Data Mining Applications with R

Data Mining Applications with R

Yanchang Zhao
Senior Data Miner, RDataMining.com, Australia
Yonghua Cen
Associate Professor, Nanjing University of Science and Technology, China

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SYDNEY • TOKYO
Academic Press is an imprint of Elsevier

Academic Press is an imprint of Elsevier

225 Wyman Street, Waltham, MA 02451, USA
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-411511-8

Printed and Bound in United States of America
13 14 15 16 17 10 9 8 7 6 5 4 3 2 1

Preface

This book presents 15 real-world applications on data mining with R, selected from 44 submissions based on peer-reviewing. Each application is presented as one chapter, covering business background and problems, data extraction and exploration, data preprocessing, modeling, model evaluation, findings, and model deployment. The applications involve a diverse set of challenging problems in terms of data size, data type, data mining goals, and the methodologies and tools to carry out analysis. The book helps readers to learn to solve real-world problems with a set of data mining methodologies and techniques and then apply them to their own data mining projects.

R code and data for the book are provided at the RDataMining.com Website http://www.rdatamining.com/books/dmar so that readers can easily learn the techniques by running the code themselves.

Background

R is one of the most widely used data mining tools in scientific and business applications, among dozens of commercial and open-source data mining software. It is free and expandable with over 4000 packages, supported by a lot of R communities around the world. However, it is not easy for beginners to find appropriate packages or functions to use for their data mining tasks. It is more difficult, even for experienced users, to work out the optimal combination of multiple packages or functions to solve their business problems and the best way to use them in the data mining process of their applications. This book aims to facilitate using R in data mining applications by presenting real-world applications in various domains.

Objectives and Significance

This book is not only a reference for R knowledge but also a collection of recent work of data mining applications.

As a reference material, this book does not go over every individual facet of statistics and data mining, as already covered by many existing books. Instead, by integrating the concepts

and techniques of statistical computation and data mining with concrete industrial cases, this book constructs real-world application scenarios. Accompanied with the cases, a set of freely available data and R code can be obtained at the book's Website, with which readers can easily reconstruct and reflect on the application scenarios, and acquire the abilities of problem solving in response to other complex data mining tasks. This philosophy is consistent with constructivist learning. In other words, instead of passive delivery of information and knowledge pieces, the book encourages readers' active thinking by involving them in a process of knowledge construction. At the same time, the book supports knowledge transfer for readers to implement their own data mining projects. We are positive that readers can find cases or cues approaching their problem requirements, and apply the underlying procedure and techniques to their projects.

As a collection of research reports, each chapter of the book is a presentation of the recent research of the authors regarding data mining modeling and application in response to practical problems. It highlights detailed examination of real-world problems and emphasizes the comparison and evaluation of the effects of data mining. As we know, even with the most competitive data mining algorithms, when facing real-world requirements, the ideal laboratory setting will be broken. The issues associated with data size, data quality, parameters, scalability, and adaptability are much more complex and research work on data mining grounded in standard datasets provides very limited solutions to these practical issues. From this point, this book forms a good complement to existing data mining text books.

Target Audience

The audience includes but does not limit to data miners, data analysts, data scientists, and R users from industry, and university students and researchers interested in data mining with R. It can be used not only as a primary text book for industrial training courses on data mining but also as a secondary text book in university courses for university students to learn data mining through practicing.

Acknowledgments

This book dates back all the way to January 2012, when our book prospectus was submitted to Elsevier. After its approval, this project started in March 2012 and completed in February 2013. During the one-year process, many e-mails have been sent and received, interacting with authors, reviewers, and the Elsevier team, from whom we received a lot of support. We would like to take this opportunity to thank them for their unreserved help and support.

We would like to thank the authors of 15 accepted chapters for contributing their excellent work to this book, meeting deadlines and formatting their chapters by following guidelines closely. We are grateful for their cooperation, patience, and quick response to our many requests. We also thank authors of all 44 submissions for their interest in this book.

We greatly appreciate the efforts of 42 reviewers, for responding on time, their constructive comments, and helpful suggestions in the detailed review reports. Their work helped the authors to improve their chapters and also helped us to select high-quality papers for the book.

Our thanks also go to Dr. Graham Williams, who wrote an excellent foreword for this book and provided many constructive suggestions to it.

Last but not the least, we would like to thank the Elsevier team for their supports throughout the one-year process of book development. Specifically, we thank Paula Callaghan, Jessica Vaughan, Patricia Osborn, and Gavin Becker for their help and efforts on project contract and book development.

Yanchang Zhao
RDataMining.com, Australia
Yonghua Cen
Nanjing University of Science and Technology, China

Review Committee

Sercan Taha Ahi	Tokyo Institute of Technology, Japan
Ronnie Alves	Instituto Tecnológico Vale Desenvolvimento Sustentável, Brazil
Nick Ball	National Research Council, Canada
Satrajit Basu	University of South Florida, USA
Christian Bauckhage	Fraunhofer IAIS, Germany
Julia Belford	UC Berkeley, USA
Eithon Cadag	Lawrence Livermore National Laboratory, USA
Luis Cavique	Universidade Aberta, Portugal
Alex Deng	Microsoft, USA
Kalpit V. Desai	Data Mining Lab at GE Research, India
Xiangjun Dong	Shandong Polytechnic University, China
Fernando Figueiredo	Customs and Border Protection Service, Australia
Mohamed Medhat Gaber	University of Portsmouth, UK
Andrew Goodchild	NEHTA, Australia
Yingsong Hu	Department of Human Services, Australia
Radoslaw Kita	Onet.pl SA, Poland
Ivan Kuznetsov	HeiaHeia.com, Finland
Luke Lake	Department of Immigration and Citizenship, Australia
Gang Li	Deakin University, Australia
Chao Luo	University of Technology, Sydney, Australia
Wei Luo	Deakin University, Australia
Jun Ma	University of Wollongong, Australia
B. D. McCullough	Drexel University, USA
Ronen Meiri	Chi Square Systems LTD, Israel
Heiko Miertzsch	EODA, Germany
Wayne Murray	Department of Human Services, Australia
Radina Nikolic	British Columbia Institute of Technology, Canada
Kok-Leong Ong	Deakin University, Australia
Charles O'Riley	USA
Jean-Christophe Paulet	JCP Analytics, Belgium
Evgeniy Perevodchikov	Tomsk State University of Control Systems and Radioelectronics, Russia

Clifton Phua	Institute for Infocomm Research, Singapore
Juana Canul Reich	Universidad Juarez Autonoma de Tabasco, Mexico
Joseph Rickert	Revolution Analytics, USA
Yin Shan	Department of Human Services, Australia
Kyong Shim	University of Minnesota, USA
Murali Siddaiah	Department of Immigration and Citizenship, Australia
Mingjian Tang	Department of Human Services, Australia
Xiaohui Tao	The University of Southern Queensland, Australia
Blanca A. Vargas-Govea	Monterrey Institute of Technology and Higher Education, Mexico
Shanshan Wu	Commonwealth Bank, Australia
Liang Xie	Travelers Insurance, USA

Additional Reviewers

Ping Xiong
Tianqing Zhu

Foreword

As we continue to collect more data, the need to analyze that data ever increases. We strive to add value to the data by turning it from data into information and knowledge, and one day, perhaps even into wisdom. The data we analyze provide insights into our world. This book provides insights into how we analyze our data.

The idea of demonstrating how we do data mining through practical examples is brought to us by Dr. Yanchang Zhao. His tireless enthusiasm for sharing knowledge of doing data mining with a broader community is admirable. It is great to see another step forward in unleashing the most powerful and freely available open source software for data mining through the chapters in this collection.

In this book, Yanchang has brought together a collection of chapters that not only talk about doing data mining but actually demonstrate the doing of data mining. Each chapter includes examples of the actual code used to deliver results. The vehicle for the *doing* is the R Statistical Software System (R Core Team, 2012), which is today's Lingua Franca for Data Mining and Statistics. Through the use of R, we can learn how others have analyzed their data, and we can build on their experiences directly, by taking their code and extending it to suit our own analyses.

Importantly, the R Software is free and open source. We are free to download the software, without fee, and to make use of the software for whatever purpose we desire, without placing restrictions on our freedoms. We can even modify the software to better suit our purposes. That's what we mean by *free*—the software offers us freedom.

Being open source software, we can learn by reviewing what others have done in the coding of the software. Indeed, we can stand on the shoulders of those who have gone before us, and extend and enhance their software to make it even better, and share our results, without limitation, for the common good of all.

As we read through the chapters of this book, we must take the opportunity to *try out* the R code that is presented. This is where we get the real value of this book—learning to do data mining, rather than just reading about it. To do so, we can install R quite simply by visiting http://www.r-project.org and downloading the installation package for

Windows or the Macintosh, or else install the packages from our favorite GNU/Linux distribution.

Chapter 1 sets the pace with a focus on Big Data. Being memory based, R can be challenged when all of the data cannot fit into the memory of our computer. Augmenting R's capabilities with the Big Data engine that is Hadoop ensures that we can indeed analyze massive datasets. The authors' experiences with power grid data are shared through examples using the Rhipe package for R (Guha, 2012).

Chapter 2 continues with a presentation of a visualization tool to assist in building Bayesian classifiers. The tool is developed using gWidgetsRGtk2 (Lawrence and Verzani, 2012) and ggplot2 (Wickham and Chang, 2012).

In Chapters 3 and 4, we are given insights into the text mining capabilities of R. The twitteR package (Gentry, 2012) is used to source data for analysis in Chapter 3. The data are analyzed for emergent issues using the tm package (Feinerer and Hornik, 2012). The tm package is again used in Chapter 4 to analyze documents using latent Dirichlet allocation. As always there is ample R code to illustrate the different steps of collecting data, transforming the data, and analyzing the data.

In Chapter 5, we move on to another larger area of application for data mining: recommender systems. The recommenderlab package (Hahsler, 2011) is extensively illustrated with practical examples. A number of different model builders are employed in Chapter 6, looking at data mining in direct marketing. This theme of marketing and customer management is continued in Chapter 7 looking at the profiling of customers for insurance. A link to the dataset used is provided in order to make it easy to follow along.

Continuing with a business-orientation, Chapter 8 discusses the critically important task of feature selection in the context of identifying customers who may default on their bank loans. Various R packages are used and a selection of visualizations provide insights into the data. Travelers and their preferences for hotels are analyzed in Chapter 9 using Rfmtool.

Chapter 10 begins a focus on some of the spatial and mapping capabilities of R for data mining. Spatial mapping and statistical analyses combine to provide insights into real estate pricing. Continuing with the spatial theme in data mining, Chapter 11 deploys randomForest (Leo Breiman et al., 2012) for the prediction of the spatial distribution of seabed hardness. Chapter 12 makes extensive use of the zooimage package (Grosjean and Francois, 2013) for image classification. For prediction, randomForest models are used, and throughout the chapter, we see the effective use of plots to illustrate the data and the modeling. The analysis of crime data rounds out the spatial analyses with Chapter 13. Time and location play a role in this analysis, relying again on gaining insights through effective visualizations of the data.

Modeling many covariates in Chapter 14 to identify the most important ones takes us into the final chapters of the book. Italian football data, recording the outcome of matches, provide the basis for exploring a number of predictive model builders. Principal component analysis also plays a role in delivering the data mining project.

The book is rounded out with the application of data mining to the analysis of domain name system data. The aim is to deliver efficiencies for DNS servers. Cluster analysis using *k*means and *k*medoids forms the primary tool, and the authors again make effective use of very many different types of visualizations.

The authors of all the chapters of this book provide and share a breadth of insights, illustrated through the use of R. There is much to learn by watching masters at work, and that is what we can gain from this book. Our focus should be on replicating the variety of analyses demonstrated throughout the book using our own data. There is so much we can learn about our own applications from doing so.

Graham Williams
February 20, 2013

References

R Core Team, 2012. R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN: 3-900051-07-0. http://www.R-project.org/.

Feinerer, I., Hornik, K., 2012. tm: Text Mining Package. R package version 0.5-8.1. http://CRAN.R-project.org/package=tm.

Gentry, J., 2012. twitteR: R based Twitter client. R package version 0.99.19. http://CRAN.R-project.org/package=twitteR.

Grosjean, P., Francois, K.D.R., 2013. zooimage: analysis of numerical zooplankton images. R package version 3.0-3. http://CRAN.R-project.org/package=zooimage.

Guha, S., 2012. Rhipe: R and Hadoop Integrated Programming Environment. R package version 0.69. http://www.rhipe.org/.

Hahsler, M., 2011. recommenderlab: lab for developing and testing recommender algorithms. R package version 0.1-3. http://CRAN.R-project.org/package=recommenderlab.

Lawrence, M., Verzani, J., 2012. gWidgetsRGtk2: toolkit implementation of gWidgets for RGtk2. R package version 0.0-81. http://CRAN.R-project.org/package=gWidgetsRGtk2.

Original by Leo Breiman, F., Cutler, A., port by Andy Liaw, R., Wiener, M., 2012. randomForest: Breiman and Cutler's random forests for classification and regression. R package version 4.6-7. http://CRAN.R-project.org/package=randomForest.

Wickham, H., Chang, W., 2012. ggplot2: an implementation of the Grammar of Graphics. R package version 0.9.3. http://had.co.nz/ggplot2/.

CHAPTER 1

Power Grid Data Analysis with R and Hadoop

Ryan Hafen, Tara Gibson, Kerstin Kleese van Dam, Terence Critchlow
Pacific Northwest National Laboratory, Richland, Washington, USA

1.1 Introduction

This chapter presents an approach to analysis of large-scale time series sensor data collected from the electric power grid. This discussion is driven by our analysis of a real-world data set and, as such, does not provide a comprehensive exposition of either the tools used or the breadth of analysis appropriate for general time series data. Instead, we hope that this section provides the reader with sufficient information, motivation, and resources to address their own analysis challenges.

Our approach to data analysis is on the basis of exploratory data analysis techniques. In particular, we perform an analysis over the entire data set to identify sequences of interest, use a small number of those sequences to develop an analysis algorithm that identifies the relevant pattern, and then run that algorithm over the entire data set to identify all instances of the target pattern. Our initial data set is a relatively modest 2TB data set, comprising just over 53 billion records generated from a distributed sensor network. Each record represents several sensor measurements at a specific location at a specific time. Sensors are geographically distributed but reside in a fixed, known location. Measurements are taken 30 times per second and synchronized using a global clock, enabling a precise reconstruction of events. Because all of the sensors are recording on the status of the same, tightly connected network, there should be a high correlation between all readings.

Given the size of our data set, simply running R on a desktop machine is not an option. To provide the required scalability, we use an analysis package called RHIPE (pronounced ree-pay) (RHIPE, 2012). RHIPE, short for the R and Hadoop Integrated Programming Environment, provides an R interface to Hadoop. This interface hides much of the complexity of running parallel analyses, including many of the traditional Hadoop management tasks. Further, by providing access to all of the standard R functions, RHIPE allows the analyst to focus instead on the analysis of code development, even when exploring large data sets. A brief

Data Mining Applications with R

introduction to both the Hadoop programming paradigm, also known as the MapReduce paradigm, and RHIPE is provided in Section 1.3. We assume that readers already have a working knowledge of R.

As with many sensor data sets, there are a large number of erroneous records in the data, so a significant focus of our work has been on identifying and filtering these records. Identifying bad records requires a variety of analysis techniques including summary statistics, distribution checking, autocorrelation detection, and repeated value distribution characterization, all of which are discovered or verified by exploratory data analysis. Once the data set has been cleaned, meaningful events can be extracted. For example, events that result in a network partition or isolation of part of the network are extremely interesting to power engineers.

The core of this chapter is the presentation of several example algorithms to manage, explore, clean, and apply basic feature extraction routines over our data set. These examples are generalized versions of the code we use in our analysis. Section 1.3.3.2.2 describes these examples in detail, complete with sample code. Our hope is that this approach will provide the reader with a greater understanding of how to proceed when unique modifications to standard algorithms are warranted, which in our experience occurs quite frequently.

Before we dive into the analysis, however, we begin with an overview of the power grid, which is our application domain.

1.2 A Brief Overview of the Power Grid

The U.S. national power grid, also known as "the electrical grid" or simply "the grid," was named the greatest engineering achievement of the twentieth century by the U.S. National Academy of Engineering (Wulf, 2000). Although many of us take for granted the flow of electricity when we flip a switch or plug in our chargers, it takes a large and complex infrastructure to reliably support our dependence on energy.

Built over 100 years ago, at its core the grid connects power producers and consumers through a complex network of transmission and distribution lines connecting almost every building in the country. Power producers use a variety of generator technologies, from coal to natural gas to nuclear and hydro, to create electricity. There are hundreds of large and small generation facilities spread across the country. Power is transferred from the generation facility to the transmission network, which moves it to where it is needed. The transmission network is comprised of high-voltage lines that connect the generators to distribution points. The network is designed with redundancy, which allows power to flow to most locations even when there is a break in the line or a generator goes down unexpectedly. At specific distribution points, the voltage is decreased and then transferred to the consumer. The distribution networks are disconnected from each other.

The US grid has been divided into three smaller grids: the western interconnection, the eastern interconnection, and the Texas interconnection. Although connections between these regions exist, there is limited ability to transfer power between them and thus each operates essentially as an independent power grid. It is interesting to note that the regions covered by these interconnections include parts of Canada and Mexico, highlighting our international interdependency on reliable power. In order to be manageable, a single interconnect may be further broken down into regions which are much more tightly coupled than the major interconnects, but are operated independently.

Within each interconnect, there are several key roles that are required to ensure the smooth operation of the grid. In many cases, a single company will fill multiple roles—typically with policies in place to avoid a conflict of interest. The goal of the power producers is to produce power as cheaply as possible and sell it for as much as possible. Their responsibilities include maintaining the generation equipment and adjusting their generation based on guidance from a balancing authority. The *balancing authority* is an independent agent responsible for ensuring the transmission network has sufficient power to meet demand, but not a significant excess. They will request power generators to adjust production on the basis of the real-time status of the entire network, taking into account not only demand, but factors such as transmission capacity on specific lines. They will also dynamically reconfigure the network, opening and closing switches, in response to these factors. Finally, the utility companies manage the distribution system, making sure that power is available to consumers. Within its distribution network, a utility may also dynamically reconfigure power flows in response to both planned and unplanned events. In addition to these primary roles, there are a variety of additional roles a company may play—for example, a company may lease the physical transmission or distribution lines to another company which uses those to move power within its network. Significant communication between roles is required in order to ensure the stability of the grid, even in normal operating circumstances. In unusual circumstances, such as a major storm, communication becomes critical to responding to infrastructure damage in an effective and efficient manner.

Despite being over 100 years old, the grid remains remarkably stable and reliable. Unfortunately, new demands on the system are beginning to affect it. In particular, energy demand continues to grow within the United States—even in the face of declining usage per person (DOE, 2012). New power generators continue to come online to address this need, with new capacity increasingly either being powered by natural gas generators (projected to be 60% of new capacity by 2035) or based on renewable energy (29% of new capacity by 2035) such as solar or wind power (DOE, 2012). Although there are many advantages to the development of renewable energy sources, they provide unique challenges to grid stability due to their unpredictability. Because electricity cannot be easily stored, and renewables do not provide a consistent supply of power, ensuring there is sufficient power in the system to meet demand without significant overprovisioning (i.e., wasting energy) is a major challenge facing grid

operators. Further complicating the situation is the distribution of the renewable generators. Although some renewable sources, such as wind farms, share many properties with traditional generation capabilities—in particular, they generate significant amounts of power and are connected to the transmission system—consumer-based systems, such as solar panels on a business, are connected to the distribution network, not the transmission network. Although this distributed generation system can be extremely helpful at times, it is very different from the current model and introduces significant management complexity (e.g., it is not currently possible for a transmission operator to control when or how much power is being generated from solar panels on a house).

To address these needs, power companies are looking toward a number of technology solutions. One potential solution being considered is transitioning to real-time pricing of power. Today, the price of power is fixed for most customers—a watt used in the middle of the afternoon costs the same as a watt used in the middle of the night. However, the demand for power varies dramatically during the course of a day, with peak demand typically being during standard business hours. Under this scenario, the price for electricity would vary every few minutes depending on real-time demand. In theory, this would provide an incentive to minimize use during peak periods and transfer that utilization to other times. Because the grid infrastructure is designed to meet its peak load demands, excess capacity is available off-hours. By redistributing demand, the overall amount of energy that could be delivered with the same infrastructure is increased. For this scenario to work, however, consumers must be willing to adjust their power utilization habits. In some cases, this can be done by making appliances cost aware and having consumers define how they want to respond to differences in price. For example, currently water heaters turn on and off solely on the basis of the water temperature in the tank—as soon as the temperature dips below a target temperature, the heater goes on. This happens without considering the time of day or water usage patterns by the consumer, which might indicate if the consumer even needs the water in the next few hours. A price-aware appliance could track usage patterns and delay heating the water until either the price of electricity fell below a certain limit or the water was expected to be needed soon. Similarly, an air conditioner might delay starting for 5 or 10 min to avoid using energy during a time of peak demand/high cost without the consumer even noticing.

Interestingly, the increasing popularity of plug-in electric cars provides both a challenge and a potential solution to the grid stability problems introduced by renewables. If the vehicles remain price insensitive, there is the potential for them to cause sudden, unexpected jumps in demand if a large number of them begin charging at the same time. For example, one car model comes from the factory preset to begin charging at midnight local time, with the expectation that this is a low-demand time. However, if there are hundreds or thousands of cars within a small area, all recharging at the same time, the sudden surge in demand becomes significant. If the cars are price aware, however, they can charge whenever demand is lowest, as long as they are fully charged when their owner is ready to go. This would spread out the charging over

the entire night, smoothing out the demand. In addition, a price-aware car could sell power to the grid at times of peak demand by partially draining its battery. This would benefit the owner through a buy low, sell high strategy and would mitigate the effect of short-term spikes in demand. This strategy could help stabilize the fluctuations caused by renewables by, essentially, providing a large-scale power storage capability.

In addition to making devices cost aware, the grid itself needs to undergo a significant change in order to support real-time pricing. In particular, the distribution system needs to be extended to support real-time recording of power consumption. Current power meters record overall consumption, but not when the consumption occurred. To enable this, many utilities are in the process of converting their customers to *smart meters*. These new meters are capable of sending the utility real-time consumption information and have other advantages, such as dramatically reducing the time required to read the meters, which have encouraged their adoption. On the transmission side, existing sensors provide operators with the status of the grid every 4 seconds. This is not expected to be sufficient given increasing variability, and thus new sensors called phasor measurement units (PMUs) are being deployed. PMUs provide information 30-60 times per second. The sensors are time synchronized to a global clock so that the state of the grid at a specific time can be accurately reconstructed. Currently, only a few hundred PMUs are deployed; however, the NASPI project anticipates having over 1000 PMUs online by the end of 2014 (Silverstein, 2012) with a final goal of between 10,000 and 50,000 sensors deployed over the next 20 years.

An important side effect of the new sensors on the transmission and distribution networks is a significant increase in the amount of information that power companies need to collect and process. Currently, companies are using the real-time streams to identify some critical events, but are not effectively analyzing the resulting data set. The reasons for this are twofold. First, the algorithms that have been developed in the past are not scaling to these new data sets. Second, exactly what new insights can be gleaned from this more refined data is not clear. Developing scalable algorithms for known events is clearly a first step. However, additional investigation into the data set using techniques such as exploratory analysis is required to fully utilize this new source of information.

1.3 Introduction to MapReduce, Hadoop, and RHIPE

Before presenting the power grid data analysis, we first provide an overview of MapReduce and associated topics including Hadoop and RHIPE. We present and discuss the implementation of a simple MapReduce example using RHIPE. Finally, we discuss other parallel R approaches for dealing with large-scale data analysis. The goal is to provide enough background for the reader to be comfortable with the examples provided in the following section.

The example we provide in this section is a simple implementation of a MapReduce operation using RHIPE on the `iris` data (Fisher, 1936) included with R. The goal is to solidify our

description of MapReduce through a concrete example, introduce basic RHIPE commands, and prepare the reader to follow the code examples on our power grid work presented in the following section. In the interest of space, our explanations focus on the various aspects of RHIPE, and not on R itself. A reasonable skill level of R programming is assumed.

A lengthy exposition on all of the facets of RHIPE is not provided. For more details, including information about installation, job monitoring, configuration, debugging, and some advanced options, we refer the reader to RHIPE (2012) and White (2010).

1.3.1 MapReduce

MapReduce is a simple but powerful programming model for breaking a task into pieces and operating on those pieces in an embarrassingly parallel manner across a cluster. The approach was popularized by Google (Dean and Ghemawat, 2008) and is in wide use by companies processing massive amounts of data.

MapReduce algorithms operate on data structures represented as *key/value* pairs. The data are split into blocks; each block is represented as a key and value. Typically, the key is a descriptive data structure of the data in the block, whereas the value is the actual data for the block. MapReduce methods perform independent parallel operations on input key/value pairs and their output is also key/value pairs. The MapReduce model is comprised of two phases, the *map* and the *reduce*, which work as follows:

Map: A map function is applied to each input key/value pair, which does some user-defined processing and emits new key/value pairs to intermediate storage to be processed by the reduce.

Shuffle/*Sort*: The map output values are collected for each unique map output key and passed to a reduce function.

Reduce: A reduce function is applied in parallel to all values corresponding to each unique map output key and emits output key/value pairs.

1.3.1.1 An Example: The Iris Data

The iris data are very small and methods can be applied to it in memory, within R, without splitting it into pieces and applying MapReduce algorithms. It is an accessible introductory example nonetheless, as it is easy to verify computations done with MapReduce to those with the traditional approach. It is the MapReduce principles—not the size of the data—that are important: Once an algorithm has been expressed in MapReduce terms, it theoretically can be applied unchanged to much larger data.

The iris data are a data frame of 150 measurements of iris petal and sepal lengths and widths, with 50 measurements for each species of "setosa," "versicolor," and "virginica." Let us assume that we are doing some computation on the sepal length. To simulate the notion of data

being partitioned and distributed, consider the data being randomly split into 3 blocks. We can achieve this in R with the following:

```
> set.seed(4321)           # make sure that we always get the same partition
> permute <- sample(1:150, 150)
> splits <- rep(1:3, 50)
> irisSplit <- tapply(permute, splits, function(x) {
     iris[x, c("Sepal.Length", "Species")]
  })
> irisSplit
$`1`
   Sepal.Length    Species
51          7.0 versicolor
7           4.6     setosa
... # output truncated
```

Throughout this chapter, code blocks that also display output will distinguish lines containing code with "> " at the beginning of the line. Code blocks that do not display output do not add this distinction.

This partitions the `Sepal.Length` and `Species` variables into three random subsets, having the keys "1," "2," or "3," which correspond to our blocks. Consider a calculation of the maximum sepal length by species with `irisSplit` as the set of input key/value pairs. This can be achieved with MapReduce by the following steps:

Map: Apply a function to each division of the data which, for each species, computes the maximum sepal length and outputs key=species and value=max sepal length.

Shuffle/*Sort*: Gather all map outputs with key "setosa" and send to one reduce, then all with key "versicolor" to another reduce, etc.

Reduce: Apply a function to all values corresponding to each unique map output key (species) which gathers and calculates the maximum of the values.

It can be helpful to view this process visually, as is shown in Figure 1.1. The input data are the `irisSplit` set of key/value pairs. As described in the steps above, applying the map to each input key/value pair emits a key/value pair of the maximum sepal length per species. These are gathered by key (species) and the reduce step is applied which calculates a maximum of maximums, finally outputting a maximum sepal length per species. We will revisit this Figure with a more detailed explanation of the calculation in Section 1.3.3.2.

1.3.2 Hadoop

Hadoop is an open-source distributed software system for writing MapReduce applications capable of processing vast amounts of data, in parallel, on large clusters of commodity hardware, in a fault-tolerant manner. It consists of the Hadoop Distributed File

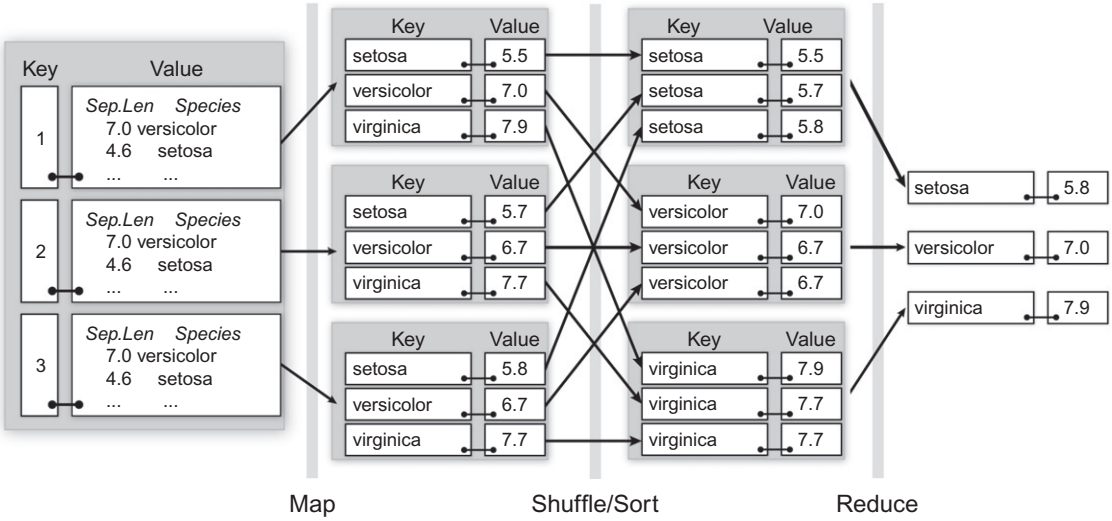


Figure 1.1
An Illustration of applying a MapReduce job to calculate the maximum Sepal. Length by Species for the irisSplit data.

System (HDFS) and the MapReduce parallel compute engine. Hadoop was inspired by papers written about Google's MapReduce and Google File System (Dean and Ghemawat, 2008).

Hadoop handles data by distributing key/value pairs into the HDFS. Hadoop schedules and executes the computations on the key/value pairs in parallel, attempting to minimize data movement. Hadoop handles load balancing and automatically restarts jobs when a fault is encountered.

Hadoop has changed the way many organizations work with their data, bringing cluster computing to people with little knowledge of the complexities of distributed programming. Once an algorithm has been written the "MapReduce way," Hadoop provides concurrency, scalability, and reliability for free.

1.3.3 RHIPE: R with Hadoop

RHIPE is a merger of R and Hadoop. It enables an analyst of large data to apply numeric or visualization methods in R. Integration of R and Hadoop is accomplished by a set of components written in R and Java. The components handle the passing of information between R and Hadoop, making the internals of Hadoop transparent to the user. However, the user must be aware of MapReduce as well as parameters for tuning the performance of a Hadoop job.

One of the main advantages of using R with Hadoop is that it allows rapid prototyping of methods and algorithms. Although R is not as fast as pure Java, it was designed as a programming environment for working with data and has a wealth of statistical methods and tools for data analysis and manipulation.

1.3.3.1 Installation

RHIPE depends on Hadoop, which can be tricky to install and set up. Excellent references for setting up Hadoop can be found on the web. The site www.rhipe.org provides installation instructions for RHIPE as well as a virtual machine with a local single-node Hadoop cluster. A single-node setup is good for experimenting with RHIPE syntax and for prototyping jobs before attempting to run at scale. Whereas we used a large institutional cluster to perform analyses on our real data, all examples in this chapter were run on a single-node virtual machine. We are using RHIPE version 0.72. Although RHIPE is a mature project, software can change over time. We advise the reader to check www.rhipe.org for notes of any changes since version 0.72.

Once RHIPE is installed, it can be loaded into your R session as follows:

```
> library(Rhipe)
------------------------------------------------------
| IMPORTANT: Before using Rhipe call rhinit()        |
| Rhipe will not work or most probably crash         |
------------------------------------------------------
> rhinit()
Rhipe initialization complete
Rhipe first run complete
Initializing mapfile caches
[1] TRUE
> hdfs.setwd("/")
```

RHIPE initialization starts a JVM on the local machine that communicates with the cluster. The `hdfs.setwd()` command is similar to R's `setwd()` in that it specifies the base directory to which all references to files on HDFS will be subsequently based on.

1.3.3.2 Iris MapReduce Example with RHIPE

To execute the example described in Section 1.3.1.1, we first need to modify the `irisSplit` data to have the key/value pair structure that RHIPE expects. Key/value pairs in RHIPE are represented as an R list, where each list element is another list of two elements, the first being a key and the second being the associated value. Keys and values can be arbitrary R objects. The list of key-value pairs is written to HDFS using rhwrite ():

```
> irisSplit <- lapply(seq_along(irisSplit), function(i)
+ list(i, irisSplit[[i]])
+ )
> rhwrite(irisSplit, "irisData")
Wrote 3 pairs occupying 2100 bytes
```

This creates an HDFS directory `irisData` (relative to the current HDFS working directory), which contains a Hadoop sequence file. Sequence files are flat files consisting of key/value pairs. Typically, data are automatically split across multiple sequence files in the named data

directory, but since this data set is so small, it only requires one file. This file can now serve as an input to RHIPE MapReduce jobs.

1.3.3.2.1 The Map Expression

Below is a map expression for the MapReduce task of computing the maximum sepal length by species. This expression transforms the random data splits in the irisData file into a partial answer by computing the maximum of each species within each of the three splits. This significantly reduces the amount of information passed to the shuffle sort, since there will be three sets (one for each of the map keys) of at most three key-value pairs (one for each species in each map value).

```
maxMap <- expression({
   for(r in map.values) {
      by(r, r$Species, function(x) {
         rhcollect(
            as.character(x$Species[1]), # key
            max(x$Sepal.Length)         # value
         )
      })
   }
})
```

RHIPE initializes the `map.values` object to be an R list with the input keys and values for each map task to process. Typically, multiple map expressions are executed in parallel with batches of key/value pairs being passed in until all key/value pairs have been processed. Hadoop takes care of the task distribution, relaunching map tasks when failures occur and attempting to keep the computation local to the data. Because each map task operates on only the subset of the keys and values, it is important that they do not make improper assumptions about which data elements are being processed.

The above map expression cycles through the input key/value pairs for each `map.value`, calculating the maximum sepal length by species for each of the data partitions. The function `rhcollect()` emits key/value pairs to be shuffled and sorted before being passed to the reduce expression. The map expression generates three output collections, one for each of the unique keys output by the map, which is the species. Each collection contains three elements corresponding to the maximum sepal length for that species found within the associated map value. This is visually depicted by the Map step in Figure 1.1. In this example, the input `map.keys` ("1," "2," and "3") are not used since they are not meaningful for the computation.

Debugging expressions in parallel can be extremely challenging. As a validation step, it can be useful for checking code correctness to step through the map expression code manually (up to `rhcollect()`, which only works inside of a real RHIPE job), on a single processor and with a small data set, to ensure that it works as expected. Sample input `map.keys` and `map.values` can be obtained by extracting them from `irisSplit`:

```
map.keys <- lapply(irisSplit, "[[", 1)
map.values <- lapply(irisSplit, "[[", 2)
```

Then one can proceed through the map expression to investigate what the code is doing.

1.3.3.2.2 The Reduce Expression

Hadoop automatically performs the shuffle/sort step after the map task has been executed, gathering all values with the same map output key and passing them to the same reduce task. Similar to the map step, the reduce expression is passed key/value pairs through the `reduce.key` and associated `reduce.values`. Because the shuffle/sort combines values with the same key, each reduce task can assume that it will process all values for a given key (unlike the map task). Nonetheless, because the number of values can be large, the `reduce.values` are fed into the reduce expression in batches.

```
maxReduce <- expression(
   pre={
      speciesMax <- NULL
   },
   reduce={
       speciesMax <- max(c(speciesMax, do.call(c, reduce.values)))
   },
   post={
       rhcollect(reduce.key, speciesMax)
   }
)
```

To manage the sequence of values, the reduce expression is actually defined as a vector of expressions, `pre`, `reduce`, and `post`. The `pre` expression is executed once at the beginning of the reduce phase for each `reduce.key` value. The reduce expression is executed each time new `reduce.values` arrive, and the `post` expression is executed after all values have been processed for the particular `reduce.key`. In our example, pre is used to initialize the `speciesMax` value to `NULL`. The `reduce.values` arrive as a list of a collection of the emitted map values, which for this example is a list of scalars corresponding to the sepal lengths. We update `speciesMax` by calculating the maximum of the `reduce.values` and the current value of `speciesMax`. For a given reduce key, the `reduce` expression may be invoked multiple times, each time with a new batch of `reduce.values`, and updating in this manner assures us that we ultimately obtain the maximum of all maximums for the given species. The `post` expression is used to generate the final key/value pairs from this execution, each species and its maximum sepal length.

1.3.3.2.3 Running the Job

RHIPE jobs are prepared and run using the `rhwatch()` command, which at a minimum requires the specification of the map and reduce expressions and the input and output directories.

```
> maxSepalLength <- rhwatch(
+ map=maxMap,
+ reduce=maxReduce,
+ input="irisData",
+ output="irisMaxSepalLength"
```

```
+)
...
job_201301212335_0001, State: PREP, Duration: 5.112
URL: http://localhost:50030/jobdetails.jsp?jobid=job_201301212335_0001
        pct numtasks pending running complete killed failed_attempts
map       0        1       1       0        0      0                0
reduce    0        1       1       0        0      0                0
        killed_attempts
map                   0
reduce                0
Waiting 5 seconds
...
```

Note that the time it takes this job to execute (~30 s) is longer than it would take to do the simple calculation in memory in R. There is a small overhead with launching a RHIPE MapReduce job that becomes negligible as the size of the data grows.

`rhwatch()` specifies, at a minimum, the map and reduce expressions and the input and output directories. There are many more options that can be specified here, including Hadoop parameters. Choosing the right Hadoop parameters varies by the data, the algorithm, and the cluster setup, and is beyond the scope of this chapter. We direct the interested reader to the `rhwatch()` help page and to (White, 2010) for appropriate guides on Hadoop parameter selection. All examples provided in the chapter should run without problems on the virtual machine available at www.rhipe.org using the default parameter settings.

Some of the printed output of the call to `rhwatch()` is truncated in the interest of space. The printed output basically provides information about the status of the job. Above, we see output from the setup phase of the job. There is one map task and one reduce task. With larger data and on a larger cluster, the number of tasks will be different, and mainly depend on Hadoop parameters which can be set through RHIPE or in the Hadoop configuration files. Hadoop has a web-based job monitoring tool whose URL is specified when the job is launched, and the URL to this is supplied in the printed output.

The output of the MapReduce job is stored by default as a Hadoop sequence file of key/value pairs on HDFS in the directory specified by `output` (here, it is `irisMaxSepalLength`. By default, `rhwatch()` reads these data in after job completion and returns it. If the output of the job is too large, as is often the case, and we don't want to immediately read it back in but instead use it for subsequent MapReduce jobs, we can add `readback=FALSE` to our `rhwatch()` call and then later on call `rhread("irisMaxSepalLength")`. In this chapter, we will usually read back results in the examples since the output datasets are small.

1.3.3.2.4 Looking at Results

The output from a MapReduce run is the set of key/value pairs generated by the reduce expression. In exploratory data analysis, often it is important to reduce the data to a size that is manageable within a single, local R session. Typically, this is accomplished by iterative

applications of MapReduce to transform, subset, or reduce the data. In this example, the result is simply a list of three key-value pairs.

```
> maxSepalLength
 [[1]]
[[1]][[1]]
[1] "setosa"

[[1]][[2]]
[1] 5.8
...
> do.call("rbind", lapply(maxSepalLength, function(x) {
+    data.frame(species=x[[1]], max=x[[2]])
+ }))
     species max
1     setosa 5.8
2  virginica 7.9
3 versicolor 7.0
```

We see that maxSepalLength is a list of key/value pairs. This code turns this list of key/value pairs into a more suitable format, extracting the key (species) and maximum for each pair and binding them into a `data.frame`.

Before moving on, we introduce a simplified way to create map expressions. Typically, a RHIPE map expression as defined above for the iris example simply iterates over groups of key/value pairs provided as `map.keys` and `map.values` lists. To avoid this repetition, a wrapper function, `rhmap()` has been created, that is applied to each element of `map.keys` and `map.values`, where the current `map.keys` element is available as m, and the current `map.values` element is available as r. Thus, the map expression for the iris example could be rewritten as

```
maxMap <- rhmap({
   by(r, r$Species, function(x) {
      rhcollect(
         as.character(x$Species[1]), # key
         max(x$Sepal.Length)         # value
      )
   })
})
```

This simplification will be used in all subsequent examples.

1.3.4 Other Parallel R Packages

As evidenced by over 4000 R add-on packages available on the Comprehensive R Archive Network (CRAN), there are many ways to get things done in R. Parallel processing is no exception. High-performance computing with R is very dynamic, and a good place to find up-to-date information about what is available is the CRAN task view for high-performance computing.[1] Nevertheless, this chapter would not be complete without a brief overview of some

[1] http://cran.r-project.org/webviews/HighPerformanceComputing.html.

other parallel packages available at this time. The interested reader is directed to a very good, in-depth discussion about standard parallel R approaches in (McCallum and Weston, 2011).

There are a number of R packages for parallel computation that are not suited for analysis of large amounts of data. Two of the most popular parallel packages are snow (Tierney et al., 2012) and multicore (Urbanek, 2011), versions of which are now part of the base R package `parallel` (R Core Team, 2012). These packages enable embarrassingly parallel computation on multiple cores and are excellent for CPU heavy tasks. Unfortunately, it is incumbent upon the analyst to define how each process interacts with the data and how the data are stored. Using the MapReduce paradigm with these packages is tedious because the user must explicitly perform the intermediate storage and shuffle/sort tasks, which Hadoop takes care of automatically. Finally, these packages do not provide automatic fault tolerance, which is extremely important when computations are spread out over hundreds or thousands of cores.

R packages that allow for dealing with larger data outside of R's memory include `ff` (Adler et al., 2012), `bigmemory` (Kane and Emerson, 2011), and `RevoScaleR` (Revolution Analytics, 2012a,b). These packages have specialized formats to store matrices or data frames with a very large number of rows. They have corresponding packages that perform computation of several standard statistical methods on these data objects, much of which can be done in parallel. We do not have extensive experience with these packages, but we presume that they work very well for moderate-sized, well-structured data. When the data must be spread across multiple machines and is possibly unstructured, however, we turn to solutions like Hadoop.

There are multiple approaches for using Hadoop with R. Hadoop Streaming, a part of Hadoop that allows any executable, which reads from standard input and writes to standard output to be used as map and reduce processes, can be used to process R executables. The rmr package (Revolution Analytics, 2012a,b) builds upon this capability to simplify the process of creating the map and reducing tasks. RHIPE and rmr are similar in what they accomplish: using R with Hadoop without leaving the R console. The major difference is that RHIPE is integrated with Hadoop's Java API whereas rmr uses Hadoop Streaming. The rmr package is a good choice for a user satisfied with the Hadoop Streaming interface. RHIPE's design around the Java API allows for a more managed interaction with Hadoop during the analysis process. The `segue` package (Long, 2012) allows for `lapply()` style computation using Hadoop on Amazon Elastic MapReduce. It is very simple, but limited for general-purpose computing.

1.4 Power Grid Analytical Approach

This section presents both a synopsis of the methodologies we applied to our 2TB power grid data set and details about their implementation. These methodologies include aspects of exploratory analysis, data cleaning, and event detection. Some of the methods are

straightforward and could be accomplished using a variety of techniques, whereas others clearly exhibit the power and simplicity of MapReduce. Although this approach is not suited for every data mining problem, the following examples should demonstrate that it does provide a powerful and scalable rapid development environment for a breadth of data mining tasks.

1.4.1 Data Preparation

Preprocessing data into suitable formats is an important consideration for any analysis task, but particularly so when using MapReduce. In particular, the data must be partitioned into key/value pairs in a way that makes the resulting analysis efficient. This applies to both optionally reformatting the original data into a format that can be manipulated by R and partitioning the data in a way that supports the required analyses. In general, it is not uncommon to partition the data along multiple dimensions to support different analyses.

As a first step, it is worthwhile to convert the raw data into a format that can be quickly ingested by R. For example, converting data from a customized binary file into an R data frame dramatically reduces read times for subsequent analyses. The raw PMU data were provided in a proprietary binary format that uses files to partition the data. Each file contains approximately 9000 records, representing 5 min of data. Each record contains 555 variables representing the time and multiple measurements for each sensor.

We provide R code in Appendix to generate a synthetic data set of frequency measurements and flags, a subset of 76 variables out of the 555. These synthetic data are a simplified version of the actual PMU data, containing only the information required to execute the examples in this section. Where appropriate, we motivate our analysis using results pulled from the actual data set. The attentive reader will notice that these results will typically not exhibit the same properties as the same analysis performed on the synthetic data, although we tried to artificially preserve some of these properties. Although unfortunate, that difference is to be expected.

The major consideration when partitioning the data is determining how to best split it into key/value pairs. Our analyses are primarily focused on time-local behavior, and therefore the original partitioning of the data into 5-min time intervals is maintained. Five minutes is an appropriate size for analysis because interesting time-local behaviors occur in intervals spanning only a few seconds, and the blocks are of an adequate size ($\sim$11 MB per serialized block) for multiple blocks to be read in to the map function at a given time. However, many raw data files do not contain exactly 5 min of data, so some additional preprocessing is required to fill in missing information. For simplicity, the synthetic data set comprises 10 complete 5-min partitions.

If our analysis focused on behavior of individual PMUs over time, the partitioning would be by PMU. This would likely result in too much data per partition and thus an additional refinement based on time or additional partitioning within PMU would also be needed.

1.4.2 Exploratory Analysis and Data Cleaning

A good first step in the data mining process is to visually and numerically explore the data. With large data sets, this is often initially accomplished through summaries since a direct analysis of the detailed records can be overwhelming. Although summaries can mask interesting features of the full data, they can also provide immediate insights. Once the data are understood at the high level, analysis of the detailed records can be fruitful.

1.4.2.1 5-min Summaries

A simple summary of frequency over a 5-min window for each PMU provides a good starting point for understanding the data. This calculation is straightforward: since the data are already divided into 5-min blocks the computation simply calculates summary statistics at each time stamp split by PMU. The map expression for this task is:

```
map.pmusumm <- rhmap({
    # r is the data.frame of values for a 5-minute block
    # k is the key (time) for the block
    colNames <- colnames(r)
    freqColumns <- which(grepl("freq", colNames))
    pmuName <- gsub("(.*)\\.freq", "\\1", colNames[freqColumns])
    for(j in seq_along(freqColumns)) { # loop through frequency columns
       v <- r[,freqColumns[j]] / 1000+60 # convert to HZ

       rhcollect(
          pmuName[j], # key is the PMU
          data.frame(
             time   = k,
             min    = min(   v, na.rm=TRUE),
             max    = max(   v, na.rm=TRUE),
             mean   = mean(  v, na.rm=TRUE),
             stdev  = sd(    v, na.rm=TRUE),
             median = median(v, na.rm=TRUE),
             nna    = length(which(is.na(v)))
          )
       )
    }
})
```

The first three lines after the comments extract only the PMU frequency columns (because the data frame consists of column names ending with ".freq" and ".flag"). These three lines will be recycled in most of the subsequent examples. These columns are then iterated over, converting the 1000 offset from 60 HZ to a true frequency value and calculating a data frame of summary statistics which is emitted to the reduce function with the PMU as the key. This map task emits 38 key/value pairs (one for each PMU) for each input key (time interval).

The reduce collects all of the values for each unique PMU and collates them:

```
reduce.pmusumm <- expression(
   pre = {
      res <- NULL
   },
   reduce = {
      res <- do.call(rbind, c(list(res), reduce.values))
   },
   post = {
      res <- res[order(res$time),] # order results by time
      res$time <- as.POSIXct(res$time, origin="1970-01-01", tz="UTC")
      rhcollect(reduce.key, res)
   }
)
```

Recall that the reduce expression is applied to each collection of unique map output keys, which in this case is the PMU identifier. The `pre` expression is evaluated once for each key, initializing the result data frame. The `reduce` expression is evaluated as new `reduce.values` flow in iteratively building the data frame. Finally, the `post` expression orders the result by time, converts the time to an R `POSIXct` object, and writes the result.

We run the job by:

```
summ5min <- rhwatch(
    map=map.pmusumm,
    reduce=reduce.pmusumm,
    input="blocks5min",
    output="blocks5min_summary"
)
```

To look at the first key/value pair:

```
> summ5min[[1]][[1]]
[1] "AA"
> str(summ5min[[1]][[2]])
'data.frame': 10 obs. of 7 variables:
$ time : POSIXct, format: "2012-01-01 00:00:00" "2012-01-01 00:05:00" "2012-01-01 00:10:00"
...
$ min    : num 60 60 60 60 60 ...
$ max    : num 60 60 60 60 60 ...
$ mean   : num 60 60 60 60 60 ...
$ stdev  : num 0.00274 0.00313 0.0044 0.0028 0.00391 ...
$ median: num 60 60 60 60 60 ...
$ nna    : int 0 0 0 0 0 0 0 0 0 0
```

Note that the order of the key/value pairs read in by `rhread()` will not necessarily be the same for each run, depending on the order of when tasks are completed when running in parallel.

In the real data, the 5-min summaries highlight some interesting behaviors in the frequency. Specifically, some PMUs exhibit extended periods of abnormally deviant median frequencies, and more interestingly, these aberrant points typically have few values reported for the 5-min

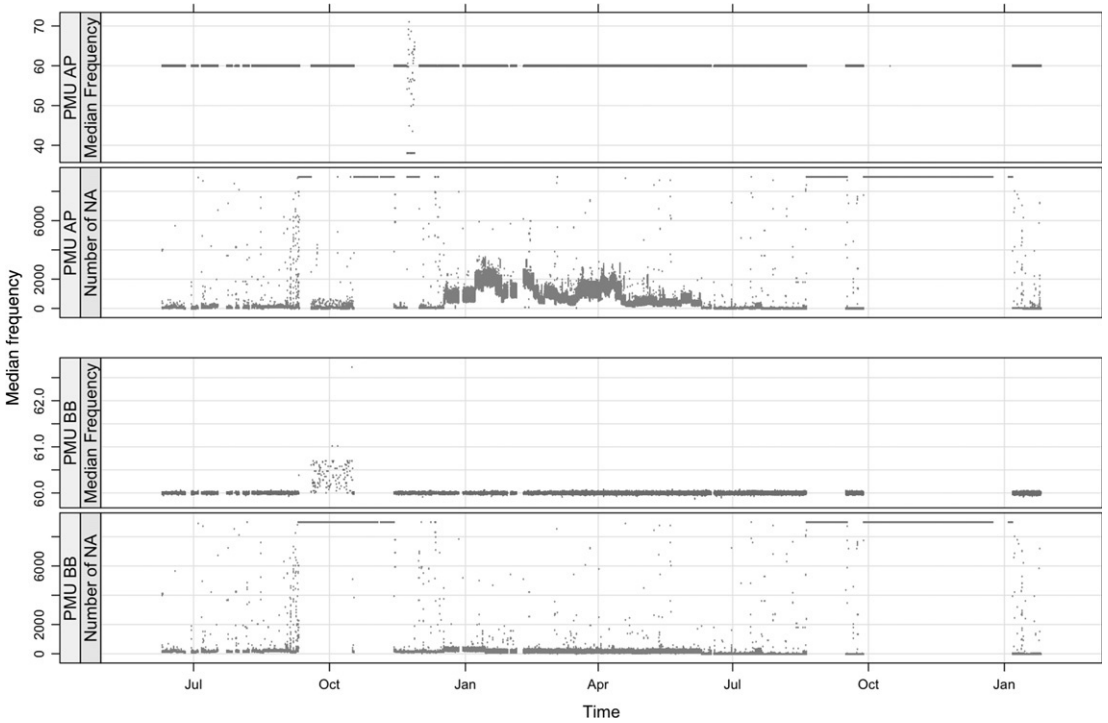


Figure 1.2
5-min medians and number of missing values versus time for two PMUs.

interval. Figure 1.2 shows a plot of 5-min median frequencies and number of missing observations across time for two PMUs as calculated from the real data.

Plots like these helped identify regions of data for which the median frequency was so out-of-bounds that it could not be trusted.

1.4.2.2 Quantile Plots of Frequency

A summary that provides information about the distribution of the frequency values for each PMU is a quantile plot. Calculating exact quantiles would require collecting all of the frequency values for each PMU, about 1.4 billion values, then applying a quantile algorithm. This is an example where a simplified algorithm that achieves an approximate solution is very useful.

An approximate quantile plot can be achieved by first discretizing the data through rounding and then tabulating the unique values. Great care must be taken in choosing how to discretize the data to ensure that the resulting distribution is not oversimplified. The PMU frequency data are already discretely reported as an integer 1000 offset from 60 HZ. Thus, our task is to tabulate the unique frequency values by PMU and translate the tabulation into quantiles. Here, as in the previous

example, we are transforming data split by time to data split by PMU. This time, we are aggregating across time as well, tabulating the occurrence of each unique frequency value:

```
map.freqquant <- rhmap({
    colNames <- colnames(r)
    freqColumns <- which(grepl("freq", colNames))
    pmuName <- gsub("(.*)\\.freq", "\\1", colNames[freqColumns])

    for(j in seq_along(freqColumns)) { # loop through frequency columns
       freqtab <- table(r[,freqColumns[j]])

       rhcollect(
           pmuName[j], # key is the PMU
           data.frame(
              level = as.integer(names(freqtab)),
              count = as.numeric(freqtab)
           )
       )
    }
})
```

As a result, this map task is similar to that of the previous example. The only difference is the data frame of tabulated counts of frequencies that is being passed to the reduce expression.[2] The map expression emits one data frame of tabulations for each PMU for each time block.

The reduce function must combine a collection of tabulations for a given PMU:

```
reduce.tab <- expression(
   pre = {
      res <- NULL
   },
   reduce = {
      tabUpdate <- do.call(rbind, c(list(res), reduce.values))
      tabUpdate <- xtabs(count ~ level, data = tabUpdate)
      res <- data.frame(
         level = as.integer(names(tabUpdate)),
         count = as.numeric(tabUpdate)
      )
   },
   post = {
      rhcollect(reduce.key, res)
   }
)
```

For each PMU, we combine the data frames of the tabulation output and create a new tabulation data frame using R's `xtabs()` function. This function sums the already-computed counts instead of simply counting the number of unique occurrences. The call to `as.numeric()` is required because the count may become too large to be represented as an integer. As is common

[2] We use the word *count* to refer to the tabulation of frequencies as opposed to *frequency* to avoid confusion with the PMU measurement of grid frequency.

in reduce steps, the output of the `reduce` expression is of the same format as the values emitted from the map, allowing it to be easily updated as new `reduce.values` arrive. This function is a generalized reduce that can be applied to other computations.

To execute the job:

```
freqtab <- rhwatch(
   map=map.freqquant, reduce=reduce.tab,
   input="blocks5min", output="frequency_quantile"
)
```

The data and computed quantiles for the first PMU can be manipulated and viewed by:

```
ftpmu <- freqtab[[1]][[2]]
cs <- cumsum(ftpmu$count)
f <- ppoints(2000, 1)
q <- ftpmu$level[sapply(f * sum(ftpmu$count), function(x)
   min(which(x <= cs)))]
plot(f, q/1000+60, xlab="Quantile", ylab="Frequency (HZ)")
```

Here, `cs` is the cumulative sum of the tabulated frequencies, and `f` is the vector of discrete values which had been recorded by the PMU. The quantile, `q[i]`, for a given value of `f[i]` is the value at which the number of counts corresponding to f[i] is less than or equal to the cumulative sum. This code essentially turns our frequency tabulation into approximate quantiles.

The distribution of frequency for many of the PMUs was approximately normal. Figure 1.3 shows normal-quantile plots for two PMUs in the real data set. The distribution of frequency for PMU A looks approximately normal, whereas that for PMU B has an abnormally high amount of frequency values right below 60 HZ. This indicates data integrity issues.

As implied by the previous two examples, there are many observations in this data set that are suspect. With RHIPE, we can perform calculations across the entire data set to uncover and confirm the bad data cases while developing algorithms for removing these cases from the data. A conservative approach filters only impossible data values, ensuring that anomalous data are not unintentionally filtered out by these algorithms. Furthermore, the original data set is unchanged by the filters, which are applied on demand to remove specific types of information from the data set.

1.4.2.3 Tabulating Frequency by Flag

Investigating the strange behavior of the quantile plots required looking at other aspects of the data including the PMU flag. The initial information from one expert was that flag 128 is a signal for bad data. Our first step was to verify this and then gain insight into the other flags.

Tabulating frequency by PMU and flag can indicate whether or not the frequency behaves differently based on the flag settings. This involves a computation similar to the quantile calculation, except that we step through each unique flag for each PMU, and emit a vector with

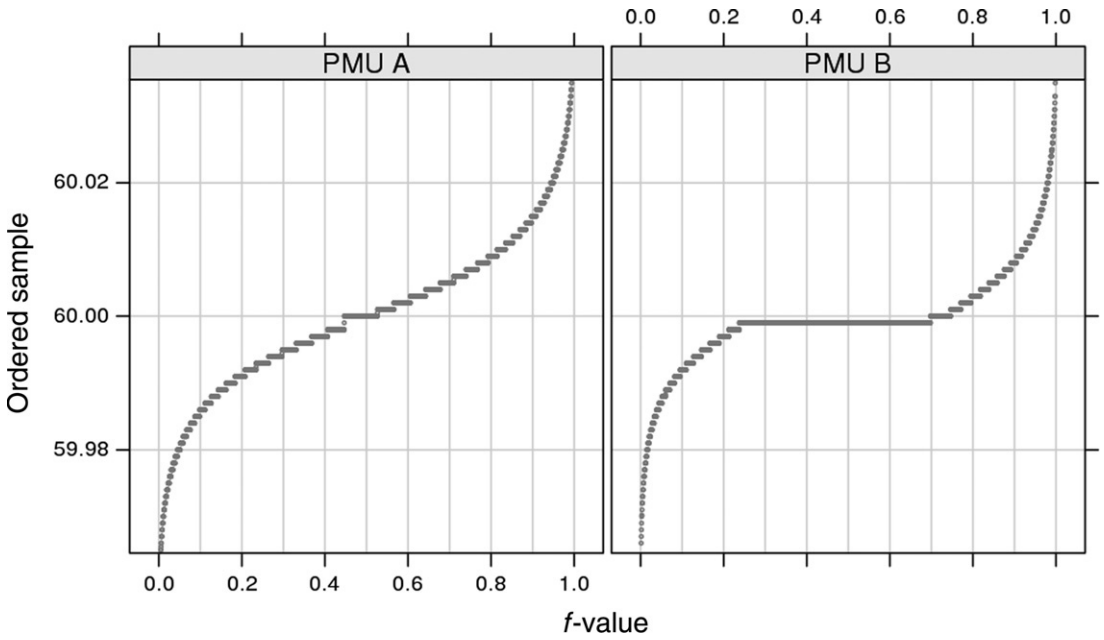


Figure 1.3
Normal-quantile plots of frequency for two real PMUs.

the current PMU and flag for which frequencies are being tabulated. For the sake of brevity, we omit the code for this example but encourage the reader to implement this task on the synthetic data provided.

When applying this to the real data, we found that for any flag greater than or equal to 128, the frequency is virtually always set at −1 (59.999 HZ). This insight, combined with visual confirmation that the values did not appear to be plausible, implied that these flags are indicative of bad data. This was later confirmed by another expert.

1.4.2.4 Distribution of Repeated Values

Even after filtering the additional bad data flags, the quantile plots indicated some values that occurred more frequently than expected. Investigating the frequency plots further, we identified some cases where extended sequences of repeated frequency values occurred. This was quantified by calculating the distribution of the sequence length of repeated values.

The sequence length distribution is calculated by stepping through a 5-min block of data and, for each PMU and given frequency value x, finding all runs of x and counting the run length. Then the number of times each run length occurs is tabulated. This can be achieved with:

```
map.repeat <- rhmap({
   colNames <- colnames(r)
```

```
    freqColumns <- which(grepl("freq", colNames))
    pmuName <- gsub("(.*)\\.freq", "\\1", colNames[freqColumns])

    for(j in seq_along(freqColumns)) { # step through frequency columns
        curFreq <- r[,freqColumns[j]]
        curFreq <- curFreq[!is.na(curFreq)] # omit missing values
        changeIndex <- which(diff(curFreq) != 0) # find index of changes
        changeIndex <- c(0, changeIndex, length(curFreq)) # pad with beg/end
        runLengths <- diff(changeIndex) # run length is diff between changes
        runValues <- curFreq[changeIndex[-1]] # get value assoc with lengths

        uRunValues <- unique(runValues)
        for(val in uRunValues) { # for each unique runValue tabulate lengths
            repeatTab <- table(runLengths[runValues == val])
            rhcollect(
                list(pmuName[j], val),
                data.frame(
                    level = as.integer(names(repeatTab)),
                    count = as.numeric(repeatTab)
                )
            )
        }
    }
})
```

`curFreq` is the current frequency time series for which we are tabulating runs, with missing values omitted. We find all places where `curFreq` changes from one number to another by finding where all first differences are not zero, storing the result as `changeIndex`. We calculate `runLengths` by taking the first difference of `changeIndex`. Then, for each unique `runValue`, we tabulate the count of each sequence length and pass it to the reduce.

Because this tabulation in the map occurs on a single 5-min window, the largest possible sequence length is 9000 (5 min * 30 records/s). For our purposes, this is acceptable because we are looking for impossibly long sequences of values, and a value repeating for even 1 min or more is impossibly long. We do not need to accurately count the length of every run; we only identify those repetitions that are too long to occur in a correctly working system. Further, if a sequence happens to slip by the filter, for example, because it is equally split across files, there is little harm done. An exact calculation could be made, but the additional algorithmic complexity is not worthwhile in our case.

Since we emit data frames of the same format as those expected by `reduce.tab` (Section 1.4), we simply define it as our reduce function. This reuse is enabled because we precede the tabulation with a nontrivial transformation of the data to sequence lengths. As previously noted, many map algorithms can be defined to generate input to this function. The job is executed by:

```
repeatTab <- rhwatch(
    map=map.repeat, reduce=reduce.tab,
    input="blocks5min", output="frequency_repeated"
)
```

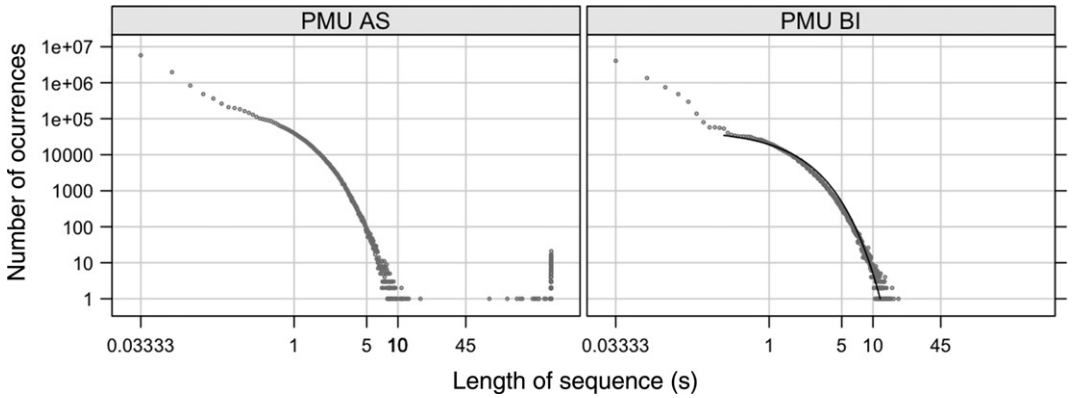


Figure 1.4

Tabulation of sequence lengths for repeated zero values for 2 real PMU frequency series. The solid black line indicates the tail of a fitted geometric distribution.

And we can visualize the results for repeated 60 HZ frequencies by:

```
repeatZero <- do.call(rbind, lapply(repeatTab, function(x) {
   if(x[[1]][[2]] == 0) {
      data.frame(pmu=x[[1]][[1]], x[[2]])
   }
}))
library(lattice)
xyplot(log2(count) ~ log2(level) | pmu, data=repeatZero)
```

The lattice package plots the distribution of repeated 60H values by PMU, shown for two PMUs in Figure 1.4 for the real PMU data.

Figure 1.4 shows this plot for real PMUs on a log-log scale. The solid black line indicates the tail of a geometric probability distribution fit to the tail of the repeated value distribution. We can use this observation to set limits based on the estimated geometric parameters which we would likely never expect to see a run length exceed. PMU AS in Figure 1.4 shows several cases well beyond the tail of a legitimate distribution. These sequences of well over 45 s correspond to bad data records. An interesting side effect of this analysis was that domain experts were quite surprised that sequence lengths as long as 3 seconds are not only completely plausible according to the distribution—but actually occur frequently in practice.

1.4.2.5 White Noise

A final type of bad sensor data occurs when the data have a high degree of randomness. Because the sensors model a physical system, with real limitations on how fast state can change, a high degree of autocorrelation in frequency is expected. When data values are essentially random, the sensor is clearly not operating correctly and the values should be

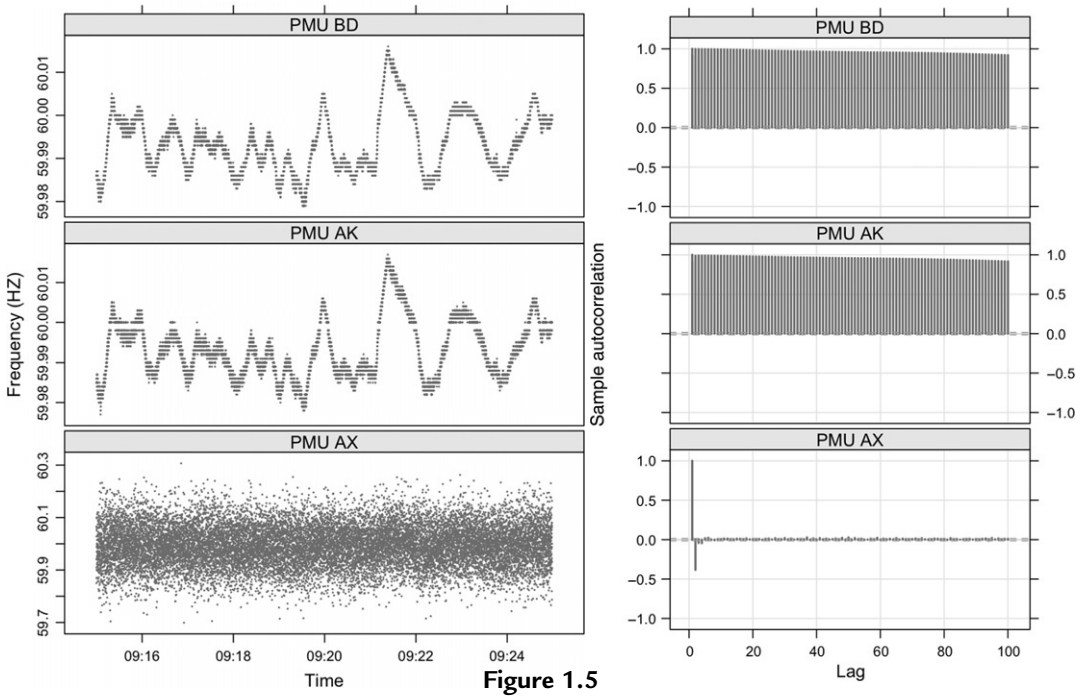


Figure 1.5
Time series (left) and sample autocorrelation function (right) for three real PMU 5-min frequency series.

discarded—even when they fall within a normal range. We call this effect "white noise." The first column in Figure 1.5 shows two PMUs with a normal frequency distribution and a third producing white noise.

One way to detect this behavior is to apply the Ljung-Box test statistic to each block of data. This determines whether any of a group of autocorrelations is significantly different from zero. As seen in the second column of Figure 1.5, the autocorrelation function for the regular frequency series trails off very slowly, whereas it quickly decreases for the final PMU indicating a lack of correlation. Applying this test to the data identifies local regions where this behavior is occurring.

This is relatively straightforward to implement:

```
map.ljung <- rhmap({
   colNames <- colnames(r)
   freqColumns <- which(grepl("freq", colNames))
   pmuName <- gsub("(.*)\\.freq", "\\1", colNames[freqColumns])

   # apply Box.test() to each column
   pvalues <- apply(r[,freqColumns], 2, function(x) {
      boxres <- try(
```

```
        Box.test(x, lag=10, type="Ljung-Box")$p.value,
        silent=TRUE
      )
      ifelse(inherits(boxres, "try-error"), NA, boxres)
   })

   rhcollect(
      "1",
      data.frame(time=k, t(pvalues))
   )
})
```

This map uses the built-in R function, `Box.test()` and builds the result across all `map.values` before calling `rhcollect()`. It also provides an example of handling R errors by checking to see if the `Box.test()` call produced an error. If RHIPE detects R errors, the job will not be completed so it is important this error is caught and handled by the map function. The map output key in all cases is simply "1," which is an arbitrary choice which ensures all map outputs go to the same reduce.

The reduce collates the *P*-values and uses `reduce.rbind` to convert them into a single data frame of *P*-values.

```
reduce.rbind <- expression(
   pre = {
      res <- NULL
   },
   reduce = {
      res <- rbind(res, do.call(rbind, reduce.values))
   },
   post = {
      rhcollect(reduce.key, res)
   }
)

ljungPval <- rhwatch(
   map=map.ljung, reduce=reduce.rbind,
   input="blocks5min", output="frequency_ljung"
)
```

We can now search for PMUs and time intervals with nonsignificant *P*-values. For the simulated data, we see that the results show the detection of the white noise that was inserted into the `AA.freq` series at the 10th time step.

1.4.3 Event Extraction

Now that we have a suite of data cleaning tools, we can proceed with our initial goal of finding events of interest in the data. In this section, we highlight two types of events: out-of-sync (OOS) frequency events and generator trips.

1.4.3.1 OOS Frequency Events

The power grid is a large, connected, synchronized machine. As a result, the frequency measured at any given time should be nearly the same irrespective of location. If frequency at one group of locations is different from another group of locations for a prolonged amount of time, there is an *islanding* of the grid. Islanding is a rare event in which a portion of the grid is disconnected from the rest, resulting in a network "island." This is one example an OOS frequency event, a general term for events where sensors appear to be measuring disconnected networks.

Finding significant differences between two PMU data streams requires first characterizing a "typical" difference. The distribution of all pairwise frequency differences between PMUs was calculated, and the upper quantiles of these distributions were defined as the cutoffs beyond which the difference is significant. As one might hypothesize, the variability of the frequency difference between two locations is greater when the locations are geographically farther apart. As a result, in practice, the cutoff value for significant PMU pair differences varies. For simplicity, a fixed cutoff of 1/100 HZ is used throughout this section.

To find all regions where there is a significant, persistent difference between the frequency for different PMUs, all pairwise differences must be considered:

```
map.oos <- rhmap({
   colNames <- colnames(r)
   freqColumns <- which(grepl("freq", colNames))
   pmuName <- gsub("(.*)\\.freq", "\\1", colNames[freqColumns])

   # make r only contain frequency information
   tt <- r$time
   r <- r[,freqColumns]
   names(r) <- pmuName

   # get all combinations of pairs
   freqPairs <- combn(ncol(r), 2)
   freqPairNames <- rbind(
      names(r)[freqPairs[2,]], names(r)[freqPairs[1,]]
   )


   # loop through all pairs and look for significant differences
   for(i in 1:ncol(freqPairs)) {
      s1 <- freqPairs[1,i]
      s2 <- freqPairs[2,i]
      isSignif <- ifelse(abs(r[,s1] - r[,s2]) > 10, 1, 0)

      changeIndex <- which(diff(isSignif) != 0) # find index of changes
      changeIndex <- c(0, changeIndex, length(isSignif)) # pad
      runLengths <- diff(changeIndex) # run length is diff between changes
      runValues <- isSignif[changeIndex[-1]]
```

```
      # we are interested in 1's that repeat more than 90 times
      signifIndex <- which(runValues == 1 & runLengths > 90)

      for(ix in signifIndex) {
         rhcollect(
         freqPairNames[,i],
         data.frame(time=tt[changeIndex[-1][ix]], length=runLengths[ix])
      )
      }
   }
})

oosFreq <- rhwatch(
   map=map.oos,
   reduce=reduce.rbind,
   input="blocks5min",
   output="frequency_outofsync"
)
```

The `combn()` function generates all combinations of PMU names. The absolute difference between the frequency series is calculated for each pair and checked to see if there is a persistent significant difference between the two. Three seconds (90 records) was chosen to represent "persistent" although this can be adjusted. If there is a significant difference, the beginning time of the run and the run length is emitted to provide data that can be used as the basis for further investigation. There is significant similarity between this expression and the repeated sequences algorithm: times where a significant difference occurs (i.e., frequencies differ by more than 10, or 1/100 HZ) are marked by a 1, then sequences of 1 longer than 90 records (i.e., 3 s at 30 records/s) are identified. The previously defined `reduce.rbind` expression collects the results into a single data frame for each PMU pair. We leave it as an exercise to the reader to read in the results and look for any significant OOS events.

Figure 1.6 shows six representative events of the 73 events returned by the OOS frequency algorithm run against the real data set. Events 05, 44, 53, and 59 have been verified to be cases of bad data not found by previous methods. For events 05 and 59, there is a shift in frequency for some PMUs, which is not physically possible. For event 53, the frequency for one PMU has been time shifted. Event 03 is an example of a generator trip, discussed in the next section, which was tagged because of the opposing oscillations between PMUs that occur after the drop in frequency. Event 27 is an example of a line fault, which was caught by the algorithm for similar reasons as the generator trip.

1.4.3.2 Finding Generator Trip Features

The generator trip events identified by the OOS algorithm were used to study the properties of generator trips and how they can be differentiated from other events, as well as normal operations. In particular, we take a feature-based approach by identifying the unique characteristics of generator trips.

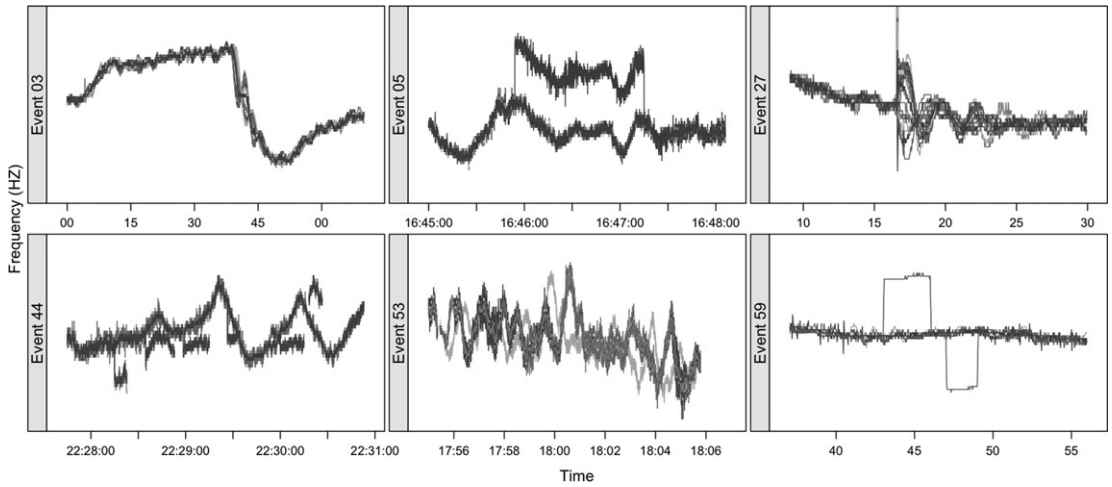


Figure 1.6
Six events from the out-of-sync detection algorithm.

Generator trips are characterized by the sudden and steep decline in frequency that occurs when a power generator goes offline. This is characterized by segmenting the data into increasing and decreasing sequences and defining a feature to be the steepest slope in that segment. In practice, additional features are required to fully specify generator trips, but for simplicity we use only this feature for the remainder of our discussion.

1.4.3.3 Creating Overlapping Frequency Data

Segmenting into increasing and decreasing sequences can be achieved through smoothing the data and identifying the critical points of the smoothed representation. Loess local polynomial regression is used to achieve the smoothing. Since local smoothing methods are not reliable at endpoints, it does not make sense to apply them to the existing 5-min block partitions. Instead, a new data set partitioning is required. A 14-s window width for loess seems to be a good choice for smoothing the data while retaining the overall pattern in the data. Adding an extra 30 s to each side of each 5-min block creates overlapping partitions which can then be smoothed correctly. Creating this new partition can be done by building on the existing one:

```
map.freqoverlap <- rhmap({
   timeVec <- r$time
   freqColumns <- which(grepl("freq", names(r)))
   r <- r[,freqColumns]
   r <- data.frame(r, time=timeVec)

   rhcollect(k, r)
   rhcollect(k-5*60, r[as.numeric(timeVec) < k+30,])
   rhcollect(k+5*60, r[as.numeric(timeVec) >= k+5*60-30,])
})
```

```
z <- rhwatch(
   map=map.freqoverlap, reduce=reduce.rbind,
   input="blocks5min", output="blocks5min_freq_overlap",
   readback=FALSE
)
```

The first `rhcollect()` emits the current 5-min time chunk, the second emits the first 30 s of the current chunk associated with the previous chunk, and the third emits the last 30 s of the current chunk associated with the next time period. `reduce.rbind` can be used as the reduce; however, in this case, it would be better to write a new reduce that sorts the resulting data frame by time. Here, we do not read the result back into R. This MapReduce job actually creates more data than its input, and with data larger than our example dataset, it would not be a good idea to try to read the result back in.

To read in sample key/value pairs for examination when the data is large, the `max` argument of `rhread()` can be useful, which limits the number of key/value pairs to be read in. Here, we look at the first 2 key/value pairs and verify that the new data blocks are 30 s longer on both ends, in which case we should have 10,800 rows instead of 9000:

```
> freqOverlap <- rhread("blocks5min_freq_overlap", max=2)
> nrow(freqOverlap[[1]][[2]])
10800
```

The generator trip algorithm is more complex than our other examples. To extract generator trip features, we create a function `getTripFeatures()`, which segments the data and calculates the maximum slope in each segment. This function takes a matrix of frequencies (`freqMat`), a time vector (`tt`), a `span` parameter (in seconds) for loess, and the minimum duration (in seconds) of a valid segment.

```
getTripFeatures <- function(freqMat, tt, span = 14, minLength = 1) {
   # get the mean frequency of the 38 PMUs at each time point
   freqMeans <- apply(freqMat, 1, function(x) mean(x, na.rm=TRUE))
   nobs <- length(freqMeans)

   startTime <- ceiling(span * 30 / 2)
   endTime <- floor(nobs - span * 30 / 2)

   # apply loess smoothing to the time point-wise means
   smoothFreq <- predict(
      loess(freqMeans ~ c(1:length(freqMeans)), degree=2,
         span=span * 30 / nobs, family="symmetric",
         control=loess.control(sufrace="direct")),
      newdata=1:nobs
   )

   # get information about each increasing and decreasing segment
   slopeDir <- sign(diff(smoothFreq))
   changePoints <- which(abs(diff(slopeDir)) == 2) + 1
   starts <- c(1, changePoints)
   ends <- c(changePoints - 1, length(smoothFreq))
```

```
   ind <- starts > span*30/2 &
      ends-starts > minLength*30 &
      ends < nobs - span*30/2
   starts <- starts[ind]
   ends <- ends[ind]
   signs <- sign(smoothFreq[ends] - smoothFreq[starts])

   sliceVec <- rep(NA, nobs)
   features <- do.call(rbind, lapply(seq_along(starts), function(i) {
      sliceVec[starts[i]:ends[i]] <<- i

      data.frame(
         start = tt[starts[i]],
         duration = (ends[i] - starts[i]) / 30,
         magnitude = abs(smoothFreq[starts[i]] - smoothFreq[ends[i]]),
         largestSlopeChange =
            max(abs(diff(smoothFreq[starts[i]:ends[i]]))) * signs[i]
      )
   }))

   list(
      features = features,
      data = data.frame(time=tt, freqMeans=freqMeans,
         smoothFreq=smoothFreq, slice=sliceVec)
   )
}
```

The function takes the frequency matrix data and calculates the point-wise mean (for a given time point, the mean of the 38 PMUs). Then it calculates a smooth representation of the averages, called `smoothFreq`, and finds all points at which the slope changes from positive to negative. Finally, it steps through each segment and calculates the maximum slope, the magnitude (difference from peak to trough), and the duration.

The algorithm can be visualized by looking at one key/value pair from our synthetic data set and plotting the result.

```
v <- freqOverlap[[2]][[2]] # get the value of the second key/value pair
v <- v[order(v$time),] # make sure they are ordered
freqColumns <- which(grepl("freq", colnames(v)))

vFeat <- getTripFeatures(v[,freqColumns], v$time)

xyplot(smoothFreq ~ time, groups=slice, data=vFeat$data, type="l", lwd=3,
   panel=function(x, y, ...) {
      panel.points(vFeat$data$time, vFeat$data$freqMeans, col="lightgray")
      panel.xyplot(x, y, ...)
   }
)
```

The smooth line represents the loess fit to the data, and the alternating colors represent different segments of increasing and decreasing slope.

Running the event detection algorithm over the entire data set requires each map process to have a local version of `getTripFeatures()`. This can be accomplished by saving the function to disk and reading it in using RHIPE `setup` expression, which specifies the code executed prior to each map or reduce step (e.g., auxiliary functions, load packages, data objects).

```
map.gentrip <- rhmap({
   r <- r[order(r$time),] # make sure it is ordered
   freqColumns <- which(grepl("freq", colnames(r)))

   rhcollect(k, getTripFeatures(r[,freqColumns], r$time)$features)
})
tripFeat <- rhwatch(
   map=map.gentrip, reduce=reduce.rbind,
   input="blocks5min", output="frequency_gentrip"
)
```

Note that `getTripFeatures()` is only defined in our local R session, but RHIPE automatically notices that it is used in the `map` expression and makes it available during MapReduce. Behind the scenes, the `getTripFeatures()` data object is stored to HDFS for all nodes to load as a shared environment. This occurs because of the `parameters` argument in `rhwatch()` which by default searches the map and reduce expressions for data objects that might be relied on that only exist in your local session.

1.5 Discussion and Conclusions

This chapter has presented an overview of using exploratory analysis techniques to analyze multi-TB power grid sensor data. We began motivating our analysis with a brief summary of the power grid and our primary source of sensor data, PMUs. Immediately following this discussion, we introduced the three foundational concepts of this chapter: MapReduce, Hadoop, and RHIPE.

Exploratory analysis is an iterative and interactive process, where the best approach and final results are not known *a priori*. As we have worked with the PMU data, we have alternated between analysis over the entire data set to identify subsets of interesting data and detailed analysis over these subsets to characterize the associated events. Typically, results of one analysis have led to additional analyses being required. Instead of attempting to provide a detailed description of our process, this chapter has focused on the algorithms we found useful for analyzing our sensor data set with the goal of providing the reader tools appropriate for analyzing their own data.

Most of the methods discussed in this chapter deal with exploration of the data to find erroneous records. Data cleaning is an important and time-intensive part of the data mining

process (Dasu and Johnson, 2003). The discovery of data quality issues is rarely achieved solely by application of automated routines known in advance. Furthermore, manual inspection of large data sets is impossible and relying on data subsets greatly increases the risk of missing infrequent but important errors. Instead, an iterative exploratory strategy that incorporates domain knowledge is usually required to identify key types of errors hidden within the data. For example, initial application of our OOS algorithm highlighted new types of errors in the sensor data that needed to be removed before OOS events could be effectively identified.

In addition to data cleaning, event extraction benefits from an exploratory approach since the underlying characteristics of the event can be extracted from the underlying data instead of a conceptual model. This feature-based approach increases the likelihood of identifying the targeted events without significant numbers of false positives. The event extraction examples included in this chapter provide insight into the complexity of analysis that can be performed with a package such as RHIPE while maintaining the scalability required to analyze a large data set.

Because of the variation in data set sizes inherent in exploratory analysis—from multiple terabytes when analyzing the entire data set to only a few kilobytes for detailed analysis of a specific data subset—it is important to use a scalable infrastructure. This eliminates the need to recode algorithms and improves analyst efficiency. We have found the combination of R and Hadoop, instantiated in the RHIPE package, to provide an excellent combination of flexibility and scalability.

Appendix

To allow the reader to follow the examples the following R code can be used to generate 10 5-min windows of synthetic data with somewhat similar properties as that of the real data. Details of what this code is doing are not important.

```
pmuNames <- apply(expand.grid(LETTERS, LETTERS)[,2:1], 1, function(x) paste(x, sep="", collapse=""))[1:38]
colNames <- as.vector(t(sapply(c("flag", "freq"), function(x) paste(pmuNames, x, sep="."))))
```

```
library(MASS)
set.seed(4321)
Sigma <- matrix(nrow=38, ncol=38, data=0.99)
diag(Sigma) <- 1
innov <- mvrnorm(n=4001, rep(0, 38), Sigma) * 0.8
freqs <- apply(innov, 2, function(x) {
     tmp <- arima.sim(3001, model=list(ar=c(0.9771, 0.1647, 0.0361, -0.1121, 0.0245, -0.1473)), start.innov=x[1:1000], innov=x[-c(1:1000)])
```

```
    as.integer(splinefun(seq(1, 90030, by=30), tmp)(1:90000)+rnorm(90000, sd=0.2))
})
flags <- matrix(nrow=90000, data=sample(c(rep(0, 2000), 128:132), 90000*38,
replace=TRUE))
freqs[flags >= 128] <- -1
pmudat <- data.frame(matrix(nrow=90000, ncol=38*2))
flagColIndex <- seq(1, 38*2, by=2)
pmudat[,flagColIndex] <- flags
pmudat[,flagColIndex + 1] <- freqs
colnames(pmudat) <- colNames

startTimes <- as.integer(as.POSIXct("2012-01-01", tz="UTC")) + c(0:9) * 5*60

pmudat <- lapply(seq_along(startTimes), function(i) {
   ind <- ((i-1) * 9000+1):(i * 9000)
   list(
      startTimes[i],
      data.frame(pmudat[ind,], time=as.POSIXct(startTimes[i],
         origin="1970-01-01", tz="UTC") + c(0:8999) / 30)
   )
})
# insert some bad data: a repeated sequence, white noise and a freq shift
pmudat[[5]][[2]][2000:3000,8] <- 0
pmudat[[10]][[2]][,2] <- rnorm(9000, sd=3.6)
pmudat[[2]][[2]][1000:2000,2] <- pmudat[[2]][[2]][1000:2000,2] + 20
```

Due to the seed being set based on the timestamp, the output of the above code should look the same in the reader's session. Here is a check for what the data should look like, looking at the first key/value pair:

```
> pmudat[[1]][[1]]
[1] 1325376000
> str(pmudat[[1]][[2]])
'data.frame': 9000 obs. of 77 variables:
 $ AA.flag: num 0 0 0 0 0 0 0 0 0 0 ...
 $ AA.freq: num 8 7 7 7 7 7 7 7 7 7 ...
...
 $ time : POSIXct, format: "2012-01-01 00:00:00" ...
```

Note that the frequency columns, denoted with the name suffix ".freq" depict frequency as an integer around zero. The raw data report frequency as the 1000 deviation of the frequency from 60 HZ. We now have a data set, `pmudat`, which is in key/value pair format ready to be written to the file system. We can achieve this with the `rhwrite()` function:

```
> library(Rhipe)
> rhinit()
> hdfs.setwd("/") # set this to the desired base directory
> rhwrite(pmudat, "blocks5min_matrix")
Wrote 10 pairs occupying 55700364 bytes
```

This will serve as the input for most of the examples in Section 1.3.3.2.2.

A useful operation with this data for stepping through the map expressions in examples is to create the `map.keys` and `map.values` objects from, say, the first 3 key/value pairs of `blocks5min_matrix`:

```
first3 <- rhread("blocks5min_matrix", max=3)
map.keys <- lapply(first3, "[[", 1)
map.values <- lapply(first3, "[[", 2)
```

References

Adler, D., Glaser, C., Nenadic, O., Oehlschlagel, J., Zucchini, W., 2012. ff: memory-efficient storage of large data on disk and fast access functions. http://CRAN.R-project.org/package-ff.

Revolution Analytics, 2012. Retrieved from http://www.revolutionanalytics.com.

Revolution Analytics, 2012. rmr: R and Hadoop Streaming Connector.

R Core Team, 2012. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, http://www.R-project.org.

Dasu, T., Johnson, T., 2003. Exploratory data mining and data cleaning.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM. 51 (1), 107–113.

DOE, 2012. Annual Energy Outlook. US Department of Energy Information Administration, DOE/EIA-0383.

Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Ann. Eugen.. 7 (2), 179–188.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B., et al., 2012. Large complex data: divide and recombine (D&R) with RHIPE. Stat. 1 (1), 53–67.

Kane, M., Emerson, J., 2011. Bigmemory: manage massive matrices with shared memory and memory-mapped files. http://CRAN.R-project.org/package=bigmemory.

Long, J., 2012. segue: a segue into parallel processing on Amazon's Web Services. http://code.google.com/p/segue.

McCallum, Q., Weston, S., 2011. *Parallel R*. O'Reilly Media, 2011.

RHIPE, 2012. http://www.rhipe.org.

Silverstein, A., 2012. NASPI Update and Technology Roadmap. *NASPI Update to the NERC Planning and Operating Committee, Dec 2011*.

Tierney, L., Rossini, A.J., Li, N., Sevcikova, H., 2012. snow: Simple Network of Workstations. http://CRAN.R-project.org/package=snow.

Urbanek, S., 2011. Multicore: parallel processing of R code on machines with multiple cores or CPUs. http://CRAN.R-project.org/package=multicore.

White, T., 2010. Hadoop: The Definitive Guide.

Wulf, W.A., 2000. Greatest achievements and grand challenges. The Bridge. 30 (3&4), 5–10.

CHAPTER 2

Picturing Bayesian Classifiers: A Visual Data Mining Approach to Parameters Optimization

Giorgio Maria Di Nunzio*, **Alessandro Sordoni**†

**Department of Information Engineering, University of Padua, Padua, Italy*

†*Déptartement d'Informatique et Recherche Opérationnelle, Université de Montréal, Montréal, Canada*

2.1 Introduction

In machine learning, "computers are programmed to optimize a performance criterion using example data or past experience" (Alpaydin, 2009). It is reasonable to assume that there is a hidden process that explains the data we observe. Though we do not know the details of this process, we know that it is not completely random. This enacts the possibility of finding a good and useful approximation even though we may not be able to identify the process completely. Mathematical models defined on parameters can be used for this task. The "learning" part of the model consists of choosing the parameters that optimize a performance criterion with respect to observed data.

"Application of machine learning methods to large databases is called data mining" (Alpaydin, 2009). Data mining applications can retrieve and explore existing information as well as extrapolate, predict, and derive new information from the given database. Classification is a special kind of prediction task which deals with the need of classifying items on the basis of previously classified training data. The research in this field has been very active in recent years.[1] For example, mining billions of user ratings for musical pieces to discover user profiles and predict which songs users will listen to or mining tweets content to capture users interests, to serve them with potentially interesting items, thus reducing information overload. In the literature, many successful algorithms for pattern classification, inference, and prediction have been presented (Hastie et al., 2009). Some of these techniques require that the user selects the dataset and performs some tuning on the algorithm's parameters—which are often difficult to determine

[1] http://www.sigkdd.org/.

Data Mining Applications with R

a priori. Moreover, some of these act as black boxes screening the user out of the analysis process. In this context, interpreting learned parameters and discovering the causal process underlying observed data become difficult tasks. Data mining applications may benefit significantly by providing visual feedback and summarization. This is the goal of visual data mining.

Visual data mining is a general approach, which aims to include the human in the data exploration process, thus gaining benefit from his perceptual abilities. In particular, users often want to validate and explore the classifier model and its output or understand the classification rationale. To address these issues, the classification system should have an intuitive and interactive explanation capability (de Oliveira and Levkowitz, 2003; Poulin et al., 2006; Wong, 1999). Visual data mining techniques have proved to be of high value in exploratory data analysis and they also have a high potential for exploring large databases (Hansen and Johnson, 2004, Part XI, pp. 819–830). Two or three-dimensional representation is probably the most "natural" metaphor a visualization system can offer to model object relationships. This is how we perceive the world as humans: two objects that are "close" to each other are probably more similar than two objects far away. The interactive visualization and navigation of such space becomes a means to browse and explore the corpus which match predetermined characteristics (Ankerst et al., 2000; Becker, 1997; Becks et al., 2000; Chalmers and Chitson, 1992; Harrell, 2006; Kohonen, 1995; Leban et al., 2006; Mozina et al., 2004; Poulet, 2008; Poulin et al., 2006; Rohrer et al., 1999; Seifert and Lex, 2009; Wise et al., 1995).

In this chapter, we focus our attention on the Bayesian classifier. Although based upon strong conditional dependence assumptions, this classifier is still widely used in practice most likely because of its tradeoff between very efficient model training and good empirical results. These characteristics make this type of classifier very suitable for analyzing large-scale datasets and synthesizing huge amounts of heterogeneous data quickly. The chapter is organized in two main parts: we present the Bayesian framework which characterizes the nature of the classification problem by introducing Bayesian data analysis; then we describe a visualization tool to support the classification process.

The chapter is divided into the following sections: we discuss related works in Section 2.2; in Section 2.3, we give the motivations of our work and list the requirements of the R code. Section 2.4 presents the Bayesian probabilistic framework we use to describe the Naïve Bayes (NB) classifiers. The two-dimensional visualization system is described in Section 2.5. Final considerations are given in Section 2.7.

2.2 Related Works

Some works in the literature have specifically tackled the problem of the visualization of NB classifiers and to this extent can be directly compared to our visualization tools. The Evidence Visualizer (Becker et al., 2002) can display Bayes model decisions as pie and bar charts.

In particular, the rows of pie charts represent each attribute, and each pie chart represents an interval or value of the attribute. The attributes are listed in order of usefulness for predicting the label. The height of the pies shows the number record having a particular value. In Mozina et al. (2004), the authors show how to adapt Naïve Bayesian classifiers and present them with a visualization technique called nomograms (Harrell, 2006). One of the main benefits of this approach is the simple and clear visualization of the complete model and the quantitative information it contains. The visualization can be used both for exploratory analysis and for decision making. It can also be used to compare different models, including those coming from logistic regression. ExplainD (Poulin et al., 2006) is a framework for explaining decisions made by classifiers that use additive evidence. It has been applied to different linear models such as support vector machines, logistic regression, and NB. The main goal of this framework is to visually explain the decisions of machine-learned classifiers and the evidence for those decisions. There are five explanation capabilities: Decision, Decision Evidence, Decision Speculation, Ranks of Evidence, Source of Evidence. Each successive capability increases the user's ability to understand and audit an aspect of the classification process, on the basis of the evidence. The Class Radial Visualization (Seifert and Lex, 2009) is an integrated visualization system that provides interactive mechanisms for a deep analysis of classification results and procedures. In this system, class items are displayed as squares and equally distributed around the perimeter of a circle. Objects to be classified are displayed as colored points in the circle and the distance between the point and the squares represent the uncertainty of assigning that object to the class.

It is worth mentioning another approach that does not specially address the NB classifier but resembles our idea of visualizing the line of decision of the classifier. In Poulet (2008), the author presents two interactive methods to improve the results of a classification task: the first is an interactive decision tree construction algorithm with a help mechanism based on SVM; the second is a visualization method used to explain the SVM results. This method can also be used to help the user in the parameter tuning step of the SVM algorithm and reduce significantly the time needed for the classification.

2.3 Motivations and Requirements

In Becker (1997), a list of desired requirements for the visualization of the structure of probabilistic classifiers are discussed. We summarize the main points here since we use them as a basis for the design of our visualization tool. In particular, users should be able:

1. to quickly grasp the primary factors influencing the classification;
2. to see the whole model and understand how it applies to classification objects, rather than the visualization being specific to a single object;
3. to compare the relative evidence contributed by every value of every attribute;

4. to see a characterization of a given class, that is a list of attributes that differentiate that class from others;
5. to infer record counts and confidence in the shown probabilities so that the reliability of the classifier's prediction for specific values can be assessed quickly from the graphics;
6. to interact with the visualization to perform classification.

Moreover, there is one last requirement which concerns the system:

7. the system should handle many attributes without creating an incomprehensible visualization or a scene that is impractical to manipulate.

In this chapter, we present a state-of-the-art visualization tool for Bayesian classifiers that can help the user understand why a classifier makes the predictions it does given a vector of parameters in input (Di Nunzio and Sordoni, 2012). The user can interact with the classifier by: (i) selecting different parametric distributions, (ii) choosing different smoothing methods, and (iii) changing the models' parameters.

During the requirements analysis, we analyzed the possible alternatives currently available for interactive R plots. The first choice was RGGobi which is one of the most powerful R tools for interactive data analysis (Lang et al., 2011). However, in our case, it was not possible to adapt it to the particular two-dimensional visualization model. A possible option could have been the RStudio[2] environment with its very useful "manipulate" package (RStudio, 2011). Unfortunately, this package is available only within the RStudio environment and we did not want to force a user to install the whole RStudio framework. For these reasons, we decided to create our own data visualization system on the basis of the R packages which are available on the Comprehensive R Archive Network.[3] In particular, we make use of the *ggplot2* plotting system for R that has a strong underlying model which supports the production of any kind of statistical graphic.[4]

2.3.1 R Packages Requirements

The code developed for this chapter requires the following R packages:

- *gWidgets*: this provides a toolkit-independent API for building interactive GUIs (Verzani, 2012);
- *gWidgetsRGtk2*: this allows the gWidgets API to use the RGtk2 package allowing the use of the GTK libraries within R (Lawrence and Verzani, 2012);

[2] http://www.rstudio.org/.

[3] http://cran.r-project.org/.

[4] http://had.co.nz/ggplot2/.

- *cairoDevice*: this allows the user to embed an R plot in a GTK user interface constructed with RGtk2 (Lawrence, 2011);
- *ggplot2*: this is an implementation of the grammar of graphics in R (Wickham, 2009);
- *tm*: a text mining framework in R (Feinerer et al., 2008);
- *Matrix*: implementation of dense and sparse matrices and fast operations on them using Lapack and SuiteSparse (Bates and Maechler, 2007).

2.4 Probabilistic Framework of NB Classifiers

In this section, we present the Bayesian framework, which characterizes the nature of the classification problem. This framework is based upon two commonly used assumptions: the data is produced by a mixture model, and there is a one-to-one correspondence between the mixture components and classes (Domingos and Pazzani, 1997; Nigam et al., 1998).

In Section 2.1, we described machine learning as the problem of guessing the process that explains the data we observe. In the classification problem, we want to characterize every object which is generated according to a probability distribution given by a mixture model. In Section 2.4.1, 2.4.2 and 2.4.3, we address the problem of choosing the parametric form of the model. For the moment, we can generically assume that all the parameters of interest are stored into a vector θ. Mixture components are encoded in a random variable $C = \{c_1, \ldots, c_n\}$ and objects in a random variable $O = \{o_1, \ldots, o_d\}$. The generative process for a classification object o_j consists of selecting a component according to the class probabilities $P(C = c_i; \theta) \equiv P(c_i; \theta)$, then picking o_j with probability $P(O = o_j | c_i; \theta) \equiv P(o_j | c_i; \theta)$. The underlying probabilistic assumption is that each object is generated by one given class. Thus, the probability of observing o_j can be found by marginalizing out the class variable:

$$P(o_j; \theta) = \sum_{i=1}^{n} P(o_j | c_i; \theta) P(c_i; \theta) \tag{2.1}$$

Let us suppose that an "oracle" tells us the optimal estimate $\hat{\theta}$ of the parameters of this problem. How do we find the class c^* of an unlabeled object o_j? One straightforward solution consists of selecting the mode of the posterior probability distribution of the class variable given the object:

$$c^* = \arg\max_{c_i} P\left(c_i | o_j; \hat{\theta}\right) = \arg\max_{c_i} \frac{P\left(o_j | c_i; \hat{\theta}\right) P\left(c_i; \hat{\theta}\right)}{P\left(o_j; \hat{\theta}\right)} = \arg\max_{c_i} P\left(o_j, c_i; \hat{\theta}\right), \tag{2.2}$$

which is obtained by a simple application of the Bayes' rule and by considering that $P(o_j; \theta)$ does not depend on c_i. In pattern classification problems, objects usually belong to more than one class (this problem is usually known as a multilabel classification). For practical and efficiency reasons, instead of building one classifier able to "attach" n labels, it is easier to build n binary classifiers able to decide whether the object belongs to class c_i or not, $\bar{c}_i = C - c_i$. This is called binary classification.

Before going into more detail, we shall present the very core "object" of our system: the `nb` function reported in Listing 2.1.

This function creates a generic empty "object" (the list `inst`, line 12) that contains all the information about the model: its parametric form and smoothing type, feature counts, sufficient statistics, and estimated parameters. These can be seen as the necessary tools to compute Equations (2.1) and (2.2). It is valuable to consider the step done in line 13: the parametric form specified in the variable `model` is added to the object class. This enables us to use R powerful "generic" function paradigm which turns out to be useful in dealing with the estimation phase (see Section 2.4.1).

In the following subsections, we describe in more detail some parts of the classification model that we believe are important for the visualization tool:

- How do we choose the parametric form of our model?
- Can we tune the estimate $\hat{\theta}$ of the optimal parameters θ?
- Once we have the estimate, can we measure how far we are from the optimal decision?

2.4.1 Choosing the Model

Choosing the model of our problem means finding the right mathematical description according to some hypotheses we can make. Actually, there are two things to consider: how to model the distribution of the class variable $P(c_i; \theta)$ and how to model the distribution of the objects given a class $P(o_j|c_i; \theta)$.

The choice of the model for the first point is easier when we consider the binary classification problem: the class we observe can be either c_i or $\bar{c}_i$. This binary setting can well be modeled by a Bernoulli random variable:

$$c_i \sim \text{Bern}(\theta_c), \tag{2.3}$$

Listing 2.1 `nb` Function (File *nb.R*)

```
1 nb <- function (model = "bernoulli", smoothing = "laplace") {
2  # Initialize a simple Naive-Bayes binary classifier
3  #
4  # Args:
```

```
 5 #  model: The parametric form of the features distribution.
 6 #  smoothing: The smoothing method to be used for the estimation.
 7 #
 8 # Returns:
 9 #  A Naive-Bayes classifier ready to be estimated.
10 # Initialize the object and the basic structures
11 inst <- list()
12 class(inst) <- c("nb", model)
13 inst$type <- model
14 inst$smoothing <- smoothing
15 inst$classes.names <- list()
16 inst$features.num <- 0
17 inst$examples.num <- 0
18 # Create distributional parameters
19 inst$smoothing.params <- list()
20 inst$features.params <- list()
21 inst$classes.params <- list()
22 # Sufficient statistics for the objects' parametric distribution
23 inst$features.freq <- list()
24 # Data structure to speed-up computation
25 inst$features.freq.tot <- list()
26 return(inst)
27 }
```

where the notation θ_c is used to specify the parameters relevant to the class distribution.

In the *nb.R* file (line 22), the parameters of the mixture are stored under the list key `classes.params`. The estimation of the Bernoulli parameter θ_c calculated by maximum-likelihood estimation (MLE) is simply the number of examples of that class divided by the total number of examples (see Section 2.4.2 for a discussion about smoothing). The *R* code for this estimation process is given in Listing 2.2.

Here, `x` is the `nb` model under estimation, `x$examples.num` is the total number of examples, `labels` a vector which specifies which objects belong to that category, and `x$classes.names` the names of the classes for this classification problem. Constructing a `table` from the labels vector is a useful trick to automatically count the number of examples in each class.

Listing 2.2 **Parameters of the Mixture (File *nb.R*)**

```
1 # Bernoulli parameter for each class
2 x$classes.params <- as.vector(table(labels)[x$classes.names]) /
x$examples.num
```

The choice of the mathematical model for the distribution of the objects requires more attention. Hereafter, we present three possible models to parametrize the conditional probability $P(o_j|c_i;\,\theta)$.

2.4.1.1 Multivariate Bernoulli model

In the multivariate Bernoulli model, an object is a binary vector over the space of features. Given a set of features F, $|F| = m$, each object o_j is represented as a vector of m Bernoulli random variables $o_j \equiv (f_1, \ldots, f_m)$ such that:

$$f_k \sim \text{Bern}\left(\theta_{f_k|c}\right) \tag{2.4}$$

We can write the probability of an object by using the NB conditional independence assumption which states that feature variables are independent given the class variable. Formally:

$$P\left(o_j|c_i;\theta\right) = \prod_{k=1}^{m} P(f_k|c_i;\theta) = \prod_{k=1}^{m} \theta_{f_k|c_i}^{x_k}\left(1-\theta_{f_k|c_i}\right)^{1-x_k}, \tag{2.5}$$

where x_k is either 0 or 1 indicating whether feature f_k is present or absent in object o_j. When the number of features is very large, this product goes quickly to zero. For this reason, a monotonic transformation, such as a log transformation, is usually performed:

$$\log\left(P\left(o_j|c_i;\theta\right)\right) = \log\left(\prod_{k=1}^{m} P(f_k|c_i;\theta)\right) = \sum_{k=1}^{m} x_k \log\left(\frac{\theta_{f_k|c_i}}{1-\theta_{f_k|c_i}}\right) + \sum_{k=1}^{m} \log\left(1-\theta_{f_k|c_i}\right), \tag{2.6}$$

This last equation is not only a way to avoid arithmetical anomalies but also a very efficient implementation of the same calculation: the last sum on the right-hand side can be precomputed and the first term ranges only over the features appearing in the current object ($x_k = 1$).

2.4.1.2 Multinomial Model

In contrast to the multivariate Bernoulli model, in the multinomial model, we have one single random variable F, which can take values over the set of features. By assuming that each feature event is independent of the other, an object o_j is represented as a vector of frequencies whose entries correspond to features. This vector is drawn from a multinomial distribution:

$$o_j \equiv \left(N_{1,j}, \ldots, N_{m,j}\right) \sim \text{Multinomial}\left(\theta_{f|c}\right) \tag{2.7}$$

where $N_{k,j}$ indicates the number of times the feature f_k appears in the object o_j. Note that, in this case, $\sum_{k=1}^{m} \theta_{f_k|c} = 1$. The probability is proportional to:

$$P\left(o_j|c_i;\theta\right) \propto \prod_{k=1}^{m} \theta_{f_k|c_i}^{N_{k,j}} \tag{2.8}$$

Again, we can use a monotonic transformation to avoid zero-probabilities:

$$\log\left(P\left(o_j|c_i;\theta\right)\right) \propto \sum_{k=1}^{m} N_{k,j} \log\left(\theta_{f_k|c_i}\right) \tag{2.9}$$

2.4.1.3 Poisson Model

In the Poisson model, an object is generated by a multivariate Poisson random variable. Each object o_j is represented as an m-dimensional vector of frequencies, $o_j \equiv (N_{1,j},\ldots,N_{m,j})$, and each feature count is governed by a Poisson random variable:

$$N_{k,j} \sim \text{Pois}\left(\theta_{f_k|c}\right) \tag{2.10}$$

Using the NB conditional independence assumption, we can write the probability of the object as:

$$P\left(o_j|c_i;\theta\right) \propto \prod_{k=1}^{m} \theta_{f_k|c_i}^{N_{k,j}} e^{-\theta_{f_k|c_i}}, \tag{2.11}$$

and, by taking the logs we obtain:

$$\log\left(P\left(o_j|c_i;\theta\right)\right) \propto \sum_{k=1}^{m} \left(N_{k,j} \log\left(\theta_{f_k|c_i}\right) - \theta_{f_k|c_i}\right) \tag{2.12}$$

With these concepts in hand, we can easily apply Equation (2.2) to calculate the posterior distribution $P(c_i|o_j;\theta)$ and classify the unlabeled object o_j. The generic function `nbClassify` allows us to call the correct classification function according to the class of the object `x` that is passed as argument. An example of the implementation of the classification function using the Bernoulli model (Equation 2.6) is reported in Listing 2.3.

The method returns the joint probabilities $P(o_j, c_i; \hat{\theta})$ for the two classes, c_i and $\bar{c}_i$, for all the objects' vectors contained in the `dataset` matrix.

Listing 2.3 Classification Function for the Bernoulli Model (File *nb.R*)

```
nbClassify.bernoulli <- function(x, dataset) {
 dataset <- prepareDataset(x, dataset)

 # precompute the sum over features of log(1 - param)
 sumnegative <- colSums(log(1 - x$features.params) )

 # compute p(d, c)
 logfeatures <- log(x$features.params)
 lognfeatures <- log(1 - x$features.params)
 scores <- dataset %*% (logfeatures - lognfeatures)
 scores <- t(t(scores) + sumnegative + log(x$classes.params) )

 return(scores)
}
```

2.4.2 Estimating the Parameters

When the model and the features which characterize the objects are defined, the next step is to estimate the parameters of our model θ. In the examples of the previous sections, we let an "oracle" tell the vectors of estimates of the parameters. In a realistic situation, how can we estimate these parameters?

In general, there are two main paths you may follow to accomplish this task: the MLE or the Bayesian approach to estimation.[5] With MLE, under the hypothesis that the data D contains observations, which are independent and identically distributed, the estimate $\hat{\theta}^{MLE}$ is calculated by maximizing the likelihood of the data with respect to the parameter:

$$\hat{\theta}^{MLE} = \arg\max_{\theta} P(D|\theta) = \arg\max_{\theta} \prod_{j \in D} P\left(o_j|\theta\right) \tag{2.13}$$

For many models, like the ones we presented in this paper, a maximum-likelihood estimator can be found as an explicit function of the observed data. However, due to data sparseness, the MLE of the probability of unseen features tends to be zero (Gelman et al., 2003). To prevent this undesirable "zero-probability" behavior, we need to smooth the estimates of our parameters: the estimation of a parameter is strongly tied to smoothing techniques.

In a Bayesian framework, smoothing is implicitly considered by treating model parameters θ as random variables governed by a probability distribution, or *prior*, indicated as $P(\theta)$. The prior can be seen as encoding our "beliefs" about how θ should behave. Intuitively, this solves the problem of zero-probability features because we relax the strict dependency of the estimates upon the statistics gathered from data. The parameters controlling the prior are usually called *hyper-parameters*.[6] Being Bayesian means adjusting the prior distribution upon observed data, known as computing the *posterior* given the data $P(\theta|D)$. This is achieved through the application of Bayes' rule:

$$\overbrace{P(\theta|D)}^{\text{posterior}} = \frac{\overbrace{P(D|\theta)}^{\text{likelihood}}\overbrace{P(\theta)}^{\text{prior}}}{P(D)}, \tag{2.14}$$

where D represents our data, and $P(D)$ is the probability of the particular instance D according to some generative model of the data.

[5] Actually, there is a third way of estimating the probabilities which is the Maximum a Posteriori (MAP) estimator and a complete account of it falls outside the scope of this chapter.

[6] The prefix "hyper" is used to distinguish the parameters of the prior from the parameters of the original model, like θ.

The Bayesian estimator $\hat{\theta}^B$ of the parameter is the posterior mean:

$$\hat{\theta}^B = E_{\theta|D}[\theta] = \int \theta P(\theta|D)\mathrm{d}\theta \tag{2.15}$$

Equation (2.15) can be mathematically hard to solve because of the integration over a product of two functions, $P(\theta|D)$ and $P(\theta)$. One way to approach this problem is to find the "conjugate" prior of the likelihood function $P(D|\theta)$ that makes the posterior function $P(\theta|D)$ come out with the same functional form as the prior. If the likelihood belongs to the exponential family, there always exists a conjugate prior. NB models have a likelihood of this type:

- The multivariate Bernoulli model conjugate prior is the Beta distribution Beta(θ; α, β),
- The Multinomial model conjugate prior is the distribution Dir(θ; $\vec{\alpha}$),
- The multivariate Poisson conjugate prior is the Gamma distribution Gamma(θ; α, β).

where α, β are hyper-parameters, and $\vec{\alpha} = \alpha_1, \ldots, \alpha_m$ is a vector of hyper-parameters, one for each feature.

A comparison of the two estimation methods is given in Table 2.1. We can easily see that the zero-probability behavior arises in MLE when the counts $n_{k,i}$ or $N_{k,i}$ are zero. By contrast, Bayesian estimation adds "pseudo-counts" that avoid the problem. Our visualization tool implements three kind of smoothing methods: Laplace, Bayesian, and Fixed Interpolation. The Laplace method is a special case of the Bayesian estimation and is obtained by setting the hyper-parameters $\alpha = 1$, $\beta = 1$ and $\alpha_k = 1$, for each k. The Fixed interpolation method solves the problem by interpolating the MLE estimator with a "collection" model, in which the statistics are gathered regardless of the class variable. A Fixed interpolation estimator $\theta^I_{f_k|c_i}$ for a Bernoulli model would be:

$$\theta^I_{f_k|c_i} = \lambda \frac{n_{k,i}}{n_i} + (1-\lambda)\frac{\sum_i n_{k,i}}{\sum_i n_i}, \tag{2.16}$$

where $\lambda \in [0, 1]$ is a free parameter.

Table 2.1 Differences Between the MLE and the Bayesian Approach Where $n_{k,i}$ Is the Number of Objects Belonging to c_i in Which Feature f_k Appears and n_i the Total Number of Objects in c_i, $N_{k,i}$ the Frequency of Feature f_k in c_i, α_k the kth Component of Vector $\vec{\alpha}$

Model	$\hat{\theta}^{MLE}$	$\hat{\theta}^B$
Bernoulli	$\frac{n_{k,i}}{n_i}$	$\frac{n_{k,i}+\alpha}{n_i+\alpha+\beta}$
Multinomial	$\frac{N_{k,i}}{\sum_k (N_{k,i})}$	$\frac{N_{k,i}+\alpha_k}{\sum_k (N_{k,i}+\alpha_k)}$
Poisson	$\frac{N_{k,i}}{n_i}$	$\frac{N_{k,i}+\alpha}{n_i+\alpha+\beta}$

The generic function *nbEstimate* (Listing 2.4) gathers sufficient statistics from the vectors contained in the variable `dataset` and selects the correct estimation function according to the class of the model `x` we have chosen. Specifically these two steps are: (i) count the occurrences of the dataset and store them in the model (Listing 2.4); (ii) call the generic function `nbUpdate` which computes posterior estimate by plugging the hyper-parameters (Listing 2.5).

Listing 2.4 Estimation of Bernoulli Model Parameters (File *nb.R*)

```
# Estimate Bernoulli parameters
nbEstimate.bernoulli <- function(x, dataset, labels, params) {
  # Preprocess dataset for Bernoulli
  dataset <- prepareDataset(x, dataset)
  # Initialize dataset specific variables
  # Sorted class names
  x$classes.names <- sort(unique(labels), decreasing=TRUE)
  x$smoothing.params <- params
  # Number of unique features
  x$features.num <- dim(dataset)[2]
  x$examples.num <- dim(dataset)[1]
  # Compute sufficient statistics for each class
  # features.freq is a features x class matrix
  # features.freq.tot is a tot_features x class matrix for the
current model
  x$features.freq <- sapply(x$classes.names,
                            function(r) colSums(dataset[c(labels==r),]))
  x$features.freq.tot <- as.vector(table(labels)[x$classes.
names])
  # Bernoulli parameter for each class
  x$classes.params <- as.vector(table(labels)[x$classes.names])
/ x$examples.num
  # Compute probability estimates
  x <- nbUpdate(x, params)
  return(x)
}
```

Listing 2.5 Updating Bernoulli Parameters (File *nb.R*)

```
# Update estimates
nbUpdate <- function(x, ...) {
  UseMethod("nbUpdate")
}
# Update Bernoulli estimates
nbUpdate.bernoulli <- function(x, params) {
```

```
 7  # Laplace smoothing for Bernoulli
 8  if (x$smoothing == "laplace") {
 9    x$features.params <- t(t(x$features.freq + 1) / (x$features.
freq.tot + 2) )
10  }
11  # Beta prior
12  else if (x$smoothing == "prior") {
13   x$smoothing.params$alpha <- (alpha <- params$alpha)
14   x$smoothing.params$beta <- (beta <- params$beta)
15   x$features.params <- t(t(x$features.freq + alpha) /
                            (x$features.freq.tot + (alpha + beta) ) )
16  }
17  # Jelinek-Mercer interpolation
18  else if (x$smoothing == "interpolation") {
19    x$smoothing.params$lambda <- (lambda <- params$lambda)
20    features.freqs  <- rowSums(x$features.freq)
21    collection.freqs  <- sum(x$features.freq.tot)
22    x$features.params <- (1 - lambda) * t(t(x$features.freq) / x
$features.freq.tot
        ) + lambda * (features.freqs / collection.freqs)
23  }
24  return(x)
25 }
```

2.5 Two-Dimensional Visualization System

The model which upholds the visualization tool defines a direct relationship between the probability of an object given a category of interest and a point on a two-dimensional space (Di Nunzio, 2009). The idea is to associate each object of the dataset to a point in the two-dimensional space: the abscissa reflects how much the object is relevant to the category, the ordinate reflects how much the object is not relevant to the category. In this light, it is possible to graph entire collections of objects on a Cartesian plane. Remembering that in a binary classification setting we classify the object o_j under category c_i if

$$P\left(c_i|o_j;\hat{\theta}\right) > P\left(\bar{c}_i|o_j;\hat{\theta}\right), \tag{2.17}$$

we can get the two coordinates of a NB classifier by simply considering:

$$\underbrace{P\left(c_i|o_j;\hat{\theta}\right)}_{x} > \underbrace{P\left(\bar{c}_i|o_j;\hat{\theta}\right)}_{y} \tag{2.18}$$

In Figure 2.1a, three points in this two-dimensional space are shown together with the decision line. The visual metaphor is immediate: if the point is "below" the decision line, the object is classified under category c_i (because $x > y$), whereas if the point is "above" the line, the object is

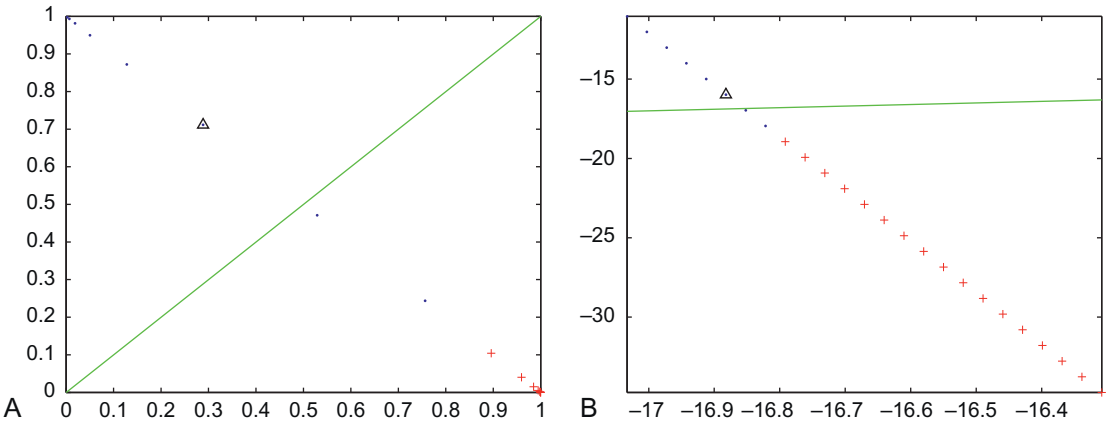


Figure 2.1
2D visualization example with normalized probabilities (left) and nonnormalized log probabilities (right).

classified under category $\bar{c}_i$ (because $x < y$). Points that lie exactly on the line are those for which we need to take an explicit decision (because they have the same chance of belonging to either category).[7] The further the point is from the line, the higher the confidence in the classification. In this figure, the point highlighted with a triangle would be classified under c_i.

2.5.1 Design Choices

The two-dimensional visualization described earlier may present the following problems:

- In a binary classification problem $P\left(c_i|o_j;\hat{\theta}\right) = 1 - P\left(\bar{c}_i|o_j;\hat{\theta}\right)$ and all the points are on the segment (0,1)-(1,0), see Figure 2.1a. With datasets of the size of thousands of objects, the points may result too cluttered.
- There may be arithmetical anomalies given the fact that the product of the probability of the features goes rapidly to zero.[8] Either $P\left(c_i|o_j;\hat{\theta}\right)$ or $P\left(\bar{c}_i|o_j;\hat{\theta}\right)$ can be approximated to zero given an insufficient number of bits.

To avoid points cluttering, we take the nonnormalized probabilities:

$$P\left(o_j|c_i;\hat{\theta}\right)P\left(c_i;\hat{\theta}\right) > P\left(o_j|\bar{c}_i;\hat{\theta}\right)P\left(\bar{c}_i;\hat{\theta}\right) \tag{2.19}$$

which means not dividing by the probability of the data $P(o_j|\theta)$. This does not change the result of the decision as it was justified in Equation (2.2). To eliminate the arithmetical anomaly, we take the logarithm of both sides of the equation:

$$\log\left(P\left(o_j|c_i;\hat{\theta}\right)\right) + \log\left(P\left(c_i;\hat{\theta}\right)\right) > \log\left(P\left(o_j|\bar{c}_i;\hat{\theta}\right)\right) + \log\left(P\left(\bar{c}_i;\hat{\theta}\right)\right) \tag{2.20}$$

[7] In this chapter, we classify these documents under category c_i.

[8] If we have 50 features and the probability of each feature given a class is 0.5, the product is $0.5^{50} \sim 10^{-15}$.

Figure 2.1a shows the result of this transformation. Note that we are now in the third quadrant of the cartesian plane, because we are adding the logarithm of probabilities, therefore the logarithms of numbers between 0 and 1.

2.5.2 Visualization Design

The system we propose consists of several visual components that correspond to the points addressed in Section 2.4. The main components of the system are:

- *View Panel*: this displays the two-dimensional plot of the dataset according to the choices of the user.
- *Interaction Panel*: this allows for the interaction between the user and the parameters of the probabilistic models.
- *Performance Panel*: this displays the performance measures of the model.

Figure 2.2 shows the Main window and the panels. The interactivity of the plot is realized by means of the *gWidgets* package and the layout is realized by grouping widgets together.

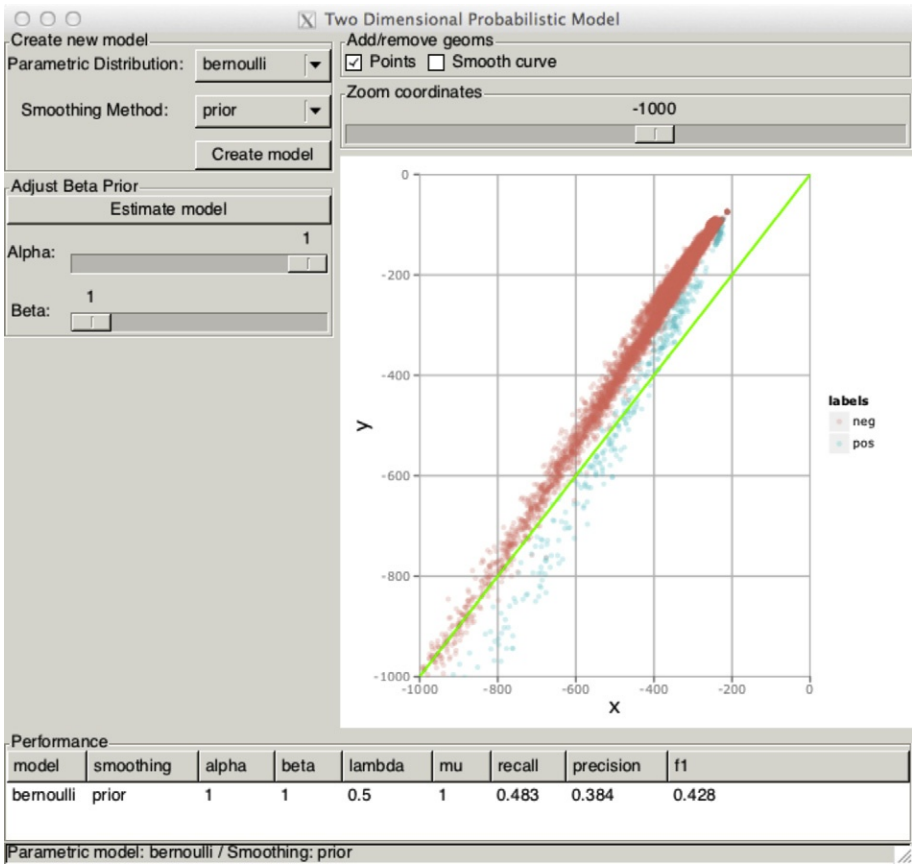


Figure 2.2
Visualization tool: Main window.

The link between the interactive window and the probabilistic model starts when the *hndleCreateModel* function is called (Listing 2.6).

Listing 2.6 Initialize Model by Calling the *hndleCreateModel* Function (File *twodm.R*).

```
hndleCreateModel <- function (h, ...) {

  # Get selected model
  model <- svalue(combo.model)
  # Get selected smoothing method
  smoothing <- svalue(combo.smooth)

  # Delete parameters layouts
  for (w in list.layouts) {
    delete(frm.params, w)
  }

  # Build widgets according to prior and model
  if (smoothing == "prior") {
    if (model == "bernoulli") {
      names(frm.params) <- "Adjust Beta Prior"
      add(frm.params, list.layouts$bernprior)
    } else if (model == "poisson") {
      names(frm.params) <- "Adjust Gamma Prior"
      add(frm.params, list.layouts$bernprior)
    } else {
      names(frm.params) <- "Adjust Dirichlet Prior"
      add(frm.params, list.layouts$multiprior)
    }
  } else if (smoothing == "interpolation") {
    names(frm.params) <- "Adjust Interpolation Factor"
    add(frm.params, list.layouts$interp)
  }

  # Initialize parameters
  params <<- lapply(list.sliders, svalue)

  # Create the model
  nbmodel <<- nb(model, smoothing)
  nbmodel <<- nbEstimate(nbmodel, dataset, labels, params)

  # Update everything
  fireUpdate()

  # Update status bar value
  svalue(status.bar) <- paste("Parametric model:", model, "/
Smoothing:",
        smoothing)
}
```

From line 21 to line 40, we first hide the interactive widgets that are used to tune the prior hyper-parameters and then we build them according to the desired model.

When the Estimate Model button in the Interaction Panel is clicked, the window requests an update of the coordinates to the model. The model computes the estimates of the probabilities according to the new parameters and sends the scores to the view, as shown in Listing 2.7.

Listing 2.7 **Update probabilities and plot results (file *twodm.R*)**

```
 1 fireUpdate <- function() {
 2   # Update the value of the parameters
     params <<- lapply(list.sliders, svalue)
 3   # Update the model
 4   nbmodel <<- nbUpdate(nbmodel, params)
 5   # Update scores
 6   scores <<- nbClassify(nbmodel, dataset)
 7   # Update plot and performances
 8   updatePlot()
 9   updatePerformances()
10 }
11 # Update the plot
12 updatePlot <- function(...) {
13   # Update dataframe
14   df$x <<- scores[, 1]
15   df$y <<- scores[, 2]
16   pp <- p
17   pp <- (pp %+% df)
18   # Add layers
19   if (svalue(chk.points) )
20   {
21     pp <- pp + geom_point(alpha = 0.2)
22   }
23   if (svalue(chk.smooth) )
24   {
25     pp <- pp + geom_smooth()
26   }
27   # Add decision line
28   pp <- pp + geom_abline(colour = "green")
29   # Adjust coordinate
30   pp <- pp + coord_cartesian(xlim = c(svalue(slider.coord), 0),
                               ylim = c(svalue(slider.coord), 0) )
31   # Plot
32   print(pp)
33 }
```

2.6 A Case Study: Text Classification

The task of text classification is to assign one or multiple predefined class labels to a textual document. It was a very popular research topic at the end of the 1990s with the rapid increase of text in digital form such as Web pages, newswire, and scientific literature and the need to organize them. Today, classification has witnessed a new wave of interest due to new technologies available to users or new specific tasks such as query classification, blog classification, patent classification, and medical document classification.

For this case study, we used the Reuters-21578 collection which consists of 21,578 news stories which appeared on the Reuters newswire in 1987. The actual number of documents manually assigned to categories is 12,902.

In order to replicate this case study, the following R packages and relative versions are needed:

- *cairoDevice* 2.19,
- *datasets* 2.15.2,
- *ggplot2* 0.9.2.1,
- *graphics* 2.15.2,
- *grDevices* 2.15.2,
- *gWidgets* 0.0-52,
- *gWidgetsRGtk2* 0.0-81,
- *lattice* 0.20-10,
- *Matrix* 1.0-10,
- *methods* 2.15.2,
- *stats* 2.15.2,
- *tm* 0.5-7.1,
- *utils* 2.15.2,
- *XML* 3.95-0.1.

2.6.1 Description of the Dataset

For this case study, we prepared two subsets of the same Reuters-21578 dataset:

- the folder *./data/reuters21578/* contains a random sample of about 900 zipped files of the original training set of the Reuters collection. This sample can be used to test the *tm* framework as explained in Section 2.6.2.
- the folder *./data/* contains a file named *reuters.matrix* that contains a term-document sparse matrix of the whole Reuters training set. Section 2.6.3 explains how to load this sparse matrix.
- the folder *./data/* also contains the files of the labels that must be loaded to classify documents into positive and negative examples. These files are named *name_category.labels*.

2.6.2 Creating Document-Term Matrices

When dealing with text collections, there are a number of steps that are necessary to clean noisy documents and obtain a nicely formatted machine readable dataset. We used the *tm* package, a text mining package in R that gives many options for processing raw text files into document-term matrices (Feinerer et al., 2008). Listing 2.8 shows the sequence of steps to load the Reuters dataset, precisely the two methods `loadReutersDataset` and `preprocessDataset`. The former method can easily be adapted to one's need by changing the parameters of `DirSource`.

Listing 2.8 Case Study: Load Reuters Collection (File *preprocess_dataset.R*).

```
require(tm)
require(XML)

# Use TextMining package to load and preprocess documents
loadReutersDataset <- function() {
  cat(sprintf("load dataset...\n") )
  reuters <- Corpus(DirSource("./data/reuters21578/"),
                    readerControl = list(reader = readReut21578XML) )
  return(reuters)
}

# This function preprocess an XML Corpus (tm package) into a plain text
dataset.
preprocessDataset <- function(dataset) {
    # Transform XML into plain text
    cat(sprintf("transform into plain text...\n") )
    dataset_plain <- tm_map(dataset, as.PlainTextDocument)

    # Remove extra white spaces
    cat(sprintf("remove extra white spaces...\n") )
    dataset_plain <- tm_map(dataset_plain, stripWhitespace)

    # To lower case
    cat(sprintf("letters to lower case...\n") )
    dataset_plain <- tm_map(dataset_plain, tolower)

    # Remove stopwords
    cat(sprintf("remove stopwords...\n") )
    dataset_plain <- tm_map(dataset_plain, removeWords, stopwords
("english") )

    # Stem words (currently not working on MAC OS X Lion)
    # dataset_plain <- tm_map(dataset_plain, stemDocument)

    return(dataset_plain)
}
```

2.6.3 Loading Existing Term-Document Matrices

Depending on the type of hardware available and on the size of the document collection, it might be more convenient to process the dataset outside R and produce a dataset in a sparse format: our system can handle datasets encoded in row-column-value triplets. For example, the first lines of the *reuters.matrix* are shown in Listing 2.9:

Listing 2.9 First Lines of the File *reuters.matrix*

```
1 1  70  1
2 1  1020  1
3 1  1045  1
4 1  1104  1
5 1  1121  1
6 [...]
```

where the columns contain the document id, the term id, and its frequency, respectively. This very compact representation can be loaded as a table and transformed into a sparse matrix, as shown in Listing 2.10.

Listing 2.10 Function for Loading an External Sparse Matrix File (File *preprocess_dataset.R*).

```
10 # Take in input a triplet form row,column,value
# and returns a sparse matrix
loadDataset <- function(dataset) {
  t <- read.table(dataset)
  return(sparseMatrix(t[,1], t[,2], x=t[,3]) )
}
```

When we load existing term-document matrices, we also need to load the labels that we use to classify documents into positive and negative classes. We use the *loadLabels function* shown in Listing 2.11.

Listing 2.11 Function for Loading an External Labels File (File *preprocess_dataset.R*).

```
11 # Take in input a column file of classes labels
# and returns a dense column matrix
loadLabels <- function(labels) {
 m <- as.matrix(read.table(labels) )
 colnames(m)[1] <- "labels"
 return(m)
}
```

2.6.4 Running the Program

Once a collection has been initialized, we can launch the two-dimensional view with the command `twodm/2` (Listing 2.12, line 18). Currently, the two-dimensional view supports the classification of one class at a time, thus we must specify which class we would like to examine as a parameter (Listing 2.12, line 16).

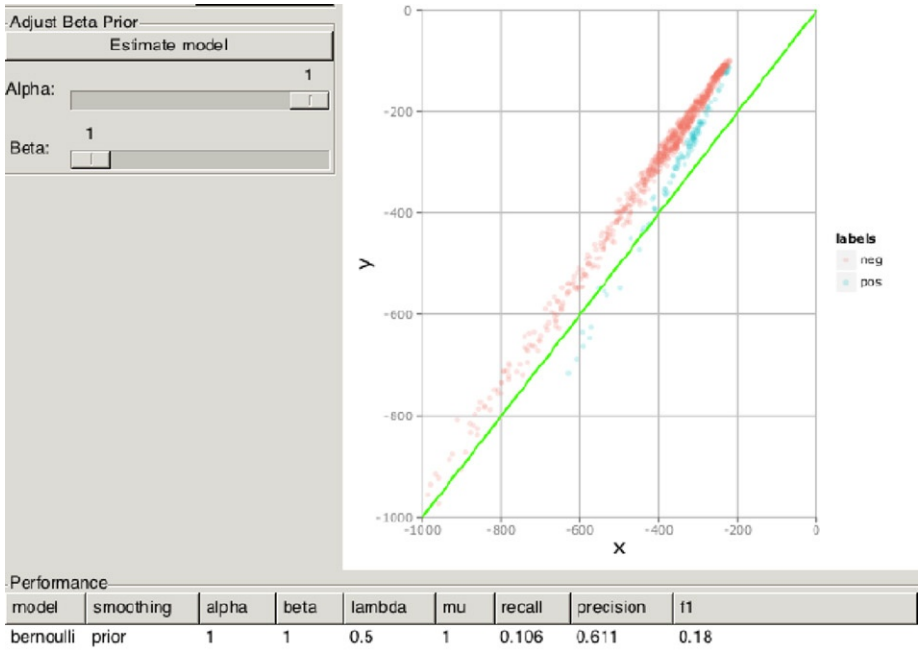
Listing 2.12 Case Study: Loading Reuters Dataset

```
 1 > source("twodm.R")
 2 > # load dataset
 3 > dataset <- loadReutersDataset()
 4 load dataset...
 5 > # transform XML documents into plain documents
 6 > dataset_plain <- preprocessDataset(dataset)
 7 transform into plain text...
 8 remove extra white spaces...
 9 letters to lower case...
10 remove stopwords...
11 > #  build document-term matrix
12 >dtm  <- createDocumentTermMatrix(dataset_plain)
13 > #  extract matrix from corpus
14 >dataset_matrix <- extractDocumentTermMatrix(dtm)
15 > #  extract labels
16 > labels <- extractLabels(dataset, "acq")
17 > # start two dimensional visualization
18 > twodm(dataset_matrix, labels)
```

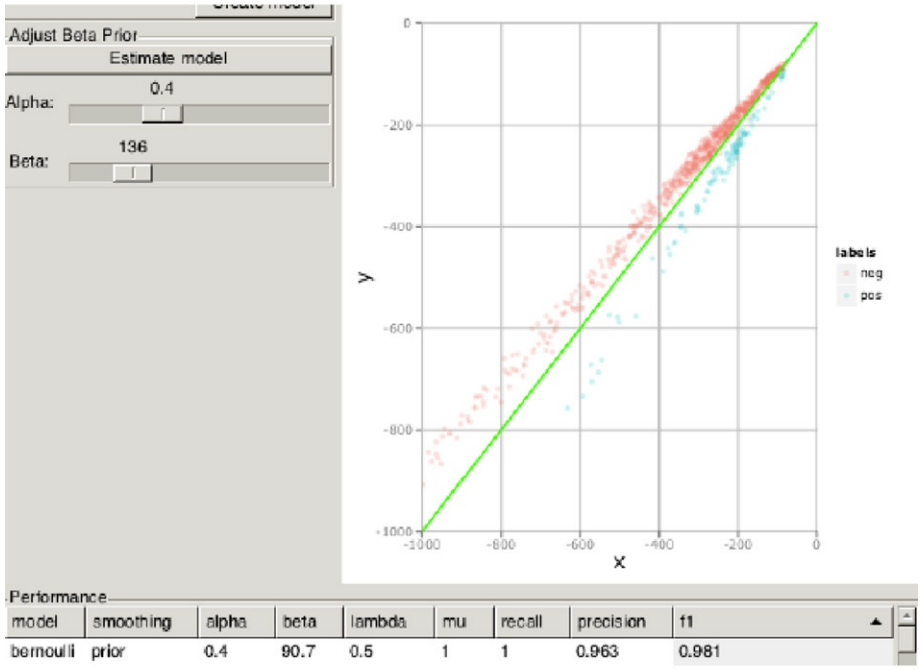
As shown in Listing 2.12, the command takes two arguments: (1) the preprocessed dataset as a sparse matrix and (2) a vector of labels indicating for each document its true class. Once the visualization is started, the result is reported in Figure 2.3a. We intentionally left the beta prior parameters equal to 1 to simulate a Laplacian smoothing ($\alpha = \beta = 1$). The visualization tool immediately shows that in theory the two sets of documents are well separated, but in practice they are misplaced with respect to the (green) decision line that is the line where $P(c_i|d) = P(\bar{c}_i|d)$. By tweaking the two hyper-parameters, we can see the effects on both the plot and in the performance panel. The results are immediately evident: the cloud of points is better aligned with the decision line and all the performance measures have significantly increased (Figure 2.3b). Note that in this figure we intentionally changed the transparency level of the points to highlight the areas of higher density.

2.6.4.1 Comparing Models

With the two-dimensional visualization, it is possible to assess the quality of a model quickly and compare it to other models. In Figures 2.4 and 2.5, the multinomial models and the Poisson model have been initialized with Laplacian smoothing (or uniform prior). In these plots, two



(a) Laplace



(b) Optimized

Figure 2.3

Case study: Bernoulli with (a) Laplacian smoothing (or Beta prior with alpha = 1, beta = 1), (b) Beta prior (alpha = 0.4, beta = 136). 2D visualization example with normalized probabilities (left) and nonnormalized log probabilities (right).

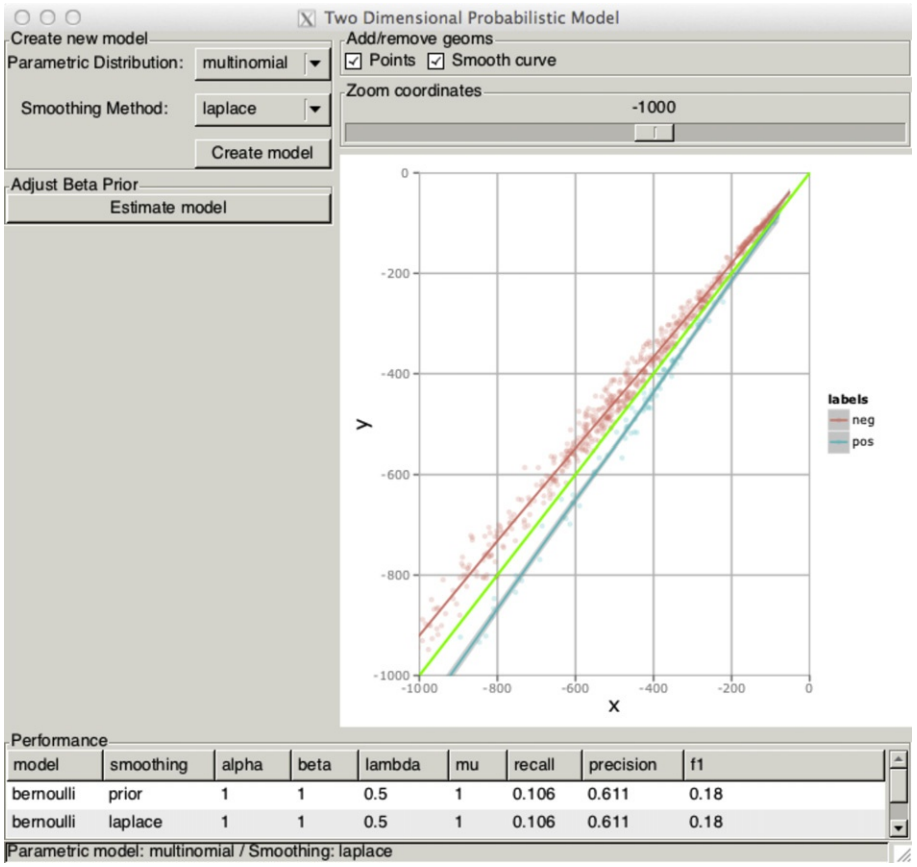


Figure 2.4
Case study: multinomial with Laplacian smoothing and fitted lines.

fitted curves, one for each class, have been computed by the `geom_smooth()` function of the *ggplot2* package. Listing 2.13 shows the function that draws the actual plot.

Listing 2.13 **Case Study: Plot Reuters Collection (File *twodm.R*).**

```
1 # Update the plot
2 updatePlot <- function(...) {
3   # Update dataframe
4   df$x <<- scores[, 1]
5   df$y <<- scores[, 2]
6   pp <- p
7   pp <- (pp %+% df)
8   # Add layers
9   if (svalue(chk.points) )
```

```
10  {
11    pp <- pp + geom_point(alpha = 0.2)
12  }
13  if (svalue(chk.smooth) )
14  {
15    pp <- pp + geom_smooth()
16  }
17  # Add decision line
18  pp <- pp + geom_abline(colour = "green")
19  # Adjust coordinate
20  pp <- pp + coord_cartesian(xlim = c(svalue(slider.coord), 0),
                               ylim = c(svalue(slider.coord), 0) )
21  # Plot
22  print(pp)
23 }
```

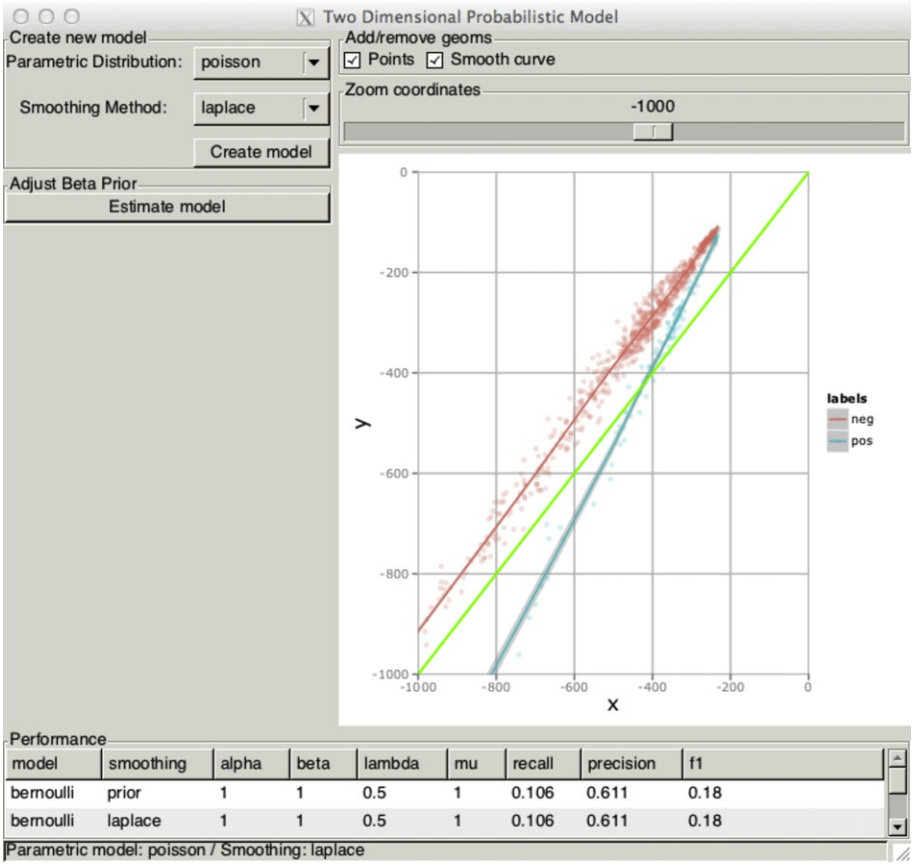


Figure 2.5
Case study: Poisson with Laplacian smoothing and fitted lines.

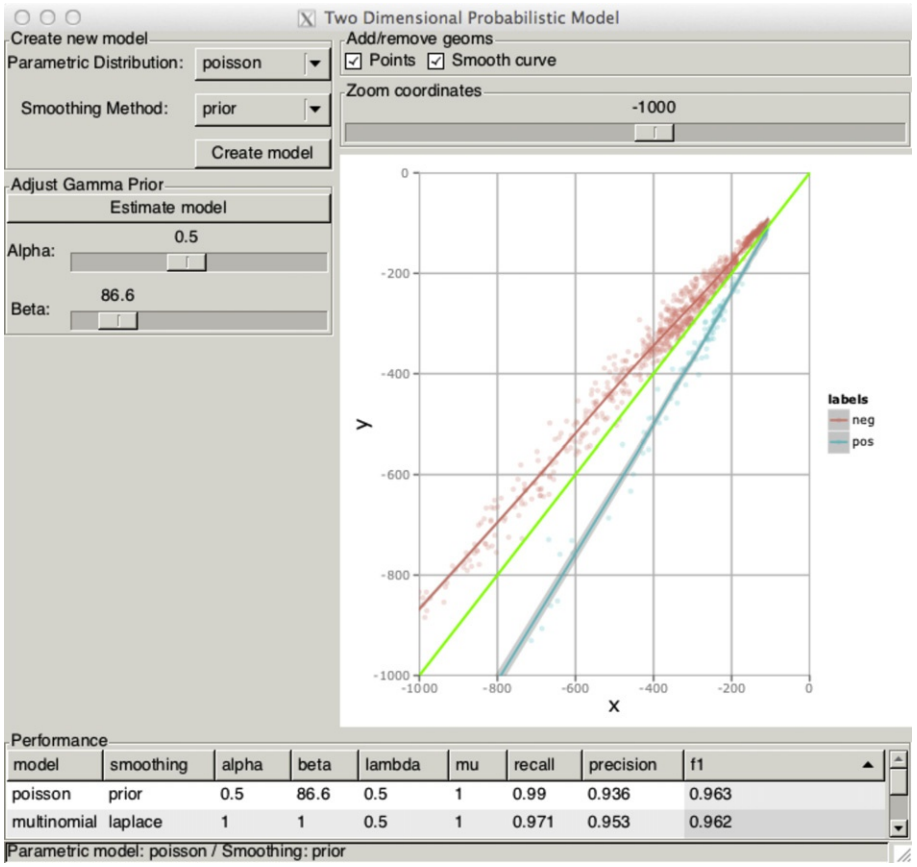


Figure 2.6
Case study: Poisson with Gamma prior smoothing and fitted lines.

In theory, when the hyper-parameters are close to the optimal values, the two lines should be perfectly aligned and specular with respect to the decision line, as, for example, in Figure 2.4.

When one of the two lines (or both) intersects the decision line, the model has not been optimized (see, for example, Figure 2.5). In these cases, a tweaking of the hyper-parameters by setting the smoothing method to prior is required. Figure 2.6 shows the performance of an optimized Poisson model. At the bottom of the window, the performances of the two models are shown.

2.7 Conclusions

In this chapter, we have presented a state-of-the-art visualization tool for Bayesian classifiers that can help (i) the user interpret the performance of a classifier and (ii) how to improve it by selecting different parametric distributions, choosing different smoothing methods, and

changing models' parameters. This two-dimensional view offers the most natural metaphor to model object relationships: the closer a document is to the decision line, the more uncertain is the classification.

We describe a Bayesian setting for modeling our prior knowledge of the distributions on the values of the parameters of the model. Within this setting, it is possible to alter the estimate of the probability of terms and consequently the decision on the classification. The direct interaction with the parameters of the model allows users and researchers to adjust the classification performance and quickly compare different alternatives.

Experimental results on a standard benchmark collection confirm that representing probabilities of Bayesian classifiers in the two-dimensional space is very useful for understanding the behavior of text classifiers for particular values of the parameters of the model. Moreover, the visualization shows that, in general, a model does not perform well not because it is not able to discriminate the categories but because the points are not aligned with the decision line.

Acknowledgments

This work has been partially supported by the QONTEXT project under grant agreement N. 247590 (FP7/2007-2013).

References

Alpaydin, E., 2009. Introduction to Machine Learning, Adaptive Computation and Machine Learning Series, second ed. The MIT Press, Cambridge, MA, USA.

Ankerst, M., Ester, M., Kriegel, H.-P., 2000. Towards an effective cooperation of the user and the computer for classification. In *KDD'00*, pages 179–188, Boston, MA, USA. ACM.

Bates, D., Maechler, M., 2007. matrix: a matrix package for r. *R package version 0.99875-2*. http://CRAN.R-project.org, 15.

Becker, B.G., 1997. Using mineset for knowledge discovery. IEEE Comput. Graph. Appl.. 17 (4), 75–78.

Becker, B., Kohavi, R., Sommerfield, D., 2002. Visualizing the simple Bayesian classifier. In: Fayyad, U., Grinstein, G.G., Wierse, A. (Eds.), Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 237–249.

Becks, A., Sklorz, S., Jarke, M., 2000. Exploring the semantic structure of technical document collections: a cooperative systems approach. In: Etzion, O., Scheuermann, P. (Eds.), CoopIS. Lecture Notes in Computer Science, 1901, Springer, Springer Berlin Heidelberg, pp. 120–125.

Chalmers, M., Chitson, P., 1992. Bead: explorations in information visualization. In: Belkin, N.J., Ingwersen, P., Pejtersen, A.M. (Eds.), SIGIR. ACM, New York, NY, USA, pp. 330–337.

de Oliveira, M.C.F., Levkowitz, H., 2003. From visual data exploration to visual data mining: a survey. IEEE Trans. Vis. Comput. Graph. 9 (3), 378–394.

Di Nunzio, G., 2009. Using scatterplots to understand and improve probabilistic models for text categorization and retrieval. Int. J. Approx. Reasoning 50 (7), 945–956.

Di Nunzio, G., Sordoni, A., 2012. A visual tool for bayesian data analysis: the impact of smoothing on Naïve Bayes text classifiers. In: SIGIR. ACM, New York, NY, USA, p. 1002.

Domingos, P., Pazzani, M.J., 1997. On the optimality of the simple Bayesian classifier under zero–one loss. Mach. Learn. 29 (2–3), 103–130.

Feinerer, I., Hornik, K., Meyer, D., 2008. Text mining infrastructure in r. J. Stat. Software. 25 (5), 1–54.

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B., 2003. Bayesian Data Analysis, Second Edition (Chapman & Hall/CRC Texts in Statistical Science), second ed. Chapman and Hall/CRC, New York, NY, USA.

Hansen, C.D., Johnson, C.R., 2004. Visualization Handbook, first ed. Academic Press, Amsterdam, The Netherlands.

Harrell, F., 2006. Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis, second ed. Springer Series in Statistics, Springer, New York, NY, USA.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed. Springer Series in Statistics, Springer, New York, NY, USA.

Kohonen, T., 1995. Self-Organizing Maps, Springer Series in Information Retrieval, second ed. Springer, New York, NY, USA.

Lang, D.T., Swayne, D., Wickham, H., Lawrence, M., 2011. RGGobi: interface between R and GGobi. R package version 2.1.17.

Lawrence, M., 2011. cairoDevice: cairo-based cross-platform antialiased graphics device driver. R package version 2.19.

Lawrence, M., Verzani, J., 2012. gWidgetsRGtk2: Toolkit implementation of gWidgets for RGtk2. R package version 0.0-81.

Leban, G., Zupan, B., Vidmar, G., Bratko, I., 2006. Vizrank: data visualization guided by machine learning. Data Min. Knowl. Discov. 13 (2), 119–136.

Mozina, M., Demsar, J., Kattan, M.W., Zupan, B., 2004. Nomograms for visualization of naive Bayesian classifier. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (Eds.), Knowledge Discovery in Databases: PKDD. Lecture Notes in Computer Science, 3202, Springer, New York, NY, USA, pp. 337–348.

Nigam, K., McCallum, A., Thrun, S., Mitchell, T.M., 1998. Learning to classify text from labeled and unlabeled documents. In: Mostow, J., Rich, C. (Eds.), AAAI/IAAI. AAAI Press/The MIT Press, Cambridge, MA, USA, pp. 792–799.

Poulet, F., 2008. Towards effective visual data mining with cooperative approaches. In: Simoff, S.J., Böhlen, M.H., Mazeika, A. (Eds.), Visual Data Mining. Lecture Notes in Computer Science, 4404, Springer, Berling Heidelberg, pp. 389–406.

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D.S., Fyshe, A., Pearcy, B., Macdonell, C., Anvik, J., 2006. Visual explanation of evidence with additive classifiers. In *AAAI*, pp. 1822–1829. AAAI Press.

Rohrer, R.M., Sibert, J.L., Ebert, D.S., 1999. A shape-based visual interface for text retrieval. IEEE Comput. Graph. Appl. 19 (5), 40–46.

RStudio, 2011. Manipulate: interactive plots for RStudio. R package version 0.96.316.

Seifert, C., Lex, E., 2009. A novel visualization approach for data-mining-related classification. In: Banissi, E., Stuart, L.J., Wyeld, T.G., Jern, M., Andrienko, G.L., Memon, N., Alhajj, R., Burkhard, R.A., Grinstein, G.G., Groth, D.P., Ursyn, A., Johansson, J., Forsell, C., Cvek, U., Trutschl, M., Marchese, F.T., Maple, C., Cowell, A.J., Moere, A.V. (Eds.), *IV*, IEEE Computer Society, 490–495.

Verzani, J., 2012. gWidgets: gWidgets API for building toolkit-independent, interactive GUIs. R package version 0.0-50.

Wickham, H., 2009. ggplot2: Elegant Graphics for Data Analysis. Springer, New York.

Wise, J.A., Thomas, J.J., Pennock, K., Lantrip, D., Pottier, M., Schur, A., Crow, V., 1995. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In: Gershon, N.D., Eick, S.G. (Eds.), INFOVIS. IEEE Computer Society, Washington, DC, USA, pp. 51–58.

Wong, P.C., 1999. Guest editor's introduction: visual data mining. IEEE Comput. Graph. Appl. 19 (5), 20–21.

CHAPTER 3

Discovery of Emergent Issues and Controversies in Anthropology Using Text Mining, Topic Modeling, and Social Network Analysis of Microblog Content

Ben Marwick
Department of Anthropology, University of Washington, Seattle, USA

3.1 Introduction

The aim of this chapter is to show some basic methods using R to analyze text content to discover emergent issues and controversies in diverse corpora. As a specific case study, I investigate the culture of microblogging academics within the dynamics of a professional conference to gain insights into the key issues and debates emergent in this community and the transformative effects of using Twitter in academic contexts. Microblogging academics can be considered a type of online community which has its own norms, rules, and communicative behaviors (Gruzd et al., 2011) that can be analyzed with anthropological methods (cf. Boellstorff, 2011; Wilson and Peterson, 2002). My hypothesis is that mining the publically available microblog text content generated in relation to the 109th Annual Meeting of the American Anthropological Association (AAA) in November 2011 can reveal the main issues and controversies that characterized the event as well as the community structure of the people generating the corpus. Although the duration of the meeting represents a narrow slice of Twitter content, it is ideal for looking at which academics are tweeting and why they tweet because academic meetings are a period of highly concentrated intellectual and social activity within the academic community. It is during these times that the distinctive patterns of shared learned knowledge, behaviors, and beliefs that characterize communities are most apparent (Egri, 1992). It is hoped that the methods presented will be suitable for the analysis of a wide variety of communities that generate large amounts of text content.

There are a number of unique and eventful characteristics of the 2011 meeting that make the related Twitter content especially worthy of investigation. These include organizational issues

Data Mining Applications with R

such as the session was convened in response to controversy surrounding the removal of the word "science" from the AAA's long-range plan statement in 2010 (Boellstorff, 2011; Lende, 2011), the AAA Presidential Address that discussed the 2010 final report of the Commission on Race and Racism in Anthropology, and the revision of the AAA code of ethics. Beyond these major organizational topics, other issues that were prominent at the time of the meeting were the Occupy movement and the future of scholarly publishing. Analysis of the Twitter messages relating to these issues gives insights into the behavior of microblogging anthropologists and their fit within the structure and culture of the discipline. Since Twitter postings are highly accessible to the public, this chapter also reveals the potential of evaluating how anthropologists use Twitter as a public face of the discipline.

Among academics in general, Twitter use is relatively rare with Priem et al. (2011) finding that about 2.5% of 8826 scholars at five U.K. and U.S. universities used Twitter weekly. Priem et al. found that no academic rank or discipline was significantly overrepresented in their sample. They also noted that although Twitter is popular as a scholarly medium for making announcements, linking to articles, and engaging in discussions about methods and literature, about 60% of the messages were personal. The use of Twitter at academic conferences has also been the subject of a number of systematic analyses, mostly aiming to identify how Twitter is used in this context and who benefits from it (Ebner, 2009; Ebner and Reinhardt, 2009; Ebner et al., 2010; Letierce et al., 2010a; McCarthy and Boyd, 2005; Reinhardt et al., 2009; Ross et al., 2011). These previous studies, summarized in Table 3.1, show that microblog content from conferences can be a corpus of substantial size comprising a large number of very short documents.

Table 3.1 Summary of Related Previous Research on Microblogging of Academic Meetings

Conference	Conference Attendees (*n*)	Authors (*n* [%])	Messages (*n*)	Source
EduCamp 2010	NA	272	2110	Ebner et al. (2010)
International Semantic Web Conference 2009	405	273 [67]	1444	Letierce et al. (2010a)
Digital Humanities 2009, That Camp 2009, Digital Resources in the Arts and Humanities 2009	542	379 [70]	4574	Ross et al. (2011)
ED-MEDIA 2009	1000	173 [17.3]	1595	Ebner and Reinhardt (2009)
ED-MEDIA 2008	1000	10 [10]	54	Ebner (2009)

The number of authors as a percentage of attendees is included in square brackets.

3.2 How Many Messages and How Many Twitter-Users in the Sample?

To obtain raw data for this study, I searched the Twitter Web site (cf. Gentry, 2011) and downloaded 1500 messages that had been labeled by each message's author as relevant to the 109th Annual Meeting of the AAA (1500 messages is the maximum number of messages that the Twitter application programming interface (API) allows to download at one time). Authors of Twitter messages frequently use a shared system of notation for identifying the subject of their messages where a hash symbol is placed before the topic word or phrase (Kwak et al., 2010). In this case, the #aaa2011 hashtag was the subject identifier, so I extracted all messages containing this hashtag as follows

```
# get package with functions for interacting with Twitter.com. Note that the Twitter API is
constantly changing so this method of searching Twitter may not work in the future.
StackOverflow is a good resource for finding the most current method to search and download
content from Twitter
require(twitteR)
# get 1500 tweets with #aaa2011 tag, note that 1500 is the max, and it's subject to filtering and
other restrictions by Twitter
aaa2011 <- searchTwitter('#aaa2011', n=1500)
# convert to data frame
df <- do.call("rbind", lapply(aaa2011, as.data.frame))
# get column names to see structure of the data
names(df)
# look at the first three rows to check content
head(df,3)
# see how many unique Twitter accounts in the sample
length(unique(df$screenName))
```

In this sample, there are 307 authors whose messages span from 11:00 am EST on 17 until 6:00 pm EST on 20 November 2011. Although the #AAA2011 hashtag was in use in the weeks leading up to the meeting and continued to be used after the meeting concluded, I chose to limit the sample to those produced only during the course of the meeting. There are two reasons for this strategy. First, the scope of this study is limited to a synchronic analysis of Twitter use at the meeting as a time of intensive intellectual and social activity among anthropologists. This is a sampling strategy that has become standard in research on Twitter use in academic and professional contexts because it is a time when people are highly active on Twitter (Ebner, 2009; Ebner and Reinhardt, 2009; Ebner et al., 2010; Letierce et al., 2010a,b; Reinhardt et al., 2009). Second, the Twitter Web site is not explicit about how it makes messages publically available, so it is not always clear if Twitter reveals only a sample of messages matching the hashtag or the entire set of matching messages. During repeated sampling of the Twitter archives, I found I could only obtain a reproducible sample of messages for the period of the meeting, excluding the first day. For the days before the meeting, I was not confident that the archive was making available all the relevant messages. Furthermore, the number of messages in each sample declined with increased time after the event to the point where a few months after the event there were no Twitter messages with the #AAA2011 hashtag. This limitation

unfortunately excludes the possibility of tracking which issues generated discussion beyond the meeting and whether Twitter posts during the meeting could predict the staying power of particular topics (cf. Ebner and Reinhardt, 2009; Reinhardt et al., 2009). The unpredictable nature of the results obtained from the Twitter Web site necessitated the exclusion of days for which I could not obtain a reproducible number of messages and more broadly is a serious limitation on the reproducibility of analyses of Twitter corpora.

3.3 Who Is Writing All These Twitter Messages?

Although all the Twitter messages used in this study were publically available at the time the sample was collected, Twitter-users can hide all of their messages at any time, so for the rest of the analysis I have anonymized individual authors here to preserve their confidentiality. The authors include individual and corporate authors (such as the AAA, The Society for the Anthropology of Food and Nutrition, and Wiley-Blackwell). About half of all individual authors in the sample use pseudonyms. The degree of anonymity of the pseudonyms varied greatly. Some authors used a cryptic username unique to their Twitter account with no implied biographical information, giving absolute anonymity to the author. Some used a pseudonym on Twitter that was linked to their physical world self elsewhere on the Internet. Others used a username that could not be linked to a specific physical person, but implied a gender, academic status (e.g., graduate student, postdoctoral scholar, etc.), scholarly interests (e.g., bioanthropology, archeology, or medical anthropology) or some combination of the three.

```
# Create a new column of random numbers in place of the usernames and redraw the plots
# find out how many random numbers we need
n <- length(unique(df$screenName))
# generate a vector of random number to replace the names, we'll get four digits just for
convenience
randuser <- round(runif(n, 1000, 9999),0)
# match up a random number to a username
screenName <- unique(df$screenName)
screenName <- sapply(screenName, as.character)
randuser <- cbind(randuser, screenName)
# Now merge the random numbers with the rest of the Twitter data, and match up the correct random
numbers with multiple instances of the usernames...
rand.df <- merge(randuser, df, by="screenName")
```

The use of real names by some of the authors is notable because it links their professional identities as scholars to their authorship of their Twitter messages, giving them ownership of and accountability for their messages. This indicates a use of Twitter by some anthropologists as instrument of professional communication and makes these users visible as the informal public faces of the discipline to Twitter-users. The anonymity preferred by other anthropologists using Twitter is an indication of the heterogeneity of the Twitter-using community and the existence of individuals who prefer to maintain varying degrees of separation between their identity as a Twitter author and other dimensions of their identity (e.g., as a scholar or a student).

Of the 233 authors with a gender-identifying Twitter username (i.e., excluding unrevealing and ambiguous usernames), 128 (55%) identified as female and 104 (45%) as male. Among the 128 authors who provided enough information to determine their academic status, about half of these are graduate students (66, 49%). The next most represented group is faculty at the rank of assistant professors (23, 17%) followed by people with sessional, fixed term appointments, or nontenure track teaching appointments (14, 10%). The remainder is made up of associate professors (11, 8%), full professors (9, 7%), community college faculty (6, 5%), postdoctoral scholars (5, 3%), and undergraduates (2, 1%). In terms of academic status, it seems reasonable to conclude that more junior members of the discipline are most frequently represented on Twitter. This subset may be analogous to Prensky's (2001) "digital natives" or people whose upbringing was immersed in information and communication technologies, although the presence of more senior academics suggests a mixed group with a range of exposures to technology. Although specific ages for the authors are unavailable for this sample, the relatively small proportion of full professors relative to assistant professors and graduate students suggests that younger scholars are more often users of this form of virtual communication than older ones.

3.4 Who Are the Influential Twitter-Users in This Sample?

Figure 3.1 shows the frequency distribution of messages per author in this sample. The distribution approximately follows a power law, consistent with previous observations of Twitter usage and other online and real-life cultural phenomena (Bentley et al., 2004; Letierce et al., 2010a). Figure 3.1 shows that the majority of the messages were authored by about half a dozen individuals (most of whom used their real names, which are not given here). Figure 3.1 also shows that the most prolific authors also tend to have their messages most frequently repeated or cited by other authors. This behavior is known as retweeting and allows messages to spread beyond the network of the original message's author. Whereas the observed motivations for retweeting are numerous and difficult to disentangle (Boyd et al., 2010), the effect of retweeting is to increase the spread of the message and in turn, the author's influence on other authors. In this sample, 451 messages (30%) are retweets, a figure consistent with samples of Twitter messages from other academic conferences, but substantially higher than the 3% observed in general Twitter data (Letierce et al., 2010b). This indicates that these authors are reading and retweeting widely among their network.

```
# determine the frequency of tweets per account
counts <- table(rand.df$randuser)
# create an ordered data frame for further manipulation and plotting
countsSort <- data.frame(user = unlist(dimnames(counts)), count = sort(counts, decreasing =
TRUE), row.names = NULL)
# create a subset of those who tweeted at least 5 times or more
countsSortSubset <- subset(countsSort,countsSort$count > 5)
```

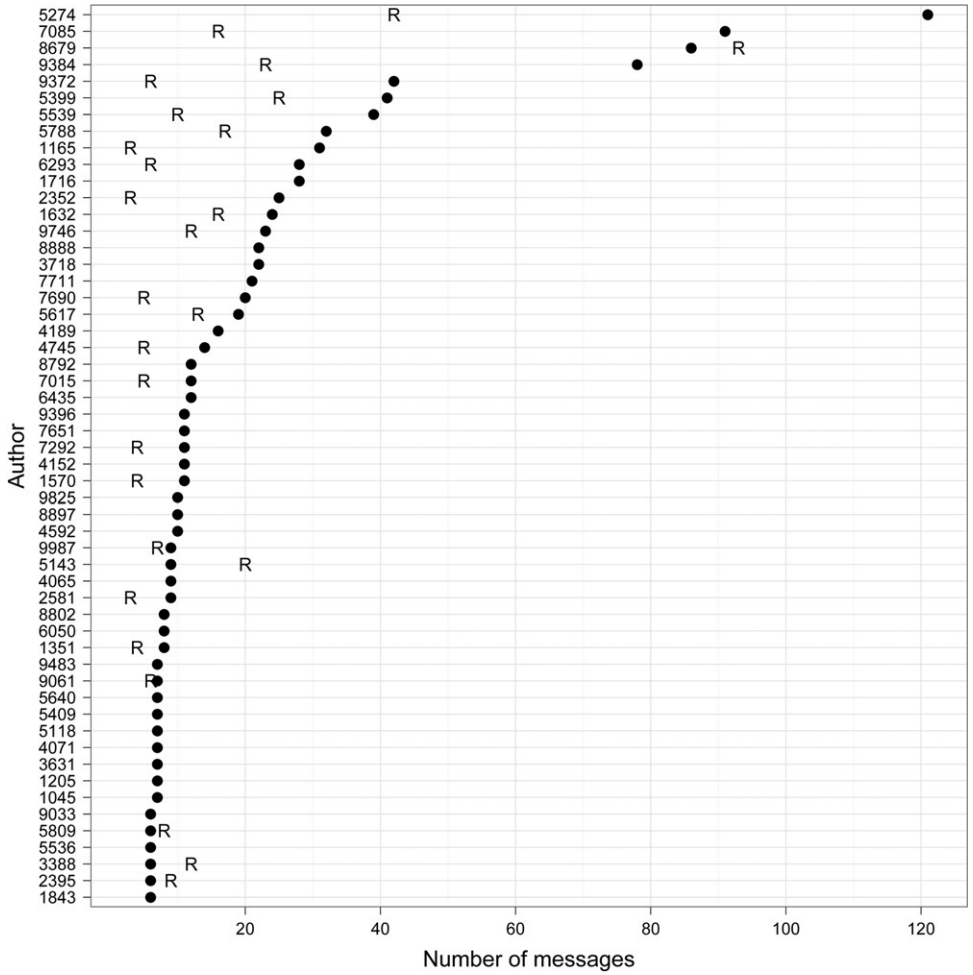


Figure 3.1

Number of messages by author, for all authors posting more than five messages (solid circle), and number of each author's messages that are repeated or cited by other authors, for all messages repeated or cited more than twice (indicated by "R").

```
# extract counts of how many tweets from each account were retweeted, this code is derived from
the excellent tutorial here http://heuristically.wordpress.com/2011/04/08/text-data-
mining-twitter-r/
# first clean the twitter messages by removing odd characters
rand.df$text=sapply(rand.df$text,function(row) iconv(row,to='UTF-8'))
# remove @ symbol from user names
trim <- function (x) sub('@','',x)
# pull out who the message is to
require(stringr)
rand.df$to <- sapply(rand.df$to,function(name) trim(name))
```

```
# extract who has been retweeted
rand.df$rt <- sapply(rand.df$text,function(tweet)
   trim(str_match(tweet,"^RT (@[[:alnum:]_]*)")[2]))
# replace names with corresponding anonymising number
randuser <- data.frame(randuser)
rand.df$rt.rand <
- as.character(randuser$randuser)[match(as.character(rand.df$rt),
                              as.character(randuser$screenName))]
# make a table with anonymised IDs and number of RTs for each account
countRT <- table(rand.df$rt.rand)
countRTSort <- sort(countRT)
# subset those people RT'd at least twice
countRTSortSubset <- subset(countRTSort,countRT>2)
# create a data frame for plotting
countRTSortSubset.df <-data.frame(people = as.factor(unlist(dimnames
(countRTSortSubset))), RT_count = as.numeric(unlist(countRTSortSubset)))
# combine tweet and retweet counts into one data frame
TweetRetweet <- merge(countsSortSubset, countRTSortSubset.df, all.x = TRUE)
# create a Cleveland dot plot of tweet counts and retweet counts per Twitter account
# solid data point = number of tweets, letter R = number of retweets
require(ggplot2)
require(grid)
ggplot() +
   geom_point(data = TweetRetweet, mapping = aes(reorder(people, count), count), size = 3) +
   geom_point(data = TweetRetweet, mapping = aes(people,
RT_count), size = 4, shape = "R") +
   xlab("Author") +
   ylab("Number of messages") +
   coord_flip() +
   theme_bw() +
   theme(axis.title.x = element_text vjust = -0.5, size = 14)) +
   theme(axis.title.y = element_text (size = 14, angle=90)) +
   theme(plot.margin = unit(c(1,1,2,2), "lines"))
```

Figure 3.2 shows that there are no clear correlations between number of messages, number of retweets, and number of followers (the number of other Twitter-users who subscribed to the messages of an author) but authors with high frequencies of messages and retweets tend to have a high number of followers (e.g., authors 8679, 5724, and 7085). There are two interesting implications from the follower data in Figure 3.2. First is that range in the number of followers is very wide, from 9 to 6979, indicating that the audiences of the authors vary from an audience of a small circle of close colleagues based on face-to-face relationships to an audience of a large number of people who might only know the author via Twitter messages. Second, the low correlation between the number of followers and the number of retweets (Kendall's $\tau = 0.28$, $p = 0.02$ from the R package "Kendall") suggests that the size of an author's Twitter audience is not a good predictor of how widely their messages are propagated.

```
# calculate the number of followers of each Twitter account
# extract the usernames from the non-anonymised dataset
users <- unique(df$screenName)
```

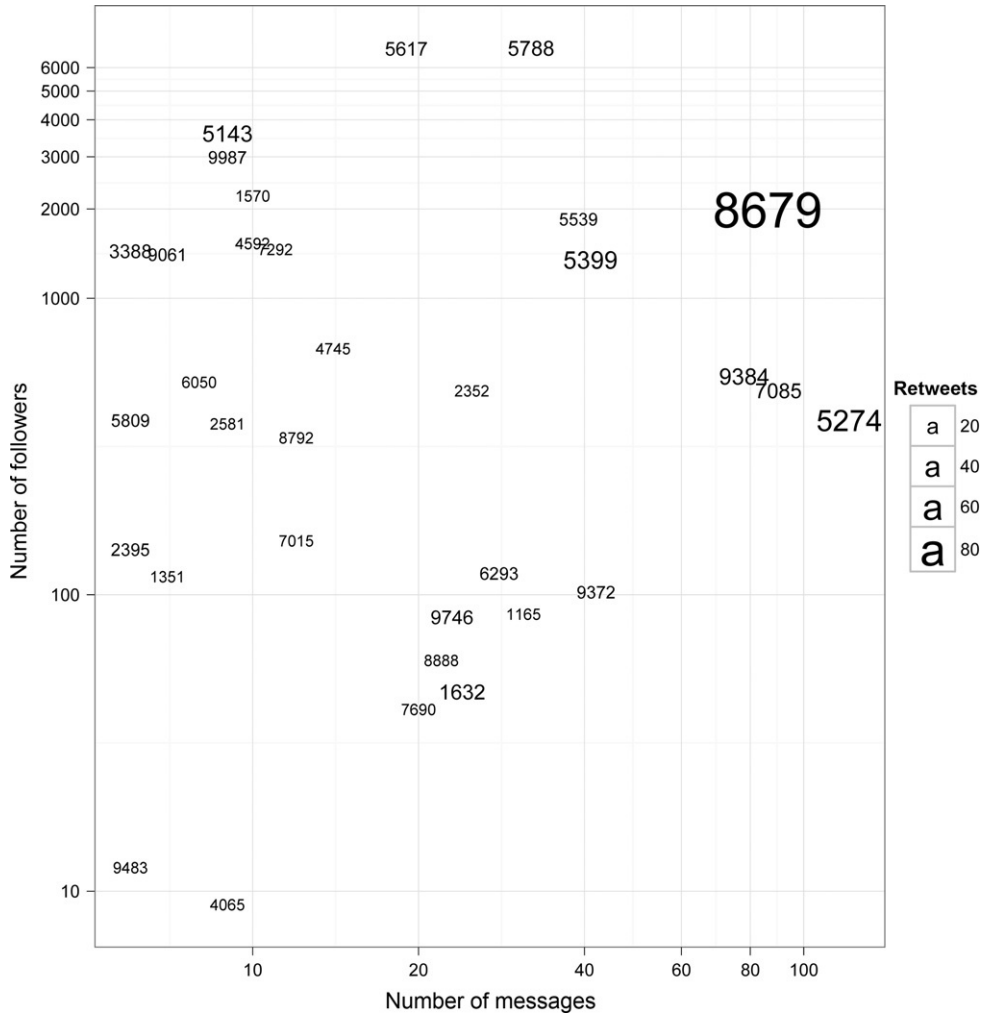


Figure 3.2
Plot of each author's number of followers on the Twitter network by the number of their messages in the corpus. The size of the author's identifying number indicates the frequency that their messages were retweeted.

```
users <- sapply(users, as.character)
# make a data frame for further manipulation
users.df <- data.frame(users = users, followers = "", stringsAsFactors = FALSE)
# loop to populate users$followers with a follower count obtained from Twitter API
for (i in 1:nrow(users.df))
{
   # tell the loop to skip a user if their account is protected
   # or some other error occurs
   result <- try(getUser(users.df$users[i])$followersCount,
silent = FALSE);
```

```
   if(class(result) == "try-error") next;
   # get the number of followers for each user
   users.df$followers[i] <-
getUser(users.df$users[i])$followersCount
   # tell the loop to pause for 60 s between iterations to
   # avoid exceeding the Twitter API request limit
   # note that this is going to take a long
   # time if there are a lot of users the sample!
   print('Sleeping for 60 seconds...')
   Sys.sleep(60);
}
# merge follower count with number of tweets per author
followerCounts <- merge(TweetRetweet, users.df, by.x = "screenName", by.y = "users")
# convert value to numeric for further analysis
followerCounts$followers <- as.numeric(followerCounts$followers)
followerCounts$counts <- as.numeric(followerCounts$counts)

# create a plot of number of followers by number of messages and number of retweets
ggplot(data = followerCounts, aes(count, followers)) +
   geom_text(aes(label = randuser, size = RT_count)) +
   scale_size(range=c(3,10)) +
   scale_x_log10(breaks = c(10,20,40,60,80,100)) +
   scale_y_log10(breaks = c(10,100,seq(1000,7000,1000))) +
   xlab("Number of Messages") +
   ylab("Number of Followers") +
   theme_bw() +
   theme(axis.title.x = element_text (vjust = -0.5, size = 14)) +
   theme(axis.title.y = element_text (size = 14, angle=90)) +
   theme(plot.margin = unit(c(1,1,2,2), "lines"))
```

Further insights into message propagation via retweets can be obtained from the ratio of retweets to original messages produced by each author (Figure 3.3). Authors with a retweet ratio greater than one had a higher number of their messages being retweeted by others than original messages, indicating a degree of influence and popularity that might not be predicted from the total number of messages they authored. It is remarkable to note that only one of the authors with a high retweet ratio is also among the most prolific (author 8679).

```
# Make table with counts of tweets per person
t <- as.data.frame(table(rand.df$randuser))
# make table with counts of retweets per person
rt <- as.data.frame(table(rand.df$rt.rand))
# combine tweet count and retweet count per person
t.rt <- merge(t,rt, by = "Var1")
# creates new col and adds ratio tweet/retweet
t.rt["ratio"] <- t.rt$Freq.y / t.rt$Freq.x
# sort it to put names in order by ratio
sort.t.rt <- t.rt[order(t.rt$ratio), ]
# exclude those with 2 tweets or less
sort.t.rt.subset <- subset(sort.t.rt, sort.t.rt$Freq.x > 2)
#
```

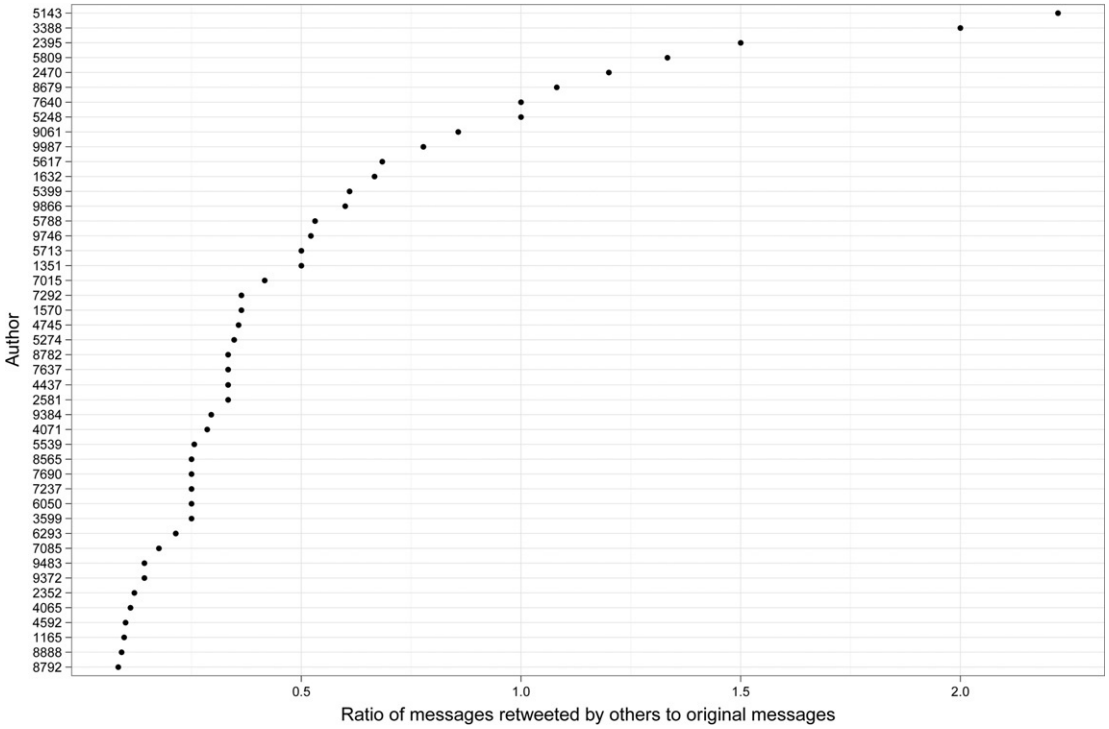


Figure 3.3
Ratio of retweeted messages to total messages by each author.

```
# drop unused levels leftover from subsetting
sort.t.rt.subset.drop <- droplevels(sort.t.rt.subset)
# plot nicely ordered counts of tweets by person for
# people > 5 tweets
ggplot(sort.t.rt.subset.drop, aes(reorder(Var1, ratio), ratio)) +
   xlab("Author") +
   ylab("Ratio of messages retweeted by others to original messages")+
   geom_point() +
   coord_flip() +
   theme_bw() +
   theme(axis.title.x = element_text(vjust = -0.5, size = 14)) +
   theme(axis.title.y = element_text(size = 14, angle=90)) +
   theme(plot.margin = unit(c(1,1,2,2), "lines"))
```

3.5 What Is the Community Structure of These Twitter-Users?

The degree of connectedness, and other related network properties of this community can be further explored by computing descriptive indices of the network graph resulting from the relationships contained in the retweet data (Butts, 2008a; Ye and Wu, 2010).

Table 3.2 Summary of Graph-Level Social Network Indices

Index	Value	Range of CUG Test Distribution	Interpretation
Density	0.012	0.492-0.508	Significantly fewer connections between community members than expected
Reciprocity	1.000	0.487-0.510	Significantly higher tendency of ties to be reciprocal rather than unidirectional
Transitivity	0.059	0.493-0.507	Significantly less instances of "a friend of a friend is a friend" than expected
Centralization	0.222	0.044-0.107	Significantly more centralized than expected

These indices provide succinct numerical summaries of the structure of the community that produced these messages. For each of these indices, a conditional uniform graph test can be undertaken to compare the observed index values against those which would be obtained by simulated data with known substantive properties similar to the data. The extent and direction of the deviation of the indices from their baseline distributions can create structural biases within the network, which may provide useful clues regarding the organization of the community (Butts, 2008b). Table 3.2 summarizes these graph indices and indicates that the community has distinctive structural properties, such as significantly fewer connections between individuals than expected, significantly higher tendency of reciprocal rather than unidirectional ties, significantly fewer triadic relationships than expected, and a significantly higher degree of centralization than expected. Some of these properties are also apparent in the network graph (Figure 3.4) which shows the network graph with a distinctive pattern of highly interconnected individuals in the center and many individuals who connect with only one highly connected individual.

```
# extract tweeter-retweeted pairs
rt <- data.frame(user=rand.df$randuser, rt=rand.df$rt.rand)
# omit pairs with NA and get only unique pairs
rt.u <- na.omit(unique(rt)) #
# begin social network analysis plotting
require(igraph)
require (sna)
degree <- sna::degree
g <- graph.data.frame(rt.u, directed = F)
g <- as.undirected(g)
g.adj <- get.adjacency(g)
# plot network graph
```

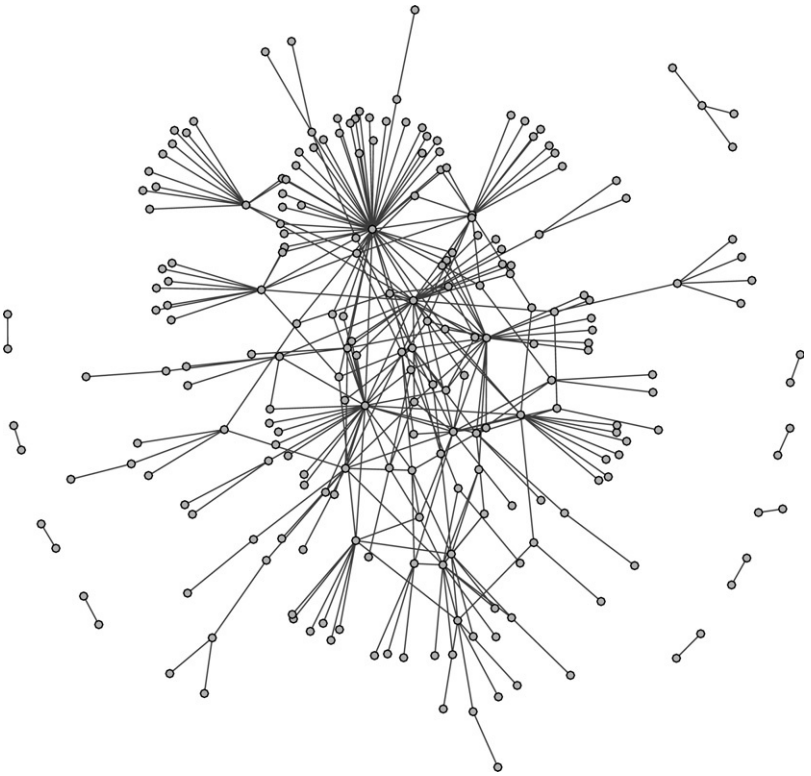


Figure 3.4
Visualization of the community of authors based on their retweeting behaviors.

```
gplot(g.adj, usearrows = FALSE,
        vertex.col = "grey",vertex.border = "black",
        displaylabels = FALSE, edge.lwd = 0.01, edge.col
         = "grey30", vertex.cex = 0.5)
# get some basic network attributes
gden(g.adj) # density
grecip(g.adj) # reciprocity
gtrans(g.adj) # transitivity
centralization(g.adj, degree)
# calculate Univariate Conditional Uniform Graph Tests
# density
print(cug.gden <- cug.test(g.adj, gden))
plot(cug.gden)
range(cug.gden$rep.stat)
# reciprocity
print(cug.recip <- cug.test(g.adj, grecip))
plot(cug.recip)
range(cug.recip$rep.stat)
# transistivity
print(cug.gtrans <- cug.test(g.adj, gtrans))
plot(cug.gtrans)
```

```
range(cug.gtrans$rep.stat)
# centralisation
print(cug.cent <- cug.test(g.adj, centralization, FUN.arg=list(FUN=degree)))
plot(cug.cent)
range(cug.cent$rep.stat)

# find out how many communities exist in the network using the walktrap
g.wc <- walktrap.community(g, steps = 1000, modularity=TRUE, labels=TRUE)
plot(as.dendrogram(g.wc, use.modularity=TRUE))
max(g.wc$membership)+1
```

An inspection of the corpus of messages indicates that many of the highly connected individuals were broadcasting snippets of information from presentations they were attending. An example of a frequently retweeted message is "Would take two hours of moderate exercise daily to bring industrialized activity budget in line with subsistence [sic] activity budget," referring to a presentation in the "Scars of Evolution" session. These snippets were being retweeted by others who do not appear to have been at the same presentation but wanted to acknowledge their interest in the presentation and circulate the details through their network of Twitter contacts. Other types of frequently retweeted messages include links to weblog posts and news articles that comment on issues of the meeting (e.g., "What role should science play in #anthropology? http://t.co/4KylJaE1," referring to Jaschik (2011)) and observations and announcements about meeting events (e.g., "Undergrad student poster session! Come by and view the wonderful research by our future academics!"). Twenty-five cohesive groups of message writers and retweeters were identified in this sample using the walktrap community structure detection algorithm (Pons and Latapy, 2005).

3.6 What Were Twitter-Users Writing About During the Meeting?

Now I turn to some basic and widely used text mining techniques (Feinerer et al., 2008) to identify the issues that captured the attention of Twitter-using anthropologists during the meeting. To prepare for this analysis, I converted all text in the corpus to lower case, removed punctuation, numbers, and stopwords (words that occur very frequently due to their importance in sentence construction, for example, *is*, *and*, *the*) and stemmed words (removing morphological affixes such as -s, -ed, -ing, leaving only the stem of the word so that there is a single token that indicates different forms of the same word that have a common meanings). The result is that each document is a string of tokens, where a token is a sequence of characters that are grouped together as a useful semantic unit (but are not always immediately recognizable as words) for further processing. A document term matrix was then created where each column represents a token and each row represents a document. From here I identified the most frequently occurring tokens in the entire corpus and repeated the stopword removal process to remove context specific but relatively uninformative high-frequency tokens such as *aaa*, *session*, *panel*.

```
require(tm)
a <- Corpus(VectorSource(df$text)) # create corpus object
a <- tm_map(a, tolower) # convert all text to lower case
a <- tm_map(a, removePunctuation)
a <- tm_map(a, removeNumbers)
a <- tm_map(a, removeWords, stopwords("english")) # this list needs to be edited and this
function repeated a few times to remove high frequency context specific words with no semantic
value
require(rJava) # needed for stemming function
library(Snowball) # also needed for stemming function
a <- tm_map(a, stemDocument, language = "english") # converts terms to tokens

a.dtm <- TermDocumentMatrix(a, control = list(minWordLength = 3)) # create a term document
matrix, keeping only tokens longer than three characters, since shorter tokens are very hard to
interpret
inspect(a.dtm[1:10,1:10]) # have a quick look at the term document matrix
findFreqTerms(a.dtm, lowfreq=30) # have a look at common words, in this case, those that appear
at least 30 times, good to get high freq words and add to stopword list and re-make the dtm, in
this case add aaa, panel, session
findAssocs(a.dtm, 'science', 0.30) # find associated words and strength of the common words. I
repeated this function for the ten most frequent words.
```

Term frequency and association analyses are simple but widely used methods in text mining because the results are relatively simple to calculate and interpret (Namey et al., 2007: 141; Ryan and Bernard, 2000: 776). Table 3.3 shows the 25 most frequently occurring tokens in the corpus. Table 3.4 shows the 10 tokens most strongly correlated with the 10 most frequently occurring tokens. Once these tokens were identified, close reading of a sample of the full text was undertaken to investigate their meaning and context. The 10 most frequently occurring tokens reflect four topics of the meeting that Twitter-using anthropologists were responding to. The most frequent token, *scar*, is contained in messages referring to the session "The scars of human evolution" in which author 8679 was a presenter. The majority of messages containing this token are either messages by this author or other authors retweeting his messages. Author 5399 also used this token in frequent messages referring to this session and was similarly

Table 3.3 High-Frequency Tokens in the Corpus

Frequency	Tokens
100	scar
60	scar, scienc[e]
50	scar, scienc[e], digita[l]
40	scar, scienc[e], digita[l], people[e]
30	scar, scienc[e], digita[l], people[e], activit[y], evoluti[ion], male, publishin[g]
20	scar, scienc[e], digita[l], people[e], activit[y], evoluti[ion], male, publishin[g], birt[h], bra[in], bud[get], domingue[z], ethic[s], foo[d], future, industrialize, occup[y], primate, ris[k], rol[e], sout[h], theor[y], virgini[a], writin[g]

The characters in square brackets show the terms that the tokens most frequently derive from.

Table 3.4 Token Associations in the Corpus for the 10 Most Frequently Occurring Tokens (Limited to the 10 Tokens with the Strongest Associations with Correlations >0.2)

Token	Associated Tokens (Strength of Correlation)
Scar	doctor (0.30), milfor (0.30), pond (0.30), wolp (0.30), birt (0.28), bra (0.26), lif (0.26), expenditure (0.24), compare (0.23), evoluti (0.23)
Scienc	humanis (0.63), scientific (0.63), rol (0.46), debat (0.45), nuance (0.33), educati (0.25), see (0.25), plac (0.25)
Digita	space (0.29), morp (0.27), archive (0.25), audi (0.25), collaboration (0.25), exhibition (0.25), includ (0.25), layere (0.25), repatriati (0.25), tur (0.25)
People	decad (0.37), pri (0.37), reproductiv (0.37), surviv (0.37), jer (0.31), wakin (0.29), archiva (0.24), destructiv (0.24), prett (0.24), sometime (0.24)
Activit	bud (0.95), exercis (0.91), subistenc (0.91), moderat (0.89), brin (0.87), dail (0.87), lin (0.78), industrialize (0.64), ironi (0.23), stretc (0.23)
Evoluti	anato (0.35), childbirth (0.35), litte (0.35), thrill (0.35), detail (0.31), stre (0.31), disabl (0.30), persona (0.28), treva (0.28), wend (0.28)
Male	subsistenc (0.80), weig (0.80), expenditure (0.64), energ (0.60), industrialize (0.51), female (0.42), competit (0.35), aggressivenes (0.34), favore (0.32), level (0.29)
publishin	futur (0.43), academi (0.40), sav (0.40), gree (0.35), brandin (0.30), speculation (0.30), vita (0.30), opportunitie (0.26), curatoria (0.24), pushin (0.24)
Birt	decad (0.59), pri (0.59), reproductive (0.58), surviv (0.58), amaz (0.50), suppor (0.49), measure (0.40), pas (0.37), los (0.36), weigh (0.36)
Bra	compare (0.67), rat (0.67), restin (0.67), mammal (0.64), metaboli (0.59), neonat (0.56), primte (0.56), siz (0.53), human (0.51), adul (0.49)

See the text for reconstruction of the terms from these tokens.

retweeted. Associated with *scar* is the name of one of the discussants of the session, Milford Wolpoff, whose name mostly occurs in the context of messages noting his reference to the television series Dr. Who, indicating the mixture of scholarly and informal messages relating to this session. The terms *birth*, *brain*, *life*, etc., can be reconstructed from the tokens in Table 3.4 and come from highlights of scholarly content in the presentations, mostly in messages by author 8679. Among the other top 10 high-frequency tokens, *peopl*, *activit*, *evoluti*, *male*, *birt*, and *bra* (most frequently *brain*, though also resulting from the unrelated term *brand*) also relate to this session. The dominance of this session in the Twitter content appears to reflect the experience of a small number of people who participated in the session and the followers of these people who rebroadcast snippets of detail from the presentation most likely for the benefit of others who were not attending the session.

The second most frequent token is *scienc*. This token is more evenly distributed across the authors and, as can been seen from the associated tokens in Table 3.4, relates to the debate about whether anthropology is more of a humanistic or scientific discipline. Messages containing this token fall into two categories. First are direct observations on the session "Science in

Anthropology: An Open Discussion," which was organized in response to controversy surrounding the removal of the word *science* from the AAA's long-range plan statement in 2010. An interesting contrast between the importance of this issue among the community of Twitter-using anthropologists and the wider group of meeting attendants is revealed by this message: "#AAAsci is dominating #AAA2011 conversation, yet the room is less full than anticipated. 516 CD. Looks like there's much discussion ahead." This indicates that the science issue had been frequently mentioned by Twitter-users, but the low attendance at the session suggested to that author that it was not a high priority for the majority of participants.

The second category of messages, discussing science, contains links to articles in *The Chronicle of Higher Education* (Berrett, 2011) and *Inside Higher Education* (Jaschik, 2011). The link to the *Inside Higher Education* story on the science debate was the most frequently shared link in the corpus and indicates the importance of this issue to Twitter-using anthropologists (Figure 3.5). In this sample, 276 messages contained links, i.e., 18% of the sample, which is a substantially lower proportion than similar datasets (Weller et al., 2011). The cited articles were

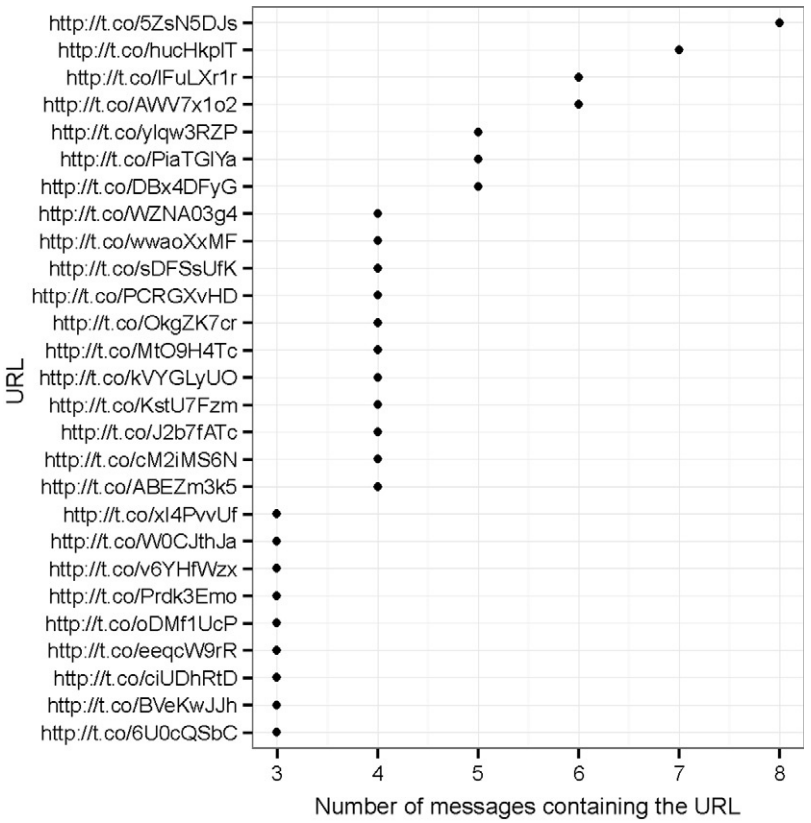


Figure 3.5
Frequency of URLs in the corpus.

published online while the meeting was in session and give history to the controversy as well as reporting on the 2011 "Science in Anthropology" session. There is no evaluation of the news articles in the messages, only broadcasting of the headline and a link to the online article. This behavior has previously been observed as a common use of Twitter during academic conferences (Weller et al., 2011). One interpretation of this behavior is that authors are trying to work around the 140-character limit of Twitter messages by linking to long-form writing that contains more complex and nuanced discussions.

```
# investigate the URLs contained in the Twitter messages
require(stringr)
require(ggplot2)
df$link <- sapply(df$text,function(tweet) str_extract(tweet,
("http[[:print:]]+")))  # creates new field and extracts the links contained in the tweet
df$link <- sapply(df$text,function(tweet)
str_extract(tweet,"http[[:print:]]{16}")) # limits to just 16 characters after http so I just
get the shortened link.
countlink <- data.frame(URL = as.character(unlist(dimnames(sort(table(df$link))))),
N = sort(table(df$link))) # get frequencies of each link and put in rank order
rownames(countlink) <- NULL # remove rownames
countlinkSub <- subset(countlink, N>2) # subset of just links appearing more than twice
# plot to see distribution of links
ggplot(countlinkSub, aes(reorder(URL, N), N)) +
  xlab("URL") +
  ylab("Number of messages containing the URL")+
  geom_point() +
  coord_flip() +
  theme_bw() +
  theme(axis.title.x = element_text(vjust = -0.5, size = 14)) +
  theme(axis.title.y = element_text(size = 14, angle=90)) +
  theme(plot.margin = unit(c(1,1,2,2), "lines"))
```

The third most frequent token, *digita*, refers to two sessions, mostly "Digital Anthropology: Projects and Projections" and to a lesser degree, "Coming of Age in the Digital Age: Youth Media Practices and Gendered Identities." Similar to the Scars session, many of the messages containing the *digita* token originate from a single author (5274), who was also a presenter in the "Digital Anthropology" session, and relate to the scholarly content of that session. A focus on digital topics is to be expected from a community of authors who exist because of their use of digital media such as Twitter.

The final theme that can be readily identified from these data relates to the token *publishin*. The associated tokens show that the authors were concerned with the future of academic publishing. Most of these tokens relate to two messages originally by author 5274 as comments on the "Digital Anthropology" session (in which this person was a presenter) that were frequently retweeted. In this prominence of the digital and scars session in the Twitter corpus, we see how the interests of a small number of authors have dominated the corpus. The high frequency of posts about these sessions, and about the science session, reflect a small but engaged

community whose interests and experiences are not necessarily reflective of others involved in the meeting. For example, although the Society for Medical Anthropology is one of the largest AAA sections, content from its sessions are not prominent in the Twitter corpus.

3.7 What Do the Twitter Messages Reveal About the Opinions of Their Authors?

The token frequency and association data give some simple insights into the issues that dominate the messages emanating from the meetings. But they are not very effective at revealing whether they had a positive or negative opinion about the issues they write about. Some insights into this may be obtained using sentiment analysis, the computational study of the opinions that people have about entities and events (Thelwall et al., 2011). The number of occurrences of positive and negative words in each document was counted to determine the document's sentiment score. To calculate the document sentiment score, each positive word counts as +1 and each negative word as −1. Although the method I used has the advantage of being simple to use, it does not handle polysemy, for example, irony and sarcasm (inspection of the corpus indicated that these forms of expression were very rare). I used a list of 6789 positive and negative words created by Hu and Liu (2004) to calculate scores for all documents in the corpus (the list is available online at http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html).

```
# This is based on Jeffrey Breen's excellent tutorial at http://jeffreybreen.wordpress.com/
2011/07/04/twitter-text-mining-r-slides/
# download sentiment word list from here: http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-
English.rar un-rar and put somewhere logical on your computer
hu.liu.pos = scan('C:/...somewhere on your computer.../opinion-lexicon-English/positive-
words.txt', what = 'character',comment.char=';') #load +ve sentiment word list
hu.liu.neg = scan('C:/...somewhere on your computer.../opinion-lexicon-English/negative-
words.txt',what = 'character',comment.char= ';') #load -ve sentiment word list
pos.words = c(hu.liu.pos)
neg.words = c(hu.liu.neg)
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
    require(plyr)
    require(stringr)
# we got a vector of sentences. plyr will handle a list
# or a vector as an "l" for us
# we want a simple array ("a") of scores back, so we use
# "l" + "a" + "ply" = "laply":
   scores = laply(sentences, function(sentence, pos.words, neg.words) {
# clean up sentences with R's regex-driven global substitute,gsub():
        sentence = gsub('[[:punct:]]', '', sentence)
        sentence = gsub('[[:cntrl:]]', '', sentence)
        sentence = gsub('\\d+', '', sentence)
        # and convert to lower case:
        sentence = tolower(sentence)
        # split into words. str_split is in the stringr package
```

```
        word.list = str_split(sentence, '\\s+')
        # sometimes a list() is one level of hierarchy too much
        words = unlist(word.list)

        # compare our words to the dictionaries of positive & negative terms
        pos.matches = match(words, pos.words)
        neg.matches = match(words, neg.words)

        # match() returns the position of the matched term or NA
        # we just want a TRUE/FALSE:
        pos.matches = !is.na(pos.matches)
        neg.matches = !is.na(neg.matches)

        # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
        score = sum(pos.matches) - sum(neg.matches)

        return(score)
    }, pos.words, neg.words, .progress=.progress )

    scores.df = data.frame(score=scores, text=sentences)
    return(scores.df)
}
aaa.text <- laply(aaa2011, function(t)t$getText()) # draw on the original object of tweets
that we first got to extract just the text of the tweets
length(aaa.text) #check how many tweets, make sure it agrees with the original sample size
head(aaa.text, 5) #check content sample, see that it looks as expected, no weird
characters, etc.
aaa.scores <-
score.sentiment(aaa.text,pos.words,neg.words,.progress='text')
# get scores for the tweet text
# create a histogram of sentiment scores
ggplot(aaa.scores, aes(x=score)) +
   geom_histogram(binwidth=1) +
   xlab("Sentiment score") +
   ylab("Frequency") +
   theme_bw() +
   opts(axis.title.x = theme_text(vjust = -0.5, size = 14)) +
   opts(axis.title.y=theme_text(size = 14, angle=90, vjust = -0.25)) +
   opts(plot.margin = unit(c(1,1,2,2), "lines"))
aaa.pos <- subset(aaa.scores,aaa.scores$score >= 2) # get tweets with only very +ve scores
aaa.neg <- subset(aaa.scores,aaa.scores$score <= -2) # get tweets with only very -ve scores

# Now create subset based on tweets with certain words, such as the high frequency words
identified in the text mining. eg. science
scien <- subset(aaa.scores, regexpr("scien", aaa.scores$text) > 0) # extract tweets
containing only 'scien'
# plot histogram for this token,
ggplot(scien, aes(x = score)) + geom_histogram(binwidth = 1) +
   xlab("Sentiment score for the token 'scien'") +
   ylab("Frequency") + theme_bw() +
   theme(axis.title.x = element_text(vjust = -0.5, size = 14)) +
   theme(axis.title.y = element_text(size = 14, angle = 90, vjust = -0.25)) +
   theme(plot.margin = unit(c(1,1,2,2), "lines"))
# repeat this block with different high frequency words
```

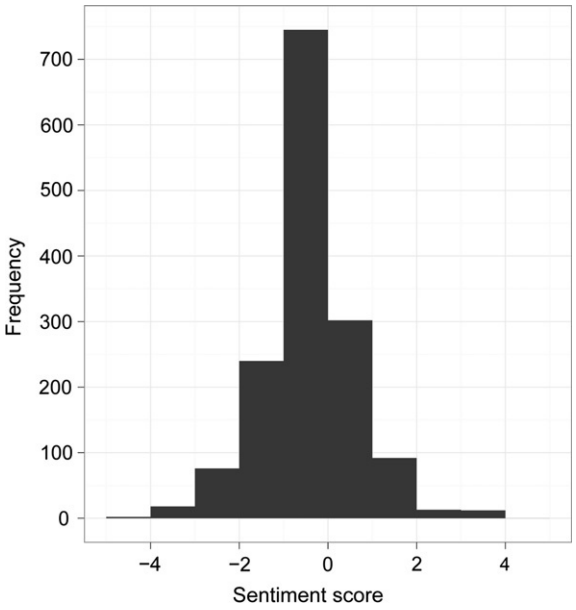


Figure 3.6
Histogram of sentiment scores for all documents.

The modal sentiment over the entire corpus is neutral to slightly negative (Figure 3.6). A small number of very positive scores (>2) counter the negative mode, resulting in a mean sentiment score of 0.08. The range of scores in this sample (−4 to 4) is smaller than a larger sample of general Twitter messages (−6 to 7, Breen, 2011). Taking the subset of documents ($n = 65$) that contain the token *scien*, a similar slightly negative mode is evident in Figure 3.7 but there are no highly positive scores. This results in a significantly more negative sentiment about the science issue than overall sentiment about the meeting ($t = 2.53$, df = 126.26, $p = 0.01$). This is consistent with the weblog and news article commentaries produced during and shortly after the meeting report that meeting participants were frustrated with the discussion of the science issue (Antrosio, 2011; Berrett, 2011; Jaschik, 2011; Lende, 2011; Marks, 2011; Van Arsdale, 2011).

Sentiment surrounding the token *digita* (54 messages) was more positive, with a mean score of 0.37, a relatively high minimum score of −2 and a positive skew in the distribution of scores (no significant difference to overall sentiment: $t = -1.46$, df = 35.34, $p = 0.15$) (Figure 3.8). These data convey the enthusiasm that some of the authors have for digital technologies in anthropology, which is evident in a sample of the full text, for example, "amazing opportunities for authors and readers," "great panel this morning," and "great papers." On a related issue, documents containing the token *publishin* had a mean score of exactly 0.0 and a range from −2 to 1 indicating a mix of

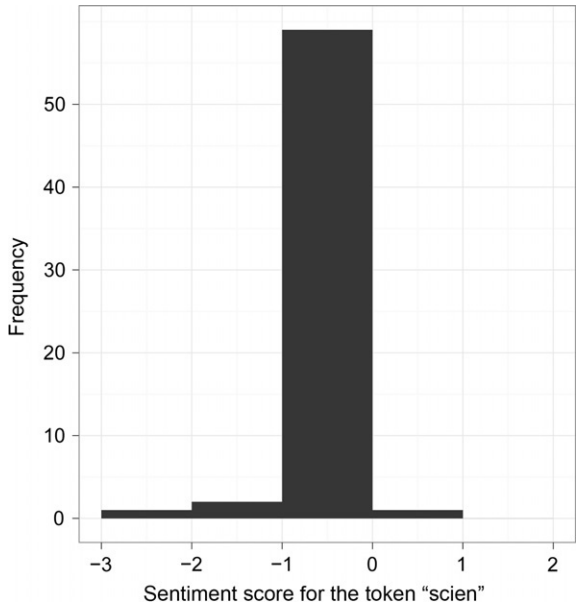


Figure 3.7
Histogram of sentiment scores for documents containing the token *scien*.

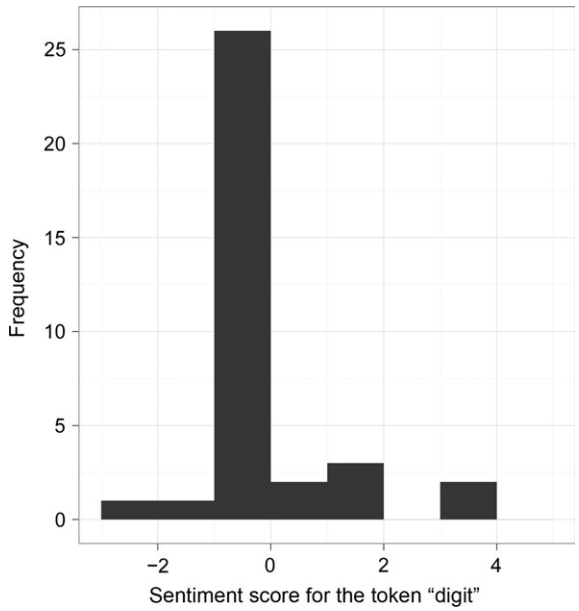


Figure 3.8
Histogram of sentiment scores for documents containing the token *digita*.

positive and negative sentiment about the future of academic publishing. Sentiment scores were not calculated for documents containing the token *scar* because inspection of the full text indicated that they contain semantic terms such as *lower* and *fail* that relate to the scholarly content of the session rather than the opinion of the author and so would have given exaggerated negative sentiment scores. Direct inspection of the corpus reveals no obviously negative messages and three messages that are explicitly positive, noting the high attendance at the session, praising the humor of the presenters and creativity of the paper titles.

3.8 What Can Be Discovered in the Less Frequently Used Words in the Sample?

Although token frequency and association analyses are simple and revealing, their focus on the highest frequencies and strongest associations means they are not sensitive to less common patterns in the text. To investigate these rarer patterns, I used hierarchical clustering methods to produce a visualization of the distances between tokens in the corpus. This method takes the document term matrix and calculates distances between all of the tokens based on their frequencies in the documents and then classifies the tokens into nested groups (Suzuki and Shimodaira, 2006) (Figure 3.9). This method is useful because it reveals correlations between rarer tokens that do not appear in the frequency and association analysis, giving additional insights into what captured the attention of Twitter-using anthropologists.

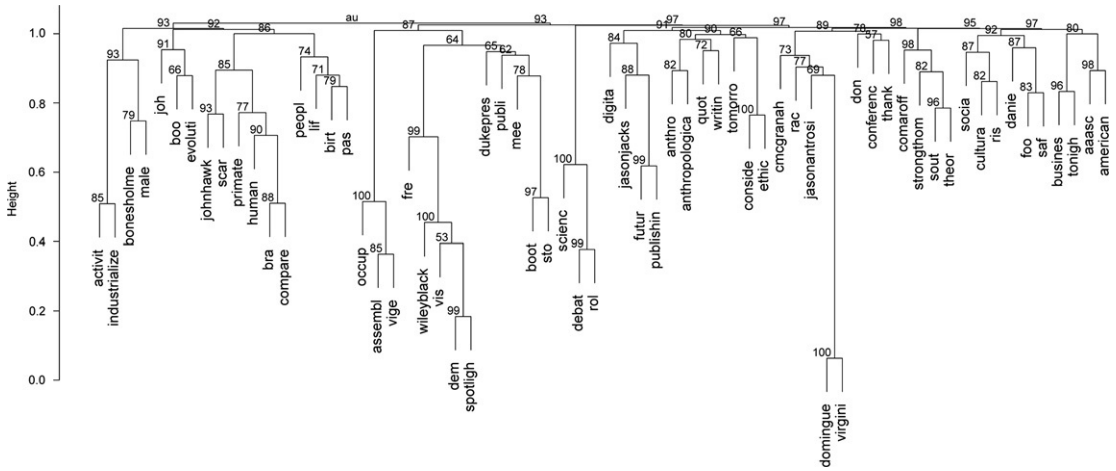


Figure 3.9

Cluster dendrogram of all documents with AU (approximately unbiased) *p*-values. For each cluster in the dendrogram, *p*-values between 0 and 1 were calculated by multiscale nonparametric bootstrap resampling (in this case, 5000 resamples). Clusters that are highly supported by the data have *p*-values closer to 1.

```
a.dtm.sp <- removeSparseTerms(a.dtm, sparse=0.989) # I found I had to iterate over this to
ensure the dtm doesn't get too small... for example: 0.990 nrow=88, 0.989, nrow=67, 0.985,
nrow=37, 0.98 nrow=23, 0.95 nrow=6
a.dtm.sp.df <- as.data.frame(inspect(a.dtm.sp)) # convert document term matrix to data frame
nrow(a.dtm.sp.df) # check to see how many words we're left with after removing sparse terms
# this analysis is based on http://www.statmethods.net/advstats/cluster.html
# scale and transpose data for cluster analysis
a.dtm.sp.df.sc.t <- t(scale(a.dtm.sp.df))
require(pvclust)
fit <- pvclust(a.dtm.sp.df.sc.t, method.hclust = "average", method.dist = "correlation",
nboot = 10000) # this method may take a few hours the bootstraping, you can reduce the nboot value
for a quicker result
plot(fit, cex = 1.5, cex.pv = 1.2, col.pv = c(1,0,0), main="", xlab="", sub="") # draw the
dendrogram
```

Support for claims based on the token frequency and association data can been seen in the large left-most cluster that includes 17 tokens relating to the scars session. The debate about the role of science is clearly evident in a tight cluster near the center containing *scienc*, *debat*, and *rol*. The issue of digital media and the future of academic publishing are captured by a cluster just to the right of the science debate cluster.

In addition to this verification of the token frequency and association analysis, several further insights into the issues contained in the corpus may be derived from the cluster analysis. The cluster of *occup*, *asssembl*, and *vige* (Viger Hall, a location at the meeting) derives from messages encouraging people to participate in a general assembly in support of the Occupy protest movement. The clusters containing *wileyblack* and *dukepres* are readily identifiable as deriving from the stream of advertisements from these publishers.

The cluster containing the names Carole McGranahan, Jason Antrosio, and Virginia Dominguez (the AAA president at the time of the meeting) refers to messages discussing the AAA Presidential Address. The focus of many of these messages is Dominguez's discussion of the 2010 final report of the Commission on Race and Racism in Anthropology, as indicated by the token *rac* in this cluster. Links to the PDF file of the report were also circulated in five messages, making it the third most frequently shared link in the corpus. Inspection of the full text reveals generally positive sentiment about the Presidential Address, for example, "an address worth thinking about" and "great presidential address." Moving further to the right of the dendrogram, a cluster including *sout*, *theor*, *comarof* derives from messages commenting on the session "Authors Meet Critics: Reading Jean and John Comaroff's 'Theory From The South: Or, How Euro-America is Evolving Toward Africa." The scholarly content of this session was reported in almost one hundred messages by a single author, whose messages were widely retweeted. The cluster containing *foo*, *saf*, and *danie* refers to 46 messages discussing papers presented in the 12 sessions (and evening reception) sponsored by the Society for the Anthropology of Food and Nutrition (identified by the hashtag #SAFN, from which the *saf* token derives). Inspection of the full text reveals that the token *danie* refers to Daniel Reichman's paper in the session "Ethnographic Approaches to Food Activism: Agency,

Democracy, and Economy" which contained the observation of a "new trend toward consumers" desire for absolute empirical knowledge about the provenance of their food', which was retweeted by a number of authors.

3.9 What Are the Topics That Can Be Algorithmically Discovered in This Sample?

My final method for discovering what was important to Twitter-using anthropologists during the meeting is topic modeling. This method allows for a more complex subject analysis than the text mining and token distance techniques employed earlier (Newman and Block, 2011). Topic models are generative models that aim to discover the hidden thematic structures in large numbers of text documents. A topic is defined as a probability distribution over all words in the corpus that captures the salient themes that run through the corpus (Blei et al., 2010; Steyvers and Griffiths, 2007). Each document in the corpus is represented as a probability distribution over some of the topics. Topic modeling aims to infer the set of topics that were responsible for generating a collection of documents. The difference between topic modeling and text mining methods is that while text mining methods assume that each token is distinctive to a topic, topic models are mixed-membership models, meaning that each word or token may simultaneously belong to several topics, each document may contain several topics, and the distributions of these topics will vary over the documents in the corpus (Grün and Hornik, 2011). The unique contribution of this method is that it can identify a topic characterized by key words that may never appear next to each other in the same document.

```
require(slam)
a.dtm.sp.t <- t(a.dtm.sp) # transpose document term matrix, necessary for the next steps using mean term frequency-inverse document frequency (tf-idf) to select the vocabulary for topic modeling
summary(col_sums(a.dtm.sp.t)) # check median...
term_tfidf <- tapply(a.dtm.sp.t$v/row_sums(a.dtm.sp.t)[a.dtm.sp.t$i], a.dtm.sp.t$j, mean) * log2(nDocs(a.dtm.sp.t)/col_sums(a.dtm.sp.t>0)) # calculate tf-idf values
summary(term_tfidf) # check median... note value for next line...
a.dtm.sp.t.tdif <- a.dtm.sp.t[,term_tfidf>=1.0] # keep only those terms that are slightly less frequent that the median
a.dtm.sp.t.tdif <- a.dtm.sp.t[row_sums(a.dtm.sp.t) > 0, ]

summary(col_sums(a.dtm.sp.t.tdif)) # have a look
# Before going right into generating the topic model and analysing the output, we need to decide on the number of topics that the model should use
# Here's a function to loop over different topic numbers, get the log likelihood of the model for each topic number and plot it so we can pick the best one
# The best number of topics is the one with the highest log likelihood value.

require(topicmodels)
best.model <- lapply(seq(2, 50, by = 1), function(d){LDA(a.dtm.sp.t.tdif, d)}) # this will make a topic model for every number of topics between 2 and 50... it will take some time!
```

```
best.model.logLik <- as.data.frame(as.matrix(lapply(best.model, logLik))) # this will
produce a list of logLiks for each model

# plot the distribution of log likelihoods by topic
best.model.logLik.df <- data.frame(topics=c(2:50), LL = as.numeric(as
.matrix(best.model.logLik)))
ggplot(best.model.logLik.df, aes(x = topics, y = LL)) +
   xlab("Number of topics") +
   ylab("Log likelihood of the model") +
   geom_line() +
   theme_bw() +
   theme(axis.title.x = element_text(vjust = -0.5, size = 14)) +
   theme(axis.title.y = element_text(size = 14, angle=90, vjust= -0.25)) +
   theme(plot.margin = unit(c(1,1,2,2), "lines")) # plot nicely
ggsave(file = "model_LL_per_topic_number.pdf") # export the plot to a PDF file
# it's not easy to see exactly which topic number has the highest LL, so let's look at the data...
best.model.logLik.df.sort <- best.model.logLik.df[order(-best.model.logLik.df$LL), ] #
sort to find out which number of topics has the highest loglik, in this case 23 topics.
best.model.logLik.df.sort # have a look to see what's at the top of the list, the one with the
highest score
```

Topic modeling identifies subject categories without *a priori* subject definitions (Newman and Block, 2011). Instead, fast algorithms for computing with hierarchical mixture models find the underlying patterns of words that are embedded in the corpus. Latent Dirichlet allocation (LDA) has been shown to be a highly effective unsupervised probabilistic method for finding distinct topics in Twitter messages (e.g., Ramage et al., 2010; Zhao et al., 2011) and a variety of other collections of documents (e.g., Blei et al., 2003; Hall et al., 2008). In brief, specifying the LDA model consists of three steps: (1) draw K topics from a symmetric Dirichlet distribution, (2) for each document d, draw topic proportions from a symmetric Dirichlet distribution, and (3) for each word n in each document d, (3a) draw a topic assignment from the topic proportions and (3b) draw the word from a multinomial probability distribution conditioned on the topic (Grün and Hornik, 2011). I generated LDA models that decomposed the corpus into its salient topics, and determined the specific distributions over the tokens for each topic and distributions of topics over each document (cf. Blei et al., 2010). To fit the LDA model to the document-term matrix, the number of topics needs to be decided in advance. To identify the optimum number of topics for this corpus, I calculated the log-likelihood of the data for all models with between 2 and 50 topics. The model with the highest log-likelihood value indicates the number of topics that are the best fit for the data (Griffiths and Steyvers, 2004), in this case 23 topics (Figure 3.10).

```
lda <- LDA(a.dtm.sp.t.tdif,23) # generate a LDA model with 23 topics, as found to be optimum
get_terms(lda, 5) # get keywords for each topic, just for a quick look
get_topics(lda, 5) # gets topic numbers per document
lda_topics<-get_topics(lda, 5)
beta <- lda@beta # create object containing parameters of the word distribution for each topic
gamma <- lda@gamma # create object containing posterior topic distribution for each document
terms <- lda@terms # create object containing terms (words) that can be used to line up with beta
and gamma
```

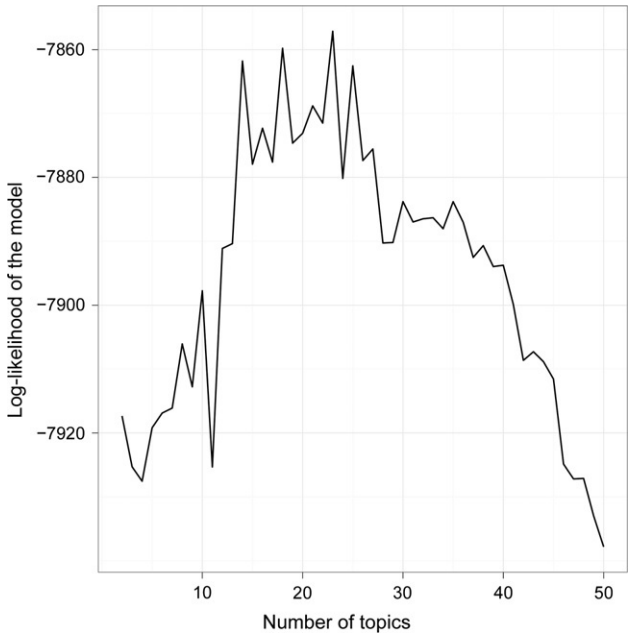


Figure 3.10
LDA model selection results showing the log-likelihood of the data for different numbers of topics.

```
colnames(beta) <- terms # puts the terms (or words) as the column names for the topic weights.
id <- t(apply(beta, 1, order)) # order the beta values
beta_ranked <- lapply(1:nrow(id),function(i)beta[i,id[i,]]) # gives table of words per
topic with words ranked in order of beta values. Useful for determining the most important words
per topic
```

Table 3.5 shows the top-ranked five tokens associated with each of the 23 topics. The topics automatically identified by the LDA model provide excellent verification of the issues identified by the token frequency and association analysis. Both methods identified the prominence of topics relating to the scars session, the "Theory from the South" session and the sessions on food, publishing, and Digital Anthropology. Other issues emerging from the topic model data include racism in anthropology, the role of science in anthropology, and changes to the AAA's code of ethics.

3.10 Conclusion

In summary, I have obtained a large number of short text messages written by participants of the 109th AAA meeting and used three methods of quantitative content analysis to discover the topical issues and controversies of the meeting according to the authors of these messages. I have also obtained some insights into the structure, rules, and practices of this community of authors. All three content analysis methods provide consistent results on the prominent topics, issues, and controversies of the meeting. Key issues for this community can be grouped

Table 3.5 Topics and Their Five Top-Ranked Tokens Produced by the LDA Model. Topic names were manually assigned based on the top-ranked tokens.

1. [code of ethics]	2. [scars session]	3. [scars session]	4. mixed	5. mixed
Conside	scar	compare	publishin	conferenc
Ethic	cultura	bra	lif	don
Jasonjacks	johnhawk	primate	scar	peopl
Quot	quot	human	johnhawk	tomorro
Dukepres	bonesholme	johnhawk	comaroff	dukepres
6. [future of publishing]	**7. [SAFN sessions]**	**8. [scars session]**	**9. mixed**	**10. [race]**
Jasonjacks	foo	writin	peopl	virgini
Future	socia	ris	dukepres	domingue
Thank	saf	danie	scar	rac
Publishin	cultura	scar	johnhawk	cmcgranah
jasonantrosi	publi	publi	jasonantrosi	jasonantrosi
11. [scars session]	**12. [scars session]**	**13. [scars session]**	**14. [role of science]**	**15. [scars session]**
Scar	male	evoluti	scienc	activit
Birt	johnhawk	bonesholme	rol	industrialize
Pas	scar	johnhawk	debat	quot
Johnhawk	bonesholme	scar	jasonantrosi	johnhawk
jasonantrosi	jasonantrosi	joh	dukepres	scar
16. [adverts]	**17. [adverts]**	**18. [social]**	**19. [Digital Anthropology]**	**20. [Occupy Montreal]**
Wileyblack	boot	quot	digita	occup
Vis	fre	boo	jasonantrosi	assembl
Dem	sto	mee	anthro	vige
Spotlight	publi	tomorro	quot	scar
Fre	wileyblack	anthro	joh	johnhawk

21. [theory from south session]	22. mixed	23. mixed
Theor	tonigh	johnhawk
Sout	anthropologica	american
strongthom	busines	aaasc
Comaroff	joh	scar
Quot	scar	quot

Tokens are ordered by the logarithmized parameters of the token distribution for each topic. I assigned the column labels in square brackets manually after inspecting the full set of topic-tokens (i.e., these column labels are not output from the model).

into scholarly concerns and concerns relating to policies specific to the AAA. Prominent scholarly concerns relate to papers presented in the scars session, the "Theory from the South" session, the Digital Anthropology session, the anthropology of food sessions, and the future of publishing forum. The concerns specific to the AAA corpus are the debate about the role of science in anthropology, racism in the discipline, and concern about revising the organization's code of ethics. Many of these issues were also represented in long-form weblog posts by anthropologists attending the meeting and the mainstream media, indicating a correlation

between issues of interest to Twitter-using anthropologists, weblog authors (most of whom are also highly active on Twitter), and the media. The most prominent controversy in the Twitter corpus, as measured by the sentiment analysis, was the role of science in anthropology. These messages were directed mainly to author peers participating in the conference, but there was limited dissent among authors, as indicated by the overall neutral and narrow range of sentiment scores. These observations are consistent with previous studies of the use of Twitter at academic meetings (Ebner, 2009; Ebner and Reinhardt, 2009; Ebner et al., 2010; Letierce et al., 2010a,b; Reinhardt et al., 2009; Ross et al., 2011).

Key attributes of the content of the corpus are the high proportion of retweeted messages and the circulation of links, indicating that sharing information and reporting news were common uses of Twitter by meeting participants. This distinctive content suggests that Twitter messages may have value for informing nonparticipants on the hot issues among Twitter-using anthropologists, contrary to previous work that found Twitter messages uninformative for nonparticipants (Ebner et al., 2010). Future research using interviews is needed to investigate the relationship between Twitter-using anthropologists and nonanthropologists. Institutional support for the use of Twitter by the AAA, such as a publically viewable projection of messages in a common space of the meeting venue, would likely stimulate more intensive use by attendees. This would result in a more complete record of the meeting in the Twitter corpus that would perhaps more credibly represent the diversity of the event to nonparticipants.

The structure of the community in this study is distinctive, with its demography biased toward more junior scholars and roughly equal representation of male and female authors. The relationship between gender and impact among Twitter-users (e.g., the number of followers and retweets) is an important issue for future investigation. A wide range of identity-signaling practices are employed with about half of the community using pseudonyms. The community has a small number of very highly interconnected individuals, and the majority of individuals are only connected to a small number of these highly connected individuals. One interpretation of this community structure is that Twitter-using anthropologists are comprised of many weakly connected groups composed of individuals sharing similar interests. For example, in several instances, we see one prolific individual broadcasting messages about the contents of a session and a group of dozen or so other individuals retweeting those messages. Among the different sessions where this occurred, few individuals appear to have been members of more than one group of retweeters.

This distinctive community structure is one of the most important emergent properties of the use of Twitter at the AAA meeting. The immediate nature of Twitter messages, compared to weblogs and other media, means that groups of individuals can rapidly and loosely self-assemble around specific events, such as conference presentations and specific people who are influential at these events. Similar phenomena have been described in the use of Twitter in political contexts (Holotescu et al., 2011). This is the transformative and emergent effect of Twitter in academia, to easily enable the spontaneous formation of information-sharing

communities bound by an interest in an event or topic. Twitter enables the kind of cross-cutting connectivity between groups of individuals that 19th-century sociologist Émile Durkheim (1893/1993) claimed was central to modern solidarity (Gruzd et al., 2011). The long-term stability of the membership and structure of these connections and communities formed by Twitter-users are important issues for future investigation.

A logical future extension of the methods presented here is for the analysis of longer texts such as weblog posts and journal articles. Furthermore, a corpus representing a longer period of time would also give insights into long-term community change and change in key issues and controversies. Although there are some pioneering examples of this kind of work (Blei and Lafferty, 2007; Griffiths and Steyvers, 2004; Hall et al., 2008; Mimno, 2012; Newman and Block, 2006, 2011), it remains for future work to take advantage of the reproducibility and accessibility that are key strengths of using R to make these methods more widely applicable.

References

Antrosio, J., 2011. Science in Anthropology: humanistic science and scientific humanism. Living Anthropologically Blog post 17-Nov-11. http://www.livinganthropologically.com/2011/11/17/science-in-anthropology/ (accessed 17.11.11).

Bentley, R., Hahn, M., Shennan, S., 2004. Random drift and culture change. Proc. R. Soc. B Biol. Sci. 271 (1547), 1443–1450.

Berrett, D., 2011. Anthropologists seek a more nuanced place for science. The chronicle of higher education. http://chronicle.com/article/Anthropologists-Seek-a-More/129823/ (accessed 17.11.11).

Blei, D.M., Lafferty, J.D., 2007. A correlated topic model of science. Ann. Appl. Stat. 1, 17–35.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Blei, D., Carin, L., Dunson, D., 2010. Probabilistic topic models. Signal Process. Mag. 27 (6), 55–65.

Boellstorff, T., 2011. Three comments on anthropology and science. Am. Anthropologist. 113 (4), 541–544.

Boyd, D., Golder, S., Lotan, G., 2010. Tweet, tweet, retweet: conversational aspects of retweeting on twitter. In: Proceedings of the 43rd Hawaii International Conference on System Sciences, 5-8 January, Koloa, Kauai, HI, USA, 2010. Institute of Electrical and Electronics Engineers, pp. 1–10.

Breen, J., 2011. Sentiment analysis—a popular use of text mining in airlines' CRM. In: Miner, G., Elder, J., Hill, T., Nisbet, R., Delen, D. (Eds.), Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications. Academic Press, New York, pp. 57–68.

Butts, C.T., 2008a. Social network analysis with sna. J. Stat. Software. 24 (6), 1–51.

Butts, C.T., 2008b. Social network analysis: a methodological introduction. Asian J. Soc. Psychol. 11 (1), 13–41.

Durkheim, É., 1893/1993. The Division of Labor in Society. Macmillan, New York, NY.

Ebner, M., 2009. Introducing live microblogging: how single presentations can be enhanced by the mass. J. Res. Innovative Teach. 2 (1), 91–100.

Ebner, M., Reinhardt, W., 2009. Social networking in scientific conferences—twitter as tool for strengthen a scientific community. Proceedings of the 1st European Conference on Technology Enhanced Learning, October 1-4, Nizza, Crete, Greece, 2009, vol. 2. Springer, pp. 1–8.

Ebner, M., et al., 2010. Getting Granular on twitter: tweets from a conference and their limited usefulness for non-participants. In: International Federation for Information Processing, World Computer Congress, Key Competencies in the Knowledge Society Conference, September 20-23, Brisbane, Australia, 2010. Springer, pp. 102–113.

Egri, C.P., 1992. Academic conferences as ceremonials: opportunities for organizational integration and socialization. J. Manag. Educ. 16 (1), 90–115.

Feinerer, I., Hornik, K., Meyer, D., 2008. Text mining infrastructure in R. J. Stat. Software. 25 (5), 1–54.
Gentry, J., 2011. twitteR: R based Twitter client. R package version 0.99.15. http://cran.r-project.org/web/packages/twitteR/ (accessed 01.10.11).
Griffiths, T.L., Steyvers, M., 2004. Finding scientific topics. Proc. Natl. Acad. Sci. U.S.A. 101 (Suppl. 1), 5228–5235.
Grün, B., Hornik, K., 2011. topicmodels: an R package for fitting topic models. J. Stat. Software. 40 (13), 1–30.
Gruzd, A., Wellman, B., Takhteyev, Y., 2011. Imagining twitter as an imagined community. Am. Behav. Sci. 55 (10), 1294–1318.
Hall, D., Jurafsky, D., Manning, C.D., 2008. Studying the history of ideas using topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 25–27 October Honolulu, Hawaii, USA, 2008. Association for Computational Linguistics, pp. 363–371.
Holotescu, C., et al., 2011. Microbolgging Meets Politics. The Influence of Communication in 140 Characters on Romanian Presidential Elections in 2009. Rom. J. Commun. Public Relations 13 (1), 37–50.
Hu, M., Liu, B., 2004. Mining and summarizing customer reviews. In: Proceedings of the ACM SIGKDD (Association for Computing Machinery Special Interest Group for Knowledge Discovery and Data Mining) International Conference on Knowledge Discovery & Data Mining, 22-25 August, Seattle, USA, 2004. Association for Computing Machinery, pp. 168–177.
Jaschik, S., 2011. Not Feeling the Kinship. Inside Higher Education. http://www.insidehighered.com/news/2011/11/18/anthropologists-debate-role-science (accessed 18.11.11).
Kwak, H., et al., 2010. What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web, 26-30 April. Association for Computing Machinery, Raleigh, North Carolina, USA, pp. 591–600.
Lende, D., 2011. The Montreal Anthropology Meetings—Recap of AAA Coverage. Neuroanthropology Blog post 22-Nov-11. http://blogs.plos.org/neuroanthropology/2011/11/22/the-montreal-anthropology-meetings-recap-of-aaa-coverage/ (accessed 22.11.11).
Letierce, J., et al., 2010a. Using Twitter during an Academic Conference: The iswc2009 Use-case. In: Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media, 23-26 May, Washington, DC, USA, 2010a. Association for the Advancement of Artificial Intelligence, pp. 279–282.
Letierce, J., et al., 2010b. Understanding how Twitter is used to spread scientific messages. In: Proceedings of the Web Science Conference 2010: Extending the Frontiers of Society On-Line, 19th International World Wide Web Conference, 26-30 April, Raleigh, NC, USA, 2010b, pp. 1–8.
Marks, J., 2011. So, est-ce la science? Anthropomics Blog post 25-Nov-11. http://anthropomics.blogspot.com/2011/11/so-est-ce-la-science.html.
McCarthy, J.F., Boyd, D., 2005. Digital backchannels in shared physical spaces: experiences at an academic conference. In: Computer-Human Interaction 2005, Conference on Human Factors in Computing Systems, 5-7 April, Portland, Oregon, USA, 2005. Association for Computing Machinery, pp. 1641–1644.
Mimno, D., 2012. Computational historiography: data mining in a century of classics journals. J. Comput. Cult. Heritage. 5 (1), 1–19.
Namey, E., et al., 2007. Data reduction techniques for large qualitative data sets. In: Guest, G., MacQueen, K. (Eds.), Handbook for Team-based Qualitative Research. AltaMira Press, Lanham, MD, pp. 137–162.
Newman, D.J., Block, S., 2006. Probabilistic topic decomposition of an eighteenth-century American newspaper. J. Am. Soc. Inform. Sci. Technol. 57 (6), 753–767
Newman, D., Block, S., 2011. What, where, when, and sometimes why: data mining two decades of women's history abstracts. J. Women's Hist. 23 (1), 81–109
Pons, P., Latapy, M., 2005. Computing communities in large networks using random walks. Comput. Inform. Sci. ISCIS. 2005, 284–293.
Prensky, M., 2001. Digital natives, digital immigrants Part 1. On the horizon 9 (5), 1–6.
Priem, J., Costello, K., Dzuba, T., 2011. Prevalence and use of Twitter among scholars. In: Metrics 2011: Symposium on Informetric and Scientometric Research., LA, USA, New Orleans, LA, USA, 2011.

Ramage, D., Dumais, S., Liebling, D., 2010. Characterizing microblogs with topic models. In: Proceedings of the Fourth International Association for the Advancement of Artificial Intelligence Conference on Weblogs and Social Media, 23-26 May Washington, DC, USA, 2010. The Association for the Advancement of Artificial Intelligence Press, pp. 1–10.

Reinhardt, W., et al., 2009. How people are using Twitter during conferences. In: Hornung-Prahauser, V., Luckmann, M. (Eds.), Kreativität und Innovationskompetenz im digitalen Netz: wie kommt das "Neue" mit Hilfe von Internettechnologien in die Welt?. Salzburg Research Forschungsgesellschaft, Salzburg, pp. 145–155.

Ross, C., et al., 2011. Enabled backchannel: conference Twitter use by digital humanists. J. Doc. 67 (2), 214–237

Ryan, G., Bernard, H.R., 2000. Data management and analysis methods. In: Denison, N., Lincoln, Y. (Eds.), Handbook of Qualitative Research. second ed. Sage Publications, Thousand Oaks, CA, pp. 769–802.

Steyvers, M., Griffiths, T., 2007. Probabilistic topic models. In: Landauer, T.K., McNamara, D.S., Dennis, S., Kintsch, W. (Eds.), Handbook of Latent Semantic Analysis. Psychology Press, London, pp. 424–440.

Suzuki, R., Shimodaira, H., 2006. Pvclust: an R package for assessing the uncertainty in hierarchical clustering. Bioinformatics 22 (12), 1540–1542.

Thelwall, M., Buckley, K., Paltoglou, G., 2011. Sentiment in Twitter events. J. Am. Soc. Inform. Sci. Technol. 62 (2), 406–441

Van Arsdale, A., 2011. Science and the Ring Species of Anthropology. A.P. Van Arsdale Biological Anthropology Lab Blog post 21-Nov-11. http://blogs.wellesley.edu/vanarsdale/2011/11/21/anthropology/science-and-the-ring-species-of-anthropology/ (accessed 21.11.11).

Weller, K., Dröge, E., Puschmann, C., 2011. Citation analysis in Twitter. In: Approaches for Defining and Measuring Information Flows within Tweets during Scientific Conferences. Making Sense of Microposts (#MSM2011), Workshop at the Extended Semantic Web Conference 2011, 30 May, Crete, Greece, 2011. CEUR-WS.org, Tilburg University.

Wilson, S.M., Peterson, L.C., 2002. The anthropology of online communities. Ann. Rev. Anthropology 31, 449–467.

Ye, S., Wu, S., 2010. Measuring Message propagation and social influence on Twitter.com social informatics. Soc. Inform. Lect. Notes Comput. Sci. 6430, 216–231.

Zhao, W., et al., 2011. Comparing Twitter and traditional media using topic models. Adv. Inform. Retrieval. 6611, 338–349.

CHAPTER 4

Text Mining and Network Analysis of Digital Libraries in R

Eric Nguyen
FinancialMood Analytical Services, Inc. Montreal, Quebec, Canada

4.1 Introduction

With the availability of open and structured data, digital libraries become an important source of data in recent data mining techniques. The inherent structure of digital libraries comes with information about date, authorship, involved institutions, geographic context, and large volumes of text.

As an illustration of digital libraries analysis, we will focus on a space of scientific publications related to some science discipline and extract patterns and text-related information from the dataset. Text mining techniques offer a reliable and efficient way to quantify many aspects of digital libraries, as well as methods to cluster and regroup digital content in relation to topical content. This chapter will discuss about text preparation techniques in R, the use of the latent Dirichlet allocation (LDA) to classify content, and the analysis of text features, like co-occurrence matrices. Simple similarity measures between authors and between papers will be used to illustrate cluster cohesion within the dataset.

On top of text analysis, scientific collaboration is often mirrored in terms of citations and references, which will allow us to build a network of interactions. Various centrality measures of impact and influence of scientific collaborators and participants are then derived from the network. This chapter will make extensive use of the `tm`, `sna`, and the `lda` R packages. Theoretical aspects of the various algorithms will also be discussed, as well as R code snippets and relevant charts.

The following outline describes the different steps of our analysis:

1. Dataset preparation:
   Using the `tm` R package, we parse the text content of our dataset. This involves using various functions to get rid of undesirable characters and terms such as punctuation characters, whitespaces, and stop words.

Data Mining Applications with R

2. Exploring the document-term matrix:
   The document-term matrix is a useful object, which describes the relationships between documents and terms. The `tm` R package provides methods to deal with term sparsity and to find associations between terms. A discussion about the term frequency-inverse document frequency measure and how it can be used in R is also part of this section.
3. Topical analysis and content clustering using the LDA:
   The LDA is a model where documents are viewed as a mixture of topics. One can use the `lda` R package to learn about a topic representation from the documents of a dataset. This section covers how to run the learning algorithm, validate the model using log-likelihood values, describe the different topics resulting from the LDA model, and how to visually paint the topic representation of each document.
4. Document cohesion using a similarity measure:
   From the LDA model, each document has a mixture of topics associated with it. These attributes can be used to derive a concept of similarity, where similar documents have similar topic distribution. In this section, we show how one can compute a simple cosine similarity measure on the documents of the dataset, and use a heatmap to illustrate the different document clusters.
5. Social Network Analysis of authors:
   As a final analysis, we build a network of authors based on coauthorship interactions between the different participants. The `igraph` R package will allow us to construct a graph representing our network, and, using the `sna` package, we apply a measure of centrality to the nodes of our graph to describe which authors are more important (in terms of collaboration). This analysis can be extended to other type of networks.

4.2 Dataset Preparation

ArXiv.org is an archive of electronic preprints of scientific publications provided by Cornell University. The available topics include mathematics, physics, computer science, statistics, quantitative finance, and biology. It was launched in 1991 and includes more than half a million articles.

The first dataset to be explored is the set of all abstracts of papers submitted during the year 2011 in the mathematics subject. The data are included in the 2011-papers.Rdata file and was generated externally from the arXiv.org website, using the `curl` command line tool and a few `perl` scripts.

A corpus is the set of all documents and can be thought of as a database of text files. To load the abstracts into a corpus in R, get to the directory and enter:

```
> setwd("Data")
> library(tm)
> s <- Corpus(DirSource("abstracts"))
```

To start our text analysis of all the abstracts, we want to remove undesirable characters, such as whitespaces, punctuations, and treat all words in their lowercase form. The `tm` library includes a few text processing functions which can be applied to our corpus (Feinerer, 2011).

```
> s <- tm_map(s, tolower)
> s <- tm_map(s, removePunctuation)
> s <- tm_map(s, removeWords, stopwords("english"))
> s <- tm_map(s, stripWhitespace)
```

All functions `tolower`, `removePunctuation`, `removeWords`, and `stripWhitespace` are applied to each document in our corpus. We use the `removeWords` function to remove various stop words, which we consider noisy in our data. The list provided by the `tm` library includes words such as *the*, *is*, *on*, *at*. The last function `stripWhitespace` will collapse repeated whitespaces resulting from the previous text manipulations.

```
> summary(s)
A corpus with 1870 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

> s[[1]]
noncommutative solitons quasideterminants masashi hamanaka discuss extension soliton theory
integrable systems noncommutative spaces focusing integrable aspects noncommutative
antiselfdual yangmills equations wide class exact solutions solving riemannhilbert
[...]
```

The `tm` package contains many more transformation functions, such as `removeCitation`, `removeMultipart`, `removeNumbers`, and more. The reader is redirected to the `tm` user manual for more information about these other functions.

4.3 Manipulating the Document-Term Matrix

4.3.1 The Document-Term Matrix

Now that we have collected all our abstracts, we can build the document-term matrix associated with the collection of abstracts. The document-term matrix is simply a matrix describing the frequencies of all terms occurring in the collection of text documents.

```
> dtm <- DocumentTermMatrix(s)
```

Note that the `tm` package also has the `TermDocumentMatrix` class, which allows the user to work in a transposed way where terms are represented as rows, as opposed to columns as in the case of `DocumentTermMatrix`. In the present example, we will use the

DocumentTermMatrix class. All functions related to the DocumentTermMatrix class can also be applied to the TermDocumentMatrix class.

Using the DocumentTermMatrix, each row corresponds to a document in the corpus and each column corresponds to a term. For example, one can get a partial view of the matrix with:

```
> inspect(dtm[1:5, 1:5])
A document-term matrix (5 documents, 5 terms)

Non-/sparse entries: 0/25
Sparsity           : 100%
Maximal term length : 10
Weighting          : term frequency (tf)

                Terms
Docs             00879873 012 0134 013indexed 01d
1101.0005.txt           0   0    0          0   0
1101.0006.txt           0   0    0          0   0
1101.0012.txt           0   0    0          0   0
1101.0013.txt           0   0    0          0   0
1101.0015.txt           0   0    0          0   0
```

The entries of the matrix are simply the frequencies associated with the terms in the corresponding document.

```
> dtm
A document-term matrix (1870 documents, 20176 terms)

Non-/sparse entries: 80702/37648418
Sparsity           : 100%
Maximal term length : 327
Weighting          : term frequency (tf)
```

When printing out the dtm object, a few statistics are displayed. First, the number of sparse and nonsparse entries is displayed. A value of 0 in the matrix is considered a sparse entry and a nonzero value is a nonsparse entry. In our case, we have 1870 documents and 20,176 terms, yielding 37,729,120 entries in total. Among these entries, 80,702 entries are nonzero values, and the rest of the matrix, the other 37,648,148 entries, is zero. Sparsity is the proportion of sparse entries in the entire matrix. The maximal term length is the number of terms in the longest document in our corpus. The default weighting method when instantiating a DocumentTermMatrix is the term-frequency weighting. A detailed discussion on this parameter is given below.

There are a few options one can use with the DocumentTermMatrix. The first setting is the lower and upper bounds of global term frequencies to be considered. Terms appearing in the collection of documents less often than the specified lower bound will be ignored, and similarly with terms appearing more frequently than the upper bound.

```
> dtm.2 <- DocumentTermMatrix(s, control=list(bounds = list(global = c(2,Inf))))
> dtm.2
A document-term matrix (1870 documents, 6404 terms)
```

```
Non-/sparse entries: 66930/11908550
Sparsity           : 99%
Maximal term length : 46
Weighting          : term frequency (tf)
```

In the example above, terms which appears only once in the whole corpus will be ignored when building the document-term matrix. Note that the number of terms in our matrix has dropped because we specified frequency bounds.

For a conventional set of text documents, the size of `DocumentTermMatrix` objects tends to grow at an undesirable rate. A high proportion of the entries are sparse, hence one can significantly reduce the size of the matrix by removing sparse terms without significantly affecting the overall analysis.

```
> removeSparseTerms(dtm, 0.99)
A document-term matrix (1870 documents, 803 terms)

Non-/sparse entries: 40560/1461050
Sparsity           : 97%
Maximal term length : 16
Weighting          : term frequency (tf)
```

The `removeSparseTerms` function returns another `DocumentTermMatrix`, duplicated from the one supplied in the argument, from which sparse terms have been removed. In our example, terms which are 80% sparse were removed from the `DocumentTermMatrix` object. In other words, terms which do not appear in at least 80% of all documents are removed from the original `DocumentTermMatrix` object.

Another interesting feature of the `DocumentTermMatrix` class in R is the availability of using an arbitrary frequency measure other than the standard frequency count. The following section covers that particular aspect about the `DocumentTermMatrix` class.

4.3.2 Term Frequency-Inverse Document Frequency

Given a corpus D, a term t_i and a document $d_j \in D$, we denote the number of occurrences of t_i in d_j by tf_{ij}. This is referred as the term frequency.

The inverse document frequency for a term t_i is defined as

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

where $|D|$ is the number of documents in our corpus, and $|\{d : t_i \in d\}|$ is the number of documents in which the term appears. If the term t_i appears in every document of the corpus, idf_i is equal to 0. The fewer documents the term t_i appears in, the higher the idf_i value.

The measure called term frequency-inverse document frequency (*tf-idf*) is defined as tf_{ij}*idf_i (Salton and McGill, 1986). It is a measure of importance of a term t_i in a given document d_j. It is a

term frequency measure which gives a larger weight to terms which are less common in the corpus. The importance of very frequent terms will then be lowered, which could be a desirable feature.

There are many variants of the *tf-idf* measure, and some variants will give better results depending on the given dataset (Salton and Buckley, 1988). For example, a variant of the *tf-idf* measure where the term frequency factor scales sub-linearly, like when $tf'_i = \log(1 + tf_i)$, will deflate the weight given to frequent terms. This could be useful when term frequencies follow a power law with respect to the rank. The tf-idf function provided with the `tm` package is the standard one defined earlier.

To construct a document-term matrix such that the entries are tf-idf values, one would use:

```
> dtm.tfidf <- DocumentTermMatrix(s, control=list(weighting=weightTfIdf))
> dtm.tfidf
A document-term matrix (1870 documents, 20176 terms)

Non-/sparse entries: 80702/37648418
Sparsity           : 100%
Maximal term length : 327
Weighting          : term frequency - inverse document frequency
 (normalized) (tf-idf)
> inspect(removeSparseTerms(dtm.tfidf, 0.9)[1:5, 1:5])
A document-term matrix (5 documents, 5 terms)

Non-/sparse entries: 3/22
Sparsity           : 88%
Maximal term length : 9
Weighting          : term frequency - inverse document frequency
 (normalized) (tf-idf)
              Terms
Docs           consider   finite   function   functions   model
1101.0005.txt        0        0 0.00000000 0.00000000       0
1101.0006.txt        0        0 0.07768253 0.00000000       0
1101.0012.txt        0        0 0.11652380 0.00000000       0
1101.0013.txt        0        0 0.00000000 0.00000000       0
1101.0015.txt        0        0 0.00000000 0.06001211       0
```

Note that by using tf-idf as a frequency measure, we are now dealing with real values instead of integral entries.

The `weighting` parameter of the `control` option of the `DocumentTermMatrix` constructor adds flexibility by allowing arbitrary term frequency measures, as long as the provided function can parse a corpus object.

4.3.3 Exploring the Document-Term Matrix

Once the document-term matrix has been decided on, one can list terms with a frequency higher than a threshold by typing:

```
> findFreqTerms(dtm, 400)
[1]  "finite"  "paper"  "prove"  "results"  "space"  "theory"
```

Here, the words with frequency higher than 400, across the whole collection of documents, are shown.

Given a term, one can also find out which words are highly correlated with that term by using the `findAssocs` function. This function also requires a correlation limit as an argument.

```
> findAssocs(dtm.2, "graph", 0.3)[1:3]
graphs    edge  vertices
  0.49    0.39      0.37
> findAssocs(dtm.2, "edge", 0.3)
   vertex  adjacent    graph    graphs    chordal    cut
   0.41      0.39      0.39     0.33       0.31    0.31
located  maintains    picks    walker
0.31       0.31      0.31     0.31
```

In this case, the terms "*graphs*," "*edge*," and "*vertices*" are correlated with the term "*graphs*." By lowering the correlation limit, more terms appear from the output of `findAssocs`.

4.4 Clustering Content by Topics Using the LDA

4.4.1 The Latent Dirichlet Allocation

The LDA is a technique developed by David Blei, Andrew Ng, and Michael Jordan and exposed in Blei et al. (2003). The LDA is a generative model, but in text mining, it introduces a way to attach topical content to text documents. Each document is viewed as a mix of multiple distinct topics. An advantage of the LDA technique is that one does not have to know in advance what the topics will look like. By tuning the LDA parameters to fit different dataset shapes, one can explore topic formation and resulting document clusters.

The mathematics behind the LDA is beyond the scope of this work, but one should be aware of the following aspects of the algorithm. The number of topics K is fixed and specified in advanced. The corpus contains documents $d_i = \left(w_{i,1}, ..., w_{i,n_i}\right)$ of length n_i. Each word $w_{i,j}$ comes from a vocabulary which consists of V different terms.

The term distribution for each topic is modeled by

$$\beta_i \sim \text{Dirichlet}(\eta)$$

where Dirichlet(η) denotes the Dirichlet distribution for parameter η.

The proportion of topic distribution for each document is distributed as

$$\omega_i \sim \text{Dirichlet}(\alpha)$$

Each word $w_{i,j}$ is associated to a topic $z_{i,j}$ which follows

$$z_{i,j} \sim \text{Multinomial}(\omega_i)$$

where Multinomial(ω_i) denotes the multinomial distribution with one trial (Grun and Hornik, 2011).

In this setup, the LDA is a "bag of words" model, where the order in which the words appear does not affect the grammar.

4.4.2 Learning the Various Distributions for LDA

Given a set of documents, one can use the LDA framework to learn the different distributions describing a model about the topic representation of the documents in the corpus. Each word is then associated with a topic, and each topic has a term distribution that helps make sense of it. One way to turn this into a learning algorithm is by using the collapsed Gibbs sampling method.

Before the first iteration, the algorithm starts by assigning a random topic to each word in each document. Then, at each iteration, the algorithm goes through each word $w_{i,j}$ in document d_i. For each topic T_k, it computes $p(T_k|d_i)$, the observed portion of words assigned to topic T_k in document d_i, and $p(w_{i,j}|T_k)$, the portion of assignment to topic T_k that comes from the word $w_{i,j}$. The algorithm then resamples a new topic $T'_{i,j}$ for $w_{i,j}$ with probability $p(T_k|d_i)*p(w_{i,j}|T_k)$ before it jumps to the next word.

An iteration is completed when all words in all documents are revisited. After a large number of iterations, the model tends to converge to a steady state of topic assignment. We will use the LDA model with R to model topics for our corpus from arXiv.org.

For the following section of the chapter, the `lda` package is required.

```
> library(lda)
```

The R implementation of LDA requires the input to be suitably formatted. The `lexicalize` function will help us address this issue.

```
[...]
> lex <- lexicalize(s)
> head(lex$vocab)
 [1] "noncommutative"    "solitons"    "quasideterminants"
 [4] "masashi"           "hamanaka"    "discuss"
> lex$documents[[1]]
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]    0    1    2    3    4    5    6    7    8
[2,]    1    1    1    1    1    1    1    1    1
[...]
```

Here, the variable `s` is the clean version of our corpus, produced earlier in the current work. The list `lex$vocab` is the vocabulary associated with the corpus. The first row of `lex $documents[[1]]` is the list of term indices, from `lex$vocab`, and the second row of `lex $documents[[1]]` is the term frequency of each term of the first document of our corpus.

We are now ready to run the LDA on the resulting `lex` object:

```
> res <- lda.collapsed.gibbs.sampler(lex$documents, 10, lex$vocab, 100, 0.1, 0.1, compute.
log.likelihood=T)
```

In our example, the `lda.collapsed.gibbs.sampler` function takes the following arguments:

- an object resulting from `lexicalize`, in our case the `lex$documents` object
- the number of topics in the model
- the vocabulary associated with the corpus, `lex$vocab`
- a number of iterations for the Gibbs sampling
- the α parameter describing the term distribution for each topic
- the η parameter describing the topic distribution for each document
- a flag indicating whether the log-likelihood should be computed and returned. The log-likelihood is useful when one wants to determine convergence of the LDA process. Details about this return value are discussed below.

The resulting object, `res` in our case, contains a few important attributes:

- `res$assignments`: a list of vectors representing the topic association of each term in each document of our corpus. Each entry of `res$assignments` corresponds to a document.
- `res$document_sums`: a matrix representing the number of times the terms in each document were associated with each of the topics. Rows represent topics and columns are documents.
- `res$log.likelihoods`: a matrix with two rows. The first row lists the full log-likelihood values for each iteration, and the second row lists the log-likelihood values of the observations conditioned on the arguments for each iteration.

```
> res$assignments[[1]]

[1]  5  7  5  5  7  5  5  7  5  7  7  5  5  7  5  5  5  5  5  7  7  7  7  7  7  5  5 [...]
[39]  5  5  5  5  5  7  5  5  5  5  5  7  5  5  7  7  5  5  7  7  7
> res$document_sums[,1:10]
     [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]   [,10]
[1,]    0      0      0      0      0      6      0      0      0       0
[2,]    0      8      0      0      0      1      0      0      0       0
[3,]    0      8      0      4      0      0      0      0      0      27
[4,]    0      0      0      0     39      0      0     10      0       0
[5,]    0      0      8      0      0     61      0      0      0       1
[6,]   36      0      0      0     11      2      0      0      3       0
```

```
[7,]    0      5      0      0      0     20     59      2      1       0
[8,]   23     21     43     25      3      0      0      3     17       0
[9,]    0      0      1      0      0      0      0      0      0       0
[10,]   0      0      0      0      0     10      0      0     20      32
> res$log.likelihoods
[...]

           [,98]      [,99]     [,100]
[1,]  -1015611.5 -1015942.5 -1015845.9
[2,]   -902856.1  -903184.8  -903122.7
```

Note that the topic assignment values from `res$assignments` range from 0 to 9, whereas the row indices, which correspond to topics, of `res$document_sums` go from 1 to 10. Note also that because the LDA process is not deterministic, the output might differ from the current example.

4.4.3 Using the Log-Likelihood for Model Validation

Given a statistical model and parameter values, one can ask what the probability of the observed outcomes is. Alternatively, given a set of outcomes, one can ask how likely the set of parameters is. This probability is what we mean by likelihood. It is a measure of the parameters being appropriate given a set of observations.

In our case, the set of observations is the set of existing documents and the different topic associations for each word occurring in each document. The parameters are the α and η parameters, the multinomial and dirichlet distributions, and K, the number of topics. By itself, the log-likelihood value does not have a significant meaning. But, different models can be compared based on their log-likelihood values. In other words, given the same set of observations, two models having different parameters can be compared based on their likelihood: a model with greater likelihood is considered more appropriate given the set of observations.

In many cases, it is more convenient to work with the natural logarithm of the likelihood value, known as the log-likelihood. Because the log function is monotone increasing, the maximum of both the log-likelihood and the likelihood coincide at the same point.

By specifying the `compute.log.likelihood` flag, one can plot the returned log-likelihood values to verify convergence (Figure 4.1). A simple way to do so could be:

```
> par(mfrow=c(2,1), pch=20)
> plot(1:100, res$log.likelihoods[1,])
> plot(1:100, res$log.likelihoods[2,])
```

From the plots, we can observe that the log-likelihood convergence slowly. After 50 iterations, the additional iterations did not seem to improve the model so much. The choice of 100 iterations in our example was a first blind attempt at generating a valid model. By plotting the log-likelihood values, one can determine whether modifying the number of iterations can significantly impact the log-likelihood of the model, hence improve the final model.

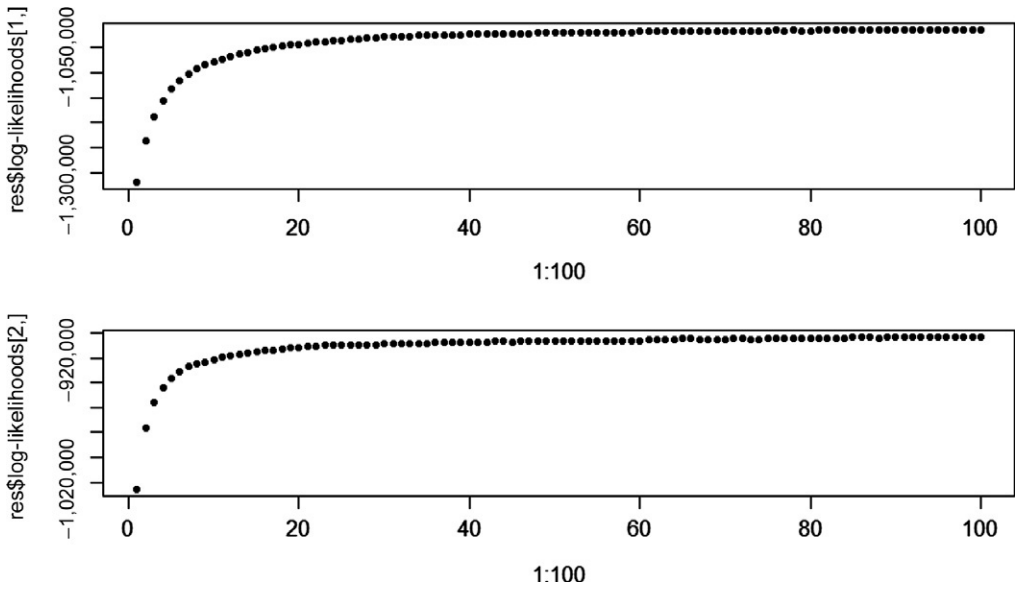


Figure 4.1
Log-likehood values from iterations.

4.4.4 Topics Representation

In our example, document 1 would have 36 terms associated with topic 6, and 23 terms with topic 8. Unfortunately, topics are not explicitly defined in our model. One way to describe each topic is to sort out associated terms with high frequency and scores. The `lda` package provides the `top.topic.words` function which outputs high-usage terms for each topic.

```
> top.topic.words(res$topics, 5, by.score=T)
     [,1]          [,2]         [,3]            [,4]          [,5]
[1,] "curvature"   "convex"     "random"        "algebra"     "space"
[2,] "surfaces"    "functions"  "process"       "algebras"    "operators"
[3,] "conjecture"  "sets"       "distribution"  "category"    "spaces"
[4,] "surface"     "set"        "processes"     "ring"        "operator"
[5,] "geometry"    "property"   "brownian"      "cohomology"  "theory"
     [,6]              [,7]        [,8]          [,9]          [,10]
[1,] "quantum"         "graph"     "equations"   "channel"     "systems"
[2,] "polynomials"     "graphs"    "solutions"   "capacity"    "stochastic"
[3,] "functions"       "set"       "equation"    "codes"       "model"
[4,] "noncommutative"  "bound"     "solution"    "network"     "numerical"
[5,] "theory"          "vertices"  "wave"        "coding"      "system"
```

In our example, topic 1 relates to mathematical physics. The LDA algorithm, in our current case, associated the majority of terms of document 1 to topic 6. In fact, the title of document 1 in arXiv.org is "Noncommutative Solitons and Quasideterminants" and is categorized in the mathematical physics subject.

The lda package also includes the `top.topic.documents` function which returns the top documents for each topic. The output is also in matrix format, just like the `top.topic.words` function.

4.4.5 Plotting the Topics Associations

Using a combination of the `reshape`, `ggplot2`, and `RColorBrewer` packages, one can create a stacked bar chart illustrating the weight of each topic for all documents in our corpus.

The `ggplot2` package is an advanced plotting system and is an implementation of the grammar of graphics in R. It gives the possibility to draw complex charts by exposing a way to manipulate plot aesthetic features to express multidimensional attributes of the data.

The `RColorBrewer` is an R package that provides predefined color palettes and ways to generate arbitrary color mappings shaded according to a variable.

The first step is the use of the `res$document_sums` matrix and reshape it into a `data.frame` object. The `melt` function allows the user to use a high-dimensional array and transform it into a dataframe. The coordinates of each matrix entry become columns in the dataframe, and each entry in the matrix gets turned into a row in the resulting dataframe.

```
> library(reshape)
     [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,]    0      0      0      0      0      6
[2,]    0      8      0      0      0      1
[3,]    0      8      0      4      0      0
[4,]    0      0      0      0     39      0
[5,]    0      0      8      0      0     61
[6,]   36      0      0      0     11      2
[7,]    0      5      0      0      0     20
[8,]   23     21     43     25      3      0
[9,]    0      0      1      0      0      0
[10,]   0      0      0      0      0     10
> d <- melt(res$document_sums)
> colnames(d) <- c("topic", "document", "value")
> head(d)
topic document value
1   1        1     0
2   2        1     0
3   3        1     0
4   4        1     0
5   5        1     0
6   6        1    36
```

Notice how the third column of the dataframe contains the entries of the matrix. Row and column indices become values in the resulting dataframe.

The following example will give us the desired plot, for the 50 first documents of our corpus (Figure 4.2).

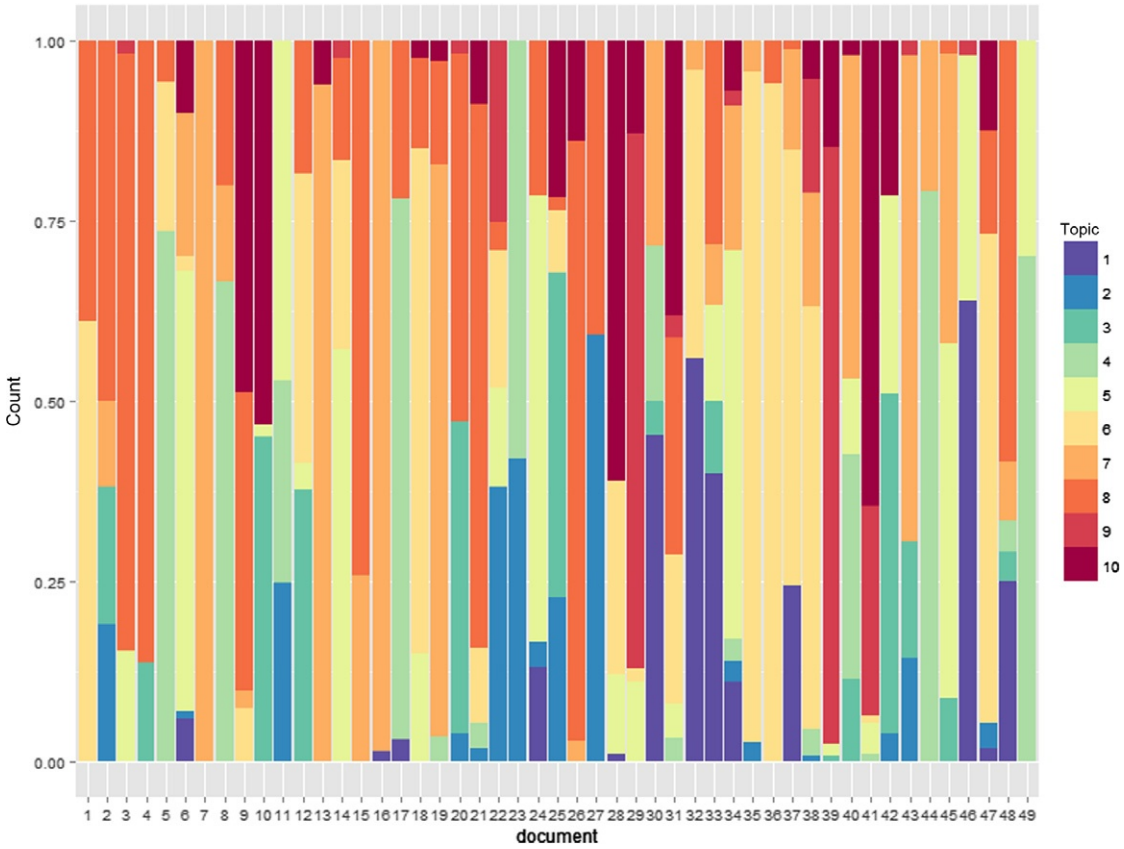


Figure 4.2
Distribution of topics in documents.

```
> library(ggplot2)
> library(RColorBrewer)
> d2 <- subset(d, d$document < 50)
> d2$topic <- as.factor(d2$topic)
> d2$document <- as.factor(d2$document)
> ggplot(d2, aes(x = document)) + geom_bar(aes(weight=value, fill = topic), position =
'fill') + scale_fill_manual(values = rev(brewer.pal(10, "Spectral")))
```

The stacked bar chart gives a more illustrative perspective of how the documents are distributed in the space of topics.

The `lda` package also includes two other methods to fit models: the mixed-membership stochastic blockmodel and the supervised LDA. They are called, respectively, by `slda.em` and `mmsb.collapsed.gibbs.sampler`. These LDA-type models have different features and are not discussed in the current chapter.

4.5 Using Similarity Between Documents to Explore Document Cohesion

4.5.1 Computing Similarities Between Documents

Let us determine how documents relate to each other in our corpus. Let t_1 and t_2 be two vectors, respectively, representing the topic associations of documents d_1 and d_2, where $t_1^{(i)}$ and $t_2^{(i)}$ are, respectively, the number of terms in d_1 and d_2, which are associated with topic i. One can then use the cosine similarity to derive a measure of document similarity:

$$\frac{\sum_i t_1^{(i)} * t_2^{(i)}}{\|t_1\| * \|t_2\|}$$

Here, $\|t_j\|$ denotes the norm of vector t_j.

In the current example, we will use the rows of the matrix `res$document_sums` as the list of features. Hence, two documents are similar if they share a similar topic distribution. The following lines will compute and output the similarity matrix for the documents.

```
> mat <- t(as.matrix(res$document_sums)) %*% as.matrix(res$document_sums)
> d <- diag(mat)
> sim <- t(t(mat/sqrt(d))/sqrt(d))
> sim[1:5, 1:5]
          [,1]       [,2]       [,3]       [,4]       [,5]
[1,] 1.0000000  0.46389797  0.52916839  0.53162745  0.26788474
[2,] 0.4638980  1.00000000  0.84688328  0.90267821  0.06361709
[3,] 0.5291684  0.84688328  1.00000000  0.97052892  0.07256801
[4,] 0.5316274  0.90267821  0.97052892  1.00000000  0.07290523
[5,] 0.2678847  0.06361709  0.07256801  0.07290523  1.00000000
```

The resulting matrix is a symmetric matrix where the entry in row i and column j represents the cosine similarity measure between documents d_i and d_j. The larger the entries, the more similar the publications are in terms of topic associations. An entry of 1 indicates identical publications in terms of topic associations.

In our example, documents 3 and 5 are completely dissimilar and documents 2 and 3 are somewhat similar. As a matter of fact, document 3 relates to the analysis of partial differential equations and document 5 discusses quantum algebra. Document 2 in our corpus is a scientific paper discussing the analysis of partial differential equations as well.

Alternatively, one can use the `res$document_sums` matrix to compute distances between the documents, instead of using cosine similarity measure. The `dist` function in R allows one to do so.

```
> as.matrix(dist(t(res$document_sums)))[1:5, 1:5]
         1        2        3        4        5
1  0.00000  38.11824  41.96427  36.27671  50.45790
2 38.11824   0.00000  26.49528  11.00000  46.03260
3 41.96427  26.49528   0.00000  20.12461  57.50652
4 36.27671  11.00000  20.12461   0.00000  46.28175
5 50.45790  46.03260  57.50652  46.28175   0.00000
```

Again, the distance between documents 2 and 3 is relatively small compared to other distance values, which reflects the fact that they are somewhat similar.

The `dist` function accepts many arguments, but the most important one is the method used for computing distances. This makes it easier to adjust the distance calculation method to the underlying dataset and objectives. The provided options are the euclidean, which happens to be the default one, the maximum, the manhattan, the canberra, the binary, and the minkowski distance methods. The different distance methods are detailed in the `dist` function help page.

4.5.2 Using a Heatmap to Illustrate Clusters of Documents

Now that the similarity matrix has been constructed, where similarity in our case is based on volume of topic associations by document, we can chart the different similarities on a heatmap and visualize which groups of documents are more likely clustered together. The simplest way to do so is to use the `heatmap` function (Figure 4.3).

```
> heatmap(sim[1:100,1:100])
> heatmap(sim[1:20, 1:20])
```

A yellow square indicates strong similarity, whereas a red square indicates distant documents. The second heatmap shows a subset of the same values for better readability. The reordering of the rows and columns are illustrated with dendrograms to the left and on the top of the chart. The default clustering method for the `heatmap` function is the hierarchical clustering, which is implemented in the `hclust` function.

The first step of hierarchical clustering assigns each document to its own cluster. Then, at each iteration, the two most similar clusters are joined together in a single cluster. Distances between clusters are then recomputed, and further iterations of cluster joins are executed. The algorithm ends when only one cluster is left (Feldman and Sanger, 2006). The dendrograms shown on the charts represent each step of the hierarchical clustering algorithm.

In this case, for the second heatmap, we can see that documents 20 and 19 are similar to each other and so are documents 5, 8, 11, and 17.

The `heatmap` function has many other arguments, such as color choices, arbitrary distance, and clustering functions and more. The help page for the `heatmap` function is a good reference for more details about the different features.

4.6 Social Network Analysis of Authors

4.6.1 Constructing the Network as a Graph

In the current corpus, the names of authors have been extracted and stored separately. To load the list of authors per publication, write

```
> load("s.authors.RData")
```

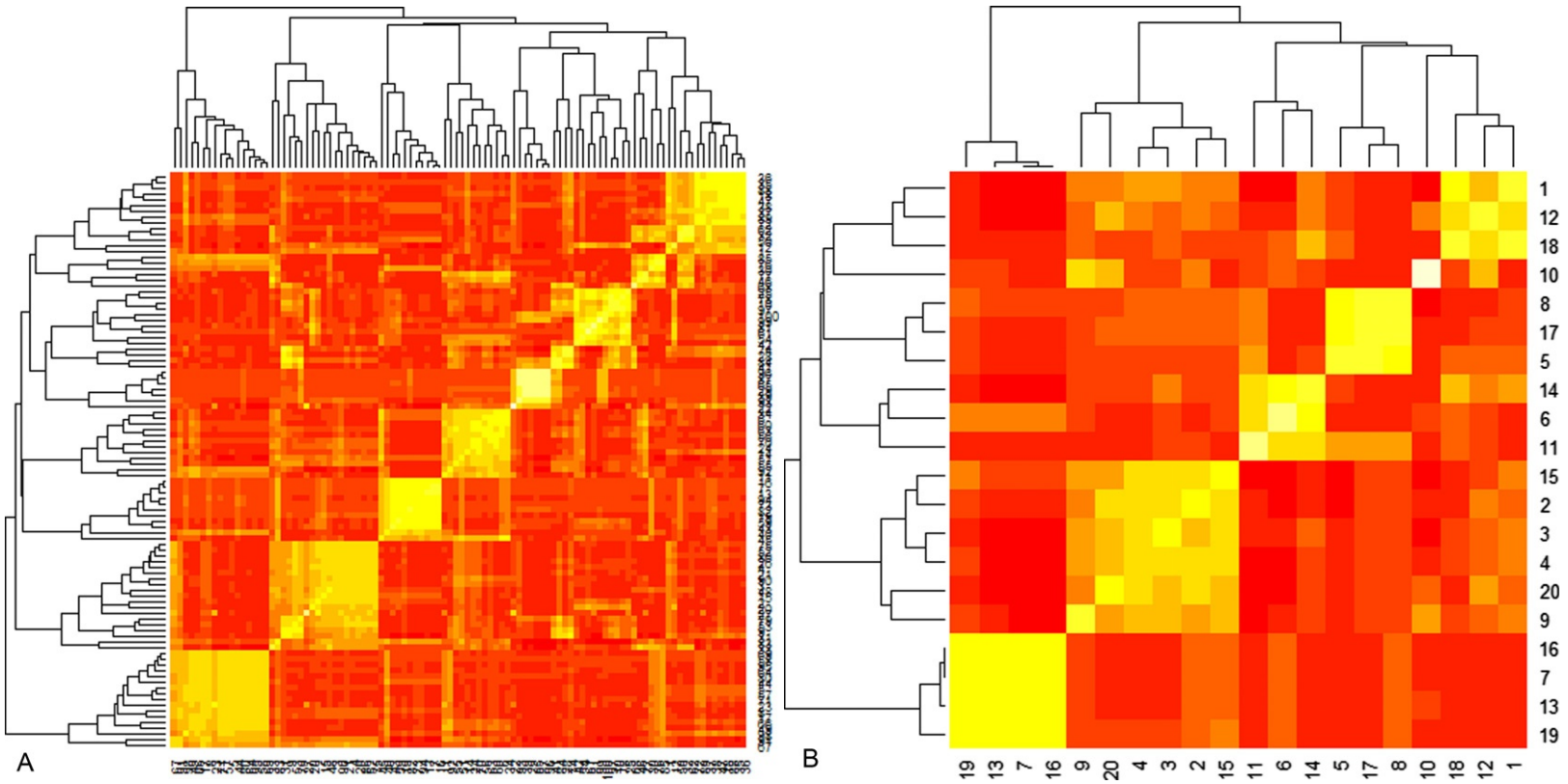


Figure 4.3

Heatmap of similarities between documents. A) The left heatmap represents the similarity matrix from the first 100 documents. B) The heatmap on the right depicts the same idea for the first 20 documents of our corpus.

which will load a `s.authors` object into your R environment.

```
> head(s.authors, 4)
$`1101.0012.txt`
[1] "Dirk Hundertmark" "Shuanglin Shao"

$`1101.0013.txt`
[1] "Jinggang Tan" "Jingang Xiong"

$`1101.0015.txt`
[1] "Michael Gekhtman" "Michael Shapiro" "Alek Vainshtein"

$`1101.0017.txt`
[1] "Tran Vu Khanh" "Stefano Pinton" "Giuseppe Zampieri"
```

The following is an attempt at a simple analysis of the network of authors from our corpus. We construct a graph where the nodes consist of authors, and an edge represents a collaboration event between authors. In our case, collaborations between authors will be defined by a joint paper.

We first remove the papers which were written by only one author.

```
> getJointPapers <- function(s) { if (length(s) > 1) s else c() }
> s.authors <- sapply(s.authors, getJointPapers)
```

The graph library used in this exercise is the `igraph` package (Csardi and Nepusz, 2006). The `igraph` package is used mainly for network analysis allowing fast analysis and prototyping (Figure 4.4). The typical usage is

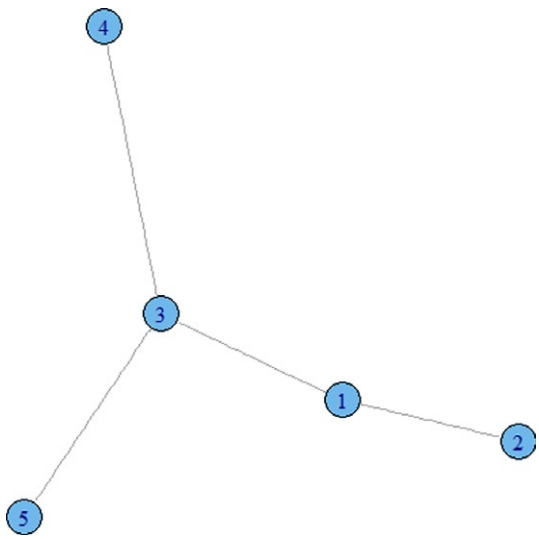


Figure 4.4
A simple graph.

```
> library(igraph)
> foo <- graph(c(1,2,1,3,3,4,3,5), directed=F)
> plot(foo)
```

The above example creates an undirected graph with edges between vertices 1 and 2, between vertices 2 and 3, and between vertices 3 and 4. The first argument of the graph function, as used in the example, is the list of edges where all pairs are merged into a single array.

The vertices for the graph object are identified by integers, which in our case has to be associated with author names. The list of all distinct authors is constructed with:

```
> authors <- unique(as.vector(unlist(s.authors)))
> head(authors)
[1] "Dirk Hundertmark" "Shuanglin Shao" "Jinggang Tan"
[4] "Jingang Xiong" "Michael Gekhtman" "Michael Shapiro"
```

We will use the indices of the `authors` list as the vertex identifier for the graph object, by using a helper function.

```
> getAuthorId <- function(n) which(authors == n)
> getAuthorId("Jinggang Tan")
[1] 3
```

Because we chose to represent our network of authors with graphs, and not hypergraphs, edges only link two authors at a time, whereas papers can have more than two coauthors. From the dataset, the third entry of the `s.authors` object lists three different authors:

```
> s.authors[[3]]
[1] "Michael Gekhtman" "Michael Shapiro" "Alek Vainshtein"
```

In this case, we would like to transform this into a list of three edges with author identifiers. In other words, an edge is required for each pair of authors from the list of three authors. A simple way to generate that is to use the `combn` function. This function generates all combinations of the elements of a vector by taking a fixed number at a time.

```
> combn(letters[1:4], 2)
     [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
[1,] "a"   "a"   "a"   "b"   "b"   "c"
[2,] "b"   "c"   "d"   "c"   "d"   "d"
> combn(letters[1:4], 3)
     [,1]  [,2]  [,3]  [,4]
[1,] "a"   "a"   "a"   "b"
[2,] "b"   "b"   "c"   "c"
[3,] "c"   "d"   "d"   "d"
```

In the above example, the first four letters of the alphabet are taken 2 and 3 at a time to form combinations. We use the previous functions to create the list of edges for our network of authors, which we achieve by defining another helper function `getEdges`.

```
> getEdges <- function(s) { if (is.null(s) || length(s) < 2) { c() } else { unlist(lapply(combn
(s, 2), getAuthorId)) } }
> getEdges(s.authors[[3]])
```

```
[1] 5 6 5 7 6 7
> authors[c(5,6,7)]
[1] "Michael Gekhtman" "Michael Shapiro" "Alek Vainshtein"
```

The graph is then created with:

```
> g <- graph(as.vector(unlist(lapply(s.authors, getEdges))), directed=F)
> g
IGRAPH  U- - - 2639 2909 - -
```

The resulting `g` object is an undirected graph with 2639 vertices and 2909 edges. Plotting this graph isn't practical, but there are a few things we can analyze from this network.

4.6.2 Author Importance Using Centrality Measures

Given a graph, one can speak of the centrality of a vertex which represents the importance of a vertex within the graph. This can be measured with many centrality measures, among which is the betweenness centrality measure (Leydesdorff, 2007). The betweenness of a specific vertex is equal to the number of shortest paths from all pairs of vertices in the graph that pass through that vertex. Informally, the more the shortest paths that go through a vertex, the more important that vertex is in terms of graph connectivity.

Formally, the betweenness centrality of a vertex v is

$$b(v) = \sum \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

where σ_{uw} is the number of shortest paths between vertices u and w, and $\sigma_{uw}(v)$ is the number of shortest paths between u and w that pass through vertex v. The sum in the expression ranges over all pairs of distinct vertices u and w.

The higher the betweenness centrality value, the more central the vertex is. An implementation of the betweenness centrality computation is done in the `sna` R package. To apply the measure to the entire graph, we use:

```
> library(sna)
> b <- betweenness(as.array(get.adjacency(g)))
> head(b)
[1] 0 0 0 0 0 0
> which(b>25)
  [1]   28   205   528   605   670   726   1044   1414   2034
```

Among the list of 2639 authors, only 9 of them have betweenness centrality measures larger than 25.

```
> authors[which(b>25)]
[1] "Xueliang Li"   "Vincent K. N. Lau"   "Jean-Marie Mirebeau"
[4] "Terence Tao"   "Shunqing Zhang"      "Yuliya Babenko"
[7] "Van Vu"        "Guoqing Wang"        "V. Soucek"
```

The above code displays the name of these 9 authors by fetching the names from the `authors` array. To retrieve the authors' betweenness centrality values and use it to sort, one can use a dataframe just as in:

```
> top <- data.frame(name=authors[which(b>25)], centrality=b[which(b>25)])
> top[order(top$centrality, decreasing=T),]
                  name     centrality
1          Xueliang Li             55
8         Guoqing Wang             54
2     Vincent K. N. Lau             40
4          Terence Tao             34
7              Van Vu             32
3  Jean-Marie Mirebeau             30
5       Shunqing Zhang             30
6       Yuliya Babenko             30
9            V. Soucek             30
```

This gives us an idea of most central authors of the set of papers we are analyzing.

The betweenness centrality measure has a few variants which can affect the analysis of some networks. The length-scaled betweenness is such an example and it is given by

$$b(v) = \sum \frac{1}{d_{uw}} \frac{\sigma_{uw}(v)}{\sigma_{uw}}$$

where d_{uw} is the geodesic distance between vertices u and w. The geodesic distance is also known as the length of the shortest path between two vertices. The length-scaled betweenness downgrades the weight given to long paths, in situations where such paths are not likely to be useful in the analysis of the network. One can look at more variants in Brandes (2007).

To use the shortest-path betweenness centrality variants with the `betweenness` function, the `cmode` argument is required.

```
> b <- betweenness(as.array(get.adjacency(g)), cmode="lengthscaled")
> top <- data.frame(name=authors[which(b>12)], centrality=b[which(b>12)])
> top[order(top$centrality, decreasing=T),]
                name       centrality
1        Xueliang Li             27.5
3       Guoqing Wang             27.0
2   Vincent K. N. Lau             16.0
4          V. Soucek             13.0
```

We can see from this example that some authors from the first list do not appear in the list of top authors when using shortest-path betweenness centrality.

Other variants of shortest-path betweenness centrality measures are available. They include a linearly scaled betweenness measure, a proximal source and target betweenness (where the fact that the vertex is closer to the start of the shortest path downgrades or upgrades the weight used

in the formula), and other methods. The full details of such options are available in the `sna` package reference manual (Butts, 2010).

The `sna` package contains a wide variety of functions, including closeness measures, functions for components analysis, hierarchy analysis, and much more. Different alternative centrality measures are also implemented, which could be useful in some cases.

In this example, we analyzed the interactions between authors of our network, where interactions represent collaborative work. A similar analysis can be applied to other types of networks as well, and the `sna` R package provides many tools to handle this type of exercise.

4.7 Conclusion

From the content of a digital library such as arXiv.org, R has been used with a wide variety of packages to analyze the textual content of the scientific publications, the term occurrences within the different documents, the co-occurrences of terms with other terms, the clustering of papers into different arbitrary topics, and the network analysis of their authors. The LDA algorithm allowed the user to learn a topic assignment model given a set of documents and words. A few plots and charts were plotted using the `ggplot2` package to visually enhance the analytical content of the different techniques used in the current chapter.

References

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. J Mach Learn Res 3, 993–1022.

Brandes, U., 2007. On variants of shortest-path betweenness centrality and their generic computation. Soc Netw 30 (2), 136–145.

Butts, C.T., 2010. sna: tools for social network analysis. R package version 2.2-2. http://cran.r-project.org/web/packages/sna/sna.pdf.

Csardi, G., Nepusz, T., 2006. The igrah software package for complex network research. http://igraph.sf.net.

Feinerer, I., 2011. tm: text mining package. R package version 0.5-5. http://cran.r-project.org/package=tm.

Feldman, R., Sanger, J., 2006. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, New York.

Grun, B., Hornik, K., 2011. Topicmodels: an R package for fitting topic models. http://cran.r-project.org/web/packages/topicmodels/vignettes/topicmodels.pdf.

Leydesdorff, L., 2007. Betweenness centrality as an indicator of the interdisciplinarity of scientific journals. J. Am. Soc. Inf. Sci.. 58 (9), 1303–1319.

Salton, G., Buckley, C., 1988. Term-weighting approaches in automatic text retrieval. Inform. Process. Manag. 24 (5), 513–523.

Salton, G., McGill, M.J., 1986. Introduction to Modern Information Retrieval. McGraw-Hill, New York.

CHAPTER 5

Recommender Systems in R

Saurabh Bhatnagar
New York, USA
http://sanealytics.wordpress.com/

5.1 Introduction

Recommender systems are pervasive. You have encountered them while buying a book on www.barnesandnoble (if you like this. . .), renting a movie on Netflix, listening to music on Pandora, finding the bar to visit (FourSquare), to possibly even finding love at match.com. In this chapter, I show you how easy it is to write a recommendation engine in *R* and more importantly, test it out with real-world data to make sure it works. Recommendation systems help users find the right choices in an increasingly complex domain. But there are places where they fall short (black sheep, cold start). Later in the chapter, we talk about how to address those issues.

5.2 Business Case

Many retailers carry a myriad of products. It becomes an issue of matching the right product to the right person. Imagine going to a small wine shop, where the sommelier knew what you had purchased and liked earlier, and could recommend other wines for your taste. He might ask you a few questions (white or red today? fish or meat? meal or table?) but he will quickly try to get you that perfect bottle for the evening. Now no sommelier can keep track of thousands of transactions per day from millions of customers on a website. This expert system is a recommender engine.

5.3 Evaluation

Before we talk about recommendation systems, we have to talk about what qualities we are looking for. This will help us judge which recommendation system works better than the others given our data. A few metrics used commonly are:

RMSE (Root Mean Squared Error): Here, we measure how far real ratings are from the ones we predicted. Mathematically, we can write it out as

Data Mining Applications with R

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\kappa}\left(r_{(i,j)} - \hat{r}_{(i,j)}\right)^2}{|\kappa|}}$$

where k is the set of all user-item pairings (i, j) for which we have a predicted rating $\hat{r}_{(i,j)}$ and a known rating $r_{(i,j)}$, which was not used to learn the recommendation model.

Precision/Recall/f-value/AUC: Precision tells us how good the predictions are. In other words, how many were a hit.

$$\text{precision} = \left|\frac{\{\text{relevantdocuments}\} \cap \{\text{retrieveddocuments}\}}{\{\text{retrieveddocuments}\}}\right.$$

Recall tells us how many of the hits were accounted for, or the coverage of the desirable outcome.

$$\text{recall} = \left|\frac{\{\text{relevantdocuments}\} \cap \{\text{retrieveddocuments}\}}{\{\text{relevantdocuments}\}}\right.$$

Precision and recall usually have an inverse relationship. This becomes an even bigger issue for rare issue phenomenon like recommendations. To tackle this problem, we will use *f*-measure. This is nothing but the harmonic mean of precision and recall.

$$f = \text{value} = \frac{2 * \text{precision.recall}}{\text{precision} + \text{recall}}$$

Another popular measure is AUC. This is roughly analogous. The higher the AUC, the better we did at guessing what the user actually picked. It does not take ratings into account. Since this is more common, we will use it for our recommendation effectiveness. We will plot ROC on true positive rate vs. false positive rate. A true positive is when the recommended item was picked by the user. A false positive is when a recommended item was not picked by the user.

ARHR (Hit Rate): When returning a ranked list

$$\text{ARHR} = \frac{1}{\#\text{users}}\sum_{i=1}^{\#\text{hits}}\frac{1}{p_i}$$

where p is the position of the item in a ranked list.

There are other factors to consider besides just these metrics, like speed of recommendation, precalculation or post. Others are more esoteric like serendipity. These will make more sense as we go over some of these methods.

5.4 Collaborative Filtering Methods

Now for the fun part. If my friend Jimmy tells me that he liked the movie "Drive," I might like it too as we have similar tastes. However, if Paula tells me she liked "The Notebook," I might avoid it because we usually don't like the same movies. This is called UBCF (User-based

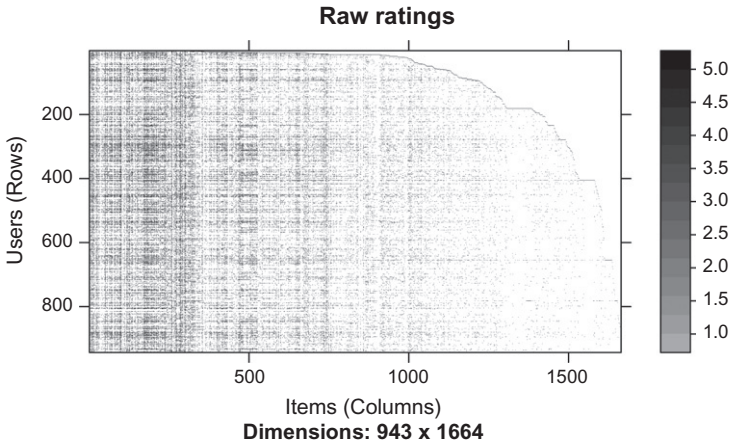


Figure 5.1
Raw ratings of MovieLense database.

collaborative filtering). Another way to think about it is soft-clustering. We find Users with similar tastes (neighborhood) and use their preferences to build yours (Figure 5.1).

Another flavor of this is IBCF (Item-Based Collaborative Filtering). If I watched "Darjeeling Limited," I might be inclined to watch "The Royal Tannenbaums" but not necessarily "Die Hard." This is because the first two are more similar in the users who have watched/rated them. This is rather simple to compute as all we need is the covariance between products to find out what this might be. You have seen this at Amazon (if you like this).

Let's compare both approaches on some real data (thanks R). We will use the packages recommenderlab for evaluation (Hahsler, 2011) and ggplot2 for graphics (Wickham, 2009).

```
> # Load required library
> library(recommenderlab)
> library(ggplot2) # For plots
> # Load the data we are going to work with
> data(MovieLense)
> MovieLense

943 × 1664 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 99392 ratings.

>
> # Visualizing a sample of this
> image(MovieLense, main = "Raw ratings")
```

It appears that some movies were not rated at all by first few users. Maybe they were released later (Figures 5.2 and 5.3).

```
> summary(getRatings(MovieLense)) # Skewed to the right

    Min.      1st Qu.      Median      Mean      3rd Qu.      Max.
    1.00      3.00         4.00        3.53      4.00         5.00
```

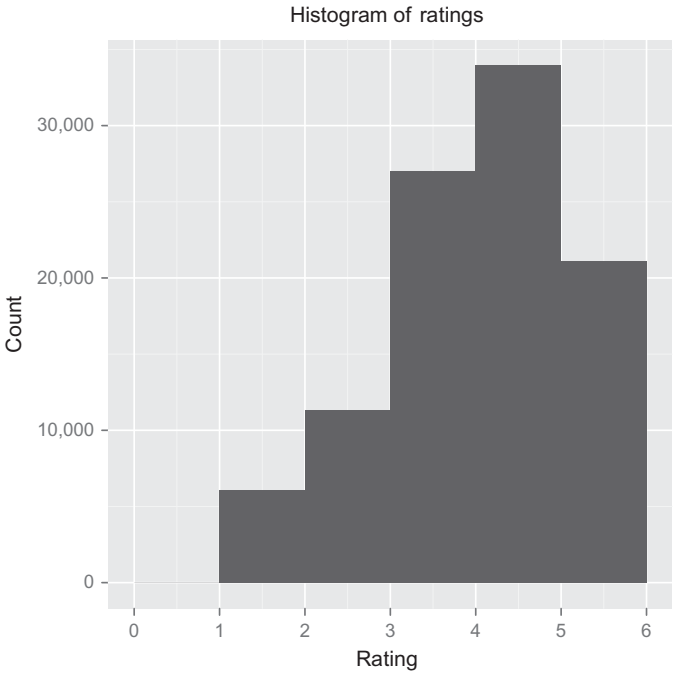


Figure 5.2
Ratings of MovieLense.

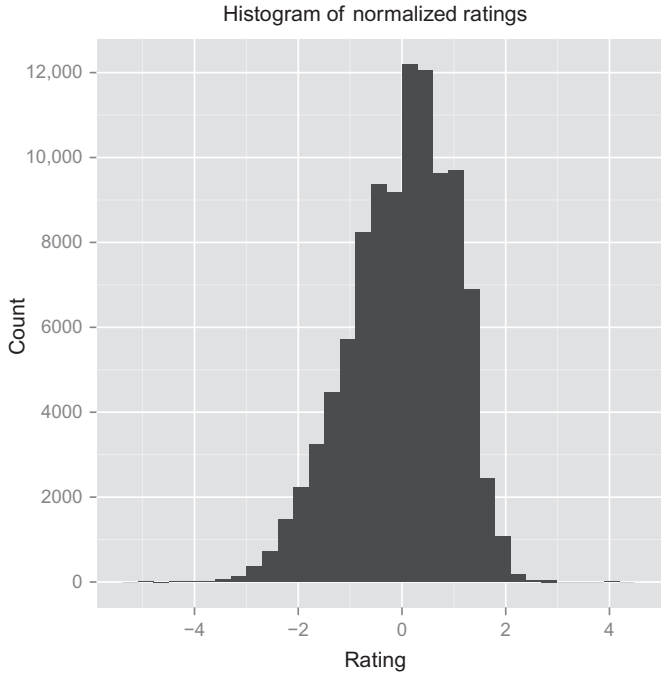


Figure 5.3
Normalized ratings of MovieLense.

```
> # Visualizing ratings
> qplot(getRatings(MovieLense), binwidth = 1,
+       main = "Histogram of ratings", xlab = "Rating")
```

Looks skewed to the right. What about after normalization?

```
> qplot(getRatings(normalize(MovieLense, method = "Z-score")),
+       main = "Histogram of normalized ratings", xlab = "Rating")
> summary(getRatings(normalize(MovieLense, method = "Z-score")))
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-4.8520  -0.6466   0.1084   0.0000   0.7506   4.1280
```

It seems better. Here is why normalization works. It adjusts for the bias of individual raters. For example, Sam might rate movies at 4 more often than Robin who usually rates them at 2. The normalization will take care of that user bias.

```
> # How many movies did people rate on average
> qplot(rowCounts(MovieLense), binwidth = 10,
+       main = "Movies Rated on average",
+       xlab = "# of users",
+       ylab = "# of movies rated")
```

Seems people get tired of rating movies at a logarithmic pace. But most rate some (Figure 5.4).

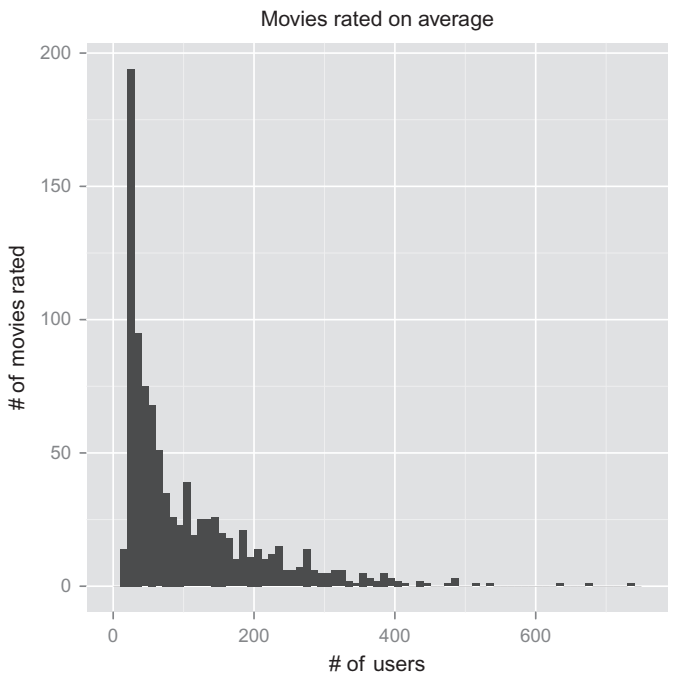


Figure 5.4
Distribution of movie raters.

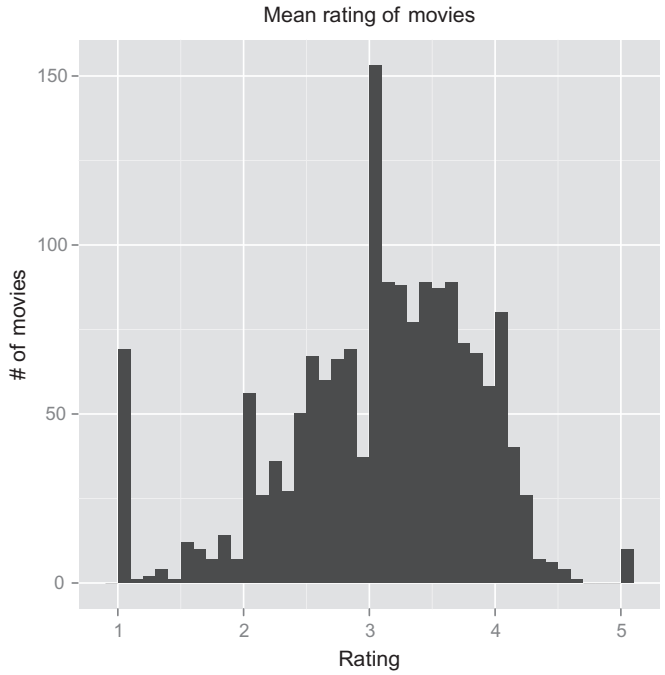


Figure 5.5
Mean rating of movies.

```
> # What is the mean rating of each movie
> qplot(colMeans(MovieLense), binwidth = .1,
+       main = "Mean rating of Movies",
+       xlab = "Rating",
+       ylab = "# of movies")
```

The big spike on 1 suggests that this could also be interpreted as binary. In other words, some people don't want to see certain movies at all. Same on 5 and on 3. We will give it the binary treatment later and see why that makes sense (Figure 5.5).

To evaluate recommender systems, we will split the data into test and training sets. We will use the training set to build our model. We will hold out some items from the test set, then make predictions using the model. Then we will compare our predictions against the holdout. If we predicted correctly, it is a true positive. If we predicted incorrectly, it is a false positive.

For production, I suggest using cross-validation. This is the same as above, except you slice it again and again. This makes sure that there was no bias in the slicing of data.

Let's see what algorithms can we test against?

```
> recommenderRegistry$get_entries(dataType = "realRatingMatrix")

$IBCF_realRatingMatrix
Recommender method: IBCF
```

```
Description: Recommender based on item-based collaborative filtering (real data).

$POPULAR_realRatingMatrix
Recommender method: POPULAR
Description: Recommender based on item popularity (real data).

$RANDOM_realRatingMatrix
Recommender method: RANDOM
Description: Produce random recommendations (real ratings).

$UBCF_realRatingMatrix
Recommender method: UBCF
Description: Recommender based on user-based collaborative filtering (real data).

> # We have a few options
>
> # Split the data into train and test. Here, train is 90%.
> # For testing it will take any 10 movie ratings by the user
> # and predict n others. Then compare to see if they match.
>
> scheme <- evaluationScheme(MovieLense, method = "split", train = .9,
+                            k = 1, given = 10, goodRating = 4)
> scheme

Evaluation scheme with 10 items given
Method: âĂŸsplitâĂŹ with 1 run(s).
Training set proportion: 0.900
Good ratings: >=4.000000
Data set: 943 × 1664 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 99392 ratings.

> # Here we are using split, but other schemes are also available
> # For production testing, I STRONGLY recommend using cross-validation scheme
>
> # Let's check some algorithms against each other
> algorithms <- list(
+    "random items" = list(name="RANDOM", param=list(normalize = "Z-score")),
+    "popular items" = list(name="POPULAR", param=list(normalize = "Z-score")),
+    "user-based CF" = list(name="UBCF", param=list(normalize = "Z-score",
+                                                   method="Cosine",
+                                                   nn=50, minRating=3)),
+    "item-based CF" = list(name="IBCF", param=list(normalize = "Z-score"
+                                                   ))
+
+ )
>
```

You probably noticed normalize = "Z-score," but by now you would have expected that. The parameter that might be perplexing is method = "Cosine" for UBCF. Remember when we said that both user-based and item-based are working on similarity of neighborhoods. To define a neighborhood, we need a measure to tell them what similarity is. Cosine is most widely used. There are many others, like Jaccard, karypis, conditional. To go into them would be beyond the scope of this chapter.

nn is simply suggesting how many neighbors to consider. Make the neighborhood too small, and you might get recommendations that are all over the place. Make it too big and the recommendations might be too general. In the extreme case, too small might result in just one neighbor (one who has seen the same movie), so no useful recommendations can be generated. Too large in extreme will be the entire population, so it will result in a general popularity index, no nice user specific recommendations that we want. I am going to put that to 50 for now (Figures 5.6 and 5.7).

```
> # run algorithms, predict next n movies
> results <- evaluate(scheme, algorithms, n=c(1, 3, 5, 10, 15, 20))

RANDOM run
        1 [0.004 sec/0.506 sec]
POPULAR run
        1 [0.054 sec/0.1 sec]
UBCF run
        1 [0.048 sec/2.55 sec]
IBCF run
        1 [55.679 sec/0.508 sec]
> # Draw ROC curve
> plot(results, annotate = 1:4, legend="topleft")
> # See precision / recall
> plot(results, "prec/rec", annotate=3)
```

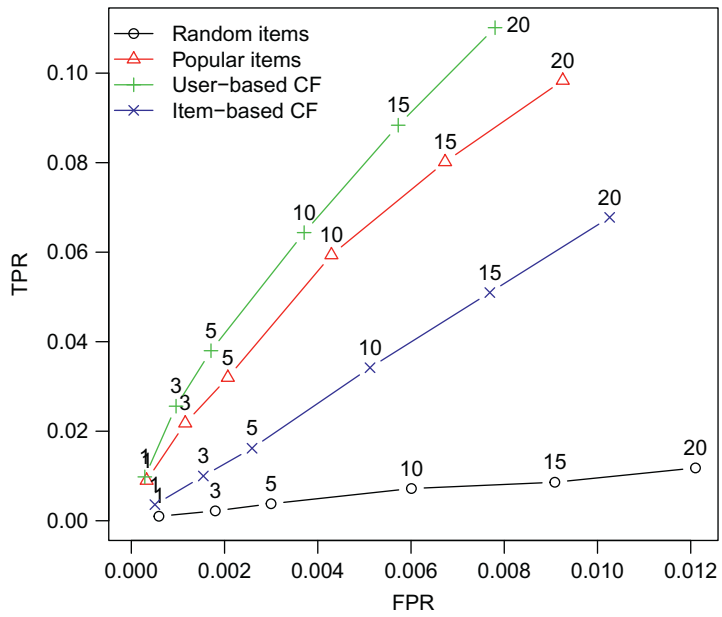


Figure 5.6
Evaluation of standard recommendation algorithms against Movie-lense.

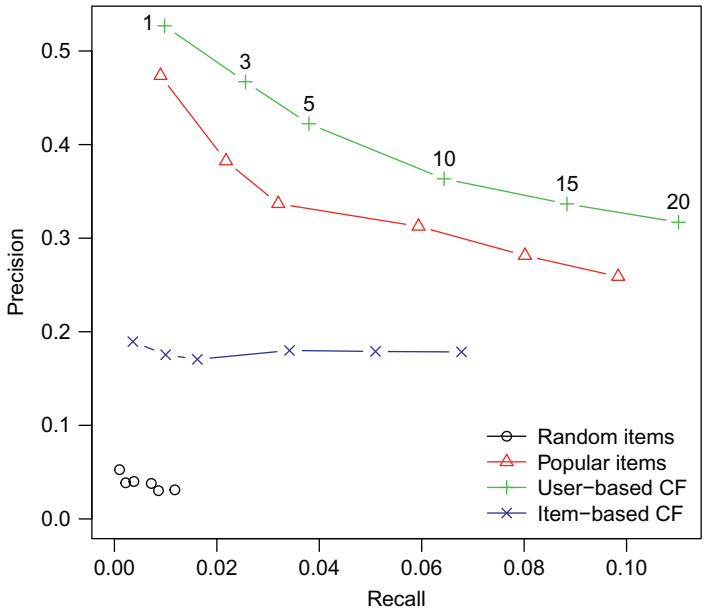


Figure 5.7
Evaluation of standard recommendation algorithms against Movie-lense.

It seems like UBCF did better than IBCF. Then why would you use IBCF? The answer lies in when and how are you generating recommendations. UBCF saves the whole matrix and then generates the recommendation at predict by finding the closest user. For large number of users-items, this becomes an issue. IBCF saves only k closest items in the matrix and doesn't have to save everything. It is precalculated and predict simply reads off the closest items.

Predictably, RANDOM is the worst but perhaps surprisingly it seems, it's hard to beat POPULAR. All this means that the movies rated high are usually liked by everyone and are safe recommendations. This might be different for other datasets.

Before we talk about other methods, I would like to draw your attention about what the ROC curves aren't telling us. UBCF does better but is more expensive to generate recommendations at recommend time, as it uses the whole matrix. On the other hand, IBCF finds what's good between items, but loses the nuances of different user groups. So the recommendations are not that surprising. If you want serendipity, UBCF does a better job. A user who is like you is more likely to tell you about a "cult" classic that might be lost on the general population.

We are gong to implement a few collaborative filtering algorithms. So let's see what realRatingMatrix class really is:

```
> # Let's start with a regular matrix of 5 users, 10 items
> set.seed(2358)
> my.mat <- matrix(sample(c(as.numeric(-2:2), NA), 50,
+                    replace=TRUE,
+                    prob=c(rep(.4/5,5), .6)), ncol=10,
+             dimnames=list(user=paste("u", 1:5, sep=''),
+                           item=paste("i", 1:10, sep='')))
> my.mat

        item
  user   i1   i2   i3   i4   i5   i6   i7   i8   i9   i10
    u1   2    NA   NA   NA   2    NA   -1   NA   -1   NA
    u2   NA   2    0    NA   NA   NA   2    2    NA   NA
    u3   NA   NA   NA   NA   -1   NA   1    NA   -1   NA
    u4   NA   NA   1    NA   -1   NA   NA   NA   NA   NA
    u5   NA   NA   -1   2    NA   NA   0    -2   NA   NA

> # Here user u2 has rated i2 as 2 and i3 as 0.
> # Please note that 0 could be a valid value
> # All unrated values are NA
>
> # Convert this to realRatingMatrix
> (my.realM <- as(my.mat, "realRatingMatrix"))

5 × 10 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 17 ratings.

> str(my.realM)

Formal class 'realRatingMatrix' [package "recommenderlab"] with 2 slots
  ..@ data     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  .. .. ..@ i        : int [1:17] 0 1 1 3 4 4 0 2 3 0 ...
  .. .. ..@ p        : int [1:11] 0 1 2 5 6 9 9 13 15 17 ...
  .. .. ..@ Dim      : int [1:2] 5 10
  .. .. ..@ Dimnames:List of 2
  .. .. .. ..$ user:  chr [1:5] "u1" "u2" "u3" "u4" ...
  .. .. .. ..$ item:  chr [1:10] "i1" "i2" "i3" "i4" ...
  .. .. ..@ x        : num [1:17] 2 2 0 1 -1 2 2 -1 -1 -1 ...
  .. .. ..@ factors  : list()
  ..@ normalize: NULL

> # Hmm, can we look at the underlying object?
> rating.obj <- my.realM@data
> # This is the class called sparse Matrix (notice the uppercase M)
> # By default all 0 s in Matrix are dropped to save space.
> # Since we expect mostly NAs, it has taken our input mat,
> # and converted it to 0 s. We can do this another way
>
> dropNA(my.mat)
5 × 10 sparse Matrix of class "dgCMatrix"
```

```
u1  2  .   .  .   2  .  −1   .  −1  .
u2  .  2   0  .   .  .   2   2   .  .
u3  .  .   .  .  −1  .   1   .  −1  .
u4  .  .   1  .  −1  .   .   .   .  .
u5  .  .  −1  2   .  .   0  −2   .  .
> identical (rating.obj, dropNA(my.mat))

[1] TRUE

> # OK, let's convert it back
> as.matrix(rating.obj)
        item
   User  i1  i2  i3  i4  i5  i6  i7  i8  i9  i10
     u1   2   0   0   0   2   0  −1   0  −1   0
     u2   0   2   0   0   0   0   2   2   0   0
     u3   0   0   0   0  −1   0   1   0  −1   0
     u4   0   0   1   0  −1   0   0   0   0   0
     u5   0   0  −1   2   0   0   0  −2   0   0
> # This is wrong. We had NAs!
> # What happened here is as.matrix applied to Matrix class,
> # and so it translated it zeroes instead of NAs.
> # For the right translation, we need -
> as (my.realM, "matrix")
        item
   User  i1  i2  i3  i4  i5  i6  i7  i8  i9  i10
     u1   2  NA  NA  NA   2  NA  −1  NA  −1  NA
     u2  NA   2   0  NA  NA  NA   2   2  NA  NA
     u3  NA  NA  NA  NA  −1  NA   1  NA  −1  NA
     u4  NA  NA   1  NA  −1  NA  NA  NA  NA  NA
     u5  NA  NA  −1   2  NA  NA   0  −2  NA  NA
>
```

So this means that if we are to run a standard function against realRatingMatrix, we have to be careful to what the conversion really is. Keeping this in mind, let's move on. Sparse Matrices (Bates and Maechler, 2012) do not store zeroes and thus save a lot of space. Most recommendation system datasets are pretty sparse (very few items rated from a large catalog), so this makes sense.

5.5 Latent Factor Collaborative Filtering

Now on to some recent algorithms that have worked well (à la Netflix prize). SVD (Singular value decomposition) is a way to decompose a matrix. So if we take a matrix R, we can reduce it to

$$R = U.S.V'$$

where S is the diagonal.

Why is this relevant? Imagine there are k categories of items. So you can create a kxi matrix from i items by putting i items in k categories. Similarly you can put all users in these k categories knowing how much they like each category. That will be a uxk matrix.

So now you have A_{uxk} and B_{kxi}. And if you were to multiply these together you would get S_{uxi}, your original matrix back. Pretty neat.

But how do you take this large matrix and figure out those A and B matrices? One way is SVD decomposition. Again, we can think of recommendation systems as soft-clustering. If you now take only k elements from U, S, and V, you can multiply them and get something close to the original matrix, but not the same. This can be represented as

$$R_k = U_k.S_k.V_k'$$

S is a diagonal, so taking k elements from that is trivial. This approximate matrix can be saved for recommendations. So if you have to, give recommendations for user u_l. You can simply read off the row from R_k for him.

Another nice thing about the diagonal is that it is ordered from the most variance to the least. So taking the first k elements works just fine.

This method is very efficient for large sparse matrixes and for Netflix prize, and has shown some nice results. See Sarwar et al. (2000).

Let's go ahead and implement this idea in *R*. We are going to use Recommenderlab's framework so that we can check our work against other results.

```
> # At the time of writing this, you still have to create function .get_parameters,
> # but you can take the code from AAA.R so it is trivial.
>
> ## helper functions and registry
> # From AAA.R
> .get_parameters <- function(p, parameter) {
+     if(!is.null(parameter) && length(parameter) != 0) {
+         o <- pmatch(names(parameter), names(p))
+
+         if(any(is.na(o)))
+         stop(sprintf(ngettext(length(is.na(o)),
+                     "Unknown option: %s",
+                     "Unknown options: %s"),
+                 paste(names(parameter)[is.na(o)],
+                     collapse = " ")))
+
+         p[o] <- parameter
+     }
+
+     p
+  }
> # Now our new method using SVD
> REAL_SVD <- function(data, parameter= NULL) {
+
```

```
+   p <- .get_parameters(list(
+     categories = 50,
+     method="Cosine",
+     normalize = "center",
+     normalize_sim_matrix = FALSE,
+     alpha = 0.5,
+     treat_na = "0",
+     minRating = NA
+     ), parameter)
+
+   # Do we need to normalize data?
+   if(!is.null(p$normalize))
+     data <- normalize(data, method=p$normalize)
+
+   # Just save everything for now.
+   model <- c(list(
+     description = "full matrix",
+     data = data
+     ), p)
+
+   predict <- function(model, newdata, n = 10,
+                       type=c("topNList", "ratings"), ...) {
+
+     type <- match.arg(type)
+     n <- as.integer(n)
+
+     # Do we need to denormalize?
+     if(!is.null(model$normalize))
+        newdata <- normalize(newdata, method=model$normalize)
+
+     # Get the old data
+     data <- model$data@data
+     # Add new data to it to create combined matrix
+     data <- rBind(data, newdata@data)
+
+     ### svd does as.matrix which sets all missing values to 0!
+     # So we have to treat missing values before we pass it to svd (fix by Michael Hahsler)
+     data <- as(data, "matrix")
+
+     if(model$treat_na=="min") data[is.na(data)] <- min(data, na.rm=TRUE)
+     else if(model$treat_na=="mean") data[is.na(data)] <- mean(data, na.rm=TRUE)
+     else if(model$treat_na=="median") data[is.na(data)] <- median(data, na.rm=TRUE)
+     else if(model$treat_na=="max") data[is.na(data)] <- max(data, na.rm=TRUE)
+     else if(model$treat_na=="0") data[is.na(data)] <- 0
+     else stop("No valid way to treat NAs specified (treat_na)!")
+
+     # Calculate SVD using available function
+     s<-svd(data)
+
+     # Get Diag but only of p elements
+     S <- diag(s$d[1:p$categories])
+
```

```
+     # Multiply it back up, but only using p elements
+     ratings <- s$u[,1:p$categories] %*% S %*% t(s$v[,1:p$categories])
+
+     # Put back correct names
+     rownames(ratings) <- rownames(data)
+     colnames(ratings) <- colnames(data)
+
+     # Only need to give back new users
+     ratings <- ratings[(dim(model$data@data)[1]+1):dim(ratings)[1],]
+
+     # Convert to right type
+     ratings <- new("realRatingMatrix", data=dropNA(ratings))
+     ## prediction done
+
+     ratings <- removeKnownRatings(ratings, newdata)
+
+     if(!is.null(model$normalize))
+        ratings <- denormalize(ratings)
+
+     if(type=="ratings") return(ratings)
+
+     getTopNLists(ratings, n=n, minRating=model$minRating)
+
+   }
+
+   ## construct recommender object
+   new("Recommender", method = "SVD", dataType = class(data),
+       ntrain = nrow(data), model = model, predict = predict)
+ }
> # Add it to registry
> recommenderRegistry$set_entry(
+   method="SVD", dataType = "realRatingMatrix", fun=REAL_SVD,
+   description="Recommender based on SVD approximation (real data).")
>
```

We had to take care of missing values for this implementation because the default SVD does as.matrix(). As shown earlier, this can be a problem with the object realRatingMatrix. So we have to convert it to a matrix properly. Now that we have added our method, let's run it and see what we get. We will continue with our previous example.

```
> algorithms <- list(
+   "random items" = list(name="RANDOM", param=list(normalize = "Z-score")),
+   "popular items" = list(name="POPULAR", param=list(normalize = "Z-score")),
+   "user-based CF" = list(name="UBCF", param=list(normalize = "Z-score",
+                                                  method="Cosine",
+                                                  nn=50, minRating=3)),
+   "item-based CF" = list(name="IBCF", param=list(normalize = "Z-score"
+                                                  )),
+   "SVD CF" = list(name="SVD", param=list(normalize = "Z-score",
+                                                  treat_na = "0"
+                                                  ))
+   )
```

```
> # run algorithms, predict next n movies
> results <- evaluate(scheme, algorithms, n=c(1, 3, 5, 10, 15, 20))

RANDOM run
        1 [0.002 sec/0.506 sec]
POPULAR run
        1 [0.051 sec/0.093 sec]
UBCF run
        1 [0.047 sec/1.703 sec]
IBCF run
        1 [55.831 sec/0.486 sec]
SVD run
        1 [0.048 sec/8.542 sec]
> # Draw ROC curve
> plot(results, annotate = 1:4, legend="topleft")
> # See precision / recall
> plot(results, "prec/rec", annotate=3)
```

Looks like it doesn't do quite as well as UBCF and the prediction time seems to be longer. Please note that there are other ways to approximate SVD as well that are more efficient. It still beats IBCF for this dataset (Figures 5.8 and 5.9).

Another take on this is PCA. If you have more items than users, you can reduce them to categories as before. But we will do it slightly differently. We can do PCA to decompose the original matrix into a matrix of its principal components.

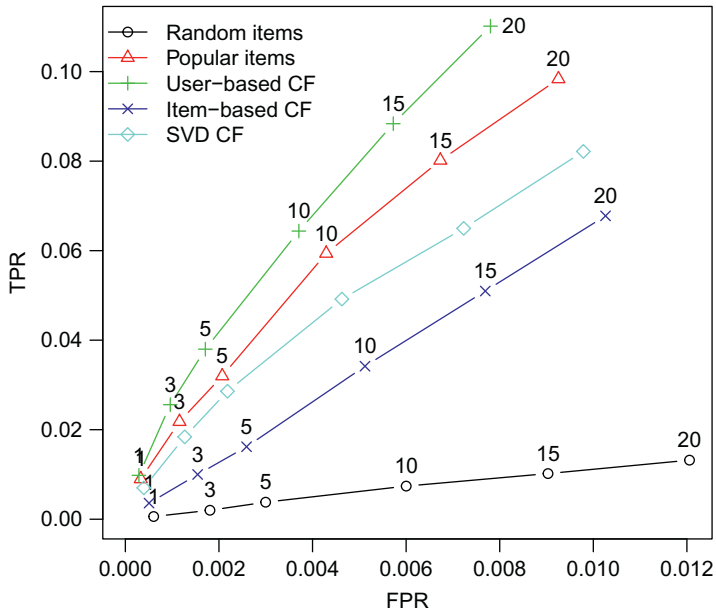


Figure 5.8
Performance of SVD CF against MovieLense.

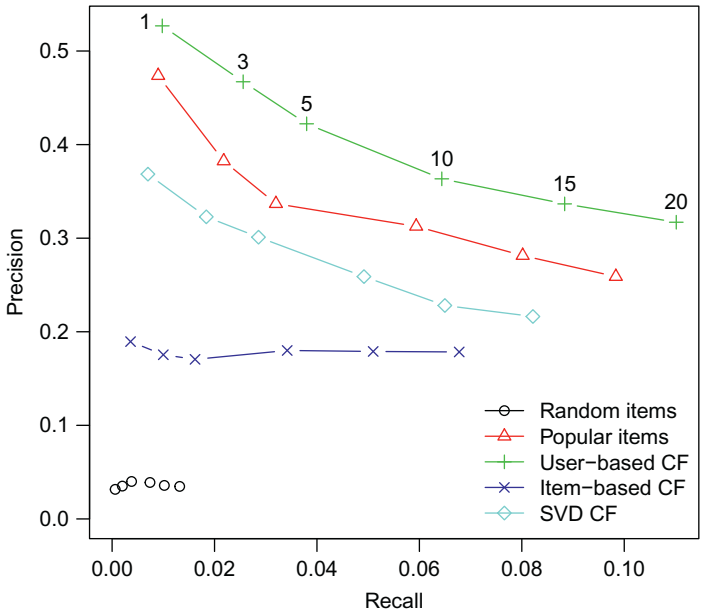


Figure 5.9
Performance of SVD CF against MovieLense.

$$R = W_1 x W_2 x W_3 \ldots x W_n$$

Multiplying these together will give us the same matrix back. You can think of each eigenvector representing a category by itself. Since the vectors are arranged from ones with most variability to the ones with least, you can take the first few vectors as a good indication of capturing most representative categories and approximate R. Again, the soft-clustering idea appears.

$$R_p = W_1 x W_2 x W_3 \ldots x W_n \, (where \, k < n)$$

For datasets with more users than items, this method is faster than regular SVD and works almost as well (Goldberg et al., 2001). Don't take my word for it, let's try it out and see how it works.

```
> REAL_PCA <- function(data, parameter= NULL) {
+
+   p <- .get_parameters(list(
+     categories = 20,
+     method="Cosine",
+     normalize = "center",
+     normalize_sim_matrix = FALSE,
+     alpha = 0.5,
+     na_as_zero = FALSE,
```

```
+     minRating = NA
+     ), parameter)
+
+
+   if(!is.null(p$normalize))
+     data <- normalize(data, method=p$normalize)
+
+   # Perform PCA
+   data <- data@data
+
+   # We will use princomp function, there are other methods available as well in R
+   # princomp does an as.matrix as well but it does not matter in this case
+   pcv<-princomp(data, cor=TRUE)
+
+   # Get the loadings
+   lpcv<-loadings(pcv)
+
+   # Total number of categories
+   cats <- min(dim(lpcv)[2], p$categories)
+
+   # det(lpcv[,1:99] %*% t(lpcv[, 1:99]))
+   # This is just a check. If this is close to 1 that means we did well.
+
+   # Convert to right type
+   itemcat <- new("realRatingMatrix",
+                              data = as(lpcv[,1:cats], "dgCMatrix"))
+
+   # Save the model
+   model <- c(list(
+     description = "PCA: Reduced item-category matrix",
+     itemcat = itemcat
+     ), p)
+
+   predict <- function(model, newdata, n = 10,
+                       type=c("topNList", "ratings"), ...) {
+
+     type <- match.arg(type)
+     n <- as.integer(n)
+
+     if(!is.null(model$normalize))
+        newdata <- normalize(newdata, method=model$normalize)
+
+     ## predict all ratings
+     u <- as(newdata, "dgCMatrix")
+     itemcat <- as(model$itemcat, "dgCMatrix")
+     ratings <- u %*% itemcat %*% t(itemcat)
+
+     ratings <- new("realRatingMatrix", data=dropNA(ratings),
+                    normalize = getNormalize(newdata))
+     ## prediction done
+
+     ratings <- removeKnownRatings(ratings, newdata)
```

```
+
+     if(!is.null(model$normalize))
+        ratings <- denormalize(ratings)
+
+     if(type=="ratings") return(ratings)
+
+     getTopNLists(ratings, n=n, minRating=model$minRating)
+
+   }
+
+   ## construct recommender object
+   new("Recommender", method = "PCA", dataType = class(data),
+       ntrain = nrow(data), model = model, predict = predict)
+ }
> # Add to registry
> recommenderRegistry$set_entry(
+   method="PCA", dataType = "realRatingMatrix", fun=REAL_PCA,
+   description="Recommender based on PCA approximation (real data).")
>
```

Now that we have implemented PCA as well, we can go ahead and compare it to what comes out of the box. We will have to use a different dataset since PCA won't work with more items than users. So we will try with Jester5k dataset. This has only 100 jokes but is evaluated by 5000 users. They are rated on a scale of −10 to 10 (Figure 5.10).

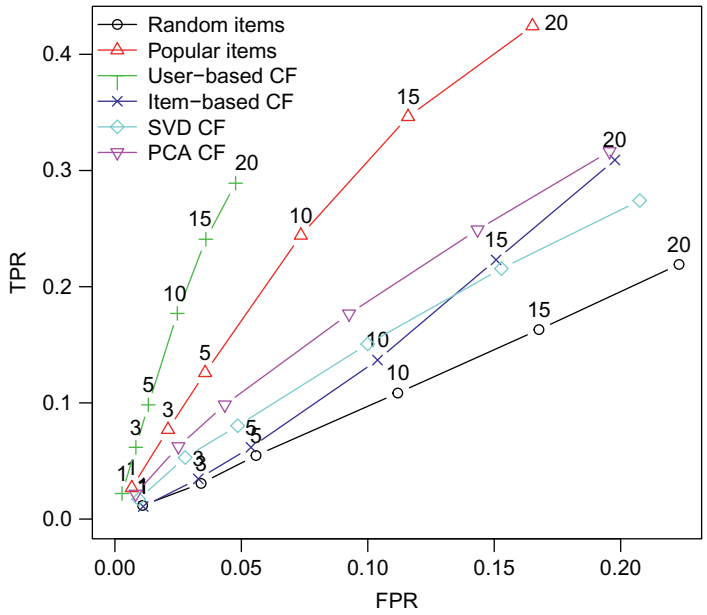


Figure 5.10
Performance of PCA CF against Jester5k.

```
> rm(MovieLense, scheme) # Clean up
> data(Jester5k) # Load another dataset
> scheme.jester <- evaluationScheme(Jester5k, method = "split", train = .9,
+                                   k = 1, given = 10, goodRating = 4)
> scheme.jester
Evaluation scheme with 10 items given
Method: âĂŸsplitâĂŹ with 1 run(s).
Training set proportion: 0.900
Good ratings: >=4.000000
Data set: 5000 × 100 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 362106 ratings.

> algorithms <- list(
+   "random items" = list(name="RANDOM", param=list(normalize = "Z-score")),
+   "popular items" = list(name="POPULAR", param=list(normalize = "Z-score")),
+   "user-based CF" = list(name="UBCF", param=list(normalize = "Z-score",
+                                                  method="Cosine",
+                                                  nn=50, minRating=3)),
+   "item-based CF" = list(name="IBCF", param=list(normalize = "Z-score"
+                                                  )),
+   "SVD CF" = list(name="SVD", param=list(normalize = "Z-score",
+                                          treat_na = "0"
+                                          )),
+   "PCA CF" = list(name="PCA", param=list(normalize = "Z-score"
+                                          ))
+   )
> # run algorithms, predict next n movies
> results <- evaluate(scheme.jester, algorithms, n=c(1, 3, 5, 10, 15, 20))


RANDOM run
         1 [0.005 sec/0.313 sec]
POPULAR run
         1 [0.222 sec/0.385 sec]
UBCF run
         1 [0.233 sec/4.909 sec]
IBCF run
         1 [0.57 sec/0.358 sec]
SVD run
         1 [0.227 sec/0.577 sec]
PCA run
         1 [0.377 sec/0.382 sec]
> # Draw ROC curve
> plot(results, annotate = 1:4, legend="topleft")
> # See precision / recall
> plot(results, "prec/rec", annotate=3)
```

As you can see, this implementation of PCA does better than SVD for this dataset and is quite fast. But if I had all the time (and memory) in the world, I would still pick UBCF. It gets closest to the intuition "Ask your friends." But as you can see from the timing, it takes a while to do so. SVD can be precomputed and results stored. That makes it very desirable. It doesn't have to be done online if the user is known from before. PCA can be done on partial information and

precomputed even earlier. It does require more items than users though (in this implementation). These two embody the intuition "What categories do you like, let me pick movies from those categories." And since the performance hit is small, these are good alternatives. In really large and sparse datasets, these might outperform UBCF. For Netflix prize, these methods were fundamental in getting a higher score (Figure 5.11).

Please note that there are many ways to decompose a matrix and many SVD implementations available in *R* (for example, irlba, model-based approximations, etc.) but I am showing here what can be done with a standard one. Feel free to switch out parts of this with better algorithms. But always test against the metric that is important to you.

Let's do one more with the same theme.

As before, imagine that for an *uxi* matrix, there are *k* categories. All items can be classified into *k* categories. For example, let's say all nontechnical books fall into four categories—romance, action, biography, misc. They can be in multiple categories with different rates. For example,

"The Godfather" = .9 action + 0 romance + .1 misc
"Lolita" = 0 action + .9 romance + .1 misc (bear with me here)
"Romeo and Juliet" = .1 action + .8 romance + .1 misc

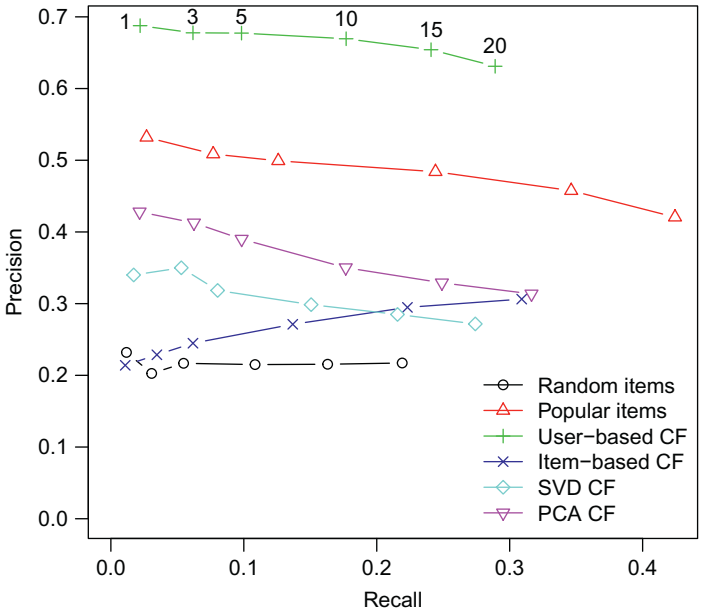


Figure 5.11
Performance of PCA CF against Jester5k.

Let's go further and say that we have gone ahead and put all our nontechnical books in these ratings manually. So our entire library is cataloged by these three categories and how much they fall in each of them.

So now if Jim comes and asks me for a book recommendation, I know he likes .8 action + .1 romance + .1 misc. So I will recommend him "The Godfather."

Let's say Adele comes and asks me for a recommendation. I don't know what she likes so I ask her for some books she has read. She tells me that she liked "Lolita." I see that must mean that she likes .8 romance + .2 misc. The closest I have to that is "Romeo and Juliet." So I will go ahead and recommend that. If I knew more books she liked, I could have a better idea of her taste breakdown, as I could simply average the categories of all the books she read.

So given a library of preclassified books, we can infer what the user's taste is. Let's say a User to category is an *uxk* matrix Θ. And Book to category is an *ixk* matrix X. To match books to users I can simply multiply these as before, $\Theta' X$.

There is one pesky detail. All books seem to have that misc category. All users will also have the same misc category. Since we don't know what it is, we will go ahead and use the intercept of 1 there, by convention. So the matrix X truly becomes:

"The Godfather" = .9 action + 0 romance + 1
"Lolita" = .0 action + .9 romance + 1
"Romeo and Juliet" = .1 action + .8 romance + 1

To measure cost, we want to see how far our prediction ($\Theta' X$) is from reality (Y):

$$J(\Theta) = \frac{1}{2}\sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(\Theta' X - Y_{i,j}\right)^2$$

We only do this for the items rated by the user as denoted by $i{:}r(i, j) = 1$. We would also have to regularize it to remove bias. Say our regularizing parameter is λ. We now have:

$$J(\Theta) = \frac{1}{2}\sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(\Theta' X - Y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n} \left(\Theta^2\right)$$

We cannot apply the regularization parameter when $k = 1$ because we have an intercept. We will have to treat that case differently.
And taking the derivative of that with respect to Theta, our gradient is

$$\Theta_k^{(j)} = \sum_{i:r(i,j)=1} \left(\left(\Theta^{(j)}\right)' X^{(i)} - Y^{(i,j)}\right) X_k^{(i)} + \lambda\left(\Theta_k^{(j)}\right)$$

(except for $k = 0$, our intercept, where there won't be a regularization term).

This is called content based filtering. It's the class of algorithms that music services like Pandora typically use, thanks to an already classified music library—Music Genome Project.

But we could think of this problem the opposite way. If we knew the taste vectors of all users (*uxk* matrix X), then we could figure out what the Θ for each book is going the opposite direction. For example, if both Adele and Mike like "Lolita," I assume that the categories for "Lolita" are the average of the taste of those two users. Mathematically, the cost function now becomes

$$J\left(X^1 \ldots X^{n_m}\right) = \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j:r(i,j)=1}\left(\Theta' X^{(i)} - Y_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(X^{(i)}\right)^2$$

So we could start with small initial guesses for both X and Θ. Then use X to estimate Θ, then Θ to estimate X, and so on. That way we will converge on appropriate values for X and Θ.

It turns out there is a way to combine both these equations. The final cost function is

$$J\left(X^1 \ldots X^{n_m}, \Theta^1 \ldots \Theta^{n_u}\right) = \frac{1}{2}\sum_{(i,j):r(i,j)=1}\left(\left(\Theta^{(j)}\right)' X^{(i)} - \left(Y^{(i,j)}\right)\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}\left(X^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\Theta^2\right)$$

And the gradients are:

$$\frac{d}{dX_k^{(i)}} = \sum_{i:r(i,j)=1}\left(\left(\Theta^{(j)}\right)' X^{(i)} - Y^{(i,j)}\right)\Theta_k^{(j)} + \lambda X_k^{(i)}$$

$$\frac{d}{d\Theta_k^{(j)}} = \sum_{j:r(i,j)=1}\left(\left(\Theta^{(j)}\right)' X^{(i)} - Y^{(i,j)}\right)X_k^{(i)} + \lambda\Theta_k^{(j)}$$

We no longer have a special case for the intercept term because we are feature learning, and so it is superfluous.

Let's go ahead and implement this low rank matrix factorization using stochastic gradient descent (Funk, 2006; Koren et al., 2009; Koren, 2008).

```
> REAL_LRMF <- function(data, parameter= NULL) {
+
+   p <- .get_parameters(list(
+     categories = min(100, round(dim(data@data)[2]/2)),
+     method="Cosine",
+     normalize = "Z-score",
+     normalize_sim_matrix = FALSE,
+     minRating = NA,
+     lambda = 1.5, # regularization
```

```
+     maxit = 2000 # Number of iterations for optim
+   ), parameter)
+
+   if(!is.null(p$normalize))
+     data <- normalize(data, method=p$normalize)
+
+   model <- c(list(
+     description = "full matrix",
+     data = data
+   ), p)
+
+   predict <- function(model, newdata, n = 10,
+                       type=c("topNList", "ratings"), ...) {
+
+     type <- match.arg(type)
+     n <- as.integer(n)
+
+     if(!is.null(model$normalize))
+        newdata <- normalize(newdata, method=model$normalize)
+
+     # Get new data, make one Matrix object
+     data <- model$data@data
+     data <- rBind(data, newdata@data)
+
+     Y <- t(data)
+
+     # initialization
+     # Users
+     theta <- Matrix(runif(p$categories * dim(Y)[2]), ncol = p$categories)
+     # Items
+     X <- Matrix(runif(dim(Y)[1] * p$categories), ncol = p$categories)
+
+     # We are going to scale the data so that optim converges quickly
+     scale.fctr <- max(abs(Y@x))
+     Y@x <- Y@x / scale.fctr
+
+     # Let's optimize
+     system.time(
+       res <- optim(c(as.vector(X), as.vector(theta)),
+                    fn = J_cost_full, gr = grad,
+                    Y=Y, lambda = model$lambda,
+                    num_users = dim(theta)[1], num_books = dim(X)[1],
+                    num_cats = model$categories,
+                    method = "CG", # Slow method, faster methods available
+                    control = list(maxit=model$maxit, factr = 1e-2)
+                    )
+     )
+
+     print(paste("final cost: ", res$value, " convergence: ", res$convergence,
+                 res$message, " counts: ", res$counts))
+
```

```
+     X_final <- unroll(res$par, num_users = dim(theta)[1],
+                         num_books = dim(X)[1], num_cats = p$categories)[[1]]
+     theta_final <- unroll(res$par, num_users = dim(theta)[1],
+                             num_books = dim(X) [1], num_cats = p$categories) [[2]]
+
+     Y_final <- (X_final %*% t(theta_final) )
+     Y_final <- Y_final * scale.fctr
+     dimnames(Y_final) = dimnames(Y)
+
+     ratings <- t(Y_final)
+
+     # Only need to give back new users
+     ratings <- ratings[(dim(model$data@data)[1]+1):dim(ratings)[1],]
+
+     ratings <- new("realRatingMatrix", data=dropNA(as.matrix(ratings)))
+     ## prediction done
+
+     ratings <- removeKnownRatings(ratings, newdata)
+
+     if(!is.null(model$normalize))
+       ratings <- denormalize(ratings)
+
+     if(type=="ratings") return(ratings)
+
+     getTopNLists(ratings, n=n, minRating=model$minRating)
+
+   }
+
+   ## construct recommender object
+   new("Recommender", method = "LRMF", dataType = class(data),
+       ntrain = nrow(data), model = model, predict = predict)
+ }
> # Helper functions
> unroll <- function (Vec, num_users, num_books, num_cats) {
+   # Unroll the vector
+   endIdx <- num_books * num_cats
+   X <- Matrix(Vec[1:endIdx], nrow = num_books)
+   theta <- Matrix(Vec[(endIdx + 1): (endIdx + (num_users * num_cats))],
+                   nrow = num_users)
+
+   return (list(X, theta))
+ }
> J_cost_full <- function (Vec, Y, lambda, num_users, num_books, num_cats) {
+   # Calculate the cost
+   # Unroll the vector
+   Vec.unrolled <- unroll(Vec, num_users, num_books, num_cats)
+   X <- Vec.unrolled[[1]]
+   theta <- Vec.unrolled[[2]]
+
+   R          <- as(Y, "nsparseMatrix") * 1 # Creates binary matrix
```

```
+   Y_dash          <- (X %*% t(theta) ) * R #(Y!=0)
+   J_cost          <- .5 * (sum ((Y_dash - Y)^2)
+                            + lambda/2 * sum(X^2)
+                            + lambda/2 * sum (theta^2) )
+
+   return (J_cost)
+ }
> grad <- function (Vec, Y, lambda, num_users, num_books, num_cats) {
+   # Unroll the vector
+   Vec.unrolled <- unroll(Vec, num_users, num_books, num_cats)
+   X <- Vec.unrolled[[1]]
+   theta <- Vec.unrolled[[2]]
+
+   # Calculate gradients
+   R          <- as(Y, "nsparseMatrix") * 1 # Creates binary matrix
+   Y_dash     <- (X %*% t(theta) ) * R #(Y!=0)
+   X_gr       <- (    (Y_dash - Y) * R ) %*% theta + lambda * X
+   theta_grad <- (  t((Y_dash - Y) * R ) %*% X     + lambda * theta)
+
+   return (c(as.vector(X_gr), as.vector(theta_grad)))
+ }
> recommenderRegistry$set_entry(
+   method="LRMF", dataType = "realRatingMatrix", fun=REAL_LRMF,
+   description="Recommender based on Low Rank Matrix Factorization (real data).")
>
```

Please note that I am using optim with gradient descent because it is standard but there are many very fast optimizers available in R (e.g., L-BFGS-B). Feel free to try them.

We have to be careful though, if optim does not converge our estimates won't be very good. We can scale our Matrix to small numbers before we call optim to take care of that. Let's test it out (Figure 5.12).

```
> algorithms <- list(
+   "random items" = list(name="RANDOM", param=list(normalize = "Z-score")),
+   "popular items" = list(name="POPULAR", param=list(normalize = "Z-score")),
+   "user-based CF" = list(name="UBCF", param=list(normalize = "Z-score",
+                                                  method="Cosine",
+                                                  nn=50, minRating=3)),
+   "item-based CF" = list(name="IBCF", param=list(normalize = "Z-score"
+                                                  )),
+   "SVD CF" = list(name="SVD", param=list(normalize = "Z-score",
+                                          treat_na = "0"
+                                          )),
+   "PCA CF" = list(name="PCA", param=list(normalize = "Z-score"
+                                          )),
+   "LRMF" = list(name="LRMF", param=list(normalize = "Z-score",
+                                         maxit = 5000
+                                          ))
+   )
> # run algorithms, predict next n movies
> results <- evaluate(scheme.jester, algorithms, n=c(1, 3, 5, 10))
```

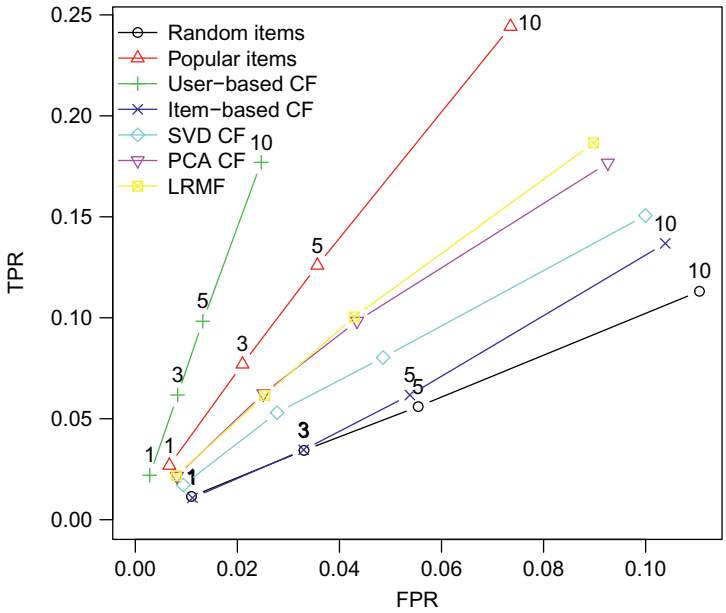


Figure 5.12
Performance of LRMF, 50 categories, 1.5 lambda against Jester5k.

```
RANDOM run
          1 [0.006 sec/0.311 sec]
POPULAR run
          1 [0.219 sec/0.383 sec]
UBCF run
          1 [0.227 sec/4.961 sec]
IBCF run
          1 [0.569 sec/0.358 sec]
SVD run
          1 [0.225 sec/0.578 sec]
PCA run
          1 [0.359 sec/0.363 sec]
LRMF run
          1 [1] "final cost: 774.063103603801 convergence: 0 counts: 967"
[2] "final cost: 774.063103603801 convergence: 0 counts: 373"
[0.227 sec/272.949 sec]
> # Draw ROC curve
> plot(results, annotate = 1:4, legend="topleft")
```

This method is especially powerful because we end up with X and Θ, which are reduced matrices of users and items in k categories. This method also takes care of worrying about NAs, as we only use the items that were rated. SVD and PCA also give roughly the same matrices, some would say the same. For example X can be thought of as the $U_k.\sqrt{S_k}$ and Θ as $\sqrt{S_k}V_k'$. You can go ahead and do user based collaborative filtering or item based with these reduced

matrices. You can read more about user-based and item-based collaborative filtering in the vignette for recommenderlab. For any of these, we don't have to know what the categories are ahead of time, the model figures that out. Finally, for LRMF we can do two passes. We can create the model and figure out X in the first pass. Then use that to figure out Θ_u for a given user. This means that we can do online modeling. I will leave the implementation as an exercise for you, dear reader. Watch out for the intercept term. A similar way to incrementally update SVD is also available, see (Sarwar et al., 2002). More advanced matrix decomposition methods have also been tried out, see (Abernethy et al., 2006).

5.6 Simplified Approach

If you don't have to predict how the users will rate the item, sometimes a simplified approach will do. In this case, you don't have to rely on a user-rating matrix but rather a binary feedback. We might only know whether the user picked an item or not. We don't have to predict what the rank given to each movie was. We can test a simple version of this by binarizing our data. In addition to UBCF and IBCF, we will also try association rules that are already available in the package. Association rules find items that happen to show up together with a high level of confidence. They are a recursive algorithm and take a long time to compute all the rules. Nevertheless, we can try it for free here (Figures 5.13 and 5.14).

```
> rm(Jester5k, scheme.jester) # Clean up
> data(MovieLense) # Load data
> # Binarize
> MovieLense.bin <- binarize(MovieLense, minRating = 3)
> # What is available to us?
> recommenderRegistry$get_entries(dataType = "binaryRatingMatrix")

$AR_binaryRatingMatrix
Recommender method: AR
Description: Recommender based on association rules.

$IBCF_binaryRatingMatrix
Recommender method: IBCF
Description: Recommender based on item-based collaborative filtering (binary rating data).

$POPULAR_binaryRatingMatrix
Recommender method: POPULAR
Description: Recommender based on item popularity (binary data).

$RANDOM_binaryRatingMatrix
Recommender method: RANDOM
Description: Produce random recommendations (binary ratings).

$UBCF_binaryRatingMatrix
Recommender method: UBCF
Description: Recommender based on user-based collaborative filtering (binary data).

> # We have a few options
>
```

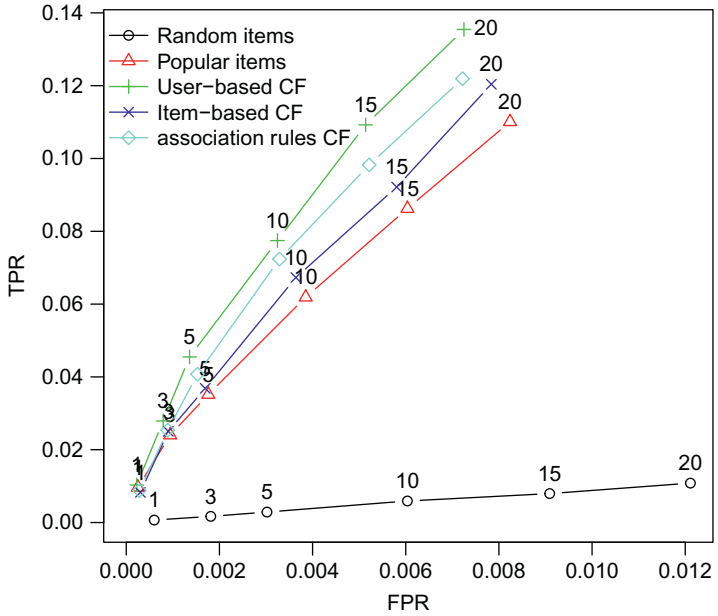


Figure 5.13
Performance of Association Rules CF against binarized Jester5k.

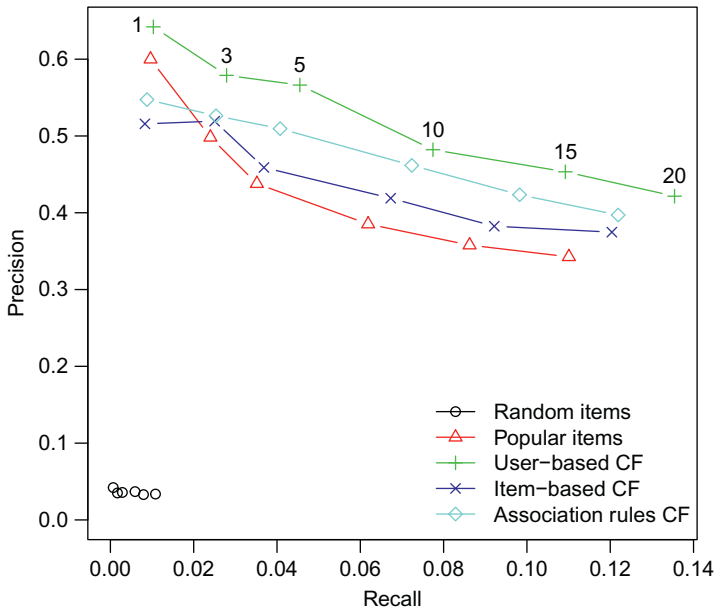


Figure 5.14
Performance of Association Rules CF against binarized Jester5k.

```
> # Let's check some algorithms against each other
> scheme.bin <- evaluationScheme(MovieLense.bin,
+                                method = "split", train = .9,
+                                k = 1, given = 6) # Had to decrease given
> scheme.bin
Evaluation scheme with 6 items given
Method: âĂŸsplitâĂŹ with 1 run(s).
Training set proportion: 0.900
Good ratings: >=NA
Data set: 943 × 1664 rating matrix of class âĂŸbinaryRatingMatrixâĂŹ with 82026 ratings.

> algorithms <- list(
+   "random items" = list(name="RANDOM", param=NULL),
+   "popular items" = list(name="POPULAR", param=NULL),
+   "user-based CF" = list(name="UBCF", param=NULL),
+   "item-based CF" = list(name="IBCF", param=NULL),
+   "association rules CF" = list(name="AR", param=NULL)
+ )
> # run algorithms, predict next n movies
> results.bin <- evaluate(scheme.bin, algorithms, n=c(1, 3, 5, 10, 15, 20))

RANDOM run
         1 [0.006 sec/0.545 sec]
POPULAR run
         1 [0.004 sec/0.584 sec]
UBCF run
         1 [0.002 sec/2.248 sec]
IBCF run
         1 [36.234 sec/0.608 sec]
AR run
         1 [0.095 sec/2.681 sec]
> # Draw ROC curve
> plot(results.bin, annotate = 1:4, legend="topleft")
> # See precision / recall
> plot(results.bin, "prec/rec", annotate=3)
```

The major lesson here is that even without rating data, we got a high AUC with just what we have. In a production system, a combination approach is often used—Ratings (if they are available), How long did the user watch the movie? Did she click on other movies as well? We can use information such as visitor's browser history, temporal data or other implicit feedback.

5.7 Roll Your Own

In the examples above, you have learned how to roll your own algorithms. You can implement even more complicated algorithms and test out your ideas against some real data. You don't have to bet that one method will work better over other, you can actually measure it. Very little work is required to convert your data into the right format (Figure 5.15).

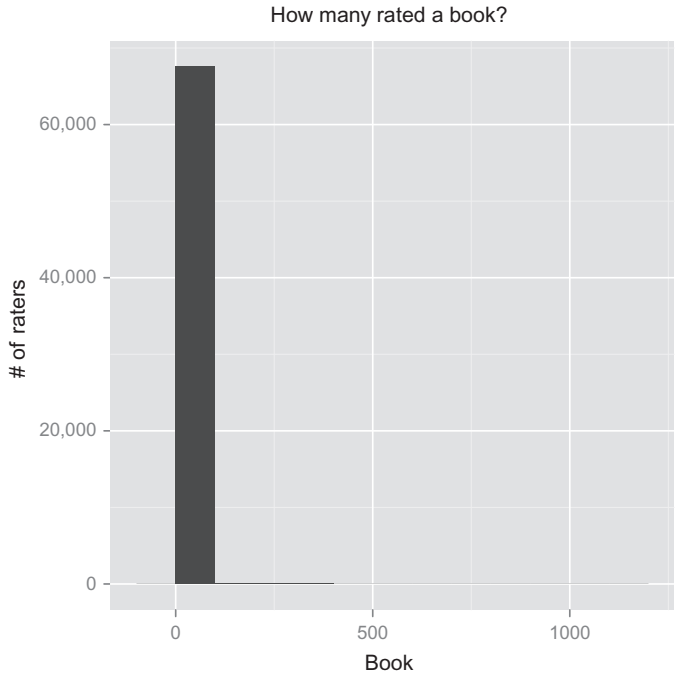


Figure 5.15
Distribution of Book Readers.

Let's take a freely available dataset from (Ziegler et al., 2005) and run with that. This dataset has books rated by users.

```
> ########################
> # Data Aquisition
> ########################
>
> # Name of download file
> temp <- tempfile(fileext = ".zip")
> # Get the file from http://www.informatik.uni-freiburg.de/~cziegler/BX/
> download.file("http://www.informatik.uni-freiburg.de/~cziegler/BX/BX-CSV-Dump.
zip",                          temp)
> # Read in bookratings
> bookratings <- read.csv(unz(temp, "BX-Book-Ratings.csv"),
+                         header=FALSE, sep = ';',
+                         stringsAsFactors = FALSE, skip = 1,
+                         col.names = c("User.ID", "ISBN", "Book.Rating"))
> # Not used here, provided to peak your curiosity
> # users <- read.csv(unz(temp, "BX-Users.csv"),
> #                   header=FALSE, sep = ';',
> #                   stringsAsFactors = FALSE, skip = 1,
> #                   col.nam = c("User-ID", "Location", "Age"))
>
> # Are there any duplicates?
```

```
> bookratings[duplicated(bookratings)]

data frame with 0 columns and 493813 rows
> # No duplicate ratings
>
> # Read the book names
> books <- read.csv(unz(temp, "BX-Books.csv"),
+                   header=FALSE, sep = ';',
+                   stringsAsFactors = FALSE, skip = 1,
+                   col.names = c("ISBN", "Book-Title", "Book-Author",
+                                 "Year-Of-Publication", "Publisher",
+                                 "Image-URL-S", "Image-URL-M", "Image-URL-L"))
> ########################
> # Data Wrangling
> ########################
>
> # Merge the two datasets
> bookratings.dtl <- merge(bookratings, books, on = ISBN)
> bookratings.dtl$Book.detail <- with(bookratings.dtl, paste(ISBN, Book.Title, Book.
Author,
> # We only need these fields
> bookratings.dtl <- bookratings.dtl[, c("User.ID", "Book.detail", "Book.Rating")]
> # Convert it to a realRatingMatrix
> (bookratings.r <- as(bookratings.dtl, "realRatingMatrix"))

26137 × 67665 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 199149 ratings.

>
> # Look at the distribution
>
> # Books were rated by how many users?
> qplot(as.vector(colCounts(bookratings.r)),
+       binwidth=100,
+       main = "How many rated a book?",
+       xlab = "Book",
+       ylab = "# of raters")
>
```

This has a very long tail. For brevity, we will go ahead and pick the top few.

```
> # Taking the 1000 most rated books (approx.)
> colIdx <- colCounts(bookratings.r)
> # Seeing the cutoff value here
> sort(colIdx, decreasing = TRUE)[1000]

0553106341::Dust to Dust::Tami Hoag
                                  24

> # using cutoff threshold
> bookratings.r@data <- bookratings.r@data[, which(colIdx >= 24)]
> bookratings.r

26137 × 1016 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 58978 ratings.

> # Let's also cut down on number of users
```

```
> # We are going to evaluate on users by giving the model 5 things they have rated
> # And ask to predict the next 5
> # So we need to have atleast 10 ratings per user
> summary(rowCounts(bookratings.r))

   Min.    1st Qu.   Median    Mean    3rd Qu.   Max.
   0.000   0.000     1.000     2.256   1.000     258.000
> rowIdx <- rowCounts(bookratings.r)
> qplot(rowIdx, binwidth = 10,
+       main = "Books Rated on average",
+       xlab = "# of users",
+       ylab = "# of books rated")
>
> # If 5 are given and 5 are predicted, need to remove <10
> length(id2remove <- which(rowIdx < 10))

[1] 25074

> bookratings.r <- bookratings.r[-1*id2remove]
> # Final realRatingMatrix
> bookratings.r

1063 × 1016 rating matrix of class âĂŸrealRatingMatrixâĂŹ with 34104 ratings.

> ########################
> # Model Evaluation
> ########################
>
> scheme <- evaluationScheme(bookratings.r, method="cross-validation", goodRating=5,
+                            k=2, given=10)
> algorithms <- list(
+   "random items" = list(name="RANDOM", param=NULL),
+   "popular items" = list(name="POPULAR", param=NULL),
+   "user-based CF" = list(name="UBCF", param=list(method="Cosine",
+                                                  nn=10, minRating=1)),
+   "Item-based CF" = list(name = "IBCF", param = list(normalize="Z-score")),
+   "LRMF (100 categories)" = list(name = "LRMF", param = list(categories=100,
+                                                              normalize="Z-score"))
+ )
> results <- evaluate(scheme, algorithms, n=c(1, 3, 5, 10))

RANDOM run
         1 [0.002 sec/2.454 sec]
         2 [0.002 sec/2.61 sec]
POPULAR run
         1 [0.008 sec/0.598 sec]
         2 [0.008 sec/0.587 sec]
UBCF run
         1 [0.005 sec/6.707 sec]
         2 [0.005 sec/6.882 sec]
IBCF run
         1 [10.043 sec/2.422 sec]
         2 [9.645 sec/2.587 sec]
```

```
LRMF run
         1 [1] "final cost: 103.409093016859 convergence: 0 counts: 137"
[2] "final cost: 103.409093016859 convergence: 0 counts: 39"
[0.022 sec/48.616 sec]
         2 [1] "final cost: 119.560837692644 convergence: 0 counts: 1469"
[2] "final cost: 119.560837692644 convergence: 0 counts: 442"
[0.021 sec/479.138 sec]
>

> plot (results, annotate=c(1,3), legend="topleft")
```

Very quickly, we were able to run our algorithms on this new dataset (Figures 5.16 and 5.17).

5.8 Final Thoughts

When I introduced metrics at the beginning, I said they are roughly analogous. But that is not the full story. Are you showing your recommendations ranked or are they jumbled? How many do you show? Do you have other business rules (certain items HAVE to be shown every × times, like at a dating site)? Do you have limited inventory (recommending a physical product)? The application of the recommender system drives the evaluation method. The right

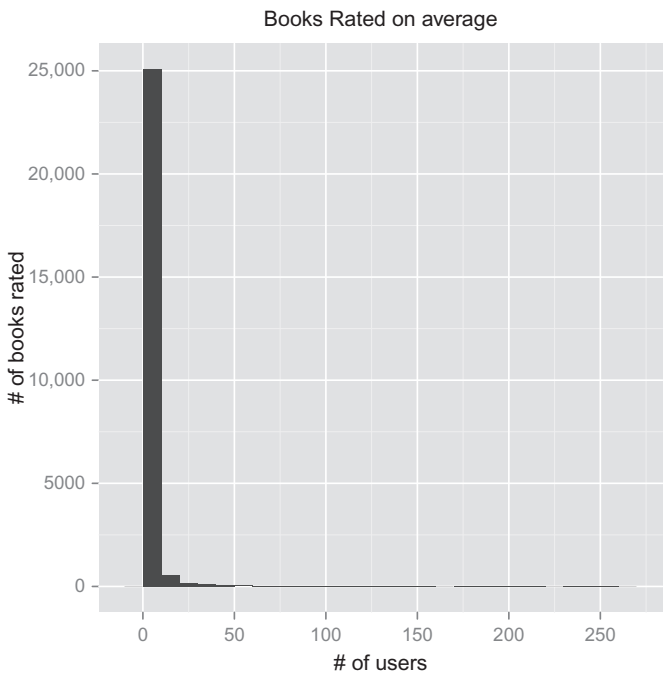


Figure 5.16
Books rated on average.

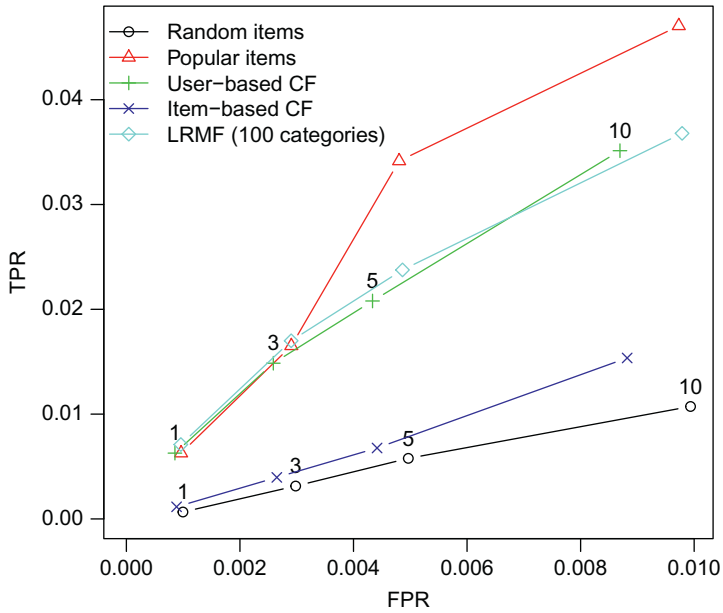


Figure 5.17
Performance of multiple CF algorithms against Book Ratings data.

metric should reflect what you want to measure. When you have that down you can use this framework to narrow down to the best answer quickly.

There are other concerns besides ROC curves. Speed of calculation could be important. UBCF gives the highest AUC for MovieLense but if speed is important, I might choose PCA over that for that dataset. If I know that I will not go over 20 items recommended to a single user, I might go with IBCF (fastest at predict time). But IBCF gives rather predictable recommendations. If serendipity is important and I want users to be surprised by the recommendations, I might do things differently. In real big data systems I might not have the luxury to get the SVD of a large sparse matrix, so I might go with model based methods or lower dimension representations. Maybe a combination is more suitable.

Then there are other issues you will hear about in the community. If we get a new user into the fold about whom we know nothing about, it would be hard to recommend the right items for him right away. This is called the cold start problem. One solution could be showing the most popular items to start with and switch to another method after you have enough history. In practice it is possible to get some data even if the user has not made any ratings, for example, geolocation, time spent, time of day, weekday, etc. This is called implicit data collection and is a fine approach.

Another dilemma is "black sheep" problem. These are users who are so unique that no other user in the system seems to follow a similar pattern. But this is an issue for real life recommenders as well, so it is OK to give reasonable recommendations to them even if they are not at par. And over time, hopefully you have enough of those kinds of users to form their own cluster/community. Then one of these methods will actually work for them. Yet another issue is how to build the user's trust in the recommendation. One solution is to show why you are showing those recommendations, for example, for IBCF (because the users who liked item *A* and item *B* liked item *C*), or for UBCF (found users *U* and *V* just like you who like item *C*).

You now know how to test out of the box algorithms, roll your own, use your own data, and questions to ask yourself before deploying this in a real-world system. I hope this has given you a practical introduction to recommender systems and gotten you excited about testing your theories in R. Good luck and happy coding.

References

Abernethy, J., Bach, F., Evgeniou, T., Vert, J.-P. 2006. Low-rank matrix factorization with attributes. *CoRR*, abs/cs/0611124.

Bates, D., Maechler, M. 2012. Matrix: Sparse and Dense Matrix Classes and Methods. http://Matrix.R-forge.R-project.org/. R package version 1.0-6.

Funk, S., 2006. Netflix update: try this at home. http://sifter.org/simon/journal/20061211.html.

Goldberg, K., Roeder, T., Gupta, D., Perkins, C., 2001. Eigentaste: a constant time collaborative filtering algorithm. Information Retrieval. 1386-45644, 133–151. http://dx.doi.org/10.1023/A:1011419012209.

Hahsler, M., 2011. recommenderlab: lab for developing and testing recommender algorithms. http://CRAN.R-project.org/package=recommenderlab. R package version 0.1-3.

Koren, Y., 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08. ACM, New York, NY, USA, pp. 426–434. http://dx.doi.org/10.1145/1401890.1401944. http://doi.acm.org/10.1145/1401890.1401944.

Koren, Y., Bell, R., Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer. 0018-916242 (8), 30–37. http://dx.doi.org/10.1109/MC.2009.263. http://dx.doi.org/10.1109/MC.2009.263.

Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T., 2000. Application of dimensionality reduction in recommender system—a case study. In: IN ACM WEBKDD WORKSHOP.

Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Fifth International Conference on Computer and Information Sciencepp. 27–28.

Wickham, H., 2009. ggplot2: Elegant Graphics for Data Analysis. Springer, New York.http://had.co.nz/ggplot2/book.

Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G., 2005. Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web, WWW '05. ACM, New York, NY, USAISBN 1-59593-046-9, pp. 22–32. http://dx.doi.org/10.1145/1060745.1060754.http://doi.acm.org/10.1145/1060745.1060754.

CHAPTER 6

Response Modeling in Direct Marketing: A Data Mining-Based Approach for Target Selection

Sadaf Hossein Javaheri, Mohammad Mehdi Sepehri, Babak Teimourpour
Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran

6.1 Introduction/Background

Traditional large-scale sales pattern is the most familiar sales pattern for companies. On the basis of this pattern, companies usually give all the customers the same sales promotion. However, this kind of sales promotions neglects the differences among customers. In most cases, these promotions cost a lot, but only get few real profits from customers. That means many promotions are wasted. A new business culture is developing today. Within it, the economics of customer relationships are changing in fundamental ways, and companies are facing the need to implement new solutions and strategies that address these changes. The concepts of mass production and mass marketing, first created during the Industrial Revolution, are being supplanted by new ideas in which customer relationships are the central business issue. The traditional process of mass marketing is being challenged by the new approach of one-to-one marketing. In the traditional process, the marketing goal is to reach more customers and expand the customer base. But given the high cost of acquiring new customers, it makes better sense to conduct business with current customers. In doing so, the marketing focus shifts away from the breadth of customer base to the depth of each customer's needs.

Businesses do not just deal with customers to conduct transactions; they turn the opportunity to sell products into a service experience and endeavor to establish a long-term relationship with each customer (Rygielski et al., 2002). Actually in direct marketing, companies or organizations try to establish and maintain a direct relationship with their customers to target them individually for specific product offers or fund raising. Nowadays this type of marketing is being used by a growing number of companies, especially financial services, banks, and insurance companies as their main strategy for interacting with their customers. Recently, expenditure on direct marketing has increased dramatically, e.g., 73.6% in the UK between 2001 and 2004. Furthermore, U.S. expenditure was as high as

Data Mining Applications with R

$161.3 billion in 2005 accounting for 10.3% of its total GDP (Bose and Chen, 2009; Ou et al., 2003). But direct marketers in a wide range of industries from banking and financial services to consumer electronics to computers to office supplies to consumer retail are faced with the challenge of continually rising printing and postage costs, as well as decreasing response rates. To combat rising costs and declining response rates, direct marketers are advised to shift from intuitive selection of their audience or the profiling method to more scientific approaches such as predictive modeling and analyzing the customers' data (demographic and historical purchase data) and selecting those customers who are most likely to respond to a promotion. Identifying customers who are more likely to respond to a product offering is an important issue in direct marketing (Deichmann et al., 2002). Data mining can solve this problem. Nowadays, a huge amount of information on customers is kept in databases. Thus data mining can be very effective for direct marketing (Ling and Li, 1998).

Data mining can be used to improve targeting by selecting the people to contact. Data mining is the process of exploration and analysis of large quantities of data to discover meaningful patterns and rules. Simply stated, data mining refers to extracting or "mining" knowledge from large amounts of data. Because of the wide availability of large amounts of data and the imminent need for turning such data into useful information and knowledge, recently data mining has attracted a great deal of attention in the information industry (Han and Kamber, 2006). Direct marketing has become an important application field for data mining. Identifying customers for marketing purposes is one of the most common applications of data mining. Actually target selection is an important data mining problem from the world of direct marketing. It aims at identifying customers who are most likely to respond positively to a new product. Large databases of customer and market data are maintained for this purpose. The customers or clients to be targeted in a specific campaign are selected from the database given different types of information such as demographic information and information on the customers' personal characteristics like profession, age, and purchase history. The main task is to determine the potential customers for a new product from a client database by identifying profiles of customers who are known to have shown interest in a product in the past, that is, the generation of customer models for a given product by analyzing customers' data obtained from similar previous marketing campaigns.

Response model is a well known technique commonly used by direct marketing analysts and it is a profitable tool in fine-tuning direct marketing strategies (Potharst et al., 2002). A response model predicts the probability that a customer is going to respond to a promotion or offer. Response models are typically built from historical purchase data. Using the model, one can identify a subset of customers who are more likely to respond than others. Actually the purpose of these models is the selection of those customers who will be most interested in a particular product offer, so that as large a percentage as possible of the targeted customers responds to the product offer. A more accurate response model will have more respondents and fewer

nonrespondents in the subset. By doing so, one can significantly reduce the overall marketing cost without sacrificing opportunities (Shin and Cho, 2006).

Response modeling is usually formulated as a binary classification problem. The customers are divided into two classes, respondents and nonrespondents. Various classification methods have been used for response modeling such as statistical and machine learning methods, for example, neural networks (Bentz and Merunkay, 2000; Bounds and Ross, 1997; Ha et al., 2005; Kim and Street, 2004; Moutinho et al., 1994; Zahavi and Levin, 1997a), decision trees (Haughton and Oulabi, 1997), and support vector machines (SVMs) (Shin and Cho, 2006; Yu and Cho, 2006). The SVM has drawn much attention and a few researchers have implemented them for response modeling. SVMs are attracting increasing attention because they rely on a solid statistical foundation and appear to perform quite effectively in many different applications (Lecun et al., 1995; Osuna et al., 1997).

Response modeling procedure consists of several steps such as data collection, data preprocessing, feature construction, feature selection, class balancing, classification, and evaluation. Various data mining techniques and algorithms have been applied for implementing each step. According to a review of the literature on response modeling very few articles deal with all these steps. Most of them focus only on two or three steps of modeling and the rest use the results of previous works. Response models can be very beneficial for companies. By using this model, companies can identify a subset of customers who are more likely to respond than others. As a result they can maximize the profits in selling the product and minimize the cost of the marketing campaign. Actually they can improve return on investment, customer relationships, and retention (Shin and Cho, 2006).

6.2 Business Problem

In general, financial services and banks in Iran use mass marketing as their strategy for offering and promoting a new product or service to their customers. In this strategy, a single communication message is broadcast to all customers through media such as print, radio, or television. In this approach companies do not establish a direct relationship with their customers for offering new products. However, this kind of sales promotions neglects the differences among customers. Such an approach is always a waste; only a small proportion of the customers will actually buy the product. In today's world where products are overwhelming and the market is highly competitive, mass marketing has become less effective. The response rate and the percentage of people who actually buy the products after seeing the promotion, is often low. So in this competitive environment, there is a need for direct marketing, which is an effective and efficient way of communicating with customers. As a result, many of banks and financial services in Iran are trying to move away from traditional aggregate-level mass marketing programs, using direct marketing as their main strategy for interacting with their customers. Direct marketing is done by sending product offers and

information directly to the customers. This can be done by sending e-mails or SMS to the customers, or by making phone calls, or by addressing the customers by post. By doing this, companies are faced with two important issues: the costs of addressing all the customers and the customers' annoyance with undesirable mail, SMS, or phone calls. In the first issue, it is imperative to reduce costs because the costs of such a full-scale mailing campaign can soon become too large and rise above the expected returns, because the product may only be interesting to a subset of the total customer base, whereas in the second issue it is important to have in mind that the customers' annoyance may lead to loss of market share because sending many uninteresting product offers to customers leads to irritation as such mail is often considered "junk" (Setnes and Kaymak, 2001). So it is important to select those who are more likely to be potential buyers of the new product or service. To address these problems, companies are searching for accurate methods to identify their most promising customers to focus specifically on those individuals.

For this study, one Iranian private bank was selected. This company offers different products and services to its customers and uses mass marketing as its strategy for offering and promoting a new product or service to its customers. This bank is faced with the challenges of increasing competition, continually rising marketing costs, decreasing response rates, and also a lack of direct relationship with its customers. As the market is highly competitive and products are overwhelming, this bank, to retain its customers, tried to move away from the traditional aggregate-level mass marketing programs and use direct marketing as their strategy for interacting with its customers. Because of two important issues in direct marketing—high marketing cost and customer annoyance—they have to select a subset of their customers. To combat these problems, the company is searching for an accurate method to identify the potential customers for a new product offering to focus specifically on those individuals. The goal of the bank is to obtain as high a percentage as possible of positive responses and minimize the marketing cost.

6.3 Proposed Response Model

The goal of this model is to predict whether an existing customer will purchase in the next marketing campaign by using information provided by the purchase behavior variables. We try to predict, using independent variables (RFM variables and demographic), whether a customer will respond to the next marketing campaign.

The response model is a classification model. The task is to classify the customers who will respond to the next marketing campaign on the basis of information collected about the customers and into two classes of respondents and nonrespondents. The input variables of the response model are the RFM variables (Recency, Frequency, Monetary (RFM)), which provide the customer purchase behavior and some demographic information. Customers with these

variables are induced into the model and the model classifies them as respondents or nonrespondents.

Response modeling overall procedure consists of several steps such as data preprocessing, feature construction, feature selection, class balancing, classification, and model evaluation; different data mining techniques and algorithms have been used for implementing each step of modeling. Very few studies deal with all these steps. Most of them focus only on two or three steps of modeling and for the rest use the result of previous works such as Ha et al. (2005), Shin and Cho (2006), and Yu and Cho (2006). The purpose of this study is to focus on all these steps. The following process has been followed for designing this model (Figure 6.1). Because of the nature of this study, different steps (components) in modeling were collected from a previous work and with some changes and amendments, integrated into a unique process. The algorithms related to each step were programmed and coded in R Open Source Language. After coding and running each step, they were combined together to make the prediction system.

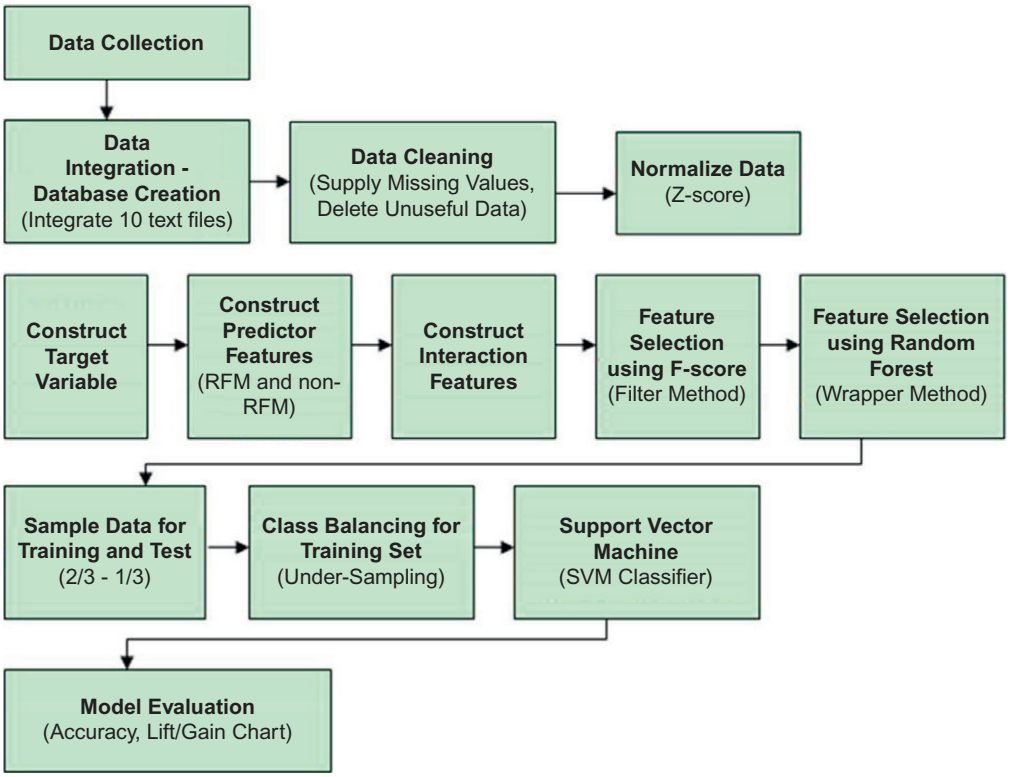


Figure 6.1
Proposed response model.

6.4 Modeling Detail

6.4.1 Data Collection

After having determined the most suitable research strategy, it is necessary to decide on how the empirical data will be collected (Yin, 1994). According to Zikmund (2003), there are two kinds of data normally used in researches: secondary and primary data. For data mining purposes, secondary data are used. The information at the level of the individual consumer is typically used to construct a response model. Response models are typically built from historical purchase data. For data mining purposes, the purchase history can often be translated into features on the basis of the measures of RFM values (Van den Poel, 2003).

The customers' data were gathered from the bank's databases for building a model. From 1,000,000 customers about 30,000 customers were randomly selected through IT section and given for modeling. In addition to customer data, campaign data were also collected from the marketing section of the company. In Table 6.1, the gathered information is shown. Following the literature and in close cooperation with a domain expert, these features (Table 6.1) were chosen directly from the database tables. These features allowed us to drive and construct all necessary purchase behavior history/RFM variables.

6.4.2 Data Preprocessing

Once the data required for the data mining process is collected, it must be in the appropriate format or distribution. Therefore, it has to be integrated, cleaned, and transformed to meet the requirements of the data mining algorithms. Data preprocessing is an essential step for

Table 6.1 Collected Data

Historical Purchase Data and Demographic	
Account's type that the customer has (long-termed investment deposit account, short-termed investment deposit account, savings account, current account) No of accounts that the customer has Open date of an account Close date of an account Type of services that the customer take (SMS, EMS, ISS, Telephone-Bank) No of services that the customer take Start date of service Age	
Transaction Data	**Campaign Data**
No. of transactions Type of each transaction (credit, debit) Amount of each transaction Date of each transaction	Date of different campaigns Content of each campaign

knowledge discovery and data mining. There are a number of data preprocessing techniques: data integration, data cleaning, data transformation, and data reduction.

6.4.2.1 Data Integration and Cleaning

Data integration merges data from multiple sources into a coherent data store. These sources may include multiple databases, data cubes, or flat files. The collected customer data from the bank's database were in the 10 text files. All the customers had unique IDs, which were the same in all text files. In this step, all 10 text files were merged into one table in the database. In this study, the database was created in the SQLite database and the 10 text files were integrated into one table in the SQLite database. After the data were integrated into one table, the data had to be cleaned and transformed to meet the requirements of the data mining algorithms. Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data. According to Han and Kamber (2006), there are several strategies to deal with missing values and noisy data.

For this data, very little data cleaning was required. Only some variables had missing values. All records with missing values and all the data that were only overheads and were not helpful in any way were deleted from the table. Data cleaning was done in R Languages Program; it can also be done in SQLite browser, as follows:

- Delete all records have missing values from the table.
- Delete all those customer names whose opening account date was after 20/01/2006 and with no account before 20/01/2006 from the table. These customers are not useful for prediction because they have no purchase history in the database.
- Delete all customers who closed their account from the table.
- Delete all records (customers) that have negative age from the table.

After data cleaning, the dataset from 30,000 customers (records) was reduced to 22,427 customers (records).

6.4.2.2 Data Normalization

Data normalization involves scaling the attribute values to make them lie numerically in the same interval/scale, and thus have the same importance. Because SVMs produce better models when the data are normalized, all data should be normalized or standardized before classification. There are three normalization techniques: Z-score Normalization, Min-Max Normalization, and Normalization by decimal scaling. There is no difference between these three techniques.

For this study the Z-score Normalization was used. The data were normalized using the mean and standard deviation. All data have a mean of zero and a standard deviation of 1. For each variable, this was done by subtracting the mean of the variable and dividing by the standard deviation, to arrive at the Z-score. Scale function in R program was used for the normalization task.

6.4.3 Feature Construction

After data integration, cleaning, and normalization, all the necessary variables, dependent and independent, for modeling had to be constructed.

For building a model and constructing variables it is very important to decide on the campaign to be used for modeling. One might consider constructing a model for each campaign, but according to Potharst et al. (2002), it is not a good choice for several reasons: the models are not likely to be of the same quality, the more recent a campaign, the more historical data available, the database has grown over time, information about more customers is available from the more recent campaigns. Therefore, for this study, the campaigns data collected from the bank's most recent campaign was used for construction of the variables and the model.

6.4.3.1 Target Variable Construction

The response model is a classification model. The task is to classify the customers who will respond to the next marketing campaign on the basis of information collected about the customers. The first step in the predictive modeling process is to determine the objective variable to be modeled. In building models for targeting direct marketing campaigns, the objective variable is usually based on the customer's historical response data to similar direct marketing campaigns (Cabena et al., 1999). The selected bank used mass marketing as its strategy for offering a new product or service to the customers and they did not have a direct marketing campaign or the customer's historical response data to the direct marketing campaign. So in this study for building the target variable customer's historical response data to mass marketing campaigns was used.

The first and critical step in creating the target (dependent) or objective variable is to select the time period in consideration. Setting the objective variable correctly is critical. The size of the time window selected is driven by the time horizon of the desired prediction. For instance, if the marketing campaign to be executed has a 6-month window for the customer to respond, the objective variable should be defined with a 6-month window (Cabena et al., 1999). Actually it is very important to decide which campaign (time window) should be used for modeling.

Among the various mass marketing campaigns that the company had, the most recent campaign (period between 21/01/2006 and 23/07/2006) was selected for building a model and target variable. Purchase information from the period between 21/01/2006 and 23/07/2006 was used as the dependent variable. All the customers were categorized as either a respondent (class 1) or as nonrespondent (class 0) depending upon whether they made the purchase during the period between 21/01/2006 and 23/07/2006 or not, respectively. All those customers whose open account date was between 21/01/2006 and 23/07/2006 were declared as respondents and rest of them were nonrespondents. The target variable in the table was assigned a value 1 for respondents (if a customer had opened an account during period of 21/01/2006 through 23/07/2006) and value 0 for nonrespondents.

6.4.3.2 Predictor Variables

In this study, the RFM variables and demographic information were used as independent variables. Cullinan (1977) is generally credited with identifying the three sets of variables most often used in response modeling: RFM (Bauer, 1988; Kestnbaum, 1992). Since then, the literature has provided so many uses for these three variable categories, that there is overwhelming evidence both from academically reviewed studies as well as from practitioners' experience that the RFM variables are an important set of predictors for response modeling.

There are various approaches to feature construction. In this study, all RFM variables were constructed from a knowledge-based approach and the literature. Using domain knowledge to construct new features is often recommended because one can thus quickly rule out many impossible combinations and determine an effective measure to evaluate new compound features. Some information on customer historical purchase data was obtained from the bank. This allowed us, in close cooperation with domain experts and guided by the extensive literature, to derive and construct all the necessary purchase behavior variables/RFM variables for a total sample size of 22,427 customers. 85 predictor features and target variables were constructed from gathered information. These features are listed in Table 6.3.

Two time horizons (Hist Horizon and Year Horizon) for constructing all RFM variables were used (Table 6.2). All the variables were measured for this two time period. The Hist horizon refers to the fact that the variable is measured for the period between 2002 and 20/01/2006. The Year horizon refers to the fact that the variable is measured over the last year (time period between 20/01/2005 and 20/01/2006). Including both time horizons allows us to check whether more recent data are more relevant than historical data.

In this study, the Recency variables were defined as:

- Number of days since the last open account
- Number of days since the first open account
- Number of days since the first transaction
- Number of days since the last transaction
- Number of days since the largest transaction
- Number of days since the first service started
- Number of days since the last service started

Recency has been found to be inversely related to the probability of the next purchase, i.e., the longer the time delay since the last purchase the lower the probability of a next purchase within the specific period (Cullinan, 1977).

In the context of direct marketing, it has generally been observed that multibuyers are more likely to repurchase than buyers who only purchased once (Bauer, 1988; Stone, 1984). The frequency variable is generally considered to be the most important of the RFM variables

Table 6.2 Constructed Features
Dependent Variable and Predictor Variables

Name	Description
Target	Dependent variable
Age	Demographic Information
Recency	
recencyfirstaccount	No. of days since first open account
recencylastaccount	No. of days since last open account
recencylastservice	No. of days since last start services
recencyfirstservice	No. of days since first start services
recencylasttran	No. of days since last transaction
recencymaxtrandate	No. of days since maximum transaction
Frequency	
Jari	Life to date no. of checking accounts
gharz	Life to date no. of gharz accounts
long	Life to date no. of long savings accounts
short	Life to date no. of short savings account
totalacc	Total accounts
telephonebank	Life to date no. of telephonebank
sms	Life to date no. of sms
ems	Life to date no. of ems
iis	Life to date no. of iis
totalservice	Life to date total services
jarilastyear	No. of checking accounts over last year
gharzlastyear	No. of gharz accounts over last year
Monetary	
totaltranjari	Total money that a customer put in checking account
totaltrangharz	Total money that a customer put in gharz account
totaltranlong	Total money that a customer put in long savings account
totaltranshort	Total money that a customer put in short savings account
totaltranjarilast	Total money that a customer put in checking account last year
totaltrangharzlast	Total money that a customer put in gharz account last year
totaltranlonglast	Total money that a customer put in long savings account last year
totaltranshortlast	Total money that a customer put in short savings account last

Table 6.3 Two Time Horizons for Constructing RFM Variables

Time Horizon	Description
Hist Horizon	Variable is measured for the period between 2002 and s20/01/2006
Year Horizon	Variable is measured over the last year

(Nash, 1994). Bauer (1988) suggests the frequency variable be defined as the number of purchases divided by the time on the customer list since the first purchase for operational purposes. In this study, the frequency variables were defined as:

- The number of accounts that the customer has in a certain time period (Hist versus Year)
- The number of services that the customer has taken in a certain time period (Hist versus Year)
- The number of transactions in a certain time period (Hist versus Year)

In the direct marketing literature, the general convention is that the more money a person has spent with a company, the higher his/her likelihood of purchasing the next offering (Zahavi and Levin, 1996). Nash (1994) suggests that the operational monetary value be taken as the highest transaction sale or as the average order size. Zahavi and Levin (1996) propose to use the average amount of money per purchase. For this study, the monetary variables were modeled as:

- Total money that the customer put in the bank (for each account) during a certain time period (Hist versus Year)
- Total money that the customer withdrew from the bank (for each account) during a certain time period (Hist versus Year)
- Maximum amount of money that the customer deposited in the bank (for each account) during a certain time period (Hist versus Year)
- Minimum amount of money that the customer deposited in the bank (for each account) during a certain time period (Hist versus Year)
- Average amount of money that the customer deposited in the bank (for each account) during a certain time period (Hist versus Year)

Apart from the RFM variables, demographic information was used as an independent variable. The only demographic information used in this research was the age of each customer, which was calculated from the given date of birth.

6.4.3.3 Interaction Variables

In addition to studying the effects of individual independent variables (predictors), it is also important to study their interactions. An interaction effect is a change in the simple main effect of one variable over levels of the second (Candade, 2004). After construction of the individual independent variables the second step is to construct interactions features. First, interactions between the predictor variables should be identified by an appropriate method or model which searches for two-way interactions between candidate predictors, and then after identifying which variables interacted, the pool of candidate predictors is augmented by adding products between the interacting variables. In this study because the appropriate method and tool for identifying interactions between predictors variable couldn't be found, all the 85 constructed features were multiplied two by two and then by means of F-score selection, in the feature selection step, appropriate ones were selected. RFM variables and their interactions were coded and constructed in R Language Program and then added to the table in SQLite database. In the next section, the feature selection step and the techniques that were used for this step are described.

After construction of all the necessary purchase behavior variables/RFM and their interaction, it is important to select the best subset of features for modeling. For this study, feature selection was performed in three steps using both the wrapper and filter method. The following sections describe these steps and also show the result of each step.

6.4.4 Feature Selection

Feature selection is a critical step in response modeling, because customer related datasets usually contain hundreds of features or variables, many of which are irrelevant and heavily correlated with others. Without prescreening or feature selection, they tend to deteriorate the performance of the model, as well as increase the model training time. Feature subset selection can be formulated as an optimization problem, which involves searching the space of possible features to identify a subset that is optimum or near-optimal with respect to performance measures such as accuracy (Yang and Honavar, 1998). Various ways exist to perform variable and feature subset selection (Guyon and Elisseeff, 2003). Feature selection can either be performed as a preprocessing step independent of the induction algorithm or as a process explicitly making use of the induction algorithm. The former approach is termed *filter*, the latter *wrapper* (John et al., 1994). Filter methods operate independent of the target learning algorithm. Undesirable inputs are filtered out of the data before induction commences. Wrapper methods make use of the actual target learning algorithm to evaluate the usefulness of inputs. Typically the input evaluation heuristic that is used is based upon inspection of the trained parameters and/or comparison of predictive performance under different input subset configurations. Input selection is then often performed in a sequential fashion. The backward selection scheme starts from a full input set and prunes the input variables that are undesirable step-wise. The forward selection scheme starts from the empty input set and adds input variables that are desirable step-wise.

Feature wrappers often achieve better results than filters because they are tuned to the specific interaction between an induction algorithm and its training data, but also tend to be more computationally expensive than the filter model (Blum and Langley, 1997). When the number of features becomes very large, the filter model is usually chosen due to its computational efficiency. Chen et al. (2005) combined SVM with various feature selection strategies, the filter type and wrapper type methods.

Various methods have been proposed to alleviate the irrelevant or redundant features in response modeling. Malthouse (2001) proposes a variable selection algorithm that optimizes the performance of the response model. This article proposes a forward selection algorithm that takes model performance into consideration when making decisions about which variables should be used in the model. Viaene et al. (2001b) applied a least squares support vector machine (LS-SVM)-based input selection wrapper to a real-life direct marketing case involving the modeling of repeat-purchase modeling in direct marketing and they showed that by reducing the number of input features, both human understanding and computational

performance can often be vastly enhanced. In this chapter, binary (buyer versus nonbuyer) classification problem is tackled by using LS-SVM classifiers. The input selection procedure was based upon a (greedy) best-first heuristic, guiding a backward search mechanism through the input space. Also Viaene et al. (2001a) in another research implemented feature selection used a typical wrapper approach with a best-first search heuristic guiding the backward search procedure toward the optimal input set. Starting with the full set, all inputs are pruned sequentially, i.e., one by one. They used multilayer neural networks as induction mechanism for their research. The results of this research indicated that elimination of redundant and/or irrelevant inputs by means of the applied input selection method allow significant reduction in model complexity without degrading the predictive generalization ability. Yu and Cho (2006) proposed the feature selection ensemble model based on a GA wrapper approach.

In this study, because the number of constructed features (individual and interaction features) were large, the wrapper method alone couldn't be used. So feature selection according to Chen et al. (2005) was performed by using both the wrapper and filter method. Filter method (F-score) was used as a preprocessing step and then the wrapper method (Random Forest—backward elimination) was used for selecting final features as input for the model. Thus feature selection was implemented in three steps: (1) the most important interaction features were selected by using F-Score selection, (2) selected interaction features were added to 85 individual features and then 50 important features were selected by using F-score selection, (3) the best subset of features were selected as input for modeling by using Random Forest (RF). In practice, the RF cannot handle too many features, and in addition, as mentioned above, when the number of features are large the wrapper approach tends to be more computationally expensive than the filter model. Thus, before using RF (wrapper approach) to select features, a subset of features was obtained using the F-score selection first.

6.4.4.1 F-Score

F-score is a simple technique which measures the discrimination between two sets of real numbers. Given training vectors x_k, $k=1, \ldots, m$, if the numbers of positive and negative instances are $n+$ and $n-$, respectively, then the F-score of the ith feature is defined as:

$$F(i) \equiv \frac{\left(\bar{x}_i^{(+)} - \bar{x}_i\right)^2 + \left(\bar{x}_i^{(-)} - \bar{x}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \bar{x}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \bar{x}_i^{(-)}\right)^2}$$

where $\bar{x}_i, \bar{x}_i^{(+)}, \bar{x}_i^{(-)}$ are the averages of the ith features of the whole, positive, and negative data sets, respectively; $x_{k,i}^{(+)}$ is the ith feature of the kth positive instance, and $x_{k,i}^{(-)}$ is the ith feature of the kth negative instance. The numerator indicates the discrimination between the positive and negative sets, and the denominator (the sum of variances of the positive and negative sets) indicates the one within each of the two sets. The larger the F-score is, the more likely this feature is more discriminative. Therefore, we use this score as a feature selection criterion (Chen et al., 2005).

A disadvantage of F-score is that is does not reveal mutual information among features. Despite this disadvantage, the F-score is simple and generally quite effective. In this study, the F-score was used as a preprocessing step and features with high F-scores were selected and then RF was applied to select final features as input for modeling. The F-score was programmed and implemented in the R language program.

6.4.4.2 Step1: Selection of Interaction Features Using F-Score

In the first step F-score, which is the filter method, was used to select the most significant interaction features. In addition to constructing all the necessary purchase behavior variables (RFM variables) from raw data, interaction between these variables must be constructed for modeling. Because the appropriate tool or algorithm for finding interaction between predictor variables couldn't be found, all the 85 features were multiplied two by two. As the numbers of interaction features were large, the most appropriate ones were selected by means of F-score selection. While constructing two-way interactions of independents variables, F-score for each interaction was calculated. After calculation of F-score for each interaction, the features were ranked in descending order. From the interaction features, the first 20 features with high F-scores were selected. These selected features with their F-scores are listed in Table 6.4. In the following table, 20 selected features are ranked in descending order based on the F-score.

Table 6.4 Selected Interaction Features Using F-Score

No.	Name	Formulation	F-Score
1	Inter 74-14	longlastyear*recencylastaccount	1.05×10^{-1}
2	Inter 74-14	notranlonglast*recencylastaccount	9.09×10^{-2}
3	Inter 58-14	notranlong*recencylastaccount	9.09×10^{-2}
4	Inter 16-14	gharzlastyear*recencylastaccount	7.16×10^{-2}
5	Inter 55-17	recencylasttran*longlastyear	6.94×10^{-2}
6	Inter 55-19	recencylasttran*totalacclastyear	6.85×10^{-2}
7	Inter 55-6	recencylasttran*totalacc	6.59×10^{-2}
8	Inter 14-4	recencylastaccount*long	6.55×10^{-2}
9	Inter 55-4	recencylasttran*long	6.45×10^{-2}
10	Inter 14-6	recencylastaccount*totalacc	6.26×10^{-2}
11	Inter 74-55	notranlonglast*recencylasttran	5.54×10^{-2}
12	Inter 58-55	notranlong*recencylasttran	5.47×10^{-2}
13	Inter 27-14	totalaccserv*recencylastaccount	5.54×10^{-2}
14	Inter 19-14	totalacclastyear*recencylastaccount	5.01×10^{-2}
15	Inter 55-27	recencylasttran*totalaccserv	4.97×10^{-2}
16	Inter 55-28	recencylasttran*totalaccservlastyear	4.93×10^{-2}
17	Inter 28-14	totalaccservlastyear*recencylastaccount	4.39×10^{-2}
18	Inter 87-55	recencymaxtrandate*recencylasttran	4.30×10^{-2}
19	Inter 17-3	longlastyear*gharz	4.08×10^{-2}
20	Inter 19-16	totalacclastyear*gharzlastyear	4.14×10^{-2}

```
fs <- matrix(0,ncol(customertarget3fdatanormal),ncol(customertarget3fdatanormal))
v <- 0
for (i in 2:ncol(fs)){
    for (j in 2:(i-1)){
v <- 0
       v <- customertarget3fdatanormal[,i]*customertarget3fdatanormal[,j]

       meanpositive <- mean(v[20716:22427])
       meannegative <- mean(v[1:20715])
       meantotal <- mean(v[1:22427])
       varpositive <- var(v[20716:22427])
       varnegative <- var(v[1:20715])

       fs[i,j] <- (((((meanpositive-meantotal)^2)+((meannegative-meantotal)^2)))/
(varpositive+varnegative))
    }
}

#Finding High F-score
f <- matrix(0,7569,3)
p <- 1
    for (i in 1:ncol(fs)){
        for (j in 1:nrow(fs)){
            f[p,2] <- i
            f[p,3] <- j
            f[p,1] <- fs[i,j]
            p <- p+1
        }
    }
f2 <- as.data.frame(f[order(f[,1]), ])
names(customertarget3fdata[55])
for (t in 7532:nrow(f2)){
    i <- f2[t,2]
    j <- f2[t,3]
 dbSendQuery(con, paste("alter table customertarget3f add column inter",
as.character(i),"_", as.character(j) ," integer", sep = ""))
    dbSendQuery(con, paste("update customertarget3f set inter", as.character(i),"_",
as.character(j) ,"=", as.character(names(customertarget3fdata[i])),"  *  ",
as.character(names(customertarget3fdata[j])), sep=""))
}
```

6.4.4.3 Step2: Selection of Features Using F-Score

After selection of the 20 interactions with high F-scores, these selected features were added to the individual independent variables, so the numbers of predictor variables became 105 (constructed features = 85 and selected interaction features = 20). As the number of features (105) were large and the RF code that was used cannot handle too many features, F-score was used again to obtain a subset of features before using RF to select the best subset of features for modeling. F-score was actually used as a preprocessing step. The F-score for every feature

(105 features) was calculated. After calculating the F-score for each feature, the features were sort in descending order, and from the 105 features, 50 of them with high F-scores were selected. This threshold was selected because there are indications in literature that when the number of features is larger than 50, RF cannot end in appropriate time.

```
fs4 <- matrix(0,ncol(customertarget3finternormal),2)
v <- 0
for (i in 1:nrow(fs4)){
       v <- 0
       v <- customertarget3finternormal[,i]

       meanpositive <- mean(v[20716:22427])
       meannegative <- mean(v[1:20715])
       meantotal <- mean(v[1:22427])
       varpositive <- var(v[20716:22427])
       varnegative <- var(v[1:20715])

       fs4[i,1] <- i
       fs4[i,2] <- (((((meanpositive-meantotal)^2)+((meannegative-
meantotal)^2)))/(varpositive+varnegative))
}
fsorder4 <- as.data.frame(fs4[order(fs4[,2]), ])

dbWriteTable(con, "customertargetselectedf", customerselectedf , overwrite = T)
customerselectedfscale <- subset(customerselectedf, select=-target)
customerselectedfscale <- subset(customerselectedfscale, select=-ID)
customerselectedfscale <- scale(customerselectedfscale)
customerselectedfscale <- cbind(customerselectedfscale,data.frame
(target=customertarget3fdata[,71]))
customerselectedfscale <- cbind(customerselectedfscale,data.frame
(ID=customertarget3fdata[,1]))
dbWriteTable(con, "customerselectedfscale", customerselectedfscale, overwrite = T)
```

6.4.4.4 Step3: Selection of Best Subset of Features Using Random Forest

After the selection of the 50 features, RF was performed to select the best subset features for modeling. RF is a classification method, but it also provides feature importance (Breiman, 2001). The RF algorithm estimates the importance of a variable by looking at how much the prediction error increases when the out-of-bag (OOB) data for that variable is permuted while all others are left unchanged. The OOB data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

RF for feature selection uses backward variable elimination approach, which is the wrapper approach, with the OOB error as a minimization criterion. The backward elimination begins with all the features in the model and at each step eliminates the feature that contributes least to the discriminatory power of the model. The process stops when all the remaining features meet the criterion to stay in the model.

In the final step of feature selection, RF was performed for selection of best subset of features for modeling with 50 features selected from previous step (step 2). varSelRF and randomForest package in R were used for implementing RF. varSelRF function in this package selects variables from RF using backward variable elimination approach. The OOB is used as a minimization measure for error, and variable elimination from RF is carried out, by successively eliminating the least important variables (with importance as returned from RF). varSelRF in R language program was performed with 50 features and with the default parameters (mtryFactor = 1, c.sd = 1, ntree = 2000, ntreeIterat = 1000, and vars.drop.frac = 0.2). After performing varSelRF with 50 features, the best subset of 26 important features was selected. The OOB estimate of error rate of this selected set of variables was 7.81%. The first forest (before any variable selection) was fitted with all 50 features and iteratively, a fraction of the least important variables used in the previous iteration was eliminated. The initial OOB error (the OOB estimate of error rate of first forest) was 7.65%. The numbers of variables that were in the forest at that stage (iteration) and the OOB error rate of all the fitted variables are shown in Table 6.5.

```
library(randomForest)
library(varSelRF)
Featureselect<-varSelRF(subset(customertargetselectedscale2data, select= target),
customertargetselectedscale2data$target, mtryFactor = 1 , ntree = 2000, ntreeIterat = 1000,
vars.drop.frac = 0.2,whole.range =FALSE, verbose = TRUE)
```

Based on the "c.sd = 1" rule, the RF selected the 26 features with 7.81% OOB error rate as the best subset of features. These selected features are listed in Table 6.6. These 26 features were used as input variables for SVM modeling.

6.4.5 Data Sampling for Training and Test

After the selection of the best subset of features for modeling, the dataset containing 22,427 customers was partitioned into training and test sets. It is important to create both a training data set, which is used to build the model, and a test or hold-back data set, which is used to test the model. A model should be tested against data that it has never seen before to ensure that there is no over-fitting. Because if we rely solely on training data, it is very easy to get exceptional results. However, these results would likely not generalize to new examples, which would be missing from the stored table. Researchers build a classifier using the train set and evaluate it

Table 6.5 Random Forest Result

Iteration	No. of Variables	OOB Error
1	40	mean = 0.0777, sd = 0.0018
2	32	mean = 0.0772; sd = 0.0018
3	26	mean = 0.0781; sd = 0.0018
4	21	mean = 0.0788; sd = 0.0018

Table 6.6 Selected Features Using Backward Elimination on Random Forest (Input Variables)
(Parameters used
mtry = 1, ntree = 2000, ntreeIterat = 1000, vars.drop.frac = 0.2, c.sd = 1)

Variables	Description
Frequency	
longlastyear	No. of long saving accounts over the last year
notranshort	No. of transactions for short savings account
notranshortlast	No. of transactions for short saving accounts over last year
notrantotal	LTD total transaction
notrantotallast	Total transaction over last year
Recency	
recencylastaccount	No. of days since the last account was opened
recencylasttran	No. of days since the last transaction
recencymaxtrandate	No. of days since the maximum transaction
Monetary	
totaldebitshort	LTD total money that the customer drew from the short account
totaldebitshortlast	Total money that the customer drew from the short saving account last year
totaltran	LTD total money that the customer put in all accounts
totaltranlast	Total money that the customer put in all accounts over the last year
totaltranlong	LTD total money that the customer put in long savings account
totaltranlonglast	Total money that the customer put in long savings account over last year
totaltranshort	LTD total money that the customer put in short savings account
totaltranshortlast	Total money that the customer put in short savings account last year
avgtranlong	Average LTD money that the customer put in long savings account
avgtranlonglast	Average money that the customer put in long savings account
avgtranshortlast	Average money that the customer put in short savings account during last year
Interaction	
inter55_27	recencylasttran*totalaccserv
inter14_4	recencylastaccount*long
inter17_14	longlastyear*recencylastaccount
inter28_14	totalaccservlastyear*recencylastaccount
inter55_6	recencylasttran*totalacc
inter87_55	recencymaxtrandate*recencylasttran
inter19_14	totalacclastyear*recencylastaccount

using the test set. They randomly split data into training and test sets. Training is typically done on a large proportion of the total data available, whereas testing is done on some small percentage of the data that has been held exclusively for this purpose (usually 2/3 for train, 1/3 for test).

In this case, we were trying to build a model that would predict a binary outcome. Dataset for building this model contains 22,427 customers with 20,715 nonrespondents and 1712 respondents. Two-thirds of the customers (14,952 records) were randomly assigned to the training set while the other one-third (7475 records) were assigned to the test set. Making use of a training set the SVM model was built; later the test set was used to evaluate the accuracy of the model.

```
#split data into a training and test set
index <- 1:nrow(customerdata)
testindex <- sample(index,trunc(length(index)/3))
testset <- customerdata[testindex,]
trainset <- customerdata[-testindex,]

non1 <- testset[testset$target==0,]
res1 <- testset[testset$target==1,]
nrow(non1)
nrow(res1)
```

6.4.6 Class Balancing

Many practical classification problems are imbalanced. The class imbalance problem typically occurs when there are many more instances of some classes than others. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones. When classifiers are faced with imbalanced datasets, where the number of negative instances far outnumbers the positive instances, the performance drops significantly. Various strategies have been proposed to deal with class imbalance problems, such as increasing the penalty associated with misclassifying the positive class relative to the negative class, over-sampling the majority class, and undersampling the minority class. Among these approaches, the undersampling method has been very popular.

In this study, the dataset contains 22,427 customers each of whom is described by 85 individual features and their two-way interactions. The response rate is 7% with 1712 respondents and 20,715 nonrespondents. Also when the dataset was partitioned into training and test sets (two-third for training and one-third for test) the training set had 13,809 nonrespondents and 1143 of respondents, which means that the class distribution is highly imbalanced. According to Shin and Cho (2006), SVMs do not work very well with unbalanced data sets, so the training set should be balanced. Undersampling the nonrespondent class was chosen for this study. Nonrespondents were randomly selected to equal twice the number of respondents in the training set. After balancing the class distribution in the training set (undersampling the nonrespondent class) the number of nonrespondents in training set were two times the number of respondents. So the response rate of the training set was 33% with 1143 respondents and 2286 nonrespondents. Table 6.7 shows the training set response rate before and after class balancing.

Table 6.7 Class Balancing in Training Set

	Training Set No. of Respondents	Training Set No. of Nonrespondents	Response Rate (%)
Before balancing	1143	13,809	7
After balancing (undersampling)	1143	2286	33

```
# Class Balancing
non <- trainset[trainset$target==0,]
res <- trainset[trainset$target==1,]
nrow(non)
nrow(res)
indexnon <-sample(1:nrow(non),2*nrow(res),replace=FALSE)
sampletrain <- rbind(res,non[indexnon,])
sampletrain <- (subset(sampletrain, select=-ID))
sampletrain$target <- factor(sampletrain$target)
```

Making use of a training set, the SVM model was built, and later the test set was used to evaluate the accuracy of the model. The performance of a model is not evaluated on the training set; because it has had a hand in creating the model and so will overstate the model's accuracy. The model's accuracy is always a measure on a test set that is drawn from the same population as the training set, but has not been used in any way to create the model (Berry and Linoff, 2004). Thus in this study, as explained earlier, the dataset was divided into training and test sets. The training set was used for model building and the test set for measuring the model's accuracy. The results returned when the model was run on the test set are shown by the confusion matrix (Table 6.8). From the confusion matrix the overall accuracy of the model was calculated. *True Negative Rate* (*TNR*) and *True Positive Rate* (*TPR*) were used to evaluate the performance of learning algorithms on imbalanced data. The overall accuracy on the test set is 81% with the 83% (5727) TNR and 57% (324) TPR. The response rate of the imbalanced data (test set) was 7% and therefore, the overall accuracy of the model was calculated on the test set as (7%*TPR) + (93%*TNR) = 81%.

6.4.7 Classifier (SVM)

Once the data preparation, feature construction, and selecting the best subset of features for modeling the response are complete, the classification algorithm and the algorithm parameters have to be selected. Various classification methods (classifiers) have been used for response modeling such as statistical and machine learning methods, neural networks, decision trees, and SVMs. In this study, SVM was used as a classifier for classification. SVMs show distinct

Table 6.8 Confusion Matrix

		Predicted		
		Class1 (negative) nonrespondent (0)	**Class2 (positive) respondent (1)**	
True	Class1 (negative) Nonrespondent (0)	5727	1179	6906
	Class2 (positive) respondent (1)	245	324	569
		5972	1503	7475

advantages such as better generalization, increased speed of learning, the ability to find a global optimum, and the ability to deal with linearly nonseparable data.

For implementing SVM in R Language Program, SVM function in e1071 package was used for classification task. The TUNE() function from e1071 package was used. This generic function tunes hyper-parameters of statistical methods using a grid-search over supplied parameter ranges. It returns the best values to use for the parameters gamma and cost.

SVM requires a certain amount of model selection (parameter search). The kernel parameter is one of the most important design choices for the SVM because it implicitly defines the structure of the high dimensional feature space where a maximal margin hyperplane will be found. Thus the choice of the SVM kernel is crucial. As was recommended in the previous work, the radial basis function (RBF) kernel was chosen for this study. There are two parameters in using RBF kernels: C (cost = misclassification tolerance parameter) and γ (gamma). In the R Language Program, the default parameter of cost is 1 and default parameter of gamma is 1/ncol (as.matrix (data)) which is 0.03571429.

In order to fine tune these parameters and find the ones which generalize better, "grid-search" on C and γ using cross-validation (fivefold cross-validation) was used. First, on the basis of the default value of C and γ, the possible intervals for C and γ with grid space were defined. Then, all grid points of (C, γ) were tested to find the one that gives the highest cross-validation accuracy. Then, the best parameter was used to train the whole training set and generate the final model. Parameter selection was performed using various values of C and γ.

Trying exponentially growing sequences of C and γ is a practical method to identify good parameters (for instance, $C = 10^{\wedge}(-2{:}4)$, $\gamma = 2^{\wedge}(-8{:}0)$). As explained different combinations of C and γ using fivefold cross-validations were examined. Pairs of (C, γ) were tried and the one with the best cross-validation accuracy was picked. Among different combinations of C and γ that were tried, the combinations of $C = 10^{\wedge}(-2{:}4)$ {0.01, 0.1, 1, 10, 100, 1000, 10,000} and $\gamma = 2^{\wedge}(-8{:}0)$ {0.00390625, 0.00781250, 0.01562500, 0.03125000, 0.06250000, 0.12500000, 0.25000000, 0.50000000, 1.00000000} were the best ones. The best (C, γ) pair was (10, 0.0078125) with the lowest error rate 0.2723793. As can be seen in Figure 6.2 the lowest error 0.27 belongs to the orange point in the whitest area which has $C = 10$ and $\gamma = 0.0078125$. Thus these values for C and γ were used for modeling. Figure 6.2 visualizes the results of parameter tuning for SVM model.

```
# create model using SVM (support vector machine)

#SVM requires tuning

library(e1071)

wts <- 100 / table(sampletrain$target)

mytune <- tune(svm, target ~ ., data=sampletrain ,class.weights = wts, probability =TRUE,
ranges=list(gamma = 2^(-8:0),cost = 10^(-2:4)),scale=FALSE, tunecontrol = tune.control
(best.model = TRUE,performances=TRUE, sampling="cross",cross=5))
```

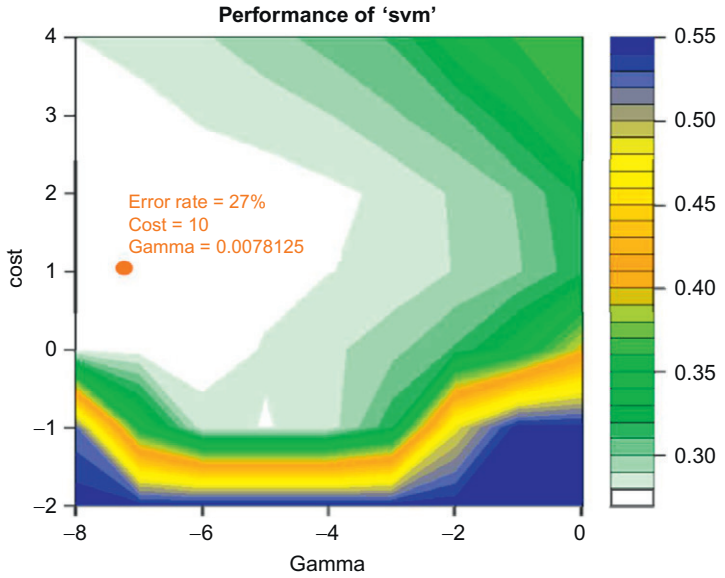


Figure 6.2
Result of parameter tuning for SVM model.

6.5 Prediction Result

After building a model on a training set, to ensure that the model is not an overfit for the data, the test set was used to evaluate the model accuracy. The exciting model was executed to assess the model performance against the test data. The results returned when the model was run on the test set is shown by the confusion matrix. The prediction result is shown by the confusion matrix. The confusion matrix contains information about actual and predicted classifications done by a classification system (model). Table 6.8, the confusion matrix, shows the result returned when the model was run on the test set. The rows of the matrix are actual classes, and the columns are the predicted classes.

```
#Use the model to predict the evaluation
testset <– (subset(testset, select=–ID))
testset$target <– factor(testset$target)
predTarget1 <– predict(model, subset(testset, select=– target),decision.values=TRUE,
probability = TRUE)
a1 <– attr(predTarget1, "probabilities")
table(pred=predTarget1,testset$target)
sum(predTarget1==testset$target)/length(testset$target)
summary(model)
```

As can be seen in confusion matrix (Table 6.8), 5972 corresponds to the number of customers that the model classified as nonrespondent and 1503 corresponds to the numbers of customers that the model classified as respondent. 5727 customers out of 6906 customers were

predicted as nonrespondents who were actually nonrespondents (TN, true negative), also 324 customers out of 569 customers were predicted as respondents who were actually respondents (TP, true positive). 1179 out of 6906 customers were predicted as respondents while they were nonrespondents (FP, false positive). There were also 245 out of 569 customers predicted as nonrespondents who were actually respondents (FN, false negative).

The accuracy of a model and the other evaluation criteria are calculated by the confusion matrix. In this study, the performance of the SVM response model was measured by using accuracy, TPR, TNR, weighted accuracy, and the lift chart. Using confusion matrix, these evaluation measurements were calculated and the lift/gain chart was drawn.

6.6 Model Evaluation

In this study, the performance of a model was measured by using accuracy, TPR, TNR, weighted accuracy, and lift chart. The model's accuracy is always a measure on a test set that is drawn from the same population as the training set, but has not been used in any way to create the model (Berry and Linoff, 2004). In this study, the training set was used for model building and test set for measuring the model's accuracy. The results returned when the model was run on the test set is shown by the confusion matrix (Table 6.8). From confusion matrix, the overall accuracy of the model was calculated. As can be seen in Table 6.9, the overall accuracy on the test set is 81%.

The accuracy (Acc) is a widely used metric for measuring the performance of learning systems. But when the prior probabilities of the classes are very different, such metrics might be misleading. A trivial classifier that predicts every case as the majority class can still achieve very high accuracy. In this study, because the number of negative cases (nonrespondents class) was much greater than the number of positive cases (respondent class), the overall classification accuracy was not an appropriate measure of performance. Thus, metrics such as TNR, TPR and weighted accuracy were used to evaluate the performance of learning algorithms on imbalanced data. All the metrics are functions of the confusion matrix. Based on the confusion matrix, the performance metrics were calculated as in Table 6.10.

Table 6.9 SVM Model Performance on Test Set

		Predicted		Accuracy
		Class1 (Negative) Nonrespondent (0)	Class2 (Positive) Respondent (1)	
True	Class1 (negative) Nonrespondent (0)	5727 TNR = 83%	1179	81.75%
	Class2 (positive) Respondent (1)	245	324 TPR = 57%	

Table 6.10 Evaluation Criteria

Evaluation Criteria	Formulation	Value (%)
Accuracy (Acc)	(TP + TN) / (TP + FP + TN + FN)	81
Error rate	1-Acc	19
True positive rate (TPR) (Acc +)	TP / (TP + FP)	57
True negative rate (TNR) (Acc −)	TN / (TN + FP)	83

For any classifier, there is always a tradeoff between TPR and TNR. In the case of learning algorithms with extremely imbalanced data, quite often the rare class is of great interest. In direct marketing application, it is desirable to have a classifier that gives high prediction accuracy over the minority class (TPR—Acc +), while maintaining reasonable accuracy for the majority class (TNR—Acc −). Actually accuracy for the respondents' group is of greater importance to direct marketers, since their primary goal is to identify the respondents, not the nonrespondents. Increasing the number of respondents is the goal of response modeling. In such situations, the weighted accuracy is often used. For these reasons, in this study, besides overall model accuracy TPR (Acc +), TNR (Acc −), weighted accuracy with $\lambda = 9$ were calculated. Table 6.10 shows that the overall model accuracy was 81% with 57% TPR (Acc +) and 83% TNR (Acc −),and the weighted accuracy was 72%.

The data in the confusion matrix were plotted in a lift or gains chart to visually evaluate the results of the model performance and to compare the performance of a constructed response model within the results achieved by random chance. Lift/gain chart is a graphical representation of the advantage of using a predictive response model to choose to contact customers with. A gain/lift chart shows the gain made by using a derived response model over random selection of targets. In this study, the lift chart was plotted in R Language Program. Prediction and Performance function in ROCR package were used for plotting the lift chart. The lift chart was created by sorting all the prospects in descending order according to their likelihood of responding as predicted by the model and selecting the instance subset starting from the one with the highest predicted probability.

Figure 6.3 shows the lift/gain chart that was created for the test set. The X-axis shows the ratio of the size of mailed group of clients to the size of the total client base ((TP+FP)/(TP+FP+TN+FN)) and the Y-axis shows the percentage of respondents that is actually captured by the model, which is the TPR. The bottom line (straight line) is the random line that indicates the customers were randomly selected for the marketing campaign. If no models were used, mailing to 20% of the population would reach 20% of the respondents. The other curve of the chart shows what happens if the model is used to select customers for the campaign. If only the top 20% of the customers are mailed, then 57% of the respondents are approached as opposed to only 20% in a randomly targeted mailing of the same size.

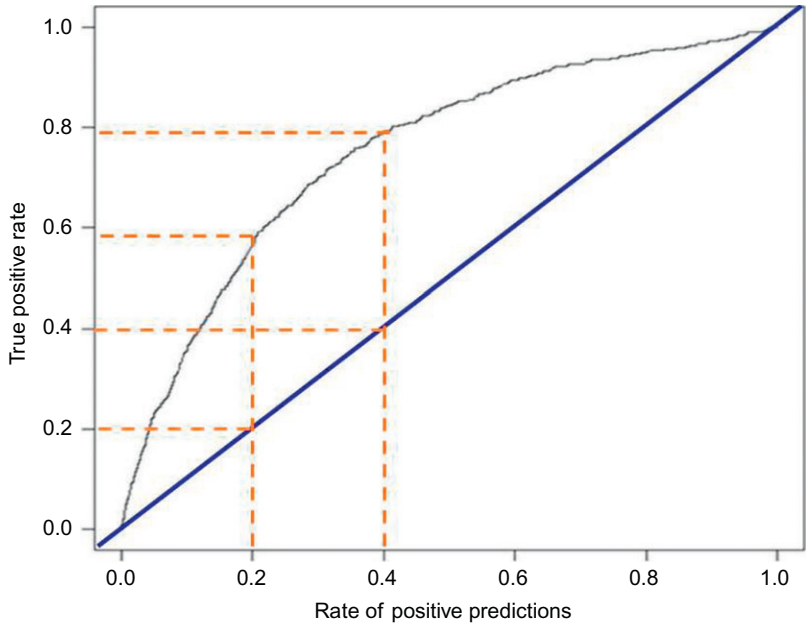


Figure 6.3
Lift chart of the model.

By using this model, the company can get three times as many respondents for its marketing expenditure than it would have received by mailing to 20% of its one million prospects at random. So it can significantly reduce the overall marketing cost.

```
#Plot the performance of the model applied to the evaluation set as an ROC curve and Lift Chart
library(ROCR)
pred <- prediction( a1[,1], testset$target)
perf <- performance(pred,"tpr","rpp")
plot(perf)
perf <- performance(pred,"lift","rpp")
plot(perf)
```

6.7 Conclusion

Data mining is a very powerful technology that can help an organization to plan effective direct marketing campaigns. Most of the research on data mining for direct marketing focuses on the theoretical and computational aspects of the technology. Yet these researches fail to achieve managerial application. In this study a response model for target selection in direct marketing with data mining techniques (with R) was constructed for one of the Iranian banks. The purpose of this model is to predict whether an existing customer will purchase the

next marketing campaign or not, based on information provided by the purchase behavior variables. We have applied SVM as a classifier for the modeling purpose.

From the company database, customers' past purchase behavior and campaigns data were obtained. This allowed us, in close cooperation with domain experts and guided by the extensive literature, to derive all the necessary purchase behavior variables (RFM variables) for a total sample size of 30,000 customers. In the feature construction step, the target variable and all the necessary purchase behavior variables (RFM) were constructed. Feature selection was done in three steps using F-score (filter method) and RF (wrapper method). The dataset was partitioned into training and test sets for performance evaluation. The data were highly unbalanced. The response rate was 7%. Thus for balancing the training set undersampling the nonrespondent class (majority class) was used. The data were also normalized and scaled before presenting it to the SVM to ease mathematical calculations as well as reduce the effect of larger attributes. The training set was used to build the SVM model and then the test set was used to evaluate the accuracy of model.

Performance of the SVM model was measured using Accuracy, TPR, TNR, weighted accuracy, and lift/gain chart. The overall model accuracy on the test set is 81% with 57% true positives rate and 83% true negatives rate. The lift chart was created by sorting all the prospects in descending order according to their likelihood of responding as predicted by the model. The lift chart shows that, for example, if only top 20% of the customers are mailed, then 57% of the respondents are approached as opposed to only 20% in a randomly targeted mailing of the same size. Actually by contacting only 20% of the customers on the basis of the predictive model the company will reach three times as many respondents as if they use no model. In conclusion, the predictive response model helps the company to identify a subset of customers who are more likely to respond than others and establish a direct relationship with them. By using this model, companies can not only reduce the overall marketing cost significantly but also prevent irritating the customers and improve customer relationship management. This facilitates the process of building long-term and strong relationship between businesses and their customers. In addition, understanding customers' behavior can help managers to plan more effective direct marketing campaigns. That is to say, data mining is a very important technology for efficient direct marketing (Chen, Chiu & Chang 2005); (Wang and Hong, 2006).

References

Bauer, C.L., 1988. A direct mail customer purchase model. J. Direct Mark. 2, 16–24.

Bentz, Y., Merunkay, D., 2000. Neural networks and the multinomial logit for brand choice modeling: a hybrid approach. J. Forecast. 19 (3), 177–200.

Blum, A., Langley, P., 1997. Selection of relevant features and examples in machine learning. Artif. Intell. 97, 245–271.

Bounds, D., Ross, D., 1997. Forecasting customer response with neural networks. Handbook Neural Comput. G6 (2), 1–7.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.
Berry, M.J.A., Linoff, G.S., 2004. Data Mining Techniques for Marketing, Sales, and Customer Relationship Management, second ed. Indianapolis Publishing Inc., Indiana.
Bose, I., Chen, X., 2009. Quantitative models for direct marketing: a review from systems perspective. Eur. J. Oper. Res. 195 (1), 1–16.
Cabena, P., Choi, H.H., Kim, I.S., Otsuka, S., Reinschmidt, J., Saarenvirta G., 1999. Intelligent Miner for Data Applications Guide. IBM Redbooks, SG24-5252-00.
Candade, N.V., 2004. Application of Support Vector Machines and Neural Networks in Digital Mammography: A Comparative Study. University of South Florida, Master Thesis.
Chen, M., Chiu, A., Chang, H., 2005. Mining changes in customer behavior in retail marketing. Expert Syst. Appl. 28 (4), 773–781.
Cullinan, G.J., 1977. Picking them by their Batting Averages' Recency-Frequency-Monetary Method of Controlling Circulation, vol. 18(1). Manual release 2103, Direct Mail/Marketing Association, New York, pp. 63–72.
Deichmann, J., Eshghi, A., Haughton, D., Sayek, S., Teebagy, N., 2002. Application of multiple adaptive splines (MARS) in direct response modeling. J. Interact. Mark. 16 (4), 15–27.
Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. J. Mach. Learn. Res. 1157–1182.
Ha, K., Cho, S., MacLachlan, D., 2005. Response models based on bagging neural networks. J. Interact. Mark. 19 (1), 17–30.
Han, J., Kamber, M., 2006. Data Mining: Concepts and Techniques, San Francisco. Morgan Kaufman Publishers, USA.
Haughton, D., Oulabi, S., 1997. Direct marketing modeling with CART and CHAID. J. Direct Mark. 11 (4), 42–52.
John, G.H., Kohavi, R., Pfleger, K., 1994. Irrelevant features and the subset selection problem. In: Cohen, W.W., Hirsh, H. (Eds.), Machine Learning: Proceeding of the Eleventh International Conference (ICML), (New Brunswick, New Jersey, July 10–13, 1994). Morgan Kaufmann Publisher, San Francisco, CA, pp. 121–129.
Kestnbaum R.D. (1992) 'Quantitative database methods', In: The direct marketing handbook, New York, McGraw Hill, pp. 588_597.
Kim, Y., Street, N., 2004. An intelligent system for customer targeting: a data mining approach. Decis. Support. Syst. 37 (2), 215–228.
Lecun, Y.J.L., Bottou, L., Brunot, A., Cortes, C., Denker, J.S., Drucker, H., Guyoin, I., Muller, A., Sackinger, E., Simard, P., Vapnik, V., 1995. Comparison of learning algorithms for hand written digit recognition. In: Gallinari, P., Fogelman, F. (Eds.), International Conference on Artificial Neural Networks, Paris, pp. 53–60.
Ling, C.X., Li, C., 1998. 'Data mining for direct marketing: Problems and solutions', Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining (KDD-98). ACM, New York, pp. 73–79.
Malthouse, E.C., 2001. Assessing the performance of direct marketing scoring models. J. Interact. Mark. 15 (1), 49–62.
Moutinho, L., et al., 1994. Neural networks in marketing. In: In Moutinho, L. (Ed.), Computer Modeling and Expert Systems in Marketing. Routledge, New York, pp. 191–212.
Nash, E.L., 1994. Direct Marketing: Strategy, Planning, Execution, third ed. McGraw Hill, New York.
Osuna, E., Freund, R., Girosi, F., 1997. Training support vector machines: an application to face detection. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 130–136.
Ou, C., Liu, C., Huang, J., Zhong, N., 2003. One Data mining for direct marketing. Springer-Verlag Berlin Heidelberg. 491–498.
Potharst, R., Kaymak, U., Pijls, W., 2002. Neural networks for target selection in direct marketing. In: Smith, K., Gupta, J. (Eds.), Neural Networks in Business: Techniques and Applications. Idea Group Publishing, London, pp. 89–110.
Rygielski, C., Wang, J.C., Yen, D.C., 2002. Data mining techniques for customer relationship management. Technol. Soc. 24, 483–502.
Setnes, M., Kaymak, U., 2001. Fuzzy modeling of client preference from large data sets: an application to target selection in direct marketing. IEEE Trans. Fuzzy Syst. 9 (1), 153–163.

Shin, H.J., Cho, S., 2006. Response modeling with support vector machines. Expert Syst. Appl. 30 (4), 746–760.
Stone, B., 1984. Successful direct marketing methods. Crain Books, Chicago.
Van Den Poel, D., 2003. Predicting Mail-Order Repeat Buying: Which Variables Matter? Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium 03/191.
Viaene, S., Baesens, B., Van den Poel, D., Dedene, G., Vanthienen, J., 2001a. Wrapped input selection using multilayer perceptrons for repeat-purchase modeling in direct marketing. Int. J. Intell. Syst. Account. Finance Manage. 10, 115–126.
Viaene, S., Baesens, B., Van Gestel, T., Suykens, J.A.K., Van den Poel, D., Vanthienen, J., 2001b. Knowledge discovery in a direct marketing case using least squares support vector machines. Int. J. Intell. Syst. 16, 1023–1036.
Yang, J., Honavar, V., 1998. Feature subset selection using a genetic algorithm. In: Motoda, H., Liu, H. (Eds.), Feture Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Dordecht, pp. 117–136.
Yin, R.K., 1994. Case Study Research, Design and Methods, Second ed. Thousand Oaks, Sage Publications, Inc., California.
Yu, E., Cho, S., 2006. Constructing response model using ensemble based on feature subset selection. Expert Syst. Appl. 30 (2), 352–360.
Wang, H., Hong, W., 2006. Managing customer profitability in a competitive market by continuous data mining. Ind. Mark. Manage. 35 (6), 715–723.
Zahavi, J., Levin, N., 1996. Segmentation analysis with managerial judgment. J. Direct Mark. 10 (3), 28–47.
Zahavi, J., Levin, N., 1997. Applying neural computing to target marketing. Journal of Direct Marketing. 11, 76–93.
Zikmund, W., 2003. Essential of Marketing Research. Thompson South-Western, USA.

CHAPTER 7

Caravan Insurance Customer Profile Modeling with R

Mukesh Patel*, Mudit Gupta†

**Adjunct Professor, Changa University of Science and Technology, Anand, India and Director, Dexter Consultancy Services, Ahmedabad, India*

†Analyst, Dun & Bradstreet, Bangalore, India

7.1 Introduction

Customer profiling for target marketing, particularly in the insurance sector, is an important area for potential application of data mining techniques and analytics. The analysis presented here is based on information about existing/current customers of an insurance company. In particular, our aim is to derive a typical profile of customers holding Caravan (Motor Homes) Insurance policy: Such a profile (or predictive model) would help identify customers with a similar profile who do not have caravan policies. This information can then be used for a much more focused marketing of caravan policies to both existing nonholders of caravan policies or new customers with similar profiles.

The obvious approach would be to profile current caravan insurance buyers, but this is not possible as the incidence of Caravan Insurance policy customers is very low as can be seen in Table 7.1. This fact indeed was the main challenge in developing a model for accurate prediction of potential target customers. The focus of our analysis was to model the binary (discrete) outcome (dependent variable), that is, to model a set of independent variables (i.e., customer related parameters such as socio-economic group, lifestyle, type of insurance policies held, etc.) such that it has a statistically significant power to predict a discrete outcome (i.e., Caravan Insurance holders). In this case, whether a certain type of existing customer profile is significantly more likely to be a holder of Caravan Insurance policy (1) or not (0).

In the rest of this chapter, we describe the process of developing customer profile models using R. In Section 7.2, we review the data and describe key results of preliminary and exploratory analysis. In Section 7.3, we describe the main steps in the model building approach. In Section 7.4, we conclude with a discussion of the data mining approach used, and how it

Data Mining Applications with R

Table 7.1 Training and Validation Dataset Split

	Number of Customer Profiles	% of Caravan Ins. holders
Total	9822	5.96
Training Dataset	5822	5.98
Validation Dataset	4000	5.95

can be refined to build models with improved predictive power. Based on insights provided by the models developed, we conclude with recommendations for a marketing approach designed to increase sales of caravan insurance.

7.2 Data Description and Initial Exploratory Data Analysis

The available data are from "Coil data mining competition 2000" which was downloaded from http://www.liacs.nl/~putten/library/cc2000/. The complete dataset has 9822 rows and 86 column headings.

The data contained a range of information on customers, which included income, age range, vehicle ownership, number of policies held, and level of contributions (premiums) paid as well as more qualitative information on lifestyle and type of households.

Before carrying out any further analysis, data were randomly split into two datasets: A Training dataset with 5822 records and a Validation dataset with 4000 records (see Table 7.1), for which the following R commands were used:

```
>describe(dataset1[145]==1)
>describe(dataset1[145]==0)
```

The Training data set was used for all the following descriptive and exploratory data analysis, as well as for models described in Sections 7.3.3–7.3.6. The Validation dataset was retained for validation of final models, results of which are given in Sections 7.3.7 and 7.3.8.

Below we describe each step of the preliminary exploratory analysis together with relevant R commands and results.

Step 1—Loading the data in R: excel format data converted into .csv format. For importing .csv file into R the following command was used:

```
>dataset1<– read.csv(file="C:/Documents and Settings/fool/Desktop/ticdata2000lrn.csv")
```

Step 2—To describe the dataset summary, matrix the following R command was used:

```
>summary(dataset1)
```

Step 3—To determine whether a column variable is a categorical value with factors (levels) or nominal values, the following R command was used:

```
>str(dataset1)
```

The results of the above two analyses are summarized in Table 7.2 which shows frequency count of nominal and categorical variables and the overall range of possible values.

Table 7.2 Total Frequency of Nominal and Categorical Variables

Variable Type	Number	Levels
Categorical	5	1-41
Nominal	81	0-10
Total	86	1-41

Please refer to Appendix A for full details of the dataset type including column headings and subheadings of the five categorical variables.

An initial review of the data revealed that for the religion parameters those customers who are Roman Catholic are further subdivided into 10 different levels with a range of percentage values (see Table Main Table and Sub Table L3 in Appendix A highlighted in grey). Despite various attempts to get more information about what such percentage value actually represents for a parameter that normally would have a simple binary value—that is, either someone is a Roman Catholic or not—we were unable to come up with a credible explanation. Given this lack of clarity, it was decided to exclude all four religion-related parameters (i.e., MGODRK—Roman Catholic; MGODPR—Protestant; MGODOV—Other religion; MGODGE—No religion) from further analysis presented here. However, before doing so, we carried out a correlation analysis to ensure that these four religion parameters are not significantly correlated with Caravan Insurance holders. For correlation between customers' religion associated with Caravan Insurance, the following R command was used:

```
>cor(dataset1[1:85],dataset1[86], method="spearman")
```

As can be seen from Table 7.3, the correlations between customers' religion and holding of Caravan Insurance is not very significant. Hence, on the basis of this all four religion parameters have been excluded from further analysis presented here. This reduced the total number of variables to 82 (from the original 86) summarized in Table 7.4.

Before carrying out any further exploration of the Training dataset, each of the remaining four categorical variables was "unfolded" (i.e., converted to a binary variable). For instance, the first categorical variable, MOSTYPE, which defines "Customer Subtype" had 41 levels such as "Career and Childcare" or "Middle Class Family" (as shown in Table L0 in Appendix A). Each subtype is treated as a separate variable with a binary

Table 7.3 Correlation Between Religion and Caravan Insurance Holding

Religion	Caravan Policy Holder's Spearman Coefficient
Romans Catholic	0.02
Protestant	0.03
Other religion	0.00
No religion	−0.04

Table 7.4 Frequency of Nominal and Categorical Variables Excluding Religion

Variable Type	Number	Levels
Categorical	4	1-41
Nominal	78	0-10

value indicating whether a customer belongs to that subtype or not. This increased the total number of variables from 82 to 145.

In the next subsection (7.2.1), we present further exploratory analysis of the data which includes simple correlations between variable and multiple regression analysis.

7.2.1 Variable Correlations and Logistic Regression Analysis

Step 4—The training dataset with 145 variables was subjected to further exploratory data analysis. The following R commands were used to get an overview of the value of each customer profile variable:

```
>library(Hmisc)
>describe(dataset1)
```

This analysis provided detailed information on the nature of the customer profile data. In particular, it gives a clear indication of the number of customers who fall into each subtype of the four main categorical variables. For example from the detailed Table (B1 in Appendix B), it is clear that the distribution of customers in each customer subtype is not uniform. It is interesting to note that there are no customers in "Junior Cosmopolitan" or "Students in apartment" subtype, a fact that is somewhat backed-up by the relatively very low number of customers in the "20-30" age group. See Table B2 in, Appendix B for values for all other variables.

Step 5—To see if there is any particularly strong relationship between customer parameters and Caravan Insurance holding we decided to review the proportion of caravan insurance holder vis-à-vis each of the remaining (independent) variables. The following R command was used for this analysis:

```
>prop.table(table(MOSTYPE, CARAVAN), 1)
```

The results did not elicit any strong link between Caravan Insurance holding and any other specific variable as can be seen from the complete results in Appendix C. Table 7.5 below lists variables with a relatively high association (i.e., proportion of 0.10 and above) with Caravan Insurance holders. Though the variables in this list are of some interest, the correlation between each pair is unlikely to be statistically significant. Overall, the results also confirmed the relatively low number of Caravan Insurance Policy holders in the database (as mentioned in Section 7.1).

Table 7.5 Variable with Higher Association with Carvan Policy Holders

Customer Profile	Proportion of Caravan Policy Holders
Average income	0.10
Cont. boat policies	0.11
Cont. car policies	0.16
Number of car policies	0.15
Number of boat policies	0.11
Middle class families	0.10

Following the exploration of dependencies between variables, we carried out a logistic linear regression analysis as follows:

```
>summary(cust.logit)

Call:
glm(formula = CARAVAN~., data = train_3)
Deviance Residuals:
Min          1Q    Median    3Q       Max
−0.70314 -0.08721 -0.04517 -0.00434 1.03944
```

Table 7.6 lists significant variables with 95% or above confidence levels (t-value > 1.64).

It is interesting to note that all but one variable refer to either the number or the contribution (premiums) of policies held by customers. This implies that Caravan Insurance holders are highly likely to hold other insurance policies. This finding is hardly surprising and not particularly insightful because the data *is* of existing customers.

Also noteworthy is that those customers with Boat and Surfboard Policies and high incomes (above 125,000) are significantly more likely to have Caravan Policies, which perhaps indicates a customer segment focused on outdoor activities. Though the inclusion of "lower-level education" is, as we will see below, an interesting indicator of another group that is also significantly likely to hold Caravan Insurance.

In the next section, Section 7.3, we describe our customer profile modeling approach and the results of four types of classifier modeling methods. This is followed by results comparing the relative performance of each model. In Section 7.4, we discuss the results and consider possible further analyses likely to lead to models with better predictive power and/or more insightful results.

7.3 Classifier Models of Caravan Insurance Holders

7.3.1 Overview of Model Building and Validating

The exploratory data analysis and logistic regression (LR) model reported in Section 7.2 provided interesting results, but no novel insights that can have a significant impact on target marketing for Caravan Insurance. In other words, in many ways they highlight the obvious; that

Table 7.6 Logistic Linear Regression Analysis

Description of variable	Data Label	Estimate	Std. Error	*t*-Value	Pr(>|t|)	Sig Lev
Intercept		1.33×10	4.82×10^{-1}	2.76	0.01	0.001
Number of boat policies	APLEZIER	3.57×10^{-1}	8.88×10^{-2}	4.03	0.00	0
Cont. car policies	PPERSAUT	1.03×10^{-2}	2.65×10^{-3}	3.88	0.00	0
Cont. fire policies	PBRAND	1.16×10^{-2}	3.63×10^{-3}	3.20	0.00	0.001
Number of life insurances	ALEVEN	4.15×10^{-2}	1.55×10^{-2}	2.68	0.01	0.001
Cont. family accidents	PGEZONG	1.94×10^{-1}	7.95×10^{-2}	2.44	0.01	0.01
Cont. disability insurance	PWAOREG	5.49×10^{-2}	2.59×10^{-2}	2.12	0.03	0.01
Number of surfboard policies	AZEILPL	5.03×10^{-1}	2.82×10^{-1}	1.78	0.07	0.05
Lower-level education	MOPLLAAG	-1.32×10^{-2}	7.43×10^{-3}	−1.78	0.08	0.05
Income >123,000	MINK123M	-1.37×10^{-2}	6.96×10^{-3}	−1.97	0.05	0.01
Number of family accidents ins.	AGEZONG	-4.08×10^{-1}	1.90×10^{-1}	−2.15	0.03	0.01
Number of private third party	AWAPART	-2.81×10^{-1}	1.21×10^{-1}	−2.32	0.02	0.01
Cont. private 3rd party ins (50-99)	PWAPART_2	-2.19×10^{-1}	9.06×10^{-2}	−2.41	0.02	0.01
Cont. private 3rd party ins (1-49)	PWAPART_1	-2.40×10^{-1}	9.21×10^{-2}	−2.61	0.01	0.001
Cont. life insurances	PLEVEN	-1.77×10^{-2}	6.52×10^{-3}	−2.71	0.01	0.001
Cont. private third insurance (0)	PWAPART_0	-5.20×10^{-1}	1.91×10^{-1}	−2.72	0.01	0.001

is, customers with more polices and (therefore) contributions and the time and money for leisure activities are more likely to also hold Caravan Insurance policies. Although such information can be useful for targeting potential customers, nevertheless, it is fairly predictable.

To fulfill our objective of seeking out a nonobvious customer profile, it was decided to construct or learn predictive models. The expectation was that insights provided by

such models would be useful to develop a more nuanced marketing campaign aimed at a customer profile (i.e., buyers) that would have not been otherwise targeted.

Given the large number customer profile parameters, and depending on the modeling methodology and approach used, it is possible to develop a number of such models that perform equally well. For this reason, it was decided that each model will be checked for performance on more than one criteria such as efficiency, accuracy, and computation (prediction) times. More precisely, to determine a models optimal performance we applied two criteria.

One is on the basis of the generally accepted convention that if two models have the same predictive power (that is, they perform equally well in identifying target customers) then the one with the least number of variables should be considered to be superior. Such a model is referred to as the most parsimonious in that it will be less complex, which for our purposes is defined as explaining or accounting for variability with fewer independent variables. In addition, such a model with fewer variables is arguably more robust as apart from correct predictions it is likely to identify fewer false positives.

Efficiency is our second criterion for comparing similar models. It is usually measured in terms of performance over time and accuracy. How quickly a model indentifies a potential or target customer is obviously critical from a marketing and selling point of view. That a model should do so with accuracy can be just as critical. A model that is fast but also tends to have a high rate of wrong predictions (i.e., false positives) may be of less practical use than one that is slower but tends to predict correctly (i.e., has none or very few false positives) at a higher rate. More technically, our efficiency criterion checks for models that over-fit the data.

Ideally, an optimal model would be both parsimonious and efficient. As we will see here, however, that is not always the case, and particularly so, for data with a large number of variables. There is often an element of trade-off between these two criteria such that a less parsimonious model is also the one that is more efficient as compared to a more parsimonious model. In such situations, it is not so straightforward to determine optimal performance solely based on these two criteria. As we suggest in Section 7.4 in such circumstances model optimality evaluation should also take business purpose or practical aspects of the model utilization into account.

Essentially model building is a learning process based on the training dataset (which is increasingly referred to as Machine Learning, ML). In our case each model learned (or was trained) to correctly identify Caravan Insurance holder in the training dataset. The resulting models can also be regarded as models (or representation) of typical customer profiles of Caravan Insurance holders. The customer profile model would include a set (usually a subset) of independent variables that have statistically significant association with the dependent variable (that is, Caravan Insurance holders). The selected independent variables collectively explain or account for some proportion of variance in the data. This accounting for variance is

also often termed as "goodness of fit" of a data. During the learning phase, the main objective would to be to construct a model with the least number of independent variables that explains the most variance. A truly optimal model would explain all the variance with one independent variable, essentially reducing the model to an if-then rule that always predicts correctly. In practice, matters are rarely that simple: As we will see, our models while accounting for a similar proportion of variance, have varying numbers of independent variables thus indicating that a different modeling approach has an effect on the resulting predictive model.

Following the model building phase, each model's predictive performance is checked with a totally new dataset. For this, we used the validation dataset of 4000 customer profiles, which had been split from the dataset right at outset. Validation essentially involves each model attempting to correctly identifying members of two classes: Caravan Insurance holders and nonholders. During the validation phase, we also measure the model's accuracy in correct identification of target customer profiles and the time taken to do so. Together these measures enable us to determine the overall optimality of the models.

In the next subsection (7.3.2), after a brief review of classifier modeling method, we motivate our selection of four different modeling approaches. In the four subsequent subsections (7.3.3, 7.3.4, 7.3.5 and 7.3.6), each model building process is described in detail, concluding with a review of the best performing model of a typical customer profile of a Caravan Insurance holder.

7.3.2 Review of Four Classifier Methods

Following initial trials with several different modeling methods such as, clustering, inductive rules,, we felt that classifier modeling would be the most suitable for our objective of developing a model of a typical customer profile of likely holders or buyers of Caravan Insurance.

One reason for this was that classification methods typically output a class of independent variables that significantly affect the dependent variable. Often the independent variables can be useful for overall understanding of the customer profile being modeled. For instance, a classifier model that includes independent variables such as "young and rising," "affluent," "unmarried couples," and "rented accommodation" or another one that includes "middle class" with "double incomes" provide useful indicators of the type and nature of a target market segment.

Such clarity in models can often greatly facilitate business or marketing decision-making. For example, knowing that the target group is largely made up of young, affluent people living in rented apartments can be very useful when it comes to deciding not only the target market profile but also the type and style of communication, the medium, and even things like product positioning and the pricing model. At the same time, such information

can also be a basis for drawing conclusions such as the fact that if the affluent middle class is a target market then "rural" or "multigeneration households" need not be targeted. The classifier models are, therefore, ideally suited for business applications where customer or market segment profiling is the key requirement.

Apart from the advantage of clarity in terms of group or set membership (i.e., either an independent variable is in the class set or not) it also would allow us to carry out fine-tuning or further optimization by pruning variables. Another advantage is to be able to combine variables that can be regarded as similar (e.g., "Social Class A" with "affluent senior apartments" or "high-status seniors").

Right at the outset, we decided to use more than one classification method. The main reason for this was to add to overall certainty and confidence that the best performing classifier model was indeed so. Using more than one method enables comparison between models such that if they tend to include similar independent variables then one is more likely to be confident about the overall reliability of the best performing model (in terms of explanatory or predictive power).

The second reason for using more than one classifier method was more theoretical and technical. Given that R offered different methods, we felt that exploring a number of them would be insightful and deepen our understanding of the different approaches to building classifier models. After initial exploration of various classifier methods available in R, we decided to compare the following four methods:

1. Recursive Partitioning (RP)
2. Bagging Ensemble (BE)
3. Support Vector Machine (SVM)
4. LR classifier

In the next four subsections (7.3.3, 7.3.4, 7.3.5 and 7.3.6), each method is briefly described (with references for more details wherever appropriate). This is followed by details of the learning process together with R commands used for development of each type of classifier model. The significant independent variables included in each model is presented in histograms.

As the independent variables for each modeling method are selected or learned by different modeling methods, the between-model absolute values of independent variables vary quite a bit. For instance, the same variable selected by two (or more) different models may have very different values, and, as readers will note at times, this difference can be of more than an order of magnitude. To aid cross model comparison, normalization of values is one way of handling such between-model variations. However, given that each modeling method uses different techniques or criteria for selecting independent variables we felt that this would not be appropriate and therefore decided to leave the values as predicted or modeled.

Finally, each model is checked for performance; that is, correct prediction of instances of Caravan Insurance holders in the validation dataset (of 4000 as illustrated in Table 7.1). As we will see, the verification process provides interesting information about the goodness of fit for each model. These results are presented in the following two different ways.

First, as already stated above, for each analysis, the objective was to develop a class of independent variables that together were the best predictors of the dependent variable (i.e., Caravan Policy holders). Ideally, this class should also be the smallest set possible—that is, the model should be parsimonious—while also accounting for the most variance. This objective can also be stated in terms of the area under ROC (i.e., AUC) or the area under the optimal cost curve which is presented in Section 7.3.7. For this analysis, we used the package ROCR (Sing et al., 2009), which is a flexible tool for visual representation of proportion of variability accounted for by each type of classifier model.

Second, in Section 7.3.8, the results on the performance and accuracy of each model is presented. For the latter analysis, we used Precision-Recall, and Accuracy-v-Cut-off analysis to evaluate each model. As will be explained in more detail, Recall value is a measure of the proportion/percentage of total customers in the validation dataset that are correctly identified as Caravan Insurance holders, and Accuracy analysis indicates the probability of proportion/percentage of customers correctly identified as Caravan Insurance holder.

In the following four Sections (7.3.3–7.3.6), we describe each modeling approach in more detail together with the model learning or building process. The resulting predictive models derived from the training dataset are also presented in histograms.

7.3.3 RP Model

RP aims to minimize the loss function (that is the mean squared error) attributed to each variable at each split. Variables tabulated at each split with the best fit (i.e., least mean squared error) are selected for the next split. The recursive splits stop when the mean square error cannot be optimized any further. In addition, because there may be candidate variables that are important but are not used in a split, the top competing variables are also tabulated at each split. For more details on regression trees, see Breiman et al. (1984) which is the standard reference on both classification and regression trees. The result outputs of the following command for RP analysis are given and explained below:

```
>library(rpart)-[This function loads the package rpart (Therneau and Atkinson, 2010) in R.]
>library(caret) - [This function loads the package caret (Kuhn, 2012) in R, command varImp is
provided in this package to identify important variables corresponding to the method used]
>cust.rp<– rpart(CARAVAN~. , data=dataset1)
>cust.rp
```

$n = 5822$

Node	Split	*n*	Deviance	*y* val
1	root	5822	327.20	0.06
2	PPERSAUT < 5.5	3459	83.86	0.02 *
3	PPERSAUT >= 5.5	2363	232.95	0.10
6	PBRAND < 2.5	1151	70.11	0.07
12	PPLEZIER < 0.5	1143	64.83	0.06 *
13	PPLEZIER >= 0.5	8	1.50	0.75*
7	PBRAND >= 2.5	1212	158.15	0.15
14	MOPLLAAG >= 2.5	933	101.58	0.12 *
15	MOPLLAAG < 2.5	279	52.93	0.25 *

* denotes final leaf node.

This tree-based model selects relevant variables, which are shown in the R output sequence above (which represents a tree, splits, and leaf nodes). Only relevant and significant variables appear in the sequence. The way to "read" the sequence is to start at the top or root marked as 1 in R, with subsequent two rows denoting a two way split, which continues till a leaf node is reached denoted by an asterisk (*).

From the results listed above, we can see that based on 5822 records analyzed with an average frequency of CARAVAN (root) as 0.06 and the deviation (sum of squared difference from the average) from this average is 327.20. From this root node, there are two branches (marked by 2 and 3). Branch 2 is for cases which are true, i.e., PPERSAUT (Contribution car policies) >5.5 (3459 records) and the branch 3 is for cases which are do not satisfy the criteria (2363 records). From node 3, we have other branches marked as 6 and 7 in R according the criteria for PBRAND (Contribution fire policies) and so on.

To generate information about the variables actually used in tree construction, the following R command was used:

```
>printcp(cust.rp$cptable)
```

The results of variables actually used in tree construction are summarized in Table 7.7, in which the column headed "xerror" gives the estimates of cross-validated prediction error for different numbers of splits. For more detail, refer to webpage hyperlink to RP, July 2012.

Next, we generated RP model details of all significant independent variables and their relative importance by using the following R command.

```
>cust4.var.imp <- varImp(cust4.rp, useModel=rpart)
```

The results are presented in the histogram in Figure 7.1.

Table 7.7 Recursive Partitioning (RP) Model

	CP	nsplit	rel.error	xerror	xstd
root	0.032	0	1.000	1.000	0.049
MOPLLAAG (lower-level education)	0.014	1	0.968	0.969	0.046
PBRAND (Cont. Fire policies)	0.012	2	0.954	0.977	0.046
PPERSAUT (Cont. Car policies)	0.011	3	0.942	0.972	0.046
PPLEZIER (Cont. Boat policies)	0.010	4	0.931	0.972	0.046

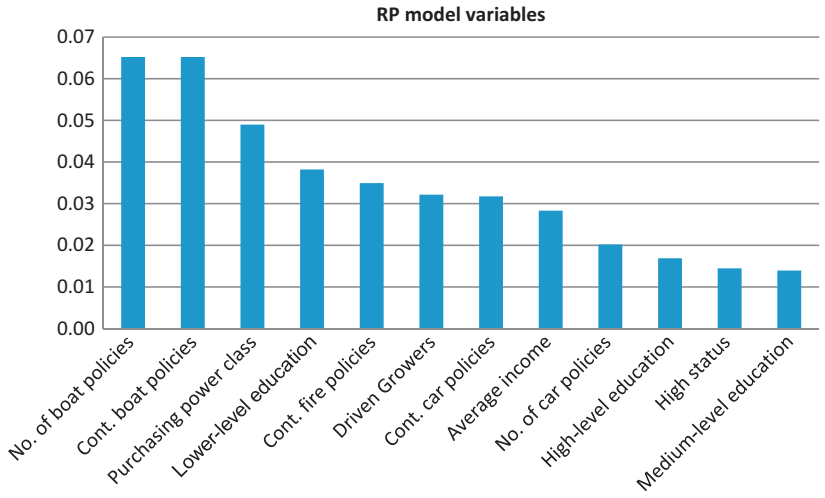


Figure 7.1
RP model significant variables.

Although there is quite a bit of overlap between the variables in the regression model and the RP model, it is interesting to see that customers with "lower-level education" and "Average Income" are significant variables. This indicates that apart from the more affluent group, those at the other end of the socio-economic scale are also more likely to be Caravan Insurance holders.

7.3.4 Bagging Ensemble

BE method is a combination of classifier and regression tree methods designed to stabilize the tree proposed by Breiman (1996a,b, 1998). Briefly, it works by splitting the data into multiple

(training) data sets to which a class of learning or optimizing methods—that is decision trees and neural networks—is applied. The method is training multiple (k) models on different sets and then averaging the predictions of each model, hence bagging. The goal is to develop a model that optimizes the accuracy of one model. The rationale is that averaging of misclassification errors on different data splits gives a better estimate of the predictive ability of a learning method. The Bagging algorithm steps are:

- Training—In each iteration i, $i = 1, \ldots, n$
- Random sampling with replacement N samples from the training set
- Training a chosen "base model" (here, regression decision trees because variables are numeric on the samples)
- Testing—For each test example
- Starting all trained base models
- Predicting by combining results of all i trained models:
  - Regression: averaging
  - Classification: a majority vote

The following is the R command sequence to run this analysis:

```
>library(ipred)-[To load the package ipred in R by Peters et al. (2002b)]
>cust.ip<- bagging(CARAVAN~. , data=dataset1, coob=TRUE)
>cust.ip.prob<- predict(cust.ip, type="prob", newdata=dataset2)
```

The output is:

```
Bagging regression trees with 25 bootstrap replications
Call: bagging.data.frame(formula = CARAVAN~., data = dataset1, coob = TRUE)
Out-of-bag estimate of root mean squared error: 0.2303
```

The records were split into 25 subsets of the training dataset. The average mean squared error is 0.23. In other words, it estimates the prediction error of the response variable CARAVAN.

Next, we generated RP model details of all significant independent variables and their relative importance using the following R command. The results are presented in the histogram in Figure 7.2:

```
>cust4.var.imp <- varImp(cust4.ip, useModel=bagging)
```

It is interesting that the BE model has the same significant independent variables as the RP. As we will see in Section 7.3.7, (Table 7.8 and ROC graph in Figure 7.5) their explanatory and predictive powers are also very similar. It would be interesting to explore the extent to which systematic similarities between the two classification methods can explain this outcome.

7.3.5 Support Vector Machine

An SVM is a (supervised) ML method for finding a decision boundary for classification of data. An SVM training algorithm is applied to a training data set with information about the class that each datum (or vector) belongs to and in doing so establishes a hyperplane(i.e., a gap or

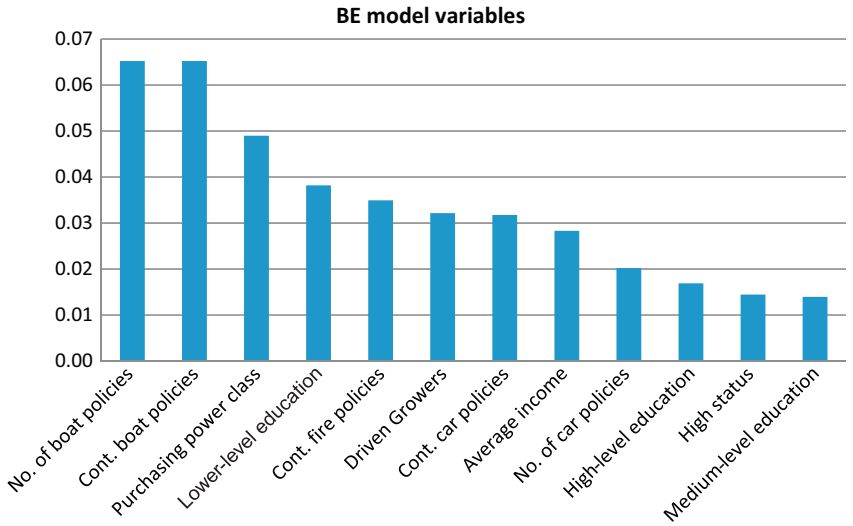


Figure 7.2
BE model significant variables.

Table 7.8 Summary Table of AUC of Classifier Models

Modeling Method	No. Ind. Variables	AUC the ROC Curve	Criteria for Selecting Variables
Recursive Partitioning (RP)	12	0.68	Mean squared error at each split
Bagging Ensemble (BE)	12	0.68	Prediction by combining results of trained models-Regression, classification
Support Vector Machine (SVM)	67	0.66	Margin maximization and feature weight greater than 1
Logistic Regression (LR)	28	0.72	Absolute *t*-value statistic

geometric margin) separating the two classes. During the model learning phase, the SVM aims to optimize the width of the gap (i.e., the maximum margin hyperplane) between classes. The resulting model can then be used to determine whether a (new) data vector is a member of a class (or not).

SVM's are highly appropriate in finding (or learning) nonlinear class boundaries represented as linear models: The maximum margin hyperplane is thus a linear model that represents a nonlinear decision boundary in the original space constructed. "The training examples that are closest to the maximum margin hyperplane are called support vectors. All other training examples are irrelevant for defining the binary class boundaries." Kim (2003).

There are four known implementations of SVM in R, namely, package e1071 by Dimitriadou et al. (2011), package kernlab by Karatzoglou et al. (2004), package klaR by Weihs et al. (2005), and package svmpath by Hastie (2004). Here, we have used Dimitriadou et al. (2011) package e1071 implementation of R using the function svm().

For full details of this modeling approach, see Chang and Lin (2001), Kim (2003), and Dimitriadou et al. (2011); more advanced consideration of SVMs can be found in Vapnik (2000) and Shawe-Taylor and Cristianini (2000).

The following R command code was used to generate the SVM model:

```
>library(e1071) [To load the package e1071 by Dimitriadou et al. (2009)]
>cust.svm<-svm(CARAVAN~.,data=dataset1,method="C-classification", kernel="radial",
cost=10,gamma=0.1,cross=0,fitted=TRUE, probability=TRUE)
```

Next, we generated SVM model details of all significant independent variables and their relative importance with the following R command. The results are presented in the histograms in Figures 7.3a-e. R commands for variable importance SVM model were:

```
>cust.svm.feature.weights = t(cust.svm$coefs) %*%cust.svm$SV [this calculates the feature
weights]
>cust.svm.feature.weights
```

The first thing to note about the SVM model is the significant increase in the number of independent variables; 67 variables as against the more modest 12 for the RP and BE models. In this respect, the SVM model cannot be regarded as particularly parsimonious. Further, despite the increase in the number of variables, its explanatory power is marginally less than that for RP and BE models (as can be seen from the Table 7.8 and ROC curve Figure 7.5 in Section 7.3.7).

It is clear from Figures 7.3a-e that the SVM model also includes significant variables that are similar to that for the RP and BE models. In particular, it is notable that "Contribution to Car" and "Contribution to Boat" policies are highly significant predictors of the dependent variable (i.e., Caravan Insurance holders) as are "Low level of education" and "Skilled laborers." These results once again suggest that that there are two distinct subgroups of actual and potential Caravan Insurance policyholders. One, that is more affluent with perhaps more leisure time denoted by "Social Class A" and "Income 75-122,000"; and, another less so with lower income denoted by "Low level of education" and "Skilled laborers" as well as "Social Class C" and "Average Income."

7.3.6 LR Classification

This function takes a regression model (with continuous value variables) and renders it in a classifier model (i.e., applying class labels to variables). This involves transforming the output of a linear regression by using a logit link function that can vary on a scale of (0, 1), so the log odds can vary on the scale of (−1, 1). The inverse of logit, logistic function will map any value

to a proportionate value between 0 and 1. For our analysis, the absolute value of the *t* statistic for each model parameter is used.

For further details, refer to the Logistic document available at the Stanford University website (Manning, 2007, accessed July 2012). Further details on Generalized linear models can be found in Dobson (1990), Hastie and Pregibon (1992), and McCullagh and Nelder (1989). Here,

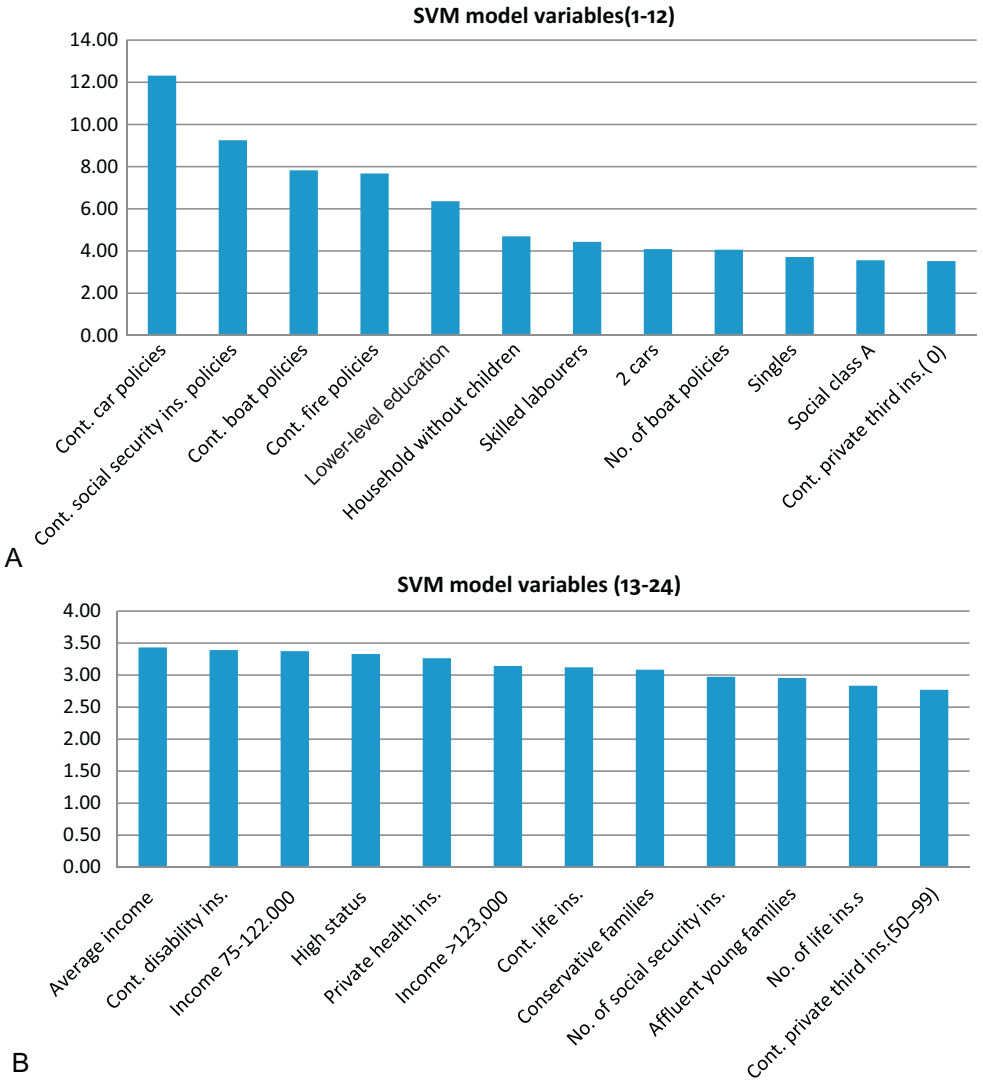
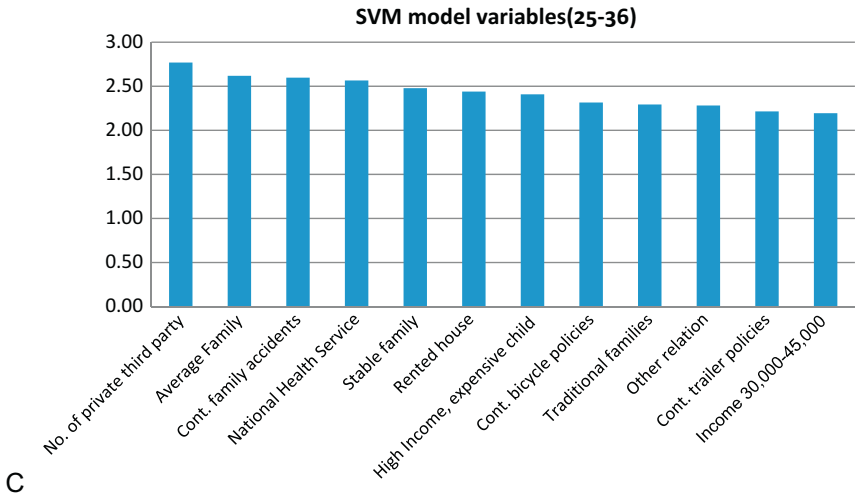


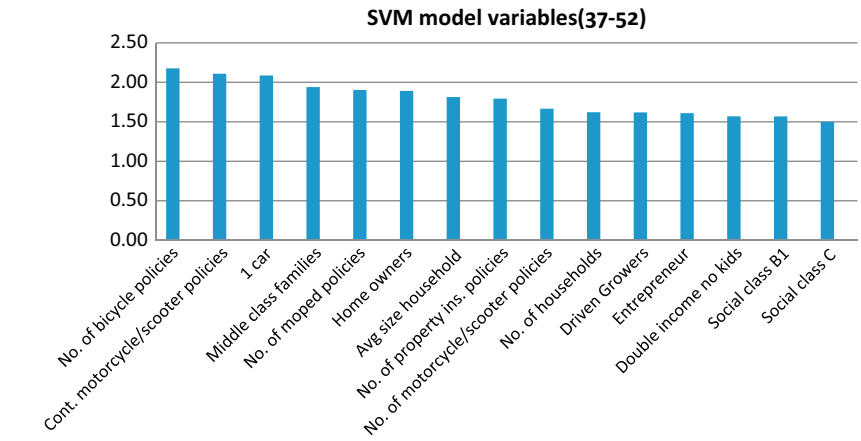
Figure 7.3

(a) SVM model significant variables (1-12). (b) SVM model significant variables (13-24).

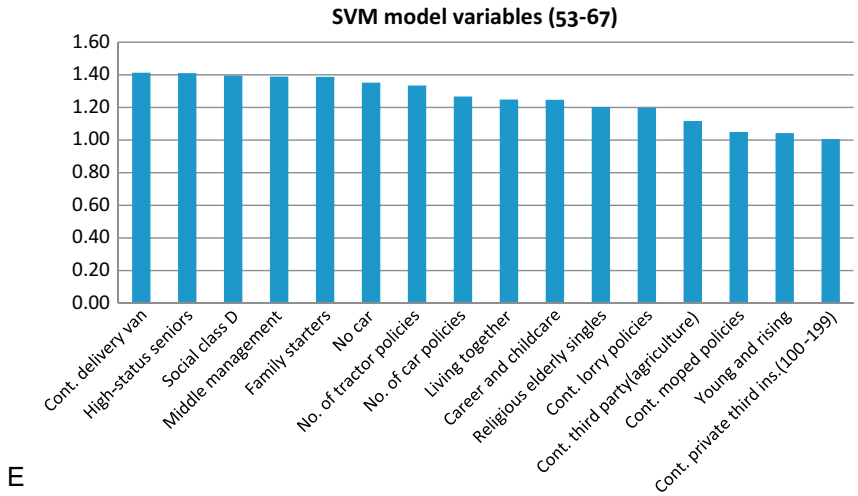
Continued



C



D



E

Figure.7.3—Cont'd(c) SVM model significant variables (25-36). (d) SVM model significant variables (37-52). (e) SVM model significant variables (53-67).

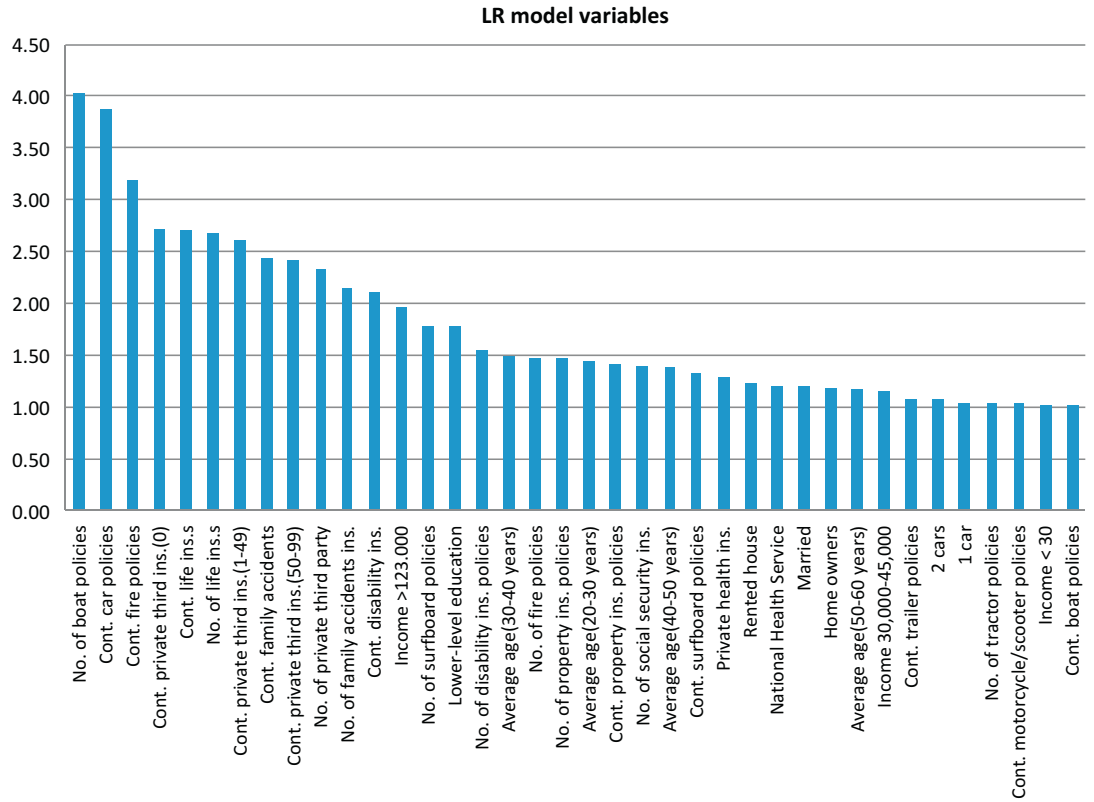


Figure 7.4
LR model significant variables.

we use the generalized linear model[1] (glm) as the variables are categorical and the output variable is also discrete. The R commands for this analysis are:

```
>cust.logit<– glm(CARAVAN∼. , data=dataset1, family=binomial(link="logit"))
>summary(cust.logit)
```

The full results are tabulated in Appendix D, and are used for building the LR model with the following R command:

```
>cust.var.imp<– varImp(cust.logit, useModel=glm)
```

The significant variables included in the LR model are illustrated in the histogram in Figure 7.4.

[1] The original R implementation of glm was written by Simon Davies working for Ross Ihaka at the University of Auckland but has since been extensively rewritten by members of the R Core team. The design was inspired by the S function of the same name described in Hastie and Pregibon (1992).

The first thing to note about the LR model is the inclusion of insurance related variables (either number or contribution and to which we will return to in Section 7.4). In that respect, the LR model is similar to the SVM. It is also of some interest that more or less all the age range-related variables are included in this model. However, the key result is the continued inclusion of two sets of variables that suggest two distinct groups of Caravan Insurance holders. Once again this model includes variables such as "Number of boat policies" and "Income >123,000" as well as "lower-level education" and "Income <30."

In the next subsection, we present a summary table of the goodness-of-fit for each classifier model. These results are presented in graphic format as ROC curves for area under curve (AUC) analysis for each model. This measure shows how well each model performs in terms of correct identification of instances of Caravan Insurance holders in the validation dataset, which is a generally accepted indication of their predictive power.

7.3.7 Comparison of Four Classifier Models: ROC and AUC

Here we present prediction results of all four classifier models described in Sections 7.3.3–7.3.6. As already explained each model attempts to correctly identify existing Caravan Insurance holders in the validation dataset of 4000 customer profiles. The overall predictive performance of each model is presented as an ROC curve for AUC. In the next subsection, we present Accuracy-v-Cut-off performance of each model and review the overall computation time for each model's validation process. Together the results of these analyses are essentially descriptive indicators of the relative predicitve power and goodness-of-fit to the data of each model. As already explained above, given the very different approaches (summaried in Table 7.8) of classification methods used it was not feasible to carry out formal comparative analysis of each model because the parameter values are not comparable across models.

For the ROC curve analysis, we used the package ROCR (Sing et al., 2009), which is a flexible tool for visual representation of evaluation metrics of classifiers. Details of R commands for computation of individual ROC curves for each model using the validation dataset are given in Appendix E. The combined ROC analysis and plots for four models are presented in Figure 7.5 and were generated with the following R commands:

```
>ppi<- 300
>png(filename="ROC curve without religion variables.png", width=6*ppi, height=6*ppi,
res=ppi)
>plot(cust.rp.perf, col=2, main="ROC curve without religion variables")
>legend(0.5,0.5, c('rpart','bagging','svm', 'logistic'), 2:5)
>plot(cust.ip.perf, col=3, add=TRUE)
>plot(cust.svm.perf, col=4, add=TRUE)
>plot(cust.logit.perf, col=5, add=TRUE)
>dev.off()
```

The AUC of each model's ROC presented in Table 7.8 is computed with the following R commands:

```
>cust.rp.perf.auc<–performance(cust.rp.prob,.rocr, 'auc')
>cust.ip.perf.auc<– performance(cust.ip.prob.rocr, 'auc')
>cust.svm.perf.auc<– performance(cust.svm.prob.rocr, 'auc')
>cust.logit.perf.auc<– performance(cust.logit.prob.rocr, 'auc')
```

As can be seen from Figure 7.5, LR is by far the best at modeling the data. The RP and BE models are the next best (represented in the graphs by the single curve as their predictive performance is very similar). The SVM model performance is not as good as that of the other three models. This is notable given that it includes many more variables than the other three models. This outcome may be because of the over-fitting of data by the SVM. However, further analysis would be necessary to validate this conclusion.

Based on the ROC curves, it would be reasonable to conclude that there is not much to choose between the four models. However, considering the AUC values given in Table 7.8, it is possible to identify models that are more parsimonious and/or those that perform better in terms of explaining or fitting the validation dataset.

As mentioned above, apart from AUC (i.e., the area under the ROC that is explained or accounted for by a particular model), the relative optimality of each model can also be compared in terms of the number of significant variables included in the model (and therefore contributing toward the predictive power). In general, it is true that all selected variables make a legitimate (i.e., statistically significant) contribution toward explaining

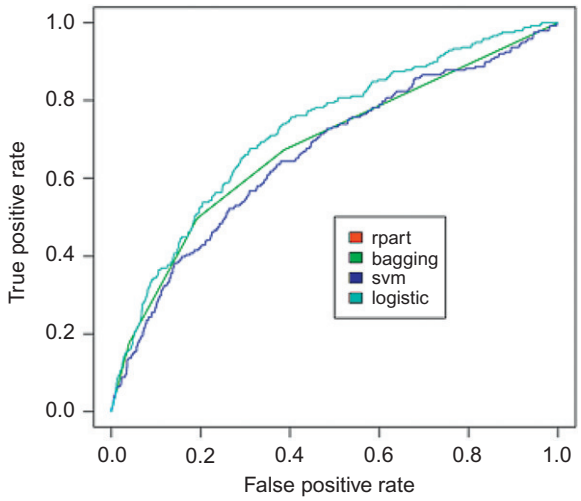


Figure 7.5
ROC curve of classifier models.

the data or predicting the dependent variable but models that do so with fewer variables are generally regarded as better. The main reason is that such parsimonious models are less complex and more elegant requiring few computational resources (all other things being equal). The reduced level of complexity is particularly desirable where increased complexity does not lead to a significant increase in the model's goodness-of-fit or predictive power.

So with respect to parsimony, it is clear from Table 7.8 that the RP and BE models are preferable and are almost alike in terms of number of variables (12 each) and explanatory power (accounting of 0.68 AUC). The LR model with 28 variables is the next most parsimonious though it does also account for higher proportion of AUC (0.72) as compared to RP and BE models. The least parsimonious SVM model with 67 variables also accounts for the least proportion of AUC (0.66). Having said that, it is unlikely that the relative differences in AUC of models is likely to be statistically significant. Therefore, on the basis of the variables involved the RP and BE are the best performing models at in terms of predictive power.

In the next subsection, we present results of Precsion-Recall, and Accuracy-v-Cut-off analysis which provide an indication of the relative precision and accuracy of prediction of each model. These results, together with the computational times, will be our basis for the evaluation of the relative merits of different models.

7.3.8 Model Comparison: Recall-Precision, Accuracy-v-Cut-off, and Computation Times

Here, we compare and evaluate each model in terms of how good they are at predicting likely Caravan Insurance holders in the validation dataset. Given that the four models were built using four different algorithms, it was felt that the most appropriate evaluation metrics are Precision-Recall (or Recall) and Accuracy-v-Cut-off. Together they measure not only the success rate of correct identification of the target (i.e., Caravan Insurance holders) but also the probability of accuracy maximization.

Recall values indicate the rate of correct identification of the target (Caravan Insurance holder), which is the same as the true positive rate (TPR) on the ROC curve presented in the previous subsection (7.3.7). To derive a measure of precision, the TPR is calculated as a fraction of the total number of true positives (i.e., all Caravan Insurance holders in the validation dataset). Recall analysis of models is particularly appropriate for skewed datasets, such as ours, that have a relatively low frequency of Caravan Insurance holders.

The Recall measure is appropriate in situations where accuracy combined with speed is valued. Examples of this include online real-time processing of data to support dynamic decision-making in a business scenario.

The following R commands were used to carry out the recall analysis for all models:

```
>cust.rp.perf.cr <- performance(cust5.rp.prob.rocr, "rec", "rpp")
>cust.ip.perf.cr <- performance(cust5.rp.prob.rocr, "rec", "rpp")
>cust.svm.perf.cr <- performance(cust5.rp.prob.rocr, "rec", "rpp")
>cust.logit.perf.cr <- performance(cust5.rp.prob.rocr, "rec", "rpp")
>ppi<- 300

>png(filename=" Cumulative recall curve(without religion variables)",width=6*ppi,
height=6*ppi,res=ppi)

>plot(cust.rp.perf.cr, col=2, main="Cumulative recall curve(without religion variables)")
>legend(0.5,0.5,c('rpart','bagging','svm','logistic'),2:5)
>plot(cust.ip.perf.cr,col=3,add=TRUE)
>plot(cust.svm.perf.cr,col=4,add=TRUE)
>plot(cust.logit.perf.cr,col=5,add=TRUE)
>dev.off()
```

The Recall curves in Figure 7.6 illustrate the rate of positive recalls for each model. It indicates a cumulative measure (over time) of the accuracy of each model in terms of the rate of identifying a likely holder of Caravan Insurance in the validation dataset. It is obvious that all models perform more or less equally well in terms of overall accuracy. Bearing in mind the very similar proportion of AUC of each model, the similarity in Recall performance of all four models is to be expected.

Overall the LR model performs slightly better and particularly so at the later stage of the prediction process, by when the RP and BE models are clearly not performing as well. Also noteworthy is the manner in which the SVM model underperforms during the initial stage of the prediction process only to improve toward the end.

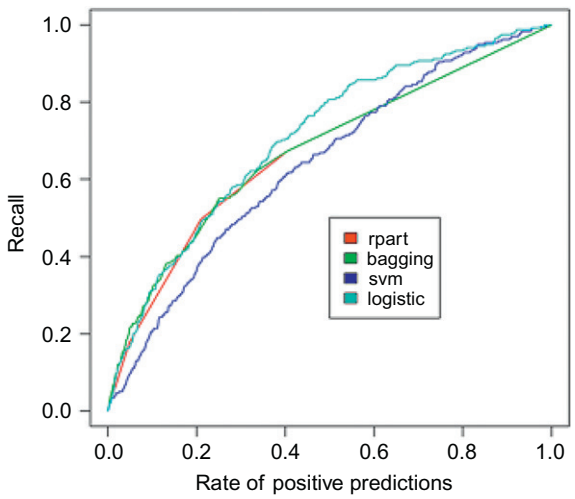


Figure 7.6
Cumulative recall curve of classifier models.

Accuracy-v-Cut-off analysis gives the probability of accuracy maximization. This measure is stated as the ratio between the sum of all true positives and true negatives (in the validation dataset) and the sum of positive and negatives predicted by the model. It is a measure of how accurate a model is at correct classification of data: that is, correct prediction of Caravan Insurance holders and nonholders in the validation dataset with minimum false predictions in each set (i.e., minimizing false positives and false negatives).

For analyzing the accuracy of prediction for each model, the following R commands were used:

```
>cust.rp.perf.acc<– performance(cust5.rp.prob.rocr,"acc")
>cust.ip.perf.acc<– performance(cust5.ip.prob.rocr,"acc")
>cust.svm.perf.acc<– performance(cust5.svm.prob.rocr,"acc")
>cust.logit.perf.acc<– performance(cust5.logit.prob.rocr,"acc")
>ppi<–300
>png(filename="Accuracy versus Cut-off curve without religion variables.png",
width=6*ppi, height=6*ppi. res=ppi)
>plot(cust.rp.perf.acc, col=2, main="Accuracy versus Cut-off curve without religion
variables")
>legend(0.5,0.5, c('rpart','bagging','svm', 'logistic'), 2:5)
>plot(cust.ip.perf.acc, col=3, add=TRUE)
>plot(cust.svm.perf.acc, col=4, add=TRUE)
>plot(cust.logit.perf.acc, col=5, add=TRUE)
>dev.off()
```

Figure 7.7 is a graphic illustration of Accuracy-v-Cut-off performance of each model. As can be seen in terms of overall accuracy by cut-off rate of 0.20, all models perform well. However, there is notable difference in the rate of increase in the probability of accuracy

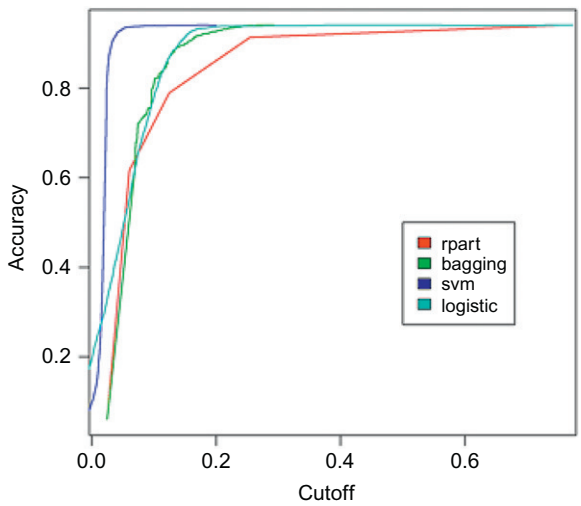


Figure 7.7
Accuracy versus cut-off curve of classifier models.

gains. This measure is an indicator of the predictive power of the model. A steeper rise in accuracy indicates that a model is good at correct prediction of true positives (or true negatives) while avoiding wrong prediction of false positives (or false negatives).

The curves for each model are thus indicative of relative predictive power of each model. For instance, SVM model has higher probability of accurate prediction of correct class member, and gaining high level of accuracy prediction probability as compared to RP and BE models. This would indicate that by including many more variables the SVM model has constructed a hyperplane gap such that it is relatively accurate in classifying (new) members of each class (i.e., instances of Insurance Policy holders and nonholders in the validation set). Therefore, although the additional variables do not result in greater predictive power (as is clear from Table 7.8), it is certainly able to predict likely target candidates with more accuracy, which can be very useful in marketing scenarios where precision in targeting the correct customer profile or type is highly important.

Next, we carried out an analysis to see if observed precision and accuracy performance of each model differed in terms of time taken to analyze or compute the predictions. This is an indication of the computational resources required by each model to deliver predictive performance presented in this subsection. To address this question, we carried out elapsed or execution run-time analysis for each model. The following R commands were used to generate execution time information:

```
>time.rp<-system.time(
+{ cust.rp<- rpart(CARAVAN~. , data=dataset1)
+cust.rp.pred<- predict(cust.rp, type="matrix", newdata=dataset2)
+cust.rp.prob.rocr<- prediction(cust.rp.pred, dataset2$CARAVAN)
+})
```

Table 7.9 presents the elapsed time (in seconds) for initiation, execution runtime, and the total time for both phases of the analysis. System Initiation time refers to time taken by the CPU to read the code; Modeling and Prediction time indicates the execution runtime. Elapsed time analysis was carried out in sequence (though we could have used the relevant R package for parallel runs). Please note that the figures below are on the basis of execution on a specific configuration of hardware and software, any changes in which is likely to give different values for initiation and computation durations.

Table 7.9 Summary of Computation Times of Classifier Models

Classification Modeling Method	System Initiation Time (s)	Modeling and Prediction Time (Sec)	Total Elapsed Time (Sec)
Recursive partitioning (RP)	0.04	14.48	14.55
Bagging Ensemble (BE)	0.57	37.29	38.00
Support Vector Machine (SVM)	0.12	15.33	15.64
Logistic Regression (LR)	0.00	1.46	1.62

The results are interesting in that though on all other criteria the RP and BE models are highly similar, in terms of processing time the BE model takes nearly three times longer (without any concurrent improvement in accuracy or speed of correct predictions as we have already seen above). In fact, the BE model takes the longest overall elapsed time for the modeling and prediction process.

The SVM model has the second highest duration for elapsed time computation though the difference from the next highest, the RP model, is little more than 1 s. This is notable given the far higher number of independent variables in the SVM model, which is an indication of higher complexity that in general would be expected to take more time to compute the predictions. Yet, the SVM model with 67 variables performs nearly as well as the RP model, which has 12 variables. Though as already discussed above, the SVM model is less optimal in terms of AUC (or explanatory power).

The LR model performs best in terms of elapsed time which is at least an order of magnitude less than all other models. It also has the highest AUC, and so in that respect despite the higher number of variables (as compared to RP and BE) this model can be regarded as the best overall performer.

Results of analysis of relative performance of each model presented in this and the previous subsection (7.3.7), collectively provide a lot of information on their explanatory or predictive power. However, such comparisons at best give us an idea of their relative performance vis-à-vis a specific criterion: It would be inappropriate to regard them as collective indicators of overall performance of each model as compared to the other three. In other words, the above analysis of comparative performance does not enable us to indentify the most *optimal* model. For that, we need to carry out a methodologically more rigorous comparison by carrying out cross validation analysis (which we have initiated as can been seen from relevant R commands that are given in Appendix E).

However, at this stage, we also feel that ideally our immediate focus ought to be on trying to improve the performance of the models presented here. In many respects, they are not too dissimilar from each other, and from a business application point of view, it would be difficult to recommend one solely on the basis of overall performance. And more to the point, although the predictive performance of the models is reasonable for many business applications it would not be sufficient: As we have already noted, for some business applications (such as automated decision-making) incorrect prediction would need to be significantly lower than observed for all four models presented here.

In Section 7.4, we briefly describe one or two possible ways in which the four models can be trained such that an increase in explanatory or predictive power is not accompanied by an increase in complexity: In fact, ideally the enhanced predictive power should be achieved with fewer variables.

7.4 Discussion of Results and Conclusion

The above analysis and model development (or ml) was carried out to derive a typical customer profile of Caravan Insurance Policy holders. To achieve this objective, we developed models using four different classification methods. One reason for the multimethod approach was to see how different classifier modeling methods affected the resulting models. As we have seen, all four models have a great deal of similarities and their overall predictive performance is comparable. Therefore, in that respect we can conclude that classification methods available in R have a high degree of methodological similarities. This outcome can also be regarded as validation of the model selected for business application: Had each method led to very distinctive models in terms of selected independent variables or predictive performance picking one of them for use as part of the marketing campaign would have been difficult.

A remarkable finding was the similarity between the RP and BE models. Both have the same subset of independent variables and similar AUC values. Although further analysis would be necessary to identify the precise nature and reasons for this similarity, this outcome is highly likely to be based on fundamental similarities in modeling methods. From a more practical point of view, this outcome reinforces the validity of both models.

With respect of to the number of variables, there is some difference between the four models. It is notable that the SVM model includes three times more independent variables than the LR model. In general, a model with more variables would be expected to over-fit (i.e., predict significantly more false positives) and/or take longer to compute predictions. Our analysis did not reveal any such serious drawbacks as far as computation runtime was concerned. If fact, it was notable that the BE model with far fewer variables required more computation time. As to over-fitting, the ROC curve seems to suggest some evidence of this though further analysis is needed to confirm that this is the case for the SVM model.

The main purpose for developing a target customer profile is to utilize it for correct identification of potential customers. A model that helps identify them precisely and accurately would enhance the success rate of a marketing campaign. All four models perform this task reasonably well. However, this is not very insightful as all models in some respect predict that the best potential Caravan Insurance purchasers are existing customers or holder of insurance policies. This sort of recommendation does not really justify the complicated data analysis or modeling presented here.

In carrying out this model building and evaluation process our aim was to see if it provided us with useful and clear insights that in turn can be used to develop a highly targeted marketing campaign, for instance, a model that unerringly predicted that customers with “high incomes” or in “senior manager position” with “boat insurance” are also highly likely to be (potential) customers of Caravan Insurance. Such a clear cut prediction would then justify a highly focused marketing campaign. Say one that involved heavy advertising in glossy Yachting or Sailing magazines with a predominantly higher socioeconomic group readership. The outcome of

which is likely to be highly positive, and perhaps even more so than trying to cross sell caravan insurance to current holders of (other types) of insurance policies.

We expect that with further refinements in the model building process it would be possible to develop such insightful models. One way of achieving this is to explore ways in which insurance policy related data can be pruned or grouped. As we have already noted all models include variables such as "Number car policies" or "Contribution car policies." The reason being their preponderance in the dataset; around a third of all variable are about either Number of or Contribution to policies. Given that these two types of variables are essentially two aspects of one (type) of insurance policy, it might be possible to discard one of them and develop models with a reduced dataset. Preliminary pairwise correlation analysis between Number and Contribution seems to suggest that that would be feasible without any significant loss in information. Classifier models based on such a reduced variable dataset are likely to have fewer independent variables, and this would be particularly so for an SVM model.

Additionally, it may be justifiable to exclude some variables completely. For instance, all four models include either Number or Contribution for car and fire policies. This really reflects the fact that over 50% of all customers held both types of policies. Their inclusion in a customer profile model is therefore not particularly informative at least from a marketing point of view. However, the inclusion of the variable related to Boat or Surfboard policies of which there are relatively few customers (33 and 3, respectively, in the training dataset of 5822) may be so. So such judicious pruning of variables may help in building models that are less complex but with equal chance of better predictive performance than those presented here. Such models are likely to prove to be of more practical use for application in business.

It can also be argued that certain variables are similar to each other. For instance, it is arguable that a customer with "Income above 125,000" may also be in "Social Class A" or have an "affluent lifestyle." A closer look at variables related to categorical variable "Customer maintype" and "Customer subtype" have revealed a lot of overlap between the levels (listed in Appendix A). On the basis of an appropriate analysis, it is possible to conflate them into one group or even have one variable act as a proxy for another that is highly similar (i.e., very significantly).

Initial analyses with pruned or grouped datasets along the lines described above, have provided encouraging results in terms of building models with enhanced predictive powers but with less complexity (i.e., reduced number of independent variables). In particular, preliminary analysis with grouped variables did not result in any significant loss of customer profile information. We plan to pursue this line of research by building predictive models based on a grouped dataset and comparing their performance with that of those presented here. We expect that such models will provide the desired nonobvious insights to support a more focused target marketing campaign.

As we have noted at various points in Section 7.3, the models presented here did provide some information about possible target customer profiles. All models included variables on

socio-economic status, levels of income, and lifestyle (also referred to as demographics). Some of these variables (e.g., "lower-level education") were included in more than two models, whereas others (e.g., "Social class C") were included by only one (SVM) model. Despite between-model differences, it is possible to discern a degree of convergence in their predictions based on the socio-economic profile. For instance, right from the early stages of the model building process, it became clear that customers with higher income or higher education (and usually "senior" or in the older age bracket) were more likely to also hold Caravan Insurance. However, more often than not models also tended to include one or the other variable such as unskilled laborers, lower or average income and lower education. This pattern seems to indicate two distinct target market groups: One, which owns a caravan or motor home as part of a leisured and affluent lifestyle. And another, that is less affluent or less skilled or with lower level of education (and possibly young families), who either own a caravan as a cost-effective way of vacationing or indeed have them as their homes (hence, the term Motor Homes). Based on the analysis carried out and presented here, this cannot be regarded as a hard and fast finding, for which, as we have already discussed, more in-depth analysis needs to be carried out.

In conclusion, given the similarity in their predictive performance, it is not possible to clearly identify the most optimal model. Although the SVM model has many more variables, and the BE model requires more computation time than the rest, these differences do not have significant impact on their accuracy or precision which from a business application point of view is perhaps the most important criteria. Taking that into account, we therefore feel that the LR model is the most optimal in terms of overall efficiency. It also accounts for marginally more variance and it displayed a predictive performance albeit with a larger number of variables than RP or BE models.

So based on the LR model, a Caravan policy marketing campaign would do well to sending out mailers (hard or soft) to all existing Fire, Boat, and Car policy holders. As part of the campaign, responses from those with lower levels of education and/or Young Families should be closely monitored. On the basis of their relative response rates, these results should be analyzed. Further analysis is also likely to lead to a more nuanced understanding of segments in the target market, and in turn be used to fine-tune the focus of the marketing campaign.

Simultaneously, the marketing campaign should target customers who in some respects are orthogonal to their main target group of the lower income and younger age group. This would include not only middle class families and/or middle management but also those are more likely to be married, homeowners, outgoing achievers possibly with private health insurance. This group may prove to be highly fragmented but that cannot be ascertained without further analysis, as well as more data and of possible feedback from actual marketing campaigns. Such information is likely to lead to a more nuanced understanding of segments in the target market.

Overall, the modeling process has provided some useful insights about the target market. In particular, it can be safely concluded that the target market is probably not a single group. There are at least two main customer profiles who are likely to own caravans (or motor homes)

and therefore are potential buyers of Caravan Insurance. For marketing purposes, each group would probably need to be approached in different ways, both in terms of the communication message as well as the medium of communication. Any such marketing campaign would then undergo further refinement based on the outcome of each phase of the campaign. In this respect a marketing campaign motivated by the optimal predictive model can be regarded as the starting point which is optimized over time.

Appendix A Details of the Full Data Set Variables

http://www.liacs.nl/~putten/library/cc2000/

Nr	Name	Description Domain
1	MOSTYPE	Customer Subtype (see L0 below)
2	MAANTHUI	Number of houses 1-10
3	MGEMOMV	Avg size household 1-6
4	MGEMLEEF	Avg age (see L1 below)
5	MOSHOOFD	Customer main type (see L2 below)
6	MGODRK	Roman Catholic (see L3 below)
7	MGODPR	Protestant
8	MGODOV	Other religion
9	MGODGE	No religion
10	MRELGE	Married
11	MRELSA	Living together
12	MRELOV	Other relation
13	MFALLEEN	Singles
14	MFGEKIND	Household without children
15	MFWEKIND	Household with children
16	MOPLHOOG	High-level education
17	MOPLMIDD	Medium-level education
18	MOPLLAAG	Lower-level education
19	MBERHOOG	High status
20	MBERZELF	Entrepreneur
21	MBERBOER	Farmer
22	MBERMIDD	Middle management
23	MBERARBG	Skilled laborers
24	MBERARBO	Unskilled laborers
25	MSKA	Social class A
26	MSKB1	Social class B1
27	MSKB2	Social class B2
28	MSKC	Social class C
29	MSKD	Social class D
30	MHHUUR	Rented house
31	MHKOOP	Home owners
32	MAUT1	1 car
33	MAUT2	2 cars
34	MAUT0	No car
35	MZFONDS	National Health Service
36	MZPART	Private health insurance

Continued

Nr	Name	Description Domain
37	MINKM30	Income < 30
38	MINK3045	Income 30,000-45,000
39	MINK4575	Income 45,000-75,000
40	MINK7512	Income 75,000-122,000
41	MINK123M	Income > 123,000
42	MINKGEM	Average income
43	MKOOPKLA	Purchasing power class
44	PWAPART	Contribution private third-party insurance (see L4 below)
45	PWABEDR	Contribution third-party insurance (firms)
46	PWALAND	Contribution third-party insurance (agriculture)
47	PPERSAUT	Contribution car policies
48	PBESAUT	Contribution delivery van policies
49	PMOTSCO	Contribution motorcycle/scooter policies
50	PVRAAUT	Contribution lorry policies
51	PAANHANG	Contribution trailer policies
52	PTRACTOR	Contribution tractor policies
53	PWERKT	Contribution agricultural machines policies
54	PBROM	Contribution moped policies
55	PLEVEN	Contribution life insurances
56	PPERSONG	Contribution private accident insurance policies
57	PGEZONG	Contribution familya accidents insurance policies
58	PWAOREG	Contribution disability insurance policies
59	PBRAND	Contribution fire policies
60	PZEILPL	Contribution surfboard policies
61	PPLEZIER	Contribution boat policies
62	PFIETS	Contribution bicycle policies
63	PINBOED	Contribution property insurance policies
64	PBYSTAND	Contribution social security insurance policies
65	AWAPART	Number of private thirdpartyinsurance1-12
66	AWABEDR	Number of third-party insurance (firms)
67	AWALAND	Number of third-party insurance (agriculture)
68	APERSAUT	Number of car policies
69	ABESAUT	Number of delivery van policies
70	AMOTSCO	Number of motorcycle/scooter policies
71	AVRAAUT	Number of lorry policies
72	AAANHANG	Number of trailer policies
73	ATRACTOR	Number of tractor policies
74	AWERKT	Number of agricultural machines policies
75	ABROM	Number of moped policies
76	ALEVEN	Number of life insurances
77	APERSONG	Number of private accident insurance policies
78	AGEZONG	Number of family accidents insurance policies
79	AWAOREG	Number of disability insurance policies
80	ABRAND	Number of fire policies
81	AZEILPL	Number of surfboard policies
82	APLEZIER	Number of boat policies
83	AFIETS	Number of bicycle policies

Nr	Name	Description Domain
84	AINBOED	Number of property insurance policies
85	ABYSTAND	Number of social security insurance policies
86	CARAVAN	Number of mobilehome policies 0-1
L0		
1	High-"income" expensive child	
2	Very important provincials	
3	High status seniors	
4	Affluent senior apartments	
5	Mixed seniors	
6	Career and childcare	
7	Dinki's (double income no kids)	
8	Middle class families	
9	"Modern" complete families	
10	Stable family	
11	Family starters	
12	Affluent young families	
13	Young all American family	
14	Junior cosmopolitan	
15	Senior cosmopolitans	
16	Students in apartments	
17	Fresh masters in the city	
18	Single youth	
19	Suburban youth	
20	Ethnically diverse	
21	Young urban have-nots	
22	Mixed apartment dwellers	
23	Young and rising	
24	Young low educated	
25	Young seniors in the city	
26	Own home elderly	
27	Seniors in apartments	
28	Residential elderly	
29	Porchless seniors: no front yard	
30	Religious elderly singles	
31	Low income Catholics	
32	Mixed seniors	
33	Lower class large families	
34	Large "family" employed child	
35	Village families	
36	Couples with teens "married with children"	
37	Mixed small town dwellers	
38	Traditional families	
39	Large religious families	
40	Large family farms	
41	Mixed rurals	
L1		
1	20-30 years	

Continued

Nr	Name	Description Domain
2	30-40 years	
3	40-50 years	
4	50-60 years	
5	60-70 years	
6	70-80 years	
L2		
1	Successful hedonists	
2	Driven growers	
3	Average family	
4	Career loners	
5	Living well	
6	Cruising seniors	
7	Retired and religious	
8	Family with grown ups	
9	Conservative families	
10	Farmers	
L3		
0	0%	
1	1-10%	
2	11-23%	
3	24-36%	
4	37-49%	
5	50-62%	
6	63-75%	
7	76-88%	
8	89-99%	
9	100%	
L4		
0	0	
1	1-49	
2	50-99	
3	100-199	
4	200-499	
5	500-999	
6	1000-4999	
7	5000-9999	
8	10,000-19,999	
9	20,000+	

Appendix B Customer Profile Data-Frequency of Binary Values

The Table B1 below shows the number of customers for each binary variables. Table B2 has the same information for nonbinary variables (i.e., with range of values between 0 and 9).

Table B2 with number of customers for each nonbinary variable. The range of (i.e., with range of values between 0 and 9).

Variables	Variable Description	Total	Missing	Unique	0	1
Subtable L0 Customer Subtype						
MOSTYPE_1	High "income" expensive child	5822	0	2	5698	124
MOSTYPE_2	Very important provincials	5822	0	2	5740	82
MOSTYPE_3	High-status seniors	5822	0	2	5573	249
MOSTYPE_4	Affluent senior apartments	5822	0	2	5770	52
MOSTYPE_5	Mixed seniors	5822	0	2	5777	45
MOSTYPE_6	Career and childcare	5822	0	2	5703	119
MOSTYPE_7	Dinki's (double income no kids)	5822	0	2	5778	44
MOSTYPE_8	Middle class families	5822	0	2	5483	339
MOSTYPE_9	"Modern" complete families	5822	0	2	5544	278
MOSTYPE_10	Stable family	5822	0	2	5657	165
MOSTYPE_11	Family starters	5822	0	2	5669	153
MOSTYPE_12	Affluent young families	5822	0	2	5711	111
MOSTYPE_13	Young all American family	5822	0	2	5643	179
MOSTYPE_14	Junior cosmopolitan	5822	0	1	5822	0
MOSTYPE_15	Senior cosmopolitans	5822	0	2	5817	5
MOSTYPE_16	Students in apartments	5822	0	1	5822	0
MOSTYPE_17	Fresh masters in the city	5822	0	2	5813	9
MOSTYPE_18	Single youth	5822	0	2	5803	19
MOSTYPE_19	Suburban youth	5822	0	2	5819	3
MOSTYPE_20	Ethnically diverse	5822	0	2	5797	25
MOSTYPE_21	Young urban have-nots	5822	0	2	5807	15

Continued

Variables	Variable Description	Total	Missing	Unique	0	1
MOSTYPE_22	Mixed apartment dwellers	5822	0	2	5724	98
MOSTYPE_23	Young and rising	5822	0	2	5571	251
MOSTYPE_24	Young low educated	5822	0	2	5642	180
MOSTYPE_25	Young seniors in the city	5822	0	2	5740	82
MOSTYPE_26	Own home elderly	5822	0	2	5774	48
MOSTYPE_27	Seniors in apartments	5822	0	2	5772	50
MOSTYPE_28	Residential elderly	5822	0	2	5797	25
MOSTYPE_29	Porchless seniors: no front yard	5822	0	2	5736	86
MOSTYPE_30	Religious elderly singles	5822	0	2	5704	118
MOSTYPE_31	Low income Catholics	5822	0	2	5617	205
MOSTYPE_32	Mixed seniors	5822	0	2	5681	141
MOSTYPE_33	Lower class large families	5822	0	2	5012	810
MOSTYPE_34	Large “family” employed child	5822	0	2	5640	182
MOSTYPE_35	Village families	5822	0	2	5608	214
MOSTYPE_36	Couples with teens “Married with children”	5822	0	2	5597	225
MOSTYPE_37	Mixed small town dwellers	5822	0	2	5690	132
MOSTYPE_38	Traditional families	5822	0	2	5483	339
MOSTYPE_39	Large religious families	5822	0	2	5494	328
MOSTYPE_40	Large family farms	5822	0	2	5751	71
MOSTYPE_41	Mixed rurals	5822	0	2	5617	205
Subtable L1 Age Range						
MGEMLEEF_1	Age 20-30	5822	0	2	5748	74
MGEMLEEF_2	Age 30-40	5822	0	2	4370	1452
MGEMLEEF_3	Age 40-50	5822	0	2	2822	3000
MGEMLEEF_4	Age 50-60	5822	0	2	4749	1073
MGEMLEEF_5	Age 60-70	5822	0	2	5629	193
MGEMLEEF_6	Age 70-80	5822	0	2	5792	30

Variables	Variable Description	Total	Missing	Unique	0	1
Subtable L2 Customer Main Type						
MOSHOOFD_1	Successful hedonists	5822	0	2	5270	552
MOSHOOFD_2	Driven Growers	5822	0	2	5320	502
MOSHOOFD_3	Average Family	5822	0	2	4936	886
MOSHOOFD_4	Career Loners	5822	0	2	5770	52
MOSHOOFD_5	Living well	5822	0	2	5253	569
MOSHOOFD_6	Cruising Seniors	5822	0	2	5617	205
MOSHOOFD_7	Retired and Religious	5822	0	2	5272	550
MOSHOOFD_8	Family with grown ups	5822	0	2	4259	1563
MOSHOOFD_9	Conservative families	5822	0	2	5155	667
MOSHOOFD_10	Farmers	5822	0	2	5546	276
Subtable L4 Cont. Private Third-Party Insurance						*5822*
PWAPART_0	0	5822	0	2	2340	3482
PWAPART_1	01-49	5822	0	2	5621	201
PWAPART_2	50-99	5822	0	2	3694	2128
PWAPART_3	100-199	5822	0	2	5811	11
PWAPART_4	200-499	5822	0	1	5822	0
PWAPART_5	500-999	5822	0	1	5822	0
PWAPART_6	1000-4999	5822	0	1	5822	0
PWAPART_7	5000-9999	5822	0	1	5822	0
PWAPART_8	10,000-19,999	5822	0	1	5822	0
PWAPART_9	20,000+	5822	0	1	5822	0
MAATHUI	Number of households	5822	0	9	5267	555
MGEMOMV	Avg size household	5822	0	5	284	4777
MRELGE	Married	5822	0	10	64	5758
MRELSA	Living together	5822	0	8	2448	3374
MRELOV	Other relation	5822	0	10	1173	4649
MFALLEE	Singles	5822	0	10	1757	4065
MFGEKIND	Household without children	5822	0	10	371	5451

Continued

Variables	Variable Description	Total	Missing	Unique	0	1
MFWEKIND	Household with children	5822	0	10	153	5669
MOPLHOOG	High-level education	5822	0	10	2147	3675
MOPLMIDD	Medium-level education	5822	0	10	423	5399
MOPLLAAG	Lower-level education	5822	0	10	299	5523
MBERHOOG	High status	5822	0	10	1524	4298
MBERZELF	Entrepreneur	5822	0	6	4171	1651
MBERBOER	Farmer	5822	0	10	4176	1646
MBERMIDD	Middle management	5822	0	10	667	5155
MBERARBG	Skilled laborers	5822	0	10	1167	4655
MBERARBO	Unskilled laborers	5822	0	10	968	4854
MSKA	Social class A	5822	0	10	1738	4084
MSKB1	Social class B1	5822	0	10	1353	4469
MSKB2	Social class B2	5822	0	10	990	4832
MSKC	Social class C	5822	0	10	364	5458
MSKD	Social class D	5822	0	9	2607	3215
MHHUUR	Rented house	5822	0	10	949	4873
MHKOOP	Home owners	5822	0	10	760	5062
MAUT1	1 car	5822	0	10	19	5803
MAUT2	2 cars	5822	0	8	1854	3968
MAUT0	No car	5822	0	10	1450	4372
MZFONDS	National Health Service	5822	0	10	55	5767
MZPART	Private health insurance	5822	0	10	852	4970
MIKM30	Income <30	5822	0	10	1304	4518
MIK3045	Income 30,000-45,000	5822	0	10	465	5357
MIK4575	Income 45,000-75,000	5822	0	10	891	4931
MIK7512	Income 75,000-122,000	5822	0	10	3246	2576
MIK123M	Income $>123,000$	5822	0	8	4900	922
MIKGEM	Average income	5822	0	10	25	5797
MKOOPKLA	Purchasing Power Class	5822	0	8	587	5235

Variables	Variable Description	Total	Missing	Unique	0	1	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	Zone 7	Zone 8	Zone 9
PWABEDR	Contribution third party (firms)	5822	0	7	5740	82	7	30	23	17	1	4			
PWALAND	Contribution third party (agri)	5822	0	4	5702	120		3	57	60					
PPERSAUT	Contribution car policies	5822	0	6	2845	2977	1	613	2319	41	3				
PBESAUT	Contribution delivery van	5822	0	4	5774	48					10	35	3		
PMOTSCO	Cont. motorcycle policies	5822	0	6	5600	222	3	136	32	49	2				
PVRAAUT	Contribution lorry policies	5822	0	4	5813	9				1		7			1
PAANHANG	Contribution trailer policies	5822	0	6	5757	65	19	38	6	1	1				
PTRACTOR	Contribution tractor policies	5822	0	5	5679	143	79	27	28	9					
PWERKT	Contribution agri. machines	5822	0	5	5801	21	4	6	8	3					
PBROM	Contribution moped policies	5822	0	6	5426	396	34	282	63	16	1				
PLEVEN	Contribution life insurances	5822	0	10	5529	293	9	28	84	94	35	38	3	1	1
PPERSONG	Contribution private accident	5822	0	7	5791	31	3	18	4	3	1	2			
PGEZONG	Contribution family accidents	5822	0	3	5784	38		25	13						
PWAOREG	Cont. disability insurance	5822	0	5	5799	23	1	1	19	2					
PBRAND	Contribution fire policies	5822	0	9	2666	3156	161	535	920	1226	149	155	9	1	
PZEILPL	Contribution surfboard policies	5822	0	3	5819	3	2		1						
PPLEZIER	Contribution boat policies	5822	0	7	5789	33	5	5	5	13	2	3			

Continued

Variables	Variable Description	Total	Missing	Unique	0	1	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	Zone 7	Zone 8	Zone 9
PFIETS	Contribution bicycle policies	5822	0	2	5675	147	147								
PINBOED	Cont. property insurance policies	5822	0	7	5777	45	18	16	6	3	1	1			
PBYSTAND	Cont. social security insurance	5822	0	5	5740	82	15	22	44	1					
AWAPART	Number of private third party	5822	0	3	3482	2340	2334	6							
AWABEDR	No. of third-party insurance	5822	0	3	5740	82	81				1				
AWALAND	No. of third-party insurance	5822	0	2	5702	120	120								
APERSAUT	No. of car policies	5822	0	7	2845	2977	2712	246	12	5	1	1			
ABESAUT	No. of delivery van policies	5822	0	5	5774	48	40	4	3	1					
AMOTSCO	No. of motorcycle policies	5822	0	4	5600	222	211	10						1	
AVRAAUT	No. of lorry policies	5822	0	4	5813	9	6	2	1						
AAANHANG	No. of trailer policies	5822	0	4	5757	65	59	4	2						
ATRACTOR	No. of tractor policies	5822	0	5	5679	143	105	29	3	6					
AWERKT	No. of agri. machines policies	5822	0	5	5801	21	12	6	2	1					
ABROM	No. of moped policies	5822	0	3	5426	396	382	14							
ALEVEN	No. of life insurances	5822	0	6	5529	293	173	100	11	8	1				
APERSONG	No. of private accident insurance	5822	0	2	5971	31		31							
AGEZONG	No. of family accidents insurance	5822	0	2	5784	38		38							

Variables	Variable Description	Total	Missing	Unique	0	1	Zone 1	Zone 2	Zone 3	Zone 4	Zone 5	Zone 6	Zone 7	Zone 8	Zone 9
AWAOREG	No. of disability insurance policies	5822	0	3	5799	23	19	4							
AZEILPL	No. of surfboard policies	5822	0	2	5819	3	3								
APLEZIER	No. of boat policies	5822	0	3	5789	33	31	2							
AFIETS	No. of bicycle policies	5822	0	4	5675	147	111	34	2						
AINBOED	No. property insurance policies	5822	0	3	5777	45	44	1							
ABYSTAND	No. of social security insurance	5822	0	3	5740	82	81	1							
ABRAND	No. of fire policies	5822	0	7	2666	3156	3017	126	7	3	2	1			
CARAVAN	No. Caravan Policies	5822	0	2	5474	348	348								

Appendix C Proportion of Caravan Insurance Holders vis-à-vis other Customer Profile Variables

Profile Parameter	Prop Holding Caravan Insurance	Profile Parameter	Prop Holding Caravan Insurance
Customer Subtype	*See Subtable L0*	Number of surfboard policies	0.03
Number of households	0.00	Number of boat policies	0.11
Avg size household	0.04	Number of bicycle policies	0.03
Age Range	*See Subtable L1*	No. of property insurance policies	0.02
Customer Main Type	*See Subtable L2*	Number of social security insurance	0.07
Married	0.07	*Sub Table L0 Customer Subtype*	
Living together	−0.03	High "income" expensive child	0.03
Other relation	−0.06	Very Important Provincials	0.01
Singles	−0.05	High-status seniors	0.04
Household without children	0.01	Affluent senior apartments	−0.01
Household with children	0.03	Mixed seniors	−0.01
High-level education	0.08	Career and childcare	0.03
Medium-level education	0.04	Dinki's (double income no kids)	0.00
Lower-level education	−0.09	Middle class families	0.10
High status	0.06	"Modern" complete families	−0.02
Entrepreneur	0.03	Stable family	0.00
Farmer	−0.06	Family starters	0.00
Middle management	0.04	Affluent young families	0.05
Skilled laborers	−0.05	Young all American family	0.01
Unskilled laborers	−0.06	Junior cosmopolitan	NA
Social class A	0.06	Senior cosmopolitans	−0.01
Social class B1	0.03	Students in apartments	NA
Social class B2	0.01	Fresh masters in the city	−0.01
Social class C	−0.05	Single youth	−0.01
Social class D	−0.06	Suburban youth	−0.01
Rented house	−0.08	Ethnically diverse	0.01
Home owners	0.08	Young urban have-nots	−0.01
1 car	0.07	Mixed apartment dwellers	−0.01
2 cars	0.01	Young and rising	−0.04
No car	−0.08	Young low educated	−0.02
National Health Service	−0.06	Young seniors in the city	−0.02
Private health insurance	0.06	Own home elderly	−0.01
Income < 30	−0.08	Seniors in apartments	−0.02
Income 30,000-45,000	−0.01	Residential elderly	−0.02
Income 45,000-75,000	0.07	Porchless seniors: no front yard	−0.02
Income 75,000-122,000	0.06	Religious elderly singles	−0.02

Profile Parameter	Prop Holding Caravan Insurance	Profile Parameter	Prop Holding Caravan Insurance
Income >123,000	0.01	Low income catholic	−0.02
Average income	0.10	Mixed seniors	0.00
Purchasing power class	0.09	Lower class large families	−0.01
Profile Parameter	*Prop Holding Caravan Insurance*	*Profile Parameter*	*Prop Holding Caravan Insurance*
Cont. Private Third-Party Insurance	*See Subtable L4*	Large "family" employed child	−0.01
Contribution third party (firms)	0.00	Village families	−0.02
Contribution third party (agriculture)	−0.02	Couples with teens 'Married with children'	0.01
Contribution car policies	0.16	Mixed small town dwellers	0.01
Contribution delivery van	−0.01	Traditional families	0.01
Cont. motorcycle/scooter policies	0.01	Large religious families	0.00
Contribution lorry policies	−0.01	Large family farms	−0.03
Contribution trailer policies	0.01	Mixed rural	−0.03
Contribution tractor policies	−0.02	*Sub Table L1 Age Range*	
Contribution agricultural machines	−0.02	Age 20-30	−0.02
Contribution moped policies	−0.05	Age 30-40	0.00
Contribution life insurances	0.02	Age 40-50	0.01
Contribution private accident	−0.01	Age 50-60	0.00
Contribution family accidents	0.03	Age 60-70	0.00
Contribution disability insurance	0.03	Age 70-80	−0.01
Contribution fire policies	0.10	*Sub Table L2 Customer Main Type*	
Contribution surfboard policies	0.03	Successful hedonists	0.04
Contribution boat policies	0.11	Driven Growers	0.09
Contribution bicycle policies	0.03	Average Family	0.01
Cont. property insurance policies	0.02	Career Loners	−0.02
Cont. social security ins. policies	0.07	Living well	−0.05
Number of private third party	0.09	Cruising Seniors	−0.03
Number of third party insurance	0.00	Retired and Religious	−0.03
Number of third party insurance	−0.02	Family with grown ups	−0.01
Number of car policies	0.15	Conservative families	0.00
Number of delivery van policies	−0.01	Farmers	−0.04

Continued

Profile Parameter	Prop Holding Caravan Insurance	Profile Parameter	Prop Holding Caravan Insurance
No of motorcycle/scooter policies	0.01	*Sub Table L4 Cont. Private Third-Party Insurance*	
Number of lorry policies	−0.01	0	−0.09
Number of trailer policies	0.01	01-99	−0.02
Number of tractor policies	−0.02	50-99	0.10
Number of agri machines policies	−0.02	100-199	0.02
Number of moped policies	−0.05	200-499	NA
Number of life insurances	0.02	500-999	NA
Number of private accident insurance	−0.01	1000-4999	NA
Number of family accidents insurance	0.03	5000-9999	NA
Number of disability ins. policies	0.03	10,000-19,999	NA
Number of fire policies	0.07	20,000+	NA

Appendix D LR Model Details

Output of >summary(cust.logit)					Signif. codes: 0 “***” 0.001 “**” 0.01 “*” 0.05 “.” 0.1 “ “ 1
Call:					(Dispersion parameter for gaussian family taken to be 0.05283494)
glm(formula = CARAVAN ~, data = train_3)					Null deviance: 327.2 on 5821 degrees of freedom
Deviance Residuals:					Residual deviance: 301.0 on 5697 degrees of freedom
Min	1Q	Med	3Q	Max	AIC: −472.31
−70,314	−0.08721	−0.04517	−0.00434	−0.03944	Number of Fisher Scoring iterations: 2

	Estimate	Std. Error	*t*-Value	Pr(>|*t*|)	Sig Lev		Estimate	Std. Error	*t*-Value	Pr(>|*t*|)	Sig Lev
MOSTYPE_14	NA	NA	NA	NA		MOSTYPE_5	2.77×10^{-2}	7.93×10^{-2}	0.35	0.73	
MOSTYPE_16	NA	NA	NA	NA		MOSTYPE_1	4.47×10^{-2}	1.32×10^{-1}	0.34	0.73	
MGEMLEEF_6	NA	NA	NA	NA		MOSTYPE_41	2.79×10^{-2}	8.34×10^{-2}	0.34	0.74	
MOSHOOFD_1	NA	NA	NA	NA		PPERSONG	1.06×10^{-2}	3.36×10^{-2}	0.31	0.75	
MOSHOOFD_2	NA	NA	NA	NA		MOSTYPE_11	3.29×10^{-2}	1.06×10^{-1}	0.31	0.76	
MOSHOOFD_3	NA	NA	NA	NA		MOSTYPE_29	2.34×10^{-2}	7.55×10^{-2}	0.31	0.76	
MOSHOOFD_4	NA	NA	NA	NA		MOSTYPE_23	2.06×10^{-2}	7.14×10^{-2}	0.29	0.77	
MOSHOOFD_5	NA	NA	NA	NA		MOPLHOOG	1.97×10^{-3}	6.98×10^{-3}	0.28	0.78	
MOSHOOFD_6	NA	NA	NA	NA		MOSTYPE_13	2.87×10^{-2}	1.05×10^{-1}	0.27	0.78	
MOSHOOFD_7	NA	NA	NA	NA		MOSTYPE_21	2.15×10^{-2}	8.58×10^{-2}	0.25	0.80	
MOSHOOFD_8	NA	NA	NA	NA		MOSTYPE_40	2.01×10^{-2}	8.02×10^{-2}	0.25	0.80	
MOSHOOFD_9	NA	NA	NA	NA		MOSTYPE_25	1.66×10^{-2}	6.73×10^{-2}	0.25	0.81	
MOSHOOFD_10	NA	NA	NA	NA		AWABEDR	1.23×10^{-2}	5.31×10^{-2}	0.23	0.82	
PWAPART_3	NA	NA	NA	NA		AWALAND	2.65×10^{-2}	1.38×10^{-1}	0.19	0.85	
PWAPART_4	NA	NA	NA	NA		PBESAUT	1.97×10^{-3}	1.49×10^{-2}	0.13	0.89	
PWAPART_5	NA	NA	NA	NA		MKOOPKLA	6.46×10^{-4}	1.60×10^{-2}	0.04	0.97	
PWAPART_6	NA	NA	NA	NA		MOSTYPE_28	2.28×10^{-3}	7.96×10^{-2}	0.03	0.98	
PWAPART_7	NA	NA	NA	NA		MOSTYPE_10	3.07×10^{-3}	1.30×10^{-1}	0.02	0.98	
PWAPART_8	NA	NA	NA	NA		MOSTYPE_7	1.62×10^{-3}	1.10×10^{-1}	0.02	0.99	
PWAPART_9	NA	NA	NA	NA		MOSTYPE_18	1.26×10^{-3}	8.22×10^{-2}	0.02	0.99	
APLEZIER	3.57×10^{-1}	8.88×10^{-2}	4.03	0.00	***	AWERKT	1.04×10^{-3}	7.32×10^{-2}	0.01	0.99	
PPERSAUT	1.03×10^{-2}	2.65×10^{-3}	3.88	0.00	***	APERSAUT	-5.27×10^{-5}	1.28×10^{-2}	0.00	1.00	
PBRAND	1.16×10^{-2}	3.63×10^{-3}	3.20	0.00	**	MBERZELF	-1.85×10^{-4}	5.54×10^{-3}	−0.03	0.97	
(Intercept)	1.33×10	4.82×10^{-1}	2.76	0.01	**	MSKB2	-3.46×10^{-4}	4.58×10^{-3}	−0.08	0.94	
ALEVEN	4.15×10^{-2}	1.55×10^{-2}	2.68	0.01	**	MOSTYPE_4	-8.74×10^{-3}	1.07×10^{-1}	−0.08	0.94	
PGEZONG	1.94×10^{-1}	7.95×10^{-2}	2.44	0.01	*	PFIETS	-5.01×10^{-3}	5.52×10^{-2}	−0.09	0.93	
PWAOREG	5.49×10^{-2}	2.59×10^{-2}	2.12	0.03	*	MGEMOMV	-7.54×10^{-4}	7.36×10^{-3}	−0.10	0.92	
AZEILPL	5.03×10^{-1}	2.82×10^{-1}	1.78	0.07	.	MBERARBG	-6.33×10^{-4}	4.60×10^{-3}	−0.14	0.89	
AINBOED	1.03×10^{-1}	7.01×10^{-2}	1.47	0.14		PWERKT	-6.19×10^{-3}	3.72×10^{-2}	−0.17	0.87	
ABYSTAND	1.39×10^{-1}	9.87×10^{-2}	1.41	0.16		MOSTYPE_19	-2.95×10^{-2}	1.52×10^{-1}	−0.19	0.85	
MRELGE	9.01×10^{-3}	7.54×10^{-3}	1.20	0.23		MFALLEEN	-1.44×10^{-3}	6.67×10^{-3}	−0.22	0.83	
MINK3045	5.76×10^{-3}	5.00×10^{-3}	1.15	0.25		MFWEKIND	-1.89×10^{-3}	7.21×10^{-3}	−0.26	0.79	
PAANHANG	6.08×10^{-2}	5.63×10^{-2}	1.08	0.28		PWABEDR	-6.10×10^{-3}	2.06×10^{-2}	−0.30	0.77	
MAUT2	7.57×10^{-3}	7.02×10^{-3}	1.08	0.28		MSKD	-1.65×10^{-3}	4.79×10^{-3}	−0.35	0.73	
MAUT1	8.17×10^{-3}	7.85×10^{-3}	1.04	0.30		MSKA	-1.85×10^{-3}	5.32×10^{-3}	−0.35	0.73	
	Estimate	**Std. Error**	***t*-Value**	**Pr(>|*t*|)**	**Sig Lev**		**Estimate**	**Std. Error**	***t*-Value**	**Pr(>|*t*|)**	**Sig Lev**
MINKM30	5.29×10^{-3}	5.21×10^{-3}	1.02	0.31		ABESAUT	-2.33×10^{-2}	6.54×10^{-2}	−0.36	0.72	
MINK7512	4.98×10^{-3}	5.30×10^{-3}	0.94	0.35		MAANTHUI	-3.58×10^{-3}	8.82×10^{-3}	−0.41	0.68	
MINKGEM	4.28×10^{-3}	4.63×10^{-3}	0.93	0.35		ABROM	-1.91×10^{-2}	4.69×10^{-2}	−0.41	0.68	
MINK4575	4.64×10^{-3}	5.08×10^{-3}	0.91	0.36		MBERBOER	-2.42×10^{-3}	5.49×10^{-3}	−0.44	0.66	
AMOTSCO	2.82×10^{-2}	3.13×10^{-2}	0.90	0.37		MFGEKIND	-3.19×10^{-3}	6.92×10^{-3}	−0.46	0.65	
MOSTYPE_20	7.19×10^{-2}	8.04×10^{-2}	0.89	0.37		MOSTYPE_17	-5.66×10^{-2}	1.22×10^{-1}	−0.47	0.64	
AFIETS	3.60×10^{-2}	4.10×10^{-2}	0.88	0.38		APERSONG	-4.56×10^{-2}	9.57×10^{-2}	−0.48	0.63	
MOSTYPE_12	1.00×10^{-1}	1.19×10^{-1}	0.84	0.40		PWALAND	-2.17×10^{-2}	3.91×10^{-2}	−0.56	0.58	
MOSTYPE_37	6.95×10^{-2}	8.36×10^{-2}	0.83	0.41		MSKB1	-3.18×10^{-3}	5.12×10^{-3}	−0.62	0.53	
MBERMIDD	3.80×10^{-3}	4.61×10^{-3}	0.83	0.41		PVRAAUT	-2.64×10^{-2}	4.17×10^{-2}	−0.63	0.53	
PTRACTOR	1.17×10^{-2}	1.43×10^{-2}	0.82	0.41		PBYSTAND	-2.06×10^{-2}	2.90×10^{-2}	−0.71	0.48	
MOSTYPE_36	5.92×10^{-2}	7.30×10^{-2}	0.81	0.42		MOPLMIDD	-5.61×10^{-3}	7.30×10^{-3}	−0.77	0.44	
MRELOV	6.09×10^{-3}	7.57×10^{-3}	0.80	0.42		MGEMLEEF_5	-3.70×10^{-2}	4.81×10^{-2}	−0.77	0.44	

Continued

	Estimate	Std. Error	*t*-Value	Pr(>|*t*|)	Sig Lev		Estimate	Std. Error	*t*-Value	Pr(>|*t*|)	Sig Lev
MOSTYPE_32	5.18×10^{-2}	6.57×10^{-2}	0.79	0.43		AAANHANG	-8.21×10^{-2}	9.52×10^{-2}	−0.86	0.39	
MOSTYPE_38	6.28×10^{-2}	7.98×10^{-2}	0.79	0.43		PPLEZIER	-2.73×10^{-2}	2.70×10^{-2}	−1.01	0.31	
MOSTYPE_8	9.18×10^{-2}	1.18×10^{-1}	0.78	0.44		PMOTSCO	-8.26×10^{-3}	7.99×10^{-3}	−1.03	0.30	
MOSTYPE_33	5.07×10^{-2}	7.23×10^{-2}	0.70	0.48		ATRACTOR	-3.68×10^{-2}	3.54×10^{-2}	−1.04	0.30	
MOSTYPE_31	4.45×10^{-2}	6.51×10^{-2}	0.68	0.49		MGEMLEEF_4	-5.56×10^{-2}	4.78×10^{-2}	−1.17	0.24	
MSKC	3.42×10^{-3}	5.00×10^{-3}	0.68	0.49		MHKOOP	-4.48×10^{-2}	3.80×10^{-2}	−1.18	0.24	
MOSTYPE_30	4.59×10^{-2}	6.84×10^{-2}	0.67	0.50		MZFONDS	-5.39×10^{-2}	4.47×10^{-2}	−1.21	0.23	
MOSTYPE_3	6.80×10^{-2}	1.04×10^{-1}	0.65	0.51		MHHUUR	-4.67×10^{-2}	3.80×10^{-2}	−1.23	0.22	
MAUT0	4.76×10^{-3}	7.51×10^{-3}	0.63	0.53		MZPART	-5.76×10^{-2}	4.46×10^{-2}	−1.29	0.20	
MOSTYPE_26	4.29×10^{-2}	7.20×10^{-2}	0.60	0.55		PZEILPL	-1.90×10^{-1}	1.44×10^{-1}	−1.32	0.19	
MOSTYPE_24	3.84×10^{-2}	6.63×10^{-2}	0.58	0.56		MGEMLEEF_3	-6.78×10^{-2}	4.88×10^{-2}	−1.39	0.16	
MBERARBO	2.61×10^{-3}	4.55×10^{-3}	0.57	0.57		PINBOED	-4.37×10^{-2}	3.09×10^{-2}	−1.42	0.16	
MOSTYPE_22	3.72×10^{-2}	6.87×10^{-2}	0.54	0.59		MGEMLEEF_1	-8.20×10^{-2}	5.70×10^{-2}	−1.44	0.15	
MOSTYPE_39	4.66×10^{-2}	9.19×10^{-2}	0.51	0.61		ABRAND	-1.73×10^{-2}	1.17×10^{-2}	−1.48	0.14	
MBERHOOG	2.32×10^{-3}	4.68×10^{-3}	0.50	0.62		MGEMLEEF_2	-7.37×10^{-2}	4.94×10^{-2}	−1.49	0.14	
MOSTYPE_6	6.38×10^{-2}	1.32×10^{-1}	0.49	0.63		AWAOREG	-1.94×10^{-1}	1.25×10^{-1}	−1.55	0.12	
MRELSA	3.19×10^{-3}	7.12×10^{-3}	0.45	0.65		MOPLLAAG	-1.32×10^{-2}	7.43×10^{-3}	−1.78	0.08	.
MOSTYPE_27	3.17×10^{-2}	7.12×10^{-2}	0.45	0.66		MINK123M	-1.37×10^{-2}	6.96×10^{-3}	−1.97	0.05	*
AVRAAUT	6.53×10^{-2}	1.60×10^{-1}	0.41	0.68		AGEZONG	-4.08×10^{-1}	1.90×10^{-1}	−2.15	0.03	*
MOSTYPE_34	4.08×10^{-2}	1.05×10^{-1}	0.39	0.70		AWAPART	-2.81×10^{-1}	1.21×10^{-1}	−2.32	0.02	*
MOSTYPE_9	3.08×10^{-2}	8.13×10^{-2}	0.38	0.70		PWAPART_2	-2.19×10^{-1}	9.06×10^{-2}	−2.41	0.02	*
MOSTYPE_2	4.06×10^{-2}	1.08×10^{-1}	0.38	0.71		PWAPART_1	-2.40×10^{-1}	9.21×10^{-2}	−2.61	0.01	**
PBROM	5.54×10^{-3}	1.53×10^{-2}	0.36	0.72		PLEVEN	-1.77×10^{-2}	6.52×10^{-3}	−2.71	0.01	**
MOSTYPE_35	3.32×10^{-2}	9.26×10^{-2}	0.36	0.72		PWAPART_0	-5.20×10^{-1}	1.91×10^{-1}	−2.72	0.01	**
MOSTYPE_15	4.29×10^{-2}	1.21×10^{-1}	0.35	0.72							

Appendix E R Commands for Computation of ROC Curves for Each Model Using Validation Dataset

Individual ROC curves of the initial model (with 145 variables). To draw ROC curves it is essential to load library ROCR by Sing et al. (2005).

R commands for ROC for RP:

```
>library(ROCR)
>library(rpart)
>cust.rp.pred<– predict(cust.rp, type="matrix", newdata=dataset2)
>cust.rp.prob.rocr<– prediction(cust.rp.pred, dataset2$CARAVAN)
>cust.rp.perf<– performance(cust.rp.prob.rocr, "tpr", "fpr")
>plot(cust.rp.perf, main="ROC curve using recursive partioning", colorize=T)
```

R commands for ROC for BE:

```
>library(ipred)
>cust.ip.prob<– predict(cust.ip, type="prob", newdata=dataset2)
>cust.ip.prob.rocr<– prediction(cust.ip.prob, dataset2$CARAVAN)
>cust.ip.perf<– performance(cust.ip.prob.rocr,"tpr", "fpr")
>plot(cust.ip.perf, main="ROC curve using bagging ensemble", colorize=T)
```

R commands for ROC for SVM:

```
>library(e1071)
>cust.svm.prob<– predict(cust.svm, type="prob" , newdata=dataset2,probability=TRUE)
>cust.svm.prob.rocr<– prediction(cust.svm.prob , dataset2$CARAVAN)
>cust.svm.perf<– performance(cust.svm.prob.rocr,"tpr", "fpr")
>plot(cust.svm.perf , main="ROC curve using SVM", colorize=T)
```

R commands for ROC for Logistics Regression (RP):

```
>cust.logit.pred<– predict(cust.logit, newdata=dataset2, type="response")
>cust.logit.prob.rocr<– prediction(cust.logit.pred, dataset2$CARAVAN)
>cust.logit.perf<– performance(cust.logit.prob.rocr, "tpr","fpr")
>plot(cust.logit.perf , main="ROC using Logistic regression", colorize=T)
```

Appendix F Commands for Cross-Validation Analysis of Classifier Models

This analysis requires a significant amount of computation resources. To address this issue, the use of parallel processing package NetworkSpaces in R is highly recommended.

R commands for cross-validation analysis of RP model:

```
>library(rpart)
>control<– rpart.control(xval=5) [xval defines number of cross validations]
>cust.rp<– rpart(CARAVAN~. , data=dataset1, control)
```

R commands for cross-validation analysis of BE model:

```
>library(adabag)
```

```
>cust.baggingcv<- bagging.cv(CARAVAN~., v=10, data=dataset12, mfinal=10)[v defines
number of cross validations]
```

R commands for cross-validation analysis of SVM model:

```
>library(e1071)
>cust.svm<- svm(CARAVAN ~., data=dataset1,method="C-classification", kernel="radial",
cost=10,gamma=0.1,cross=10,fitted=TRUE, probability=TRUE) [cross defines number of cross
validations ]
```

R commands for cross-validation analysis of LR model:

```
>library(boot)
>cv.logit<-cv.glm(dataset1, cust4.logit, K=5) [K defines number of cross validations]
>cv.logit
```

References

Breiman, L., 1996a. Bagging predictors. Machine Learn. 24 (2), 123–140.

Breiman, L., 1996b. Out-Of-Bag Estimation. Technical Report. ftp://ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps.Z.

Breiman, L., 1998. Arcing classifiers. Ann. Stat. 26 (3), 801–824.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classication and Regression Trees. Wadsworth International Group.

Chang, C.-C., Lin, C.-J., 2001. LIBSVM: a library for support vector machines, Technical Report, Department of Computer Science and Information Engineering, National Taiwan University. http://www.csie.edu.tw/~cjlin/papers/libsvm.pdf Last accessed on 12 December 2012.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A., 2011. e1071: Misc Functions of the Department of Statistics, TU Wien. R package version 1.6. http://CRAN.R-project.org/package=e1071.

Dobson, A.J., 1990. An Introduction to Generalized Linear Models. Chapman and Hall, London.

Hastie, T.J., 2004. The SVM Path algorithm. R package, Version 0.9. http://CRAN.R-project.org/.

Hastie, T.J., Pregibon, D., 1992. Generalized linear models. In: Chambers, J.M., Hastie, T.J. (Eds.), Statistical Models in S. Wadsworth & Brooks/Cole, Pacific Grove.

Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A., 2004. Kernlab An S4 package for kernel methods in R. J. Stat. Softw. 11 (9), 1–20.

Kim, K.J., 2003. Financial time series forecasting using support vector machines. Neurocomputing. 55 (1/2), 307–319.

Kuhn, M., 2012. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer and Allan Engelhardt. caret*: Classification and Regression Training. R package version 5.15-023* http://CRAN.R-project.org/package=caret.

Manning, C., 2007. Logistic regression (with R), http://nlp.stanford.edu/manning/courses/ling289/logistic.pdf. Last accessed 4 December 2012.

McCullagh, P., Nelder, J.A., 1989. Generalized Linear Models. Chapman and Hall, London.

Peters, A., Hothorn, T., Berthold, L., 2002. ipred: improved predictors. R News. 2 (2), 33–36. June 2002. URL, http://CRAN.R-project.org/doc/Rnews/.

Shawe-Taylor, J., Cristianini, N., 2000. An introduction to Support Vector Machines. Cambridge University Press, Cambridge.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T., 2005. ROCR: visualizing classifier performance in R. *Bioinformatics*. 21 (20), 7881. http://rocr.bioinf.mpi-sb.mpg.de.

Therneau, T.M., Atkinson, B., ported by Brian Ripley, 2010. rpart: recursive partitioning. R package version 3.1–46.

Vapnik, V.N., 2000. The nature of statistical learning theory, second ed. Springer-Verlag, New York. ISBN 978-1-4419-3160-3.

Weihs, C., Ligges, U., Luebke, K., Raabe, N., 2005. klaR Analyzing German Business Cycles. In: Baier, D., Decker, R., Schmidt-Thieme, L. (Eds.), Data Analysis and Decision Support. Springer-Verlag, Berlin, pp. 335–343.

CHAPTER 8

Selecting Best Features for Predicting Bank Loan Default

Zahra Yazdani*, **Mohammad Mehdi Sepehri**†, **Babak Teimourpour**†

**Group of Information Technology Management, Payam Noor University, Tehran, Iran*

†Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran

8.1 Introduction

Credit risk evaluation becomes more and more important for economic corporations and banks. The Basel committee defines rules for credit risk evaluation process. The Committee permits banks a choice between two broad methodologies for calculating their capital requirements for credit risk.

One alternative, the Standardized Approach, measures credit risk in a standardized manner, supported by external credit assessments. The other alternative, the Internal Ratings-based Approach, which is subject to the explicit approval of the bank supervisor, allows banks to use their internal rating systems for credit risk (BCBS, 2006). On the basis of the standard explanations the effective components on credit risk are probability of default, loss given by default, and exposure at default, of which probability of default (PD) is the most important one.

According to Basel II, a default is considered to have occurred when the bank considers that the obligor is unlikely to pay its credit obligations and/or the obligor is overdue by more than 90 days on any material credit obligation to the banking group (BCBS, 2005).

On the basis of the IRB approach, banks should categorize the *exposures* into classes depending on the type of assets. After categorizing the exposures, they need to estimate the values of risk components for each loan or exposure (BCBS, 2001). Determining PD is prerequisite for estimating other components and the time horizon used in PD estimation is one year (BCBS, 2006).

This chapter presents a data mining framework for building a PD model. The aim is first to identify the effective features to estimate PD and second to predict PD for new loans.

Data Mining Applications with R

8.2 Business Problem

In recent years, there has been a significant increase in the number of companies that have defaulted. It increases the bank's credit risk and requires the banks to raise their regulatory capital. Under the IRB approach, banks must categorize their exposures into classes of assets with different underlying risk characteristics: corporate, sovereign, bank, retail, and equity.

In general, a corporate exposure is defined as a debt obligation of a *corporation*, *partnership*, or *proprietorship* (BCBS, 2001).

Because the PD is the most important component of credit risk, the goal is to identify efficient features on PD and estimate PD for new loans in the category of corporate asset. There are many features but some of them are irrelevant and only some of the features may affect the model's performance. Using irrelevant features leads to very poor results. To avoid being trapped by irrelevant features, we need to analyze the data, prepare it, and choose the best subset of features that are more efficient. The main goal of the proposed method is to predict PD with the aim of decreasing the bank's credit risk.

8.3 Data Extraction

On the basis of some related research studies and by asking experts, we defined and extracted the most important features that affect the PD of corporate loans. This extraction is categorized into four groups: Loan characteristics, corporate structural features, behavioral features, and systematic factors. As the bank is not able to control systematic factors such as economic risks, we do not take these features into consideration and only use the other factors that exist in the bank's databases.

Beacuse the existent databases contain a huge amount of data, a sampling technique was applied to extract data. Ten branches of the bank were randomly chosen as samples, which involves almost 140,000 loans in corporate class during the past 15 years. The data are for the period 1995-2010. Among all those loans, only a few obligors have defaulted.

We divided the data into two separate sets; the first one contains all the data from the first 14 years [0,14] (First set) and has around 124,000 observations and the other one from the last year [14,15] (Second set) that had our main data set with 16,000 observations. The first data set was used to generate past behavioral features for customers that had transactions in the second period.

The objective function (Binary target) is to predict PD using the extracted features from the second set and the generated features from first set. We accomplished these steps by programming and using SQL.

Table 8.1 Types of Independence Data

x1	*x2*	*x3*	*x4*	*x5*	*x6*	*x7*	*x8*	*x9*	*x10*	*x11*	*x12*	*x13*	*x14*	*x15*	*x16*	*x17*	*x18*
C	N	N	N	C	C	C	R	R	B	C	N	C	C	B	B	R	R

B: Binary, C: Continuous, N: Nominal, R: Ratio.

We transferred the extracted data to an excel file (http://www.rdatamining.com/books/dmar/). The data set includes 18 independent features with different types of values and a binary target feature. The value of the target feature is zero or one.

$$y_i = \begin{cases} 1, & \textit{If the customer defaults} \\ 0, & \text{otherwise} \end{cases}$$

Table 8.1 displays types of independence data and the description of the features are given in the Appendix.

8.4 Data Exploration and Preparation

Data mining is closely related to exploratory data analysis. Data exploration is essential to mine data. It is not possible to achieve effective results by mining data without having enough knowledge about the data. Before using data, we must know more about it. It is necessary to analyze the data before applying data mining methods. There are some data visualization tools that facilitate this. We present some of these techniques that were applied to explore our data.

In order to get the clean data, we made use of different preprocessing techniques to have a flawless data set. Data cleaning methods attempt to fill in missing values, smooth out noise, and correct inconsistencies in the data (Han and Kamber, 2006).

8.4.1 Null Value Detection

Handling missing values as a part of the preprocessing stage is one of the most important aspects in data mining. Without having a clear data set, achieving an effective result is very difficult and sometimes impossible. To have an almost perfect data set, we used several techniques to fill in the missed values.

The exploration of the data set begins using aggr() function of package VIM (Lang et al., 2012). This package introduces new tools for the visualization of missing or imputed values in R, which can be used for exploring the data and the structure of the missing or imputed values.

The aggr() function calculates or plots the number of missing/imputed values in each feature and the number of missing/imputed values in certain combinations of features. It represents many missed values in a data set.

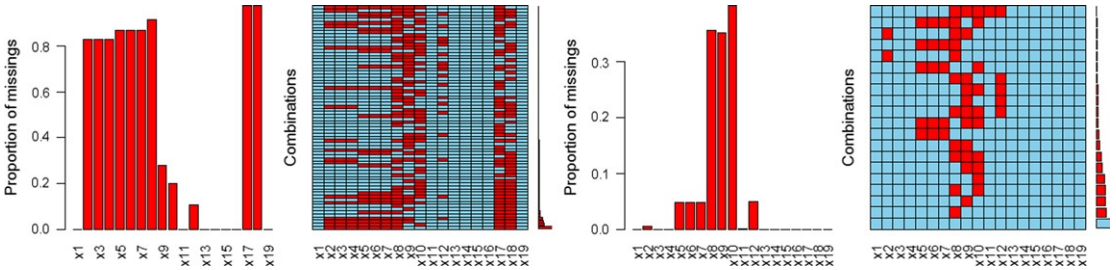


Figure 8.1
Null values in data set before and after eliminating objects and dividing the dataset.

Unfortunately, there was no possibility to estimate all those missed values. The structural characteristics are very important to predict PD, and hence, we had to delete some observations that did not have enough information about the corporate structure. This resulted in a smaller data set. We missed many observations but the remaining ones were so much more complete and efficient. There were also missing values in some behavioral features by structure because the customers were new. To achieve higher accuracy, we divided the data set into two separate data sets and analyzed each set independently. In this chapter, we represent the result of analyzing the one with behavioral features. This data set contains around 1500 observations. Figure 8.1 displays the distribution of null values in the dataset before and after eliminating objects and dividing the data set.

8.4.2 Outlier Detection

Another interesting aspect, albeit a little tricky one in data mining, is the *outlier*. The outlier is a kind of observation, which might deflect the result. A database may contain objects that involve features having very distant values from others. Because the data set contains different types of data, it is wise to investigate the data depending on its type. We used two methods to identify outliers: univariate and multivariate. In the former, we checked each feature separately. Following are some examples of outlier detection in nominal features:

```
> levels(as.factor(mydata[,"x2"]))
[1] "0" "1" "2" "3" "8"

> levels(as.factor(mydata[,"x4"]))
[1] "0" "1" "2" "3" "4" "9" "11" "12"
```

The feature $x2$ must include values: 1,2,3 and the feature $x4$ must contains values:1,2,3,9,11,12. So we changed the incorrect values to null. Binary variables were checked too. For numeric features, we applied the boxplot technique. Boxplot is a simple univariate technique for finding outliers and presenting distribution of data. Figure 8.2 displays boxplots for features. To display all the boxplots in one figure, the values were transformed into the [0,1] domain.

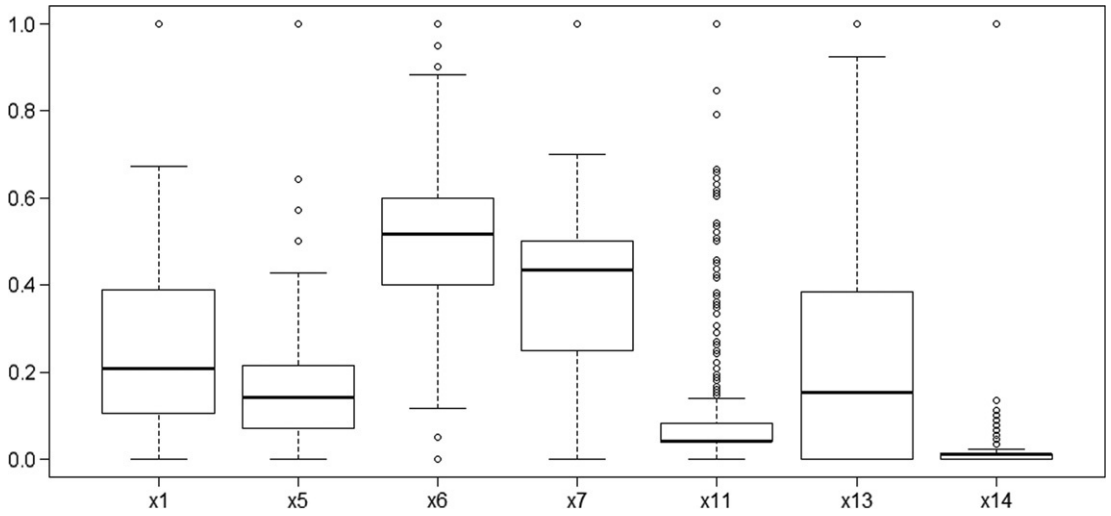


Figure 8.2
Boxplot for numeric features.

To achieve clear results, we applied the boxplot in log scale for ratio features. It makes the distribution of features more homogeneous.

```
# normalization function

norm01 <- function(data,x)
{
For (j in x)
          {
          data [!(is.na(data[,j])),j]=
          (data[!(is.na(data[,j])),j]-min(data[!(is.na(data[,j])),j])) /
          (max(data[!(is.na(data[,j])),j])-min(data[!(is.na(data[,j])),j]))
          }
return(data)
}
```

Using the normalization function, all features are transformed into the [0,1] scale and it is possible to represent the boxplot in one figure.

```
# drawing boxplot

attach (mydata)
c <- c(1,5,6,7,11,13,14)
data_norm <- norm01(mydata,c)
boxplot(data_norm[,c])
c <- c(8,9,17,18)
boxplot(log(mydata[,c]))
```

Figure 8.3 displays the boxplot for ratio features. Hence, the outliers were identified and their values were replaced with null.

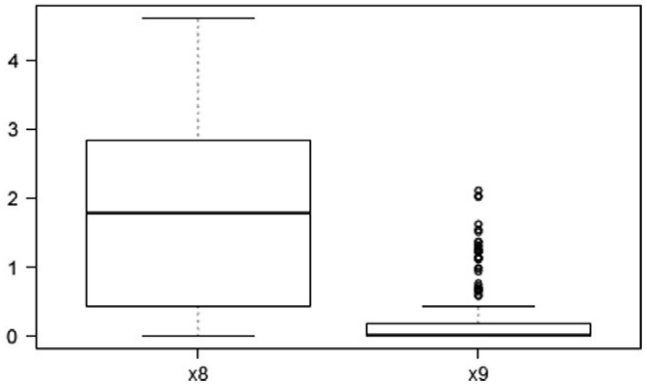


Figure 8.3
Boxplot for ratio features in log scale.

We replaced the value of outliers with null to fill them by an efficient imputation technique in the last step.

One of the multivariate methods for detecting outliers is the distance-based method; some of the related algorithms are *Fazzy clustering*, *SOM*, and *Agglomerative Hierarchical Clustering*. We used the agglomerative hierarchical algorithm because the computation is less complex and it is rather easy to understand. It is a type of clustering technique. In this algorithm, each observation is assumed to be a cluster and in each step, the observations are grouped based on the distance between them (Tan et al., 2005). Each observation that is assigned later has a lower rank. It is seen that the observations with lower rank are outliers because there are dissimilarities between them and the other observations (Torgo, 2011). Therefore, they join into the groups in later stages. For identifying outlier observations, we used the boxplot and the observations with rank under the lower limit were disregarded as outlier data.

Clustering algorithms need distance matrix, and hence, we used the daisy() function of package cluster (Maechler, 2012). As the data set contains mixed types of data the *gower* metric was used to compute distance between objects. The following code is used for outlier ranking:

```
require(cluster)
attach(mydata)

# creating proximity matrix

dist=daisy(mydata[,-19],stand=TRUE,metric=c("gower"),
                  type = list(interval=c(1,5,6,7,11,13,14),ratio=c(7,8,17,18),
                  nominal=c(2,12),binary=c(3,4,10,15,16)))

# clustring objects by agglomerative hierarchical clustering
# this function obtains a score for each object

require(DMwR)
outl=outliers.ranking(dist,test.data=NULL,method="sizeDiff", meth = "ward")
```

After ranking, the objects, whose scores are out of range, are disregarded.

8.5 Missing Imputation

Some cases have missing values, and hence, we have to determine a method for handling missing data. There are two approaches to handle missing data: single imputation and multiple imputation. The former utilizes the information of one variable. For example, any missing value of a variable is replaced by the mean or median of the completed value of the same variable or by the value of the same observation of another variable with the largest association.

However, the later utilizes a new data set which contains the complete data. The missing data are completed using an approximation between them and the complete data. These techniques are more robust than single imputation because of taking more information into account. The considered approach in this case is multiple imputation. We used k nearest neighbors algorithm by knnImputation() function of package DMwR (Torgo, 2012). The function is usable for data sets with both numeric and nominal features. It uses a variant of the Euclidean distance to identify the k nearest neighbors of each object. The distance function is:

$$d_{\{x,y\}} = \sqrt{\sum_{i=1}^{p} \delta_i(x_i, y_i)}$$

Where δ_i was determined by:

$$\delta_i(x_i, y_i) = \begin{cases} 1, & \text{if } i \text{ is nominal and } x_i \neq y_i \\ 0, & \text{if } i \text{ is nominal and } x_i = y_i \\ (x_i - y_i)^2, & \text{if } i \text{ is numeric} \end{cases}$$

Before calculating the distance between objects, the numeric features are normalized. The following code is used for imputation:

```
require(DMwR)
attach(mydata)

# missing data imputation

     co=knnImputation(data_norm, k = 5, scale = T, meth = "weighAvg",
distData = NULL)
```

For large data sets, we can cluster objects into similar groups and then implement imputation. It costs less time to impute by k nearest neighbors.

8.5.1 Relevance Analysis

Data sets may contain irrelevant or redundant features which might make the model more complicated. It takes longer to build more complicated models. Removing redundant features will speed up the building of the model. To understand the nature of the relationship between features, we need to know data types. Our dataset contains mixed types of data.

We used the correlation matrix for continuous and ratio features by plotcorr() function of package ellipse. This function plots a correlation matrix using ellipse-shaped glyphs for each entry (Murdoch et al., 2012). It shows the correlation between features in an easy way. The correlation between two features is independent of their scale. It is very useful to measure association between features. Figure 8.4 displays the correlation plot. The plot is colored for more clarity. The following code displays correlation:

```
# correlation matrix using ellipse-shaped glyphs
library(package="ellipse")
c= c(1,5,6,7,8,9,11,13,14,17,18)
                    plotcorr(cor(mydata[,c]),col=cl<-c("green","red","blue"))
```

There is a weak negative correlation between features $x1$, $x7$ and $x11$, $x18$ that do not need any action and a strong positive correlation between two features $x11, x14$ that we could disregard one of them, but after some consideration we decided to take both of them into account.

We drew the Parallel Coordinate Plot for nominal and binary features. Parallel coordinate plots (PCPs) allow one to see more than four dimensions. The idea is simple: Instead of plotting observations in an orthogonal coordinate system, one draws their coordinates in a system of parallel axes (Härdle and Simar, 2006). One first scales all variables to max = 1 and min = 0. This way of representation is very useful for high-dimensional data. It is, however, sensitive to the order of the variables, because certain patterns in the data can be shown more clearly in one

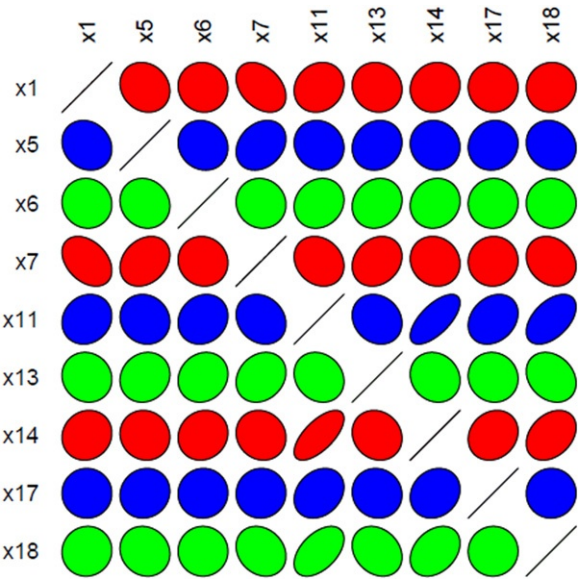


Figure 8.4
Correlation between numeric features.

ordering than in another (Hardle and Simar, 2006). Plotting parallel coordinate was accomplished by ggobi() function of package rggobi.

The rggobi package provides a command-line interface to GGobi, an interactive and dynamic graphics package. Rggobi complements GGobi's graphical user interface, providing a way to fluidly transition between analysis and exploration, as well as automating common tasks (Lang et al., 2012). The ggobi() function conveniently retrieve ggobi dataset:

```
# rggobi for nominal and binary data

c= c(1,seq(5,9,1),11,13,14,17,18)
data_norm=norm01(mydata[,c],c)
c=c(2,12)
data_nomi[,c]=mydata[,c]/max(mydata[,c])
c=c(3,4,10,15,16,19)
newdata=cbind(data_norm,data_nomi,mydata[,c])
library(rggobi)
attach(newdata)
gd <- ggobi(newdata)[1]
```

Before drawing the PCP, we divided the dataset to some more apprehended subsets. Figure 8.5 displays the PCP.

Considering the figure, we did not observe any correlated feature in this plot.

8.5.2 Data Set Balancing

Most of data sets in the real world have unbalanced class distributions that cause some problems in classification models. Most learning algorithms tend to omit the smaller class because it is not supported statistically. Hence the accuracy will rise unrealistically. We sampled train

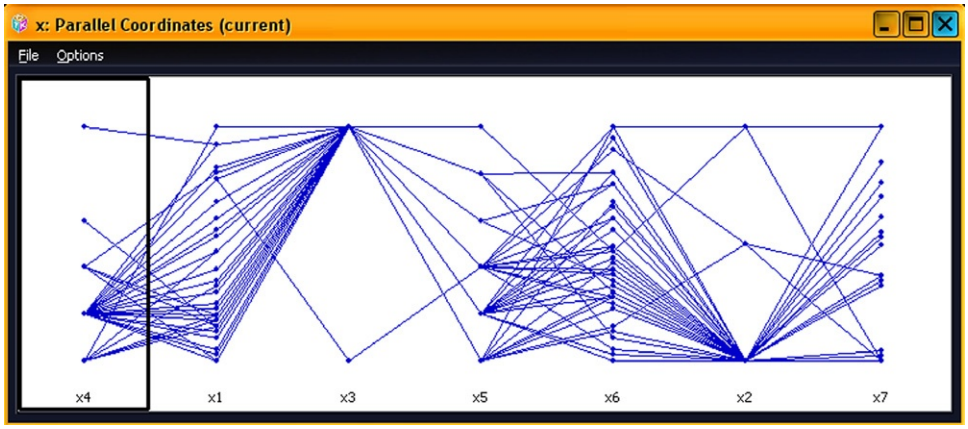


Figure 8.5
Parallel coordinate plot.

and test data sets from all available objects randomly and used the SMOTE function of package DMwR for the training data to overcome these problems. The test set included around 500 observations and the balanced train set contained around 1500 observations.

The SMOTE function handles unbalanced classification problems using the SMOTE method. Namely, it can generate a new "SMOTEd" data set that addresses the class unbalance problem. It generates new examples of minority classes artificially using the nearest neighbors of this class of elements and under samples the majority class to balance the training data set (Torgo, 2011). The following code is used for balancing the training data set:

```
library(DMwR)
attach(mydata)
tr<-sample(nrow(m), round(nrow(mydata)*0.7))
train=mydata[tr,]
test=mydata[-tr,]
data_smot=train
data_smot$defn <- factor(ifelse(data_smot$defn == 1, "def", "nondef"))
data_samp <- SMOTE(defn ~ ., data_smot, k=5,perc.over = 500)
```

Figure 8.6 shows the data distribution before and after balancing. To represent the data distribution in two statuses, we utilized the Multi Dimensional Scaling (MDS) method.

MDS is a method based on proximities between objects, subjects, or stimuli used to produce a spatial representation of these items. Proximities express the similarity or dissimilarity between data objects (Härdle and Simar, 2006). It represents the objects as the points such that the distances between them are nearly proximities between same objects. To plot objects in two dimensional space, the following code can be used:

```
library(cluster)
dist=daisy(data_samp[,-19],stand=TRUE,metric=c("gower"),
type=list(interval=c(1,5,6,7,11,13,14),
ratio=c(7,8,17,18), nominal=c(2,12),
binary=c(3,4,10,15,16))),11,15,16)))
loc=cmdscale(d,k=2)
```

Figure 8.6
Data class distribution before and after balancing.

```
x=loc[,1]
y=loc[,2]
plot(x,y,type="n")
text(x,y,labels=as.expression(as.numeric(data_samp[,19])),col=as.numeric(data_samp
[,19]))
```

8.5.3 Feature Selection

The size of the data set in many applications is very large. In the real world, the relevant features are not predetermined and more data sets contain both relevant and irrelevant features that might cause some drawbacks. Increasing running time, achieving poor results, and complex patterns are all examples of these problems. The aim of feature selection is to find a promising subset of features, which are very close to the original data set.

As each feature has a special effect on the target feature, using a powerful feature selection technique is necessary. To work with a large data set, we made use of the random feature selection method, and also *randomForest* function of the package *randomForest* is used to evaluate the weight of each feature (Breiman et al, 2012). The proposed method picks a random object from the samples and generates several trees, and on the basis of the accuracy of classifier or error ratio, features are weighted. This approach has two steps.

The first step generates a tree with all features named as primary tree, then in each iteration one feature is eliminated. Consequently, the mean value of differences between the error rate for each feature with respect to each new generated tree and the primary tree will be calculated. On the basis of the accuracy obtained from the previous stage we are able to assign a higher weight to the features with better accuracy.

The second step or second measurement calculates the mean value of decreased impurity respect to each feature. This step is inspired by Gini Index. After running this calculation, we have a weighted $n * 2$ matrix: where n is the number of features and the 2 columns are related to two measures, whose values represent the importance of each feature.

We should notice that the number for the random selection must be chosen to achieve the best accuracy. The following codes are used to make the table of important features:

```
attach(data_samp)
library(randomForest)
set.seed(4543)
data.frame(intbo_samp)
rf<-randomForest(x19~ ., data=data_samp,ntree=700, importance=TRUE, proximity=TRUE)
importance(rf, type=1, scale=TRUE)
varImpPlot(rf)
```

The *type* argument specifies the type of importance measure (1: Mean decrease in accuracy, 2: Mean decrease in Gini). According to experts, we want to select the most relevant features on the basis of the first measure. The following shows the results of the importance function:

	MeanDecreaseAccuracy
*x*1	0.9161477
*x*2	0.3610998
*x*3	0.3156747
*x*4	0.5687706
*x*5	0.7720659
*x*6	0.7910566
*x*7	0.9395147
*x*8	0.9603374
*x*9	0.9549458
*x*10	0.4534945
*x*11	0.8425741
*x*12	0.7628996
*x*13	0.7243090
*x*14	0.7692923
*x*15	0.6654196
*x*16	0.7226551
*x*17	0.7081333
*x*18	0.9568865

After obtaining the importance score of the features, we should choose the most important features, which are already based on a certain threshold. This can be accomplished using a backward elimination method.

To choose them, we used rfcv() function of package *randomForest*. This function shows the cross-validated prediction performance of models with sequentially reduced number of predictors (ranked by variable importance) via a nested cross-validation procedure (Breiman et al., 2012). The above mentioned are coded below:

```
fo=rfcv(data_samp[,−19],data_samp[,19], cv.fold=10, scale="log", step=0.9)

best <− which.max(fo$error.cv)
plot( fo$n.var,rfo$error.cv, type= "h", main = "importance",
xlab="number of features", ylab = "classifier error rate")
axis(1, best, paste("best", best, sep="\n"), col = "red", col.axis = "red")
```

As Figure 8.7 shows, the best number of features is 12, so these 12 features with the highest importance are chosen to build the model: *x*1, *x*4, *x*5, *x*6, *x*7, *x*8, *x*9, *x*11, *x*12, *x*13, *x*14, *x*18.

8.6 Modeling

Classification is one of the data analysis forms that predicts categorical labels (Han and Kamber, 2006). Classification methods are used to classify the new observations whereas the class labels are known. In other words, classification is a procedure for finding a model (Mathematical function) to predict a class label for those observations that are not clearly

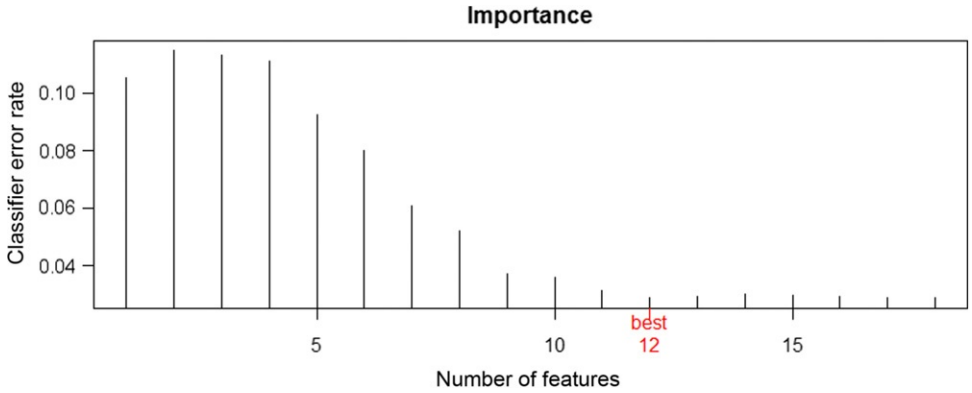


Figure 8.7
Classification error rate using different numbers of features.

labeled. This model can be illustrated in different ways such as a decision tree, classification rules, and neural network.

We used the decision tree technique to obtain a model for the loans probability of default classification. A decision tree has a structure as a tree in which each node represents a feature measurement, each branch is a result for each measurement and the class labels are shown by leaves. Decision trees can be easily converted to classification rules.

Classification method is a two-step process which makes use of a set of clearly labeled data in the first or learning step. This phase, also called the supervised learning phase, provides a classification model (Algorithm) by analyzing the data from the training data set. The algorithm generated by the first step will be run on nonlabeled observation or testing data set in the second phase. Function *rpart*() is used to build a decision tree (Therneau et al., 2010). The following code is used to implement *rpart*():

```
library(rpart)
sdf=data.frame(train)
fit=rpart(train$x19 ~.,data=sdf,method="class")
```

Since the class label ($x19$) is a factor, the method ="class" is assumed. The *rpart* package implements a cost complexity pruning (cp) to stop growing the tree. It produces a set of subtrees and chooses the tree which has the best comparison between cp, accuracy and tree size. The function *printcp*() of this package displays the cp table for fitted *rpart* object. The result of this function is shown below.

```
> printcp(fit)

Classification tree:
rpart(formula = train$x19 ~ ., data = sdf, method = "class")
```

Variables actually used in tree construction:

[1] *x*11 *x*12 *x*13 *x*14 *x*18 *x*8 *x*9

Root node error: 552/1472 = 0.375

$n = 1472$

CP	nsplit	rel error	error	xstd
1 0.731884	0	1.00000	1.00000	0.033649
2 0.022947	1	0.26812	0.26993	0.020964
3 0.018116	5	0.16667	0.21377	0.018874
4 0.016304	6	0.14855	0.17754	0.017327
5 0.014493	7	0.13225	0.16848	0.016909
6 0.010000	8	0.11775	0.12138	0.014487

The last tree is the best one and will be chosen as the final model. This tree is illustrated in Figure 8.8. The command to plot the tree is as follows:

```
plot(fit, uniform=TRUE,main="Classification Tree")
text(fit, use.n=TRUE, cex=1)
```

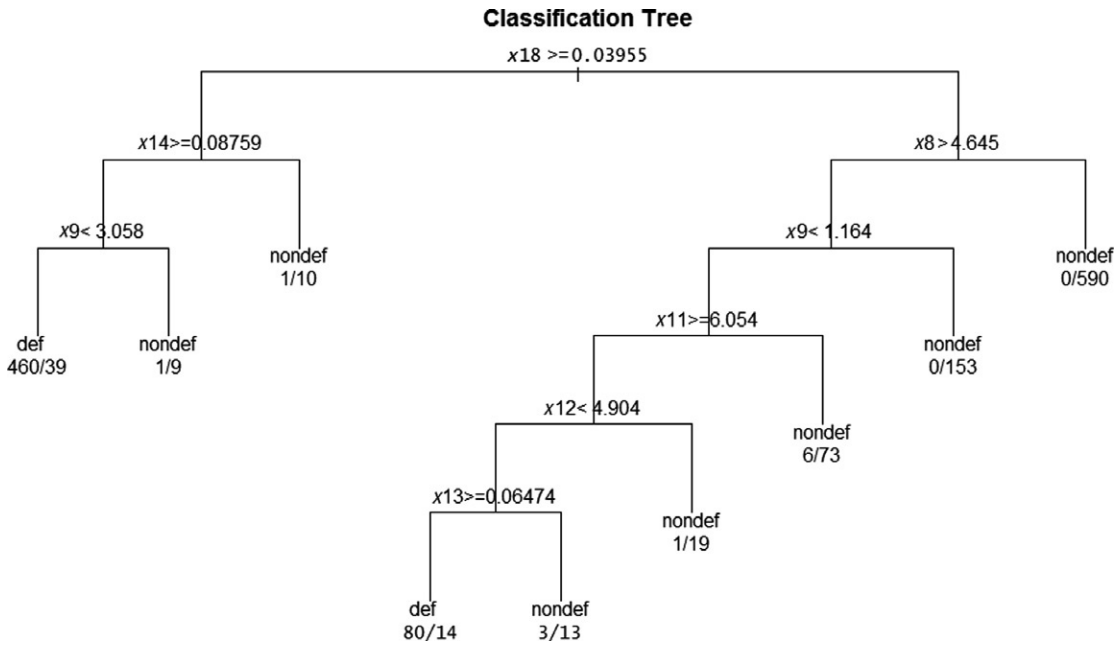


Figure 8.8
Classification Tree.

8.7 Model Evaluation

To choose the best model, it is necessary to make use of model evaluation techniques. Model evaluation is a subprocess of the modeling process. There are several variations in evaluation.

We used the combination of holdout and cross-validation methods. In the first step, we divided the data set into two subsets: training and test data sets in which each observation has an equal chance to be selected for each subset. Two-thirds of the data were selected for training data. The test data are independent of the training data. The training observations were used to build the model by using 10-fold cross-validation. In this way, the training data set partitioned into 10-folds. In each iteration, onefold was assumed as a test data. The average error rate was considered as the final error rate after 10 iterations. These tasks were accomplished by *rpart*() function.

8.8 Finding and Model Deployment

The deployment in classification models refers to the predict class labels of new objects. We tested the model for new objects from the test data set by the predict() function. The following displays the code and some predicted test data as a sample:

```
> sdfto=data.frame(test)
> a=predict(fit,sdfto)
> a
```

	def	nondef
6	0.18750000	0.81250000
8	0.00000000	1.00000000
10	0.00000000	1.00000000
15	0.92184369	0.07815631
22	0.00000000	1.00000000

We assumed each loan with predicted PD greater than 50%, as a loan in default. Then we calculated the confusion matrix by this code:

```
> table(predict(fit, test, type="class",na.action=na.pass), test[, "x19"])
```

	def	nondef
def	43	5
nondef	11	391

```
>
```

Table 8.2 shows the values of the common metrics that were calculated from the confusion matrix.

Table 8.2 The Measures from the Confusion Matrix

Accuracy	Precision	TP	FP
0.96	0.8	0.9	0.03

$$\text{TP rate} = \frac{\text{True defaults}}{\text{Total defaults}}$$

$$\text{FP rate} = \frac{\text{False defaults}}{\text{Total nondefaults}}$$

$$\text{Precision} = \frac{\text{True defaults}}{\text{True defaults} + \text{False defaults}}$$

$$\text{Accuracy} = \frac{\text{True defaults and True nondefaults}}{\text{Total test set}}$$

8.9 Lessons and Discussions

In this chapter, we presented a framework to identify effective features on the probability of default in corporate loans as well as the way to measure their importance. These features were used to predict the PD for each nonlabeled loan for a one-year horizon. PD estimation can help banks to prevent expected loss. We accomplished these tasks on the basis of the data mining approach and collected the needed data from a large bank for our method. As the preprocessing stage is very important and also a time consuming step, we used both classification and clustering techniques to make the data usable in an efficient way.

After the preprocessing step, a decision tree classifier was implemented in order to predict the class label for new loans regarding their potential ability of PD. From the selected 18 independence features, the 12 most important ones were chosen to build the model. We utilized some R packages to prepare data and to build the classification model. The R packages are very useful to implement data mining techniques as well as visualization methods.

Appendix Selecting Best Features for Predicting Bank Loan Default

Feature Description
$x1$ age of company (in years)
$x2$ ownership (1: public, 2: cooperative, 3: government)
$x3$ type (1: company, 2: institute)
$x4$ stock type (1: public joint stock, 2: private joint stock, 3: limited LTD, . . .)
$x5$ number of managers
$x6$ managers' average age (in years)
$x7$ managers' total stock

$x8$ the ratio of asset to capital
$x9$ the ratio of collateral to loan
$x10$ activity background in this segment (0: without any background, 1: with background)
$x11$ the remaining months for borrowers to withdraw the obligations
$x12$ collateral code
$x13$ duration between default so far (in years)
$x14$ duration between the last payment so far (in years)
$x15$ payment after due date (0: no, 1: yes)
$x16$ default in past years (0: nondefault, 1: default)
$x17$ the ratio of past debt to previous credit value
$x18$ the ratio of default to previous credit value
$x19$ class label (0: nondefault, 1: default)

References

BCBS, 2001. The Consultative Document: The Internal Ratings-Based Approach. Bank for International Settlements.

BCBS, 2005. Working Paper No. 14: Studies on the Validation of Internal Ratings Systems. Bank for International Settlements.

BCBS, 2006. International Convergence of Capital Measurement and Capital Standards.

Breiman, L., Cutler, A., Liaw, A., Wiener, Matthew, 2012. Breiman and cutler's random forests for classi_cation and regression. R package version 3.1-46.

Han, J., Kamber, M., 2006. Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco.

Härdle, W., Simar, L., 2006. Applied Multivariate Statistical Analysis. Springer, Berlin Heidelberg.

Lang, D.T., Swayne, D., Wickham, H., Lawrence, M., 2012. Interface between r and ggobi, R package version 2.1.19.

Maechler, M., 2012. Cluster analysis extended rousseeuw et al, R package version 1.14.2.

Murdoch, D., Chow, E.D., Celayeta, J.M.F., 2012. Functions for drawing ellipses and ellipse-like con_dence regions, R package version 0.3-7.

Tan, P.-N., Stainbach, M., Kumar, V., 2005. Introduction to Data Mining. Pearson Education, Boston.

Therneau, T.M., Atkinson, B., Ripley, B., 2010. Recursive partitioning, R package version 3.1-54.

Torgo, L., 2011. Data Mining with R: Learning with Case Studies. Chapman Hall/CRC, Boca Raton.

Torgo, L., 2012. Functions and data for "data mining with r" R package version 0.2.3.

CHAPTER 9

A Choquet Integral *Toolbox and Its Application in Customer Preference Analysis*

Huy Quan Vu, Gleb Beliakov, Gang Li
School of Information Technology, Deakin University, Melbourne, Victoria, Australia

9.1 Introduction

Customer preference is a fundamental factor for business as it governs a customer's intention of whether to buy or not. Rapid and accurate identification of customer preferences are essential to production development and marketing in business management nowadays. Researchers have long sought the preferences of customers in supporting the strategic plan of businesses (Granot et al., 2010; Yang et al., 2012).

Despite considerable efforts, business managers who expect to gain insight into customer preferences for effective planning still face a significant barrier. Customer decision making typically involves the comparison of two or more alternatives, which are evaluated against various factors according to their priorities. This operation is referred to as the *multicriteria decision making* (MCDM) process (Beliakov et al., 2007). The understanding of the MCDM process is an effective strategy for improving the product and designing focused marketing strategies. For instance, a customer may give a product a low rating on the *quality* and *service* criteria but still select it based on its *cheap* price. Other customers may select a product only if it can satisfy both the *quality* and the *service* criteria. The insight into how the criteria interact with each other in guiding a customer's intention can provide business managers with insightful knowledge into customer preferences. This often requires the simultaneous consideration of multiple criteria, whereas other techniques do not pay attention to this. Therefore, there remains a strong demand for a technique that allows the exploration of customer MCDM process.

During the past decades, *fuzzy decision support* has been an emerging area that had provided effective methods in modeling the MCDM process. *Choquet Integral* has been identified as a promising method (Beliakov et al., 2007). The *Choquet Integral* method is an aggregation technique defined with respect to a *fuzzy measure* that allows it to assign importance to all possible groups of criteria to model the MCDM process. It takes account of the dependencies and the interaction among variables, which can be interpreted via two metrics (*Shapley* value

Data Mining Applications with R

and *Interaction Index*). The *Shapley* value estimates the overall importance of each criterion (Grabisch and Lange, 2007) and the *Interaction Index* measures the interaction among criteria (Grabisch and Labreuche, 2007). With these advantages, the *Choquet Integral* offers a flexible method for relationship modeling between multiple criteria in the decision-making process.

Aiming to address the challenges of MCDM modeling for discovering customer preferences, the objectives of this chapter are:

- To deploy a new aggregation function, the *Choquet Integral*, for effective modeling of the MCDM process. Using the *Shapley* value and the *Interaction Index*, the insight into the preferences of customers, along with interaction among criteria, can be explored.
- To introduce a new toolbox, the `Rfmtool` package, for customer preferences analysis, based on the *Choquet Integral*. The operations of the toolbox can be performed in the freely available *R* environment.
- To demonstrate the practical advantages of the *Choquet Integral* and the use of `Rfmtool` package, by using a case study to model the behavior of international travelers in the hotel selection process.

Having set out these aims for undertaking this work, the rest of the chapter is organized as follows. Section 9.2 provides the background of aggregation functions, the *Choquet Integral*, as well as its metrics. Section 9.3 describes the `Rfmtool` toolbox and its operation. Section 9.4 presents the experimental design and the data collection process. Evaluation results of the *Choquet* model and analysis of the findings with reference to customer preferences and preference changes are also provided in Section 9.4. Finally, Section 9.5 summarizes the chapter and provides possibilities for future applications.

9.2 Background

This section provides the background on the formulation of aggregation functions, along with a description of the *Choquet Integral* for modeling the MCDM process.

9.2.1 Aggregation Functions

The process of MCDM involves the comparison of two or more alternatives. Each alternative is evaluated against multiple or n criteria according to the priorities of the decision maker. The degree, to which an alternative satisfies a particular criterion, corresponds to a utility value. The scores must then be combined somehow to give an overall rating on that alternative. For instance, when a customer is planning to buy a new car, with so many possibilities, he has to evaluate every offer against n criteria, such as *price*, *efficiency*, *service*, and *warranty conditions*, to identify the most suitable option. The purpose of aggregation functions is to

combine multiple numerical inputs, usually interpreted as *degrees of membership*, *strength of evidence*, etc., into a single numerical value, which in some sense represents all the inputs.

A simple example of aggregation function is the *arithmetic mean* (AM). This is commonly employed in everyday language as the "*average*," and a more general form of the AM is the *weighted arithmetic mean* (WAM), which takes the importance of criteria into account.

The AM of n values is the function:

$$\mathrm{AM}(x) = \frac{1}{n}(x_1 + x_2 + \cdots + x_n) \tag{9.1}$$

The WAM is a linear $\sum_{i=1}^{n} w_i = 1$ function with respect to a positive weighting vector $\mathbf{w}$ with:

$$\mathrm{WAM}_w(x) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{9.2}$$

In the context of modeling the MCDM process, the above aggregation functions have a limitation mainly from their natural assumption that input criteria are independent of each other (Beliakov et al., 2007). This is not always true in reality, where the independence of criteria cannot be assumed. Some interaction among different criteria does exist, including the *independence*, the *complement*, and the *correlation* (Grabisch and Roubens, 2000). To take all of the interaction among the attributes into account, it has been proposed to use the *fuzzy measure* in the calculation of the overall aggregation result (Grabisch and Roubens, 2000). The *Choquet Integral* is one such function, whose detail is provided in the following subsection 9.2.2.

9.2.2 Choquet Integral

Choquet Integral is an aggregation function defined with respect to the fuzzy measure. A fuzzy measure is a set function, acting on the domain of all possible combinations of a set of criteria. The complexity is therefore exponential of 2^n subsets, where n is the number of criteria.

Formally, let $N = \{1,2,\ldots,n\}$, a *general fuzzy measure* is a set function $v: 2^N \rightarrow [0,1]$ which is monotonic (i.e., $v(A) \leq v(B)$ whenever $A \subset B$) and satisfies $v(\varnothing) = 0$ and $v(N) = 1$. Since subset $A \subseteq N$ can be considered as a group of criteria, $v(A)$ can represent the importance or weight of this group.

The use of fuzzy measure allows the *Choquet Integral* to assign importance to all possible groups of criteria, and thus offers a much greater flexibility for aggregation. Both the inputs and the outputs are usually defined on the unit interval [0,1]; however, other choices are also possible.

Let $x_1, x_2, \ldots, x_n$ be a set of criteria, and v, its fuzzy measure. The *Choquet Integral* computed from the *general fuzzy measure* is given by:

$$C_v(x) = \sum_{i=1}^{n} [x_i - x_{i-1}] v(H_i). \tag{9.3}$$

where $x_0 = 0$ by convention, and $H_i = \{i, \ldots, n\}$ is the subsets of indexes of the $n - i + 1$ largest components of x.

The advantages of the *Choquet Integral* is based on the use of fuzzy measure in its computation, which allows it to consider the interaction between all possible combinations of criteria. The following example will illustrate this advantage.

Example 1 *A customer wants to select a product based on three criteria - price, quality, and service. The options have been narrowed down to four products with the utility values in Table 9.1.*

The customer starts his/her decision-making process by giving a ranking to the products. Let us assume that, after a reasoning process, the customer suggests the ranking order $A > B > C > D$.

Here, a question arises: Can an additive mode such as WAM produce this partial ranking? Let w_1, w_2, w_3 be the weight of the product criteria. If the WAM model is used, the order $A > B$ holds when $w_1 > w_2$, and the order $C > D$ holds when $w_1 < w_2$. There is no such weight for WAM to produce the proposed ranking order.

Suppose the Choquet Integral is used to model the decision-making process of this customer with the general fuzzy measure and it arrives at the solution in Table 9.2. Then, the Choquet Integral values are computed and shown in *Table 9.3.*

The ranking based on the Choquet Integral values satisfies the ranking order given by the customer ($A > B > C > D$). It should be noted that the fuzzy measure weight was assigned to

Table 9.1 Utility Values

	Price	Quality	Service
Product A	0.8	0.5	0.9
Product B	0.5	0.8	0.9
Product C	0.5	0.8	0.4
Product D	0.8	0.5	0.4

Table 9.2 Weight of Fuzzy Measure

Fuzzy Measure	Weight
$v(\{\varnothing\})$	0
$v(\{\text{price}\})$	0.3
$v(\{\text{quality}\})$	0.4
$v(\{\text{service}\})$	0.75
$v(\{\text{price,quality}\})$	0.2
$v(\{\text{price,service}\})$	0.9
$v(\{\text{quality,service}\})$	0.6
$v(\{\text{quality,qualityservice}\})$	1

Table 9.3 Choquet Integral Values

	Product A	Product B	Product C	Product D
C(.)	0.85	0.76	0.54	0.51

each group of criteria rather than to individual criterion. It offers a flexible way of modeling the complex decision-making process of the customer.

9.2.3 Fuzzy Measure Representation

Beside the *general* representation discussed in the previous section, the fuzzy measure is also known under the name of the *Mobius* representation (Chateauneuf and Jaffray, 1989). Its values can be interpreted as a measure of the interaction between the groups of criteria.

Let v be a *general fuzzy measure* for every $A \subseteq N$, the *Mobius fuzzy measure* can be computed as:

$$M(A) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} v(B) \tag{9.4}$$

The transformation to *Mobius* representation is invertible, and in one-to-one correspondence with the *general* representation. The fuzzy measure v can be determined from the *Mobius* representation following an inverse operation, called the *Zeta transform* (Beliakov et al., 2007):

$$v(A) = \sum_{B \subseteq A} M(B), \forall A \subseteq N \tag{9.5}$$

The *Choquet Integral* can also be computed from the *Mobius fuzzy measure* as:

$$C_{M_v}(\mathrm{x}) = \sum_{A \subseteq N} M(A) \min_{i \in A} x_i \tag{9.6}$$

Various representations of fuzzy measure allow the aggregation function, like the *Choquet Integral*, to work interchangeably, depending on which is more convenient.

A fuzzy measure is usually stored in an array of size 2^n, following a binary indexing system (Beliakov et al., 2007), which makes it convenient for the computational purpose.

Example 2 *In case $N = \{1, 2, 3\}$, there are $2^3 = 8$ possible combination of criteria. The fuzzy measure representation for 8 groups of criteria is defined as:*

$$[v\{\phi\}, v\{1\}, v\{2\}, v\{1,2\}, v\{3\}, v\{1,3\}, v\{2,3\}, v\{1,2,3\}]$$

Assume that it takes the values:

$$[0, 0.3, 0.5, 0.6, 0.4, 0.8, 0.9, 1]$$

We can interprets these values as, for instances: $v\{1\}=0.3$ *is the weight of the group containing criterion 1;* $v\{2, 3\}=0.9$ *is the weight of the group containing criteria* 2 *and* 3; $v\{1, 2, 3\}=1$ *is the weight of the group containing criteria 1,* 2, *and* 3.

Example 3 *Given a general fuzzy measure with the values*:

$$[0, 0.3, 0.5, 0.6, 0.4, 0.8, 0.9, 1]$$

Following Equation (9.4), its equivalent values in Mobius representation are:

$$[0, 0.3, 0.5, -0.2, 0.4, 0.1, 0, -0.1]$$

Assume we have an input $x=(0.4, 0.3, 0.8)$. *The Choquet Integral value of the input* x *can be computed with the general fuzzy measure using Equation (9.3), or with the Mobius fuzzy measure using Equation (9.6). Either way produces the identical Choquet Integral value of 0.54.*

There is an issue with the fuzzy measure that the model complexity grows exponentially (2^n) with the input criteria n. Grabisch (Grabisch, 1997) developed the concept of a *k-additive* fuzzy measure to deal with this problem by reducing the number of variables to define the fuzzy measure. The interactions between criteria are only considered for subsets of k elements or less, which allows for a trade-off between modeling ability and complexity. Users can decide how complex a fuzzy measure is to be considered by choosing a *k-additive* value ($1 \leq k \leq n$). It should be noted that check WAM is equivalent to the *Choquet Integral* with 1-*additive* fuzzy measure. When $k=n$, the fuzzy measure is said to be unrestricted. The influence of *k-additive* on the modeling performance of the *Choquet Integral* will be discussed further in Section 9.4.3.

9.2.4 Shapley Value and Interaction Index

Shapley value measures the overall importance of each criteria in terms of its contribution to the score of each group of criteria. Let v be a fuzzy measure. The *Shapley* value for every input $i \in N$ is

$$\phi_i = \sum_{A \subseteq N \setminus \{i\}} \frac{(n - |A| - 1)!|A|!}{n!} [v(A \cup \{i\}) - v(A)] \tag{9.7}$$

The *Shapley* value is the vector $\Phi(v)=\phi_1, \ldots, \phi_n$. The index ϕ_i can be interpreted as a kind of average value of the contributions of i criteria in all groups.

The *Interaction Index* values reflect the behaviors of criteria in groups, and they can be considered as a measurement of the interaction between criteria in the decision-making process. For every set $A \subseteq N$:

$$I(A) = \sum_{B \subseteq N \setminus A} \frac{(n - |B| - |A|)!|B|!}{(n - |A| + 1)!} \sum_{C \subseteq A} (-1)^{|A \setminus C|} v(B \cup C) \tag{9.8}$$

It should be noted that $I(A) \in [-1, 1]$ and the computation of the *Interaction Index* can include all combination of criteria. However, the *Interaction Index* I_{ij} for each pair $A = \{i, j\}$ of criteria is used most often, due to its convenience in interpretation.

If there exists a correlating relationship between a pair of criteria x_i and x_j, then we have $I_{ij} < 0$. If they have a complementary relationship, we have $I_{ij} > 0$. When x_i and x_j are independent or have little interaction, we have $I_{ij} \approx 0$. The use of the Shapley and Interaction Index for customer preference analysis will be demonstrated in section 9.3.

9.3 Rfmtool Package

This section provides a detailed description of the `Rfmtool` software package, which was developed to perform operations on the *Choquet Integral* for preference analysis applications. This section begins with the installation process, followed by an overall description of the toolbox. The usage of the toolbox for the preference analysis is then described with an example to illustrate.

9.3.1 Installation

The `Rfmtool` toolbox is distributed as a standard *R* package containing source code files, data sample, and examples. The toolbox can be downloaded from `www.tulip.org.au/resources/rfmtool`. There are two distribution files; the file `Rfmtool. zip` is for installation and running on the *Windows* platform and the file `Rfmtool.tar.gz` is for the *Linux* platform.

It should be noted that the routine code for fuzzy measure operation is mainly written in *C/C++* (Beliakov, 2007). The source files of *Rfmtool* also contain the source code of the `Lp_solve` library (Berkelaar and Buttrey, 2011), because the operation of `Rfmtool` depends on this library for solving linear, integer, and mixed integer programs. We supplied a "*wrapper*" function, which allows all of the operations and data input/output to be performed in the *R* environment.

After downloading, *Windows* users can install this toolbox by selecting the *installation package from the local zip files* menu bar in the *R* graphic user interface. For *Linux* users, the package can be installed by typing the command:

```
$ R CMD INSTALL Rfmtool.tar.gz
```

Note that the `Rfmtool` package requires the C/C++ compiler `gcc` version 4.6.0 or above to be available for installation in *Linux*.

Users can verify if the package has been installed successfully by calling a testing function in the *R* environment:

```
> # Load the package
> library("Rfmtool")
> # Call the testing function.
> fm.test()
> # The following output is expected.
[1] "Choquet Integral from general fuzzy measure"
[1] 0.62
[1] "Choquet Integral from Mobius fuzzy measure"
[1] 0.62
[1] "Mobius transform"
[1] 0.0 0.3 0.5 -0.2 0.4 0.1 -0.2 0.1
[1] "Zeta transform"
[1] 0.00 0.18 0.15 0.28 0.23 0.48 0.56 1.00
[1] "Shapley value"
[1] 0.2833 0.3333 0.3833
[1] "Interaction Index"
[1] 0.00 0.18 0.15 0.28 0.23 0.48 0.56 1.00
```

9.3.2 Toolbox Description

The following core functions are provided in the `Rfmtool` package:

`fm.fitting` (`data`, `kadd`): This estimates the fuzzy measure based on the empirical data. The first input argument (`data`) is the empirical data set in pairs (x_i,y_i), where $x_i \in [0,1]^n$ is a vector containing the utility values of n input criteria $\{x_{i1},x_{i2},\ldots,x_{in}\}$, $y_i \in [0,1]$ is a single aggregated value given by decision makers. The second argument (`kadd`) is the value of *k-additive* for reducing the complexity of fuzzy measures (Grabisch, 1997). `kadd` is defined as an optional argument, and its default value is `kadd`$=n$. The estimated fuzzy measure v is in the *Mobius* representation and is stored in an array of size 2^n following a binary ordering.

`fm.Choquet`(`x`, `v`): This calculates the *Choquet Integral* based on the general fuzzy measure. The argument `x` is a vector containing the input criteria, and `v` is a vector of the *general fuzzy measure*.

`fm.ChoquetMob`(`x`, `Mob`): This calculates the *Choquet Integral* based on the *Mobius* fuzzy measure. The argument `x` is a vector containing the input criteria, and `Mob` is a vector of the *Mobius fuzzy measure*.

`fm.Mobius`(`v`): This transforms the general fuzzy measure `v` into the *Mobius* representation.

`fm.Zeta`(`Mob`): This transforms the *Mobius* fuzzy measure `Mob` to the general representation.

`fm.Shapley`(`v`): This calculates the *Shapley* value from the general fuzzy measure `v`.

`fm.Interaction`(`Mob`): This calculates the *Interaction Index* from the *Mobius* fuzzy measure `Mob`.

During the operation, users can view a list of all functions included in this toolbox for reference at any time by typing:

```
> fm()
```

User can view the manual by using the help functions.

```
> help(Rfmtool)
```

9.3.3 Preference Analysis Example

We explore the operations of the `Rfmtool` package via a simple example. Suppose a company wants to know which preference of customers to support during their product development task. A data set was collected with customer ratings on three product selection criteria (`price`, `quality`, and `service`), and an overall rating which represents the final decision of the customer. The rating values are normalized into the unit interval [0,1], as shown in Table 9.4:

The data set is saved in a file `data.txt`, containing only the rating values. We can apply the `Rfmtool` to this data set as follows:

```
> # Load data from files. Assume that the file data.txt is
> # stored in the working directory.
> data <- as.matrix(read.table("data.txt"))

       V1   V2   V3   V4
 [1,] 0.6  0.9  0.5  0.8
 [2,] 0.8  0.3  0.4  0.3
 [3,] 0.2  0.6  0.3  0.4
 [4,] 0.7  1.0  0.5  0.8
 [5,] 0.3  0.6  0.8  0.7
 [6,] 0.5  0.4  0.3  0.5
 [7,] 0.8  0.9  0.7  0.9
 [8,] 0.3  0.8  0.9  0.6
 [9,] 0.6  0.8  0.4  0.5
[10,] 0.3  0.6  0.2  0.2
```

Table 9.4 A Customer Rating Data Set

Record ID	Price	Quality	Service	Overall Rating
R_1	0.6	0.9	0.5	0.8
R_2	0.8	0.3	0.4	0.3
R_3	0.2	0.6	0.3	0.4
R_4	0.7	1.0	0.5	0.8
R_5	0.3	0.6	0.8	0.7
R_6	0.5	0.4	0.3	0.5
R_7	0.8	0.9	0.7	0.9
R_8	0.3	0.8	0.9	0.6
R_9	0.6	0.8	0.4	0.5
R_{10}	0.3	0.6	0.2	0.2

```
> # Note that the data.txt file contain only the rating values.
> # The record ID and labels for product features are not included.

> # Estimate fuzzy measure (in Mobius representation) from
> # empirical data sets. Here, the kadd value is not specified,
> # thus kadd is assigned with the default value as kadd = n = 3.
> Mobfuzzy <- fm.fitting(data)
[1] 0.00 0.00 0.50 0.25 0.50 0.00 −0.50 0.25
> # Note that estimated fuzzy measure is in Mobius representation
> # and it is stored in an array containing 2^3 = 8 values.

> # Transform the estimated Mobius fuzzy measure into general
> # fuzzy measure by calling Zeta transform function.
> Genfuzzy <- fm.Zeta(Mobfuzzy)
[1] 0.00 0.00 0.50 0.75 0.50 0.50 0.50 1.00

> # User can try converting the general fuzzy measure back to
> # Mobius fuzzy measure by calling Mobius transform function.
> fm.Mobius(Genfuzzy)

[1] 0.00 0.00 0.50 0.25 0.50 0.00 -0.50 0.25
> # The result is expected to be the same as in Mobfuzzy.

> # User can try computing the Choquet Integral from general
> # fuzzy measure for an input x as below.
> x <- c(0.8, 0.4, 0.6)
> fm.Choquet(x,Genfuzzy)
[1] 0.5

> # Compute the Choquet Integral from Mobius fuzzy measure
> # for the input x.
> fm.ChoquetMob(x,Mobfuzzy)
[1] 0.5
> # Note that the computed Choquet Integral values from
> # general and Mobius fuzzy measure are identical.

> # Compute Shapley values
> ShapleyVal <- fm.Shapley(Genfuzzy)
[1] 0.2083 0.4583 0.3333

> # Compute Interaction Index values
> InterVal <- fm.Interaction(Mobfuzzy)


        [,1]      [,2]
[1,]    0       0.4792
[2,]    1       0.2083
[3,]    2       0.4583
[4,]    12      0.3750
[5,]    3       0.3333
```

```
[6,]     13       0.1250
[7,]     23       −0.3750
[8,]     123      0.2500
> # The Interaction Index values is stored in a matrix.
> # The first column stores the indexes of criteria in groups
> # and the second column stores the interaction index values.
```

We can discover the behavior of customers by interpreting the output values of the *Shapley* and *Interaction Index*. It is convenient to associate the computed *Shapley* values with the labels of the input criteria as in Table 9.5.

The preference of the customer is interpreted as follows. Product *quality* is the most important criterion for customers as shown by the highest value of 0.4583. The *price* is considered least as indicated by the lowest value of 0.2083.

Insight into customer behavior can also be understood by interpreting the *Interaction Index*. Here, the *Interaction Index* values are computed for every subset of input criteria. However, it is suggested the values of pair-wise interactions be used for preference analysis as they are easy to interpret and understand. The extracted pairwise *Interaction Index* values for analysis are {1, 2} (0.375), {1, 3} (0.125), and {2, 3} (−0.3750). For the purpose of interpretation, they are presented as an interaction matrix in Table 9.6.

The behavior of customers can be analyzed as follows. There is a significant *complementary* relationship between the `price` and `quality` criteria as indicated by a positive *Interaction Index* value (0.375). This means that the customers will be more interested in the product if it offers cheap price and good quality. On the contrary, the pair `quality` and `service` appear to have a correlating relationship as indicated by a negative *Interaction Index* value (−0.375). Apparently, the preference of the customers does not increase even if such product has good quality and good service. In addition, there is a slight complement of 0.125 between pair `price` and `service`. However, this interaction is insignificant as shown by the low value; therefore, it may not be necessary to include this in the finding.

Table 9.5 Shapley Values

	Price	Quality	Service
Shapley	0.2083	0.4583	0.3333

Table 9.6 Interaction Index Values

	Price	Quality	Service
Price	-	0.375	0.125
Quality		-	−0.375
Service			-

In summary, `Rfmtool` provides a tool for performing operations on the *Choquet Integral* and for computing its related metrics. When it comes to real-life applications for preference discovery, the role of users is important when interpreting the computed *Shapley* values and the *Interaction Index* values into meaningful information. Section 9.4 provides a real case study, which can assist users to better understand the usage of this toolbox.

9.4 Case Study

In this section, we demonstrate the use of `Rfmtool` in an application to discover customer preference in practice. First, we provide the background on the modeling of the behavior of international travelers in the hotel selection process. We then describe the process of online hotel review collections. The experiment design and result analysis are then presented in tables of *Shapley* values and *Interaction Index* values. The last subsection contains a summary of this case study with managerial implications on how to improve the travelers' satisfaction level in their hotel business.

9.4.1 Traveler Preference Study and Hotel Management

The choice of hotel is a high priority for most overseas travelers, and it is an example of a complicated decision-making process (Sohrabi et al., 2012). Hotel managers are interested in the influencing factors in the hotel selection process of travelers (Lockyer, 2005a). Efforts have been made to study hotel criteria, which may influence the choice of travelers. Factors such as *location*, *price*, *facilities*, and *cleanliness* have been proven to have strong relationships to the hotel selection process (Lockyer, 2005b). In further research, hotel features such as *hotel location*, *hotel guest room size*, *staff*, *hotel facilities*, and *breakfast* have also been examined (Stringam et al., 2010). Recently, Merlo et al. (Merlo and Joao, 2011) attempted to identify attributes of low price hotel segments that have more value in improving the satisfaction of more travelers, and found that three features emerged. These included *cleanliness*, *silence in rooms*, and *airconditioning*. Personal factors including *gender*, *purpose of stay*, *nationality*, *culture*, and *private domain of hospitality* were proven to have an influence in expectations about hotel hospitality by Ariffin and Maghzi (Ariffin and Maghzi, 2011).

In addition, the relative importance of each hotel factor in determining a traveler's overall satisfaction levels has also been of interest. Once these criteria are identified and evaluated, the managerial practices and considerations can be used to focus on what is important to travelers, so the quality of service can be improved, and traveler' satisfaction can be increased (Israeli, 2000). For instance, Choi and Chu (Choi and Chu, 2001) found that *Service Quality*, *Room Qualities*, and *Value* were three of the most influential factors in determining the overall impression of travelers toward the selected hotels. Shergill and Sun (Shergill and Sun, 2004) conducted research on *overall facilities*, *room facilities*, and *service* of hotels in *New Zealand*,

to identify factors that are likely to influence the vacation traveler's perceptions. Recently, Tsai et al. (Tsai et al., 2011) investigated the importance of hotel selection criteria between *Mainland Chinese* and foreign individual travelers to *Hong Kong*. Despite significant effort, hotel managers still face significant barriers in gaining insight into the behavior of travelers and their decision making for effective planning.

In this case study, we aim to address these challenges in hotel management by applying the `Rfmtool` for more accurate modeling of traveler's MCDM process in hotel selection. Our case study focuses on modeling of international travelers to *Singapore*, which is one of the major tourist destinations in *South East Asia*.

9.4.2 Data Collection and Experiment Design

The data set used in this case study was collected from *tripadvisor.com*, a well-known travel review web site for traveler's opinion analysis. Each review contains the ratings on popular hotel criteria including *value for money* (*Value*), *hotel location* (*Location*), *quality of sleep* (*Sleep*), *quality of room* (*Room*), *room cleanliness* (*Cleanliness*), and *additional service* (*Service*), as well as an *overall rating*. The ratings are on a scale from 1 (*very unsatisfied*) to 5 (*very satisfied*).

Because the `Rfmtool` is designed for the purpose of preference analysis, the data collection process was done outside of this toolbox. Users may choose to write a script or use a particular data extraction software. In the context of this chapter, we used *Visual Web Ripper* (*www.visualwebripper.com*), a professional data extraction software to collect those ratings from the web site. The software navigated through all listed hotels in *Singapore* and extracted all review ratings, including user ratings of hotel criteria together with the reviewer's demographic data such as *travel types* (*business*, *family*, *couple*) and *continents of origin*. It should be noted, this is an automated process and some reviews did not provide ratings on all six listed hotel features, and a number of data instances are with missing attributes. Those missing data were removed, and therefore, a total of 8561 records were used in this case study.

In tourism research, it has been suggested that traveler behavior and decisions are guided by the deep effects of national culture. People from different continents generally have different backgrounds (Khadaroo and Seetanah, 2008), which makes it convenient to group the regions according to a reviewer's *continent of origin* as a preparatory step for preference analysis. We also noted that the majority of people who traveled to *Singapore* and posted review comments were from *Asia*, *Europe*, *North America*, and *Oceania*, with very few people from *South America* or *Africa*. For this reason, only the first four of these continents have been considered here. Table 9.7 shows the structure of our data sets with respect to *travel types* and *regions*.

In order to demonstrate the use of `Rfmtool` in addressing the challenges in hotel management, we performed the following analysis:

Table 9.7 Hotel Rating Data Collections

Travel Type	Region	Size
Business	Asia	1210 instances
	Europe	581 instances
	North America	407 instances
	Oceania	381 instances
Couple	Asia	1169 instances
	Europe	1389 instances
	North America	320 instances
	Oceania	1188 instances
Family	Asia	951 instances
	Europe	309 instances
	Noth America	131 instances
	Oceania	525 instances
	Total	8561 instances

Preference profile construction: We constructed a detailed hotel preference profile with respect to *travel types* (*business*, *couple*, and *family*) and *regions* (*Asia*, *Europe*, *North America*, and *Oceania*) of travelers to *Singapore*, by analyzing the *Shapley* values.

Interaction behavior analysis: We analyzed the *interaction index* for each travel group, to demonstrate the capability of *Choquet Integral* in assessing the interaction between criteria.

In a research application, users are often required to compare their selected methods against others. For instance, *Choquet* was claimed to be a suitable candidate for effective MCDM modeling; however, it is also necessary to evaluate its performance against other typical aggregation methods. The next section provides instruction about such evaluation procedures in the *R* environment. Notably, the input data to the aggregation functions is assumed to be in the range [0,1]. It is necessary to perform normalization for all the hotel ratings into this range before any other operation.

9.4.3 Model Evaluation

In this section, we evaluate the performance of the *Choquet Integral* (CI) and other algorithms such as *AM* and *WAM*. Three subdata sets are constructed from the hotel rating data collection with respect to *travel types* for evaluation purposes. The subdata sets contain 2579 instances for *business* travelers, 4066 instances for *couple* travelers, and 1916 instances for *family* travelers. They are converted into the range [0,1] and saved in three data files (`Business.txt`, `Couple.txt`, and `Family.txt`).

The evaluation procedure is follows a 10-fold cross validation strategy. More specifically, each of the subdata sets are randomly divided into 10 sets. The algorithms are applied on

9 sets and tested on 1 set, and this process is repeated 10 times. *Mean Absolute Error* (MAE) is used to measure the prediction error of the algorithms, and is reflective of their performance.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)| \tag{9.9}$$

where y_i is the value of the ith input instance to be predicted, and $f(\mathrm{x}_i)$ is the predicted value of y_i.

The evaluation procedure for CI is as given below.

```
# Firstly, we define a function to compute Mean Absolute Error.
mae <- function(obs, pred) mean(abs(obs-pred))

# Then, we define a function containing the evaluation
# routine with 10-fold cross validation.
evalfunc <- function(datafile,kadd)
{
     # datafile is the filename containing the empirical data set.
     # kadd is the value of k-additive.

     # Read the data set from file.
     data <- as.matrix(read.table(datafile));
     size <- dim(as.matrix(data));
     row <- size [1];
     col <- size [2];
     inputdim <- col - 1;

     # Evaluate the Choquet Integral using 10−fold cross validation.
     k <- 10;
     id <- sample(rep(seq_len(k), length.out=nrow(data)));

     # Initialize an array to save the prediction errors.
     maeVal <- array(0,c(1,k));
     for (i in seq_len(k))
     {
          # Generate a training and a testing data set.
          test_matrix <- data [id==i, ];
          train_matrix <- data [id!=i, ];

          # Estimate fuzzy measure from training data.
          estfuzzy <- fm.fitting(train_matrix,kadd);

          # Initialize arrays to save the aggregated values.
          predicVal <- array(0,c(1,nrow(test_matrix)));
          originVal <- array(0,c(1,nrow(test_matrix)));

          count <- 1;
          for (f in seq_len(nrow(test_matrix)))
          {
```

```
            eachrec <- test_matrix[f,];
            # Compute Choquet Integral for each testing input.
            ChoVal <- fm.ChoquetMob(eachrec[1:col-1],estfuzzy);

            predicVal[count] <- ChoVal;
            originVal[count] <- eachrec[col];
            count <- count +1;
        }
        # Compute the Mean Absolute Error for each iteration.
        maeVal[i] <- mae(originVal,predicVal);
    }
    # Return the average of Mean Absolute Error.
    return(mean(maeVal));
}
```

The main evaluation routine called `evalfunc` to evaluate the CI with the empirical data sets is shown below.

```
> # Specify the value for k-additive.
> # Here kadd is set to 6.
> kadd <- 6;
> # Get the data file names.
> busidata <- "Business.txt";
> coupdata <- "Couple.txt";
> famidata <- "Family.txt";
> # Users are suggested to repeated the evaluation process
> # for several time and get the average error as the indicator
> # of the performance for Choquet on each data set.
> # Here, we repeated them 10 times.
> busierror <- array(0,c(1,10));
> couperror <- array(0,c(1,10));
> famierror <- array(0,c(1,10));
> for (i in seq_len(10)) {
  busierror[i] <- evalfunc(busidata,kadd);
  couperror[i] <- evalfunc(coupdata,kadd);
  famierror[i] <- evalfunc(famidata,kadd);
  }
> # Get the average error on each empirical data set.
> mean(busierror)
[1] 0.05762073
> mean(couperror)
[1] 0.05421118
> mean(famierror)
[1] 0.05861229
```

The evaluation of AM and WAM are performed similarly. Their routine code is provided in the files `AMEvaluation.r` and `WAMEvaluation.r` in the `Rfmtool` package. The prediction errors for CI, AM, and WAM are shown in Table 9.8. In particular, for the CI evaluation results with different `kadd` values are recorded.

Table 9.8 MAE Values of the Evaluated Algorithms

		Data Sets		
	Algorithms	Business	Couple	Family
CI	AM	0.0757	0.0689	0.0706
	WAM	0.0701	0.0633	0.0666
	Kadd = 1	0.0701	0.0633	0.0666
	Kadd = 2	0.0662	0.0613	0.0649
	Kadd = 3	0.0615	0.0556	0.0612
	Kadd = 4	0.0622	0.0561	0.0620
	Kadd = 5	0.0578	0.0543	0.0600
	Kadd = 6	*0.0576*	*0.0542*	*0.0586*

Table 9.9 Sub-Data Set for Hotel Selection Behavior Analysis

Data Set	File Name	Size
D_1	Business-Asia.txt	1210 instances
D_2	Business-Europe.txt	581 instances
D_3	Business-NorthAmeica.txt	407 instances
D_4	Business-Oceania.txt	381 instances
D_5	Couple-Asia.txt	1169 instances
D_6	Couple-Europe.txt	1389 instances
D_7	Couple-NothAmeric.txt	320 instances
D_8	Couple-Oceania.txt	1188 instances
D_9	Family-Asia.txt	951 instances
D_{10}	Family-Europe.txt	309 instances
D_{11}	Family-NothAmerica.txt	131 instances
D_{12}	Family-Oceania.txt	525 instances

The performance of CI is similar to WAM when `kadd = 1` because the fuzzy measure is restricted to individual criteria, which is equivalent to the WAM. When `kadd = 6`, CI performs best with a significantly low prediction error. This is the result of fuzzy measures being unrestricted, where, all possible combinations of input criteria are considered. When the value of `kadd` is increased, its performance tends to increase with lower prediction error. However, this is not always true because of various numerical issues, particularly when data does not fully cover the domain of the function. In general, CI appears to outperform other algorithms as indicated by low MAE values on all three data sets when `kadd > 1`.

9.4.4 Result Analysis

In this section, we construct and analyze the hotel selection behavior of travelers with respect to regions and traveling purposes. The collected data set is divided into sub-data sets according to the *travel types* and the *regions* as shown in Table 9.9. Note that these sub-data sets are

included in the `Rfmtool` package, and the rating values in the data sets are normalized into the range [0,1]. The procedure for preference analysis and interaction behavior is performed in the subsequent sections 9.4.4.1 and 9.4.4.2 respectively.

9.4.4.1 Preference Profile Construction

The preference of travelers in each group can be discovered through the *Shapley* value as presented in Section 9.3.3. For instance, the *Shapley* value reflecting the preference of business travelers from *Asia* can be computed from data set D_1 in the file `Business-Asia.txt` following the below procedure.

```
> # Load the data set.
> data <- as.matrix(read.table("Business-Asia.txt"))
> # The data matrix has 1210 records.

> # Estimate the fuzzy measure.
> estfuzzy <- fm.fitting(data)

 [1]  0.0   0.0   0.0   0.0   0.0   0.0  0.0   0.0  0.0   1.0  0.0   0.0
[13]  0.0   0.0   0.0   0.0   0.0   0.0  0.0   0.0  0.0   0.0  0.0   1.0
[25]  0.0   0.0   0.0   0.0   0.0   0.0  0.5  -1.5  0.0   0.0  0.0   0.0
[37]  0.0   0.0   0.0   1.0   0.0   0.0  0.0   0.0  1.0  -1.0  0.0  -1.0
[49]  0.0   1.0   0.0   0.0   1.0  -1.0  0.0  -2.0  1.0  -2.0  0.0   0.0
[61] -2.0   2.0  -0.5   2.5

> # Note that the estimated fuzzy measure in estfuzzy is in
> # Mobius representation containing 2^6 = 64 values.

> # Tranform Mobius fuzzy measure to general fuzzy measure.
> genfuzzy <- fm.Zeta(estfuzzy)

 [1]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  1.0
[13]  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
[25]  0.0  1.0  0.0  1.0  0.0  1.0  0.5  1.0  0.0  0.0  0.0  0.0
[37]  0.0  0.0  0.0  1.0  0.0  1.0  0.0  1.0  1.0  1.0  1.0  1.0
[49]  0.0  1.0  0.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
[61]  1.0  1.0  1.0  1.0

> # Compute Shapley values
> ShapleyVal <- fm.Shapley(genfuzzy)
[1] 0.2500 0.0417 0.1083 0.2583 0.1417 0.2000
> # ShapleyVal contains 6 values corresponding 6 hotel features.
```

Similarly, the *Shapley* values for all traveler groups can be computed. Their outputs are extracted and presented in Table 9.10.

From the *Shapley* value in Table 9.10, the hotel preference profile of people traveling to *Singapore* can be constructed as follows:

Business Group: Travelers from *Asia* care more about features such as *value for money*, *room quality*, and *services*, with the *room quality* as the most important feature. For *European* and *Oceania* travelers, their preferred features were *room quality* and *service* as indicated with high *Shapley* values. *Service* is highly valued by *Oceania* people (0.367). On the other hand,

Table 9.10 Shapley Values for Different Groups of Travelers

		Hotel Features					
Travel Type	**Region**	**Value**	**Location**	**Sleep**	**Room**	**Cleanliness**	**Service**
Business	Asia	0.250	0.042	0.108	0.258	0.142	0.200
	Europe	0.192	0.050	0.125	**0.300**	0.067	0.267
	North America	0.267	0.150	0.233	0.100	0.067	0.183
	Oceania	0.017	0.067	0.117	**0.317**	0.117	**0.367**
Couple	Asia	0.265	0.117	0.158	0.153	0.153	0.154
	Europe	0.225	0.075	0.125	0.158	0.142	0.275
	North America	0.150	0.067	0.114	**0.303**	0.150	0.217
	Oceania	0.100	0.100	0.183	0.217	0.133	0.267
Family	Asia	0.217	0.133	0.067	0.167	0.167	0.250
	Europe	0.142	0.125	0.167	0.233	0.117	0.217
	North America	0.092	0.092	0.042	**0.317**	0.117	**0.342**
	Oceania	0.100	0.203	0.119	0.092	0.186	0.300

business people from *North American* countries consider the *value for money* and *sleep quality* features of the *Singapore* hotels as more important. *Location* feature appears to be less important, especially for *Asian* travelers. *Cleanliness* is also considered as unimportant for business travelers from *Europe* and *North America*.

Couple Group: It is interesting to see that *Asian* and *European* travelers pay high attention to the *value for money* feature when they travel with their partners. However, *European* people are also concerned more about the *service* feature. For *North American* and *Oceania* couples, their preferred features were the *room quality* and the *service quality*, with the *room quality* being the most important for *North American* people (0.303). In general, most hotel features have certain levels of influence on the overall rating of people traveling as a *couple* as shown with relatively significant *Shapley* values (above 0.1). The *location* feature is of less concern for *couples* from *Europe* and *North America*.

Family Group: The importance of the *value for money* feature remains significant for *Asian* travelers in this group, while *European* travelers care more about *room quality*. However, the preference for *room quality* is still high for *North American* travelers (0.317). Most interestingly, *service quality* is the feature with a heavy effect on the decision-making process for people from all regions when traveling with *family*, particularly for *North American* and *Oceania* travelers. The *sleep quality* appears to be unimportant for *Asian* and *North American* travelers who are traveling with their family.

9.4.4.2 Interaction Behavior Analysis

Another advantage of the *Choquet Integral* is the ability to provide insight into the interaction between criteria. We compute the *Interaction Index* for each traveler group following the below procedure.

```
> # Load the data set.
> data <- as.matrix(read.table("Business-Asia.txt"))
> # Estimate the fuzzy measure.
> estfuzzy <- fm.fitting(data)

 [1]  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0   0.0
[13]  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   1.0
[25]  0.0   0.0   0.0   0.0   0.0   0.0   0.5  -1.5   0.0   0.0   0.0   0.0
[37]  0.0   0.0   0.0   1.0   0.0   0.0   0.0   0.0   1.0  -1.0   0.0  -1.0
[49]  0.0   1.0   0.0   0.0   1.0  -1.0   0.0  -2.0   1.0  -2.0   0.0   0.0
[61] -2.0   2.0  -0.5   2.5

> # Compute Interaction Index from Mobius fuzzy measure.
> InteracVal <- fm.Interaction(estfuzzy)

      [,1]    [,2]
[1,]  0       0.4905
[2,]  1       0.2500
[3,]  2       0.0417
[4,]  12      0.0417
[5,]  3       0.1083
[6,]  13      -0.1250
[7,]  23      0.0833
[8,]  123     0.1250
[9,]  4       0.2583
. . . . . .
> # The output interaction index matrix InteracVal has 64 rows
> # Only some first records are shown above.

> # Extract the pair-wise interaction matrix.
> InterMatrix <- array(,c(6,6));
> for (i in seq_len(nrow(InteracVal))){
+   PairIndex <- InteracVal[i,1];
+   if(PairIndex > 10 && PairIndex <100){
+       ColInd <- PairIndex %% 10;
+       RowInd <- (PairIndex - ColInd)/10;
+       InterMatrix [RowInd,ColInd] = InteracVal[i,2];
+ }}
> InterMatrix

      [,1]   [,2]     [,3]     [,4]     [,5]     [,6]
[1,]  NA     0.0417  -0.1250   0.3750  -0.0417  -0.2500
[2,]  NA     NA       0.0833  -0.0833   0.0000  -0.0417
[3,]  NA     NA       NA      -0.0833   0.0000   0.1250
[4,]  NA     NA       NA       NA      -0.1667  -0.0417
[5,]  NA     NA       NA       NA       NA       0.2083
[6,]  NA     NA       NA       NA       NA       NA
```

The *Interaction Index* values for other travel groups are computed similarly. The pairwise *Interaction Index* values are extracted and presented in Tables 9.11–9.13 for interaction analysis.

Table 9.11 Interaction Index Values for Business Travelers

Region	Interaction Index						
Asia		Value	Location	Sleep	Room	Cleanliness	Service
	Value	–	0.042	−0.125	**0.375**	−0.042	*−0.250*
	Location		–	−0.083	−0.083	0.000	−0.042
	Sleep			–	−0.083	0.000	0.125
	Room				–	−0.167	−0.042
	Cleanliness					–	0.208
	Service						–
Europe	Value	–	0.058	−0.025	**0.350**	0.017	*−0.400*
	Location		–	0.017	−0.108	0.058	−0.025
	Sleep			–	*−0.275*	0.058	0.225
	Room				–	−0.150	0.183
	Cleanliness					–	0.017
	Service						–
North	Value	–	0.233	−0.100	−0.017	0.067	−0.183
America	Location		–	*−0.350*	0.067	−0.017	−0.067
	Sleep			–	−0.067	0.150	0.233
	Room				–	−0.100	−0.017
	Cleanliness					–	−0.100
	Service						–
Oceania	Value	–	−0.050	0.033	−0.050	0.033	0.033
	Location		–	0.117	0.033	0.117	−0.217
	Sleep			–	−0.217	0.200	−0.133
	Room				–	−0.217	**0.450**
	Cleanliness					–	−0.133
	Service						–

From the *Interaction Index* values of hotel criteria in the tables, we can see, there are some significant interactions between different criteria in the selection process of travelers. The interactions are also quite different for different travel groups. This indicates that people from different regions, who travel for different purposes have different decision-making processes. We make the following observations:

Business Traveler: The *value for money* criterion appears to have a significant complementary effect with *room quality*, as shown by a high positive index. But, this is redundant with *service* criterion as indicated by a high negative index, for business travelers from *Asia* and *Europe*.

The positive index values indicate that the preference of these travelers for a hotel will increase significantly only if it can satisfy all of those criteria, whereas the negative index values inform us that no improvement on a traveler's preference is made even if the hotel can satisfy both of the criteria. Particularly for business travelers from *Europe*, slight correlation is found between the *sleep quality* and the *room quality* criteria. For people from *North America*, there is a

Table 9.12 Interaction Index Values for Couple Travelers

Region		Interaction Index					
Asia		Value	Location	Sleep	Room	Cleanliness	Service
	Value	–	0.011	0.074	0.046	0.046	−0.176
	Location		–	−0.072	0.067	−0.100	0.094
	Sleep			–	0.004	0.004	−0.010
	Room				–	−0.079	−0.037
	Cleanliness					–	0.129
	Service						–
Europe	Value	–	−0.025	0.017	−0.150	0.142	0.017
	Location		–	0.017	0.100	−0.108	0.017
	Sleep Room			–	−0.025	0.017	−0.025
	Room				–	0.017	0.058
	Cleanliness					–	−0.067
	Service						–
North	Value	–	−0.092	−0.022	−0.064	0.117	0.061
America	Location		–	0.075	0.006	0.019	−0.008
	Sleep			–	−0.036	0.144	−0.161
	Room				–	*−0.314*	0.075
	Cleanliness					–	0.033
	Service						–
Oceania	Value	–	0.067	0.067	−0.017	−0.183	0.067
	Location		–	−0.100	−0.017	−0.183	**0.233**
	Sleep			–	0.150	0.150	*−0.267*
	Room				–	0.067	−0.183
	Cleanliness					–	0.150
	Service						–

strong negative interaction between the pair of criteria *location* and *sleep quality*. Business travelers from *Oceania* have a high preference for a hotel if it offers good *room* and *service quality*.

Couple Traveler: Most hotel criteria pairs for this travel type show little interaction, as indicated by the low *Interaction Index* values. Only a few significant interactions were found. For instance, the *room quality* shows a slight redundancy with the *cleanliness* criterion for *North American* couples. Couples from *Oceania* have a high preference for a hotel which offers good quality in both *location* and *service*, as this pair has a considerable positive interaction. However, the preference of this group does not increase if the hotel has good *service* together with *sleep quality*, as indicated by a negative interaction.

Family Traveler: Service criterion appears to play a vital role in the decision-making process of people traveling with their families. More specifically, it has a strong complementary interaction with *cleanliness* (for *Asian* families), *room quality* (for *North American* families), and *location* (for *Oceania* families). On the contrary, *service* also has a strong negative interaction with *value for money* and the *cleanliness* for families from *Asian* and *North*

Table 9.13 Interaction Index Values for Family Travelers

Region		Interaction Index					
Asia		Value	Location	Sleep	Room	Cleanliness	Service
	Value	–	0.083	0.083	0.167	0.000	*−0.333*
	Location		–	−0.083	0.000	−0.167	0.167
	Sleep			–	0.000	0.000	0.000
	Room				–	−0.083	−0.083
	Cleanliness					–	**0.250**
	Service						–
Europe	Value	–	−0.033	0.050	0.175	−0.158	−0.033
	Location		–	0.092	*−0.283*	0.050	0.175
	Sleep			–	−0.033	0.133	−0.242
	Room				–	0.008	0.133
	Cleanliness					–	−0.033
	Service						–
North America	Value	–	0.083	0.000	*−0.292*	0.125	0.083
	Location		–	0.000	0.042	0.125	*−0.250*
	Sleep			–	−0.042	0.042	0.000
	Room				–	−0.083	**0.375**
	Cleanliness					–	−0.208
	Service						–
Oceania	Value	–	0.033	−0.133	−0.064	0.047	0.117
	Location		–	0.006	−0.064	−0.231	**0.256**
	Sleep			–	−0.064	0.103	0.089
	Room				–	0.200	−0.008
	Cleanliness					–	−0.119
	Service						–

American travelers respectively. The *room quality* is found to have a slight correlation with the *location* for *European* travelers, and *value for money* for *North American* travelers.

9.4.5 Discussion

Analysis has demonstrated the use of the `Rfmtool` package in studying the preferences of travelers in the hotel selection process. By using fuzzy measures, the *Choquet Integral* is able to provide hotel managers with a better understanding of traveler profiles and preferences for hotel features. The detailed analysis of preference profiles using the *Shapley* value in Section 9.4.4 has highlighted important hotel criteria for travelers with respect to *travel types* and *regions*, which hotel managers should give high priority to improving. Furthermore, the advantage of the fuzzy decision support technique using the *Choquet Integral* is the ability for assessing the interaction between criteria as presented in Section 9.4.4. The positive interactions suggest that hotel quality must be met for both of these criteria at the same time to satisfy the expectation of travelers in this group. However, negative interactions suggest it is unnecessary to improve both

criteria as they are redundant. Such information is useful for hotel managers to decide what criteria should be improved to achieve the best outcome with minimal effort. The marketing strategy could also be improved with a profile construction for different groups of travelers, thus enabling consumer behaviors to be better understood.

The `Rfmtool` package provides a useful tool to support organizations, companies, and business managers in modeling the decision-making process of customers. The output result is easy to interpret and understand, and therefore, can be used directly by nontechnical users. The `Rfmtool` can be applied in many areas such as the retail industry for customer preferences in products, the education industry for student preferences on course contents, in politics for understanding the demands of people in society, etc. The case study presented for Hotels in *Singapore* demonstrates its effectiveness in the tourism industry. It should be noted that the input data of this toolbox is in the form of rating of each criterion as the data was collected from the *TripAdvisor* website. In case the information is not already available for online review, the rating can be obtained by using other existing web data mining techniques for online review analysis such as introduced in Liu (2010, 2011). However, a discussion of these techniques is beyond the scope of this chapter. On the other hand, one possible way to make use of this toolbox in practice is to design surveys for collecting consumer ratings on products; the analysis can then be performed in a similar way as presented in this case study.

The evaluation result in Section 9.4.3 suggests the use of the fuzzy measure (`kadd = n`) that can model the decision-making process of customers closest to a real-life situation. This was proved by the best performance of the lowest MAE on all three tested data sets. However, when using the `Rfmtool`, it is important to note that the setting of a larger value for `kadd` can make the fuzzy measure more flexible, but the model will also become more complex. The use of large `kadd` should be accompanied by large training data sets, otherwise, the use of `kadd < n` is suggested.

9.5 Conclusions

The understanding of customers' preferences has always been of interest for researchers in supporting the strategic planning and decision making of business managers. Efforts have been made to study the preferences by proposing techniques and modeling the multicriteria decision-making process of managers. However, a traditional technique, such as *weighted averaging*, is unable to effectively perform this task, because it is unable to consider multiple criteria simultaneously. There remains a strong demand for a technique that allows for the exploration of the MCDM process.

This chapter has addressed such a demand by introducing a new aggregation function technique, the *Choquet Integral*, for effective modeling of the MCDM process. By using the

Shapley and *Interaction Index* values, the insight into customer preferences as well as the interaction among criteria, has been explored.

The `Rfmtool` toolbox is also introduced for customer preference analysis. The operations of this toolbox can be performed easily in the *R* environment, which is freely available for the research community. A case study of modeling the international traveler's behavior in the hotel selection process has also demonstrated the use of the `Rfmtool` in practice. The *Choquet Integral* and `Rfmtool` toolbox have significant potential in supporting researchers and industry practitioners in dealing with the MCDM process and developing strategic plans. The findings are also helpful for tourism managers who are seeking to explore the potential of the growing tourism market.

References

Ariffin, A.A.M., Maghzi, A., 2011. A preliminary study on customer expectations of hotel hospitality: Influences of personal and hotel factors. Int. J. Hosp. Manag. 31 (1), 191–198.

Beliakov, G., 2007. fmtools package, version 1.0. http://www.deakin.edu.au/gleb/aotool.html.

Beliakov, G., Pradera, A., Calvo, T., 2007. Aggregation Functions: A Guide for Practitioners. Springer, Heidelberg, Berlin, New York.

Berkelaar, M., Buttrey, S., 2011. Interface to lp solve v. 5.5 to solve linear/integer programs. Version: 5.6.6. http://cran.rproject.org/web/packages/lpSolve/index.html.

Chateauneuf, A., Jaffray, J.Y., 1989. Some characterization of lower probabilities and other monotone capacities through the use of mobius inversion. Math. Soc. Sci. 17 (3), 263–283.

Choi, T.Y., Chu, R., 2001. Determinants of hotel guests's satisfaction and repeat patronage in the hong kong hotel industry. Int. J. Hosp. Manag. 20 (3), 277–297.

Grabisch, M., 1997. k-Order additive discrete fuzzy measures and their representation. Fuzzy Set Syst. 92 (2), 167–189.

Grabisch, M., Labreuche, C., 2007. Derivative of functions over lattices as a basis for the notion of interaction between attributes. Ann. Math. Artif. Intell. 49, 151–170.

Grabisch, M., Lange, F., 2007. Games on lattices, multichoice games and the shapley value: a new approach. Math. Methods Operations Res. 65, 153–167.

Grabisch, M., Roubens, M., 2000. Application of the Choquet Integral in Multicriteria Decision Making. Physica Verlag, Wurzburg.

Granot, E., Greene, H., Brashear, T.G., 2010. Female consumers: decision-making in brand-driven retail. J. Bus. Res. 63 (8), 801–808.

Israeli, A.A., 2000. Exploring the importance of hotel features among guests using a multi-attribute scaling approach. Int. J. Tourism Hospitality Res. 11 (2), 141–158.

Khadaroo, J., Seetanah, B., 2008. The role of transport infrastructure in international tourism development: a gravity model approach. Tour. Manage. 29, 831–840.

Liu, B., 2010. Sentiment analysis and subjectivity. Handbook of Natural Language Processing. pages 1–38.

Liu, B., 2011. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Springer, Berlin, Heidelberg chapter 11.

Lockyer, T., 2005a. The perceived importance of price as one hotel selection dimension. Tour. Manage. 26 (4), 529–537.

Lockyer, T., 2005b. Understanding the dynamics of the hotel accommodation purchase decision. Int. J. Contemp. Hosp. Manag. 17 (6), 481–492.

Merlo, E.M., Joao, I.D.S., 2011. Consumers attribute analysis of economic hotels: an exploratory study. J. Bus. 5 (21), 8410–8416.
Shergill, G.S., Sun, W., 2004. Tourists' perceptions towards hotel services in New Zealand. Int. J. Hosp. Tourism Admin. 5 (4), 1–29.
Sohrabi, B., Raeesi, I., Tahmasebipur, K., Fazli, S., 2012. An exploratory analysis of hotel selection factors: a comprehensive survey of tehran hotels. Int. J. Hosp. Manag. 31 (1), 96–106.
Stringam, B.B., Gerdes, J., Vanleeuwen, D.M., 2010. Assessing the importance and relationships of ratings on user-generated traveler reviews. J. Qual. Assur. Hosp. Tourism. 11 (2), 73–92.
Tsai, H., Yeung, S., Yim, P.H.L., 2011. Hotel selection criteria used by mainland Chinese and foreign individual travelers to Hong Kong. Int. J. Hosp. Tourism Admin. 12 (3), 252–267.
Yang, J.B., Wang, Y.M., Xu, D.L., Chin, K.S., Chatton, L., 2012. Expert systems with applications belief rule-based methodology for mapping consumer preferences and setting product targets. Expert Syst. Appl. 39 (5), 4749–4759.

CHAPTER 10

A Real-Time Property Value Index Based on Web Data

M.J. Bárcena*, P. Menéndez†, M.B. Palacios‡, F. Tusell§

*University of the Basque Country (UPV/EHU), Spain
†University of Queensland, Australia
‡Public University of Navarra (UPNA), Spain
§University of the Basque Country (UPV/EHU), Spain

10.1 Introduction

Since the accession of Spain to the EEC in 1986, and, in particular, during the first years of the present century, housing prices experienced substantial increases. This phenomenon has been observed in a large part of the western world, but in Spain has been exacerbated by a number of circumstances: monetary stability, with low or even negative real interest rates, easy borrowing, high economic growth, and fiscal allowances for home buyers. Since the first years of this century, it became commonplace to refer to the *real estate bubble*.

The financial crisis started in 2007 brought an abrupt end to this state of affairs. Credit tightened and house sales slowed down putting substantial downward pressure in the residential property prices. There has been considerable interest since then to quantify the surge in prices prior to 2007 and the subsequent price drop. Widely different figures have been given, which highlights the difficult underlying problem of measurement. Our work in this chapter offers yet another approach that emphasizes the usefulness of public data sources, data mining techniques, and software tools such as R to obtain reliable and potentially real time information on housing prices.

10.2 Housing Prices and Indices

Houses are traded at relatively infrequent times and can hardly be standardized: two equally built and furnished houses may command widely different prices in the market on account of their different location or even orientation. Clearly, the computation of an index such as Laspeyres'

Data Mining Applications with R

$$I_0(t) = \frac{\sum_{i=1}^{I} p_{it} q_{i0}}{\sum_{i=1}^{I} p_{i0} q_{i0}} \tag{10.1}$$

where p_{it} and q_{it} are, respectively, prices and quantities of good i at time t, is unfeasible, because no given set of houses are traded at regular intervals, and no exact replicates are available.

Both problems—lack of homogeneity in houses and irregular observation times—have been addressed in a variety of ways, which include the estimation of hedonic models (Baranzini et al., 2008; Rosen, 1974), construction of "cells" of relatively similar houses (Gila García and Novás Filgueira, 2012), and repeated sales methods (Rodríguez López, 2007).

Hedonic models attempt to measure the contribution of individual attributes to the total value of the house through what is in essence a regression model. Each house is regarded as a "basket" of characteristics like size, quality of the construction, age, and quality attributes of the surroundings (e.g., clean air or noise). Location is typically included in the form of proxies which measure distance to transportation networks, municipal services, recreational areas, and so on.

Because these proxies are not always observable, and rarely capture fully all factors affecting the desirability of a given location, spatial effects are likely to remain. Such effects can be introduced in different ways: (a) through the error structure (Dubin, 1988); (b) as additional effects in the regression, typically in the form of smooth surfaces over space, nonparametrically estimated (Hastie and Tibshirani, 1991); or (c) by allowing the coefficients of the hedonic model to change in space (geographically weighted regression (GWR) (Fotheringham et al., 2002; Kestens et al., 2005) used later.

Hedonic models have also been used to measure the effect of time on selling prices of homes, which is of particular interest here. Time is not necessarily viewed as a cause, but as a proxy of a variety of factors which change over time and cannot be explicitly accounted for. The specification typically takes log price as the response variable in a regression model; the estimated coefficients of the dummy time variables are used to measure the cumulative percentage of change in constant quality house prices up to and including the associated time period.

Besides lack of homogeneity and infrequent trading, an additional problem exists with the housing market: opacity. There are incentives to misrepresent the price at which transactions take place, mainly for fiscal reasons. Therefore, it is hard, if at all possible, to obtain such information.

10.3 A Data Mining Approach

Our goal is to develop a methodology to obtain reliable estimates of the price level of housing in a given area. The approach followed in our work has been to use publicly available information, such as property sales offers in web sites. Although the offered price is rarely the price at which

transactions are finalized, for the purpose of index computations this is a minor problem, as long as the ratio of offered prices to transaction prices is reasonably stable in time. Offered prices appear to be good proxies and have been used in a number of studies, including Kryvobokov and Wilhelmsson (2007), Henneberry (1998), and Pace et al. (2000).

As a case study, we have used data from www.idealista.com, one of the main web sites in Spain carrying real estate offers. Data consist of house offers in the city of Bilbao, a large industrial town located in the north of Spain, at about 43.25N, 2.93W. Observations extend from 2004-11-17 to 2012-05-09, although they are sparse prior to 2007. Data consist of a description of the property offered for sale, including selling price, location (often down to the street number, in other cases only street or district), square footage, floor, age of the building, number of bedrooms and bathrooms, parking space if available, availability of central or individual hot water and heating, elevator, etc.

10.3.1 Data Capture

As it often happens, data capture, preparation, and cleaning represent a large proportion of the effort necessary to analyze information such as this. In our case, this was compounded by the fact that offers are user-inputed, and in a relatively free format with respect, for instance, to addresses. We cannot give here a full account of the raw data preprocessing step, but a few details will convey the flavor of it, and display the power of R for this particular task.

We start from two files, `Bilbao.csv` (see http://www.rdatamining.com/books/dmar/) containing offers, and `BilbaoUTM.csv` (see http://www.rdatamining.com/books/dmar/), derived from a different source, containing address location information, described below. These two files are read with:

```
> Bilbao     <- read.csv(file="Bilbao.csv")
> BilbaoUTM  <- read.csv(file="BilbaoUTM.csv")
```

The dataframe `Bilbao` contains 9202 observations, although precise addresses are available on a far smaller number of observations. Dates are given as strings in yyyy-mm-dd format, which is turned to R's `Date` format:

```
> Bilbao$Date <- as.Date(Bilbao$Date)
```

Variables in dataframe `Bilbao` include, among others, those whose names and meanings are listed in Table 10.1.

A total of 11,965 individual addresses are listed in turn in dataframe `BilbaoUTM` with coordinates of the entrance of the building, far more precise than needed for our purposes. Data have been obtained from the Web site of the city of Bilbao. Variables included are listed in Table 10.2.

Table 10.1 Contents of Dataframe `Bilbao`

Variable	Type	Description
Date	Date	Date offer appears in Web
Price	Numeric	Price of property (in €)
Condition	Factor	Conservation state: "Newly built," "Second hand," "Needs reform"
Age	Factor	Age in six categories, from "<5 years" to "+30 years" and "unknown"
HouseType	Factor	Dwelling type, e.g., "flat," "detached house," etc.
Storey	Numeric	Levels, for multistory buildings
Orientation	Factor	Orientation
Community	Numeric	Cost of community services
M2build	Numeric	House's built surface (in square meters)
Bedrooms	Numeric	Number of bedrooms
Bathrooms	Numeric	Number of bathrooms
Elevator	Boolean	Is there an elevator?
Heating	Factor	Heating ("central," "individual," "none")
ParkingPlaces	Numeric	Parking places included in price
CP	Factor	Postal code
District	Factor	District (12 categories)
Type	Factor	Type of address ("Avenue," "Street," ...)
Street	Factor	Name of street, square, avenue, ...
Num	Factor	Number within street, avenue, ...
Origin	Factor	Who offers? ("owner," "agent")

Table 10.2 Contents of Dataframe `BilbaoUTM`

Variable	Type	Description
Type	Factor	Type of address ("Street", "Avenue"...)
Direccion	Factor	Name of street, avenue...
Num	Factor	Number within street, avenue...
Distrito	Factor	District with eight categories
CP	Factor	Postal code (15 categories)
UTMX	Numeric	UTM "easting" within zone 30N in meters
UTMY	Numeric	UTM "northing" within zone 30N in meters

Files such as `Bilbao.csv` (see http://www.rdatamining.com/books/dmar/) containing up-to-date information are made available to us quarterly. Because our goal is to process real estate information in near real time, we capture new information from the Web site of our provider on a daily basis. R also provides tools which make this an easy task. Each new offer, identified by `URL`, is downloaded to a `File` by:

```
> download.file(URL, File, quiet=0, method="wget")
```

Then, `File` is processed by parsing the HTML source using the `XML` package (Lang, 2012; see also Goldberg, 2008) for a general description of XML processing:

```
> doc <- htmlTreeParse(file=File)$children$html[["body"]]
```

The resulting `doc` object is traversed and relevant pieces of information picked up and stored in a row of a dataframe. Although writing the required code is tedious and requires "reverse engineering" the HTML source of the offers, once it is set up it works smoothly. There is every now and then the odd unavoidable problem, given the free format of some fields. This process is embedded in a loop and applied to each new offer.

10.3.2 Geocoding

After reading the data, the first task is to geocode the observations, whenever a complete address for a property in sale is given. This is often the case for houses sold directly by their owners, whereas ads placed by real estate agents usually give only approximate indications of the location, to prevent their being bypassed by prospective buyers.

The geocoding is complicated by the fact that two official languages coexist in Bilbao (Basque and Spanish), so streets may have dual names, not even close in Levenshtein distance (Gusfield, 1997) or any similar metric among character strings. For instance, "Pintores Arrúe" and "Arrue Margolarien" designate the same street in Spanish and Basque. There are streets, avenues, squares, etc. sharing the same name, so the "type of address" in variable `Type`, listed in both Tables 10.1 and 10.2, is vital to decode the correct address. To further complicate things, several writings may coexist, even in a single language: "Zumalacarregui," "Zumalacárregui" (with accent), "Zumalakarregui," "Zumalakarregi" are all accepted names of an avenue.

There are powerful facilities in R to perform geocoding using the API provided by Google Maps; package `ggmap`, Kahle and Wickham (2012), for instance, provides an easy-to-use function `geocode` that does a very respectable job. It was soon clear, however, that the geocoding provided by Google Maps was insufficient for our purposes, and we had trouble making sense of the information in the raw state in which it was made available to us. In the end, we made use of the power of R to reduce as much as possible the diversity of written addresses and matched them to the geographical information read in `BilbaoUTM`.

The process involves trimming useless white space at beginning and end of each address, converting to upper case and getting rid of the accents,

```
> Bilbao$Street <- gsub("(^+)|( +$)", "",Bilbao$Street)
> Bilbao$Street <- toupper(Bilbao$Street)
> Bilbao$Street <- chartr("ÁÉÍÓÚ", "AEIUO",Bilbao$Street)
```

Next, normalized complete addresses were formed by pasting the address type (street, avenue, ..., in `Type`), the address itself, the street number, and the postal code, all separated by dashes. This normalization is performed for addresses both in `Bilbao` and `BilbaoUTM` prior to matching.

```
> Addr2Search <- with(Bilbao,paste(Type,Street,Num,CP,sep="-"))
> Searchable  <- with(BilbaoUTM,paste(Type,Direccion,Num,CP,sep="-"))
```

This is what one of those normalized addresses from `Bilbao` looks like:

```
> Key <- Addr2Search[611]
> Key

[1] "AVENIDA-ZUMALAKARREGI-46-48006"
```

As it happens, the unique name for that address in `BilbaoUTM` is `AVENIDA-ZUMALACARREGUI-46-48007`, so a direct attempt to match fails:

```
> match(Key,Searchable)
[1] NA
```

We need to map all identified variants to that unique name: regular expression matching and substitution is used (Friedl, 2006). For this particular case,

```
> Key <- gsub("ZUMALA[C|K]AR(R*)EG(U*)I",
            "ZUMALACARREGUI", Key, perl=TRUE)
> Key

[1] "AVENIDA-ZUMALACARREGUI-46-48006"
```

after which:

```
> i <- match(Key,Searchable)
> BilbaoUTM[i,c("Type","Direccion","Num","CP","UTMX","UTMY")]

         Type       Direccion  Num     CP     UTMX     UTMY
11754 AVENIDA  ZUMALACARREGUI   46  48006  507191.4  4789543
```

Over 300 like substitutions are used to map input variants in `Addr2Search` to accepted addresses as in `Searchable`, built from `BilbaoUTM`. After this is done, matching is easy and the dataframe `Bilbao` can be completed with UTM coordinates in `BilbaoUTM`:

```
> found <- match(Addr2Search, Searchable)
> Bilbao <- cbind(Bilbao, BilbaoUTM[found, c("UTMX","UTMY")])
```

Location is the single most important factor influencing the value of real estate. Fortunately, R provides tools that make it easy to access cartography, draw maps, and overlay polygons or points to discover patterns for easy interpretation. In the sequel, use is made at various points of packages `maptools` (Lewin-Koh et al., 2012), `rgdal` (Keitt et al., 2012), `RgoogleMaps` (Loecher and Berlin, School of Economics and Law, 2012), and their respective dependencies notably package `sp` described in Pebesma and Bivand (2005); see also Bivand et al. (2008).

Figure 10.1 shows a polygon outline with a map of Bilbao[1] as background, for reference. Figure 10.1 was constructed by first reading a.`shp` file, in ESRI shapefile format, giving the outline of the city of Bilbao and several subdivisions. The information about the projection used is added at the time of reading.

Figure 10.1
Outline of the city of Bilbao.

[1] From Google Maps, `http://maps.google.com`.

```
> require(maptools)
> Bilbao.utm <- readShapePoly("BilbaoDistricts.shp",
    proj4string=CRS("+proj=utm +zone=30 +ellps=intl"))
```

As we are going to overlay this on a raster background which uses lat/long coordinates and the WGS84 datum, we need to reproject. Function `spTransform` in package `rgdal` makes this easy:

```
> require(rgdal)
> Bilbao.wgs84 <- spTransform(Bilbao.utm,
                CRS("+proj=longlat +datum=WGS84 +ellps=WGS84"))
```

We next transform to PolySet format, as this will facilitate the overlay.

```
> Bilbao.PS <- SpatialPolygons2PolySet(Bilbao.wgs84)
```

Next, we obtain a raster background from Google Maps using the package `RgoogleMaps`. In `lat, long` we have to pass a bounding box which covers the area of interest:

```
> require(RgoogleMaps)
> lat <- c(43.222,43.2822)
> lon <- c(-2.880,-2.985)
> bb <- qbbox(lat, lon, margin=list(m=c(1,1,1,1), TYPE="perc"))
> Bilbo.raster <- GetMap.bbox(bb$lonR,bb$latR,maptype="mobile",
                              destfile="Bilbao.png")
```

Finally, the overlay is produced by

```
> PlotPolysOnStaticMap(Bilbo.raster,Bilbao.PS,lwd=1.5,
                       col=NULL,add=FALSE)
```

A similar technique can be used to overlay points, so individual houses (perhaps with abnormal prices) can be placed on the map to investigate patterns. In what follows, we will drop the background to avoid distraction and use only the outline in Figure 10.2, where we have also placed the names of the districts used later.

10.3.3 Price Evolution

As a first approximation, we may think of plotting the evolution in time of the price per square meter. A boxplot sequence of price per square meter in € is produced by the code which follows and is shown in Figure 10.3. Function `as.yearmon` belongs to package `zoo` (Zeileis and Grothendiek, 2005), which is also used for time series manipulations below.

```
> require(zoo)
> boxplot(Price/M2built ~ as.yearmon(Date), data=Bilbao,
        ylab="EUR per square meter", xlab="Date",
        varwidth=TRUE, cex=0.5)
```

Although the general pattern of price evolution can be ascertained from this graph, it is apparent that the large spread in prices within any given month dwarfs the month-to-month mean (or median) change. This, of course, is the result of the high heterogeneity in the quality and location of houses entering the market at any given month.

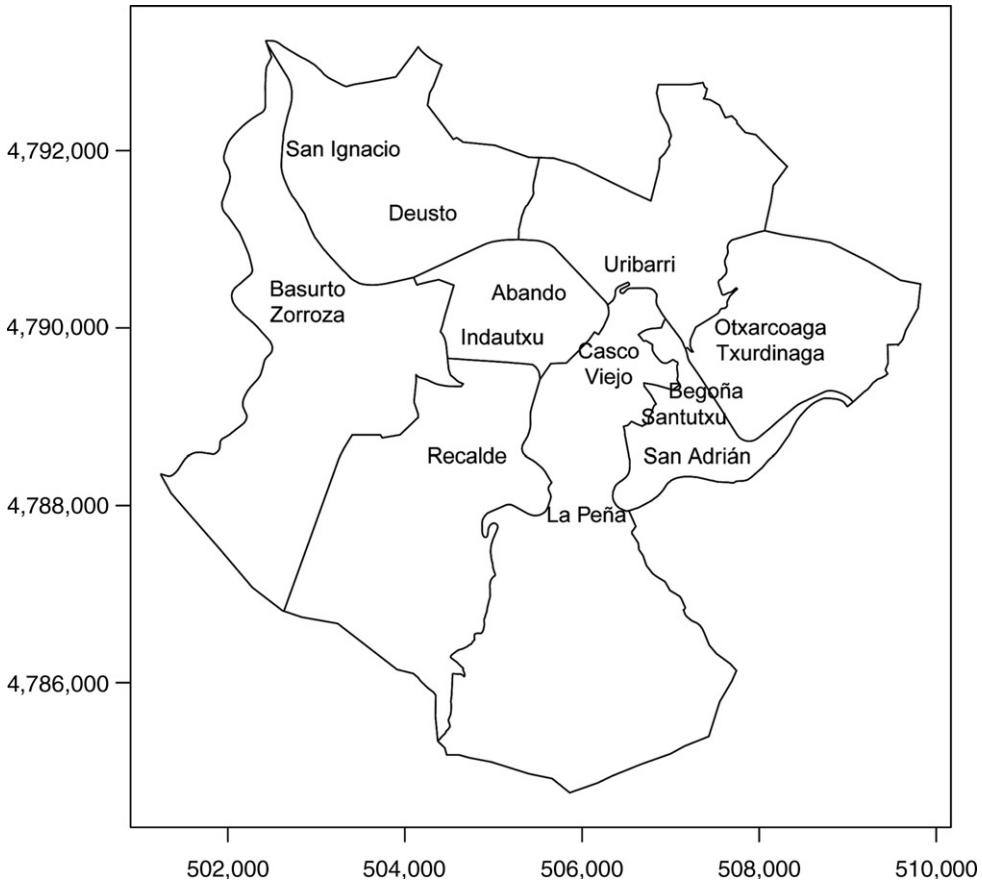


Figure 10.2
Districts of Bilbao. The axis units are UTM coordinates within the sheet 30N, in meters.

Because our goal is to obtain a price index, the fact that this heterogeneity may present a temporal evolution is particularly worrisome. The data are observational rather than the outcome of a well designed experiment, and hence, subject to selection biases because of temporal changes in quality or location. We may investigate for instance the distribution of offers from different districts over time using a code such as:

```
> attach(Bilbao)
> work1 <- table(Date,District)
> work2 <- zoo(x=work1, order.by=as.Date(rownames(work1)))
> work3 <- aggregate(x=work2, by=as.yearmon, FUN=sum)
> coredata(work3) <- 100*coredata(work3) / rowSums(work3)
```

`work3` is now a multivariate time series, each of whose columns contains the percentage of offers from a district. To avoid clutter, we aggregate districts with high, middle and low

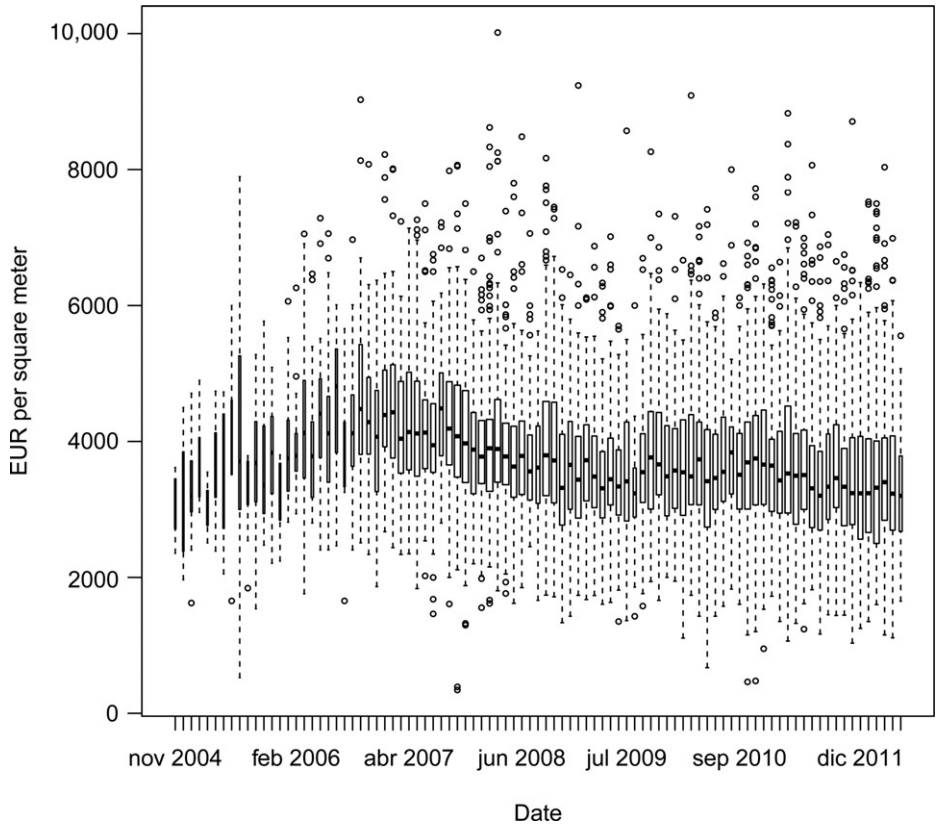


Figure 10.3

Housing prices in Bilbao. Monthly evolution and spread of offered prices in € per m^2. The median price can be taken as a rudimentary price index, but notice its erratic behavior.

prices in time series `work4` (not shown); then, the following code produces the plot in Figure 10.4.

```
> require(RColorBrewer)
> myPalette <- colorRampPalette(brewer.pal(3, "Spectral"))
> oldpar <- par()
> par(xpd=T, mar=par()$mar+c(9,0,0,0))
> barplot(t(coredata(work4)), names.arg=index(work4),
        ylab="House offers (% total)",col=myPalette(3))
> legend(7,-23,colnames(work4)[c(1,3,2)],cex=0.8,ncol=1,
        fill=myPalette(3)[c(1,3,2)])
> detach(Bilbao)
> par(oldpar)
```

What we see is that the share in the total number of offers of each group of districts changes appreciably from month to month. The variable relative abundance of offers from differently

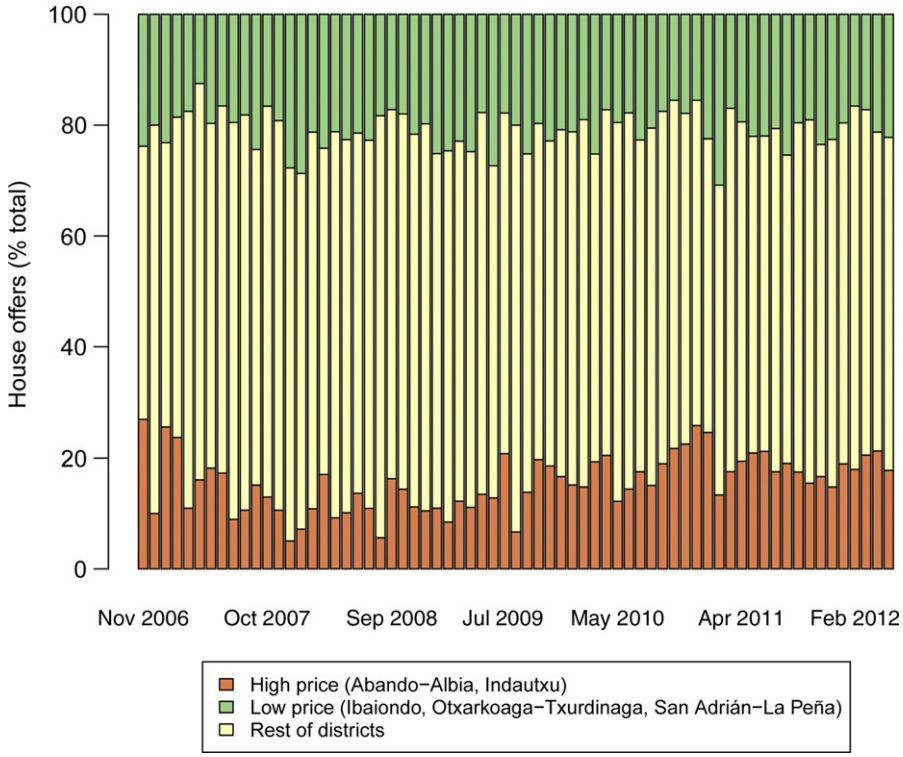


Figure 10.4
Housing prices in Bilbao. Monthly breakdown of number of offers by groups of similar price level districts.

priced groups explains in part the fluctuations observed in Figure 10.3. On top of that, of course, there is the heterogeneity of houses within each group, which also contributes to the high dispersion seen in Figure 10.3.

10.4 Real Estate Pricing Models

House attributes and locations change over time beyond our control, inducing the high variability in median (or mean) prices per square meter shown in Figure 10.3 and more clearly in Figure 10.5. One way to isolate the effect of interest to us—intrinsic, quality and location adjusted, real estate price level over time—is to fit a model using as response Price/m^2 or log(Price/m^2) and as explanatory variables all those available with plausible influence on the price *plus* a term capturing price evolution. This last term provides the basis for a constant quality and location price index.

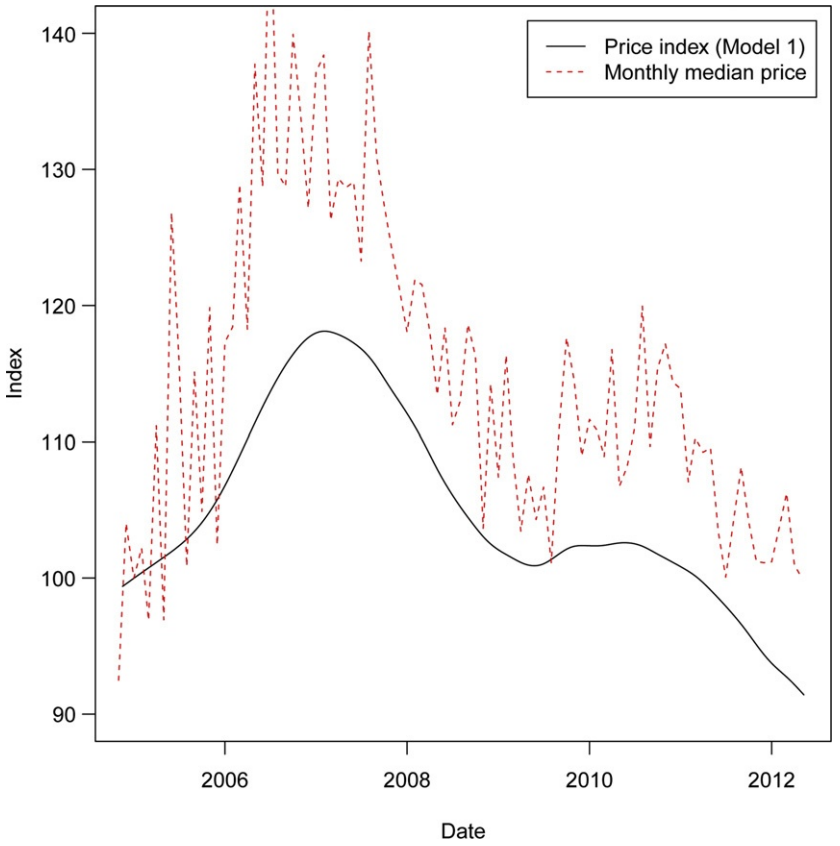


Figure 10.5
Housing price index for Bilbao: nonparametric estimate from Model 1 (solid) and median price (dashed). Base 100 January, 2005.

We have fitted several such models, whose relative merits are discussed next. It is remarkable that different specifications, using location information with various degrees of sophistication, still yield fairly similar estimates of the price index.

10.4.1 Model 1: Hedonic Model Plus Smooth Term

Our first model is a standard hedonic model in which the response is $\log(\text{Price}/\text{m}^2)$. Location enters the model through a categorical variable, which codes the district where each offered house is located, of which there are 13: refer to Figure 10.2. Other effects introduced are type of dwelling, type of heating, number of parking places, number of bathrooms, number of bedrooms, existence of an elevator, the monthly cost of shared services, and age of the building, coded in six categories.

The effect of time is modeled nonparametrically, as a smooth function of time, $s(t)$. Thus, Model 1 can be written as:

$$\log\left(\text{Price}/\text{m}^2\right) = \sum_{i=1}^{p} \beta_i x_i + s(t) + \varepsilon \tag{10.2}$$

where the x_i ($i = 1, \ldots, n$) are the observed regressors, $s(t)$ is a cubic spline whose suitably normalized value provides our estimation of the price index and ε is a random disturbance.

Estimation is easily accomplished in R. We first generate the `x` variable (time) as number of days from first observed date, and then fit the model with a single invocation of function `gam`, in the R package of the same name (Hastie, 2011). Alternative functions exist in package `mgcv` (Wood, 2001, 2004).

```
> require(gam)
> x  <-  as.numeric( Bilbao$Date - min(Bilbao$Date) + 1)
> mod.1  <-  gam(log(Price/M2built) ~ Bedrooms + Bathrooms
                 + Elevator + log(M2built) + Community
                 + ParkingPlaces + Heating + HouseType +
                 + District + Age + s(x,12), data=Bilbao)
```

The smooth function in `x` is modeled as a cubic spline. Degrees of freedom have been set by trial and error. In this example, 12 degrees of freedom appear to give a good compromise between fidelity and variance.

The values of the estimated parameters can be seen in Table 10.3. The R^2 is 0.5455. Because the response variable is measured in the log scale, the term $\exp(s(t))$ enters as a factor in Price/m^2 and our estimation for the index with base 100 at time t_0 is obtained as

$$I(t) = 100 \times \frac{\exp(s(t))}{\exp(s(t_0))}; \tag{10.3}$$

and the profile of $I(t)$ is shown in Figure 10.5, with an index of the median prices per square meter from Figure 10.3 overlaid. Both have been aligned so that their value at the base period (January 2005) is 100. The important point is that the model filters out the month-to-month variability, giving a much more plausible profile for a price index than the naive monthly median price can give.

We see that offered prices reached their maximum around the end of 2007, accumulating an increase of almost 25% since the beginning of 2005, then dropped until the first quarter of 2009, when they appear to have stabilized slightly above the early 2005 levels, only to resume their drop in 2010 and onward.

Examination of coefficients in Table 10.3 shows that all estimated coefficients (or groups of estimated coefficients, in the case of categorical variables) are highly significant and have the expected signs. We see for instance that new houses are clearly more valued, and that districts of Abando-Albia and Indautxu command distinctly higher prices than the rest. The

Table 10.3 Model 1
$\log(\text{Price}/\text{m}^2) = \sum_{i=1}^{p} \beta_i x_i + s(t) + \varepsilon$

	Estimate	Std. Error	*t* Value	Pr(>|*t*|)
(Intercept)	9.8357	0.0461	213.53	0.0000
Bedrooms	−0.0186	0.0036	−5.18	0.0000
Bathrooms	0.1102	0.0059	18.75	0.0000
ElevatorTRUE	0.1657	0.0059	28.15	0.0000
log(Square meters)	−0.3637	0.0121	−30.05	0.0000
Community	0.0007	0.0001	9.25	0.0000
ParkingPlaces	0.0754	0.0062	12.16	0.0000
Type of heating	**(ref. level: none)**			
Individual	0.1309	0.0069	19.01	0.0000
Collective	0.1738	0.0089	19.55	0.0000
Type of house	**(ref. level: penthouse)**			
Duplex	−0.0137	0.0247	−0.55	0.5809
Studio	−0.1972	0.0330	−5.97	0.0000
Flat	−0.0141	0.0097	−1.45	0.1475
Single house	0.0382	0.0687	0.56	0.5780
House in a block	−0.0978	0.0448	−2.18	0.0291
Paired house	0.1622	0.0731	2.22	0.0264
District	**(ref. level: Abando-Albia)**			
Basurto-Zorroza	−0.2798	0.0104	−26.88	0.0000
Begoña-Santutxu	−0.2437	0.0111	−22.04	0.0000
Casco Viejo	−0.1073	0.0126	−8.52	0.0000
Deusto	−0.2138	0.0116	−18.49	0.0000
Ibaiondo	−0.3384	0.0101	−33.55	0.0000
Indautxu	0.0445	0.0107	4.15	0.0000
Otxarkoaga-Txurdinaga	−0.3549	0.0146	−24.37	0.0000
Rekalde	−0.2237	0.0098	−22.73	0.0000
San Adrián-La Peña	−0.3419	0.0143	−23.83	0.0000
San Ignacio	−0.2438	0.0153	−15.98	0.0000
Uribarri	−0.2453	0.0103	−23.75	0.0000
Age of building	**(ref. level: unknown)**			
<5 years	0.0884	0.0119	7.43	0.0000
5-10 years	0.0527	0.0123	4.29	0.0000
10-20 years	0.0139	0.0136	1.02	0.3061
20-30 years	−0.0155	0.0084	−1.85	0.0649
Age + 30 years	−0.0283	0.0065	−4.32	0.0000
Time trend				
$s(x, 12)$	−0.0001	0.0000	−27.98	0.0000

negative coefficient of log(Square meters) reflects that large houses are cheaper in proportion to size, and hence, have a lower price per square meter with all other things being equal.

Model 1 in (10.2) uses the district where each house is located as a very coarse spatial indicator. For a subset of the sample, there are precise UTM coordinates of the entrance of the building which we can use instead.

10.4.2 Model 2: GWR Plus a Smooth Term

We now consider the model

$$\log\left(Price/\mathrm{m}^2\right)_i = \sum_{j=1}^{p} \beta_{i,j} x_{ij} + s(t) + \varepsilon_i \tag{10.4}$$

where x_{ij} is the value of the jth regressor for house i. Coefficients $\beta_{i,j}$ are estimated for each regressor j at each house location i.[2]

Model 2 in (10.4) is similar to Model 1 in (10.2), but the beta coefficients are now allowed to change in space. Thus, we allow for the fact that parking space or existence of an elevator, for instance, may be differently valued at different places. The time trend $s(t)$, on the other hand, is unique.

Except for the smooth term $s(t)$ in (10.4), the estimation can be done by least squares, with observations weighted less as their distance to the house i increases. This is GWR, Harris et al. (2010). Weighting is usually done with a two dimensional kernel, here an isotropic Gaussian kernel, whose bandwidth is set by cross-validation or selected by the analyst at some plausible value. With the sample used, the bandwidth selected[3] (defined as the standard deviation of the Gaussian kernel) has been 400, which implies that observations 400 m away are given weights about 6% of those right at the space point at which we estimate the coefficients ($6\% \approx 0.0585$, density of an isotropic bivariate normal density one standard deviation away from the mean).

Although there is software readily available to perform GWR (like package `spgwr`, R. Bivand and Yu (2012), that we have used), the inclusion of a smooth term such as $s(t)$ in (10.4) cannot be handled directly. However, given routines for GWR and spline smoothing, it is fairly straightforward to implement a back-fitting estimation routine, Hastie and Tibshirani (1991), Section 4.4, for Model 2. The procedure can be sketched as follows:

[2] Although for simplicity we are assuming that we estimate parameters only at locations where we observe a house, this need not be the case. We can estimate parameters at any point in space, irrespective of whether or not an observation is present there. In particular, we might estimate parameters over a grid of points to construct maps by smoothing and interpolation of the obtained values.

[3] As the geocoding is made to UTM coordinates in meters, the bandwidth is also in meters.

Step 0. Take as initial estimate $s^{(0)}(t)$ of $s(t)$ in (10.4) the smooth function estimated from (10.2) (or any other available approximation).

Step 1. At iteration k, let $\beta_{ij}^{(k)}$ be the estimates of the $\beta_{i,j}$ using GWR to fit

$$\log(P_{it}) - s^{(k)}(t) = \sum_{j=1}^{p} \beta_{i,j}^{(k)} x_{ij} + \varepsilon_{it}^{(k)} \tag{10.5}$$

Step 2. At iteration k, obtain $s^{(k+1)}(t)$ by smoothing over time the partial residuals $\log(P_{it}) - \sum_{j=1}^{p} \beta_{ij}^{(k)} x_{ij}$.

Step 3. For a pre-set tolerance η, if max $|s^{(k)}(t) - s^{(k+1)}(t)| < \eta$, return as estimates the $\beta_{i,j}^{(k+1)}$ and $s^{(k+1)}(t)$ computed in the final iteration, otherwise return to Step 1.

Essentially, the back-fitting algorithm iterates Steps 1 and 2, each time estimating the parametric or nonparametric part of the fit given the best current approximation of the other. This can be easily implemented in a few lines of R. First, we subset the observations with coordinates and define the auxiliary variable `logP` as the log of price per square meter:

```
> Bilbao.g  <-  subset(Bilbao, !is.na(UTMX + UTMY))
> scratch  <-  with(Bilbao.g, log(Price/M2built))
> Bilbao.g  <-  cbind(Bilbao.g, logPM2=scratch)
```

Dataframe `Bilbao.g` contains 3225 observations. To start the iteration, we can use as $s^{(0)}(t)$, the smooth function estimated in Model 1. We align its values with the dates of observations in `Bilbao.g`:

```
> r  <- range(Bilbao.g$Date)
> dates  <- match(Bilbao.g$Date, seq(from=r[1],to=r[2],by="day"))
> smooth.old <- s0[dates]        # smooth term from Model 1
```

Then, we can write the back-fitting algorithm as follows:

```
> require(spgwr) ; gpclibPermit()
> tol  <- 0.0035 ; iter <- 0 ; bw <- 400
> xy  <- as.matrix(Bilbao.g[,c("UTMX","UTMY")])
> repeat {
    def1 <- Bilbao.g$logPM2 - smooth.old
    mod2 <- gwr(def1 ~ Bedrooms + Bathrooms + Elevator
        + log(M2built) + Community + ParkingPlaces
        + Heating + HouseType + Age,
        coords=xy, bandwidth=bw, data=Bilbao.g)
resids <- Bilbao.g$logPM2 - mod2$SDF@data$pred
smooth.new <- gam(resids ~ s(dates,12))$fitted.values
if ( max(abs(smooth.new-smooth.old)) < tol)
  break
smooth.old <- smooth.new ; iter <- iter + 1
}
```

The `repeat` loop exits when the largest absolute difference among two consecutive estimations of the smooth term drops below a preset tolerance (0.0035 in our case, leading to 2

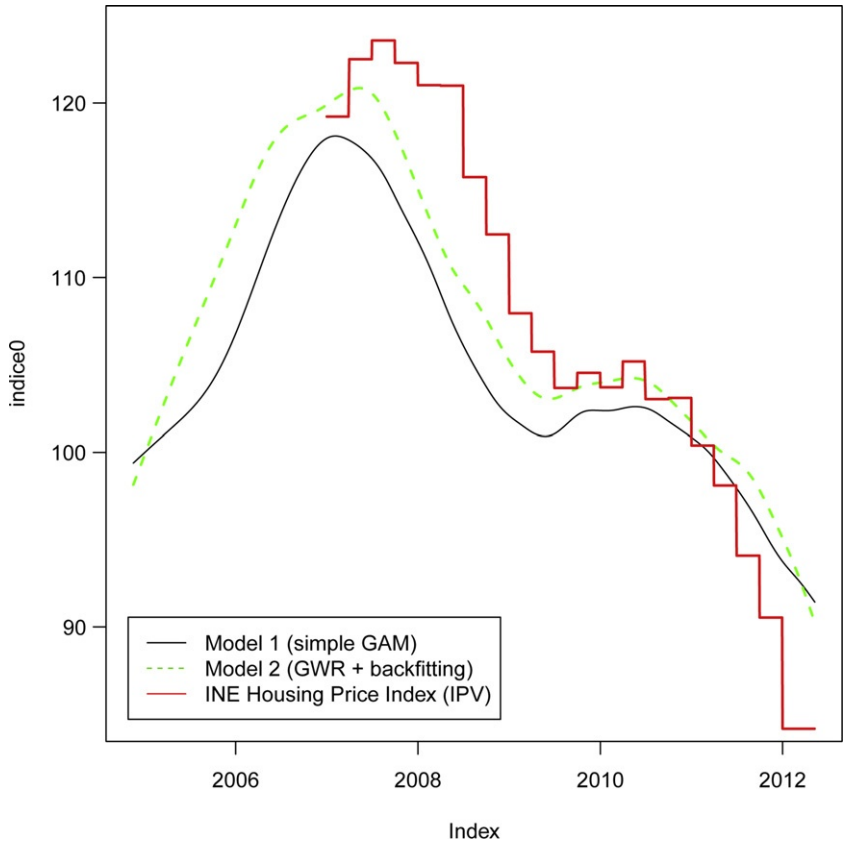


Figure 10.6

Asking prices indices. The index from INE (base 2007) has been aligned with the average of our two indices at 2007-02-15, mid point of the first quarter for which it was published.

iterations). From `smooth.new`, we can compute the new back-fitted index (`ibf`) using formula (10.3) and make it a time series for plotting:

```
> sel  <-  match(unique(dates),dates)
> ibf  <-  zoo(smooth.new[sel], order.by=Bilbao.g$Date[sel])
> base <-  match(as.Date("2005-01-05"),index(ibf))
> ibf  <-  100* exp(ibf) / exp(coredata(ibf)[base])
```

Figure 10.6 shows the index obtained from Model 1, the index obtained from Model 2, and the Índice de Precios de la Vivienda, or Housing Price Index (IPV) computed quarterly by the Instituto Nacional de Estadística, or National Statistical Institute (INE) for the whole of the Basque Country[4]; for a description of the methodology, see Gila García and

[4] Available from their web page at `www.ine.es`.

Novás Filgueira (2012). Note that since the IPV is computed with base 2007 = 100, we have aligned their first published figure with the half sum of our two indices.

Although the IPV is computed for a larger area and with different methodology, its profile is remarkably similar to our two indices. In particular, all of them nicely reflect a short-lived upturn in late 2009, likely due to the announced end of fiscal rebates for house buyers, which prompted a small surge of activity in the housing market.

The similar profiles of the IPV and offered prices indices are also significant in that they help to allay fears that they could measure entirely different things. One might hypothesize that offered prices are relatively sticky, and softening market conditions would surface rather in transaction prices, which are input to the IPV. This does not seem to be the case. It is reassuring that an index computed with a much larger information base (and much greater cost), with a totally different methodology, still yields similar results.

Model 2 produces a wealth of geographically disaggregated information. The parametric part (i.e., the right hand side of Equation (10.5) evaluated with the estimated $\beta_{i,j}^{(k)}$, suitably scaled, gives estimates of the deflated log prices per square meter $\log(P_{it}/m^2) - s(t)$. Aligning values to the base period and transforming back to the original scale (€/m^2), we get the values mapped in Figure 10.7, produced by the code next. First, predictions are recovered from the object `mod2` returned by function `gwr` and exponentiated:

```
> val <- mod2$SDF[!is.na(mod2$SDF$pred),]
> val@data$pred <- exp(val@data$pred)
```

Next, a single call to function `spplot` in package `sp` produces the graph. Function `spplot` calls functions in the `lattice` package (Sarkar, 2008): some of the arguments passed to `spplot` are documented in the last package. We have chosen a divergence palette from package `colorspace` (Zeileis et al., 2009).

```
> require(colorspace)
> cuts   <- c(0,3000,3250,3750,4500,Inf)
> colors <- diverge_hcl(length(cuts)-1)
> fig    <- spplot(val,c("pred"), col.regions=colors,
                   cuts=cuts,cex=0.6, key.space="bottom",
                   scales=list(draw=TRUE),xlim=c(500000,510000),
                   ylim=c(4784500,4793500),
                   sp.layout=list("sp.polygons",Bilbao.utm))
> print(fig)
```

We have made no attempt to correct biases due to the nonlinear log transformation (Jensen's inequality). Consider the case where we target the mean value of x_t and we have a model such as $y_t = \log(x_t) = m_t + \varepsilon_t$, with ε_t Gaussian of zero mean. If $\hat{m}_t$ is unbiased for $E[y_t] = m_t$, it can be shown (Granger and Newbold, 1976; see also Mayr and Ulbricht, 2007) that $\exp(\hat{m}_t)$ is biased for $E[\exp(y_t)] = E[x_t]$; in fact,

$$\hat{x}_t = E[\exp(m_t + \varepsilon_t)] = \exp(\hat{m}_t) E[\exp(\hat{\varepsilon}_t)]. \tag{10.6}$$

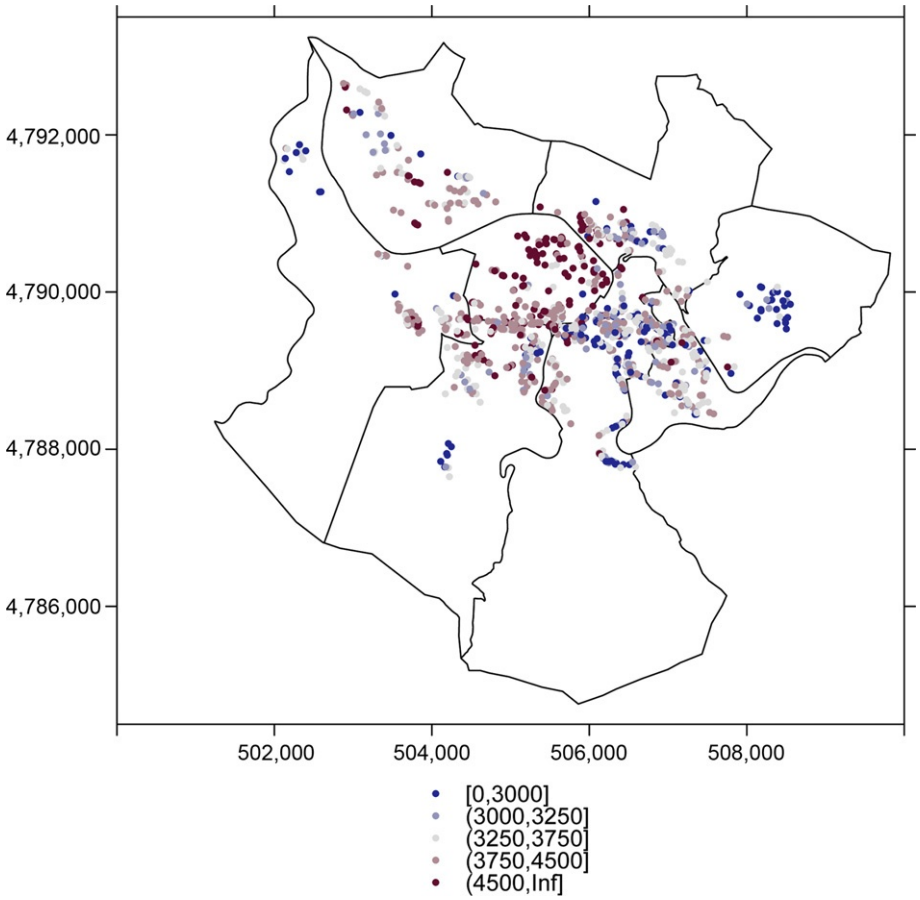


Figure 10.7
Fitted values in € per square meter at the base period (January, 2005).

Thus, the naive estimates $\exp(\hat{m}_t)$ will be off by a multiplicative factor $E[\exp(\hat{\varepsilon}_t)]$. In index construction, however, only relative prices at different time points matter (refer to formula (10.3)), so a multiplicative factor is of no concern, as long as it is constant or nearly constant. In Figure 10.7, on the other hand, prices incorporate a small bias, but we are not concerned with absolute prices (which would be anyway *offered* prices, useful only as proxies for unobserved transaction prices).

The pattern in Figure 10.7 is clear; houses in the central part of the city command the highest prices, while peripheral districts of San Ignacio, Txurdinaga, or Basurto exhibit lower prices.

It is also interesting to check the fit, which now is local. Because we compute a different regression for each location, we have local R^2 coefficients that we have plotted in Figure 10.8.

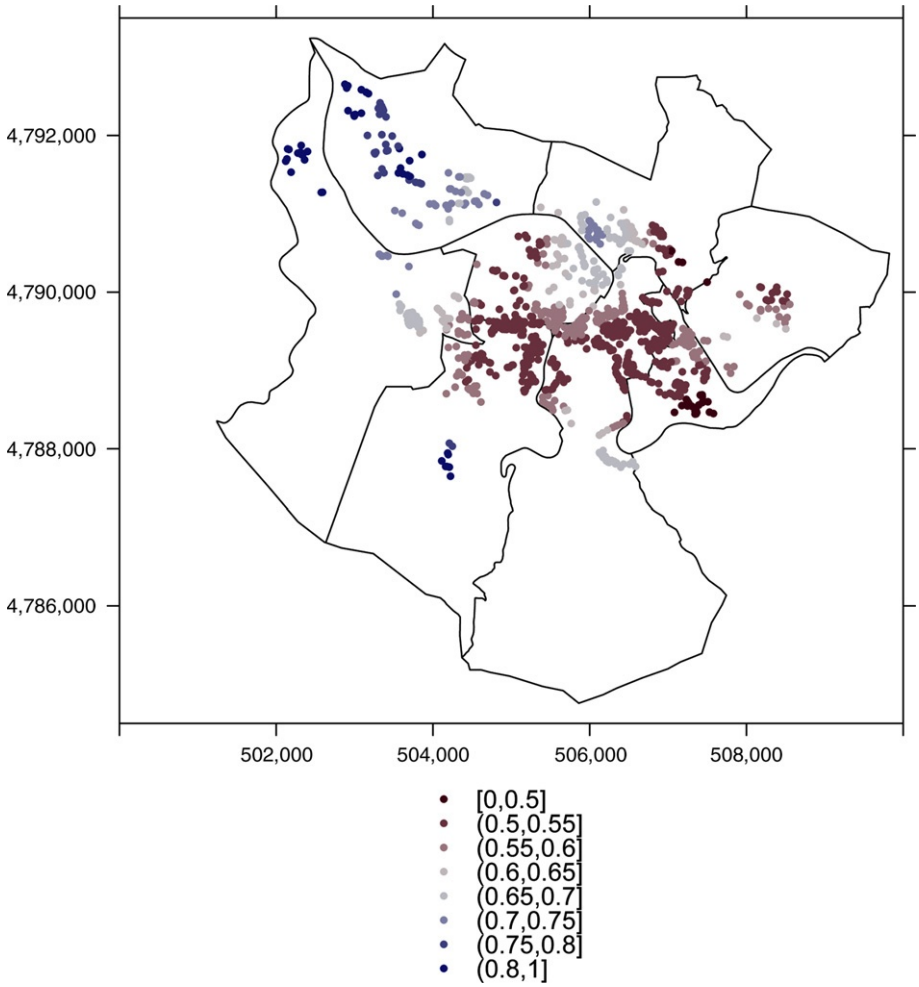


Figure 10.8
Local (partial) R^2 values for the regression $\log(P_{it}) - s^{(k)}(t) = \sum_{j=1}^{p} \beta_{i,j}^{(k)} x_{ij} + \varepsilon_{it}^{(k)}$.

The fit of deflated log prices per square meter with the available regressors is reasonably good, with the median R^2 equal to 0.5502. There are localized areas of worse and better-than-average fit for which we have no obvious explanation.

In principle, local values for any of the $\beta_{i,j}^{0}$ coefficients in Equation (10.4) could be spatially represented. For instance, we could have an estimate of how much an additional bedroom or the existence or absence of an elevator or a garage enhances or depresses the price of a house at each chosen location.

This may be misleading: GWR has been the subject of much controversy recently, because estimated coefficients appear sometimes to be unstable and take counter-

intuitive signs. As pointed out in Wheeler and Tiefelsdorf (2005), local multicollinearity is usually a problem. Some remedies have been proposed (Wheeler, 2006) which we have not attempted to implement, because for the purpose of price index construction we only require a good estimate of $\sum_{j=1}^{p} \beta_{i,j}^{(k)} x_{ij}$ in (10.5) and are less concerned with apportioning that sum into the contributions of each predictor.

10.4.3 Relationship to Other Work

Ours is but a possible approach to a much researched problem; for a collection of papers from a spatio-temporal vantage point, see Biggeri and Ferrari (2010). Specifically related to housing prices and close to our approach are, among many others, Borst (2008), Geniaux and Napoléone (2008), Clapp (2004), Meese and Wallace (1997) Bourassa et al. (2006), McMillen (2011), and McMillen and Redfearn (2007). To highlight similarities or differences with the approach followed in Model 2, comments on some of their respective approaches follow and Table 10.4 gives a summary. We can make no attempt to even list all contributions in this area, which has produced a copious literature.

McMillen's (2011) goal is to examine the effect over time on house prices of the distance to "employment subcenters" in the city of Chicago. He uses a repeated sales estimator and a smooth function to capture the effect of time. His model takes the form

$$Y_{i,t} - Y_{i,s} = g(t) - g(s) + u_{i,t} - u_{i,s}, \tag{10.7}$$

where $Y_{i,t}$ is the (log) price of house i at time t, u_{it} the corresponding random term and $g(t)$ is a smooth function of time given by

Table 10.4 Summary and Comparison of Some Models and Indices Discussed

Model or Index	Quality Adjustment	Spatial Effects	Time Effects
Geniaux and Napoleone (2008): $Y_i = \beta X_i + Z_i s(u_i, v_i) + s(u_i, v_i) + \varepsilon_i$	Hedonic, fixed β	Smooth $s(u_i, v_i)$	None
McMillen (2011): $Y_{i,t} - Y_{i,s} + g(t) - g(s) + \varepsilon_{i,t} - \varepsilon_{i,s}$	Repeated sales	Not explicitly	$g(t)$
Clapp (2004): $Y_{it} = \beta X_i + s(u_i, v_i, t) + \varepsilon_{it}$	Hedonic, fixed β	Joint smooth $s(u_i, v_i, t)$	
Meese and Wallace (1997): $Y_{i(t)} - \bar{Y}_t = G(x_{i(t)}) + \varepsilon_{i(t)}$	Hedonic	LWR[†] in attributes	Estimated for each t
Housing Prices Survey (IPV): $Y_{i,c,t} = X_c \beta_t + e_{i,c,t}$	Hedonic, variable β	Not explicitly	Estimated for each t
Model 2 (Section 4.2 above): $Y_{it} = \beta_i X_{it} + s(t) + \varepsilon_{it}$	Hedonic, variable β	GWR[‡]	Smooth $s(t)$

[†]LWR, locally weighted regression.
[‡]GWR, geographically weighted regression.

$$g(t) = \alpha_0 + \alpha_1 z + \alpha_2 z^2 + \sum_q \left(\lambda_q \sin(qz) + \gamma_q \cos(qz)\right), \tag{10.8}$$

with $z = 2\pi t/\max(t)$. As compared with our Model 2, he does not need to fit a hedonic model, because the effect of the quality of house i cancels in the difference $Y_{i,t} - Y_{i,s}$ (assuming the characteristics of the house i remained constant from time s to time t); the downside is that only houses with repeated sales in the sample can be used (with a possible selection effect, as poor quality houses might be more likely to enter the market repeatedly). To capture the effect of time, a second order polynomial plus a combination of periodic functions is used. The profile of $g(t)$ suitably normalized would give an estimate of a price index similar to ours, although for a given number of degrees of freedom we regard the penalized spline approach we have used as more flexible.

Geniaux and Napoléone (2008) investigate spatial effects, in particular the distance to the urban center, on house prices. Although they are not particularly concerned with the problem of modeling the effect of time (which enters some of their models only in the form of year dummies), their models are close to ours; in their Section 5.4.3, they propose an estimation method for the MGWR model which is close to the back-fitting algorithm sketched above for Model 2.

Close to this approach is also Clapp (2004). He considers fixed betas in the hedonic part and a smooth function in time and space, estimated by the local polynomial regression, using a product kernel, to account for trends in time and space. The estimated $s(u_i, v_i, t)$ as a function of time for given u_i, v_i gives a *local* price index.

Meese and Wallace (1997) compare several methods, providing a wealth of insight into their respective merits. They propose yet another model,

$$Y_{i(t)} - \bar{Y}_t = G\left(x_{i(t)}\right) + u_{i(t)}; \tag{10.9}$$

Equation (10.9) is estimated for each quarter; $Y_{i(t)}$ is the log price of the i house traded at quarter t, and $\bar{Y}_t$ is the average log prices of all houses traded in that same quarter. $G(x_{i(t)})$ is the hedonic part, estimated by locally weighted regression (LWR); weighting is done not in terms of geographical distance, but rather distance in the attribute space to the attributes median. The implicit prices of the attributes are then used to correct $\bar{Y}_t$ with the valuation of the attributes of the current or first quarter. This gives Laspeyres and Paasche indices whose geometric mean produces a final Fisher's ideal index, Vogt and Barta (1996).

Our Model 2 produces an index for the whole area sampled, rather than indices for each particular location—like, e.g., Clapp (2004). Where other models, such as Geniaux and Napoléone (2008), use smooth nonparametric functions in geographical coordinates to account for spatial effects, our model accounts for spatial variation through GWR of hedonic coefficients, the $s(t)$ term smoothing along the time dimension, much like McMillen (2011). Our model is geared towards production of an index for a region, rather than the discovery of subregions with different price trends.

10.5 Conclusion

Our goal was to assess the feasibility of building a house price index with publicly available information on proxy variables of the true magnitude to estimate. This is quite in keeping with the modern trend to exploit administrative records or automatically collected information to replace, to the extent possible, costly surveys.

The limited evidence shown in the preceding section is encouraging: with off-the-shelf software and limited human resources, basically spent in data cleaning and geocoding, we have been able to compute an index which captures remarkably well the patterns in the IPV. The method is straightforward, simple to implement, understand, and track.

There is no question that R provides unmatched power and flexibility to attack the problem through all the stages, from data collecting and Web scrapping to data preprocessing, visualization, model estimation, and diagnosis and results summarization. It is this flexibility and breadth that has given R such a prominent place in the toolbox of the applied data analyst, as evinced by the large and growing number applications and monographs on data mining with R, e.g., Torgo (2011), Spector (2008), or Zhao (2013).

Acknowledgments

Partial support from Grants ECO2008-05622 (MCyT) and IT-347-10 (Basque Government) is gratefully acknowledged.

References

Baranzini, A., Ramirez, J., Schaerer, C., Thalmann, P. (Eds.), 2008. Hedonic methods in housing markets. Springer Verlag, New York, NY.

Biggeri, L., Ferrari, G. (Eds.), 2010. Price Indexes in Time and Space. Springer Verlag, New York.

Bivand, R., Yu, D., 2012. spgwr: geographically weighted regression [computer software manual]. Available from: http://CRAN.R-project.org/package=spgwr (R package version 0.6–18).

Bivand, R.S., Pebesma, E.J., Gómez-Rubio, V., 2008. Applied spatial data analysis with R. Springer Verlag, New York.

Borst, R., 2008. Time-varying model parameters: obtaining time trends in a hedonic model without specifying their functional form. J. Property Tax Asses. Adm. 6 (4), 29–36.

Bourassa, S., Hoesli, M., Sun, J., 2006. A simple alternative house price index method. J. Hous. Econ. 15 (1), 80–97.

Clapp, J.M., 2004. A semiparametric method for estimating local house price indices. Real Estate Econ. 32 (1), 127–160.

Dubin, R., 1988. Estimation of regression coefficients in the presence of spatially autocorrelated error terms. Rev. Econ. Stat. 70 (3), 466–474.

Fotheringham, S., Charlton, M., Brunsdon, C., 2002. Geographically weighted regression: The analysis of spatially varying relationships. Wiley, West Sussex, England.

Friedl, J. (2006). *Mastering regular expressions*. O'Reilly Media, Inc. Sebastopol, CA, USA.

Geniaux, G., Napoléone, C., 2008. Semi-parametric tools for spatial hedonic models: An introduction to mixed geographically weighted regression and geoadditive models. In *Hedonic methods in housing markets* (p. 101–127). Springer Verlag, New York.

Gila García, A., Novás Filgueira, M., 2012. The regression model in the House Price Index. Bol. Estad. Inv. Oper. 28 (3), 247–260.
Goldberg, K.H., 2008. XML: Visual Quickstart Guide. Peachpit Press, Berkeley, USA.
Granger, C., Newbold, P., 1976. Forecasting transformed series. J. R. Stat. Soc. Ser. B. 38 (2), 189–203.
Gusfield, D., 1997. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, USA.
Harris, P., Fotheringham, A.S., Crespo, R., Charlton, M., 2010. The use of geographically weighted regression for spatial prediction: an evaluation of models using simulated data sets. Math. Geosci. 42 (6), 657–680.
Hastie, T., 2011. gam: generalized additive models [computer software manual]. Available from: http://CRAN.R-project.org/package=gam (R package version 1.06.2).
Hastie, T., Tibshirani, R., 1991. Generalized additive models, second ed. Chapman & Hall, London.
Henneberry, J., 1998. Transport Investment and House Prices. J. Prop. Valuat. Invest. 16 (2), 144–157.
Kahle, D., Wickham, H., 2012. ggmap: a package for spatial visualization with google maps and openstreetmap [computer software manual]. Available from: http://CRAN.R-project.org/package=ggmap (R package version 2.1).
Keitt, T.H., Bivand, R., Pebesma, E., Rowlingson, B., 2012. rgdal: bindings for the geospatial data abstraction library [computer software manual]. Available from: http://CRAN.R-project.org/package=rgdal (R package version 0.7-12).
Kestens, Y., Thériault, M., Des Rosiers, F., 2005. Heterogeneity in hedonic modelling of house prices: looking at buyers' household profiles. J. Geogr. Syst. 8 (1), 61–96.
Kryvobokov, M., Wilhelmsson, M., 2007. Analysing location attributes with a hedonic model for apartment prices in Donetsk, Ukraine. Int. J. Strateg. Prop. Manag. 11 (3), 157–178
Lang, D.T., 2012. XML: tools for parsing and generating XML within R and S-Plus. [computer software manual]. Available from: http://www.omegahat.org/RSXML (R package version 3.93–0).
LeSage, J., Pace, R. (Eds.), 2011. Advances in Econometrics. Emerald, New York, NY.
Lewin-Koh, N.J., Bivand, R., 2012. maptools: tools for reading and handling spatial objects [computer software manual]. Available from: http://CRAN.R-project.org/package=maptools (R package version 0.8-16).
Loecher, M., Berlin School of Economics and Law, 2012. Rgooglemaps: overlays on google map tiles in R [computer software manual]. Available from: http://CRAN.R-project.org/package=RgoogleMaps (R package version 1.2.0).
Mayr, J., Ulbricht, D., 2007. Log versus level in var forecasting: 16 million empirical answers—expect the unexpected (Tech. Rep. No. 42). IFO Institute for Economic Research, University of Munich.
McMillen, D., Redfearn, C., 2007. Estimation, Interpretation, and Hypothesis Testing for Nonparametric Hedonic House Price Functions. Unpublished report.
McMillen, D.P., 2011. Employment subcenters and home price appreciation rates in metropolitan Chicago. In: LeSage, J., Pace, R. (Eds.), Emerald, New York, NY, pp. 237–257.
Meese, R.A., Wallace, N.E., 1997. The construction of residential housing price indices: a comparison of repeat-sales, hedonic-regression, and hybrid approaches. J. Real Estate Finance Econ. 73, 51–73.
Neuwirth, E., 2011. Rcolorbrewer: colorbrewer palettes [computer software manual]. Available from: http://CRAN.R-project.org/package=RColorBrewer (R package version 1.0–5).
Pace, R., Barry, R., Gilley, O., Sirmans, C., 2000. A method for spatial-temporal forecasting with an application to real state prices. Int. J. Forecast. 16, 229–246.
Pebesma, E., Bivand, R., 2005. Classes and methods for spatial data in R. R News, 5(2). Available from: http://cran.r-project.org/doc/Rnews.
Rodríguez López, J., 2007. Los índices de precios de la vivienda, Problemática. Indice. Revista de Estadística y Sociedad. 22, 14–16.
Rosen, S., 1974. Hedonic prices and implicit markets: product differentiation in pure competition. J. Polit. Econ. 82, 34–55.
Sarkar, D., 2008. Lattice. Multivariate Data Visualization with R. Springer, New York.

Spector, P., 2008. Data Manipulation with R. Springer, New York.
Torgo, L., 2011. Data Mining with R. CRC Press, Boca Raton, USA.
Vogt, A., Barta, J., 1996. The making of tests for index numbers. Physica-Verlag, Heidelberg, Germany.
Wheeler, D., Tiefelsdorf, M., 2005. Multicollinearity and correlation among local regression coefficients in geographically weighted regression. J. Geogr. Syst. 7 (2), 161–187.
Wheeler, D.C., 2006. Diagnostic tools and remedial methods for collinearity in linear regression models with spatially varying coeffcients. Unpublished doctoral dissertation, Ohio State University.
Wood, S., 2001. mgcv: GAMs and generalized ridge regression for R. R News, 1(2), pp. 20–25. Available from: http://CRAN.R-project.org/doc/Rnews/.
Wood, S., 2004. Stable and efficient multiple smoothing parameter estimation for generalized additive models. J. Am. Stat. Assoc. 99, 673–686.
Zeileis, A., Grothendiek, G., 2005. Zoo: S3 infrastructure for regular and irregular time series. J. Stat. Softw. 14 (6), 1–27.Available from:http://www.jstatsoft.org/.
Zeileis, A., Hornik, K., Murrell, P., 2009. Escaping RGBland: selecting colors for statistical graphics. Comput. Stat. Data Anal. 53, 3259–3270.
Zhao, Y., 2013. R and Data Mining. Academic Press, Amsterdam, forthcoming.

CHAPTER 11

Predicting Seabed Hardness Using Random Forest in R

Jin Li, P. Justy, W. Siwabessy, Maggie Tran, Zhi Huang, Andrew D. Heap
Geoscience Australia, GPO Box 378, Canberra, ACT 2601, Canberra, Australia

11.1 Introduction

Seabed hardness is an important environmental property for predicting marine biodiversity that can be used to support marine zone management in Australia. This is because the seabed substrate is one of the most important factors controlling the spatial distribution of benthic marine communities: it influences the colonization and formation of ecological communities and the abundance of benthic organisms (Buhl-Mortensen et al., 2012; Newell et al., 2001; Post et al., 2006; Thrush et al., 2001; Warwick and Davies, 1977). Broadly, substrate hardness may also influence the nature of attachment of an organism to the seabed (Williams and Leach, 1999). Hard substrates provide environments that generally support sessile suspension feeders and soft (unconsolidated) substrates generally support discretely motile invertebrates (McArthur et al., 2010). Therefore, a significant aid to mapping and predicting the spatial distribution of benthic marine communities would be to have a spatially continuous measurement of seabed hardness.

Despite its importance, seabed hardness is difficult to measure. Traditional methods of measuring seabed hardness with a cone penetrometer only provide data at (often widely dispersed) discrete locations. Seabed hardness can be inferred by categorizing substratum from underwater video footage (Stein et al., 1992); however, the footage is usually also available at discrete locations. Furthermore, it is expensive and time consuming to acquire underwater video even over small areas. Seabed hardness has also been inferred from multibeam backscatter data (Anderson et al., 2008a; Basu and Saxena, 1999; Kloser et al., 2010) because it is correlated with multibeam backscatter (reflection) signal strength. Crucially, inferring seabed hardness over large areas has become possible with the recent widespread use and development of multibeam sonar systems for seabed mapping purposes.

Physical properties derived from multibeam backscatter and bathymetry may be useful for predicting seabed hardness. The nature of the seabed is a fundamental factor in controlling the

Data Mining Applications with R

received backscatter signals. Soft seabed substrates generally produce lower backscatter intensity; in contrast, hard seabed substrates are more likely to produce higher backscatter intensity (De Falco et al., 2010; Ferrini and Flood, 2006; Goff et al., 2000; Hamilton and Parnum, 2011). However, the strength of the return signal can be significantly attenuated by scattering from surface topography and the benthic organisms, leading to an apparent reduction in backscatter intensity. Likewise, well-sorted unconsolidated sediments can produce very strong seabed reflections in the absence of any surface topography. The existence of these factors can affect the reliability of acoustic data and needs to be appropriately addressed. At regional and continental scales, seabed sediments are often strongly correlated to bathymetry (Li et al., 2010, 2012b). At these scales, bathymetry and its derived properties are informative predictors of seabed substrate type. Given the added spatial seabed complexity, more work is required to determine whether they are equally informative predictors of seabed hardness at regional and local scales.

The recent technological advancements in sonar equipment have meant that significantly more biophysical data are available for making seabed hardness predictions. A vital decision is identifying the best modeling technique to generate a surface that most accurately represents the seabed hardness. Random forest (RF) was developed by Ho (1995, 1998), Breiman (2001), and Breiman et al. (1984) and the predictive ability of RF has been demonstrated in data mining and other disciplines (Cutler et al., 2007; Diaz-Uriarte and de Andres, 2006; Marmion et al., 2009; Okun and Priisalu, 2007; Prasad et al., 2006). The RF method has been used previously for generating continuous layers from point samples in the marine environmental sciences and outperformed a number of other statistical modeling techniques (Li et al., 2010, 2011b). Therefore, in this study, RF as being implemented via the function *randomForest* in R (Liaw and Wiener, 2002) was used to develop a model to predict seabed hardness.

In this study, we predict the spatial distribution of seabed hardness using RF based on video classification and seabed biophysical variables. We illustrate the effects of cross-validation methods including a new cross-validation function (*rf.cv*) on the selection of the most optimal RF predictive model. We also test the effects of the various predictor sets on the accuracy of RF predictive models. The most accurate model is used to predict the spatial distribution of "hard" and "soft" substrates. This study provides an example of predicting the spatial distribution of environmental properties using *randomForest* in R. All data manipulation and modeling were implemented in R (R Development Core Team, 2011). Snippets of the R code used in this study are embedded in the relevant sections. A dataset and the new R cross-validation function are provided in the appendices for readers who are interested in adopting our modeling work.

11.2 Study Region and Data Processing

This section describes relevant concepts and the procedures for deriving relevant data. It includes (1) study region, (2) point samples derived for seabed hardness, and (3) information about 15 environmental predictors.

11.2.1 Study Region

The study region is located in the eastern Joseph Bonaparte Gulf, northern Australian marine margin (Figure 11.1a) and four areas (A-D) in the region were used in this study (Figure 11.1b). Two marine surveys completed in 2009 (Heap et al., 2010) and 2010 (Anderson et al., 2011) collected high-resolution multibeam bathymetry and backscatter data and colocated seabed samples and underwater video transects across the four study areas (Figure 11.1b and c). The study areas comprise a spatially complex suite of geomorphic features including shallow flat-topped banks, terraces, ridges, deep valleys, and plains (Figure 11.1c).

11.2.2 Data Processing of Seabed Hardness

In total, 140 samples of seabed hardness were considered in this study. The hardness of the seabed substrate at these sampling locations was determined based on the underwater video transects taken at either the sampling locations or the nearest adjacent area (Siwabessy et al., 2012) (Figure 11.1b). The video footage was analyzed on the basis of a 15-s sequence of the transect to classify substratum composition (Anderson et al., 2008b). Substratum composition (i.e., rock, boulder, cobble, rubble, gravel, sand, and mud) as defined by Wentworth (1992) was estimated to a precision of 10% (i.e., 0%, 10%, 20%, . . . , 100%). We then categorized the

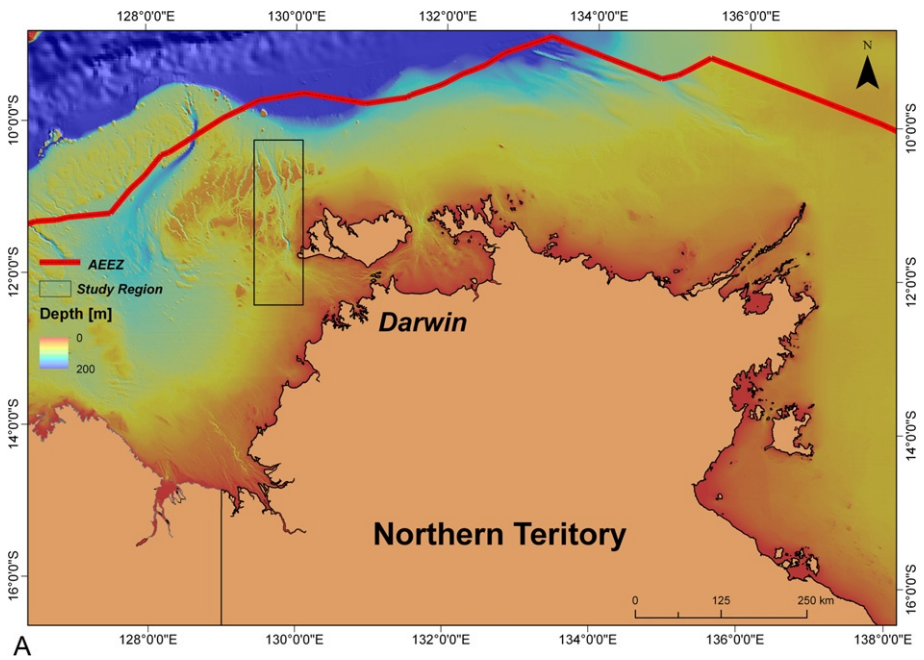


Figure 11.1

(a) Location of the study region in the eastern Joseph Bonaparte Gulf, northern Australian marine margin overlaid with bathymetry;

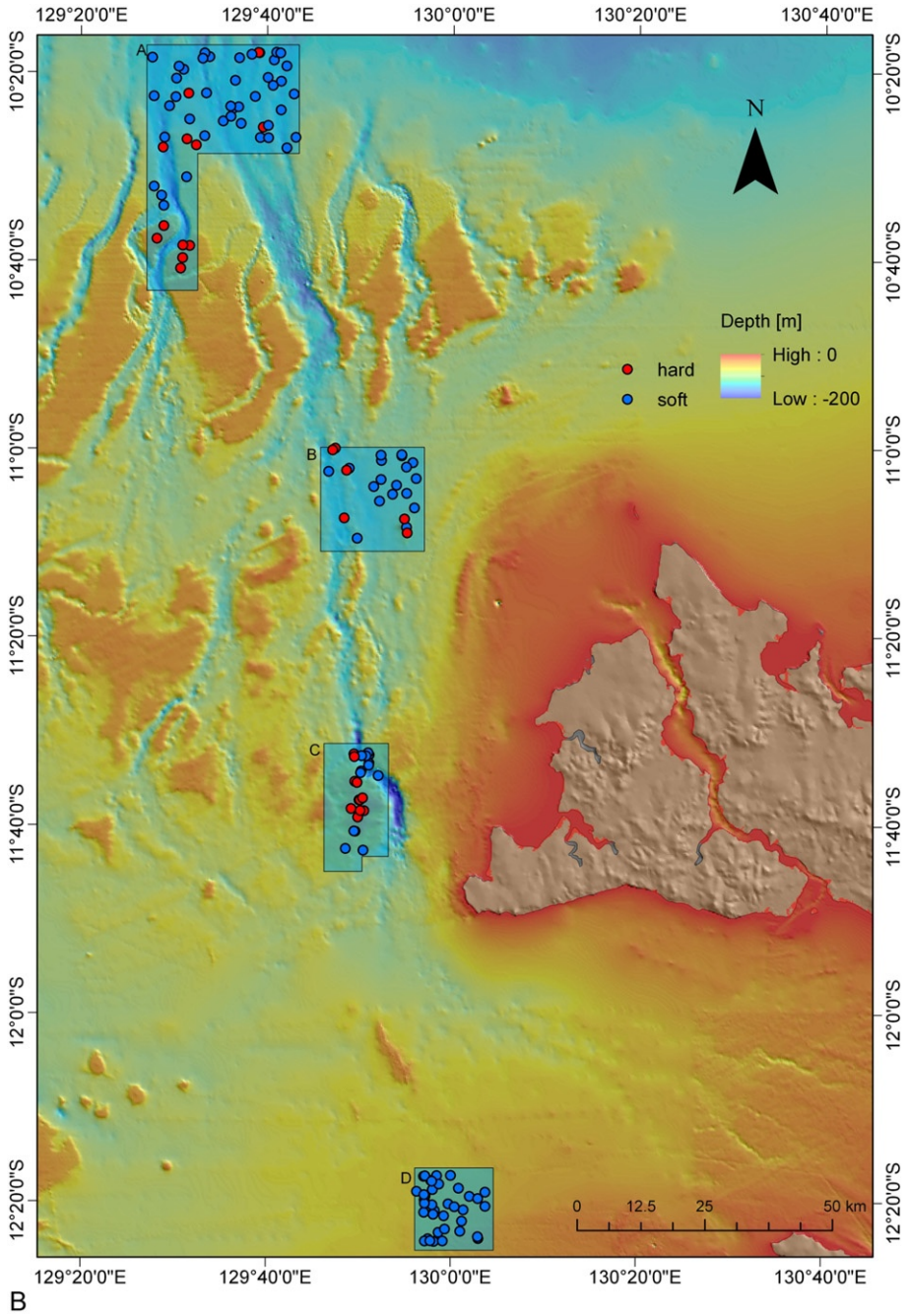


Figure 11.1—cont'd

(b) location of the four study areas (A-D) in the study region and video-derived classes of seabed hardness (hard and soft) overlaid with bathymetry at each sampling station; and

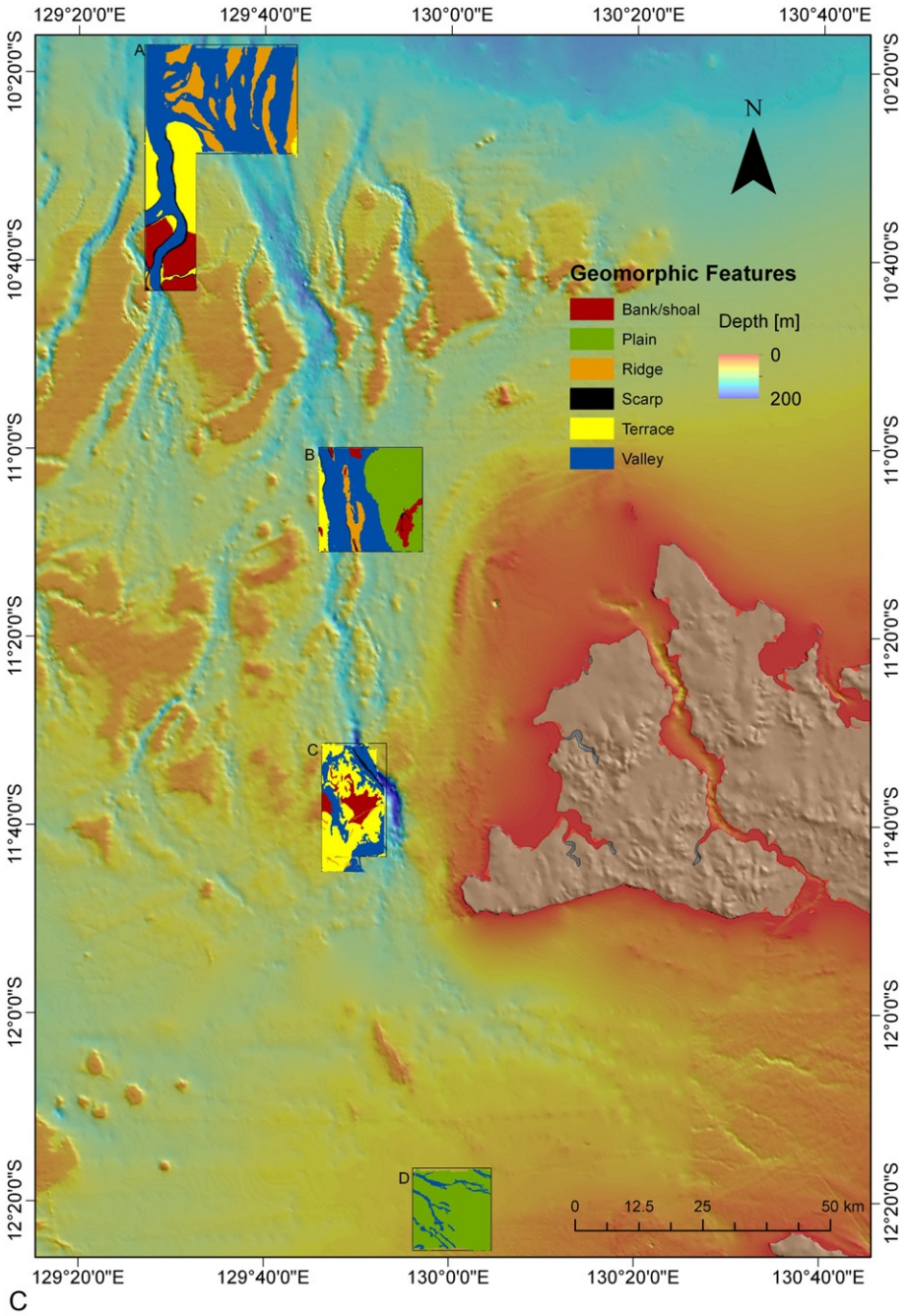


Figure 11.1—cont'd
(c) the geomorphic features of the four study areas.

substrate by primary (>50% of the area viewed) and secondary (>20% of the area viewed) percent cover using the protocol of Stein et al. (1992). The categorized substratum composition was then used to form two simple video classes: "hard" and "soft." Rock, boulder, cobble, and rubble identified in either primary or secondary substratum were merged to form "hard" class whereas gravel, sand, and mud found in both primary and secondary substratum were combined to form "soft" class. This process unavoidably introduced a mixed class in which the primary and secondary substratum types were different. In this study, this class was merged into the "hard" class on the basis of a preliminary analyses of backscatter data and the presence of certain biological entities (e.g., corals, rhodoliths, octocorals, or sponges) (Siwabessy et al., 2012).

Good agreement between the assigned data and the values of the probability of hard substrate (prock; for detailed information, see Section 11.2.3) were expected; however, some samples displayed extreme discrepancies between the assigned data and the prock. This is most likely due to a number of factors affecting the reliability of the assigned hardness data as discussed above and in Siwabessy et al. (2012). Therefore, the assigned data were reassessed by examining neighboring trends in backscatter and bathymetry data (Siwabessy et al., 2012). Consequently, 13 samples (of which 11 samples are "soft") were reassigned to different data; and for the remaining samples we did not have sufficient evidence to reassign them to different categories, so they remained unchanged. The resultant dataset was used to predict seabed hardness (Figure 11.1b).

11.2.3 Predictors

Following a preliminary analysis based on data availability and the relationship with seabed hardness as discussed above, 15 variables were selected and used in this study.

1. Bathymetry (bathy): water depth between the seabed and the sea surface
2. Seabed slope (slope)
3. Topographic relief (relief)
4. Surface area (surface): the ratio of the "true" surface area and its "planar" surface area (Jenness, 2004)
5. Topographic position index (tpi): a measure of where a location is in the overall landscape (e.g., slope position) (Weiss, 2001)
6. Planar curvature (planar.curv): the curvature of the surface perpendicular to the slope direction
7. Profile curvature (profile.curv): the curvature of the surface in the direction of the slope
8. Local Moran I (bathy.moran): a measure of local spatial autocorrelation in bathymetry,
9. Backscatter (bs): a diffused reflection of acoustic energy due to scattering process back to the direction from which it's been generated, measured as the ratio of the acoustic energy sent to a seabed to that bounced back from the seabed,

10. Homogeneity of backscatter (homogeneity): a measure of the closeness of the distribution of elements in the Gray-Level Co-occurrence Matrix (GLCM) to the GLCM diagonal,
11. Variance of backscatter (variance): a measure of the dispersion of the values around the mean within the GLCM,
12. Local Moran I of backscatter (bs.moran): a measure of local spatial autocorrelation in backscatter,
13. Prock: the probability of hard substrate,
14. Easting, and
15. Northing.

The first eight predictors are bathymetry and its derived variables. The next four predictors are backscatter and its derived variables. The processes of acquiring multibeam bathymetry and multibeam backscatter and their derived variables are detailed in Siwabessy et al. (2012).

The prock was estimated by comparing all angular response curves for incidence angles between 0° and 60° derived from the backscatter to the lowest angular backscatter response curve of the hard substrate for each study area using the Kolmogorov-Smirnov goodness of fit for each of the four study areas (Siwabessy et al., 2012). The video was used to locate the area of hard substrate. The inverse distance weighting method was used to produce a continuous layer of the prock for each of the study areas.

Finally, the last two predictors are the coordinates of sample locations.

All these predictors were available at each grid cell to a 10-m resolution in the four study areas for predicting seabed hardness.

Initial analysis and interpretation of the multibeam backscatter indicated that the shallow banks have the highest backscatter value (i.e., acoustically hard). Terraces also show strong acoustic returns. In contrast, sediment deposits in the valleys between the banks and terraces are associated with the weakest acoustic returns.

Seabed sediment properties (e.g., gravel, sand, or mud) often indicate the hardness of seabed substrate (De Falco et al., 2010; Ferrini and Flood, 2006; Goff et al., 2000; Hamilton and Parnum, 2011) and could be informative predictors. However, they were not included in this study, because they were available only at the point locations, whereas predictors with spatially continuous data were required.

11.3 Dataset Manipulation and Exploratory Analyses

This section introduces the dataset (Appendix A) and explores the relationships among the predictors and between the response variable and the predictors.

11.3.1 Features of the Dataset

```
> setwd("C:/temp/")
> hard <- read.table("hardness.csv", header = TRUE, sep = ",") # input dataset
> class(hard)
[1] "data.frame"
```

Since some missing data were recorded as "9999" during data procession, they were replaced with "NA."

```
> hard[hard==9999] <- NA
```

The following code displays the structure of the dataset, so all variables can be checked to ensure they were recorded correctly.

```
> str(hard)
'data.frame':  140 obs. of 17 variables:
$ region       : Factor w/ 4 levels "A","B","C","D": 3 3 3 3 3 3 3 3 3 3 ...
$ easting      : num  591804 590054 590455 591297 592836 ...
$ northing     : num  8712946 8718689 8718471 8720739 8723872 ...
$ prock        : num  1 0.119 0.042 0 0 0.2 0 1 1 0 ...
$ bathy        : num  22.1 32.8 38.6 48.2 92.3 ...
$ bs           : num  -10.5 -15.6 -15.8 -16.5 -23.4 ...
$ bathy.moran  : num  29.85 13.59 7.38 1.24 38.68 ...
$ planar.curv  : num  3.95 -1.54 0.34 0.85 -0.2 -0.56 -0.08 -2.2 -0.77 1.99 ...
$ profile.curv: num  -0.02 -0.02 0.01 -0.02 -0.02 0.06 0.11 0 -0.01 -0.02 ...
$ relief       : num  1.53 0.73 1.85 1.48 8.6 ...
$ slope        : num  0.39 0.53 0.85 1.24 3.01 2.88 7.2 0.26 0.25 2.35 ...
$ surface      : num  100 100 100 100 100 ...
$ tpi          : num  -0.16 -0.04 -0.03 -0.28 0.09 0.29 0.72 0.09 -0.11 -0.17 .
$ homogeneity  : num  0.85 0.9 0.87 0.71 0.7 0.77 0.62 0.91 0.86 0.84 ...
$ bs.moran     : num  83.96 12.71 9.42 1.05 24.3 ...
$ variance     : num  0.12 0.07 0.16 0.52 0.6 0.33 0.77 0.09 0.18 0.25 ...
$ hardness     : Factor w/ 2 levels "hard","soft": 1 1 1 2 2 1 1 1 1 2 ...
```

In total, the data of 15 predictors are available at 140 sample locations for developing models to predict seabed hardness. The study areas and hardness are factors and all the predictors are numerical.

11.3.2 Exploratory Data Analyses

Given that the seabed hardness was recorded in terms of "hard" and "soft," the data need to be transformed into a numerical variable in order to explore the relationships of hardness and the predictive variables. So the hardness variables were converted into a numeric variable of either 0 ("soft") or 1 ("hard") as below.

```
> hard$hardness2 <- 0
> hard$hardness2[hard$hardness == "hard"] <- 1
```

Now the dataset is ready for exploratory analyses. First, the correlations among the predictors and hardness were analyzed using Spearman's rank correlation method.

```
> hcor <- round(cor(hard[,-c(1, 17)], use = "na.or.complete", method = "spearman"), 2)
> library(gridExtra)
> grid.draw(tableGrob(hcor[, -16], show.csep = TRUE, show.rsep = TRUE, show.box + = TRUE,
separator = "grey"))
```

The results from the correlation analysis showed that seabed hardness is strongly correlated with prock, bathy, and bs; whereas it is weakly correlated with bathy.moran, relief, slope, surface, homogeneity, and bs.moran (Table 11.1). There are strong correlations ($\rho >= 0.8$) among some predictors, such as, easting and northing, prock and bathy, prock and bs, bathy and bs, relief and slop, relief and surface, slope and surface, homogeneity and variance (Table 11.1). The relationships among predictors are further illustrated in Figure 11.2 with the following code.

```
> pairs(hard[, -c(1, 17, 18)])
```

The results showed that some relationships are linear such as bathy and bs, relief and slope, relief and surface, slope and surface; but others are nonlinear, such as prock and bathy, prock and bs, homogeneity and variance. The relationships between seabed hardness and the predictors are depicted in Figure 11.3 by using the following code.

```
> h2 <- hard[,-c(1, 17)]
> par(mfrow = c(3,5))
> for (i in 1:15){
+     h3 <- subset(h2, abs(h2[,i]) >= 0)
+     plot(h3[, i], h3[,16], ylab = "Hardness", xlab = names(h3)[i])
+     lines(lowess(h3[,16]~h3[,i]), col = "blue")
+ }
```

High prock values are typically associated with the "hard" substrate (Figure 11.3). In contrast, low prock values are associated with the "soft" substrate. A similar pattern was observed for backscatter, whereas bathymetry showed an opposite pattern. These relationships are typically non-linear. These variables could potentially be good predictors of seabed hardness.

11.4 Application of RF for Predicting Seabed Hardness

This section briefly describes RF (Breiman, 2001; Breiman et al., 1984; Ho, 1995, 1998) and then introduces the application of RF via an R function, *randomForest* (Liaw and Wiener, 2002). RF is an ensemble method that combines many individual regression or classification trees in the following way: from the original sample many bootstrap samples and portions of predictors are drawn, and an unpruned regression or classification tree is fitted to each bootstrap sample using the sampled predictors. From the complete forest the status of the response variable is usually predicted as the average of the predictions of all trees for regression and as

Table 11.1 Spearman Correlation Coefficients (ρ) Among Seabed Hardness (Hardness2) and 15 Predictors ($n = 137$).

	easting	northing	prock	bathy	bs	bathy. moran	planar. curv	profile. curv	relief	slope	surface	tpi	homogeneity	bs.moran	variance
easting	1	−0.85	0.37	−0.52	0.43	0.06	0.05	−0.11	−0.35	−0.25	−0.38	−0.1	0.3	−0.05	−0.33
northing	−0.85	1	−0.53	0.7	−0.65	−0.22	−0.05	0.09	0.19	0.13	0.2	0.07	−0.41	−0.07	0.4
prock	0.37	−0.53	1	−0.78	0.8	0.45	−0.13	−0.05	−0.14	−0.11	−0.14	0.07	0.42	0.32	−0.37
bathy	−0.52	0.7	−0.78	1	−0.91	−0.19	−0.01	0.12	0.21	0.19	0.19	0.08	−0.59	−0.18	0.55
bs	0.43	−0.65	0.8	−0.91	1	0.24	−0.04	−0.03	−0.1	−0.08	−0.07	0.03	0.56	0.1	−0.52
bathy.moran	0.06	−0.22	0.45	−0.19	0.24	1	−0.15	0.04	0.16	0.12	0.14	0.15	−0.02	0.57	0.06
planar.curv	0.05	−0.05	−0.13	−0.01	−0.04	−0.15	1	−0.19	0.01	−0.04	−0.04	−0.51	−0.03	0.01	0.01
profile.curv	−0.11	0.09	−0.05	0.12	−0.03	0.04	−0.19	1	0.14	0.1	0.19	0.7	−0.05	−0.09	0.05
relief	−0.35	0.19	−0.14	0.21	−0.1	0.16	0.01	0.14	1	0.91	0.93	0.14	−0.32	0.19	0.37
slope	−0.25	0.13	−0.11	0.19	−0.08	0.12	−0.04	0.1	0.91	1	0.83	0.18	−0.24	0.09	0.28
surface	−0.38	0.2	−0.14	0.19	−0.07	0.14	−0.04	0.19	0.93	0.83	1	0.17	−0.33	0.16	0.38
tpi	−0.1	0.07	0.07	0.08	0.03	0.15	−0.51	0.7	0.14	0.18	0.17	1	0.01	−0.01	−0.02
homogeneity	0.3	−0.41	0.42	−0.59	0.56	−0.02	−0.03	−0.05	−0.32	−0.24	−0.33	0.01	1	−0.06	−0.92
bs.moran	−0.05	−0.07	0.32	−0.18	0.1	0.57	0.01	−0.09	0.19	0.09	0.16	−0.01	−0.06	1	0.14
variance	−0.33	0.4	−0.37	0.55	−0.52	0.06	0.01	0.05	0.37	0.28	0.38	−0.02	−0.92	0.14	1
hardness 2	−0.1	−0.07	0.62	−0.47	0.6	0.3	−0.13	−0.04	0.18	0.14	0.21	0.08	0.2	0.31	−0.13

Figure 11.2
The relationships among the 15 predictors.

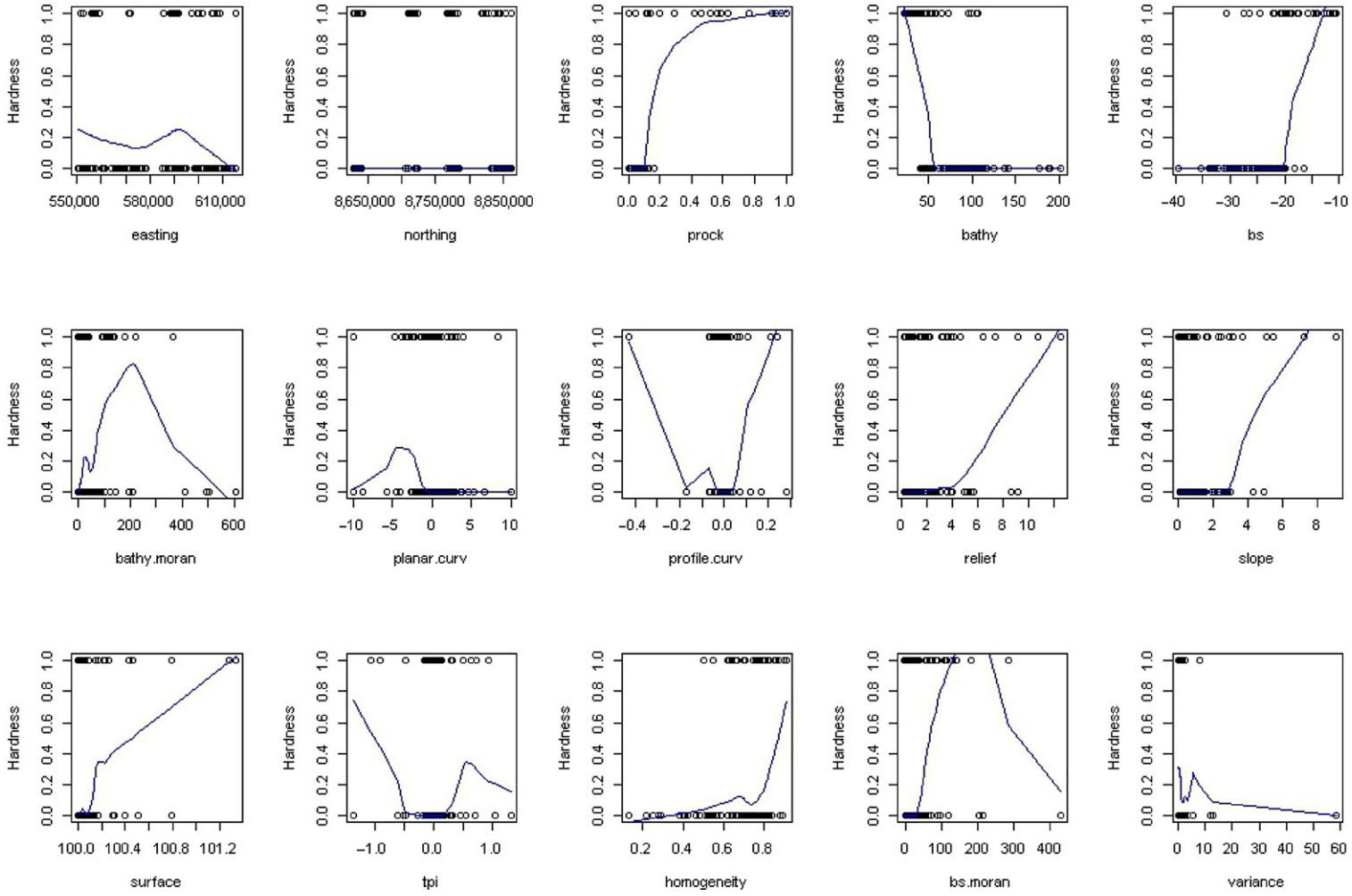


Figure 11.3

The relationships between seabed hardness (i.e., 0 = "soft" and 1 = "hard") and 15 predictors.

the classes with majority vote for classification (Breiman, 2001; Li et al., 2010). The R function, *randomForest*, is employed to develop a model to predict the spatial distribution of seabed hardness. In *randomForest*, there are a few parameters that need to be specified. One of these parameters is *mtry*, that is, the "number of variables randomly sampled as candidates at each split." It is often selected based on the result of *tuneRF*, whereby the *tuneRF* function searches for the optimal value of *mtry*.

Given that NA is not permitted in *tuneRF*, all NAs were removed in the dataset using the following code.

```
> hard2 <- na.omit(hard)
> dim(hard2)
[1] 137 18
```

This indicates that 3 of 140 samples contained NAs and were subsequently excluded. Consequently only 137 samples are left for developing a predictive model.

```
> library(randomForest)
> tuneRF(hard2[,-c(1,17,18)], hard2[,17], ntreeTry = 100)
```

The results of *tuneRF* indicated an *mtry* of 3 (Figure 11.4a). This value is equivalent to the default value for *mtry*. Hence in this study the default value for *mtry* was used. We also need to specify some other parameters such as *importance*, *ntree*, and *proximity* in the *randomForest* function to develop a predictive model.

```
> set.seed(123)
> rf.1 <- randomForest(hard2[,-c(1,17,18)], hard2[,17], data = hard2,
+ importance = TRUE, ntree = 500, proximity = TRUE)
> varImpPlot(rf.1)
```

The above code produced a figure illustrating the importance of predictors in the RF model (*rf.1*) (Figure 11.4b). It indicates that prock, bs, and bathy are the three most important predictors.

The contents of *rf.1* are displayed below.

```
> names(rf.1)
[1] "call"  "type"  "predicted"  "err.rate"  "confusion"  "votes"
[7] "oob.times"  "classes"  "importance"  "importanceSD"
[11] "localImportance"  "proximity"  "ntree"  "mtry"  "forest"  "y"
[17]"test"    "inbag"
```

Of these outputs, *predicted* is sometimes taken as the predictions produced by the model. This is not true because the values of *predicted* are actually the predicted values of the input data based on the out-of-bag (OOB) samples, which is clearly stated in the help file of the *randomForest* package. The *err.rate* and *confusion* are also based on the OOB samples.

To acquire the correct predictions and confusion matrix, we used the following code.

```
> dev.rf1 <- predict(rf.1, hard2)
> grid.table(table(hard2[,17], dev.rf1))
```

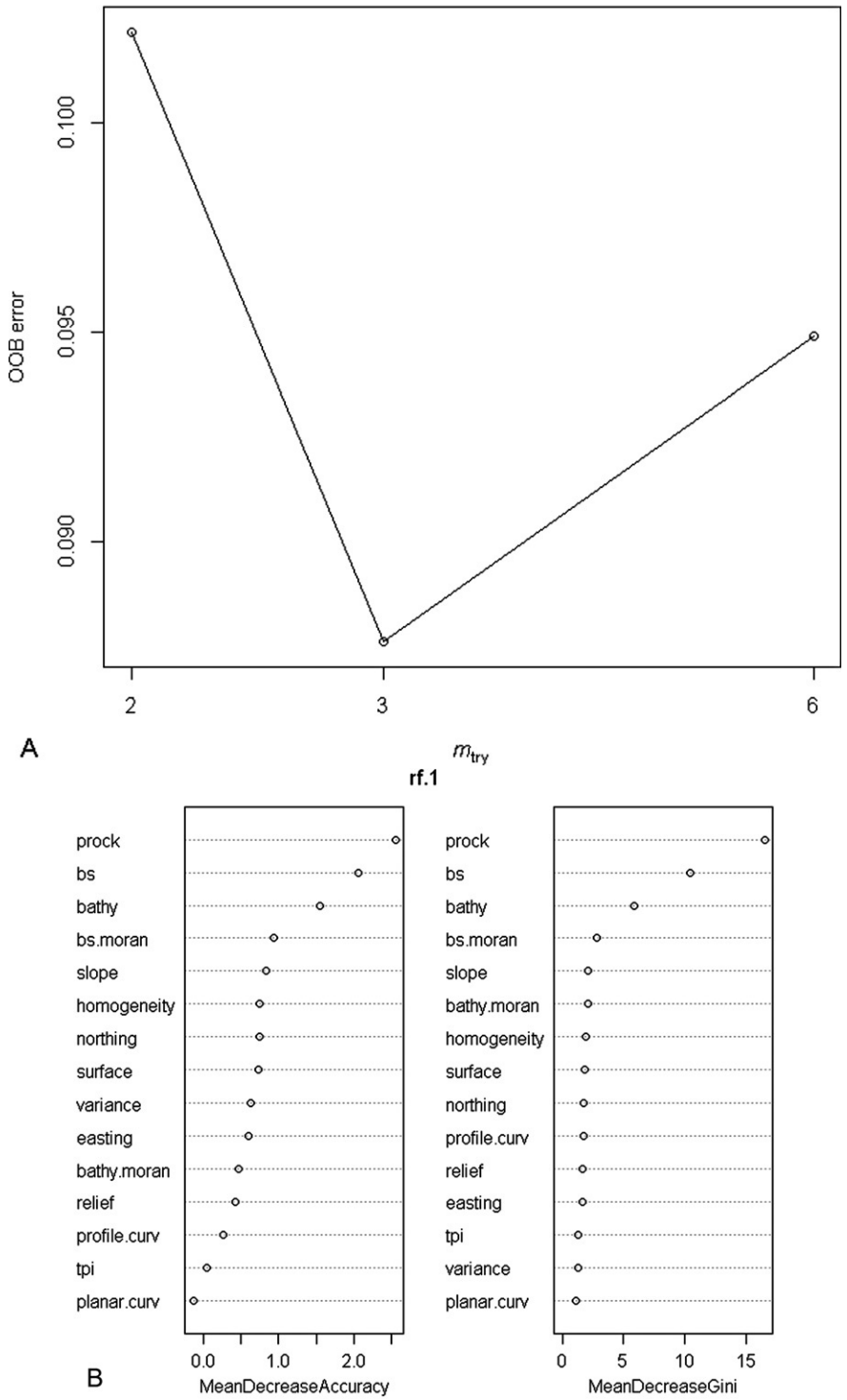


Figure 11.4

(a) A number of an *mtry* and (b) the importance of the predictors for modeling the hardness using the RF method.

	Hard	Soft
Hard	38	0
Soft	0	99

The resultant confusion matrix (above) shows that the predictive model correctly classified all samples, that is, it was a perfect fit to the data. However, this does not necessarily mean the model can make perfect predictions.

11.5 Model Validation Using rfcv

To validate the predictive ability of the model, we used 10-fold cross-validation (Hastie et al., 2009). To test the random error associated with each 10-fold cross validation, the cross validation procedure were repeated 100 times. The final results were based on the average of 100 iterations of the cross-validation.

A cross-validation function (*rfcv*) is provided with the *randomForest* packages, and it only provides one statistic, *error.cv* that is the corresponding vector of error rates for classification or mean square errors (MSEs) for regression at each step (Svetnik et al., 2004).

This *rfcv* also allows the removal of the least important variable(s) at each step, generates a new model, and thus performs automated variable selections. It validates the accuracy of the new model until a desired number of variables is reached. In this study, we allow *rfcv* to remove the least important variable at each step by providing a *non.log* to *scale* and by setting $step = -1$ as below.

```
> result1 <- replicate(100, rfcv(hard2[,-c(1,17,18)], hard2[,17], scale =
+ "non.log", cv.fold = 10, step = -1), simplify = FALSE)
```

The *result1* contains a list of 100 sets of sublists that contain the following information as detailed in the help file of *rfcv* (Svetnik et al., 2004):

n.var vector of number of variables used at each step

error.cv corresponding vector of error rates or MSEs at each step

predicted list of *n.var* components, each containing the predicted values from the cross-validation

The following codes were used to extract and plot the 10-fold cross-validation error against the number of variables used at each step (Figure 11.5). The application of *rfcv* resulted in 15 models, leaving the last model with only one predictor.

```
> error.cv <- sapply(result1, "[[", "error.cv")
> matplot(result1[[1]]$n.var, cbind(rowMeans(error.cv), error.cv), type = "l",
+ lwd = c(2, rep(1, ncol(error.cv))), col = c(2, rep(1, ncol(error.cv))), lty = + 1,
xlab = "Number of variables", ylab = "CV Error")
```

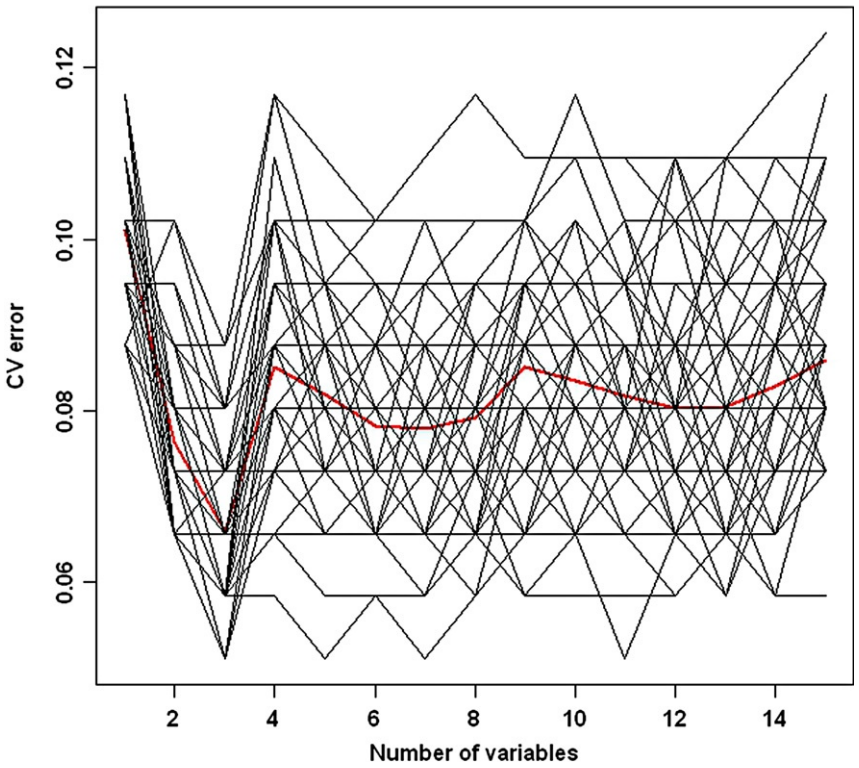


Figure 11.5

The number of variables used at each step and the10-fold cross-validation error based on 100 iterations, with the results from each of 100 iterations (black line) and the overall average of the 100 iterations (red line).

We further plotted the averaged correct classification rate (*ccr*) based on 100 iterations of the 10-fold cross-validation against the number of variables used at each step in the predictive models (Figure 11.6).

```
> plot(result1[[1]]$n.var, (1-rowMeans(error.cv))*100, type = "l", lwd = 2, col += 2,
lty = 1, xlab = "Number of variables", ylab = "Correct classification
+ rate (%)")
```

It is apparent that the most accurate model was developed when 3 variables were used, while a few suboptimal models were also developed when 7, 12 and 13 variables were used (Figure 11.6). The most accurate result was achieved when the three most important predictors were used (Figure 11.5), with an error rate of 0.0659 that is equivalent to a *ccr* of 93.41% (Figure 11.6). This suggests that only the three most important predictors should be used in the predictive model.

However, this result does not tell us which predictors were used and which predictors should be used at each step. In fact, because of the randomness associated with the importance of

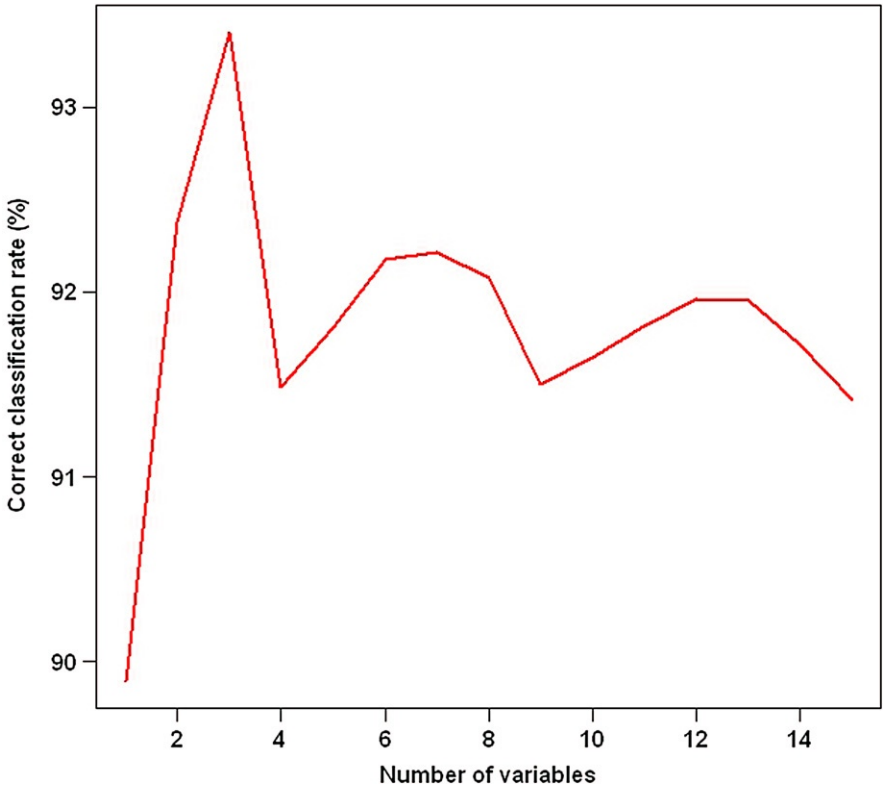


Figure 11.6
The number of variables used in the predictive models and averaged *ccr* based on 100 iterations of the 10-fold cross-validation.

predictors generated by the RF algorithm, the least important variable(s) may change with individual iterations. Hence the predictors used at each of the 100 iterations were not always identical to other iterations for a given number of variables. Consequently, the results (Figures 11.5 and 11.6) reflect the effects of two factors. One is the difference in datasets generated by 10-fold cross-validation among iterations, and the other is the changes in predictors among iterations.

11.6 Optimal Predictive Model

To identify the most accurate predictive model (i.e., the optimal predictive model), we need to know which set of predictors should be used in the model. To achieve this, we modified the *rfcv* into *rf.cv* (Appendix B) that validates one model with fixed predictors for all iterations for a given number of variables. This function will only allow variation in datasets generated by cross-validation. Given that the response variable is categorical, the *ccr* (Fielding and Bell, 1997; Li and Hilbert, 2008) and *kappa* (Cohen, 1960) were used to measure the accuracy of the

predictive model in *rf.cv*. This function produces two statistics: *kappa*, and *ccr*. This function is only for categorical response variables and was applied to the RF model above.

```
> ccr.cv.1 <- NULL; kappa.cv.1 <- NULL; mccr.cv.1 <- NULL; mkappa.cv.1 <- NULL
> for (i in 1:100){
+ rfcv.1 <- rf.cv(hard2[,-c(1,17,18)], hard2[,17], cv.fold = 10, ntree = 500)
+ ccr.cv.1[i] <- rfcv.1$ccr.cv; kappa.cv.1[i] <- rfcv.1$kappa.cv
+ mccr.cv.1[i] <- mean(ccr.cv.1); mkappa.cv.1[i] <- mean(kappa.cv.1)
+ }
```

The above loop can be replaced with the *replicate* function as for *rfcv*, and the accuracy measures need to be extracted afterwards. We repeated this validation process 100 times to check the variability in these two measures among iterations. We also calculated the cumulative averages of these two measures to show the contribution or influence of individual iterations and to stabilize the overall accuracy. The results of this procedure were extracted and plotted, as follows:

```
> x <- c(1:100)
> par(mfrow=c(2,1), font.axis = 2, font.lab = 2)
> plot(ccr.cv.1 ~ x, xlab = "Iteration", ylab = "Correct classification rate")
> points(mccr.cv.1 ~ x, col = 2)
> plot(kappa.cv.1 ~ x, xlab = "Iteration", ylab = "Kappa")
> points(kappa.cv.1 ~ x, col = 2)
```

Both *ccr* and *kappa* displayed considerable dynamics among different iterations (Figure 11.7). The *ccr* changed from 89.05% to 93.43%; while *kappa* varied from 0.7104 to 0.8263. After 100 iterations, their averages stabilised at 91.30% and 0.7689, respectively.

To identify the most accurate predictive model, we adopted the same principle as used in *rfcv*, that is, removing the least important variables based on the importance of predictors, but keeping the predictors unchanged among iterations. Since tpi and planar.curv were the two least important variables (Figure 11.4b), we excluded them and re-ran the code above. On the basis of the importance of the predictors of each model, we sequentially excluded: (1) profile.curv and variance; (2) homogeneity and easting; (3) bathy.moran and northing; (4) bs.moran and relief; (5) slope; (6) bathy; and (7) surface in the following model. Because bathymetry and slope may play an important role in predicting seabed hardness (Li et al., 2010, 2012b; and Table 11.1), we first added bathy back and then added slope back to the model. The modeling results are summarised in Table 11.2.

Models with four or more predictors (i.e., models 1-7, and 12) produced perfect fits to the data (Table 11.2). The models with three or less predictors (i.e., models 8-11) made good fits to the data, with a high predictive accuracy.

The effects of various predictor sets on the predictive accuracy are illustrated in Figure 11.8. The predictive accuracies of these models based on 100 iterations of 10-fold cross validation indicated that model 8 is the most accurate model (Figure 11.8). There is one suboptimal

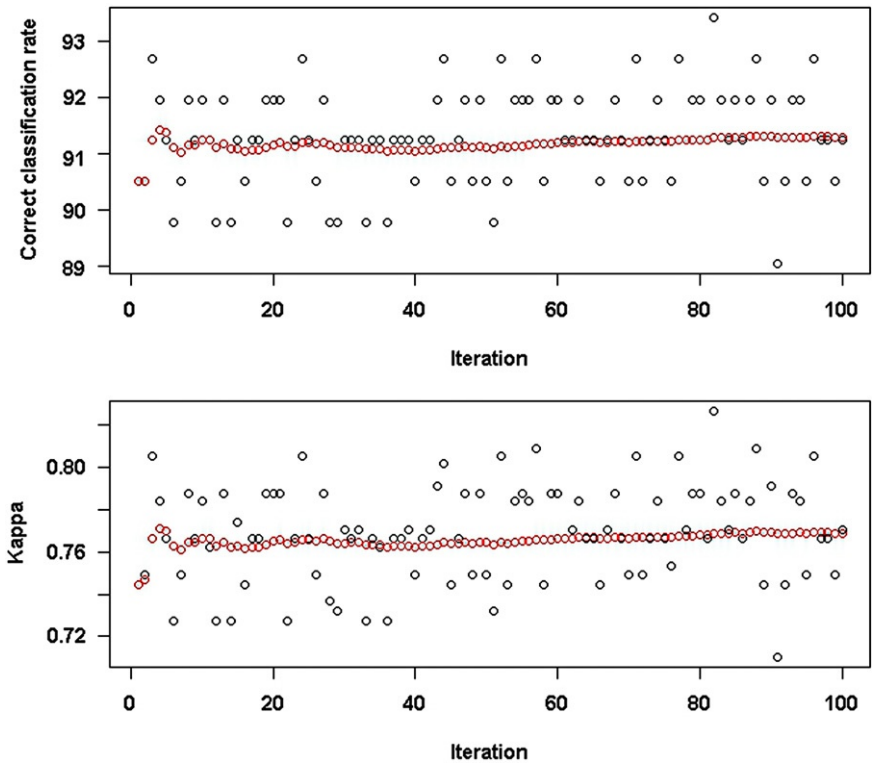


Figure 11.7

The predictive accuracies of RF using all 15 predictive variables based on the 10-fold cross-validation over 100 iterations and the average of the cumulative accuracy based on 1-100 iterations.

model (model 6) in terms of *kappa*. However, replacing surface in model 8 with bathy resulted in a new model 11 that becomes the most accurate model, with a mean *ccr* of 93.51% and a mean *kappa* of 0.8307. Further adding slope to model 11 resulted in a model (model 12) that is, however, less accurate.

On the basis of the above analyses, we can conclude that model 11 is the most accurate model and its accuracy also matches the results in Figure 11.6 in terms of *ccr*. The dynamics of the predictive accuracies with iterations of model 11 are shown in Figure 11.9, suggesting that the averaged accuracies stabilised after 60 iterations.

To explore how each of the three predictors contributed to the most accurate predictive model, we used partial dependence plots that showed the marginal effect of a variable on the class probability (Figure 11.10).

```
> par(mfrow = c(3,1), font.axis = 2, font.lab = 2)
> partialPlot(rf.1, hard2, prock)
> partialPlot(rf.1, hard2, bs)
> partialPlot(rf.1, hard2, bathy)
```

Table 11.2 Brief Summary of RF Modeling Process with Different Predictors

Model	Modeling Process	Predictors	No. of Variables	Model.fit
1	All 15 predictors	All 15 variables	15	100.00
2	Exclude planar.curv and tpi from model 1	easting, northing, prock, bathy, bs, bathy.moran, profile curv, relief, slope, surface, homogeneity, bs.moran, variance	13	100.00
3	Exclude profile.curv and variance from model 2	easting, northing, prock, bathy, bs, bathy.moran, relief, slope, surface, homogeneity, bs.moran	11	100.00
4	Exclude homogeneity and easting from model 3	northing, prock, bathy, bs, bathy.moran, relief, slope, surface, bs.moran	9	100.00
5	Exclude bathy.moran and northing from model 4	prock, bathy, bs, relief, slope, surface, bs. moran	7	100.00
6	Exclude bs.moran and relief from model 5	Prock, bathy, bs, slope, surface	5	100.00
7	Exclude slope from model 6	prock, bathy, bs, surface	4	100.00
8	Exclude bathy from model 7	prock, bs, surface	3	97.81
9	Exclude surface from model 8	prock, bs	2	96.35
10	Exclude bs from model 9	prock	1	95.62
11	Add bathy back to model 9	prock, bathy, bs	3	96.35
12	Add slope back to model 10	prock, bathy, bs, slope	4	100.00

Model.fit is the predictive accuracy (*ccr*) of each RF model developed.

The partial plots bear some similarity to those of prock, bs, and bathy in Figure 11.3. It showed that as prock increases, the probability of a sample being classified as "hard" increases. The relationship between the probability and bs is interesting and showed that when bs is below −32.5, the probability of "hard" is low, when bs is above −14, the probability of "hard" is high, and for bs between −32.5 and −14, the shape of the trend is nonlinear, with an increasing trend in the probability of "hard." For bathy, the probability of "hard" is high at less than 30 m, the probability of "soft" is high between 80-90 m, and for other depths, the relationship is not clear.

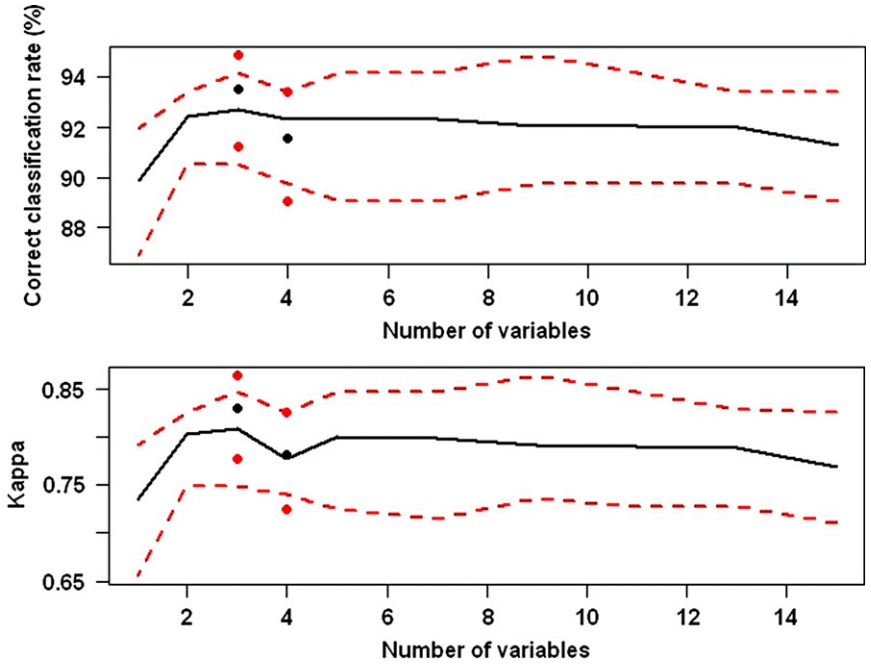


Figure 11.8

Summary statistics of RF models with different predictor sets based on the averages over 100 iterations of 10-fold cross validation. The solid black line is the mean accuracy for models 1-10 (see Table 11.2), while the dashed red lines are the minimum and maximum of the accuracies of the 100 iterations. The mean (black), minimum (red), and maximum (red) accuracy of models 11 and 12 based on the averages of over 100 iterations of 10-fold cross validation are represented as dot points.

11.7 Application of the Optimal Predictive Model

The application of RF to predict seabed hardness involves two essential steps. First, a predictive model of seabed hardness is developed based on a set of predictors as above. Second, the most accurate predictive model is used to predict seabed hardness at each 10 m grid cell in the study areas where the values of the predictors are known.

Here we predicted seabed hardness of a portion of area A (A1) using the most accurate predictive model developed in the previous section to demonstrate the application of the RF predictive model.

```
> ra <- read.csv("area_a1.csv", sep = ",", header = FALSE)
> dim(ra);
[1] 4653653      15
> pred.a1 <- ra[, c(1,2)]
> pred.a1$hardness <- predict(rf.1, ra)
> write.table(pred.a1, "hardness.pred.a1_3vars.csv", sep = ",", row.names =
+ FALSE)
```

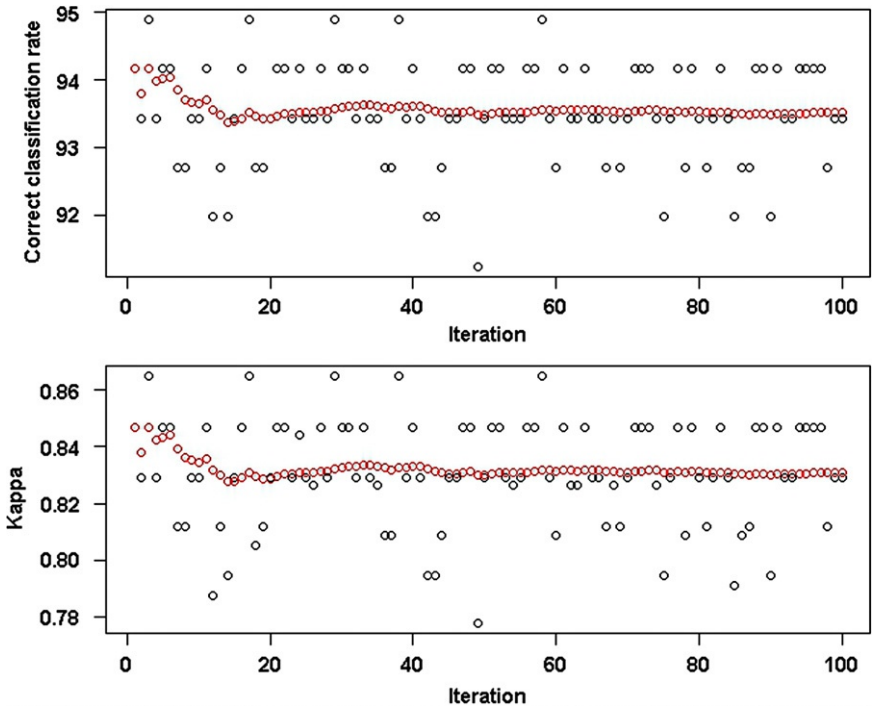


Figure 11.9

The predictive accuracies of the most accurate RF model using three predictive variables (i.e., prock, bathy, and bs) based on the 10-fold cross-validation over 100 iterations (black circles) and the average of the cumulative accuracy based on 1-100 iterations (red circles).

The predicted results were further processed in ArcGIS to generate the prediction map by overlaying the predicted values with the bathymetry layer (Figure 11.11a). The three predictors are also illustrated for A1 (Figure 11.11b–d) (Siwabessy et al., 2012) to show how each predictor contributed to the predictions. Because of the size of the dataset and its high resolution (10 m), we are not able to provide the file with this study. However the above code shows how to generate the final predictions.

Obvious trends were observed between the hardness prediction (Figure 11.11a), prock (Figure 11.11b), bs (Figure 11.11c) as well as bathy (Figure 11.11d). The hard substrates are mostly found in the shallow water and are mostly associated with the highest backscatter values, and geomorphic features such as banks and slope (Figure 11.1c). In comparison, valleys in relatively deeper water are mostly associated with the lowest backscatter values and fall within the predicted soft substrate category (Figure 11.1c).

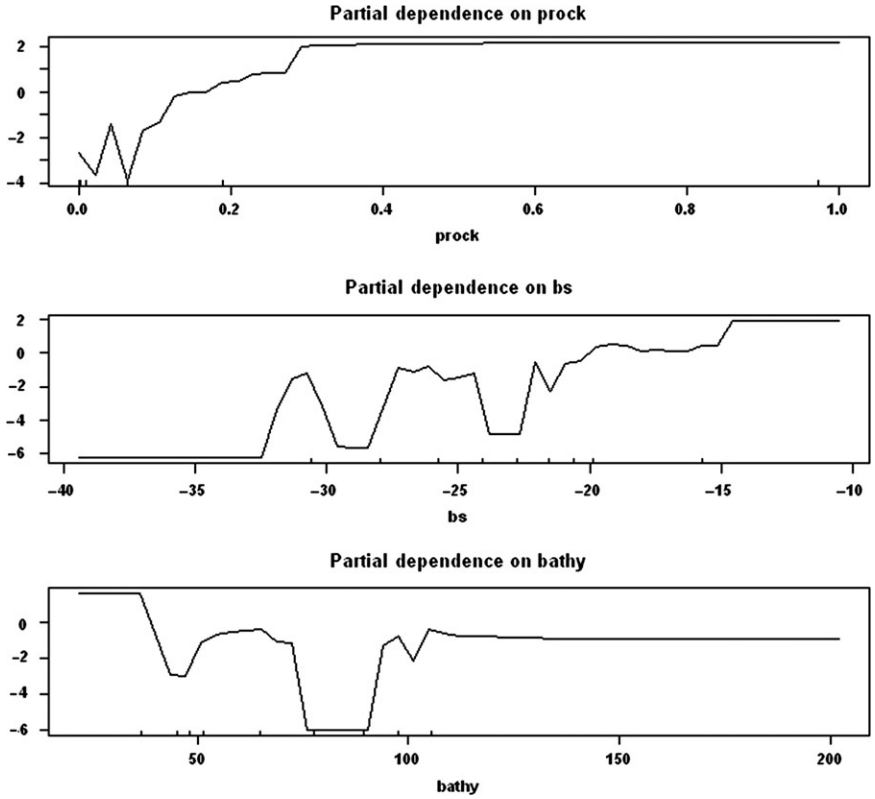


Figure 11.10

Partial dependence plots for RF classifications for seabed hardness and three predictors (i.e., prock, bs, and bathy), showing the marginal effect of the predictors on the class probability of hardness in model 11.

11.8 Discussion and Conclusions

11.8.1 Selection of Relevant Predictors and the Consequences of Missing the Most Important Predictors

The probability of hard substrate (prock) has been shown to be the most important predictor in the final predictive model. By using prock alone, RF can achieve accuracy as high as 89.85% (ranging from 86.86% to 91.97%) in terms of *ccr* and 0.7344 (ranging from 0.6555 to 0.7912) in terms of *kappa*. This is because it is derived from the full angular backscatter response which is a direct inherent property of the seabed. Additionally, this is supported by the success of the classification-based approach utilising the full angular backscatter response alone (Siwabessy et al., 2012). However, it requires an interpolation to generate a continuous

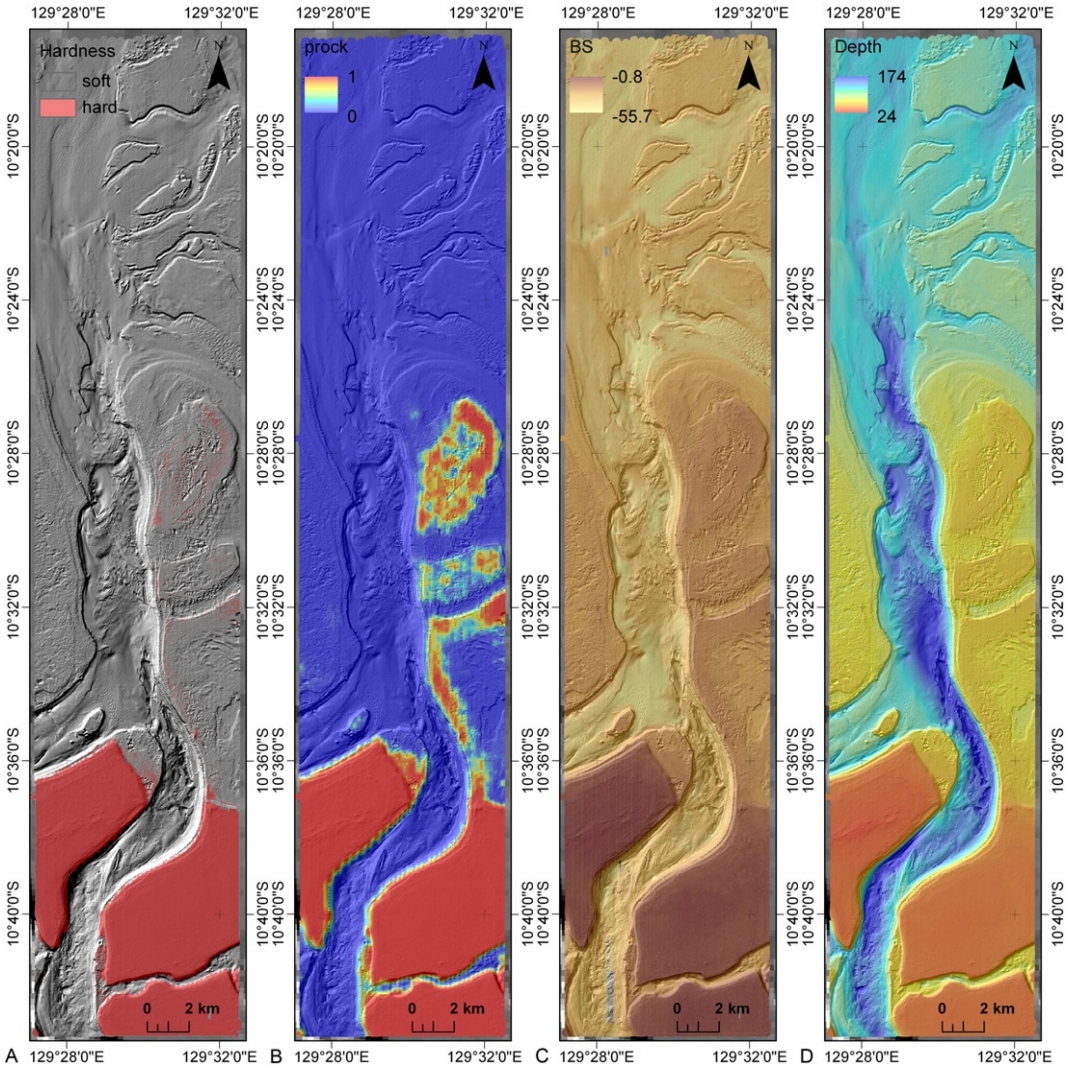


Figure 11.11

(a) Predicted seabed hardness, (b) prock, (c) backscatter in dB unit, and (d) bathymetry in meter for a section of area A (A1).

layer of prock to be used in developing the predictive model. One disadvantage of this is that its accuracy is dependent on the interpolation method (Li and Heap, 2011).

The importance of multibeam backscatter as revealed by RF largely supports deriving the inference of seabed hardness from the backscatter data in previous studies (Anderson et al., 2008a; Basu and Saxena, 1999; Kloser et al., 2010). The backscatter is the second most important predictor of seabed hardness. It contributed to the improvement in the predictive accuracy of RF model.

Bathymetry is also one of the most important predictors of seabed hardness. It further improved the predictive accuracy of the RF model. In addition, due to the availability of bathymetry data throughout the Australian Exclusive Economic Zone, it offers a fast and inexpensive assessment of seabed hardness.

The predictive accuracy of RF models with different sets of predictors demonstrated that reliance upon variable importance can lead to a suboptimal model (model 8) (Table 11.2, Figure 11.8). The most accurate predictive model may be unidentified but a suboptimal model may be developed and taken as the most accurate predictive model if one of the most important predictors is not included but a less important predictor is included. Therefore, to identify the most accurate predictive model we need to rely on both previous knowledge and model selection technique.

11.8.2 Issues with Searching for the Most Accurate Predictive Model Using RF

The essence of RF is its double randomisation: random trees and randomly sampled predictors at each split of individual trees. Such randomisation is controlled by its two parameters, *ntree* and *mtry*. The function of *rfcv* is also with a spirit of randomness, again double randomisation, in addition to the randomness associated with RF: random samples controlled by *cv.fold* and random predictors for each number of predictor(s) controlled via *step* based on the importance produced via RF. The partial dependence plot (*partialPlot*) is also associated with the nature of randomness inherited from RF. These developments exploited randomness to a great extent and often produced highly accurate predictive models, as we discussed above/below.

It should be noted that the results of *tuneRF*, *randomForest*, *rfcv* and *partialPlot* are often variable due to the randomness associated with the method. For example, *tuneRF* often generates inconsistent values of *mtry*. This could be overcome by running this function several times and then assigning the value with the highest frequency of occurrence to *mtry*. However, we may not need to be concerned because the choice of the *mtry* has been proven to have little impact on the predictive accuracy (Cutler et al., 2007; Diaz-Uriarte and de Andres, 2006; Li et al., 2012a,b; Liaw and Wiener, 2002; Okun and Priisalu, 2007).

The importance of predictors in the RF model is not very stable. The order of the importance is usually inconsistent among different iterations (Figure 11.4b). This is due to the randomness associated with RF. Individual iterations often produce different results; but the order of the most important predictors usually remains relatively stable. The order generally depends on the correlations of predictors with the response variable. The higher the correlation coefficient of a predictor with the response variable is, the more important the predictor. However, the least important variable(s) often change with iterations, as mentioned above, and different least important variable(s) may result, consequently leading to different models if *rfcv* is repeatedly applied as we observed in this study.

This study clearly demonstrated that there is considerable variation among different iterations for models at each "number of variables," suggesting that the accuracy of RF is affected by the input datasets generated for each 10-fold cross-validation (Figure 11.5). The results indicated that we cannot obtain reliable cross-validation results based on a single 10-fold cross-validation. Such variation may also be a result of the possible different predictors used for models at each "number of variables" in *rfcv*. In this study, we repeated the validation process 100 times to check the variability in these two measures among iterations, and stable results were achieved after 60 iterations. The choice of iteration number is often arbitrary, but we recommend 100 times should be as a minimum. In fact, the larger the iteration times, the more stable the accuracy measures will be, with the disadvantage of longer computational time.

As for cross-validation, we can achieve more consistent results by using *rf.cv* and by increasing the number of iteration as discussed above. The *rf.cv* also incorporates randomness but with the removal of the randomness of *rfcv* associated with the number of predictors. Additionally, more consistent results of RF could be achieved by increasing the number of *ntree*.

The development of the most accurate model and a few suboptimal models implies that we may miss the opportunity to find the most accurate model if we had taken one of these suboptimal models as the most accurate model (Figure 11.6). By comparing Figures 11.6 and 11.8, the randomness is further evidenced because there are two suboptimal models in Figure 11.6 but only one in Figure 11.8. So we suggest running all possible models with at least 100 iterations to identify the most accurate model, unless you have evidence to show that the model developed has already met the accuracy target. This approach has identified key predictors required to predict the seabed hardness. This study further demonstrated that model selection is an essential step for identifying the most accurate predictive model (Li et al., 2011b, 2012b).

11.8.3 Predictive Accuracy of RF and Prediction Maps of Seabed Hardness

All the models developed produced a good or even perfect fit to the data (Table 11.2). Consequently the predictive accuracy is also very high (Figure 11.8). This suggests that (1) the data of the response variable is of high quality and no outliers existed and (2) the predictors we selected are informative and useful. However, models with highest fit (i.e., models 1-7, and 12) are not necessarily the most accurate predictive model (Table 11.2, Figure 11.8).

According to Fielding and Bell (1997), the agreement between the predicted values and the tested values is excellent if *kappa* is more than 0.75. For the most accurate predictive model in this study, *kappa* ranged from 0.7779 to 0.8649, with a mean *kappa* of 0.8307, and the *ccr* varied from 91.24% to 94.89%, with a mean *ccr* over 93.51%. Hence the accuracy of the model developed for predicting the seabed hardness is exceptionally high. Therefore, we can conclude that:

- sample size used in this study was adequate for modeling the seabed hardness;
- the hardness was properly classified based on underwater video footage;
- informative predictors were identified and incorporated;
- an appropriate modeling method was employed;
- a robust predictive model was developed;
- seabed hardness is predictable;
- seabed hardness can be predicted with high accuracy;
- RF is an effective modeling method with high predictive ability; and
- RF can be applied to "small p and large n" problems, with the number of predictors as low as only one.

RF accurately distinguished hard from soft substrates in the four geomorphically complex areas in eastern Joseph Bonaparte Gulf. Unlike the clustering technique (Siwabessy et al., 2012), RF enables generation of a continuous layer of the seabed hardness. The predicted maps reflect the influence of various geomorphic features such as banks, terraces, valleys, ridges, scraps, and plains. More importantly, the model could potentially be applied to areas where multibeam acoustic data exist, especially to large areas where predictions of seabed types are needed for marine planning and management. Consequently, more reliable predictions of Australia's seabed biodiversity could be generated.

Additionally, as banks are the shallowest geomorphic feature, they were often found with the highest diversity of epibenthic assemblages such as hard corals, sponges, and octocorals often on rocky outcrops (Anderson et al., 2011; Przeslawski et al., 2011). This further supports our predicted hardness distribution, as the epibenthic organisms listed above require a hard substratum to grow. In comparison, valleys fell within the predicted soft substrate category (Figure 11.1c), which is also consistent with that valleys supported significantly lower proportions of epibenthic organisms than banks (Anderson et al., 2011; Przeslawski et al., 2011).

This study further affirms the superior performance of RF in marine environmental sciences (Li et al., 2010, 2011a–c, 2012b). The excellent performance of RF has been attributed to a number of features associated with RF including the ability to model a nonlinear relationship as discussed in the previous studies (Li et al., 2011a,b). Since RF produced continuous predictions with high predictive accuracy, it is recommended for future work if continuous predictions of seabed hardness are required and if the information of relevant predictors is available.

11.8.4 Limitations

To verify and validate the acoustic prediction, we relied on video data and colocated seabed samples. The limitation of these approaches is that they provide little to no information about the subsurface. It is therefore difficult to rely solely on the video footage to validate hard

grounds that is underneath a thin veneer of soft sediments. In addition, the information used for validation relied on a video-derived classification which can be somehow subjective if only one observer was involved in the classification process.

Angular backscatter response curves have become popular in various fields for seabed classification and characterization as they preserve seabed properties in high angular resolution in comparison to what backscatter mosaics can offer. However, they are lacking in spatial resolution due to the nature of their construction because they are generated by taking the average of a stack of backscatter intensity as a function of incidence angles within a sliding window. In addition, this is based on the assumption that the seabed is homogeneous across half of the swath which is not always the case.

Acknowledgments

This study was undertaken as an activity of the Seabed Mapping and Coastal Information Section of the Coastal, Marine and Climate Change Group, Environmental Geoscience Division, Geoscience Australia (GA). We thank the Master and crew of the RV Solander and scientific staff at the Australian Institute of Marine Science (AIMS) for their support in conducting the survey, Marcus Stowar (AIMS) for towed-video and still photography acquisition, and the Field and Engineering Support staff at GA. We also thank Augusto Sanabria and Wenping Jiang (GA) for their valuable reviews of the initial manuscript. Tanya Whiteway carefully proofread and also commented on the chapter. This study is published with the permission of the Chief Executive Officer, Geoscience Australia.

Appendix AA Dataset of Seabed Hardness and 15 Predictors

See http://www.rdatamining.com/books/dmar/.

Appendix BA R Function, rf.cv, Shows the Cross-Validated Prediction Performance of a Predictive Model

```
rf.cv <- function (trainx, trainy, cv.fold = 10,
    mtry = function(p) max(1, floor(sqrt(p))), ntree=500, ...) {
    classRF <- is.factor(trainy)
    n <- nrow(trainx)
    p <- ncol(trainx)
    cv.pred <- NULL
    if (classRF) {
        f <- trainy
    }     else {
        stop ("This function is only for categorical response variable")
    }
    nlvl <- table(f)
```

```
    idx <- numeric(n)
    for (i in 1:length(nlvl)) {
        idx[which(f == levels(f)[i])] <- sample(rep(1:cv.fold,
            length = nlvl[i]))
    }
    for (i in 1:cv.fold) {
        all.rf <- randomForest(trainx[idx != i, , drop = FALSE],
            trainy[idx != i], trainx[idx == i, , drop = FALSE],
            trainy[idx == i], mtry = mtry(p), ntree=ntree)
        cv.pred[idx == i] <- all.rf$test$predicted
    }
    require(psy)
        data1<-as.data.frame(cbind(cv.pred, trainy))
        kappa.cv<-ckappa(data1)$kappa
        ccr.cv<-sum(diag(table(data1)))/sum(table(data1))*100
    list(kappa.cv = kappa.cv, ccr.cv = ccr.cv, predicted = cv.pred)
}
```

References

Anderson, J.T., Van Holliday, D., Kloser, R., Reid, D.G., Simard, Y., 2008a. Acoustic seabed classification: current practice and future directions. ICES J. Mar. Sci. 65, 1004–1011.

Anderson, T.J., Cochrane, G.R., Roberts, D.A., Chezar, H., Hatcher, G., 2008b. A rapid method to characterize seabed habitats and associated macro-organisms. In: Todd, B.J., Greene, H.G. (Eds.), Mapping the Seafloor for Habitat Characterization. Geological Association of Canada c/o Department of Earth Sciences, Memorial University of Newfoundland, St. John's, Newfoundland & Labrador, Canada A1B 3X5, pp. 71–79.

Anderson T.J., Nichol S., Radke L., Heap A.D., Battershill C., Hughes M., Siwabessy P.J., Barrie V., Alvarez de Glasby B., Tran M., Daniell J., and Party S., 2011. Seabed Environments of the Eastern Joseph Bonaparte Gulf, Northern Australia: GA0325/Sol5117 - Post-Survey Report. Geoscience Australia, Canberra. Record 2011/08, 59pp.

Basu, A., Saxena, N.K., 1999. A review of shallow-water mapping systems. Mar. Geod. 22, 249–257.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and regression trees. Wadsworth, Belmont.

Buhl-Mortensen, L., Buhl-Mortensen, P., Dolan, M.F.J., Danmheim, J., Bellec, V., Holte, B., 2012. Habitat complexity and bottom fauna composition at different scales on the continental shelf and slope of northern Norway. Hydrobiologia. 685, 191–219.

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educ. Psychol. Meas. 20, 37–46.

Cutler, D.R., Edwards, T.C.J., Beard, K.H., Cutler, A., Hess, K.T., Gibson, J., Lawler, J.J., 2007. Random forests for classification in ecology. Ecography. 88, 2783–2792.

De Falco, D., Tonielli, R., Di Martino, G., Innangi, S., Simeone, S., Parnum, I.M., 2010. Relationships between multibeam backscatter, sediment grain size and *Posidonia oceanica* seagrass distribution. Cont. Shelf Res. 30, 1941–1950.

Diaz-Uriarte, R., de Andres, S.A., 2006. Gene selection and classification of microarray data using random forest. BMC Bioinforma. 7, 1–13.

Ferrini, V.L., Flood, R.D., 2006. The effects of fine-scale surface roughness and grain size on 300 kHz multibeam backscatter intensity in sandy marine sedimentary environments. Mar. Geol. 228, 153–172.

Fielding, A.H., Bell, J.F., 1997. A review of methods for the assessment of prediction errors in conservation presence/absence models. Environ. Conserv. 24, 38–49.

Goff, J.A., Olson, H.C., Duncan, C.S., 2000. Correlation of side-scan backscatter intensity with grain-size distribution of shelf sediments, New Jersey margin. Geo-Mar. Lett. 20, 43–49.

Hamilton, L.J., Parnum, I., 2011. Acoustic seabed segmentation from direct statistical clustering of entire multibeam sonar backscatter curves. Cont. Shelf Res. 31, 138–148.
Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.
Heap A.D., Przeslawski R., Radke L., Trafford J., Battershill C., and Party S. 2010. Seabed Environments of the Eastern Joseph Bonaparte Gulf, Northern Australia. Sol4934 - Post-survey Report. Geoscience Australia, Canberra. Record 2010/09, 78pp.
Ho, T.K., 1995. Random decision forests. In: Proceedings of the Third International Conference on Document Analysis and Recognition, 278-282. Montreal, Que., Canada.
Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20, 832–844.
Jenness, J.S., 2004. Calculating landscape surface area from digital elevation models. Wildl. Soc. Bull. 32, 829–839.
Kloser, R.J., Penrose, J.D., Butler, A.J., 2010. Multi-beam backscatter measurements used to infer seabed habitats. Cont. Shelf Res. 30, 1772–1782.
Li, J., Heap, A., 2011. A review of comparative studies of spatial interpolation methods: performance and impact factors. Ecol. Inform. 6, 228–241.
Li, J., Hilbert, D.W., 2008. LIVES: a new habitat modelling technique for predicting the distributions of species' occurrence using presence-only data based on limiting factor theory. Biodiversity Conserv. 17, 3079–3095.
Li, J., Potter, A., Huang, Z., Daniell, J.J., Heap, A., 2010. Predicting Seabed Mud Content across the Australian Margin: Comparison of Statistical and Mathematical Techniques Using a Simulation Experiment. Geoscience Australia, Canberra, 2010/11, 146pp.
Li, J., Heap, A., Potter, A., Daniell, J.J., 2011a. Predicting Seabed Mud Content across the Australian Margin II: Performance of Machine Learning Methods and Their Combination with Ordinary Kriging and Inverse Distance Squared. Geoscience Australia, Canberra, Record 2011/07, 69pp.
Li, J., Heap, A.D., Potter, A., Daniell, J., 2011b. Application of machine learning methods to spatial interpolation of environmental variables. Environ. Model. Software. 26, 1647–1659.
Li, J., Heap, A.D., Potter, A., Huang, Z., Daniell, J., 2011c. Can we improve the spatial predictions of seabed sediments? A case study of spatial interpolation of mud content across the southwest Australian margin. Cont. Shelf Res. 31, 1365–1376.
Li, J., Potter, A., Heap, A., 2012a. Irrelevant Inputs and Parameter Choices: Do They Matter to Random Forest for Predicting Marine Environmental Variables? In: Australian Statistical Conference 2012. Adelaide.
Li, J., Potter, A., Huang, Z., Heap, A., 2012b. Predicting Seabed Sand Content across the Australian Margin Using Machine Learning and Geostatistical Methods. Geoscience Australia, Canberra, Record 2012/48, 115pp.
Liaw, A., Wiener, M., 2002. Classification and regression by radomForest. R News. 2, 18–22.
Marmion, M., Luoto, M., Heikkinen, R.K., Thuiller, W., 2009. The performance of state-of-the-art modelling techniques depends on geographical distribution of species. Ecol. Model. 220, 3512–3520.
McArthur, M.A., Brooke, B.P., Przeslawski, R., Ryan, D.A., Lucieer, V.L., Nichol, S., McCallum, A.W., Mellin, C., Cresswell, I.D., Radke, L.C., 2010. On the use of abiotic surrogates to describe marine benthic biodiversity. Estuarine Coastal Shelf Sci. 88, 21–32.
Newell, R.C., Seiderer, L.J., Robinson, J.E., 2001. Animal/sediment relationships in coastal deposits of the eastern English Channel. J. Mar. Biol. Assoc. U.K. 81, 1–9.
Okun, O., Priisalu, H., 2007. Random forest for gene expression based cancer classification: overlooked issues. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (Eds.), Pattern Recognition and Image Analysis: Third Iberian Conference, IbPRIA 2007. Lecture Notes in Computer Science4478, Springer-Verlag, Berlin, Heidelberg, pp. 483–490.
Post, A.L., Wassenberg, T.J., Passlow, V., 2006. Physical surrogates for macrofaunal distribution and abundance in a tropical gulf. Mar. Freshw. Res. 57, 469–483.
Prasad, A.M., Iverson, L.R., Liaw, A., 2006. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. Ecosystems. 9, 181–199.
Przeslawski, R., Daniell, J., Anderson, T., Vaughn Barrie, J., Heap, A., Hughes, M., Li, J., Potter, A., Radke, L., Siwabessy, J., Tran, M., Whiteway, T., Nichol, S., 2011. Seabed Habitats and Hazards of the Joseph Bonaparte Gulf and Timor Sea, Northern Australia. Geoscience Australia, Canberra, Record 2008/23, 69pp.

R Development Core Team, 2011. R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna.

Siwabessy, P.J.W., Daniell, J., Li, J., Huang, Z., Heap, A.D., Nichol, S., Anderson, T.J., Tran, M., 2013. Methodologies for seabed substrate characterisation using multibeam bathymetry, backscatter and video data: a case study from the carbonate banks of the Timor Sea, Northern Australia. Geoscience Australia, Canberra, Record 2013/11, 82pp.

Stein, D.L., Tissot, B.N., Hixon, M.A., Barss, W.H., 1992. Fish–habitat associations on a deep reef at the edge of the Oregon continental shelf. Fish. Bull. 90, 540–551.

Svetnik, V., Liaw, A., Tong, C., Wang, T., 2004. Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules. In: Roli, F., Windeatt, T. (Eds.), MCS 2004, pp. 334–343.

Thrush, S.F., Hewitt, J.E., Funnell, G.A., Cummings, V.J., Ellis, J., Schultz, D., Talley, D., Norkko, A., 2001. Fishing disturbance and marine biodiversity: role of habitat structure in simple soft-sediment systems. Mar. Ecol. Prog. Ser. 221, 255–264.

Warwick, R.M., Davies, J.R., 1977. The distribution of sublitoral macrofauna communities in the Bristol Channel in relation to the substrate. Estuarine Coastal Shelf Sci. 5, 267–288.

Weiss, A.D., 2001. Topographic Position and Landforms Analysis. In: ESRI International User Conference. San Diego, CA.

Wentworth, C.K., 1992. A scale of grade and class terms for clastic sediments. J. Geol. 30, 377–392.

Williams, I.M., Leach, J.H.J., 1999. The relationship between depth, substrate and ecology: a drop video study from the southeastern Australian coast. Oceanol. Acta. 22, 651–662.

CHAPTER 12

Supervised Classification of Images, Applied to Plankton Samples Using R and Zooimage

Grosjean Philippe, Denis Kevin
Numerical Ecology of Aquatic Systems, Institute of Complex Systems (Complexys), University of Mons, Belgium

12.1 Background

Supervised classification, a subset of data mining techniques, is particularly useful when a large quantity of cases must be rapidly classified into predefined classes. The machine learning approach uses a model built on a training set in which identification of each case is *a priori* known to learn how to automatically sort other, unknown, cases. In this chapter, we show how general machine learning approaches, usable in a wide range of situations like fraud detection, documents retrieval, disease detection, or microarray classification (Torgo, 2010) are applied to the classification of images. This chapter is focused on the classification of plankton images (Benfield et al., 2007) (Figure 12.1), but the proposed approach can be generalized to any digital images, e.g., pictures of butterflies (MacLeod et al., 2010).

There are several good reasons to survey the small creatures that live in aquatic environments (so-called, plankton). Marine plankton is a large community of plants, animals, bacteria, and viruses that live in the open seas, that is, in the water column and far away from solid substrates. It is very diverse (e.g., hundreds of species, or even more, can be found in just one liter of seawater). It includes organisms covering a wide range of sizes, from microscopic organisms like bacteria or viruses to large animals like jellyfishes (Lenz, 2000). Phytoplankton contains major primary producers in the sea (Reynolds, 2006), that is, those organisms that use the energy of light to build new organic compounds that fuel the ecosystem. Zooplankton constitutes a central component of the aquatic food web by transferring material synthesized by the phytoplankton to large predators like fishes, mammals, or birds (Lenz, 2000; Miller, 2004).

Plankton also constitutes a good bioindicator of environmental changes (Hays et al., 2005). It quickly varies both in space and time as a function of modifications in their environment, for

Data Mining Applications with R

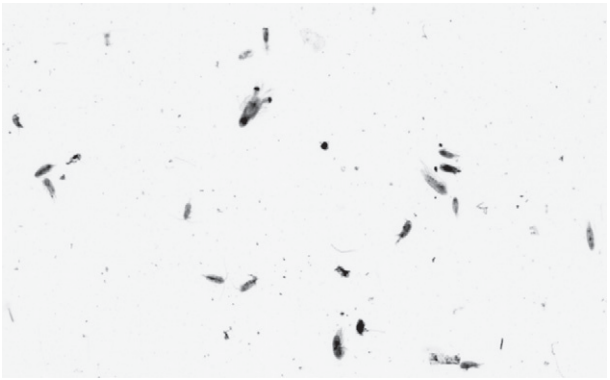


Figure 12.1
A small portion (1940 × 1200 pixels) of one original scan (10,000 × 6000 pixels) used in our example dataset. Several plankton organisms are visible (mainly copepods and one larger *Malacostraca*). Note also the large proportion of small detritus surrounding plankters.

instance, climatic changes or pollutions (Hays et al., 2005). Also, some harmful algae can produce toxins able to damage a whole ecosystem or to cause human diseases after ingestion of contaminated shellfishes (Graneli and Turner, 2006).

The study of plankton is far from an easy task because of the patchiness in its distribution and the inherently limited sampling effort that can be reasonably done in oceanographic campaigns. Despite a continuous effort of designing better sampling methods for plankton,[1] it is clear that anytime plankton communities are studied, the largest possible set of data needs to be gathered and processed, with "largest as possible" being limited mainly by the duration required to process the samples.

12.2 Challenges

The traditional analysis of preserved plankton samples consists in subsampling, counting, and sorting particles under a binocular or a microscope. Such an analysis is labor-intensive and results in a major time lag between sample collection and data interpretation (Benfield et al., 2007). Such a manual analysis requires taxonomists able to identify the different plankton species (Lenz, 2000; MacLeod et al., 2010). A well-trained taxonomist is able to analyze between one and two samples per day for the abundance in the different taxonomic groups, but without any additional measurements (e.g., the size of particles or the calculation of biomass) (Culverhouse et al., 2003). These experts are subject to fatigue when they work a long time with a microscope, which in turn, leads to mistakes (Culverhouse et al., 2003). Manual analysis of

[1] A review of traditional plankton sampling methods can be found in Wiebe and Benfield (Wiebe and Benfield, 2003).

plankton samples is thus not suitable to obtain enough spatial and temporal resolution required to study the patchy distribution of plankton in most oceanographic studies.

Since the 1980s, plankton digitization devices have been developed to speed up plankton sample analysis.[2] The recently developed plankton imagers like the ZOOSCAN (Gorsky et al., 2010; Grosjean et al., 2004) or the FlowCAM (Lancelot et al., 2012; Sieracki et al., 1998) have led to the accumulation of a large quantity of raw data and images. It now shifts to classifying the same particles on digital images rather than under a microscope (Benfield et al., 2007; Gaston and O'Neill, 2004; Grosjean et al., 2004). Image analysis and supervised classification of images offer the potential to automate, at least partly, this classification. It also allows for other measurements like the size or biomass of the plankton creatures (Bell and Hopcroft, 2008; Benfield et al., 2007).

Particles "numerically fixed" on digital pictures can be easily measured and counted by image analysis software. Several features for each region of interest (ROI), supposed to closely match the silhouette of each individual plankter can be used to characterize it, at least down to a given taxonomic level usable in most ecological studies (Sieracki et al., 1985). How to "fix numerically" plankton samples on digital pictures and how to perform the analysis of the particles visible on these images will not been explained here. We discuss instead how to analyze such kind of data.

A package called **zooimage** (http://CRAN.R-project.org/package=zooimage), which itself depends on the **mlearning** package (http://CRAN.R-project.org/package=mlearning) was developed to ease integration of data mining tools available in R (R Core Team, 2012), and more particularly machine learning algorithms, for plankton samples processing, but the **zooimage** package can also be used to analyze images from other sources than plankton samples. The **zooimage** package has already been used worldwide for zooplankton, phytoplankton, or bacteria analysis since its first public version in 2007 (Bachiller and Fernandes, 2011; Bell and Hopcroft, 2008; Di Mauro et al., 2011; Fernandes et al., 2009; Gillan et al., 2012; Gislason and Silva, 2009; Irigoien et al., 2009; Lancelot et al., 2012; Manriquez et al., 2009; Zarauz et al., 2008, 2009). Image analysis is done with ImageJ (a free software available at http://rsbweb.nih.gov/ij) (Schneider et al., 2012) using a series of dedicated plugins. Both the **zooimage** package and the dedicated ImageJ plugins constitute an integrated software called Zoo/PhytoImage (http://www.sciviews.org/zooimage) (Grosjean and Denis, 2007). In the planktonology/oceanography communities, no distinction is made between the image analysis part in ImageJ and the machine learning part in R and the whole is also referred to as ZooImage for zooplankton, or PhytoImage for phytoplankton in the scientific literature.

[2] For reviews of plankton imaging systems, see Benfield et al. (2007), Culverhouse et al. (2006), and Sieracki et al. (2010).

One goal of this chapter is to show how machine learning algorithms in the R environment can be applied to the classification of images, especially in the context of plankton images analysis. This chapter also shows why additional R code was required to ease this task in real-life application, including the definition and use of associated metadata and calculation of derived statistics at the sample level like abundances or biomasses per classes, or size spectra which are used, in practice, by ecologists. As any statistical approach, data mining requires correct preparation of the data: sampling of a representative subset, detection of outliers, elimination of missing values, and selection of pertinent descriptors. Such a preparation of the data is largely initiated in ImageJ, during the analysis of the raw images, but R is, of course, rich in tools for such a task too.

The general workflow of the process in Zoo/PhytoImage, from the raw images to the final descriptive statistics at the sample level, is presented in Figure 12.2. It can be split into three main stages: (1) image analysis and importation of data and metadata in R, (2) elaboration of training set(s) and classifier(s) coupled with their associated calibration and performance analysis, and (3) their use for automatic classification of unknown

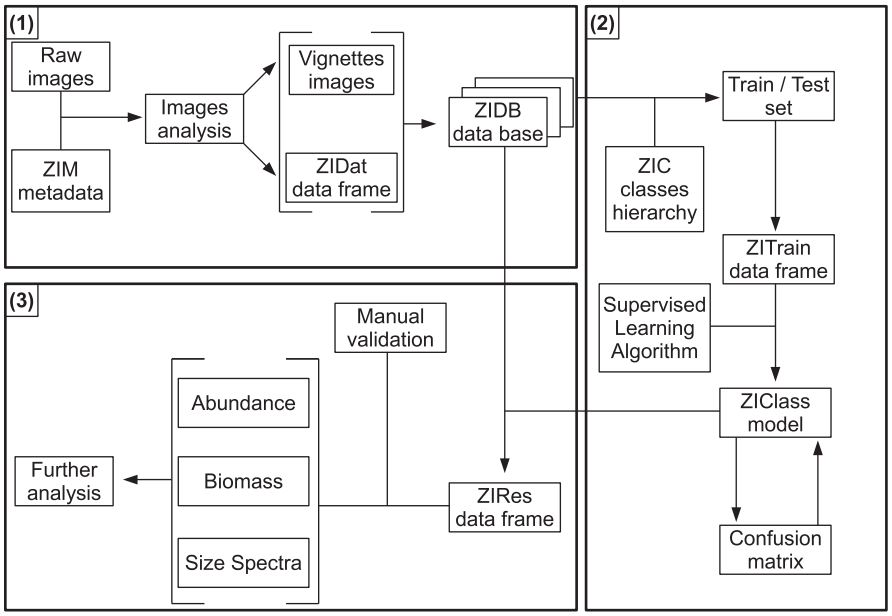


Figure 12.2

Workflow in **Zoo/PhytoImage**. There are three main stages in the process: (1) image analysis and data acquisition/preparation, (2) elaboration of training sets and classifiers and their calibration, and (3) classification of unknown samples and calculation of statistics at the sample level. Different specific S3 objects are created and used at each step.

samples.[3] Zoo/PhytoImage can analyze almost any kind of digital images. These images are imported in the first stage of the process and the integrated workflow, which is facilitated by the creation of dedicated R objects at each step of the process (Figure 12.2), is followed. Raw images are imported together with the metadata contained in the ASCII files with the "ZIM" (ZooImage Metadata) extension. Image analysis extracts small subimages of the particles of interest for future visual identification of the organisms (so-called vignettes) and numerical information is collected in "ZIDat" objects (ZooImage Data) containing a data frame of features measured on each particle along with the metadata. Both vignettes and "ZIDat" objects are stored on disk in the so-called ZIDB files (ZooImage DataBase). One "ZIDB" file is created for each sample and contains information about hundreds or thousands of particles. In the second stage, data from selected "ZIDB" files are extracted from the root of a series of directories representing the various organism classes (Figure 12.2). At this stage, a "ZIC" file (ZooImage Classes) is used to indicate the hierarchy in these classes. Experts can then manually classify representative items in the different classes by drag and drop of the corresponding vignettes in the different classes directories. This manual classification is then used to create the training set, stored in a "ZITrain" object (ZooImage Training set). The training set is used to train machine learning algorithms and to create a suitable model included in a "ZlClass" object (ZooImage Classifier). Performances of this classifier can be determined by the analysis of a confusion matrix, or by means of other metrics (e.g., based on ROC curves, lift charts, or cost plots). When the model is calibrated and, if performances are judged sufficient, the "ZlClass" object can be used in the third stage to automatically classify unknown particles from other "ZIDB" files (Figure 12.2). This creates a "ZIRes" object (Zoolmage Results) containing various statistics at the sample or station level, like abundance, biomass, and size spectra per class coupled with some of the initial metadata. In addition, partial or total manual validation can be done on automatic predictions to remove errors to obtain the final results. These results can be further analyzed in **R** or exported to any other software (Figure 12.2).

From the machine learning point of view, plankton classification constitutes probably one of the most difficult and challenging applications. This is typically a multiclass problem with hundreds of possible classes. A compromise needs to be made between the number of classes to discriminate (more or less detailed separation of the plankton taxonomic groups) and misclassification rate. Some particles are hard to identify by eye in the pictures, or do not belong to classes of interest (e.g., detritus). Training sets are contaminated by an unavoidable fraction of erroneous manual identifications (Culverhouse et al., 2003). Characteristics of the particles can slightly change in time or space and this is not completely caught in the training sets, most of the time. It is not the purpose of this chapter to discuss these difficulties in details,

[3] Note that Zoo/PhytoImage also proposes a toolbar and a menu (GUI) which allows users to drive their analyses without any knowledge of the R language. This has been a key point in its acceptation by a community of, chiefly, biologists. Here we don't present the Zoo/PhytoImage GUI, but show how analyses of plankton samples can be done using R code directly.

and we focus on showing how the tools in **R** are integrated in the Zoo/PhytoImage workflow to turn it into a system usable in routine (plankton) image studies. The system is flexible enough for specialists to adapt it to their specific applications or to derive better processes to address these difficulties. Up to now, this framework being used in practice by biologists to *partly* automate plankton classification: the general consensus is that a complete manual validation of the automatic classification is still required (Gorsky et al., 2010). This suboptimal situation is likely to change in the future. We discuss this important point later in this chapter.

12.3 Data Extraction and Exploration

In the particular case of plankton, specialized digitization devices, like the ZOOSCAN (Grosjean et al., 2004) or the FlowCAM (Sieracki et al., 1998), are used to provide high-resolution images. Similar systems can be used directly at sea for *in situ* digitization like the FlowCytoBot (Olson and Sosik, 2007) or the Video Plankton Recorder (VPR) (Davis et al., 1992). Digitization of plankton samples using a flatbed scanner or a camera coupled with a macro lens have also been used (Bell and Hopcroft, 2008; Gislason and Silva, 2009; Irigoien et al., 2009), and these systems can also be used for other applications.

Our example dataset[4] is obtained from 16-bit grayscale images (Figure 12.1) digitized with a flatbed scanner (Epson Perfection 4870) at 2400 dpi. A series of preserved plankton samples collected in the lagoon and the south pass of the coral reef off Toliara in Madagascar were placed in petri dishes with distilled water and digitized using transmitted light (standard light cover provided with the scanner for slides digitization).

The resulting images have been analyzed using ImageJ with one of the Zoo/PhytoImage plugins (Scanner_Gray16). ROIs are detected by binary segmentation in the images and 26 measurements are performed on each ROI.[5] In addition to the table of measurements, small subimages of each ROIs that we call "vignettes" are also generated (Figures 12.2 and 12.3). After raw images are analyzed, a series of vignettes and associated "ZIM" (ZooImage Metadata) files are collected together in one directory per sample on the disk.[6] In our example dataset, the first sample (MTPS.2004-10-20.V5) contains 1756 vignettes extracted from six scans. The **zooimage** package provides functions to compact all these items in one file per sample. In version 1 and 2, it was just a zipped archive with a special extension (".zid," for ZooImage Data). Starting from version 3, a more efficient approach is used by

[4] Follow instructions at http://www.sciviews.org/zooimage/Data mining with R/ for using these examples on your own computer.

[5] Image analysis is not detailed here and interested readers are encouraged to read the Zoo/PhytoImage manual (Grosjean and Denis, 2007) or the ?calcVars help page for more details about the different features.

[6] See the "/Samples" subdirectory in the example dataset. Open one of these ZIM files in a plain text editor to get an idea of the kind of data it contains.

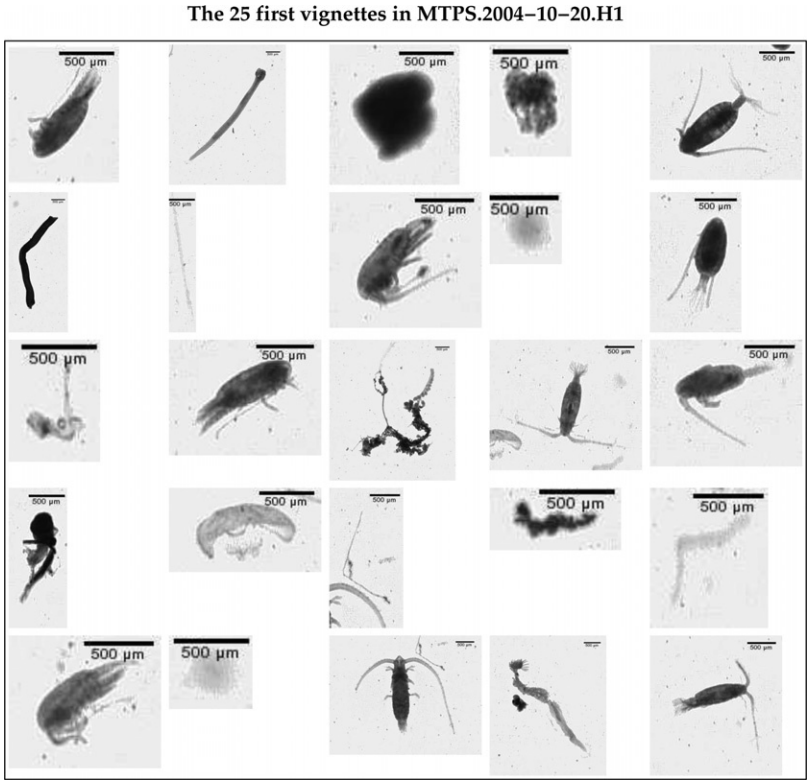


Figure 12.3
Collage of the 25 first vignettes included in the ZIDB file displayed in an **R** graph. The sample contains particles of heterogeneous shapes and sizes with a large quantity of crustaceans but also long animals called *Chaetognatha*, see the "/**Training set**" subdirectory in the example dataset for more examples of vignettes.

means of a flat file storage (with ".zidb" extension), as implemented in the **filehash** R package. They are called "ZIDB" files (ZooImage database, Figure 12.2). The conversion of the content of one sample directory (zidir argument) into a ZIDB file is done using:

```
> require(zooimage) # Load zooimage and mlearning among others

Loading required package: zooimage
Loading required package: svMisc
Loading required package: svDialogs
Loading required package: svGUI
Loading required package: mlearning

> ## Change this to the path where you placed example datasets
> rootdir <- "~/Data Mining with R"
> smp1file <- file.path(rootdir, "data", "Samples", "MTPS.2004-10-20.V5")
> zidbMake(smp1file)
```

The process of one sample directory with zidbMake() does this: (1) it creates the ZIDB file, (2) it adds vignettes and ZIM files, (3) it creates one "ZIDat" object (a data frame with all data and metadata), and (4) it adds the "ZIDat" object to the file. At the end of this process, the sample is ready to be used for the next steps of the workflow.

Because we usually have to deal with a series of samples in a routine, a batch mode is available in the **zooimage** package for all key functions. For instance, batch creation of many ZIDB files can be done using zidbMakeAll(). Batch functions have the following characteristics: (1) for long processes, they first check all items quickly to make sure they can be processed (e.g., no missing or corrupted files to process) and issue early warnings in case of problems, (2) each individual process never uses stop(), but always issues warning()s and invisibly returns TRUE or FALSE, and (3) a count-down of the batch process is displayed in the **R** console. That design allows for continuation of the batch process to the next item in case of failure, which would not be possible when using stop() messages (unless one uses something like try() or tryCatch()). Here is how you would convert all samples contained in one common directory (say the work done so far) into ZIDB files:

```
> ## Batch creation of ZIDB files for all samples in a common directory
> ## (don't delete source)
> smpdir <- file.path(rootdir, "data", "Samples")
> zidbMakeAll(smpdir, delete.source = FALSE, replace = TRUE)
```

The first argument to zidbMakeAll() is the root directory that contains all the sample subdirectories to batch convert. The package **filehash** (Peng, 2006) allows to link to ZIDB files data and to manipulate its content as if it was already loaded in memory, taking advantage of the lazy loading mechanism build in **R**. The **zooimage** package builds on this feature to ease access to measurements or vignettes:

```
> smpsfiles <- c("MTPS.2004-10-20.V5.zidb", "MTLG.2005-05-24.H1.zidb")
> smps <- file.path(smpdir, smpsfiles)
> ## Load and explore data from one ZIDB file
> dat1 <- zidbDatRead(smps[1])
> head(dat1, n = 2)
                    Label  Item    ECD    Area    Mean  StdDev   Mode    Min    Max
  1  MTPS.2004-10-20.V5+A1    1.   5422  0.2309  0.1979  0.1278  0.036  0.004  0.444
  2  MTPS.2004-10-20.V5+A1    2.  .7306  0.4192  0.1558  0.1312  0.008  0.004  0.448
         X       Y     XM      YM  Perim.     BX      BY   Width  Height   Major
  1   82.51  0.6430  82.49  0.6962   2.624  82.17  0.1693  0.6665  0.8464  0.9375
  2   23.72  0.9492  23.73  0.9517   6.571  22.92  0.2222  1.4071  1.3119  0.9057
    Minor   Angle   Circ.   Feret  IntDen  Median    Skew    Kurt  XStart  YStart
  0.3136   52.03  0.4213  0.9918  0.0457   0.208  0.0658  -1.333    7822      16
  0.5893  130.65  0.1220  1.6782  0.0653   0.124  0.3978  -1.323    2280      21
    Dil
0.7461
0.7461
```

The general structure of "ZIDat" object is a data frame containing all features measured on each vignette. In addition, the "ZIDat" object also contains original metadata:

```
> attr(dat1, "metadata")
  $Fraction
                     Label   Code   Min   Max
  1   MTPS.2004-10-20.V5+A      A   500    -1
  2   MTPS.2004-10-20.V5+B      B   500    -1

  $Image
                      Label        Author     Hardware         Software
  1   MTPS.2004-10-20.V5+A   Kevin Denis    EPSON 4870   VueScan 8.0.10
  2   MTPS.2004-10-20.V5+B   Kevin Denis    EPSON 4870   VueScan 8.0.10
                  ImageType
1 trans 16bits gray 2400dpi
2 trans 16bits gray 2400dpi

  $Process
                      Label   Version               Method   MinSize   MaxSize
  1   MTPS.2004-10-20.V5+A     0.4-0   default [0.25 - 10]      0.25        10
  2   MTPS.2004-10-20.V5+B     0.4-0   default [0.25 - 10]      0.25        10

     Calibration   ProcessPixSize   WhitePoint   BlackPoint
  1    EPSON 4870          0.01058         1806        12510
  2    EPSON 4870          0.01058         1806        12510

  $Subsample
                     Label   SubPart   SubMethod   CellPart   Replicates   VolIni
  1   MTPS.2004-10-20.V5+A    0.3060   volumetry       0.73            3        2
  2   MTPS.2004-10-20.V5+B    0.0927   volumetry       0.73            3        2
     VolPrec
  1      0.2
  2      0.2
```

Thus metadata remain coupled with the tables of measurements along the Zoo/PhytoImage workflow and it is possible to add further information there for traceability, if needed.[7] Note, for instance, *SubPart*, *CellPart*, *Replicates*, and *VolIni* that are used to calculate the *Dil*

[7] Metadata fields are not detailed here, readers are encouraged to consult the Zoo/PhytoImage manual (Grosjean and Denis, 2007).

column of the "ZIDat" object that provides coefficients to use for conversion from observed frequencies to number of individuals per volume of seawater:

$$\text{Dil} = \frac{1}{\text{SubPart}\cdot\text{CellPart}\cdot\text{Replicates}\cdot\text{VolIni}}$$

where *Dil* is the "dilution" coefficient to apply to each vignette; *SubPart* is the proportion of each fraction obtained after separation of the original sample on one or more sieves to split large and small organisms and possibly use different dilutions for these fractions (here two fractions (A) higher and (B) smaller than 500 µm, respectively); *CellPart* is the proportion of items analyzed by the Zoo/PhytoImage—ImageJ plugin; *Replicates* is the number of replicates digitized for each fraction; and *VolIni* is the volume of water column filtered to collect the original sample.

Of course, you can use all R functions here to explore "ZIDat" object, e.g., summary() or plot():

```
> summary(dat1[, c("Area", "Perim.", "Skew", "Kurt")])
       Area             Perim.             Skew               Kurt        
 Min.   :  0.052   Min.   :  0.90   Min.   : -1.181   Min.   :  -1.66  
 1st Qu.:  0.100   1st Qu.:  1.64   1st Qu.:  0.363   1st Qu.:  -1.08  
 Median :  0.185   Median :  2.49   Median :  0.686   Median :  -0.53  
 Mean   :  0.306   Mean   :  3.56   Mean   :  0.870   Mean   :   0.97  
 3rd Qu.:  0.279   3rd Qu.:  3.95   3rd Qu.:  1.134   3rd Qu.:   0.81  
 Max.   :  8.002   Max.   : 35.34   Max.   :  8.468   Max.   : 119.29  

> plot(dat1$Area, dat1$Perim., xlab = "Area", ylab = "Perimeter"i)
>## (Plot is not reproduced here)
```

Particles included in the first sample (MTPS.2004-10-20.V5) show a large range of sizes with areas varying from 0.05 to 8 mm^2 and perimeters varying from 0.9 to 35 mm. These parameters will be used to classify images (Figure 12.2).

You can also display thumbnails of vignettes directly in R. After linking to a ZIDB database, you can see, e.g., the 25 first vignettes like this (result displayed in Figure 12.3):

```
> ## Lazy loading data from one ZIDB file in R
> db1 <- zidbLink(smps[1])
> ## Contains data in *_dat1 and vignettes in *_nn
>items1 <- ls(db1)
> vigs1 <- items1[-grep("_dat1", items1)]
> ## Display a 5*5 thumbnail of the first 25 vignettes (Figure 12.3)
> zidbPlotNew("The 25 first vignettes in MTPS.2004-10-20.H1")
> for (i in 1:25) zidbDrawVignette(db1[[vigs1[i]]], item = i, nx = 5, ny = 5)
```

12.4 Data Preprocessing

Once all ZIDB files are created, the user moves to the second stage: creation of the training set (Figure 12.2). This is a little time-consuming because it involves the manual classification of vignettes from a series of samples into several classes. The number of classes to include in the training set depends on the purpose of the study. Ideally, all taxonomic groups should be separated but in the case of plankton analysis, it is an impossible task because of the large diversity of plankton communities, especially at Madagascar (Conway, 2005). The number of classes must be chosen as a trade-off between taxonomic resolution and potentials to discriminate items based on their vignettes (Embleton et al., 2003; Grosjean et al., 2004; Hu and Davis, 2006; Irigoien et al., 2009; Lancelot et al., 2012; Luo et al., 2003; Sosik and Olson, 2007; Tang et al., 1998). In such studies, typically, a couple of dozens classes are made. These are possibly simplified a little further on, depending on the final objectives (e.g., study of larger ecological functional groups).

The training set is imported in a data frame containing the discriminant variables plus one categorical variable, named *Class*, with manual identification. In **zooimage**, a multilevel hierarchy of the classes can be specified and used (larger classes containing more detailed classes). A subsample of vignettes must be manually sorted into a structure of directories and subdirectories reflecting that hierarchy on the disk. To prepare the directories, **zooimage** uses specifications written in a "ZIC" file (ZooImage Classification specification file, see Figure 12.2). Among the directories that are created, a special one is dedicated to unclassified items: it is a folder named "_" (underscore). The "unclassified items" directory will, of course, be used here to store all vignettes from the ZIDB files waiting for manual identification. Vignettes left unidentified will simply remain there (or in one of its subdirectories), and will be further ignored in the training set.

In order to prepare the training set for manual identification of vignettes, you do:

```
> train1dir <- file.path(rootdir, "data", "_train")
> prepareTrain(train1dir, zidbfiles = smps, template = "[Detailed]")
```

The first argument of prepareTrain() is the name of the main training set directory; the argument zidbfiles is a character string indicating the path of the ZIDB files to be used (i.e., all their vignettes are placed in the "unclassified items" directory on the creation of the training set and you have to move them into the respective class directories) and the argument template is the path to a file that describes the initial classes hierarchy to use. In the present case, the default "[Detailed]" tree structure provided with the **zooimage** package is used. Here is how to look at its content:

```
> file.show(system.file("etc", "Detailed.zic", package = "zooimage"))
> ## (Content of the file not reproduced here)
```

Use any file explorer or images browser program to inspect the training set and drag and drop vignettes with the mouse from the "unclassified items" directory to the appropriate folders (Figure 12.4) while you build your training set. You are, of course, free to create new subdirectories for other classes. Once the manual sorting of a representative subset of your vignettes is done, you can collect data for your training set using:

```
> train2dir <- file.path(rootdir, "data", "Training set")
> train <- getTrain(train2dir)
```

The function getTrain() reads the content of all directories, except the "unclassified" one and its subdirectories. It then combines manual identifications of the vignettes with their associated

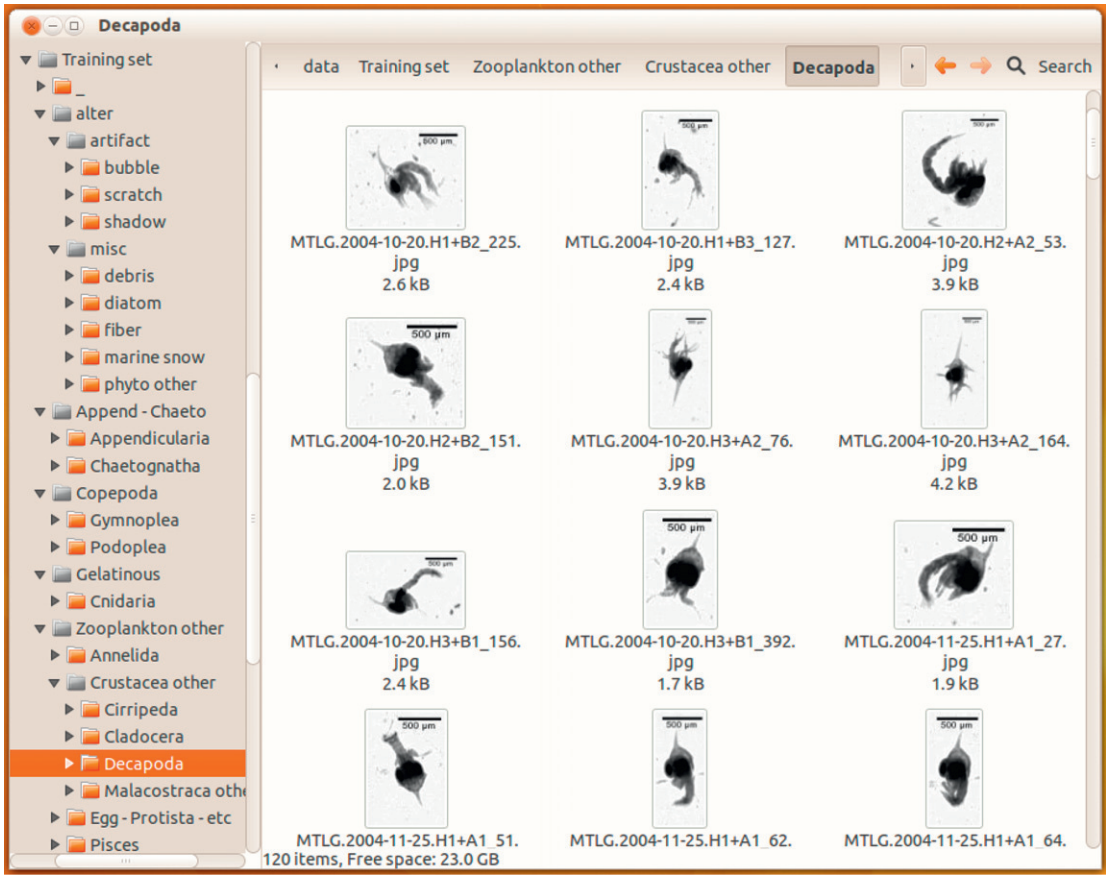


Figure 12.4
The example training set with its hierarchy of classes (left) as it appears in a typical file explorer. Vignettes (right) can be moved into the corresponding directories (here, "Decapoda"). Snapshot of the Nautilus file browser under Ubuntu 12.04.

measurements taken from the ZIDB files. The training set (Figure 12.2) information is packed together in one R object of class "ZITrain."

```
> head(train, n = 2)
                            Id                  Label Item    ECD   Area
1 MTLG.2004-10-20.H1+A1_106 MTLG.2004-10-20.H1+A1  106 0.7698 0.4654
2 MTLG.2004-10-20.H1+A1_109 MTLG.2004-10-20.H1+A1  109 0.7004 0.3853
    Mean StdDev  Mode   Min   Max     X     Y    XM    YM Perim.    BX
1 0.3590 0.2287 0.032 0.004 0.900 87.98 27.09 87.98 27.08  4.447 87.39
2 0.3569 0.1811 0.492 0.024 0.668 14.71 27.79 14.71 27.78  2.325 14.36
     BY  Width Height  Major  Minor Angle  Circ.  Feret IntDen Median
1 26.76 1.3013 0.8464 1.1640 0.5091 18.47 0.2958 1.3168 0.1671  0.380
2 27.43 0.6983 0.7089 0.7128 0.6882 52.89 0.8959 0.7281 0.1375  0.376
     Skew   Kurt XStart YStart    Dil             Class
1  0.0115 -1.116   8336   2529 0.2393 Poecilostomatoida
2 -0.2125 -1.202   1388   2593 0.2393       Egg - round
```

The training set used in this example contains the data of 1931 vignettes, with 31 features measured on the images and which have been manually sorted into 25 classes, that is $\{x_i, y_i\}_{i=1}^{1931}$ where $x \in \mathbb{R}^{31} \rightarrow y \in \{1, 2, \ldots, 25\}$. In the perspective of the zooplankton analysis, not all classes are of interest. Classes with lowercase names ("bubble," "marine snow," etc.) are those we want to discard further in the analysis. Anyway, we need them to allow sorting of zooplankton versus the rest.

```
> sort(table(train$Class))
             Cnidaria             Cirripeda                bubble
                   21                    22                    26
         Harpacticoida              Annelida            Egg - round
                   39                    45                    49
              Protista             Cladocera          Egg - elongated
                   49                    50                    50
                Pisces               scratch             Gastropoda
                   50                    50                    51
               diatom            Cyclopoida           Chaetognatha
                   51                    58                    65
       Appendicularia           phyto other                shadow
                   69                    69                   100
                fiber                debris       Malacostraca other
                  110                   111                   119
              Decapoda           marine snow       Poecilostomatoida
                  120                   121                   159
           Calanoida
                  277
```

The hierarchy of the classes is also recorded in the "ZITrain" object (as "path" attribute), and it can be used to recode the classes at various levels of details:

```
> table(recode(train, depth = 2)$Class)


             Annelida       Cnidaria     Crustacea other
                   58             21                270
    Egg - Protista - etc      Gymnoplea          Pisces
                  367            277                 49
             Podoplea       artifact           misc
                  251            188                450

> table(recode(train, depth = 1)$Class)

    Append - Chaeto    Copepoda    Gelatinous Zooplankton    other
                114         414                        21      744
              Alter
                638
```

12.5 Modeling

The model creation is, of course, a critical stage. It involves a series of steps: selection of variables (possibly calculating also derived or transformed ones), choice of a machine learning algorithm, its calibration, and choice of best set of parameters for the selected algorithm. Cross-validation on the training set is often used at this stage. With zooimage, these different steps are embedded into one specific object: "ZIClass" (ZooImage Classifier, see Figure 12.2). It is built on top of "mlearning" objects that ensure a unique, unified interface that can be used for each machine learning algorithm. Hence, to grow a random forest (Breiman, 2001) classifier with default values, and to perform 10 times cross-validation on our example training set, one makes:

```
> classif <- ZIClass(Class ~ ., train, method = "mlRforest", calc.vars = calcVars,
+     cv.k = 10)
> classif
> ## (Output not shown here)
```

Building a "ZIClass" object using the ZIClass() function involves internally three steps: (1) calculation and/or selection of the variables, (2) creation of a classifier, and (3) *x*fold cross-validation of the classifier on the training set.

Calculation and/or selection of the variables is done by applying an external function on the original data frame. The function to use is specified by the calc.vars argument and it points by

default to the calcVars() function in the **zooimage** package.[8] Here are the original versus transformed variables after running calcVars():

```
> names(train) # Original variables
 [1] "Id"      "Label"   "Item"    "ECD"     "Area"    "Mean"    "StdDev"
 [8] "Mode"    "Min"     "Max"     "X"       "Y"       "XM"      "YM"
[15] "Perim."  "BX"      "BY"      "Width"   "Height"  "Major"   "Minor"
[22] "Angle"   "Circ."   "Feret"   "IntDen"  "Median"  "Skew"    "Kurt"
[29] "XStart"  "YStart"  "Dil"     "Class"
> ## Variables as seen by the classifier
> names(traincalc <- calcVars(train))
 [1] "ECD"          "Area"         "Mean"         "StdDev"
 [5] "Mode"         "Min"          "Max"          "Perim."
 [8] "Major"        "Minor"        "Circ."        "Feret"
[13] "IntDen"       "Median"       "Skew"         "Kurt"
[17] "Class"        "AspectRatio"  "CentBoxD"     "GrayCentBoxD"
[21] "CentroidsD"   "Range"        "MeanPos"      "SDNorm"
[25] "CV"           "MeanDia"      "MeanFDia"     "Transp1"
[29] "Transp2"      "Elongation"   "Compactness"  "Roundness"
```

With a carefully designed calcVars() function that embeds all required variables calculation, one ends up with a very simple formula specification for the machine learning model as Class ~. (variable *Class* is modeled as a function of all the other variables present in the data frame after calcVars() processing). Code in the "mlearning" object that is called internally ZIClass() is optimized for memory usage and processing speed with such a formula. Of course, one can still specify a more complex formula for the model (specifying descriptors and transformed descriptors directly in the formula): this is an optimized call on one hand versus flexibility on the other hand thanks to the R "formula" objects.

As an illustration of a variables selection procedure, we use here the decrease in Gini criterion, calculated by the random forest algorithm (Figure 12.5), to select only most discriminant predictors (Torgo, 2010):

```
> require(randomForest)
> ## Importance of the predictors
> Imp <- classif$importance
> varImpPlot(classif, n.var = nrow(Imp))
> ## Plot shown at Figure 12.5
```

[8] Type calcVars at the R console to see how final variables are calculated and look at its help page ?calcVars to understand its logic.

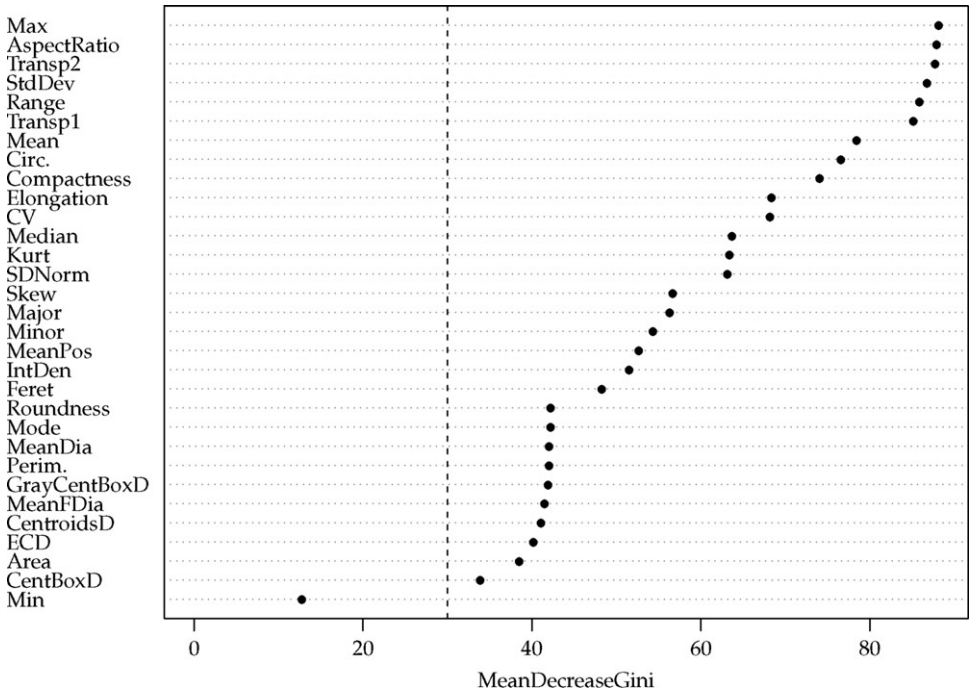


Figure 12.5
Importance of predictor variables as measured by the mean decrease of the Gini index for the random forest classifier *classif*.

```
> Threshold <- 30
> abline(v = Threshold, lty = 2)
> ## Drop variables with low Gini decrease
> VarsToDrop <- rownames(Imp)[Imp < Threshold]
```

The list of predictors we want to use in **zooimage** is specified in the ZI.dropVars option this way:

```
> options(ZI.dropVars = VarsToDrop)
> ## Now, 'Min' is drop from the dataset
> names(traincalc <- calcVars(train))
     [1] "ECD"         "Area"        "Mean"        "StdDev"
     [5] "Mode"        "Min"         "Max"         "Perim."
     [9] "Major"       "Minor"       "Circ."       "Feret"
    [13] "IntDen"      "Median"      "Skew"        "Kurt"
    [17] "Class"       "AspectRatio" "CentBoxD"    "GrayCentBoxD"
    [21] "CentroidsD"  "Range"       "MeanPos"     "SDNorm"
    [25] "CV"          "MeanDia"     "MeanFDia"    "Transp1"
    [29] "Transp2"     "Elongation"  "Compactness" "Roundness"
```

In the present case, only *Min* variable is considered as uninformative and is eliminated. If needed, you can replace calcVars() by your own function in order to calculate a different set of predictors.

Another usual step here consists in choosing better parameters for the classification algorithm. For instance, with random forest, one can determine the best number of trees to be used:

```
> plot(classif$err.rate[, "OOB"], type = "l", xlab = "trees", ylab = "Error")
> ## Plot shown at Figure 12.6
> abline(v = 200, lty = 2)
> ## 200 trees is largely enough here (default is to use 500 trees)
```

The error is stable after 200 trees and the ntree argument of mlRforest() can thus be changed from the default value of 500 to 200. The classifier becomes:

```
> classifrf <- mlearning(Class ~ ., data = traincalc,
+     method = "mlRforest", ntree = 200)
```

The third and final step in ZIClass() is an *x*fold cross-validation on the transformed data frame using the same machine learning algorithm. This is equivalent to:

```
> ## 10-fold cross-validation set by using cv.k = 10
> classifcv <- predict(classifrf, method = "cv", cv.k = 10)
> classifcv[1:6] # The six first predictions
  [1] Malacostraca other Egg - round marine snow
  [4] Poecilostomatoida Poecilostomatoida Poecilostomatoida
  25 Levels: Annelida Appendicularia Calanoida Chaetognatha ... shadow
```

All "mlearning" objects allow for method = "cv" in its predict() method, which is equivalent to using the cvpredict() method on the same object. It calls internally errorest() from package

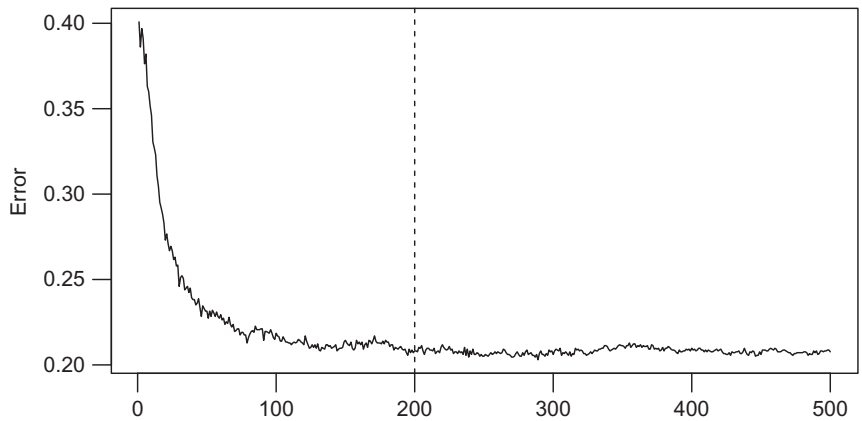


Figure 12.6

The out-of-bag error as a function of the number of trees in the random forest classifier.

ipred (Peters and Hothorn, 2012) for (by default) 10-fold cross-validation using a stratified sampling, but other strategies are accessible too. See ?cvpredict and ?errorest.

All these three steps can be done with one line of code, by calling ZIClass(). To recreate the previous classifier integrating both algorithm optimization and variables selection directly in the ZIClass() function, one makes:

```
> classif <- ZIClass(Class ~ ., data = train,
+     method = "mlRforest",calc.vars = calcVars,
+     ntree = 200, cv.k = 10)
```

In the **mlearning** package version 1.0-0, several machine learning algorithms are available, like linear discriminant analysis, quadratic linear discriminant analysis, learning vector quantization, neural network, support vector machine, naive Bayes, or random forest. The "mlearning" objects are unified interfaces on top of algorithms implemented in other R packages, like **MASS**, **class**, and **nnet** (Venables and Ripley, 2002), **e1071** (Dimitriadou et al., 2011), or **randomForest** (Liaw and Wiener, 2002). In future versions of **mlearning**, there will be an increasing number of machine learning algorithms that will be made compatible with "mlearning" objects, including Weka code, through the **RWeka** package (Hornik et al., 2009).

12.6 Model Evaluation

ROC curves are popular tools to evaluate binary classifiers (Drummond and Holte, 2006; Hanczar et al., 2010; Vuk and Curk, 2006). However, they do not easily scale up for multiclass problems (Hand and Till, 2001), especially when the number of classes is high, like in the present case. The confusion matrix and the statistics based on the confusion matrix like the F-score (Hand, 2009) are still best indicators in multiclass cases. Hence, the **mlearning** package proposes the "confusion" object and methods print(), summary(), and plot() (with four different types) to evaluate a classifier, and **zooimage** builds on it:

```
## All proposed statistics descriptors
summary(classif)
## (Output not displayed here)
```

The summary() method of the "ZIClass" object is identical to summary (confusion(ZIClass)). In this case, it always uses cross-validated predictions, but in the case of some algorithms, like random forest, you could use the out-of-bag as well.[9] You can, of course, select the statistics you want among all possibles ones.

```
> ## Get selected statistics only
> summary(classif, type = c("Fscore", "Recall", "Precision"))[1:3, ]
```

[9] To get the confusion matrix from out-of-bag prediction, you make: confusion(classif, predict(classif, method = "oob")).

```
    1931 items classified with 1525 true positives (error = 21%)

    Global statistics on reweighted data:
    Error rate: 21%, F(micro-average): 0.79, F(macro-average): 0.783
                    Fscore  Recall  Precision
    Egg - elongated 0.9515  0.9800     0.9245
    scratch         0.9505  0.9600     0.9412
    Protista        0.9485  0.9388     0.9583
> ## Only the output for the three first classes is printed.
```

If you want to access and further inspect the confusion matrix itself, you should build the "confusion" object explicitly. This is very easy:

```
> conf <- confusion(classif)
> conf
> ## (Output not shown here)
```

The "confusion" object inherits from the standard R "table" object, and it proposes specific print(), summary(), and plot() methods (Figure 12.7).

```
> ## type = 'image' is default and thus facultative
> plot(conf, type = "image")
> ## Plot shown in Figure 12.7
```

Note that the confusion matrix is, by default, displayed with the classes sorted as a function of their mutual false positive/false negative rates, so that classes with higher confusion are as close to each other as possible (Figure 12.7). Sorting is done using hierarchical clustering (hclust() R function) on a symmetric matrix with the average of false positives and false negatives out of the diagonal (that is, indeed, conf + t(conf)/2) and by default, the Wald aggregation criterion. You can see the dendrogram of that hierarchical clustering too (Figure 12.8):

```
> plot(conf, type = "dendrogram")
> ## Plot shown in Figure 12.8
```

Two other useful representations to visualize classifier performances, related to recall *versus* precision for each class, or its combination as F1-score are also available (Figures 12.9 and 12.10):

```
> plot(conf, type = "barplot") # See Figure 12.9
> plot(conf, type = "stars") # (not shown here)
```

In the probable situation where one would like to compare two different classifiers, there is a graph that allows such a visual comparison. Let's build a support vector machine classifier with a linear kernel on the same training set and let's compare it with random forest, class-by-class according to recall and precision (Figure 12.10):

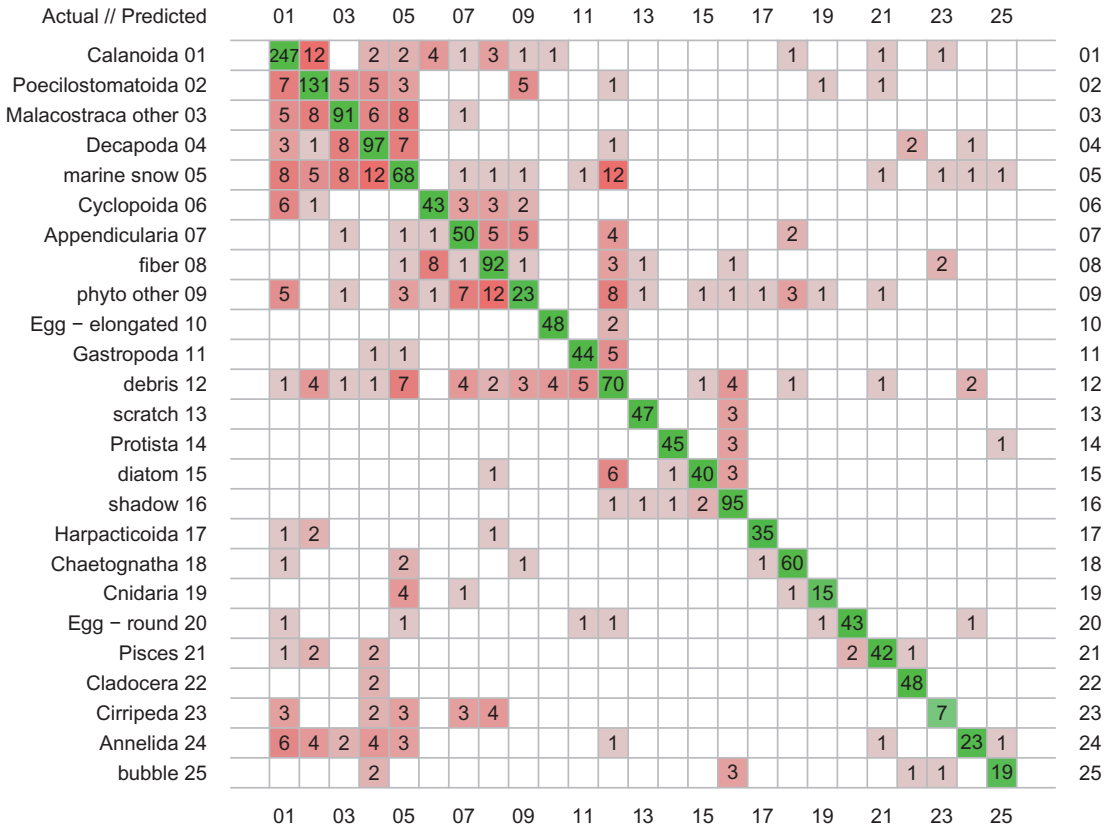


Figure 12.7

The confusion matrix calculated by cross-validation from a "**ZIClass**" object using its default **plot()** method. This is the graphical representation of the numerical matrix obtained by **confusion(classif)**.

```
> classif2 <- ZIClass(Class ~ ., train, method = "mlSvm", kernel = "linear")
> classif2
> # [...]
> summary(classif2)
> # [...]
> conf2 <- confusion(classif2)
> conf2
> # [...]
ConfuDist
hclust (*, "ward")

> plot(conf2) # (not shown here)
> plot(conf2, type = "barplot") # Idem
```

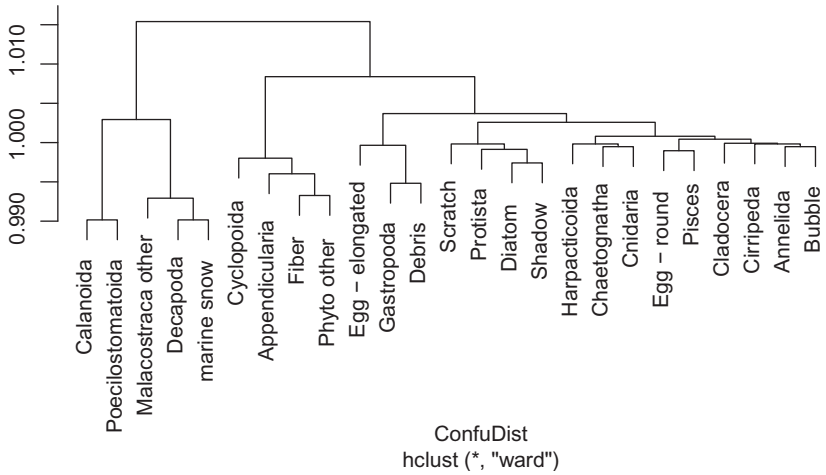


Figure 12.8

Hierarchical clustering of classes in the confusion matrix, based on the average between false positives and false negatives as interclass distances and obtained with **plot(conf, type = "dendrogram")**.

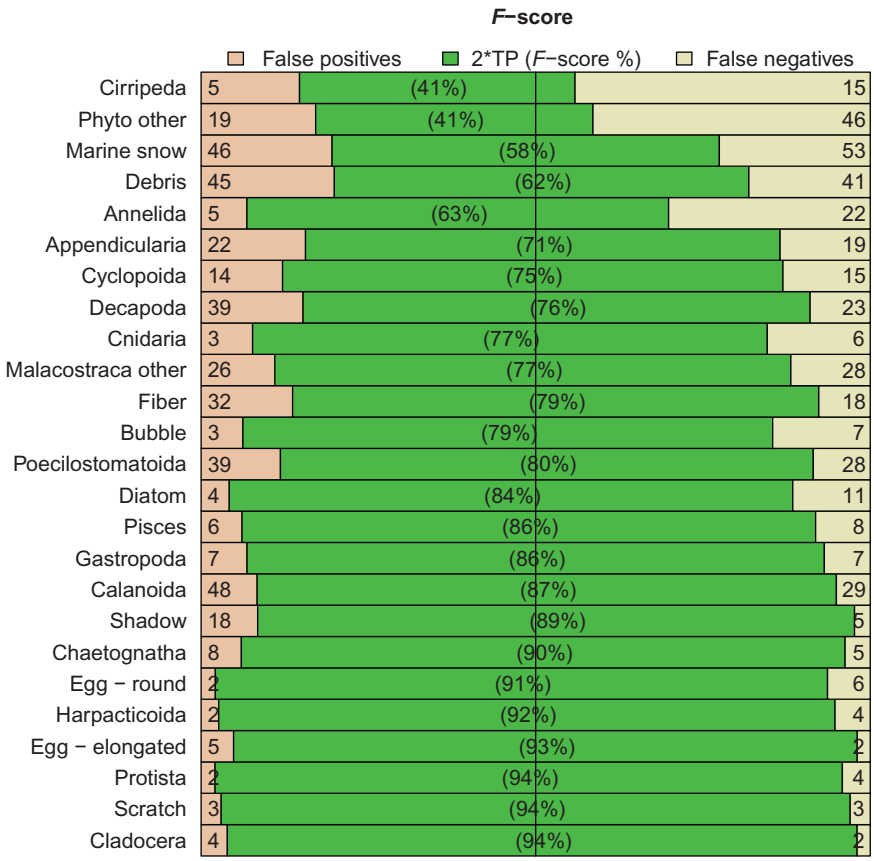


Figure 12.9

Graphical representation of the F-score and false positive *versus* false negative rate, as obtained using **plot(conf, type = "barplot")**.

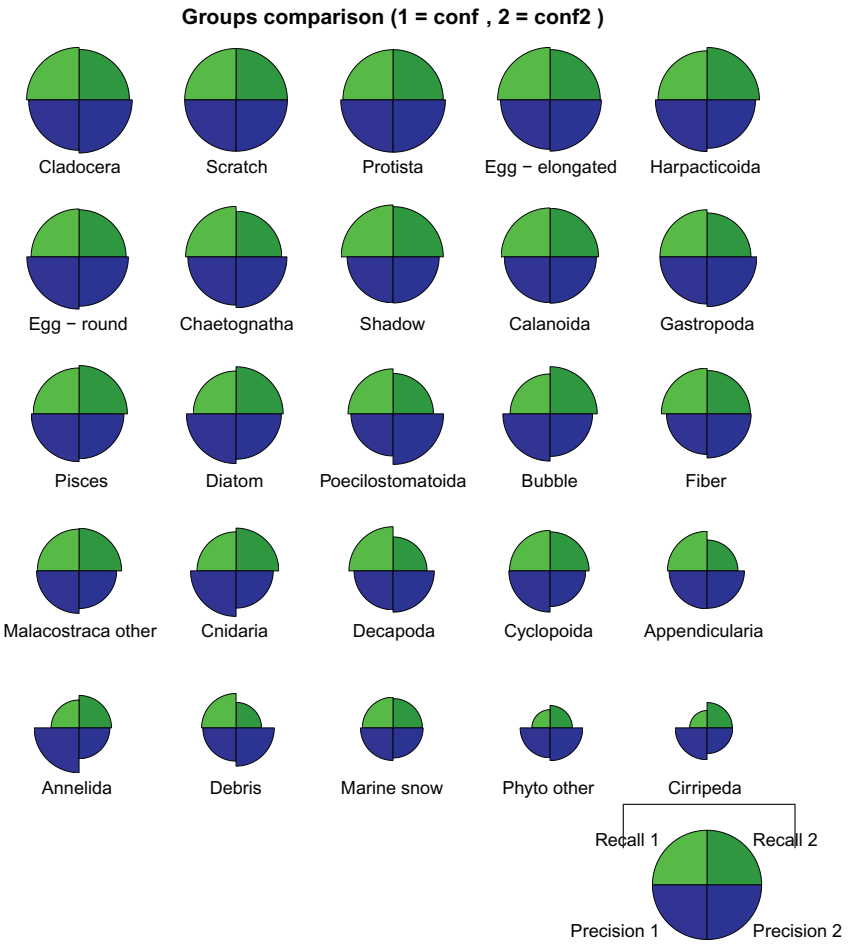


Figure 12.10
Visual comparison of two classifiers, based on recall and precision for each class as obtained by plot(conf, conf2), the two respective confusion objects. Classifier 1 is random forest; classifier 2 is support vector machine.

```
> ## Comparison of both classifiers
> plot(conf, conf2) # See Figure 12.10
```

Both algorithms show accuracies close to 80% with similar recall and precision for the large majority of classes (Figure 12.10). When larger differences are observed, algorithms show an opposite behavior with a higher recall for random forest and a higher precision for support vector machine as observed for *Chaetognatha* or *Appendicularia*, two elongated creatures (Figure 12.10). The opposite is observed for other classes as *Cnidaria* or *Annelida*. The smallest

values and the highest dissimilarities between the tested algorithms are observed for the *Cirripeda* class.

As the statistics based on the confusion matrix are dependent on the relative proportions of the items in the different classes, it is important to define good priors. Plankton communities can change drastically, with different taxonomic groups that dominate the communities at different times or geographical locations. Moreover, the number of particles included in the training set tends to be more balanced, with an oversampling of the rare groups in order to get enough vignettes for learning (a couple of dozens, at least). This means that prior probabilities need to be adjusted in the confusion matrix to reflect the relative proportions of each class in a given practical situation. This can be done easily with the "**confusion**" object. For instance, one can inspect the performances of the random forest algorithm with a realistic sample containing half of nonliving particles (so-called marine snow, fiber, or debris in the training set) and a zooplankton community dominated by calanoid copepods and chaetognaths this way:

```
> priors <- rep(20, 25)
> names(priors) <- rownames(conf)
> priors[c("Calanoida", "Chaetognatha")] <- c(300, 100)
> priors[c("debris", "fiber", "marine snow")] <- 266
> priors
```

```
         Annelida     Appendicularia    Calanoida
               20                 20          300
     Chaetognatha          Cirripeda    Cladocera
              100                 20           20
         Cnidaria         Cyclopoida     Decapoda
               20                 20           20
   Egg - elongated         Egg - round   Gastropoda
               20                 20           20
    Harpacticoida  Malacostracaother       Pisces
               20                 20           20
Poecilostomatoida           Protista       bubble
               20                 20           20
           debris             diatom        fiber
              266                 20          266
       marine snow         phyto other      scratch
              266                 20           20
           Shadow
               20
```

```
> prior(conf) <- priors
> summary(conf, type = c("Fscore", "Recall", "Precision"))[1:3, ]
1931 items classified with 1525 true positives (error = 21%)

Global statistics on reweighted data:
Error rate: 24%, F(micro-average): 0.728, F(macro-average): 0.703

                Fscore    Recall    Precision
scratch         0.9627    0.9600       0.9653
Protista        0.9539    0.9388       0.9694
Chaetognatha    0.9267    0.9077       0.9465
> ## Only the output for the three first classes is printed.
> ## Reset prior to frequencies in the training set
> prior(conf) <- NULL
```

A better approach than to manipulate the confusion matrix calculated after the training set to estimate the performances of the classifier is to use a separate test set with one representative sample. Here is what we got with the first sample in our example dataset:

```
> prepareTest(file.path(rootdir, "data", "_test"), smps[1], template = classif)
> ## (now you are supposed to sort the vignettes) Let's pretend this is the
> ## result...
> test <- getTest(file.path(rootdir, "data", "Test sets", "MTPS.2004-10-20.V5"))
> ## Create and study the confusion matrix
> conf3 <- confusion(predict(classif, test), test$Class)
> conf3
> # 1756 items classified with 914 true positives (error rate = 47.9%) [...]
```

Half of all the particles in our testing sample are misclassified! The presence of visually unrecognizable particles in the vignettes contributes to increase in this error rate (Bell and Hopcroft, 2008). Such a score leads to unusable results by biologists or ecologists, unless they decrease the misclassification fraction by performing a so-called manual validation in the plankton literature (Gorsky et al., 2010). It consists in a manual correction of the automatic classification proposed by the machine learning algorithm for all vignettes. This manual validation is done very similarly to the creation of a test set, but first with seeding the vignettes in the different folders according to their automatic classification.[10]

Here is an example of a separate sample with its validation set. The additional argument classes = classif in prepareTest() allows for presorting of the vignettes on the disk by using the *classif* classifier:

[10] Note the subtle difference between a test set where all vignettes are first located in the "unclassified items" folder and a validation set where the vignettes are presorted according to their predicted classes. Thus, the test set is independent from the automatic classification, while the validation set is not.

```
> prepareTest(file.path(rootdir, "data", "_test2"), smps[2], template = classif,
+     classes = classif)
> ## (now, you can inspect the subdirectories and manually reclassify wrong
> ## items) Let's pretend this is the result...
> test2 <- getTest(file.path(rootdir, "data", "Test sets", "MTLG.2005-05-24.H1"))
> ## Create and study the confusion matrix
> conf4 <- confusion(predict(classif, test2), test2$Class)
> conf4
> # 3110 items classified with 2022 true positives (error rate = 35%) [...]
```

At this stage, one could wonder if this approach is of any use in practice, since all the vignettes still must be visually checked. In fact, it is much faster to perform that validation than to classify all items by hand from scratch in a test set. So, there is still a significant gain to process plankton samples that way, but see the discussion for ideas on further possible improvements.

12.7 Model Deployment

In R, both the "ZITrain" training set and the "ZIClass" classifier can be easily save()d in native ".RData" files and reload()ed in a different session or on another computer. Provided you are ready to manually validate the output of this automatic classification, all the tools are in place in **zooimage** to integrate it in your (plankton) images analysis workflow. Given the inherent limitations in the speed of the preparation and digitization of the plankton samples at the beginning of the workflow, and the even stronger time bottleneck introduced by the manual validation at the end, one can actually process no more than a couple of dozen samples per day. This is already a vast improvement from the manual enumeration (remember that one to two samples could be processed in one day). In this context, all calculations can be easily performed on an average personal computer, and there is no need to deploy the classifier on massively parallel infrastructures, or on cloud computers.

One key point is worth mentioning. In such plankton studies or other community analyses, nobody is interested in the identification of every single particle *per se*. The only interesting outputs are summary statistics at (at least) the sample level, like abundances per class, biomass (in dry weight per water volume) per class, or size distribution of the particles (size spectrum in total, or per class). Hence, no matter if one or the other ROI is perfectly segmented, measured, and classified, the important question is whether one can derive usable statistics from the classification of a few thousands of them. A dedicated function, processSample(), takes the dilution coefficients (see Equation 12.1) of the digitized subsamples into account for such calculations:

```
> dat2 <- zidbDatRead(smps[2]) # Read a ZIDat object
> ## Add manually validated items
```

```
> dat2 <- addClass(dat2, test2)
> ## Add a 'Predicted' column
> dat2 <- predict(classif, dat2, class.only = FALSE)
> ## Calculate abundances per classes
> ## using 'Dil', which is (eq. 12.1):
> subsmp <- attr(dat2, "metadata")$Subsample
> 1/subsmp$SubPart/subsmp$CellPart/subsmp$Replicates/subsmp$VolIni [1] 0.4612 0.3038
> unique(dat2$Dil) # Verification [1] 0.4612 0.3038
> ## Total abundance in ind./m^3
> processSample(dat2)
                          Id    Abd [total]
   1    MTLG.2005-05-24.H1    1127
```

But here, we considered all particles in the calculations, including those that are not zooplankton. Thus, we need to exclude all the classes of particles that do not belong to zooplankton:

```
> zoo <- levels(dat2$Predicted) # All classes
> ## We used first uppercase letters to denote
> ## zooplankton classes, thus:
> zoo <- zoo[grepl("[A-Z]", zoo)] # Zooplankton only
> zoo
   [1]    "Annelida"              "Appendicularia"        "Calanoida"
   [4]    "Chaetognatha"          "Cirripeda"             "Cladocera"
   [7]    "Cnidaria"              "Cyclopoida"            "Decapoda"
  [10]    "Egg - elongated"       "Egg - round"           "Gastropoda"
  [13]    "Harpacticoida"         "Malacostraca other"    "Pisces"
  [16]    "Poecilostomatoida"     "Protista"
> ## Abundance of zooplankton in ind./m^3
> processSample(dat2, keep = zoo)
                      Id    Abd [total]
1    MTLG.2005-05-24.H1    326.8
```

This is the total abundance, but we may be also interested by partial abundances for the most frequent classes (e.g., those where we have, say, at least 50 particles caught in the scans):

```
> lev <- levels(dat2$Class)
> freqs <- table(dat2$Class)
> detail <- zoo[zoo %in% lev[freqs >= 50]]
> detail
[1]    "Calanoida"    "Decapoda"    "Poecilostomatoida"
[4]    "Protista"
```

```
> ## More detailed zooplankton abundances
> processSample(dat2, keep = zoo, detail = detail)
                     Id     Abd Calanoida     Abd Decapoda     Abd Poecilostomatoida
1    MTLG.2005-05-24.H1              111            100.3                     53.94
     Abd Protista     Abd [other]     Abd [total]
1            19.2           42.39           326.8
```

This returns the abundance for the four most frequent zooplankton classes, for the rest ([other]), and for the [total] of all zooplankton. Although these calculations can be done by hand using standard R functions, the code is not trivial (inspect the content of the function processSample() to see how it was done).

Calculation of the biomasses involves the use of an empirically defined conversion function to convert from the area of the particles as measured in the digital images into the dry weight of the organisms, or their carbon content (Alcaraz et al., 2003; Baguley et al., 2004; Di Mauro et al., 2011). Providing you have such relationships available, processSample() is also able to calculate plankton biomasses. As an illustration, we use here the biomass conversion function calculated by Hernandez-Leon and Montero (Hernandez-Leon and Montero, 2006) established for "mesozooplankton" (plankton with size ranging from 0.2 to 20 mm, Equation 12.2).

$$\text{biomass}(\mu\text{g dry weight/ind.}) = 45.7.\text{area}(\text{mm})^{2.38} \tag{12.2}$$

Since our code uses the *ECD*, "equivalent circular diameter" with $area = \pi.\ (ECD/2)^2$, an equivalent function using *ECD* is (Equation 12.3):

$$\text{biomass}(\mu\text{g dry weight/ind.}) = 34.3.\text{ECD}(\text{mm})^{2.38} \tag{12.3}$$

The biomass argument of processSample() expects three parameters for a slightly more general conversion function (Equation 12.4), but we set here $P2 = 0$:

$$\text{biomass} = P1 \cdot \text{ECD}^{\text{P3}} + P2 \tag{12.4}$$

Thus:

```
> processSample(dat2, keep = zoo, detail = detail,
+     biomass = c(P1 = 34.3, P2 =0, P3 = 2.38))
                     Id     Abd Calanoida     Abd Decapoda     Abd Poecilostomatoida
  1  MTLG.2005-05-24.H1              111            100.3                     53.94
     Abd Protista     Abd [other]     Abd [total]     Bio Calanoida     Bio Decapoda
  1           19.2           42.39           326.8              1904             1063
     Bio Poecilostomatoida     Bio Protista     Bio [other]     Bio [total]
  1                 273.9            41.91           571.4            3854
```

This returns both the abundances and the biomasses for our four most abundant zooplankton classes, for the rest ([other]), and for the [total] of all zooplankton. It is hard to compare these results with the literature because ecology of the plankton at Madagascar is still poorly studied (Conway, 2005). However, such results are in the range of observed biomasses in another tropical reef lagoon (Roman et al., 1990). You can also provide a table to the biomass argument, with a separate specification of $P1$, $P2$, and $P3$ for each class, if you have separate relationships available. It looks like this (here default parameters $P1 = 1$, $P2 = 0$, and $P3 = 1$ that obviously need to be changed):

```
> read.delim(system.file("etc", "Conversion.txt",
      + package = "zooimage"), nrows = 3)
            Class    P1    P2    P3
1        Copepoda     1     0     1
2    Cope lateral     1     0     1
3     Cope dorsal     1     0     1
```

Finally, the break argument indicates the size ranges you want to use in the size spectra (in term of the *ECD*). Providing this argument triggers this size spectra calculation:

```
> processSample(dat2, keep = zoo, detail = detail,
+ breaks = seq(0.2, 2, by = 0.2))
                      Id    Abd Calanoida    Abd Decapoda    Abd Poecilostomatoida
1    MTLG.2005-05-24.H1             111           100.3                    53.94
     Abd Protista    Abd [other]    Abd [total]
1            19.2          42.39          326.8
With size spectrum:
$'MTLG.2005-05-24.H1'

                    (0.2,0.4]   (0.4,0.6]   (0.6,0.8]   (0.8,1]   (1,1.2]   (1.2,1.4]
Calanoida              2.4415      37.702      31.466     31.960     6.457      0.9225
Decapoda               0.3038      57.649      37.722      3.229     1.384      0.0000
Poecilostomatoida      9.7439      41.900       2.295      0.000     0.000      0.0000
Protista              17.9688       1.226       0.000      0.000     0.000      0.0000
[other]               11.8705      10.857      14.591      1.845     1.384      0.9225
[total]               42.3285     149.335      86.074     37.034     9.225      1.8449
[total]             (1.4,1.6]   (1.6,1.8]     (1.8,2]

Calanoida                          0.0000                  0.0000                  0
Decapoda                           0.0000                  0.0000                  0
Poecilostomatoida                  0.0000                  0.0000                  0
Protista                           0.0000                  0.0000                  0
[other]                            0.4612                  0.4612                  0
[total]                            0.4612                  0.4612                  0
```

Because you most probably have a series of samples to process, a batch mode of the function, called processSampleAll() is also available:

```
> ## Process all samples located in a directory in batch
> processSampleAll(smpdir, ZIClass = classif, keep = zoo,
>     detail = detail,
>     biomass = c(P1 = 34.3, P2 =0, P3= 2.38),
>     breaks = seq(0.2, 2, by = 0.2))
> ## (Output not displayed here)
```

The result is a "ZIRes" object (ZooImage Results, see Figure 12.2). It inherits from data frame and is thus very easily handled with standard R code for further analyses.

12.8 Lessons, Discussion, and Conclusions

This chapter presents automatic classification of data originating from images. As a concrete example, we used images from plankton samples but a similar approach can be applied on any kind of image data. The use of R to automatically classify plankton raises an opportunity to the oceanographic community to speed up samples analyses. The **zooimage** and **mlearning** packages leverage that potential by providing a series of specialized functions on top of the already large palette of R functions available for data mining. In particular, it integrates the process in a wider workflow, including image acquisition/analysis (not presented here, but see the Zoo/PhytoImage manual (Grosjean and Denis, 2007)), metadata processing, elaboration of training and testing sets, and final processing of the data into sample-based statistics. Thanks to **zooimage**, R was rapidly and largely adopted by a community of plankton biologists and oceanographers who did not necessarily use it before, as demonstrated by the multiple works already published. It could also be used or adapted for other applications (e.g., it has also been used to count bacteria (Gillan et al., 2012)).

In this chapter, we have applied two machine learning algorithms on one example dataset: random forest and support vector machine using linear kernel function. They are among the most efficient algorithms to classify plankton images. They both achieve an error rate close to 20% with 25 groups, that is, 80% of accuracy. The optimization of the obtained classifiers was not the purpose of the chapter, but it is of course possible (see the help pages of corresponding R functions). Selection of discriminant variables is also possible with algorithms like random forest (and this was illustrated by using the decrease in Gini index criterion, see (Torgo, 2010)). The classifier performances highly depend on image quality, image analysis, selected features, the number of classes to discriminate, and the parameters of the algorithms used for the automatic classification of images. In order to show the workflow when using **zooimage**, the performances obtained with the two tested algorithms are compared with other studies in plankton classification. Microscopic images of 23 dinoflagellates species (unicellular phytoplankton) have been classified with 83% of accuracy using a neural network algorithm

(Culverhouse et al., 1996). In this case, the classifier was considered as good as trained taxonomists. Real-time classification of five to seven major planktonic classes with, respectively, 84% and 69% of accuracy using the same learning vector quantization algorithm allows the VPR to provide distribution maps of plankton at sea (Davis et al., 2004). Other algorithms have also been used with VPR images, including support vector machine that reached 72% of accuracy for seven classes (Hu and Davis, 2005). With a dual classifier combining a neural network and support vector machine, more than 90% of accuracy was achieved for the same seven classes (Hu and Davis, 2006). The SIPPER is an experimental device that offers lower resolution images than the VPR, but automatic classification with support vector machine still achieved 76% of accuracy for six classes (Luo et al., 2003). The same algorithm was able to reach 88% of accuracy for five groups using SIPPER II images (Luo et al., 2005). The Imaging FlowCytoBot (Olson and Sosik, 2007) is a prototype which can be compared with the FlowCAM, a device already commercialized (http://www.fluidimaging.com/). A support vector machine classifier allows to discriminate 22 groups of phytoplankton digitized with Imaging FlowCytoBot (Olson and Sosik, 2007) with an accuracy of 88% (Sosik and Olson, 2007). The ZOOSCAN has been used to discriminate 29 classes of zooplankton with 83% accuracy using a combination of algorithms (Grosjean et al., 2004). In another study, images from the ZOOSCAN allowed to classify particles into 20 groups of zooplankton with 79% of accuracy using random forest algorithm (Gorsky et al., 2010). Zoo/PhytoImage coupled with a high-resolution flatbed scanner constitute a system rather similar to the ZOOSCAN (Bachiller and Fernandes, 2011; Bell and Hopcroft, 2008; Gislason and Silva, 2009). With this combination, 63 and 53 groups of mesozooplankton from Alaska have been classified using a random forest algorithm with, respectively, 85% and 88% accuracy (Bell and Hopcroft, 2008). In Iceland, 34 and 25 groups of plankton have been classified with a similar system, still with random forest, with respectively 75% and 82% of accuracy (Gislason and Silva, 2009). Plankton is sometimes stained before being digitized, for better discrimination of living planktonic organisms from dead or mineral particles (Bachiller and Fernandes, 2011; Fernandes et al., 2009; Irigoien et al., 2009). That way, 17 groups of plankton have been classified with 88% accuracy using random forest (Irigoien et al., 2009), and 24 groups were discriminated with 86% accuracy using a naive Bayes algorithm (Fernandes et al., 2009). Phytoplankton from the North Sea digitized with a FlowCAM and analyzed with Zoo/PhytoImage was classified in real-time in 25 groups using random forest with an accuracy of 79% (Lancelot et al., 2012). All these performances are similar to those obtained in the present chapter.

The performances estimated through global accuracy on cross-validated confusion matrix in all these studies do not necessarily reflect real potential when classifying particles from separate samples, as we have shown here. This is partly because (1) new samples are likely to contain particles belonging to taxonomic groups that where ignored or unknown when the training set was built, (2) the shape of the particles within each class can slightly change as a

function of environmental conditions, and (3) we have typically 5-15% of the particles that the taxonomist cannot identify and it inflates the overall misclassification rate. With **mlearning**, prior probabilities can be changed in two places: when training the classifier[11] and in the confusion matrix to test how the composition of a test sample would affect performances of a preexisting classifier. In the context of plankton samples analysis, the second case is most likely to be of interest, and we have illustrated how to do this with **mlearning** and **zooimage**.

In plankton studies and more generally in ecology, scientists are more interested in general statistics such as abundance and biomass or size spectrum by taxonomic or ecologically functional groups than in the identification of every single plankton organism in the samples. This point of view highly diverges from usual machine learning approaches where methods are focused on finding or extracting all cases corresponding to a given criteria (e.g., fraud detection, disease diagnostic) (Torgo, 2010). Binary classifiers are often used there. Several R packages, like **ROCR** (Sing et al., 2009) or **pROC** (Robin et al., 2011), propose tools to analyze such binary classifiers. Plankton classification is typically a multiclass problem where an important number of classes have to be simultaneously discriminated. For such a purpose, diagnostic tools like ROC curves are not convenient because they calculate all possible one-on-one or one-on-all curves (Hand and Till, 2001). The resulting complexity is difficult to manage and interpret. To our knowledge, only one multiclass ROC curve statistic is available in R.[12] In the **zooimage** and **mlearning** packages, the confusion matrix is promoted as a central tool to analyze classifier performances. Despite its apparent simplicity and its identified shortcomings (it is dependent on priors, costs, and thresholds), this double-entry contingency table is convenient in the case of multiclass problems and allows us to rapidly spot where the highest error rate is located or to extract several statistics for each class or for the whole classifier (recall, precision, F-score, etc.). In addition, the **zooimage** package proposes different graphical representations which ease the analysis of classifier performances or to compare two classifiers. Statistical correction of the error can be applied, using the information contained in the confusion matrix, to better estimate abundance or biomass by classes (Hu and Davis, 2006; Solow et al., 2001). This certainly requires further investigations.

Another field of research is the optimal definition of the plankton classes to include in the training set. It is not clear yet at which taxonomical level the analysis of plankton communities is the most efficient, considering sampling effort and scarcity of available data. In other words,

[11] See, for instance, the sampsize argument for mlRforest() or classwt argument for mlSvm().

[12] The multiclass .roc() function from pROC package calculates AUC (area under the curve) for a multiclass classifier. But that function was not able to perform its computation with the 25 classes used in our sample (too much memory was required).

the classes themselves are not completely fixed. The hierarchical representation of the classes in **zooimage** raises the opportunity to study plankton samples at different taxonomic levels quite easily. The simplification of training sets by pooling some of the initial classes has been shown as a viable approach in a quest for the best trade-off between higher taxonomic separation and error rate minimization. Fernandes et al. (Fernandes et al., 2009) have proposed an iterative optimization algorithm to obtain such a trade-off. Further research, using the hierarchical organization of the classes in **zooimage**, would be certainly useful for future improvements.

Many packages available in CRAN (http://cran.r-project.org) provide supervised classification tools, but inevitably, the user interface differs slightly among them. In particular, the way they handle missing data or empty classes in the training set is diverging widely. The **zooimage** package proposes a standardized workflow based mainly on the unified interface offered in the **mlearning** package. All steps from data importation to automatic classification have been smoothed through dedicated R objects which embed the required information. Metadata follow objects all along the process. This is a key feature when maximal traceability is required.

Despite evident limitations, the automatic classification of biological specimens based on image analysis and supervised classification is actually considered as a possible way of automation in computer-assisted taxonomy (MacLeod et al., 2010). Recent studies have already used this approach to map the distribution of major plankton groups (Irigoien et al., 2009), to analyze time series of weekly samples at fixed stations (Gorsky et al., 2010) or to get estimation of plankton distributions in real-time aboard oceanographic ships (Lancelot et al., 2012). The TARA expedition was a recent worldwide campaign to study marine biota using plankton digitization devices (http://oceans.taraexpeditions.org). Millions of plankton images were gathered, and this huge amount of data must now be compiled, which is not possible without automatic, or at least, semiautomatic techniques. New methods are under study (Chang et al., 2012; Fernandes et al., 2009; Ye et al., 2011) to optimize performances in such multiclass problems. Together with the development of better image analysis and measurement algorithms and the elaboration of specialized machine learning algorithms, statistical error correction tools would perhaps allow us to get rid of the manual validation step in the future. This would turn that approach into a truly fast and automated way of processing plankton samples.

Acknowledgments

This work was partly supported by the Belgian Science Policy in the framework of the AMORE III project [SD/NS/03A], by IFREMER and by UMONS. The example dataset comes from the digitization of plankton samples collected at Madagascar by Prof. Igor Eeckhaut (BOMB laboratory, “Biologie des Organismes Marins et Biomimetisme” at UMONS).

References

Alcaraz, M., Saiz, E., Calbet, A., Trepat, I., Broglio, E., 2003. Estimating zooplankton biomass through image analysis. Mar. Biol. 143, 307–315.

Bachiller, E., Fernandes, J.A., 2011. Zooplankton image analysis manual: automated identification by means of scanner and digital camera as imaging devices. Rev. Invest. Mar. 18 (2), 16–37.

Baguley, J.G., Hyde, L.J., Montagna, P.A., 2004. A semi-automated digital microphotographic approach to measure meiofaunal biomass. Limnol. Oceanogr. Methods. 2, 181–190.

Bell, J.L., Hopcroft, R.R., 2008. Assessment of ZooImage as a tool for the classification of zooplankton. J. Plankton Res. 30 (12), 1351–1367.

Benfield, M.C., Grosjean, P., Culverhouse, P.F., Irigoien, X., Sieracki, M.E., Lopez-Urrutia, A., Dam, H.G., Hu, Q., Davis, C.S., Hansen, A., Pilskaln, C.H., Riseman, E.M., Schultz, H., Utgoff, P.E., Gorsky, G., 2007. RAPID: research on automated plankton identification. Oceanography. 20 (2), 172–187.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Chang, C.-Y., Ho, P.-C., Sastri, A.R., Lee, Y.-C., Gong, G.-C., Hsieh, C.-h., 2012. Methods of training set construction: towards improving performance for automated mesozooplankton image classification systems. Cont. Shelf Res. 36, 19–28.

Conway, D.V.P., 2005. Island-coastal and oceanic epipelagic zooplankton biodiversity in the southwestern Indian Ocean. Indian J.Mar. Sci. 34 (1), 50–56.

Core Team, R., 2012. R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Culverhouse, P., Simpson, R., Ellise, R., Lindley, J., Williams, R., Parisini, T., Reguera, B., Bravo, I., Zoppoli, R., Earnshaw, G., McCall, H., Smith, G., 1996. Automatic classification of field-collected dinoflagellates by artificial neural network. Mar. Ecol. Prog. Ser. 139, 281–287.

Culverhouse, P.F., Williams, R., Reguera, B., Herry, V., Gonzalez-Gil, S., 2003. Do experts make mistakes? A comparison of human and machine identification of dinoflagellates. Mar. Ecol. Prog. Ser. 247, 17–25.

Culverhouse, P.F., Williams, R., Benfield, M., Flood, P.R., Sell, A.F., Mazzocchi, M.G., Buttino, I., Sieracki, M., 2006. Automatic image analysis of plankton: future perspectives. Mar. Ecol. Prog. Ser. 312, 297–309.

Davis, C., Gallager, S., Berman, M., Haury, L., Strickler, J., 1992. The Video Plankton Recorder (VPR): design and initial results. Arch. Hydrobiol. 36, 67–81.

Davis, C., Hu, Q., Gallager, S., Tang, X., Ashjian, C., 2004. Real-time observation of taxa-specific plankton distributions: an optical sampling method. Mar. Ecol. Prog. Ser. 284, 77–96.

Di Mauro, R., Cepeda, G., Capitanio, F., Vinas, M., 2011. Using ZooImage automated system for the estimation of biovolume of copepods from the northern Argentine Sea. J. Sea Res. 66 (2), 69–75.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A., 2011. e1071: Misc functions of the Department of Statistics (e1071), TU Wien. R package version 1.6.

Drummond, C., Holte, R.C., 2006. Cost curves: an improved method for visualizing classifier performance. Mach. Learn. 65 (1), 95–130.

Embleton, K.V., Gibson, C.E., Heaney, S.I., 2003. Automated counting of phytoplankton by pattern recognition: a comparison with a manual counting method. J. Plankton Res. 25 (6), 669–681.

Fernandes, J.A., Irigoien, X., Boyra, G., Lozano, J.A., Inza, I., 2009. Optimizing the number of classes in automated zooplankton classification. J. Plankton Res. 31 (1), 19–29.

Gaston, K.J., O'Neill, M.A., 2004. Automated species identification: why not? Philos. Trans. Roy. Soc. Lond. Ser. B Biol. Sci. 359, 655–667.

Gillan, D.C., Baeyens, W., Bechara, R., Billon, G., Denis, K., Grosjean, P., Leermakers, M., Lesven, L., Pede, A., Sabbe, K., Gao, Y., 2012. Links between bacterial communities in marine sediments and trace metal geochemistry as measured by *in situ* DET/DGT approaches. Mar. Pollut. Bull. 64, 353–362.

Gislason, A., Silva, T., 2009. Comparison between automated analysis of zooplankton using ZooImage and traditional methodology. J. Plankton Res. 31 (12), 1505–1516.

Gorsky, G., Ohman, M.D., Picheral, M., Gasparini, S., Stemmann, L., Romagnan, J.-B., Cawood, A., Pesant, S., Garcia-Comas, C., Prejger, F., 2010. Digital zooplankton image analysis using the ZOOSCAN integrated system. J. Plankton Res. 32 (3), 285–303.

Graneli, E., Turner, J., 2006. An introduction to harmful algae. In: Graneli, E., Turner, T. (Eds.), Ecology of Harmful Algae, Ecological Studies. Springer-Verlag Edition, Berlin, pp. 3–8.

Grosjean, P., Denis, K., 2007. Zoo/PhytoImage user's manual. 57pp, http://www.sciviews.org/zooimage/docs/ZooPhytoImageManual.pdf.

Grosjean, P., Picheral, M., Warembourg, C., Gorsky, G., 2004. Enumeration, measurement, and identification of net zooplankton samples using the ZOOSCAN digital imaging system. ICES J. Mar. Sci. 61 (4), 518–525.

Hanczar, B., Hua, J., Sima, C., Weinstein, J., Bittner, M., Dougherty, E.R., 2010. Small-sample precision of ROC-related estimates. Bioinformatics. 26 (6), 822–830.

Hand, D.J., 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. Mach. Learn. 77 (1), 103–123.

Hand, D.J., Till, R.J., 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach. Learn. 45, 171–186.

Hays, G.C., Richardson, A.J., Robinson, C., 2005. Climate change and marine plankton. Trends Ecol. Evol. 20 (6), 337–344.

Hernandez-Leon, S., Montero, I., 2006. Zooplankton biomass estimated from digitalized images in Antarctic waters: a calibration exercise. J. Geophys. Res. 111 (C5), 2156–2202.

Hornik, K., Buchta, C., Zeileis, A., 2009. Open-source machine learning: R meets weka. Comput. Stat. 24 (2), 225–232.

Hu, Q., Davis, C., 2005. Automatic plankton image recognition with co-occurrence matrices and support vector machine. Mar. Ecol. Prog. Ser. 295, 21–31.

Hu, Q., Davis, C., 2006. Accurate automatic quantification of taxa-specific plankton abundance using dual classification with correction. Mar. Ecol. Prog. Ser. 306, 51–61.

Irigoien, X., Fernandes, J.A., Grosjean, P., Denis, K., Albaina, A., Santos, M., 2009. Spring zooplankton distribution in the Bay of Biscay from 1998 to 2006 in relation with anchovy recruitment. J. Plankton Res. 31 (1), 1–17.

Lancelot, C., Rousseau, V., Lacroix, G., Denis, K., Gypens, N., Grosjean, P., Van Nieuwenhove, K., Desmit, X., Parent, J.-Y., Lillo, N.T., Ruddick, K., Delbare, D., 2012. Combined effect of changing hydroclimate and human activity on coastal ecosystem health-AMORE III. Final Report. Brussels: Belgian Science Policy Office 2012, 56 p.

Lenz, J., 2000. Introduction. In: Harris, R., Wiebe, P., Lenz, J., Skjoldal, H., Huntley, M. (Eds.), Zooplankton Methodology Manual. Academic Press, London, pp. 1–30.

Liaw, A., Wiener, M., 2002. Classification and regression by randomforest. R News. 2 (3), 18–22.

Luo, T., Kramer, K., Samson, S., Remsen, A., Goldgof, D., Hall, L.O., Hopkins, T., 2003. Learning to recognize plankton. In: IEEE International Conf. on Systems, Man and Cyberneticspp. 888–893.

Luo, T., Kramer, K., Goldgof, D., Hall, L.O., Samson, S., Remsen, A., Hopkins, T., 2005. Active learning to recognize multiple types of plankton. J. Mach. Learn. Res. 6, 589–613.

MacLeod, N., Benfield, M., Culverhouse, P., 2010. Time to automate identification. Nature. 467 (7312), 154–155.

Manriquez, K., Escribano, R., Hidalgo, P., 2009. The influence of coastal upwelling on the mesozooplankton community structure in the coastal zone off Central/Southern Chile as assessed by automated image analysis. J. Plankton Res. 31 (9), 1075–1088.

Miller, C., 2004. The spring phytoplankton bloom. In: Miller, C. (Ed.), Biological Oceanography. Blackwell Publishing company, Oxford, pp. 1–19.

Olson, R.J., Sosik, H.M., 2007. A submersible imaging-in-flow instrument to analyze nano and microplankton: imaging FlowCytobot. Limnol. Oceanogr. Methods. 5, 195–203.

Peng, R.D., 2006. Interacting with data using the filehash package. R News. 6 (4), 19–24.

Peters, A., Hothorn, T., 2012. ipred: Improved predictors. R package version 0.8–13.

Reynolds, C., 2006. Phytoplankton. In: Usher, M., Saunders, D., Peet, R., Dobson, A. (Eds.), Ecology of Phytoplankton. Cambridge edition, Cambridge, pp. 1–36.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C., Muller, M., 2011. pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinform. 12, 77.

Roman, M., Furnas, M., Mullin, M., 1990. Zooplankton abundance and grazing at Davies Reef, Great Barrier Reef, Australia. Mar. Biol. 105, 73–82.

Schneider, C.A., Wayne, S.R., Eliceiri, K.W., 2012. NIH Image to ImageJ: 25 years of image analysis. Nat. Methods. 9, 671–675.

Sieracki, M.E., Johnson, P.W., Sieburth, J.M., 1985. Detection, enumeration, and sizing of planktonic bacteria by image-analyzed epifluorescence microscopy. Appl. Environ. Microbiol. 49 (4), 799–810.

Sieracki, C., Sieracki, M., Yentsch, C., 1998. An imaging-in-flow system for automated analysis of marine microplankton. Mar. Ecol. Prog. Ser. 168, 285–296.

Sieracki, M.E., Benfield, M., Hanson, A., Davis, C., Pilskaln, C.H., Sosik, H.M., Ashjian, C., Culverhouse, P., Cowen, R., Lopes, R., Balch, W., Irigoien, X., 2010. Optical plankton imaging and analysis systems for ocean observation. Proc. OceanObs'09 Sust. Ocean Observ. Inform. Soc. 2 (1), 21–25.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T., 2009. ROCR: visualizing the performance of scoring classifiers. R package version 1.0–4.

Solow, A., Davis, C., Hu, Q., 2001. Estimating the taxonomic composition of a sample when individuals are classified with error. Mar. Ecol. Prog. Ser. 216 (2), 309–311.

Sosik, H.M., Olson, R.J., 2007. Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. Limnol. Oceanogr. Methods. 5, 204–216.

Tang, X., Stewart, W.K., Vincent, L.U.C., Huang, H.E., Marra, M., Gallager, S.M., Davis, C.S., 1998. Automatic plankton image recognition. Artif. Intell. Rev. 12, 177–199.

Torgo, L., 2010. Data Mining with R: Learning with Case Studies, first ed. CRC Press, Boca Raton.

Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S, fourth ed. Springer, New York. ISBN 0-387-95457-0.

Vuk, M., Curk, T., 2006. ROC curve, lift chart and calibration plot. Metodoloski zvezki. 3 (1), 89–108.

Wiebe, P., Benfield, M.C., 2003. From the Hensen net toward four-dimensional biological oceanography. Prog. Oceanogr. 56 (1), 7–136.

Ye, L., Chang, C., Hsieh, C., 2011. Bayesian model for semi-automated zooplankton classification with predictive confidence and rapid category aggregation. Mar. Ecol. Prog. Ser. 441, 185–196.

Zarauz, L., Irigoien, X., Fernandes, J.A., 2008. Modelling the influence of abiotic and biotic factors on plankton distribution in the Bay of Biscay, during three consecutive years (20042006). J. Plankton Res. 30 (8), 857–872.

Zarauz, L., Irigoien, X., Fernandes, J.A., 2009. Changes in plankton size structure and composition, during the generation of a phytoplankton bloom, in the central Cantabrian Sea. J. Plankton Res. 31 (2), 193–207.

CHAPTER 13

Crime Analyses Using R

Anindya Sengupta*, **Madhav Kumar***, **Shreyes Upadhyay**†

**Fractal Analytics, India, †Diamond Management and Technology Consultants, India*

13.1 Introduction

The past couple of years have witnessed an overall declining trend in crime rate in the United States. This, in part, is attributable to the improvement in law enforcement strategies especially the inclusion of computer-aided technology for effective and efficient deployment of police resources. These advancements have been complemented by the availability of a vast amount of data and the capability to handle them. Many police departments have turned to data science to translate these bytes into actionable insights. Ranging from trend reports and correlation analyses to aggregated behavioral modeling, the developments in the field of crime analysis have paved the way for predictive policing and strategic foresight (Figure 13.1).

Predictive policing is an upcoming and growing area of research where statistical techniques are used to identify criminal hot-spots dynamically in order to facilitate anticipatory and precautionary deployment of police resources. However, the efforts that go into designing an effective predictive policing strategy involve a series of challenges. The most pertinent of these, especially concerning statisticians and analysts, is the one relating to data. How does one gather, process, and harness the copious real-time data? How does one develop a predictive engine that is simple enough to be understood and at the same time accurate enough to be useful? How does one implement a solution that gives stable results and yet updates dynamically? These are some of the concerns that we address in this chapter. We look at how crime data are stored, how they need to be handled, the different dimensions involved, the kind of techniques that would be applicable, and the limitations of such analysis. We look at all this, within the central theme of the powerful statistical programming language R.

The chapter is organized into eight sections including Section 13.1 After defining the problem in Section 13.2, we dive straight into the data through R with data extraction in Section 13.3. Data exploration and preprocessing in Section 13.4 provide details on how to deal with crime data in R and how to clean and process them for modeling. We visualize the data in Section 13.5 to get a better understanding of what we are dealing with and how we can reorganize it to get optimal results through modeling. In Sections 13.6 and 13.7, we build a

Data Mining Applications with R

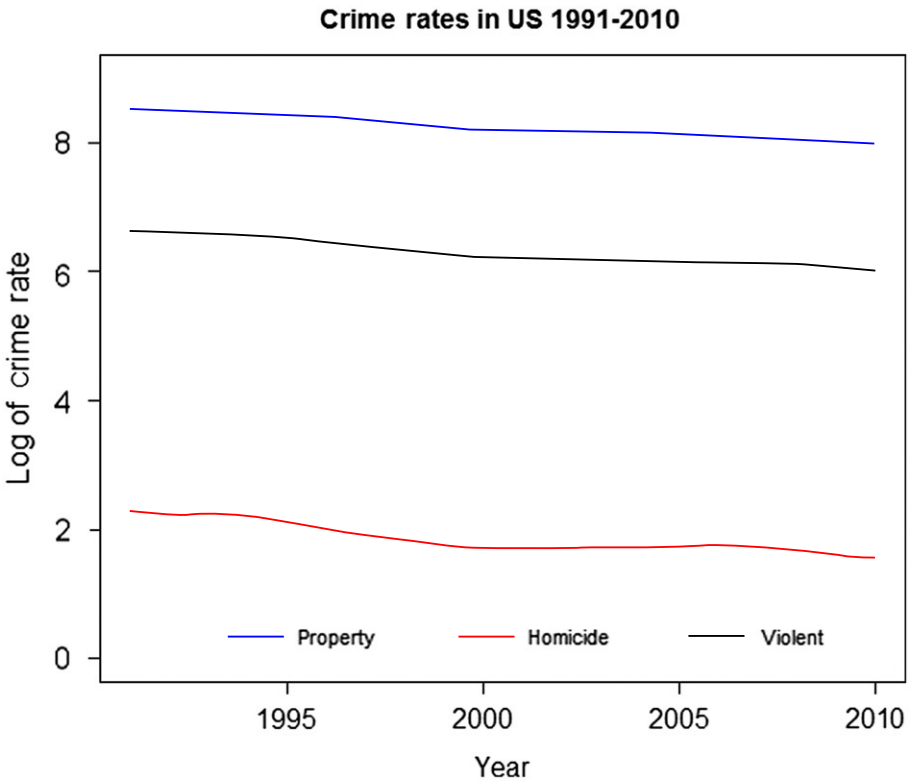


Figure 13.1
U.S. crime rates over time.

multivariate regression model to predict crime counts using historical crime data for the city of Chicago and evaluate its performance. Section 13.8 closes the chapter with discussions on the limitations of the model and the scope for improvement.

13.2 Problem Definition

Predictive policing is a multidimensional optimization problem where law enforcement agencies try to efficiently utilize a scarce resource to minimize instances of crime over time and across geographies. But how do we optimize? This is precisely what we answer in this chapter by using real and publicly available criminal data of Chicago. Using statistics and computer-aided technologies, we try to devise a solution for this optimization problem. In the attempt to address a real world problem, our primary focus here is on the fundamentals of data rather than on acrobatics with techniques.

Crime analysis includes looking at the data from two different dimensions—spatial and temporal. Spatial dimension involves observing the characteristics of a particular region along

with its neighbors. Temporal dimension involves observing the characteristics of a particular region over time. The question then is how far away, from the epicenter, do we look for similar patterns and how far back in time, from the date of event, do we go to capture the trend. Ideally we would like to have as much data as possible. But we don't. And that makes data science a creative process. A process bound by mathematical logic and centered around statistical validity.

Crime data are not easy to deal with. With both spatial and temporal attributes, processing them can be a challenging task. The challenge is not limited to handling spatial and temporal data but also deriving information from them at these levels. Any predictive model for crime will have these two dimensions attached to it. And to make an effort toward effective predictive policing strategies, this inherent structure of the data needs to be leveraged.

13.3 Data Extraction

For this exercise, we use the crime data for the city of Chicago which are available from 2001 onwards on the city's open data portal. To make analysis manageable, we utilize the past one year of data from the current date.

R has the capability of reading files and data tables directly from the Web. We can do this by specifying the connection string instead of the file name in the read.csv() function.[1]

```
> url.data <- "https://data.cityofchicago.org/api/views/x2n5-
8w5q/rows.csv?accessType=DOWNLOAD"

> crime.data <- read.csv(url.data, na.strings= '')
```

13.4 Data Exploration and Preprocessing

Before we start playing with data, it is important to understand how the data are organized, what fields are present in the table, and how they are stored. str() is a useful command for this which displays the internal structure of the data neatly.

```
> str(crime.data)
'data.frame': 337793 obs. of 17 variables:
$ CASE.                    : Factor w/ 337775 levels "","223432","C431426",..:
64 71 5316 109 66 95 1526 133 65 2534 ...
$ DATE..OF.OCCURRENCE      : Factor w/ 121027 levels "1/1/2012 0:00",..: 71649
71650 71650 71651 71652 71652 71652 71653 71653 71654 ...
$ BLOCK                    : Factor w/ 28472 levels "0000X E 100TH PL",..: 3951
12626 16621 5479 889 16317 7765 11248 5132 7765 ...
$ IUCR                     : Factor w/ 323 levels "031A","031B",..: 301 307 23
173 291 192 25 226 45 23 ...
```

[1] The url can also be directly fed as input to read.csv() as `crime.data <- read.csv(https://data.cityofchicago.org/api/views/x2n5-8w5q/rows.csv?accessType=DOWNLOAD)`.

```
$ PRIMARY.DESCRIPTION    : Factor w/ 31 levels "ARSON","ASSAULT",..: 30 30 8 5
2 26 8 22 6 8 ...
$ SECONDARY.DESCRIPTION  : Factor w/ 303 levels "$500 AND UNDER",..: 1 240 96
230 252 180 145 299 277 96 ...
$ LOCATION.DESCRIPTION   : Factor w/ 117 levels "","ABANDONED BUILDING",..:
108 106 19 69 88 105 22 94 113 24 ...
$ ARREST                 : Factor w/ 2 levels "N","Y": 2 2 1 1 2 1 1 1 1 1 ...
$ DOMESTIC               : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 2 2 1 ...
$ BEAT                   : int 1322 1924 1922 1433 1832 1131 1922 2523 1011
1922 ...
$ WARD                   : int 32 44 47 1 42 24 47 31 24 47 ...
$ FBI.CD                 : Factor w/ 26 levels "01A","01B","04A",..: 24 24 8
17 5 20 8 21 11 8 ...
$ X.COORDINATE           : int 1163854 1169774 1165297 1164854 1174827
1147597 1163099 1145928 1151256 1163099 ...
$ Y.COORDINATE           : int 1906393 1921705 1930328 1909126 1905650
1897631 1931867 1918688 1893861 1931867 ...
$ LATITUDE               : num 41.9 41.9 42 41.9 41.9 ...
$ LONGITUDE              : num -87.7 -87.7 -87.7 -87.7 -87.6 ...
$ LOCATION               : Factor w/ 182456 levels "","(41.64471369097245, -
87.6128856478878)",..: 125420 155666 167007 132060 123200 104650 169311
150477 98599 169311 ...
```

str() is a compact version of summary(), which provides a detailed summary of the data.

```
> summary(crime.data)
     CASE.             DATE..OF.OCCURRENCE                BLOCK
HT572234:      3   1/1/2012 0:01 :     68    008XX  N  MICHIGAN AVE:      701
HT576362:      3   9/1/2011 9:00 :     62    001XX  N  STATE ST     :      582
HV217424:      3   12/1/2011 9:00:     50    008XX  N  STATE ST     :      519
HT341462:      2   3/1/2012 9:00 :     50    0000X  W  TERMINAL ST  :      484
HT387816:      2   10/1/2011 9:00:     47    076XX  S  CICERO AVE   :      470
HT421725:      2   8/1/2011 9:00 :     47    0000X  N  STATE ST     :      444
(Other) :337778   (Other)        :337469    (Other)                 :334593
   IUCR             PRIMARY.DESCRIPTION
486     : 29717   THEFT           :72233
820     : 28160   BATTERY         :59892
460     : 20321   NARCOTICS       :37856
1811    : 20151   CRIMINAL DAMAGE :36023
1320    : 16505   BURGLARY        :24827
610     : 16462   ASSAULT         :19492
(Other):206477   (Other)         :87470
LOCATION.DESCRIPTION                        ARREST          DOMESTIC
STREET                          : 79864     N:245198        N:289939
RESIDENCE                       : 53826     Y: 92595        Y: 47854
SIDEWALK                        : 41390
APARTMENT                       : 40530
OTHER                           : 11249
PARKING LOT/GARAGE(NON.RESID.): 10370
(Other)                         :100564
     BEAT            WARD         FBI.CD        X.COORDINATE
Min.   : 111    Min.   : 1.00   6     :72233    Min.   :1094469
1st Qu.: 622    1st Qu. :10.00   08B   :51842    1st Qu.:1152885
```

```
Median :1032    Median :22.00   14    :36023    Median :1165987
Mean   :1188    Mean   :22.64   18    :35958    Mean   :1164633
3rd Qu.:1724    3rd Qu.:34.00   26    :30328    3rd Qu.:1176390
Max.   :2535    Max.   :50.00   5     :24827    Max.   :1205078
                  NA's :13    (Other):86582    NA's   :331
  Y.COORDINATE        LATITUDE      LONGITUDE
Min.   :1813949  Min.   :41.64    Min.   :-87.93
1st Qu.:1858248  1st Qu.:41.77    1st Qu.:-87.71
Median :1889370  Median :41.85    Median :-87.67
Mean   :1884885  Mean   :41.84    Mean   :-87.67
3rd Qu.:1908824  3rd Qu.:41.91    3rd Qu.:-87.63
Max.   :1951532  Max.   :42.02    Max.   :-87.52
NA's   :331      NA's   :331      NA's   :331
                                     LOCATION
(41.896757773203376, -87.62835155256259): 490
(41.754641620170695, -87.74138515773002): 464
(41.883559699067106, -87.62773649602941): 357
                                          : 331
(41.978893800537726, -87.90646758438514): 269
(41.909763298422995, -87.74331203591734): 266
(Other)                                   :335616
```

The data are stored at a crime incident level, that is, there is one observation for each crime incident in the data table. Each incident has a unique identifier associated with it which is stored in the CASE. variable. By definition then, CASE. should have all unique values. However, we see that some instances are duplicated, i.e., there are two or more rows which have the same case value. For example, there are three rows in the data that have a case value equal to HT572234. These duplicated rows need to be removed. We can do this using a combination of the subset() and the duplicated()[2] function.[3]

```
> crime.data <- subset(crime.data, !duplicated(crime.data$CASE.))

> summary(crime.data)
```

Most raw data sets will have issues like duplicated rows, missing values, incorrectly imputed values, and outliers. This is the case with our data as well with some of the variables having missing values.

Missing value imputation is a critical ingredient to any modeling recipe. Depending on the meaning and type of the variable, the missing values need to be substituted logically. There are certain cases, however, where missing value imputation does not apply. For example, the longitude and latitude variables in our data represent the coordinates of the location where the crime incident occurred. We cannot substitute these values using simple mathematical logic. In such a scenario, depending also on the percentage of rows with missing values, we can ignore these observations. A neat function to deal with rows which have missing values is the is.na function.

[2] A negation of a command can be used by prefixing it with an exclamation mark.

[3] Output has been suppressed due to space constraints.

```
> crime.data <- subset(crime.data, !is.na(crime.data$LATITUDE))

> crime.data <- subset(crime.data, !is.na(crime.data$WARD))
```

Another way to remove observations with missing rows from the data is to explicitly find the ones that have missing values and then remove those from the data frame. We can use the which() function to determine the rows with the missing values.

```
> which(is.na(crime.data$LOCATION))

> crime.data <- crime.data[-which(is.na(crime.data$LOCATION)), ]
```

In our data, we also have illogical values in certain rows. For example, one of the values in the CASE. variable is "CASE#." This seems to be some sort of a data storage or data import issue and we can safely ignore these rows.

```
> crime.data <- crime.data[crime.data$CASE. != 'CASE#',]
```

The date of occurrence gives an approximate date and time stamp as to when the crime incident might have happened. To see how this variable is stored, we can use the head() function which shows the first few observations of the data/column.

```
> head(crime.data$DATE..OF.OCCURRENCE)
[1] 5/22/2011 18:03 5/22/2011 18:05 5/22/2011 18:05 5/22/2011 18:06 5/22/2011 18:10 5/22/2011
18:10
121027 Levels: 1/1/2012 0:00 1/1/2012 0:01 1/1/2012 0:02 1/1/2012 0:03 1/1/2012 0:05 1/1/2012
0:06 ... 9/9/2011 9:58
```

Currently, date is stored as a factor variable. To make R recognize that it is in fact a date, we need to present it to R as a date object. One way to do this is using the as.POSIXlt() function.[4]

```
> crime.data$date <- as.POSIXlt(crime.data$DATE..OF.OCCURRENCE,
            format= "%m/%d/%Y %H:%M")

> head(crime.data$date)
[1] "2011-05-22 18:03:00" "2011-05-22 18:05:00" "2011-05-22 18:05:00" "2011-05-22 18:06:00"
"2011-05-22 18:10:00"
[6] "2011-05-22 18:10:00"
```

R can now understand that the data stored in the column are date and time stamps. Processing the data a bit further, we can separate the time stamps from the date part using the times() function in the chron (James and Hornik, 2011) library. The chron library provides a bunch of useful functions to handle dates and time in R.

```
> library(chron)

> crime.data$time <- times(format(crime.data$date, "%H:%M:%S"))

> head(crime.data$time)

[1] 18:03:00 18:05:00 18:05:00 18:06:00 18:10:00 18:10:00
```

[4] During our exercise, we downloaded the data several times and noticed inconsistency in the format in which time was stored. In case, time appears as 05:30 PM instead of 17:30, please use

```
> crime.data$date <- as.POSIXlt(crime.data$DATE..OF.OCCURRENCE,
            format= "%m/%d/%Y %I:%M %p").
```

The frequency of crimes need not be consistent throughout the day. There could be certain time intervals of the day where criminal activity is more prevalent as compared to other intervals. To check this, we can bucket the timestamps into a few categories and then see the distribution across the buckets. As an example, we can create four six-hour time windows beginning at midnight to bucket the time stamps. The four time intervals we then get are—midnight to 6AM, 6AM to noon, noon to 6PM, and 6PM to midnight.[5]

For bucketing, we first create variable bins using the four time intervals mentioned above. Once the bins are created, the next step is to map each timestamp in the data to one of these time intervals. This can be done using the cut() function.

```
> time.tag <- chron(times = c("00:00:00", "06:00:00", "12:00:00", "18:00:00",
            "23:59:00"))

> time.tag
[1] 00:00:00 06:00:00 12:00:00 18:00:00 23:59:00

> crime.data$time.tag <- cut(crime.data$time, breaks = time.tag,
             labels = c("00-06","06-12", "12-18", "18-00"), include.lowest =
             TRUE)

> table(crime.data$time.tag)
00-06   06-12   12-18   18-00
56161   76224  103336  101723
```

The distribution of crime incidents across the day suggests that crimes are more frequent during the latter half of the day.

With our timestamps in order and stored in a separate variable, we can recode the date variable to contain just the date part by stripping the timestamps.

```
> crime.data$date <- as.POSIXlt(strptime(crime.data$date,
             format = "%Y-%m-%d"))

> head(crime.data$date)
[1] "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22"
```

One of the core aspects of data mining is deriving increasingly more information from the limited data we have. We will see a few examples of what we mean by this as we go along. Let's start with something simple and intuitive.

We can use the date of incidence to determine which day of the week and which month of the year the crime occurred. It is possible that there is a pattern in the way crimes occur (or are committed) depending on the day of the week and month.

```
> crime.data$day <- weekdays(crime.data$date, abbreviate = TRUE)

> crime.data$month <- months(crime.data$date, abbreviate = TRUE)
```

[5] There is no theoretical reasoning behind choosing four 6-h buckets. This is just indicative of how to look for patterns in the data.

There are two fields in the data which provide the description of the crime incident. The first, primary description provides a broad category of the crime type and the second provides more detailed information about the first. We use the primary description to categorize different crime types.[6]

```
> table(crime.data$PRIMARY.DESCRIPTION)
```

```
                ARSON                       ASSAULT                       BATTERY
                  465                         19484                         59855
             BURGLARY                 CRIM SEXUAL ASSAULT               CRIMINAL DAMAGE
                24807                          1295                         35995
   CRIMINAL TRESPASS                 DECEPTIVE PRACTICE                      GAMBLING
                 8338                         11772                           759
            HOMICIDE     INTERFERE  WITH  PUBLIC  OFFICER   INTERFERENCE WITH PUBLIC OFFICER
                  474                           168                          1022
        INTIMIDATION                      KIDNAPPING               LIQUOR LAW VIOLATION
                  161                           245                           618
MOTOR VEHICLE THEFT                       NARCOTICS                     NON-CRIMINAL
                17704                         37828                             2
          OBSCENITY          OFFENSE INVOLVING CHILDREN          OTHER NARCOTIC VIOLATION
                   29                          2095                             5
      OTHER OFFENSE                      OTHER OFFENSE                     PROSTITUTION
                18323                             2                          2449
   PUBLIC INDECENCY              PUBLIC PEACE VIOLATION                    ROBBERY
                   16                          2957                         13389
        SEX OFFENSE                       STALKING                          THEFT
                  994                           186                         72115
  WEAPONS VIOLATION
                 3892
```

```
> length(unique(crime.data$PRIMARY.DESCRIPTION))
[1] 31
```

The data contain about 31 crime types, not all of which are mutually exclusive. We can combine two or more similar categories into one to reduce this number and make the analysis a bit easier.[7]

```
> crime.data$crime <- as.character(crime.data$PRIMARY.DESCRIPTION)

> crime.data$crime <- ifelse(crime.data$crime %in% c("CRIM SEXUAL ASSAULT",
            "PROSTITUTION", "SEX OFFENSE"), 'SEX', crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("MOTOR VEHICLE THEFT"),
            "MVT", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("GAMBLING", "INTERFERE
```

[6] At the time the data were downloaded, there was an issue with one row of data set which had primary description set to " Primary." We can remove the row using `> crime.data <- crime.data[crime.data$PRIMARY.DESCRIPTION != " PRIMARY",]`

[7] The statements below can be written as one ifelse() statement. They have been broken out to improve readability.

```
            WITH PUBLIC OFFICER", "INTERFERENCE WITH PUBLIC OFFICER", "INTIMIDATION",
            "LIQUOR LAW VIOLATION", "OBSCENITY", "NON-CRIMINAL", "PUBLIC PEACE VIOLATION",
            "PUBLIC INDECENCY", "STALKING", "NON-CRIMINAL (SUBJECT SPECIFIED)"),
            "NONVIO", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "CRIMINAL DAMAGE", "DAMAGE",
            crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "CRIMINAL TRESPASS",
            "TRESPASS", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("NARCOTICS", "OTHER
            NARCOTIC VIOLATION", "OTHER NARCOTIC VIOLATION"), "DRUG", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "DECEPTIVE PRACTICE",
            "FRAUD", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("OTHER OFFENSE", "OTHER
            OFFENSE"), "OTHER", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("KIDNAPPING", "WEAPONS
            VIOLATION", "OFFENSE INVOLVING CHILDREN"), "VIO", crime.data$crime)

> table(crime.data$crime)
ARSON    ASSAULT    BATTERY    BURGLARY     DAMAGE       DRUG      FRAUD    HOMICIDE
  465      19484      59855      24807      35995      37833      11772        474
  MVT    NONVIO      OTHER    ROBBERY        SEX      THEFT   TRESPASS        VIO
17704       5918      18325      13389       4738      72115       8338       6232
```

In addition, we also have data on whether the crime incident led to an arrest or not. This is currently stored with Yes/No inputs. For now, let us convert this to a binary numeric variable. We will come back to it later and see how we can use it.

```
> crime.data$ARREST <- ifelse(as.character(crime.data$ARREST) == "Y", 1, 0)
```

With a couple of basic variables in place, we can start with a few visualizations to see how, when, and where are the crime incidents occurring.

13.5 Visualizations

Visualizing data is a powerful way to derive high-level insights about the underlying patterns in the data. Visualizations provide helpful clues as to where we need to dig down deeper for the gold. To see a few examples, we start with some simple plots of variables we processed in the previous section using the powerful ggplot2 (Wickham, 2009) library.

```
> library(ggplot2)

> qplot(crime.data$crime, xlab = "Crime", main ="Crimes in Chicago) +
            scale_y_continuous("Number of crimes")
```

qplot() is a quick plotting function from the ggplot2 library which works similar to the plot() function in the base package (Figure 13.2).

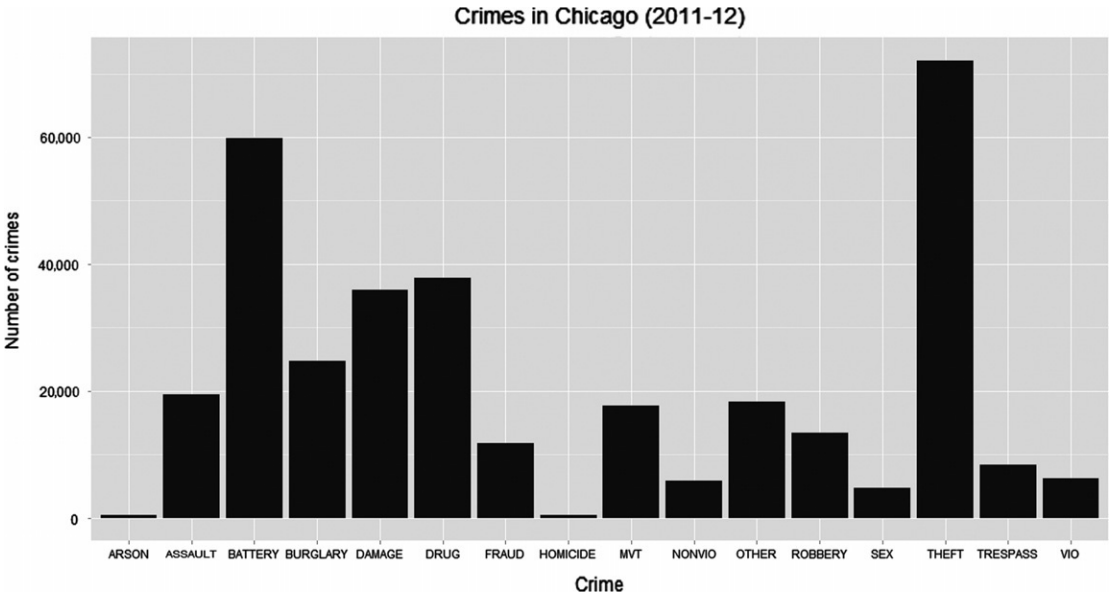


Figure 13.2
Frequency of different crimes in Chicago (2011-2012).

Prevalence of different crimes seems to be unevenly distributed in Chicago with theft and battery being much more frequent. It would be interesting to look at how crimes are distributed with respect to time of day, day of week, and month (Figures 13.3–13.5).

```
> qplot(crime.data$time.tag, xlab="Time of day", main = "Crimes by time of
            day") + scale_y_continuous("Number of crimes")

> crime.data$day <- factor(crime.data$day, levels= c("Mon", "Tue", "Wed",
            "Thu", "Fri", "Sat", "Sun"))
> qplot(crime.data$day, xlab= "Day of week", main= "Crimes by day of week") +
            scale_y_continuous("Number of crimes")

> crime.data$month <- factor(crime.data$month, levels= c("Jan", "Feb", "Mar",
            "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))

> qplot(crime.data$month, xlab= "Month", main= "Crimes by month") +
            scale_y_continuous("Number of crimes")
```

There does seem to a pattern in the occurrence of crime with respect to the dimension of time. The latter part of the day, Fridays, and summer months witness more crime incidents, on average, with respect to other corresponding time periods.

These plots show the combined distribution of all the crime with respect to different intervals of time. We can demonstrate the same plots with additional information by splitting out the different crime types. For example, we can see how different crimes vary by different times of the day. To get the number of different crimes by time of day, we need to roll-up the data

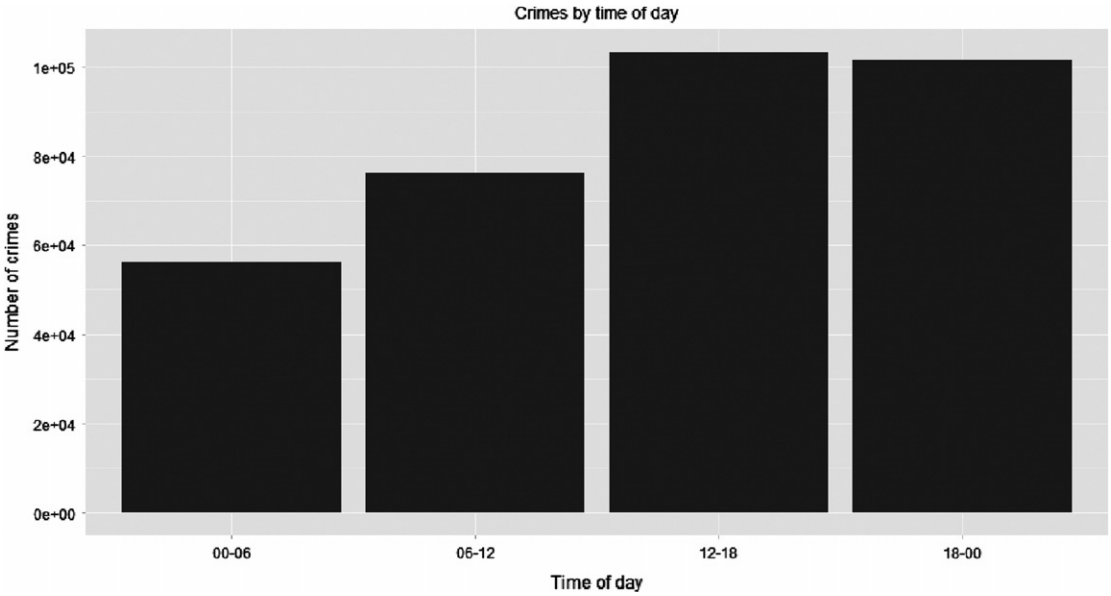


Figure 13.3
Distribution of crime by time of day.

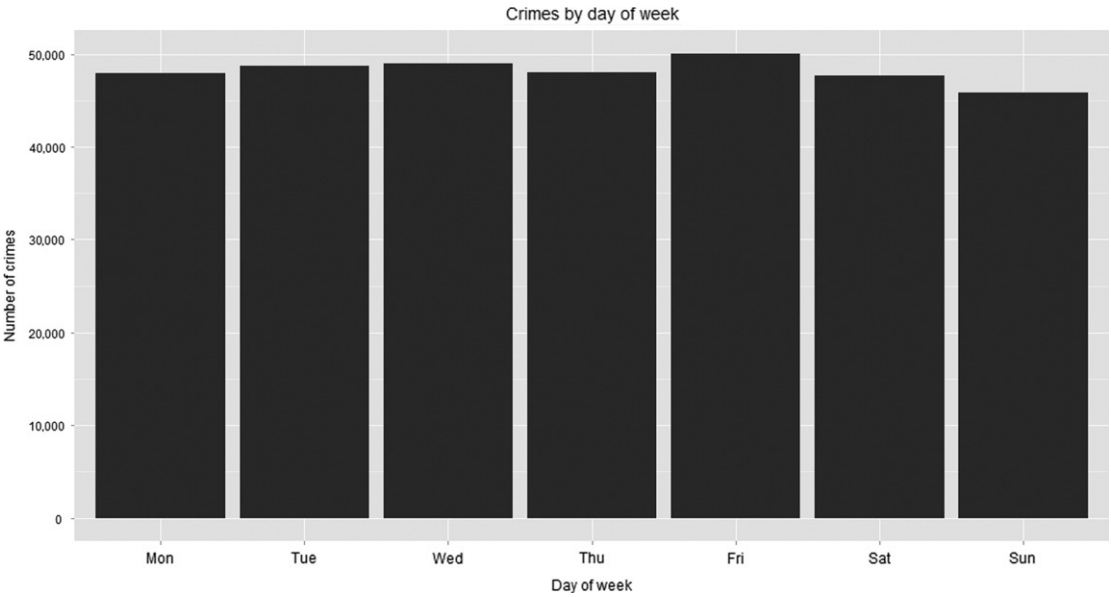


Figure 13.4
Distribution of crimes by day of week.

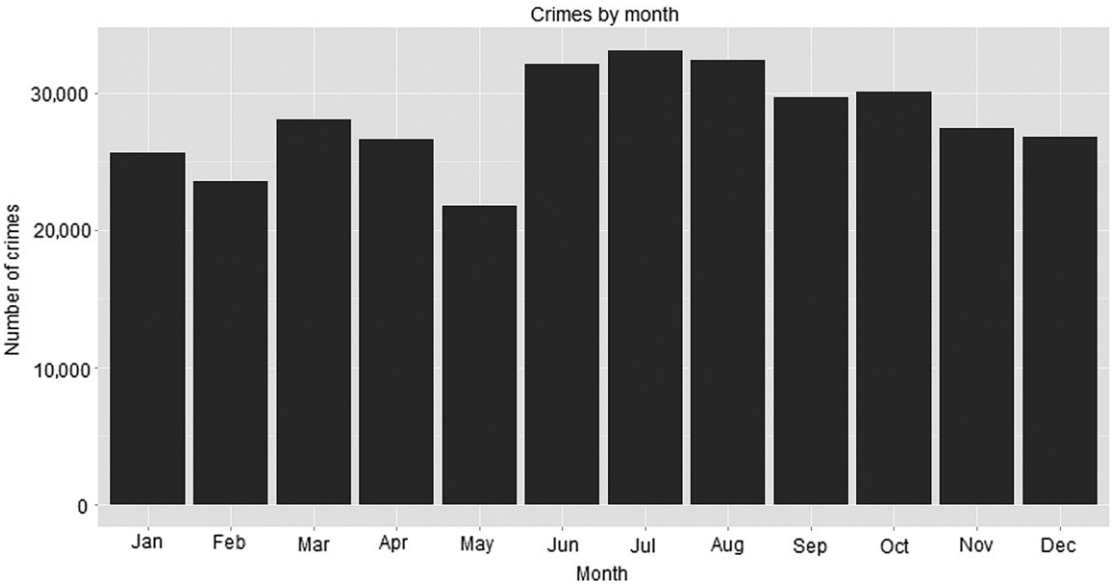


Figure 13.5
Distribution of crimes by month.

at a crime – time.tag level. That is, four rows for each crime type – one for each time interval of the day. An easy way to roll-up data is by using the aggregate() function (Figure 13.6).

```
> temp <- aggregate(crime.data$crime, by= list(crime.data$crime,
            crime.data$time.tag), FUN= length)
> names(temp) <- c("crime", "time.tag", "count")
```

To construct the plot, we use the ggplot() function from the ggplot2 (Wickham, 2009) library.

```
> ggplot(temp, aes(x= crime, y= factor(time.tag))) +
      geom_tile(aes(fill= count)) + scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Time of day", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high = "steelblue") +
      theme_bw() + ggtitle("Crimes by time of day") +
      theme(panel.grid.major = element_line(colour = NA), panel.grid.minor = element_line
      (colour = NA))
```

A quick look at the heat map above shows that most of the theft incidents occur in the afternoon whereas drug related crimes are more prevalent in the evening.

We can do a similar analysis by day of week and month as well and see the distribution of crimes with respect to these parameters. For this purpose, we show two other powerful aggregating functions in R. One is called ddply() from the plyr (Wickham, 2011) library and the other is summaryBy() from the doBy (Højsgaard and Ulrich, 2012) library. Note that all these three functions, in this case, serve the same purpose and can be used interchangeably (Figures 13.7 and 13.8).

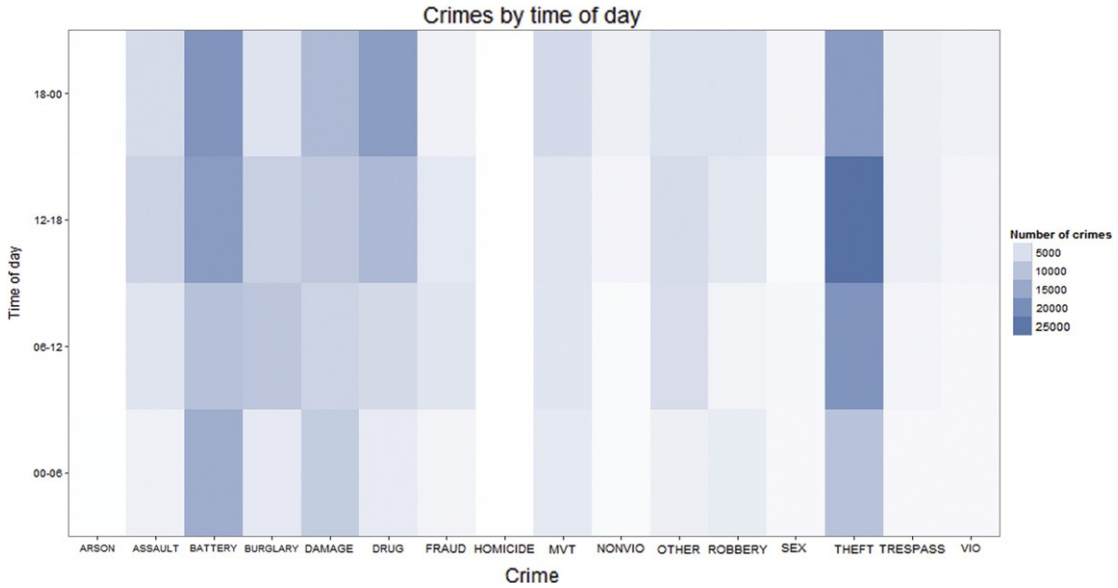


Figure 13.6
Heat map of different crimes by time of day.

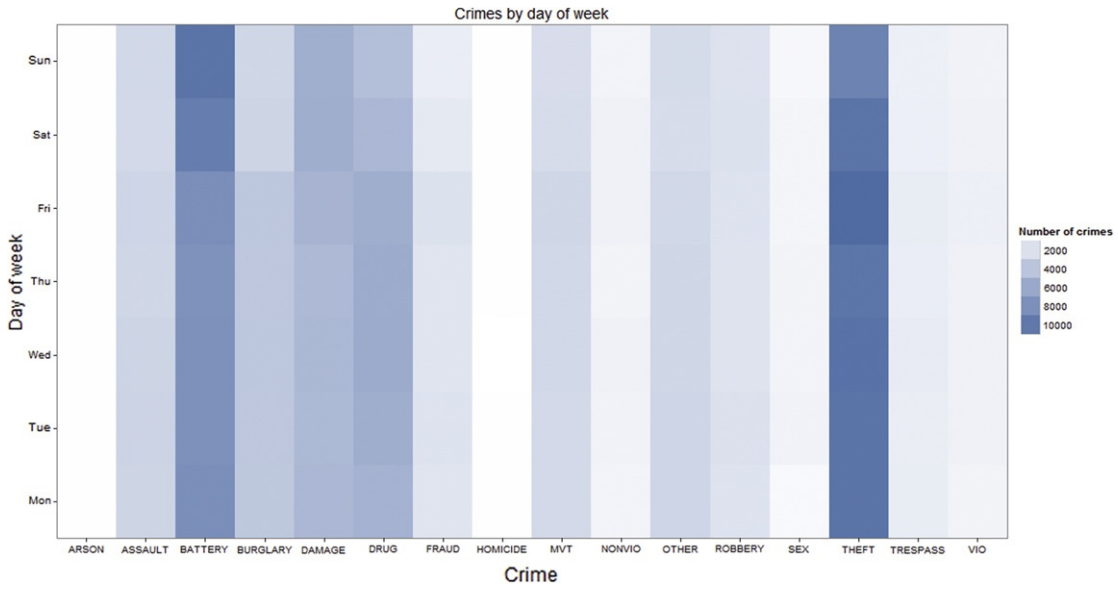


Figure 13.7
Heat map of different crimes by day of week.

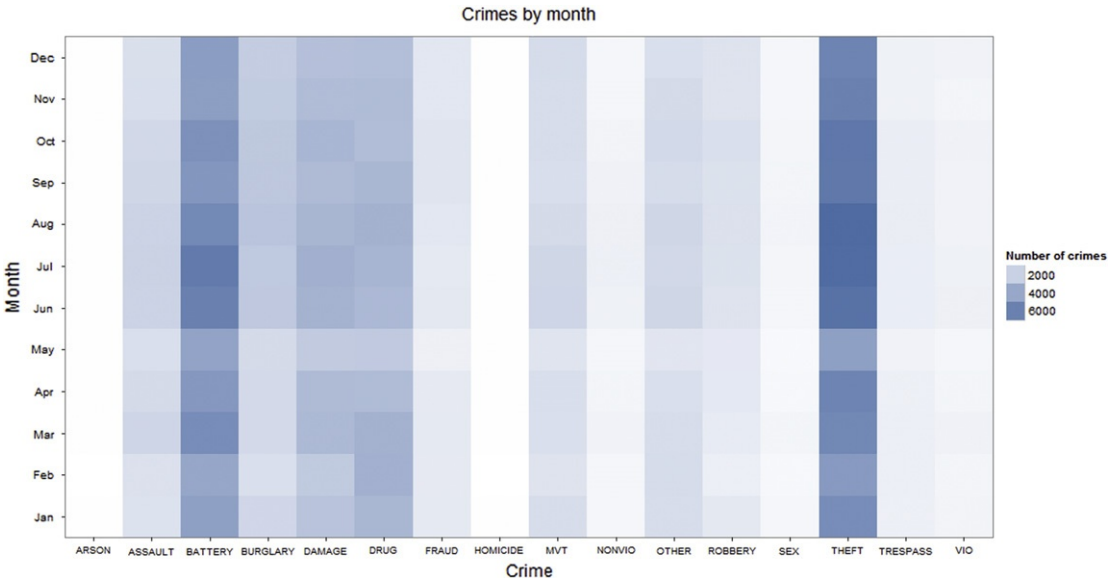


Figure 13.8
Heat map of different crimes by month.

```
> library(plyr)

> temp <- ddply(crime.data, .(crime, day), summarise, count = length(date))

> ggplot(temp, aes(x= crime, y= day, fill= count)) +
      geom_tile(aes(fill= count)) +
      scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Day of week", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high =
      "steelblue") +
      theme_bw() + ggtitle("Crimes by day of week") +
      theme(panel.grid.major = element_line(colour = NA), panel.grid.minor =
      element_line(colour = NA))

> library(doBy)

> temp <- summaryBy(CASE. ~ crime + month, data= crime.data, FUN= length)
      names(temp)[3] <- 'count'

> ggplot(temp, aes(x= crime, y= month, fill= count)) +
      geom_tile(aes(fill= count)) +
      scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Month", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high = "steelblue") +
      theme_bw() + ggtitle("Crimes by Month") + theme(panel.grid.major = element_line
      (colour = NA), panel.grid.minor = element_line(colour = NA))
```

Till now we have only looked at the temporal distribution of crimes. But there is also a spatial element attached to them. Crimes vary considerably with respect to geographies. Typically,

within an area like a zip code, city, or county, there will be pockets or zones which observe higher criminal activity as compared to the others. These zones are labeled as crime hot-stops and are often the focus areas for effective predictive policing. We have the location of each crime incident in our data that can be used to look for these spatial patterns in the city of Chicago. For this purpose, we will utilize the shape files for Chicago Police Department's beats[8] by processing them in R using the maptools (Lewin-Koh and Bivand, 2012) library.

```
> library(maptools)
> beat.shp <- readShapePoly("PoliceBeats/PoliceBeat.shp")
> plot(beat.shp)
```

If we plot the shape file, we get the city of Chicago cut up into beats. We can plot our crime incidences by mapping them to the coordinates of the shape files (Figure 13.9).

To add more value to our analysis, we can also plot the locations of the police stations in Chicago and see if there is any relation between hot-spots of crime and proximity of police stations.

```
> station.shp <- readShapePoly("Police_20Stations-2/police_stations.shp")[9]
```

Unfortunately, we don't have exact locations of the police stations in a ready format. They are embedded in the shape files and need to be extracted out. We have written a function to do just that. It takes a shape file as an input and returns the location (lat and long) of each police station in Chicago as the input.

```
> getPoliceData <- function(shp.file){
            PoliceData <- data.frame()
            NumOfStations = nrow(shp.file @data)
                  for(ii in 1:NumOfStations){
                   PoliceData[ii, 1] <- shp.file@data$DESCRIPTIO[ii]
                   PoliceData[ii, 2] <- shp.file@polygons[[ii]]@labpt[1]
                   PoliceData[ii, 3] <- shp.file@polygons[[ii]]@labpt[2]
                  }
            names(PoliceData) <- c("description", "lat", "long")
            return(PoliceData)
}
> police.data <- getPoliceData(station.shp)
```

Before we can plot the data on the map, we need to process it a bit further and get crime counts for each location. ddply() from the plyr library is a convenient function for this. ddply() takes a portion of a data frame, applies a function to that portion, and returns the result back in the form of a data frame. It works similar to the aggregate() function in the base package but is more flexible, versatile, and powerful.

---

[8] The shape files for CPD beats can be downloaded from https://data.cityofchicago.org/Public-Safety/Boundaries-Police-Beats/kd6k-pxkv.

[9] Shape files for CPD police stations can be downloaded from https://data.cityofchicago.org/Public-Safety/Police-Stations-Shapefiles/tc9m-x6u6.

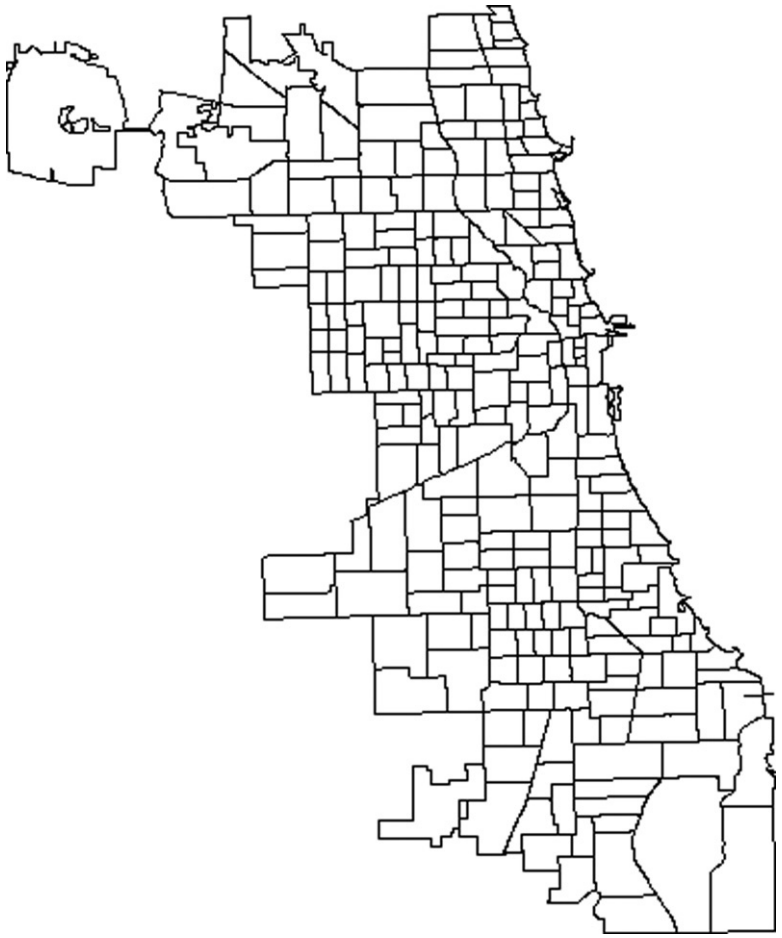


Figure 13.9
Chicago city police beats.

```
> crime.agg <- ddply(crime.data, .(crime, ARREST, BEAT, date, X.COORDINATE,
            Y.COORDINATE, time.tag, day, month), summarise, count = length(date),
            .progress= 'text')
```

We are going to utilize ggplot2's (Wickham, 2009) extensive plotting capabilities for this exercise as well. To plot the shape file using ggplot(), we need to convert the data in the shape file into a data frame using the fortify.SpatialPolygons() function.

```
> source('fortify.R')
```

```
> beat.shp.df <- fortify.SpatialPolygons(beat.shp)
```
[10]

[10] The function fortify.SpatialPolygons() has been written by Hadley Wickham and is available on his GitHub page https://github.com/hadley/ggplot2/blob/master/R/fortify-spatial.r.

```
> crime.plot <- ggplot(beat.shp.df, aes(x=long, y=lat)) +
            geom_path(aes(group = group)) + theme_bw() +
            geom_point(data= police.data, aes(x= lat, y= long))
```

To see how the crimes in Chicago are spread out spatially on a given day, we take the first date in our data set and the subset data for observations relating to this date only. In addition, to plot the crime incidents on the map, we convert our coordinates from factor to numeric (Figure 13.10).

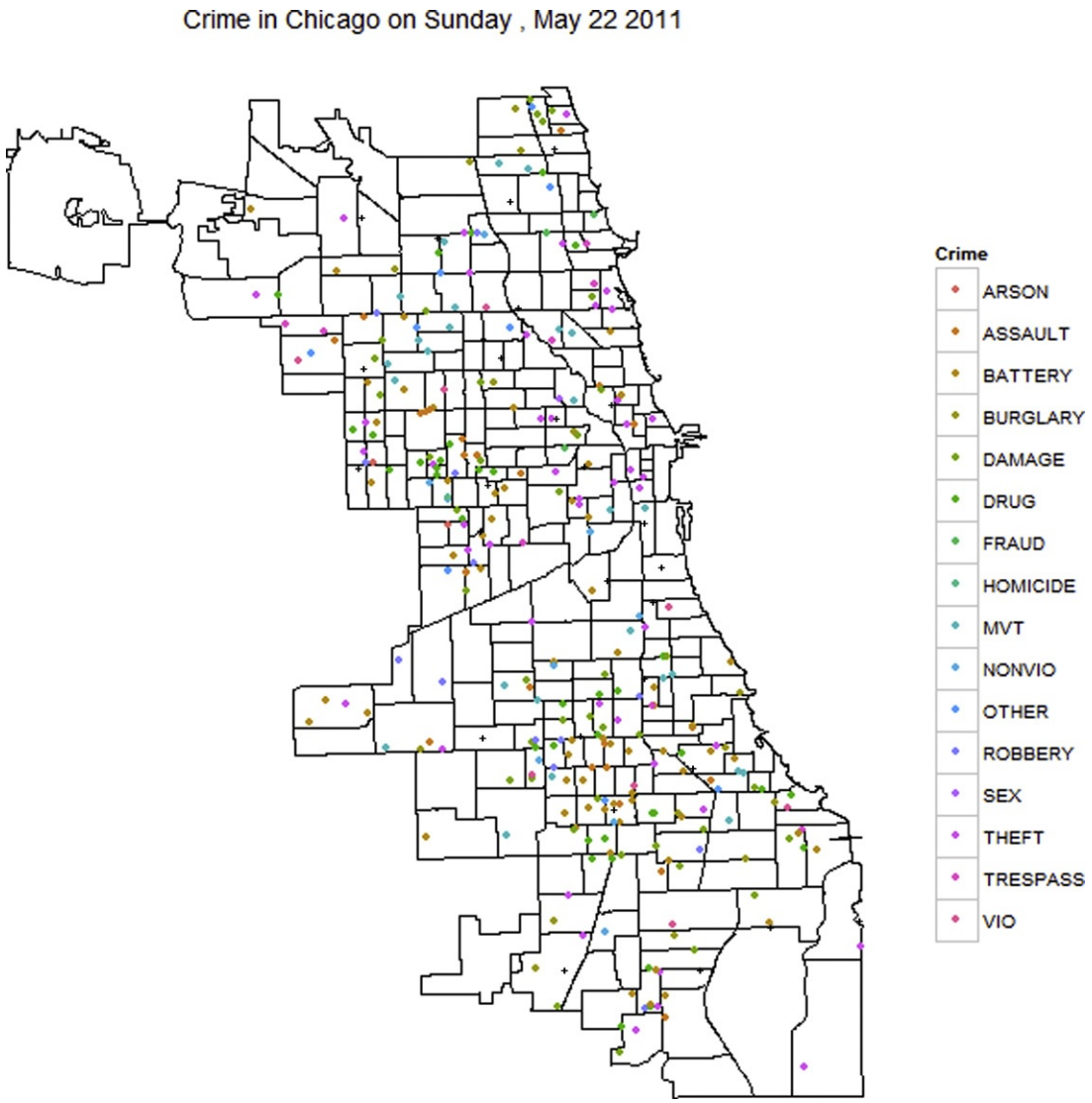


Figure 13.10
Crimes in Chicago on May 22, 2011.

```
> today <- crime.agg[1, 'date']

> crime.agg$X.COORDINATE <- as.numeric(crime.agg$X.COORDINATE)
> crime.agg$Y.COORDINATE <- as.numeric(crime.agg$Y.COORDINATE)

> crime.plot <- crime.plot + geom_point(data= subset(crime.agg,
            as.character(crime.agg$date) == today),
            aes(x=X.COORDINATE, y= Y.COORDINATE, color= crime))

> crime.plot <- crime.plot + theme_bw() +
            ggtitle(paste("Crime in Chicago on", weekdays(today), ",",
            months(today), substr(today, 9, 12), substr(today, 1, 4))) +
            theme(panel.grid.major= element_blank(), panel.grid.minor= element_blank(),
            panel.border= element_blank(),
            axis.title.x= element_blank(), axis.title.y= element_blank(),
            axis.text.x= element_blank(), axis.text.y= element_blank(), axis.ticks=
            element_blank())
> crime.plot
```

The "+" symbols on the plot are the police stations, whereas the other dots represent different crimes. A quick look at the plot indicates that quite a few police stations are located in and around high-crime areas except for the ones in the south-east corner where the incidents of crime appear much lesser. This plot, however, just shows the crime instances for one particular day. A better way to understand the pattern would be look at these crimes for a larger time period, for example, a month or a year. However, plotting the data for the entire year on one plot would not be visually conclusive since there will be too many data points over-crowding the limited space. An effective alternative is to plot the crimes for each day and see how the pattern is changing. We, however, don't want to run the code above for 365 days and then look at 365 different files. This is where R's powerful plotting capabilities come out in full glory. We can use the animation (Xie, 2012) library to create a rolling window plot that updates for each date, plots crimes for the entire year, and provides only one file as the output.

We plot the crimes by taking the subset of the data for each date in the data set, plotting it, saving it, and repeating the process again. We then use the saveMovie()[11] function in the animation package to convert the temporarily saved png files into a gif image.[12]

```
> back.plot <- ggplot(beat.shp.df, aes(x=long, y=lat)) +
            geom_path(aes(group = group)) + theme_bw() +
            geom_point(data= police.data, aes(x= lat, y= long))

> crime.plot <- function(dates){
      for (today in dates){
            date.crime <- crime.agg[as.character(crime.agg$date) == today, ]
            today <- as.Date(today, origin = "1970-01-01")
```

[11] saveMovie() requires ImageMagick which can be downloaded from http://www.imagemagick.org/script/index.php.

[12] A visually more appealing plot on the same lines has been made by Drew Conway and can be found on his blog at http://www.drewconway.com/zia/?p=2741.

```
            date.plot <- back.plot + theme_bw() +
            ggtitle(paste("Crime in Chicago on", weekdays(today), ",", months(today),
            substr(today, 9, 12), substr(today, 1, 4))) +
            theme(panel.grid.major=element_blank(), panel.grid.minor=element_blank(),
            panel.border= element_blank(),
            axis.title.x= element_blank(), axis.title.y= element_blank(),
            axis.text.x= element_blank(), axis.text.y= element_blank(),
            axis.ticks=element_blank())

            date.plot <- date.plot + geom_point(data= date.crime,
            aes(x=X.COORDINATE, y= Y.COORDINATE, color= crime))
            + scale_size(guide= "none", range= c(1.5,2.5)) +
            scale_alpha(guide= "none", range=c(0.3,0.5)) +
            scale_colour_discrete(name="Type of Crime") +
            xlab("") + ylab("")

      plot(date.plot)
  }
}

> dates <- as.character(unique(crime.agg$date))

> saveMovie(crime.plot(dates), movie.name= "chicago_crime_animation.gif",
            img.name= "Rplot", convert= "convert", cmd.fun= system, clean= TRUE, outdir=
      getwd(), interval = 1, width = 580, height = 400)
```

We have seen some interesting visualizations that helped us gauge the data better and provide clues to potential predictor variables for the model. Visualizations, like these and many others, are a great resource for data explorers as they guide us towards information hidden deep down somewhere among these large piles of data.

13.6 Modeling

Before we get to our model, we need to process the data a bit more. Specifically, we need to determine at what level of data are we going to build the model and what kind of independent variables can we derive from the existing data set.

The data we have are at a crime incident level, that is, for each recorded each crime incident in the city of Chicago that occurred in the past 12 months, we have the location, date, type, beat, and ward. The location is available at a very granular level with the exact geographical coordinates present alongside the larger territorial identity of police beats. Such information is vital, but it needs to be understood that not all information can be directly transformed into an implementable solution. We cannot predict exactly when and where the next crime is going to happen. Data scientists don't make prophecies. They derive actionable insights in the form of statistical aggregates. Therefore, understanding the nature of the problem is critical for designing a workable solution within a given set of constraints.

Keeping the ultimate goal of predictive policing in mind, our modeling engine would need to be constructed at a reasonably sized predefined geographical area for reasonably long

predefined time interval. For example, the geographical area can be a block, or a couple of blocks taken together, a zip code, a beat etc. Similarly, the time interval can be an hour or a couple hours or a day.

For our purpose, we will build a multivariate regression model with a negative binomial distribution for errors[13] at a beat-day level for all the crimes taken together. It is debatable whether building the model in this fashion is the right approach or not. We agree to the fact that in terms of accuracy, a better solution would be to deal with different crime types separately and generate predictions for smaller time periods. However, this is just an indicative exercise which has been broken down for understanding and at times, brevity. Following these lines, we try to explain how a solution can be designed through a well suited approach without worrying too much about the exact details and metrics.

To create the modeling data set, we first create our base data set which has all possible unique combinations of beats and dates. The number of rows for the base data will be the product of the total number of unique beats and total number of unique dates.

```
> length(unique(crime.agg$BEAT))
[1] 297

> length(unique(crime.agg$date))[1] 359

> beats <- sort(unique(crime.agg$BEAT))

> dates <- sort(as.character(unique(crime.agg$date)))
```

The function expand.grid() will help create the desired data frame with the unique combination of beats and dates.

```
> temp <- expand.grid(beats, dates)

> names(temp) <- c("BEAT", "date")
```

For convenience, we can sort/order the data frame by beats

```
> temp <- orderBy(~ BEAT, data= temp)
```

The total number of crimes in a beat on a given day can be found by aggregating the data by beats and dates.[14]

```
> model.data <- aggregate(crime.agg[, c('count', 'ARREST')], by=
            list(crime.agg$BEAT, as.character(crime.agg$date)), FUN= sum)

> names(model.data) <- c("BEAT", "date", "count", "ARREST")
```

We then overlap the aggregated crimes data with the temp data set we created to get our final modeling data set.

```
> model.data <- merge(temp, model.data, by= c('BEAT', 'date'), all.x= TRUE)

> View(model.data)
```

[13] There are many approaches that can be taken to develop crime prediction models. We chose a negative binomial model because of its simplicity, general acceptance and understanding, and ease of implementation.

[14] Note that we could have used the ddply() function here as well.

Our modeling data set has all the crime incidents that were recorded in the past 12 months. However, during this period, there were beats in which no crime occurred (or was not recorded) on certain days. For these rows, we see NAs in the count and arrest fields. We can replace the NAs with zero to indicate that there were no crimes, and therefore no arrests, in these beats on these days.

```
> model.data$count[is.na(model.data$count)] <- 0

> model.data$ARREST[is.na(model.data$ARREST)] <- 0
```

We saw in the visualization exercise that there was variation in the crime incidents with respect to days of the week and month, suggesting that they can be used as predictor variables in the model. These variables can be created using the weekdays() and months() functions, respectively.

```
> model.data$day <- weekdays(as.Date(model.data$date), abbreviate= TRUE)

> model.data$month <- months(as.Date(model.data$date), abbreviate= TRUE)
```

A potential important indicator of criminal activity in a particular area could be the history of criminal activities in the past. We can calculate the crime history of each beat for different time periods. To calculate the crime histories, we will use the group averages function ave() and supplement it with a customized input function that will help us look back in time for any number of days by supplying only one argument.

The pastDays() function below creates a vector beginning with a zero followed a number ones, with the ones indicating how many days in the past do we need to look back. For example, to look back at the crime history of the past week, we need to supply 7 as an input to the pastDays() function.

```
> pastDays <- function(x) {
            c(0, rep(1, x))
            }
```

The pastDays() function is supplied further to the ave() function to calculate the number of crimes in the past for each beat.

```
> model.data$past.crime.1 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(1), sides= 1))
```

The ave() function takes three arguments—the numeric variable to be manipulated, the group or level at which the aggregation is required, and the function to be applied to the numeric variable. The filter function applied to the numeric variable "count" in this case takes the sum of the number of rows going back equal to the argument supplied in the pastDays() function and returns the value for each beat date combination expect for the first date for which do not have any past observations.

We can calculate the crimes for the past 7 and 30 days as well.

```
> model.data$past.crime.7 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(7), sides= 1))

> model.data$past.crime.30 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(30), sides= 1))
```

If we look at the data, we can see that the past crime information will not be available for certain dates. For example, if the first observation in our data is from May 22, 2011. This implies that the first observation for the past.crime.30 variable can only be calculated from June 21, 2011 onwards. Since we have 297 beats in our data, we will have missing values for 8910 rows, which is about 8% of our total observations. Simply removing these rows can lead to loss of valuable information. To circumvent this barrier, we need to impute for these missing observations with logical estimates from the data. One way to approach this is by taking the mean for each of these variables and replace the missing cells with this value.

By default, R will give an NA if asked to calculate the mean of a variable which has missing values. It needs to be instructed to not consider the rows with missing values while doing the calculation. meanNA() below, is a simple function to do that.

```
> meanNA <- function(x){
      mean(x, na.rm= TRUE)
}

> model.data$past.crime.1 <- ifelse(is.na(model.data$past.crime.1),
            meanNA(model.data$past.crime.1), model.data$past.crime.1)

> model.data$past.crime.7 <- ifelse(is.na(model.data$past.crime.7),
            meanNA(model.data$past.crime.7), model.data$past.crime.7)

> model.data$past.crime.30 <- ifelse(is.na(model.data$past.crime.30),
            meanNA(model.data$past.crime.30), model.data$past.crime.30)
```

We are not done with past crimes as yet. There is more valuable information that we can extract from these and, typically, such will be the case in most data mining exercises we do. With a few base variables, we can tweak, twist, and turn them to look at the response from different perspectives.

We have information on the number of arrests made in each beat on each day. We calculate similar past variables for arrests and see the effects of police actions in the past on crimes in the future. For example, we can calculate the number of arrests in the past 30 days for each beat.

```
> model.data$past.arrest.30 <- ave(model.data$ARREST, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(30), sides= 1))

> model.data$past.arrest.30 <- ifelse(is.na(model.data$past.arrest.30),
            meanNA(model.data$past.arrest.30), model.data$past.arrest.30)
```

Simply using the past arrests variable will not bring out the effect we are looking for since it will be highly correlated with the past crimes variable (the correlation between past.crime.30 and past.arrest.30 is ~83%)

```
> cor(model.data$past.crime.30, model.data$past.arrest.30)
```

[1] 0.8330903

To bring out the real effect of policing, we can normalize past arrests by past crimes and see if more arrests per crime in the past have deterred criminals from committing crimes in the future.

```
> model.data$policing <- ifelse(model.data$past.crime.30 == 0, 0,
            model.data$past.arrest.30/model.data$past.crime.30)
```

Another useful extension of past crimes variables can be the interaction between crimes in the past 7 days to crime in the past 30 days to see if crime has been increasing or decreasing in the recent past and what effect does this have on future crimes.

```
> model.data$crime.trend <- ifelse(model.data$past.crime.30 == 0, 0,
            model.data$past.crime.7/model.data$past.crime.30)
```

Our visualization exercise showed that there is considerable variation in crime incidents with respect to the time of the year. But we have only 1 year of data for both building the model and validating it. To use the month variable effectively, we can combine the similar months to form a new variable that reflects the changes in crime incidents with seasons.

```
> model.data$season <- as.factor(ifelse(model.data$month %in% c("Mar", "Apr",
            "May"), "spring",
            ifelse(model.data$month %in% c("Jun", "Jul",
            "Aug"), "summer",
            ifelse(model.data$month %in% c("Sep", "Oct",
            "Nov"), "fall", "winter"))))
```

We have created a number of predictor variables, including interactions. The motivation behind deriving these was to get a set of variables that explains a good amount of variation in the dependent variable. One possible way to check this is the correlation between the dependent variable and the independent variables. The psych (Revelle, 2012) library has a convenient function that will allow us to do just that (Figure 13.11).[15]

```
> library(psych)

> model.cor <- cor(model.data[, c('count', 'past.crime.1', 'past.crime.7',
            'past.crime.30','policing', 'crime.trend')])

> cor.plot(model.cor)
```

Though data mining is a creative process, it does have some rules. One of the most stringent ones being that the performance of any predictive model is to be tested on a separate out-of-sample dataset. Analysts and scientists who are ignorant of this are bound to face the wrath of over-fitting leading to poor results when the model is deployed.

[15] In case of an error message regarding the margins being too small for the plot, try adjusting the margins using

```
> op <- par(mar= rep(0,4))
> cor.plot(model.cor)
> par(op)
> dev.off()
```

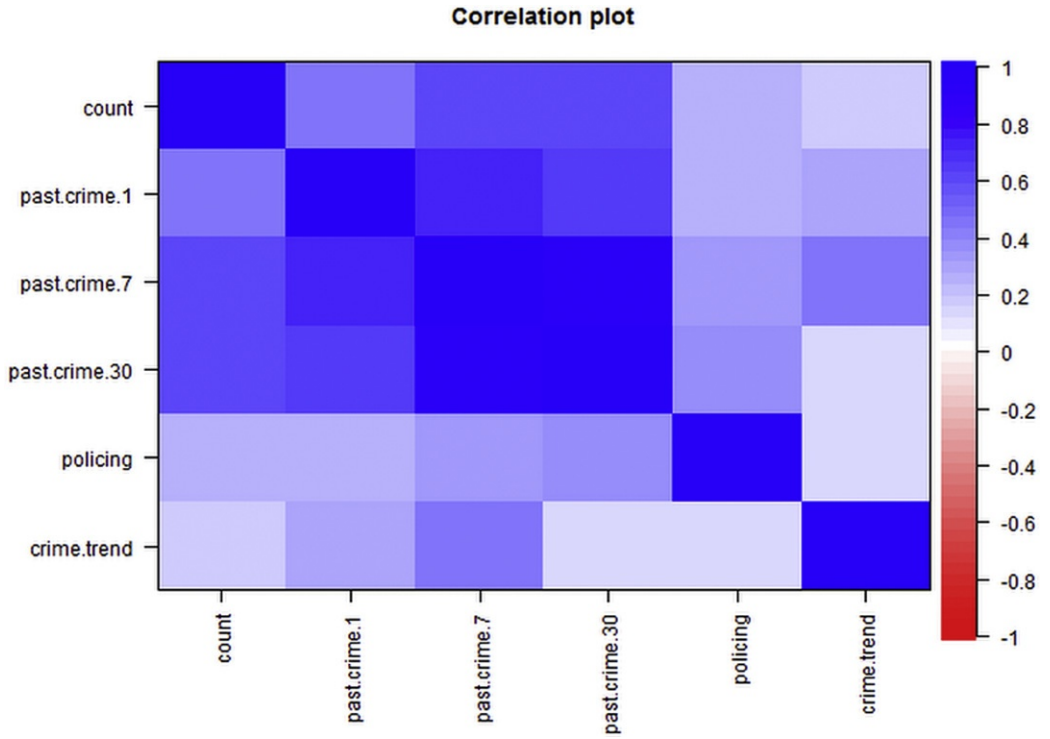


Figure 13.11
Correlation matrix plot.

In our case, to measure the performance of our model, we will use an out-of-time validation sample. We will divide our data into two portions—a 90% development sample where we can train or develop our model and a 10% validation sample where we can test the performance of our model. We can do this by ordering the data by date and then splitting them into train and test using proportional numbers for observations required in each.

```
> model.data <- orderBy(~ date, data= model.data)
> rows <- c(1:floor(nrow(model.data)*0.9))
> test.data <- model.data[-rows, ]
> model.data <- model.data[rows, ]
```

The dependent variable here is a count variable. It is a discrete variable that counts the number of crimes instances in a particular beat on a given day. Typically, for modeling counts, a poisson regression model is a preferred choice that seems to fit the data well. However, the poisson regression model comes with the strong assumption that the mean of the dependent variable is equal to its variance. In our case, this assumption does not hold.

```
> mean(model.data$count)
[1] 3.178985

> var(model.data$count)
[1] 6.028776
```

The variance is much greater than the mean indicating that the distribution is overdispersed. A suitable way to model for such overdispersion is using the negative binomial distribution. We will use the glm.nb() function in the MASS (Venables and Ripley, 2002) package to build the model.

As a basic model, we will include all linear variables and interactive effects we created in the previous section.

```
> crime.model <- glm.nb(count ~ past.crime.1 + past.crime.7 + past.crime.30 +
            + policing + crime.trend + factor(day) + season, data= model.data)

> summary(crime.model)

Call:
glm.nb(formula = count ~ past.crime.1 + past.crime.7 + past.crime.30 +
    policing + crime.trend + day + season, data = model.data,
    init.theta = 11.62188729, link = log)

Deviance Residuals:
    Min 1Q Median 3Q Max
-3.8253 -0.8946 -0.1503 0.5434 5.4160

Coefficients:
                Estimate  Std. Error z value  Pr(>|z|)
(Intercept)   -0.1747152   0.0121101  -14.427   < 2e-16 ***
past.crime.1   0.0121566   0.0010234   11.879   < 2e-16 ***
past.crime.7   0.0009976   0.0005072    1.967  0.049199 *
past.crime.30  0.0086720   0.0001221   71.042   < 2e-16 ***
policing       0.2290357   0.0195652   11.706   < 2e-16 ***
crime.trend    1.3358112   0.0374330   35.685   < 2e-16 ***
dayMon        -0.0568357   0.0073488   -7.734  1.04e-14 ***
daySat        -0.0500302   0.0073600   -6.798  1.06e-11 ***
daySun        -0.1028851   0.0074047  -13.894   < 2e-16 ***
dayThu        -0.0441040   0.0073510   -6.000  1.98e-09 ***
dayTue        -0.0244230   0.0073278   -3.333  0.000859 ***
dayWed        -0.0227957   0.0073220   -3.113  0.001850 **
seasonspring   0.0313964   0.0057203    5.489  4.05e-08 ***
seasonsummer   0.0428420   0.0054100    7.919  2.39e-15 ***
seasonwinter  -0.0059497   0.0057152   -1.041  0.297866
—
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(11.6219) family taken to be 1)

    Null deviance: 174906 on 106622 degrees of freedom
Residual deviance: 119530 on 106608 degrees of freedom
AIC: 422674
```

```
Number of Fisher Scoring iterations: 1

              Theta:  11.622
          Std. Err.:  0.235

2 x log-likelihood: -422642.430
```

Most of the variables are significant at the 0.1% level except crimes in the past 7 days given by past.crime.7 and the winter season. This shows that the visualization exercise was fruitful and our insights based on those did represent the structure present in the data. The significance of variables, however, is not a deciding factor for the model being good or bad. For that, we need to go a step further.

13.7 Model Evaluation

One widely used industry practice to decide whether a model is good or bad is to check its performance on an out-of-sample or out-of-time data set. Before starting the modeling exercise, we kept a portion of data for this purpose. We can check the performance on the out-of-sample data set by scoring it using the predict() function and comparing it to the actual values by using the root mean squared error (RMSE).

```
> crime.model.pred <- predict(crime.model, test.data, type= "response")

> sqrt(mean((test.data$count - crime.model.pred)^2))
[1] 1.950504
```

The RMSE is a popular industry metric that gives us a value of how far we are on average from the actual values. We establish a benchmark with the first iteration of the model. This benchmark then initiates an iterative process of tweaking the model constituents, adding or deleting variables, transforming them till we get some improvement. As an example, consider the scenario where we believe that crimes in the future increase as crimes in the past increase, but at a decreasing rate, that is, the relationship between then is concave. In such a situation including another variable which is a higher-order term for past.crime.30 might help reduce the RMSE.[16,17]

```
> crime.model <- glm.nb(count ~ past.crime.1 + past.crime.7 + past.crime.30 +
            + crime.trend + policing + factor(day) + season + I(past.crime.30^2),
            data= model.data)

> summary(crime.model)

> crime.model.pred <- predict(crime.model, test.data, type= "response")

> sqrt(mean((test.data$count - crime.model.pred)^2))
[1] 1.916148
```

[16] In R, to include a functional form in a glm(), we do not have to create a separate variable in the data set that is the transformation of the original variable. We can simply indicate it directly in the model statement as shown in the example.

[17] The output of the model has been suppressed due to space constraints.

Inclusion of an additional important variable dropped the RMSE from 1.95 to 1.92. Though not a very big jump, it still is an indication that we are headed in the right direction.

Typically, with limited amount of data and time one can only improve so much on a given model. The key is then to decide the optimal point where the trade-off between accuracy and effort is minimized.

RMSE is just one of the many metrics that can be employed to determine the predictive power of a model. Another common method used is the actual versus predicted plot that gives a visual representation of how far away our predictions are from the actual values.

Directly comparing the actual and predicted values may not yield any insights considering there are about ~10,000 data points within a small range. To make the comparison visually convenient and conclusive, we can bin the predictions and the actuals into groups and compare the average values for corresponding groups. For example, we can create 10 groups of the predicted values and take the average of the predicted and actual values in each group. The idea is that if the predictions were reasonably accurate, we would get reasonably overlapping lines. This gives us a comprehensive and condensed view of the differences between our predictions and the actual crimes.

For the plot, we first bring the actual and the predicted values into one data frame and then use the cut() function to create 10 groups of the predicted values and calculate the average values of the predicted and actual crimes by applying the mean() function to each group (Figure 13.12).

```
> validate <- data.frame(test.data$count, crime.model.pred)

> names(validate) <- c("actual", "predicted")

> validate$bucket <- with(validate, cut(predicted, breaks=
            quantile(predicted, probs= seq(0, 1, 0.1)),
            include.lowest= TRUE, labels= c(1:10)))

> validate <- aggregate(validate[, c('actual', 'predicted')], by=
            list(validate$bucket), FUN = mean)
```

For comparison, we plot the actual and predicted means on the same graph.

```
> plot(validate$predicted, col= "red", type= "l", lwd= 1.5, ylab= "No. of Crashes", xlab=
"Predicted Crimes Decile", main= "Actual vs. Predicted")

> lines(validate$actual, col= "blue", lwd= 1.5)

> legend("topright", c("Actual", "Predicted"), col= c("blue", "red"), lwd= c(1.5, 1.5),
bty= "n")
```

The graph shows that the actual and predicted values come quite close in the middle and diverge in the extremes. We are, on average, under predicting in the first few deciles (except the first decile) and over predicting in the last few ones.

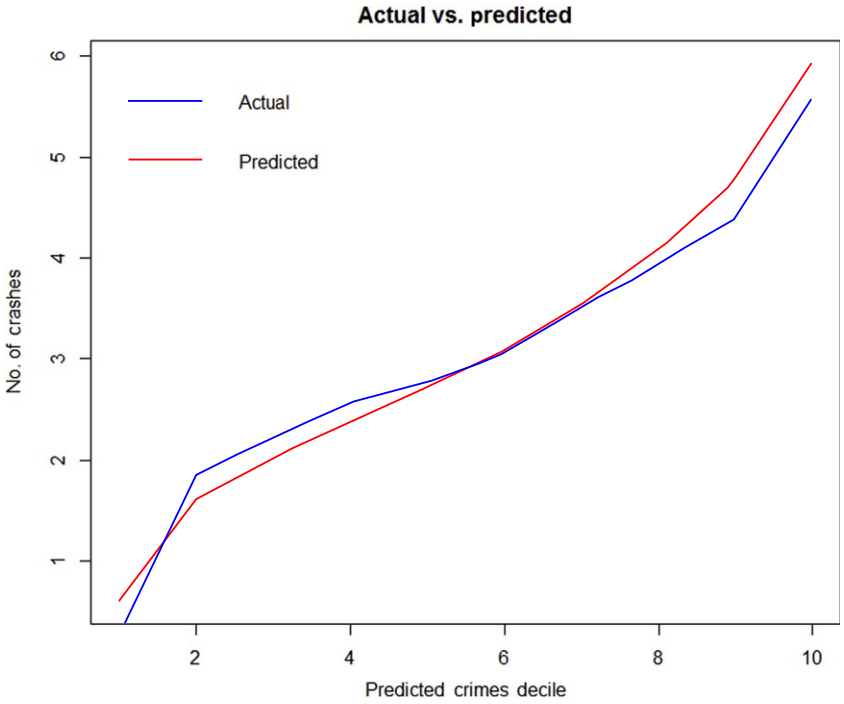


Figure 13.12
Actual versus predicted.

13.8 Discussions and Improvements

It has been a lengthy and informative exercise till now. We learnt how to—pull data directly from web, handle, clean, and process crime data, utilize geo-coded information through shape files, retrieve hidden information through visualizations, create new variables from limited data, build a predictive engine for crime, and test how good it is. There are still some issues that we need to address related to deployment, limitation, and improvements.

The model we built predicts the expected number of crimes in each beat in the city of Chicago for each day. The purpose behind building the model was to build a resource that would help law enforcement agencies deploy their resources proactively and efficiently. In purview of this, one way the model could be deployed is by using color-coded maps identifying the crime hot-spots in the city for a particular day. The crime hot-spots can then be administered effectively by using patrolling teams.

These strategies, however, do have their limitations. First, the 24-h prediction window might appear too large and static for effective patrolling. For example, we saw that crimes follow a pattern during a particular day. Their frequency tends to be much higher during the latter half

of the day. A spatial dimension can be added to this pattern, that is, there might be certain beats which expect more crimes during a particular time interval within a day. A possible workaround could be to build the model for smaller time intervals and allow the crime hot-spots to change during the day. Second, our crime model predicts the expected number of crimes without being able to differentiate among them and by treating them equally. In reality, certain crimes types have totally different characteristics from the others. For example, crimes like murder, aggravated assault, and rape, along with other violent crimes, may need special attention from modelers and the police alike. These nuances could be better addressed if there were separate models for violent crimes, nonviolent and organized crimes. Third, zoning down to one beat as a crime hotspot ignores the impact of crimes in neighboring areas. Since crimes have a spatial distribution attached to them, there might be high correlation between criminal activities in adjoining beats. A simple way to control for this is to include crime history of the adjoining beats in the given model and check their impact on the predictions.

The limitations discussed earlier indicate that there is a lot of scope for further research and work in the area of crime prediction. And the scope is not limited to the data itself. There are different predictive techniques that can be employed to see if we can get incremental lift from these. The suggestions, however, do not undermine the work presented in the chapter which is expected to provide the reader with a detailed understanding of processes involved in mining crime data with R.

References

Højsgaard, S., Ulrich, H., 2012. doBy: doBy—groupwise summary statistics, general linear contrasts, population means (least-squares-means), and other utilities.

James, D., Hornik, K., 2011. chron: chronological objects which can handle dates and times.

Lewin-Koh, N.J., Bivand, R., 2012. maptools: tools for reading and handling spatial objects.

Revelle, W., 2012. Procedures for psychological, psychometric, and personality research.

Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S. Springer, New York.

Wickham, H., 2009. ggplot2: elegant graphics for data analysis. Springer, New York.

Wickham, H., 2011. The split-apply-combine strategy for data analysis. Journal of Statistical Software. 40, 1–29.

Xie, Y., 2012. animation: a gallery of animations in statistics and utilities to create.

CHAPTER 14

Football Mining with R

Maurizio Carpita, Marco Sandri, Anna Simonetto, Paola Zuccolotto
Research Center "Data, Methods and Systems"
Department of Economics and Management of the University of Brescia, Italy

14.1 Introduction to the Case Study and Organization of the Analysis

The use of statistical methods to identify factors determining sporting event outcomes has increased greatly in recent years. This growing interest is motivated by the financial incentives coaches and team owners have to win, the desire of gamblers to improve betting strategies, and by the scientific curiosity of statisticians interested in modeling real-world events (Hopkins, 2012). Early papers from the 1970s highlight the use of statistical thinking in sports. Since then, scientific journals have regularly published articles on the statistical analysis of sports data. The Journal of Quantitative Analysis in Sports, launched in 2005, was the first academic journal devoted solely to this research field (Albert and Koning, 2008). Also in 2005, Albert et al. published an insightful collection of statistical analyses applied to a wide range of sports, including American football, baseball, basketball, and ice hockey (Albert et al., 2005).

Football (known as soccer in the United States) is one of the most popular sports in the world, and many teams use data analysis techniques to study player performance (Kuper, 2011; Slaton, 2012). One frequent goal of statistical studies of football data is the prediction of match outcome (see, for example, Min et al., 2008; Rue and Salvesen, 2000). Another especially fascinating challenge for statisticians is the identification of optimal game strategies (Stern, 2005) in order to improve decision-making based on variables related to match play. There are several examples of statistical studies with this objective in the literature (see, for instance, Carroll et al., 1988; Pollard and Reep, 1997). From a methodological point of view, the huge amount of raw data available on the actions and strategies used in football—combined with the absence of a sound theory explaining the relationships between the many variables affecting match outcome—make the sport an interesting subject for exploratory data analyses using data mining (Han et al., 2011; Hand et al., 2001).

In this chapter, we examine a dataset from the 2010-2011 season of the Italian Football League "Serie A" using a data mining approach and R statistical computing software. The aim of this case study is to identify the key factors that determine the outcome of a football match

Data Mining Applications with R

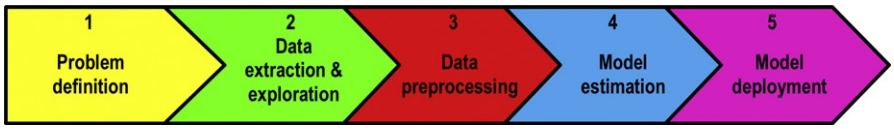


Figure 14.1
The data mining process.

(win, lose, or draw). It is worth noting that when the outcome variable is categorical, it is common to present case studies dealing with dichotomous variables. Here, the outcome is a categorical variable with three categories (win, lose, or draw); the use of this type of variable is one of the distinguishing features of this case study. The analysis applies a wide variety of modern statistical techniques implemented in different R packages, using parallel computing when possible. The data mining process consists of the following tasks: (i) the selection of informative variables using the Gini variable importance measure (VIM) computed by Random forests (RFs) and corrected with the approach proposed by Sandri and Zuccolotto (2008); (ii) the use of the selected variables to construct composite indicators of match outcome by principal component analysis (PCA); and (iii) the training and comparison of five different classification models and algorithms (RF, Classification Neural Network (NNET), KNN, Naïve Bayes classifier, and Multinomial Logit regression) to develop a rule explaining how the composite indicators affect the outcome of the match.

The structure of the chapter follows the framework of a data mining process (Figure 14.1). The code presented here was developed and tested using R version 2.15.2 (R Core Team, 2012), and the scripts are available to the reader as supplementary materials together with the source data. The R packages used in the data mining process and the corresponding supplementary materials are summarized in Table 14.1.

14.2 Background of the Analysis: The Italian Football Championship

Our aim is to investigate the relationship between the outcome of a match and the strategies used by each team. To do this, we analyze data from the top Italian professional football league "Serie A."

The league has 20 teams, selected based on their performance during the previous season, the best 17 from "Serie A" plus the top three teams from the next highest ranking Italian league "Serie B." The season lasts from August to June. A true round-robin format is used: each of the 20 teams plays twice (as home and away team) against each of the other 19 teams, for a total of 38 matches altogether. The season is divided into two parts: the "going round" (andata) and the "return round" (ritorno). Teams are awarded three points for a win, one point for a draw, and no points for a loss. The top four teams in the final ranking qualify for the UEFA Champions League and the fifth

Table 14.1 Main R Packages and Commands Used in the Chapter

Step	Section	Aim	Statistical Method	Supplementary Material		R Package
2	3.1	Data extraction		3_Data	football	Hmisc, ggplot2
	3.2	Data exploration	Descriptive statistics	_Extraction _and_ Exploration.r	_data.Rdata	
3	4.1	Variable importance evaluation	Random forest	4_1_Variable_ Importance_ Evaluation.r		randomForest, snowfall, ggplot2
	4.2	Composite indicators construction	Principal component analysis	4_2_Composite_ Indicators _Construction.r	selected _covariates .RData	
4	5.1	Model development	Random forest	5_1_Model	object	caret, doParallel
			Classification neural network	_Development.r	_scores.RData	caret, doParallel
			k-Nearest neighbor			caret, doParallel
			Naïve Bayesian classification			klaR
			Multinomial logistic regression			globaltest, nnet
	5.2	Model selection	Confusion matrix indices	5_2_Model _Selection.r	classifiers .RData	nnet, randomForest, caret, klaR, doParallel, RSNNS, ggplot2, raster
	5.3	Model refinement	Weighted multinomial logistic regression	5_3_Model_ Refinement.r		nnet, ggplot2, raster
5	6	Model deployment		6_Model _Deployment.r		nnet, ggplot2

and sixth teams qualify for the UEFA Europa League Tournament. The three lowest placed teams in "Serie A" are relegated to "Serie B" for the next season.

14.3 Data Extraction and Exploration

In the first phase of the data mining process, we retrieve the data necessary for a preliminary explorative analysis. These two activities—data extraction and data exploration—are used to define an effective strategy for solving the problem under investigation.

14.3.1 Data Extraction

We use the Panini Digital football database for the 2010-2011 "Serie A" season. Panini Digital is a leader in the collection of statistical data on football, providing data services to clubs and the media (www.paninidigital.com). The football database contains detailed information about plays made during each match (free kicks and shots, action type, fouls, crosses, recovered balls,

goal assists, average time of ball possession, saves, goals on free kicks, etc.). The data collection technique is based on the proprietary software *DigitalScout* and aims to introduce objectivity into the technical/tactical observation of a football match. With *DigitalScout*, a trained operator views the match and records each action in real time, entering data using a point-and-click device and voice recorder using a coded vocabulary shared by leading coaches across the world. For each match, approximately 1300 events are processed, summarized, and recorded in a dataset with 482 variables, which we use in the statistical analysis described later.

The statistical unit of interest for our analysis is the single match. The database is composed of 380 observations (10 matches for each of the 38 rounds) and 482 variables: one categorical target variable Y describing the outcome of the match and 481 covariates $\boldsymbol{X}$ that describe the actions during the match.

We store our data in the R format in the `football_data.Rdata` file:

```
> load(file="football_data.Rdata")
> ls()
[1] "dtset"
> str(dtset)
'data.frame': 380 obs. of 482 variables:
$ RIS           : Factor w/ 3 levels "1","2","X": 1 3 1 3 1 3 1 3 1 2 ...
$ G_MINUTI_C    : num 97 98 95 94 96 94 96 97 94 97 ...
$ G_PALLE_GIOC_C: num 570 441 582 526 690 575 540 639 315 508 ...
$ G_POS_PAL_C   : num 2745 2011 2500 2137 3119 ...
$ G_SUP_TER_C   : num 939 502 905 1011 1331 ...
... [list output truncated]
```

The `dtset` object is a data frame with $n = 380$ observations (rows) and $(p+1) = 482$ variables (columns). The target variable is the first variable (`RIS`), a `factor` with the three categories "1," "2," and "*X*." We copy this variable into the $(n \times 1)$ target vector `y` and all the explanatory variables into the matrix `Xf` with dimension $(n \times p)$. Categories "1", "2," and "*X*" of `RIS` are recoded as "*W*" (win), "*L*" (loss), and "*D*" (draw), respectively:

```
> y  <- factor(dtset[,1], labels=c("W","L","D"))
> Xf <- dtset[,2:ncol(dtset)]
```

According to the provisions of Panini Digital, in order to avoid the identification of the matches two variables have been deleted and the order of records has been randomized.

14.3.2 Data Exploration

As mentioned earlier, the aim of our analysis is to select and identify the factors most strongly associated with match outcome. Before proceeding with model specification, we implement an exploratory analysis to determine the main characteristics of the data. For ease of interpretation, we have decided to avoid transformation of covariates. Focusing on the target variable, we investigate its distribution by computing a table of absolute and percentage frequencies (`table` and `prop.table` commands). The `cbind` function is used to join the two vectors:

```
> AbsFreq <- table(y)
> PerFreq <- round(prop.table(AbsFreq)*100,1)
> cbind(AbsFreq,PerFreq)
AbsFreq  PerFreq
W   180      47.4
L   104      27.4
D    96      25.3
```

The distribution of the target variable shows that approximately half of the matches (47%) ended with victory for the home team, while the frequencies of losses and draws are similar (27% and 25%, respectively). This result supports the importance of the "home team advantage," which is well known in football (and other sports). To visualize this distribution (Figure 14.2a), we create a bar chart using the graphic library `ggplot2` (Wickham, 2009):

```
> library(ggplot2)
> Freq <- data.frame(PerFreq)
> ggplot(Freq, aes(x="", fill=y, weight=Freq))+geom_bar(width = 1)+
+  scale_fill_manual(values=c("green","yellow","red"))+
+  scale_y_continuous("Percentage frequency")+scale_x_discrete(name="")
```

Focusing on the covariates in matrix $\boldsymbol{X}_f$, we calculate selected univariate descriptive statistics. The `describe` command yields a synthetic representation of covariate distributions, including minimum, maximum, mean, median, and percentiles for numerical variables as well as a table of absolute frequencies for categorical variables:

```
> library(Hmisc)
> describe(Xf)
Xf
481  Variables  380  Observations
---------------------------------------------------------------------------
G_MINUTI_C
  n  missing  unique   Mean  .05  .10  .25  .50  .75
380        0      12  95.98   94   94   95   96   97
.90      .95
 98       99

          91  92  93  94  95  96  97  98  99  100  101  104
Frequency  2   3  13  52  82  92  76  33  17    6    3    1
%          1   1   3  14  22  24  20   9   4    2    1    0
---------------------------------------------------------------------------
...
---------------------------------------------------------------------------
O_OCCAS_C
     n  missing  unique   Mean     .05     .10     .25     .50     .75     .90     .95
   380        0      15  4.942       1       2       3       5       6       8      10

          0   1   2   3   4   5   6   7   8  9  10  11  12  13  14
Frequency 6  18  37  69  50  59  51  39  14  9  11   7   7   1   2
%         2   5  10  18  13  16  13  10   4  2   3   2   2   0   1
---------------------------------------------------------------------------
...[list output truncated]
```

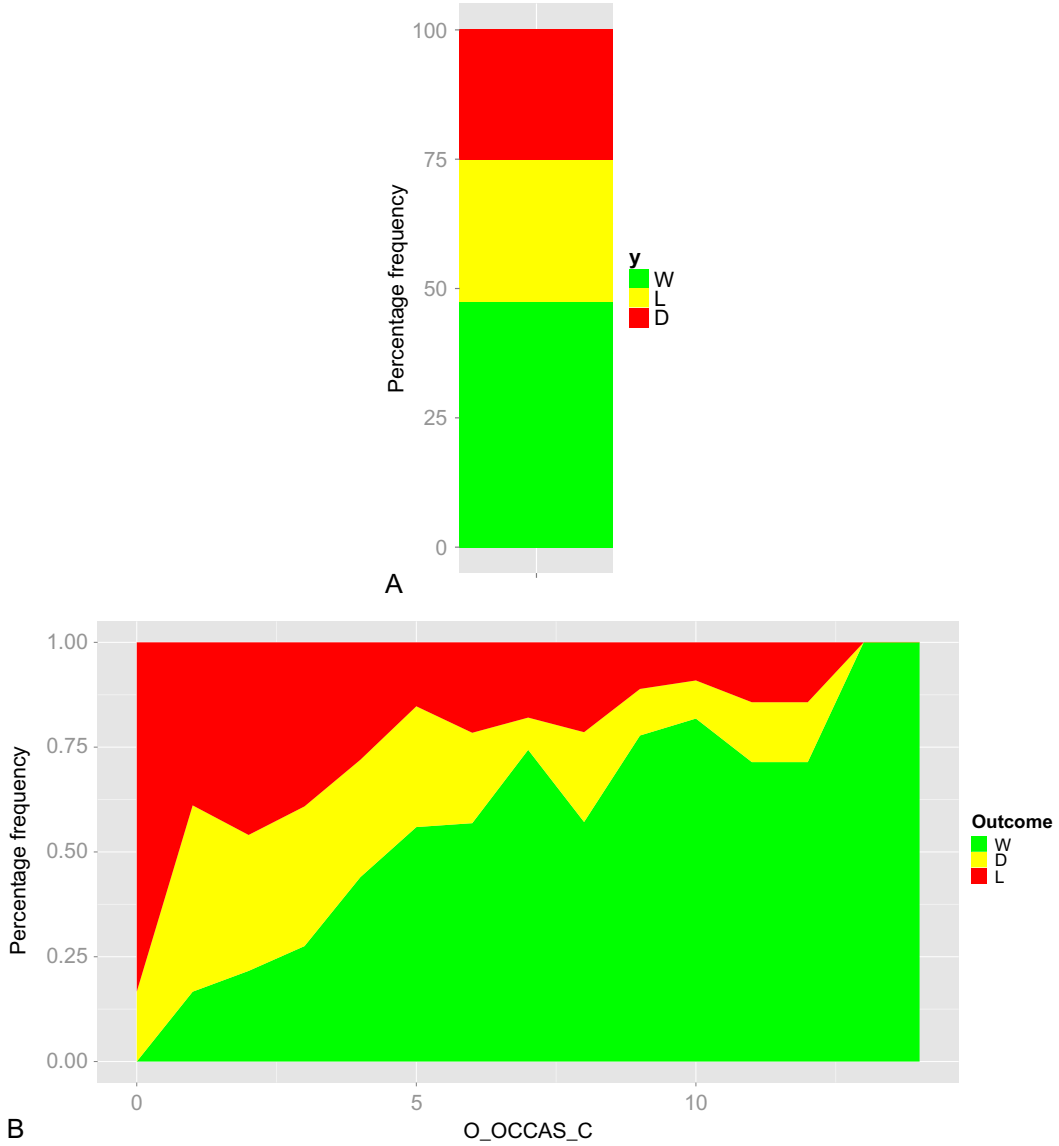


Figure 14.2
(a) Bar chart of *Y* (match outcome); (b) area graph of *Y* given *X* = "*O_OCCAS_C*".

For example, the descriptive statistics for the O_OCCAS_C variable (number of scoring opportunities created by the home team) are mean = 4.942, median = 5, minimum = 0, and maximum = 14.

Focusing on the relationship between the target variable and the covariates, we investigate how the distribution of the target variable changes for different values of a single covariate. First, we

create the matrix `pf` containing the percentages of W, L, and D outcomes for each value of O_OCCAS_C. These frequencies are then collected in the `dtst` data frame and formatted according to the requirements of the `ggplot2` graphics environment:

```
x.name <- "O_OCCAS_C"
x <- Xf[,names(Xf) %in% x.name]
pf <- prop.table(table(x,y),1)[,c(1,3,2)]
dtst <- data.frame(PctFreq=c(t(pf)),
      x=rep(as.numeric(rownames(pf)),each=ncol(pf)),
      Outcome=ordered(rep(1:3,nrow(pf)), labels=colnames(pf)))
```

Using `geom_area`, we easily draw an area graph of W, L, and D percentages as the value of O_OCCAS_C varies:

```
ggplot(dtst, aes(x=x, y=PctFreq, group=Outcome, fill=Outcome)) +
+    geom_area(position="fill") + scale_x_continuous(x.name) +
+    scale_y_continuous("Percentage frequency") +
+    scale_fill_manual(values = c("green", "yellow", "red"))
```

The plot in Figure 14.2b shows that when the number of home team scoring opportunities increases the frequency of wins for the home team increases. Conversely, the frequencies of draws and losses for the home team decrease.

14.4 Data Preprocessing

In the third step of the data mining process, we study the relationship between the $(n \times p) = (380 \times 481)\,\boldsymbol{X}_f$ covariate data matrix and the $(n \times 1) = (380 \times 1)$ $\boldsymbol{y}$ target vector representing match outcome. These objects are defined at the end of Section 3.1 (`Xf` and `y`, respectively). Since the number of observed covariates is higher than the number of sample units ($n < p$), we have decided to include a preliminary variable selection step in our procedure.

14.4.1 Variable Importance Evaluation

The basic function of variable selection is to identify those variables with the greatest influence on outcome. We used the RF algorithm (Breiman, 2001a), because it is well suited to treat the case: $n < p$. This methodology allows us to approximate relationships, between the p covariates in $\boldsymbol{X}$ and the target variable Y, even those that are very complex. At the same time, we can identify any redundant or noninfluential covariates (Liaw and Wiener, 2002).

An RF with randomly selected inputs consists of a sequence of classification or regression trees (CART, Breiman et al., 1984) grown by selecting randomly from $\boldsymbol{X}$, at each node, a small group of h covariates on which to split the node. As is customary when using the CART methodology, the splitting criterion is the maximum heterogeneity reduction in the target variable Y, which has three categories: $(Y_1, Y_2, Y_3) = (W, L, D)$. This procedure is used together

with *bagging* (Breiman, 1996), which is the random selection of a subsample from the original training set at each tree. This simple and effective idea is based on a theoretical framework described by Breiman (2001a) in his seminal work. The RF prediction is an average of the tree predictions for Y, computed by passing down each tree only the observations that did not contribute to its construction (*out-of-bag* predictions).

The RF algorithm returns two main VIMs that are used to identify the most important predictors (Breiman, 2002) in $\boldsymbol{X}$. The first VIM is the *mean decrease in accuracy* (MDA). For each tree in the RF, the out-of-bag prediction accuracy is recorded and compared to the accuracy of the same tree when the values for the jth covariate are randomly permuted. The MDA VIM of the jth covariate is then obtained by averaging over all trees the decrease in accuracy due to this permutation. The second VIM is the *total decrease in node impurities* (TDNI). At each split of the RF in each tree, the heterogeneity reduction in the target variable Y is defined as the importance measure attributed to the splitting variable in that particular split. The TDNI VIM is then obtained by summing up these measures, separately for each covariate, for all of the trees in the RF.

In general, the heterogeneity reduction in the target variable Y due to the split of node w into the two daughter nodes wl and wr, is measured as

$$d_w = \frac{n_w}{n}\left\{\hat{H}_{Y|w} - \left(\frac{n_{wl}}{n}\hat{H}_{Y|wl} + \frac{n_{wr}}{n}\hat{H}_{Y|wr}\right)\right\},$$

where $\hat{H}_{Y|w}$, $\hat{H}_{Y|wl}$, and $\hat{H}_{Y|wr}$ are the estimated heterogeneities, of Y in node w and in the left and right splits, respectively. Similarly, n_w, n_{wl}, and n_{wr} are the sample sizes in node w and in the left and right splits.

The heterogeneity measure $\hat{H}$ we adopt in our analysis is the Gini index,

$$\hat{H}_{Y|w} = \hat{G}_{Y|w} = 1 - \sum_{k=1}^{3} f_{kw}^2,$$

where f_{kw}^2 is the observed relative frequency of kth category in node w. The corresponding TDNI measure is the Gini VIM. In the variable selection analysis below, we rely on the Gini VIM, as recommended by Calle and Urrea (2010). These authors showed that variable ranking according to the Gini VIM is generally more stable than the ranking obtained by the MDA VIM. Nonetheless, VIMs should be used cautiously for variable selection because they may lead to spurious results under some circumstances (Nicodemus, 2011). In addition, as observed by Strobl et al. (2007) and Sandri and Zuccolotto (2008), the Gini VIM is biased in favor of variables that have either more distinct numerical values (or categories) or fewer missing values. In this case study, we deal with bias in the Gini VIM using the heuristic correction strategy proposed by Sandri and Zuccolotto (2008, 2010). The idea behind this correction

algorithm is to estimate bias by means of p "pseudo-covariates" $\boldsymbol{Z}$, independent on Y (thus, uninformative) and having the same marginal distributions of $\boldsymbol{X}$. A set of S observations for $\boldsymbol{Z}$ is generated from $\boldsymbol{X}$ by a heuristic permutation procedure, and the estimated VIMs of the pseudo-covariates are then shown to approximate the unknown bias (Sandri and Zuccolotto, 2008, 2010).

The main steps of this algorithm are:

1. Generate a matrix $\boldsymbol{Z}_f$ of pseudo-covariates by randomly permuting the rows of $\boldsymbol{X}_f$;
2. Build an RF for the prediction of Y using the covariates in $\boldsymbol{X}_f$ and the pseudo-covariates in $\boldsymbol{Z}_f$ as explanatory variables;
3. Compute the Gini VIMs of covariates and pseudo-covariates;
4. For each $j = 1, 2, \ldots, p$, calculate the difference between the estimated VIM of variable X_j and the corresponding pseudo-covariate Z_j;
5. Repeat steps 1 to 4 a total of S times.
6. Calculate the mean values of these differences over the S replications.

The rows of $\boldsymbol{X}_f$ can be randomly permuted using the `sample` command:

```
> Zf <- Xf[sample(nrow(Xf)),]
```

Covariate and pseudo-covariate matrices, along with the target vector, are first collected in a data frame:

```
> dtset.pseudo <- data.frame(cbind(Xf, Zf, y))
```

and then used for the construction of an RF using the `randomForest` package:

```
> library(randomForest)
> rf <- randomForest(y ~ ., data=dtset.pseudo, ntree=500)
```

We set the number of trees in the RF to 500 to limit the computational time, but a higher number would be preferable. The `mtry` parameter (denoting the number h of covariates randomly selected at each node) is set to the default value `floor(sqrt(ncol(x)))`, the square root of the number of covariates, because the bias-correction algorithm is fairly insensitive to this parameter. The (biased) mean decrease in node impurity (the Gini VIM) is estimated using the `importance` command with the `type=2` option (use `type=1` for the MDA VIM):

```
> VIMs <- importance(rf, type=2)
```

and the differences for step 4 are given by:

```
> p <- ncol(Xf)
> VIMs.unb <- VIMs[1:p,] - VIMs[(p+1):(2*p),].
```

The above steps must be repeated S times, which may take a long time when the number of covariates is in the order of hundreds or greater. For this reason, we advise using the parallel computing capabilities of R when more processing units are available. We start defining an R function that collects the four steps of the algorithms with:

```
VIMs.unb <- function(k) {
  set.seed(k)
  Zf <- Xf[sample(nrow(Xf)),]
  dtset.pseudo <- data.frame(cbind(Xf,Zf,y))
  rf <- randomForest(y ~ ., data=dtset.pseudo, ntree=500)
  VIMs <- importance(rf, type=2)
  VIMs[1:p,] - VIMs[(p+1):(2*p),]
}
```

We then show how the bias-correction algorithm of Sandri and Zuccolotto (2008) can be parallelized using the `snowfall` R package (Knaus, 2010), a usability wrapper for the `snow` library (Tierney et al., 2012):

```
> library(snowfall)
> sfInit(parallel=TRUE, cpus=6, type="SOCK")
> sfLibrary(randomForest)
> sfExport("Xf", "y", "p")
> VIMs.list <- sfLapply(x=1:100, VIMs.unb)
> sfStop()
```

Using the `sfInit` command, we initialize the `snowfall` functions with parallel computing enabled (`parallel=TRUE`) and request a socket cluster (`type="SOCK"`) with six CPUs. The `sfLibrary` command loads the `randomForest` package onto all computing nodes and `sfExport` exports objects `Xf`, `y`, and `p` to each node. The parallel version of `lapply` is `sfLapply`, which applies the `VIMs.unb` function to each element of the vector `x=1:100`. The output of `VIMs.unb` is the object `VIMs.list` containing a list of the same length as `x`. Each element of this list is equal to the expression `VIMs[1:p,]-VIMs[(p+1):(2*p),]` for one of the $S = 100$ replications of the bias-correction algorithm. Reproducibility is ensured by the use of a fixed random seed (the `set.seed(k)` command in the `VIM.unb` function) for each run of the worker processes (Knaus and Porzelius, 2009). Parallel computation stops after the execution of the `sfLapply` command using `sfStop`.

At this point, we convert the list `VIMs.list` into a matrix `VIMs` of dimension $S \times p$ and calculate unbiased Gini VIMs of the p covariates by averaging the values of the differences contained in `VIMs.list` over the S replications:

```
> VIMs <- t(matrix(unlist(VIMs.list),p))
> GINI.unb <- apply(VIMs,2,mean)
```

The box-plots of the bias-corrected Gini VIMs for the 481 covariates obtained with the $S = 100$ replications displayed in Figure 14.3 are generated using the following code:

```
> idx <- order(GINI.unb,decreasing=T)
> vm <- c(VIMs[,idx])
> grp <- c(t(matrix(rep(1:ncol(VIMs),nrow(VIMs)),ncol(VIMs))))
> dt <- data.frame(vm,grp=factor(grp))
> ggplot(dt, aes(grp,vm)) + geom_boxplot(outlier.size = 0)+
+ scale_x_discrete(breaks=c(1,100,200,300,400,p), name="") +
```

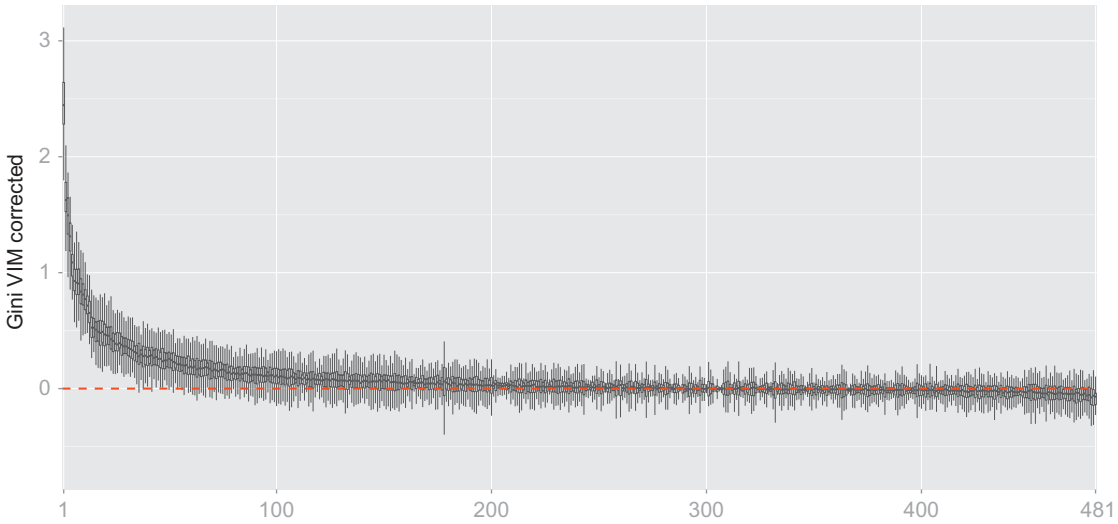


Figure 14.3
Box-plots of bias-corrected Gini VIMs for the *481* covariates (100 replications).

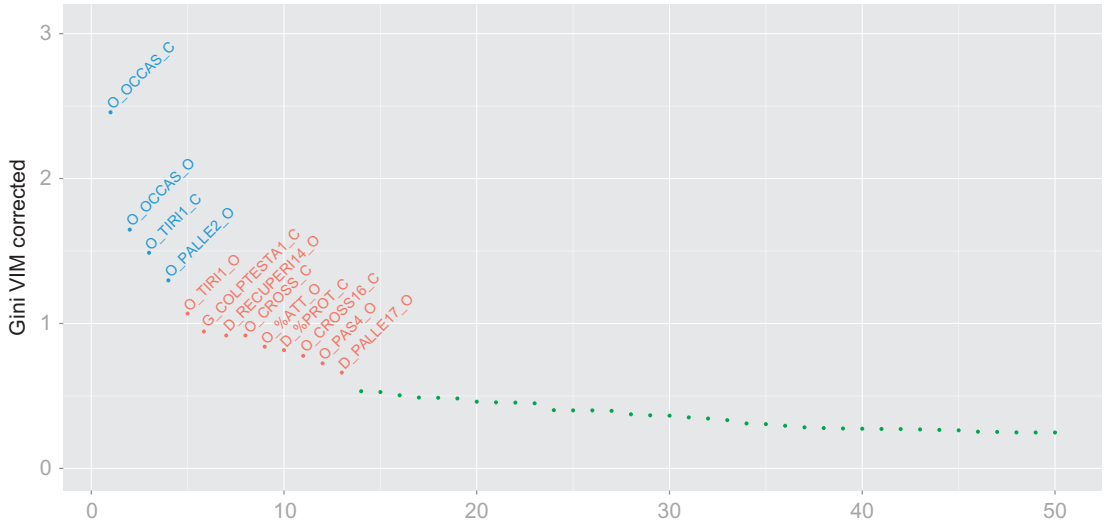


Figure 14.4
The 50 most important covariates according to the mean bias-corrected Gini VIM.

```
+ scale_y_continuous(name="Gini VIM corrected")+
+ geom_hline(yintercept=0, colour="red", lty=2, lwd=1)
```

Figure 14.4 shows the mean unbiased Gini VIMs for the 50 most important covariates (code reported in the supplementary materials). The graph highlights the presence of three groups of

covariates: four "highly" informative covariates (blue dots), nine covariates with "medium" importance (red dots), and a residual group (green dots) of covariates having low VIMs (<0.5) that are approximately constant. According to these criteria, the first 13 explanatory variables selected are:

- O_OCCAS_C: number of scoring opportunities created by the home team;
- O_OCCAS_O: number of scoring opportunities created by the away team;
- O_TIRI1_C: number of shots on goal for the home team;
- O_PALLE2_O: number of ball kicks to bypass the midfield by the away team;
- O_TIRI1_O: number of shots on goal by the away team;
- D_RECUPERI14_O: number of defensive headings in the penalty area by the away team;
- O_CROSS_C: number of crosses by the home team;
- G_COLPTESTA1_C: number of heading shots by the home team in the away team's penalty area;
- O_%ATT_O: goal attack percentage by the away team;
- D_%PROT_C: percentage of penalty area's defense by the home team;
- O_CROSS16_C: number of crosses on free kicks by the home team;
- O_PAS4_O: number of long passes forward from the defense of the away team;
- D_PALLE17_O: number of balls lost during forward actions by home team.

Finally, we store the selected covariates in the (380×13) matrix $\boldsymbol{X}_s$:

```
> Xs <- Xf[,idx[1:13]]
```

14.4.2 Composite Indicators Construction

In the second phase of data preprocessing, after selecting the 13 variables that most influence match results, we combine those variables into a smaller number of easily interpretable key indicators. To do this, we use PCA. The main function of PCA is to reduce the dimensionality of a data set of q correlated variables while retaining as much of the variation within the data set as possible. The earliest descriptions of the technique now known as PCA were published by Pearson (1901) and Hotelling (1933), while the most widely cited modern reference is Jolliffe (2002).

Given an $(n \times q)$ data matrix $\boldsymbol{X}$, containing n observations of q numerical variables $\boldsymbol{X}$, PCA results in a linear transformation to a new coordinate system such that the new set of q numerical variables Z, the principal components (PCs), are uncorrelated. The orthogonal projections of data onto the one-dimensional spaces spanned by Z have the greatest variance, the second greatest variance, and so on, respectively. This reduction in complexity is achieved by selecting components containing a significant portion of the total observed variance.

PCA is based on the singular value decomposition of the data matrix $\boldsymbol{X} = \boldsymbol{UDV}'$ (Hastie et al., 2001), where $\boldsymbol{U}$ is an $n \times q$ orthogonal matrix of left singular vectors, $\boldsymbol{V}$ is a $q \times q$ orthogonal

matrix whose column vectors $\boldsymbol{v}_j$ are right singular vectors, and $\boldsymbol{D}$ is a $q \times q$ diagonal matrix of the singular values d_j of $\boldsymbol{X}$. The number of PCs is equal to q. The PCs are in the form $\boldsymbol{Z} = \boldsymbol{XV}$, so that $\boldsymbol{z}_1 = \boldsymbol{X}\boldsymbol{v}_1$ has the largest sample variance, $\mathrm{Var}(\boldsymbol{z}_1) = \mathrm{Var}(\boldsymbol{X}\boldsymbol{v}_1) = d1^2/q$, among all of the normalized linear combinations of the original variables. To reduce the dimensionality of our data, we keep the first b PCs; the amount of variance they account for is referred to as the cumulative variance accounted for (CVAF):

$$\mathrm{CVAF} = \sum_{j=1}^{b} \mathrm{Var}(\mathbf{z}_j) = \sum_{j=1}^{b} \frac{d_j^2}{q}.$$

To facilitate interpretation of the results, we choose to conduct two separate PCAs, one for variables directly related to the home team and one for variables directly related to the away team.

14.4.2.1 PCA for the Home Team

We start selecting from $\boldsymbol{X}_s$ the six variables related to the home team (O_OCCAS_C, O_TIRI1_C, O_CROSS_C, G_COLPTESTA1_C, D_%PROT_C, and O_CROSS16_C) to build the data matrix $\boldsymbol{X}$ described above. In this case, $n = 380$ and $q = 6$:

```
> X <- Xs[,grep("_C$", names(Xs))]
```

Below, we build a `PCA` function that calculates eigenvalues, loadings, and scores using the `princomp` command. In order to facilitate interpretation, the first `b` loadings are orthogonally rotated using the `varimax` command, standardized by the `scale` command, and multiplied by the rotation matrix. The result yields the first `b` varimax-rotated object scores. The inputs of the `PCA` function are the `X` covariate matrix and the number of PCs to be retained `(b)`. The output of the function is an object of class `list` with five components: `eigvals`, `loadings`, `obj.scores`, `loadings.rot`, and `obj.scores.rot`.

```
PCA <- function(X,b) {
  pca <- princomp(X, cor=T, scores=T)
  loadings <- pca$loadings
  obj.scores <- pca$scores[,1:b]
  Rot <- varimax(loadings[,1:b])
  list(eigvals=(pca$sdev)^2, loadings=loadings, obj.scores=obj.scores,
  loadings.rot=Rot$loadings, obj.scores.rot=scale(obj.scores)%*%Rot$rotmat)
}
```

Here is an example of the use of this function:

```
> dims <- 3
> pca <- PCA(X,dims)
```

Figure 14.5 shows the scree plot and the CVAF plot. The code for generating these graphics is:

```
> p <- length(pca$eigvals)
> plot(pca$eigvals, xaxp=c(1,p,p-1), type='b', main='Scree Plot',
+    xlab='Principal Components', ylab='Eigenvalues')
> lines(c(0,p+1),c(1,1),lty='dashed')
> text(pca$eigvals, as.character(round(pca$eigvals,digits=2)),
```

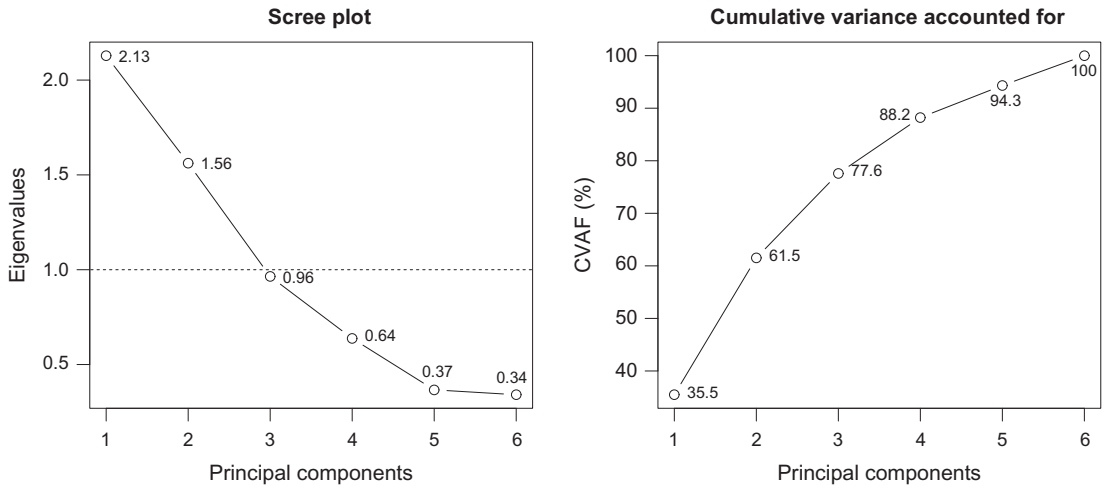


Figure 14.5
Scree plot and CVAF plot for the first PCA (home team).

```
+     cex=0.6, pos=c(4,4,4,4,3,3))
> plot(100*cumsum(pca$eigvals)/p, xaxp=c(1,p,p-1), type='b',
+     xlab='Principal Components', ylab='CVAF (%)',
+     main='Cumulative Variance Accounted For ')
> text(100*cumsum(pca$eigvals)/p,
+     as.character(round((cumsum(pca$eigvals)/p)*100,digits=1)),
+     cex=0.6, pos=c(4,4,4,2,1,1))
```

The commonly used rule for retaining a PC is that its eigenvalue be greater than 1.0. Two of our PCs meet this eigenvalue cutoff. We also choose to retain a third PC with an eigenvalue very close to 1.0(0.96). Thanks to the addition of a third PC, the percentage of CVAF increases from 61.5% to 77.6%. This gives us a total of $b = 3$ PCs.

For interpretation of the rotated solution, we refer to the loadings generated using the code:

```
> print(pca$loadings.rot, cutoff=0)
```

For ease of interpretation, a common practice is to change the sign of a PC if it is negatively correlated with the variables having the greatest weight in its linear combination. In this case we change the sign of PC1, with the following code:

```
> objs.rot.home[,1] <- -objs.rot.home[,1]
> pca$loadings.rot[,1] <- -pca$loadings.rot[,1]
```

The loadings after the sign changes are displayed in Table 14.2. The first (rotated) PC represents aerial abilities (crosses and heading) when the home team is on the attack (also referred to as on the offense), so we name it "*aerial.attack.home*." The second PC represents the ability of the home team to create the opportunity and to make shots on goal, so we name it "*shot.attack.home*." The third PC represents the capability of the home team defense, so we name it "*defense.home*."

Table 14.2 Loadings of the First PCA (home teams) after Varimax Rotation and Sign Changes

Variables	PC1 aerial.attack.home	PC2 shot.attack.home	PC3 defense.home
O_OCCAS_C	0.029	**0.698**	0.028
O_TIRI1_C	−0.018	**0.709**	−0.030
G_COLPTESTA1_C	**0.429**	−0.083	**0.341**
O_CROSS_C	**0.643**	0.036	−0.072
D_%PROT_C	−0.089	0.035	**0.936**
O_CROSS16_C	**0.627**	0.013	−0.028

The three rotated scores are stored in the `objs.rot.home` object:

```
> objs.rot.home <- pca$obj.scores.rot
```

14.4.2.2 PCA for the Away Team

In the second analysis, we select from $\boldsymbol{X}_s$ the seven variables describing the away team's abilities (O_OCCAS_O, O_PALLE2_O, O_TIRI1_O, D_RECUPERI14_O, O_%ATT_O, O_PAS4_O, D_PALLE17_O) to build a 380×7 matrix $\boldsymbol{X}$:

```
> X <- Xs[,grep("_O$", names(Xs))]
```

We can use exactly the same code lines presented for the home team PCA:

```
> dims <- 3
> pca <- PCA(X,dims)
> objs.rot.away <- pca$obj.scores.rot
```

Figure 14.6 shows the scree plot and the CVAF plot, which are analogous to those presented for the previous analysis of the home team. For the away team, we use similar criteria as for the home team (see above) to determine the number of PCs to retain and

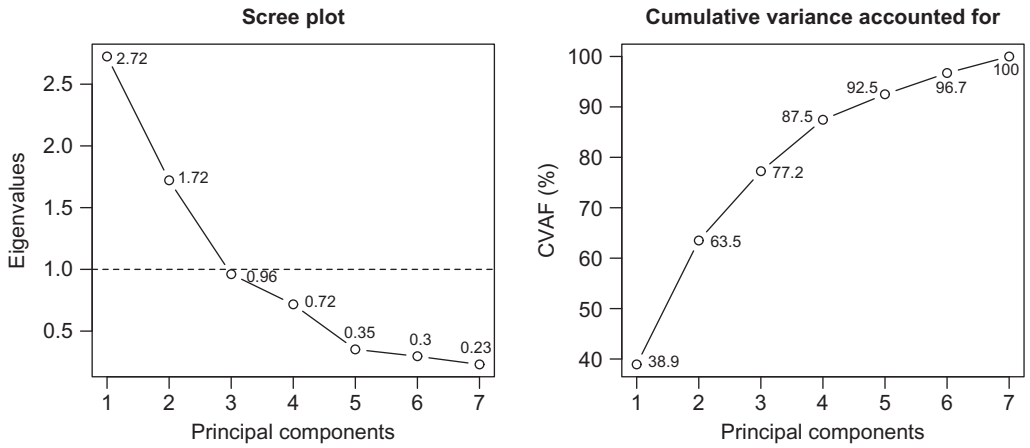


Figure 14.6
Scree plot and CVAF for the second PCA (away team).

Table 14.3 Loadings of the Second PCA (away team) after Varimax Rotation and Sign Changes

Variables	PC1 defense.away	PC2 shot.attack.away	PC3 counterattack.away
O_OCCAS_O	0.058	**0.675**	−0.022
O_PALLE2_O	0.283	−0.084	**0.504**
O_TIRI1_O	0.028	**0.660**	−0.045
D_RECUPERI14_O	**0.623**	0.136	−0.002
O_%ATT_O	**−0.443**	0.285	**0.433**
O_PAS4_O	−0.029	−0.053	**0.734**
D_PALLE17_O	**0.575**	0.011	0.129

decide whether the sign of any of the variables should be changed. As a result, we choose to retain $b = 3$ PCs, with a CVAF of 77.2%, and also to change the signs of PC1 and PC3:

```
> objs.rot.away[,c(1,3)] <- -objs.rot.away[,c(1,3)]
> pca$loadings.rot[,c(1,3)] <- -pca$loadings.rot[,c(1,3)]
```

For interpretation of the rotated solution, we refer to loadings shown in Table 14.3. The first (rotated) PC represents the defense abilities of the away team, so we name it "*defense.away*." The second PC represents the ability of the away team to create the opportunity and to make shots on goal, so we name it "*shot.attack.away*." The third PC represents the ability of the away team to make long-range kicks and sudden counterattacks, so we name it "*counterattack.away*."

Finally, the six composite indicators obtained by the two PCAs are collected in a unique data frame $\mathbf{X}_c$ with dimension (380×6) for subsequent analyses:

```
> Xc <- data.frame(objs.rot.home,objs.rot.away)
> names(Xc) <- c("air.attack.home", "shot.attack.home", "defense.home", + "defense.away",
            "shot.attack.away", "counterattack.away")
```

By construction, $\boldsymbol{X}_c$ is composed of two blocks with three standardized variables each. Within blocks, the variables are mutually uncorrelated; however, variables belonging to different blocks may show a nonzero correlation:

```
> round(cor(Xc),3)
                  air.attack.home shot.attack.home defense.home defense.away shot.attack.away counterattack.away
air.attack.home             1.000            0.000        0.000        0.593            0.109             0.384
shot.attack.home            0.000            1.000        0.000        0.036            0.008            -0.168
defense.home                0.000            0.000        1.000        0.535           -0.421            -0.365
defense.away                0.593            0.036        0.535        1.000            0.000             0.000
shot.attack.away            0.109            0.008       -0.421        0.000            1.000             0.000
counterattack.away          0.384           -0.168       -0.365        0.000            0.000             1.000
```

14.5 Model Development: Building Classifiers

According to Breiman (2001b), the use of statistical modeling to draw conclusions from data can be approached from two viewpoints: data modeling and algorithmic modeling. The former assumes that the data are generated by a given stochastic model, while the latter

assumes that the data generation mechanism is unknown. Although algorithmic modeling was not initially used in the field of statistics, in recent years it has been adopted by statisticians for use in data mining.

We begin the model development phase by building classification rules for the categorical target variable Y, approaching the problem from both data modeling and algorithmic modeling perspectives. The explanatory variables are the six key factors identified by the PCAs presented earlier. These variables are collected in matrix $\boldsymbol{X}_c$, which for simplicity we will refer to as $\boldsymbol{X}$. We denote with $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ic})$, $c = 6$, the ith row of $\boldsymbol{X}$ and with $\boldsymbol{y}$ the vector of the target variable Y, which has $m = 3$ categories (W, L, and D).

We follow a three-step procedure: (i) develop five different classifiers that can be used to predict match outcome; (ii) evaluate the performance of the each classifier and select the "best" one; (3) refine the selected classifier to enhance its performance.

Step (i) consists of a supervised learning phase where the classifier models the association between the target variable Y and the explanatory variables using a training set randomly selected from the (380×7) data set $(\boldsymbol{y}, \boldsymbol{X})$.

This allows us in step (ii) to check the performance of the trained classifiers against a testing set, evaluating their predictive accuracy with data not used in the training step. This procedure can be repeated ($S =$ repetitions), each time selecting a different training set in order to compute average performance indices, thus allowing us a more careful comparison between the different classifiers.

14.5.1 Learning Step

We consider five classification methods during the learning step, chosen to represent a wide range of approaches in statistics: two machine learning algorithms (RF and NNET), a nonparametric model (KNN), a probabilistic model (Naïve Bayes), and a model that belongs to the class of generalized linear models (multinomial logistic (MLogit) regression). It is worth mentioning that there are additional techniques that can be used in the data mining process. In addition to RF, there are other algorithms belonging to the broad tree-based ensemble learning class, for example, gradient boosting machine (Friedman, 2001; `gbm` package), conditional trees (Hothorn et al., 2006; Strobl et al., 2007; `party` package), and logic forests (Wolf et al., 2010; `LogicForest` package). Alternatively, one could choose a Bayesian network classifier (Friedman et al., 1997), such as the Bayesian belief network (Pearl, 1986; `bnlearn` package). In the machine learning context, support vector machine (Vapnik, 1998) is implemented in several R packages including `e1071`, `kernlab`, `klar`, and `svmpath`.

Our analysis starts with a random split of the dataset containing the $c = 6$ composite indicators `Xc` and the outcome `y` into two parts: a learning set and a testing set. The learning set (`learn`)

contains data from 300 matches used to train the five classifiers. The remaining 80 matches (`test`) are used to evaluate the "out-of-sample" performance of the classifiers. We randomly select 80 of the 380 available matches using the `sample` command:

```
> dtset.ind <- data.frame(Xc, y)
> set.seed(987654)
> idx <- sample(1:nrow(dtset.ind),80)
```

and then we build the learning and testing sets:

```
> learn <- dtset.ind[-idx,]
> test <- dtset.ind[idx,]
```

The training of the five classifiers is performed using the package `caret`, short for classification and regression training (Kuhn et al., 2012). This package is an interesting set of tools for training and testing a large number of classification and regression algorithms and models. Using the `train` command in `caret`, it is possible to easily tune (using resampling methods such as cross-validation, repeated cross-validation, or bootstrapping) the critical parameters of a selected method and to identify the model/algorithm that provides the most accurate prediction. Another interesting feature of `caret` is that it can use the parallel processing features of R (Kuhn, 2008). Many parallel computing techniques can work in conjunction with this package, including NetworkSpaces or MPI.

14.5.1.1 Random Forest

We use the RF algorithm already introduced in Section 4.1. Here, we employ the `doParallel` package (Weston and Calaway, 2012) and the `caret` package to build an RF classifier:

```
> library(caret)
> library(doParallel)
> clus <- makeCluster(spec=6, type='PSOCK')
socket cluster with 6 nodes on host 'localhost'
> registerDoParallel(clus)
```

To set the parameters controlling the train function, we select a repeated cross-validation resampling method with 15 complete sets of 10-folds:

```
> ctrl.train <- trainControl(method='repeatedcv',number=10,repeats=15)
```

We perform the tuning and training of the RF using the `train` command with the option `method='rf'`. The parameter that needs tuning is `mtry`, the number h of variables randomly sampled as candidates at each split. Setting `tuneGrid=expand.grid(.mtry=1:6)`, `train` varies `mtry` from 1 to 6 and stores the model with the highest accuracy (`metric='Accuracy'`) in the `finalModel` object of the output list (`fit.rf`):

```
> fit.rf <- train(y~., data=learn, method='rf', metric='Accuracy',
+            tuneGrid=expand.grid(.mtry=1:6),trControl=ctrl.train,
+            ntree=1000)
> stopCluster(clus)
```

The `print` (and `plot`) command of the `fit.rf` object shows that the estimated value of the optimal `mtry` parameter is 1. An important warning: Reproducibility of results

cannot be achieved when using parallel processing with the current version of `caret`. Hence, the output of the `train` command reported below is only an example of what one can obtain:

```
> print(fit.rf)

300 samples
  6 predictors
  3 classes: 'W', 'L', 'D'

No pre-processing
Resampling: Cross-Validation (10 fold, repeated 15 times)

Summary of sample sizes: 269, 271, 269, 271, 271, 270, ...

Resampling results across tuning parameters:

mtry  Accuracy  Kappa  Accuracy  SD  Kappa  SD
1     0.609     0.368  0.0787        0.131
2     0.607     0.369  0.0727        0.12
3     0.602     0.362  0.0724        0.118
4     0.603     0.365  0.0724        0.117
5     0.602     0.365  0.0725        0.117
6     0.606     0.371  0.0734        0.119

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 1.
```

The RF out-of-sample classification of the 80 matches in the training set is obtained using `predict` with the `type='class'` option, which assigns to the *i*th profile the category (class *W*, *L*, or *D*) with the highest estimated class probability:

```
>  y.rf  <- predict(fit.rf$finalModel, newdata=test[,1:6],type='class')
```

14.5.1.2 Neural Network

A NNET is a set of learning methods based on the estimation of nonlinear combinations of the covariates to model a response variable. NNETs are composed of simple computational units (neurons), structured on different levels and interconnected through a defined system of linkages (architecture). In this case study, we consider a multilayer perceptron (MLP) with $c = 6$ neurons in the input layer, one for each explanatory variable. The number of neurons in the output layer is $m = 3$, one for each category of Y. The number of hidden layers and the number of neurons in each hidden layer is determined by the investigator.

For ease of explanation, we consider the case of a NNET with a single hidden layer, h hidden neurons, and no bias neurons. The ith element of the kth output node, $O_{ik} \in \{0,1\}$, is equal to 1 if the classification NNET assigns to the ith feature vector $\boldsymbol{x}_i$ the kth level of Y. Otherwise, the output node is equal to 0. The outputs are modeled as functions of the linear combinations (called activation states: $\boldsymbol{A}_o$) of the variables in the hidden layer $\boldsymbol{Z}$, which in

turn are (nonlinear) functions of the linear combinations (activation states $\boldsymbol{A}_z$) of the covariates X_j. Following Hastie et al. (2001), the NNET is specified by:

$$\boldsymbol{A}_z = \boldsymbol{X}\cdot\boldsymbol{W}_1; \quad \boldsymbol{Z} = f_1(\boldsymbol{A}_z)$$
$$\boldsymbol{A}_o = \boldsymbol{Z}\cdot\boldsymbol{W}_2; \quad \boldsymbol{O} = f_2(\boldsymbol{A}_o)$$

where

- $\boldsymbol{Z}$ is the $n \times h$ output matrix of hidden neurons and $\boldsymbol{O}$ is the $n \times m$ matrix of the NNET outputs;
- $\boldsymbol{W}_1$ is the $c \times h$ matrix of weights for connections between inputs and hidden neurons; $\boldsymbol{W}_2$ is the $h \times m$ weight matrix for connections between hidden neurons and outputs;
- The elements of the $\boldsymbol{A}_z$ and $\boldsymbol{A}_o$ matrices are the activation states of the hidden and output neurons, respectively;
- f_1,f_2 are the activation functions that specify the response of the neuron to the activation state; typically these functions act as squashing functions, such that the output of a neuron lies within a given interval of the real line (for example, sigmoid function or hyperbolic tangent).

Here is the code for training the classifier based on a MLP with a single layer of hidden neurons:

```
> clus <- makeCluster(spec=6, type='PSOCK')
> registerDoParallel(clus)
```

The `train` command performs the tuning and training of the MLP (`method='mlp'`) by standard backpropagation, with the number of hidden neurons (h) ranging from one to 15 (`tuneGrid=expand.grid(.size=1:15)`). As before, the model with the highest accuracy (`metric='Accuracy'`) will be selected and stored in the `finalModel` object of the output list:

```
> fit.nnet <- train(y~., data=learn, method='mlp', metric='Accuracy',
+             tuneGrid=expand.grid(.size=1:15),trControl=ctrl.train)
> stopCluster(clus)
```

The `summary` command gives a detailed description of the optimal NNET structure:

```
> summary(fit.nnet$finalModel)

SNNS network definition file V1.4-3D
generated at Thu Jul 26 17:31:48 2012

network name   :  RSNNS_untitled
source files   :
no. of units   :  13
no. of connections   :  36
no. of unit types   :  0
no. of site types   :  0

learning function  :  Std_Backpropagation
update function    :  Topological_Order
```

```
unit default section :
act      | bias    | st | subnet | layer | act func      | out func
---------|---------|----|--------|-------|---------------|----------
0.00000  | 0.00000 | i  | 0      | 1     | Act_Logistic  | Out_Identity
---------|---------|----|--------|-------|---------------|----------

unit definition section :
no. | typeName | unitName   | act      | bias     | st | position | act func      | out func | sites
----|----------|------------|----------|----------|----|----------|---------------|----------|------
  1 |          | Input_1    | -0.21367 |  0.09263 | i  | 1,0,0    | Act_Identity  |          |
  2 |          | Input_2    |  0.03144 | -0.08992 | i  | 2,0,0    | Act_Identity  |          |
  3 |          | Input_3    | -0.36410 |  0.05476 | i  | 3,0,0    | Act_Identity  |          |
  4 |          | Input_4    | -0.15385 | -0.08583 | i  | 4,0,0    | Act_Identity  |          |
  5 |          | Input_5    | -0.44387 |  0.10217 | i  | 5,0,0    | Act_Identity  |          |
  6 |          | Input_6    | -0.95274 | -0.20035 | i  | 6,0,0    | Act_Identity  |          |
  7 |          | Hidden_2_1 |  0.13939 | -1.84450 | h  | 1,2,0    |               |          |
  8 |          | Hidden_2_2 |  0.00457 | -1.22991 | h  | 2,2,0    |               |          |
  9 |          | Hidden_2_3 |  0.01680 | -1.92949 | h  | 3,2,0    |               |          |
 10 |          | Hidden_2_4 |  0.68588 | -2.74513 | h  | 4,2,0    |               |          |
 11 |          | Output_1   |  0.86291 | -0.97549 | o  | 1,4,0    |               |          |
 12 |          | Output_2   |  0.05260 | -0.93177 | o  | 2,4,0    |               |          |
 13 |          | Output_3   |  0.16987 |  0.06701 | o  | 3,4,0    |               |          |
----|----------|------------|----------|----------|----|----------|---------------|----------|------

connection definition section :
target | site | source:weight
-------|------|-----------------------------------------------------
  7    |      |  6: 1.90252,  5:-3.13627,  4: 0.33107,  3:-0.87872,  2: 2.42884,  1:-0.46435
  8    |      |  6: 3.57530,  5:-0.59082,  4: 2.38882,  3: 1.14327,  2:-1.13969,  1: 0.88770
  9    |      |  6: 0.37165,  5: 2.95812,  4: 0.51659,  3: 0.71586,  2:-1.93326,  1: 0.33686
 10    |      |  6:-2.92389,  5:-1.83936,  4: 2.17533,  3: 1.54707,  2: 3.19895,  1:-3.37565
 11    |      | 10: 3.48102,  9:-0.46043,  8:-2.72008,  7: 3.21276
 12    |      | 10:-2.38589,  9: 2.11569,  8: 0.80565,  7:-2.59738
 13    |      | 10:-1.96299,  9:-3.39130,  8: 1.92892,  7:-1.85862
-------|------|-----------------------------------------------------
```

The output shows that the optimal NNET has four neurons in the hidden layer. In the connection definition section, we find the input-hidden weights $\boldsymbol{W}_1$ in the first four rows and the hidden-output weights $\boldsymbol{W}_2$ in the last three rows.

For objects of class `nnet`, the `predict` command returns only class probabilities:

```
> probs.nnet <- predict(fit.nnet$finalModel, newdata=test[,1:6])
> head(probs.nnet)

          [,1]         [,2]       [,3]
320  0.3689758  0.235318542  0.2666312
360  0.7592122  0.046294659  0.2143974
277  0.2315165  0.388963103  0.2892783
193  0.2607177  0.348797619  0.2837963
150  0.9358161  0.008980305  0.1737476
188  0.8515148  0.024579775  0.1977452...
```

For each of the 80 matches in the training set, the predicted result is the category (W, L, or D) with the highest estimated class probability:

```
> y.nnet <- apply(probs.nnet,1,which.max)
> y.nnet <- factor(y.nnet,levels=1:3,labels=levels(dtset$y))
```

As in RF, NNET also cannot achieve the reproducibility of results due to the use of `caret` package in conjunction with parallel computation. Because of the stability of tree-based ensemble learning algorithms, the output of RF does not significantly change from one run to another. Conversely, with NNET we obtain markedly different solutions. Following Breiman (2001b), model instability occurs when there are many different models that have approximately the same in-sample or out-of-sample prediction error, and a slight perturbation of the data or in the model estimation causes a skip from one model to another. These different models are close to each other in terms of error, but can be simultaneously distant in terms of model form. This is similar to what happens with NNET.

14.5.1.3 k-Nearest Neighbor Algorithm

The KNN algorithm is one of the simplest machine learning algorithms: It assigns to the profile or feature vector $\boldsymbol{x}_i$ the most common modality of Y among its k "nearest neighbors." The number of neighboring profiles to be considered is equal to k. When $k=1$, the algorithm assigns to $\boldsymbol{x}_i$ the category y of its nearest neighbor; when $k>1$, the algorithm assigns to $\boldsymbol{x}_i$ the most common category y among its nearest neighbors.

The training phase of the algorithm consists of storing the profiles of the training sample and the corresponding categories. In the classification phase, we define k, and we classify the profiles of the testing set. Here is the R code for the KNN classifier:

```
> clus <- makeCluster(spec=6, type='PSOCK')
> print(clus)
> registerDoParallel(clus)
```

The `train` command tunes the KNN method (`method='knn')`, with k ranging from 5 to 100 (`tuneGrid=expand.grid(.k=5:100)`). We choose the value of k that maximizes the accuracy of the final model (`metric='Accuracy'`):

```
> fit.knn <- train(y~., data=learn, method='knn',
+           tuneGrid=expand.grid(.k=5:100),
+           metric='Accuracy', trControl=ctrl.train)
> stopCluster(clus)
> yhat.knn<-predict(fit.knn$finalModel,newdata=test[,1:6],type="class")
```

14.5.1.4 Naïve Bayesian Classification

The Naïve Bayesian (NBayes) classification algorithm is based on a probabilistic model that incorporates strong (naïve) independence assumptions. It postulates that, given a response category $Y_k \in \{W,D,L\}$, a particular characteristic assumed by a covariate X_j is independent of

any other feature. For the ith profile $\boldsymbol{x}_i$, the classifier assigns the category Y_k with the highest posterior probability conditioned on $\boldsymbol{x}_i$:

$$P(Y_k|\boldsymbol{x}_i) > P(Y_l|\boldsymbol{x}_i) \, for \; 1 \leq l \leq m, l \neq k.$$

Using Bayes' theorem, we can compute the three posterior probabilities using the formula:

$$P(Y_k|\boldsymbol{x}_i) = \frac{P(\boldsymbol{x}_i|Y_k) \cdot P(Y_k)}{P(\boldsymbol{x}_i)} \text{ with } k = 1, 2, \ldots, m.$$

As $P(\boldsymbol{x}_i)$ is constant for all categories, we need to consider only $P(\boldsymbol{x}_i|Y_k) \cdot P(Y_k)$. The category prior probabilities can be easily estimated by $P(Y_k) = n_k/n$, where n_k is the number of the n training profiles of $\boldsymbol{X}_\mathrm{T}$ belonging to category Y_k. Because of the naïve assumption of conditional independence, the covariate values are conditionally independent from one another:

$$P(\boldsymbol{x}_i|Y_k) = \prod_{j=1}^{c} P\left(x_{ij}|Y_k\right).$$

NBayes is implemented in the `klaR` package (Weihs et al., 2005):

```
> library(klaR)
```

The `NaiveBayes` command in this package estimates the parameters of the probabilistic classifier using a Gaussian probability distribution for $P(x_{ij}|Y_k)$ with conditioned μ and σ parameters:

```
> fit.NB <- NaiveBayes(y~., data=learn)
```

The `predict` command gives posterior probabilities (`probs.NB`) as well as the categories (`y.nb`) with the highest predicted posterior probabilities:

```
> pred.NB <- predict(fit.NB, newdata=test)
> probs.NB <- pred.NB$posterior
> y.nb <- pred.NB$class
```

14.5.1.5 Multinomial Logistic Regression Model

MLogit regression is a generalized linear model used to estimate the probabilities for the m categories of a qualitative dependent variable Y, using a set of explanatory variables $\boldsymbol{X}$:

$$\Pr(Y_{ik}) = \Pr(Y_i = k \,|\, \boldsymbol{x}_i; \boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m) = \frac{\exp\left(\beta_{0k} + \boldsymbol{x}_i \boldsymbol{\beta}_{\boldsymbol{k}}^{'}\right)}{\sum_{j=1}^{\mathrm{m}} \exp\left(\beta_{0j} + \boldsymbol{x}_i \boldsymbol{\beta}_{\boldsymbol{j}}^{'}\right)} \text{ with } k = 1, 2, \ldots, m$$

where $\boldsymbol{\beta}_k$ is the row vector of regression coefficients of $\boldsymbol{X}$ for the kth category of Y.

Unfortunately, this model specification is nonidentifiable; the probabilities are identical for each $\beta_{0k} + q, q \in R$ and $\boldsymbol{\beta_k} + \boldsymbol{q}, \boldsymbol{q} \in \boldsymbol{R^c}$. Hence, some convenient normalization method has to be

applied. In the `mlogit` command from the `globaltest` package, the ability to identify model parameters is ensured by the requirement that the sum of regression coefficients over the m categories is zero:

```
> library(globaltest)
> fit.mlog1 <- mlogit(y ~., data=learn)
> print(cf.mlog1 <- fit.mlog1@coefficients)

                            W           L             D
(Intercept)         0.4840504  -0.5634251   0.079374777
air.attack.home    -0.5505973   0.2900597   0.260537580
shot.attack.home    0.9069024  -0.8025341  -0.104368340
defense.home       -0.1258031   0.1324323  -0.006629216
defense.away       -0.0331553   0.1782980  -0.145142731
shot.attack.away   -0.7787624   1.0660781  -0.287315651
counterattack.away -0.5057221   0.3801720   0.125550132
```

Another convenient normalization choice for solving the identification problem is to assume that $\beta_{01}=0$ and $\boldsymbol{\beta}_1=0$ so that:

$$\Pr(Y_{i1}) = \frac{1}{1+\sum_{j=2}^{m}\exp\left(\beta_{0j}+\boldsymbol{x}_i\boldsymbol{\beta}_j^{'}\right)}$$

and

$$\Pr(Y_{ik}) = \frac{\exp\left(\beta_{0k}+\boldsymbol{x}_i\boldsymbol{\beta}_k^{'}\right)}{1+\sum_{j=2}^{m}\exp\left(\beta_{0j}+\boldsymbol{x}_i\boldsymbol{\beta}_j^{'}\right)} \text{ with } k=2,3,\ldots,m.$$

Note that this model adopts a linear predictor function to estimate the natural logarithm of the probability (log-odds) ratio for the kth category of Y:

$$\ln\left(\frac{\Pr(Y_{ik})}{\Pr(Y_{i1})}\right) = \beta_{0k}+\boldsymbol{x}_i\boldsymbol{\beta}_k^{'} \text{ with } k=2,3,\ldots,m.$$

With this parameterization, the jth element of vector $\boldsymbol{\beta}_k$ can be interpreted as the increase in the log-odds ratio of the kth category versus the reference category (in this case W) resulting from a one-unit increase in the jth covariate, holding the other covariates constant to their mean values.

In this case, considering the $n=380$ observations and the observed target variable category y_i for the ith profile $\boldsymbol{x}_i$, the estimate of the parameter vector $\boldsymbol{\beta}$ of length $(m\text{-}1)\cdot(c+1)=2\cdot 7=14$ is obtained, maximizing the log-likelihood function:

$$\max_{\boldsymbol{\beta}\in\mathbb{R}^{(m-1)\cdot(c+1)}} \sum_{i=1}^{n}\ln[Pr(Y_i=y_i\,|\,\boldsymbol{x}_i;\boldsymbol{\beta})].$$

This version of MLogit is implemented using the `multinom` command of the `nnet` package (Venables and Ripley, 2002):

```
> library(nnet)
> fit.mlog2 <- multinom(y ~., data=learn)
> print(t(coef(fit.mlog2)))
```

```
                            L           D
(Intercept)        -1.0474669  -0.4046781
air.attack.home     0.8406374   0.8111295
shot.attack.home   -1.7094407  -1.0112680
defense.home        0.2582328   0.1191700
defense.away        0.2114578  -0.1119894
shot.attack.away    1.8448428   0.4914361
counterattack.away  0.8858884   0.6312683
```

Again, `predict` yields the out-of-sample classification for the multinomial logistic model:

```
> y.mlog <- predict(fit.mlog2, newdata=test)
```

14.5.2 Model Selection

The second step of model evaluation is model selection. After developing the five classifiers, we define the criteria for evaluating their performances. The dilemma of Occam's razor is well discussed in the literature. In terms of model evaluation, we interpret Occam's razor as the need to find a trade-off between accuracy and simplicity. Usually, complex models/algorithms such as NNETs produce good predictions, but their interpretability is often very limited. On the other hand, simple statistical models such as logistic regression are easily interpretable, but they are often a poor fit to the data (Breiman, 2001b).

The performance of the five classifiers A through E for our target variable Y (with $m = 3$ categories) can be described by the "confusion matrix", a squared contingency table with m rows (the categories predicted by the model) and m columns (the categories observed in the sample). Many indices have been proposed for evaluating the predictive performance of classifiers. Two "overall" measures of model performance are the Accuracy and Kappa statistics. *Accuracy* (a) is the proportion of observations correctly classified, i.e., the ratio between the sum of the diagonal elements of the confusion matrix and the sample size n. Cohen's *Kappa* (κ) statistic corrects the a statistic by including the probability of a correct classification occurring by chance:

$$\kappa = \frac{a - e}{1 - e},$$

where e is the estimated probability of a chance agreement, computed as the sum of the products of the pairs of marginal frequencies related to the same predicted and observed categories. Indices a and κ assume values in [0;1] and are used with the simple decision rule "greater is better."

There is also a set of indices for assessing the performance of classifiers for each single category of the target variable Y that can be computed using the m reduced (2×2) confusion matrices obtained by collapsing $(m\text{-}1)$ rows and columns of the original $(m \times m)$ confusion matrix (i.e., adopting the "one-versus-all" criteria). For our performance assessment ($m = 3$), we use three *Sensitivity* (s_k) indices:

$$s_k = \frac{n_{kk}}{n_k} \text{ with } k = 1, 2, 3$$

where s_k is the proportion of cases of the kth category of the target variable Y that are correctly predicted by the model (n_{kk}) compared to the total number of cases in the kth category (n_k).

The `confusionMatrix` command of the `caret` package cross-tabulates observed and predicted categories, producing a confusion matrix (3×3 for our match outcome variable). In addition, it produces estimates of a number of performance indices for classifiers: overall accuracy and Kappa statistics, sensitivity, specificity, positive/negative predicted values, and more. For the multinomial model of the previous section, we get the following results:

```
> CM.mlogit  <-  caret::confusionMatrix(y.mlog, test$y)
> print(CM.mlogit)

Confusion Matrix and Statistics

          Reference
Prediction  W   L   D
         W 36  3  10
         L  3 11   6
         D  1  6   4

Overall Statistics
               Accuracy : 0.6375
                 95% CI : (0.5224, 0.7421)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.009158

                  Kappa : 0.3927
 Mcnemar's Test P-Value : 0.061168

Statistics by Class:

                     Class: W  Class: L  Class: D
Sensitivity            0.9000    0.5500    0.2000
Specificity            0.6750    0.8500    0.8833
Pos Pred Value         0.7347    0.5500    0.3636
Neg Pred Value         0.8710    0.8500    0.7681
Prevalence             0.5000    0.2500    0.2500
Detection Rate         0.4500    0.1375    0.0500
Detection Prevalence   0.6125    0.2500    0.1375
```

To evaluate the variability of results, the code of Section 5.1 is replicated 1000 times. For each replication, the data set is randomly split into a learning set (`learn`, 300 matches) and a testing

set (`test`, 80 matches). For each learning set, the five classifiers are built, and their performances are evaluated for the corresponding testing set by computing the performance indices a, k, and s_k. Due to the computational burden of some algorithms (e.g., NNETs), this procedure is time consuming: the time required for 100 replications is 8.5 h using the iMAC (one 4-core processor Intel Core i7 with 8 Gb RAM DDR3). The R code for this analysis is included in the Supplementary Materials.

Table 14.4 shows summary statistics (average, standard deviation, and percentage coefficient of variation) for the results obtained from 1000 replications for each classifier. The overall performances of the five classifiers are similar: average Accuracy ranges from 62% (NNET) to 65% (MLogit), and average Kappa ranges from 39% (RF) to 44% (MLogit). Thus, on the basis of these two indices, the MLogit classifier shows a modest performance advantage.

Classifier performance for each category of the target variable Y is evaluated using the sensitivity index (s_k). The best and the worst predictions are for the categories W (win) and D (draw), respectively. The range of the average sensitivity for W predictions is 80% (NNET) to 85% (KNN), the sensitivity for L ranges from 56% (NBayes) to 70% (NNET), and for D the range is 22% (NNET) to 36% (NBayes). There is no one classifier that clearly outperforms the other four, but MLogit is the second best classifier for each of the three categories.

Figure 14.7 shows, using box-plots, the distributions of the performance indices from 1000 replications for each of the five classifiers. These graphs confirm the conclusions drawn from Table 14.4.

These results suggest that the MLogit classifier is the model with the best balance between complexity and accuracy of predictions. In addition, MLogit presents the relationship between outcome and covariate in an understandable way (interpretability). Hence, it is reasonable to use MLogit to predict the final outcome of football matches in our dataset.

In the next section, we examine this predictive model and try to improve target classification, paying special attention to the D (draw) outcome.

Table 14.4 Statistics of the Performance Indices for the Five Classifiers (1000 Replications)

Indices	Accuracy			Kappa			Sensitivity *W*			Sensitivity *L*			Sensitivity *D*		
Classifiers	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%
RF	0.63	0.05	7.5	0.39	0.07	17.6	0.83	0.06	7.8	0.59	0.11	18.5	0.29	0.09	32.5
NNET	0.62	0.05	7.7	0.40	0.07	17.2	0.80	0.08	9.5	0.70	0.14	19.7	0.22	0.16	72.7
KNN	0.64	0.05	7.6	0.42	0.07	17.2	0.85	0.06	6.8	0.61	0.11	17.5	0.30	0.10	34.4
NBayes	0.63	0.05	7.6	0.40	0.07	17.6	0.82	0.06	7.8	0.56	0.10	18.2	0.36	0.10	27.4
MLogit	0.65	0.05	7.4	0.44	0.07	16.3	0.83	0.06	7.5	0.65	0.10	15.4	0.32	0.10	32.0

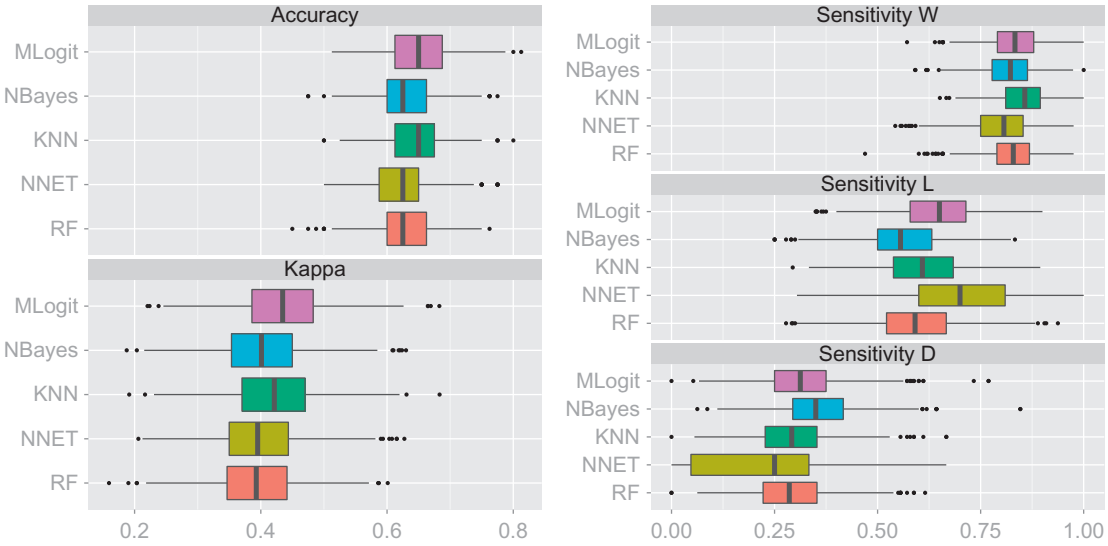


Figure 14.7
Box-plots of the performance indices for the five classifiers (1000 replications).

14.5.3 Model Refinement

The performance analysis described in the Section 14.5.2 highlights some interesting facts. Despite the simplicity and the intrinsically linear nature of MLogit regression, this model compares favorably with more complex methods. This suggests that the relationships under investigation could be characterized by an essentially linear structure. However, the predictive ability of the model is substantially different for the three categories of Y, with the D (draw) result being particularly difficult to predict. This could be due, in part, to the problem of class imbalance.

In the context of machine learning, several approaches have been proposed. Notable methods for machine learning using imbalanced data include oversampling, undersampling, boosting, bagging, and repeated random subsampling (Khalilia et al., 2011). Chawla et al. (2004) provide an interesting historical overview of techniques described in the machine learning literature. A more recent article from He and Garcia (2009) presents a useful comprehensive review of the analysis of unbalance data using machine learning.

The `caret` package has two commands (`downSample` and `upSample`) for simple random down- and up-sampling of imbalanced data. Chawla et al. (2002) proposed the synthetic minority oversampling technique. The basic idea behind this technique is to artificially generate new examples of the minority class using the nearest neighbors of these cases, simultaneously

undersampling the other classes in order to obtain a more balanced dataset. This method is implemented in the `DMwR` package of R using the `SMOTE` function (Torgo, 2010). Qiao and Liu (2009) proposed three weighted learning procedures: (1) consider only the relative class frequency; (2) take into account the within-group misclassification rates; and (3) adaptively modify the weights according to both the classification information and the estimated class frequencies.

Since the data in our case study are not strongly imbalanced, we can limit ourselves to a simple refinement of MLogit by assigning different weights to the observations in order to rebalance the categories of the target variable. Since the frequency of *W* is about twice that of *L* or *D*, the matches won by the home team (*W*) are weighted 0.5 and the *L* and *D* matches are weighted 1:

```
> wgt <- rep(1,length(learn$y))
> wgt[learn$y=="W"] <- .5
```

Then, the model is fitted using the command `multinom`, as described earlier:

```
> fit.mlogit.bal <- multinom(y ~., weights=wgt, data=learn)
```

and relative risk ratios are computed:

```
> print(t(exp(coef(fit.mlogit.bal))))
                           L         D
(Intercept)        0.7015747 1.3394566
air.attack.home    2.2779604 2.2342969
shot.attack.home   0.1740850 0.3551985
defense.home       1.3942270 1.1633536
defense.away       1.1711399 0.8592654
shot.attack.away   6.9808374 1.6829456
counterattack.away 2.5963145 1.9968368
```

Finally, predictive ability is evaluated using the same reasoning described in Section 5.2:

```
> y.mlogit.bal <- predict(fit.mlogit.bal,newdata=test)
> CM.mlogit <- caret::confusionMatrix(y.mlogit.bal, test$y)
> print(CM.mlogit)
Confusion Matrix and Statistics

          Reference
Prediction  W  L  D
         W 31  1  6
         L  3 11  6
         D  6  8  8
Overall Statistics
               Accuracy : 0.625
                 95% CI : (0.5096, 0.7308)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.0165

                  Kappa : 0.4059
 Mcnemar's Test P-Value : 0.7325
```

```
Statistics by Class:

                     Class: W  Class: L  Class: D
Sensitivity            0.7750    0.5500    0.4000
Specificity            0.8250    0.8500    0.7667
Pos Pred Value         0.8158    0.5500    0.3636
Neg Pred Value         0.7857    0.8500    0.7931
Prevalence             0.5000    0.2500    0.2500
Detection Rate         0.3875    0.1375    0.1000
Detection Prevalence   0.4750    0.2500    0.2750
```

Results are summarized in Table 14.5 and compared with those reported in Table 14.4 for the imbalanced MLogit classifier.

As expected, the proposed refinement improves the sensitivity for category *D* but yields a corresponding worsening of sensitivity for category *W* so that the overall performance, according to both the Accuracy and the Kappa index, remains broadly unchanged.

One may choose to use either the MLogit regression or its balanced refinement based on the goal of the analysis. From a statistical perspective, a model that can satisfactorily predict all categories is preferable; however, from a practical point of view, a coach might be more interested in the key factors that would help his team win the match. In that case, he would prefer the model with the best sensitivity for the *W* outcome regardless of the sensitivity for the *D* or *L* classes.

14.6 Model Deployment

In the final step of the analysis, we use the fitted imbalanced and balanced MLogit models to extract insightful information about key factors affecting the match outcome. We accomplish this by interpretation of relative risk ratios. Relative risk ratios can be computed from the two formulations for the unbalanced model (`mlogit` and `multinom` in Section 5.1.5) as follows:

```
> cbind(exp(cf.mlog1[,2])/exp(cf.mlog1[,1]),
        exp(cf.mlog1[,3])/exp(cf.mlog1[,1]))
                          L         D
(Intercept)         0.3508223 0.6671932
air.attack.home     2.3178895 2.2504606
```

Table 14.5 Statistics of the Performance Indices for the MLogit Classifiers (1000 Replications)

Indices	Accuracy			Kappa			Sensitivity *W*			Sensitivity *L*			Sensitivity *D*		
Classifiers	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%	Ave	SD	CV%
MLogit	0.65	0.05	7.6	0.43	0.07	17.1	0.83	0.06	7.6	0.64	0.10	15.5	0.32	0.10	31.7
Balanced MLogit	0.63	0.05	7.8	0.42	0.07	17.1	0.70	0.08	10.9	0.67	0.10	15.5	0.46	0.11	23.8

```
shot.attack.home    0.1809677  0.3637564
defense.home        1.2946436  1.1265658
defense.away        1.2354723  0.8940555
shot.attack.away    6.3270905  1.6346795
counterattack.away  2.4251516  1.8800008
```

and

```
> t(exp(cf.mlog2))

                           L          D
(Intercept)         0.3508253  0.6671915
air.attack.home     2.3178440  2.2504484
shot.attack.home    0.1809670  0.3637575
defense.home        1.2946402  1.1265614
defense.away        1.2354778  0.8940538
shot.attack.away    6.3271054  1.6346620
counterattack.away  2.4251379  1.8799934
```

The relative risk ratios for both the imbalanced and balanced models are displayed in Table 14.6. The results of the two models are largely similar. Each model shows that increases in the factor "*shot.attack.home*" reduce the probability of both *L* and *D* with respect to *W*. This turns out to be the most important game strategy in terms of the probability of winning the match. Surprisingly, "*aerial.attack.home*" tends to favor the probability of *L* and *D*, meaning that this game strategy tends to be very poorly effective or even counterproductive. On the other hand, increments in "*shot.attack.away*" largely increase the probability of *L* and increments in "*counterattack.away*" increase the probability of both *L* and *D* with respect to *W*, but only to a minor extent. It is worth noting that factors related to the defense play only a marginal role in determining match outcome.

Some insightful graphics can be created in order to visualize the relationships identified by our classifiers. The first graph is a partial dependence plot that depicts the marginal effect of a variable on estimated outcome probabilities. Partial dependence plots are useful tools especially when predictions are obtained by means of "black box" machine learning

Table 14.6 Relative Risk Ratios Estimated by Unbalanced and Balanced Multinomial Logistic Models

	MLogit		Balanced MLogit	
Variables	***L***	***D***	***L***	***D***
aerial.attack.home	2.32	2.25	2.28	2.23
shot.attack.home	0.18	0.36	0.17	0.36
defense.home	1.29	1.13	1.39	1.16
defense.away	1.24	0.89	1.71	0.86
shot.attack.away	6.33	1.63	6.98	1.68
counterattack.away	2.43	1.88	2.60	2.00

algorithms. In fact, they are often implemented in the corresponding R packages (for example, the function `partialPlot()` in the package `randomForest`). In these cases, partial dependence plots provide some information about the unknown mechanism generating predictions. In our case study, the function explaining the relation between the variables in $\boldsymbol{X}$ and the probability of the match outcomes is known to be logistic, so we use the graphs only for demonstrative purposes. Partial dependence plots for *shot.attack.home* and *shot.attack.away* are shown in Figure 14.8. The code for the plot on the left generates a data matrix `X0` with `k`=200 rows and six columns (one for each covariate). The selected predictor varies within a given range. In our example, the range is from −3.0 to 3.0 and includes 200 equally spaced values. The other predictors are held constant to their mean values:

```
> x.name <- "shot.attack.home"
> j <- which(names(learn) %in% x.name)
> k <- 200
> x <- seq(-3,3,length.out=k)
> X0 <- matrix(0,k,6)
> X0[,j] <- x
> dimnames(X0)[[2]] <- names(learn)[1:6]
```

We then use the fitted model to estimate class probabilities corresponding to the values in `X0`:

```
> probs.mlog <- predict(fit.mlog2, newdata=X0, type="probs")
```

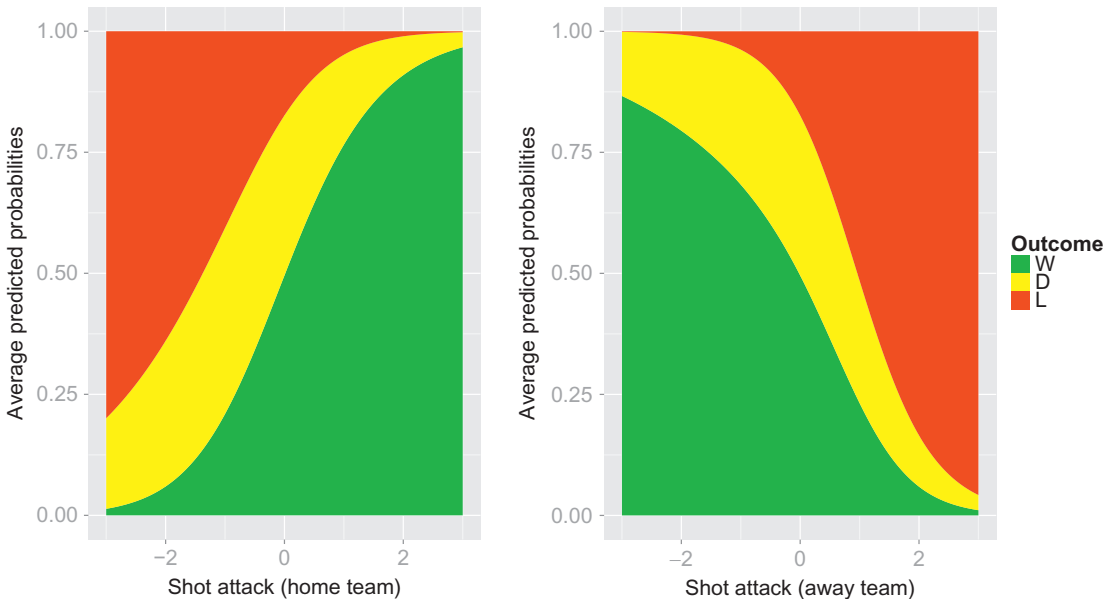


Figure 14.8
Area graphs of the estimated probabilities for $\boldsymbol{Y}$ given "*shot.attack.home*" (left) and "*shot.attack.away*" (right).

Finally, the estimated probabilities are plotted using the `geom_area` command, which is available in the `ggplot2` package, to produce an area graph (see Figure 14.2):

```
> dtst  <-  data.frame(Xj=rep(x,3), Prbs=c(probs.mlog[,c(1,3,2)]),
+           Outcome=factor(rep(1:3,each=k),labels=c("W","D","L")))
> ggplot(dtst, aes(x=Xj, y=Prbs, group=Outcome, fill=Outcome)) +
+    geom_area(position="fill") +
+    scale_x_continuous(name="Shot attack (home team)") +
+    scale_y_continuous(name="Average predicted probabilities") +
+    scale_fill_manual(values=c("W"="green","D"="yellow","L"="red"))
```

Further insight can be obtained by identifying the most challenging cases, the matches with the highest prediction error. To this aim, we plotted a bar chart of the probability assigned by the model to the category corresponding to the real result for each of the 380 matches (Figure 14.9).

The green bars denote matches that are correctly predicted by the model (i.e., the model assigned the highest probability to the true outcome). Orange and red bars denote matches with incorrect outcome predictions for which the difference between the highest estimated probability and the probability assigned to the true outcome is either low or high (i.e., lower or greater than a given threshold), respectively. This graph allows us to identify matches that, for some reason, did not follow the usual pattern. For example, a match could have an outcome difficult to predict by means of the statistical model because of unpredictable factors such as refereeing mistakes, extraordinary game strategies, changes in the team roster, or even match fixing.

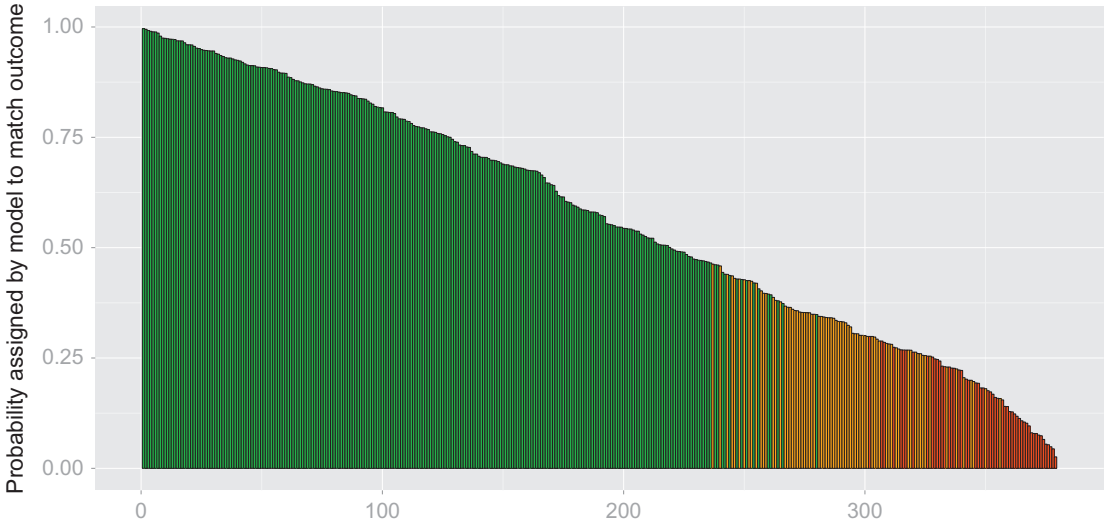


Figure 14.9
Bar chart of the probabilities assigned by the model to the true outcomes for all *380* matches.

First, for each match we estimate the three class probabilities: the probability assigned to the true outcome (`assign.pr`), the highest probability (`max.pr`), and the difference between the highest and assigned probabilities (`diff.prob`):

```
> probs.mlog <- predict(fit.mlog2, newdata=dtset, type="probs")
> assign.pr <- apply(cbind(probs.mlog,dtset$y),1,function(x){x[x[4]]})
> idx <- order(assign.pr, decreasing=T)
> assign.pr <- assign.pr[idx]
> max.pr <- apply(probs.mlog,1,max)[idx]
> diff.prob <- max.pr - assign.pr
```

If `diff.probs` is zero, the bars are then colored with green. If `diff.probs` is less than the threshold, the bars are orange. Otherwise, the bars are red. In our case, the threshold is set at the mean value of the nonzero differences, approximately equal to 0.3:

```
> thres <- mean(diff.prob[diff.prob>0])
> cols <- cut(diff.prob, c(-1,0,thres,1), labels=c("1","2","3"))
> dtst <- data.frame(assign.pr, cols, cnt=1:nrow(dtset))
> ggplot(aes(x=cnt, y=assign.pr, fill=cols), data=dtst) +
+  geom_bar(aes(width=1), stat="identity", colour="black")+
+  scale_fill_manual(values=c("1"="green", "2"="orange", "3"="red")) +
+  scale_x_continuous(limits=c(0,nrow(dtset)),name="")+
+  scale_y_continuous(name="Probability assigned by model to match
+  outcome") + theme(legend.position="none")
```

14.7 Concluding Remarks

In this chapter, we describe a data mining process we developed to identify the key factors determining the outcome of a football match. We tested this process using a large dataset comprised of 481 variables with values recorded for each match of the Italian football championship league "Serie A" during the 2010-2011 season. The aim of this analysis was twofold: First, we demonstrated that a structured collection of data mining techniques can extract information from a complex dataset. Second, we showed that the entire data mining process can be performed using R software, thanks to the wide choice of available R packages.

Our analysis followed a five-step data mining process involving three intermediate tasks: (i) the selection of informative variables, (ii) the construction of composite indicators describing match play, and (iii) the extrapolation of the relationship between the composite indicators and match outcome.

The main results can be briefly summarized as follows:

i. Variable selection was carried out by RF using the Gini VIM with bias correction according to the innovative heuristic strategy proposed by Sandri and Zuccolotto (2008). In this step, a valuable development in the present work is the implementation of the bias-correction strategy

in R with parallel computing, which appreciably lightens the computational burden. Among the 481 variables, 13 were selected as particularly important in influencing match outcome.

ii. The 13 selected variables were then summarized, using PCA, in 6 composite indicators: three related to the home team and three related to the away team. Each indicator represented different elements of match play. The extent to which one indicator is particularly influential during a given match suggests the use of a specific game strategy by the corresponding team.

iii. To determine how composite indicators affect match outcome, five different classifiers were used: RF, NNET, KNN, Naïve Bayes, and MLogit regression. The performances of the five models/algorithms were evaluated "out-of-sample" using a random test set of 80 matches and a cross-validation approach. From a computational point of view, the most significant aspect of this step is the implementation of training and the comparison of the five classifiers via the `caret` R package together with parallel processing features. The model selection was done according to the dilemma of Occam's razor, pursuing the best balance between model fitting and simplicity. Thus, since the five classifiers exhibited similar overall performance indices, we gave preference to the MLogit classifier, which combines good performance with easier interpretation.

In terms of football strategy, this analysis highlights that the home team's ability to create opportunities to make a shot on the goal increases the probability of scoring and, ultimately, winning the match. On the other hand, we found that a strategy on the basis of aerial abilities is positively associated with the probability of a draw or defeat; therefore, teams with a high degree of aerial play are less likely to win. In addition, we determined that the two composite indicators related to defense play do not appear to play crucial roles in determining match outcome.

Acknowledgments

The authors are grateful to dott. Aldo Maccagni of Panini Digital for his interest in this work and for making his data freely available for the readers of this book. Thanks to prof. Eugenio Brentari for having purchased the dataset and to dott. Roberta Tengattini for the first exploratory analyses of data in her degree dissertation, tutored by prof. Paola Zuccolotto. Thanks to dott. Samantha Sartori and to dott. Elisa Zorzi for helpful discussions. Special thanks to dott. Amy Volpert for her careful editing and proofreading. Finally, the authors acknowledge the anonymous referees for interesting suggestions.

References

Albert, J., Koning, R.H. (Eds.), 2008. Statistical Thinking in Sports. Chapman & Hall, Boca Raton.

Albert, J., Bennet, J., Cochran, J.J. (Eds.), 2005. Anthology of Statistics in Sport, ASA-SIAM Series in Statistics and Probablity. SIAM, Philadelphia, ASA, Alexandria.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24, 123–140.

Breiman, L., 2001a. Random forests. Mach. Learn. 45 (1), 5–32.
Breiman, L., 2001b. Statistical modeling: the two cultures. Stat. Sci. 16 (3), 199–231.
Breiman, L., 2002. Manual on setting up, using, and understanding Random Forests v3.1. Technical report. http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf.
Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Chapman & Hall, New York.
Calle, M.L., Urrea, V., 2010. Letter to the Editor: Stability of Random Forest Importance Measures. Brief. Bioinform. 12 (1), 86–89.
Carroll, B., Palmer, P., Thorn, J., 1988. The Hidden Game of Football. Warner Books, New York.
Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, P., 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chawla, N.V., Japkowicz, N., Kotcz, A., 2004. Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. 6 (1), 1–6.
R Core Team, 2012. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.r-project.org/.
Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 29, 1189–1232.
Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Mach. Learn. 29, 131–161.
Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques, third ed. The Morgan Kaufmann Publishers, San Francisco.
Hand, D.J., Mannila, H., Smyth, P., 2001. Principles of Data Mining: Adaptive Computation and Machine Learning. MIT Press, Cambridge.
Hastie, T., Tibshirani, R., Friedman, J.H., 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.
He, H., Garcia, E., 2009. Learning from Imbalanced Data. IEEE Trans. Knowledge Data Eng. 21 (9), 1263–1284.
Hopkins, W.G., 2012. The impact-factor Olympics for journals in sport and exercise science and medicine. Sportscience. 16, 17–19.http://sportsci.org/2012/wghif.htm.
Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. 24 (417–441), 498–520.
Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: a conditional inference framework. J. Comput. Graph. Stat. 15 (3), 651–674.
Jolliffe, I.T., 2002. Principal Component Analysis. Springer Verlag, New York.
Khalilia, M., Chakraborty, S., Popescu, M., 2011. Predicting disease risks from highly imbalanced data using random forest. BMC Med. Inform. Decis. Mak. 11, 51.
Knaus, J., 2010. Snowfall: Easier cluster computing (based on snow), R package version 1.84. http://cran.r-project.org/package=snowfall.
Knaus, J., Porzelius, C., 2009. Tutorial: Parallel computing using R package snowfall, http://www.imbi.uni-freiburg.de/parallel/docs/Reisensburg2009_TutParallelComputing_Knaus_Porzelius.pdf.
Kuhn, M., 2008. Building Predictive Models in R Using the caret Package. J. Stat. Software. 28 (5), 1–26. http://www.jstatsoft.org/v28/i05/paper.
Kuhn, M., Contributions from Wing, J., Weston, S., Williams, A., Keefer, C., Engelhardt, A., 2012. caret: Classification and Regression Training. R package version 5.15-045, http://cran.r-project.org/package=caret.
Kuper, S., 2011. A football revolution (downloadable as the numbers game), Financial Times Magazine, June 17, 2011, http://gilesrevell.com/files/championsleague.pdf.
Liaw, A., Wiener, M., 2002. Classification and regression by random forest. R News. 2 (3), 18–22.
Min, B., Kim, J., Choe, C., Eom, H., McKay, R.I., 2008. A compound framework for sports results prediction: a football case study. Knowledge Based Syst. 21, 551–562.
Nicodemus, K.K., 2011. Letter to the Editor: on the stability and ranking of predictors from random forest variable importance measures. Brief. Bioinform. 12 (4), 369–373.
Pearl, J., 1986. Fusion, propagation and structuring in belief networks. Artif. Intell. 29, 241–288.

Pearson, K., 1901. On lines and planes of closest fit to systems of points in space. Philos. Mag. Series 6. 2 (11), 559–572.
Pollard, R., Reep, C., 1997. Measuring the effectiveness of playing strategies at soccer. Statistician. 46 (4), 541–550.
Qiao, X., Liu, Y., 2009. Adaptive weighted learning for unbalanced multicategory classification. Biometrics. 65 (1), 159–168.
Rue, H., Salvesen, O., 2000. Prediction and retrospective analysis of soccer matches in a league. Statistician. 49 (3), 399–418.
Sandri, M., Zuccolotto, P., 2008. A bias correction algorithm for the Gini variable importance measure in classification trees. J. Comput. Graph. Stat. 17 (3), 611–628.
Sandri, M., Zuccolotto, P., 2010. Analysis and correction of bias in total decrease in node impurity measures for tree-based algorithms. Stat. Comput. 20, 393–407.
Slaton, Z., 2012. A beautiful numbers game—statistically informed soccer writing. http://www.abeautifulnumbersgame.com.
Stern, H., 2005. Introduction to the football articles. In: Albert, J., Bennet, J., Cochran, J.J. (Eds.), Anthology of Statistics in Sport. ASA-SIAM Series in Statistics and Probability, Philadelphia, ASA, Alexandria, 13–16.
Strobl, C., Boulesteix, A.-L., Zeileis, A., Hothorn, T., 2007. Bias in random forest variable importance measures: illustrations, sources and a solution. BMC Bioinform. 8 (25).
Tierney, L., Rossini, A.J., Na, L., Sevcikova, H., 2012. Snow: Simple Network of Workstations, R package version 0.3-10, http://cran.r-project.org/package=snow.
Torgo, L., 2010. Data Mining with R, Learning with Case Studies. Chapman and Hall/CRC, Boca Raton, FL.
Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.
Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S, fourth ed. Springer, New York.
Weihs, C., Ligges, U., Luebke, K., Raabe, N., 2005. `klaR` analyzing German business cycles. In: Baier, D., Decker, R., Schmidt-Thieme, L. (Eds.), Data Analysis and Decision Support. Springer-Verlag, Berlin, pp. 335–343.
Weston, S., Calaway, R., 2012. Getting Started with `doParallel` and `foreach`, http://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf.
Wickham, H., 2009. ggplot2. Elegant Graphics for Data Analysis, second ed. Springer, New York.
Wolf, B.J., Hill, E.G., Slate, E.H., 2010. Logic forest: an ensemble classifier for discovering logical combinations of binary markers. Bioinformatics. 26 (17), 2183–2189.

CHAPTER 15

Analyzing Internet DNS(SEC) Traffic with R for Resolving Platform Optimization

Emmanuel Herbert[*,†], **Daniel Migault**[*], **Stephane Senecal**[*], **Stanislas Francfort**[*], **Maryline Laurent**[†]

*Orange Labs, Issy-les-Moulineaux, France

†CNRS Samovar UMR 5157, Institut Mines-Télécom/Télécom SudParis, Ivry, France

15.1 Introduction

Domain Name System (DNS) (Mockapetris, 1987a,b) is the computer protocol that facilitates Internet communication using hostnames by matching an Internet Protocol (IP) address and a Fully Qualified Domain Name (FQDN), e.g., "www.google.com." DNS servers, which host the IP addresses of the queried web sites—that is to say the DNS responses—are called *Authoritative Servers*. Because *Authoritative Servers* would not be able to support all end users' queries, the DNS architecture introduces *Resolving Servers* that cache the responses during Time to Live (TTL) seconds. Internet Service Providers (ISPs) manage such servers for their end users. Thanks to the caching mechanism, *Resolving Servers* do not need to ask *Authoritative Servers* if the response is still in their cache. This provides faster responses to the end user and reduces the traffic load on the DNS *Authoritative Servers*.

For multiple reasons, ISPs consider operating DNSSEC, the security extension of DNS defined in the standards (Arends et al., 2005a,b,c; Sawyer, 2005). With DNSSEC, a DNS response is signed so that its authenticity (generation by a legitimate Authoritative Server) and its integrity (nonmodification of response) can be checked. With DNSSEC, resolutions require multiple signature checks so that responses are around seven times longer than traditional DNS responses. Migault (2010), Migault et al. (2010), and Griffiths (2009) show that DNSSEC resolution platforms require up to five times more servers than DNS resolution platforms. Migault et al. (2010) measures that a DNSSEC resolution involves three signature checks and costs up to 4.25 times more than a regular DNS resolution. With the DNS traffic doubling every

Data Mining Applications with R

year and the deployment of its secure extension DNSSEC, DNS resolving platforms require more and more resources.

The operational problem faced is to reduce the resources needed by a resolving platform. The resolving platform consists of several DNS *resolving servers* behind a load balancer device. The load balancer splits the incoming traffic to distribute queries on resolving servers. The classical way of load balancing is performed by assigning a pool of clients to be served to each server.

One way to reduce the load on a server is to lower the number of resolutions. To reduce the number of resolutions, Migault and Laurent (2011) and Francfort et al. (2011) evaluate the advantage of splitting the DNS traffic according to the queried FQDN rather than according to the IP addresses. This increases the efficiency provided by caching mechanisms, reduces the number of signatures to be checked, and can result in a 1.32 times more efficient architecture.

To design this new load balancing mechanism, we first need to characterize the DNS traffic and to evaluate how the DNSSEC traffic looks like. We perform data extraction from raw network captures taken from a DNS resolving platform. The main challenge here is to define the variables, which are taken and computed for each FQDN. The goal is to define a routing table mapping each frequently requested FQDN to a server of the resolving platform.

15.2 Data Extraction from PCAP to CSV File

To conduct this study, we first gather pieces of DNS data. They consist of real outbound and inbound DNS traffic of the platform stored in PCAP files. Then, for each FQDN found in a traffic sample, we compute a series of variables. Given the application considered, these variables are related to the FQDN's resolution cost.

Network costs: servers occupation times associated to a FQDN (time between a query and its response) and different rates:

- mean open context times observed for resolvers: Mean Internet Resolution Time (*MIRT*) and Mean Platform Resolution Time (*MPRT*)
- end user and platform query rates (*euQR* and *reQR*)
- end user and platform bit rates (*euBR* and *reBR*)

Computation costs: signature checks related variables:

- number of signature checks (*SigCheck*)
- cache hit rate (*CHR*)

Memory costs: cache length and cache update related variables:

- mean TTL observed (*MTTL*)
- query and response length (*Qlen* and *Rlen*)
- response time for cached response

These variables are exported into a CSV file. CSV is a standard format that can be read from many softwares and languages, including *R*. To generate this CSV file, we use a homemade python script. This file is composed of lines terminated with the UNIX-compliant end of line character ("\n"), each line containing the variables corresponding to a FQDN into fields. The separator that separates fields is the classical space and fields are not enclosed between quotation marks. The first field is the FQDN, i.e., the label of the vector corresponding to the FQDN. Variables labels are not included in the CSV file to ease some common operations like split or concatenation of several CSV input files. The first arrow in Figure 15.1 represents this step.

Our dataset is now stored in a CSV file that consists of vectors corresponding to FQDN and composed of cost-related measures.

15.3 Data Importation from CSV File to R

Once the dataset is extracted from PCAP files to CSV files, we import these files into R. To do so, we use the code in the Listing 15.1. This step is represented by the last arrow in Figure 15.1. As described in Section 15.2, the CSV file does not contain dimension labels. In line 3 of Listing 15.1, we construct a vector containing all labels in the correct order. These labels will be used

Listing 15.1 R Code Used to Load Dataset from CSV File

```
filename = "inputfile.CSV" # input filename

clab <- c("euQR", "reQR", "tR", "euBR", "reBR", "tBR", "cQRT",
    "reOCC", "euOCC", "tOCC", "CHR", "eQR", "eCPU", "ePRT",
    "succRatio", "failRatio", "cltQNbr", "pltQNbr", "Qlen", "Rlen",
    "Sigcheck", "MIRT", "SDIRT", "MPRT", "SDPRT", "MTTL", "SDTTL")

mat_ent <- read.table (filename, row.names=1, col.names=clab)

mat_ent <- subset (mat_ent, cQRT > 0) # python script return -1 if
     no request is present and cQRT is used to plot several variables

mat <- subset (mat_ent, MTTL > 0) # remove non valid TTL
```

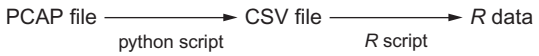


Figure 15.1
Extraction and importation from PCAP files to *R*.

later to ease dimension selection. The *R*-function used to import data is *read.table()*, in line 5. This function returns a matrix stored in a *data.frame* object whose dimensions are labeled thanks to the *row.names* and *col.names* arguments. We set the name of the input file at the beginning of our code (line 1) to ease the readability and the further modification of the input file name. Note that the way our CSV file is defined allows us to keep default values for most of the *read.table()* function's parameters. The last lines of Listing 15.1 (lines 7 and 8) are used to remove lines (i.e., FQDN), which present non acceptable values for the variables, from our data. This is a step to delete FQDN whose variables are not coherent or not in the expected intervals.

We now have a *data.frame* containing the input dataset for further *R* processing.

15.4 Dimension Reduction Via PCA

The dataset consists of several thousands of 27-dimensional vectors, each vector corresponding to a FQDN. For a better understanding, we aim at reducing this dataset volume by shrinking the number of its dimensions, i.e., the number of FQDN characteristics. To perform this dimension reduction, we use principal component analysis (PCA; cf. Cox and Cox, 2001), for instance. PCA is an efficient way to reduce the number of noninformative dimensions and to eliminate correlated variables. The PCA algorithm is implemented in *R* through the function *prcomp()*. The code used to perform PCA is presented in Listing 15.2.

Listing 15.2 R Code Used for PCA

```
r =0.9  # threshold for PCA
output_file = paste (format (Sys.time(), "%F-%T"), "-Rout.txt", sep=" ")
    # file where to print
tmp_file = "/tmp/foo" #tmp file
sink(output_file)
clabf <- c("euBR", "reBR", "QNbr", "pltQNbr", "CHR", "cQRT", "MIRT",
    "MPRT", "MTTL")

mat <- subset (mat, select=clabf)

pca <- prcomp(mat, scale=TRUE, center=TRUE)
mag <- sum(pca$sdev * pca$sdev)       # total magnitude
pca <- prcomp(mat, tol = (1 — r)*mag/(pca$sdev [1] * pca$sdev [1]),
    scale=TRUE, center=TRUE)
write.table(pca$x, file=tmp_file)
d<-read.table(tmp_file, header=TRUE, row.names=1)
write.table(pca$rotation, file=tmp_file)
rot<-read.table(tmp_file, header=TRUE, row.names=1)
print(pca$rotation)       # new vectors

sink()
```

To ease the exploitation of the PCA's results, we dump the output stream into a file which can be read thanks to any text editor. To do so, we first construct the name of this file. We want to keep and distinguish results from this script run at different times. The filename (line 2 of

Listing 15.2) contains the date when the script is run (*Sys.date()*) in a friendly format (*format()*). This timestamp is concatenated with another string (*"-Rout.txt"*) thanks to the function *paste*. We change the separator of this function to the empty string (*sep=" "*) to avoid space in the filename.

We also use a temporary file to write and read some data. This is a trick to reformat a *matrix* object into a *data.frame* object which can be avoided using *as.data.frame()*. Moreover *as.data.frame()* takes less time as it does not require hard disk access. To open an output flow, we use the function *sink()* (line 4 of Listing 15.2) with the filename constructed line 2. This redirects all the output into the file. Note that the file should be closed (line 18 of Listing 15.2). As a preparation step, we also store a subset of dimension labels in a vector (line 5) to be used later to select a subset of initial data thanks to the *subset* function (line 7). This subset concerns only nine variables and ignores others which are linear combinations of the first ones.

In the PCA, we define a threshold to decide which components are kept. We aim at keeping a percentage of the total magnitude. We used *prcomp()* the first time (line 9 of Listing 15.2) to get the whole magnitude, i.e., the whole variance, and to compute the variance kept. We recompute a PCA using this variance value (line 11). The result of this step is the rotation matrix and the vectors in the new basis. These results are stored in variables (line 13 of Listing 15.2) and printed into the output file (line 16) (Figure 15.2).

Thanks to the rotation matrix and the screegraph (plot of variance explained by each principal component represented in Figure 15.3), we can see that:

- the two first principal components hold a significative part of magnitude (35% in our application case)
- the two first principal components are mainly due to *euQR* and *reQR*

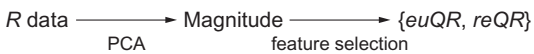


Figure 15.2
Reduction dimension via PCA.

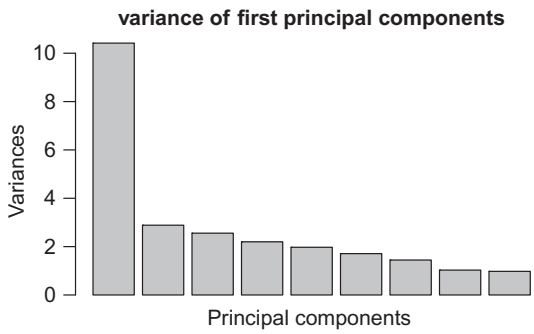


Figure 15.3
Screegraph of PCA on initial variables.

This result could have been obtained by analyzing the variance for each variable individually. *euQR* and *reQR* are the most discriminative variables from the viewpoint of second-order statistical information (variance).

15.5 Initial Data Exploration Via Graphs

To be more familiar with the data considered in this problem, also to learn how they look, and to define which process to apply, we perform an exploration phase. We conduct this initial exploration through graphs. All graphs performed with *R* are drawn into a postscript file to be edited with external tools if needed. There are multiple types of graphs depending on what we plot with *R* and what parameters we provide to the *plot()* function.

To draw points, we provide the list of coordinates (list of abscissas and list of ordinates) to the *plot()* function. For a *matrix* or a *data.frame*, the *plot()* function performs a *scatterplot*. This consists of a series of graphs, each being the representation of data in a two-dimensional space. All possible couples are represented. Such a graph can be seen in Figure 15.4.

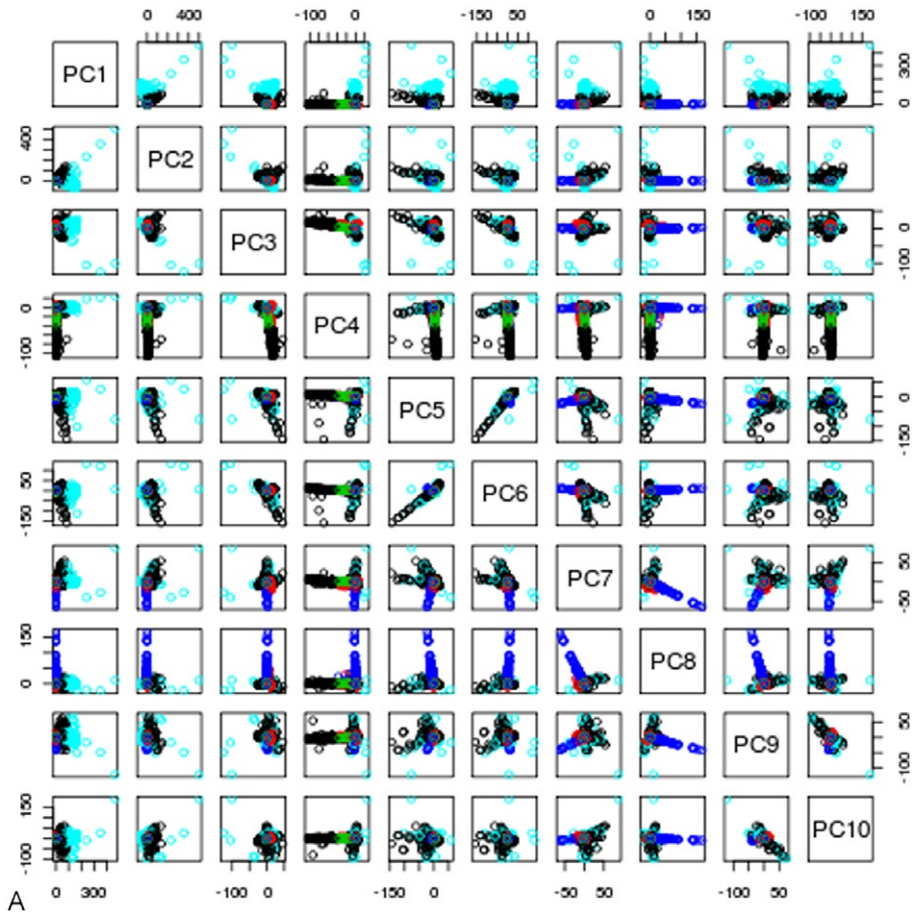


Figure 15.4
Multidimensional representation of FQDN. (a) Principal components.

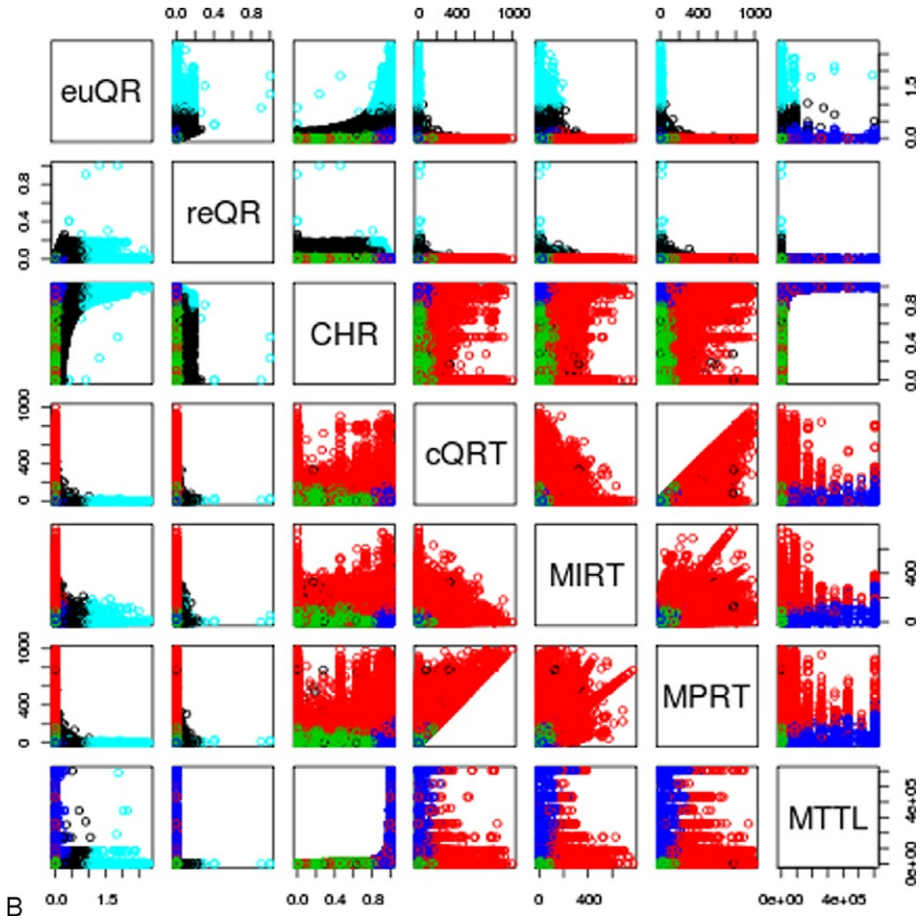


Figure 15.4—cont'd (b) Subset of initial parameters.

The function *density()* returns a kernel density estimate which is drawn with the *plot()* function. This graph is useful to know if the distribution is multimodal (see Figure 15.5).

Boxplots highlight median, quartiles, minimum, maximum, and outliers. When the input of the *boxplot()* function is a *data.frame*, it traces a boxplot for every dimension. This simple drawing allows us to see immediately if a dimension seems discriminant and highlights outliers and imbalances in the distribution. This representation also helps to visualize the differences between variables.

Thanks to these simple graphs, we can define further processes to apply to the dataset to get more balanced features.

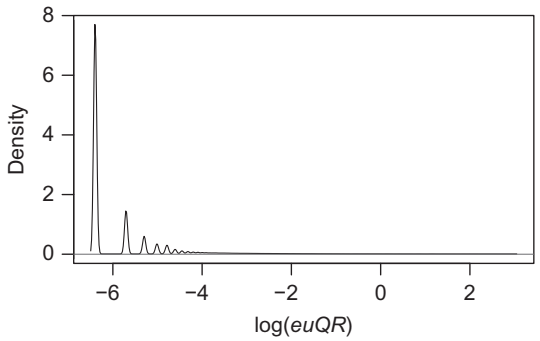


Figure 15.5
euQR density plot.

15.6 Variables Scaling and Samples Selection

As seen in Section 15.5, all the variables are not equivalently informative to discriminate the FQDN, see, for instance, Figure 15.4b. Moreover, the distribution of the queries and the responses rates highlighted in Section 15.5 suggests that these variables should be processed using a log function. Indeed, the range and the distribution of values for these variables do not give an informative representation. In this case, a standard linear representation is not very relevant. Instead, we choose to apply a logarithmic transformation to grasp more precisely the value amplitudes for the variables of interest. We also decide to remove the less requested FQDN (*euQR* less than a threshold) because many FQDN are requested only a couple of times during the timeslot used for the traffic capture. The code used to perform this processing is presented in Listing 15.4. We add to the original data (stored in *mat*) three variables (cf. lines 1-3 of Listing 15.4).

Listing 15.3 Generation of Several Graphs into Postscript Files

```
postscript ("pca_magnitude.ps")
plot(pca)
dev.off ()

postscript ("boxplot.ps")
boxplot (mat)
dev.off()

postscript ("scatter.ps")
plot(mat)
dev.off()

postscript ("euQR_density.ps")
plot (density (log(mat$euQR)), xlab=" log (euQR), main=" ")
dev.off()
```

Listing 15.4 **Application of log() on Initial Dataset, Sample Selection, and Feature Selection**

```
mat$logeuQR <- log(mat$euQR)
mat$logreQR <- log(mat$reQR)
mat$logSigcheck <- log(mat$Sigcheck)

mat <- subset(mat, euQR > threshold)
m_mat <- subset(mat, select=c("reQR", "euQR"))
```

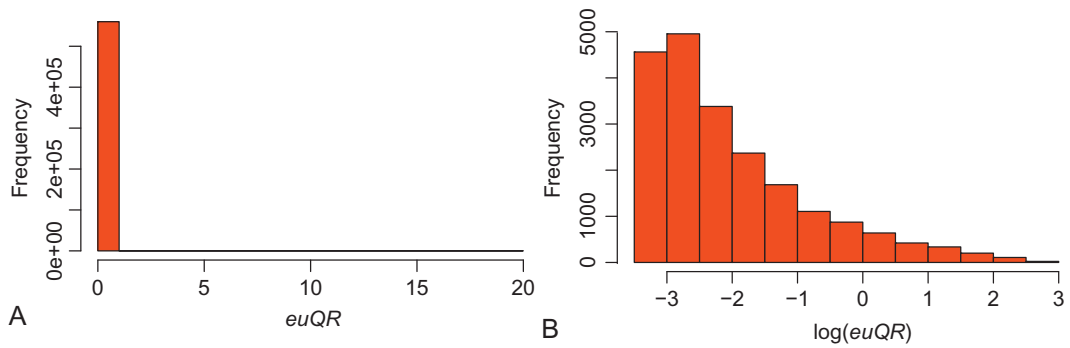


Figure 15.6
Preprocessing the variable *euQR*. (a) *euQR* (q s^{-1}) without any transformation. (b) log(*euQR*) without less requested FQDN.

Once PCA has been applied, we select the most informative variables for the problem considered. *euQR* is the variable with the greatest variance. The operation consisting in applying the log() function and removing the less requested FQDN (lines 5 and 6 of Listing 15.4) can be considered as preprocessing. To visualize the effects of this preprocessing, we use histograms (Figure 15.6). As the kernel smoother used by *density()*, histogram is a density estimator and allows us to visualize the distribution (Figure 15.7).

15.7 Clustering for Segmenting the FQDN

The goal pursued is to separate FQDN into different groups depending on their costs. The initial idea we investigate is to define for each FQDN a set of cost-related variables (Section 15.6) and to cluster FQDN using an unsupervised machine learning technique. We aim at clustering the data in groups of FQDN having the same cost origins (e.g., frequently requested, long response, low TTL).

We use simple clustering algorithms: K-means and K-medoïds, for instance (cf. Hastie et al., 2008; Kogan, 2007). The K-means is a clustering algorithm grouping similar pieces of data together. A group is characterized by its *centroïd* which is a vector minimizing the distances to all other elements of the group. The K-medoïds algorithm uses *medoïds* instead of centroïds. The difference between centroïds and medoïds is that medoïds are necessarily

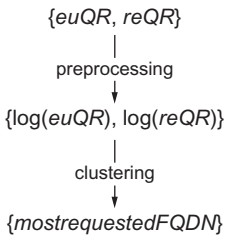


Figure 15.7
Preprocessing and clustering after feature selection.

points belonging to the initial dataset. This characteristic prevents us from finding objects in the original space that are not in our dataset. It also enables the precomputation of all the samples interdistances (i.e., the use of a dissimilarity matrix).

The K-means algorithm is implemented in *R* through the function *kmeans()*. This function is part of the *stats* package (R Development Core Team, 2010). This function returns a *kmeans* object consisting of clusters and some cluster characteristics. Also, the K-medoïds algorithm is implemented through the *pam()* function. *pam* stands for *Partition Around Medoïds*. This function is provided by the package *cluster* (Maechler et al., 2005) loaded in line 1 in Listing 15.5. The *pam()* function returns a *cluster* object.

Listing 15.5 Clustering and Silhouette Visualization

```
library("cluster)

# silhouette width for pam and kmeans
swlqr <- numeric(25)
kswlqr <- numeric(25)

sink(file=output_file, split=TRUE)

for (k in c(2:3)) {
    # kmean log qr
   km <- kmeans(clqrmat, center=k, iter.max=1000)
   kswlqr[k] <- summary(silhouette (km$cluster, daisy(clqrmat))) $avg.width
   png(paste("k0", k, "qr_log_kmean.png", sep=" "))
   par(cex=2); plot(qrmat, col=km$cluster * 5, log="xy", pch=km$ cluster)
   dev.off()
   print(paste(" - log qr kmean - k =", k))
   mysummarykmean(km)

   # kmed log qr
   km <- pam(clqrmat, k=k)
   swlqr[k] <- km $ silinfo $ avg.width
   png(paste("k0", k, "qr_log_kmed.png", sep=" "))
   par(cex=2); plot(qrmat, col=km$ clustering * 5, log="xy", pch=km$ cluster)
   dev.off()
   print(paste(" - log qr kmed - k =", k))
   print(km$clusinfo)
}
```

As suggested in Section 15.6, we cluster the data using a subset of initial variables (*euQR* and *reQR*) and another subset of preprocessed variables (log(*euQR*) and log(*reQR*)). In practice, the K-means and K-medoïds algorithms applied to the dataset exhibits a convergence in less than 15 iterations. As a result, the data are well segmented into groups corresponding to the FQDN which are either rarely requested or frequently requested among the traffic (cf. Xu et al., 2011). We also observe that the K-means and the K-medoïds schemes converge to similar clustering results. It can be explained by the samples distribution and shape of the data, in which the centroïds are located quickly quite close to the medoïds data points.

To determine the relevant number of clusters, we used the *silhouette* as defined in Rousseeuw (1987). The silhouette is defined for each sample and takes values between -1 and 1.

- it is close to 1 when the sample is near the center of the cluster it belongs to.
- it is almost null if the sample is located near the frontier between its cluster and the nearest cluster.
- it is negative if the sample is in a cluster it should not belong to.

For each $FQDN_i$ in cluster C_i, we measure a_i the average distance between $FQDN_i$ and other FQDN of C_i. a_i measures the average dissimilarity of $FQDN_i$ with C_i. Then, we measure b_i the minimum average distance between $FQDN_i$ and other FQDN in clusters $(C_j)_{j\neq i}$. b_i measures similarity with other clusters. The silhouette for $FQDN_i$ is given by:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

By construction of the K-means and the K-medoïds algorithms, the silhouette cannot be negative. We run the clustering algorithms for several numbers of clusters (*for* loop from lines 9 to 27 in Listing 15.5). At each iteration, we compute the average silhouette and store the result in a vector (created lines 4 and 5). For human readability, we draw the average silhouette thanks to the code presented in Listing 15.6. To monitor the evolution of the *for* loop, we

Listing 15.6 R Code Used to Plot Silhouette

```
# plot barplot of sil value for k in c(2:15) for

aabb <- mat.or.vec(2,15)
aabb[1, 1:15] <- kswlqr[1:15]
aabb[2, 1:15] <- swlqr[1:15]
par(cex=2)
barplot(aabb[,2:15], beside=TRUE, col=c("dark blue", "pink"), names.arg=c(2:15),
    xlab="cluster number", ylab="average silhouette width", legend=c("kmean",
    "kmedoid"))
```

decide to split the output flow. The argument of the *sink()* function (line 7) makes two identical copies of the output flow:

- one flow for the standard output to monitor the evolution of the *R* script.
- one flow written in the file whose name is stored in the variable *output_file*.

As the objects returned by *kmeans()* and *pam()* are not the same, the silhouette is not computed the same way. For the *cluster* object returned by the *pam()* function, we immediately access the silhouette information (line 21). For the *kmeans* object returned by *kmeans()*, we compute the silhouette thanks to the *silhouette()* function included in the *R* package *cluster* (Maechler et al., 2005). We provide to the *silhouette* function clusters as returned by *kmeans()* and dissimilarity between samples computed by *daisy()*. *daisy()* is also part of the *cluster* package (Maechler et al., 2005). We use the *summary()* function because the *summary.silhouette* object returned is easier to manipulate than the *silhouette* object returned by the *silhouette()* function. This is illustrated in line 12 of Listing 15.5.

We now handle silhouette values for multiple numbers of clusters (*k* values) and for the two clustering algorithms (K-means and K-medoïds). To visually compare the results, we use *barplot()*. The code used is written in Listing 15.6.

First, we cast all data into a two-dimensional array. This array is declared and filled (lines 3-5 from Listing 15.6) with data from Listing 15.5. To enhance readability, we increase label size thanks to the *par()* function (line 6). This function controls layout parameters for graphs. The *cex* parameter controls the size of text and symbols. The colors used (*dark blue* and *pink*) are chosen to be quite different if the graph is printed in black and white. The results are presented in Figure 15.8. They show that the highest silhouette values for both clustering algorithms are obtained for 5, 3, and 2 clusters. This gives reliable estimates of the number of clusters fixed *a priori* to run the clustering algorithms. We perform an analysis of the DNS traffic through feature selection and clustering in Section 15.4 and above. Now, we devote the three following sections to the construction of a routing table for the identified heavily requested FQDN.

15.8 Building Routing Table Thanks to Clustering

As explained in Section 15.1, our goal is to build a routing table for the most requested FQDN to balance the load of the incoming DNS traffic in our resolution platform. The routing table is a function mapping a FQDN to a server of the platform, the platform being a set of resolution servers. This mapping is composed of explicit entries mapping FQDN to servers. For FQDN, which are not frequently requested, we compute the mapping *on the fly* based on a hashing function. We focus on the most requested FQDN for building the explicit mapping. Our first idea is that clustering outputs homogeneous groups of FQDN, each one having a

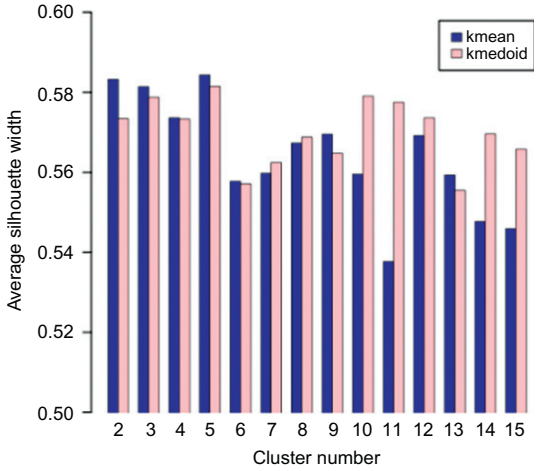


Figure 15.8
Average *silhouette* versus number of clusters.

different main source of cost. To balance the resources used between the servers of the platform, the idea is to distribute each group of FQDN homogeneously between servers. Doing this should dispatch the consumption of each kind of resource (network resources, memory, CPU, etc.) equally on each server. Unfortunately, as shown in Section 15.4, two variables (*euQR* and *reQR*) are more discriminative than the other because of their variance. To build the routing table, we proceed cluster by cluster. For each cluster, we distribute FQDN in a round-robin fashion. The algorithm is detailed in algorithm 1.

This algorithm outputs a routing table for the frequently requested FQDN, which maps the FQDN to different resolving servers. This table is not used directly after its generation but will be considered in Section 15.11 to be compared with routing tables built thanks to other approaches.

Algorithm 1 Building routing table based on clustering

Require: `cluster_number` // number of clusters for the clustering algorithms
Require: `server_number` // number of servers in our platform
`server` ← 1 // used to indicate to which server current FQDN will be mapped
for `k` = 0 to `cluster_number` **do**
  for `fqdn` ∈ `k` // enumerate FQDN belonging to cluster *k* **do**
    maps FQDN *fqdn* to server *server* mod `server_number` // add an explicit entry for the mapping
    `server` ← `server` + 1
  end for
end for

15.9 Building Routing Table Thanks to Mixed Integer Linear Programming

Another approach to building an efficient mapping between the FQDN and the servers of the resolving platform is to use linear programming. This idea is driven by the fact that we face an optimization problem. We used *GLPK* (Theussl and Hornik, 2010) to solve this problem. Although *GLPK* can be used as an *R* package, we use it as a standalone program.

Operational teams evaluate the efficiency of different load balancing techniques by comparing the CPU load of each server. However, providing an estimation of the CPU load for a server relies on experimental measurements, and as (Migault et al., 2010) mentioned, measured values for the CPU load depend on the hardware, the DNS server implementation, the nature of the traffic, etc. Since we do not want to depend on these factors, we evaluate the difference by considering the number of queries and resolutions performed by each server of the platform. Such evaluation requires defining specific notations we will use in the later in this chapter. Furthermore, these notations are also used to build a routing table with a mixed integer linear programming (MILP) method (cf. Schrijver, 1998) for instance:

I: Set of FQDN requested by the end users

J: Set of servers composing the DNS *Resolving Platform*

q_i, $i \in I$: Queries number associated with FQDN i

r_i, $i \in I$: Number of resolutions associated with FQDN i

$X = x_{i,j}$, $(i, j) \in I \times J$: Matrix binding FQDN i to server j

Q_j, $j \in J$: Number of queries supported by server j

R_j, $j \in J$: Number of resolutions supported by server j

We have immediately:

$$\forall i \in I, \forall j \in J, x_{i,j} = \begin{cases} 1 & \text{if FQDN } i \text{ is hosted by server } j \\ 0 & \text{otherwise} \end{cases}$$

$$\forall j \in J, Q_j = \sum_{i \in I} x_{i,j} . q_i$$

$$R_j = \sum_{i \in I} x_{i,j} . r_i$$

$$\forall i \in I, \sum_{j \in J} x_{i,j} \geq 1 \text{ (each FQDN is hosted by (at least) one server)}$$

We use this method to build a routing table for the most requested FQDN *milp-200* as we consider 200 FQDN. This number is the result of an operational evaluation. It is a compromise between the minimization of the computation time and the minimization of the number of

FQDN that are not balanced thanks to the routing table. For each FQDN, the number of resolutions is computed thanks to the number of queries and the mean *TTL* value observed for the FQDN. Because we consider the popular FQDN, we assume that a resolution occurs every TTL seconds.

Although this method makes it possible to build a routing table thanks to a technically sound scientific approach, in practice it happens to be heavy to implement because of the computational burden which limits its applicability. Also, this method needs to evaluate *a priori* the number of FQDN to be processed, which relies only on an empirical estimation.

The MILP method is based on solving a system of equations. We define a given set I of FQDN (line 1 of Listing 15.7). For a given distribution of these FQDN on the servers $(x_{i,j})_{(i,j)\in I\times J}$, we compute the number of queries and resolutions supported by each server $(Q_j, R_j)_{j\in J}$. The distribution we seek minimizes the differences between the servers of the platform in terms of $(Q_j, R_j)_{j\in J}$: ΔQ and ΔR.

Listing 15.7 **Mixed Integer Linear Program Used to Build a Routing Table**

```
set I;    /* set of fqdn */
set J;    /* set of servers */
param k;  /* k parameter */

param c{i in I, j in 1..2};    /* costs[r, q] */

var S{j in J};  /* sum of requests */
var T{j in J};  /* sum of resolutions */

var x{i in I, j in J} binary; /* 1 if Fi affected to Sj */
var deltar;
var deltaq;
var max;

minimize cost : max;/* objectives */

s.t. slack{(j1, j2) in (J cross J)} : k * S[j1] + (1 - k)*10000* T[j2] <= max;
s.t. aff{i in I} : sum{j in J} (x[i, j]) >= 1; /* one fqdn affected to at least 1 server */
s.t. q{j in J} : sum{i in I} (x[i, j] * c[i, 1]) = S[j] ;
s.t. r{j in J} : sum{i in I} (x[i, j] * c[i, 2]) = T[j] ;

solve;
end;
```

Trying all possible combinations for such a distribution is not feasible, so we formulate our problem as a MILP and use a solver (GLPK in this case, Theussl and Hornik, 2010) to find a proper distribution. Although an *R* application programming interface for GLPK exists (Theussl and Hornik, 2010), we decide not to use it. Writing the problem using the *GNU Mathematical Programming Language*, the native language for GLPK, is easier once the problem is modeled.

The challenge for the solver is to find a distribution that is close to the optimal distribution, even though we do not know the optimal solution. Considering 200 FQDN is a compromise between the total number of FQDN to process and the resources needed for the computation as shown in (Francfort et al., 2011). The integer linear optimization problem with two objectives consists in minimizing ΔQ and ΔR defined as follows:

$$\begin{aligned} \Delta Q &= \max_{j_1 \in J} Q_{j_1} - \min_{j_2 \in J} Q_{j_2} \\ \Delta R &= \max_{j_1 \in J} R_{j_1} - \min_{j_2 \in J} R_{j_2} \end{aligned}$$

To ease the resolution by the solver, we reduce the number of objectives by defining a FQDN cost:

$$c_i = \lambda q_i + (1-\lambda) r_i \forall_i \in I \quad \text{where } \lambda \in [0,1] \tag{15.1}$$

λ is a weighting parameter which determines the parts of q and r in the definition of the cost. In that sense, the important parameter is not λ itself but the ratio $\frac{\lambda}{1-\lambda}$ $\left(or\ \frac{1-\lambda}{\lambda}\right)$. Note that λ is introduced here only for resolving purpose, and has *a priori* no physical meaning. The problem is then rewritten as:

$$\begin{aligned} &\text{minimize}: \max_{j \in J} C_j \\ &\text{with}: \forall j \in J, C_j = \sum_{i \in I} x_{i,j} c_i \end{aligned}$$

C_j represents the cost supported by the server j. This objective function ensures the minimization of the cost supported by each server, which leads to the minimization of the difference of costs between the servers of the platform. The cost that is not supported by the most loaded server is reported on other servers, increasing the cost supported by the less loaded server, which thus reduces the difference of costs supported by the servers.

We run a solver for different values of λ and for each value we consider ΔQ and ΔR. Figure 15.9 represents the solutions (ΔQ, ΔR) obtained for specific values of λ, although λ values themselves are not represented. Some values of λ do not provide an optimal solution whereas others do. In the Figure 15.9, these optimal solutions for λ are pointed by the line called *Pareto front*. Among the values on the front, there is no mathematical way to decide whether one is better than the other. Choosing a solution is based on choosing whether we prefer to minimize ΔQ or ΔR. Thus, the decision should consider other aspects such as operational criteria. Also, we should mention that λ values on the *Pareto front* are difficult to characterize. For instance, if λ_0 corresponds to a point located on the *Pareto front*, then $(\lambda_0 + \varepsilon)$ does not necessarily provide a value close to the front. This value can be located anywhere on the (ΔQ, ΔR) plane. This reflects the nonconvex aspect of our problem because we use a discrete space for $(x_{i,j})_{i \in I, j \in J}$. We use the solver *GLPK* (Theussl and Hornik, 2010) running for 1000 s to

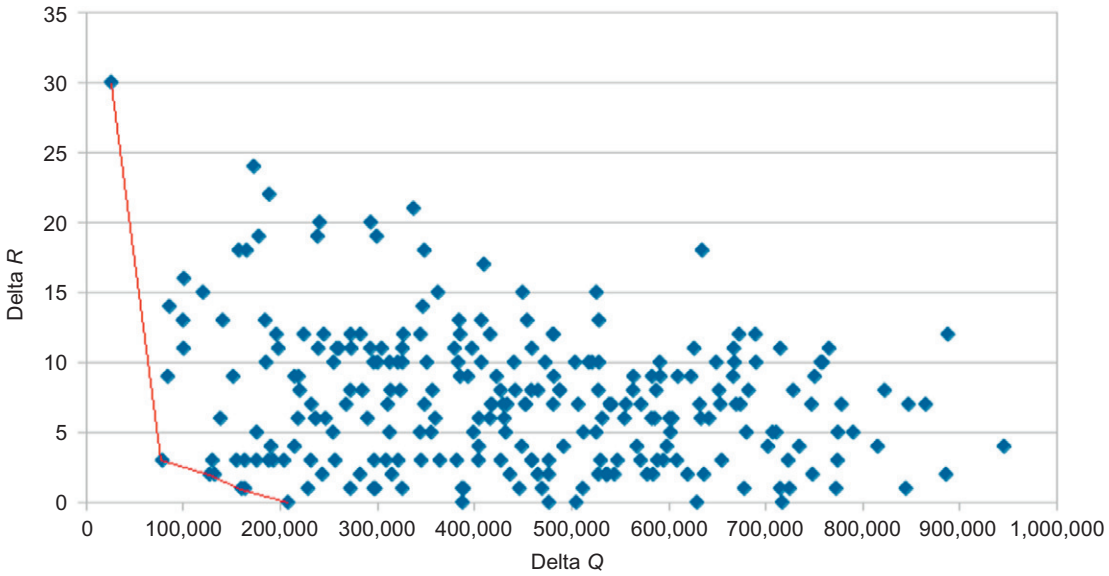


Figure 15.9
Bi-criteria MILP results.

solve this problem. Running *GLPK* longer does not necessarily give better results for our 200 FQDN set. The number of FQDN chosen is limited by resources used by the solver. We note this algorithm *milp-200*, and Figure 15.9 shows that ΔR as well as ΔQ can be very small.

15.10 Building Routing Table Via a Heuristic

The method described in Section 15.9 gives us promising results, but can only consider a limited number of FQDN. We thus evaluate another approach based on a heuristic.

The goal of this algorithm is similar to *milp-200*: minimizing jointly ΔQ and ΔR. However, the way we build the routing table provides less accurate results as *milp-200*. As a result, we need to consider a much larger set—namely, 18 times larger—of FQDN to build a routing table which balances properly the load among the servers. Even though the routing table is roughly 18 times larger, it takes less than 0.5 s to build it. Compared to 1000 s with *milp-200*, this method may present an operational advantage over *milp-200*. The algorithm starts with I, the set of the most requested FQDN. From the current set of FQDN, it takes the costliest FQDN, assigns it to the less charged server (i.e., a server $j_{\min}$ verifying $C_{j_{\min}} = \min_{j \in J} C_j$) and removes it from the FQDN set. This step is performed until the set of FQDN is empty.

To compare this method with *milp-200*, we compute ΔQ and ΔR for different values of λ. We first choose a set of 200 FQDN to be compared with the results from Section 15.9. Then,

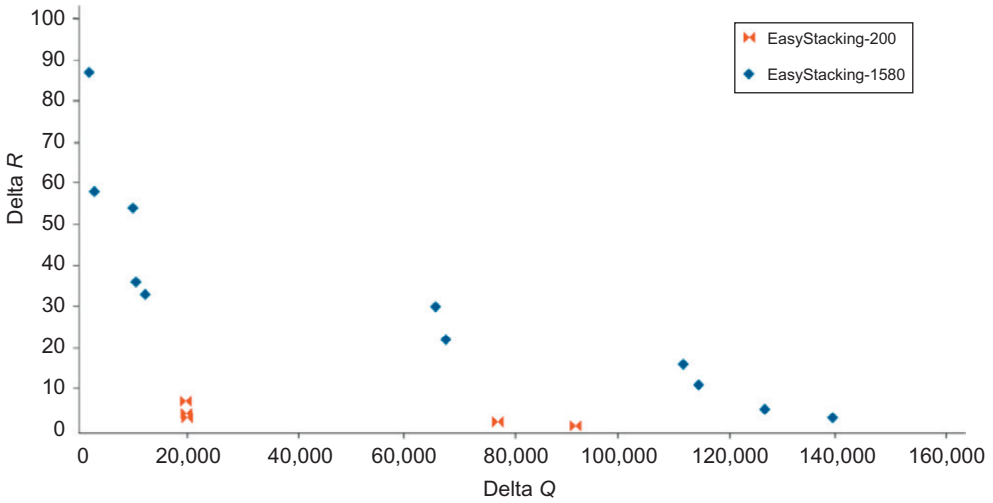


Figure 15.10
Pareto front with easy stacking.

by construction, an upper bound of the difference between the cost on different servers is $\min_{i \in I} c_i$, c_i being the cost of the less costly FQDN. Thus, we choose I, the set of FQDN, such that the last element is associated with a cost that is roughly the difference of costs generated by *milp-200*. This leads to consider 1580 FQDN. We denote these algorithms as *stacking-200* and *stacking-1580*. Note that 200 FQDN represent 16% of the number of queries and that 1580 FQDN represent 46% of the number of queries. Figure 15.10 shows that *stacking-200* presents a lower front compared to *stacking-1580*. However, ΔQ and ΔR are computed according to the set of FQDN I. This set is definitely not the same in *stacking-200* and in *stacking-1580*, which makes the comparison between *stacking-200* and *stacking-1580* difficult according to Figure 15.10.

One must keep in mind that this comparison takes into account the FQDN used for building the routing table. This is motivated by the fact that we evaluate the routing table and not the imbalance owing to the less requested FQDN. At this point, we can deal with the most requested FQDN and see what happens when adding the less requested FQDN.

15.11 Final Evaluation

Once routing tables are built (cf. methods detailed in previous Sections 15.8–15.10), we evaluate them. A routing table takes into account only the most requested FQDN. To validate the previously built routing tables and to decide which one is the best, we perform simulations. A simulation consists in replaying the traffic on a simulator. The traffic replayed is a 10-min slot received on one of our resolving platforms at a rush hour. This allows us to perform

evaluation including FQDN which are not balanced thanks to the previously built routing table. The simulator is a program implementing the load balancing task and some basic functions of the DNS servers to reproduce the behavior of a DNS resolving platform. The functions implemented are the only ones needed to evaluate performance. The indicators computed by the simulator are for each DNS server.

Network related indicators

- Query rate
- Response rate

DNS-Related Indicators

- Number of signatures to be checked if DNSSEC is used
- Cache management
  - *CHR*
  - Cache length
- Number of resolutions to perform on the Internet

The result of the simulation consists of an array containing these indicators for each server of the platform. The next step is to analyze the various indicators computed from simulations. To do so, we seek for a comprehensive representation of these indicators.

We define efficient load balancing as a load balance minimizing the resources used by the whole platform and minimizing jointly the differences of resources used for each server.

To visualize the repartition of the resources over the platform and to compare the different routing tables, we use the graphical function *boxplot*. Handling a boxplot allows us to see immediately the median, the quartiles, and the minimum and maximum in term of resources needed by servers. Note that the median is a more interesting indicator than the mean as every FQDN-based load balancing generates exactly the same number of requests and resolutions for the whole platform. Boxplots are directly drawn thanks to the *boxplot()* function.

Francfort et al. (2011) show that the computational cost can be modeled as a linear combination of the query rate and the resolution rate. To compare the different routing tables, we compute the repartition of CPU load. For each routing table, we evaluate the total CPU load for each server ($cpu\_load = query\_cost.nb\_query + resulution\_cost.nb\_resolution$) and then draw boxplots. The simulator's output consists of a two-dimensional array containing the total number of queries and the total number of resolutions for each server of our platform. Once this is done for a routing table, we store the result (i.e., the total resources needed for each server) in a column of a new two-dimensional array, the column being labeled with the name of the algorithm used to create the routing table. Then, all the routing tables being tested, we simply use the *boxplot()* function. The results are shown in Figure 15.11. In this figure, we denote by *XOR* the IP-based routing table as the load balancing is based on a *exclusive or* operation between source and destination IP addresses, and by *SHA1* the load balancing technique based

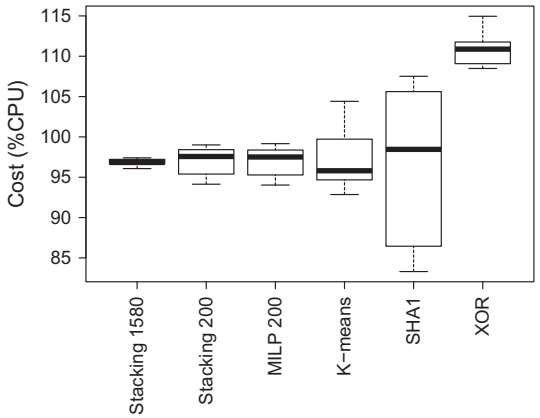


Figure 15.11
Repartition of costs.

on the application of the SHA1 hash function National Institute of Science and Technology (1993) on the FQDN. In Figure 15.11, we should keep in mind that our goal is to balance the resources needed, i.e., to minimize the differences between servers. The CPU load shown in this figure is not representative of the load on a real platform as the servers used to evaluate query and resolution costs are not representative of production servers. However, the values themselves are not so important but the ratio between them matters. We see that:

- *Stacking 1580* is the best algorithm as it minimizes the differences between resolution servers and results in a maximum of 2% CPU load difference between servers.
- All algorithms considered on the basis of the queried FQDN (all but XOR) are better than the algorithm based on IP addresses (XOR) in minimizing the resources needed.
- FQDN-based algorithms (all but XOR) outperform the IP addresses-based algorithm (XOR) by needing between 1.14 (DNS) and 1.32 (DNSSEC) times less servers.
- DNSSEC increases differences between the IP addresses-based and FQDN-based algorithms.
- The routing table length is a parameter influencing the resources repartition but the algorithm used to generate the routing table is also important. For example, *K-means* uses a bigger routing table than the one used by *milp-200* but does not perform better in balancing traffic.

15.12 Conclusion

R is an attractive tool to explore data and to design scripts on the basis of statistical methods. It is also efficient to visualize data. Same as *python*, it is useful for fast prototyping. The interactive mode allows us to find and test different options for the different built-in functions to be

included in scripts. Script mode is useful for process automation. As in *python*, one can take advantage of the object oriented possibilities offered by this language to ease scripts design.

R is complete with its extensions provided thanks to a variety of officially supported packages which ease its use. For common statistical-oriented usages, functions already exist. Moreover, the documentation is complete and gives us references to the algorithms the functions implement. One of the advantages of *R* is the possibility of making graphs with almost every *R* object. This is useful to visualize the effects of the processing performed on the data.

To our knowledge, FQDN-based load balancing techniques and the methods used to build the related routing tables are novel approaches to address the problem of Internet resolving platforms optimization. In the application case of data mining methods implemented in *R* described in this chapter, it was demonstrated that FQDN-based load balancing is efficient for improving the *CHR* and for reducing the resources needed to process DNS(SEC) traffic on a resolving platform. We can take advantage of the most popular FQDN distribution to improve this load balancing. The few most requested FQDN can be easily processed according to a small routing table, whereas handling the other FQDN is performed dynamically. A compromise must be found for the appropriate size of the routing table: the bigger it is, the better is the *CHR* and the balance between servers, but the slower is its generation. Further works on the platform optimization problem described in this chapter include a more efficient processing of the rarely requested FQDN and the study of robustness for the proposed load balancing techniques.

References

Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., 2005a. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard). Updated by RFC 6014.

Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., 2005b. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard). Updated by RFCs 4470, 6014.

Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., 2005c. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard). Updated by RFCs 4470, 6014.

Cox, T.F., Cox, M.A.A., 2001. Multidimensional Scaling. Chapman and Hall, Boca Raton, FL.

Development Core Team, R., 2010. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Francfort, S., Migault, D., Senecal, S., 2011. A bi-objective Mixed Integer Linear Program for load balancing DNS (SEC) requests. In: Proceedings of DNS EASY 2011, extended version in International Journal of Critical Infrastructure Protection, Elsevier, 2012.

Griffiths, C., 2009. Comcast DNSSEC Trail Test Bed. North American Network Operator Group (NANOG45).

Hastie, T., Tibshirani, R., Friedman, J., 2008. The Elements of Statistical Learning: Data Mining, Inference and Prediction. In: second ed. Springer.

Kogan, J., 2007. Introduction to Clustering Large and High-Dimensional Data. Cambridge University Press, New York.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., 2005. Cluster analysis basics and extensions. Rousseeuw et al provided the S original which has been ported to R by Kurt Hornik and has since been enhanced by Martin Maechler: speed improvements, silhouette() functionality, bug fixes, etc. See the 'Changelog' file (in the package source).

Migault, D., 2010. Performance measurements on bind9/nsd/unbound. In *IETF79*. IEPG.
Migault, D., Laurent, M., 2011. How DNSSEC resolution platforms benefit from load balancing traffic according to fully qualified domain name. In: Proceedings of CSNA.
Migault, D., Girard, C., Laurent, M., 2010. A performance view on DNSSEC migration. In: Proceedings of CNSM 2010.
Mockapetris, P., 1987a. Domain names—concepts and facilities. RFC 1034 (Standard). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.
Mockapetris, P., 1987b. Domain names - implementation and specification. RFC 1035 (Standard). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966.
National Institute of Science and Technology, 1993. Secure hash standard. Technical report, Federal Informational Processing Standard (FIPS), USA.
Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Comput. Appl. Math. 20, 53–65.
Sawyer, W., 2005. Management Information Base for Data Over Cable Service Interface Specification (DOCSIS) Cable Modem Termination Systems for Subscriber Management. RFC 4036 (Proposed Standard).
Schrijver, A., 1998. Theory of linear and integer programming. John Wiley & Sons Inc., New York.
Theussl, S., Hornik, K., 2010. *Rglpk: R/GNU Linear Programming Kit Interface*. R package version 0.3-5.
Xu, Q., Migault, D., Senecal, S., Francfort, S., 2011. K-means and adaptive k-means algorithms for clustering DNS traffic. In: Proceedings of the 5th International ACM/IEEE Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2011.

Index

Note: Page numbers followed by *f* indicate figures and *t* indicate tables.

C

D

L

M

R

S

T