EMC²

Data Science and Big Data Analytics

Discovering, Analyzing, Visualizing and Presenting Data

EMC Education Services

Data Science & Big Data Analytics

Data Science & Big Data Analytics

Discovering, Analyzing, Visualizing and Presenting Data

EMC Education Services

WILEY

Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-118-87613-8
ISBN: 978-1-118-87622-0 (ebk)
ISBN: 978-1-118-87605-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2014946681

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Credits

Executive Editor
Carol Long
Project Editor
Kelly Talbot
Production Manager
Kathleen Wisor
Copy Editor
Karen Gill
Manager of Content Development and Assembly
Mary Beth Wakefield
Marketing Director
David Mayhew
Marketing Manager
Carrie Sherrill
Professional Technology and Strategy Director
Barry Pruett
Business Manager
Amy Knies
Associate Publisher
Jim Minatel
Project Coordinator, Cover
Patrick Redmond
Proofreader
Nancy Carrasco
Indexer
Johnna VanHoose Dinse
Cover Designer
Mallesh Gurram

About the Key Contributors

David Dietrich heads the data science education team within EMC Education Services, where he leads the curriculum, strategy and course development related to Big Data Analytics and Data Science. He co-authored the first course in EMC's Data Science curriculum, two additional EMC courses focused on teaching leaders and executives about Big Data and data science, and is a contributing author and editor of this book. He has filed 14 patents in the areas of data science, data privacy, and cloud computing.

David has been an advisor to several universities looking to develop academic programs related to data analytics, and has been a frequent speaker at conferences and industry events. He also has been a a guest lecturer at universities in the Boston area. His work has been featured in major publications including Forbes, Harvard Business Review, and the 2014 Massachusetts Big Data Report, commissioned by Governor Deval Patrick.

Involved with analytics and technology for nearly 20 years, David has worked with many Fortune 500 companies over his career, holding multiple roles involving analytics, including managing analytics and operations teams, delivering analytic consulting engagements, managing a line of analytical software products for regulating the US banking industry, and developing Software-as-a-Service and BI-as-a-Service offerings. Additionally, David collaborated with the U.S. Federal Reserve in developing predictive models for monitoring mortgage portfolios.

Barry Heller is an advisory technical education consultant at EMC Education Services. Barry is a course developer and curriculum advisor in the emerging technology areas of Big Data and data science. Prior to his current role, Barry was a consultant research scientist leading numerous analytical initiatives within EMC's Total Customer Experience organization. Early in his EMC career, he managed the statistical engineering group as well as led the data warehousing efforts in an Enterprise Resource Planning (ERP) implementation. Prior to joining EMC, Barry held managerial and analytical roles in reliability engineering functions at medical diagnostic and technology companies. During his career, he has applied his quantitative skill set to a myriad of business applications in the Customer Service, Engineering, Manufacturing, Sales/Marketing, Finance, and Legal arenas. Underscoring the importance of strong executive stakeholder engagement, many of his successes have resulted from not only focusing on the technical details of an analysis, but on the decisions that will be resulting from the analysis. Barry earned a B.S. in Computational Mathematics from the Rochester Institute of Technology and an M.A. in Mathematics from the State University of New York (SUNY) New Paltz.

Beibei Yang is a Technical Education Consultant of EMC Education Services, responsible for developing several open courses at EMC related to Data Science and Big Data Analytics. Beibei has seven years of experience in the IT industry. Prior to EMC she worked as a software engineer, systems manager, and network manager for a Fortune 500 company where she introduced new technologies to improve efficiency and encourage collaboration. Beibei has published papers to prestigious conferences and has filed multiple patents. She received her Ph.D. in computer science from the University of Massachusetts Lowell. She has a passion toward natural language processing and data mining, especially using various tools and techniques to find hidden patterns and tell stories with data. Data Science and Big Data Analytics is an exciting domain where the potential of digital information is maximized for making intelligent business decisions. We believe that this is an area that will attract a lot of talented students and professionals in the short, mid, and long term.

Acknowledgments

EMC Education Services embarked on learning this subject with the intent to develop an "open" curriculum and certification. It was a challenging journey at the time as not many understood what it would take to be a true data scientist. After initial research (and struggle), we were able to define what was needed and attract very talented professionals to work on the project. The course, "Data Science and Big Data Analytics," has become well accepted across academia and the industry.

Led by EMC Education Services, this book is the result of efforts and contributions from a number of key EMC organizations and supported by the office of the CTO, IT, Global Services, and Engineering. Many sincere thanks to many key contributors and subject matter experts **David Dietrich, Barry Heller,** and **Beibei Yang** for their work developing content and graphics for the chapters. A special thanks to subject matter experts **John Cardente** and **Ganesh Rajaratnam** for their active involvement reviewing multiple book chapters and providing valuable feedback throughout the project.

We are also grateful to the following experts from EMC and Pivotal for their support in reviewing and improving the content in this book:

Aidan O'Brien	Joe Kambourakis
Alexander Nunes	Joe Milardo
Bryan Miletich	John Sopka
Dan Baskette	Kathryn Stiles
Daniel Mepham	Ken Taylor
Dave Reiner	Lanette Wells
Deborah Stokes	Michael Hancock
Ellis Kriesberg	Michael Vander Donk
Frank Coleman	Narayanan Krishnakumar
Hisham Arafat	Richard Moore
Ira Schild	Ron Glick
Jack Harwood	Stephen Maloney
Jim McGroddy	Steve Todd

Jody Goncalves	Suresh Thankappan
Joe Dery	Tom McGowan

We also thank Ira Schild and Shane Goodrich for coordinating this project, Mallesh Gurram for the cover design, Chris Conroy and Rob Bradley for graphics, and the publisher, John Wiley and Sons, for timely support in bringing this book to the industry.

Nancy Gessler
Director, Education Services, EMC Corporation
Alok Shrivastava
Sr. Director, Education Services, EMC Corporation

Contents

Introduction ..... *xvii*

Chapter 1 • Introduction to Big Data Analytics **1**

1.1 Big Data Overview 2

1.1.1 Data Structures *5*

1.1.2 Analyst Perspective on Data Repositories *9*

1.2 State of the Practice in Analytics 11

1.2.1 BI Versus Data Science *12*

1.2.2 Current Analytical Architecture *13*

1.2.3 Drivers of Big Data *15*

1.2.4 Emerging Big Data Ecosystem and a New Approach to Analytics *16*

1.3 Key Roles for the New Big Data Ecosystem 19

1.4 Examples of Big Data Analytics 22

Summary 23

Exercises 23

Bibliography 24

Chapter 2 • Data Analytics Lifecycle **25**

2.1 Data Analytics Lifecycle Overview 26

2.1.1 Key Roles for a Successful Analytics Project *26*

2.1.2 Background and Overview of Data Analytics Lifecycle *28*

2.2 Phase 1: Discovery 30

2.2.1 Learning the Business Domain *30*

2.2.2 Resources *31*

2.2.3 Framing the Problem *32*

2.2.4 Identifying Key Stakeholders *33*

2.2.5 Interviewing the Analytics Sponsor *33*

2.2.6 Developing Initial Hypotheses *35*

2.2.7 Identifying Potential Data Sources *35*

2.3 Phase 2: Data Preparation 36

2.3.1 Preparing the Analytic Sandbox *37*

2.3.2 Performing ETLT *38*

2.3.3 Learning About the Data *39*

2.3.4 Data Conditioning *40*

2.3.5 Survey and Visualize *41*

2.3.6 Common Tools for the Data Preparation Phase *42*

2.4 Phase 3: Model Planning 42

2.4.1 Data Exploration and Variable Selection *44*

2.4.2 Model Selection *45*

2.4.3 Common Tools for the Model Planning Phase *45*

2.5 Phase 4: Model Building . . . 46
2.5.1 Common Tools for the Model Building Phase . . . 48
2.6 Phase 5: Communicate Results . . . 49
2.7 Phase 6: Operationalize . . . 50
2.8 Case Study: Global Innovation Network and Analysis (GINA) . . . 53
2.8.1 Phase 1: Discovery . . . 54
2.8.2 Phase 2: Data Preparation . . . 55
2.8.3 Phase 3: Model Planning . . . 56
2.8.4 Phase 4: Model Building . . . 56
2.8.5 Phase 5: Communicate Results . . . 58
2.8.6 Phase 6: Operationalize . . . 59
Summary . . . 60
Exercises . . . 61
Bibliography . . . 61

Chapter 3 • Review of Basic Data Analytic Methods Using R . . . 63

3.1 Introduction to R . . . 64
3.1.1 R Graphical User Interfaces . . . 67
3.1.2 Data Import and Export . . . 69
3.1.3 Attribute and Data Types . . . 71
3.1.4 Descriptive Statistics . . . 79
3.2 Exploratory Data Analysis . . . 80
3.2.1 Visualization Before Analysis . . . 82
3.2.2 Dirty Data . . . 85
3.2.3 Visualizing a Single Variable . . . 88
3.2.4 Examining Multiple Variables . . . 91
3.2.5 Data Exploration Versus Presentation . . . 99
3.3 Statistical Methods for Evaluation . . . 101
3.3.1 Hypothesis Testing . . . 102
3.3.2 Difference of Means . . . 104
3.3.3 Wilcoxon Rank-Sum Test . . . 108
3.3.4 Type I and Type II Errors . . . 109
3.3.5 Power and Sample Size . . . 110
3.3.6 ANOVA . . . 110
Summary . . . 114
Exercises . . . 114
Bibliography . . . 115

Chapter 4 • Advanced Analytical Theory and Methods: Clustering . . . 117

4.1 Overview of Clustering . . . 118
4.2 K-means . . . 118
4.2.1 Use Cases . . . 119
4.2.2 Overview of the Method . . . 120
4.2.3 Determining the Number of Clusters . . . 123
4.2.4 Diagnostics . . . 128

4.2.5 Reasons to Choose and Cautions *130*
4.3 Additional Algorithms 134
Summary 135
Exercises 135
Bibliography 136

Chapter 5 • Advanced Analytical Theory and Methods: Association Rules 137
5.1 Overview 138
5.2 Apriori Algorithm 140
5.3 Evaluation of Candidate Rules 141
5.4 Applications of Association Rules 143
5.5 An Example: Transactions in a Grocery Store 143
5.5.1 The Groceries Dataset *144*
5.5.2 Frequent Itemset Generation *146*
5.5.3 Rule Generation and Visualization *152*
5.6 Validation and Testing 157
5.7 Diagnostics 158
Summary 158
Exercises 159
Bibliography 160

Chapter 6 • Advanced Analytical Theory and Methods: Regression 161
6.1 Linear Regression 162
6.1.1 Use Cases *162*
6.1.2 Model Description *163*
6.1.3 Diagnostics *173*
6.2 Logistic Regression 178
6.2.1 Use Cases *179*
6.2.2 Model Description *179*
6.2.3 Diagnostics *181*
6.3 Reasons to Choose and Cautions 188
6.4 Additional Regression Models 189
Summary 190
Exercises 190

Chapter 7 • Advanced Analytical Theory and Methods: Classification 191
7.1 Decision Trees 192
7.1.1 Overview of a Decision Tree *193*
7.1.2 The General Algorithm *197*
7.1.3 Decision Tree Algorithms *203*
7.1.4 Evaluating a Decision Tree *204*
7.1.5 Decision Trees in R *206*
7.2 Naïve Bayes 211
7.2.1 Bayes' Theorem *212*
7.2.2 Naïve Bayes Classifier *214*

7.2.3 Smoothing *217*
7.2.4 Diagnostics *217*
7.2.5 Naïve Bayes in R *218*
7.3 Diagnostics of Classifiers 224
7.4 Additional Classification Methods 228
Summary 229
Exercises 230
Bibliography 231

Chapter 8 • Advanced Analytical Theory and Methods: Time Series Analysis 233

8.1 Overview of Time Series Analysis 234
8.1.1 Box-Jenkins Methodology *235*
8.2 ARIMA Model 236
8.2.1 Autocorrelation Function (ACF) *236*
8.2.2 Autoregressive Models *238*
8.2.3 Moving Average Models *239*
8.2.4 ARMA and ARIMA Models *241*
8.2.5 Building and Evaluating an ARIMA Model *244*
8.2.6 Reasons to Choose and Cautions *252*
8.3 Additional Methods 253
Summary 254
Exercises 254

Chapter 9 • Advanced Analytical Theory and Methods: Text Analysis 255

9.1 Text Analysis Steps 257
9.2 A Text Analysis Example 259
9.3 Collecting Raw Text 260
9.4 Representing Text 264
9.5 Term Frequency—Inverse Document Frequency (TFIDF) 269
9.6 Categorizing Documents by Topics 274
9.7 Determining Sentiments 277
9.8 Gaining Insights 283
Summary 290
Exercises 290
Bibliography 291

Chapter 10 • Advanced Analytics—Technology and Tools: MapReduce and Hadoop 295

10.1 Analytics for Unstructured Data 296
10.1.1 Use Cases *296*
10.1.2 MapReduce *298*
10.1.3 Apache Hadoop *300*
10.2 The Hadoop Ecosystem 306
10.2.1 Pig *306*
10.2.2 Hive *308*
10.2.3 HBase *311*
10.2.4 Mahout *319*

10.3 NoSQL 322
Summary 323
Exercises 324
Bibliography 324

Chapter 11 • Advanced Analytics—Technology and Tools: In-Database Analytics 327

11.1 SQL Essentials 328
11.1.1 Joins 330
11.1.2 Set Operations 332
11.1.3 Grouping Extensions 334
11.2 In-Database Text Analysis 338
11.3 Advanced SQL 343
11.3.1 Window Functions 343
11.3.2 User-Defined Functions and Aggregates 347
11.3.3 Ordered Aggregates 351
11.3.4 MADlib 352
Summary 356
Exercises 356
Bibliography 357

Chapter 12 • The Endgame, or Putting It All Together 359

12.1 Communicating and Operationalizing an Analytics Project 360
12.2 Creating the Final Deliverables 362
12.2.1 Developing Core Material for Multiple Audiences 364
12.2.2 Project Goals 365
12.2.3 Main Findings 367
12.2.4 Approach 369
12.2.5 Model Description 371
12.2.6 Key Points Supported with Data 372
12.2.7 Model Details 372
12.2.8 Recommendations 374
12.2.9 Additional Tips on Final Presentation 375
12.2.10 Providing Technical Specifications and Code 376
12.3 Data Visualization Basics 377
12.3.1 Key Points Supported with Data 378
12.3.2 Evolution of a Graph 380
12.3.3 Common Representation Methods 386
12.3.4 How to Clean Up a Graphic 387
12.3.5 Additional Considerations 392
Summary 393
Exercises 394
References and Further Reading 394
Bibliography 394

Index 397

Foreword

Technological advances and the associated changes in practical daily life have produced a rapidly expanding "parallel universe" of new content, new data, and new information sources all around us. Regardless of how one defines it, the phenomenon of Big Data is ever more present, ever more pervasive, and ever more important. There is enormous value potential in Big Data: innovative insights, improved understanding of problems, and countless opportunities to predict—and even to shape—the future. Data Science is the principal means to discover and tap that potential. Data Science provides ways to deal with and benefit from Big Data: to see patterns, to discover relationships, and to make sense of stunningly varied images and information.

Not everyone has studied statistical analysis at a deep level. People with advanced degrees in applied mathematics are not a commodity. Relatively few organizations have committed resources to large collections of data gathered primarily for the purpose of exploratory analysis. And yet, while applying the practices of Data Science to Big Data is a valuable differentiating strategy at present, it will be a standard core competency in the not so distant future.

How does an organization operationalize quickly to take advantage of this trend? We've created this book for that exact purpose.

EMC Education Services has been listening to the industry and organizations, observing the multi-faceted transformation of the technology landscape, and doing direct research in order to create curriculum and content to help individuals and organizations transform themselves. For the domain of Data Science and Big Data Analytics, our educational strategy balances three things: *people*—especially in the context of data science teams, *processes*—such as the analytic lifecycle approach presented in this book, and *tools and technologies*—in this case with the emphasis on proven analytic tools.

So let us help you capitalize on this new "parallel universe" that surrounds us. We invite you to learn about Data Science and Big Data Analytics through this book and hope it significantly accelerates your efforts in the transformational process.

Introduction

Big Data is creating significant new opportunities for organizations to derive new value and create competitive advantage from their most valuable asset: information. For businesses, Big Data helps drive efficiency, quality, and personalized products and services, producing improved levels of customer satisfaction and profit. For scientific efforts, Big Data analytics enable new avenues of investigation with potentially richer results and deeper insights than previously available. In many cases, Big Data analytics integrate structured and unstructured data with real-time feeds and queries, opening new paths to innovation and insight.

This book provides a practitioner's approach to some of the key techniques and tools used in Big Data analytics. Knowledge of these methods will help people become active contributors to Big Data analytics projects. The book's content is designed to assist multiple stakeholders: business and data analysts looking to add Big Data analytics skills to their portfolio; database professionals and managers of business intelligence, analytics, or Big Data groups looking to enrich their analytic skills; and college graduates investigating data science as a career field.

The content is structured in twelve chapters. The first chapter introduces the reader to the domain of Big Data, the drivers for advanced analytics, and the role of the data scientist. The second chapter presents an analytic project lifecycle designed for the particular characteristics and challenges of hypothesis-driven analysis with Big Data.

Chapter 3 examines fundamental statistical techniques in the context of the open source R analytic software environment. This chapter also highlights the importance of exploratory data analysis via visualizations and reviews the key notions of hypothesis development and testing.

Chapters 4 through 9 discuss a range of advanced analytical methods, including clustering, classification, regression analysis, time series and text analysis.

Chapters 10 and 11 focus on specific technologies and tools that support advanced analytics with Big Data. In particular, the MapReduce paradigm and its instantiation in the Hadoop ecosystem, as well as advanced topics in SQL and in-database text analytics form the focus of these chapters.

Chapter 12 provides guidance on operationalizing Big Data analytics projects. This chapter focuses on creating the final deliverables, converting an analytics project to an ongoing asset of an organization's operation, and creating clear, useful visual outputs based on the data.

EMC Academic Alliance

University and college faculties are invited to join the Academic Alliance program to access unique "open" curriculum-based education on the following topics:

- Data Science and Big Data Analytics
- Information Storage and Management
- Cloud Infrastructure and Services
- Backup Recovery Systems and Architecture

The program provides faculty with course resources to prepare students for opportunities that exist in today's evolving IT industry at no cost. For more information, visit `http://education.EMC.com/academicalliance`.

EMC Proven Professional Certification

EMC Proven Professional is a leading education and certification program in the IT industry, providing comprehensive coverage of information storage technologies, virtualization, cloud computing, data science/Big Data analytics, and more.

Being proven means investing in yourself and formally validating your expertise.

This book prepares you for Data Science Associate (EMCDSA) certification. Visit `http://education.EMC.com` for details.

1

Introduction to Big Data Analytics

Key Concepts

Big Data overview
State of the practice in analytics
Business Intelligence versus Data Science
Key roles for the new Big Data ecosystem
The Data Scientist
Examples of Big Data analytics

Much has been written about Big Data and the need for advanced analytics within industry, academia, and government. Availability of new data sources and the rise of more complex analytical opportunities have created a need to rethink existing data architectures to enable analytics that take advantage of Big Data. In addition, significant debate exists about what Big Data is and what kinds of skills are required to make best use of it. This chapter explains several key concepts to clarify what is meant by Big Data, why advanced analytics are needed, how Data Science differs from Business Intelligence (BI), and what new roles are needed for the new Big Data ecosystem.

1.1 Big Data Overview

Data is created constantly, and at an ever-increasing rate. Mobile phones, social media, imaging technologies to determine a medical diagnosis—all these and more create new data, and that must be stored somewhere for some purpose. Devices and sensors automatically generate diagnostic information that needs to be stored and processed in real time. Merely keeping up with this huge influx of data is difficult, but substantially more challenging is analyzing vast amounts of it, especially when it does not conform to traditional notions of data structure, to identify meaningful patterns and extract useful information. These challenges of the data deluge present the opportunity to transform business, government, science, and everyday life.

Several industries have led the way in developing their ability to gather and exploit data:

- Credit card companies monitor every purchase their customers make and can identify fraudulent purchases with a high degree of accuracy using rules derived by processing billions of transactions.
- Mobile phone companies analyze subscribers' calling patterns to determine, for example, whether a caller's frequent contacts are on a rival network. If that rival network is offering an attractive promotion that might cause the subscriber to defect, the mobile phone company can proactively offer the subscriber an incentive to remain in her contract.
- For companies such as LinkedIn and Facebook, data itself is their primary product. The valuations of these companies are heavily derived from the data they gather and host, which contains more and more intrinsic value as the data grows.

Three attributes stand out as defining Big Data characteristics:

- **Huge volume of data:** Rather than thousands or millions of rows, Big Data can be billions of rows and millions of columns.
- **Complexity of data types and structures:** Big Data reflects the variety of new data sources, formats, and structures, including digital traces being left on the web and other digital repositories for subsequent analysis.
- **Speed of new data creation and growth:** Big Data can describe high velocity data, with rapid data ingestion and near real time analysis.

Although the volume of Big Data tends to attract the most attention, generally the variety and velocity of the data provide a more apt definition of Big Data. (Big Data is sometimes described as having 3 Vs: volume, variety, and velocity.) Due to its size or structure, Big Data cannot be efficiently analyzed using only traditional databases or methods. Big Data problems require new tools and technologies to store, manage, and realize the business benefit. These new tools and technologies enable creation, manipulation, and

management of large datasets and the storage environments that house them. Another definition of Big Data comes from the McKinsey Global report from 2011:

> ***Big Data is data whose scale, distribution, diversity, and/or timeliness require the use of new technical architectures and analytics to enable insights that unlock new sources of business value.***
>
> McKinsey & Co.; Big Data: The Next Frontier for Innovation, Competition, and Productivity [1]

McKinsey's definition of Big Data implies that organizations will need new data architectures and analytic sandboxes, new tools, new analytical methods, and an integration of multiple skills into the new role of the data scientist, which will be discussed in Section 1.3. Figure 1-1 highlights several sources of the Big Data deluge.

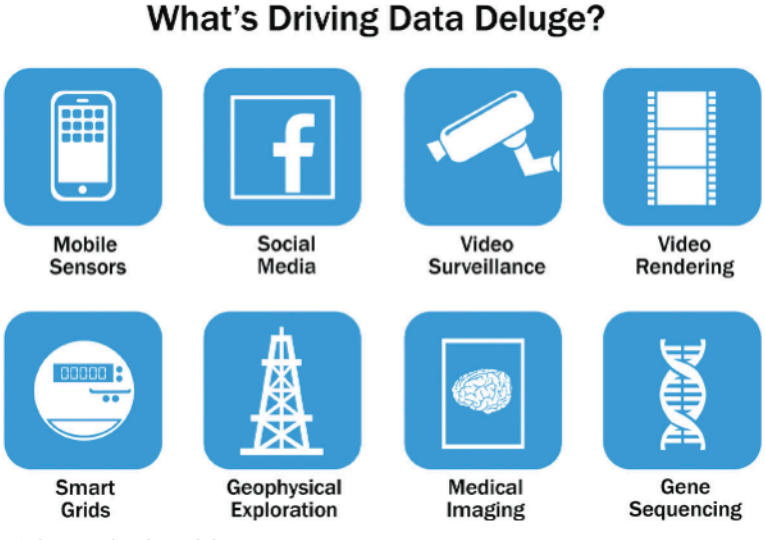


FIGURE 1-1 *What's driving the data deluge*

The rate of data creation is accelerating, driven by many of the items in Figure 1-1.

Social media and genetic sequencing are among the fastest-growing sources of Big Data and examples of untraditional sources of data being used for analysis.

For example, in 2012 Facebook users posted 700 status updates per second worldwide, which can be leveraged to deduce latent interests or political views of users and show relevant ads. For instance, an update in which a woman changes her relationship status from "single" to "engaged" would trigger ads on bridal dresses, wedding planning, or name-changing services.

Facebook can also construct social graphs to analyze which users are connected to each other as an interconnected network. In March 2013, Facebook released a new feature called "Graph Search," enabling users and developers to search social graphs for people with similar interests, hobbies, and shared locations.

Another example comes from genomics. Genetic sequencing and human genome mapping provide a detailed understanding of genetic makeup and lineage. The health care industry is looking toward these advances to help predict which illnesses a person is likely to get in his lifetime and take steps to avoid these maladies or reduce their impact through the use of personalized medicine and treatment. Such tests also highlight typical responses to different medications and pharmaceutical drugs, heightening risk awareness of specific drug treatments.

While data has grown, the cost to perform this work has fallen dramatically. The cost to sequence one human genome has fallen from $100 million in 2001 to $10,000 in 2011, and the cost continues to drop. Now, websites such as 23andme (Figure 1-2) offer genotyping for less than $100. Although genotyping analyzes only a fraction of a genome and does not provide as much granularity as genetic sequencing, it does point to the fact that data and complex analysis is becoming more prevalent and less expensive to deploy.

FIGURE 1-2 *Examples of what can be learned through genotyping, from 23andme.com*

As illustrated by the examples of social media and genetic sequencing, individuals and organizations both derive benefits from analysis of ever-larger and more complex datasets that require increasingly powerful analytical capabilities.

1.1.1 Data Structures

Big data can come in multiple forms, including structured and non-structured data such as financial data, text files, multimedia files, and genetic mappings. Contrary to much of the traditional data analysis performed by organizations, most of the Big Data is unstructured or semi-structured in nature, which requires different techniques and tools to process and analyze. [2] Distributed computing environments and massively parallel processing (MPP) architectures that enable parallelized data ingest and analysis are the preferred approach to process such complex data.

With this in mind, this section takes a closer look at data structures.

Figure 1-3 shows four types of data structures, with 80–90% of future data growth coming from non-structured data types. [2] Though different, the four are commonly mixed. For example, a classic Relational Database Management System (RDBMS) may store call logs for a software support call center. The RDBMS may store characteristics of the support calls as typical structured data, with attributes such as time stamps, machine type, problem type, and operating system. In addition, the system will likely have unstructured, quasi- or semi-structured data, such as free-form call log information taken from an e-mail ticket of the problem, customer chat history, or transcript of a phone call describing the technical problem and the solution or audio file of the phone call conversation. Many insights could be extracted from the unstructured, quasi- or semi-structured data in the call center data.

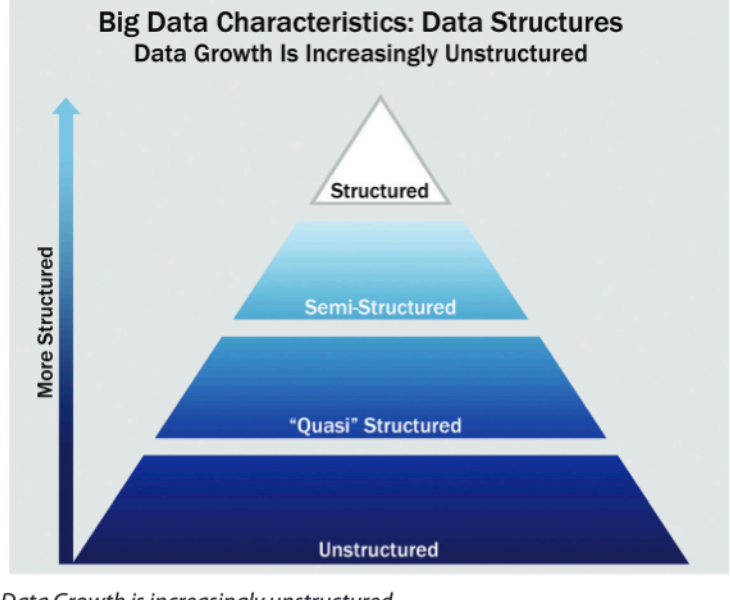


Figure 1-3 *Big Data Growth is increasingly unstructured*

Although analyzing structured data tends to be the most familiar technique, a different technique is required to meet the challenges to analyze semi-structured data (shown as XML), quasi-structured (shown as a clickstream), and unstructured data.

Here are examples of how each of the four main types of data structures may look.

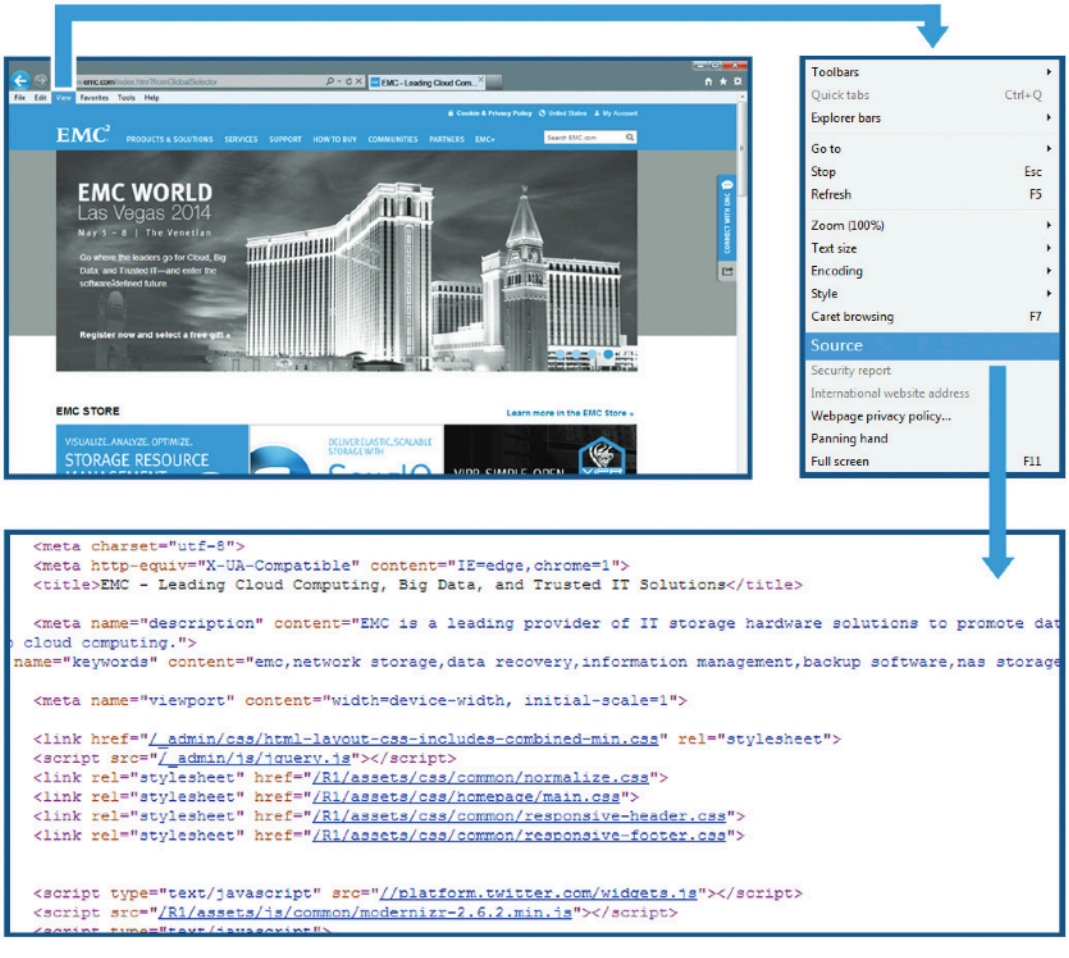
- **Structured data:** Data containing a defined data type, format, and structure (that is, transaction data, online analytical processing [OLAP] data cubes, traditional RDBMS, CSV files, and even simple spreadsheets). See Figure 1-4.

SUMMER FOOD SERVICE PROGRAM 1]				
(Data as of August 01, 2011)				
Fiscal Year	Number of Sites	Peak (July) Participation	Meals Served	Total Federal Expenditures 2]
	----------Thousands----------		--Mil.--	---Million $---
1969	1.2	99	2.2	0.3
1970	1.9	227	8.2	1.8
1971	3.2	569	29.0	8.2
1972	6.5	1,080	73.5	21.9
1973	11.2	1,437	65.4	26.6
1974	10.6	1,403	63.6	33.6
1975	12.0	1,785	84.3	50.3
1976	16.0	2,453	104.8	73.4
TQ 3]	22.4	3,455	198.0	88.9
1977	23.7	2,791	170.4	114.4
1978	22.4	2,333	120.3	100.3
1979	23.0	2,126	121.8	108.6
1980	21.6	1,922	108.2	110.1
1981	20.6	1,726	90.3	105.9
1982	14.4	1,397	68.2	87.1
1983	14.9	1,401	71.3	93.4
1984	15.1	1,422	73.8	96.2
1985	16.0	1,462	77.2	111.5
1986	16.1	1,509	77.1	114.7
1987	16.9	1,560	79.9	129.3
1988	17.2	1,577	80.3	133.3
1989	18.5	1,652	86.0	143.8
1990	19.2	1,692	91.2	163.3

FIGURE 1-4 *Example of structured data*

- **Semi-structured data:** Textual data files with a discernible pattern that enables parsing (such as Extensible Markup Language [XML] data files that are self-describing and defined by an XML schema). See Figure 1-5.
- **Quasi-structured data:** Textual data with erratic data formats that can be formatted with effort, tools, and time (for instance, web clickstream data that may contain inconsistencies in data values and formats). See Figure 1-6.
- **Unstructured data:** Data that has no inherent structure, which may include text documents, PDFs, images, and video. See Figure 1-7.

Quasi-structured data is a common phenomenon that bears closer scrutiny. Consider the following example. A user attends the EMC World conference and subsequently runs a Google search online to find information related to EMC and Data Science. This would produce a URL such as `https://www.google.com/#q=EMC+ data+science` and a list of results, such as in the first graphic of Figure 1-5.


EMC WORLD
Las Vegas 2014
EMC STORE
STORAGE RESOURCE
Toolbars
Quick tabs Ctrl+Q
Explorer bars
Go to
Stop Esc
Refresh F5
Zoom (100%)
Text size
Encoding
Style
Caret browsing F7
Source
Security report
International website address
Webpage privacy policy...
Panning hand
Full screen F11
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<title>EMC - Leading Cloud Computing, Big Data, and Trusted IT Solutions</title>
<meta name="description" content="EMC is a leading provider of IT storage hardware solutions to promote dat
cloud computing.">
name="keywords" content="emc,network storage,data recovery,information management,backup software,nas storage
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="/_admin/css/html-layout-css-includes-combined-min.css" rel="stylesheet">
<script src="/_admin/js/jquery.js"></script>
<link rel="stylesheet" href="/R1/assets/css/common/normalize.css">
<link rel="stylesheet" href="/R1/assets/css/homepage/main.css">
<link rel="stylesheet" href="/R1/assets/css/common/responsive-header.css">
<link rel="stylesheet" href="/R1/assets/css/common/responsive-footer.css">
<script type="text/javascript" src="//platform.twitter.com/widgets.js"></script>
<script src="/R1/assets/js/common/modernizr-2.6.2.min.js"></script></script>

.aspx, that displays the page shown as (2) in Figure 1-6. Arriving at this site, the user may decide to click to learn more about the process of becoming certified in data science. The user chooses a link toward the top of the page on Certifications, bringing the user to a new URL: `https://education.emc.com/guest/certification/framework/stf/data_science.aspx`, which is (3) in Figure 1-6.

Visiting these three websites adds three URLs to the log files monitoring the user's computer or network use. These three URLs are:

```
https://www.google.com/#q=EMC+data+science
https://education.emc.com/guest/campaign/data_science.aspx
https://education.emc.com/guest/certification/framework/stf/data_
science.aspx
```

FIGURE 1-6 *Example of EMC Data Science search results*

FIGURE 1-7 *Example of unstructured data: video about Antarctica expedition [3]*

This set of three URLs reflects the websites and actions taken to find Data Science information related to EMC. Together, this comprises a *clickstream* that can be parsed and mined by data scientists to discover usage patterns and uncover relationships among clicks and areas of interest on a website or group of sites.

The four data types described in this chapter are sometimes generalized into two groups: structured and unstructured data. Big Data describes new kinds of data with which most organizations may not be used to working. With this in mind, the next section discusses common technology architectures from the standpoint of someone wanting to analyze Big Data.

1.1.2 Analyst Perspective on Data Repositories

The introduction of spreadsheets enabled business users to create simple logic on data structured in rows and columns and create their own analyses of business problems. Database administrator training is not required to create spreadsheets: They can be set up to do many things quickly and independently of information technology (IT) groups. Spreadsheets are easy to share, and end users have control over the logic involved. However, their proliferation can result in "many versions of the truth." In other words, it can be challenging to determine if a particular user has the most relevant version of a spreadsheet, with the most current data and logic in it. Moreover, if a laptop is lost or a file becomes corrupted, the data and logic within the spreadsheet could be lost. This is an ongoing challenge because spreadsheet programs such as Microsoft Excel still run on many computers worldwide. With the proliferation of data islands (or spreadmarts), the need to centralize the data is more pressing than ever.

As data needs grew, so did more scalable data warehousing solutions. These technologies enabled data to be managed centrally, providing benefits of security, failover, and a single repository where users

could rely on getting an "official" source of data for financial reporting or other mission-critical tasks. This structure also enabled the creation of OLAP cubes and BI analytical tools, which provided quick access to a set of dimensions within an RDBMS. More advanced features enabled performance of in-depth analytical techniques such as regressions and neural networks. Enterprise Data Warehouses (EDWs) are critical for reporting and BI tasks and solve many of the problems that proliferating spreadsheets introduce, such as which of multiple versions of a spreadsheet is correct. EDWs—and a good BI strategy—provide direct data feeds from sources that are centrally managed, backed up, and secured.

Despite the benefits of EDWs and BI, these systems tend to restrict the flexibility needed to perform robust or exploratory data analysis. With the EDW model, data is managed and controlled by IT groups and database administrators (DBAs), and data analysts must depend on IT for access and changes to the data schemas. This imposes longer lead times for analysts to get data; most of the time is spent waiting for approvals rather than starting meaningful work. Additionally, many times the EDW rules restrict analysts from building datasets. Consequently, it is common for additional systems to emerge containing critical data for constructing analytic datasets, managed locally by power users. IT groups generally dislike existence of data sources outside of their control because, unlike an EDW, these datasets are not managed, secured, or backed up. From an analyst perspective, EDW and BI solve problems related to data accuracy and availability. However, EDW and BI introduce new problems related to flexibility and agility, which were less pronounced when dealing with spreadsheets.

A solution to this problem is the analytic sandbox, which attempts to resolve the conflict for analysts and data scientists with EDW and more formally managed corporate data. In this model, the IT group may still manage the analytic sandboxes, but they will be purposefully designed to enable robust analytics, while being centrally managed and secured. These sandboxes, often referred to as *workspaces,* are designed to enable teams to explore many datasets in a controlled fashion and are not typically used for enterprise-level financial reporting and sales dashboards.

Many times, analytic sandboxes enable high-performance computing using in-database processing—the analytics occur within the database itself. The idea is that performance of the analysis will be better if the analytics are run in the database itself, rather than bringing the data to an analytical tool that resides somewhere else. In-database analytics, discussed further in Chapter 11, "Advanced Analytics—Technology and Tools: In-Database Analytics," creates relationships to multiple data sources within an organization and saves time spent creating these data feeds on an individual basis. In-database processing for deep analytics enables faster turnaround time for developing and executing new analytic models, while reducing, though not eliminating, the cost associated with data stored in local, "shadow" file systems. In addition, rather than the typical structured data in the EDW, analytic sandboxes can house a greater variety of data, such as raw data, textual data, and other kinds of unstructured data, without interfering with critical production databases. Table 1-1 summarizes the characteristics of the data repositories mentioned in this section.

TABLE 1-1 *Types of Data Repositories, from an Analyst Perspective*

Data Repository	Characteristics
Spreadsheets and data marts ("spreadmarts")	Spreadsheets and low-volume databases for recordkeeping Analyst depends on data extracts.

Data Warehouses	Centralized data containers in a purpose-built space Supports BI and reporting, but restricts robust analyses Analyst dependent on IT and DBAs for data access and schema changes Analysts must spend significant time to get aggregated and disaggregated data extracts from multiple sources.
Analytic Sandbox (workspaces)	Data assets gathered from multiple sources and technologies for analysis Enables flexible, high-performance analysis in a nonproduction environment; can leverage in-database processing Reduces costs and risks associated with data replication into "shadow" file systems "Analyst owned" rather than "DBA owned"

There are several things to consider with Big Data Analytics projects to ensure the approach fits with the desired goals. Due to the characteristics of Big Data, these projects lend themselves to decision support for high-value, strategic decision making with high processing complexity. The analytic techniques used in this context need to be iterative and flexible, due to the high volume of data and its complexity. Performing rapid and complex analysis requires high throughput network connections and a consideration for the acceptable amount of latency. For instance, developing a real-time product recommender for a website imposes greater system demands than developing a near-real-time recommender, which may still provide acceptable performance, have slightly greater latency, and may be cheaper to deploy. These considerations require a different approach to thinking about analytics challenges, which will be explored further in the next section.

1.2 State of the Practice in Analytics

Current business problems provide many opportunities for organizations to become more analytical and data driven, as shown in Table 1-2.

TABLE 1-2 *Business Drivers for Advanced Analytics*

Business Driver	Examples
Optimize business operations	Sales, pricing, profitability, efficiency
Identify business risk	Customer churn, fraud, default
Predict new business opportunities	Upsell, cross-sell, best new customer prospects
Comply with laws or regulatory requirements	Anti-Money Laundering, Fair Lending, Basel II-III, Sarbanes-Oxley (SOX)

Table 1-2 outlines four categories of common business problems that organizations contend with where they have an opportunity to leverage advanced analytics to create competitive advantage. Rather than only performing standard reporting on these areas, organizations can apply advanced analytical techniques to optimize processes and derive more value from these common tasks. The first three examples do not represent new problems. Organizations have been trying to reduce customer churn, increase sales, and cross-sell customers for many years. What is new is the opportunity to fuse advanced analytical techniques with Big Data to produce more impactful analyses for these traditional problems. The last example portrays emerging regulatory requirements. Many compliance and regulatory laws have been in existence for decades, but additional requirements are added every year, which represent additional complexity and data requirements for organizations. Laws related to anti-money laundering (AML) and fraud prevention require advanced analytical techniques to comply with and manage properly.

1.2.1 BI Versus Data Science

The four business drivers shown in Table 1-2 require a variety of analytical techniques to address them properly. Although much is written generally about analytics, it is important to distinguish between BI and Data Science. As shown in Figure 1-8, there are several ways to compare these groups of analytical techniques.

One way to evaluate the type of analysis being performed is to examine the time horizon and the kind of analytical approaches being used. BI tends to provide reports, dashboards, and queries on business questions for the current period or in the past. BI systems make it easy to answer questions related to quarter-to-date revenue, progress toward quarterly targets, and understand how much of a given product was sold in a prior quarter or year. These questions tend to be closed-ended and explain current or past behavior, typically by aggregating historical data and grouping it in some way. BI provides hindsight and some insight and generally answers questions related to "when" and "where" events occurred.

By comparison, Data Science tends to use disaggregated data in a more forward-looking, exploratory way, focusing on analyzing the present and enabling informed decisions about the future. Rather than aggregating historical data to look at how many of a given product sold in the previous quarter, a team may employ Data Science techniques such as time series analysis, further discussed in Chapter 8, "Advanced Analytical Theory and Methods: Time Series Analysis," to forecast future product sales and revenue more accurately than extending a simple trend line. In addition, Data Science tends to be more exploratory in nature and may use scenario optimization to deal with more open-ended questions. This approach provides insight into current activity and foresight into future events, while generally focusing on questions related to "how" and "why" events occur.

Where BI problems tend to require highly structured data organized in rows and columns for accurate reporting, Data Science projects tend to use many types of data sources, including large or unconventional datasets. Depending on an organization's goals, it may choose to embark on a BI project if it is doing reporting, creating dashboards, or performing simple visualizations, or it may choose Data Science projects if it needs to do a more sophisticated analysis with disaggregated or varied datasets.

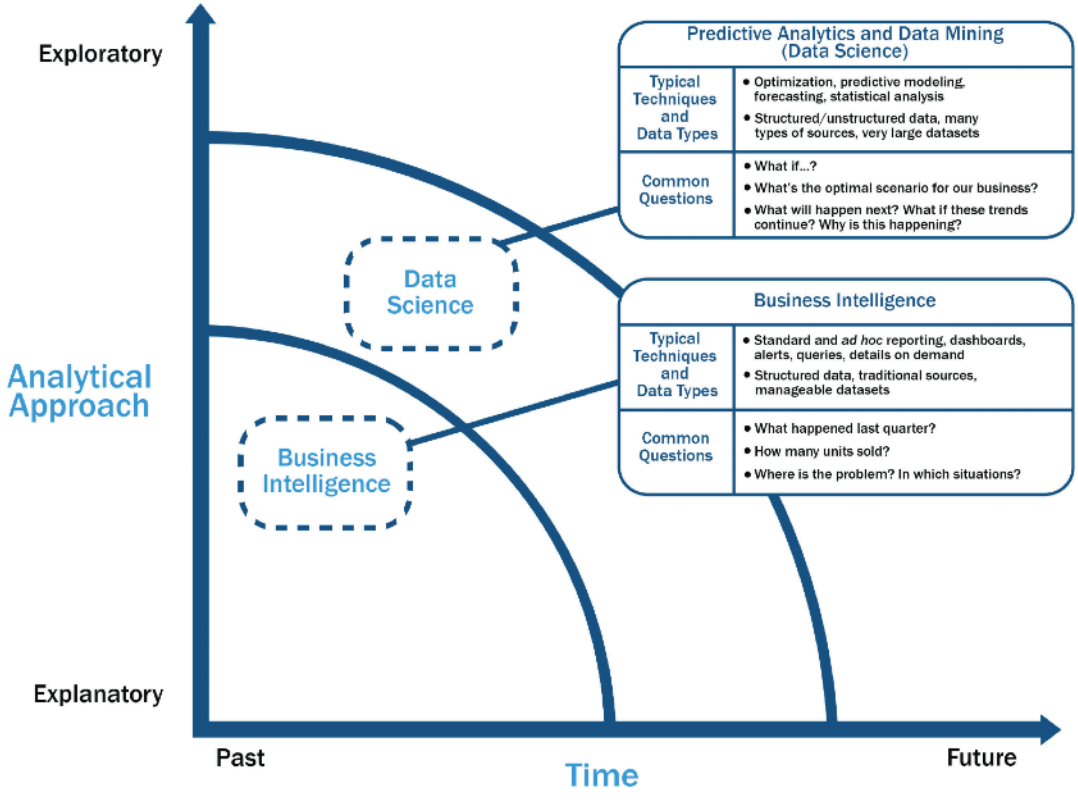


FIGURE 1-8 *Comparing BI with Data Science*

1.2.2 Current Analytical Architecture

As described earlier, Data Science projects need workspaces that are purpose-built for experimenting with data, with flexible and agile data architectures. Most organizations still have data warehouses that provide excellent support for traditional reporting and simple data analysis activities but unfortunately have a more difficult time supporting more robust analyses. This section examines a typical analytical data architecture that may exist within an organization.

Figure 1-9 shows a typical data architecture and several of the challenges it presents to data scientists and others trying to do advanced analytics. This section examines the data flow to the Data Scientist and how this individual fits into the process of getting data to analyze on projects.

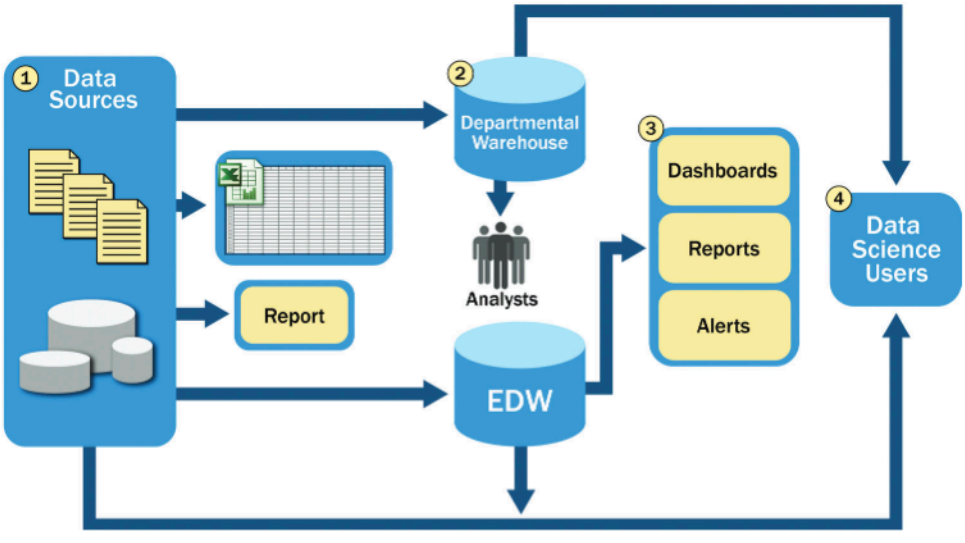


FIGURE 1-9 *Typical analytic architecture*

1. For data sources to be loaded into the data warehouse, data needs to be well understood, structured, and normalized with the appropriate data type definitions. Although this kind of centralization enables security, backup, and failover of highly critical data, it also means that data typically must go through significant preprocessing and checkpoints before it can enter this sort of controlled environment, which does not lend itself to data exploration and iterative analytics.
2. As a result of this level of control on the EDW, additional local systems may emerge in the form of departmental warehouses and local data marts that business users create to accommodate their need for flexible analysis. These local data marts may not have the same constraints for security and structure as the main EDW and allow users to do some level of more in-depth analysis. However, these one-off systems reside in isolation, often are not synchronized or integrated with other data stores, and may not be backed up.
3. Once in the data warehouse, data is read by additional applications across the enterprise for BI and reporting purposes. These are high-priority operational processes getting critical data feeds from the data warehouses and repositories.
4. At the end of this workflow, analysts get data provisioned for their downstream analytics. Because users generally are not allowed to run custom or intensive analytics on production databases, analysts create data extracts from the EDW to analyze data offline in R or other local analytical tools. Many times these tools are limited to in-memory analytics on desktops analyzing samples of data, rather than the entire population of a dataset. Because these analyses are based on data extracts, they reside in a separate location, and the results of the analysis—and any insights on the quality of the data or anomalies—rarely are fed back into the main data repository.

Because new data sources slowly accumulate in the EDW due to the rigorous validation and data structuring process, data is slow to move into the EDW, and the data schema is slow to change.

Departmental data warehouses may have been originally designed for a specific purpose and set of business needs, but over time evolved to house more and more data, some of which may be forced into existing schemas to enable BI and the creation of OLAP cubes for analysis and reporting. Although the EDW achieves the objective of reporting and sometimes the creation of dashboards, EDWs generally limit the ability of analysts to iterate on the data in a separate nonproduction environment where they can conduct in-depth analytics or perform analysis on unstructured data.

The typical data architectures just described are designed for storing and processing mission-critical data, supporting enterprise applications, and enabling corporate reporting activities. Although reports and dashboards are still important for organizations, most traditional data architectures inhibit data exploration and more sophisticated analysis. Moreover, traditional data architectures have several additional implications for data scientists.

- High-value data is hard to reach and leverage, and predictive analytics and data mining activities are last in line for data. Because the EDWs are designed for central data management and reporting, those wanting data for analysis are generally prioritized after operational processes.
- Data moves in batches from EDW to local analytical tools. This workflow means that data scientists are limited to performing in-memory analytics (such as with R, SAS, SPSS, or Excel), which will restrict the size of the datasets they can use. As such, analysis may be subject to constraints of sampling, which can skew model accuracy.
- Data Science projects will remain isolated and ad hoc, rather than centrally managed. The implication of this isolation is that the organization can never harness the power of advanced analytics in a scalable way, and Data Science projects will exist as nonstandard initiatives, which are frequently not aligned with corporate business goals or strategy.

All these symptoms of the traditional data architecture result in a slow "time-to-insight" and lower business impact than could be achieved if the data were more readily accessible and supported by an environment that promoted advanced analytics. As stated earlier, one solution to this problem is to introduce analytic sandboxes to enable data scientists to perform advanced analytics in a controlled and sanctioned way. Meanwhile, the current Data Warehousing solutions continue offering reporting and BI services to support management and mission-critical operations.

1.2.3 Drivers of Big Data

To better understand the market drivers related to Big Data, it is helpful to first understand some past history of data stores and the kinds of repositories and tools to manage these data stores.

As shown in Figure 1-10, in the 1990s the volume of information was often measured in terabytes. Most organizations analyzed structured data in rows and columns and used relational databases and data warehouses to manage large stores of enterprise information. The following decade saw a proliferation of different kinds of data sources—mainly productivity and publishing tools such as content management repositories and networked attached storage systems—to manage this kind of information, and the data began to increase in size and started to be measured at petabyte scales. In the 2010s, the information that organizations try to manage has broadened to include many other kinds of data. In this era, everyone and everything is leaving a digital footprint. Figure 1-10 shows a summary perspective on sources of Big Data generated by new applications and the scale and growth rate of the data. These applications, which generate data volumes that can be measured in exabyte scale, provide opportunities for new analytics and driving new value for organizations. The data now comes from multiple sources, such as these:

- Medical information, such as genomic sequencing and diagnostic imaging
- Photos and video footage uploaded to the World Wide Web
- Video surveillance, such as the thousands of video cameras spread across a city
- Mobile devices, which provide geospatial location data of the users, as well as metadata about text messages, phone calls, and application usage on smart phones
- Smart devices, which provide sensor-based collection of information from smart electric grids, smart buildings, and many other public and industry infrastructures
- Nontraditional IT devices, including the use of radio-frequency identification (RFID) readers, GPS navigation systems, and seismic processing

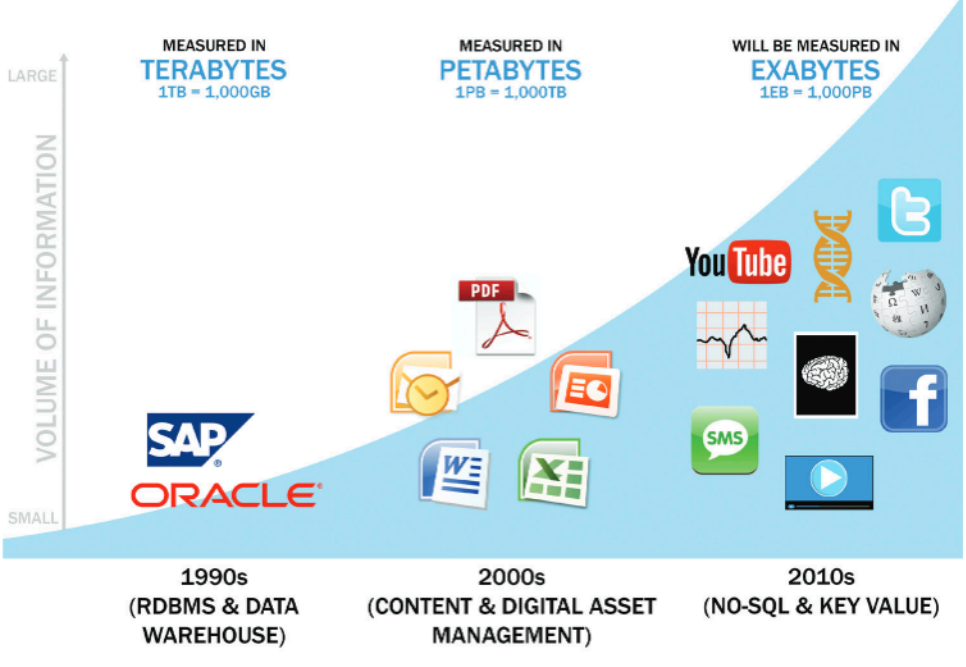


FIGURE 1-10 *Data evolution and the rise of Big Data sources*

The Big Data trend is generating an enormous amount of information from many new sources. This data deluge requires advanced analytics and new market players to take advantage of these opportunities and new market dynamics, which will be discussed in the following section.

1.2.4 Emerging Big Data Ecosystem and a New Approach to Analytics

Organizations and data collectors are realizing that the data they can gather from individuals contains intrinsic value and, as a result, a new economy is emerging. As this new digital economy continues to

evolve, the market sees the introduction of data vendors and data cleaners that use crowdsourcing (such as Mechanical Turk and GalaxyZoo) to test the outcomes of machine learning techniques. Other vendors offer added value by repackaging open source tools in a simpler way and bringing the tools to market. Vendors such as Cloudera, Hortonworks, and Pivotal have provided this value-add for the open source framework Hadoop.

As the new ecosystem takes shape, there are four main groups of players within this interconnected web. These are shown in Figure 1-11.

- **Data devices** [shown in the (1) section of Figure 1-11] and the "Sensornet" gather data from multiple locations and continuously generate new data about this data. For each gigabyte of new data created, an additional petabyte of data is created about that data. [2]
    - For example, consider someone playing an online video game through a PC, game console, or smartphone. In this case, the video game provider captures data about the skill and levels attained by the player. Intelligent systems monitor and log how and when the user plays the game. As a consequence, the game provider can fine-tune the difficulty of the game, suggest other related games that would most likely interest the user, and offer additional equipment and enhancements for the character based on the user's age, gender, and interests. This information may get stored locally or uploaded to the game provider's cloud to analyze the gaming habits and opportunities for upsell and cross-sell, and identify archetypical profiles of specific kinds of users.
    - Smartphones provide another rich source of data. In addition to messaging and basic phone usage, they store and transmit data about Internet usage, SMS usage, and real-time location. This metadata can be used for analyzing traffic patterns by scanning the density of smartphones in locations to track the speed of cars or the relative traffic congestion on busy roads. In this way, GPS devices in cars can give drivers real-time updates and offer alternative routes to avoid traffic delays.
    - Retail shopping loyalty cards record not just the amount an individual spends, but the locations of stores that person visits, the kinds of products purchased, the stores where goods are purchased most often, and the combinations of products purchased together. Collecting this data provides insights into shopping and travel habits and the likelihood of successful advertisement targeting for certain types of retail promotions.
- **Data collectors** [the blue ovals, identified as (2) within Figure 1-11] include sample entities that collect data from the device and users.
    - Data results from a cable TV provider tracking the shows a person watches, which TV channels someone will and will not pay for to watch on demand, and the prices someone is willing to pay for premium TV content
    - Retail stores tracking the path a customer takes through their store while pushing a shopping cart with an RFID chip so they can gauge which products get the most foot traffic using geospatial data collected from the RFID chips
- **Data aggregators** (the dark gray ovals in Figure 1-11, marked as (3)) make sense of the data collected from the various entities from the "SensorNet" or the "Internet of Things." These organizations compile data from the devices and usage patterns collected by government agencies, retail stores,

and websites. In turn, they can choose to transform and package the data as products to sell to list brokers, who may want to generate marketing lists of people who may be good targets for specific ad campaigns.

- **Data users and buyers** are denoted by (4) in Figure 1-11. These groups directly benefit from the data collected and aggregated by others within the data value chain.
  - Retail banks, acting as a data buyer, may want to know which customers have the highest likelihood to apply for a second mortgage or a home equity line of credit. To provide input for this analysis, retail banks may purchase data from a data aggregator. This kind of data may include demographic information about people living in specific locations; people who appear to have a specific level of debt, yet still have solid credit scores (or other characteristics such as paying bills on time and having savings accounts) that can be used to infer credit worthiness; and those who are searching the web for information about paying off debts or doing home remodeling projects. Obtaining data from these various sources and aggregators will enable a more targeted marketing campaign, which would have been more challenging before Big Data due to the lack of information or high-performing technologies.
  - Using technologies such as Hadoop to perform natural language processing on unstructured, textual data from social media websites, users can gauge the reaction to events such as presidential campaigns. People may, for example, want to determine public sentiments toward a candidate by analyzing related blogs and online comments. Similarly, data users may want to track and prepare for natural disasters by identifying which areas a hurricane affects first and how it moves, based on which geographic areas are tweeting about it or discussing it via social media.

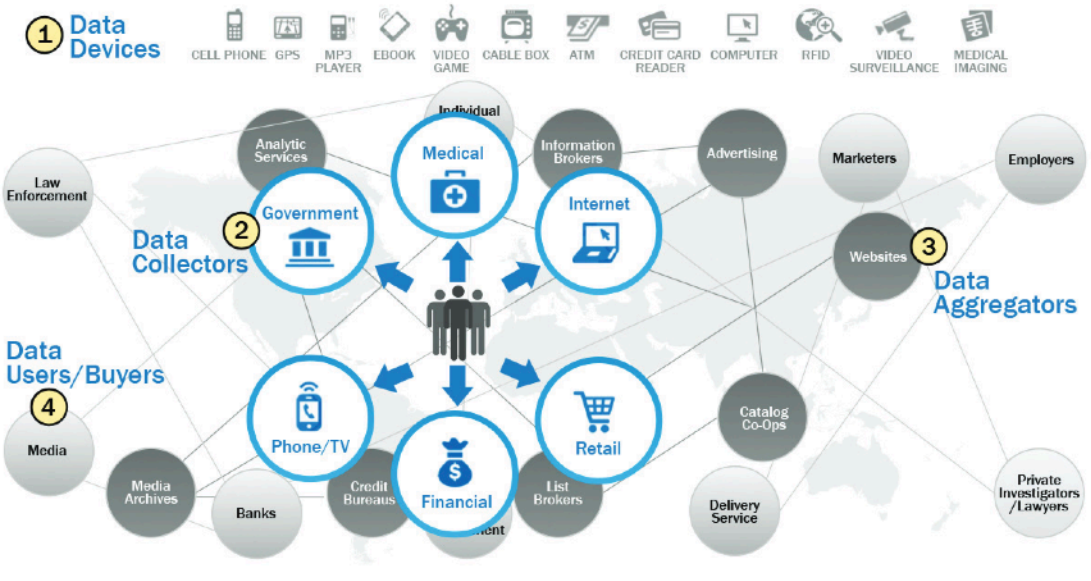


FIGURE 1-11 *Emerging Big Data ecosystem*

As illustrated by this emerging Big Data ecosystem, the kinds of data and the related market dynamics vary greatly. These datasets can include sensor data, text, structured datasets, and social media. With this in mind, it is worth recalling that these datasets will not work well within traditional EDWs, which were architected to streamline reporting and dashboards and be centrally managed. Instead, Big Data problems and projects require different approaches to succeed.

Analysts need to partner with IT and DBAs to get the data they need within an analytic sandbox. A typical analytical sandbox contains raw data, aggregated data, and data with multiple kinds of structure. The sandbox enables robust exploration of data and requires a savvy user to leverage and take advantage of data in the sandbox environment.

1.3 Key Roles for the New Big Data Ecosystem

As explained in the context of the Big Data ecosystem in Section 1.2.4, new players have emerged to curate, store, produce, clean, and transact data. In addition, the need for applying more advanced analytical techniques to increasingly complex business problems has driven the emergence of new roles, new technology platforms, and new analytical methods. This section explores the new roles that address these needs, and subsequent chapters explore some of the analytical methods and technology platforms.

The Big Data ecosystem demands three categories of roles, as shown in Figure 1-12. These roles were described in the McKinsey Global study on Big Data, from May 2011 [1].

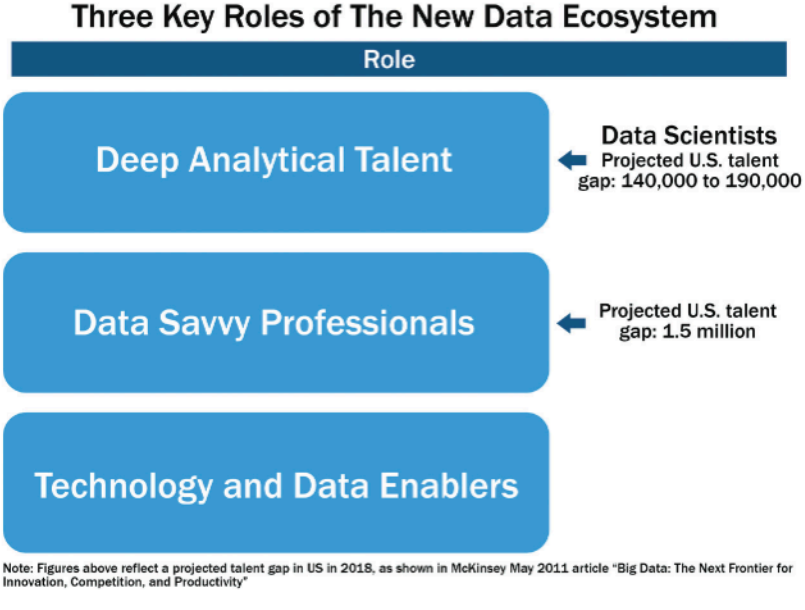


FIGURE 1-12 *Key roles of the new Big Data ecosystem*

The first group—Deep Analytical Talent— is technically savvy, with strong analytical skills. Members possess a combination of skills to handle raw, unstructured data and to apply complex analytical techniques at

massive scales. This group has advanced training in quantitative disciplines, such as mathematics, statistics, and machine learning. To do their jobs, members need access to a robust analytic sandbox or workspace where they can perform large-scale analytical data experiments. Examples of current professions fitting into this group include statisticians, economists, mathematicians, and the new role of the Data Scientist.

The McKinsey study forecasts that by the year 2018, the United States will have a talent gap of 140,000–190,000 people with deep analytical talent. This does not represent the number of people needed with deep analytical talent; rather, this range represents the difference between what will be available in the workforce compared with what will be needed. In addition, these estimates only reflect forecasted talent shortages in the United States; the number would be much larger on a global basis.

The second group—Data Savvy Professionals—has less technical depth but has a basic knowledge of statistics or machine learning and can define key questions that can be answered using advanced analytics. These people tend to have a base knowledge of working with data, or an appreciation for some of the work being performed by data scientists and others with deep analytical talent. Examples of data savvy professionals include financial analysts, market research analysts, life scientists, operations managers, and business and functional managers.

The McKinsey study forecasts the projected U.S. talent gap for this group to be 1.5 million people by the year 2018. At a high level, this means for every Data Scientist profile needed, the gap will be ten times as large for Data Savvy Professionals. Moving toward becoming a data savvy professional is a critical step in broadening the perspective of managers, directors, and leaders, as this provides an idea of the kinds of questions that can be solved with data.

The third category of people mentioned in the study is Technology and Data Enablers. This group represents people providing technical expertise to support analytical projects, such as provisioning and administrating analytical sandboxes, and managing large-scale data architectures that enable widespread analytics within companies and other organizations. This role requires skills related to computer engineering, programming, and database administration.

These three groups must work together closely to solve complex Big Data challenges. Most organizations are familiar with people in the latter two groups mentioned, but the first group, Deep Analytical Talent, tends to be the newest role for most and the least understood. For simplicity, this discussion focuses on the emerging role of the Data Scientist. It describes the kinds of activities that role performs and provides a more detailed view of the skills needed to fulfill that role.

There are three recurring sets of activities that data scientists perform:

- **Reframe business challenges as analytics challenges.** Specifically, this is a skill to diagnose business problems, consider the core of a given problem, and determine which kinds of candidate analytical methods can be applied to solve it. This concept is explored further in Chapter 2, "Data Analytics Lifecycle."

- **Design, implement, and deploy statistical models and data mining techniques on Big Data.** This set of activities is mainly what people think about when they consider the role of the Data Scientist:

namely, applying complex or advanced analytical methods to a variety of business problems using data. Chapter 3 through Chapter 11 of this book introduces the reader to many of the most popular analytical techniques and tools in this area.

- **Develop insights that lead to actionable recommendations.** It is critical to note that applying advanced methods to data problems does not necessarily drive new business value. Instead, it is important to learn how to draw insights out of the data and communicate them effectively. Chapter 12, "The Endgame, or Putting It All Together," has a brief overview of techniques for doing this.

Data scientists are generally thought of as having five main sets of skills and behavioral characteristics, as shown in Figure 1-13:

- **Quantitative skill:** such as mathematics or statistics
- **Technical aptitude:** namely, software engineering, machine learning, and programming skills
- **Skeptical mind-set and critical thinking:** It is important that data scientists can examine their work critically rather than in a one-sided way.
- **Curious and creative**: Data scientists are passionate about data and finding creative ways to solve problems and portray information.
- **Communicative and collaborative:** Data scientists must be able to articulate the business value in a clear way and collaboratively work with other groups, including project sponsors and key stakeholders.

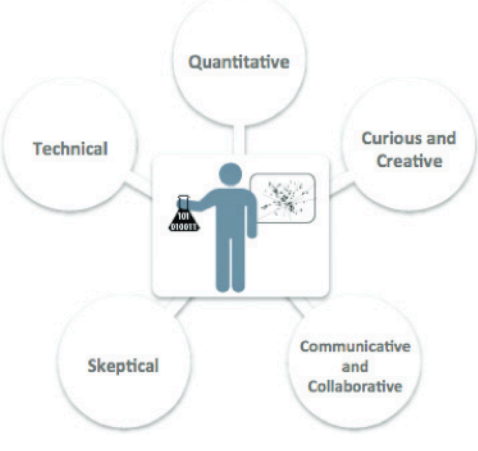


FIGURE 1-13 *Profile of a Data Scientist*

Data scientists are generally comfortable using this blend of skills to acquire, manage, analyze, and visualize data and tell compelling stories about it. The next section includes examples of what Data Science teams have created to drive new value or innovation with Big Data.

1.4 Examples of Big Data Analytics

After describing the emerging Big Data ecosystem and new roles needed to support its growth, this section provides three examples of Big Data Analytics in different areas: retail, IT infrastructure, and social media.

As mentioned earlier, Big Data presents many opportunities to improve sales and marketing analytics. An example of this is the U.S. retailer Target. Charles Duhigg's book *The Power of Habit* [4] discusses how Target used Big Data and advanced analytical methods to drive new revenue. After analyzing consumer-purchasing behavior, Target's statisticians determined that the retailer made a great deal of money from three main life-event situations.

- Marriage, when people tend to buy many new products
- Divorce, when people buy new products and change their spending habits
- Pregnancy, when people have many new things to buy and have an urgency to buy them

Target determined that the most lucrative of these life-events is the third situation: pregnancy. Using data collected from shoppers, Target was able to identify this fact and predict which of its shoppers were pregnant. In one case, Target knew a female shopper was pregnant even before her family knew [5]. This kind of knowledge allowed Target to offer specific coupons and incentives to their pregnant shoppers. In fact, Target could not only determine if a shopper was pregnant, but in which month of pregnancy a shopper may be. This enabled Target to manage its inventory, knowing that there would be demand for specific products and it would likely vary by month over the coming nine- to ten-month cycles.

Hadoop [6] represents another example of Big Data innovation on the IT infrastructure. Apache Hadoop is an open source framework that allows companies to process vast amounts of information in a highly parallelized way. Hadoop represents a specific implementation of the MapReduce paradigm and was designed by Doug Cutting and Mike Cafarella in 2005 to use data with varying structures. It is an ideal technical framework for many Big Data projects, which rely on large or unwieldy datasets with unconventional data structures. One of the main benefits of Hadoop is that it employs a distributed file system, meaning it can use a distributed cluster of servers and commodity hardware to process large amounts of data. Some of the most common examples of Hadoop implementations are in the social media space, where Hadoop can manage transactions, give textual updates, and develop social graphs among millions of users. Twitter and Facebook generate massive amounts of unstructured data and use Hadoop and its ecosystem of tools to manage this high volume. Hadoop and its ecosystem are covered in Chapter 10, "Advanced Analytics—Technology and Tools: MapReduce and Hadoop."

Finally, social media represents a tremendous opportunity to leverage social and professional interactions to derive new insights. LinkedIn exemplifies a company in which data itself is the product. Early on, LinkedIn founder Reid Hoffman saw the opportunity to create a social network for working professionals.

As of 2014, LinkedIn has more than 250 million user accounts and has added many additional features and data-related products, such as recruiting, job seeker tools, advertising, and InMaps, which show a social graph of a user's professional network. Figure 1-14 is an example of an InMap visualization that enables a LinkedIn user to get a broader view of the interconnectedness of his contacts and understand how he knows most of them.

FIGURE 1-14 *Data visualization of a user's social network using InMaps*

Summary

Big Data comes from myriad sources, including social media, sensors, the Internet of Things, video surveillance, and many sources of data that may not have been considered data even a few years ago. As businesses struggle to keep up with changing market requirements, some companies are finding creative ways to apply Big Data to their growing business needs and increasingly complex problems. As organizations evolve their processes and see the opportunities that Big Data can provide, they try to move beyond traditional BI activities, such as using data to populate reports and dashboards, and move toward Data Science- driven projects that attempt to answer more open-ended and complex questions.

However, exploiting the opportunities that Big Data presents requires new data architectures, including analytic sandboxes, new ways of working, and people with new skill sets. These drivers are causing organizations to set up analytic sandboxes and build Data Science teams. Although some organizations are fortunate to have data scientists, most are not, because there is a growing talent gap that makes finding and hiring data scientists in a timely manner difficult. Still, organizations such as those in web retail, health care, genomics, new IT infrastructures, and social media are beginning to take advantage of Big Data and apply it in creative and novel ways.

Exercises

1. What are the three characteristics of Big Data, and what are the main considerations in processing Big Data?
2. What is an analytic sandbox, and why is it important?
3. Explain the differences between BI and Data Science.
4. Describe the challenges of the current analytical architecture for data scientists.
5. What are the key skill sets and behavioral characteristics of a data scientist?

Bibliography

[1] C. B. B. D. Manyika, "Big Data: The Next Frontier for Innovation, Competition, and Productivity," McKinsey Global Institute, 2011.

[2] D. R. John Gantz, "The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," IDC, 2013.

[3] `http://www.willisresilience.com/emc-datalab` [Online].

[4] C. Duhigg, *The Power of Habit: Why We Do What We Do in Life and Business*, New York: Random House, 2012.

[5] K. Hill, "How Target Figured Out a Teen Girl Was Pregnant Before Her Father Did," Forbes, February 2012.

[6] `http://hadoop.apache.org` [Online].

2

Data Analytics Lifecycle

Key Concepts

Discovery
Data preparation
Model planning
Model execution
Communicate results
Operationalize

Data science projects differ from most traditional Business Intelligence projects and many data analysis projects in that data science projects are more exploratory in nature. For this reason, it is critical to have a process to govern them and ensure that the participants are thorough and rigorous in their approach, yet not so rigid that the process impedes exploration.

Many problems that appear huge and daunting at first can be broken down into smaller pieces or actionable phases that can be more easily addressed. Having a good process ensures a comprehensive and repeatable method for conducting analysis. In addition, it helps focus time and energy early in the process to get a clear grasp of the business problem to be solved.

A common mistake made in data science projects is rushing into data collection and analysis, which precludes spending sufficient time to plan and scope the amount of work involved, understanding requirements, or even framing the business problem properly. Consequently, participants may discover mid-stream that the project sponsors are actually trying to achieve an objective that may not match the available data, or they are attempting to address an interest that differs from what has been explicitly communicated. When this happens, the project may need to revert to the initial phases of the process for a proper discovery phase, or the project may be canceled.

Creating and documenting a process helps demonstrate rigor, which provides additional credibility to the project when the data science team shares its findings. A well-defined process also offers a common framework for others to adopt, so the methods and analysis can be repeated in the future or as new members join a team.

2.1 Data Analytics Lifecycle Overview

The Data Analytics Lifecycle is designed specifically for Big Data problems and data science projects. The lifecycle has six phases, and project work can occur in several phases at once. For most phases in the lifecycle, the movement can be either forward or backward. This iterative depiction of the lifecycle is intended to more closely portray a real project, in which aspects of the project move forward and may return to earlier stages as new information is uncovered and team members learn more about various stages of the project. This enables participants to move iteratively through the process and drive toward operationalizing the project work.

2.1.1 Key Roles for a Successful Analytics Project

In recent years, substantial attention has been placed on the emerging role of the data scientist. In October 2012, Harvard Business Review featured an article titled "Data Scientist: The Sexiest Job of the 21st Century" [1], in which experts DJ Patil and Tom Davenport described the new role and how to find and hire data scientists. More and more conferences are held annually focusing on innovation in the areas of Data Science and topics dealing with Big Data. Despite this strong focus on the emerging role of the data scientist specifically, there are actually seven key roles that need to be fulfilled for a high-functioning data science team to execute analytic projects successfully.

Figure 2-1 depicts the various roles and key stakeholders of an analytics project. Each plays a critical part in a successful analytics project. Although seven roles are listed, fewer or more people can accomplish the work depending on the scope of the project, the organizational structure, and the skills of the participants. For example, on a small, versatile team, these seven roles may be fulfilled by only 3 people, but a very large project may require 20 or more people. The seven roles follow.

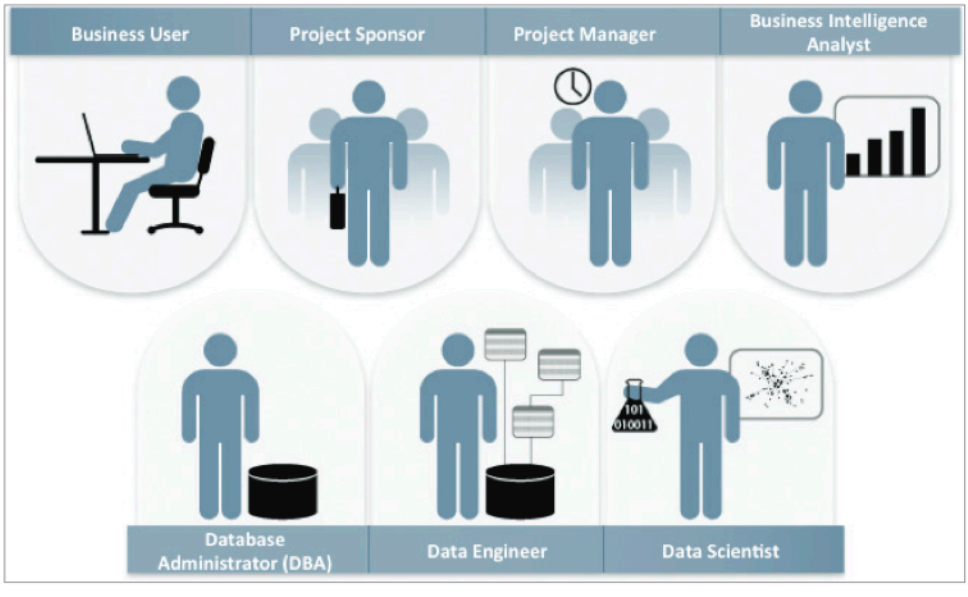


FIGURE 2-1 *Key roles for a successful analytics project*

- **Business User:** Someone who understands the domain area and usually benefits from the results. This person can consult and advise the project team on the context of the project, the value of the results, and how the outputs will be operationalized. Usually a business analyst, line manager, or deep subject matter expert in the project domain fulfills this role.
- **Project Sponsor:** Responsible for the genesis of the project. Provides the impetus and requirements for the project and defines the core business problem. Generally provides the funding and gauges the degree of value from the final outputs of the working team. This person sets the priorities for the project and clarifies the desired outputs.
- **Project Manager:** Ensures that key milestones and objectives are met on time and at the expected quality.
- **Business Intelligence Analyst:** Provides business domain expertise based on a deep understanding of the data, key performance indicators (KPIs), key metrics, and business intelligence from a reporting perspective. Business Intelligence Analysts generally create dashboards and reports and have knowledge of the data feeds and sources.
- **Database Administrator (DBA):** Provisions and configures the database environment to support the analytics needs of the working team. These responsibilities may include providing access to key databases or tables and ensuring the appropriate security levels are in place related to the data repositories.
- **Data Engineer:** Leverages deep technical skills to assist with tuning SQL queries for data management and data extraction, and provides support for data ingestion into the analytic sandbox, which

was discussed in Chapter 1, "Introduction to Big Data Analytics." Whereas the DBA sets up and configures the databases to be used, the data engineer executes the actual data extractions and performs substantial data manipulation to facilitate the analytics. The data engineer works closely with the data scientist to help shape data in the right ways for analyses.

- **Data Scientist:** Provides subject matter expertise for analytical techniques, data modeling, and applying valid analytical techniques to given business problems. Ensures overall analytics objectives are met. Designs and executes analytical methods and approaches with the data available to the project.

Although most of these roles are not new, the last two roles—data engineer and data scientist—have become popular and in high demand [2] as interest in Big Data has grown.

2.1.2 Background and Overview of Data Analytics Lifecycle

The Data Analytics Lifecycle defines analytics process best practices spanning discovery to project completion. The lifecycle draws from established methods in the realm of data analytics and decision science. This synthesis was developed after gathering input from data scientists and consulting established approaches that provided input on pieces of the process. Several of the processes that were consulted include these:

- **Scientific method** [3], in use for centuries, still provides a solid framework for thinking about and deconstructing problems into their principal parts. One of the most valuable ideas of the scientific method relates to forming hypotheses and finding ways to test ideas.
- **CRISP-DM** [4] provides useful input on ways to frame analytics problems and is a popular approach for data mining.
- Tom Davenport's **DELTA** framework [5]: The DELTA framework offers an approach for data analytics projects, including the context of the organization's skills, datasets, and leadership engagement.
- Doug Hubbard's **Applied Information Economics (AIE)** approach [6]: AIE provides a framework for measuring intangibles and provides guidance on developing decision models, calibrating expert estimates, and deriving the expected value of information.
- **"MAD Skills"** by Cohen et al. [7] offers input for several of the techniques mentioned in Phases 2–4 that focus on model planning, execution, and key findings.

Figure 2-2 presents an overview of the Data Analytics Lifecycle that includes six phases. Teams commonly learn new things in a phase that cause them to go back and refine the work done in prior phases based on new insights and information that have been uncovered. For this reason, Figure 2-2 is shown as a cycle. The circular arrows convey iterative movement between phases until the team members have sufficient information to move to the next phase. The callouts include sample questions to ask to help guide whether each of the team members has enough information and has made enough progress to move to the next phase of the process. Note that these phases do not represent formal stage gates; rather, they serve as criteria to help test whether it makes sense to stay in the current phase or move to the next.

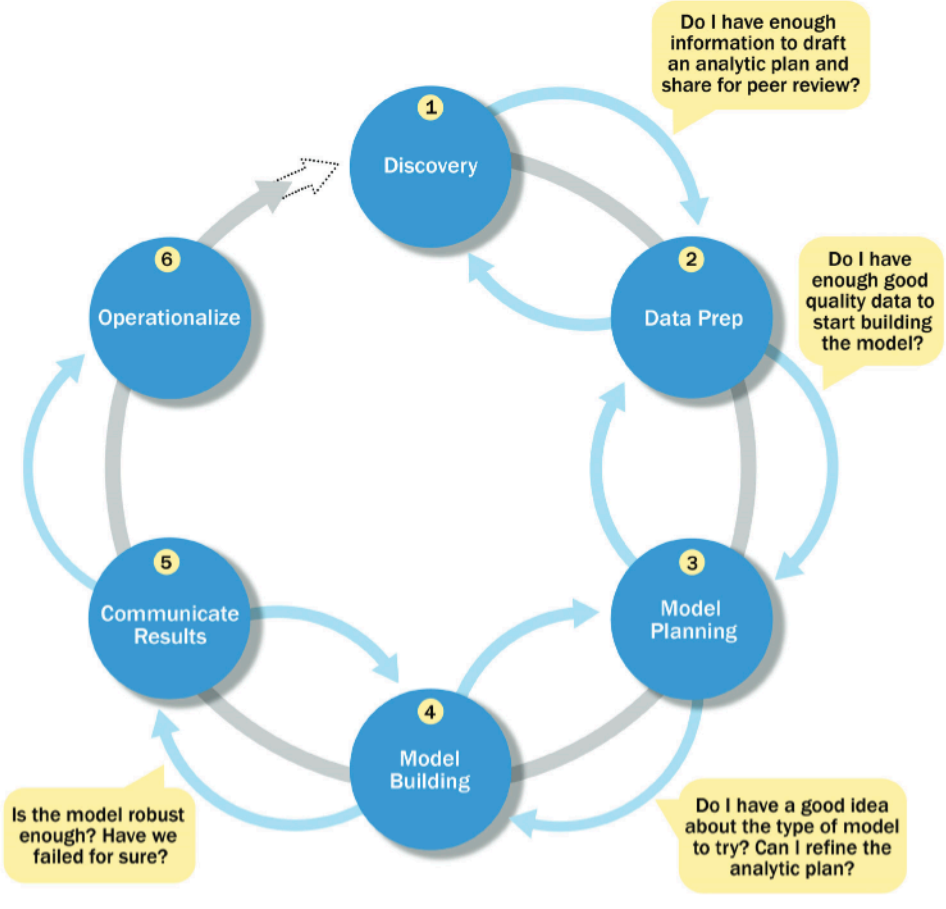


FIGURE 2-2 *Overview of Data Analytics Lifecycle*

Here is a brief overview of the main phases of the Data Analytics Lifecycle:

- **Phase 1—Discovery:** In Phase 1, the team learns the business domain, including relevant history such as whether the organization or business unit has attempted similar projects in the past from which they can learn. The team assesses the resources available to support the project in terms of people, technology, time, and data. Important activities in this phase include framing the business problem as an analytics challenge that can be addressed in subsequent phases and formulating initial hypotheses (IHs) to test and begin learning the data.
- **Phase 2—Data preparation:** Phase 2 requires the presence of an analytic sandbox, in which the team can work with data and perform analytics for the duration of the project. The team needs to execute extract, load, and transform (ELT) or extract, transform and load (ETL) to get data into the sandbox. The ELT and ETL are sometimes abbreviated as ETLT. Data should be transformed in the ETLT process so the team can work with it and analyze it. In this phase, the team also needs to familiarize itself with the data thoroughly and take steps to condition the data (Section 2.3.4).

- **Phase 3—Model planning:** Phase 3 is model planning, where the team determines the methods, techniques, and workflow it intends to follow for the subsequent model building phase. The team explores the data to learn about the relationships between variables and subsequently selects key variables and the most suitable models.
- **Phase 4—Model building:** In Phase 4, the team develops datasets for testing, training, and production purposes. In addition, in this phase the team builds and executes models based on the work done in the model planning phase. The team also considers whether its existing tools will suffice for running the models, or if it will need a more robust environment for executing models and workflows (for example, fast hardware and parallel processing, if applicable).
- **Phase 5—Communicate results:** In Phase 5, the team, in collaboration with major stakeholders, determines if the results of the project are a success or a failure based on the criteria developed in Phase 1. The team should identify key findings, quantify the business value, and develop a narrative to summarize and convey findings to stakeholders.
- **Phase 6—Operationalize:** In Phase 6, the team delivers final reports, briefings, code, and technical documents. In addition, the team may run a pilot project to implement the models in a production environment.

Once team members have run models and produced findings, it is critical to frame these results in a way that is tailored to the audience that engaged the team. Moreover, it is critical to frame the results of the work in a manner that demonstrates clear value. If the team performs a technically accurate analysis but fails to translate the results into a language that resonates with the audience, people will not see the value, and much of the time and effort on the project will have been wasted.

The rest of the chapter is organized as follows. Sections 2.2–2.7 discuss in detail how each of the six phases works, and Section 2.8 shows a case study of incorporating the Data Analytics Lifecycle in a real-world data science project.

2.2 Phase 1: Discovery

The first phase of the Data Analytics Lifecycle involves discovery (Figure 2-3). In this phase, the data science team must learn and investigate the problem, develop context and understanding, and learn about the data sources needed and available for the project. In addition, the team formulates initial hypotheses that can later be tested with data.

2.2.1 Learning the Business Domain

Understanding the domain area of the problem is essential. In many cases, data scientists will have deep computational and quantitative knowledge that can be broadly applied across many disciplines. An example of this role would be someone with an advanced degree in applied mathematics or statistics.

These data scientists have deep knowledge of the methods, techniques, and ways for applying heuristics to a variety of business and conceptual problems. Others in this area may have deep knowledge of a domain area, coupled with quantitative expertise. An example of this would be someone with a Ph.D. in life sciences. This person would have deep knowledge of a field of study, such as oceanography, biology, or genetics, with some depth of quantitative knowledge.

At this early stage in the process, the team needs to determine how much business or domain knowledge the data scientist needs to develop models in Phases 3 and 4. The earlier the team can make this assessment

the better, because the decision helps dictate the resources needed for the project team and ensures the team has the right balance of domain knowledge and technical expertise.

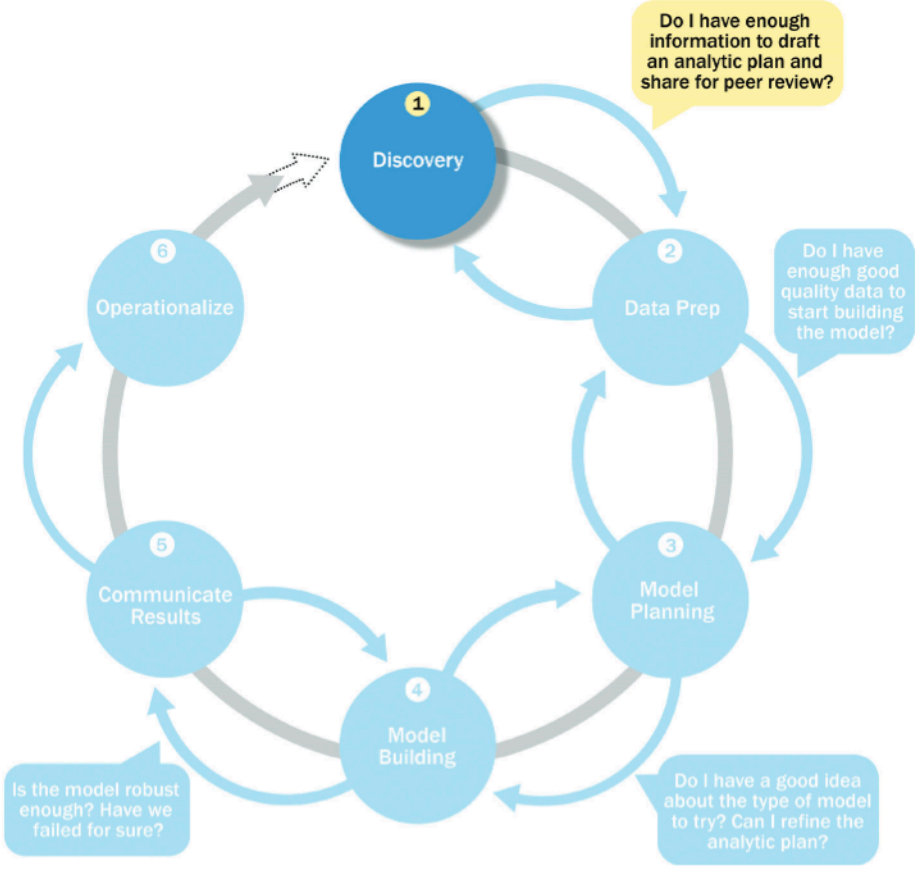


FIGURE 2-3 *Discovery phase*

2.2.2 Resources

As part of the discovery phase, the team needs to assess the resources available to support the project. In this context, resources include technology, tools, systems, data, and people.

During this scoping, consider the available tools and technology the team will be using and the types of systems needed for later phases to operationalize the models. In addition, try to evaluate the level of analytical sophistication within the organization and gaps that may exist related to tools, technology, and skills. For instance, for the model being developed to have longevity in an organization, consider what types of skills and roles will be required that may not exist today. For the project to have long-term success,

what types of skills and roles will be needed for the recipients of the model being developed? Does the requisite level of expertise exist within the organization today, or will it need to be cultivated? Answering these questions will influence the techniques the team selects and the kind of implementation the team chooses to pursue in subsequent phases of the Data Analytics Lifecycle.

In addition to the skills and computing resources, it is advisable to take inventory of the types of data available to the team for the project. Consider if the data available is sufficient to support the project's goals. The team will need to determine whether it must collect additional data, purchase it from outside sources, or transform existing data. Often, projects are started looking only at the data available. When the data is less than hoped for, the size and scope of the project is reduced to work within the constraints of the existing data.

An alternative approach is to consider the long-term goals of this kind of project, without being constrained by the current data. The team can then consider what data is needed to reach the long-term goals and which pieces of this multistep journey can be achieved today with the existing data. Considering longer-term goals along with short-term goals enables teams to pursue more ambitious projects and treat a project as the first step of a more strategic initiative, rather than as a standalone initiative. It is critical to view projects as part of a longer-term journey, especially if executing projects in an organization that is new to Data Science and may not have embarked on the optimum datasets to support robust analyses up to this point.

Ensure the project team has the right mix of domain experts, customers, analytic talent, and project management to be effective. In addition, evaluate how much time is needed and if the team has the right breadth and depth of skills.

After taking inventory of the tools, technology, data, and people, consider if the team has sufficient resources to succeed on this project, or if additional resources are needed. Negotiating for resources at the outset of the project, while scoping the goals, objectives, and feasibility, is generally more useful than later in the process and ensures sufficient time to execute it properly. Project managers and key stakeholders have better success negotiating for the right resources at this stage rather than later once the project is underway.

2.2.3 Framing the Problem

Framing the problem well is critical to the success of the project. ***Framing*** is the process of stating the analytics problem to be solved. At this point, it is a best practice to write down the problem statement and share it with the key stakeholders. Each team member may hear slightly different things related to the needs and the problem and have somewhat different ideas of possible solutions. For these reasons, it is crucial to state the analytics problem, as well as why and to whom it is important. Essentially, the team needs to clearly articulate the current situation and its main challenges.

As part of this activity, it is important to identify the main objectives of the project, identify what needs to be achieved in business terms, and identify what needs to be done to meet the needs. Additionally, consider the objectives and the success criteria for the project. What is the team attempting to achieve by doing the project, and what will be considered "good enough" as an outcome of the project? This is critical to document and share with the project team and key stakeholders. It is best practice to share the statement of goals and success criteria with the team and confirm alignment with the project sponsor's expectations.

Perhaps equally important is to establish failure criteria. Most people doing projects prefer only to think of the success criteria and what the conditions will look like when the participants are successful. However, this is almost taking a best-case scenario approach, assuming that everything will proceed as planned

and the project team will reach its goals. However, no matter how well planned, it is almost impossible to plan for everything that will emerge in a project. The failure criteria will guide the team in understanding when it is best to stop trying or settle for the results that have been gleaned from the data. Many times people will continue to perform analyses past the point when any meaningful insights can be drawn from the data. Establishing criteria for both success and failure helps the participants avoid unproductive effort and remain aligned with the project sponsors

2.2.4 Identifying Key Stakeholders

Another important step is to identify the key stakeholders and their interests in the project. During these discussions, the team can identify the success criteria, key risks, and stakeholders, which should include anyone who will benefit from the project or will be significantly impacted by the project. When interviewing stakeholders, learn about the domain area and any relevant history from similar analytics projects. For example, the team may identify the results each stakeholder wants from the project and the criteria it will use to judge the success of the project.

Keep in mind that the analytics project is being initiated for a reason. It is critical to articulate the pain points as clearly as possible to address them and be aware of areas to pursue or avoid as the team gets further into the analytical process. Depending on the number of stakeholders and participants, the team may consider outlining the type of activity and participation expected from each stakeholder and participant. This will set clear expectations with the participants and avoid delays later when, for example, the team may feel it needs to wait for approval from someone who views himself as an adviser rather than an approver of the work product.

2.2.5 Interviewing the Analytics Sponsor

The team should plan to collaborate with the stakeholders to clarify and frame the analytics problem. At the outset, project sponsors may have a predetermined solution that may not necessarily realize the desired outcome. In these cases, the team must use its knowledge and expertise to identify the true underlying problem and appropriate solution.

For instance, suppose in the early phase of a project, the team is told to create a recommender system for the business and that the way to do this is by speaking with three people and integrating the product recommender into a legacy corporate system. Although this may be a valid approach, it is important to test the assumptions and develop a clear understanding of the problem. The data science team typically may have a more objective understanding of the problem set than the stakeholders, who may be suggesting solutions to a given problem. Therefore, the team can probe deeper into the context and domain to clearly define the problem and propose possible paths from the problem to a desired outcome. In essence, the data science team can take a more objective approach, as the stakeholders may have developed biases over time, based on their experience. Also, what may have been true in the past may no longer be a valid working assumption. One possible way to circumvent this issue is for the project sponsor to focus on clearly defining the requirements, while the other members of the data science team focus on the methods needed to achieve the goals.

When interviewing the main stakeholders, the team needs to take time to thoroughly interview the project sponsor, who tends to be the one funding the project or providing the high-level requirements. This person understands the problem and usually has an idea of a potential working solution. It is critical

to thoroughly understand the sponsor's perspective to guide the team in getting started on the project. Here are some tips for interviewing project sponsors:

- Prepare for the interview; draft questions, and review with colleagues.
- Use open-ended questions; avoid asking leading questions.
- Probe for details and pose follow-up questions.
- Avoid filling every silence in the conversation; give the other person time to think.
- Let the sponsors express their ideas and ask clarifying questions, such as "Why? Is that correct? Is this idea on target? Is there anything else?"
- Use active listening techniques; repeat back what was heard to make sure the team heard it correctly, or reframe what was said.
- Try to avoid expressing the team's opinions, which can introduce bias; instead, focus on listening.
- Be mindful of the body language of the interviewers and stakeholders; use eye contact where appropriate, and be attentive.
- Minimize distractions.
- Document what the team heard, and review it with the sponsors.

Following is a brief list of common questions that are helpful to ask during the discovery phase when interviewing the project sponsor. The responses will begin to shape the scope of the project and give the team an idea of the goals and objectives of the project.

- What business problem is the team trying to solve?
- What is the desired outcome of the project?
- What data sources are available?
- What industry issues may impact the analysis?
- What timelines need to be considered?
- Who could provide insight into the project?
- Who has final decision-making authority on the project?
- How will the focus and scope of the problem change if the following dimensions change:
    - **Time:** Analyzing 1 year or 10 years' worth of data?
    - **People:** Assess impact of changes in resources on project timeline.
    - **Risk:** Conservative to aggressive
    - **Resources:** None to unlimited (tools, technology, systems)
    - **Size and attributes of data:** Including internal and external data sources

2.2.6 Developing Initial Hypotheses

Developing a set of IHs is a key facet of the discovery phase. This step involves forming ideas that the team can test with data. Generally, it is best to come up with a few primary hypotheses to test and then be creative about developing several more. These IHs form the basis of the analytical tests the team will use in later phases and serve as the foundation for the findings in Phase 5. Hypothesis testing from a statistical perspective is covered in greater detail in Chapter 3, "Review of Basic Data Analytic Methods Using R."

In this way, the team can compare its answers with the outcome of an experiment or test to generate additional possible solutions to problems. As a result, the team will have a much richer set of observations to choose from and more choices for agreeing upon the most impactful conclusions from a project.

Another part of this process involves gathering and assessing hypotheses from stakeholders and domain experts who may have their own perspective on what the problem is, what the solution should be, and how to arrive at a solution. These stakeholders would know the domain area well and can offer suggestions on ideas to test as the team formulates hypotheses during this phase. The team will likely collect many ideas that may illuminate the operating assumptions of the stakeholders. These ideas will also give the team opportunities to expand the project scope into adjacent spaces where it makes sense or design experiments in a meaningful way to address the most important interests of the stakeholders. As part of this exercise, it can be useful to obtain and explore some initial data to inform discussions with stakeholders during the hypothesis-forming stage.

2.2.7 Identifying Potential Data Sources

As part of the discovery phase, identify the kinds of data the team will need to solve the problem. Consider the volume, type, and time span of the data needed to test the hypotheses. Ensure that the team can access more than simply aggregated data. In most cases, the team will need the raw data to avoid introducing bias for the downstream analysis. Recalling the characteristics of Big Data from Chapter 1, assess the main characteristics of the data, with regard to its volume, variety, and velocity of change. A thorough diagnosis of the data situation will influence the kinds of tools and techniques to use in Phases 2-4 of the Data Analytics Lifecycle. In addition, performing data exploration in this phase will help the team determine the amount of data needed, such as the amount of historical data to pull from existing systems and the data structure. Develop an idea of the scope of the data needed, and validate that idea with the domain experts on the project.

The team should perform five main activities during this step of the discovery phase:

- **Identify data sources:** Make a list of candidate data sources the team may need to test the initial hypotheses outlined in this phase. Make an inventory of the datasets currently available and those that can be purchased or otherwise acquired for the tests the team wants to perform.
- **Capture aggregate data sources:** This is for previewing the data and providing high-level understanding. It enables the team to gain a quick overview of the data and perform further exploration on specific areas. It also points the team to possible areas of interest within the data.
- **Review the raw data:** Obtain preliminary data from initial data feeds. Begin understanding the interdependencies among the data attributes, and become familiar with the content of the data, its quality, and its limitations.

- **Evaluate the data structures and tools needed:** The data type and structure dictate which tools the team can use to analyze the data. This evaluation gets the team thinking about which technologies may be good candidates for the project and how to start getting access to these tools.
- **Scope the sort of data infrastructure needed for this type of problem:** In addition to the tools needed, the data influences the kind of infrastructure that's required, such as disk storage and network capacity.

Unlike many traditional stage-gate processes, in which the team can advance only when specific criteria are met, the Data Analytics Lifecycle is intended to accommodate more ambiguity. This more closely reflects how data science projects work in real-life situations. For each phase of the process, it is recommended to pass certain checkpoints as a way of gauging whether the team is ready to move to the next phase of the Data Analytics Lifecycle.

The team can move to the next phase when it has enough information to draft an analytics plan and share it for peer review. Although a peer review of the plan may not actually be required by the project, creating the plan is a good test of the team's grasp of the business problem and the team's approach to addressing it. Creating the analytic plan also requires a clear understanding of the domain area, the problem to be solved, and scoping of the data sources to be used. Developing success criteria early in the project clarifies the problem definition and helps the team when it comes time to make choices about the analytical methods being used in later phases.

2.3 Phase 2: Data Preparation

The second phase of the Data Analytics Lifecycle involves data preparation, which includes the steps to explore, preprocess, and condition data prior to modeling and analysis. In this phase, the team needs to create a robust environment in which it can explore the data that is separate from a production environment. Usually, this is done by preparing an analytics sandbox. To get the data into the sandbox, the team needs to perform ETLT, by a combination of extracting, transforming, and loading data into the sandbox. Once the data is in the sandbox, the team needs to learn about the data and become familiar with it. Understanding the data in detail is critical to the success of the project. The team also must decide how to condition and transform data to get it into a format to facilitate subsequent analysis. The team may perform data visualizations to help team members understand the data, including its trends, outliers, and relationships among data variables. Each of these steps of the data preparation phase is discussed throughout this section.

Data preparation tends to be the most labor-intensive step in the analytics lifecycle. In fact, it is common for teams to spend at least 50% of a data science project's time in this critical phase. If the team cannot obtain enough data of sufficient quality, it may be unable to perform the subsequent steps in the lifecycle process.

Figure 2-4 shows an overview of the Data Analytics Lifecycle for Phase 2. The data preparation phase is generally the most iterative and the one that teams tend to underestimate most often. This is because most teams and leaders are anxious to begin analyzing the data, testing hypotheses, and getting answers to some of the questions posed in Phase 1. Many tend to jump into Phase 3 or Phase 4 to begin rapidly developing models and algorithms without spending the time to prepare the data for modeling. Consequently, teams come to realize the data they are working with does not allow them to execute the models they want, and they end up back in Phase 2 anyway.

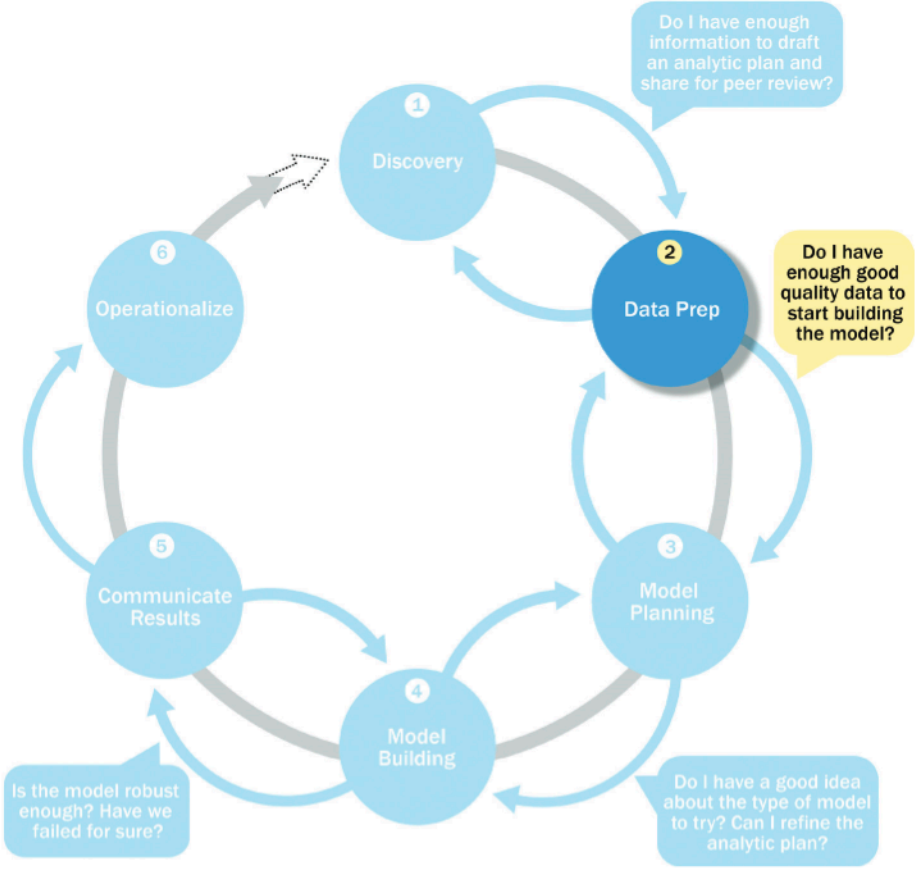


FIGURE 2-4 *Data preparation phase*

2.3.1 Preparing the Analytic Sandbox

The first subphase of data preparation requires the team to obtain an analytic sandbox (also commonly referred to as a ***workspace***), in which the team can explore the data without interfering with live production databases. Consider an example in which the team needs to work with a company's financial data. The team should access a copy of the financial data from the analytic sandbox rather than interacting with the production version of the organization's main database, because that will be tightly controlled and needed for financial reporting.

When developing the analytic sandbox, it is a best practice to collect all kinds of data there, as team members need access to high volumes and varieties of data for a Big Data analytics project. This can include

everything from summary-level aggregated data, structured data, raw data feeds, and unstructured text data from call logs or web logs, depending on the kind of analysis the team plans to undertake.

This expansive approach for attracting data of all kind differs considerably from the approach advocated by many information technology (IT) organizations. Many IT groups provide access to only a particular subsegment of the data for a specific purpose. Often, the mindset of the IT group is to provide the minimum amount of data required to allow the team to achieve its objectives. Conversely, the data science team wants access to everything. From its perspective, more data is better, as oftentimes data science projects are a mixture of purpose-driven analyses and experimental approaches to test a variety of ideas. In this context, it can be challenging for a data science team if it has to request access to each and every dataset and attribute one at a time. Because of these differing views on data access and use, it is critical for the data science team to collaborate with IT, make clear what it is trying to accomplish, and align goals.

During these discussions, the data science team needs to give IT a justification to develop an analytics sandbox, which is separate from the traditional IT-governed data warehouses within an organization. Successfully and amicably balancing the needs of both the data science team and IT requires a positive working relationship between multiple groups and data owners. The payoff is great. The analytic sandbox enables organizations to undertake more ambitious data science projects and move beyond doing traditional data analysis and Business Intelligence to perform more robust and advanced predictive analytics.

Expect the sandbox to be large. It may contain raw data, aggregated data, and other data types that are less commonly used in organizations. Sandbox size can vary greatly depending on the project. A good rule is to plan for the sandbox to be at least 5–10 times the size of the original datasets, partly because copies of the data may be created that serve as specific tables or data stores for specific kinds of analysis in the project.

Although the concept of an analytics sandbox is relatively new, companies are making progress in this area and are finding ways to offer sandboxes and workspaces where teams can access datasets and work in a way that is acceptable to both the data science teams and the IT groups.

2.3.2 Performing ETLT

As the team looks to begin data transformations, make sure the analytics sandbox has ample bandwidth and reliable network connections to the underlying data sources to enable uninterrupted read and write. In ETL, users perform extract, transform, load processes to extract data from a datastore, perform data transformations, and load the data back into the datastore. However, the analytic sandbox approach differs slightly; it advocates extract, load, and then transform. In this case, the data is extracted in its raw form and loaded into the datastore, where analysts can choose to transform the data into a new state or leave it in its original, raw condition. The reason for this approach is that there is significant value in preserving the raw data and including it in the sandbox before any transformations take place.

For instance, consider an analysis for fraud detection on credit card usage. Many times, outliers in this data population can represent higher-risk transactions that may be indicative of fraudulent credit card activity. Using ETL, these outliers may be inadvertently filtered out or transformed and cleaned before being loaded into the datastore. In this case, the very data that would be needed to evaluate instances of fraudulent activity would be inadvertently cleansed, preventing the kind of analysis that a team would want to do.

Following the ELT approach gives the team access to clean data to analyze after the data has been loaded into the database and gives access to the data in its original form for finding hidden nuances in the data. This approach is part of the reason that the analytic sandbox can quickly grow large. The team may want clean data and aggregated data and may need to keep a copy of the original data to compare against or

look for hidden patterns that may have existed in the data before the cleaning stage. This process can be summarized as ETLT to reflect the fact that a team may choose to perform ETL in one case and ELT in another.

Depending on the size and number of the data sources, the team may need to consider how to parallelize the movement of the datasets into the sandbox. For this purpose, moving large amounts of data is sometimes referred to as Big ETL. The data movement can be parallelized by technologies such as Hadoop or MapReduce, which will be explained in greater detail in Chapter 10, "Advanced Analytics—Technology and Tools: MapReduce and Hadoop." At this point, keep in mind that these technologies can be used to perform parallel data ingest and introduce a huge number of files or datasets in parallel in a very short period of time. Hadoop can be useful for data loading as well as for data analysis in subsequent phases.

Prior to moving the data into the analytic sandbox, determine the transformations that need to be performed on the data. Part of this phase involves assessing data quality and structuring the datasets properly so they can be used for robust analysis in subsequent phases. In addition, it is important to consider which data the team will have access to and which new data attributes will need to be derived in the data to enable analysis.

As part of the ETLT step, it is advisable to make an inventory of the data and compare the data currently available with datasets the team needs. Performing this sort of gap analysis provides a framework for understanding which datasets the team can take advantage of today and where the team needs to initiate projects for data collection or access to new datasets currently unavailable. A component of this subphase involves extracting data from the available sources and determining data connections for raw data, online transaction processing (OLTP) databases, online analytical processing (OLAP) cubes, or other data feeds.

Application programming interface (API) is an increasingly popular way to access a data source [8]. Many websites and social network applications now provide APIs that offer access to data to support a project or supplement the datasets with which a team is working. For example, connecting to the Twitter API can enable a team to download millions of tweets to perform a project for sentiment analysis on a product, a company, or an idea. Much of the Twitter data is publicly available and can augment other datasets used on the project.

2.3.3 Learning About the Data

A critical aspect of a data science project is to become familiar with the data itself. Spending time to learn the nuances of the datasets provides context to understand what constitutes a reasonable value and expected output versus what is a surprising finding. In addition, it is important to catalog the data sources that the team has access to and identify additional data sources that the team can leverage but perhaps does not have access to today. Some of the activities in this step may overlap with the initial investigation of the datasets that occur in the discovery phase. Doing this activity accomplishes several goals.

- Clarifies the data that the data science team has access to at the start of the project
- Highlights gaps by identifying datasets within an organization that the team may find useful but may not be accessible to the team today. As a consequence, this activity can trigger a project to begin building relationships with the data owners and finding ways to share data in appropriate ways. In addition, this activity may provide an impetus to begin collecting new data that benefits the organization or a specific long-term project.
- Identifies datasets outside the organization that may be useful to obtain, through open APIs, data sharing, or purchasing data to supplement already existing datasets

Table 2-1 demonstrates one way to organize this type of data inventory.

TABLE 2-1 *Sample Dataset Inventory*

Dataset	Data Available and Accessible	Data Available, but not Accessible	Data to Collect	Data to Obtain from Third Party Sources
Products shipped	●			
Product Financials		●		
Product Call Center Data		●		
Live Product Feedback Surveys			●	
Product Sentiment from Social Media				●

2.3.4 Data Conditioning

Data conditioning refers to the process of cleaning data, normalizing datasets, and performing transformations on the data. A critical step within the Data Analytics Lifecycle, data conditioning can involve many complex steps to join or merge datasets or otherwise get datasets into a state that enables analysis in further phases. Data conditioning is often viewed as a preprocessing step for the data analysis because it involves many operations on the dataset before developing models to process or analyze the data. This implies that the data-conditioning step is performed only by IT, the data owners, a DBA, or a data engineer. However, it is also important to involve the data scientist in this step because many decisions are made in the data conditioning phase that affect subsequent analysis. Part of this phase involves deciding which aspects of particular datasets will be useful to analyze in later steps. Because teams begin forming ideas in this phase about which data to keep and which data to transform or discard, it is important to involve multiple team members in these decisions. Leaving such decisions to a single person may cause teams to return to this phase to retrieve data that may have been discarded.

As with the previous example of deciding which data to keep as it relates to fraud detection on credit card usage, it is critical to be thoughtful about which data the team chooses to keep and which data will be discarded. This can have far-reaching consequences that will cause the team to retrace previous steps if the team discards too much of the data at too early a point in this process. Typically, data science teams would rather keep more data than too little data for the analysis. Additional questions and considerations for the data conditioning step include these.

- What are the data sources? What are the target fields (for example, columns of the tables)?
- How clean is the data?

- How consistent are the contents and files? Determine to what degree the data contains missing or inconsistent values and if the data contains values deviating from normal.
- Assess the consistency of the data types. For instance, if the team expects certain data to be numeric, confirm it is numeric or if it is a mixture of alphanumeric strings and text.
- Review the content of data columns or other inputs, and check to ensure they make sense. For instance, if the project involves analyzing income levels, preview the data to confirm that the income values are positive or if it is acceptable to have zeros or negative values.
- Look for any evidence of systematic error. Examples include data feeds from sensors or other data sources breaking without anyone noticing, which causes invalid, incorrect, or missing data values. In addition, review the data to gauge if the definition of the data is the same over all measurements. In some cases, a data column is repurposed, or the column stops being populated, without this change being annotated or without others being notified.

2.3.5 Survey and Visualize

After the team has collected and obtained at least some of the datasets needed for the subsequent analysis, a useful step is to leverage data visualization tools to gain an overview of the data. Seeing high-level patterns in the data enables one to understand characteristics about the data very quickly. One example is using data visualization to examine data quality, such as whether the data contains many unexpected values or other indicators of dirty data. (Dirty data will be discussed further in Chapter 3.) Another example is skewness, such as if the majority of the data is heavily shifted toward one value or end of a continuum.

Shneiderman [9] is well known for his mantra for visual data analysis of "overview first, zoom and filter, then details-on-demand." This is a pragmatic approach to visual data analysis. It enables the user to find areas of interest, zoom and filter to find more detailed information about a particular area of the data, and then find the detailed data behind a particular area. This approach provides a high-level view of the data and a great deal of information about a given dataset in a relatively short period of time.

When pursuing this approach with a data visualization tool or statistical package, the following guidelines and considerations are recommended.

- Review data to ensure that calculations remained consistent within columns or across tables for a given data field. For instance, did customer lifetime value change at some point in the middle of data collection? Or if working with financials, did the interest calculation change from simple to compound at the end of the year?
- Does the data distribution stay consistent over all the data? If not, what kinds of actions should be taken to address this problem?
- Assess the granularity of the data, the range of values, and the level of aggregation of the data.
- Does the data represent the population of interest? For marketing data, if the project is focused on targeting customers of child-rearing age, does the data represent that, or is it full of senior citizens and teenagers?
- For time-related variables, are the measurements daily, weekly, monthly? Is that good enough? Is time measured in seconds everywhere? Or is it in milliseconds in some places? Determine the level of granularity of the data needed for the analysis, and assess whether the current level of timestamps on the data meets that need.

- Is the data standardized/normalized? Are the scales consistent? If not, how consistent or irregular is the data?
- For geospatial datasets, are state or country abbreviations consistent across the data? Are personal names normalized? English units? Metric units?

These are typical considerations that should be part of the thought process as the team evaluates the datasets that are obtained for the project. Becoming deeply knowledgeable about the data will be critical when it comes time to construct and run models later in the process.

2.3.6 Common Tools for the Data Preparation Phase

Several tools are commonly used for this phase:

- **Hadoop** [10] can perform massively parallel ingest and custom analysis for web traffic parsing, GPS location analytics, genomic analysis, and combining of massive unstructured data feeds from multiple sources.
- **Alpine Miner** [11] provides a graphical user interface (GUI) for creating analytic workflows, including data manipulations and a series of analytic events such as staged data-mining techniques (for example, first select the top 100 customers, and then run descriptive statistics and clustering) on Postgres SQL and other Big Data sources.
- **OpenRefine** (formerly called Google Refine) [12] is "a free, open source, powerful tool for working with messy data." It is a popular GUI-based tool for performing data transformations, and it's one of the most robust free tools currently available.
- Similar to OpenRefine, **Data Wrangler** [13] is an interactive tool for data cleaning and transformation. Wrangler was developed at Stanford University and can be used to perform many transformations on a given dataset. In addition, data transformation outputs can be put into Java or Python. The advantage of this feature is that a subset of the data can be manipulated in Wrangler via its GUI, and then the same operations can be written out as Java or Python code to be executed against the full, larger dataset offline in a local analytic sandbox.

For Phase 2, the team needs assistance from IT, DBAs, or whoever controls the Enterprise Data Warehouse (EDW) for data sources the data science team would like to use.

2.4 Phase 3: Model Planning

In Phase 3, the data science team identifies candidate models to apply to the data for clustering, classifying, or finding relationships in the data depending on the goal of the project, as shown in Figure 2-5. It is during this phase that the team refers to the hypotheses developed in Phase 1, when they first became acquainted with the data and understanding the business problems or domain area. These hypotheses help the team frame the analytics to execute in Phase 4 and select the right methods to achieve its objectives.

Some of the activities to consider in this phase include the following:

- Assess the structure of the datasets. The structure of the datasets is one factor that dictates the tools and analytical techniques for the next phase. Depending on whether the team plans to analyze textual data or transactional data, for example, different tools and approaches are required.
- Ensure that the analytical techniques enable the team to meet the business objectives and accept or reject the working hypotheses.

- Determine if the situation warrants a single model or a series of techniques as part of a larger analytic workflow. A few example models include association rules (Chapter 5, "Advanced Analytical Theory and Methods: Association Rules") and logistic regression (Chapter 6, "Advanced Analytical Theory and Methods: Regression"). Other tools, such as Alpine Miner, enable users to set up a series of steps and analyses and can serve as a front-end user interface (UI) for manipulating Big Data sources in PostgreSQL.

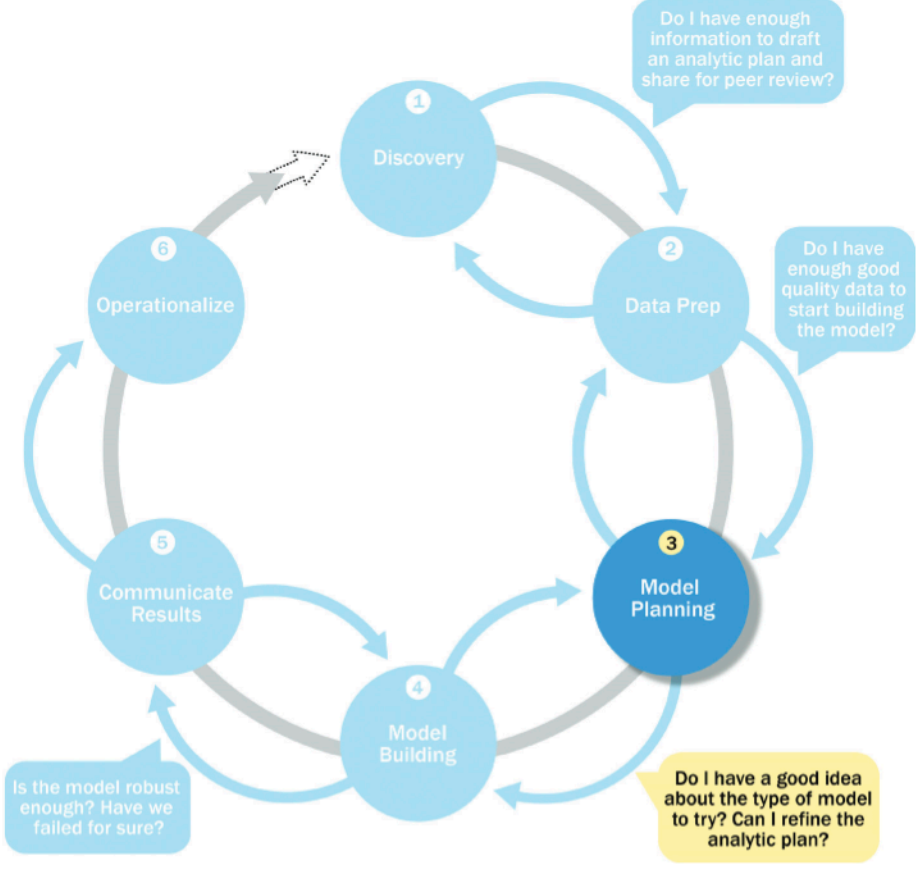


FIGURE 2-5 *Model planning phase*

In addition to the considerations just listed, it is useful to research and understand how other analysts generally approach a specific kind of problem. Given the kind of data and resources that are available, evaluate whether similar, existing approaches will work or if the team will need to create something new. Many times teams can get ideas from analogous problems that other people have solved in different industry verticals or domain areas. Table 2-2 summarizes the results of an exercise of this type, involving several domain areas and the types of models previously used in a classification type of problem after conducting research on churn models in multiple industry verticals. Performing this sort of diligence gives the team

ideas of how others have solved similar problems and presents the team with a list of candidate models to try as part of the model planning phase.

TABLE 2-2 *Research on Model Planning in Industry Verticals*

Market Sector	Analytic Techniques/Methods Used
Consumer Packaged Goods	Multiple linear regression, automatic relevance determination (ARD), and decision tree
Retail Banking	Multiple regression
Retail Business	Logistic regression, ARD, decision tree
Wireless Telecom	Neural network, decision tree, hierarchical neurofuzzy systems, rule evolver, logistic regression

2.4.1 Data Exploration and Variable Selection

Although some data exploration takes place in the data preparation phase, those activities focus mainly on data hygiene and on assessing the quality of the data itself. In Phase 3, the objective of the data exploration is to understand the relationships among the variables to inform selection of the variables and methods and to understand the problem domain. As with earlier phases of the Data Analytics Lifecycle, it is important to spend time and focus attention on this preparatory work to make the subsequent phases of model selection and execution easier and more efficient. A common way to conduct this step involves using tools to perform data visualizations. Approaching the data exploration in this way aids the team in previewing the data and assessing relationships between variables at a high level.

In many cases, stakeholders and subject matter experts have instincts and hunches about what the data science team should be considering and analyzing. Likely, this group had some hypothesis that led to the genesis of the project. Often, stakeholders have a good grasp of the problem and domain, although they may not be aware of the subtleties within the data or the model needed to accept or reject a hypothesis. Other times, stakeholders may be correct, but for the wrong reasons (for instance, they may be correct about a correlation that exists but infer an incorrect reason for the correlation). Meanwhile, data scientists have to approach problems with an unbiased mind-set and be ready to question all assumptions.

As the team begins to question the incoming assumptions and test initial ideas of the project sponsors and stakeholders, it needs to consider the inputs and data that will be needed, and then it must examine whether these inputs are actually correlated with the outcomes that the team plans to predict or analyze. Some methods and types of models will handle correlated variables better than others. Depending on what the team is attempting to solve, it may need to consider an alternate method, reduce the number of data inputs, or transform the inputs to allow the team to use the best method for a given business problem. Some of these techniques will be explored further in Chapter 3 and Chapter 6.

The key to this approach is to aim for capturing the most essential predictors and variables rather than considering every possible variable that people think may influence the outcome. Approaching the problem in this manner requires iterations and testing to identify the most essential variables for the intended analyses. The team should plan to test a range of variables to include in the model and then focus on the most important and influential variables.

If the team plans to run regression analyses, identify the candidate predictors and outcome variables of the model. Plan to create variables that determine outcomes but demonstrate a strong relationship to the outcome rather than to the other input variables. This includes remaining vigilant for problems such as serial correlation, multicollinearity, and other typical data modeling challenges that interfere with the validity of these models. Sometimes these issues can be avoided simply by looking at ways to reframe a given problem. In addition, sometimes determining correlation is all that is needed ("black box prediction"), and in other cases, the objective of the project is to understand the causal relationship better. In the latter case, the team wants the model to have explanatory power and needs to forecast or stress test the model under a variety of situations and with different datasets.

2.4.2 Model Selection

In the model selection subphase, the team's main goal is to choose an analytical technique, or a short list of candidate techniques, based on the end goal of the project. For the context of this book, a *model* is discussed in general terms. In this case, a model simply refers to an abstraction from reality. One observes events happening in a real-world situation or with live data and attempts to construct models that emulate this behavior with a set of rules and conditions. In the case of machine learning and data mining, these rules and conditions are grouped into several general sets of techniques, such as classification, association rules, and clustering. When reviewing this list of types of potential models, the team can winnow down the list to several viable models to try to address a given problem. More details on matching the right models to common types of business problems are provided in Chapter 3 and Chapter 4, "Advanced Analytical Theory and Methods: Clustering."

An additional consideration in this area for dealing with Big Data involves determining if the team will be using techniques that are best suited for structured data, unstructured data, or a hybrid approach. For instance, the team can leverage MapReduce to analyze unstructured data, as highlighted in Chapter 10. Lastly, the team should take care to identify and document the modeling assumptions it is making as it chooses and constructs preliminary models.

Typically, teams create the initial models using a statistical software package such as R, SAS, or Matlab. Although these tools are designed for data mining and machine learning algorithms, they may have limitations when applying the models to very large datasets, as is common with Big Data. As such, the team may consider redesigning these algorithms to run in the database itself during the pilot phase mentioned in Phase 6.

The team can move to the model building phase once it has a good idea about the type of model to try and the team has gained enough knowledge to refine the analytics plan. Advancing from this phase requires a general methodology for the analytical model, a solid understanding of the variables and techniques to use, and a description or diagram of the analytic workflow.

2.4.3 Common Tools for the Model Planning Phase

Many tools are available to assist in this phase. Here are several of the more common ones:

- **R** [14] has a complete set of modeling capabilities and provides a good environment for building interpretive models with high-quality code. In addition, it has the ability to interface with databases via an ODBC connection and execute statistical tests and analyses against Big Data via an open source connection. These two factors make R well suited to performing statistical tests and analytics on Big Data. As of this writing, R contains nearly 5,000 packages for data analysis and graphical representation. New packages are posted frequently, and many companies are providing value-add

services for R (such as training, instruction, and best practices), as well as packaging it in ways to make it easier to use and more robust. This phenomenon is similar to what happened with Linux in the late 1980s and early 1990s, when companies appeared to package and make Linux easier for companies to consume and deploy. Use R with file extracts for offline analysis and optimal performance, and use RODBC connections for dynamic queries and faster development.

- **SQL Analysis services** [15] can perform in-database analytics of common data mining functions, involved aggregations, and basic predictive models.
- **SAS/ACCESS** [16] provides integration between SAS and the analytics sandbox via multiple data connectors such as OBDC, JDBC, and OLE DB. SAS itself is generally used on file extracts, but with SAS/ACCESS, users can connect to relational databases (such as Oracle or Teradata) and data warehouse appliances (such as Greenplum or Aster), files, and enterprise applications (such as SAP and Salesforce.com).

2.5 Phase 4: Model Building

In Phase 4, the data science team needs to develop datasets for training, testing, and production purposes. These datasets enable the data scientist to develop the analytical model and train it ("training data"), while holding aside some of the data ("hold-out data" or "test data") for testing the model. (These topics are addressed in more detail in Chapter 3.) During this process, it is critical to ensure that the training and test datasets are sufficiently robust for the model and analytical techniques. A simple way to think of these datasets is to view the training dataset for conducting the initial experiments and the test sets for validating an approach once the initial experiments and models have been run.

In the model building phase, shown in Figure 2-6, an analytical model is developed and fit on the training data and evaluated (scored) against the test data. The phases of model planning and model building can overlap quite a bit, and in practice one can iterate back and forth between the two phases for a while before settling on a final model.

Although the modeling techniques and logic required to develop models can be highly complex, the actual duration of this phase can be short compared to the time spent preparing the data and defining the approaches. In general, plan to spend more time preparing and learning the data (Phases 1–2) and crafting a presentation of the findings (Phase 5). Phases 3 and 4 tend to move more quickly, although they are more complex from a conceptual standpoint.

As part of this phase, the data science team needs to execute the models defined in Phase 3.

During this phase, users run models from analytical software packages, such as R or SAS, on file extracts and small datasets for testing purposes. On a small scale, assess the validity of the model and its results. For instance, determine if the model accounts for most of the data and has robust predictive power. At this point, refine the models to optimize the results, such as by modifying variable inputs or reducing correlated variables where appropriate. In Phase 3, the team may have had some knowledge of correlated variables or problematic data attributes, which will be confirmed or denied once the models are actually executed. When immersed in the details of constructing models and transforming data, many small decisions are often made about the data and the approach for the modeling. These details can be easily forgotten once the project is completed. Therefore, it is vital to record the results and logic of the model during this phase. In addition, one must take care to record any operating assumptions that were made in the modeling process regarding the data or the context.

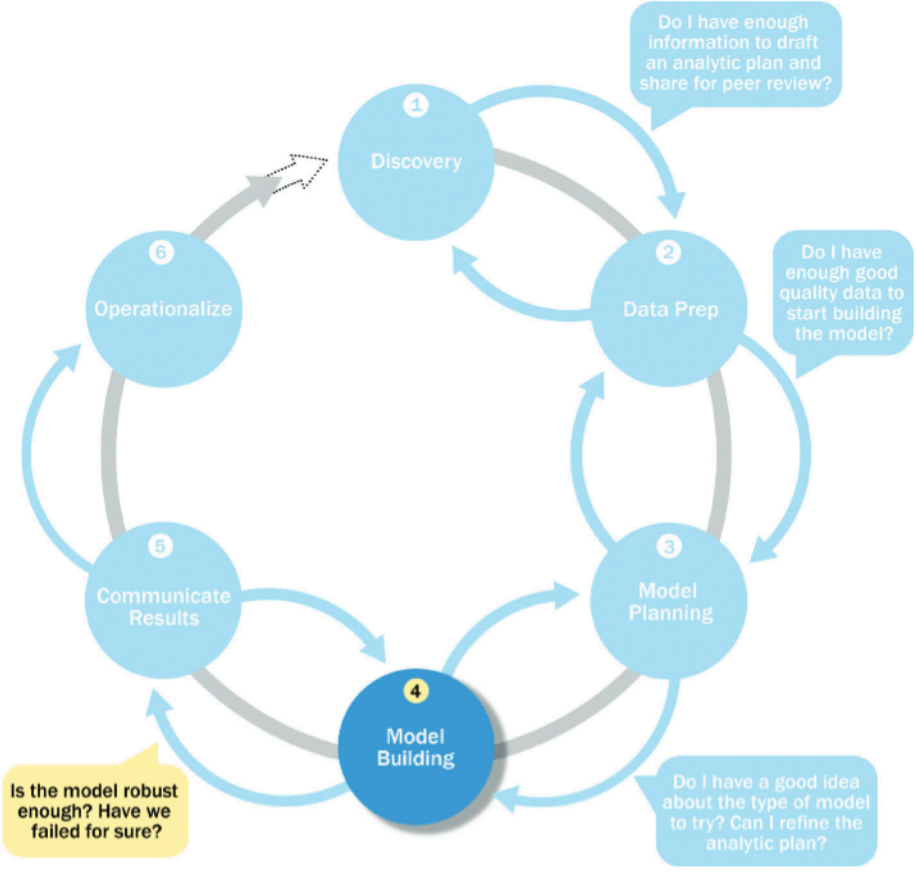


FIGURE 2-6 *Model building phase*

Creating robust models that are suitable to a specific situation requires thoughtful consideration to ensure the models being developed ultimately meet the objectives outlined in Phase 1. Questions to consider include these:

- Does the model appear valid and accurate on the test data?
- Does the model output/behavior make sense to the domain experts? That is, does it appear as if the model is giving answers that make sense in this context?
- Do the parameter values of the fitted model make sense in the context of the domain?
- Is the model sufficiently accurate to meet the goal?
- Does the model avoid intolerable mistakes? Depending on context, false positives may be more serious or less serious than false negatives, for instance. (False positives and false negatives are discussed further in Chapter 3 and Chapter 7, "Advanced Analytical Theory and Methods: Classification.")

- Are more data or more inputs needed? Do any of the inputs need to be transformed or eliminated?
- Will the kind of model chosen support the runtime requirements?
- Is a different form of the model required to address the business problem? If so, go back to the model planning phase and revise the modeling approach.

Once the data science team can evaluate either if the model is sufficiently robust to solve the problem or if the team has failed, it can move to the next phase in the Data Analytics Lifecycle.

2.5.1 Common Tools for the Model Building Phase

There are many tools available to assist in this phase, focused primarily on statistical analysis or data mining software. Common tools in this space include, but are not limited to, the following:

- Commercial Tools:
  - **SAS Enterprise Miner** [17] allows users to run predictive and descriptive models based on large volumes of data from across the enterprise. It interoperates with other large data stores, has many partnerships, and is built for enterprise-level computing and analytics.
  - **SPSS Modeler** [18] (provided by IBM and now called IBM SPSS Modeler) offers methods to explore and analyze data through a GUI.
  - **Matlab** [19] provides a high-level language for performing a variety of data analytics, algorithms, and data exploration.
  - **Alpine Miner** [11] provides a GUI front end for users to develop analytic workflows and interact with Big Data tools and platforms on the back end.
  - **STATISTICA** [20] and **Mathematica** [21] are also popular and well-regarded data mining and analytics tools.
- Free or Open Source tools:
  - **R and PL/R** [14] R was described earlier in the model planning phase, and PL/R is a procedural language for PostgreSQL with R. Using this approach means that R commands can be executed in database. This technique provides higher performance and is more scalable than running R in memory.
  - **Octave** [22], a free software programming language for computational modeling, has some of the functionality of Matlab. Because it is freely available, Octave is used in major universities when teaching machine learning.
  - **WEKA** [23] is a free data mining software package with an analytic workbench. The functions created in WEKA can be executed within Java code.
  - **Python** is a programming language that provides toolkits for machine learning and analysis, such as scikit-learn, numpy, scipy, pandas, and related data visualization using matplotlib.
  - **SQL** in-database implementations, such as **MADlib** [24], provide an alterative to in-memory desktop analytical tools. MADlib provides an open-source machine learning library of algorithms that can be executed in-database, for PostgreSQL or Greenplum.

2.6 Phase 5: Communicate Results

After executing the model, the team needs to compare the outcomes of the modeling to the criteria established for success and failure. In Phase 5, shown in Figure 2-7, the team considers how best to articulate the findings and outcomes to the various team members and stakeholders, taking into account caveats, assumptions, and any limitations of the results. Because the presentation is often circulated within an organization, it is critical to articulate the results properly and position the findings in a way that is appropriate for the audience.

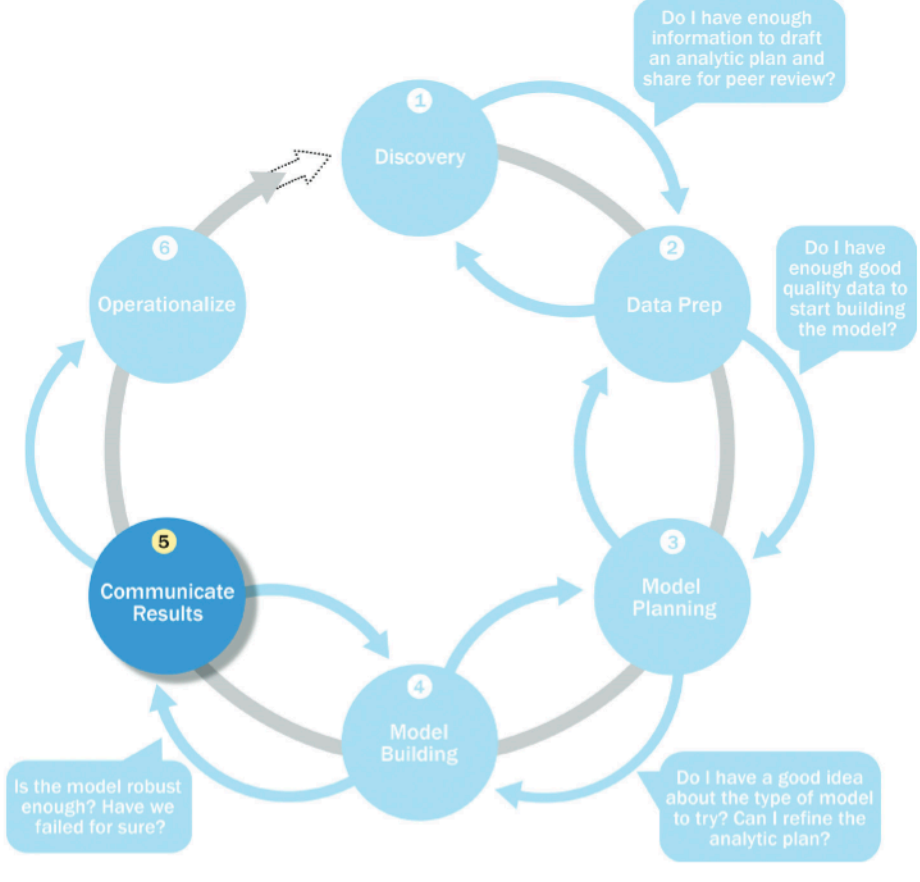


FIGURE 2-7 *Communicate results phase*

As part of Phase 5, the team needs to determine if it succeeded or failed in its objectives. Many times people do not want to admit to failing, but in this instance failure should not be considered as a true failure, but rather as a failure of the data to accept or reject a given hypothesis adequately. This concept can be counterintuitive for those who have been told their whole careers not to fail. However, the key is

to remember that the team must be rigorous enough with the data to determine whether it will prove or disprove the hypotheses outlined in Phase 1 (discovery). Sometimes teams have only done a superficial analysis, which is not robust enough to accept or reject a hypothesis. Other times, teams perform very robust analysis and are searching for ways to show results, even when results may not be there. It is important to strike a balance between these two extremes when it comes to analyzing data and being pragmatic in terms of showing real-world results.

When conducting this assessment, determine if the results are statistically significant and valid. If they are, identify the aspects of the results that stand out and may provide salient findings when it comes time to communicate them. If the results are not valid, think about adjustments that can be made to refine and iterate on the model to make it valid. During this step, assess the results and identify which data points may have been surprising and which were in line with the hypotheses that were developed in Phase 1. Comparing the actual results to the ideas formulated early on produces additional ideas and insights that would have been missed if the team had not taken time to formulate initial hypotheses early in the process.

By this time, the team should have determined which model or models address the analytical challenge in the most appropriate way. In addition, the team should have ideas of some of the findings as a result of the project. The best practice in this phase is to record all the findings and then select the three most significant ones that can be shared with the stakeholders. In addition, the team needs to reflect on the implications of these findings and measure the business value. Depending on what emerged as a result of the model, the team may need to spend time quantifying the business impact of the results to help prepare for the presentation and demonstrate the value of the findings. Doug Hubbard's work [6] offers insights on how to assess intangibles in business and quantify the value of seemingly unmeasurable things.

Now that the team has run the model, completed a thorough discovery phase, and learned a great deal about the datasets, reflect on the project and consider what obstacles were in the project and what can be improved in the future. Make recommendations for future work or improvements to existing processes, and consider what each of the team members and stakeholders needs to fulfill her responsibilities. For instance, sponsors must champion the project. Stakeholders must understand how the model affects their processes. (For example, if the team has created a model to predict customer churn, the Marketing team must understand how to use the churn model predictions in planning their interventions.) Production engineers need to operationalize the work that has been done. In addition, this is the phase to underscore the business benefits of the work and begin making the case to implement the logic into a live production environment.

As a result of this phase, the team will have documented the key findings and major insights derived from the analysis. The deliverable of this phase will be the most visible portion of the process to the outside stakeholders and sponsors, so take care to clearly articulate the results, methodology, and business value of the findings. More details will be provided about data visualization tools and references in Chapter 12, "The Endgame, or Putting It All Together."

2.7 Phase 6: Operationalize

In the final phase, the team communicates the benefits of the project more broadly and sets up a pilot project to deploy the work in a controlled way before broadening the work to a full enterprise or ecosystem of users. In Phase 4, the team scored the model in the analytics sandbox. Phase 6, shown in Figure 2-8, represents the first time that most analytics teams approach deploying the new analytical methods or models in a production environment. Rather than deploying these models immediately on a wide-scale

basis, the risk can be managed more effectively and the team can learn by undertaking a small scope, pilot deployment before a wide-scale rollout. This approach enables the team to learn about the performance and related constraints of the model in a production environment on a small scale and make adjustments before a full deployment. During the pilot project, the team may need to consider executing the algorithm in the database rather than with in-memory tools such as R because the run time is significantly faster and more efficient than running in-memory, especially on larger datasets.

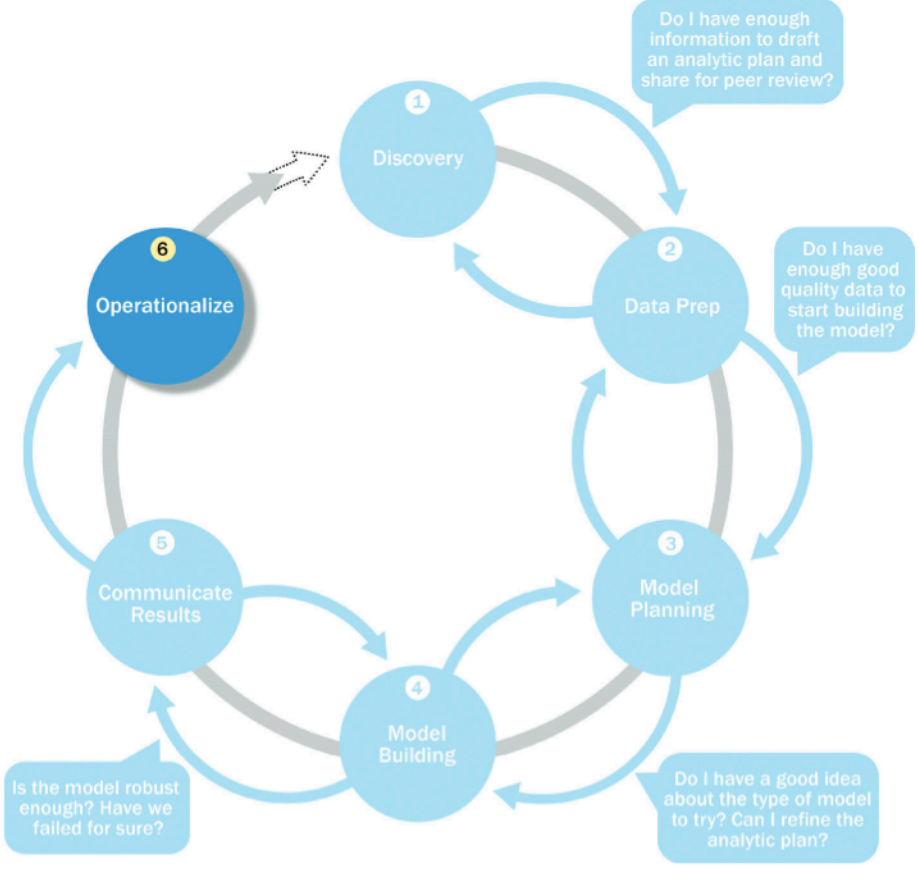


FIGURE 2-8 *Model operationalize phase*

While scoping the effort involved in conducting a pilot project, consider running the model in a production environment for a discrete set of products or a single line of business, which tests the model in a live setting. This allows the team to learn from the deployment and make any needed adjustments before launching the model across the enterprise. Be aware that this phase can bring in a new set of team members—usually the engineers responsible for the production environment who have a new set of issues and concerns beyond those of the core project team. This technical group needs to ensure that

running the model fits smoothly into the production environment and that the model can be integrated into related business processes.

Part of the operationalizing phase includes creating a mechanism for performing ongoing monitoring of model accuracy and, if accuracy degrades, finding ways to retrain the model. If feasible, design alerts for when the model is operating "out-of-bounds." This includes situations when the inputs are beyond the range that the model was trained on, which may cause the outputs of the model to be inaccurate or invalid. If this begins to happen regularly, the model needs to be retrained on new data.

Often, analytical projects yield new insights about a business, a problem, or an idea that people may have taken at face value or thought was impossible to explore. Four main deliverables can be created to meet the needs of most stakeholders. This approach for developing the four deliverables is discussed in greater detail in Chapter 12.

Figure 2-9 portrays the key outputs for each of the main stakeholders of an analytics project and what they usually expect at the conclusion of a project.

- **Business User** typically tries to determine the benefits and implications of the findings to the business.
- **Project Sponsor** typically asks questions related to the business impact of the project, the risks and return on investment (ROI), and the way the project can be evangelized within the organization (and beyond).
- **Project Manager** needs to determine if the project was completed on time and within budget and how well the goals were met.
- **Business Intelligence Analyst** needs to know if the reports and dashboards he manages will be impacted and need to change.
- **Data Engineer** and **Database Administrator** (DBA) typically need to share their code from the analytics project and create a technical document on how to implement it.
- **Data Scientist** needs to share the code and explain the model to her peers, managers, and other stakeholders.

Although these seven roles represent many interests within a project, these interests usually overlap, and most of them can be met with four main deliverables.

- Presentation for project sponsors: This contains high-level takeaways for executive level stakeholders, with a few key messages to aid their decision-making process. Focus on clean, easy visuals for the presenter to explain and for the viewer to grasp.
- Presentation for analysts, which describes business process changes and reporting changes. Fellow data scientists will want the details and are comfortable with technical graphs (such as Receiver Operating Characteristic [ROC] curves, density plots, and histograms shown in Chapter 3 and Chapter 7).
- Code for technical people.
- Technical specifications of implementing the code.

As a general rule, the more executive the audience, the more succinct the presentation needs to be. Most executive sponsors attend many briefings in the course of a day or a week. Ensure that the presentation gets to the point quickly and frames the results in terms of value to the sponsor's organization. For instance, if the team is working with a bank to analyze cases of credit card fraud, highlight the frequency of fraud, the number of cases in the past month or year, and the cost or revenue impact to the bank

(or focus on the reverse—how much more revenue the bank could gain if it addresses the fraud problem). This demonstrates the business impact better than deep dives on the methodology. The presentation needs to include supporting information about analytical methodology and data sources, but generally only as supporting detail or to ensure the audience has confidence in the approach that was taken to analyze the data.

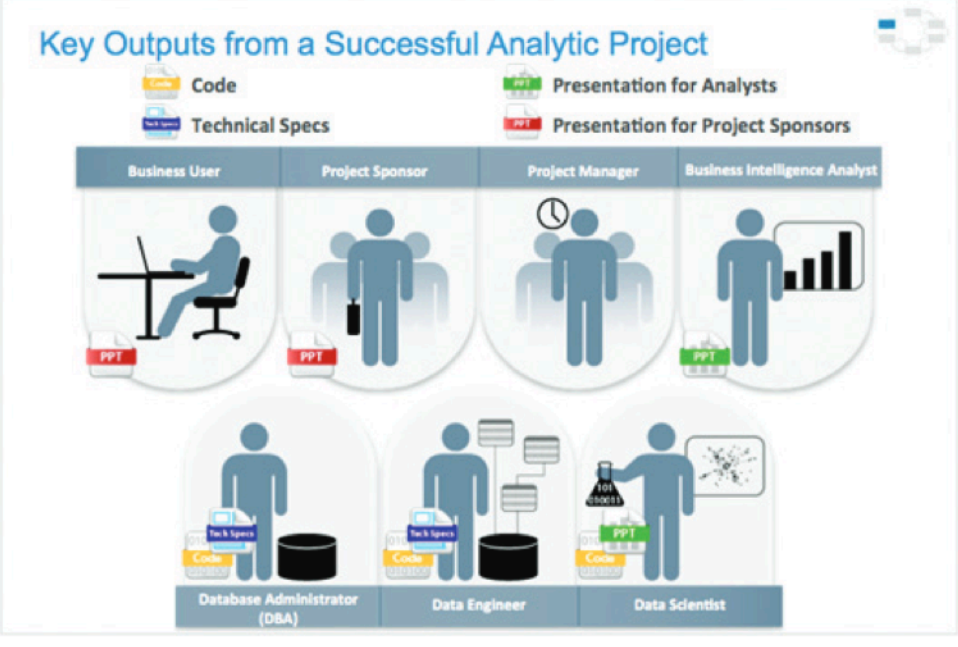


FIGURE 2-9 *Key outputs from a successful analytics project*

When presenting to other audiences with more quantitative backgrounds, focus more time on the methodology and findings. In these instances, the team can be more expansive in describing the outcomes, methodology, and analytical experiment with a peer group. This audience will be more interested in the techniques, especially if the team developed a new way of processing or analyzing data that can be reused in the future or applied to similar problems. In addition, use imagery or data visualization when possible. Although it may take more time to develop imagery, people tend to remember mental pictures to demonstrate a point more than long lists of bullets [25]. Data visualization and presentations are discussed further in Chapter 12.

2.8 Case Study: Global Innovation Network and Analysis (GINA)

EMC's Global Innovation Network and Analytics (GINA) team is a group of senior technologists located in centers of excellence (COEs) around the world. This team's charter is to engage employees across global COEs to drive innovation, research, and university partnerships. In 2012, a newly hired director wanted to

improve these activities and provide a mechanism to track and analyze the related information. In addition, this team wanted to create more robust mechanisms for capturing the results of its informal conversations with other thought leaders within EMC, in academia, or in other organizations, which could later be mined for insights.

The GINA team thought its approach would provide a means to share ideas globally and increase knowledge sharing among GINA members who may be separated geographically. It planned to create a data repository containing both structured and unstructured data to accomplish three main goals.

- Store formal and informal data.
- Track research from global technologists.
- Mine the data for patterns and insights to improve the team's operations and strategy.

The GINA case study provides an example of how a team applied the Data Analytics Lifecycle to analyze innovation data at EMC. Innovation is typically a difficult concept to measure, and this team wanted to look for ways to use advanced analytical methods to identify key innovators within the company.

2.8.1 Phase 1: Discovery

In the GINA project's discovery phase, the team began identifying data sources. Although GINA was a group of technologists skilled in many different aspects of engineering, it had some data and ideas about what it wanted to explore but lacked a formal team that could perform these analytics. After consulting with various experts including Tom Davenport, a noted expert in analytics at Babson College, and Peter Gloor, an expert in collective intelligence and creator of CoIN (Collaborative Innovation Networks) at MIT, the team decided to crowdsource the work by seeking volunteers within EMC.

Here is a list of how the various roles on the working team were fulfilled.

- **Business User, Project Sponsor, Project Manager:** Vice President from Office of the CTO
- **Business Intelligence Analyst:** Representatives from IT
- **Data Engineer** and **Database Administrator (DBA):** Representatives from IT
- **Data Scientist:** Distinguished Engineer, who also developed the social graphs shown in the GINA case study

The project sponsor's approach was to leverage social media and blogging [26] to accelerate the collection of innovation and research data worldwide and to motivate teams of "volunteer" data scientists at worldwide locations. Given that he lacked a formal team, he needed to be resourceful about finding people who were both capable and willing to volunteer their time to work on interesting problems. Data scientists tend to be passionate about data, and the project sponsor was able to tap into this passion of highly talented people to accomplish challenging work in a creative way.

The data for the project fell into two main categories. The first category represented five years of idea submissions from EMC's internal innovation contests, known as the Innovation Roadmap (formerly called the Innovation Showcase). The Innovation Roadmap is a formal, organic innovation process whereby employees from around the globe submit ideas that are then vetted and judged. The best ideas are selected for further incubation. As a result, the data is a mix of structured data, such as idea counts, submission dates, inventor names, and unstructured content, such as the textual descriptions of the ideas themselves.

The second category of data encompassed minutes and notes representing innovation and research activity from around the world. This also represented a mix of structured and unstructured data. The structured data included attributes such as dates, names, and geographic locations. The unstructured documents contained the "who, what, when, and where" information that represents rich data about knowledge growth and transfer within the company. This type of information is often stored in business silos that have little to no visibility across disparate research teams.

The 10 main IHs that the GINA team developed were as follows:

- **IH1:** Innovation activity in different geographic regions can be mapped to corporate strategic directions.
- **IH2:** The length of time it takes to deliver ideas decreases when global knowledge transfer occurs as part of the idea delivery process.
- **IH3:** Innovators who participate in global knowledge transfer deliver ideas more quickly than those who do not.
- **IH4:** An idea submission can be analyzed and evaluated for the likelihood of receiving funding.
- **IH5:** Knowledge discovery and growth for a particular topic can be measured and compared across geographic regions.
- **IH6:** Knowledge transfer activity can identify research-specific boundary spanners in disparate regions.
- **IH7:** Strategic corporate themes can be mapped to geographic regions.
- **IH8:** Frequent knowledge expansion and transfer events reduce the time it takes to generate a corporate asset from an idea.
- **IH9:** Lineage maps can reveal when knowledge expansion and transfer did not (or has not) resulted in a corporate asset.
- **IH10:** Emerging research topics can be classified and mapped to specific ideators, innovators, boundary spanners, and assets.

The GINA (IHs) can be grouped into two categories:

- Descriptive analytics of what is currently happening to spark further creativity, collaboration, and asset generation
- Predictive analytics to advise executive management of where it should be investing in the future

2.8.2 Phase 2: Data Preparation

The team partnered with its IT department to set up a new analytics sandbox to store and experiment on the data. During the data exploration exercise, the data scientists and data engineers began to notice that certain data needed conditioning and normalization. In addition, the team realized that several missing datasets were critical to testing some of the analytic hypotheses.

As the team explored the data, it quickly realized that if it did not have data of sufficient quality or could not get good quality data, it would not be able to perform the subsequent steps in the lifecycle process. As a result, it was important to determine what level of data quality and cleanliness was sufficient for the

project being undertaken. In the case of the GINA, the team discovered that many of the names of the researchers and people interacting with the universities were misspelled or had leading and trailing spaces in the datastore. Seemingly small problems such as these in the data had to be addressed in this phase to enable better analysis and data aggregation in subsequent phases.

2.8.3 Phase 3: Model Planning

In the GINA project, for much of the dataset, it seemed feasible to use social network analysis techniques to look at the networks of innovators within EMC. In other cases, it was difficult to come up with appropriate ways to test hypotheses due to the lack of data. In one case (IH9), the team made a decision to initiate a longitudinal study to begin tracking data points over time regarding people developing new intellectual property. This data collection would enable the team to test the following two ideas in the future:

- **IH8:** Frequent knowledge expansion and transfer events reduce the amount of time it takes to generate a corporate asset from an idea.
- **IH9:** Lineage maps can reveal when knowledge expansion and transfer did not (or has not) result(ed) in a corporate asset.

For the longitudinal study being proposed, the team needed to establish goal criteria for the study. Specifically, it needed to determine the end goal of a successful idea that had traversed the entire journey. The parameters related to the scope of the study included the following considerations:

- Identify the right milestones to achieve this goal.
- Trace how people move ideas from each milestone toward the goal.
- Once this is done, trace ideas that die, and trace others that reach the goal. Compare the journeys of ideas that make it and those that do not.
- Compare the times and the outcomes using a few different methods (depending on how the data is collected and assembled). These could be as simple as t-tests or perhaps involve different types of classification algorithms.

2.8.4 Phase 4: Model Building

In Phase 4, the GINA team employed several analytical methods. This included work by the data scientist using Natural Language Processing (NLP) techniques on the textual descriptions of the Innovation Roadmap ideas. In addition, he conducted social network analysis using R and RStudio, and then he developed social graphs and visualizations of the network of communications related to innovation using R's `ggplot2` package. Examples of this work are shown in Figures 2-10 and 2-11.

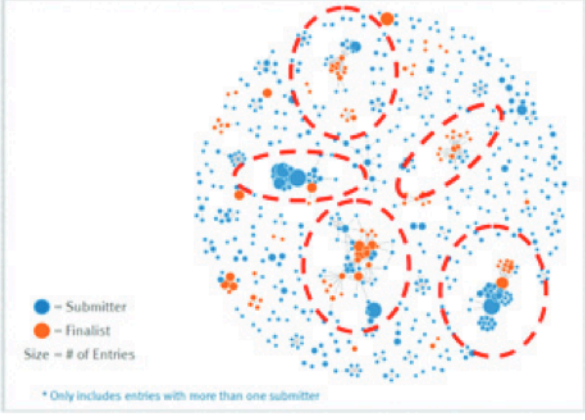


FIGURE 2-10 *Social graph [27] visualization of idea submitters and finalists*

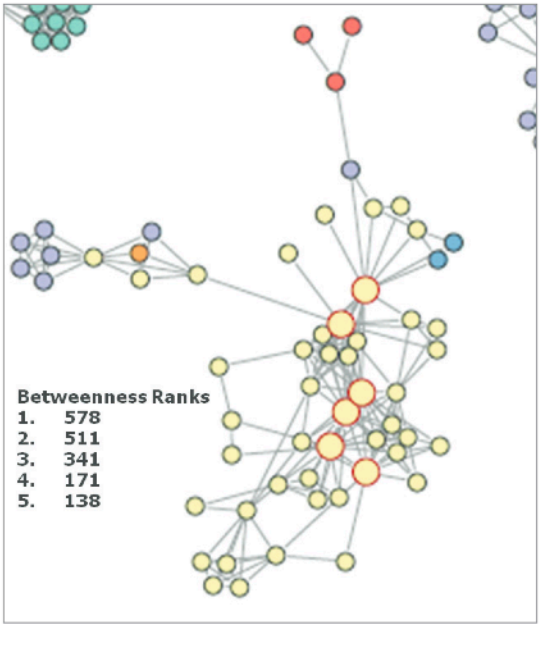


FIGURE 2-11 *Social graph visualization of top innovation influencers*

Figure 2-10 shows social graphs that portray the relationships between idea submitters within GINA. Each color represents an innovator from a different country. The large dots with red circles around them represent hubs. A ***hub*** represents a person with high connectivity and a high "betweenness" score. The cluster in Figure 2-11 contains geographic variety, which is critical to prove the hypothesis about geographic boundary spanners. One person in this graph has an unusually high score when compared to the rest of the nodes in the graph. The data scientist identified this person and ran a query against his name within the analytic sandbox. These actions yielded the following information about this research scientist (from the social graph), which illustrated how influential he was within his business unit and across many other areas of the company worldwide:

- In 2011, he attended the ACM SIGMOD conference, which is a top-tier conference on large-scale data management problems and databases.
- He visited employees in France who are part of the business unit for EMC's content management teams within Documentum (now part of the Information Intelligence Group, or IIG).
- He presented his thoughts on the SIGMOD conference at a virtual brownbag session attended by three employees in Russia, one employee in Cairo, one employee in Ireland, one employee in India, three employees in the United States, and one employee in Israel.
- In 2012, he attended the SDM 2012 conference in California.
- On the same trip he visited innovators and researchers at EMC federated companies, Pivotal and VMware.
- Later on that trip he stood before an internal council of technology leaders and introduced two of his researchers to dozens of corporate innovators and researchers.

This finding suggests that at least part of the initial hypothesis is correct; the data can identify innovators who span different geographies and business units. The team used Tableau software for data visualization and exploration and used the Pivotal Greenplum database as the main data repository and analytics engine.

2.8.5 Phase 5: Communicate Results

In Phase 5, the team found several ways to cull results of the analysis and identify the most impactful and relevant findings. This project was considered successful in identifying boundary spanners and hidden innovators. As a result, the CTO office launched longitudinal studies to begin data collection efforts and track innovation results over longer periods of time. The GINA project promoted knowledge sharing related to innovation and researchers spanning multiple areas within the company and outside of it. GINA also enabled EMC to cultivate additional intellectual property that led to additional research topics and provided opportunities to forge relationships with universities for joint academic research in the fields of Data Science and Big Data. In addition, the project was accomplished with a limited budget, leveraging a volunteer force of highly skilled and distinguished engineers and data scientists.

One of the key findings from the project is that there was a disproportionately high density of innovators in Cork, Ireland. Each year, EMC hosts an innovation contest, open to employees to submit innovation ideas that would drive new value for the company. When looking at the data in 2011, 15% of the finalists and 15% of the winners were from Ireland. These are unusually high numbers, given the relative size of the Cork COE compared to other larger centers in other parts of the world. After further research, it was learned that the COE in Cork, Ireland had received focused training in innovation from an external consultant, which

was proving effective. The Cork COE came up with more innovation ideas, and better ones, than it had in the past, and it was making larger contributions to innovation at EMC. It would have been difficult, if not impossible, to identify this cluster of innovators through traditional methods or even anecdotal, word-of-mouth feedback. Applying social network analysis enabled the team to find a pocket of people within EMC who were making disproportionately strong contributions. These findings were shared internally through presentations and conferences and promoted through social media and blogs.

2.8.6 Phase 6: Operationalize

Running analytics against a sandbox filled with notes, minutes, and presentations from innovation activities yielded great insights into EMC's innovation culture. Key findings from the project include these:

- The CTO office and GINA need more data in the future, including a marketing initiative to convince people to inform the global community on their innovation/research activities.
- Some of the data is sensitive, and the team needs to consider security and privacy related to the data, such as who can run the models and see the results.
- In addition to running models, a parallel initiative needs to be created to improve basic Business Intelligence activities, such as dashboards, reporting, and queries on research activities worldwide.
- A mechanism is needed to continually reevaluate the model after deployment. Assessing the benefits is one of the main goals of this stage, as is defining a process to retrain the model as needed.

In addition to the actions and findings listed, the team demonstrated how analytics can drive new insights in projects that are traditionally difficult to measure and quantify. This project informed investment decisions in university research projects by the CTO office and identified hidden, high-value innovators. In addition, the CTO office developed tools to help submitters improve ideas using topic modeling as part of new recommender systems to help idea submitters find similar ideas and refine their proposals for new intellectual property.

Table 2-3 outlines an analytics plan for the GINA case study example. Although this project shows only three findings, there were many more. For instance, perhaps the biggest overarching result from this project is that it demonstrated, in a concrete way, that analytics can drive new insights in projects that deal with topics that may seem difficult to measure, such as innovation.

Table 2-3 *Analytic Plan from the EMC GINA Project*

Components of Analytic Plan	GINA Case Study
Discovery Business Problem Framed	Tracking global knowledge growth, ensuring effective knowledge transfer, and quickly converting it into corporate assets. Executing on these three elements should accelerate innovation.
Initial Hypotheses	An increase in geographic knowledge transfer improves the speed of idea delivery.
Data	Five years of innovation idea submissions and history; six months of textual notes from global innovation and research activities

(*continues*)

TABLE 2-3 *Analytic Plan from the EMC GINA Project (Continued)*

Components of Analytic Plan	GINA Case Study
Model Planning Analytic Technique	Social network analysis, social graphs, clustering, and regression analysis
Result and Key Findings	1. Identified hidden, high-value innovators and found ways to share their knowledge 2. Informed investment decisions in university research projects 3. Created tools to help submitters improve ideas with idea recommender systems

Innovation is an idea that every company wants to promote, but it can be difficult to measure innovation or identify ways to increase innovation. This project explored this issue from the standpoint of evaluating informal social networks to identify boundary spanners and influential people within innovation sub-networks. In essence, this project took a seemingly nebulous problem and applied advanced analytical methods to tease out answers using an objective, fact-based approach.

Another outcome from the project included the need to supplement analytics with a separate data-store for Business Intelligence reporting, accessible to search innovation/research initiatives. Aside from supporting decision making, this will provide a mechanism to be informed on discussions and research happening worldwide among team members in disparate locations. Finally, it highlighted the value that can be gleaned through data and subsequent analysis. Therefore, the need was identified to start formal marketing programs to convince people to submit (or inform) the global community on their innovation/research activities. The knowledge sharing was critical. Without it, GINA would not have been able to perform the analysis and identify the hidden innovators within the company.

Summary

This chapter described the Data Analytics Lifecycle, which is an approach to managing and executing analytical projects. This approach describes the process in six phases.

1. Discovery
2. Data preparation
3. Model planning
4. Model building
5. Communicate results
6. Operationalize

Through these steps, data science teams can identify problems and perform rigorous investigation of the datasets needed for in-depth analysis. As stated in the chapter, although much is written about the analytical methods, the bulk of the time spent on these kinds of projects is spent in preparation—namely,

in Phases 1 and 2 (discovery and data preparation). In addition, this chapter discussed the seven roles needed for a data science team. It is critical that organizations recognize that Data Science is a team effort, and a balance of skills is needed to be successful in tackling Big Data projects and other complex projects involving data analytics.

Exercises

1. In which phase would the team expect to invest most of the project time? Why? Where would the team expect to spend the least time?
2. What are the benefits of doing a pilot program before a full-scale rollout of a new analytical methodology? Discuss this in the context of the mini case study.
3. What kinds of tools would be used in the following phases, and for which kinds of use scenarios?
   a. Phase 2: Data preparation
   b. Phase 4: Model building

Bibliography

[1] T. H. Davenport and D. J. Patil, "Data Scientist: The Sexiest Job of the 21st Century," *Harvard Business Review,* October 2012.

[2] J. Manyika, M. Chiu, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big Data: The Next Frontier for Innovation, Competition, and Productivity," McKinsey Global Institute, 2011.

[3] "Scientific Method" [Online]. Available: `http://en.wikipedia.org/wiki/Scientific_method`.

[4] "CRISP-DM" [Online]. Available: `http://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining`.

[5] T. H. Davenport, J. G. Harris, and R. Morison, *Analytics at Work: Smarter Decisions, Better Results*, 2010, Harvard Business Review Press.

[6] D. W. Hubbard, *How to Measure Anything: Finding the Value of Intangibles in Business*, 2010, Hoboken, NJ: John Wiley & Sons.

[7] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein and C. Welton, *MAD Skills: New Analysis Practices for Big Data*, Watertown, MA 2009.

[8] "List of APIs" [Online]. Available: `http://www.programmableweb.com/apis`.

[9] B. Shneiderman [Online]. Available: `http://www.ifp.illinois.edu/nabhcs/abstracts/shneiderman.html`.

[10] "Hadoop" [Online]. Available: `http://hadoop.apache.org`.

[11] "Alpine Miner" [Online]. Available: `http://alpinenow.com`.

[12] "OpenRefine" [Online]. Available: `http://openrefine.org`.

[13] "Data Wrangler" [Online]. Available: `http://vis.stanford.edu/wrangler/`.

[14] "CRAN" [Online]. Available: `http://cran.us.r-project.org`.

[15] "SQL" [Online]. Available: `http://en.wikipedia.org/wiki/SQL`.

[16] "SAS/ACCESS" [Online]. Available: `http://www.sas.com/en_us/software/data-management/access.htm`.

[17] "SAS Enterprise Miner" [Online]. Available: http://www.sas.com/en_us/software/analytics/enterprise-miner.html.
[18] "SPSS Modeler" [Online]. Available: http://www-03.ibm.com/software/products/en/category/business-analytics.
[19] "Matlab" [Online]. Available: http://www.mathworks.com/products/matlab/.
[20] "Statistica" [Online]. Available: https://www.statsoft.com.
[21] "Mathematica" [Online]. Available: http://www.wolfram.com/mathematica/.
[22] "Octave" [Online]. Available: https://www.gnu.org/software/octave/.
[23] "WEKA" [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.
[24] "MADlib" [Online]. Available: http://madlib.net.
[25] K. L. Higbee, *Your Memory—How It Works and How to Improve It*, New York: Marlowe & Company, 1996.
[26] S. Todd, "Data Science and Big Data Curriculum" [Online]. Available: http://stevetodd.typepad.com/my_weblog/data-science-and-big-data-curriculum/.
[27] T. H Davenport and D. J. Patil, "Data Scientist: The Sexiest Job of the 21st Century," *Harvard Business Review,* October 2012.

3

Review of Basic Data Analytic Methods Using R

Key Concepts

Basic features of R

Data exploration and analysis with R

Statistical methods for evaluation

The previous chapter presented the six phases of the Data Analytics Lifecycle.

- Phase 1: Discovery
- Phase 2: Data Preparation
- Phase 3: Model Planning
- Phase 4: Model Building
- Phase 5: Communicate Results
- Phase 6: Operationalize

The first three phases involve various aspects of data exploration. In general, the success of a data analysis project requires a deep understanding of the data. It also requires a toolbox for mining and presenting the data. These activities include the study of the data in terms of basic statistical measures and creation of graphs and plots to visualize and identify relationships and patterns. Several free or commercial tools are available for exploring, conditioning, modeling, and presenting data. Because of its popularity and versatility, the open-source programming language R is used to illustrate many of the presented analytical tasks and models in this book.

This chapter introduces the basic functionality of the R programming language and environment. The first section gives an overview of how to use R to acquire, parse, and filter the data as well as how to obtain some basic descriptive statistics on a dataset. The second section examines using R to perform exploratory data analysis tasks using visualization. The final section focuses on statistical inference, such as hypothesis testing and analysis of variance in R.

3.1 Introduction to R

R is a programming language and software framework for statistical analysis and graphics. Available for use under the GNU General Public License [1], R software and installation instructions can be obtained via the Comprehensive R Archive and Network [2]. This section provides an overview of the basic functionality of R. In later chapters, this foundation in R is utilized to demonstrate many of the presented analytical techniques.

Before delving into specific operations and functions of R later in this chapter, it is important to understand the flow of a basic R script to address an analytical problem. The following R code illustrates a typical analytical situation in which a dataset is imported, the contents of the dataset are examined, and some modeling building tasks are executed. Although the reader may not yet be familiar with the R syntax, the code can be followed by reading the embedded comments, denoted by #. In the following scenario, the annual sales in U.S. dollars for 10,000 retail customers have been provided in the form of a comma-separated-value (CSV) file. The `read.csv()` function is used to import the CSV file. This dataset is stored to the R variable *`sales`* using the assignment operator `<-`.

```
# import a CSV file of the total annual sales for each customer
sales <- read.csv("c:/data/yearly_sales.csv")

# examine the imported dataset
head(sales)
```

```
summary(sales)

# plot num_of_orders vs. sales
plot(sales$num_of_orders,sales$sales_total,
        main="Number of Orders vs. Sales")

# perform a statistical analysis (fit a linear regression model)
results <- lm(sales$sales_total ~ sales$num_of_orders)
summary(results)

# perform some diagnostics on the fitted model
# plot histogram of the residuals
hist(results$residuals, breaks = 800)
```

In this example, the data file is imported using the `read.csv()` function. Once the file has been imported, it is useful to examine the contents to ensure that the data was loaded properly as well as to become familiar with the data. In the example, the `head()` function, by default, displays the first six records of *`sales`*.

```
# examine the imported dataset
head(sales)
  cust_id sales_total num_of_orders gender
1  100001      800.64             3      F
2  100002      217.53             3      F
3  100003       74.58             2      M
4  100004      498.60             3      M
5  100005      723.11             4      F
6  100006       69.43             2      F
```

The `summary()` function provides some descriptive statistics, such as the mean and median, for each data column. Additionally, the minimum and maximum values as well as the 1st and 3rd quartiles are provided. Because the *`gender`* column contains two possible characters, an "F" (female) or "M" (male), the `summary()` function provides the count of each character's occurrence.

```
summary(sales)

    cust_id        sales_total       num_of_orders     gender
 Min.   :100001   Min.   :  30.02   Min.   : 1.000   F:5035
 1st Qu.:102501   1st Qu.:  80.29   1st Qu.: 2.000   M:4965
 Median :105001   Median : 151.65   Median : 2.000
 Mean   :105001   Mean   : 249.46   Mean   : 2.428
 3rd Qu.:107500   3rd Qu.: 295.50   3rd Qu.: 3.000
 Max.   :110000   Max.   :7606.09   Max.   :22.000
```

Plotting a dataset's contents can provide information about the relationships between the various columns. In this example, the `plot()` function generates a scatterplot of the number of orders (`sales$num_of_orders`) against the annual sales (`sales$sales_total`). The `$` is used to reference a specific column in the dataset `sales`. The resulting plot is shown in Figure 3-1.

```
# plot num_of_orders vs. sales
plot(sales$num_of_orders,sales$sales_total,
        main="Number of Orders vs. Sales")
```

FIGURE 3-1 *Graphically examining the data*

Each point corresponds to the number of orders and the total sales for each customer. The plot indicates that the annual sales are proportional to the number of orders placed. Although the observed relationship between these two variables is not purely linear, the analyst decided to apply linear regression using the `lm()` function as a first step in the modeling process.

```
results <- lm(sales$sales_total ~ sales$num_of_orders)
results

Call:
lm(formula = sales$sales_total ~ sales$num_of_orders)

Coefficients:
        (Intercept)  sales$num_of_orders
             -154.1                166.2
```

The resulting intercept and slope values are –154.1 and 166.2, respectively, for the fitted linear equation. However, *`results`* stores considerably more information that can be examined with the `summary()` function. Details on the contents of *`results`* are examined by applying the `attributes()` function. Because regression analysis is presented in more detail later in the book, the reader should not overly focus on interpreting the following output.

```
summary(results)

Call:
lm(formula = sales$sales_total ~ sales$num_of_orders)

Residuals:
   Min     1Q Median     3Q    Max
-666.5 -125.5  -26.7   86.6 4103.4

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         -154.128      4.129  -37.33   <2e-16 ***
sales$num_of_orders  166.221      1.462  113.66   <2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 210.8 on 9998 degrees of freedom
Multiple R-squared:  0.5637,      Adjusted R-squared:  0.5637
F-statistic: 1.292e+04 on 1 and 9998 DF,  p-value: < 2.2e-16
```

The `summary()` function is an example of a generic function. A ***generic function*** is a group of functions sharing the same name but behaving differently depending on the number and the type of arguments they receive. Utilized previously, `plot()` is another example of a generic function; the plot is determined by the passed variables. Generic functions are used throughout this chapter and the book. In the final portion of the example, the following R code uses the generic function `hist()` to generate a histogram (Figure 3-2) of the residuals stored in *`results`*. The function call illustrates that optional parameter values can be passed. In this case, the number of *`breaks`* is specified to observe the large residuals.

```
# perform some diagnostics on the fitted model
# plot histogram of the residuals
hist(results$residuals, breaks = 800)
```

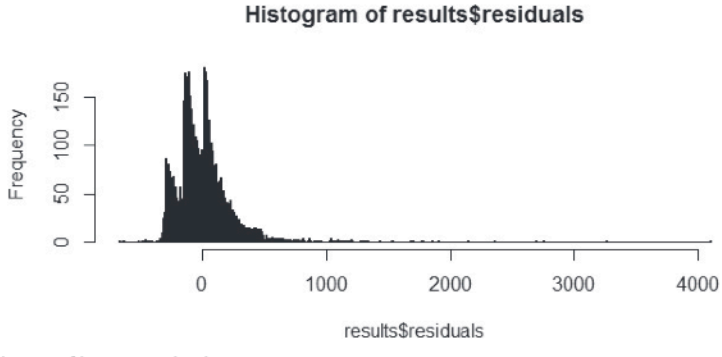


FIGURE 3-2 *Evidence of large residuals*

This simple example illustrates a few of the basic model planning and building tasks that may occur in Phases 3 and 4 of the Data Analytics Lifecycle. Throughout this chapter, it is useful to envision how the presented R functionality will be used in a more comprehensive analysis.

3.1.1 R Graphical User Interfaces

R software uses a command-line interface (CLI) that is similar to the BASH shell in Linux or the interactive versions of scripting languages such as Python. UNIX and Linux users can enter command `R` at the terminal prompt to use the CLI. For Windows installations, R comes with RGui.exe, which provides a basic graphical user interface (GUI). However, to improve the ease of writing, executing, and debugging R code, several additional GUIs have been written for R. Popular GUIs include the R commander [3], Rattle [4], and RStudio [5]. This section presents a brief overview of RStudio, which was used to build the R examples in this book. Figure 3-3 provides a screenshot of the previous R code example executed in RStudio.

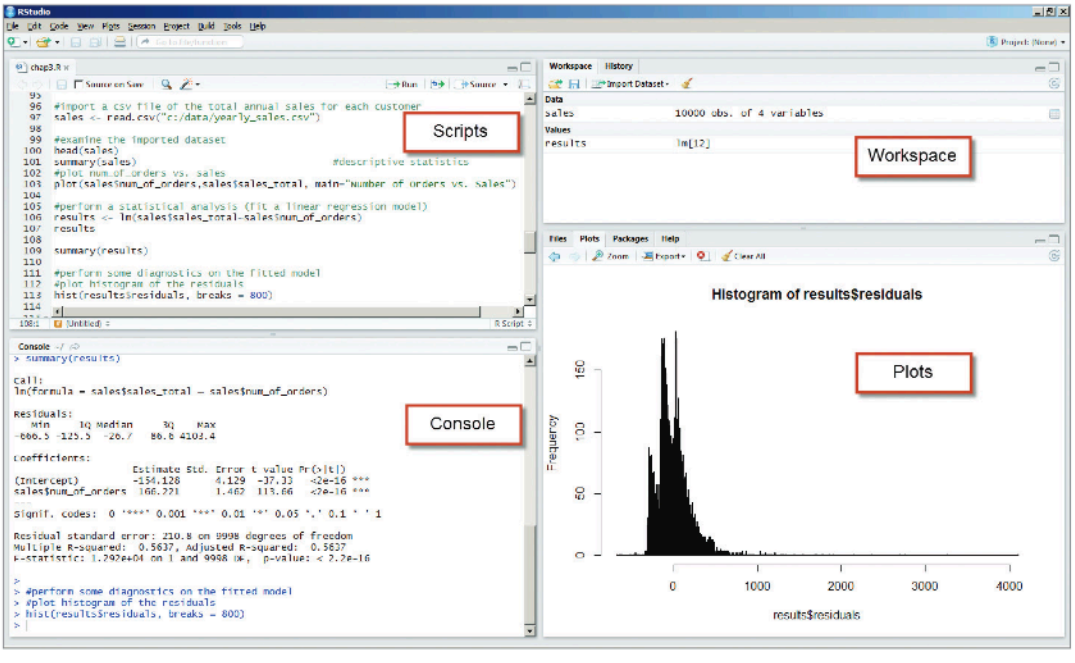


FIGURE 3-3 *RStudio GUI*

The four highlighted window panes follow.

- **Scripts:** Serves as an area to write and save R code
- **Workspace:** Lists the datasets and variables in the R environment
- **Plots:** Displays the plots generated by the R code and provides a straightforward mechanism to export the plots
- **Console:** Provides a history of the executed R code and the output

Additionally, the console pane can be used to obtain help information on R. Figure 3-4 illustrates that by entering `?lm` at the console prompt, the help details of the `lm()` function are provided on the right. Alternatively, `help(lm)` could have been entered at the console prompt.

Functions such as `edit()` and `fix()` allow the user to update the contents of an R variable. Alternatively, such changes can be implemented with RStudio by selecting the appropriate variable from the workspace pane.

R allows one to save the workspace environment, including variables and loaded libraries, into an `.Rdata` file using the `save.image()` function. An existing `.Rdata` file can be loaded using the `load.image()` function. Tools such as RStudio prompt the user for whether the developer wants to save the workspace connects prior to exiting the GUI.

The reader is encouraged to install R and a preferred GUI to try out the R examples provided in the book and utilize the help functionality to access more details about the discussed topics.

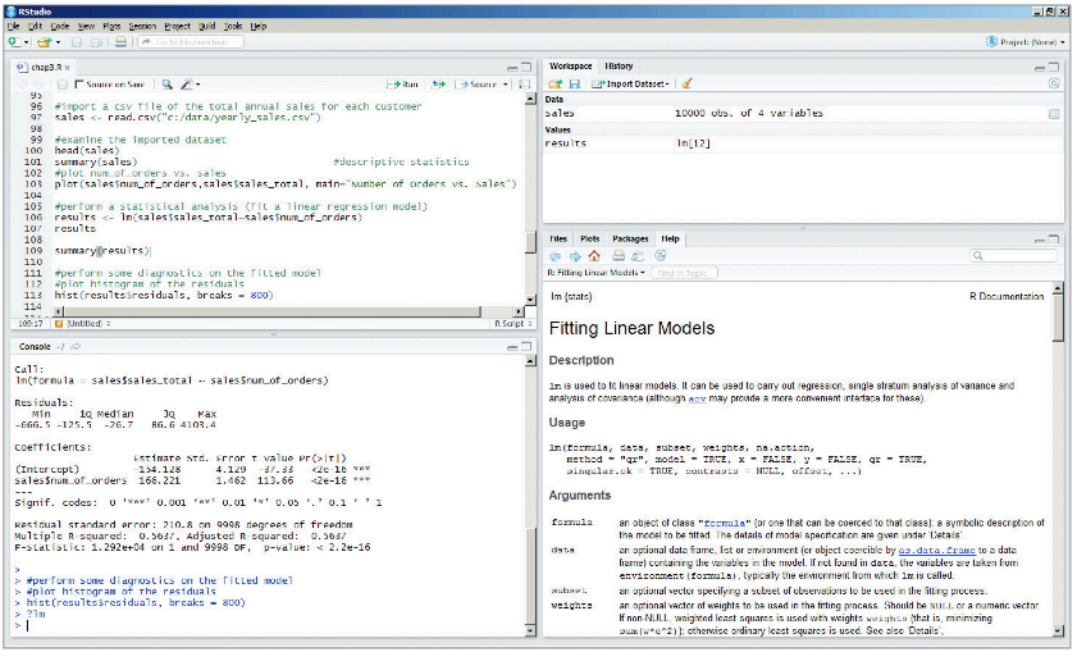


FIGURE 3-4 *Accessing help in Rstudio*

3.1.2 Data Import and Export

In the annual retail sales example, the dataset was imported into R using the `read.csv()` function as in the following code.

```
sales <- read.csv("c:/data/yearly_sales.csv")
```

R uses a forward slash (/) as the separator character in the directory and file paths. This convention makes script files somewhat more portable at the expense of some initial confusion on the part of Windows users, who may be accustomed to using a backslash (\) as a separator. To simplify the import of multiple files with long path names, the `setwd()` function can be used to set the working directory for the subsequent import and export operations, as shown in the following R code.

```
setwd("c:/data/")
sales <- read.csv("yearly_sales.csv")
```

Other import functions include `read.table()` and `read.delim()`, which are intended to import other common file types such as TXT. These functions can also be used to import the `yearly_sales.csv` file, as the following code illustrates.

```
sales_table <- read.table("yearly_sales.csv", header=TRUE, sep=",")
sales_delim <- read.delim("yearly_sales.csv", sep=",")
```

The main difference between these import functions is the default values. For example, the `read.delim()` function expects the column separator to be a tab ("\t"). In the event that the numerical data

in a data file uses a comma for the decimal, R also provides two additional functions—`read.csv2()` and `read.delim2()`—to import such data. Table 3-1 includes the expected defaults for headers, column separators, and decimal point notations.

TABLE 3-1 *Import Function Defaults*

Function	Headers	Separator	Decimal Point
`read.table()`	FALSE	""	"."
`read.csv()`	TRUE	","	"."
`read.csv2()`	TRUE	";"	","
`read.delim()`	TRUE	"\t"	"."
`read.delim2()`	TRUE	"\t"	","

The analogous R functions such as `write.table()`, `write.csv()`, and `write.csv2()` enable exporting of R datasets to an external file. For example, the following R code adds an additional column to the sales dataset and exports the modified dataset to an external file.

```
# add a column for the average sales per order
sales$per_order <- sales$sales_total/sales$num_of_orders

# export data as tab delimited without the row names
write.table(sales,"sales_modified.txt", sep="\t", row.names=FALSE
```

Sometimes it is necessary to read data from a database management system (DBMS). R packages such as `DBI` [6] and `RODBC` [7] are available for this purpose. These packages provide database interfaces for communication between R and DBMSs such as MySQL, Oracle, SQL Server, PostgreSQL, and Pivotal Greenplum. The following R code demonstrates how to install the `RODBC` package with the `install.packages()` function. The `library()` function loads the package into the R workspace. Finally, a connector (*conn*) is initialized for connecting to a Pivotal Greenplum database *`training2`* via open database connectivity (ODBC) with user `user`. The *`training2`* database must be defined either in the `/etc/ODBC.ini` configuration file or using the Administrative Tools under the Windows Control Panel.

```
install.packages("RODBC")
library(RODBC)
conn <- odbcConnect("training2", uid="user", pwd="password")
```

The connector needs to be present to submit a SQL query to an ODBC database by using the `sqlQuery()` function from the `RODBC` package. The following R code retrieves specific columns from the *`housing`* table in which household income (`hinc`) is greater than $1,000,000.

```
housing_data <- sqlQuery(conn, "select serialno, state, persons, rooms
                                from housing
                                where hinc > 1000000")
head(housing_data)
  serialno state persons rooms
1  3417867     6       2     7
2  3417867     6       2     7
```

```
3   4552088     6        5      9
4   4552088     6        5      9
5   8699293     6        5      5
6   8699293     6        5      5
```

Although plots can be saved using the RStudio GUI, plots can also be saved using R code by specifying the appropriate graphic devices. Using the `jpeg()` function, the following R code creates a new JPEG file, adds a histogram plot to the file, and then closes the file. Such techniques are useful when automating standard reports. Other functions, such as `png()`, `bmp()`, `pdf()`, and `postscript()`, are available in R to save plots in the desired format.

```
jpeg(file="c:/data/sales_hist.jpeg") # create a new jpeg file
hist(sales$num_of_orders)            # export histogram to jpeg
dev.off()                            # shut off the graphic device
```

More information on data imports and exports can be found at `http://cran.r-project.org/doc/manuals/r-release/R-data.html`, such as how to import datasets from statistical software packages including Minitab, SAS, and SPSS.

3.1.3 Attribute and Data Types

In the earlier example, the *`sales`* variable contained a record for each customer. Several characteristics, such as total annual sales, number of orders, and gender, were provided for each customer. In general, these characteristics or attributes provide the qualitative and quantitative measures for each item or subject of interest. Attributes can be categorized into four types: nominal, ordinal, interval, and ratio (NOIR) [8]. Table 3-2 distinguishes these four attribute types and shows the operations they support. Nominal and ordinal attributes are considered categorical attributes, whereas interval and ratio attributes are considered numeric attributes.

TABLE 3-2 *NOIR Attribute Types*

	Categorical (Qualitative)		Numeric (Quantitative)	
	Nominal	**Ordinal**	**Interval**	**Ratio**
Definition	The values represent labels that distinguish one from another.	Attributes imply a sequence.	The difference between two values is meaningful.	Both the difference and the ratio of two values are meaningful.
Examples	ZIP codes, nationality, street names, gender, employee ID numbers, `TRUE` or `FALSE`	Quality of diamonds, academic grades, magnitude of earthquakes	Temperature in Celsius or Fahrenheit, calendar dates, latitudes	Age, temperature in Kelvin, counts, length, weight
Operations	$=, \neq$	$=, \neq$, $<, \leq, >, \geq$	$=, \neq$, $<, \leq, >, \geq$, $+, -$	$=, \neq$, $<, \leq, >, \geq$, $+, -$, $\times, \div$

Data of one attribute type may be converted to another. For example, the *quality* of diamonds {Fair, Good, Very Good, Premium, Ideal} is considered ordinal but can be converted to nominal {Good, Excellent} with a defined mapping. Similarly, a ratio attribute like *Age* can be converted into an ordinal attribute such as {Infant, Adolescent, Adult, Senior}. Understanding the attribute types in a given dataset is important to ensure that the appropriate descriptive statistics and analytic methods are applied and properly interpreted. For example, the mean and standard deviation of U.S. postal ZIP codes are not very meaningful or appropriate. Proper handling of categorical variables will be addressed in subsequent chapters. Also, it is useful to consider these attribute types during the following discussion on R data types.

Numeric, Character, and Logical Data Types

Like other programming languages, R supports the use of numeric, character, and logical (Boolean) values. Examples of such variables are given in the following R code.

```
i <- 1                          # create a numeric variable
sport <- "football"             # create a character variable
flag <- TRUE                    # create a logical variable
```

R provides several functions, such as `class()` and `typeof()`, to examine the characteristics of a given variable. The `class()` function represents the abstract class of an object. The `typeof()` function determines the way an object is stored in memory. Although *i* appears to be an integer, *i* is internally stored using double precision. To improve the readability of the code segments in this section, the inline R comments are used to explain the code or to provide the returned values.

```
class(i)                        # returns "numeric"
typeof(i)                       # returns "double"

class(sport)                    # returns "character"
typeof(sport)                   # returns "character"

class(flag)                     # returns "logical"
typeof(flag)                    # returns "logical"
```

Additional R functions exist that can test the variables and coerce a variable into a specific type. The following R code illustrates how to test if *i* is an integer using the `is.integer()` function and to coerce *i* into a new integer variable, *j*, using the `as.integer()` function. Similar functions can be applied for double, character, and logical types.

```
is.integer(i)                   # returns FALSE
j <- as.integer(i)              # coerces contents of i into an integer
is.integer(j)                   # returns TRUE
```

The application of the `length()` function reveals that the created variables each have a length of 1. One might have expected the returned length of *sport* to have been 8 for each of the characters in the string `"football"`. However, these three variables are actually one element, ***vectors***.

```
length(i)                       # returns 1
length(flag)                    # returns 1
length(sport)                   # returns 1 (not 8 for "football")
```

Vectors

Vectors are a basic building block for data in R. As seen previously, simple R variables are actually vectors. A vector can only consist of values in the same class. The tests for vectors can be conducted using the `is.vector()` function.

```
is.vector(i)                    # returns TRUE
is.vector(flag)                 # returns TRUE
is.vector(sport)                # returns TRUE
```

R provides functionality that enables the easy creation and manipulation of vectors. The following R code illustrates how a vector can be created using the combine function, `c()` or the colon operator, `:`, to build a vector from the sequence of integers from 1 to 5. Furthermore, the code shows how the values of an existing vector can be easily modified or accessed. The code, related to the *z* vector, indicates how logical comparisons can be built to extract certain elements of a given vector.

```
u <- c("red", "yellow", "blue") # create a vector "red" "yellow" "blue"
u                               # returns "red" "yellow" "blue"
u[1]                            # returns "red" (1st element in u)
v <- 1:5                        # create a vector 1 2 3 4 5
v                               # returns 1 2 3 4 5
sum(v)                          # returns 15
w <- v * 2                      # create a vector 2 4 6 8 10
w                               # returns 2 4 6 8 10
w[3]                            # returns 6 (the 3rd element of w)
z <- v + w                      # sums two vectors element by element
z                               # returns 3  6  9 12 15
z > 8                           # returns FALSE FALSE  TRUE  TRUE  TRUE
z[z > 8]                        # returns 9 12 15
z[z > 8 | z < 5]                # returns 3  9 12 15 ("|" denotes "or")
```

Sometimes it is necessary to initialize a vector of a specific length and then populate the content of the vector later. The `vector()` function, by default, creates a logical vector. A vector of a different type can be specified by using the `mode` parameter. The vector *c*, an integer vector of length 0, may be useful when the number of elements is not initially known and the new elements will later be added to the end of the vector as the values become available.

```
a <- vector(length=3)           # create a logical vector of length 3
a                               # returns FALSE FALSE FALSE
b <- vector(mode="numeric", 3)  # create a numeric vector of length 3
typeof(b)                       # returns "double"
b[2] <- 3.1                     # assign 3.1 to the 2nd element
b                               # returns 0.0 3.1 0.0
c <- vector(mode="integer", 0)  # create an integer vector of length 0
c                               # returns integer(0)
length(c)                       # returns 0
```

Although vectors may appear to be analogous to arrays of one dimension, they are technically dimensionless, as seen in the following R code. The concept of arrays and matrices is addressed in the following discussion.

```
length(b)                          # returns 3
dim(b)                             # returns NULL (an undefined value)
```

Arrays and Matrices

The `array()` function can be used to restructure a vector as an array. For example, the following R code builds a three-dimensional array to hold the quarterly sales for three regions over a two-year period and then assign the sales amount of $158,000 to the second region for the first quarter of the first year.

```
# the dimensions are 3 regions, 4 quarters, and 2 years
quarterly_sales <- array(0, dim=c(3,4,2))
quarterly_sales[2,1,1] <-  158000
quarterly_sales

, , 1

       [,1] [,2] [,3] [,4]
[1,]      0    0    0    0
[2,] 158000    0    0    0
[3,]      0    0    0    0

, , 2

     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
```

A two-dimensional array is known as a ***matrix***. The following code initializes a matrix to hold the quarterly sales for the three regions. The parameters `nrow` and `ncol` define the number of rows and columns, respectively, for the *`sales_matrix`*.

```
sales_matrix <- matrix(0, nrow = 3, ncol = 4)
sales_matrix

     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
```

R provides the standard matrix operations such as addition, subtraction, and multiplication, as well as the transpose function `t()` and the inverse matrix function `matrix.inverse()` included in the `matrixcalc` package. The following R code builds a 3×3 matrix, M, and multiplies it by its inverse to obtain the identity matrix.

```
library(matrixcalc)
M <- matrix(c(1,3,3,5,0,4,3,3,3),nrow = 3,ncol = 3) # build a 3x3 matrix
```

```
M %*% matrix.inverse(M)                        # multiply M by inverse(M)

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Data Frames

Similar to the concept of matrices, data frames provide a structure for storing and accessing several variables of possibly different data types. In fact, as the `is.data.frame()` function indicates, a data frame was created by the `read.csv()` function at the beginning of the chapter.

```
#import a CSV file of the total annual sales for each customer
sales <- read.csv("c:/data/yearly_sales.csv")
is.data.frame(sales)            # returns TRUE
```

As seen earlier, the variables stored in the data frame can be easily accessed using the `$` notation. The following R code illustrates that in this example, each variable is a vector with the exception of `gender`, which was, by a `read.csv()` default, imported as a ***factor***. Discussed in detail later in this section, a factor denotes a categorical variable, typically with a few finite levels such as "F" and "M" in the case of *`gender`*.

```
length(sales$num_of_orders)     # returns 10000 (number of customers)

is.vector(sales$cust_id)        # returns TRUE
is.vector(sales$sales_total)    # returns TRUE
is.vector(sales$num_of_orders)  # returns TRUE
is.vector(sales$gender)         # returns FALSE

is.factor(sales$gender)         # returns TRUE
```

Because of their flexibility to handle many data types, data frames are the preferred input format for many of the modeling functions available in R. The following use of the `str()` function provides the structure of the *`sales`* data frame. This function identifies the integer and numeric (double) data types, the factor variables and levels, as well as the first few values for each variable.

```
str(sales)              # display structure of the data frame object

'data.frame': 10000 obs. of 4 variables:
 $ cust_id      : int  100001 100002 100003 100004 100005 100006 ...
 $ sales_total  : num  800.6 217.5 74.6 498.6 723.1 ...
 $ num_of_orders: int  3 3 2 3 4 2 2 2 2 2 ...
 $ gender       : Factor w/ 2 levels "F","M": 1 1 2 2 1 1 2 2 1 2 ...
```

In the simplest sense, data frames are lists of variables of the same length. A subset of the data frame can be retrieved through ***subsetting operators***. R's subsetting operators are powerful in that they allow one to express complex operations in a succinct fashion and easily retrieve a subset of the dataset.

```
# extract the fourth column of the sales data frame
sales[,4]
# extract the gender column of the sales data frame
```

```
sales$gender
# retrieve the first two rows of the data frame
sales[1:2,]
# retrieve the first, third, and fourth columns
sales[,c(1,3,4)]
# retrieve both the cust_id and the sales_total columns
sales[,c("cust_id", "sales_total")]
# retrieve all the records whose gender is female
sales[sales$gender=="F",]
```

The following R code shows that the class of the `sales` variable is a data frame. However, the type of the `sales` variable is a list. A ***list*** is a collection of objects that can be of various types, including other lists.

```
class(sales)
"data.frame"
typeof(sales)
"list"
```

Lists

Lists can contain any type of objects, including other lists. Using the vector v and the matrix M created in earlier examples, the following R code creates *`assortment`*, a list of different object types.

```
# build an assorted list of a string, a numeric, a list, a vector,
# and a matrix
housing <- list("own", "rent")
assortment <- list("football", 7.5, housing, v, M)
assortment

[[1]]
[1] "football"

[[2]]
[1] 7.5

[[3]]
[[3]][[1]]
[1] "own"

[[3]][[2]]
[1] "rent"


[[4]]
[1] 1 2 3 4 5

[[5]]
```

```
     [,1] [,2] [,3]
[1,]    1    5    3
[2,]    3    0    3
[3,]    3    4    3
```

In displaying the contents of *assortment*, the use of the double brackets, `[[]]`, is of particular importance. As the following R code illustrates, the use of the single set of brackets only accesses an item in the list, not its content.

```
# examine the fifth object, M, in the list
class(assortment[5])            # returns "list"
length(assortment[5])           # returns 1

class(assortment[[5]])          # returns "matrix"
length(assortment[[5]])         # returns 9 (for the 3x3 matrix)
```

As presented earlier in the data frame discussion, the `str()` function offers details about the structure of a list.

```
str(assortment)
List of 5
 $ : chr "football"
 $ : num 7.5
 $ :List of 2
  ..$ : chr "own"
  ..$ : chr "rent"
 $ : int [1:5] 1 2 3 4 5
 $ : num [1:3, 1:3] 1 3 3 5 0 4 3 3 3
```

Factors

Factors were briefly introduced during the discussion of the *gender* variable in the data frame *sales*. In this case, *gender* could assume one of two levels: `F` or `M`. Factors can be ordered or not ordered. In the case of *gender*, the levels are not ordered.

```
class(sales$gender)             # returns "factor"
is.ordered(sales$gender)        # returns FALSE
```

Included with the `ggplot2` package, the *diamonds* data frame contains three ordered factors. Examining the *cut* factor, there are five levels in order of improving cut: Fair, Good, Very Good, Premium, and Ideal. Thus, `sales$gender` contains nominal data, and `diamonds$cut` contains ordinal data.

```
head(sales$gender)     # display first six values and the levels

F F M M F F
Levels: F M

library(ggplot2)
data(diamonds)          # load the data frame into the R workspace
```

```
str(diamonds)
'data.frame': 53940 obs. of 10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2  ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...

head(diamonds$cut)        # display first six values and the levels
Ideal     Premium   Good      Premium   Good      Very Good
Levels: Fair < Good < Very Good < Premium < Ideal
```

Suppose it is decided to categorize *sales$sales_totals* into three groups—small, medium, and big—according to the amount of the sales with the following code. These groupings are the basis for the new ordinal factor, spender, with levels {small, medium, big}.

```
# build an empty character vector of the same length as sales
sales_group <- vector(mode="character",
                      length=length(sales$sales_total))

# group the customers according to the sales amount
sales_group[sales$sales_total<100] <- "small"
sales_group[sales$sales_total>=100 & sales$sales_total<500] <- "medium"
sales_group[sales$sales_total>=500] <- "big"

# create and add the ordered factor to the sales data frame
spender <- factor(sales_group,levels=c("small", "medium", "big"),
                              ordered = TRUE)
sales <- cbind(sales,spender)

str(sales$spender)
Ord.factor w/ 3 levels "small"<"medium"<..: 3 2 1 2 3 1 1 1 2 1 ...

head(sales$spender)
big    medium small  medium big    small
Levels: small < medium < big
```

The `cbind()` function is used to combine variables column-wise. The `rbind()` function is used to combine datasets row-wise. The use of factors is important in several R statistical modeling functions, such as analysis of variance, `aov()`, presented later in this chapter, and the use of contingency tables, discussed next.

Contingency Tables

In R, ***table*** refers to a class of objects used to store the observed counts across the factors for a given dataset. Such a table is commonly referred to as a contingency table and is the basis for performing a statistical test on the independence of the factors used to build the table. The following R code builds a contingency table based on the `sales$gender` and `sales$spender` factors.

```
# build a contingency table based on the gender and spender factors
sales_table <- table(sales$gender,sales$spender)
sales_table
  small medium  big
F  1726   2746  563
M  1656   2723  586

class(sales_table)              # returns "table"
typeof(sales_table)             # returns "integer"
dim(sales_table)                # returns 2 3

# performs a chi-squared test
summary(sales_table)
Number of cases in table: 10000
Number of factors: 2
Test for independence of all factors:
  Chisq = 1.516, df = 2, p-value = 0.4686
```

Based on the observed counts in the table, the `summary()` function performs a chi-squared test on the independence of the two factors. Because the reported *p*-value is greater than 0.05, the assumed independence of the two factors is not rejected. Hypothesis testing and *p*-values are covered in more detail later in this chapter. Next, applying descriptive statistics in R is examined.

3.1.4 Descriptive Statistics

It has already been shown that the `summary()` function provides several descriptive statistics, such as the mean and median, about a variable such as the *`sales`* data frame. The results now include the counts for the three levels of the *`spender`* variable based on the earlier examples involving factors.

```
summary(sales)
    cust_id        sales_total      num_of_orders    gender     spender
 Min.   :100001   Min.   :  30.02   Min.   : 1.000  F:5035  small :3382
 1st Qu.:102501   1st Qu.:  80.29   1st Qu.: 2.000  M:4965  medium:5469
 Median :105001   Median : 151.65   Median : 2.000          big   :1149
 Mean   :105001   Mean   : 249.46   Mean   : 2.428
 3rd Qu.:107500   3rd Qu.: 295.50   3rd Qu.: 3.000
 Max.   :110000   Max.   :7606.09   Max.   :22.000
```

The following code provides some common R functions that include descriptive statistics. In parentheses, the comments describe the functions.

```
# to simplify the function calls, assign
x <- sales$sales_total
y <- sales$num_of_orders

cor(x,y)                        # returns 0.7508015 (correlation)
cov(x,y)                        # returns 345.2111 (covariance)
IQR(x)                          # returns 215.21 (interquartile range)
mean(x)                         # returns 249.4557 (mean)
median(x)                       # returns 151.65 (median)
range(x)                        # returns 30.02 7606.09 (min max)
sd(x)                           # returns 319.0508 (std. dev.)
var(x)                          # returns 101793.4 (variance)
```

The `IQR()` function provides the difference between the third and the first quartiles. The other functions are fairly self-explanatory by their names. The reader is encouraged to review the available help files for acceptable inputs and possible options.

The function `apply()` is useful when the same function is to be applied to several variables in a data frame. For example, the following R code calculates the standard deviation for the first three variables in *sales*. In the code, setting `MARGIN=2` specifies that the `sd()` function is applied over the columns. Other functions, such as `lapply()` and `sapply()`, apply a function to a list or vector. Readers can refer to the R help files to learn how to use these functions.

```
apply(sales[,c(1:3)], MARGIN=2, FUN=sd)
      cust_id   sales_total num_of_orders
  2886.895680    319.050782      1.441119
```

Additional descriptive statistics can be applied with user-defined functions. The following R code defines a function, `my_range()`, to compute the difference between the maximum and minimum values returned by the `range()` function. In general, user-defined functions are useful for any task or operation that needs to be frequently repeated. More information on user-defined functions is available by entering `help("function")` in the console.

```
# build a function to provide the difference between
# the maximum and the minimum values
my_range <- function(v) {range(v)[2] - range(v)[1]}
my_range(x)
7576.07
```

3.2 Exploratory Data Analysis

So far, this chapter has addressed importing and exporting data in R, basic data types and operations, and generating descriptive statistics. Functions such as `summary()` can help analysts easily get an idea of the magnitude and range of the data, but other aspects such as linear relationships and distributions are more difficult to see from descriptive statistics. For example, the following code shows a summary view of a data frame *data* with two columns *x* and *y*. The output shows the range of *x* and *y*, but it's not clear what the relationship may be between these two variables.

```
summary(data)
       x                   y
 Min.   :-1.90483   Min.   :-2.16545
 1st Qu.:-0.66321   1st Qu.:-0.71451
 Median : 0.09367   Median :-0.03797
 Mean   : 0.02522   Mean   :-0.02153
 3rd Qu.: 0.65414   3rd Qu.: 0.55738
 Max.   : 2.18471   Max.   : 1.70199
```

A useful way to detect patterns and anomalies in the data is through the exploratory data analysis with visualization. Visualization gives a succinct, holistic view of the data that may be difficult to grasp from the numbers and summaries alone. Variables *x* and *y* of the data frame `data` can instead be visualized in a scatterplot (Figure 3-5), which easily depicts the relationship between two variables. An important facet of the initial data exploration, visualization assesses data cleanliness and suggests potentially important relationships in the data prior to the model planning and building phases.

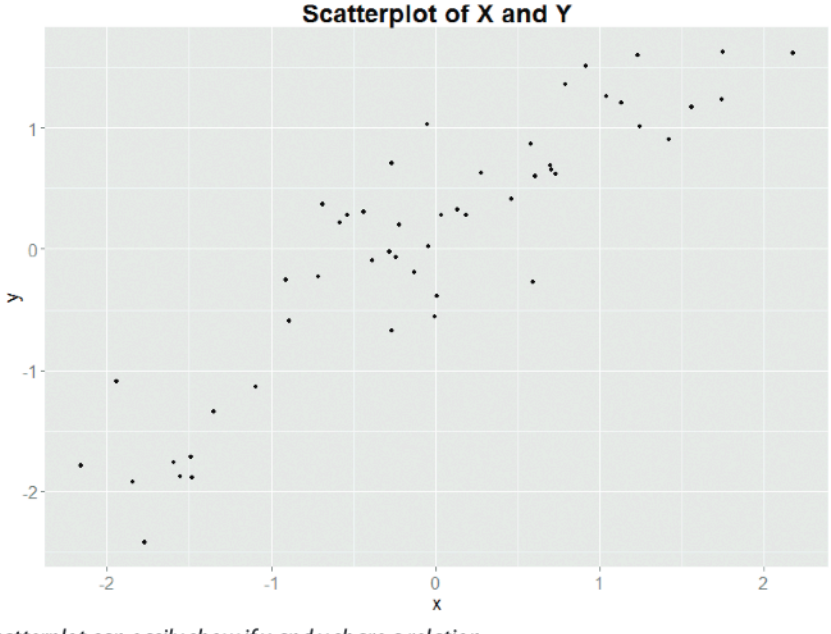


FIGURE 3-5 *A scatterplot can easily show if x and y share a relation*

The code to generate `data` as well as Figure 3-5 is shown next.

```
x <- rnorm(50)
y <- x + rnorm(50, mean=0, sd=0.5)

data <- as.data.frame(cbind(x, y))
```

```
summary(data)

library(ggplot2)
ggplot(data, aes(x=x, y=y)) +
  geom_point(size=2) +
  ggtitle("Scatterplot of X and Y") +
  theme(axis.text=element_text(size=12),
        axis.title = element_text(size=14),
        plot.title = element_text(size=20, face="bold"))
```

Exploratory data analysis [9] is a data analysis approach to reveal the important characteristics of a dataset, mainly through visualization. This section discusses how to use some basic visualization techniques and the plotting feature in R to perform exploratory data analysis.

3.2.1 Visualization Before Analysis

To illustrate the importance of visualizing data, consider Anscombe's quartet. Anscombe's quartet consists of four datasets, as shown in Figure 3-6. It was constructed by statistician Francis Anscombe [10] in 1973 to demonstrate the importance of graphs in statistical analyses.

# 1		# 2		# 3		# 4	
x	y	x	y	x	y	x	y
4	4.26	4	3.10	4	5.39	8	5.25
5	5.68	5	4.74	5	5.73	8	5.56
6	7.24	6	6.13	6	6.08	8	5.76
7	4.82	7	7.26	7	6.42	8	6.58
8	6.95	8	8.14	8	6.77	8	6.89
9	8.81	9	8.77	9	7.11	8	7.04
10	8.04	10	9.14	10	7.46	8	7.71
11	8.33	11	9.26	11	7.81	8	7.91
12	10.84	12	9.13	12	8.15	8	8.47
13	7.58	13	8.74	13	12.74	8	8.84
14	9.96	14	8.10	14	8.84	19	12.50

FIGURE 3-6 *Anscombe's quartet*

The four datasets in Anscombe's quartet have nearly identical statistical properties, as shown in Table 3-3.

TABLE 3-3 *Statistical Properties of Anscombe's Quartet*

Statistical Property	Value
Mean of *x*	9
Variance of *y*	11
Mean of *y*	7.50 (to 2 decimal points)

Variance of *y*	4.12 or 4.13 (to 2 decimal points)
Correlations between *x* and *y*	0.816
Linear regression line	$y = 3.00 + 0.50x$ (to 2 decimal points)

Based on the nearly identical statistical properties across each dataset, one might conclude that these four datasets are quite similar. However, the scatterplots in Figure 3-7 tell a different story. Each dataset is plotted as a scatterplot, and the fitted lines are the result of applying linear regression models. The estimated regression line fits Dataset 1 reasonably well. Dataset 2 is definitely nonlinear. Dataset 3 exhibits a linear trend, with one apparent outlier at $x = 13$. For Dataset 4, the regression line fits the dataset quite well. However, with only points at two x values, it is not possible to determine that the linearity assumption is proper.

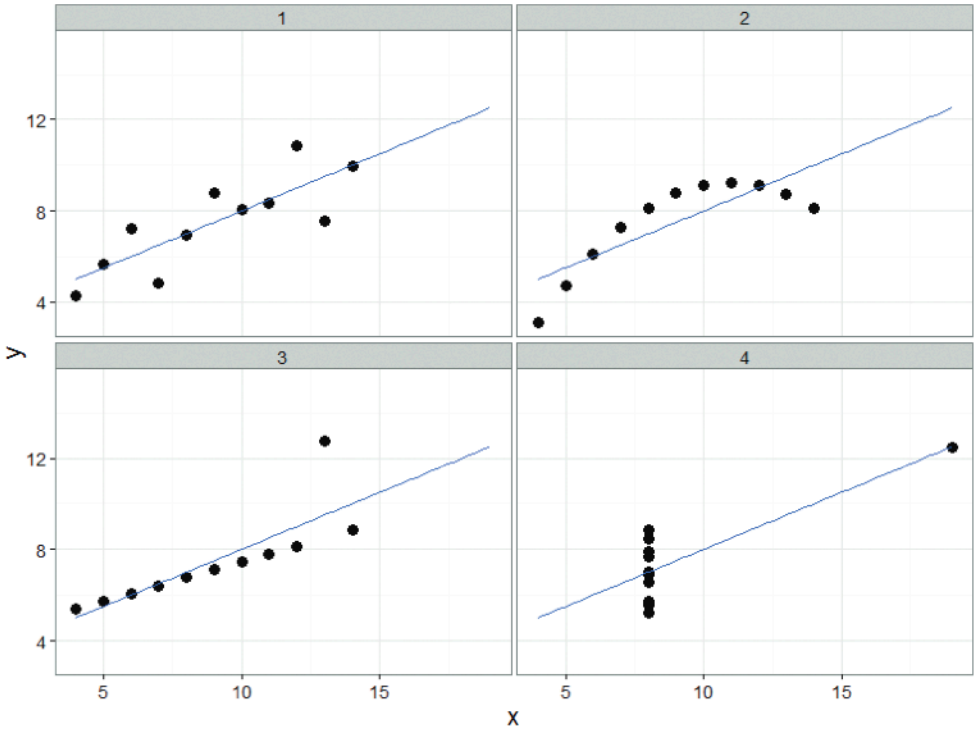


FIGURE 3-7 *Anscombe's quartet visualized as scatterplots*

The R code for generating Figure 3-7 is shown next. It requires the R package `ggplot2` [11], which can be installed simply by running the command `install.packages("ggplot2")`. The `anscombe`

dataset for the plot is included in the standard R distribution. Enter `data()` for a list of datasets included in the R base distribution. Enter `data(DatasetName)` to make a dataset available in the current workspace.

In the code that follows, variable *levels* is created using the `gl()` function, which generates factors of four levels (1, 2, 3, and 4), each repeating 11 times. Variable *mydata* is created using the `with(data, expression)` function, which evaluates an ***expression*** in an environment constructed from ***data***. In this example, the ***data*** is the `anscombe` dataset, which includes eight attributes: *x1*, *x2*, *x3*, *x4*, *y1*, *y2*, *y3*, and *y4*. The ***expression*** part in the code creates a data frame from the `anscombe` dataset, and it only includes three attributes: *x*, *y*, and the group each data point belongs to (*mygroup*).

```
install.packages("ggplot2") # not required if package has been installed

data(anscombe)     # load the anscombe dataset into the current workspace
anscombe
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
4   9  9  9  8  8.81 8.77  7.11  8.84
5  11 11 11  8  8.33 9.26  7.81  8.47
6  14 14 14  8  9.96 8.10  8.84  7.04
7   6  6  6  8  7.24 6.13  6.08  5.25
8   4  4  4 19  4.26 3.10  5.39 12.50
9  12 12 12  8 10.84 9.13  8.15  5.56
10  7  7  7  8  4.82 7.26  6.42  7.91
11  5  5  5  8  5.68 4.74  5.73  6.89

nrow(anscombe)     # number of rows
[1] 11

# generates levels to indicate which group each data point belongs to
levels <- gl(4, nrow(anscombe))
levels
[1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
[34] 4 4 4 4 4 4 4 4 4 4 4
Levels: 1 2 3 4

# Group anscombe into a data frame
mydata <- with(anscombe, data.frame(x=c(x1,x2,x3,x4), y=c(y1,y2,y3,y4),
               mygroup=levels))

mydata
    x     y mygroup
1  10  8.04       1
2   8  6.95       1
3  13  7.58       1
4   9  8.81       1
...
```

```
41 19 12.50       4
42  8  5.56       4
43  8  7.91       4
44  8  6.89       4

# Make scatterplots using the ggplot2 package
library(ggplot2)
theme_set(theme_bw())     # set plot color theme

# create the four plots of Figure 3-7
ggplot(mydata, aes(x,y)) +
      geom_point(size=4) +
      geom_smooth(method="lm", fill=NA, fullrange=TRUE) +
      facet_wrap(~mygroup)
```

3.2.2 Dirty Data

This section addresses how dirty data can be detected in the data exploration phase with visualizations. In general, analysts should look for anomalies, verify the data with domain knowledge, and decide the most appropriate approach to clean the data.

Consider a scenario in which a bank is conducting data analyses of its account holders to gauge customer retention. Figure 3-8 shows the age distribution of the account holders.

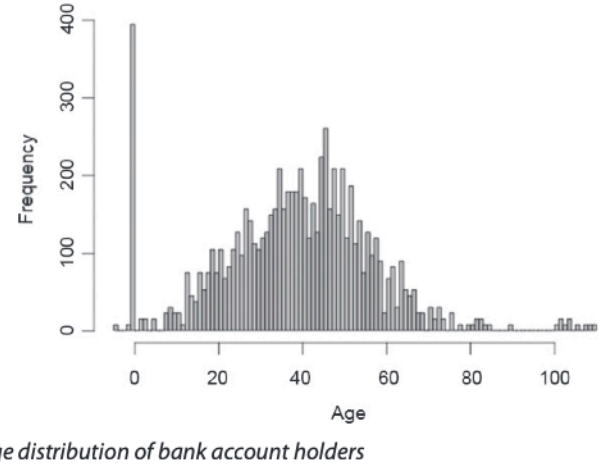


FIGURE 3-8 *Age distribution of bank account holders*

If the age data is in a vector called *age*, the graph can be created with the following R script:

```
hist(age, breaks=100, main="Age Distribution of Account Holders",
     xlab="Age", ylab="Frequency", col="gray")
```

The figure shows that the median age of the account holders is around 40. A few accounts with account holder age less than 10 are unusual but plausible. These could be custodial accounts or college savings accounts set up by the parents of young children. These accounts should be retained for future analyses.

However, the left side of the graph shows a huge spike of customers who are zero years old or have negative ages. This is likely to be evidence of ***missing data***. One possible explanation is that the null age values could have been replaced by 0 or negative values during the data input. Such an occurrence may be caused by entering age in a text box that only allows numbers and does not accept empty values. Or it might be caused by transferring data among several systems that have different definitions for null values (such as NULL, NA, 0, –1, or –2). Therefore, ***data cleansing*** needs to be performed over the accounts with abnormal age values. Analysts should take a closer look at the records to decide if the missing data should be eliminated or if an appropriate age value can be determined using other available information for each of the accounts.

In R, the `is.na()` function provides tests for missing values. The following example creates a vector *x* where the fourth value is not available (`NA`). The `is.na()` function returns `TRUE` at each `NA` value and `FALSE` otherwise.

```
x <- c(1, 2, 3, NA, 4)
is.na(x)
[1] FALSE FALSE FALSE TRUE FALSE
```

Some arithmetic functions, such as `mean()`, applied to data containing missing values can yield an `NA` result. To prevent this, set the `na.rm` parameter to `TRUE` to remove the missing value during the function's execution.

```
mean(x)
[1] NA
mean(x, na.rm=TRUE)
[1] 2.5
```

The `na.exclude()` function returns the object with incomplete cases removed.

```
DF <- data.frame(x = c(1, 2, 3), y = c(10, 20, NA))
DF
  x  y
1 1 10
2 2 20
3 3 NA

DF1 <- na.exclude(DF)
DF1
  x  y
1 1 10
2 2 20
```

Account holders older than 100 may be due to bad data caused by typos. Another possibility is that these accounts may have been passed down to the heirs of the original account holders without being updated. In this case, one needs to further examine the data and conduct data cleansing if necessary. The dirty data could be simply removed or filtered out with an age threshold for future analyses. If removing records is not an option, the analysts can look for patterns within the data and develop a set of heuristics to attack the problem of dirty data. For example, wrong age values could be replaced with ***approximation*** based on the nearest neighbor—the record that is the most similar to the record in question based on analyzing the differences in all the other variables besides age.

Figure 3-9 presents another example of dirty data. The distribution shown here corresponds to the age of mortgages in a bank's home loan portfolio. The mortgage age is calculated by subtracting the origination date of the loan from the current date. The vertical axis corresponds to the number of mortgages at each mortgage age.

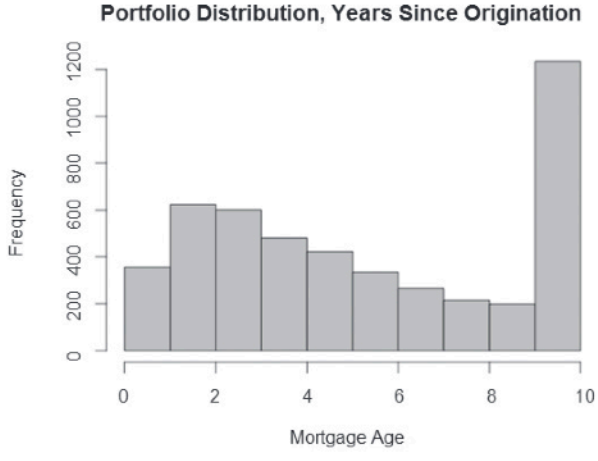


FIGURE 3-9 *Distribution of mortgage in years since origination from a bank's home loan portfolio*

If the data is in a vector called `mortgage`, Figure 3-9 can be produced by the following R script.

```
hist(mortgage, breaks=10, xlab="Mortgage Age", col="gray",
     main="Portfolio Distribution, Years Since Origination")
```

Figure 3-9 shows that the loans are no more than 10 years old, and these 10-year-old loans have a disproportionate frequency compared to the rest of the population. One possible explanation is that the 10-year-old loans do not only include loans originated 10 years ago, but also those originated earlier than that. In other words, the 10 in the *x*-axis actually means ≥ 10. This sometimes happens when data is ported from one system to another or because the data provider decided, for some reason, not to distinguish loans that are more than 10 years old. Analysts need to study the data further and decide the most appropriate way to perform data cleansing.

Data analysts should perform sanity checks against domain knowledge and decide if the dirty data needs to be eliminated. Consider the task to find out the probability of mortgage loan default. If the past observations suggest that most defaults occur before about the 4th year and 10-year-old mortgages rarely default, it may be safe to eliminate the dirty data and assume that the defaulted loans are less than 10 years old. For other analyses, it may become necessary to track down the source and find out the true origination dates.

Dirty data can occur due to acts of omission. In the `sales` data used at the beginning of this chapter, it was seen that the minimum number of orders was 1 and the minimum annual sales amount was $30.02. Thus, there is a strong possibility that the provided dataset did not include the sales data on all customers, just the customers who purchased something during the past year.

3.2.3 Visualizing a Single Variable

Using visual representations of data is a hallmark of exploratory data analyses: letting the data speak to its audience rather than imposing an interpretation on the data *a priori*. Sections 3.2.3 and 3.2.4 examine ways of displaying data to help explain the underlying distributions of a single variable or the relationships of two or more variables.

R has many functions available to examine a single variable. Some of these functions are listed in Table 3-4.

TABLE 3-4 *Example Functions for Visualizing a Single Variable*

Function	Purpose
`plot(data)`	Scatterplot where x is the index and y is the value; suitable for low-volume data
`barplot(data)`	Barplot with vertical or horizontal bars
`dotchart(data)`	Cleveland dot plot [12]
`hist(data)`	Histogram
`plot(density(data))`	Density plot (a continuous histogram)
`stem(data)`	Stem-and-leaf plot
`rug(data)`	Add a rug representation (1-d plot) of the data to an existing plot

Dotchart and Barplot

Dotchart and barplot portray continuous values with labels from a discrete variable. A dotchart can be created in R with the function `dotchart(x, label=...)`, where *x* is a numeric vector and *label* is a vector of categorical labels for *x*. A barplot can be created with the `barplot(height)` function, where *height* represents a vector or matrix. Figure 3-10 shows (a) a dotchart and (b) a barplot based on the `mtcars` dataset, which includes the fuel consumption and 10 aspects of automobile design and performance of 32 automobiles. This dataset comes with the standard R distribution.

The plots in Figure 3-10 can be produced with the following R code.

```
data(mtcars)
dotchart(mtcars$mpg,labels=row.names(mtcars),cex=.7,
         main="Miles Per Gallon (MPG) of Car Models",
         xlab="MPG")
barplot(table(mtcars$cyl), main="Distribution of Car Cylinder Counts",
          xlab="Number of Cylinders")
```

Histogram and Density Plot

Figure 3-11(a) includes a histogram of household income. The histogram shows a clear concentration of low household incomes on the left and the long tail of the higher incomes on the right.

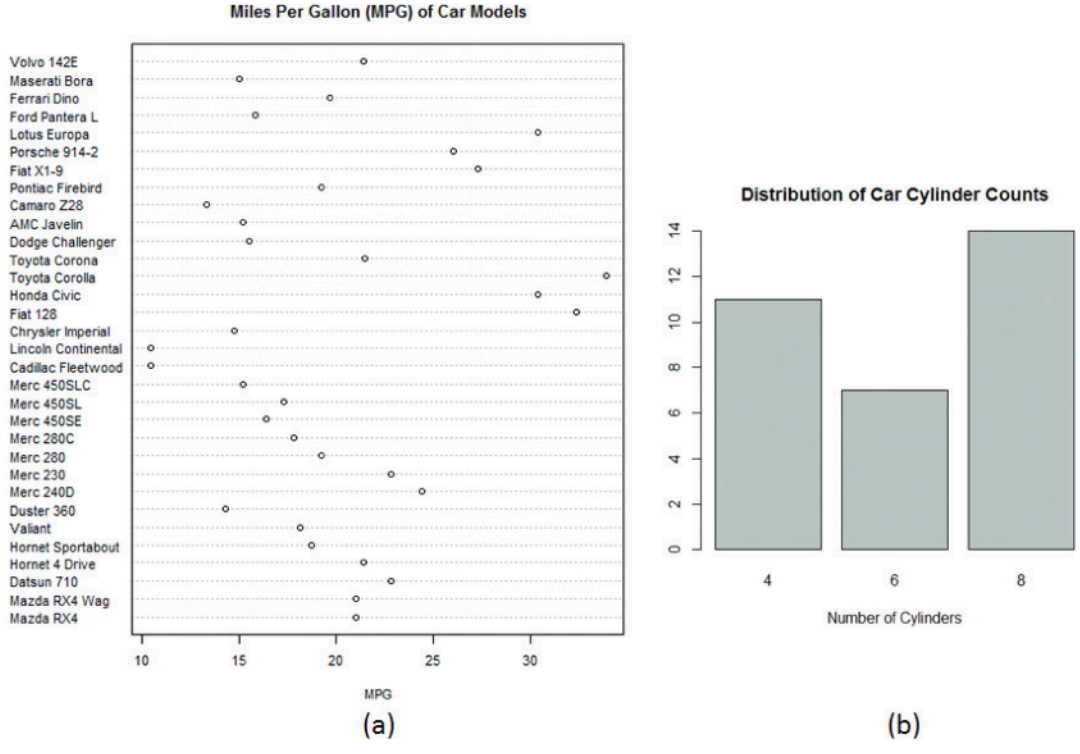


FIGURE 3-10 *(a) Dotchart on the miles per gallon of cars and (b) Barplot on the distribution of car cylinder counts*

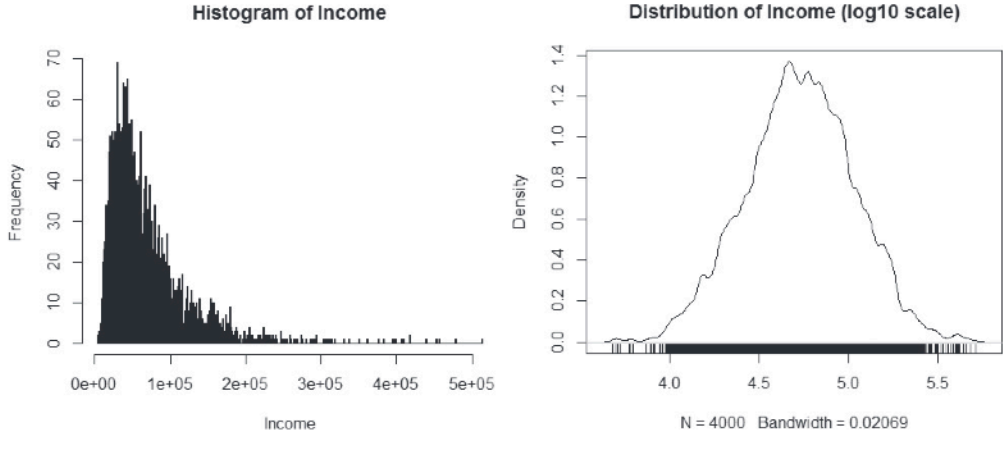


FIGURE 3-11 *(a) Histogram and (b) Density plot of household income*

Figure 3-11(b) shows a density plot of the logarithm of household income values, which emphasizes the distribution. The income distribution is concentrated in the center portion of the graph. The code to generate the two plots in Figure 3-11 is provided next. The `rug()` function creates a one-dimensional density plot on the bottom of the graph to emphasize the distribution of the observation.

```
# randomly generate 4000 observations from the log normal distribution
income <- rlnorm(4000, meanlog = 4, sdlog = 0.7)
summary(income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.301  33.720  54.970  70.320  88.800 659.800
income <- 1000*income
summary(income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4301   33720   54970   70320   88800  659800
# plot the histogram
hist(income, breaks=500, xlab="Income", main="Histogram of Income")
# density plot
plot(density(log10(income), adjust=0.5),
     main="Distribution of Income (log10 scale)")
# add rug to the density plot
rug(log10(income))
```

In the data preparation phase of the Data Analytics Lifecycle, the data range and distribution can be obtained. If the data is skewed, viewing the logarithm of the data (if it's all positive) can help detect structures that might otherwise be overlooked in a graph with a regular, nonlogarithmic scale.

When preparing the data, one should look for signs of dirty data, as explained in the previous section. Examining if the data is unimodal or multimodal will give an idea of how many distinct populations with different behavior patterns might be mixed into the overall population. Many modeling techniques assume that the data follows a normal distribution. Therefore, it is important to know if the available dataset can match that assumption before applying any of those modeling techniques.

Consider a density plot of diamond prices (in USD). Figure 3-12(a) contains two density plots for premium and ideal cuts of diamonds. The group of premium cuts is shown in red, and the group of ideal cuts is shown in blue. The range of diamond prices is wide—in this case ranging from around $300 to almost $20,000. Extreme values are typical of monetary data such as income, customer value, tax liabilities, and bank account sizes.

Figure 3-12(b) shows more detail of the diamond prices than Figure 3-12(a) by taking the logarithm. The two humps in the premium cut represent two distinct groups of diamond prices: One group centers around $\log_{10}$ price = 2.9 (where the price is about $794), and the other centers around $\log_{10}$ price = 3.7 (where the price is about $5,012). The ideal cut contains three humps, centering around 2.9, 3.3, and 3.7 respectively.

The R script to generate the plots in Figure 3-12 is shown next. The *`diamonds`* dataset comes with the `ggplot2` package.

```
library("ggplot2")
data(diamonds)    # load the diamonds dataset from ggplot2

# Only keep the premium and ideal cuts of diamonds
```

```
niceDiamonds <- diamonds[diamonds$cut=="Premium" |
                         diamonds$cut=="Ideal",]

summary(niceDiamonds$cut)
     Fair      Good Very Good   Premium     Ideal
        0         0         0     13791     21551

# plot density plot of diamond prices
ggplot(niceDiamonds, aes(x=price, fill=cut)) +
  geom_density(alpha = .3, color=NA)

# plot density plot of the log10 of diamond prices
ggplot(niceDiamonds, aes(x=log10(price), fill=cut)) +
  geom_density(alpha = .3, color=NA)
```

As an alternative to `ggplot2`, the `lattice` package provides a function called `densityplot()` for making simple density plots.

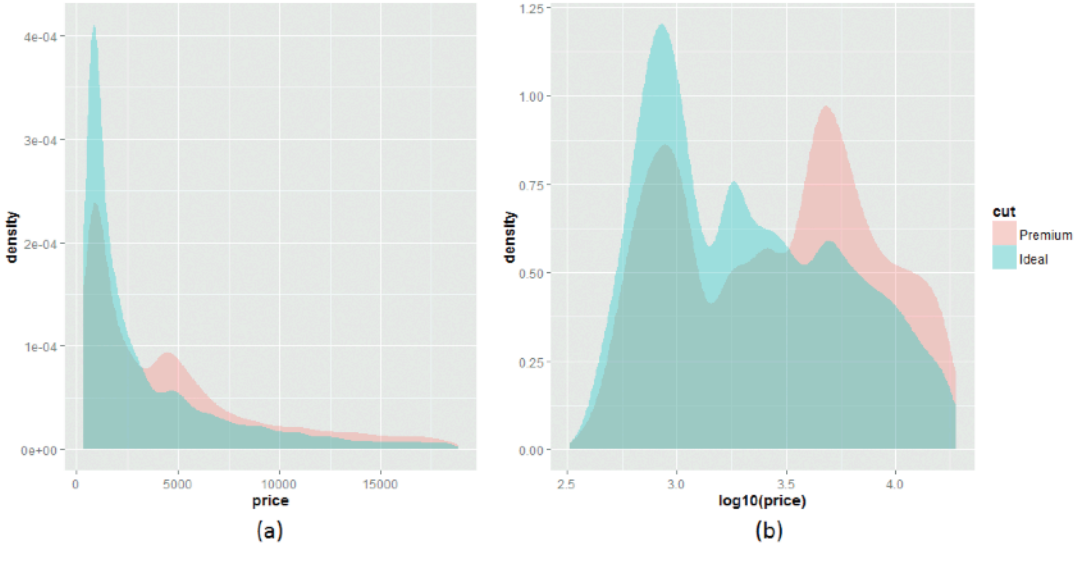


FIGURE 3-12 *Density plots of (a) diamond prices and (b) the logarithm of diamond prices*

3.2.4 Examining Multiple Variables

A scatterplot (shown previously in Figure 3-1 and Figure 3-5) is a simple and widely used visualization for finding the relationship among multiple variables. A scatterplot can represent data with up to five variables using x-axis, y-axis, size, color, and shape. But usually only two to four variables are portrayed in a scatterplot to minimize confusion. When examining a scatterplot, one needs to pay close attention

to the possible relationship between the variables. If the functional relationship between the variables is somewhat pronounced, the data may roughly lie along a straight line, a parabola, or an exponential curve. If variable y is related exponentially to x, then the plot of x versus `log(y)` is approximately linear. If the plot looks more like a cluster without a pattern, the corresponding variables may have a weak relationship.

The scatterplot in Figure 3-13 portrays the relationship of two variables: x and y. The red line shown on the graph is the fitted line from the linear regression. Linear regression will be revisited in Chapter 6, "Advanced Analytical Theory and Methods: Regression." Figure 3-13 shows that the regression line does not fit the data well. This is a case in which linear regression cannot model the relationship between the variables. Alternative methods such as the `loess()` function can be used to fit a nonlinear line to the data. The blue curve shown on the graph represents the LOESS curve, which fits the data better than linear regression.

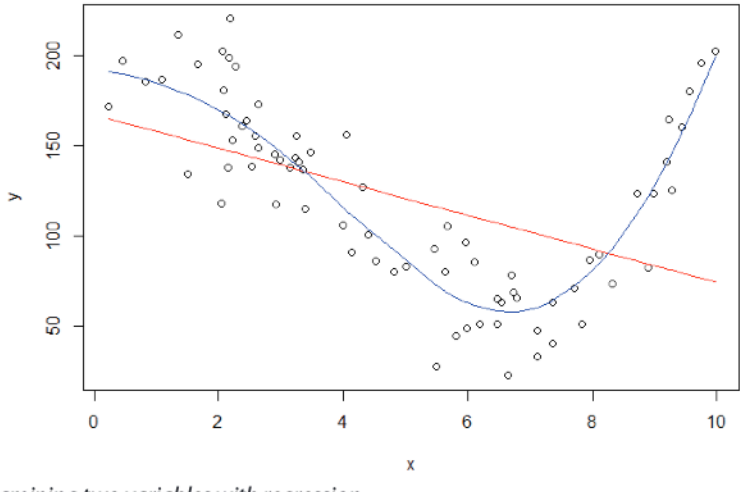


FIGURE 3-13 *Examining two variables with regression*

The R code to produce Figure 3-13 is as follows. The `runif(75,0,10)` generates 75 numbers between 0 to 10 with random deviates, and the numbers conform to the uniform distribution. The `rnorm(75,0,20)` generates 75 numbers that conform to the normal distribution, with the mean equal to 0 and the standard deviation equal to 20. The `points()` function is a generic function that draws a sequence of points at the specified coordinates. Parameter `type="l"` tells the function to draw a solid line. The `col` parameter sets the color of the line, where 2 represents the red color and 4 represents the blue color.

```
# 75 numbers between 0 and 10 of uniform distribution
x <- runif(75, 0, 10)

x <- sort(x)
y <- 200 + x^3 - 10 * x^2 + x + rnorm(75, 0, 20)


lr <- lm(y ~ x)        # linear regression
poly <- loess(y ~ x)   # LOESS
```

```
fit <- predict(poly)  # fit a nonlinear line

plot(x,y)

# draw the fitted line for the linear regression
points(x, lr$coefficients[1] + lr$coefficients[2] * x,
       type = "l", col = 2)

# draw the fitted line with LOESS
points(x, fit, type = "l", col = 4)
```

Dotchart and Barplot

Dotchart and barplot from the previous section can visualize multiple variables. Both of them use color as an additional dimension for visualizing the data.

For the same `mtcars` dataset, Figure 3-14 shows a dotchart that groups vehicle cylinders at the y-axis and uses colors to distinguish different cylinders. The vehicles are sorted according to their MPG values. The code to generate Figure 3-14 is shown next.

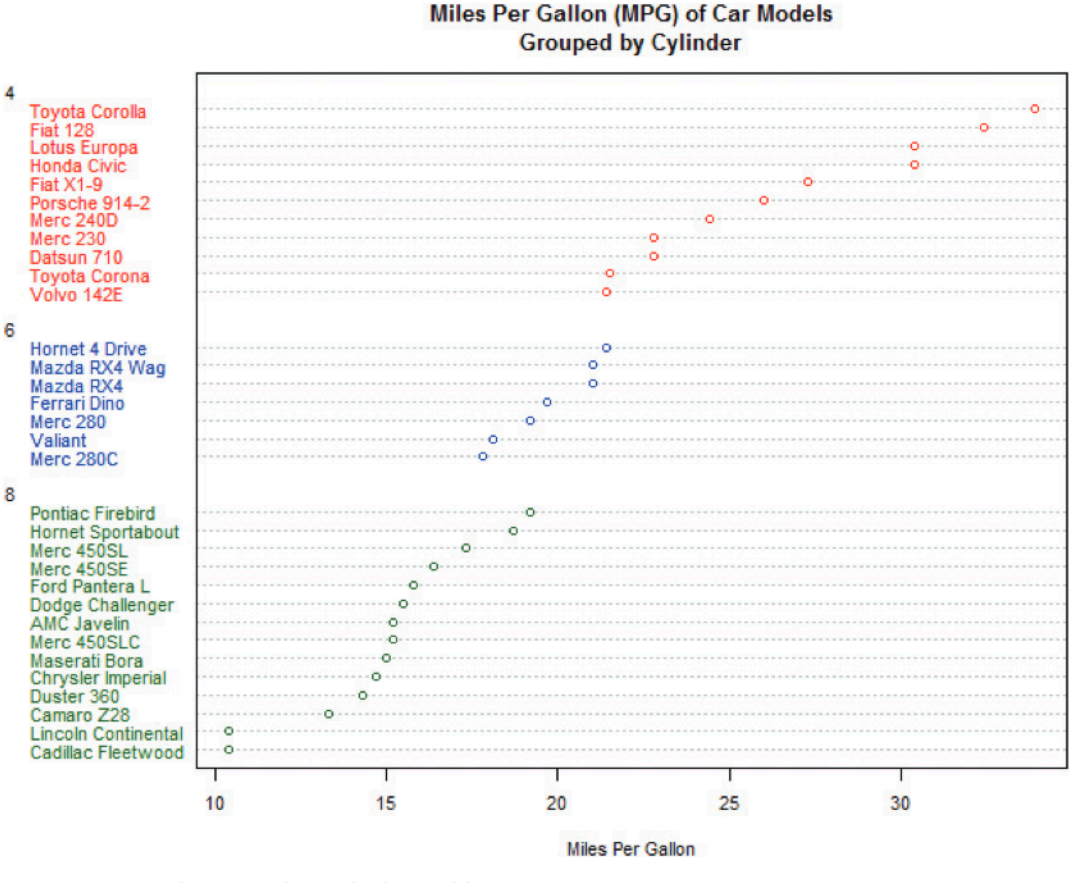


FIGURE 3-14 *Dotplot to visualize multiple variables*

```
# sort by mpg
cars <- mtcars[order(mtcars$mpg),]

# grouping variable must be a factor
cars$cyl <- factor(cars$cyl)

cars$color[cars$cyl==4] <- "red"
cars$color[cars$cyl==6] <- "blue"
cars$color[cars$cyl==8] <- "darkgreen"

dotchart(cars$mpg, labels=row.names(cars), cex=.7, groups= cars$cyl,
       main="Miles Per Gallon (MPG) of Car Models\nGrouped by Cylinder",
       xlab="Miles Per Gallon", color=cars$color, gcolor="black")
```

The barplot in Figure 3-15 visualizes the distribution of car cylinder counts and number of gears. The x-axis represents the number of cylinders, and the color represents the number of gears. The code to generate Figure 3-15 is shown next.

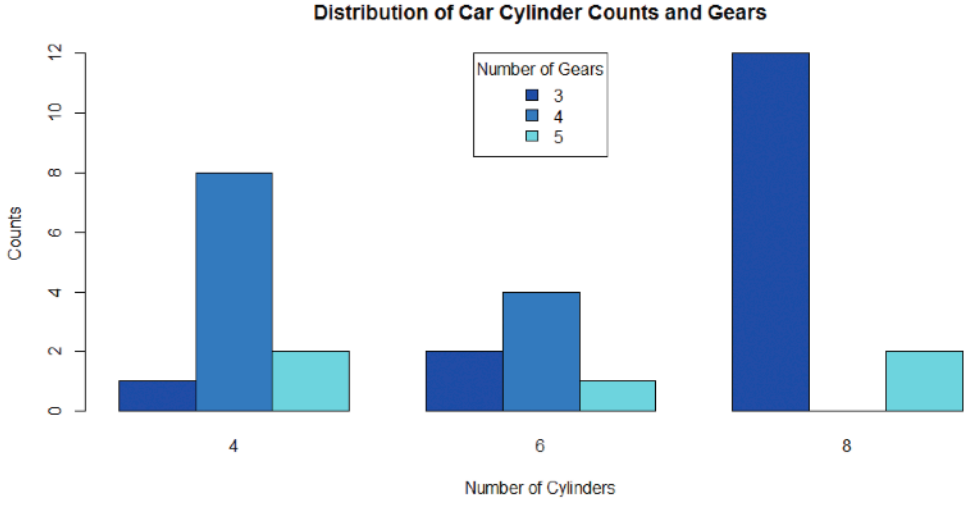


FIGURE 3-15 *Barplot to visualize multiple variables*

```
counts <- table(mtcars$gear, mtcars$cyl)
barplot(counts, main="Distribution of Car Cylinder Counts and Gears",
        xlab="Number of Cylinders", ylab="Counts",
        col=c("#0000FFFF", "#0080FFFF", "#00FFFFFF"),
        legend = rownames(counts), beside=TRUE,
        args.legend = list(x="top", title = "Number of Gears"))
```

Box-and-Whisker Plot

Box-and-whisker plots show the distribution of a continuous variable for each value of a discrete variable. The box-and-whisker plot in Figure 3-16 visualizes mean household incomes as a function of region in the United States. The first digit of the U.S. postal ("ZIP") code corresponds to a geographical region in the United States. In Figure 3-16, each data point corresponds to the mean household income from a particular zip code. The horizontal axis represents the first digit of a zip code, ranging from 0 to 9, where 0 corresponds to the northeast region of the United States (such as Maine, Vermont, and Massachusetts), and 9 corresponds to the southwest region (such as California and Hawaii). The vertical axis represents the logarithm of mean household incomes. The logarithm is taken to better visualize the distribution of the mean household incomes.

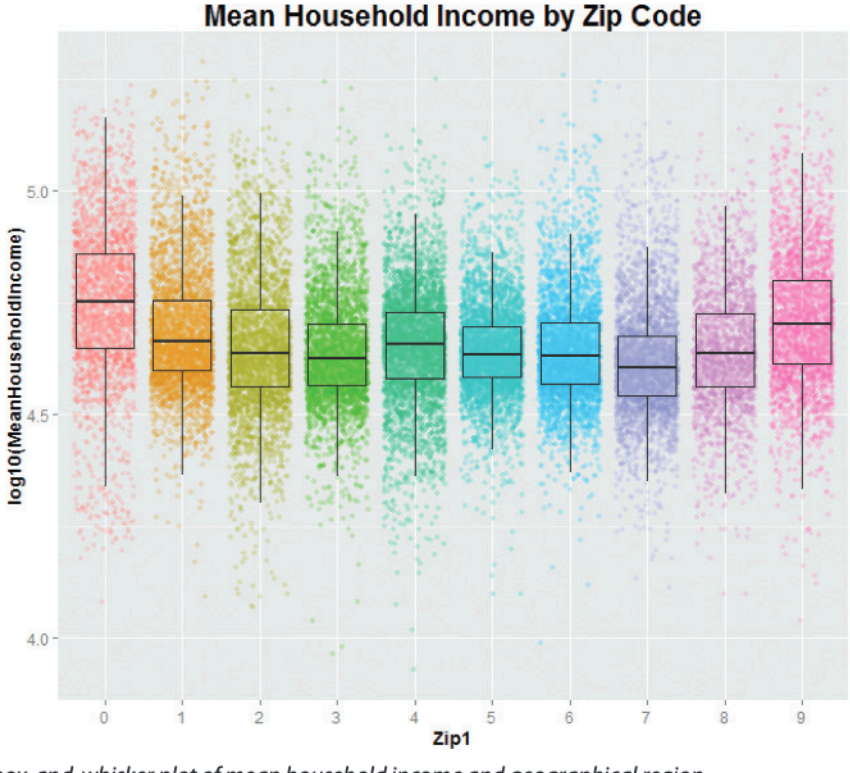


FIGURE 3-16 *A box-and-whisker plot of mean household income and geographical region*

In this figure, the scatterplot is displayed beneath the box-and-whisker plot, with some jittering for the overlap points so that each line of points widens into a strip. The "box" of the box-and-whisker shows the range that contains the central 50% of the data, and the line inside the box is the location of the median value. The upper and lower hinges of the boxes correspond to the first and third quartiles of the data. The upper whisker extends from the hinge to the highest value that is within 1.5 * IQR of the hinge. The lower whisker extends from the hinge to the lowest value within 1.5 * IQR of the hinge. IQR is the inter-quartile range, as discussed in Section 3.1.4. The points outside the whiskers can be considered possible outliers.

The graph shows how household income varies by region. The highest median incomes are in region 0 and region 9. Region 0 is slightly higher, but the boxes for the two regions overlap enough that the difference between the two regions probably is not significant. The lowest household incomes tend to be in region 7, which includes states such as Louisiana, Arkansas, and Oklahoma.

Assuming a data frame called *DF* contains two columns (*MeanHouseholdIncome* and *Zip1*), the following R script uses the `ggplot2` library [11] to plot a graph that is similar to Figure 3-16.

```
library(ggplot2)
# plot the jittered scatterplot w/ boxplot
# color-code points with zip codes
# the outlier.size=0 prevents the boxplot from plotting the outlier
ggplot(data=DF, aes(x=as.factor(Zip1), y=log10(MeanHouseholdIncome))) +
  geom_point(aes(color=factor(Zip1)), alpha=0.2, position="jitter") +
  geom_boxplot(outlier.size=0, alpha=0.1) +
  guides(colour=FALSE) +
  ggtitle ("Mean Household Income by Zip Code")
```

Alternatively, one can create a simple box-and-whisker plot with the `boxplot()` function provided by the R base package.

Hexbinplot for Large Datasets

This chapter has shown that scatterplot as a popular visualization can visualize data containing one or more variables. But one should be careful about using it on high-volume data. If there is too much data, the structure of the data may become difficult to see in a scatterplot. Consider a case to compare the logarithm of household income against the years of education, as shown in Figure 3-17. The cluster in the scatterplot on the left (a) suggests a somewhat linear relationship of the two variables. However, one cannot really see the structure of how the data is distributed inside the cluster. This is a Big Data type of problem. Millions or billions of data points would require different approaches for exploration, visualization, and analysis.

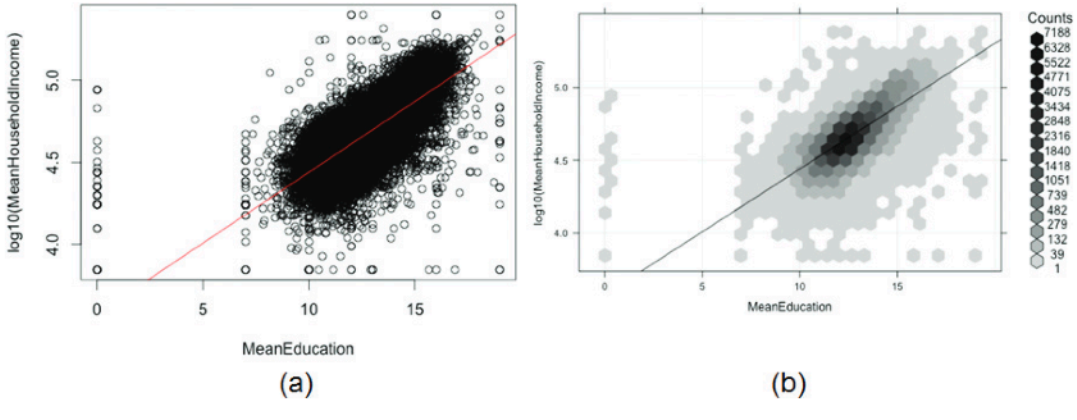


FIGURE 3-17 *(a) Scatterplot and (b) Hexbinplot of household income against years of education*

Although color and transparency can be used in a scatterplot to address this issue, a hexbinplot is sometimes a better alternative. A hexbinplot combines the ideas of scatterplot and histogram. Similar to a scatterplot, a hexbinplot visualizes data in the *x*-axis and *y*-axis. Data is placed into hexbins, and the third dimension uses shading to represent the concentration of data in each hexbin.

In Figure 3-17(b), the same data is plotted using a hexbinplot. The hexbinplot shows that the data is more densely clustered in a streak that runs through the center of the cluster, roughly along the regression line. The biggest concentration is around 12 years of education, extending to about 15 years.

In Figure 3-17, note the outlier data at `MeanEducation=0`. These data points may correspond to some missing data that needs further cleansing.

Assuming the two variables *`MeanHouseholdIncome`* and *`MeanEducation`* are from a data frame named *`zcta`*, the scatterplot of Figure 3-17(a) is plotted by the following R code.

```
# plot the data points
plot(log10(MeanHouseholdIncome) ~ MeanEducation, data=zcta)
# add a straight fitted line of the linear regression
abline(lm(log10(MeanHouseholdIncome) ~ MeanEducation, data=zcta), col='red')
```

Using the *`zcta`* data frame, the hexbinplot of Figure 3-17(b) is plotted by the following R code. Running the code requires the use of the `hexbin` package, which can be installed by running `install.packages("hexbin")`.

```
library(hexbin)
# "g" adds the grid, "r" adds the regression line
# sqrt transform on the count gives more dynamic range to the shading
# inv provides the inverse transformation function of trans
hexbinplot(log10(MeanHouseholdIncome) ~ MeanEducation,
  data=zcta, trans = sqrt, inv = function(x) x^2, type=c("g", "r"))
```

Scatterplot Matrix

A scatterplot matrix shows many scatterplots in a compact, side-by-side fashion. The scatterplot matrix, therefore, can visually represent multiple attributes of a dataset to explore their relationships, magnify differences, and disclose hidden patterns.

Fisher's *`iris`* dataset [13] includes the measurements in centimeters of the sepal length, sepal width, petal length, and petal width for 50 flowers from three species of iris. The three species are *setosa*, *versicolor*, and *virginica*. The iris dataset comes with the standard R distribution.

In Figure 3-18, all the variables of Fisher's iris dataset (sepal length, sepal width, petal length, and petal width) are compared in a scatterplot matrix. The three different colors represent three species of iris flowers. The scatterplot matrix in Figure 3-18 allows its viewers to compare the differences across the iris species for any pairs of attributes.

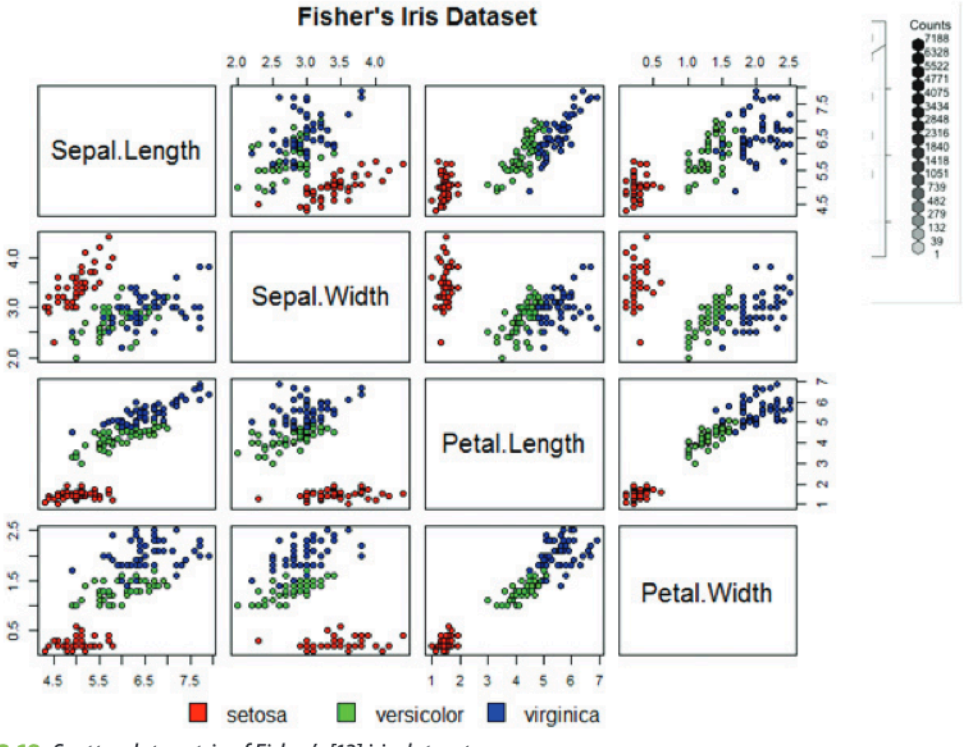


FIGURE 3-18 *Scatterplot matrix of Fisher's [13] iris dataset*

Consider the scatterplot from the first row and third column of Figure 3-18, where sepal length is compared against petal length. The horizontal axis is the petal length, and the vertical axis is the sepal length. The scatterplot shows that *versicolor* and *virginica* share similar sepal and petal lengths, although the latter has longer petals. The petal lengths of all *setosa* are about the same, and the petal lengths are remarkably shorter than the other two species. The scatterplot shows that for *versicolor* and *virginica*, sepal length grows linearly with the petal length.

The R code for generating the scatterplot matrix is provided next.

```
# define the colors
colors <- c("red", "green", "blue")

# draw the plot matrix
pairs(iris[1:4], main = "Fisher's Iris Dataset",
      pch = 21, bg = colors[unclass(iris$Species)] )

# set graphical parameter to clip plotting to the figure region
par(xpd = TRUE)

# add legend
legend(0.2, 0.02, horiz = TRUE, as.vector(unique(iris$Species)),
       fill = colors, bty = "n")
```

The vector `colors` defines the color scheme for the plot. It could be changed to something like `colors <- c("gray50", "white", "black")` to make the scatterplots grayscale.

Analyzing a Variable over Time

Visualizing a variable over time is the same as visualizing any pair of variables, but in this case the goal is to identify time-specific patterns.

Figure 3-19 plots the monthly total numbers of international airline passengers (in thousands) from January 1940 to December 1960. Enter `plot(AirPassengers)` in the R console to obtain a similar graph. The plot shows that, for each year, a large peak occurs mid-year around July and August, and a small peak happens around the end of the year, possibly due to the holidays. Such a phenomenon is referred to as a ***seasonality effect***.

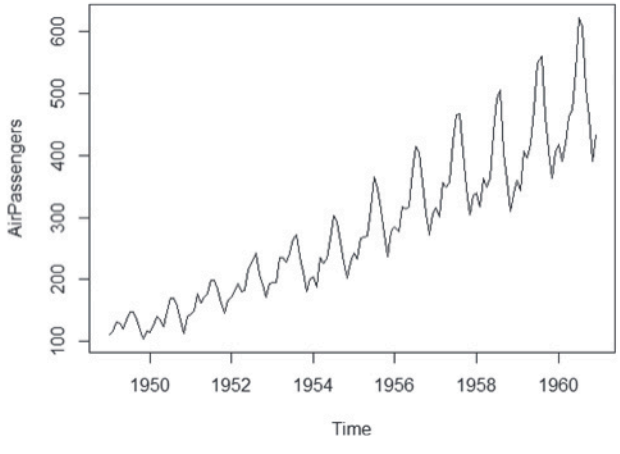


FIGURE 3-19 *Airline passenger counts from 1949 to 1960*

Additionally, the overall trend is that the number of air passengers steadily increased from 1949 to 1960. Chapter 8, "Advanced Analytical Theory and Methods: Time Series Analysis," discusses the analysis of such datasets in greater detail.

3.2.5 Data Exploration Versus Presentation

Using visualization for data exploration is different from presenting results to stakeholders. Not every type of plot is suitable for all audiences. Most of the plots presented earlier try to detail the data as clearly as possible for data scientists to identify structures and relationships. These graphs are more technical in nature and are better suited to technical audiences such as data scientists. Nontechnical stakeholders, however, generally prefer simple, clear graphics that focus on the message rather than the data.

Figure 3-20 shows the density plot on the distribution of account values from a bank. The data has been converted to the $\log_{10}$ scale. The plot includes a rug on the bottom to show the distribution of the variable. This graph is more suitable for data scientists and business analysts because it provides information that

can be relevant to the downstream analysis. The graph shows that the transformed account values follow an approximate normal distribution, in the range from $100 to $10,000,000. The median account value is approximately $30,000 ($10^{4.5}$), with the majority of the accounts between $1,000 ($10^3$) and $1,000,000 ($10^6$).

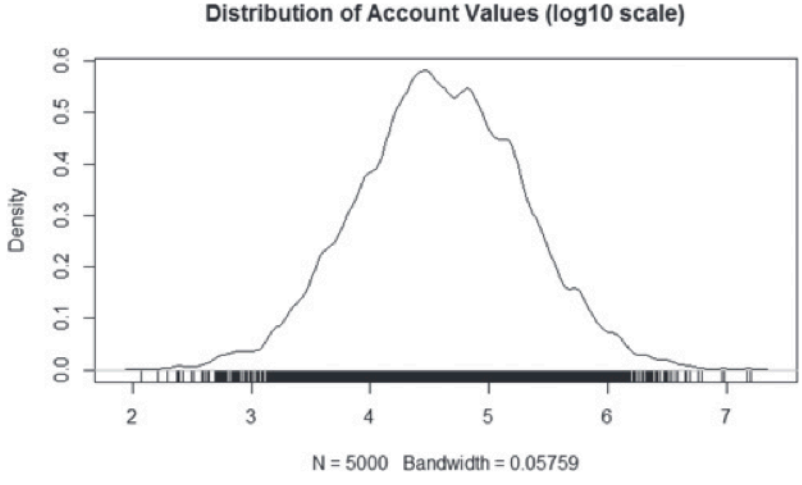


FIGURE 3-20 *Density plots are better to show to data scientists*

Density plots are fairly technical, and they contain so much information that they would be difficult to explain to less technical stakeholders. For example, it would be challenging to explain why the account values are in the $\log_{10}$ scale, and such information is not relevant to stakeholders. The same message can be conveyed by partitioning the data into log-like bins and presenting it as a histogram. As can be seen in Figure 3-21, the bulk of the accounts are in the $1,000–1,000,000 range, with the peak concentration in the $10–50K range, extending to $500K. This portrayal gives the stakeholders a better sense of the customer base than the density plot shown in Figure 3-20.

Note that the bin sizes should be carefully chosen to avoid distortion of the data. In this example, the bins in Figure 3-21 are chosen based on observations from the density plot in Figure 3-20. Without the density plot, the peak concentration might be just due to the somewhat arbitrary appearing choices for the bin sizes.

This simple example addresses the different needs of two groups of audience: analysts and stakeholders. Chapter 12, "The Endgame, or Putting It All Together," further discusses the best practices of delivering presentations to these two groups.

Following is the R code to generate the plots in Figure 3-20 and Figure 3-21.

```
# Generate random log normal income data
income = rlnorm(5000, meanlog=log(40000), sdlog=log(5))

# Part I: Create the density plot
plot(density(log10(income), adjust=0.5),
     main="Distribution of Account Values (log10 scale)")
# Add rug to the density plot
```

```
rug(log10(income))

# Part II: Make the histogram
# Create "log-like bins"
breaks = c(0, 1000, 5000, 10000, 50000, 100000, 5e5, 1e6, 2e7)
# Create bins and label the data
bins = cut(income, breaks, include.lowest=T,
           labels = c("< 1K", "1-5K", "5-10K", "10-50K",
                      "50-100K", "100-500K", "500K-1M", "> 1M"))
# Plot the bins
plot(bins, main = "Distribution of Account Values",
     xlab = "Account value ($ USD)",
     ylab = "Number of Accounts", col="blue")
```

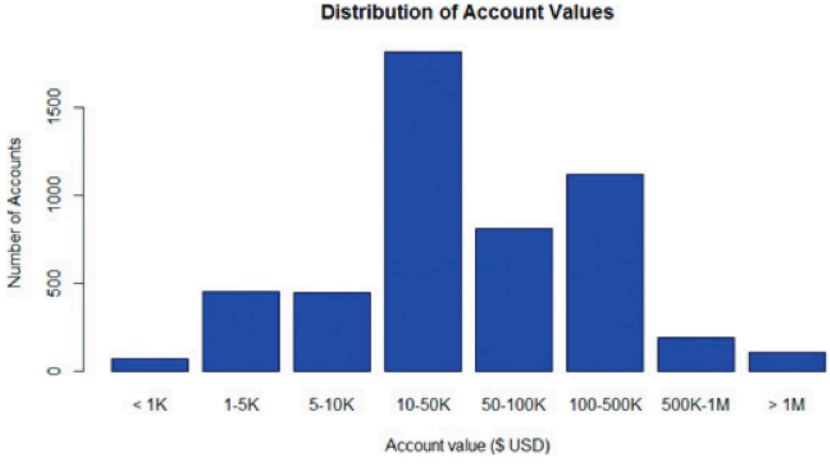


FIGURE 3-21 *Histograms are better to show to stakeholders*

3.3 Statistical Methods for Evaluation

Visualization is useful for data exploration and presentation, but statistics is crucial because it may exist throughout the entire Data Analytics Lifecycle. Statistical techniques are used during the initial data exploration and data preparation, model building, evaluation of the final models, and assessment of how the new models improve the situation when deployed in the field. In particular, statistics can help answer the following questions for data analytics:

- Model Building and Planning
  - What are the best input variables for the model?
  - Can the model predict the outcome given the input?

- Model Evaluation
  - Is the model accurate?
  - Does the model perform better than an obvious guess?
  - Does the model perform better than another candidate model?
- Model Deployment
  - Is the prediction sound?
  - Does the model have the desired effect (such as reducing the cost)?

This section discusses some useful statistical tools that may answer these questions.

3.3.1 Hypothesis Testing

When comparing populations, such as testing or evaluating the difference of the means from two samples of data (Figure 3-22), a common technique to assess the difference or the significance of the difference is ***hypothesis testing***.

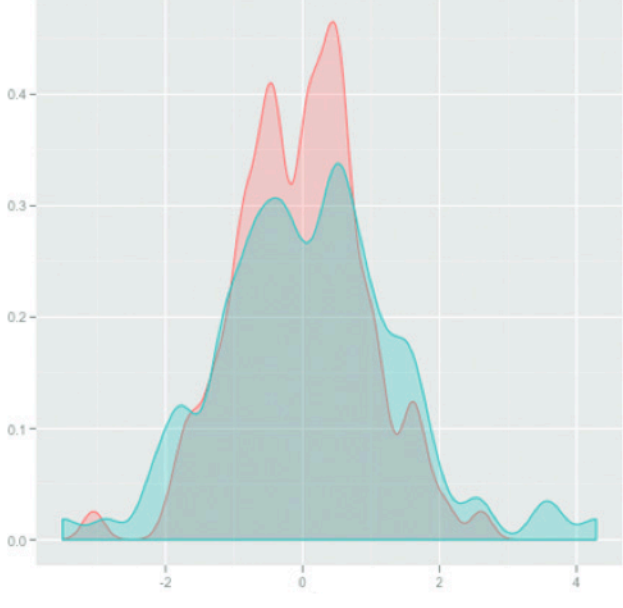


FIGURE 3-22 *Distributions of two samples of data*

The basic concept of hypothesis testing is to form an assertion and test it with data. When performing hypothesis tests, the common assumption is that there is no difference between two samples. This assumption is used as the default position for building the test or conducting a scientific experiment. Statisticians refer to this as the ***null hypothesis*** (H_0). The ***alternative hypothesis*** (H_A) is that there is a

difference between two samples. For example, if the task is to identify the effect of drug A compared to drug B on patients, the null hypothesis and alternative hypothesis would be this.

- H_0: Drug A and drug B have the same effect on patients.
- H_A: Drug A has a greater effect than drug B on patients.

If the task is to identify whether advertising Campaign C is effective on reducing customer churn, the null hypothesis and alternative hypothesis would be as follows.

- H_0: Campaign C does not reduce customer churn better than the current campaign method.
- H_A: Campaign C does reduce customer churn better than the current campaign.

It is important to state the null hypothesis and alternative hypothesis, because misstating them is likely to undermine the subsequent steps of the hypothesis testing process. A hypothesis test leads to either rejecting the null hypothesis in favor of the alternative or not rejecting the null hypothesis.

Table 3-5 includes some examples of null and alternative hypotheses that should be answered during the analytic lifecycle.

TABLE 3-5 *Example Null Hypotheses and Alternative Hypotheses*

Application	Null Hypothesis	Alternative Hypothesis
Accuracy Forecast	Model X *does not predict* better than the existing model.	Model X *predicts* better than the existing model.
Recommendation Engine	Algorithm Y *does not produce* better recommendations than the current algorithm being used.	Algorithm Y *produces* better recommendations than the current algorithm being used.
Regression Modeling	This variable *does not affect* the outcome because its coefficient is *zero*.	This variable *affects* outcome because its coefficient is not *zero*.

Once a model is built over the training data, it needs to be evaluated over the testing data to see if the proposed model predicts better than the existing model currently being used. The null hypothesis is that the proposed model does not predict better than the existing model. The alternative hypothesis is that the proposed model indeed predicts better than the existing model. In accuracy forecast, the null model could be that the sales of the next month are the same as the prior month. The hypothesis test needs to evaluate if the proposed model provides a better prediction. Take a recommendation engine as an example. The null hypothesis could be that the new algorithm does not produce better recommendations than the current algorithm being deployed. The alternative hypothesis is that the new algorithm produces better recommendations than the old algorithm.

When evaluating a model, sometimes it needs to be determined if a given input variable improves the model. In regression analysis (Chapter 6), for example, this is the same as asking if the regression coefficient for a variable is zero. The null hypothesis is that the coefficient is zero, which means the variable does not have an impact on the outcome. The alternative hypothesis is that the coefficient is nonzero, which means the variable does have an impact on the outcome.

A common hypothesis test is to compare the means of two populations. Two such hypothesis tests are discussed in Section 3.3.2.

3.3.2 Difference of Means

Hypothesis testing is a common approach to draw inferences on whether or not the two populations, denoted *pop*1 and *pop*2, are different from each other. This section provides two hypothesis tests to compare the means of the respective populations based on samples randomly drawn from each population. Specifically, the two hypothesis tests in this section consider the following null and alternative hypotheses.

- H_0: $\mu_1 = \mu_2$
- H_A: $\mu_1 \neq \mu_2$

The μ_1 and μ_2 denote the population means of *pop*1 and *pop*2, respectively.

The basic testing approach is to compare the observed sample means, $\bar{X}_1$ and $\bar{X}_2$, corresponding to each population. If the values of $\bar{X}_1$ and $\bar{X}_2$ are approximately equal to each other, the distributions of $\bar{X}_1$ and $\bar{X}_2$ overlap substantially (Figure 3-23), and the null hypothesis is supported. A large observed difference between the sample means indicates that the null hypothesis should be rejected. Formally, the difference in means can be tested using Student's *t*-test or the Welch's *t*-test.

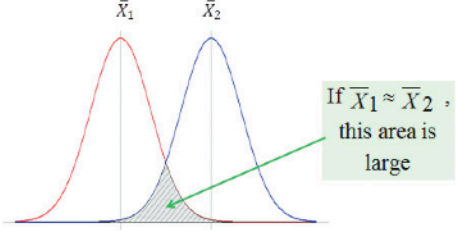


FIGURE 3-23 *Overlap of the two distributions is large if* $\bar{X}_1 \approx \bar{X}_2$

Student's t-test

Student's *t*-test assumes that distributions of the two populations have equal but unknown variances. Suppose n_1 and n_2 samples are randomly and independently selected from two populations, *pop*1 and *pop*2, respectively. If each population is normally distributed with the same mean ($\mu_1 = \mu_2$) and with the same variance, then *T* (the ***t-statistic***), given in Equation 3-1, follows a ***t-distribution*** with $n_1 + n_2 - 2$ ***degrees of freedom (df).***

$$T = \frac{\bar{X}_1 - \bar{X}_2}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

where

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} \tag{3-1}$$

The shape of the *t*-distribution is similar to the normal distribution. In fact, as the degrees of freedom approaches 30 or more, the *t*-distribution is nearly identical to the normal distribution. Because the numerator of *T* is the difference of the sample means, if the observed value of *T* is far enough from zero such that the probability of observing such a value of *T* is unlikely, one would reject the null hypothesis that the population means are equal. Thus, for a small probability, say $\alpha = 0.05$, T^* is determined such that $P(|T| \geq T^*) = 0.05$. After the samples are collected and the observed value of *T* is calculated according to Equation 3-1, the null hypothesis ($\mu_1 = \mu_2$) is rejected if $|T| \geq T^*$.

In hypothesis testing, in general, the small probability, α, is known as the ***significance level*** of the test. The significance level of the test is the probability of rejecting the null hypothesis, when the null hypothesis is actually TRUE. In other words, for $\alpha = 0.05$, if the means from the two populations are truly equal, then in repeated random sampling, the observed magnitude of *T* would only exceed T^* 5% of the time.

In the following R code example, 10 observations are randomly selected from two normally distributed populations and assigned to the variables x and y. The two populations have a mean of 100 and 105, respectively, and a standard deviation equal to 5. Student's *t*-test is then conducted to determine if the obtained random samples support the rejection of the null hypothesis.

```
# generate random observations from the two populations
x <- rnorm(10, mean=100, sd=5)    # normal distribution centered at 100
y <- rnorm(20, mean=105, sd=5)    # normal distribution centered at 105

t.test(x, y, var.equal=TRUE)      # run the Student's t-test
Two Sample t-test

data:  x and y
t = -1.7828, df = 28, p-value = 0.08547
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -6.1611557  0.4271893
sample estimates:
  mean of x mean of y
102.2136  105.0806
```

From the R output, the observed value of *T* is $t = -1.7828$. The negative sign is due to the fact that the sample mean of x is less than the sample mean of y. Using the `qt()` function in R, a *T* value of 2.0484 corresponds to a 0.05 significance level.

```
# obtain t value for a two-sided test at a 0.05 significance level
qt(p=0.05/2, df=28, lower.tail= FALSE)
2.048407
```

Because the magnitude of the observed *T* statistic is less than the *T* value corresponding to the 0.05 significance level ($|-1.7828| < 2.0484$), the null hypothesis is not rejected. Because the alternative hypothesis is that the means are not equal ($\mu_1 \neq \mu_2$), the possibilities of both $\mu_1 > \mu_2$ and $\mu_1 < \mu_2$ need to be considered. This form of Student's *t*-test is known as a ***two-sided hypothesis test***, and it is necessary for the sum of the probabilities under both tails of the *t*-distribution to equal the significance level. It is customary to evenly

divide the significance level between both tails. So, $p = 0.05/2 = 0.025$ was used in the `qt()` function to obtain the appropriate *t*-value.

To simplify the comparison of the *t*-test results to the significance level, the R output includes a quantity known as the ***p-value***. In the preceding example, the *p*-value is 0.08547, which is the sum of $P(T \leq -1.7828)$ and $P(T \geq 1.7828)$. Figure 3-24 illustrates the *t*-statistic for the area under the tail of a *t*-distribution. The -*t* and *t* are the observed values of the *t*-statistic. In the R output, $t = 1.7828$. The left shaded area corresponds to the $P(T \leq -1.7828)$, and the right shaded area corresponds to the $P(T \geq 1.7828)$.

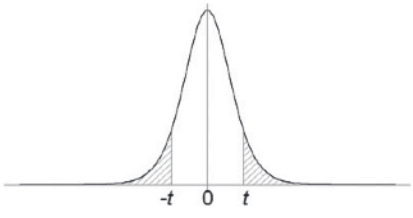


FIGURE 3-24 *Area under the tails (shaded) of a student's t-distribution*

In the R output, for a significance level of 0.05, the null hypothesis would not be rejected because the likelihood of a *T* value of magnitude 1.7828 or greater would occur at higher probability than 0.05. However, based on the *p*-value, if the significance level was chosen to be 0.10, instead of 0.05, the null hypothesis would be rejected. In general, the *p*-value offers the probability of observing such a sample result given the null hypothesis is `TRUE`.

A key assumption in using Student's *t*-test is that the population variances are equal. In the previous example, the `t.test()` function call includes `var.equal=TRUE` to specify that equality of the variances should be assumed. If that assumption is not appropriate, then Welch's *t*-test should be used.

Welch's t-test

When the equal population variance assumption is not justified in performing Student's *t*-test for the difference of means, Welch's *t*-test [14] can be used based on *T* expressed in Equation 3-2.

$$T_{welch} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \tag{3-2}$$

where $\bar{X}_i$, S_i^2, and n_i correspond to the *i*-th sample mean, sample variance, and sample size. Notice that Welch's *t*-test uses the sample variance (S_i^2) for each population instead of the pooled sample variance.

In Welch's test, under the remaining assumptions of random samples from two normal populations with the same mean, the distribution of *T* is approximated by the *t*-distribution. The following R code performs the Welch's *t*-test on the same set of data analyzed in the earlier Student's *t*-test example.

```
t.test(x, y, var.equal=FALSE)        # run the Welch's t-test

Welch Two Sample t-test

data:  x and y
t = -1.6596, df = 15.118, p-value = 0.1176
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -6.546629  0.812663
sample estimates:
  mean of x mean of y
102.2136  105.0806
```

In this particular example of using Welch's *t*-test, the *p*-value is 0.1176, which is greater than the *p*-value of 0.08547 observed in the Student's *t*-test example. In this case, the null hypothesis would not be rejected at a 0.10 or 0.05 significance level.

It should be noted that the degrees of freedom calculation is not as straightforward as in the Student's *t*-test. In fact, the degrees of freedom calculation often results in a non-integer value, as in this example. The degrees of freedom for Welch's *t*-test is defined in Equation 3-3.

$$df = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\left(\frac{S_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{S_2^2}{n_2}\right)^2}{n_2 - 1}} \tag{3-3}$$

In both the Student's and Welch's *t*-test examples, the R output provides 95% confidence intervals on the difference of the means. In both examples, the confidence intervals straddle zero. Regardless of the result of the hypothesis test, the confidence interval provides an interval estimate of the difference of the population means, not just a point estimate.

A ***confidence interval*** is an interval estimate of a population parameter or characteristic based on sample data. A confidence interval is used to indicate the uncertainty of a point estimate. If $\bar{x}$ is the estimate of some unknown population mean μ, the confidence interval provides an idea of how close $\bar{x}$ is to the unknown μ. For example, a 95% confidence interval for a population mean straddles the `TRUE`, but unknown mean 95% of the time. Consider Figure 3-25 as an example. Assume the confidence level is 95%. If the task is to estimate the mean of an unknown value μ in a normal distribution with known standard deviation σ and the estimate based on *n* observations is *x*, then the interval $\bar{x} \pm \frac{2\sigma}{\sqrt{n}}$ straddles the unknown value of μ with about a 95% chance. If one takes 100 different samples and computes the 95% confidence interval for the mean, 95 of the 100 confidence intervals will be expected to straddle the population mean μ.

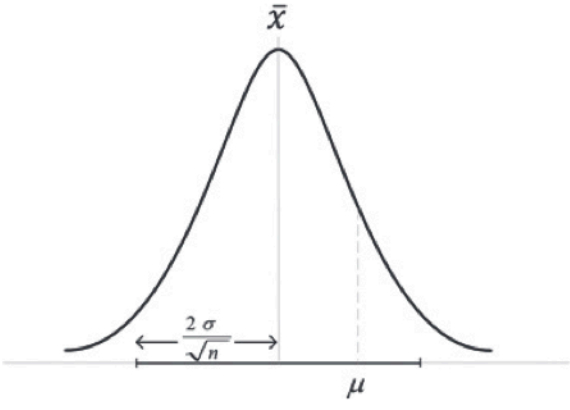


FIGURE 3-25 *A 95% confidence interval straddling the unknown population mean μ*

Confidence intervals appear again in Section 3.3.6 on ANOVA. Returning to the discussion of hypothesis testing, a key assumption in both the Student's and Welch's *t*-test is that the relevant population attribute is normally distributed. For non-normally distributed data, it is sometimes possible to transform the collected data to approximate a normal distribution. For example, taking the logarithm of a dataset can often transform skewed data to a dataset that is at least symmetric around its mean. However, if such transformations are ineffective, there are tests like the Wilcoxon rank-sum test that can be applied to see if two population distributions are different.

3.3.3 Wilcoxon Rank-Sum Test

A *t*-test represents a ***parametric test*** in that it makes assumptions about the population distributions from which the samples are drawn. If the populations cannot be assumed or transformed to follow a normal distribution, a ***nonparametric test*** can be used. The ***Wilcoxon rank-sum test*** [15] is a nonparametric hypothesis test that checks whether two populations are identically distributed. Assuming the two populations are identically distributed, one would expect that the ordering of any sampled observations would be evenly intermixed among themselves. For example, in ordering the observations, one would not expect to see a large number of observations from one population grouped together, especially at the beginning or the end of ordering.

Let the two populations again be `pop1` and `pop2`, with independently random samples of size n_1 and n_2 respectively. The total number of observations is then $N = n_1 + n_2$. The first step of the Wilcoxon test is to rank the set of observations from the two groups as if they came from one large group. The smallest observation receives a rank of 1, the second smallest observation receives a rank of 2, and so on with the largest observation being assigned the rank of *N*. Ties among the observations receive a rank equal to the average of the ranks they span. The test uses ranks instead of numerical outcomes to avoid specific assumptions about the shape of the distribution.

After ranking all the observations, the assigned ranks are summed for at least one population's sample. If the distribution of `pop1` is shifted to the right of the other distribution, the rank-sum corresponding to `pop1`'s sample should be larger than the rank-sum of `pop2`. The Wilcoxon rank-sum test determines the

significance of the observed rank-sums. The following R code performs the test on the same dataset used for the previous *t*-test.

```
wilcox.test(x, y, conf.int = TRUE)

Wilcoxon rank sum test

data:  x and y
W = 55, p-value = 0.04903
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
  -6.2596774 -0.1240618
sample estimates:
  difference in location
-3.417658
```

The `wilcox.test()` function ranks the observations, determines the respective rank-sums corresponding to each population's sample, and then determines the probability of such rank-sums of such magnitude being observed assuming that the population distributions are identical. In this example, the probability is given by the *p*-value of 0.04903. Thus, the null hypothesis would be rejected at a 0.05 significance level. The reader is cautioned against interpreting that one hypothesis test is clearly better than another test based solely on the examples given in this section.

Because the Wilcoxon test does not assume anything about the population distribution, it is generally considered more robust than the *t*-test. In other words, there are fewer assumptions to violate. However, when it is reasonable to assume that the data is normally distributed, Student's or Welch's *t*-test is an appropriate hypothesis test to consider.

3.3.4 Type I and Type II Errors

A hypothesis test may result in two types of errors, depending on whether the test accepts or rejects the null hypothesis. These two errors are known as type I and type II errors.

- A ***type I error*** is the rejection of the null hypothesis when the null hypothesis is `TRUE`. The probability of the type I error is denoted by the Greek letter α.
- A ***type II error*** is the acceptance of a null hypothesis when the null hypothesis is `FALSE`. The probability of the type II error is denoted by the Greek letter β.

Table 3-6 lists the four possible states of a hypothesis test, including the two types of errors.

TABLE 3-6 *Type I and Type II Error*

	H_0 is true	H_0 is false
H_0 is accepted	Correct outcome	*Type II Error*
H_0 is rejected	*Type I error*	Correct outcome

The significance level, as mentioned in the Student's *t*-test discussion, is equivalent to the type I error. For a significance level such as $\alpha = 0.05$, if the null hypothesis ($\mu_1 = \mu_2$) is `TRUE`, there is a 5% chance that the observed *T* value based on the sample data will be large enough to reject the null hypothesis. By selecting an appropriate significance level, the probability of committing a type I error can be defined before any data is collected or analyzed.

The probability of committing a Type II error is somewhat more difficult to determine. If two population means are truly not equal, the probability of committing a type II error will depend on how far apart the means truly are. To reduce the probability of a type II error to a reasonable level, it is often necessary to increase the sample size. This topic is addressed in the next section.

3.3.5 Power and Sample Size

The ***power*** of a test is the probability of correctly rejecting the null hypothesis. It is denoted by $1-\beta$, where β is the probability of a type II error. Because the power of a test improves as the sample size increases, power is used to determine the necessary sample size. In the difference of means, the power of a hypothesis test depends on the true difference of the population means. In other words, for a fixed significance level, a larger sample size is required to detect a smaller difference in the means. In general, the magnitude of the difference is known as the ***effect size***. As the sample size becomes larger, it is easier to detect a given effect size, δ, as illustrated in Figure 3-26.

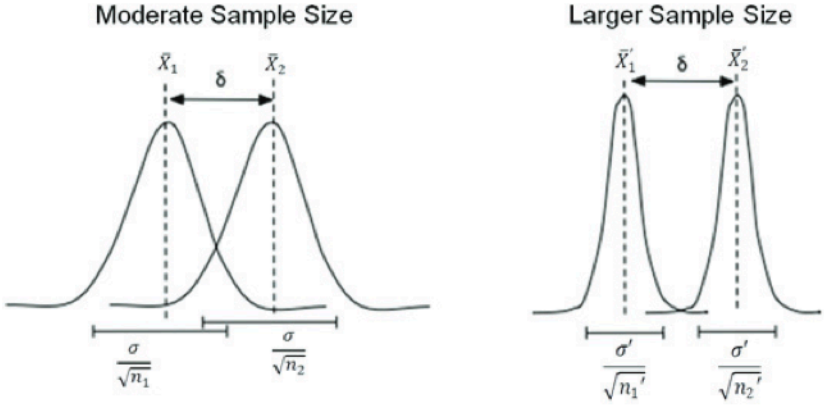


FIGURE 3-26 *A larger sample size better identifies a fixed effect size*

With a large enough sample size, almost any effect size can appear statistically significant. However, a very small effect size may be useless in a practical sense. It is important to consider an appropriate effect size for the problem at hand.

3.3.6 ANOVA

The *hypothesis tests* presented in the previous sections are good for analyzing means between two populations. But what if there are more than two populations? Consider an example of testing the impact of

nutrition and exercise on 60 candidates between age 18 and 50. The candidates are randomly split into six groups, each assigned with a different weight loss strategy, and the goal is to determine which strategy is the most effective.

- Group 1 only eats junk food.
- Group 2 only eats healthy food.
- Group 3 eats junk food and does cardio exercise every other day.
- Group 4 eats healthy food and does cardio exercise every other day.
- Group 5 eats junk food and does both cardio and strength training every other day.
- Group 6 eats healthy food and does both cardio and strength training every other day.

Multiple *t*-tests could be applied to each pair of weight loss strategies. In this example, the weight loss of Group 1 is compared with the weight loss of Group 2, 3, 4, 5, or 6. Similarly, the weight loss of Group 2 is compared with that of the next 4 groups. Therefore, a total of 15 *t*-tests would be performed.

However, multiple *t*-tests may not perform well on several populations for two reasons. First, because the number of *t*-tests increases as the number of groups increases, analysis using the multiple *t*-tests becomes cognitively more difficult. Second, by doing a greater number of analyses, the probability of committing at least one type I error somewhere in the analysis greatly increases.

Analysis of Variance (***ANOVA***) is designed to address these issues. ANOVA is a generalization of the hypothesis testing of the difference of two population means. ANOVA tests if any of the population means differ from the other population means. The null hypothesis of ANOVA is that all the population means are equal. The alternative hypothesis is that at least one pair of the population means is not equal. In other words,

- $\boldsymbol{H_0}$: $\mu_1 = \mu_2 = \ldots = \mu_n$
- $\boldsymbol{H_A}$: $\mu_i \neq \mu_j$ for at least one pair of i, j

As seen in Section 3.3.2, "Difference of Means," each population is assumed to be normally distributed with the same variance.

The first thing to calculate for the ANOVA is the test statistic. Essentially, the goal is to test whether the clusters formed by each population are more tightly grouped than the spread across all the populations.

Let the total number of populations be k. The total number of samples N is randomly split into the k groups. The number of samples in the i-th group is denoted as n_i, and the mean of the group is $\overline{X}_i$ where $i \in [1,k]$. The mean of all the samples is denoted as $\overline{X}_0$.

The ***between-groups mean sum of squares***, S_B^2, is an estimate of the ***between-groups variance***. It measures how the population means vary with respect to the grand mean, or the mean spread across all the populations. Formally, this is presented as shown in Equation 3-4.

$$S_B^2 = \frac{1}{k-1}\sum_{i=1}^{k} n_i \cdot (\overline{x}_i - \overline{x}_0)^2 \tag{3-4}$$

The ***within-group mean sum of squares***, S_W^2, is an estimate of the ***within-group variance***. It quantifies the spread of values within groups. Formally, this is presented as shown in Equation 3-5.

$$S_W^2 = \frac{1}{n-k}\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2 \tag{3-5}$$

If S_B^2 is much larger than S_W^2, then some of the population means are different from each other.

The *F*-test statistic is defined as the ratio of the between-groups mean sum of squares and the within-group mean sum of squares. Formally, this is presented as shown in Equation 3-6.

$$F = \frac{S_B^2}{S_W^2} \tag{3-6}$$

The *F*-test statistic in ANOVA can be thought of as a measure of how different the means are relative to the variability within each group. The larger the observed *F*-test statistic, the greater the likelihood that the differences between the means are due to something other than chance alone. The *F*-test statistic is used to test the hypothesis that the observed effects are not due to chance—that is, if the means are significantly different from one another.

Consider an example that every customer who visits a retail website gets one of two promotional offers or gets no promotion at all. The goal is to see if making the promotional offers makes a difference. ANOVA could be used, and the null hypothesis is that neither promotion makes a difference. The code that follows randomly generates a total of 500 observations of purchase sizes on three different offer options.

```
offers <- sample(c("offer1", "offer2", "nopromo"), size=500, replace=T)

# Simulated 500 observations of purchase sizes on the 3 offer options
purchasesize <- ifelse(offers=="offer1", rnorm(500, mean=80, sd=30),
                ifelse(offers=="offer2", rnorm(500, mean=85, sd=30),
                      rnorm(500, mean=40, sd=30)))

# create a data frame of offer option and purchase size
offertest <- data.frame(offer=as.factor(offers),
                        purchase_amt=purchasesize)
```

The summary of the `offertest` data frame shows that 170 `offer1`, 161 `offer2`, and 169 `nopromo` (no promotion) offers have been made. It also shows the range of purchase size (`purchase_amt`) for each of the three offer options.

```
# display a summary of offertest where offer="offer1"
summary(offertest[offertest$offer=="offer1",])
     offer       purchase_amt
 nopromo:  0   Min.   :  4.521
 offer1 :170   1st Qu.: 58.158
 offer2 :  0   Median : 76.944
               Mean   : 81.936
               3rd Qu.:104.959
               Max.   :180.507

# display a summary of offertest where offer="offer2"
summary(offertest[offertest$offer=="offer2",])
```

```
     offer      purchase_amt
 nopromo:  0   Min.   : 14.04
 offer1 :  0   1st Qu.: 69.46
 offer2 :161   Median : 90.20
               Mean   : 89.09
               3rd Qu.:107.48
               Max.   :154.33

# display a summary of offertest where offer="nopromo"
summary(offertest[offertest$offer=="nopromo",])
     offer      purchase_amt
 nopromo:169   Min.   :-27.00
 offer1 :  0   1st Qu.: 20.22
 offer2 :  0   Median : 42.44
               Mean   : 40.97
               3rd Qu.: 58.96
               Max.   :164.04
```

The `aov()` function performs the ANOVA on purchase size and offer options.

```
# fit ANOVA test
model <- aov(purchase_amt ~ offers, data=offertest)
```

The `summary()` function shows a summary of the model. The degrees of freedom for offers is 2, which corresponds to the $k-1$ in the denominator of Equation 3-4. The degrees of freedom for residuals is 497, which corresponds to the $n-k$ in the denominator of Equation 3-5.

```
summary(model)
             Df Sum Sq Mean Sq F value Pr(>F)
offers        2 225222  112611   130.6 <2e-16 ***
Residuals   497 428470     862
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output also includes the S_B^2 (112,611), S_W^2 (862), the F-test statistic (130.6), and the p-value (< 2e–16). The F-test statistic is much greater than 1 with a p-value much less than 1. Thus, the null hypothesis that the means are equal should be rejected.

However, the result does not show whether `offer1` is different from `offer2`, which requires additional tests. The `TukeyHSD()` function implements Tukey's Honest Significant Difference (HSD) on all pair-wise tests for difference of means.

```
TukeyHSD(model)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = purchase_amt ~ offers, data = offertest)

$offers
                    diff        lwr      upr     p adj
offer1-nopromo 40.961437 33.4638483 48.45903 0.0000000
```

```
offer2-nopromo 48.120286 40.5189446 55.72163 0.0000000
offer2-offer1   7.158849 -0.4315769 14.74928 0.0692895
```

The result includes *p*-values of pair-wise comparisons of the three offer options. The *p*-values for `offer1-nopromo` and `offer-nopromo` are equal to 0, smaller than the significance level 0.05. This suggests that both `offer1` and `offer2` are significantly different from `nopromo`. A *p*-value of 0.0692895 for `offer2` against `offer1` is greater than the significance level 0.05. This suggests that `offer2` is *not* significantly different from `offer1`.

Because only the influence of one factor (offers) was executed, the presented ANOVA is known as one-way ANOVA. If the goal is to analyze two factors, such as offers and day of week, that would be a two-way ANOVA [16]. If the goal is to model more than one outcome variable, then multivariate ANOVA (or MANOVA) could be used.

Summary

R is a popular package and programming language for data exploration, analytics, and visualization. As an introduction to R, this chapter covers the R GUI, data I/O, attribute and data types, and descriptive statistics. This chapter also discusses how to use R to perform exploratory data analysis, including the discovery of dirty data, visualization of one or more variables, and customization of visualization for different audiences. Finally, the chapter introduces some basic statistical methods. The first statistical method presented in the chapter is the hypothesis testing. The Student's *t*-test and Welch's *t*-test are included as two example hypothesis tests designed for testing the difference of means. Other statistical methods and tools presented in this chapter include confidence intervals, Wilcoxon rank-sum test, type I and II errors, effect size, and ANOVA.

Exercises

1. How many levels does `fdata` contain in the following R code?

```
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
```

2. Two vectors, `v1` and `v2`, are created with the following R code:

```
v1 <- 1:5
v2 <- 6:2
```

   What are the results of `cbind(v1,v2)` and `rbind(v1,v2)`?

3. What R command(s) would you use to remove null values from a dataset?

4. What R command can be used to install an additional R package?

5. What R function is used to encode a vector as a category?

6. What is a rug plot used for in a density plot?

7. An online retailer wants to study the purchase behaviors of its customers. Figure 3-27 shows the density plot of the purchase sizes (in dollars). What would be your recommendation to enhance the plot to detect more structures that otherwise might be missed?

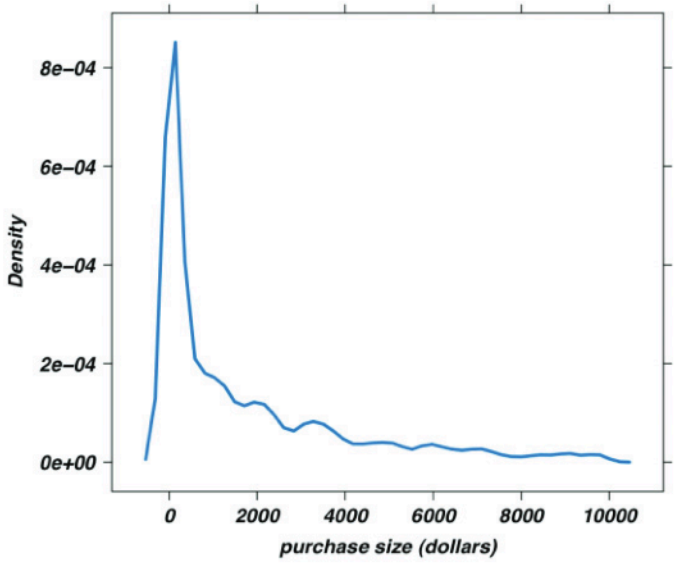


FIGURE 3-27 *Density plot of purchase size*

8. How many sections does a box-and-whisker divide the data into? What are these sections?
9. What attributes are correlated according to Figure 3-18? How would you describe their relationships?
10. What function can be used to fit a nonlinear line to the data?
11. If a graph of data is skewed and all the data is positive, what mathematical technique may be used to help detect structures that might otherwise be overlooked?
12. What is a type I error? What is a type II error? Is one always more serious than the other? Why?
13. Suppose everyone who visits a retail website gets one promotional offer or no promotion at all. We want to see if making a promotional offer makes a difference. What statistical method would you recommend for this analysis?
14. You are analyzing two normally distributed populations, and your null hypothesis is that the mean μ_1 of the first population is equal to the mean μ_2 of the second. Assume the significance level is set at 0.05. If the observed *p*-value is 4.33e-05, what will be your decision regarding the null hypothesis?

Bibliography

[1] The R Project for Statistical Computing, "R Licenses." [Online]. Available: `http://www.r-project.org/Licenses/`. [Accessed 10 December 2013].

[2] The R Project for Statistical Computing, "The Comprehensive R Archive Network." [Online]. Available: `http://cran.r-project.org/`. [Accessed 10 December 2013].

[3] J. Fox and M. Bouchet-Valat, "The R Commander: A Basic-Statistics GUI for R," CRAN. [Online]. Available: http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/. [Accessed 11 December 2013].

[4] G. Williams, M. V. Culp, E. Cox, A. Nolan, D. White, D. Medri, and A. Waljee, "Rattle: Graphical User Interface for Data Mining in R," CRAN. [Online]. Available: http://cran.r-project.org/web/packages/rattle/index.html. [Accessed 12 December 2013].

[5] RStudio, "RStudio IDE" [Online]. Available: http://www.rstudio.com/ide/. [Accessed 11 December 2013].

[6] R Special Interest Group on Databases (R-SIG-DB), "DBI: R Database Interface." CRAN [Online]. Available: http://cran.r-project.org/web/packages/DBI/index.html. [Accessed 13 December 2013].

[7] B. Ripley, "RODBC: ODBC Database Access," CRAN. [Online]. Available: http://cran.r-project.org/web/packages/RODBC/index.html. [Accessed 13 December 2013].

[8] S. S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, no. 2684, p. 677–680, 1946.

[9] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*, New York: Wiley, 1983.

[10] F. J. Anscombe, "Graphs in Statistical Analysis," *The American Statistician*, vol. 27, no. 1, pp. 17–21, 1973.

[11] H. Wickham, "ggplot2," 2013. [Online]. Available: http://ggplot2.org/. [Accessed 8 January 2014].

[12] W. S. Cleveland, *Visualizing Data*, Lafayette, IN: Hobart Press, 1993.

[13] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[14] B. L. Welch, "The Generalization of "Student's" Problem When Several Different Population Variances Are Involved," *Biometrika*, vol. 34, no. 1–2, pp. 28–35, 1947.

[15] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[16] J. J. Faraway, "Practical Regression and Anova Using R," July 2002. [Online]. Available: http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf. [Accessed 22 January 2014].

4

Advanced Analytical Theory and Methods: Clustering

Key Concepts

Centroid
Clustering
K-means
Unsupervised
Within Sum of Squares

Building upon the introduction to R presented in Chapter 3, "Review of Basic Data Analytic Methods Using R," Chapter 4, "Advanced Analytical Theory and Methods: Clustering" through Chapter 9, "Advanced Analytical Theory and Methods: Text Analysis" describe several commonly used analytical methods that may be considered for the Model Planning and Execution phases (Phases 3 and 4) of the Data Analytics Lifecycle. This chapter considers clustering techniques and algorithms.

4.1 Overview of Clustering

In general, clustering is the use of ***unsupervised*** techniques for grouping similar objects. In machine learning, unsupervised refers to the problem of finding hidden structure within unlabeled data. Clustering techniques are unsupervised in the sense that the data scientist does not determine, in advance, the labels to apply to the clusters. The structure of the data describes the objects of interest and determines how best to group the objects. For example, based on customers' personal income, it is straightforward to divide the customers into three groups depending on arbitrarily selected values. The customers could be divided into three groups as follows:

- Earn less than $10,000
- Earn between $10,000 and $99,999
- Earn $100,000 or more

In this case, the income levels were chosen somewhat subjectively based on easy-to-communicate points of delineation. However, such groupings do not indicate a natural affinity of the customers within each group. In other words, there is no inherent reason to believe that the customer making $90,000 will behave any differently than the customer making $110,000. As additional dimensions are introduced by adding more variables about the customers, the task of finding meaningful groupings becomes more complex. For instance, suppose variables such as age, years of education, household size, and annual purchase expenditures were considered along with the personal income variable. What are the natural occurring groupings of customers? This is the type of question that clustering analysis can help answer.

Clustering is a method often used for exploratory analysis of the data. In clustering, there are no predictions made. Rather, clustering methods find the similarities between objects according to the object attributes and group the similar objects into clusters. Clustering techniques are utilized in marketing, economics, and various branches of science. A popular clustering method is k-means.

4.2 K-means

Given a collection of objects each with n measurable attributes, ***k-means*** [1] is an analytical technique that, for a chosen value of k, identifies k clusters of objects based on the objects' proximity to the center of the k groups. The center is determined as the arithmetic average (mean) of each cluster's n-dimensional vector of attributes. This section describes the algorithm to determine the k means as well as how best to apply this technique to several use cases. Figure 4-1 illustrates three clusters of objects with two attributes. Each object in the dataset is represented by a small dot color-coded to the closest large dot, the mean of the cluster.

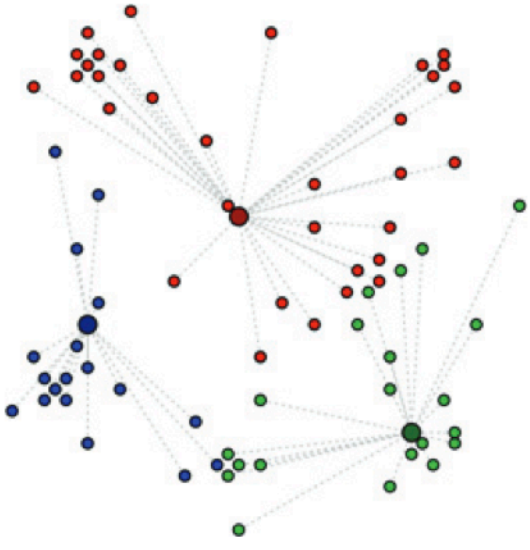


FIGURE 4-1 *Possible k-means clusters for k=3*

4.2.1 Use Cases

Clustering is often used as a lead-in to classification. Once the clusters are identified, labels can be applied to each cluster to classify each group based on its characteristics. Classification is covered in more detail in Chapter 7, "Advanced Analytical Theory and Methods: Classification." Clustering is primarily an exploratory technique to discover hidden structures of the data, possibly as a prelude to more focused analysis or decision processes. Some specific applications of k-means are image processing, medical, and customer segmentation.

Image Processing

Video is one example of the growing volumes of unstructured data being collected. Within each frame of a video, k-means analysis can be used to identify objects in the video. For each frame, the task is to determine which pixels are most similar to each other. The attributes of each pixel can include brightness, color, and location, the x and y coordinates in the frame. With security video images, for example, successive frames are examined to identify any changes to the clusters. These newly identified clusters may indicate unauthorized access to a facility.

Medical

Patient attributes such as age, height, weight, systolic and diastolic blood pressures, cholesterol level, and other attributes can identify naturally occurring clusters. These clusters could be used to target individuals for specific preventive measures or clinical trial participation. Clustering, in general, is useful in biology for the classification of plants and animals as well as in the field of human genetics.

Customer Segmentation

Marketing and sales groups use k-means to better identify customers who have similar behaviors and spending patterns. For example, a wireless provider may look at the following customer attributes: monthly bill, number of text messages, data volume consumed, minutes used during various daily periods, and years as a customer. The wireless company could then look at the naturally occurring clusters and consider tactics to increase sales or reduce the customer ***churn rate***, the proportion of customers who end their relationship with a particular company.

4.2.2 Overview of the Method

To illustrate the method to find k clusters from a collection of M objects with n attributes, the two-dimensional case (n = 2) is examined. It is much easier to visualize the k-means method in two dimensions. Later in the chapter, the two-dimension scenario is generalized to handle any number of attributes.

Because each object in this example has two attributes, it is useful to consider each object corresponding to the point (x_i, y_i), where x and y denote the two attributes and i = 1, 2 … M. For a given cluster of m points ($m \leq M$), the point that corresponds to the cluster's mean is called a ***centroid***. In mathematics, a centroid refers to a point that corresponds to the center of mass for an object.

The k-means algorithm to find k clusters can be described in the following four steps.

1. Choose the value of k and the k initial guesses for the centroids.

   In this example, k = 3, and the initial centroids are indicated by the points shaded in red, green, and blue in Figure 4-2.

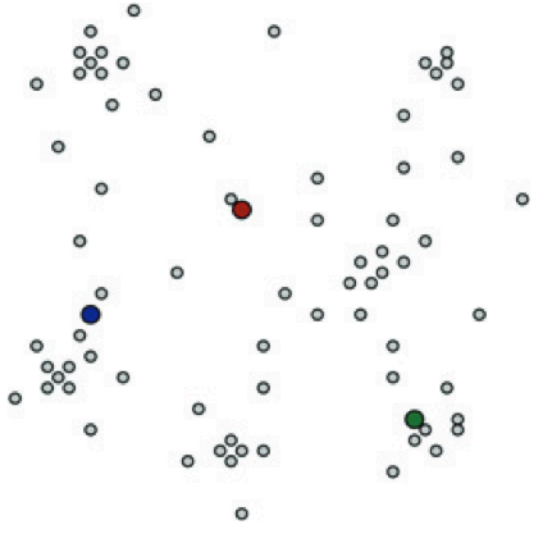


FIGURE 4-2 *Initial starting points for the centroids*

2. Compute the distance from each data point (x_i, y_i) to each centroid. Assign each point to the closest centroid. This association defines the first k clusters.

   In two dimensions, the distance, d, between any two points, (x_1, y_1) and (x_2, y_2), in the Cartesian plane is typically expressed by using the Euclidean distance measure provided in Equation 4-1.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4-1}$$

   In Figure 4-3, the points closest to a centroid are shaded the corresponding color.

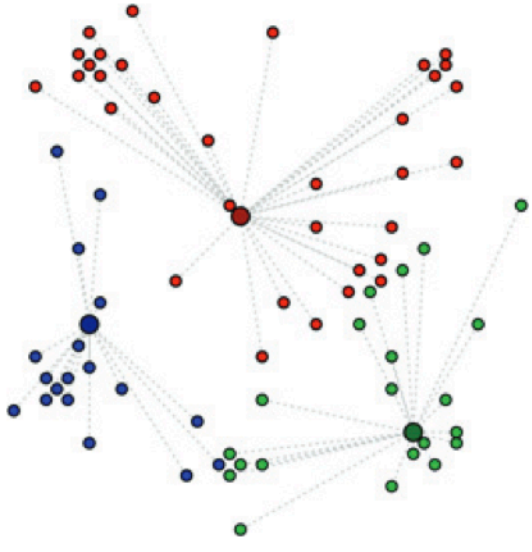


FIGURE 4-3 *Points are assigned to the closest centroid*

3. Compute the centroid, the center of mass, of each newly defined cluster from Step 2.

   In Figure 4-4, the computed centroids in Step 3 are the lightly shaded points of the corresponding color. In two dimensions, the centroid (x_C, y_C) of the m points in a k-means cluster is calculated as follows in Equation 4-2.

$$(x_C, y_C) = \left(\frac{\sum_{i=1}^{m} x_i}{m}, \frac{\sum_{i=1}^{m} y_i}{m} \right) \tag{4-2}$$

   Thus, (x_C, y_C) is the ordered pair of the arithmetic means of the coordinates of the m points in the cluster. In this step, a centroid is computed for each of the k clusters.

4. Repeat Steps 2 and 3 until the algorithm converges to an answer.

a. Assign each point to the closest centroid computed in Step 3.

b. Compute the centroid of newly defined clusters.

c. Repeat until the algorithm reaches the final answer.

Convergence is reached when the computed centroids do not change or the centroids and the assigned points oscillate back and forth from one iteration to the next. The latter case can occur when there are one or more points that are equal distances from the computed centroid.

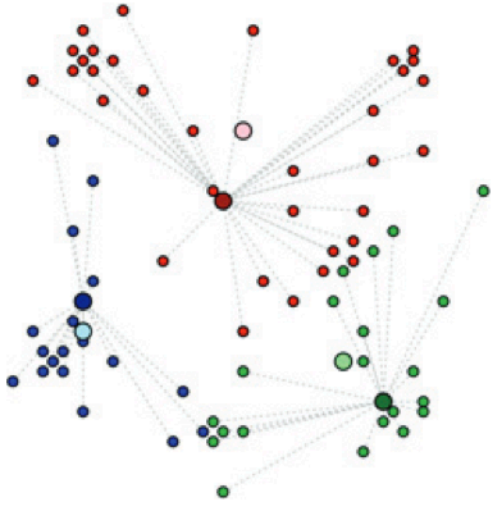


FIGURE 4-4 *Compute the mean of each cluster*

To generalize the prior algorithm to n dimensions, suppose there are M objects, where each object is described by n attributes or property values $(p_1, p_2, \ldots p_n)$. Then object i is described by $(p_{i1}, p_{i2}, \ldots p_{in})$ for $i = 1,2,\ldots$, M. In other words, there is a matrix with M rows corresponding to the M objects and n columns to store the attribute values. To expand the earlier process to find the k clusters from two dimensions to n dimensions, the following equations provide the formulas for calculating the distances and the locations of the centroids for n ≥ 1.

For a given point, $\boldsymbol{p}_i$, at $(p_{i1}, p_{i2}, \ldots p_{in})$ and a centroid, $\boldsymbol{q}$, located at $(q_1, q_2, \ldots q_n)$, the distance, $\boldsymbol{d}$, between $\boldsymbol{p}_i$ and $\boldsymbol{q}$, is expressed as shown in Equation 4-3.

$$d(p_i, q) = \sqrt{\sum_{j=1}^{n} (p_{ij} - q_j)^2} \tag{4-3}$$

The centroid, $\boldsymbol{q}$, of a cluster of m points, $(p_{i1}, p_{i2}, \ldots p_{in})$, is calculated as shown in Equation 4-4.

$$(q_1, q_2, \ldots, q_n) = \left(\frac{\sum_{i=1}^{m} p_{i1}}{m}, \frac{\sum_{i=1}^{m} p_{i2}}{m}, \ldots, \frac{\sum_{i=1}^{m} p_{in}}{m} \right) \tag{4-4}$$

4.2.3 Determining the Number of Clusters

With the preceding algorithm, k clusters can be identified in a given dataset, but what value of k should be selected? The value of k can be chosen based on a reasonable guess or some predefined requirement. However, even then, it would be good to know how much better or worse having k clusters versus k − 1 or k + 1 clusters would be in explaining the structure of the data. Next, a heuristic using the Within Sum of Squares (WSS) metric is examined to determine a reasonably optimal value of k. Using the distance function given in Equation 4-3, WSS is defined as shown in Equation 4-5.

$$WSS = \sum_{i=1}^{M} d(p_i, q^{(i)})^2 = \sum_{i=1}^{M} \sum_{j=1}^{n} \left(p_{ij} - q_j^{(i)}\right)^2 \tag{4-5}$$

In other words, WSS is the sum of the squares of the distances between each data point and the closest centroid. The term $q^{(i)}$ indicates the closest centroid that is associated with the ith point. If the points are relatively close to their respective centroids, the WSS is relatively small. Thus, if k + 1 clusters do not greatly reduce the value of WSS from the case with only k clusters, there may be little benefit to adding another cluster.

Using R to Perform a K-means Analysis

To illustrate how to use the WSS to determine an appropriate number, k, of clusters, the following example uses R to perform a k-means analysis. The task is to group 620 high school seniors based on their grades in three subject areas: English, mathematics, and science. The grades are averaged over their high school career and assume values from 0 to 100. The following R code establishes the necessary R libraries and imports the CSV file containing the grades.

```
library(plyr)
library(ggplot2)
library(cluster)
library(lattice)
library(graphics)
library(grid)
library(gridExtra)

#import the student grades
grade_input = as.data.frame(read.csv("c:/data/grades_km_input.csv"))
```

The following R code formats the grades for processing. The data file contains four columns. The first column holds a student identification (ID) number, and the other three columns are for the grades in the three subject areas. Because the student ID is not used in the clustering analysis, it is excluded from the k-means input matrix, *kmdata*.

```
kmdata_orig = as.matrix(grade_input[,c("Student","English", "Math","Science")])
kmdata <- kmdata_orig[,2:4]
```

```
kmdata[1:10,]

      English Math Science
 [1,]      99   96      97
 [2,]      99   96      97
 [3,]      98   97      97
 [4,]      95  100      95
 [5,]      95   96      96
 [6,]      96   97      96
 [7,]     100   96      97
 [8,]      95   98      98
 [9,]      98   96      96
[10,]      99   99      95
```

To determine an appropriate value for k, the k-means algorithm is used to identify clusters for k = 1, 2, …, 15. For each value of k, the WSS is calculated. If an additional cluster provides a better partitioning of the data points, the WSS should be markedly smaller than without the additional cluster.

The following R code loops through several k-means analyses for the number of centroids, *k*, varying from 1 to 15. For each k, the option *nstart*=25 specifies that the k-means algorithm will be repeated 25 times, each starting with *k* random initial centroids. The corresponding value of WSS for each k-mean analysis is stored in the *wss* vector.

```
wss <- numeric(15)
for (k in 1:15) wss[k] <- sum(kmeans(kmdata, centers=k, nstart=25)$withinss)
```

Using the basic R plot function, each WSS is plotted against the respective number of centroids, 1 through 15. This plot is provided in Figure 4-5.

```
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within Sum of
Squares")
```

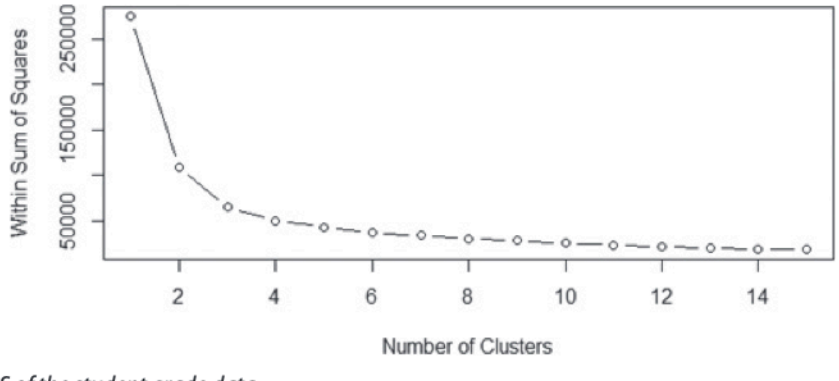


FIGURE 4-5 *WSS of the student grade data*

As can be seen, the WSS is greatly reduced when k increases from one to two. Another substantial reduction in WSS occurs at k = 3. However, the improvement in WSS is fairly linear for k > 3. Therefore, the k-means analysis will be conducted for k = 3. The process of identifying the appropriate value of k is referred to as finding the "elbow" of the WSS curve.

```
km = kmeans(kmdata,3, nstart=25)
km

K-means clustering with 3 clusters of sizes 158, 218, 244

Cluster means:
   English     Math  Science
1 97.21519 93.37342 94.86076
2 73.22018 64.62844 65.84862
3 85.84426 79.68033 81.50820

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
 [41] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
 [81] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      1 1 1 1 1 1 1 1 1 1
[121] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
      3 3 3 3 3 3 3 3 3 3
[161] 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 3 3 3 1 3 3 3 3
      3 3 1 3 3 3 3 3 3 3
[201] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[241] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[281] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[321] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
      3 3 3 3 3 3 3 3 3 3
[361] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 3 2 3 2 3 3 3
      2 2 2 2 3 3 2 2 2 2
[401] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[441] 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 3
      2 2 2 2 2 2 2 2 3 2
[481] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[521] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[561] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
      2 2 2 2 2 2 2 2 2 2
[601] 3 3 2 2 3 3 3 3 1 1 3 3 3 2 2 3 2 3 3 3

Within cluster sum of squares by cluster:
[1]  6692.589 34806.339 22984.131
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"   "centers"  "totss"    "withinss"   "tot.withinss"
[6] "betweenss" "size"     "iter"     "ifault"
```

The displayed contents of the variable *km* include the following:

- The location of the cluster means
- A clustering vector that defines the membership of each student to a corresponding cluster 1, 2, or 3
- The WSS of each cluster
- A list of all the available k-means components

The reader can find details on these components and using k-means in R by employing the help facility.

The reader may have wondered whether the k-means results stored in *km* are equivalent to the WSS results obtained earlier in generating the plot in Figure 4-5. The following check verifies that the results are indeed equivalent.

```
c( wss[3] , sum(km$withinss) )
```

```
[1] 64483.06 64483.06
```

In determining the value of k, the data scientist should visualize the data and assigned clusters. In the following code, the `ggplot2` package is used to visualize the identified student clusters and centroids.

```
#prepare the student data and clustering results for plotting
df = as.data.frame(kmdata_orig[,2:4])
df$cluster = factor(km$cluster)
centers=as.data.frame(km$centers)

g1= ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
  geom_point() + theme(legend.position="right") +
  geom_point(data=centers,
             aes(x=English,y=Math, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
  geom_point() +
  geom_point(data=centers,
             aes(x=English,y=Science, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
  geom_point() +
  geom_point(data=centers,
             aes(x=Math,y=Science, color=as.factor(c(1,2,3))),
             size=10, alpha=.3, show_guide=FALSE)

tmp = ggplot_gtable(ggplot_build(g1))
```

```
grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),
                         g2 + theme(legend.position="none"),
                         g3 + theme(legend.position="none"),
                         main ="High School Student Cluster Analysis",
                         ncol=1))
```

The resulting plots are provided in Figure 4-6. The large circles represent the location of the cluster means provided earlier in the display of the *km* contents. The small dots represent the students corresponding to the appropriate cluster by assigned color: red, blue, or green. In general, the plots indicate the three clusters of students: the top academic students (red), the academically challenged students (green), and the other students (blue) who fall somewhere between those two groups. The plots also highlight which students may excel in one or two subject areas but struggle in other areas.

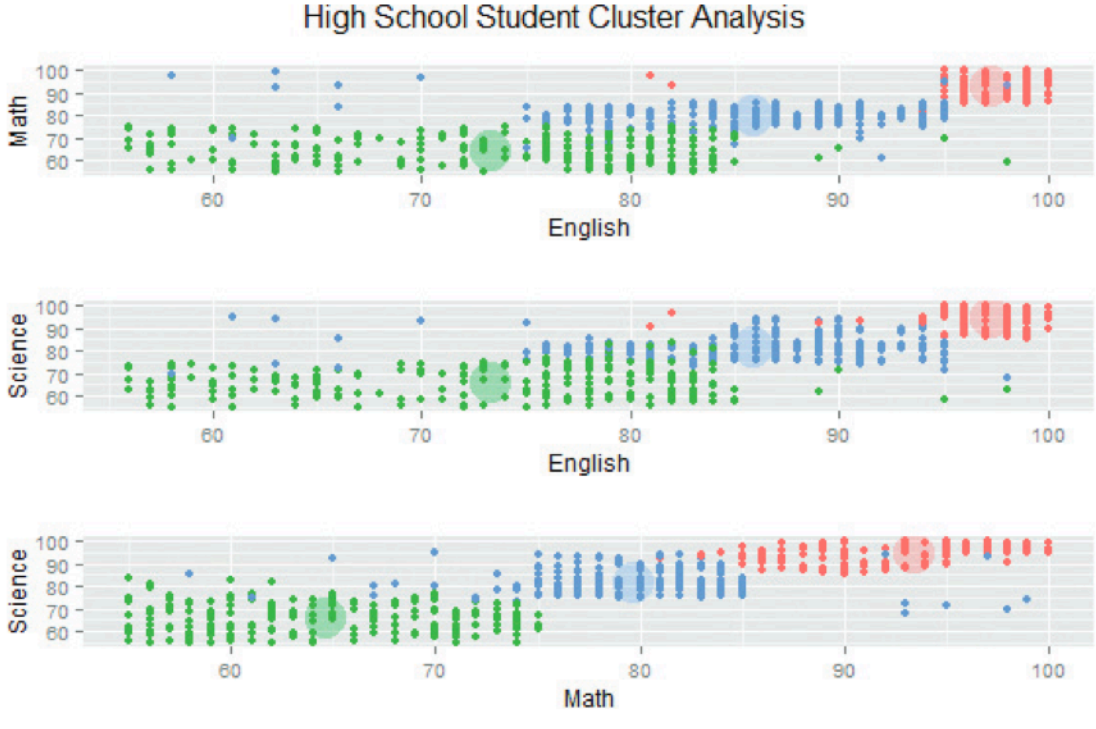


FIGURE 4-6 *Plots of the identified student clusters*

Assigning labels to the identified clusters is useful to communicate the results of an analysis. In a marketing context, it is common to label a group of customers as frequent shoppers or big spenders. Such designations are especially useful when communicating the clustering results to business users or executives. It is better to describe the marketing plan for big spenders rather than Cluster #1.

4.2.4 Diagnostics

The heuristic using WSS can provide at least several possible k values to consider. When the number of attributes is relatively small, a common approach to further refine the choice of k is to plot the data to determine how distinct the identified clusters are from each other. In general, the following questions should be considered.

- Are the clusters well separated from each other?
- Do any of the clusters have only a few points?
- Do any of the centroids appear to be too close to each other?

In the first case, ideally the plot would look like the one shown in Figure 4-7, when n = 2. The clusters are well defined, with considerable space between the four identified clusters. However, in other cases, such as Figure 4-8, the clusters may be close to each other, and the distinction may not be so obvious.

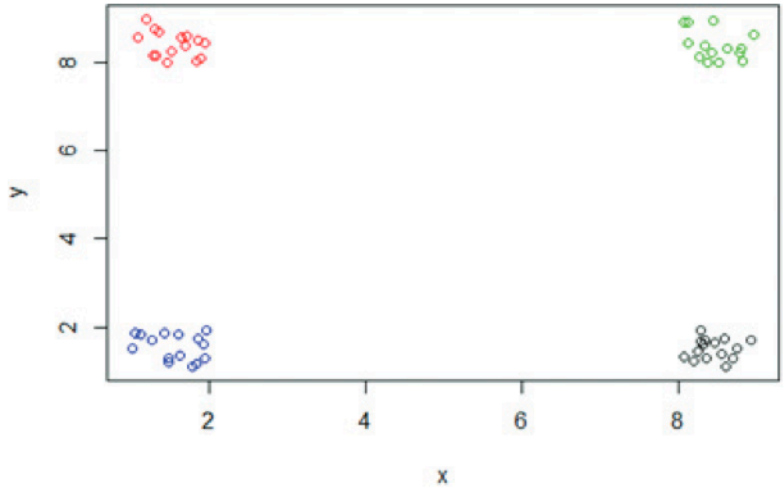


FIGURE 4-7 *Example of distinct clusters*

In such cases, it is important to apply some judgment on whether anything different will result by using more clusters. For example, Figure 4-9 uses six clusters to describe the same dataset as used in Figure 4-8. If using more clusters does not better distinguish the groups, it is almost certainly better to go with fewer clusters.

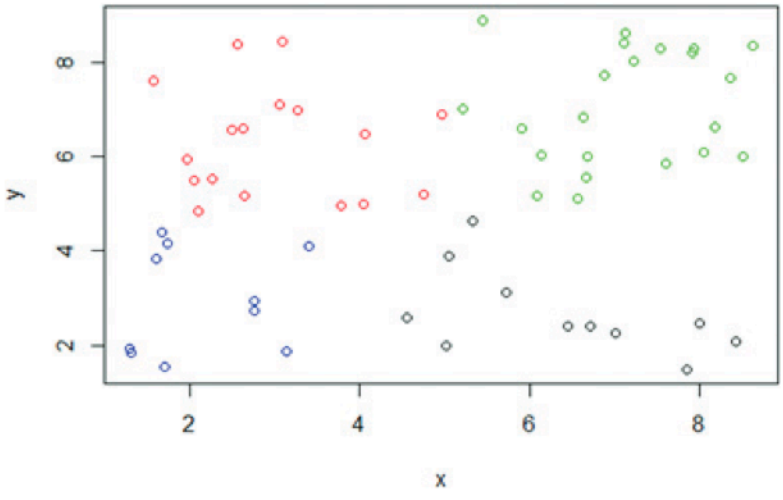


FIGURE 4-8 *Example of less obvious clusters*

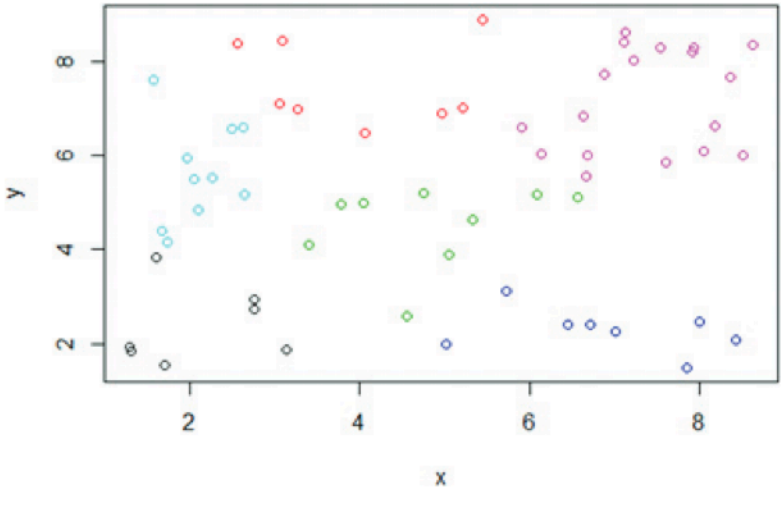


FIGURE 4-9 *Six clusters applied to the points from Figure 4-8*

4.2.5 Reasons to Choose and Cautions

K-means is a simple and straightforward method for defining clusters. Once clusters and their associated centroids are identified, it is easy to assign new objects (for example, new customers) to a cluster based on the object's distance from the closest centroid. Because the method is unsupervised, using k-means helps to eliminate subjectivity from the analysis.

Although k-means is considered an unsupervised method, there are still several decisions that the practitioner must make:

- What object attributes should be included in the analysis?
- What unit of measure (for example, miles or kilometers) should be used for each attribute?
- Do the attributes need to be rescaled so that one attribute does not have a disproportionate effect on the results?
- What other considerations might apply?

Object Attributes

Regarding which object attributes (for example, age and income) to use in the analysis, it is important to understand what attributes will be known at the time a new object will be assigned to a cluster. For example, information on existing customers' satisfaction or purchase frequency may be available, but such information may not be available for potential customers.

The Data Scientist may have a choice of a dozen or more attributes to use in the clustering analysis. Whenever possible and based on the data, it is best to reduce the number of attributes to the extent possible. Too many attributes can minimize the impact of the most important variables. Also, the use of several similar attributes can place too much importance on one type of attribute. For example, if five attributes related to personal wealth are included in a clustering analysis, the wealth attributes dominate the analysis and possibly mask the importance of other attributes, such as age.

When dealing with the problem of too many attributes, one useful approach is to identify any highly correlated attributes and use only one or two of the correlated attributes in the clustering analysis. As illustrated in Figure 4-10, a scatterplot matrix, as introduced in Chapter 3, is a useful tool to visualize the pair-wise relationships between the attributes.

The strongest relationship is observed to be between `Attribute3` and `Attribute7`. If the value of one of these two attributes is known, it appears that the value of the other attribute is known with near certainty. Other linear relationships are also identified in the plot. For example, consider the plot of `Attribute2` against `Attribute3`. If the value of `Attribute2` is known, there is still a wide range of possible values for `Attribute3`. Thus, greater consideration must be given prior to dropping one of these attributes from the clustering analysis.

Another option to reduce the number of attributes is to combine several attributes into one measure. For example, instead of using two attribute variables, one for Debt and one for Assets, a Debt to Asset ratio could be used. This option also addresses the problem when the magnitude of an attribute is not of real interest, but the relative magnitude is a more important measure.

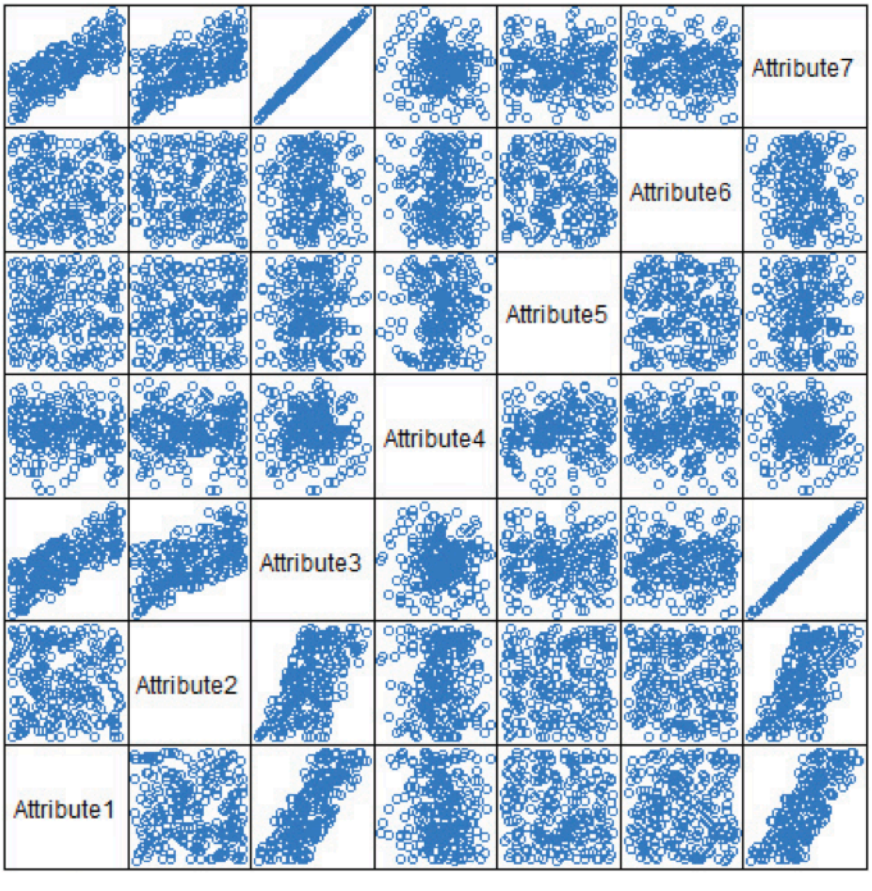


FIGURE 4-10 *Scatterplot matrix for seven attributes*

Units of Measure

From a computational perspective, the k-means algorithm is somewhat indifferent to the units of measure for a given attribute (for example, meters or centimeters for a patient's height). However, the algorithm will identify different clusters depending on the choice of the units of measure. For example, suppose that k-means is used to cluster patients based on age in years and height in centimeters. For k=2, Figure 4-11 illustrates the two clusters that would be determined for a given dataset.

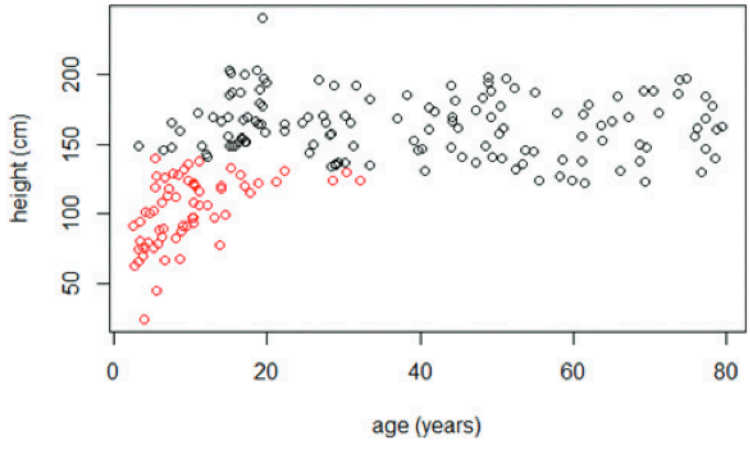


FIGURE 4-11 *Clusters with height expressed in centimeters*

But if the height was rescaled from centimeters to meters by dividing by 100, the resulting clusters would be slightly different, as illustrated in Figure 4-12.

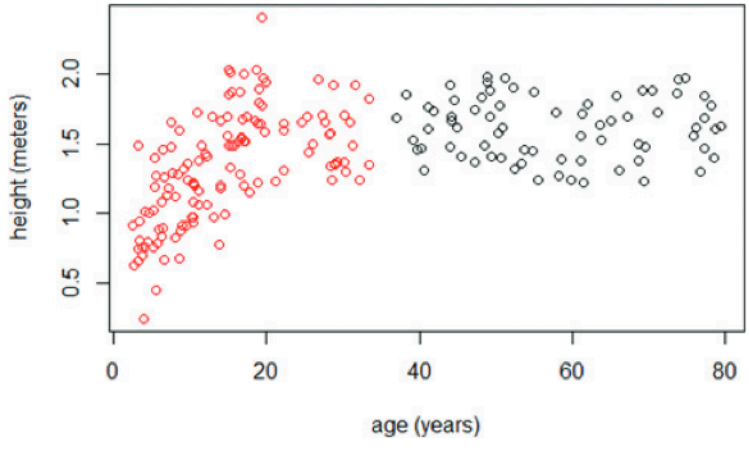


FIGURE 4-12 *Clusters with height expressed in meters*

When the height is expressed in meters, the magnitude of the ages dominates the distance calculation between two points. The height attribute provides only as much as the square between the difference of the maximum height and the minimum height or $(2.0 - 0)^2 = 4$ to the radicand, the number under the square root symbol in the distance formula given in Equation 4-3. Age can contribute as much as $(80 - 0)^2 = 6,400$ to the radicand when measuring the distance.

Rescaling

Attributes that are expressed in dollars are common in clustering analyses and can differ in magnitude from the other attributes. For example, if personal income is expressed in dollars and age is expressed in years, the income attribute, often exceeding $10,000, can easily dominate the distance calculation with ages typically less than 100 years.

Although some adjustments could be made by expressing the income in thousands of dollars (for example, 10 for $10,000), a more straightforward method is to divide each attribute by the attribute's standard deviation. The resulting attributes will each have a standard deviation equal to 1 and will be without units. Returning to the age and height example, the standard deviations are 23.1 years and 36.4 cm, respectively. Dividing each attribute value by the appropriate standard deviation and performing the k-means analysis yields the result shown in Figure 4-13.

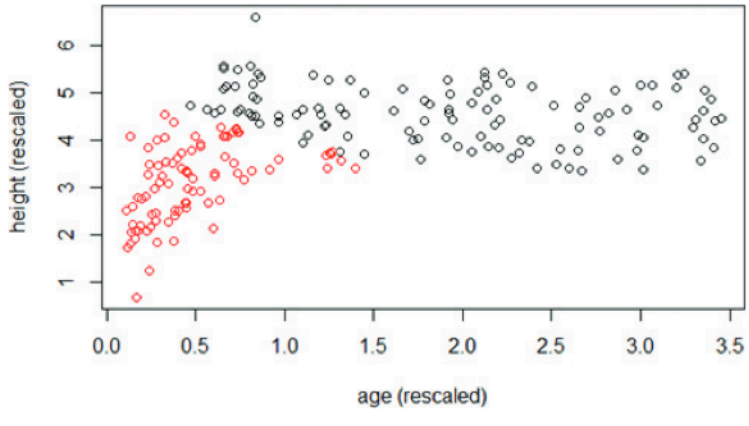


FIGURE 4-13 *Clusters with rescaled attributes*

With the rescaled attributes for age and height, the borders of the resulting clusters now fall somewhere between the two earlier clustering analyses. Such an occurrence is not surprising based on the magnitudes of the attributes of the previous clustering attempts. Some practitioners also subtract the means of the attributes to center the attributes around zero. However, this step is unnecessary because the distance formula is only sensitive to the scale of the attribute, not its location.

In many statistical analyses, it is common to transform typically skewed data, such as income, with long tails by taking the logarithm of the data. Such transformation can also be appied in k-means, but the Data Scientist needs to be aware of what effect this transformation will have. For example, if log_{10} of income expressed in dollars is used, the practitioner is essentially stating that, from a clustering perspective, $1,000 is as close to $10,000 as $10,000 is to $100,000 (*because* $log_{10} 1{,}000 = 3, log_{10} 10{,}000 = 4,$ *and* $log_{10} 100{,}000 = 5$). In many cases, the skewness of the data may be the reason to perform the clustering analysis in the first place.

Additional Considerations

The k-means algorithm is sensitive to the starting positions of the initial centroid. Thus, it is important to rerun the k-means analysis several times for a particular value of k to ensure the cluster results provide the overall minimum WSS. As seen earlier, this task is accomplished in R by using the `nstart` option in the `kmeans()` function call.

This chapter presented the use of the Euclidean distance function to assign the points to the closest centroids. Other possible function choices include the cosine similarity and the Manhattan distance functions. The cosine similarity function is often chosen to compare two documents based on the frequency of each word that appears in each of the documents [2]. For two points, p and q, at $(p_1, p_2, \ldots p_n)$ and $(q_1, q_2, \ldots q_n)$, respectively, the Manhattan distance, d_1, between p and q is expressed as shown in Equation 4-6.

$$d_1(p, q) = \sum_{j=1}^{n} \left| p_j - q_j \right| \tag{4-6}$$

The Manhattan distance function is analogous to the distance traveled by a car in a city, where the streets are laid out in a rectangular grid (such as city blocks). In Euclidean distance, the measurement is made in a straight line. Using Equation 4-6, the distance from (1, 1) to (4, 5) would be $|1 - 4| + |1 - 5| = 7$. From an optimization perspective, if there is a need to use the Manhattan distance for a clustering analysis, the median is a better choice for the centroid than use of the mean [2].

K-means clustering is applicable to objects that can be described by attributes that are numerical with a meaningful distance measure. From Chapter 3, interval and ratio attribute types can certainly be used. However, k-means does not handle categorical variables well. For example, suppose a clustering analysis is to be conducted on new car sales. Among other attributes, such as the sale price, the color of the car is considered important. Although one could assign numerical values to the color, such as red = 1, yellow = 2, and green = 3, it is not useful to consider that yellow is as close to red as yellow is to green from a clustering perspective. In such cases, it may be necessary to use an alternative clustering methodology. Such methods are described in the next section.

4.3 Additional Algorithms

The k-means clustering method is easily applied to numeric data where the concept of distance can naturally be applied. However, it may be necessary or desirable to use an alternative clustering algorithm. As discussed at the end of the previous section, k-means does not handle categorical data. In such cases, k-modes [3] is a commonly used method for clustering categorical data based on the number of differences in the respective components of the attributes. For example, if each object has four attributes, the distance from (a, b, e, d) to (d, d, d, d) is 3. In R, the function `kmode()` is implemented in the `klaR` package.

Because k-means and k-modes divide the entire dataset into distinct groups, both approaches are considered partitioning methods. A third partitioning method is known as Partitioning around Medoids (PAM) [4]. In general, a medoid is a representative object in a set of objects. In clustering, the ***medoids*** are the objects in each cluster that minimize the sum of the distances from the medoid to the other objects in the cluster. The advantage of using PAM is that the "center" of each cluster is an actual object in the dataset. PAM is implemented in R by the `pam()` function included in the `cluster` R package. The `fpc` R package includes a function `pamk()`, which uses the `pam()` function to find the optimal value for k.

Other clustering methods include hierarchical agglomerative clustering and density clustering methods. In hierarchical agglomerative clustering, each object is initially placed in its own cluster. The clusters are then combined with the most similar cluster. This process is repeated until one cluster, which includes all the objects, exists. The R `stats` package includes the `hclust()` function for performing hierarchical agglomerative clustering. In density-based clustering methods, the clusters are identified by the concentration of points. The `fpc` R package includes a function, `dbscan()`, to perform density-based clustering analysis. Density-based clustering can be useful to identify irregularly shaped clusters.

Summary

Clustering analysis groups similar objects based on the objects' attributes. Clustering is applied in areas such as marketing, economics, biology, and medicine. This chapter presented a detailed explanation of the k-means algorithm and its implementation in R. To use k-means properly, it is important to do the following:

- Properly scale the attribute values to prevent certain attributes from dominating the other attributes.
- Ensure that the concept of distance between the assigned values within an attribute is meaningful.
- Choose the number of clusters, k, such that the sum of the Within Sum of Squares (WSS) of the distances is reasonably minimized. A plot such as the example in Figure 4-5 can be helpful in this respect.

If k-means does not appear to be an appropriate clustering technique for a given dataset, then alternative techniques such as k-modes or PAM should be considered.

Once the clusters are identified, it is often useful to label these clusters in some descriptive way. Especially when dealing with upper management, these labels are useful to easily communicate the findings of the clustering analysis. In clustering, the labels are not preassigned to each object. The labels are subjectively assigned after the clusters have been identified. Chapter 7 considers several methods to perform the classification of objects with predetermined labels. Clustering can be used with other analytical techniques, such as regression. Linear regression and logistic regression are covered in Chapter 6, "Advanced Analytical Theory and Methods: Regression."

Exercises

1. Using the age and height clustering example in section 4.2.5, algebraically illustrate the impact on the measured distance when the height is expressed in meters rather than centimeters. Explain why different clusters will result depending on the choice of units for the patient's height.
2. Compare and contrast five clustering algorithms, assigned by the instructor or selected by the student.

3. Using the `ruspini` dataset provided with the `cluster` package in R, perform a k-means analysis. Document the findings and justify the choice of k. Hint: use `data(ruspini)` to load the dataset into the R workspace.

Bibliography

[1] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley, CA, 1967.

[2] P.-N. Tan, V. Kumar, and M. Steinbach, *Introduction to Data Mining*, Upper Saddle River, NJ: Person, 2013.

[3] Z. Huang, "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining," 1997. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.83&rep=rep1&type=pdf`. [Accessed 13 March 2014].

[4] L. Kaufman and P. J. Rousseeuw, "Partitioning Around Medoids (Program PAM)," in *Finding Groups in Data: An Introduction to Cluster Analysis,* Hoboken, NJ, John Wiley & Sons, Inc, 2008, p. 68-125, Chapter 2.

5

Advanced Analytical Theory and Methods: Association Rules

Key Concepts

Association rules
Apriori algorithm
Support
Confidence
Lift
Leverage

This chapter discusses an unsupervised learning method called association rules. This is a descriptive, not predictive, method often used to discover interesting relationships hidden in a large dataset. The disclosed relationships can be represented as rules or frequent itemsets. Association rules are commonly used for mining transactions in databases.

Here are some possible questions that association rules can answer:

- Which products tend to be purchased together?
- Of those customers who are similar to this person, what products do they tend to buy?
- Of those customers who have purchased this product, what other similar products do they tend to view or purchase?

5.1 Overview

Figure 5-1 shows the general logic behind association rules. Given a large collection of transactions (depicted as three stacks of receipts in the figure), in which each transaction consists of one or more items, association rules go through the items being purchased to see what items are frequently bought together and to discover a list of rules that describe the purchasing behavior. The goal with association rules is to discover interesting relationships among the items. (The relationship occurs too frequently to be random and is meaningful from a business perspective, which may or may not be obvious.) The relationships that are interesting depend both on the business context and the nature of the algorithm being used for the discovery.

FIGURE 5-1 *The general logic behind association rules*

Each of the uncovered rules is in the form X → Y, meaning that when item X is observed, item Y is also observed. In this case, the left-hand side (LHS) of the rule is X, and the right-hand side (RHS) of the rule is Y.

Using association rules, patterns can be discovered from the data that allow the association rule algorithms to disclose rules of related product purchases. The uncovered rules are listed on the right side of

Figure 5-1. The first three rules suggest that when cereal is purchased, 90% of the time milk is purchased also. When bread is purchased, 40% of the time milk is purchased also. When milk is purchased, 23% of the time cereal is also purchased.

In the example of a retail store, association rules are used over transactions that consist of one or more items. In fact, because of their popularity in mining customer transactions, association rules are sometimes referred to as ***market basket analysis***. Each transaction can be viewed as the shopping basket of a customer that contains one or more items. This is also known as an itemset. The term ***itemset*** refers to a collection of items or individual entities that contain some kind of relationship. This could be a set of retail items purchased together in one transaction, a set of hyperlinks clicked on by one user in a single session, or a set of tasks done in one day. An itemset containing ***k*** items is called a ***k-itemset***. This chapter uses curly braces like $\{\texttt{item}\ 1, \texttt{item}\ 2, \ldots \texttt{item}\ k\}$ to denote a k-itemset. Computation of the association rules is typically based on itemsets.

The research of association rules started as early as the 1960s. Early research by Hájek et al. [1] introduced many of the key concepts and approaches of association rule learning, but it focused on the mathematical representation rather than the algorithm. The framework of association rule learning was brought into the database community by Agrawal et al. [2] in the early 1990s for discovering regularities between products in a large database of customer transactions recorded by point-of-sale systems in supermarkets. In later years, it expanded to web contexts, such as mining path traversal patterns [3] and usage patterns [4] to facilitate organization of web pages.

This chapter chooses Apriori as the main focus of the discussion of association rules. Apriori [5] is one of the earliest and the most fundamental algorithms for generating association rules. It pioneered the use of support for pruning the itemsets and controlling the exponential growth of candidate itemsets. Shorter candidate itemsets, which are known to be frequent itemsets, are combined and pruned to generate longer frequent itemsets. This approach eliminates the need for all possible itemsets to be enumerated within the algorithm, since the number of all possible itemsets can become exponentially large.

One major component of Apriori is support. Given an itemset L, the ***support*** [2] of L is the percentage of transactions that contain L. For example, if 80% of all transactions contain itemset $\{\texttt{bread}\}$, then the support of $\{\texttt{bread}\}$ is 0.8. Similarly, if 60% of all transactions contain itemset $\{\texttt{bread}, \texttt{butter}\}$, then the support of $\{\texttt{bread}, \texttt{butter}\}$ is 0.6.

A ***frequent itemset*** has items that appear together often enough. The term "often enough" is formally defined with a ***minimum support*** criterion. If the minimum support is set at 0.5, any itemset can be considered a frequent itemset if at least 50% of the transactions contain this itemset. In other words, the support of a frequent itemset should be greater than or equal to the minimum support. For the previous example, both $\{\texttt{bread}\}$ and $\{\texttt{bread}, \texttt{butter}\}$ are considered frequent itemsets at the minimum support 0.5. If the minimum support is 0.7, only $\{\texttt{bread}\}$ is considered a frequent itemset.

If an itemset is considered frequent, then any subset of the frequent itemset must also be frequent. This is referred to as the ***Apriori property*** (or ***downward closure property***). For example, if 60% of the transactions contain $\{\texttt{bread}, \texttt{jam}\}$, then at least 60% of all the transactions will contain $\{\texttt{bread}\}$ or $\{\texttt{jam}\}$. In other words, when the support of $\{\texttt{bread}, \texttt{jam}\}$ is 0.6, the support of $\{\texttt{bread}\}$ or $\{\texttt{jam}\}$ is at least 0.6. Figure 5-2 illustrates how the Apriori property works. If itemset $\{\texttt{B}, \texttt{C}, \texttt{D}\}$ is frequent, then all the subsets of this itemset, shaded, must also be frequent itemsets. The Apriori property provides the basis for the Apriori algorithm.

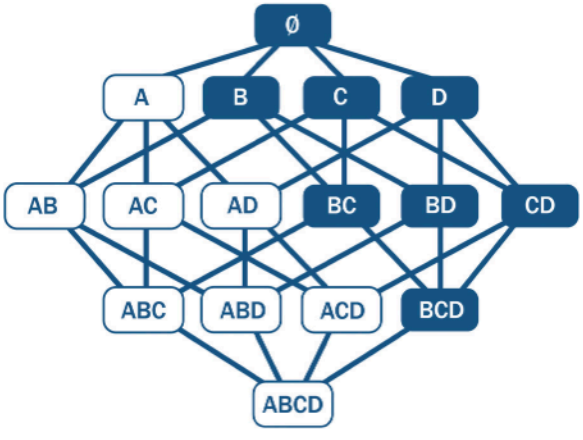


FIGURE 5-2 *Itemset {A,B,C,D} and its subsets*

5.2 Apriori Algorithm

The Apriori algorithm takes a bottom-up iterative approach to uncovering the frequent itemsets by first determining all the possible items (or 1-itemsets, for example `{bread}`, `{eggs}`, `{milk}`, ...) and then identifying which among them are frequent.

Assuming the minimum support threshold (or the minimum support criterion) is set at 0.5, the algorithm identifies and retains those itemsets that appear in at least 50% of all transactions and discards (or "prunes away") the itemsets that have a support less than 0.5 or appear in fewer than 50% of the transactions. The word ***prune*** is used like it would be in gardening, where unwanted branches of a bush are clipped away.

In the next iteration of the Apriori algorithm, the identified frequent 1-itemsets are paired into 2-itemsets (for example, `{bread,eggs}`, `{bread,milk}`, `{eggs,milk}`, ...) and again evaluated to identify the frequent 2-itemsets among them.

At each iteration, the algorithm checks whether the support criterion can be met; if it can, the algorithm grows the itemset, repeating the process until it runs out of support or until the itemsets reach a predefined length. The Apriori algorithm [5] is given next. Let variable C_k be the set of candidate k-itemsets and variable L_k be the set of k-itemsets that satisfy the minimum support. Given a transaction database D, a minimum support threshold δ, and an optional parameter N indicating the maximum length an itemset could reach, Apriori iteratively computes frequent itemsets L_{k+1} based on L_k.

```
Apriori (D, δ, N)
  k ← 1
  L_k ← {1-itemsets that satisfy minimum support δ}
  while L_k ≠ ∅
    if ∄N ∨ (∃N ∧ k < N)
```

```
        C_{k+1} ← candidate itemsets generated from L_k
        for each transaction t in database D do
          increment the counts of C_{k+1} contained in t
        L_{k+1} ← candidates in C_{k+1} that satisfy minimum support δ
        k ← k+1
    return ∪_k L_k
```

The first step of the Apriori algorithm is to identify the frequent itemsets by starting with each item in the transactions that meets the predefined minimum support threshold δ. These itemsets are 1-itemsets denoted as L_1, as each 1-itemset contains only one item. Next, the algorithm grows the itemsets by joining L_1 onto itself to form new, grown 2-itemsets denoted as L_2 and determines the support of each 2-itemset in L_2. Those itemsets that do not meet the minimum support threshold δ are pruned away. The growing and pruning process is repeated until no itemsets meet the minimum support threshold. Optionally, a threshold N can be set up to specify the maximum number of items the itemset can reach or the maximum number of iterations of the algorithm. Once completed, output of the Apriori algorithm is the collection of all the frequent k-itemsets.

Next, a collection of candidate rules is formed based on the frequent itemsets uncovered in the iterative process described earlier. For example, a frequent itemset {milk, eggs} may suggest candidate rules {milk}→{eggs} and {eggs}→{milk}.

5.3 Evaluation of Candidate Rules

Frequent itemsets from the previous section can form candidate rules such as X implies Y (X → Y). This section discusses how measures such as confidence, lift, and leverage can help evaluate the appropriateness of these candidate rules.

Confidence [2] is defined as the measure of certainty or trustworthiness associated with each discovered rule. Mathematically, confidence is the percent of transactions that contain both X and Y out of all the transactions that contain X (see Equation 5-1).

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)} \tag{5-1}$$

For example, if {bread, eggs, milk} has a support of 0.15 and {bread, eggs} also has a support of 0.15, the confidence of rule {bread, eggs}→{milk} is 1, which means 100% of the time a customer buys bread and eggs, milk is bought as well. The rule is therefore correct for 100% of the transactions containing bread and eggs.

A relationship may be thought of as interesting when the algorithm identifies the relationship with a measure of confidence greater than or equal to a predefined threshold. This predefined threshold is called the ***minimum confidence***. A higher confidence indicates that the rule (X → Y) is more interesting or more trustworthy, based on the sample dataset.

So far, this chapter has talked about two common measures that the Apriori algorithm uses: support and confidence. All the rules can be ranked based on these two measures to filter out the uninteresting rules and retain the interesting ones.

Even though confidence can identify the interesting rules from all the candidate rules, it comes with a problem. Given rules in the form of $X \rightarrow Y$, confidence considers only the antecedent (X) and the co-occurrence of X and Y; it does not take the consequent of the rule (Y) into concern. Therefore, confidence cannot tell if a rule contains true implication of the relationship or if the rule is purely coincidental. X and Y can be statistically independent yet still receive a high confidence score. Other measures such as lift [6] and leverage [7] are designed to address this issue.

Lift measures how many times more often X and Y occur together than expected if they are statistically independent of each other. Lift is a measure [6] of how X and Y are really related rather than coincidentally happening together (see Equation 5-2).

$$Lift(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X) * Support(Y)} \tag{5-2}$$

Lift is 1 if X and Y are statistically independent of each other. In contrast, a lift of $X \rightarrow Y$ greater than 1 indicates that there is some usefulness to the rule. A larger value of lift suggests a greater strength of the association between X and Y.

Assuming 1,000 transactions, with `{milk, eggs}` appearing in 300 of them, `{milk}` appearing in 500, and `{eggs}` appearing in 400, then $Lift(milk \rightarrow eggs) = 0.3/(0.5 * 0.4) = 1.5$. If `{bread}` appears in 400 transactions and `{milk, bread}` appears in 400, then $Lift(milk \rightarrow bread) = 0.4/(0.5 * 0.4) = 2$. Therefore it can be concluded that milk and bread have a stronger association than milk and eggs.

Leverage [7] is a similar notion, but instead of using a ratio, leverage uses the difference (see Equation 5-3). Leverage measures the difference in the probability of X and Y appearing together in the dataset compared to what would be expected if X and Y were statistically independent of each other.

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y) \tag{5-3}$$

In theory, leverage is 0 when X and Y are statistically independent of each other. If X and Y have some kind of relationship, the leverage would be greater than zero. A larger leverage value indicates a stronger relationship between X and Y. For the previous example, $Leverage(milk \rightarrow eggs) = 0.3 - (0.5 * 0.4) = 0.1$ and $Leverage(milk \rightarrow bread) = 0.4 - (0.5 * 0.4) = 0.2$. It again confirms that milk and bread have a stronger association than milk and eggs.

Confidence is able to identify trustworthy rules, but it cannot tell whether a rule is coincidental. A high-confidence rule can sometimes be misleading because confidence does not consider support of the itemset in the rule consequent. Measures such as lift and leverage not only ensure interesting rules are identified but also filter out the coincidental rules.

This chapter has discussed four measures of significance and interestingness for association rules: support, confidence, lift, and leverage. These measures ensure the discovery of interesting and strong rules from sample datasets. Besides these four rules, there are other alternative measures, such as correlation [8], collective strength [9], conviction [6], and coverage [10]. Refer to the Bibliography to learn how these measures work.

5.4 Applications of Association Rules

The term ***market basket analysis*** refers to a specific implementation of association rules mining that many companies use for a variety of purposes, including these:

- Broad-scale approaches to better merchandising—what products should be included in or excluded from the inventory each month
- Cross-merchandising between products and high-margin or high-ticket items
- Physical or logical placement of product within related categories of products
- Promotional programs—multiple product purchase incentives managed through a loyalty card program

Besides market basket analysis, association rules are commonly used for recommender systems [11] and clickstream analysis [12].

Many online service providers such as Amazon and Netflix use recommender systems. Recommender systems can use association rules to discover related products or identify customers who have similar interests. For example, association rules may suggest that those customers who have bought product A have also bought product B, or those customers who have bought products A, B, and C are more similar to this customer. These findings provide opportunities for retailers to cross-sell their products.

Clickstream analysis refers to the analytics on data related to web browsing and user clicks, which is stored on the client or the server side. Web usage log files generated on web servers contain huge amounts of information, and association rules can potentially give useful knowledge to web usage data analysts. For example, association rules may suggest that website visitors who land on page X click on links A, B, and C much more often than links D, E, and F. This observation provides valuable insight on how to better personalize and recommend the content to site visitors.

The next section shows an example of grocery store transactions and demonstrates how to use R to perform association rule mining.

5.5 An Example: Transactions in a Grocery Store

An example illustrates the application of the Apriori algorithm to a relatively simple case that generalizes to those used in practice. Using R and the `arules` and `arulesViz` packages, this example shows how to use the Apriori algorithm to generate frequent itemsets and rules and to evaluate and visualize the rules.

The following commands install these two packages and import them into the current R workspace:

```
install.packages('arules')
install.packages('arulesViz')

library('arules')
library('arulesViz')
```

5.5.1 The Groceries Dataset

The example uses the *Groceries* dataset from the R `arules` package. The *Groceries* dataset is collected from 30 days of real-world point-of-sale transactions of a grocery store. The dataset contains 9,835 transactions, and the items are aggregated into 169 categories.

```
data(Groceries)
Groceries
transactions in sparse format with
 9835 transactions (rows) and
 169 items (columns)
```

The summary shows that the most frequent items in the dataset include items such as whole milk, other vegetables, rolls/buns, soda, and yogurt. These items are purchased more often than the others.

```
summary(Groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda
            2513             1903             1809             1715
          yogurt          (Other)
            1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77
  15   16   17   18   19   20   21   22   23   24   26   27   28   29
  55   46   29   14   14    9   11    4    6    1    1    1    1    3
  32
   1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000

includes extended item information - examples:
       labels  level2           level1
1 frankfurter sausage meet and sausage
2     sausage sausage meet and sausage
3  liver loaf sausage meet and sausage
```

The class of the dataset is `transactions`, as defined by the `arules` package. The `transactions` class contains three slots:

- **`transactionInfo`:** A data frame with vectors of the same length as the number of transactions
- **`itemInfo`:** A data frame to store item labels
- **`data`:** A binary incidence matrix that indicates which item labels appear in every transaction

```
class(Groceries)
[1] "transactions"
attr(,"package")
[1] "arules"
```

For the *Groceries* dataset, the `transactionInfo` is not being used. Enter `Groceries@itemInfo` to display all 169 grocery labels as well as their categories. The following command displays only the first 20 grocery labels. Each grocery label is mapped to two levels of categories—`level2` and `level1`—where `level1` is a superset of `level2`. For example, grocery label `sausage` belongs to the `sausage` category in `level2`, and it is part of the `meat and sausage` category in `level1`. (Note that "`meet`" in `level1` is a typo in the dataset.)

```
Groceries@itemInfo[1:20,]
              labels     level2               level1
1        frankfurter    sausage     meet and sausage
2            sausage    sausage     meet and sausage
3         liver loaf    sausage     meet and sausage
4                ham    sausage     meet and sausage
5               meat    sausage     meet and sausage
6  finished products    sausage     meet and sausage
7    organic sausage    sausage     meet and sausage
8            chicken    poultry     meet and sausage
9             turkey    poultry     meet and sausage
10              pork       pork     meet and sausage
11              beef       beef     meet and sausage
12    hamburger meat       beef     meet and sausage
13              fish       fish     meet and sausage
14      citrus fruit      fruit fruit and vegetables
15    tropical fruit      fruit fruit and vegetables
16         pip fruit      fruit fruit and vegetables
17            grapes      fruit fruit and vegetables
18           berries      fruit fruit and vegetables
19       nuts/prunes      fruit fruit and vegetables
20   root vegetables vegetables fruit and vegetables
```

The following code displays the 10th to 20th transactions of the `Groceries` dataset. The `[10:20]` can be changed to `[1:9835]` to display all the transactions.

```
apply(Groceries@data[,10:20], 2,
      function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")
      )
```

Each row in the output shows a transaction that includes one or more products, and each transaction corresponds to everything in a customer's shopping cart. For example, in the first transaction, a customer has purchased whole milk and cereals.

```
 [1] "whole milk, cereals"
 [2] "tropical fruit, other vegetables, white bread, bottled water,
chocolate"
 [3] "citrus fruit, tropical fruit, whole milk, butter, curd, yogurt,
flour, bottled water, dishes"
 [4] "beef"
```

```
 [5] "frankfurter, rolls/buns, soda"
 [6] "chicken, tropical fruit"
 [7] "butter, sugar, fruit/vegetable juice, newspapers"
 [8] "fruit/vegetable juice"
 [9] "packaged fruit/vegetables"
[10] "chocolate"
[11] "specialty bar"
```

The next section shows how to generate frequent itemsets from the `Groceries` dataset.

5.5.2 Frequent Itemset Generation

The `apriori()` function from the `arule` package implements the Apriori algorithm to create frequent itemsets. Note that, by default, the `apriori()` function executes all the iterations at once. However, to illustrate how the Apriori algorithm works, the code examples in this section manually set the parameters of the `apriori()` function to simulate each iteration of the algorithm.

Assume that the minimum support threshold is set to 0.02 based on management discretion. Because the dataset contains 9,853 transactions, an itemset should appear at least 198 times to be considered a frequent itemset. The first iteration of the Apriori algorithm computes the support of each product in the dataset and retains those products that satisfy the minimum support. The following code identifies 59 frequent 1-itemsets that satisfy the minimum support. The parameters of `apriori()` specify the minimum and maximum lengths of the itemsets, the minimum support threshold, and the target indicating the type of association mined.

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, maxlen=1,
                       support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      1
 maxlen            target   ext
      1 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 done [0.00s].
writing ... [59 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The summary of the itemsets shows that the support of 1-itemsets ranges from 0.02105 to 0.25552. Because the maximum support of the 1-itemsets in the dataset is only 0.25552, to enable the discovery of interesting rules, the minimum support threshold should not be set too close to that number.

```
summary(itemsets)
set of 59 itemsets

most frequent items:
frankfurter     sausage         ham        meat     chicken
          1           1           1           1           1
    (Other)
         54

element (itemset/transaction) length distribution:sizes
 1
59

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       1       1       1       1

summary of quality measures:
    support
 Min.   :0.02105
 1st Qu.:0.03015
 Median :0.04809
 Mean   :0.06200
 3rd Qu.:0.07666
 Max.   :0.25552

includes transaction ID lists: FALSE

mining info:
      data ntransactions support confidence
 Groceries          9835    0.02          1
```

The following code uses the `inspect()` function to display the top 10 frequent 1-itemsets sorted by their support. Of all the transaction records, the 59 1-itemsets such as {`whole milk`}, {`other vegetables`}, {`rolls/buns`}, {`soda`}, and {`yogurt`} all satisfy the minimum support. Therefore, they are called frequent 1-itemsets.

```
inspect(head(sort(itemsets, by = "support"), 10))
   items                     support
1  {whole milk}            0.25551601
2  {other vegetables}      0.19349263
3  {rolls/buns}            0.18393493
4  {soda}                  0.17437722
5  {yogurt}                0.13950178
6  {bottled water}         0.11052364
```

```
7  {root vegetables}       0.10899847
8  {tropical fruit}        0.10493137
9  {shopping bags}         0.09852567
10 {sausage}               0.09395018
```

In the next iteration, the list of frequent 1-itemsets is joined onto itself to form all possible candidate 2-itemsets. For example, 1-itemsets {whole milk} and {soda} would be joined to become a 2-itemset {whole milk, soda}. The algorithm computes the support of each candidate 2-itemset and retains those that satisfy the minimum support. The output that follows shows that 61 frequent 2-itemsets have been identified.

```
itemsets <- apriori(Groceries, parameter=list(minlen=2, maxlen=2,
                        support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      2
 maxlen            target   ext
      2 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [61 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The summary of the itemsets shows that the support of 2-itemsets ranges from 0.02003 to 0.07483.

```
summary(itemsets)
set of 61 itemsets

most frequent items:
      whole milk other vegetables           yogurt       rolls/buns
              25               17                9                9
            soda          (Other)
               9               53

element (itemset/transaction) length distribution:sizes
 2
61
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

summary of quality measures:
    support
 Min.   :0.02003
 1st Qu.:0.02227
 Median :0.02613
 Mean   :0.02951
 3rd Qu.:0.03223
 Max.   :0.07483

includes transaction ID lists: FALSE

mining info:
      data ntransactions support confidence
 Groceries          9835    0.02          1
```

The top 10 most frequent 2-itemsets are displayed next, sorted by their support. Notice that whole milk appears six times in the top 10 2-itemsets ranked by support. As seen earlier, {`whole milk`} has the highest support among all the 1-itemsets. These top 10 2-itemsets with the highest support may not be interesting; this highlights the limitations of using support alone.

```
inspect(head(sort(itemsets, by ="support"),10))
   items                  support
1  {other vegetables,
    whole milk}       0.07483477
2  {whole milk,
    rolls/buns}       0.05663447
3  {whole milk,
    yogurt}           0.05602440
4  {root vegetables,
    whole milk}       0.04890696
5  {root vegetables,
    other vegetables} 0.04738180
6  {other vegetables,
    yogurt}           0.04341637
7  {other vegetables,
    rolls/buns}       0.04260295
8  {tropical fruit,
    whole milk}       0.04229792
9  {whole milk,
    soda}             0.04006101
10 {rolls/buns,
    soda}             0.03833249
```

Next, the list of frequent 2-itemsets is joined onto itself to form candidate 3-itemsets. For example {`other vegetables,whole milk`} and {`whole milk,rolls/buns`} would be joined as {`other vegetables,whole milk,rolls/buns`}. The algorithm retains those itemsets

that satisfy the minimum support. The following output shows that only two frequent 3-itemsets have been identified.

```
itemsets <- apriori(Groceries, parameter=list(minlen=3, maxlen=3,
                        support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      3
 maxlen            target   ext
      3 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [2 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The 3-itemsets are displayed next:

```
inspect(sort(itemsets, by ="support"))
  items                  support
1 {root vegetables,
   other vegetables,
   whole milk}       0.02318251
2 {other vegetables,
   whole milk,
   yogurt}           0.02226741
```

In the next iteration, there is only one candidate 4-itemset `{root vegetables,other vegetables,whole milk,yogurt}`, and its support is below 0.02. No frequent 4-itemsets have been found, and the algorithm converges.

```
itemsets <- apriori(Groceries, parameter=list(minlen=4, maxlen=4,
                        support=0.02, target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      4
 maxlen            target   ext
      4 frequent itemsets FALSE
```

```
algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [0 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The previous steps simulate the Apriori algorithm at each iteration. For the *Groceries* dataset, the iterations run out of support when $k = 4$. Therefore, the frequent itemsets contain 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

When the `maxlen` parameter is not set, the algorithm continues each iteration until it runs out of support or until k reaches the default `maxlen=10`. As shown in the code output that follows, 122 frequent itemsets have been identified. This matches the total number of 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

```
itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,
                                        target="frequent itemsets"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.8    0.1    1 none FALSE            TRUE    0.02      1
 maxlen            target   ext
     10 frequent itemsets FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [122 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Note that the results are assessed based on the specific business context of the exercise using the specific dataset. If the dataset changes or a different minimum support threshold is chosen, the Apriori algorithm must run each iteration again to retrieve the updated frequent itemsets.

5.5.3 Rule Generation and Visualization

The `apriori()` function can also be used to generate rules. Assume that the minimum support threshold is now set to a lower value 0.001, and the minimum confidence threshold is set to 0.6. A lower minimum support threshold allows more rules to show up. The following code creates 2,918 rules from all the transactions in the *Groceries* dataset that satisfy both the minimum support and the minimum confidence.

```
rules <- apriori(Groceries, parameter=list(support=0.001,
                        confidence=0.6, target = "rules"))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen
        0.6    0.1    1 none FALSE            TRUE   0.001      1
 maxlen target   ext
     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
writing ... [2918 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
```

The summary of the rules shows the number of rules and ranges of the support, confidence, and lift.

```
summary(rules)
set of 2918 rules

rule length distribution (lhs + rhs):sizes
   2    3    4    5    6
   3  490 1765  626   34

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.000   4.000   4.068   4.000   6.000

summary of quality measures:
    support           confidence          lift
 Min.   :0.001017   Min.   :0.6000   Min.   : 2.348
 1st Qu.:0.001118   1st Qu.:0.6316   1st Qu.: 2.668
 Median :0.001220   Median :0.6818   Median : 3.168
 Mean   :0.001480   Mean   :0.7028   Mean   : 3.450
 3rd Qu.:0.001525   3rd Qu.:0.7500   3rd Qu.: 3.692
 Max.   :0.009354   Max.   :1.0000   Max.   :18.996
```

```
mining info:
      data ntransactions support confidence
 Groceries          9835   0.001        0.6
```

Enter `plot(rules)` to display the scatterplot of the 2,918 rules (Figure 5-3), where the horizontal axis is the support, the vertical axis is the confidence, and the shading is the lift. The scatterplot shows that, of the 2,918 rules generated from the *Groceries* dataset, the highest lift occurs at a low support and a low confidence.

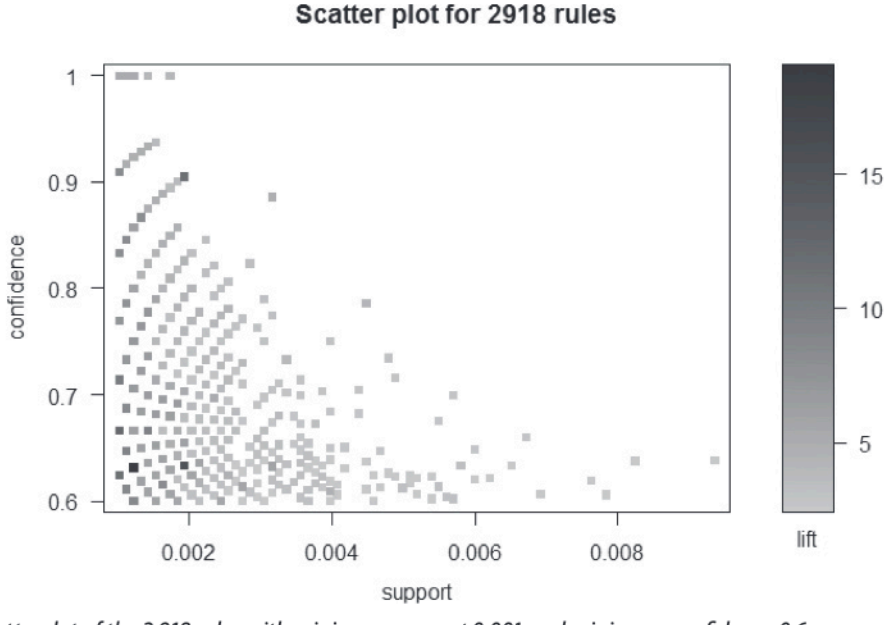


FIGURE 5-3 *Scatterplot of the 2,918 rules with minimum support 0.001 and minimum confidence 0.6*

Entering `plot(rules@quality)` displays a scatterplot matrix (Figure 5-4) to compare the support, confidence, and lift of the 2,918 rules.

Figure 5-4 shows that lift is proportional to confidence and illustrates several linear groupings. As indicated by Equation 5-2 and Equation 5-3, $Lift = Confidence / Support(Y)$. Therefore, when the support of Y remains the same, lift is proportional to confidence, and the slope of the linear trend is the reciprocal of $Support(Y)$. The following code shows that, of the 2,918 rules, there are only 18 different values for $\frac{1}{Support(Y)}$, and the majority occurs at slopes 3.91, 5.17, 7.17, 9.17, and 9.53. This matches the slopes shown in the third row and second column of Figure 5-4, where the x-axis is the confidence and the y-axis is the lift.

```
# compute the 1/Support(Y)
slope <- sort(round(rules@quality$lift / rules@quality$confidence, 2))
# Display the number of times each slope appears in the dataset
unlist(lapply(split(slope,f=slope),length))
```

```
 3.91  5.17  5.44  5.73  7.17  9.05  9.17  9.53 10.64 12.08
 1585   940    12     7   188     1   102    55     1     4
12.42 13.22 13.83 13.95 18.05 23.76 26.44 30.08
1     5     2     9     3     1     1     1
```

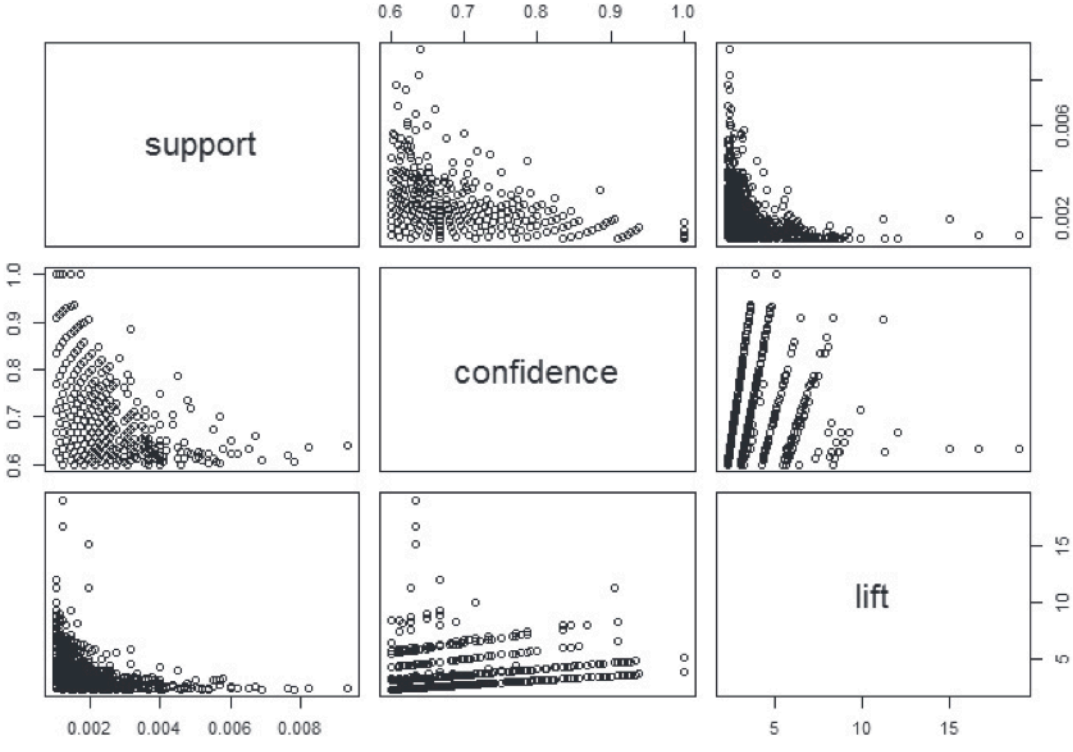


FIGURE 5-4 *Scatterplot matrix on the support, confidence, and lift of the 2,918 rules*

The `inspect()` function can display content of the rules generated previously. The following code shows the top ten rules sorted by the lift. Rule `{Instant food products,soda}→{hamburger meat}` has the highest lift of 18.995654.

```
inspect(head(sort(rules, by="lift"), 10))
   lhs                          rhs
support  confidence   lift
1  {Instant food products,
    soda}                   => {hamburger meat}
0.001220132  0.6315789 18.995654
2  {soda,
    popcorn}                => {salty snack}
0.001220132  0.6315789 16.697793
3  {ham,
    processed cheese}       => {white bread}
0.001931876  0.6333333 15.045491
```

```
4  {tropical fruit,
    other vegetables,
    yogurt,
    white bread}           => {butter}
0.001016777  0.6666667 12.030581
5  {hamburger meat,
    yogurt,
    whipped/sour cream}    => {butter}
0.001016777  0.6250000 11.278670
6  {tropical fruit,
    other vegetables,
    whole milk,
    yogurt,
    domestic eggs}         => {butter}
0.001016777  0.6250000 11.278670
7  {liquor,
    red/blush wine}        => {bottled beer}
0.001931876  0.9047619 11.235269
8  {other vegetables,
    butter,
    sugar}                 => {whipped/sour cream}
0.001016777  0.7142857  9.964539
9  {whole milk,
    butter,
    hard cheese}           => {whipped/sour cream}
0.001423488  0.6666667  9.300236
10 {tropical fruit,
    other vegetables,
    butter,
     fruit/vegetable juice} => {whipped/sour cream}
0.001016777  0.6666667  9.300236
```

The following code fetches a total of 127 rules whose confidence is above 0.9:

```
confidentRules <- rules[quality(rules)$confidence > 0.9]
confidentRules
set of 127 rules
```

The next command produces a matrix-based visualization (Figure 5-5) of the LHS versus the RHS of the rules. The legend on the right is a color matrix indicating the lift and the confidence to which each square in the main matrix corresponds.

```
plot(confidentRules, method="matrix", measure=c("lift", "confidence"),
     control=list(reorder=TRUE))
```

As the previous `plot()` command runs, the R console would simultaneously display a distinct list of the LHS and RHS from the 127 rules. A segment of the output is shown here:

```
Itemsets in Antecedent (LHS)
  [1] "{citrus fruit,other vegetables,soda,fruit/vegetable juice}"
  [2] "{tropical fruit,other vegetables,whole milk,yogurt,oil}"
```

```
  [3] "{tropical fruit,butter,whipped/sour cream,fruit/vegetable
juice}"
  [4] "{tropical fruit,grapes,whole milk,yogurt}"
  [5] "{ham,tropical fruit,pip fruit,whole milk}"
...
[124] "{liquor,red/blush wine}"
Itemsets in Consequent (RHS)
[1] "{whole milk}"       "{yogurt}"           "{root vegetables}"
[4] "{bottled beer}"     "{other vegetables}"
```

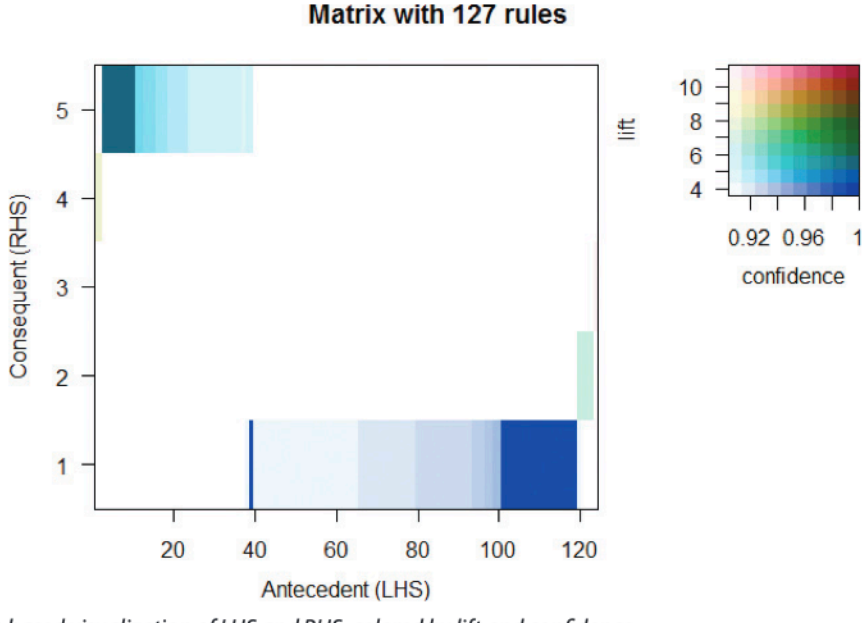


FIGURE 5-5 *Matrix-based visualization of LHS and RHS, colored by lift and confidence*

The following code provides a visualization of the top five rules with the highest lift. The plot is shown in Figure 5-6. In the graph, the arrow always points from an item on the LHS to an item on the RHS. For example, the arrows that connect ham, processed cheese, and white bread suggest rule `{ham,processed cheese}→{white bread}`. The legend on the top right of the graph shows that the size of a circle indicates the support of the rules ranging from 0.001 to 0.002. The color (or shade) represents the lift, which ranges from 11.279 to 18.996. The rule with the highest lift is `{Instant food products,soda} → {hamburger meat}`.

```
highLiftRules <- head(sort(rules, by="lift"), 5)
plot(highLiftRules, method="graph", control=list(type="items"))
```

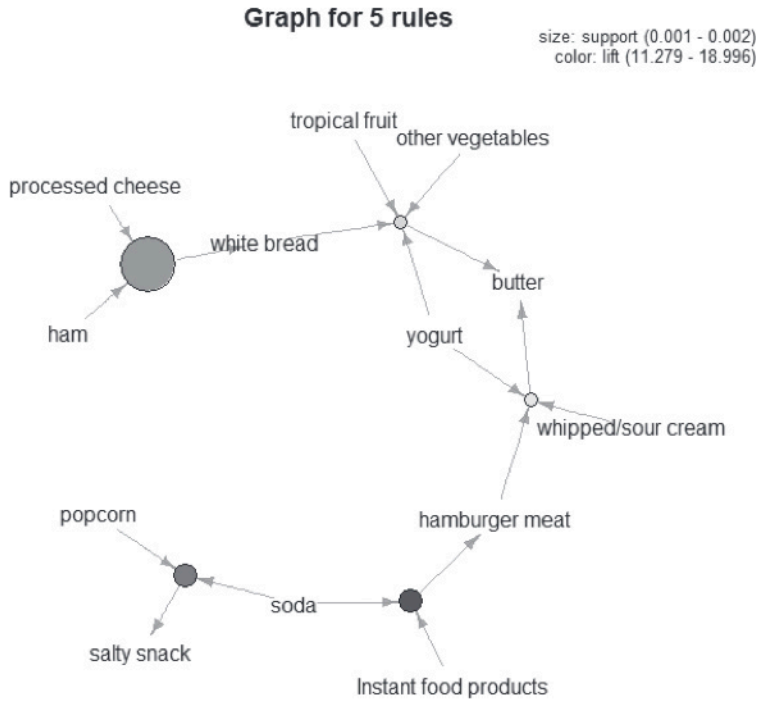


FIGURE 5-6 *Graph visualization of the top five rules sorted by lift*

5.6 Validation and Testing

After gathering the output rules, it may become necessary to use one or more methods to validate the results in the business context for the sample dataset. The first approach can be established through statistical measures such as confidence, lift, and leverage. Rules that involve mutually independent items or cover few transactions are considered uninteresting because they may capture spurious relationships.

As mentioned in Section 5.3, confidence measures the chance that X and Y appear together in relation to the chance X appears. Confidence can be used to identify the interestingness of the rules.

Lift and leverage both compare the support of X and Y against their individual support. While mining data with association rules, some rules generated could be purely coincidental. For example, if 95% of customers buy X and 90% of customers buy Y, then X and Y would occur together at least 85% of the time, even if there is no relationship between the two. Measures like lift and leverage ensure that interesting rules are identified rather than coincidental ones.

Another set of criteria can be established through subjective arguments. Even with a high confidence, a rule may be considered subjectively uninteresting unless it reveals any unexpected profitable actions. For example, rules like `{paper}→{pencil}` may not be subjectively interesting or meaningful despite high support and confidence values. In contrast, a rule like `{diaper}→{beer}` that satisfies both minimum support and minimum confidence can be considered subjectively interesting because this rule is

unexpected and may suggest a cross-sell opportunity for the retailer. This incorporation of subjective knowledge into the evaluation of rules can be a difficult task, and it requires collaboration with domain experts. As seen in Chapter 2, "Data Analytics Lifecycle," the domain experts may serve as the business users or the business intelligence analysts as part of the Data Science team. In Phase 5, the team can communicate the results and decide if it is appropriate to operationalize them.

5.7 Diagnostics

Although the Apriori algorithm is easy to understand and implement, some of the rules generated are uninteresting or practically useless. Additionally, some of the rules may be generated due to coincidental relationships between the variables. Measures like confidence, lift, and leverage should be used along with human insights to address this problem.

Another problem with association rules is that, in Phase 3 and 4 of the Data Analytics Lifecycle (Chapter 2), the team must specify the minimum support prior to the model execution, which may lead to too many or too few rules. In related research, a variant of the algorithm [13] can use a predefined target range for the number of rules so that the algorithm can adjust the minimum support accordingly.

Section 5.2 presented the Apriori algorithm, which is one of the earliest and the most fundamental algorithms for generating association rules. The Apriori algorithm reduces the computational workload by only examining itemsets that meet the specified minimum threshold. However, depending on the size of the dataset, the Apriori algorithm can be computationally expensive. For each level of support, the algorithm requires a scan of the entire database to obtain the result. Accordingly, as the database grows, it takes more time to compute in each run. Here are some approaches to improve Apriori's efficiency:

- **Partitioning:** Any itemset that is potentially frequent in a transaction database must be frequent in at least one of the partitions of the transaction database.
- **Sampling:** This extracts a subset of the data with a lower support threshold and uses the subset to perform association rule mining.
- **Transaction reduction:** A transaction that does not contain frequent ***k***-itemsets is useless in subsequent scans and therefore can be ignored.
- **Hash-based itemset counting:** If the corresponding hashing bucket count of a ***k***-itemset is below a certain threshold, the ***k***-itemset cannot be frequent.
- **Dynamic itemset counting:** Only add new candidate itemsets when all of their subsets are estimated to be frequent.

Summary

As an unsupervised analysis technique that uncovers relationships among items, association rules find many uses in activities, including market basket analysis, clickstream analysis, and recommendation engines. Although association rules are not used to predict outcomes or behaviors, they are good at identifying "interesting" relationships within items from a large dataset. Quite often, the disclosed relationships that the association rules suggest do not seem obvious; they, therefore, provide valuable insights for institutions to improve their business operations.

The Apriori algorithm is one of the earliest and most fundamental algorithms for association rules. This chapter used a grocery store example to walk through the steps of Apriori and generate frequent *k*-itemsets and useful rules for downstream analysis and visualization. A few measures such as support, confidence, lift, and leverage were discussed. These measures together help identify the interesting rules and eliminate the coincidental rules. Finally, the chapter discussed some pros and cons of the Apriori algorithm and highlighted a few methods to improve its efficiency.

Exercises

1. What is the Apriori property?

2. Following is a list of five transactions that include items A, B, C, and D:

   - T1 : { `A,B,C` }
   - T2 : { `A,C` }
   - T3 : { `B,C` }
   - T4 : { `A,D` }
   - T5 : { `A,C,D` }

   Which itemsets satisfy the minimum support of 0.5? (Hint: An itemset may include more than one item.)

3. How are interesting rules identified? How are interesting rules distinguished from coincidental rules?

4. A local retailer has a database that stores 10,000 transactions of last summer. After analyzing the data, a data science team has identified the following statistics:

   - {`battery`} appears in 6,000 transactions.
   - {`sunscreen`} appears in 5,000 transactions.
   - {`sandals`} appears in 4,000 transactions.
   - {`bowls`} appears in 2,000 transactions.
   - {`battery,sunscreen`} appears in 1,500 transactions.
   - {`battery,sandals`} appears in 1,000 transactions.
   - {`battery,bowls`} appears in 250 transactions.
   - {`battery,sunscreen,sandals`} appears in 600 transactions.

   Answer the following questions:

   a. What are the support values of the preceding itemsets?

   b. Assuming the minimum support is 0.05, which itemsets are considered frequent?

   c. What are the confidence values of {`battery`}→{`sunscreen`} and {`battery,sunscreen`}→{`sandals`}? Which of the two rules is more interesting?

   d. List all the candidate rules that can be formed from the statistics. Which rules are considered interesting at the minimum confidence 0.25? Out of these interesting rules, which rule is considered the most useful (that is, least coincidental)?

Bibliography

[1] P. Hájek, I. Havel, and M. Chytil, "The GUHA Method of Automatic Hypotheses Determination," *Computing*, vol. 1, no. 4, pp. 293–308, 1966.

[2] R. Agrawal, T. Imieliński, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *SIGMOD '93 Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207–216, 1993.

[3] M.-S. Chen, J. S. Park, and P. Yu, "Efficient Data Mining for Path Traversal Patterns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 2, pp. 209–221, 1998.

[4] R. Cooley, B. Mobasher, and J. Srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web," *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*, pp. 558–567, 1997.

[5] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, 1994.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *SIGMOD*, vol. 26, no. 2, pp. 255–264, 1997.

[7] G. Piatetsky-Shapiro, "Discovery, Analysis and Presentation of Strong Rules," *Knowledge Discovery in Databases*, pp. 229–248, 1991.

[8] S. Brin, R. Motwani, and C. Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations," *Proceedings of the ACM SIGMOD/PODS '97 Joint Conference*, vol. 26, no. 2, pp. 265–276, 1997.

[9] C. C. Aggarwal and P. S. Yu, "A New Framework for Itemset Generation," in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*, Seattle, Washington, USA, 1998.

[10] M. Hahsler, "A Comparison of Commonly Used Interest Measures for Association Rules," 9 March 2011. [Online]. Available: `http://michael.hahsler.net/research/association_rules/measures.html`. [Accessed 4 March 2014].

[11] W. Lin, S. A. Alvarez, and C. Ruiz, "Efficient Adaptive-Support Association Rule Mining for Recommender Systems," *Data Mining and Knowledge Discovery*, vol. 6, no. 1, pp. 83–105, 2002.

[12] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective Personalization Based on Association Rule Discovery from Web Usage Data," in ***ACM***, 2011.

[13] W. Lin, S. A. Alvarez, and C. Ruiz, "Collaborative Recommendation via Adaptive Association Rule Mining," in *Proceedings of the International Workshop on Web Mining for E-Commerce (WEBKDD)*, Boston, MA, 2000.

6

Advanced Analytical Theory and Methods: Regression

Key Concepts

Categorical Variable
Linear Regression
Logistic Regression
Ordinary Least Squares (OLS)
Receiver Operating Characteristic (ROC) Curve
Residuals

In general, regression analysis attempts to explain the influence that a set of variables has on the outcome of another variable of interest. Often, the outcome variable is called a ***dependent variable*** because the outcome depends on the other variables. These additional variables are sometimes called the ***input variables*** or the ***independent variables***. Regression analysis is useful for answering the following kinds of questions:

- What is a person's expected income?
- What is the probability that an applicant will default on a loan?

Linear regression is a useful tool for answering the first question, and logistic regression is a popular method for addressing the second. This chapter examines these two regression techniques and explains when one technique is more appropriate than the other.

Regression analysis is a useful explanatory tool that can identify the input variables that have the greatest statistical influence on the outcome. With such knowledge and insight, environmental changes can be attempted to produce more favorable values of the input variables. For example, if it is found that the reading level of 10-year-old students is an excellent predictor of the students' success in high school and a factor in their attending college, then additional emphasis on reading can be considered, implemented, and evaluated to improve students' reading levels at a younger age.

6.1 Linear Regression

Linear regression is an analytical technique used to model the relationship between several input variables and a continuous outcome variable. A key assumption is that the relationship between an input variable and the outcome variable is linear. Although this assumption may appear restrictive, it is often possible to properly transform the input or outcome variables to achieve a linear relationship between the modified input and outcome variables. Possible transformations will be covered in more detail later in the chapter.

The physical sciences have well-known linear models, such as Ohm's Law, which states that the electrical current flowing through a resistive circuit is linearly proportional to the voltage applied to the circuit. Such a model is considered deterministic in the sense that if the input values are known, the value of the outcome variable is precisely determined. A linear regression model is a probabilistic one that accounts for the randomness that can affect any particular outcome. Based on known input values, a linear regression model provides the expected value of the outcome variable based on the values of the input variables, but some uncertainty may remain in predicting any particular outcome. Thus, linear regression models are useful in physical and social science applications where there may be considerable variation in a particular outcome based on a given set of input values. After presenting possible linear regression use cases, the foundations of linear regression modeling are provided.

6.1.1 Use Cases

Linear regression is often used in business, government, and other scenarios. Some common practical applications of linear regression in the real world include the following:

- **Real estate:** A simple linear regression analysis can be used to model residential home prices as a function of the home's living area. Such a model helps set or evaluate the list price of a home on the market. The model could be further improved by including other input variables such as number of bathrooms, number of bedrooms, lot size, school district rankings, crime statistics, and property taxes.

- **Demand forecasting:** Businesses and governments can use linear regression models to predict demand for goods and services. For example, restaurant chains can appropriately prepare for the predicted type and quantity of food that customers will consume based upon the weather, the day of the week, whether an item is offered as a special, the time of day, and the reservation volume. Similar models can be built to predict retail sales, emergency room visits, and ambulance dispatches.
- **Medical:** A linear regression model can be used to analyze the effect of a proposed radiation treatment on reducing tumor sizes. Input variables might include duration of a single radiation treatment, frequency of radiation treatment, and patient attributes such as age or weight.

6.1.2 Model Description

As the name of this technique suggests, the linear regression model assumes that there is a linear relationship between the input variables and the outcome variable. This relationship can be expressed as shown in Equation 6-1.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{p-1} x_{p-1} + \varepsilon \tag{6-1}$$

where:

y is the outcome variable

x_j are the input variables, for $j = 1, 2, \ldots, p - 1$

β_0 is the value of y when each x_j equals zero

β_j is the change in y based on a unit change in x_j, for $j = 1, 2, \ldots, p - 1$

ε is a random error term that represents the difference in the linear model and a particular observed value for y

Suppose it is desired to build a linear regression model that estimates a person's annual income as a function of two variables—age and education—both expressed in years. In this case, income is the outcome variable, and the input variables are age and education. Although it may be an over generalization, such a model seems intuitively correct in the sense that people's income should increase as their skill set and experience expand with age. Also, the employment opportunities and starting salaries would be expected to be greater for those who have attained more education.

However, it is also obvious that there is considerable variation in income levels for a group of people with identical ages and years of education. This variation is represented by ε in the model. So, in this example, the model would be expressed as shown in Equation 6-2.

$$Income = \beta_0 + \beta_1 Age + \beta_2 Education + \varepsilon \tag{6-2}$$

In the linear model, the β_js represent the unknown p parameters. The estimates for these unknown parameters are chosen so that, on average, the model provides a reasonable estimate of a person's income based on age and education. In other words, the fitted model should minimize the overall error between the linear model and the actual observations. Ordinary Least Squares (OLS) is a common technique to estimate the parameters.

To illustrate how OLS works, suppose there is only one input variable, x, for an outcome variable y. Furthermore, n observations of (x, y) are obtained and plotted in Figure 6-1.

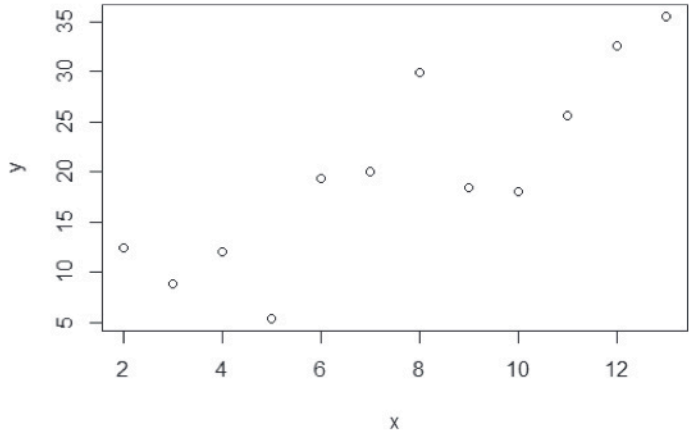


FIGURE 6-1 *Scatterplot of y versus x*

The goal is to find the line that best approximates the relationship between the outcome variable and the input variables. With OLS, the objective is to find the line through these points that minimizes the sum of the squares of the difference between each point and the line in the vertical direction. In other words, find the values of β_0 and β_1 such that the summation shown in Equation 6-3 is minimized.

$$\sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2 \tag{6-3}$$

The n individual distances to be squared and then summed are illustrated in Figure 6-2. The vertical lines represent the distance between each observed y value and the line $y = \beta_0 + \beta_1 x$.

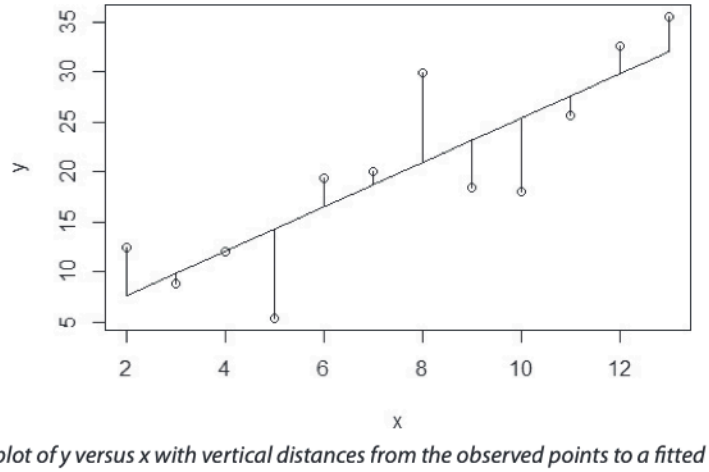


FIGURE 6-2 *Scatterplot of y versus x with vertical distances from the observed points to a fitted line*

In Figure 3-7 of Chapter 3, "Review of Basic Data Analytic Methods Using R," the Anscombe's Quartet example used OLS to fit the linear regression line to each of the four datasets. OLS for multiple input variables is a straightforward extension of the one input variable case provided in Equation 6-3.

The preceding discussion provided the approach to find the best linear fit to a set of observations. However, by making some additional assumptions on the error term, it is possible to provide further capabilities in utilizing the linear regression model. In general, these assumptions are almost always made, so the following model, built upon the earlier described model, is simply called the linear regression model.

Linear Regression Model (with Normally Distributed Errors)

In the previous model description, there were no assumptions made about the error term; no additional assumptions were necessary for OLS to provide estimates of the model parameters. However, in most linear regression analyses, it is common to assume that the error term is a normally distributed random variable with mean equal to zero and constant variance. Thus, the linear regression model is expressed as shown in Equation 6-4.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon \tag{6-4}$$

where:

y is the outcome variable

x_j are the input variables, for j = 1, 2,…, p – 1

β_0 is the value of y when each x_j equals zero

β_j is the change in y based on a unit change in x_j, for j = 1, 2,. . ., p – 1

$\varepsilon \sim N(0, \sigma^2)$ and the εs are independent of each other

This additional assumption yields the following result about the expected value of y, E(y) for given $(x_1, x_2, \ldots x_{p-1})$:

$$\begin{aligned} E(y) &= E(\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + E(\varepsilon) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} \end{aligned}$$

Because β_j and x_j are constants, the E(y) is the value of the linear regression model for the given $(x_1, x_2, \ldots x_{p-1})$. Furthermore, the variance of y, V(y), for given $(x_1, x_2, \ldots x_{p-1})$ is this:

$$\begin{aligned} V(y) &= V(\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} + \varepsilon) \\ &= 0 + V(\varepsilon) = \sigma^2 \end{aligned}$$

Thus, for a given $(x_1, x_2, \ldots x_{p-1})$, y is normally distributed with mean $\beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1}$ and variance σ^2. For a regression model with just one input variable, Figure 6-3 illustrates the normality assumption on the error terms and the effect on the outcome variable, y, for a given value of x.

For $x = 8$, one would expect to observe a value of y near 20, but a value of y from 15 to 25 would appear possible based on the illustrated normal distribution. Thus, the regression model estimates the expected value of y for the given value of x. Additionally, the normality assumption on the error term provides some useful properties that can be utilized in performing hypothesis testing on the linear regression model and

providing confidence intervals on the parameters and the mean of y given $(x_1, x_2, \ldots x_{p-1})$. The application of these statistical techniques is demonstrated by applying R to the earlier linear regression model on income.

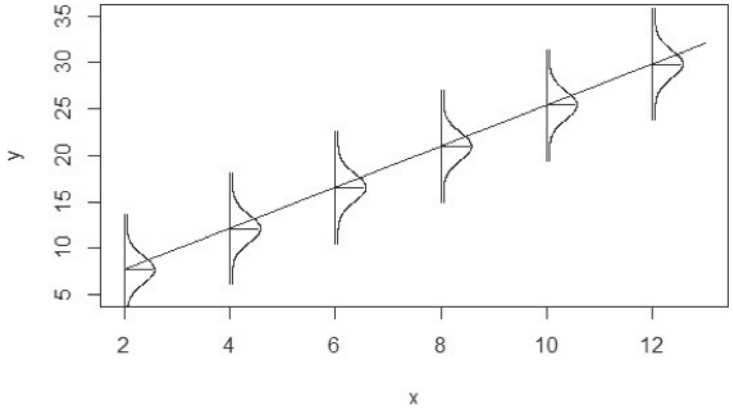


FIGURE 6-3 *Normal distribution about y for a given value of x*

Example in R

Returning to the `Income` example, in addition to the variables age and education, the person's gender, female or male, is considered an input variable. The following code reads a comma-separated-value (CSV) file of 1,500 people's incomes, ages, years of education, and gender. The first 10 rows are displayed:

```
income_input = as.data.frame(  read.csv("c:/data/income.csv")   )
income_input[1:10,]
```

```
   ID Income Age Education Gender
1   1    113  69        12      1
2   2     91  52        18      0
3   3    121  65        14      0
4   4     81  58        12      0
5   5     68  31        16      1
6   6     92  51        15      1
7   7     75  53        15      0
8   8     76  56        13      0
9   9     56  42        15      1
10 10     53  33        11      1
```

Each person in the sample has been assigned an identification number, `ID`. `Income` is expressed in thousands of dollars. (For example, 113 denotes $113,000.) As described earlier, `Age` and `Education` are expressed in years. For `Gender`, a 0 denotes female and a 1 denotes male. A summary of the imported data reveals that the incomes vary from $14,000 to $134,000. The ages are between 18 and 70 years. The education experience for each person varies from a minimum of 10 years to a maximum of 20 years.

```
summary(income_input)
```

```
       ID              Income            Age           Education
 Min.   :   1.0   Min.   : 14.00   Min.   :18.00   Min.   :10.00
```

```
 1st Qu.: 375.8   1st Qu.: 62.00   1st Qu.:30.00   1st Qu.:12.00
 Median : 750.5   Median : 76.00   Median :44.00   Median :15.00
 Mean   : 750.5   Mean   : 75.99   Mean   :43.58   Mean   :14.68
 3rd Qu.:1125.2   3rd Qu.: 91.00   3rd Qu.:57.00   3rd Qu.:16.00
 Max.   :1500.0   Max.   :134.00   Max.   :70.00   Max.   :20.00

     Gender
 Min.   :0.00
 1st Qu.:0.00
 Median :0.00
 Mean   :0.49
 3rd Qu.:1.00
 Max.   :1.00
```

As described in Chapter 3, a scatterplot matrix is an informative tool to view the pair-wise relationships of the variables. The basic assumption of a linear regression model is that there is a linear relationship between the outcome variable and the input variables. Using the `lattice` package in R, the scatterplot matrix in Figure 6-4 is generated with the following R code:

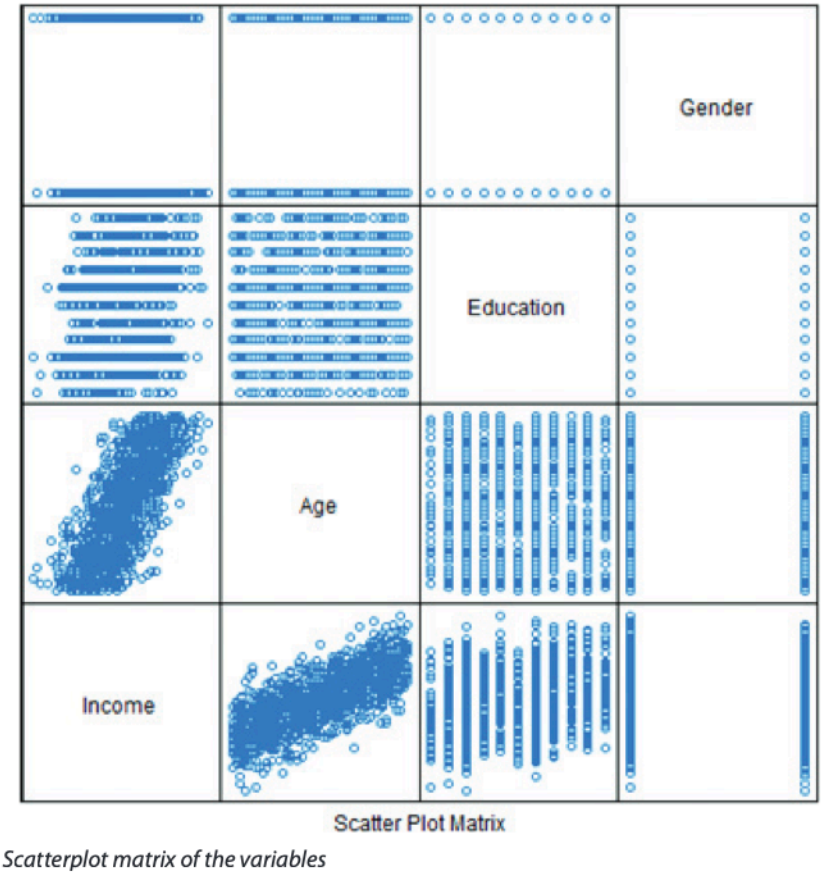


FIGURE 6-4 *Scatterplot matrix of the variables*

```
library(lattice)

splom(~income_input[c(2:5)],  groups=NULL, data=income_input,
      axis.line.tck = 0,
      axis.text.alpha = 0)
```

Because the dependent variable is typically plotted along the y-axis, examine the set of scatterplots along the bottom of the matrix. A strong positive linear trend is observed for `Income` as a function of `Age`. Against `Education`, a slight positive trend may exist, but the trend is not quite as obvious as is the case with the `Age` variable. Lastly, there is no observed effect on `Income` based on `Gender`.

With this qualitative understanding of the relationships between `Income` and the input variables, it seems reasonable to quantitatively evaluate the linear relationships of these variables. Utilizing the normality assumption applied to the error term, the proposed linear regression model is shown in Equation 6-5.

$$Income = \beta_0 + \beta_1 Age + \beta_2 Education + \beta_3 Gender + \varepsilon \tag{6-5}$$

Using the linear model function, `lm()`, in R, the income model can be applied to the data as follows:

```
results <- lm(Income~Age + Education + Gender, income_input)
summary(results)

Call:
lm(formula = Income ~ Age + Education + Gender, data = income_input)

Residuals:
    Min      1Q  Median      3Q     Max
-37.340  -8.101   0.139   7.885  37.271

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.26299    1.95575   3.714 0.000212 ***
Age          0.99520    0.02057  48.373  < 2e-16 ***
Education    1.75788    0.11581  15.179  < 2e-16 ***
Gender      -0.93433    0.62388  -1.498 0.134443
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.07 on 1496 degrees of freedom
Multiple R-squared:  0.6364,  Adjusted R-squared:  0.6357
F-statistic:   873 on 3 and 1496 DF,  p-value: < 2.2e-16
```

The intercept term, β_0, is implicitly included in the model. The `lm()` function performs the parameter estimation for the parameters β_j (j = 0, 1, 2, 3) using ordinary least squares and provides several useful calculations and results that are stored in the variable called `results` in this example.

After the stated call to `lm()`, a few statistics on the residuals are displayed in the output. The residuals are the observed values of the error term for each of the n observations and are defined for i = 1, 2, …n, as shown in Equation 6-6.

$$e_i = y_i - (b_0 + b_1 x_{i,1} + b_2 x_{i,2} \ldots + b_{p-1} x_{i,p-1}) \tag{6-6}$$

where b_j denotes the estimate for parameter β_j for $j = 0, 1, 2, \ldots p - 1$

From the R output, the residuals vary from approximately −37 to +37, with a median close to 0. Recall that the residuals are assumed to be normally distributed with a mean near zero and a constant variance. The normality assumption is examined more carefully later.

The output provides details about the coefficients. The column `Estimate` provides the OLS estimates of the coefficients in the fitted linear regression model. In general, the `(Intercept)` corresponds to the estimated response variable when all the input variables equal zero. In this example, the intercept corresponds to an estimated income of $7,263 for a newborn female with no education. It is important to note that the available dataset does not include such a person. The minimum age and education in the dataset are 18 and 10 years, respectively. Thus, misleading results may be obtained when using a linear regression model to estimate outcomes for input values not representative within the dataset used to train the model.

The coefficient for `Age` is approximately equal to one. This coefficient is interpreted as follows: For every one unit increase in a person's age, the person's income is expected to increase by $995. Similarly, for every unit increase in a person's years of education, the person's income is expected to increase by about $1,758.

Interpreting the `Gender` coefficient is slightly different. When `Gender` is equal to zero, the `Gender` coefficient contributes nothing to the estimate of the expected income. When `Gender` is equal to one, the expected `Income` is decreased by about $934.

Because the coefficient values are only estimates based on the observed incomes in the sample, there is some uncertainty or sampling error for the coefficient estimates. The `Std. Error` column next to the coefficients provides the sampling error associated with each coefficient and can be used to perform a hypothesis test, using the *t*-distribution, to determine if each coefficient is statistically different from zero. In other words, if a coefficient is not statistically different from zero, the coefficient and the associated variable in the model should be excluded from the model. In this example, the associated hypothesis tests' p-values, `Pr(>|t|)`, are very small for the `Intercept`, `Age`, and `Education` parameters. As seen in Chapter 3, a small p-value corresponds to a small probability that such a large *t* value would be observed under the assumptions of the null hypothesis. In this case, for a given j = 0, 1, 2, . . ., p − 1, the null and alternate hypotheses follow:

$$H_0 : \beta_j = 0 \quad \text{versus} \quad H_A : \beta_j \neq 0$$

For small p-values, as is the case for the `Intercept`, `Age`, and `Education` parameters, the null hypothesis would be rejected. For the `Gender` parameter, the corresponding p-value is fairly large at 0.13. In other words, at a 90% confidence level, the null hypothesis would not be rejected. So, dropping the variable `Gender` from the linear regression model should be considered. The following R code provides the modified model results:

```
results2 <- lm(Income ~ Age + Education, income_input)
summary(results2)

Call:
lm(formula = Income ~ Age + Education, data = income_input)

Residuals:
    Min      1Q  Median      3Q     Max
-36.889  -7.892   0.185   8.200  37.740

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  6.75822    1.92728   3.507 0.000467 ***
Age          0.99603    0.02057  48.412  < 2e-16 ***
Education    1.75860    0.11586  15.179  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.08 on 1497 degrees of freedom
Multiple R-squared:  0.6359,  Adjusted R-squared:  0.6354
F-statistic:  1307 on 2 and 1497 DF,  p-value: < 2.2e-16
```

Dropping the `Gender` variable from the model resulted in a minimal change to the estimates of the remaining parameters and their statistical significances.

The last part of the displayed results provides some summary statistics and tests on the linear regression model. The ***residual standard error*** is the standard deviation of the observed residuals. This value, along with the associated degrees of freedom, can be used to examine the variance of the assumed normally distributed error terms. R-squared (R^2) is a commonly reported metric that measures the variation in the data that is explained by the regression model. Possible values of R^2 vary from 0 to 1, with values closer to 1 indicating that the model is better at explaining the data than values closer to 0. An R^2 of exactly 1 indicates that the model explains perfectly the observed data (all the residuals are equal to 0). In general, the R^2 value can be increased by adding more variables to the model. However, just adding more variables to explain a given dataset but not to improve the explanatory nature of the model is known as ***overfitting***. To address the possibility of overfitting the data, the adjusted R^2 accounts for the number of parameters included in the linear regression model.

The F-statistic provides a method for testing the entire regression model. In the previous *t*-tests, individual tests were conducted to determine the statistical significance of each parameter. The provided F-statistic and corresponding p-value enable the analyst to test the following hypotheses:

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_{p-1} = 0 \quad \text{versus} \quad H_A : \beta_j \neq 0$$
$$\text{for at least one } j = 1, 2, \ldots, p-1$$

In this example, the p-value of 2.2e – 16 is small, which indicates that the null hypothesis should be rejected.

Categorical Variables

In the previous example, the variable `Gender` was a simple binary variable that indicated whether a person is female or male. In general, these variables are known as ***categorical variables***. To illustrate how to use categorical variables properly, suppose it was decided in the earlier `Income` example to include an additional variable, `State`, to represent the U.S. state where the person resides. Similar to the use of the `Gender` variable, one possible, but incorrect, approach would be to include a `State` variable that would take a value of 0 for Alabama, 1 for Alaska, 2 for Arizona, and so on. The problem with this approach is that such a numeric assignment based on an alphabetical ordering of the states does not provide a meaningful measure of the difference in the states. For example, is it useful or proper to consider Arizona to be one unit greater than Alaska and two units greater that Alabama?

In regression, a proper way to implement a categorical variable that can take on *m* different values is to add *m-1* binary variables to the regression model. To illustrate with the *Income* example, a binary variable for each of 49 states, excluding Wyoming (arbitrarily chosen as the last of 50 states in an alphabetically sorted list), could be added to the model.

```
results3 <- lm(Income~Age + Education,
               + Alabama,
               + Alaska,
               + Arizona,
               .
               .
               .
               + WestVirginia,
               + Wisconsin,
               income_input)
```

The input file would have 49 columns added for these variables representing each of the first 49 states. If a person was from Alabama, the Alabama variable would be equal to 1, and the other 48 variables would be set to 0. This process would be applied for the other state variables. So, a person from Wyoming, the one state not explicitly stated in the model, would be identified by setting all 49 state variables equal to 0. In this representation, Wyoming would be considered the reference case, and the regression coefficients of the other state variables would represent the difference in income between Wyoming and a particular state.

Confidence Intervals on the Parameters

Once an acceptable linear regression model is developed, it is often helpful to use it to draw some inferences about the model and the population from which the observations were drawn. Earlier, we saw that *t*-tests could be used to perform hypothesis tests on the individual model parameters, β_j, j = 0, 1, …, p − 1. Alternatively, these *t*-tests could be expressed in terms of confidence intervals on the parameters. R simplifies the computation of confidence intervals on the parameters with the use of the `confint()` function. From the *Income* example, the following R command provides 95% confidence intervals on the intercept and the coefficients for the two variables, *Age* and *Education*.

```
confint(results2, level = .95)

                2.5 %    97.5 %
(Intercept) 2.9777598 10.538690
Age         0.9556771  1.036392
Education   1.5313393  1.985862
```

Based on the data, the earlier estimated value of the *Education* coefficient was 1.76. Using `confint()`, the corresponding 95% confidence interval is (1.53, 1.99), which provides the amount of uncertainty in the estimate. In other words, in repeated random sampling, the computed confidence interval straddles the true but unknown coefficient 95% of the time. As expected from the earlier *t*-test results, none of these confidence intervals straddles zero.

Confidence Interval on the Expected Outcome

In addition to obtaining confidence intervals on the model parameters, it is often desirable to obtain a confidence interval on the expected outcome. In the *Income* example, the fitted linear regression provides the expected income for a given *Age* and *Education*. However, that particular point estimate does not provide information on the amount of uncertainty in that estimate. Using the `predict()` function in R, a confidence interval on the expected outcome can be obtained for a given set of input variable values.

In this illustration, a data frame is built containing a specific age and education value. Using this set of input variable values, the `predict()` function provides a 95% confidence interval on the expected *Income* for a 41-year-old person with 12 years of education.

```
Age <- 41
Education <- 12
new_pt <- data.frame(Age, Education)

conf_int_pt <- predict(results2,new_pt,level=.95,interval="confidence")
conf_int_pt

       fit      lwr      upr
1 68.69884 67.83102 69.56667
```

For this set of input values, the expected income is $68,699 with a 95% confidence interval of ($67,831, $69,567).

Prediction Interval on a Particular Outcome

The previous confidence interval was relatively close (+/– approximately $900) to the fitted value. However, this confidence interval should not be considered as representing the uncertainty in estimating a particular person's income. The `predict()` function in R also provides the ability to calculate upper and lower bounds on a particular outcome. Such bounds provide what are referred to as ***prediction intervals***. Returning to the *Income* example, in R the 95% prediction interval on the *Income* for a 41-year-old person with 12 years of education is obtained as follows:

```
pred_int_pt <- predict(results2,new_pt,level=.95,interval="prediction")
pred_int_pt

       fit      lwr      upr
1 68.69884 44.98867 92.40902
```

Again, the expected income is $68,699. However, the 95% prediction interval is ($44,988, $92,409). If the reason for this much wider interval is not obvious, recall that in Figure 6-3, for a particular input variable value, the expected outcome falls on the regression line, but the individual observations are normally distributed about the expected outcome. The confidence interval applies to the expected outcome that falls on the regression line, but the prediction interval applies to an outcome that may appear anywhere within the normal distribution.

Thus, in linear regression, confidence intervals are used to draw inferences on the population's expected outcome, and prediction intervals are used to draw inferences on the next possible outcome.

6.1.3 Diagnostics

The use of hypothesis tests, confidence intervals, and prediction intervals is dependent on the model assumptions being true. The following discussion provides some tools and techniques that can be used to validate a fitted linear regression model.

Evaluating the Linearity Assumption

A major assumption in linear regression modeling is that the relationship between the input variables and the outcome variable is linear. The most fundamental way to evaluate such a relationship is to plot the outcome variable against each input variable. In the `Income` example, such scatterplots were generated in Figure 6-4. If the relationship between `Age` and `Income` is represented as illustrated in Figure 6-5, a linear model would not apply. In such a case, it is often useful to do any of the following:

- Transform the outcome variable.
- Transform the input variables.
- Add extra input variables or terms to the regression model.

Common transformations include taking square roots or the logarithm of the variables. Another option is to create a new input variable such as the age squared and add it to the linear regression model to fit a quadratic relationship between an input variable and the outcome.

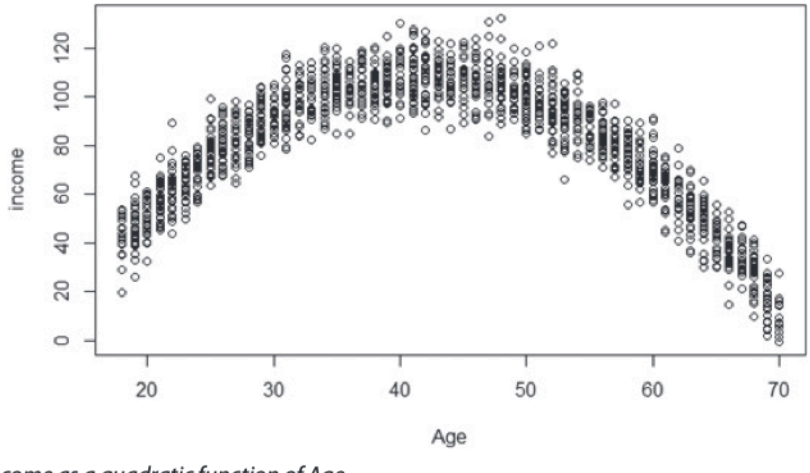


FIGURE 6-5 *Income as a quadratic function of Age*

Additional use of transformations will be considered when evaluating the residuals.

Evaluating the Residuals

As stated previously, it is assumed that the error terms in the linear regression model are normally distributed with a mean of zero and a constant variance. If this assumption does not hold, the various inferences that were made with the hypothesis tests, confidence intervals, and prediction intervals are suspect.

To check for constant variance across all y values along the regression line, use a simple plot of the residuals against the fitted outcome values. Recall that the residuals are the difference between the observed

outcome variables and the fitted value based on the OLS parameter estimates. Because of the importance of examining the residuals, the `lm()` function in R automatically calculates and stores the fitted values and the residuals, in the components *`fitted.values`* and *`residuals`* in the output of the `lm()` function. Using the Income regression model output stored in *`results2`*, Figure 6-6 was generated with the following R code:

```
with(results2, {
plot(fitted.values, residuals,ylim=c(-40,40) )
         points(c(min(fitted.values),max(fitted.values)),
                c(0,0), type = "l")})
```

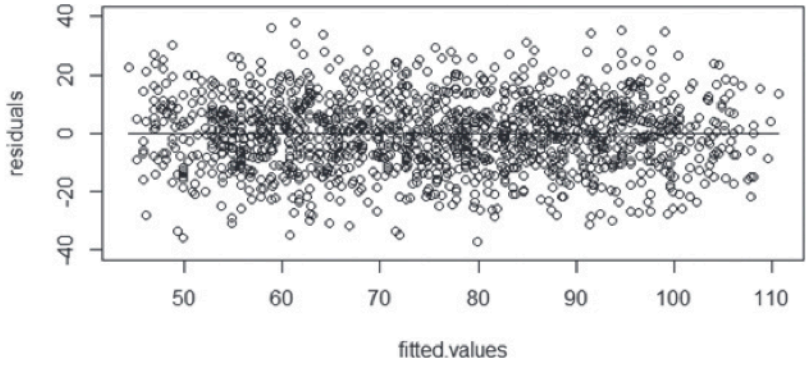


FIGURE 6-6 *Residual plot indicating constant variance*

The plot in Figure 6-6 indicates that regardless of income value along the fitted linear regression model, the residuals are observed somewhat evenly on both sides of the reference zero line, and the spread of the residuals is fairly constant from one fitted value to the next. Such a plot would support the mean of zero and the constant variance assumptions on the error terms.

If the residual plot appeared like any of those in Figures 6-7 through 6-10, then some of the earlier discussed transformations or possible input variable additions should be considered and attempted. Figure 6-7 illustrates the existence of a nonlinear trend in the residuals. Figure 6-8 illustrates that the residuals are not centered on zero. Figure 6-9 indicates a linear trend in the residuals across the various outcomes along the linear regression model. This plot may indicate a missing variable or term from the regression model. Figure 6-10 provides an example in which the variance of the error terms is not a constant but increases along the fitted linear regression model.

Evaluating the Normality Assumption

The residual plots are useful for confirming that the residuals were centered on zero and have a constant variance. However, the normality assumption still has to be validated. As shown in Figure 6-11, the following R code provides a histogram plot of the residuals from `results2`, the output from the *`Income`* example:

```
hist(results2$residuals, main="")
```

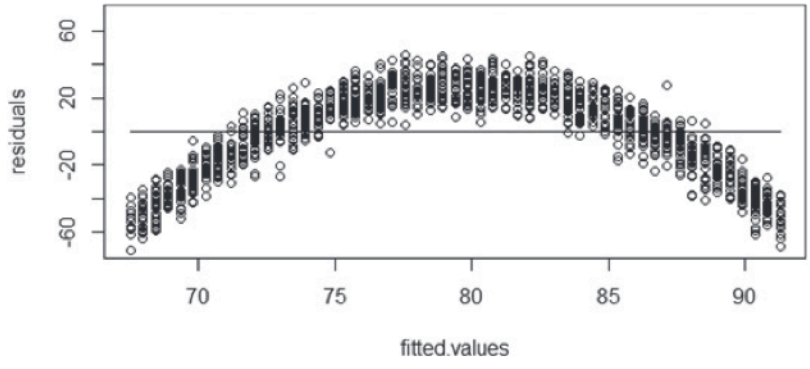


FIGURE 6-7 *Residuals with a nonlinear trend*

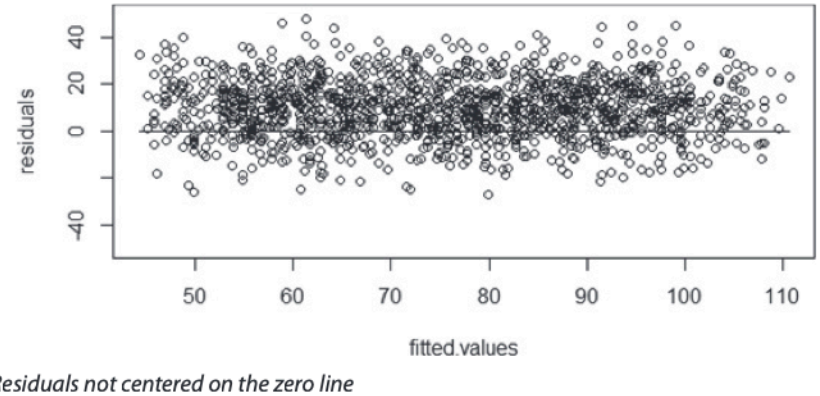


FIGURE 6-8 *Residuals not centered on the zero line*

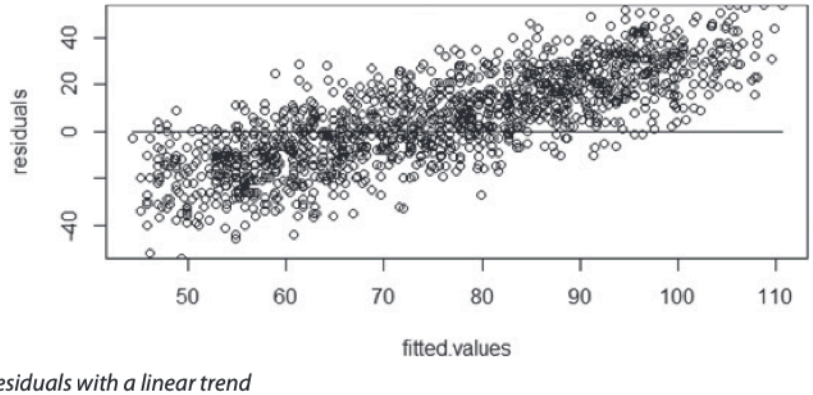


FIGURE 6-9 *Residuals with a linear trend*

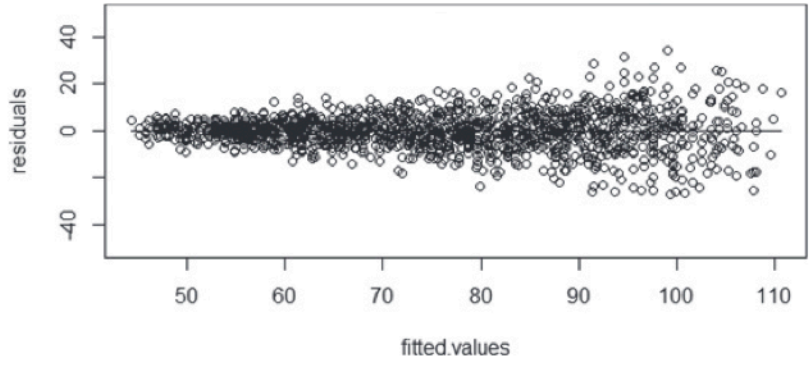


FIGURE 6-10 *Residuals with nonconstant variance*

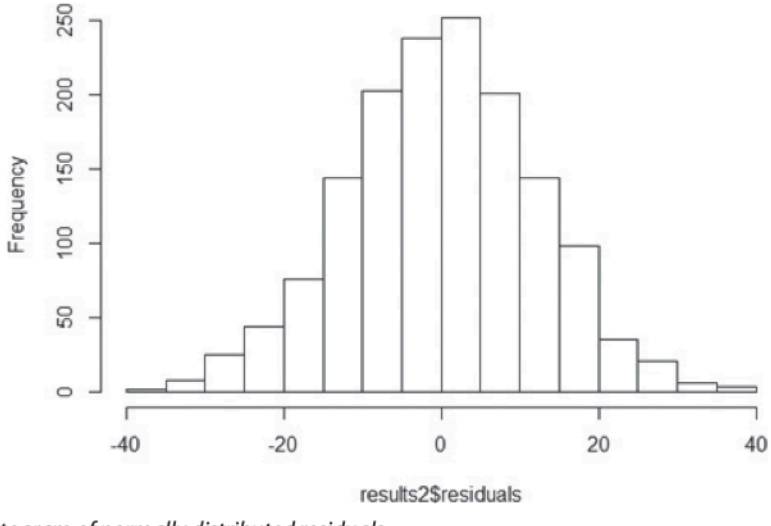


FIGURE 6-11 *Histogram of normally distributed residuals*

From the histogram, it is seen that the residuals are centered on zero and appear to be symmetric about zero, as one would expect for a normally distributed random variable. Another option is to examine a Q-Q plot that compares the observed data against the quantiles (Q) of the assumed distribution. In R, the following code generates the Q-Q plot shown in Figure 6-12 for the residuals from the *Income* example and provides the line that the points should follow for values from a normal distribution.

```
qqnorm(results2$residuals, ylab="Residuals", main="")
qqline(results2$residuals)
```

A Q-Q plot as provided in Figure 6-13 would indicate that additional refinement of the model is required to achieve normally distributed error terms.

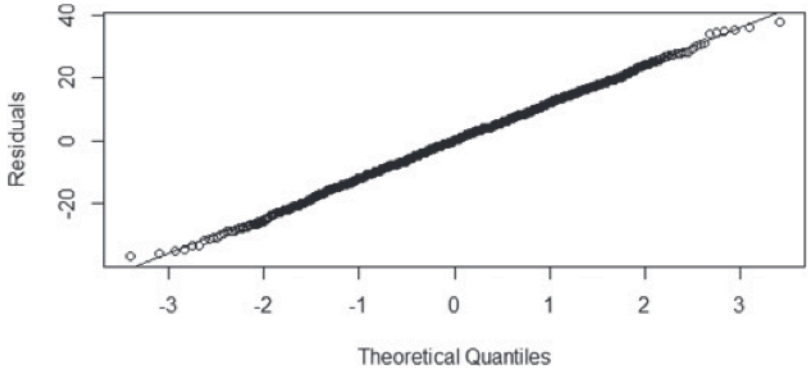


FIGURE 6-12 *Q-Q plot of normally distributed residuals*

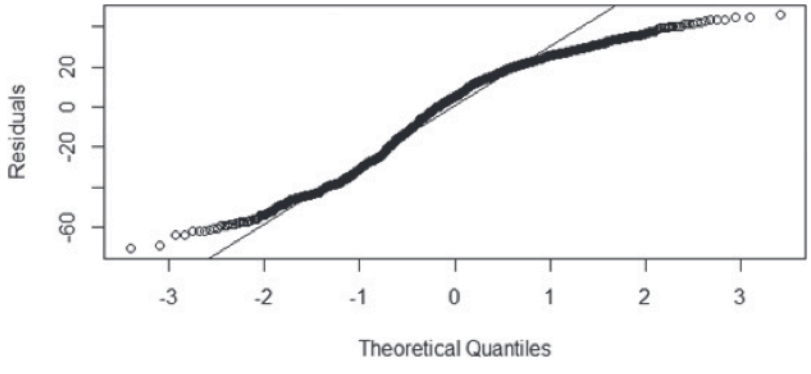


FIGURE 6-13 *Q-Q plot of non-normally distributed residuals*

N-Fold Cross-Validation

To prevent overfitting a given dataset, a common practice is to randomly split the entire dataset into a training set and a testing set. Once the model is developed on the training set, the model is evaluated against the testing set. When there is not enough data to create training and testing sets, an N-fold cross-validation technique may be helpful to compare one fitted model against another. In N-fold cross-validation, the following occurs:

- The entire dataset is randomly split into N datasets of approximately equal size.
- A model is trained against N – 1 of these datasets and tested against the remaining dataset. A measure of the model error is obtained.
- This process is repeated a total of N times across the various combinations of N datasets taken N – 1 at a time. Recall:

$$\binom{N}{N-1} = N$$

- The observed N model errors are averaged over the N folds.

The averaged error from one model is compared against the averaged error from another model. This technique can also help determine whether adding more variables to an existing model is beneficial or possibly overfitting the data.

Other Diagnostic Considerations

Although a fitted linear regression model conforms with the preceding diagnostic criteria, it is possible to improve the model by including additional input variables not yet considered. In the previous `Income` example, only three possible input variables—`Age`, `Education`, and `Gender`—were considered. Dozens of other additional input variables such as `Housing` or `Marital_Status` may improve the fitted model. It is important to consider all possible input variables early in the analytic process.

As mentioned earlier, in reviewing the R output from fitting a linear regression model, the adjusted R^2 applies a penalty to the R^2 value based on the number of parameters added to the model. Because the R^2 value will always move closer to one as more variables are added to an existing regression model, the adjusted R^2 value may actually decrease after adding more variables.

The residual plots should be examined for any ***outliers***, observed points that are markedly different from the majority of the points. Outliers can result from bad data collection, data processing errors, or an actual rare occurrence. In the `Income` example, suppose that an individual with an income of a million dollars was included in the dataset. Such an observation could affect the fitted regression model, as seen in one of the examples of Anscombe's Quartet.

Finally, the magnitudes and signs of the estimated parameters should be examined to see if they make sense. For example, suppose a negative coefficient for the Education variable in the `Income` example was obtained. Because it is natural to assume that more years of education lead to higher incomes, either something very unexpected has been discovered, or there is some issue with the model, how the data was collected, or some other factor. In either case, further investigation is warranted.

6.2 Logistic Regression

In linear regression modeling, the outcome variable is a continuous variable. As seen in the earlier `Income` example, linear regression can be used to model the relationship between age and education to income. Suppose a person's actual income was not of interest, but rather whether someone was wealthy or poor. In such a case, when the outcome variable is categorical in nature, logistic regression can be used to predict the likelihood of an outcome based on the input variables. Although logistic regression can be applied to an outcome variable that represents multiple values, the following discussion examines the case in which the outcome variable represents two values such as true/false, pass/fail, or yes/no.

For example, a logistic regression model can be built to determine if a person will or will not purchase a new automobile in the next 12 months. The training set could include input variables for a person's age, income, and gender as well as the age of an existing automobile. The training set would also include the outcome variable on whether the person purchased a new automobile over a 12-month period. The logistic regression model provides the likelihood or probability of a person making a purchase in the next 12 months. After examining a few more use cases for logistic regression, the remaining portion of this chapter examines how to build and evaluate a logistic regression model.

6.2.1 Use Cases

The logistic regression model is applied to a variety of situations in both the public and the private sector. Some common ways that the logistic regression model is used include the following:

- **Medical:** Develop a model to determine the likelihood of a patient's successful response to a specific medical treatment or procedure. Input variables could include age, weight, blood pressure, and cholesterol levels.
- **Finance:** Using a loan applicant's credit history and the details on the loan, determine the probability that an applicant will default on the loan. Based on the prediction, the loan can be approved or denied, or the terms can be modified.
- **Marketing:** Determine a wireless customer's probability of switching carriers (known as churning) based on age, number of family members on the plan, months remaining on the existing contract, and social network contacts. With such insight, target the high-probability customers with appropriate offers to prevent churn.
- **Engineering:** Based on operating conditions and various diagnostic measurements, determine the probability of a mechanical part experiencing a malfunction or failure. With this probability estimate, schedule the appropriate preventive maintenance activity.

6.2.2 Model Description

Logistic regression is based on the logistic function $f(y)$, as given in Equation 6-7.

$$f(y) = \frac{e^y}{1+e^y} \quad \text{for } -\infty < y < \infty \tag{6-7}$$

Note that as $y \to \infty$, $f(y) \to 1$, and as $y \to -\infty$, $f(y) \to 0$. So, as Figure 6-14 illustrates, the value of the logistic function $f(y)$ varies from 0 to 1 as y increases.

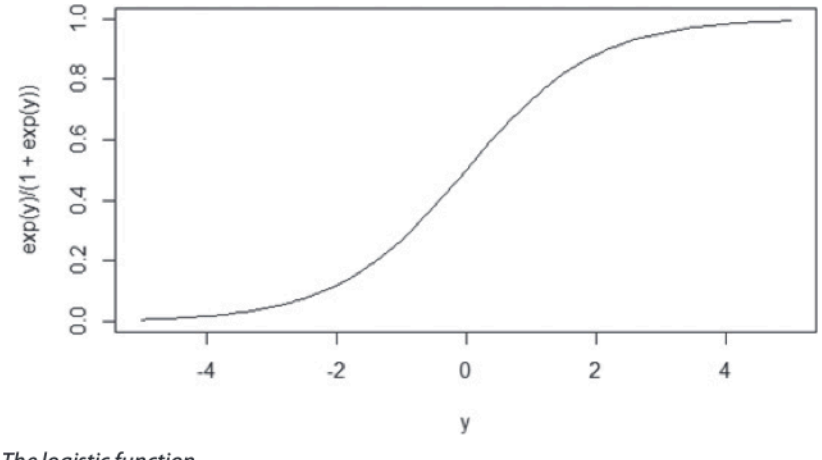


FIGURE 6-14 *The logistic function*

Because the range of $f(y)$ is (0, 1), the logistic function appears to be an appropriate function to model the probability of a particular outcome occurring. As the value of y increases, the probability of the outcome occurring increases. In any proposed model, to predict the likelihood of an outcome, y needs to be a function of the input variables. In logistic regression, y is expressed as a linear function of the input variables. In other words, the formula shown in Equation 6-8 applies.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_{p-1} x_{p-1} \tag{6-8}$$

Then, based on the input variables $x_1, x_2, \ldots, x_{p-1}$, the probability of an event is shown in Equation 6-9.

$$p(x_1, x_2, \ldots, x_{p-1}) = f(y) = \frac{e^y}{1+e^y} \quad \text{for} -\infty < y < \infty \tag{6-9}$$

Equation 6-8 is comparable to Equation 6-1 used in linear regression modeling. However, one difference is that the values of y are not directly observed. Only the value of $f(y)$ in terms of success or failure (typically expressed as 1 or 0, respectively) is observed.

Using p to denote $f(y)$, Equation 6-9 can be rewritten in the form provided in Equation 6-10.

$$ln\left(\frac{p}{1-p}\right) = y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_p x_{p-1} \tag{6-10}$$

The quantity $ln\left(\frac{p}{1-p}\right)$, in Equation 6-10 is known as the log odds ratio, or the logit of p. Techniques such as Maximum Likelihood Estimation (MLE) are used to estimate the model parameters. MLE determines the values of the model parameters that maximize the chances of observing the given dataset. However, the specifics of implementing MLE are beyond the scope of this book.

The following example helps to clarify the logistic regression model. The mechanics of using R to fit a logistic regression model are covered in the next section on evaluating the fitted model. In this section, the discussion focuses on interpreting the fitted model.

Customer Churn Example

A wireless telecommunications company wants to estimate the probability that a customer will churn (switch to a different company) in the next six months. With a reasonably accurate prediction of a person's likelihood of churning, the sales and marketing groups can attempt to retain the customer by offering various incentives. Data on 8,000 current and prior customers was obtained. The variables collected for each customer follow:

- `Age` (years)
- `Married` (true/false)
- `Duration` as a customer (years)
- `Churned_contacts` (count)—Number of the customer's contacts that have churned (count)
- `Churned` (true/false)—Whether the customer churned

After analyzing the data and fitting a logistic regression model, `Age` and `Churned_contacts` were selected as the best predictor variables. Equation 6-11 provides the estimated model parameters.

$$y = 3.50 - 0.16 * Age + 0.38 * Churned_contacts \tag{6-11}$$

Using the fitted model from Equation 6-11, Table 6-1 provides the probability of a customer churning based on the customer's age and the number of churned contacts. The computed values of y are also provided in the table. Recalling the previous discussion of the logistic function, as the value of y increases, so does the probability of churning.

TABLE 6-1 *Estimated Churn Probabilities*

Customer	Age (Years)	Churned_Contacts	y	Prob. of Churning
1	50	1	–4.12	0.016
2	50	3	–3.36	0.034
3	50	6	–2.22	0.098
4	30	1	–0.92	0.285
5	30	3	–0.16	0.460
6	30	6	0.98	0.727
7	20	1	0.68	0.664
8	20	3	1.44	0.808
9	20	6	2.58	0.930

Based on the fitted model, there is a 93% chance that a 20-year-old customer who has had six contacts churn will also churn. (See the last row of Table 6-1.) Examining the sign and values of the estimated coefficients in Equation 6-11, it is observed that as the value of *Age* increases, the value of y decreases. Thus, the negative *Age* coefficient indicates that the probability of churning decreases for an older customer. On the other hand, based on the positive sign of the *Churned_Contacts* coefficient, the value of y and subsequently the probability of churning increases as the number of churned contacts increases.

6.2.3 Diagnostics

The churn example illustrates how to interpret a fitted logistic regression model. Using R, this section examines the steps to develop a logistic regression model and evaluate the model's effectiveness. For this example, the `churn_input` data frame is structured as follows:

```
head(churn_input)

  ID Churned Age Married Cust_years Churned_contacts
1  1       0  61       1          3                1
2  2       0  50       1          3                2
3  3       0  47       1          2                0
4  4       0  50       1          3                3
5  5       0  29       1          1                3
6  6       0  43       1          4                3
```

A *Churned* value of 1 indicates that the customer churned. A *Churned* value of 0 indicates that the customer remained as a subscriber. Out of the 8,000 customer records in this dataset, 1,743 customers (~22%) churned.

```
sum(churn_input$Churned)
```

```
[1] 1743
```

Using the Generalized Linear Model function, `glm()`, in R and the specified family/link, a logistic regression model can be applied to the variables in the dataset and examined as follows:

```
Churn_logistic1 <- glm (Churned~Age + Married + Cust_years +
                        Churned_contacts, data=churn_input,
                        family=binomial(link="logit"))
summary(Churn_logistic1)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        3.415201   0.163734  20.858   <2e-16 ***
Age               -0.156643   0.004088 -38.320   <2e-16 ***
Married            0.066432   0.068302   0.973    0.331
Cust_years         0.017857   0.030497   0.586    0.558
Churned_contacts   0.382324   0.027313  13.998   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As in the linear regression case, there are tests to determine if the coefficients are significantly different from zero. Such significant coefficients correspond to small values of `Pr(>|z|)`, which denote the p-value for the hypothesis test to determine if the estimated model parameter is significantly different from zero. Rerunning this analysis without the *Cust_years* variable, which had the largest corresponding p-value, yields the following:

```
Churn_logistic2 <- glm (Churned~Age + Married +  Churned_contacts,
                        data=churn_input, family=binomial(link="logit"))
summary(Churn_logistic2)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        3.472062   0.132107  26.282   <2e-16 ***
Age               -0.156635   0.004088 -38.318   <2e-16 ***
Married            0.066430   0.068299   0.973    0.331
Churned_contacts   0.381909   0.027302  13.988   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because the p-value for the *Married* coefficient remains quite large, the *Married* variable is dropped from the model. The following R code provides the third and final model, which includes only the *Age* and *Churned_contacts* variables:

```
Churn_logistic3 <- glm (Churned~Age + Churned_contacts,
```

```
                    data=churn_input, family=binomial(link="logit"))
summary(Churn_logistic3)

Call:
glm(formula = Churned ~ Age + Churned_contacts,
               family = binomial(link = "logit"), data = churn_input)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4599  -0.5214  -0.1960  -0.0736   3.3671

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       3.502716   0.128430   27.27   <2e-16 ***
Age              -0.156551   0.004085  -38.32   <2e-16 ***
Churned_contacts  0.381857   0.027297   13.99   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8387.3  on 7999  degrees of freedom
Residual deviance: 5359.2  on 7997  degrees of freedom
AIC: 5365.2

Number of Fisher Scoring iterations: 6
```

For this final model, the entire summary output is provided. The output offers several values that can be used to evaluate the fitted model. It should be noted that the model parameter estimates correspond to the values provided in Equation 6-11 that were used to construct Table 6-1.

Deviance and the Pseudo-R^2

In logistic regression, deviance is defined to be $-2 * logL$, where L is the maximized value of the likelihood function that was used to obtain the parameter estimates. In the R output, two deviance values are provided. The ***null deviance*** is the value where the likelihood function is based only on the intercept term ($y = \beta_0$). The ***residual deviance*** is the value where the likelihood function is based on the parameters in the specified logistic model, shown in Equation 6-12.

$$y = \beta_0 + \beta_1 * Age + \beta_2 * Churned_contacts \tag{6-12}$$

A metric analogous to R^2 in linear regression can be computed as shown in Equation 6-13.

$$\text{pseudo-}R^2 = 1 - \frac{residual\ dev.}{null\ dev.} = \frac{null\ dev. - res.\ dev.}{null\ dev.} \tag{6-13}$$

The pseudo-R^2 is a measure of how well the fitted model explains the data as compared to the default model of no predictor variables and only an intercept term. A $pseudo - R^2$ value near 1 indicates a good fit over the simple null model.

Deviance and the Log-Likelihood Ratio Test

In the *pseudo* $-R^2$ calculation, the –2 multipliers simply divide out. So, it may appear that including such a multiplier does not provide a benefit. However, the multiplier in the deviance definition is based on the log-likelihood test statistic shown in Equation 6-14:

$$\begin{aligned} T &= -2 * log\left(\frac{L_{null}}{L_{alt.}}\right) \\ &= -2 * \log(L_{null}) - (-2) * \log(L_{alt.}) \end{aligned} \tag{6-14}$$

where T is approximately Chi-squared distributed (χ_k^2) with

k *degrees of freedom* $(df) = df_{null} - df_{alternate}$

The previous description of the log-likelihood test statistic applies to any estimation using MLE. As can be seen in Equation 6-15, in the logistic regression case,

$$T = \text{null deviance} - \text{residual deviance} \sim \chi^2_{p-1} \tag{6-15}$$

where p is the number of parameters in the fitted model.

So, in a hypothesis test, a large value of T would indicate that the fitted model is significantly better than the null model that uses only the intercept term.

In the churn example, the log-likelihood ratio statistic would be this:

$T = 8387.3 - 5359.2 = 3028.1$ with 2 degrees of freedom and a corresponding p-value that is essentially zero.

So far, the log-likelihood ratio test discussion has focused on comparing a fitted model to the default model of using only the intercept. However, the log-likelihood ratio test can also compare one fitted model to another. For example, consider the logistic regression model when the categorical variable `Married` is included with `Age` and `Churned_contacts` in the list of input variables. The partial R output for such a model is provided here:

```
summary(Churn_logistic2)
Call:
glm(formula = Churned ~ Age + Married + Churned_contacts,
    family = binomial(link = "logit"),
    data = churn_input)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       3.472062   0.132107  26.282   <2e-16 ***
Age              -0.156635   0.004088 -38.318   <2e-16 ***
Married           0.066430   0.068299   0.973    0.331
Churned_contacts  0.381909   0.027302  13.988   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8387.3  on 7999  degrees of freedom
Residual deviance: 5358.3  on 7996  degrees of freedom
```

```
predObj = prediction(pred, churn_input$Churned )

rocObj = performance(predObj, measure="tpr", x.measure="fpr")
aucObj = performance(predObj, measure="auc")

plot(rocObj, main = paste("Area under the curve:",
                          round(aucObj@y.values[[1]] ,4)))
```

The usefulness of this plot in Figure 6-15 is that the preferred outcome of a classifier is to have a low FPR and a high TPR. So, when moving from left to right on the FPR axis, a good model/classifier has the TPR rapidly approach values near 1, with only a small change in FPR. The closer the ROC curve tracks along the vertical axis and approaches the upper-left hand of the plot, near the point (0,1), the better the model/classifier performs. Thus, a useful metric is to compute the area under the ROC curve (AUC). By examining the axes, it can be seen that the theoretical maximum for the area is 1.

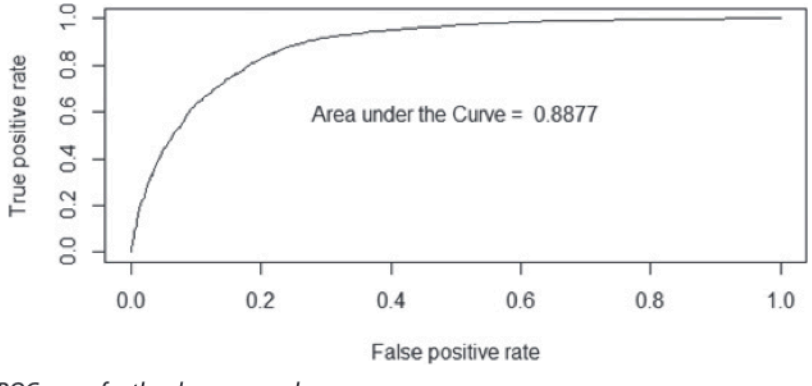


FIGURE 6-15 *ROC curve for the churn example*

To illustrate how the FPR and TPR values are dependent on the threshold value used for the classifier, the plot in Figure 6-16 was constructed using the following R code:

```
# extract the alpha(threshold), FPR, and TPR values from rocObj
alpha <- round(as.numeric(unlist(rocObj@alpha.values)),4)
fpr <- round(as.numeric(unlist(rocObj@x.values)),4)
tpr <- round(as.numeric(unlist(rocObj@y.values)),4)

# adjust margins and plot TPR and FPR
par(mar = c(5,5,2,5))
plot(alpha,tpr, xlab="Threshold", xlim=c(0,1),
                ylab="True positive rate", type="l")
par(new="True")
plot(alpha,fpr, xlab="", ylab="", axes=F, xlim=c(0,1), type="l" )
axis(side=4)
mtext(side=4, line=3, "False positive rate")
text(0.18,0.18,"FPR")
text(0.58,0.58,"TPR")
```

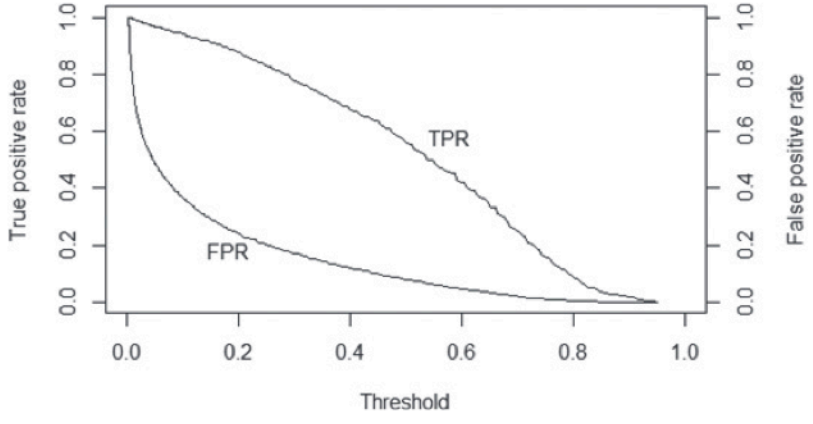


FIGURE 6-16 *The effect of the threshold value in the churn example*

For a threshold value of 0, every item is classified as a positive outcome. Thus, the TPR value is 1. However, all the negatives are also classified as a positive, and the FPR value is also 1. As the threshold value increases, more and more negative class labels are assigned. Thus, the FPR and TPR values decrease. When the threshold reaches 1, no positive labels are assigned, and the FPR and TPR values are both 0.

For the purposes of a classifier, a commonly used threshold value is 0.5. A positive label is assigned for any probability of 0.5 or greater. Otherwise, a negative label is assigned. As the following R code illustrates, in the analysis of the Churn dataset, the 0.5 threshold corresponds to a TPR value of 0.56 and a FPR value of 0.08.

```
i <- which(round(alpha,2) == .5)
paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])

[1] "Threshold= 0.5004  TPR= 0.5571  FPR= 0.0793"
```

Thus, 56% of customers who will churn are properly classified with the `Churn` label, and 8% of the customers who will remain as customers are improperly labeled as `Churn`. If identifying only 56% of the churners is not acceptable, then the threshold could be lowered. For example, suppose it was decided to classify with a `Churn` label any customer with a probability of churning greater than 0.15. Then the following R code indicates that the corresponding TPR and FPR values are 0.91 and 0.29, respectively. Thus, 91% of the customers who will churn are properly identified, but at a cost of misclassifying 29% of the customers who will remain.

```
i <- which(round(alpha,2) == .15)
paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])

[1] "Threshold= 0.1543  TPR= 0.9116  FPR= 0.2869"
[2] "Threshold= 0.1518  TPR= 0.9122  FPR= 0.2875"
[3] "Threshold= 0.1479  TPR= 0.9145  FPR= 0.2942"
[4] "Threshold= 0.1455  TPR= 0.9174  FPR= 0.2981"
```

The ROC curve is useful for evaluating other classifiers and will be utilized again in Chapter 7, "Advanced Analytical Theory and Methods: Classification."

Histogram of the Probabilities

It can be useful to visualize the observed responses against the estimated probabilities provided by the logistic regression. Figure 6-17 provides overlaying histograms for the customers who churned and for the customers who remained as customers. With a proper fitting logistic model, the customers who remained tend to have a low probability of churning. Conversely, the customers who churned have a high probability of churning again. This histogram plot helps visualize the number of items to be properly classified or misclassified. In the Churn example, an ideal histogram plot would have the remaining customers grouped at the left side of the plot, the customers who churned at the right side of the plot, and no overlap of these two groups.

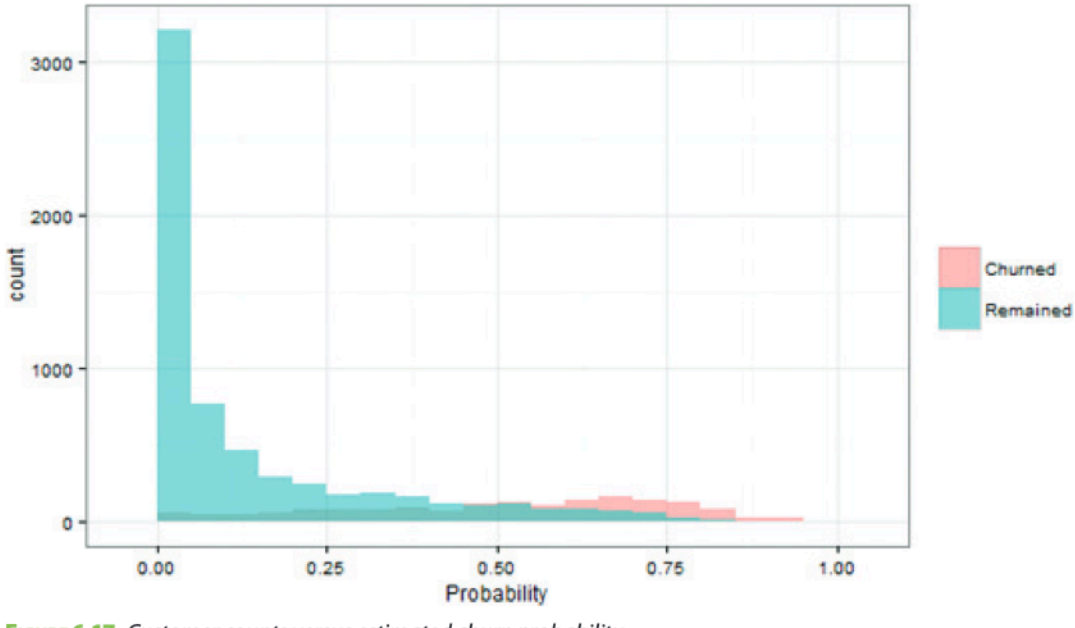


FIGURE 6-17 *Customer counts versus estimated churn probability*

6.3 Reasons to Choose and Cautions

Linear regression is suitable when the input variables are continuous or discrete, including categorical data types, but the outcome variable is continuous. If the outcome variable is categorical, logistic regression is a better choice.

Both models assume a linear additive function of the input variables. If such an assumption does not hold true, both regression techniques perform poorly. Furthermore, in linear regression, the assumption of normally distributed error terms with a constant variance is important for many of the statistical inferences that can be considered. If the various assumptions do not appear to hold, the appropriate transformations need to be applied to the data.

Although a collection of input variables may be a good predictor for the outcome variable, the analyst should not infer that the input variables directly cause an outcome. For example, it may be identified that those individuals who have regular dentist visits may have a reduced risk of heart attacks. However, simply sending someone to the dentist almost certainly has no effect on that person's chance of having a heart attack. It is possible that regular dentist visits may indicate a person's overall health and dietary choices, which may have a more direct impact on a person's health. This example illustrates the commonly known expression, "Correlation does not imply causation."

Use caution when applying an already fitted model to data that falls outside the dataset used to train the model. The linear relationship in a regression model may no longer hold at values outside the training dataset. For example, if income was an input variable and the values of income ranged from $35,000 to $90,000, applying the model to incomes well outside those incomes could result in inaccurate estimates and predictions.

The income regression example in Section 6.1.2 mentioned the possibility of using categorical variables to represent the 50 U.S. states. In a linear regression model, the state of residence would provide a simple additive term to the income model but no other impact on the coefficients of the other input variables, such as `Age` and `Education`. However, if state does influence the other variables' impact to the income model, an alternative approach would be to build 50 separate linear regression models: one model for each state. Such an approach is an example of the options and decisions that the data scientist must be willing to consider.

If several of the input variables are highly correlated to each other, the condition is known as ***multicollinearity***. Multicollinearity can often lead to coefficient estimates that are relatively large in absolute magnitude and may be of inappropriate direction (negative or positive sign). When possible, the majority of these correlated variables should be removed from the model or replaced by a new variable that is a function of the correlated variables. For example, in a medical application of regression, `height` and `weight` may be considered important input variables, but these variables tend to be correlated. In this case, it may be useful to use the Body Mass Index (BMI), which is a function of a person's height and weight.

$$BMI = \frac{weight}{height^2} \qquad \text{where weight is in kilograms and height is in meters}$$

However, in some cases it may be necessary to use the correlated variables. The next section provides some techniques to address highly correlated variables.

6.4 Additional Regression Models

In the case of multicollinearity, it may make sense to place some restrictions on the magnitudes of the estimated coefficients. ***Ridge regression***, which applies a penalty based on the size of the coefficients, is one technique that can be applied. In fitting a linear regression model, the objective is to find the values of the coefficients that minimize the sum of the residuals squared. In ridge regression, a penalty term proportional to the sum of the squares of the coefficients is added to the sum of the residuals squared. ***Lasso regression*** is a related modeling technique in which the penalty is proportional to the sum of the absolute values of the coefficients.

Only binary outcome variables were examined in the use of logistic regression. If the outcome variable can assume more than two states, ***multinomial logistic regression*** can be used.

Summary

This chapter discussed the use of linear regression and logistic regression to model historical data and to predict future outcomes. Using R, examples of each regression technique were presented. Several diagnostics to evaluate the models and the underlying assumptions were covered.

Although regression analysis is relatively straightforward to perform using many existing software packages, considerable care must be taken in performing and interpreting a regression analysis. This chapter highlighted that in a regression analysis, the data scientist needs to do the following:

- Determine the best input variables and their relationship to the outcome variable .
- Understand the underlying assumptions and their impact on the modeling results.
- Transform the variables, as appropriate, to achieve adherence to the model assumptions.
- Decide whether building one comprehensive model is the best choice or consider building many models on partitions of the data.

Exercises

1. In the Income linear regression example, consider the distribution of the outcome variable `Income`. `Income` values tend to be highly skewed to the right (distribution of value has a large tail to the right). Does such a non-normally distributed outcome variable violate the general assumption of a linear regression model? Provide supporting arguments.
2. In the use of a categorical variable with n possible values, explain the following:
   a. Why only n – 1 binary variables are necessary
   b. Why using n variables would be problematic
3. In the example of using Wyoming as the reference case, discuss the effect on the estimated model parameters, including the intercept, if another state was selected as the reference case.
4. Describe how logistic regression can be used as a classifier.
5. Discuss how the ROC curve can be used to determine an appropriate threshold value for a classifier.
6. If the probability of an event occurring is 0.4, then
   a. What is the odds ratio?
   b. What is the log odds ratio?
7. If $b_3 = -.5$ is an estimated coefficient in a linear regression model, what is the effect on the odds ratio for every one unit increase in the value of x_3?

7

Advanced Analytical Theory and Methods: Classification

Key Concepts

Classification learning
Naïve Bayes
Decision tree
ROC curve
Confusion matrix

In addition to analytical methods such as clustering (Chapter 4, "Advanced Analytical Theory and Methods: Clustering"), association rule learning Chapter 5, "Advanced Analytical Theory and Methods: Association Rules", and modeling techniques like regression (Chapter 6, "Advanced Analytical Theory and Methods: Regression"), classification is another fundamental learning method that appears in applications related to data mining. In classification learning, a classifier is presented with a set of examples that are already classified and, from these examples, the classifier learns to assign unseen examples. In other words, the primary task performed by classifiers is to assign class labels to new observations. Logistic regression from the previous chapter is one of the popular classification methods. The set of labels for classifiers is predetermined, unlike in clustering, which discovers the structure without a training set and allows the data scientist optionally to create and assign labels to the clusters.

Most classification methods are supervised, in that they start with a training set of prelabeled observations to learn how likely the attributes of these observations may contribute to the classification of future unlabeled observations. For example, existing marketing, sales, and customer demographic data can be used to develop a classifier to assign a "purchase" or "no purchase" label to potential future customers.

Classification is widely used for prediction purposes. For example, by building a classifier on the transcripts of United States Congressional floor debates, it can be determined whether the speeches represent support or opposition to proposed legislation [1]. Classification can help health care professionals diagnose heart disease patients [2]. Based on an e-mail's content, e-mail providers also use classification to decide whether the incoming e-mail messages are spam [3].

This chapter mainly focuses on two fundamental classification methods: ***decision trees*** and ***naïve Bayes***.

7.1 Decision Trees

A ***decision tree*** (also called ***prediction tree***) uses a tree structure to specify sequences of decisions and consequences. Given input $X = \{x_1, x_2, \ldots x_n\}$, the goal is to predict a response or output variable Y. Each member of the set $\{x_1, x_2, \ldots x_n\}$ is called an ***input variable***. The prediction can be achieved by constructing a decision tree with test points and branches. At each test point, a decision is made to pick a specific branch and traverse down the tree. Eventually, a final point is reached, and a prediction can be made. Each test point in a decision tree involves testing a particular input variable (or attribute), and each branch represents the decision being made. Due to its flexibility and easy visualization, decision trees are commonly deployed in data mining applications for classification purposes.

The input values of a decision tree can be categorical or continuous. A decision tree employs a structure of test points (called ***nodes***) and branches, which represent the decision being made. A node without further branches is called a ***leaf node***. The leaf nodes return class labels and, in some implementations, they return the probability scores. A decision tree can be converted into a set of decision rules. In the following example rule, `income` and `mortgage_amount` are input variables, and the response is the output variable `default` with a probability score.

```
IF income < $50,000 AND mortgage_amount > $100K
THEN default = True WITH PROBABILITY 75%
```

Decision trees have two varieties: ***classification trees*** and ***regression trees***. Classification trees usually apply to output variables that are categorical—often binary—in nature, such as yes or no, purchase or not purchase, and so on. Regression trees, on the other hand, can apply to output variables that are numeric or continuous, such as the predicted price of a consumer good or the likelihood a subscription will be purchased.

Decision trees can be applied to a variety of situations. They can be easily represented in a visual way, and the corresponding decision rules are quite straightforward. Additionally, because the result is a series of logical `if-then` statements, there is no underlying assumption of a linear (or nonlinear) relationship between the input variables and the response variable.

7.1.1 Overview of a Decision Tree

Figure 7-1 shows an example of using a decision tree to predict whether customers will buy a product. The term ***branch*** refers to the outcome of a decision and is visualized as a line connecting two nodes. If a decision is numerical, the "greater than" branch is usually placed on the right, and the "less than" branch is placed on the left. Depending on the nature of the variable, one of the branches may need to include an "equal to" component.

Internal nodes are the decision or test points. Each internal node refers to an input variable or an attribute. The top internal node is called the ***root***. The decision tree in Figure 7-1 is a binary tree in that each internal node has no more than two branches. The branching of a node is referred to as a ***split***.

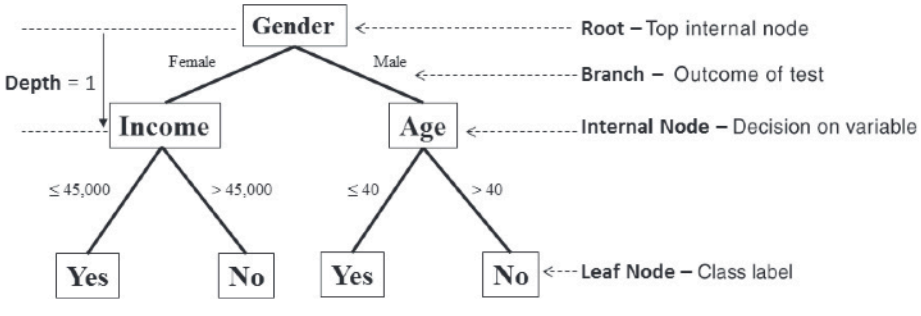


FIGURE 7-1 *Example of a decision tree*

Sometimes decision trees may have more than two branches stemming from a node. For example, if an input variable `Weather` is categorical and has three choices—`Sunny`, `Rainy`, and `Snowy`—the corresponding node `Weather` in the decision tree may have three branches labeled as `Sunny`, `Rainy`, and `Snowy`, respectively.

The ***depth*** of a node is the minimum number of steps required to reach the node from the root. In Figure 7-1 for example, nodes `Income` and `Age` have a depth of one, and the four nodes on the bottom of the tree have a depth of two.

Leaf nodes are at the end of the last branches on the tree. They represent class labels—the outcome of all the prior decisions. The path from the root to a leaf node contains a series of decisions made at various internal nodes.

In Figure 7-1, the root node splits into two branches with a `Gender` test. The right branch contains all those records with the variable `Gender` equal to `Male`, and the left branch contains all those records with the variable `Gender` equal to `Female` to create the depth 1 internal nodes. Each internal node effectively acts as the root of a subtree, and a best test for each node is determined independently of the other internal nodes. The left-hand side (LHS) internal node splits on a question based on the `Income` variable to create leaf nodes at depth 2, whereas the right-hand side (RHS) splits on a question on the `Age` variable.

The decision tree in Figure 7-1 shows that females with income less than or equal to $45,000 and males 40 years old or younger are classified as people who would purchase the product. In traversing this tree, age does not matter for females, and income does not matter for males.

Decision trees are widely used in practice. For example, to classify animals, questions (like cold-blooded or warm-blooded, mammal or not mammal) are answered to arrive at a certain classification. Another example is a checklist of symptoms during a doctor's evaluation of a patient. The artificial intelligence engine of a video game commonly uses decision trees to control the autonomous actions of a character in response to various scenarios. Retailers can use decision trees to segment customers or predict response rates to marketing and promotions. Financial institutions can use decision trees to help decide if a loan application should be approved or denied. In the case of loan approval, computers can use the logical `if-then` statements to predict whether the customer will default on the loan. For customers with a clear (strong) outcome, no human interaction is required; for observations that may not generate a clear response, a human is needed for the decision.

By limiting the number of splits, a short tree can be created. Short trees are often used as ***components*** (also called ***weak learners*** or ***base learners***) in ***ensemble methods***. Ensemble methods use multiple predictive models to vote, and decisions can be made based on the combination of the votes. Some popular ensemble methods include random forest [4], bagging, and boosting [5]. Section 7.4 discusses these ensemble methods more.

The simplest short tree is called a ***decision stump***, which is a decision tree with the root immediately connected to the leaf nodes. A decision stump makes a prediction based on the value of just a single input variable. Figure 7-2 shows a decision stump to classify two species of an iris flower based on the petal width. The figure shows that, if the petal width is smaller than 1.75 centimeters, it's Iris versicolor; otherwise, it's Iris virginica.

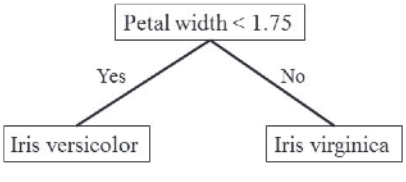


FIGURE 7-2 *Example of a decision stump*

To illustrate how a decision tree works, consider the case of a bank that wants to market its term deposit products (such as Certificates of Deposit) to the appropriate customers. Given the demographics of clients and their reactions to previous campaign phone calls, the bank's goal is to predict which clients would subscribe to a term deposit. The dataset used here is based on the original dataset collected from a Portuguese bank on directed marketing campaigns as stated in the work by Moro et al. [6]. Figure 7-3 shows a subset of the modified bank marketing dataset. This dataset includes 2,000 instances randomly drawn

from the original dataset, and each instance corresponds to a customer. To make the example simple, the subset only keeps the following categorical variables: (1) *job*, (2) *marital* status, (3) *education* level, (4) if the credit is in *default*, (5) if there is a *housing* loan, (6) if the customer currently has a personal *loan*, (7) *contact* type, (8) result of the previous marketing campaign contact (*poutcome*), and finally (9) if the client actually *subscribed* to the term deposit. Attributes (1) through (8) are input variables, and (9) is considered the outcome. The outcome *subscribed* is either `yes` (meaning the customer will subscribe to the term deposit) or `no` (meaning the customer won't subscribe). All the variables listed earlier are categorical.

	job	marital	education	default	housing	loan	contact	poutcome	subscribed
1	management	single	tertiary	no	yes	no	cellular	unknown	no
2	entrepreneur	married	tertiary	no	yes	yes	cellular	unknown	no
3	services	divorced	secondary	no	no	no	cellular	unknown	yes
4	management	married	tertiary	no	yes	no	cellular	unknown	no
5	management	married	secondary	no	yes	no	unknown	unknown	no
6	management	single	tertiary	no	yes	no	unknown	unknown	no
7	entrepreneur	married	tertiary	no	yes	no	cellular	failure	yes
8	admin.	married	secondary	no	no	no	cellular	unknown	no
9	blue-collar	married	secondary	no	yes	no	cellular	other	no
10	management	married	tertiary	yes	no	no	cellular	unknown	no
11	blue-collar	married	secondary	no	yes	no	cellular	unknown	no
12	management	divorced	secondary	no	no	no	unknown	unknown	no
13	blue-collar	married	secondary	no	yes	no	cellular	unknown	no
14	retired	married	secondary	no	no	no	cellular	unknown	no
15	management	single	tertiary	no	yes	no	cellular	unknown	no
16	retired	married	secondary	yes	yes	no	cellular	unknown	no
17	unemployed	married	secondary	no	yes	no	telephone	unknown	no
18	management	divorced	tertiary	no	yes	no	cellular	unknown	no
19	management	married	tertiary	no	yes	no	cellular	unknown	no
20	blue-collar	married	secondary	no	yes	no	unknown	unknown	no
21	management	divorced	tertiary	no	yes	yes	cellular	failure	yes
22	blue-collar	divorced	secondary	no	yes	no	cellular	failure	no
23	blue-collar	single	secondary	no	yes	no	cellular	failure	no
24	admin.	single	secondary	no	no	no	unknown	unknown	no
25	blue-collar	married	secondary	no	yes	no	cellular	failure	no
26	blue-collar	single	secondary	no	yes	no	unknown	unknown	no
27	housemaid	married	secondary	no	no	no	cellular	unknown	no
28	technician	married	tertiary	no	no	no	cellular	unknown	no

FIGURE 7-3 *A subset of the bank marketing dataset*

A summary of the dataset shows the following statistics. For ease of display, the summary only includes the top six most frequently occurring values for each attribute. The rest are displayed as `(Other)`.

```
         job          marital          education     default
 blue-collar:435   divorced: 228   primary  : 335   no :1961
 management :423   married :1201   secondary:1010   yes:  39
 technician :339   single  : 571   tertiary : 564
 admin.     :235                    unknown  :  91
 services   :168
 retired    : 92
 (Other)    :308
```

```
housing      loan           contact          month        poutcome
no : 916    no :1717    cellular :1287    may     :581    failure: 210
yes:1084    yes: 283    telephone: 136    jul     :340    other  :  79
                        unknown  : 577    aug     :278    success:  58
                                          jun     :232    unknown:1653
                                          nov     :183
                                          apr     :118
                                          (Other):268

subscribed
no :1789
yes: 211
```

Attribute *job* includes the following values.

```
     admin.   blue-collar  entrepreneur     housemaid
        235           435            70            63
  management       retired self-employed      services
        423            92            69           168
     student    technician    unemployed       unknown
          36           339            60            10
```

Figure 7-4 shows a decision tree built over the bank marketing dataset. The root of the tree shows that the overall fraction of the clients who have not subscribed to the term deposit is 1,789 out of the total population of 2,000.

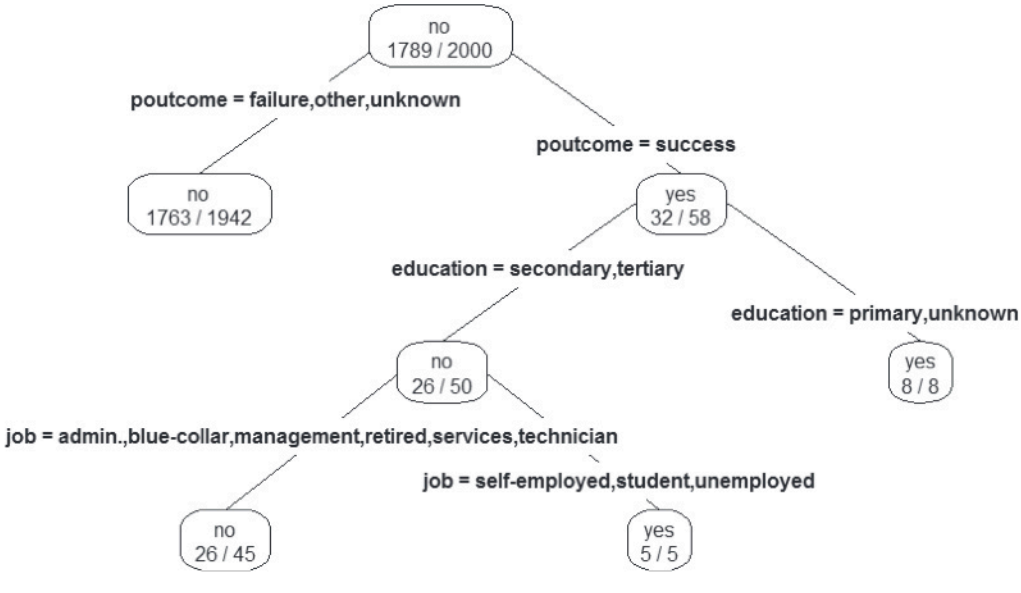


FIGURE 7-4 *Using a decision tree to predict if a client will subscribe to a term deposit*

At each split, the decision tree algorithm picks the most informative attribute out of the remaining attributes. The extent to which an attribute is informative is determined by measures such as entropy and information gain, as detailed in Section 7.1.2.

At the first split, the decision tree algorithm chooses the `poutcome` attribute. There are two nodes at depth=1. The left node is a leaf node representing a group for which the outcome of the previous marketing campaign contact is a `failure`, `other`, or `unknown`. For this group, 1,763 out of 1,942 clients have not subscribed to the term deposit.

The right node represents the *rest* of the population, for which the outcome of the previous marketing campaign contact is a `success`. For the population of this node, 32 out of 58 clients have subscribed to the term deposit.

This node further splits into two nodes based on the education level. If the `education` level is either `secondary` or `tertiary`, then 26 out of 50 of the clients have not subscribed to the term deposit. If the education level is `primary` or `unknown`, then 8 out of 8 times the clients have subscribed.

The left node at depth 2 further splits based on attribute `job`. If the occupation is `admin`, `blue collar`, `management`, `retired`, `services`, or `technician`, then 26 out of 45 clients have not subscribed. If the occupation is `self-employed`, `student`, or `unemployed`, then 5 out of 5 times the clients have subscribed.

7.1.2 The General Algorithm

In general, the objective of a decision tree algorithm is to construct a tree T from a training set S. If all the records in S belong to some class C (*subscribed* = yes, for example), or if S is sufficiently pure (greater than a preset threshold), then that node is considered a leaf node and assigned the label C. The ***purity*** of a node is defined as its probability of the corresponding class. For example, in Figure 7-4, the root $P(\textit{subscribed} = \text{yes}) = 1 - \frac{1789}{2000} = 10.55\%$; therefore, the root is only 10.55% pure on the *subscribed* = yes class. Conversely, it is 89.45% pure on the *subscribed* = no class.

In contrast, if not all the records in S belong to class C or if S is not sufficiently pure, the algorithm selects the next most informative attribute A (duration, marital, and so on) and partitions S according to A's values. The algorithm constructs subtrees $T_1, T_2 \ldots$ for the subsets of S recursively until one of the following criteria is met:

- All the leaf nodes in the tree satisfy the minimum purity threshold.
- The tree cannot be further split with the preset minimum purity threshold.
- Any other stopping criterion is satisfied (such as the maximum depth of the tree).

The first step in constructing a decision tree is to choose the most informative attribute. A common way to identify the most informative attribute is to use entropy-based methods, which are used by decision tree learning algorithms such as ID3 (or Iterative Dichotomiser 3) [7] and C4.5 [8]. The entropy methods select the most informative attribute based on two basic measures:

- ***Entropy,*** which measures the *impurity* of an attribute
- ***Information gain,*** which measures the *purity* of an attribute

Given a class X and its label $x \in X$, let $P(x)$ be the probability of x. H_x, the entropy of X, is defined as shown in Equation 7-1.

$$H_X = -\sum_{\forall x \in X} P(x) \log_2 P(x) \tag{7-1}$$

Equation 7-1 shows that entropy H_X becomes 0 when all $P(x)$ is 0 or 1. For a binary classification (`true` or `false`), H_X is zero if $P(x)$ the probability of each label x is either zero or one. On the other hand, H_X achieves the maximum entropy when all the class labels are equally probable. For a binary classification, $H_X = 1$ if the probability of all class labels is 50/50. The maximum entropy increases as the number of possible outcomes increases.

As an example of a binary random variable, consider tossing a coin with known, not necessarily fair, probabilities of coming up heads or tails. The corresponding entropy graph is shown in Figure 7-5. Let $x = 1$ represent heads and $x = 0$ represent tails. The entropy of the unknown result of the next toss is maximized when the coin is fair. That is, when heads and tails have equal probability $P(x=1) = P(x=0) = 0.5$, entropy $H_X = -(0.5 \times \log_2 0.5 + 0.5 \times \log_2 0.5) = 1$. On the other hand, if the coin is not fair, the probabilities of heads and tails would not be equal and there would be less uncertainty. As an extreme case, when the probability of tossing a head is equal to 0 or 1, the entropy is minimized to 0. Therefore, the entropy for a completely pure variable is 0 and is 1 for a set with equal occurrences for both the classes (head and tail, or yes and no).

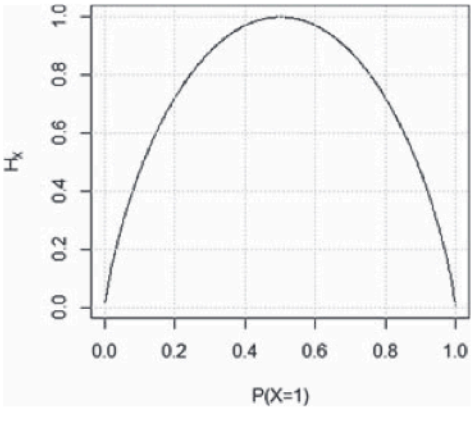


FIGURE 7-5 *Entropy of coin flips, where X=1 represents heads*

For the bank marketing scenario previously presented, the output variable is `subscribed`. The base entropy is defined as entropy of the output variable, that is $H_{subscribed}$. As seen previously, $P(subscribed = \text{yes}) = 0.1055$ and $P(subscribed = \text{no}) = 0.8945$. According to Equation 7-1, the base entropy $H_{subscribed} = -0.1055 \cdot \log_2 0.1055 - 0.8945 \cdot \log_2 0.8945 \approx 0.4862$.

The next step is to identify the conditional entropy for each attribute. Given an attribute X, its value x, its outcome Y, and its value y, conditional entropy $H_{Y|X}$ is the remaining entropy of Y given X, formally defined as shown in Equation 7-2.

$$\begin{aligned} H_{Y|X} &= \sum_{x} P(x) H(Y|X=x) \\ &= -\sum_{\forall x \in x} P(x) \sum_{\forall y \in Y} P(y|x) \log_2 P(y|x) \end{aligned} \tag{7-2}$$

Consider the banking marketing scenario, if the attribute `contact` is chosen, X = {`cellular`, `telephone`, `unknown`}. The conditional entropy of `contact` considers all three values.

Table 7-1 lists the probabilities related to the `contact` attribute. The top row of the table displays the probabilities of each value of the attribute. The next two rows contain the probabilities of the class labels conditioned on the `contact`.

TABLE 7-1 *Conditional Entropy Example*

	Cellular	Telephone	Unknown
P(contact)	0.6435	0.0680	0.2885
P(subscribed=yes | contact)	0.1399	0.0809	0.0347
P(subscribed=no | contact)	0.8601	0.9192	0.9653

The conditional entropy of the `contact` attribute is computed as shown here.

$$\begin{aligned} H_{subscribed|contact} = -\big[&0.6435 \cdot \left(0.1399 \cdot \log_2 0.1399 + 0.8601 \cdot \log_2 0.8601\right) \\ &+ 0.0680 \cdot \left(0.0809 \cdot \log_2 0.0809 + 0.9192 \cdot \log_2 0.9192\right) \\ &+ 0.2885 \cdot \left(0.0347 \cdot \log_2 0.0347 + 0.9653 \cdot \log_2 0.9653\right)\big] \\ &= 0.4661 \end{aligned}$$

Computation inside the parentheses is on the entropy of the class labels within a single `contact` value. Note that the conditional entropy is always less than or equal to the base entropy—that is, $H_{subscribed|marital} \leq H_{subscribed}$. The conditional entropy is smaller than the base entropy when the attribute and the outcome are correlated. In the worst case, when the attribute is uncorrelated with the outcome, the conditional entropy equals the base entropy.

The information gain of an attribute A is defined as the difference between the base entropy and the conditional entropy of the attribute, as shown in Equation 7-3.

$$InfoGain_A = H_S - H_{S|A} \tag{7-3}$$

In the bank marketing example, the information gain of the `contact` attribute is shown in Equation 7-4.

$$\begin{aligned} InfoGain_{contact} &= H_{subscribed} - H_{contact|subscribed} \\ &= 0.4862 - 0.4661 = 0.0201 \end{aligned} \tag{7-4}$$

Information gain compares the degree of purity of the parent node before a split with the degree of purity of the child node after a split. At each split, an attribute with the greatest information gain is considered the most informative attribute. Information gain indicates the purity of an attribute.

The result of information gain for all the input variables is shown in Table 7-2. Attribute `poutcome` has the most information gain and is the most informative variable. Therefore, `poutcome` is chosen for the first split of the decision tree, as shown in Figure 7-4. The values of information gain in Table 7-2 are small in magnitude, but the relative difference matters. The algorithm splits on the attribute with the largest information gain at each round.

TABLE 7-2 *Calculating Information Gain of Input Variables for the First Split*

Attribute	Information Gain
`poutcome`	0.0289
`contact`	0.0201
`housing`	0.0133
`job`	0.0101
`education`	0.0034
`marital`	0.0018
`loan`	0.0010
`default`	0.0005

Detecting Significant Splits

Quite often it is necessary to measure the significance of a split in a decision tree, especially when the information gain is small, like in Table 7-2.

Let N_A and N_B be the number of class A and class B in the parent node. Let N_{AL} represent the number of class A going to the left child node, N_{BL} represent the number of class B going to the left child node, N_{AR} represent the number of class B going to the right child node, and N_{BR} represent the number of class B going to the right child node.

Let p_L and p_R denote the proportion of data going to the left and right node, respectively.

$$p_L = \frac{N_{AL} + N_{BL}}{N_A + N_B}$$

$$p_R = \frac{N_{AR} + N_{BR}}{N_A + N_B}$$

The following measure computes the significance of a split. In other words, it measures how much the split deviates from what would be expected in the random data.

$$K = \frac{\left(N'_{AL} - N_{AL}\right)^2}{N'_{AL}} + \frac{\left(N'_{BL} - N_{BL}\right)^2}{N'_{BL}} + \frac{\left(N'_{AR} - N_{AR}\right)^2}{N'_{AR}} + \frac{\left(N'_{BR} - N_{BR}\right)^2}{N'_{BR}}$$

where

$$N'_{AL} = N_A \times p_L$$

$$N'_{BL} = N_B \times p_L$$

$$N'_{AR} = N_A \times p_R$$

$$N'_{BR} = N_B \times p_R$$

If K is small, the information gain from the split is not significant. If K is big, it would suggest the information gain from the split is significant.

Take the first split of the decision tree in Figure 7-4 on variable `poutcome` for example. $N_A = 1789, N_B = 211, N_{AL} = 1763, N_{BL} = 179, N_{AR} = 26, N_{BR} = 32$.

Following are the proportions of data going to the left and right node.

$$p_L = 1942/2000 = 0.971 \text{ and } p_R = 58/2000 = 0.029.$$

The N'_{AL}, N'_{BL}, N'_{AR}, and N'_{BR} represent the number of each class going to the left or right node if the data is random. Their values follow.

$$N'_{AL} = 1737.119, N'_{BL} = 204.881, N'_{AR} = 51.881 \text{ and } N'_{BR} = 6.119$$

Therefore, $K = 126.0324$, which suggests the split on `poutcome` is significant.

After each split, the algorithm looks at all the records at a leaf node, and the information gain of each candidate attribute is calculated again over these records. The next split is on the attribute with the highest information gain. A record can only belong to one leaf node after all the splits, but depending on the implementation, an attribute may appear in more than one split of the tree. This process of partitioning the records and finding the most informative attribute is repeated until the nodes are pure enough, or there is insufficient information gain by splitting on more attributes. Alternatively, one can stop the growth of the tree when all the nodes at a leaf node belong to a certain class (for example, `subscribed` = yes) or all the records have identical attribute values.

In the previous bank marketing example, to keep it simple, the dataset only includes categorical variables. Assume the dataset now includes a continuous variable called `duration`–representing the number of seconds the last phone conversation with the bank lasted as part of the previous marketing campaign. A continuous variable needs to be divided into a disjoint set of regions with the highest information gain.

A brute-force method is to consider every value of the continuous variable in the training data as a candidate split position. This brute-force method is computationally inefficient. To reduce the complexity, the training records can be sorted based on the duration, and the candidate splits can be identified by taking the midpoints between two adjacent sorted values. An examples is if the duration consists of sorted values {140, 160, 180, 200} and the candidate splits are 150, 170, and 190.

Figure 7-6 shows what the decision tree may look like when considering the `duration` attribute. The root splits into two partitions: those clients with duration < 456 seconds, and those with duration ≥ 456 seconds. Note that for aesthetic purposes, labels for the `job` and `contact` attributes in the figure are abbreviated.

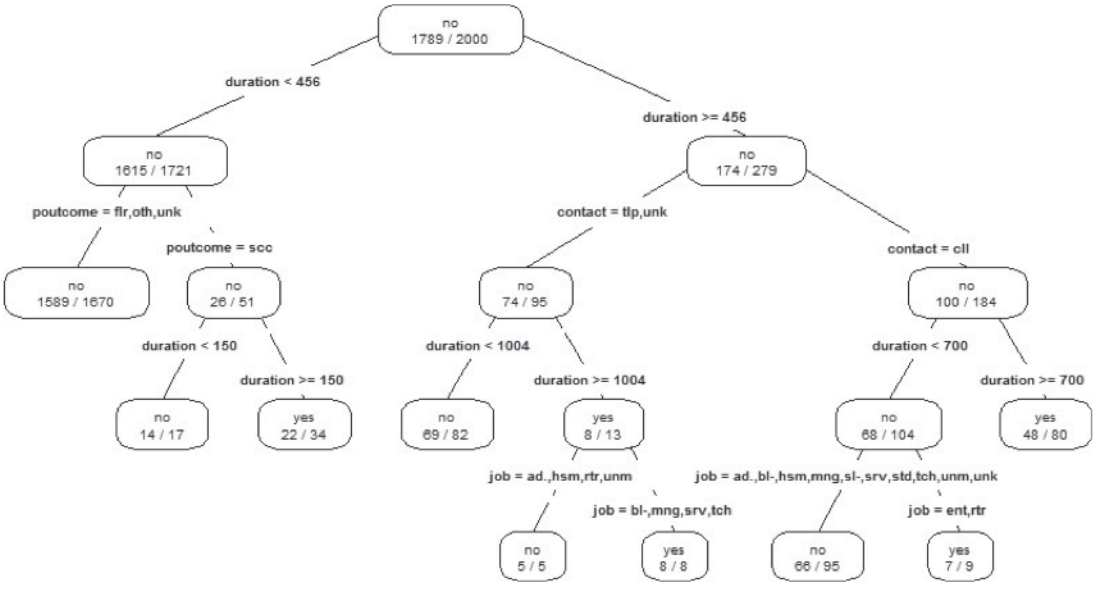


FIGURE 7-6 *Decision tree with attribute duration*

With the decision tree in Figure 7-6, it becomes trivial to predict if a new client is going to subscribe to the term deposit. For example, given the record of a new client shown in Table 7-3, the prediction is that this client will subscribe to the term deposit. The traversed paths in the decision tree are as follows.

- `duration` ≥ 456
- `contact` = cll (cellular)
- `duration` < 700
- `job` = ent (entrepreneur), rtr (retired)

TABLE 7-3 *Record of a New Client*

Job	Marital	Education	Default	Housing	Loan	Contact	Duration	Poutcome
retired	married	secondary	no	yes	No	cellular	598	unknown

7.1.3 Decision Tree Algorithms

Multiple algorithms exist to implement decision trees, and the methods of tree construction vary with different algorithms. Some popular algorithms include ID3 [7], C4.5[8], and CART [9].

ID3 Algorithm

ID3 (or Iterative Dichotomiser 3) [7] is one of the first decision tree algorithms, and it was developed by John Ross Quinlan. Let A be a set of categorical input variables, P be the output variable (or the predicted class), and T be the training set. The ID3 algorithm is shown here.

```
ID3 (A, P, T)
 if T ∈ φ
   return φ
 if all records in T have the same value for P
   return a single node with that value
 if A ∈ φ
   return a single node with the most frequent value of P in T
 Compute information gain for each attribute in A relative to T
 Pick attribute D with the largest gain
 Let { d1,d2...dm } be the values of attribute D
 Partition T into { T1,T2...Tm } according to the values of D
 return a tree with root D and branches labeled d1,d2...dm
        going respectively to trees ID3(A-{D}, P, T1),
        ID3(A-{D}, P, T2), . . . ID3(A-{D}, P, Tm)
```

C4.5

The C4.5 algorithm [8] introduces a number of improvements over the original ID3 algorithm. The C4.5 algorithm can handle missing data. If the training records contain unknown attribute values, the C4.5 evaluates the gain for an attribute by considering only the records where the attribute is defined.

Both categorical and continuous attributes are supported by C4.5. Values of a continuous variable are sorted and partitioned. For the corresponding records of each partition, the gain is calculated, and the partition that maximizes the gain is chosen for the next split.

The ID3 algorithm may construct a deep and complex tree, which would cause overfitting (Section 7.1.4). The C4.5 algorithm addresses the overfitting problem in ID3 by using a bottom-up technique called ***pruning*** to simplify the tree by removing the least visited nodes and branches.

CART

CART (or Classification And Regression Trees) [9] is often used as a generic acronym for the decision tree, although it is a specific implementation.

Similar to C4.5, CART can handle continuous attributes. Whereas C4.5 uses entropy-based criteria to rank tests, CART uses the Gini diversity index defined in Equation 7-5.

$$Gini_X = 1 - \sum_{\forall x \in X} P(x)^2 \tag{7-5}$$

Whereas C4.5 employs stopping rules, CART constructs a sequence of subtrees, uses cross-validation to estimate the misclassification cost of each subtree, and chooses the one with the lowest cost.

7.1.4 Evaluating a Decision Tree

Decision trees use ***greedy algorithms***, in that they always choose the option that seems the best available at that moment. At each step, the algorithm selects which attribute to use for splitting the remaining records. This selection may not be the best overall, but it is guaranteed to be the best at that step. This characteristic reinforces the efficiency of decision trees. However, once a bad split is taken, it is propagated through the rest of the tree. To address this problem, an ensemble technique (such as random forest) may randomize the splitting or even randomize data and come up with a multiple tree structure. These trees then vote for each class, and the class with the most votes is chosen as the predicted class.

There are a few ways to evaluate a decision tree. First, evaluate whether the splits of the tree make sense. Conduct sanity checks by validating the decision rules with domain experts, and determine if the decision rules are sound.

Next, look at the depth and nodes of the tree. Having too many layers and obtaining nodes with few members might be signs of overfitting. In overfitting, the model fits the training set well, but it performs poorly on the new samples in the testing set. Figure 7-7 illustrates the performance of an overfit model. The *x*-axis represents the amount of data, and the *y*-axis represents the errors. The blue curve is the training set, and the red curve is the testing set. The left side of the gray vertical line shows that the model predicts well on the testing set. But on the right side of the gray line, the model performs worse and worse on the testing set as more and more unseen data is introduced.

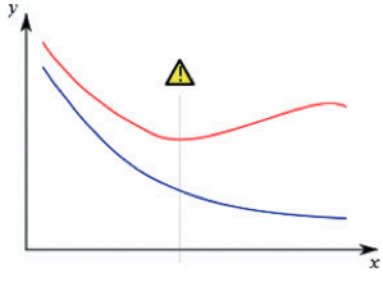


FIGURE 7-7 *An overfit model describes the training data well but predicts poorly on unseen data*

For decision tree learning, overfitting can be caused by either the lack of training data or the biased data in the training set. Two approaches [10] can help avoid overfitting in decision tree learning.

- Stop growing the tree early before it reaches the point where all the training data is perfectly classified.
- Grow the full tree, and then post-prune the tree with methods such as reduced-error pruning and rule-based post pruning.

Last, many standard diagnostics tools that apply to classifiers can help evaluate overfitting. These tools are further discussed in Section 7.3.

Decision trees are computationally inexpensive, and it is easy to classify the data. The outputs are easy to interpret as a fixed sequence of simple tests. Many decision tree algorithms are able to show the importance of each input variable. Basic measures, such as information gain, are provided by most statistical software packages.

Decision trees are able to handle both numerical and categorical attributes and are robust with redundant or correlated variables. Decision trees can handle categorical attributes with many distinct values, such as country codes for telephone numbers. Decision trees can also handle variables that have a nonlinear effect on the outcome, so they work better than linear models (for example, linear regression and logistic regression) for highly nonlinear problems. Decision trees naturally handle variable interactions. Every node in the tree depends on the preceding nodes in the tree.

In a decision tree, the decision regions are rectangular surfaces. Figure 7-8 shows an example of five rectangular decision surfaces (A, B, C, D, and E) defined by four values—$\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$—of two attributes ($x_1$ and x_2). The corresponding decision tree is on the right side of the figure. A decision surface corresponds to a leaf node of the tree, and it can be reached by traversing from the root of the tree following by a series of decisions according to the value of an attribute. The decision surface can only be axis-aligned for the decision tree.

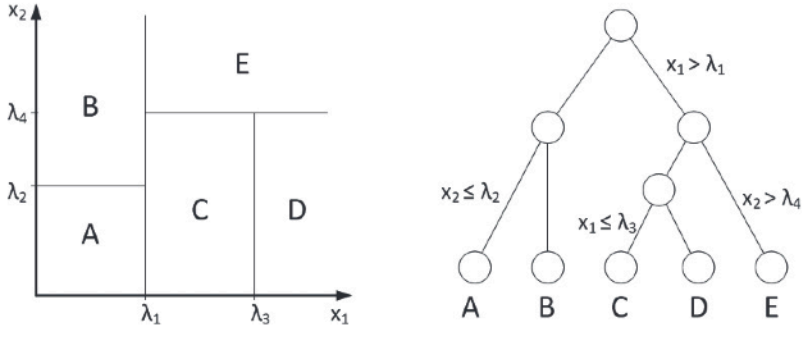


FIGURE 7-8 *Decision surfaces can only be axis-aligned*

The structure of a decision tree is sensitive to small variations in the training data. Although the dataset is the same, constructing two decision trees based on two different subsets may result in very different trees. If a tree is too deep, overfitting may occur, because each split reduces the training data for subsequent splits.

Decision trees are not a good choice if the dataset contains many irrelevant variables. This is different from the notion that they are robust with redundant variables and correlated variables. If the dataset

contains redundant variables, the resulting decision tree ignores all but one of these variables because the algorithm cannot detect information gain by including more redundant variables. On the other hand, if the dataset contains irrelevant variables and if these variables are accidentally chosen as splits in the tree, the tree may grow too large and may end up with less data at every split, where overfitting is likely to occur. To address this problem, feature selection can be introduced in the data preprocessing phase to eliminate the irrelevant variables.

Although decision trees are able to handle correlated variables, decision trees are not well suited when most of the variables in the training set are correlated, since overfitting is likely to occur. To overcome the issue of instability and potential overfitting of deep trees, one can combine the decisions of several randomized shallow decision trees—the basic idea of another classifier called random forest [4]—or use ensemble methods to combine several weak learners for better classification. These methods have been shown to improve predictive power compared to a single decision tree.

For binary decisions, a decision tree works better if the training dataset consists of records with an even probability of each result. In other words, the root of the tree has a 50% chance of either classification. This occurs by randomly selecting training records from each possible classification in equal numbers. It counteracts the likelihood that a tree will stump out early by passing purity tests because of bias in the training data.

When using methods such as logistic regression on a dataset with many variables, decision trees can help determine which variables are the most useful to select based on information gain. Then these variables can be selected for the logistic regression. Decision trees can also be used to prune redundant variables.

7.1.5 Decision Trees in R

In R, `rpart` is for modeling decision trees, and an optional package `rpart.plot` enables the plotting of a tree. The rest of this section shows an example of how to use decision trees in R with `rpart.plot` to predict whether to play golf given factors such as weather outlook, temperature, humidity, and wind.

In R, first set the working directory and initialize the packages.

```
setwd("c:/")
install.packages("rpart.plot")   # install package rpart.plot
library("rpart")   # load libraries
library("rpart.plot")
```

The working directory contains a comma-separated-value (CSV) file named `DTdata.csv`. The file has a header row, followed by 10 rows of training data.

```
Play,Outlook,Temperature,Humidity,Wind
yes,rainy,cool,normal,FALSE
no,rainy,cool,normal,TRUE
yes,overcast,hot,high,FALSE
no,sunny,mild,high,FALSE
yes,rainy,cool,normal,FALSE
yes,sunny,cool,normal,FALSE
yes,rainy,cool,normal,FALSE
yes,sunny,hot,normal,FALSE
yes,overcast,mild,high,TRUE
no,sunny,mild,high,TRUE
```

The CSV file contains five attributes: *Play, Outlook, Temperature, Humidity,* and *Wind. Play* would be the output variable (or the predicted class), and *Outlook, Temperature, Humidity,* and *Wind* would be the input variables. In R, read the data from the CSV file in the working directory and display the content.

```
play_decision <- read.table("DTdata.csv",header=TRUE,sep=",")
play_decision
   Play  Outlook Temperature Humidity  Wind
1   yes    rainy        cool   normal FALSE
2    no    rainy        cool   normal  TRUE
3   yes overcast         hot     high FALSE
4    no    sunny        mild     high FALSE
5   yes    rainy        cool   normal FALSE
6   yes    sunny        cool   normal FALSE
7   yes    rainy        cool   normal FALSE
8   yes    sunny         hot   normal FALSE
9   yes overcast        mild     high  TRUE
10   no    sunny        mild     high  TRUE
```

Display a summary of *play_decision*.

```
summary(play_decision)

  Play       Outlook  Temperature   Humidity    Wind
 no :3   overcast:2   cool:5      high  :4   Mode :logical
 yes:7   rainy   :4   hot :2      normal:6   FALSE:7
         sunny   :4   mild:3                 TRUE :3
                                             NA's :0
```

The `rpart` function builds a model of recursive partitioning and regression trees [9]. The following code snippet shows how to use the `rpart` function to construct a decision tree.

```
fit <- rpart(Play ~ Outlook + Temperature + Humidity + Wind,
             method="class",
             data=play_decision,
             control=rpart.control(minsplit=1),
             parms=list(split='information'))
```

The `rpart` function has four parameters. The first parameter, `Play ~ Outlook + Temperature + Humidity + Wind`, is the model indicating that attribute *Play* can be predicted based on attributes *Outlook, Temperature, Humidity,* and *Wind*. The second parameter, *method*, is set to "class," telling R it is building a classification tree. The third parameter, *data*, specifies the dataframe containing those attributes mentioned in the formula. The fourth parameter, *control*, is optional and controls the tree growth. In the preceding example, `control=rpart.control(minsplit=1)` requires that each node have at least one observation before attempting a split. The `minsplit=1` makes sense for the small dataset, but for larger datasets `minsplit` could be set to 10% of the dataset size to combat overfitting. Besides `minsplit`, other parameters are available to control the construction of the decision tree. For example, `rpart.control(maxdepth=10,cp=0.001)` limits the depth of the

tree to no more than 10, and a split must decrease the overall lack of fit by a factor of 0.001 before being attempted. The last parameter (`parms`) specifies the purity measure being used for the splits. The value of `split` can be either `information` (for using the information gain) or `gini` (for using the Gini index).

Enter **`summary(fit)`** to produce a summary of the model built from `rpart`.

The output includes a summary of every node in the constructed decision tree. If a node is a leaf, the output includes both the predicted class label (*`yes`* or *`no`* for *`Play`*) and the class probabilities—`P(Play)`. The leaf nodes include node numbers 4, 5, 6, and 7. If a node is internal, the output in addition displays the number of observations that lead to each child node and the improvement that each attribute may bring for the next split. These internal nodes include numbers 1, 2, and 3.

```
summary(fit)
Call:
rpart(formula = Play ~ Outlook + Temperature + Humidity + Wind,
  data = play_decision, method = "class",
  parms = list(split = "information"),
      control = rpart.control(minsplit = 1))
  n= 10
         CP nsplit rel error   xerror      xstd
1 0.3333333      0         1 1.000000 0.4830459
2 0.0100000      3         0 1.666667 0.5270463

Variable importance
       Wind     Outlook Temperature
         51          29          20

Node number 1: 10 observations,    complexity param=0.3333333
  predicted class=yes  expected loss=0.3  P(node) =1
    class counts:     3     7
   probabilities: 0.300 0.700
  left son=2 (3 obs) right son=3 (7 obs)
  Primary splits:
      Temperature splits as  RRL,     improve=1.3282860, (0 missing)
      Wind        < 0.5 to the right, improve=1.3282860, (0 missing)
      Outlook     splits as  RLL,     improve=0.8161371, (0 missing)
      Humidity    splits as  LR,      improve=0.6326870, (0 missing)
  Surrogate splits:
      Wind < 0.5 to the right, agree=0.8, adj=0.333, (0 split)

Node number 2: 3 observations,    complexity param=0.3333333
  predicted class=no   expected loss=0.3333333  P(node) =0.3
    class counts:     2     1
   probabilities: 0.667 0.333
```

```
  left son=4 (2 obs) right son=5 (1 obs)
  Primary splits:
      Outlook splits as  R-L,     improve=1.9095430, (0 missing)
      Wind    < 0.5 to the left,  improve=0.5232481, (0 missing)

Node number 3: 7 observations,    complexity param=0.3333333
  predicted class=yes  expected loss=0.1428571  P(node) =0.7
    class counts:     1     6
   probabilities: 0.143 0.857
  left son=6 (1 obs) right son=7 (6 obs)
  Primary splits:
      Wind        < 0.5 to the right, improve=2.8708140, (0 missing)
      Outlook     splits as  RLR,     improve=0.6214736, (0 missing)
      Temperature splits as  LR-,     improve=0.3688021, (0 missing)
      Humidity    splits as  RL,      improve=0.1674470, (0 missing)

Node number 4: 2 observations
  predicted class=no   expected loss=0  P(node) =0.2
    class counts:     2     0
   probabilities: 1.000 0.000

Node number 5: 1 observations
  predicted class=yes  expected loss=0  P(node) =0.1
    class counts:     0     1
   probabilities: 0.000 1.000

Node number 6: 1 observations
  predicted class=no   expected loss=0  P(node) =0.1
    class counts:     1     0
   probabilities: 1.000 0.000

Node number 7: 6 observations
  predicted class=yes  expected loss=0  P(node) =0.6
    class counts:     0     6
   probabilities: 0.000 1.000
```

The output produced by the summary is difficult to read and comprehend. The `rpart.plot()` function from the `rpart.plot` package can visually represent the output in a decision tree. Enter the following command to see the help file of `rpart.plot`:

```
?rpart.plot
```

Enter the following R code to plot the tree based on the model being built. The resulting tree is shown in Figure 7-9. Each node of the tree is labeled as either `yes` or `no` referring to the *`Play`* action of whether to play outside. Note that, by default, R has converted the values of *`Wind`* (`True/False`) into numbers.

```
rpart.plot(fit, type=4, extra=1)
```

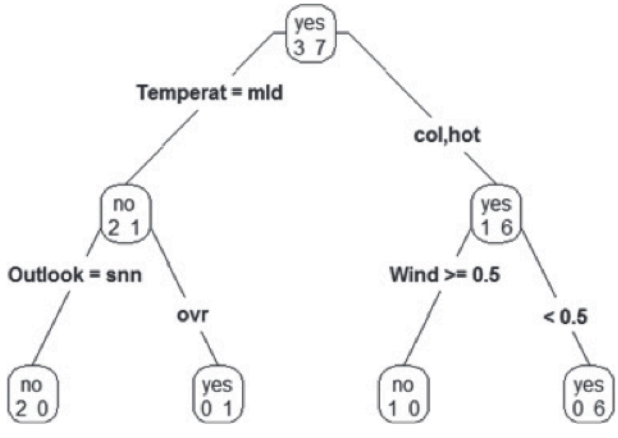


FIGURE 7-9 *A decision tree built from DTdata.csv*

The decisions in Figure 7-9 are abbreviated. Use the following command to spell out the full names and display the classification rate at each node.

```
rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE,
           varlen=0, faclen=0)
```

The decision tree can be used to predict outcomes for new datasets. Consider a testing set that contains the following record.

```
Outlook="rainy", Temperature="mild", Humidity="high", Wind=FALSE
```

The goal is to predict the play decision of this record. The following code loads the data into R as a data frame *`newdata`*. Note that the training set does not contain this case.

```
newdata <- data.frame(Outlook="rainy", Temperature="mild",
                      Humidity="high", Wind=FALSE)

newdata
  Outlook Temperature Humidity  Wind
1   rainy        mild     high FALSE
```

Next, use the *`predict`* function to generate predictions from a fitted *`rpart`* object. The format of the `predict` function follows.

```
predict(object, newdata = list(),
        type = c("vector", "prob", "class", "matrix"))
```

Parameter *type* is a character string denoting the type of the predicted value. Set it to either `prob` or `class` to predict using a decision tree model and receive the result as either the class probabilities or just the class. The output shows that one instance is classified as *Play*=`no`, and zero instances are classified as *Play*=`yes`. Therefore, in both cases, the decision tree predicts that the play decision of the testing set is not to play.

```
predict(fit,newdata=newdata,type="prob")

  no yes
1  1   0


predict(fit,newdata=newdata,type="class")

  1
  no
Levels: no yes
```

7.2 Naïve Bayes

Naïve Bayes is a probabilistic classification method based on Bayes' theorem (or Bayes' law) with a few tweaks. Bayes' theorem gives the relationship between the probabilities of two events and their conditional probabilities. Bayes' law is named after the English mathematician Thomas Bayes.

A naïve Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of other features. For example, an object can be classified based on its attributes such as shape, color, and weight. A reasonable classification for an object that is spherical, yellow, and less than 60 grams in weight may be a tennis ball. Even if these features depend on each other or upon the existence of the other features, a naïve Bayes classifier considers all these properties to contribute *independently* to the probability that the object is a tennis ball.

The input variables are generally categorical, but variations of the algorithm can accept continuous variables. There are also ways to convert continuous variables into categorical ones. This process is often referred to as the ***discretization of continuous variables***. In the tennis ball example, a continuous variable such as weight can be grouped into intervals to be converted into a categorical variable. For an attribute such as *income*, the attribute can be converted into categorical values as shown below.

- **Low Income:** income < \$10,000
- **Working Class:** \$10,000 ≤ income < \$50,000
- **Middle Class:** \$50,000 ≤ income < \$1,000,000
- **Upper Class:** income ≥ \$1,000,000

The output typically includes a class label and its corresponding probability score. The probability score is not the true probability of the class label, but it's proportional to the true probability. As shown later in the chapter, in most implementations, the output includes the log probability for the class, and class labels are assigned based on the highest values.

Because naïve Bayes classifiers are easy to implement and can execute efficiently even without prior knowledge of the data, they are among the most popular algorithms for classifying text documents. Spam filtering is a classic use case of naïve Bayes text classification. Bayesian spam filtering has become a popular mechanism to distinguish spam e-mail from legitimate e-mail. Many modern mail clients implement variants of Bayesian spam filtering.

Naïve Bayes classifiers can also be used for fraud detection [11]. In the domain of auto insurance, for example, based on a training set with attributes such as driver's rating, vehicle age, vehicle price, historical claims by the policy holder, police report status, and claim genuineness, naïve Bayes can provide probability-based classification of whether a new claim is genuine [12].

7.2.1 Bayes' Theorem

The ***conditional probability*** of event C occurring, given that event A has already occurred, is denoted as $P(C|A)$, which can be found using the formula in Equation 7-6.

$$P(C|A) = \frac{P(A \cap C)}{P(A)} \tag{7-6}$$

Equation 7-7 can be obtained with some minor algebra and substitution of the conditional probability:

$$P(C|A) = \frac{P(A|C) \cdot P(C)}{P(A)} \tag{7-7}$$

where C is the class label $C \in \{c_1, c_2, \ldots c_n\}$ and A is the observed attributes $A = \{a_1, a_2, \ldots a_m\}$. Equation 7-7 is the most common form of the ***Bayes' theorem***.

Mathematically, Bayes' theorem gives the relationship between the probabilities of *C* and *A*, *P*(*C*) and *P*(*A*), and the conditional probabilities of *C* given *A* and *A* given *C*, namely $P(C|A)$ and $P(A|C)$.

Bayes' theorem is significant because quite often $P(C|A)$ is much more difficult to compute than $P(A|C)$ and $P(C)$ from the training data. By using Bayes' theorem, this problem can be circumvented.

An example better illustrates the use of Bayes' theorem. John flies frequently and likes to upgrade his seat to first class. He has determined that if he checks in for his flight at least two hours early, the probability that he will get an upgrade is 0.75; otherwise, the probability that he will get an upgrade is 0.35. With his busy schedule, he checks in at least two hours before his flight only 40% of the time. Suppose John did not receive an upgrade on his most recent attempt. What is the probability that he did not arrive two hours early?

Let C = {John arrived at least two hours early}, and A = {John received an upgrade}, then $\neg C$ = {John did not arrive two hours early}, and $\neg A$ = {John did not receive an upgrade}.

John checked in at least two hours early only 40% of the time, or $P(C) = 0.4$. Therefore, $P(\neg C) = 1 - P(C) = 0.6$.

The probability that John received an upgrade given that he checked in early is 0.75, or $P(A|C) = 0.75$.

The probability that John received an upgrade given that he did not arrive two hours early is 0.35, or $P(A|\neg C) = 0.35$. Therefore, $P(\neg A|\neg C) = 0.65$.

The probability that John received an upgrade $P(A)$ can be computed as shown in Equation 7-8.

$$\begin{aligned} P(A) &= P(A \cap C) + P(A \cap \neg C) \\ &= P(C) \cdot P(A|C) + P(\neg C) \cdot P(A|\neg C) \\ &= 0.4 \times 0.75 + 0.6 \times 0.35 \\ &= 0.51 \end{aligned} \tag{7-8}$$

Thus, the probability that John did not receive an upgrade $P(\neg A) = 0.49$. Using Bayes' theorem, the probability that John did not arrive two hours early given that he did not receive his upgrade is shown in Equation 7-9.

$$\begin{aligned} P(\neg C|\neg A) &= \frac{P(\neg A|\neg C) \cdot P(\neg C)}{P(\neg A)} \\ &= \frac{0.65 \times 0.6}{0.49} \approx 0.796 \end{aligned} \tag{7-9}$$

Another example involves computing the probability that a patient carries a disease based on the result of a lab test. Assume that a patient named Mary took a lab test for a certain disease and the result came back positive. The test returns a positive result in 95% of the cases in which the disease is actually present, and it returns a positive result in 6% of the cases in which the disease is not present. Furthermore, 1% of the entire population has this disease. What is the probability that Mary actually has the disease, given that the test is positive?

Let C = {having the disease} and A = {testing positive}. The goal is to solve the probability of having the disease, given that Mary has a positive test result, $P(C|A)$. From the problem description, $P(C) = 0.01$, $P(\neg C) = 0.99$, $P(A|C) = 0.95$ and $P(A|\neg C) = 0.06$.

Bayes' theorem defines $P(C|A) = P(A|C)P(C)/P(A)$. The probability of testing positive, that is $P(A)$, needs to be computed first. That computation is shown in Equation 7-10.

$$\begin{aligned} P(A) &= P(A \cap C) + P(A \cap \neg C) \\ &= P(C) \cdot P(A|C) + P(\neg C) \cdot P(A|\neg C) \\ &= 0.01 \times 0.95 + 0.99 \times 0.06 = 0.0689 \end{aligned} \tag{7-10}$$

According to Bayes' theorem, the probability of having the disease, given that Mary has a positive test result, is shown in Equation 7-11.

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)} = \frac{0.95 \times 0.01}{0.0689} \approx 0.1379 \tag{7-11}$$

That means that the probability of Mary actually having the disease given a positive test result is only 13.79%. This result indicates that the lab test may not be a good one. The likelihood of having the disease

was 1% when the patient walked in the door and only 13.79% when the patient walked out, which would suggest further tests.

A more general form of Bayes' theorem assigns a classified label to an object with multiple attributes $A = \{a_1, a_2, \ldots, a_m\}$ such that the label corresponds to the largest value of $P(c_i|A)$. The probability that a set of attribute values A (composed of m variables $a_1, a_2, \ldots, a_m$) should be labeled with a classification label c_i equals the probability that the set of variables $a_1, a_2, \ldots, a_m$ given c_i is true, times the probability of c_i divided by the probability of $a_1, a_2, \ldots, a_m$. Mathematically, this is shown in Equation 7-12.

$$P(c_i|A) = \frac{P(a_1, a_2, \ldots, a_m|c_i) \cdot P(c_i)}{P(a_1, a_2, \ldots, a_m)}, i = 1, 2, \ldots n \tag{7-12}$$

Consider the bank marketing example presented in Section 7.1 on predicting if a customer would subscribe to a term deposit. Let A be a list of attributes {`job`, `marital`, `education`, `default`, `housing`, `loan`, `contact`, `poutcome`}. According to Equation 7-12, the problem is essentially to calculate $P(c_i|A)$, where $c_i \in \{subscribed = yes, subscribed = no\}$.

7.2.2 Naïve Bayes Classifier

With two simplifications, Bayes' theorem can be extended to become a naïve Bayes classifier.

The first simplification is to use the conditional independence assumption. That is, each attribute is conditionally independent of every other attribute given a class label c_i. See Equation 7-13.

$$P(a_1, a_2, \ldots, a_m|c_i) = P(a_1|c_i)P(a_2|c_i) \cdots P(a_m|c_i) = \prod_{j=1}^{m} P(a_j|c_i) \tag{7-13}$$

Therefore, this naïve assumption simplifies the computation of $P(a_1, a_2, \ldots, a_m|c_i)$.

The second simplification is to ignore the denominator $P(a_1, a_2, \ldots, a_m)$. Because $P(a_1, a_2, \ldots, a_m)$ appears in the denominator of $P(c_i|A)$ for all values of i, removing the denominator will have no impact on the relative probability scores and will simplify calculations.

Naïve Bayes classification applies the two simplifications mentioned earlier and, as a result, $P(c_i|a_1, a_2, \ldots, a_m)$ is proportional to the product of $P(a_j|c_i)$ times $P(c_i)$. This is shown in Equation 7-14.

$$P(c_i|A) \propto P(c_i) \cdot \prod_{j=1}^{m} P(a_j|c_i) \qquad i = 1, 2, \ldots n \tag{7-14}$$

The mathematical symbol $\propto$ indicates that the LHS $P(c_i|A)$ is directly proportional to the RHS.

Section 7.1 has introduced a bank marketing dataset (Figure 7-3). This section shows how to use the naïve Bayes classifier on this dataset to predict if the clients would subscribe to a term deposit.

Building a naïve Bayes classifier requires knowing certain statistics, all calculated from the training set. The first requirement is to collect the probabilities of all class labels, $P(c_i)$. In the presented example, these would be the probability that a client will subscribe to the term deposit and the probability the client will not. From the data available in the training set, $P(subscribed = \text{yes}) \approx 0.11$ and $P(subscribed = \text{no}) \approx 0.89$.

The second thing the naïve Bayes classifier needs to know is the conditional probabilities of each attribute a_j given each class label c_i, namely $P(a_j|c_i)$. The training set contains several attributes: `job`, `marital`, `education`, `default`, `housing`, `loan`, `contact`, and `poutcome`. For each attribute and its possible values, computing the conditional probabilities given *subscribed* = yes or *subscribed* = no is required. For example, relative to the `marital` attribute, the following conditional probabilities are calculated.

$$P(single \mid subscribed = yes) \approx 0.35$$

$$P(married \mid subscribed = yes) \approx 0.53$$

$$P(divorced \mid subscribed = yes) \approx 0.12$$

$$P(single \mid subscribed = no) \approx 0.28$$

$$P(married \mid subscribed = no) \approx 0.61$$

$$P(divorced \mid subscribed = no) \approx 0.11$$

After training the classifier and computing all the required statistics, the naïve Bayes classifier can be tested over the testing set. For each record in the testing set, the naïve Bayes classifier assigns the classifier label c_i that maximizes $P(c_i) \cdot \prod_{j=1}^{m} P(a_j|c_i)$.

Table 7-4 contains a single record for a client who has a career in management, is married, holds a secondary degree, has credit not in default, has a housing loan but no personal loans, prefers to be contacted via cellular, and whose outcome of the previous marketing campaign contact was a success. Is this client likely to subscribe to the term deposit?

TABLE 7-4 *Record of an Additional Client*

Job	Marital	Education	Default	Housing	Loan	Contact	Poutcome
management	married	secondary	no	yes	no	cellular	Success

The conditional probabilities shown in Table 7-5 can be calculated after building the classifier with the training set.

TABLE 7-5 *Compute Conditional Probabilities for the New Record*

j	a_j	$P(a_j \mid \text{subscribed} = \text{yes})$	$P(a_j \mid \text{subscribed} = \text{no})$
1	job = management	0.22	0.21
2	marital = married	0.53	0.61
3	education = secondary	0.46	0.51
4	default = no	0.99	0.98
5	housing = yes	0.35	0.57
6	loan = no	0.90	0.85
7	contact = cellular	0.85	0.62
8	poutcome = success	0.15	0.01

Because $P(c_i|a_1, a_2, \ldots, a_m)$ is proportional to the product of $P(a_j|c_i)(j \in [1,m])$ times (c_i), the naïve Bayes classifier assigns the class label c_i, which results in the greatest value over all i. Thus, $P(c_i|a_1, a_2, \ldots, a_m)$ is computed for each c_i with $P(c_i|A) \propto P(c_i) \cdot \prod_{j=1}^{m} P\left(a_j|c_i\right)$.

For A = {management, married, secondary, no, yes, no, cellular, success},

$$P\left(yes|A\right) \propto 0.11 \cdot \left(0.22 \cdot 0.53 \cdot 0.46 \cdot 0.99 \cdot 0.35 \cdot 0.90 \cdot 0.85 \cdot 0.15\right) \approx 0.00023$$

$$P\left(no|A\right) \propto 0.89 \cdot \left(0.21 \cdot 0.61 \cdot 0.51 \cdot 0.98 \cdot 0.57 \cdot 0.85 \cdot 0.62 \cdot 0.01\right) \approx 0.00017$$

Because $P(subscribed = yes|A) > P(subscribed = no|A)$, the client shown in Table 7-4 is assigned with the label *subscribed* = *yes*. That is, the client is classified as likely to subscribe to the term deposit.

Although the scores are small in magnitude, it is the ratio of $P\left(yes|A\right)$ and $P\left(no|A\right)$ that matters. In fact, the scores of $P\left(yes|A\right)$ and $P\left(no|A\right)$ are not the true probabilities but are only proportional to the true probabilities, as shown in Equation 7-14. After all, if the scores were indeed the true probabilities, the sum of $P\left(yes|A\right)$ and $P\left(no|A\right)$ would be equal to one. When looking at problems with a large number of attributes, or attributes with a high number of levels, these values can become very small in magnitude (close to zero), resulting in even smaller differences of the scores. This is the problem of ***numerical underflow***, caused by multiplying several probability values that are close to zero. A way to alleviate the problem is to compute

the logarithm of the products, which is equivalent to the summation of the logarithm of the probabilities. Thus, the naïve Bayes formula can be rewritten as shown in Equation 7-15.

$$P(c_i|A) \propto \log P(c_i) + \sum_{j=1}^{m} \log P(a_j|c_i) \qquad i = 1,2,\ldots n \tag{7-15}$$

Although the risk of underflow may increase as the number of attributes increases, the use of logarithms is usually applied regardless of the number of attribute dimensions.

7.2.3 Smoothing

If one of the attribute values does not appear with one of the class labels within the training set, the corresponding $P(a_j|c_i)$ will equal zero. When this happens, the resulting $P(c_i|A)$ from multiplying all the $P(a_j|c_i)(j \in [1,m])$ immediately becomes zero regardless of how large some of the conditional probabilities are. Therefore overfitting occurs. Smoothing techniques can be employed to adjust the probabilities of $P(a_j|c_i)$ and to ensure a nonzero value of $P(c_i|A)$. A smoothing technique assigns a small nonzero probability to rare events not included in the training dataset. Also, the smoothing addresses the possibility of taking the logarithm of zero that may occur in Equation 7-15.

There are various smoothing techniques. Among them is the ***Laplace smoothing*** (or add-one) technique that pretends to see every outcome once more than it actually appears. This technique is shown in Equation 7-16.

$$P^*(x) = \frac{count(x)+1}{\Sigma_x[count(x)+1]} \tag{7-16}$$

For example, say that 100 clients subscribe to the term deposit, with 20 of them single, 70 married, and 10 divorced. The "raw" probability is $P(single|subscribed = yes) = 20/100 = 0.2$. With Laplace smoothing adding one to the counts, the adjusted probability becomes $P'(single|subscribed = yes) = (20+1)/[(20+1)+(70+1)+(10+1)] \approx 0.2039$.

One problem of the Laplace smoothing is that it may assign too much probability to unseen events. To address this problem, Laplace smoothing can be generalized to use ε instead of 1, where typically $\varepsilon \in [0,1]$. See Equation 7-17.

$$P^{**}(x) = \frac{count(x)+\varepsilon}{\Sigma_x[count(x)+\varepsilon]} \tag{7-17}$$

Smoothing techniques are available in most standard software packages for naïve Bayes classifiers. However, if for some reason (like performance concerns) the naïve Bayes classifier needs to be coded directly into an application, the smoothing and logarithm calculations should be incorporated into the implementation.

7.2.4 Diagnostics

Unlike logistic regression, naïve Bayes classifiers can handle missing values. Naïve Bayes is also robust to irrelevant variables—variables that are distributed among all the classes whose effects are not pronounced.

The model is simple to implement even without using libraries. The prediction is based on counting the occurrences of events, making the classifier efficient to run. Naïve Bayes is computationally efficient and is able to handle high-dimensional data efficiently. Related research [13] shows that the naive Bayes classifier in many cases is competitive with other learning algorithms, including decision trees and neural networks. In some cases naïve Bayes even outperforms other methods. Unlike logistic regression, the naïve Bayes classifier can handle categorical variables with many levels. Recall that decision trees can handle categorical variables as well, but too many levels may result in a deep tree. The naïve Bayes classifier overall performs better than decision trees on categorical values with many levels. Compared to decision trees, naïve Bayes is more resistant to overfitting, especially with the presence of a smoothing technique.

Despite the benefits of naïve Bayes, it also comes with a few disadvantages. Naïve Bayes assumes the variables in the data are conditionally independent. Therefore, it is sensitive to correlated variables because the algorithm may double count the effects. As an example, assume that people with low income and low credit tend to default. If the task is to score "default" based on both income and credit as two separate attributes, naïve Bayes would experience the double-counting effect on the default outcome, thus reducing the accuracy of the prediction.

Although probabilities are provided as part of the output for the prediction, naïve Bayes classifiers in general are not very reliable for probability estimation and should be used only for assigning class labels. Naïve Bayes in its simple form is used only with categorical variables. Any continuous variables should be converted into a categorical variable with the process known as discretization, as shown earlier. In common statistical software packages, however, naïve Bayes is implemented in a way that enables it to handle continuous variables as well.

7.2.5 Naïve Bayes in R

This section explores two methods of using the naïve Bayes classifier in R. The first method is to build from scratch by manually computing the probability scores, and the second method is to use the *naiveBayes* function from the *e1071* package. The examples show how to use naïve Bayes to predict whether employees would enroll in an onsite educational program.

In R, first set up the working directory and initialize the packages.

```
setwd("c:/")
install.packages("e1071")   # install package e1071
library(e1071)   # load the library
```

The working directory contains a CSV file (`sample1.csv`). The file has a header row, followed by 14 rows of training data. The attributes include *Age*, *Income*, *JobSatisfaction*, and *Desire*. The output variable is *Enrolls*, and its value is either `Yes` or `No`. Full content of the CSV file is shown next.

```
Age,Income,JobSatisfaction,Desire,Enrolls
<=30,High,No,Fair,No
<=30,High,No,Excellent,No
31 to 40,High,No,Fair,Yes
>40,Medium,No,Fair,Yes
>40,Low,Yes,Fair,Yes
>40,Low,Yes,Excellent,No
31 to 40,Low,Yes,Excellent,Yes
```

```
<=30,Medium,No,Fair,No
<=30,Low,Yes,Fair,Yes
>40,Medium,Yes,Fair,Yes
<=30,Medium,Yes,Excellent,Yes
31 to 40,Medium,No,Excellent,Yes
31 to 40,High,Yes,Fair,Yes
>40,Medium,No,Excellent,No
<=30,Medium,Yes,Fair,
```

The last record of the CSV is used later for illustrative purposes as a test case. Therefore, it does not include a value for the output variable *Enrolls*, which should be predicted using the naïve Bayes classifier built from the training set.

Execute the following R code to read data from the CSV file.

```
# read the data into a table from the file
sample <- read.table("sample1.csv",header=TRUE,sep=",")
# define the data frames for the NB classifier
traindata <- as.data.frame(sample[1:14,])
testdata <- as.data.frame(sample[15,])
```

Two data frame objects called *traindata* and *testdata* are created for the naïve Bayes classifier. Enter *traindata* and *testdata* to display the data frames.

The two data frames are printed on the screen as follows.

```
traindata
        Age Income JobSatisfaction    Desire Enrolls
1      <=30   High              No      Fair      No
2      <=30   High              No Excellent      No
3  31 to 40   High              No      Fair     Yes
4       >40 Medium              No      Fair     Yes
5       >40    Low             Yes      Fair     Yes
6       >40    Low             Yes Excellent      No
7  31 to 40    Low             Yes Excellent     Yes
8      <=30 Medium              No      Fair      No
9      <=30    Low             Yes      Fair     Yes
10      >40 Medium             Yes      Fair     Yes
11     <=30 Medium             Yes Excellent     Yes
12 31 to 40 Medium              No Excellent     Yes
13 31 to 40   High             Yes      Fair     Yes
14      >40 Medium              No Excellent      No

testdata
    Age Income JobSatisfaction Desire Enrolls
15 <=30 Medium             Yes   Fair
```

The first method shown here is to build a naïve Bayes classifier from scratch by manually computing the probability scores. The first step in building a classifier is to compute the prior probabilities of the attributes,

including *Age*, *Income*, *JobSatisfaction*, and *Desire*. According to the naïve Bayes classifier, these attributes are conditionally independent. The dependent variable (output variable) is *Enrolls*.

Compute the prior probabilities $P(c_i)$ for *Enrolls*, where $c_i \in C$ and $C = \{Yes, No\}$.

```
tprior <- table(traindata$Enrolls)
tprior
     No Yes
  0   5   9

tprior <- tprior/sum(tprior)
tprior
                 No       Yes
0.0000000 0.3571429 0.6428571
```

The next step is to compute conditional probabilities $P(A|C)$, where $A = \{\text{Age, Income, JobSatisfaction, Desire}\}$ and $C = \{Yes, No\}$. Count the number of "No" and "Yes" entries for each *Age* group, and normalize by the total number of "No" and "Yes" entries to get the conditional probabilities.

```
ageCounts <- table(traindata[,c("Enrolls", "Age")])
ageCounts
       Age
Enrolls <=30 >40 31 to 40
           0   0        0
    No     3   2        0
   Yes     2   3        4


ageCounts <- ageCounts/rowSums(ageCounts)
ageCounts
       Age
Enrolls      <=30       >40  31 to 40

    No  0.6000000 0.4000000 0.0000000
    Yes 0.2222222 0.3333333 0.4444444
```

Do the same for the other attributes including *Income*, *JobSatisfaction*, and *Desire*.

```
incomeCounts <- table(traindata[,c("Enrolls", "Income")])
incomeCounts <- incomeCounts/rowSums(incomeCounts)
incomeCounts
       Income
Enrolls      High       Low    Medium

    No  0.4000000 0.2000000 0.4000000
    Yes 0.2222222 0.3333333 0.4444444

jsCounts <- table(traindata[,c("Enrolls", "JobSatisfaction")])
jsCounts <- jsCounts/rowSums(jsCounts)
```

```
jsCounts
       Jobsatisfaction
Enrolls        No       Yes

    No  0.8000000 0.2000000
    Yes 0.3333333 0.6666667

desireCounts <- table(traindata[,c("Enrolls", "Desire")])
desireCounts <- desireCounts/rowSums(desireCounts)
desireCounts
       Desire
Enrolls Excellent      Fair

    No  0.6000000 0.4000000
    Yes 0.3333333 0.6666667
```

According to Equation 7-7, probability $P(c_i|A)$ is determined by the product of $P(a_j|c_i)$ times the (c_i) where $c_1 =$ Yes and $c_2 =$ No. The larger value of $P(\text{Yes}|A)$ and $P(\text{No}|A)$ determines the predicted result of the output variable. Given the test data, use the following code to predict the `Enrolls`.

```
prob_yes <-
         ageCounts["Yes",testdata[,c("Age")]]*
         incomeCounts["Yes",testdata[,c("Income")]]*
         jsCounts["Yes",testdata[,c("JobSatisfaction")]]*
         desireCounts["Yes",testdata[,c("Desire")]]*
         tprior["Yes"]

prob_no <-
         ageCounts["No",testdata[,c("Age")]]*
         incomeCounts["No",testdata[,c("Income")]]*
         jsCounts["No",testdata[,c("JobSatisfaction")]]*
         desireCounts["No",testdata[,c("Desire")]]*
         tprior["No"]

max(prob_yes,prob_no)
```

As shown below, the predicted result of the test set is `Enrolls=Yes`.

```
prob_yes
       Yes
0.02821869

prob_no
         No
0.006857143

max(prob_yes, prob_no)
[1] 0.02821869
```

The *e1071* package in R has a built-in *naiveBayes* function that can compute the conditional probabilities of a categorical class variable given independent categorical predictor variables using the Bayes rule. The function takes the form of `naiveBayes(formula, data,...)`, where the arguments are defined as follows.

- **formula:** A formula of the form `class ~ x1 + x2 + ...` assuming x1, x2... are conditionally independent
- **data:** A data frame of factors

Use the following code snippet to execute the model and display the results.

```
model <- naiveBayes(Enrolls ~ Age+Income+JobSatisfaction+Desire,
                    traindata)
# display model
model
```

The output that follows shows that the probabilities of *model* match the probabilities from the previous method. The default `laplace=laplace` setting enables the Laplace smoothing.

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y                 No       Yes
0.0000000 0.3571429 0.6428571

Conditional probabilities:
     Age
Y          <=30       >40  31 to 40

  No  0.6000000 0.4000000 0.0000000
  Yes 0.2222222 0.3333333 0.4444444

     Income
Y          High       Low    Medium

  No  0.4000000 0.2000000 0.4000000
  Yes 0.2222222 0.3333333 0.4444444

     JobSatisfaction
Y            No       Yes

  No  0.8000000 0.2000000
  Yes 0.3333333 0.6666667
```

```
     Desire
Y     Excellent      Fair

  No  0.6000000 0.4000000
  Yes 0.3333333 0.6666667
```

Next, predicting the outcome of *Enrolls* with the *testdata* shows the result is *Enrolls*=Yes.

```
# predict with testdata
results <- predict (model,testdata)
# display results
results
[1] Yes
Levels:  No Yes
```

The *naiveBayes* function accepts a Laplace parameter that allows the customization of the ε value of Equation 7-17 for the Laplace smoothing. The code that follows shows how to build a naïve Bayes classifier with Laplace smoothing $\varepsilon = 0.01$ for prediction.

```
# use the NB classifier with Laplace smoothing
model1 = naiveBayes(Enrolls ~., traindata, laplace=.01)


# display model
model1
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
                  No       Yes
0.0000000 0.3571429 0.6428571

Conditional probabilities:
     Age
Y            <=30         >40    31 to 40
      0.333333333 0.333333333 0.333333333
  No  0.598409543 0.399602386 0.001988072
  Yes 0.222591362 0.333333333 0.444075305

     Income
Y          High       Low    Medium
      0.3333333 0.3333333 0.3333333
  No  0.3996024 0.2007952 0.3996024
  Yes 0.2225914 0.3333333 0.4440753
```

```
     JobSatisfaction
Y            No       Yes
      0.5000000 0.5000000
  No  0.7988048 0.2011952
  Yes 0.3337029 0.6662971

     Desire
Y     Excellent      Fair
      0.5000000 0.5000000
  No  0.5996016 0.4003984
  Yes 0.3337029 0.6662971
```

The test case is again classified as *Enrolls*=Yes.

```
# predict with testdata
results1 <- predict (model1,testdata)


# display results
results1
[1] Yes
Levels:  No Yes
```

7.3 Diagnostics of Classifiers

So far, this book has talked about three classifiers: logistic regression, decision trees, and naïve Bayes. These three methods can be used to classify instances into distinct groups according to the similar characteristics they share. Each of these classifiers faces the same issue: how to evaluate if they perform well.

A few tools have been designed to evaluate the performance of a classifier. Such tools are not limited to the three classifiers in this book but rather serve the purpose of assessing classifiers in general.

A ***confusion matrix*** is a specific table layout that allows visualization of the performance of a classifier.

Table 7-6 shows the confusion matrix for a two-class classifier. ***True positives*** (TP) are the number of positive instances the classifier correctly identified as positive. ***False positives*** (FP) are the number of instances in which the classifier identified as positive but in reality are negative. ***True negatives*** (TN) are the number of negative instances the classifier correctly identified as negative. ***False negatives*** (FN) are the number of instances classified as negative but in reality are positive. In a two-class classification, a preset threshold may be used to separate positives from negatives. TP and TN are the correct guesses. A good classifier should have large TP and TN and small (ideally zero) numbers for FP and FN.

TABLE 7-6 *Confusion Matrix*

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

In the bank marketing example, the training set includes 2,000 instances. An additional 100 instances are included as the testing set. Table 7-7 shows the confusion matrix of a naïve Bayes classifier on 100 clients to predict whether they would subscribe to the term deposit. Of the 11 clients who subscribed to the term deposit, the model predicted 3 subscribed and 8 not subscribed. Similarly, of the 89 clients who did not subscribe to the term, the model predicted 2 subscribed and 87 not subscribed. All correct guesses are located from top left to bottom right of the table. It's easy to visually inspect the table for errors, because they will be represented by any nonzero values outside the diagonal.

TABLE 7-7 *Confusion Matrix of Naïve Bayes from the Bank Marketing Example*

		Predicted Class		
		Subscribe	Not Subscribed	**Total**
Actual Class	Subscribed	3	8	11
	Not Subscribed	2	87	89
Total		5	95	100

The ***accuracy*** (or the ***overall success rate***) is a metric defining the rate at which a model has classified the records correctly. It is defined as the sum of TP and TN divided by the total number of instances, as shown in Equation 7-18.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{7-18}$$

A good model should have a high accuracy score, but having a high accuracy score alone does not guarantee the model is well established. The following measures can be introduced to better evaluate the performance of a classifier.

As seen in Chapter 6, the ***true positive rate*** (TPR) shows what percent of positive instances the classifier correctly identified. It's also illustrated in Equation 7-19.

$$TPR = \frac{TP}{TP + FN} \tag{7-19}$$

The ***false positive rate*** (FPR) shows what percent of negatives the classifier marked as positive. The FPR is also called the ***false alarm rate*** or the ***type I error rate*** and is shown in Equation 7-20.

$$FPR = \frac{FP}{FP + TN} \tag{7-20}$$

The ***false negative rate*** (FNR) shows what percent of positives the classifier marked as negatives. It is also known as the ***miss rate*** or ***type II error rate*** and is shown in Equation 7-21. Note that the sum of TPR and FNR is 1.

$$FNR = \frac{FN}{TP + FN} \tag{7-21}$$

A well-performed model should have a high TPR that is ideally 1 and a low FPR and FNR that are ideally 0. In reality, it's rare to have TPR = 1, FPR = 0, and FNR = 0, but these measures are useful to compare the performance of multiple models that are designed for solving the same problem. Note that in general, the model that is more preferable may depend on the business situation. During the discovery phase of the data analytics lifecycle, the team should have learned from the business what kind of errors can be tolerated. Some business situations are more tolerant of type I errors, whereas others may be more tolerant of type II errors. In some cases, a model with a TPR of 0.95 and an FPR of 0.3 is more acceptable than a model with a TPR of 0.9 and an FPR of 0.1 even if the second model is more accurate overall. Consider the case of e-mail spam filtering. Some people (such as busy executives) only want important e-mail in their inbox and are tolerant of having some less important e-mail end up in their spam folder as long as no spam is in their inbox. Other people may not want any important or less important e-mail to be specified as spam and are willing to have some spam in their inboxes as long as no important e-mail makes it into the spam folder.

Precision and recall are accuracy metrics used by the information retrieval community, but they can be used to characterize classifiers in general. ***Precision*** is the percentage of instances marked positive that really are positive, as shown in Equation 7-22.

$$Precision = \frac{TP}{TP + FP} \tag{7-22}$$

Recall is the percentage of positive instances that were correctly identified. Recall is equivalent to the TPR. Chapter 9, "Advanced Analytical Theory and Methods: Text Analysis," discusses how to use precision and recall for evaluation of classifiers in the context of text analysis.

Given the confusion matrix from Table 7-7, the metrics can be calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% = \frac{3 + 87}{3 + 87 + 2 + 8} \times 100\% = 90\%$$

$$TPR\ (or\ Recall) = \frac{TP}{TP + FN} = \frac{3}{3 + 8} \approx 0.273$$

$$FPR = \frac{FP}{FP + TN} = \frac{2}{2 + 87} \approx 0.022$$

$$FNR = \frac{FN}{TP + FN} = \frac{8}{3 + 8} \approx 0.727$$

$$Precision = \frac{TP}{TP + FP} = \frac{3}{3 + 2} = 0.6$$

These metrics show that for the bank marketing example, the naïve Bayes classifier performs well with accuracy and FPR measures and relatively well on precision. However, it performs poorly on TPR and FNR. To improve the performance, try to include more attributes in the datasets to better distinguish the characteristics of the records. There are other ways to evaluate the performance of a classifier in general, such as *N*-fold cross validation (Chapter 6) or bootstrap [14].

Chapter 6 has introduced the ***ROC curve,*** which is a common tool to evaluate classifiers. The abbreviation stands for ***receiver operating characteristic***, a term used in signal detection to characterize the trade-off between hit rate and false-alarm rate over a noisy channel. A ROC curve evaluates the performance of a classifier based on the TP and FP, regardless of other factors such as class distribution and error costs. The vertical axis is the True Positive Rate (TPR), and the horizontal axis is the False Positive Rate (FPR).

As seen in Chapter 6, any classifier can achieve the bottom left of the graph where TPR = FPR = 0 by classifying everything as negative. Similarly, any classifier can achieve the top right of the graph where TPR = FPR = 1 by classifying everything as positive. If a classifier performs "at chance" by random guessing the results, it can achieve any point on the diagonal line TPR=FPR by choosing an appropriate threshold of positive/negative. An ideal classifier should perfectly separate positives from negatives and thus achieve the top-left corner (TPR = 1, FPR = 0). The ROC curve of such classifiers goes straight up from TPR = FPR = 0 to the top-left corner and moves straight right to the top-right corner. In reality, it can be difficult to achieve the top-left corner. But a better classifier should be closer to the top left, separating it from other classifiers that are closer to the diagonal line.

Related to the ROC curve is the ***area under the curve*** (AUC). The AUC is calculated by measuring the area under the ROC curve. Higher AUC scores mean the classifier performs better. The score can range from 0.5 (for the diagonal line TPR=FPR) to 1.0 (with ROC passing through the top-left corner).

In the bank marketing example, the training set includes 2,000 instances. An additional 100 instances are included as the testing set. Figure 7-10 shows a ROC curve of the naïve Bayes classifier built on the training set of 2,000 instances and tested on the testing set of 100 instances. The figure is generated by the following R script. The `ROCR` package is required for plotting the ROC curve. The 2,000 instances are in a data frame called *`banktrain`*, and the additional 100 instances are in a data frame called *`banktest`*.

```
library(ROCR)

# training set
banktrain <- read.table("bank-sample.csv",header=TRUE,sep=",")
# drop a few columns
drops <- c("balance", "day", "campaign", "pdays", "previous", "month")
banktrain <- banktrain [,!(names(banktrain) %in% drops)]

# testing set
banktest <- read.table("bank-sample-test.csv",header=TRUE,sep=",")
banktest <- banktest [,!(names(banktest) %in% drops)]

# build the naïve Bayes classifier
nb_model <- naiveBayes(subscribed~.,
                       data=banktrain)
# perform on the testing set
nb_prediction <- predict(nb_model,
                         # remove column "subscribed"
                         banktest[,-ncol(banktest)],
                         type='raw')
score <- nb_prediction[, c("yes")]

actual_class <- banktest$subscribed == 'yes'

pred <- prediction(score, actual_class)
```

```
perf <- performance(pred, "tpr", "fpr")

plot(perf, lwd=2, xlab="False Positive Rate (FPR)",
     ylab="True Positive Rate (TPR)")
abline(a=0, b=1, col="gray50", lty=3)
```

The following R code shows that the corresponding AUC score of the ROC curve is about 0.915.

```
auc <- performance(pred, "auc")
auc <- unlist(slot(auc, "y.values"))
auc
[1] 0.9152196
```

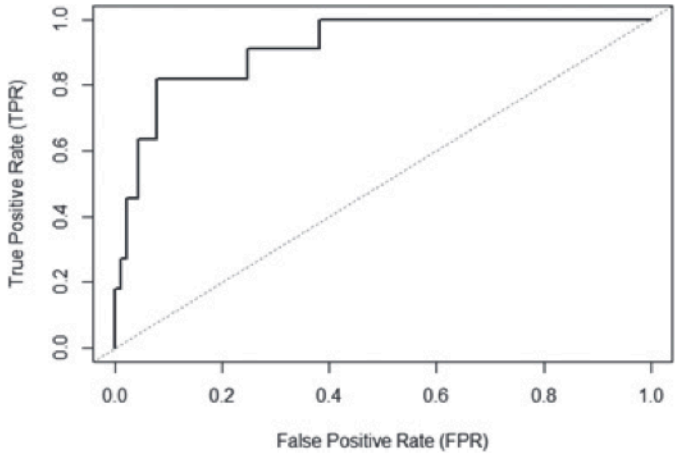


FIGURE 7-10 *ROC curve of the naïve Bayes classifier on the bank marketing dataset*

7.4 Additional Classification Methods

Besides the two classifiers introduced in this chapter, several other methods are commonly used for classification, including bagging [15], boosting [5], random forest [4], and support vector machines (SVM) [16]. Bagging, boosting, and random forest are all examples of ensemble methods that use multiple models to obtain better predictive performance than can be obtained from any of the constituent models.

Bagging (or bootstrap aggregating) [15] uses the bootstrap technique that repeatedly samples with replacement from a dataset according to a uniform probability distribution. "With replacement" means that when a sample is selected for a training or testing set, the sample is still kept in the dataset and may be selected again. Because the sampling is with replacement, some samples may appear several times in a training or testing set, whereas others may be absent. A model or base classifier is trained separately on each bootstrap sample, and a test sample is assigned to the class that received the highest number of votes.

Similar to bagging, boosting (or AdaBoost) [17] uses votes for classification to combine the output of individual models. In addition, it combines models of the same type. However, boosting is an iterative procedure

where a new model is influenced by the performances of those models built previously. Furthermore, boosting assigns a weight to each training sample that reflects its importance, and the weight may adaptively change at the end of each boosting round. Bagging and boosting have been shown to have better performances [5] than a decision tree.

Random forest [4] is a class of ensemble methods using decision tree classifiers. It is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. A special case of random forest uses bagging on decision trees, where samples are randomly chosen with replacement from the original training set.

SVM [16] is another common classification method that combines linear models with instance-based learning techniques. Support vector machines select a small number of critical boundary instances called support vectors from each class and build a linear decision function that separates them as widely as possible. SVM by default can efficiently perform linear classifications and can be configured to perform nonlinear classifications as well.

Summary

This chapter focused on two classification methods: decision trees and naïve Bayes. It discussed the theory behind these classifiers and used a bank marketing example to explain how the methods work in practice. These classifiers along with logistic regression (Chapter 6) are often used for the classification of data. As this book has discussed, each of these methods has its own advantages and disadvantages. How does one pick the most suitable method for a given classification problem? Table 7-8 offers a list of things to consider when selecting a classifier.

TABLE 7-8 *Choosing a Suitable Classifier*

Concerns	Recommended Method(s)
Output of the classification should include class probabilities in addition to the class labels.	Logistic regression, decision tree
Analysts want to gain an insight into how the variables affect the model.	Logistic regression, decision tree
The problem is high dimensional.	Naïve Bayes
Some of the input variables might be correlated.	Logistic regression, decision tree
Some of the input variables might be irrelevant.	Decision tree, naïve Bayes
The data contains categorical variables with a large number of levels.	Decision tree, naïve Bayes
The data contains mixed variable types.	Logistic regression, decision tree
There is nonlinear data or discontinuities in the input variables that would affect the output.	Decision tree

After the classification, one can use a few evaluation tools to measure how well a classifier has performed or compare the performances of multiple classifiers. These tools include confusion matrix, TPR, FPR, FNR, precision, recall, ROC curves, and AUC.

In addition to the decision trees and naïve Bayes, other methods are commonly used as classifiers. These methods include but are not limited to bagging, boosting, random forest, and SVM.

Exercises

1. For a binary classification, describe the possible values of entropy. On what conditions does entropy reach its minimum and maximum values?
2. In a decision tree, how does the algorithm pick the attributes for splitting?
3. John went to see the doctor about a severe headache. The doctor selected John at random to have a blood test for swine flu, which is suspected to affect 1 in 5,000 people in this country. The test is 99% accurate, in the sense that the probability of a false positive is 1%. The probability of a false negative is zero. John's test came back positive. What is the probability that John has swine flu?
4. Which classifier is considered computationally efficient for high-dimensional problems? Why?
5. A data science team is working on a classification problem in which the dataset contains many correlated variables, and most of them are categorical variables. Which classifier should the team consider using? Why?
6. A data science team is working on a classification problem in which the dataset contains many correlated variables, and most of them are continuous. The team wants the model to output the probabilities in addition to the class labels. Which classifier should the team consider using? Why?
7. Consider the following confusion matrix:

		Predicted Class		
		Good	Bad	Total
Actual Class	Good	671	29	300
	Bad	38	262	700
Total		709	291	1000

What are the true positive rate, false positive rate, and false negative rate?

Bibliography

[1] M. Thomas, B. Pang, and L. Lee, "Get Out the Vote: Determining Support or Opposition from Congressional Floor-Debate Transcripts," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 2006.

[2] M. Shouman, T. Turner, and R. Stocker, "Using Decision Tree for Diagnosing Heart Disease Patients," in *Australian Computer Society, Inc.*, Ballarat, Australia, in *Proceedings of the Ninth Australasian Data Mining Conference (AusDM '11).*

[3] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An Evaluation of Naïve Bayesian Anti-Spam Filtering," in *Proceedings of the Workshop on Machine Learning in the New Information Age,* Barcelona, Spain, 2000.

[4] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5–32, 2001.

[5] J. R. Quinlan, "Bagging, Boosting, and C4. 5," *AAAI/IAAI,* vol. 1, 1996.

[6] S. Moro, P. Cortez, and R. Laureano, "Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology," in *Proceedings of the European Simulation and Modelling Conference - ESM'2011,* Guimaraes, Portugal, 2011.

[7] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81–106, 1986.

[8] J. R. Quinlan, *C4. 5: Programs for Machine Learning,* Morgan Kaufmann, 1993.

[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees,* Belmont, CA: Wadsworth International Group, 1984.

[10] T. M. Mitchell, "Decision Tree Learning," in *Machine Learning,* New York, NY, USA, McGraw-Hill, Inc., 1997, p. 68.

[11] C. Phua, V. C. S. Lee, S. Kate, and R. W. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection," *CoRR,* vol. abs/1009.6119, 2010.

[12] R. Bhowmik, "Detecting Auto Insurance Fraud by Data Mining Techniques," *Journal of Emerging Trends in Computing and Information Sciences,* vol. 2, no. 4, pp. 156–162, 2011.

[13] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification,* New York: Ellis Horwood, 1994.

[14] I. H. Witten, E. Frank, and M. A. Hall, "The Bootstrap," in *Data Mining,* Burlington, Massachusetts, Morgan Kaufmann, 2011, pp. 155–156.

[15] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123–140, 1996.

[16] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods,* Cambridge, United Kingdom: Cambridge university press, 2000.

[17] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences,* vol. 55, no. 1, pp. 119–139, 1997.

8

Advanced Analytical Theory and Methods: Time Series Analysis

Key Concepts

ACF
ARIMA
Autoregressive
Moving average
PACF
Stationary
Time series

This chapter examines the topic of time series analysis and its applications. Emphasis is placed on identifying the underlying structure of the time series and fitting an appropriate Autoregressive Integrated Moving Average (ARIMA) model.

8.1 Overview of Time Series Analysis

Time series analysis attempts to model the underlying structure of observations taken over time. A ***time series***, denoted $Y = a + bX$, is an ordered sequence of equally spaced values over time. For example, Figure 8-1 provides a plot of the monthly number of international airline passengers over a 12-year period.

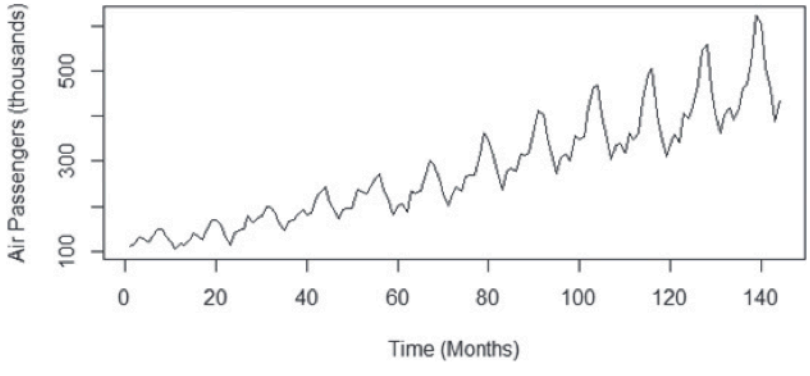


FIGURE 8-1 *Monthly international airline passengers*

In this example, the time series consists of an ordered sequence of 144 values. The analyses presented in this chapter are limited to equally spaced time series of one variable. Following are the goals of time series analysis:

- Identify and model the structure of the time series.
- Forecast future values in the time series.

Time series analysis has many applications in finance, economics, biology, engineering, retail, and manufacturing. Here are a few specific use cases:

- **Retail sales:** For various product lines, a clothing retailer is looking to forecast future monthly sales. These forecasts need to account for the seasonal aspects of the customer's purchasing decisions. For example, in the northern hemisphere, sweater sales are typically brisk in the fall season, and swimsuit sales are the highest during the late spring and early summer. Thus, an appropriate time series model needs to account for fluctuating demand over the calendar year.
- **Spare parts planning:** Companies' service organizations have to forecast future spare part demands to ensure an adequate supply of parts to repair customer products. Often the spares inventory consists of thousands of distinct part numbers. To forecast future demand, complex models for each part number can be built using input variables such as expected part failure rates, service diagnostic effectiveness, forecasted new product shipments, and forecasted trade-ins/decommissions.

However, time series analysis can provide accurate short-term forecasts based simply on prior spare part demand history.

- **Stock trading:** Some high-frequency stock traders utilize a technique called ***pairs trading***. In pairs trading, an identified strong positive correlation between the prices of two stocks is used to detect a market opportunity. Suppose the stock prices of Company A and Company B consistently move together. Time series analysis can be applied to the difference of these companies' stock prices over time. A statistically larger than expected price difference indicates that it is a good time to buy the stock of Company A and sell the stock of Company B, or vice versa. Of course, this trading approach depends on the ability to execute the trade quickly and be able to detect when the correlation in the stock prices is broken. Pairs trading is one of many techniques that falls into a trading strategy called ***statistical arbitrage***.

8.1.1 Box-Jenkins Methodology

In this chapter, a time series consists of an ordered sequence of equally spaced values over time. Examples of a time series are monthly unemployment rates, daily website visits, or stock prices every second. A time series can consist of the following components:

- Trend
- Seasonality
- Cyclic
- Random

The ***trend*** refers to the long-term movement in a time series. It indicates whether the observation values are increasing or decreasing over time. Examples of trends are a steady increase in sales month over month or an annual decline of fatalities due to car accidents.

The ***seasonality*** component describes the fixed, periodic fluctuation in the observations over time. As the name suggests, the seasonality component is often related to the calendar. For example, monthly retail sales can fluctuate over the year due to the weather and holidays.

A ***cyclic*** component also refers to a periodic fluctuation, but one that is not as fixed as in the case of a seasonality component. For example, retails sales are influenced by the general state of the economy. Thus, a retail sales time series can often follow the lengthy boom-bust cycles of the economy.

After accounting for the other three components, the ***random*** component is what remains. Although noise is certainly part of this random component, there is often some underlying structure to this random component that needs to be modeled to forecast future values of a given time series.

Developed by George Box and Gwilym Jenkins, the Box-Jenkins methodology for time series analysis involves the following three main steps:

1. Condition data and select a model.
   - Identify and account for any trends or seasonality in the time series.
   - Examine the remaining time series and determine a suitable model.
2. Estimate the model parameters.
3. Assess the model and return to Step 1, if necessary.

The primary focus of this chapter is to use the Box-Jenkins methodology to apply an ARIMA model to a given time series.

8.2 ARIMA Model

To fully explain an ARIMA (Autoregressive Integrated Moving Average) model, this section describes the model's various parts and how they are combined. As stated in the first step of the Box-Jenkins methodology, it is necessary to remove any trends or seasonality in the time series. This step is necessary to achieve a time series with certain properties to which autoregressive and moving average models can be applied. Such a time series is known as a stationary time series. A time series, y_t *for* $t = 1, 2, 3, \ldots$, is a ***stationary time series*** if the following three conditions are met:

- (a) The expected value (mean) of y_t is a constant for all values of t.
- (b) The variance of y_t is finite.
- (c) The covariance of y_t *and* y_{t+h} depends only on the value of $h = 0, 1, 2, \ldots$ for all t.

The covariance of y_t *and* y_{t+h} is a measure of how the two variables, y_t *and* y_{t+h}, vary together. It is expressed in Equation 8-1.

$$cov\left(y_t, y_{t+h}\right) = E[(y_t - \mu_t)(y_{t+h} - \mu_{t+h})] \tag{8-1}$$

If two variables are independent of each other, their covariance is zero. If the variables change together in the same direction, the variables have a positive covariance. Conversely, if the variables change together in the opposite direction, the variables have a negative covariance.

For a stationary time series, by condition (a), the mean is a constant, say μ. So, for a given stationary sequence, y_t, the covariance notation can be simplified to what's shown in Equation 8-2.

$$cov\left(h\right) = E\left[\left(y_t - \mu\right)\left(y_{t+h} - \mu\right)\right] \tag{8-2}$$

By part (c), the covariance between two points in the time series can be nonzero, as long as the value of the covariance is only a function of h. Equation 8-3 is an example for $h = 3$.

$$cov\,(3) = cov\,(y_1, y_4) = cov\,(y_2, y_5) = \ldots \tag{8-3}$$

It is important to note that for $h = 0$, the $cov\,(0) = cov(y_t, y_t) = var(y_t)$ for all t. Because the $var\,(y_t) < \infty$, by condition (b), the variance of y_t is a constant for all t. So the constant variance coupled with part (a), $E\,[y_t] = \mu$, for all t and some constant μ, suggests that a stationary time series can look like Figure 8-2. In this plot, the points appear to be centered about a fixed constant, zero, and the variance appears to be somewhat constant over time.

8.2.1 Autocorrelation Function (ACF)

Although there is not an overall trend in the time series plotted in Figure 8-2, it appears that each point is somewhat dependent on the past points. The difficulty is that the plot does not provide insight into the covariance of the variables in the time series and its underlying structure. The plot of ***autocorrelation function (ACF)*** provides this insight. For a stationary time series, the ACF is defined as shown in Equation 8-4.

$$ACF(h) = \frac{cov(y_t, y_{t+h})}{\sqrt{cov(y_t, y_t)\, cov(y_{t+h}, y_{t+h})}} = \frac{cov(h)}{cov(0)} \tag{8-4}$$

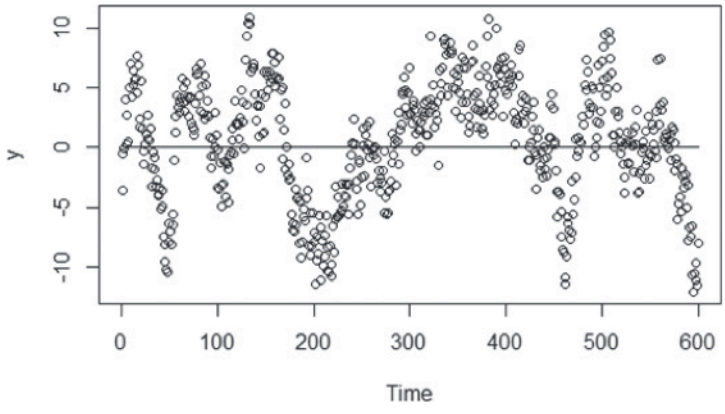


FIGURE 8-2 *A plot of a stationary series*

Because the cov(0) is the variance, the ACF is analogous to the correlation function of two variables, $corr(y_t, y_{t+h})$, and the value of the ACF falls between –1 and 1. Thus, the closer the absolute value of ACF(h) is to 1, the more useful y_t can be as a predictor of y_{t+h}.

Using the same dataset plotted in Figure 8-2, the plot of the ACF is provided in Figure 8-3.

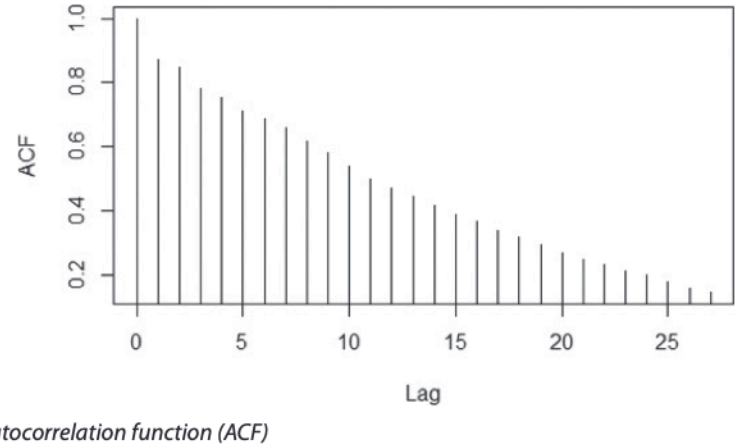


FIGURE 8-3 *Autocorrelation function (ACF)*

By convention, the quantity h in the ACF is referred to as the ***lag***, the difference between the time points t and $t + h$. At lag 0, the ACF provides the correlation of every point with itself. So ACF(0) always equals 1. According to the ACF plot, at lag 1 the correlation between y_t *and* y_{t-1} is approximately 0.9, which is very close to 1. So y_{t-1} appears to be a good predictor of the value of y_t. Because ACF(2) is around 0.8, y_{t-2} also appears to be a good predictor of the value of y_t. A similar argument could be made for lag 3 to lag 8. (All the autocorrelations are greater than 0.6.) In other words, a model can be considered that would express y_t as a linear sum of its previous 8 terms. Such a model is known as an autoregressive model of order 8.

8.2.2 Autoregressive Models

For a stationary time series, y_t $t = 1, 2, 3, \ldots$, an ***autoregressive model of order p***, denoted AR(p), is expressed as shown in Equation 8-5:

$$y_t = \delta + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \tag{8-5}$$

where δ is a constant for a nonzero-centered time series:
ϕ_j is a constant for j = 1, 2, . . ., p
y_{t-j} is the value of the time series at time $t - j$
$\phi_p \neq 0$
$\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ for all t

Thus, a particular point in the time series can be expressed as a linear combination of the prior p values, y_{t-j} for j = 1, 2, . . . p, of the time series plus a random error term, ε_t. In this definition, the ε_t time series is often called a ***white noise process*** and is used to represent random, independent fluctuations that are part of the time series.

From the earlier example in Figure 8-3, the autocorrelations are quite high for the first several lags. Although it appears that an AR(8) model might be a good candidate to consider for the given dataset, examining an AR(1) model provides further insight into the ACF and the appropriate value of p to choose. For an AR(1) model, centered around $\delta = 0$, Equation 8-5 simplifies to Equation 8-6.

$$y_t = \phi_1 y_{t-1} + \varepsilon_t \tag{8-6}$$

Based on Equation 8-6, it is evident that $y_{t-1} = \phi_1 y_{t-2} + \varepsilon_{t-1}$. Thus, substituting for y_{t-1} yields Equation 8-7.

$$\begin{aligned} y_t &= \phi_1(\phi y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t \\ &= \phi_1^2 y_{t-2} + \phi_1 \varepsilon_{t-1} + \varepsilon_t \end{aligned} \tag{8-7}$$

Therefore, in a time series that follows an AR(1) model, considerable autocorrelation is expected at lag 2. As this substitution process is repeated, y_t can be expressed as a function of y_{t-h} for $h = 3, 4$ ...and a sum of the error terms. This observation means that even in the simple AR(1) model, there will be considerable autocorrelation with the larger lags even though those lags are not explicitly included in the model. What is needed is a measure of the autocorrelation between y_t and y_{t+h} for h = 1, 2, 3... with the effect of the y_{t+1} to y_{t+h-1} values excluded from the measure. The ***partial autocorrelation function (PACF)*** provides such a measure and is expressed as shown in Equation 8-8.

$$\begin{aligned} PACF(h) &= corr(y_t - y_t^*, y_{t+h} - y_{t+h}^*) \text{ for } h \geq 2 \\ &= corr(y_t, y_{t+1}) \qquad\qquad \text{for } h = 1 \end{aligned} \tag{8-8}$$

where $y_t^* = \beta_1 y_{t+1} + \beta_2 y_{t+2} \cdots + \beta_{h-1} y_{t+h-1}$,
$y_{t+h}^* = \beta_1 y_{t+h-1} + \beta_2 y_{t+h-2} \cdots + \beta_{h-1} y_{t+1}$, and
the h − 1 values of the βs are based on linear regression .

In other words, after linear regression is used to remove the effect of the variables between y_t and y_{t+h} on y_t and y_{t+h}, the PACF is the correlation of what remains. For $h=1$, there are no variables between y_t and y_{t+1}. So the PACF(1) equals ACF(1). Although the computation of the PACF is somewhat complex, many software tools hide this complexity from the analyst.

For the earlier example, the PACF plot in Figure 8-4 illustrates that after lag 2, the value of the PACF is sharply reduced. Thus, after removing the effects of y_{t+1} and y_{t+2}, the partial correlation between y_t and y_{t+3} is relatively small. Similar observations can be made for h = 4, 5, …. Such a plot indicates that an AR(2) is a good candidate model for the time series plotted in Figure 8-2. In fact, the time series data for this example was randomly generated based on $y_t = 0.6y_{t-1} + 0.35y_{t-2} + \varepsilon_t$ where $\varepsilon_t \sim N(0, 4)$.

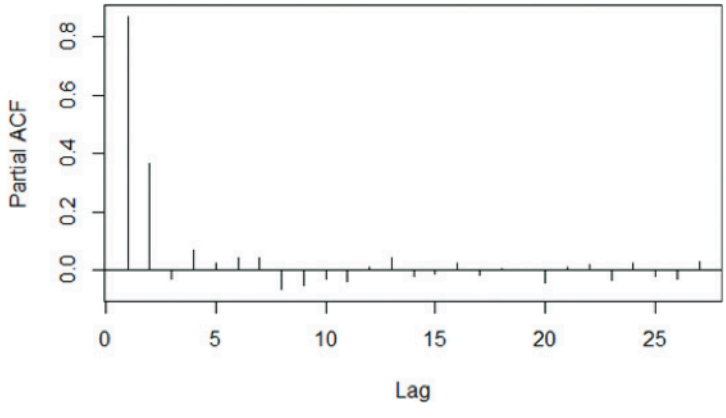


FIGURE 8-4 *Partial autocorrelation function (PACF) plot*

Because the ACF and PACF are based on correlations, negative and positive values are possible. Thus, the magnitudes of the functions at the various lags should be considered in terms of absolute values.

8.2.3 Moving Average Models

For a time series, y_t, centered at zero, a ***moving average model of order q***, denoted MA(q), is expressed as shown in Equation 8-9.

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \tag{8-9}$$

where θ_k is a constant for k = 1, 2, ..., q
$\theta_q \neq 0$
$\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ for all t

In an MA(q) model, the value of a time series is a linear combination of the current white noise term and the prior q white noise terms. So earlier random shocks directly affect the current value of the time series. For MA(q) models, the behavior of the ACF and PACF plots are somewhat swapped from the behavior of these plots for AR(p) models. For a simulated MA(3) time series of the form $y_t = \varepsilon_t - 0.4\varepsilon_{t-1} + 1.1\varepsilon_{t-2} - 2.5\varepsilon_{t-3}$ where $\varepsilon_t \sim N(0, 1)$, Figure 8-5 provides the scatterplot of the simulated data over time.

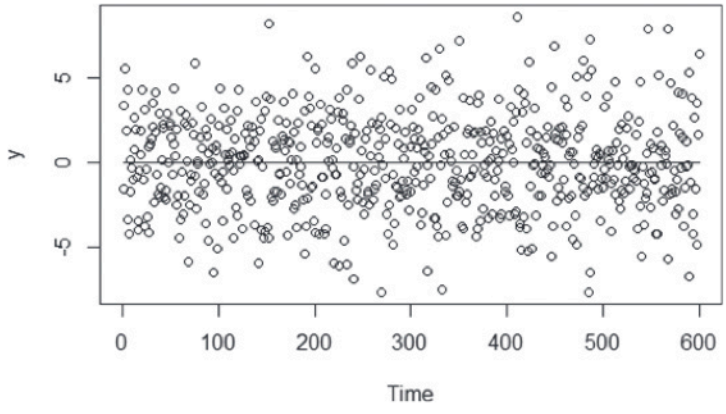


FIGURE 8-5 *Scatterplot of a simulated MA(3) time series*

Figure 8-6 provides the ACF plot for the simulated data. Again, the ACF(0) equals 1, because any variable is perfectly correlated with itself. At lags 1, 2, and 3, the value of the ACF is relatively large in absolute value compared to the subsequent terms. In an autoregressive model, the ACF slowly decays, but for an MA(3) model, the ACF somewhat abruptly cuts off after lag 3. In general, this pattern can be extended to any MA(q) model.

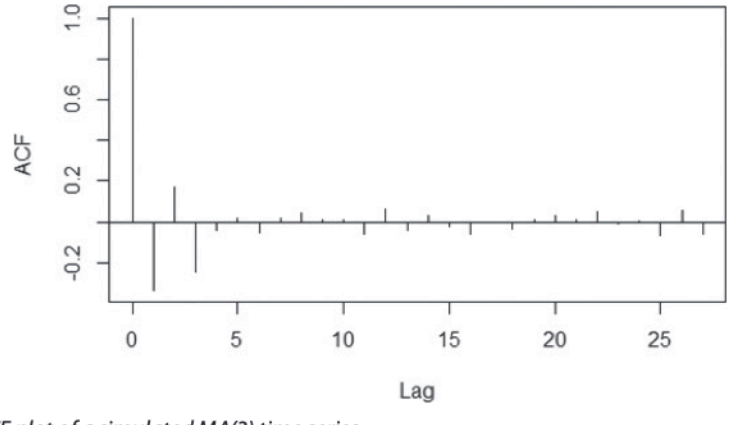


FIGURE 8-6 *ACF plot of a simulated MA(3) time series*

To understand why this phenomenon occurs, it is useful to examine Equations 8-10 through 8-14 for an MA(3) time series model:

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3} \tag{8-10}$$

$$y_{t-1} = \varepsilon_{t-1} + \theta_1 \varepsilon_{t-2} + \theta_2 \varepsilon_{t-3} + \theta_3 \varepsilon_{t-4} \tag{8-11}$$

$$y_{t-2} = \varepsilon_{t-2} + \theta_1 \varepsilon_{t-3} + \theta_2 \varepsilon_{t-4} + \theta_3 \varepsilon_t \tag{8-12}$$

$$y_{t-3} = \varepsilon_{t-3} + \theta_1 \varepsilon_{t-4} + \theta_2 \varepsilon_{t-5} + \theta_3 \varepsilon \tag{8-13}$$

$$y_{t-4} = \varepsilon_{t-4} + \theta_1 \varepsilon_{t-5} + \theta_2 \varepsilon_{t-6} + \theta_3 \tag{8-14}$$

Because the expression of y_t shares specific white noise variables with the expressions for y_{t-1} through y_{t-3}, inclusive, those three variables are correlated to y_t. However, the expression of y_t in Equation 8-10 does not share white noise variables with y_{t-4} in Equation 8-14. So the theoretical correlation between y_t and y_{t-4} is zero. Of course, when dealing with a particular dataset, the theoretical autocorrelations are unknown, but the observed autocorrelations should be close to zero for lags greater than q when working with an MA(q) model.

8.2.4 ARMA and ARIMA Models

In general, the data scientist does not have to choose between an AR(p) and an MA(q) model to describe a time series. In fact, it is often useful to combine these two representations into one model. The combination of these two models for a stationary time series results in an ***Autoregressive Moving Average model, ARMA(p,q)***, which is expressed as shown in Equation 8-15.

$$\begin{aligned} y_t = \delta &+ \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} \\ &+ \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} \end{aligned} \tag{8-15}$$

where δ is a constant for a nonzero-centered time series

ϕ_j is a constant for j = 1, 2, ..., p

$\phi_p \neq 0$

θ_k is a constant for k = 1, 2, ..., q

$\theta_q \neq 0$

$\varepsilon_t \sim N(0, \sigma_\varepsilon^2)$ for all t

If $p = 0$ and $q \neq 0$, then the ARMA(p,q) model is simply an AR(p) model. Similarly, if $p = 0$ and $q \neq 0$, then the ARMA(p,q) model is an MA(q) model.

To apply an ARMA model properly, the time series must be a stationary one. However, many time series exhibit some trend over time. Figure 8-7 illustrates a time series with an increasing linear trend over time. Since such a time series does not meet the requirement of a constant expected value (mean), the data needs to be adjusted to remove the trend. One transformation option is to perform a regression analysis on the time series and then to subtract the value of the fitted regression line from each observed y-value.

If detrending using a linear or higher order regression model does not provide a stationary series, a second option is to compute the difference between successive y-values. This is known as ***differencing***. In other words, for the n values in a given time series compute the differences as shown in Equation 8-16.

$$d_t = y_t - y_{t-1} \quad \text{for } t = 2, 3, \ldots, n \tag{8-16}$$

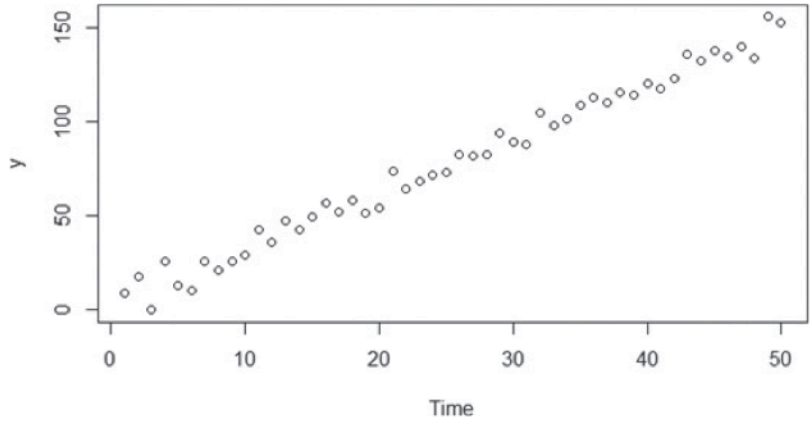


FIGURE 8-7 *A time series with a trend*

The mean of the time series plotted in Figure 8-8 is certainly not a constant. Applying differencing to the time series results in the plot in Figure 8-9. This plot illustrates a time series with a constant mean and a fairly constant variance over time.

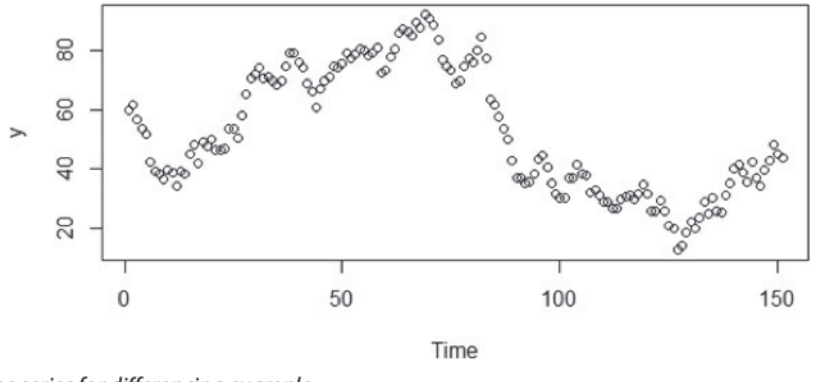


FIGURE 8-8 *Time series for differencing example*

If the differenced series is not reasonably stationary, applying differencing additional times may help. Equation 8-17 provides the twice differenced time series for $t = 3, 4, \ldots n$.

$$\begin{aligned} d_{t-1} - d_{t-2} &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned} \tag{8-17}$$

Successive differencing can be applied, but over-differencing should be avoided. One reason is that over-differencing may unnecessarily increase the variance. The increased variance can be detected by plotting the possibly over-differenced values and observing that the spread of the values is much larger, as seen in Figure 8-10 after differencing the values of y twice.

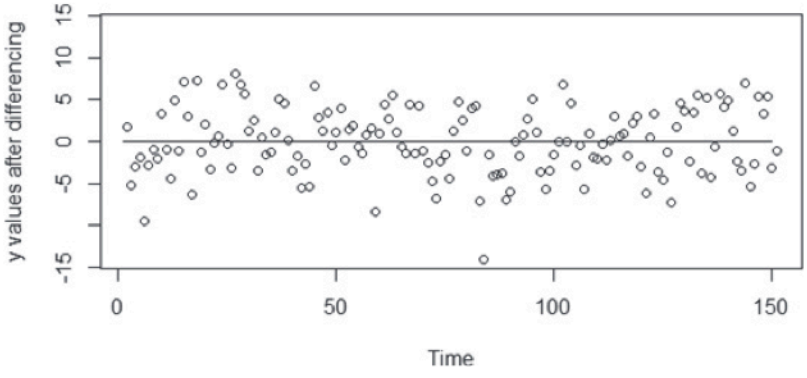


FIGURE 8-9 *Detrended time series using differencing*

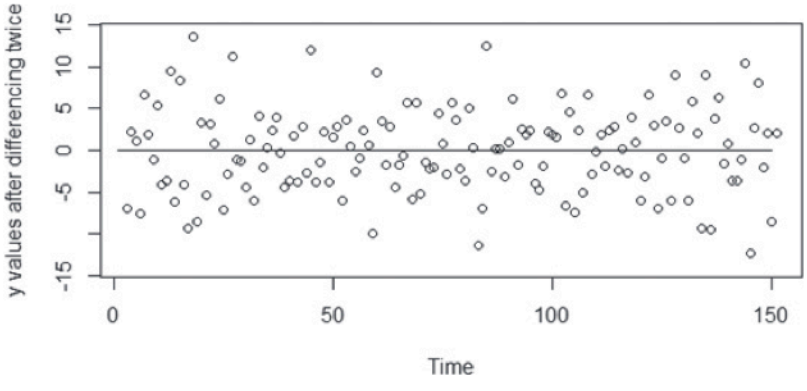


FIGURE 8-10 *Twice differenced series*

Because the need to make a time series stationary is common, the differencing can be included (integrated) into the ARMA model definition by defining the ***Autoregressive Integrated Moving Average*** model, denoted ARIMA(p,d,q). The structure of the ARIMA model is identical to the expression in Equation 8-15, but the ARMA(p,q) model is applied to the time series, y_t, after applying differencing d times.

Additionally, it is often necessary to account for seasonal patterns in time series. For example, in the retail sales use case example in Section 8.1, monthly clothing sales track closely with the calendar month. Similar to the earlier option of detrending a series by first applying linear regression, the seasonal pattern could be determined and the time series appropriately adjusted. An alternative is to use a ***seasonal autoregressive integrated moving average model***, denoted ARIMA(p,d,q) × $(P,D,Q)_s$ where:

- p, d, and q are the same as defined previously.
- s denotes the seasonal period.
- P is the number of terms in the AR model across the s periods.
- D is the number of differences applied across the s periods.
- Q is the number of terms in the MA model across the s periods.

For a time series with a seasonal pattern, following are typical values of s:

- 52 for weekly data
- 12 for monthly data
- 7 for daily data

The next section presents a seasonal ARIMA example and describes several techniques and approaches to identify the appropriate model and forecast the future.

8.2.5 Building and Evaluating an ARIMA Model

For a large country, the monthly gasoline production measured in millions of barrels has been obtained for the past 240 months (20 years). A market research firm requires some short-term gasoline production forecasts to assess the petroleum industry's ability to deliver future gasoline supplies and the effect on gasoline prices.

```
library(forecast)

# read in gasoline production time series
# monthly gas production expressed in millions of barrels
gas_prod_input <- as.data.frame( read.csv("c:/data/gas_prod.csv") )

# create a time series object
gas_prod <- ts(gas_prod_input[,2])

#examine the time series
plot(gas_prod, xlab = "Time (months)",
     ylab = "Gasoline production (millions of barrels)")
```

Using R, the dataset is plotted in Figure 8-11.

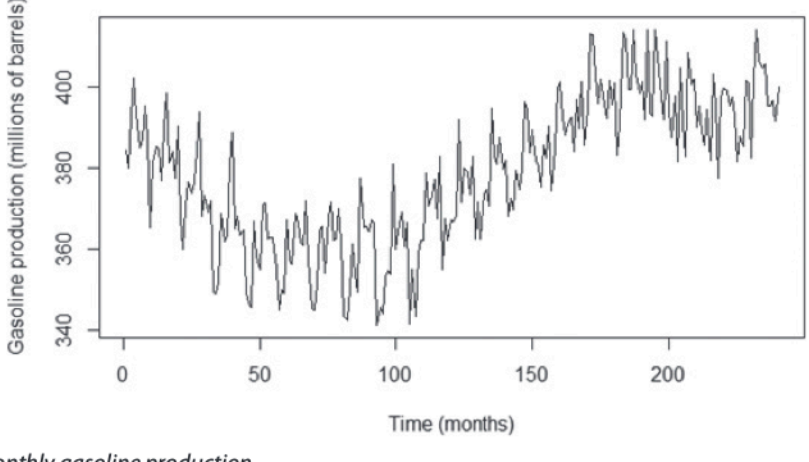


FIGURE 8-11 *Monthly gasoline production*

In R, the `ts()` function creates a time series object from a vector or a matrix. The use of time series objects in R simplifies the analysis by providing several methods that are tailored specifically for handling equally time spaced data series. For example, the `plot()` function does not require an explicitly specified variable for the x-axis.

To apply an ARMA model, the dataset needs to be a stationary time series. Using the `diff()` function, the gasoline production time series is differenced once and plotted in Figure 8-12.

```
plot(diff(gas_prod))
abline(a=0, b=0)
```

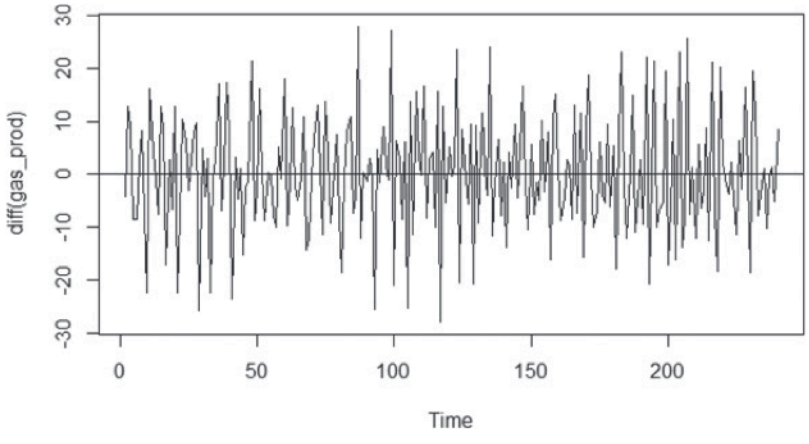


FIGURE 8-12 *Differenced gasoline production time series*

The differenced time series has a constant mean near zero with a fairly constant variance over time. Thus, a stationary time series has been obtained. Using the following R code, the ACF and PACF plots for the differenced series are provided in Figures 8-13 and 8-14, respectively.

```
# examine ACF and PACF of differenced series
acf(diff(gas_prod), xaxp = c(0, 48, 4), lag.max=48, main="")
pacf(diff(gas_prod), xaxp = c(0, 48, 4), lag.max=48, main="")
```

The dashed lines provide upper and lower bounds at a 95% significance level. Any value of the ACF or PACF outside of these bounds indicates that the value is significantly different from zero.

Figure 8-13 shows several significant ACF values. The slowly decaying ACF values at lags 12, 24, 36, and 48 are of particular interest. A similar behavior in the ACF was seen in Figure 8-3, but for lags 1, 2, 3,… Figure 8-13 indicates a seasonal autoregressive pattern every 12 months. Examining the PACF plot in Figure 8-14, the PACF value at lag 12 is quite large, but the PACF values are close to zero at lags 24, 36, and 48. Thus, a seasonal AR(1) model with period = 12 will be considered. It is often useful to address the seasonal portion of the overall ARMA model before addressing the nonseasonal portion of the model.

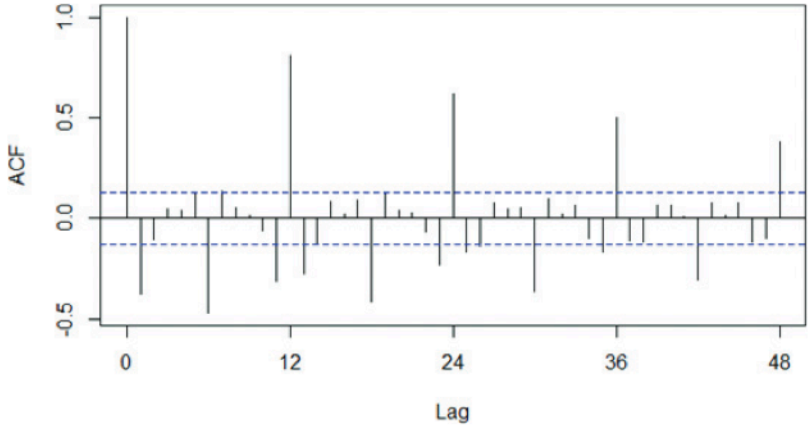


FIGURE 8-13 *ACF of the differenced gasoline time series*

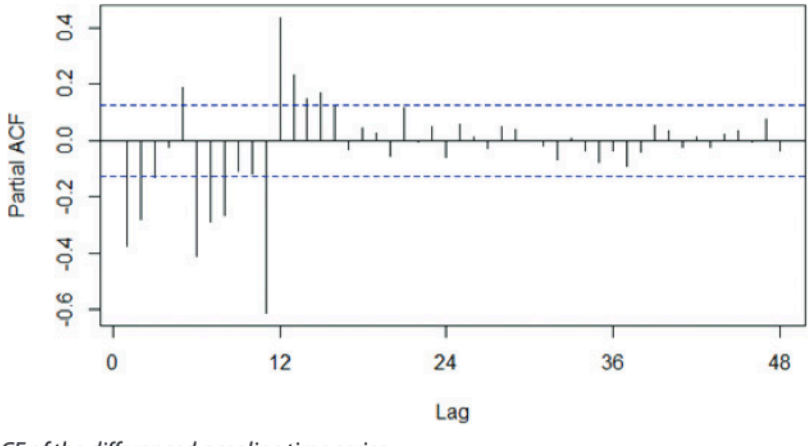


FIGURE 8-14 *PACF of the differenced gasoline time series*

The `arima()` function in R is used to fit a $(0,1,0) \times (1,0,0)_{12}$ model. The analysis is applied to the original time series variable, *gas_prod*. The differencing, d = 1, is specified by the order = c(0,1,0) term.

```
arima_1 <- arima (gas_prod,
                  order=c(0,1,0),
                  seasonal = list(order=c(1,0,0),period=12))
arima_1

Series: gas_prod
ARIMA(0,1,0)(1,0,0)[12]

Coefficients:
        sar1
      0.8335
```

```
s.e.  0.0324

sigma^2 estimated as 37.29:  log likelihood=-778.69
AIC=1561.38   AICc=1561.43   BIC=1568.33
```

The value of the coefficient for the seasonal AR(1) model is estimated to be 0.8335 with a standard error of 0.0324. Because the estimate is several standard errors away from zero, this coefficient is considered significant. The output from this first pass ARIMA analysis is stored in the variable `arima_1`, which contains several useful quantities including the residuals. The next step is to examine the residuals from fitting the $(0,1,0) \times (1,0,0)_{12}$ ARIMA model. The ACF and PACF plots of the residuals are provided in Figures 8-15 and 8-16, respectively.

```
# examine ACF and PACF of the (0,1,0)x(1,0,0)12 residuals
acf(arima_1$residuals, xaxp = c(0, 48, 4), lag.max=48, main="")
pacf(arima_1$residuals, xaxp = c(0, 48, 4), lag.max=48, main="")
```

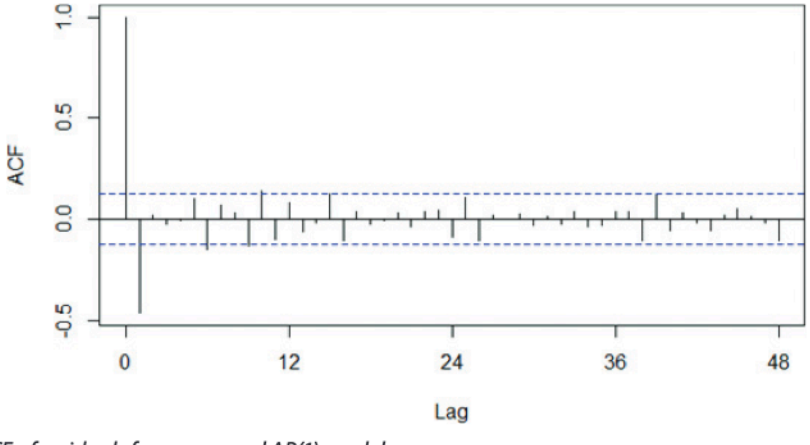


FIGURE 8-15 *ACF of residuals from seasonal AR(1) model*

The ACF plot of the residuals in Figure 8-15 indicates that the autoregressive behavior at lags 12, 24, 26, and 48 has been addressed by the seasonal AR(1) term. The only remaining ACF value of any significance occurs at lag 1. In Figure 8-16, there are several significant PACF values at lags 1, 2, 3, and 4.

Because the PACF plot in Figure 8-16 exhibits a slowly decaying PACF, and the ACF cuts off sharply at lag 1, an MA(1) model should be considered for the nonseasonal portion of the ARMA model on the differenced series. In other words, a $(0,1,1) \times (1,0,0)_{12}$ ARIMA model will be fitted to the original gasoline production time series.

```
arima_2 <- arima (gas_prod,
                  order=c(0,1,1),
                  seasonal = list(order=c(1,0,0),period=12))
arima_2

Series: gas_prod
ARIMA(0,1,1)(1,0,0)[12]

Coefficients:
          ma1     sar1
```

```
      -0.7065   0.8566
s.e.   0.0526   0.0298

sigma^2 estimated as 25.24:  log likelihood=-733.22
AIC=1472.43   AICc=1472.53   BIC=1482.86

acf(arima_2$residuals, xaxp = c(0, 48, 4), lag.max=48, main="")
pacf(arima_2$residuals, xaxp = c(0, 48,4), lag.max=48, main="")
```

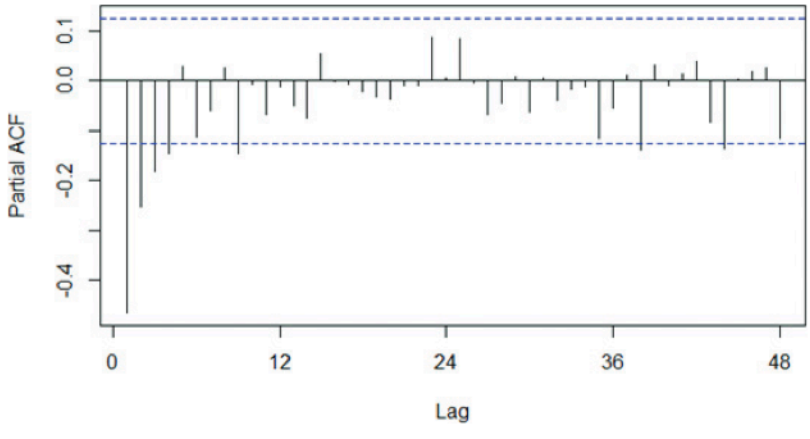


FIGURE 8-16 *PACF of residuals from seasonal AR(1) model*

Based on the standard errors associated with each coefficient estimate, the coefficients are significantly different from zero. In Figures 8-17 and 8-18, the respective ACF and PACF plots for the residuals from the second pass ARIMA model indicate that no further terms need to be considered in the ARIMA model.

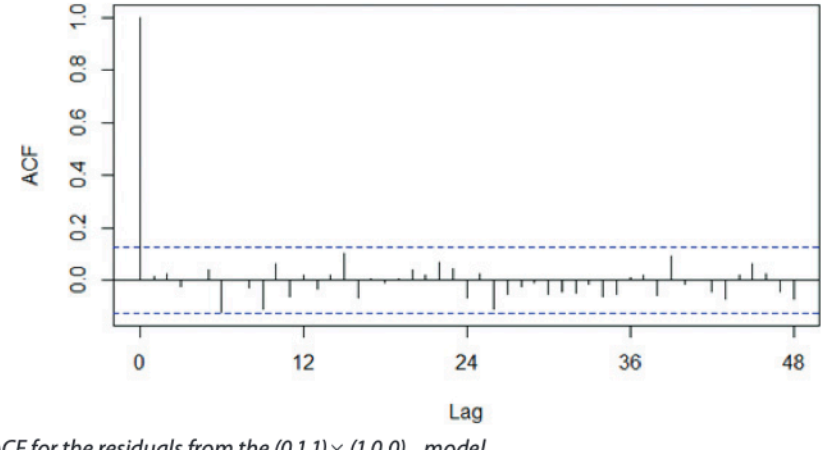


FIGURE 8-17 *ACF for the residuals from the $(0,1,1) \times (1,0,0)_{12}$ model*

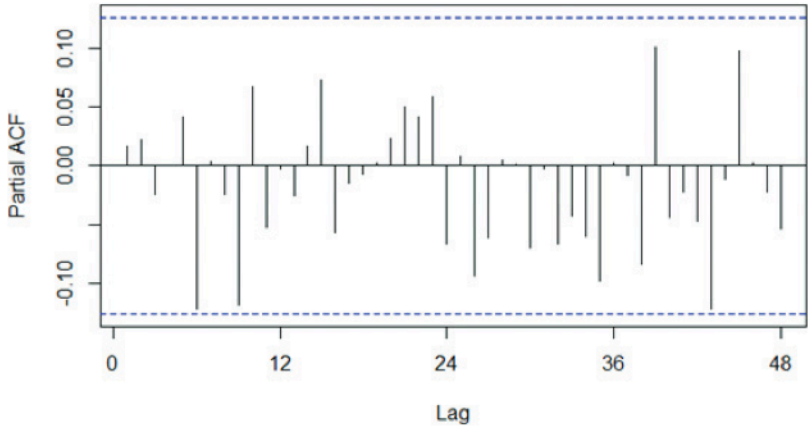


FIGURE 8-18 *PACF for the residuals from the $(0,1,1) \times (1,0,0)_{12}$ model*

It should be noted that the ACF and PACF plots each have several points that are close to the bounds at a 95% significance level. However, these points occur at relatively large lags. To avoid overfitting the model, these values are attributed to random chance. So no attempt is made to include these lags in the model. However, it is advisable to compare a reasonably fitting model to slight variations of that model.

Comparing Fitted Time Series Models

The `arima()` function in R uses Maximum Likelihood Estimation (MLE) to estimate the model coefficients. In the R output for an ARIMA model, the log-likelihood ($\log L$) value is provided. The values of the model coefficients are determined such that the value of the log likelihood function is maximized. Based on the $\log L$ value, the R output provides several measures that are useful for comparing the appropriateness of one fitted model against another fitted model. These measures follow:

- AIC (Akaike Information Criterion)
- AICc (Akaike Information Criterion, corrected)
- BIC (Bayesian Information Criterion)

Because these criteria impose a penalty based on the number of parameters included in the models, the preferred model is the fitted model with the smallest AIC, AICc, or BIC value. Table 8-1 provides the information criteria measures for the ARIMA models already fitted as well as a few additional fitted models. The highlighted row corresponds to the fitted ARIMA model obtained previously by examining the ACF and PACF plots.

TABLE 8-1 *Information Criteria to Measure Goodness of Fit*

ARIMA Model $(p,d,q) \times (P,Q,D)_S$	AIC	AICc	BIC
$(0,1,0) \times (1,0,0)_{12}$	1561.38	1561.43	1568.33
$(0,1,1) \times (1,0,0)_{12}$	1472.43	1472.53	1482.86
$(0,1,2) \times (1,0,0)_{12}$	1474.25	1474.42	1488.16
$(1,1,0) \times (1,0,0)_{12}$	1504.29	1504.39	1514.72
$(1,1,1) \times (1,0,0)_{12}$	1474.22	1474.39	1488.12

In this dataset, the $(0,1,1) \times (1,0,0)_{12}$ model does have the lowest AIC, AICc, and BIC values compared to the same criterion measures for the other ARIMA models.

Normality and Constant Variance

The last model validation step is to examine the normality assumption of the residuals in Equation 8-15. Figure 8-19 indicates residuals with a mean near zero and a constant variance over time. The histogram in Figure 8-20 and the Q-Q plot in Figure 8-21 support the assumption that the error terms are normally distributed. Q-Q plots were presented in Chapter 6, "Advanced Analytical Theory and Methods: Regression."

```
plot(arima_2$residuals, ylab = "Residuals")
abline(a=0, b=0)

hist(arima_2$residuals, xlab="Residuals", xlim=c(-20,20))

qqnorm(arima_2$residuals, main="")
qqline(arima_2$residuals)
```

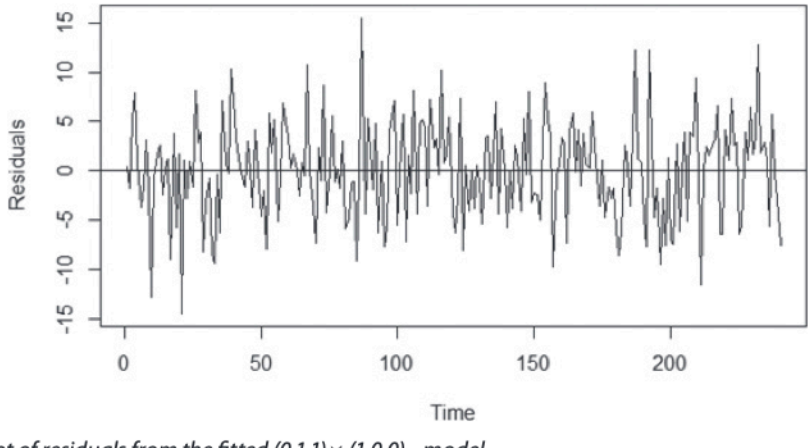


FIGURE 8-19 *Plot of residuals from the fitted $(0,1,1) \times (1,0,0)_{12}$ model*

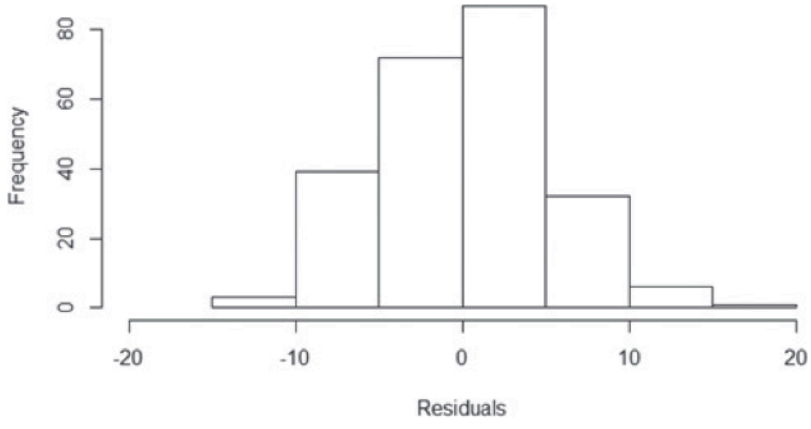


FIGURE 8-20 *Histogram of the residuals from the fitted (0,1,1) × (1,0,0)$_{12}$ model*

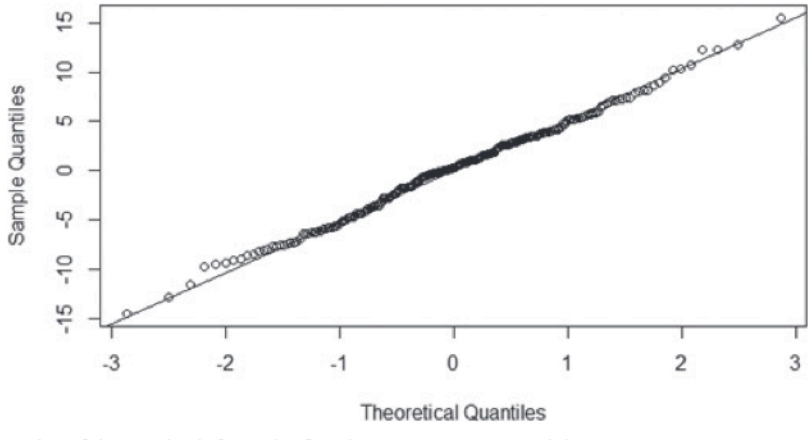


FIGURE 8-21 *Q-Q plot of the residuals from the fitted (0,1,1) × (1,0,0)$_{12}$ model*

If the normality or the constant variance assumptions do not appear to be true, it may be necessary to transform the time series prior to fitting the ARIMA model. A common transformation is to apply a logarithm function.

Forecasting

The next step is to use the fitted (0,1,1) × (1,0,0)$_{12}$ model to forecast the next 12 months of gasoline production. In R, the forecasts are easily obtained using the `predict()` function and the fitted model already stored in the variable *`arima_2`*. The predicted values along with the associated upper and lower bounds at a 95% confidence level are displayed in R and plotted in Figure 8-22.

```
#predict the next 12 months
arima_2.predict <- predict(arima_2,n.ahead=12)

matrix(c(arima_2.predict$pred-1.96*arima_2.predict$se,
```

```
          arima_2.predict$pred,
          arima_2.predict$pred+1.96*arima_2.predict$se), 12,3,
          dimnames=list( c(241:252) ,c("LB","Pred","UB"))  )

          LB     Pred       UB
241 394.9689 404.8167 414.6645
242 378.6142 388.8773 399.1404
243 394.9943 405.6566 416.3189
244 405.0188 416.0658 427.1128
245 397.9545 409.3733 420.7922
246 396.1202 407.8991 419.6780
247 396.6028 408.7311 420.8594
248 387.5241 399.9920 412.4598
249 387.1523 399.9507 412.7492
250 387.8486 400.9693 414.0900
251 383.1724 396.6076 410.0428
252 390.2075 403.9500 417.6926
plot(gas_prod, xlim=c(145,252),
     xlab = "Time (months)",
     ylab = "Gasoline production (millions of barrels)",
     ylim=c(360,440))
lines(arima_2.predict$pred)
lines(arima_2.predict$pred+1.96*arima_2.predict$se, col=4, lty=2)
lines(arima_2.predict$pred-1.96*arima_2.predict$se, col=4, lty=2)
```

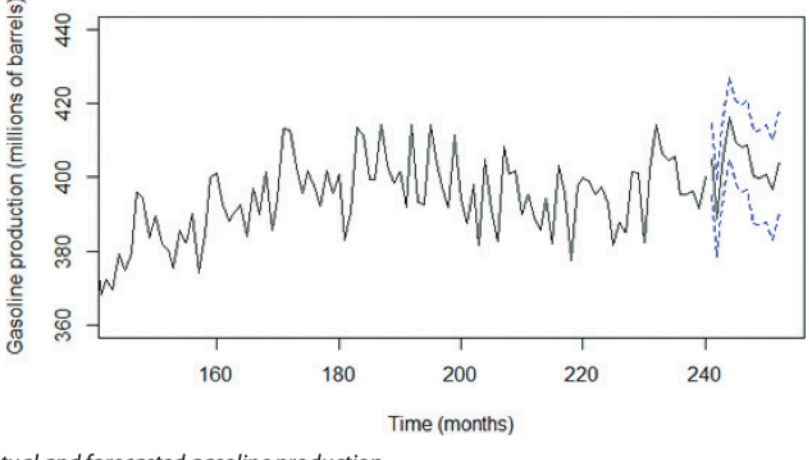


FIGURE 8-22 *Actual and forecasted gasoline production*

8.2.6 Reasons to Choose and Cautions

One advantage of ARIMA modeling is that the analysis can be based simply on historical time series data for the variable of interest. As observed in the chapter about regression (Chapter 6), various input variables

need to be considered and evaluated for inclusion in the regression model for the outcome variable. Because ARIMA modeling, in general, ignores any additional input variables, the forecasting process is simplified. If regression analysis was used to model gasoline production, input variables such as Gross Domestic Product (GDP), oil prices, and unemployment rate may be useful input variables. However, to forecast the gasoline production using regression, predictions are required for the GDP, oil price, and unemployment rate input variables.

The minimal data requirement also leads to a disadvantage of ARIMA modeling; the model does not provide an indication of what underlying variables affect the outcome. For example, if ARIMA modeling was used to forecast future retail sales, the fitted model would not provide an indication of what could be done to increase sales. In other words, causal inferences cannot be drawn from the fitted ARIMA model.

One caution in using time series analysis is the impact of severe shocks to the system. In the gas production example, shocks might include refinery fires, international incidents, or weather-related impacts such as hurricanes. Such events can lead to short-term drops in production, followed by persistently high increases in production to compensate for the lost production or to simply capitalize on any price increases.

Along similar lines of reasoning, time series analysis should only be used for short-term forecasts. Over time, gasoline production volumes may be affected by changing consumer demands as a result of more fuel-efficient gasoline-powered vehicles, electric vehicles, or the introduction of natural gas–powered vehicles. Changing market dynamics in addition to shocks will make any long-term forecasts, several years into the future, very questionable.

8.3 Additional Methods

Additional time series methods include the following:

- **Autoregressive Moving Average with Exogenous inputs (ARMAX)** is used to analyze a time series that is dependent on another time series. For example, retail demand for products can be modeled based on the previous demand combined with a weather-related time series such as temperature or rainfall.
- **Spectral analysis** is commonly used for signal processing and other engineering applications. Speech recognition software uses such techniques to separate the signal for the spoken words from the overall signal that may include some noise.
- **Generalized Autoregressive Conditionally Heteroscedastic (GARCH)** is a useful model for addressing time series with nonconstant variance or volatility. GARCH is used for modeling stock market activity and price fluctuations.
- **Kalman filtering** is useful for analyzing real-time inputs about a system that can exist in certain states. Typically, there is an underlying model of how the various components of the system interact and affect each other. A Kalman filter processes the various inputs, attempts to identify the errors in the input, and predicts the current state. For example, a Kalman filter in a vehicle navigation system can process various inputs, such as speed and direction, and update the estimate of the current location.
- **Multivariate time series analysis** examines multiple time series and their effect on each other. Vector ARIMA (VARIMA) extends ARIMA by considering a vector of several time series at a particular time, t. VARIMA can be used in marketing analyses that examine the time series related to a company's price and sales volume as well as related time series for the competitors.

Summary

This chapter presented time series analysis using ARIMA models. Time series analysis is different from other statistical techniques in the sense that most statistical analyses assume the observations are independent of each other. Time series analysis implicitly addresses the case in which any particular observation is somewhat dependent on prior observations.

Using differencing, ARIMA models allow nonstationary series to be transformed into stationary series to which seasonal and nonseasonal ARMA models can be applied. The importance of using the ACF and PACF plots to evaluate the autocorrelations was illustrated in determining ARIMA models to consider fitting. Akaike and Bayesian Information Criteria can be used to compare one fitted ARIMA model against another. Once an appropriate model has been determined, future values in the time series can be forecasted.

Exercises

1. Why use autocorrelation instead of autocovariance when examining stationary time series?
2. Provide an example that if the cov(X, Y) = 0, the two random variables, X and Y, are not necessarily independent.
3. Fit an appropriate ARIMA model on the following datasets included in R. Provide supporting evidence on why the fitted model was selected, and forecast the time series for 12 time periods ahead.
   a. **faithful:** Waiting times (in minutes) between Old Faithful geyser eruptions
   b. **JohnsonJohnson:** Quarterly earnings per J&J share
   c. **sunspot.month:** Monthly sunspot activity from 1749 to 1997
4. When should an ARIMA(p,d,q) model in which d > 0 be considered instead of an ARMA(p,q) model?

9

Advanced Analytical Theory and Methods: Text Analysis

Key Concepts

Term
Corpus
Text normalization
TFIDF
Topic modeling
Sentiment analysis

Text analysis, sometimes called text analytics, refers to the representation, processing, and modeling of textual data to derive useful insights. An important component of text analysis is text mining, the process of discovering relationships and interesting patterns in large text collections.

Text analysis suffers from the curse of high dimensionality. Take the popular children's book *Green Eggs and Ham* [1] as an example. Author Theodor Geisel (Dr. Seuss) was challenged to write an entire book with just 50 distinct words. He responded with the book *Green Eggs and Ham*, which contains 804 total words, only 50 of them distinct. These 50 words are:

> a, am, and, anywhere, are, be, boat, box, car, could, dark, do, eat, eggs, fox, goat, good, green, ham, here, house, I, if, in, let, like, may, me, mouse, not, on, or, rain, Sam, say, see, so, thank, that, the, them, there, they, train, tree, try, will, with, would, you

There's a substantial amount of repetition in the book. Yet, as repetitive as the book is, modeling it as a vector of counts, or features, for each distinct word still results in a 50-dimension problem.

Green Eggs and Ham is a simple book. Text analysis often deals with textual data that is far more complex. A ***corpus*** (plural: corpora) is a large collection of texts used for various purposes in Natural Language Processing (NLP). Table 9-1 lists a few example corpora that are commonly used in NLP research.

TABLE 9-1 *Example Corpora in Natural Language Processing*

Corpus	Word Count	Domain	Website
Shakespeare	0.88 million	Written	`http://shakespeare.mit.edu/`
Brown Corpus	1 million	Written	http://icame.uib.no/brown/bcm.html
Penn Treebank	1 million	Newswire	`http://www.cis.upenn.edu/~treebank/`
Switchboard Phone Conversations	3 million	Spoken	http://catalog.ldc.upenn.edu/LDC97S62
British National Corpus	100 million	Written and spoken	`http://www.natcorp.ox.ac.uk/`
NA News Corpus	350 million	Newswire	http://catalog.ldc.upenn.edu/LDC95T21
European Parliament Proceedings Parallel Corpus	600 million	Legal	`http://www.statmt.org/europarl/`
Google N-Grams Corpus	1 trillion	Written	http://catalog.ldc.upenn.edu/LDC2006T13

The smallest corpus in the list, the complete works of Shakespeare, contains about 0.88 million words. In contrast, the Google *n*-gram corpus contains one trillion words from publicly accessible web pages. Out of the one trillion words in the Google *n*-gram corpus, there might be one million distinct words, which would correspond to one million dimensions. The high dimensionality of text is an important issue, and it has a direct impact on the complexities of many text analysis tasks.

Another major challenge with text analysis is that most of the time the text is not structured. As introduced in Chapter 1, "Introduction to Big Data Analytics," this may include quasi-structured, semi-structured, or unstructured data. Table 9-2 shows some example data sources and data formats that text analysis may have to deal with. Note that this is not meant as an exhaustive list; rather, it highlights the challenge of text analysis.

TABLE 9-2 *Example Data Sources and Formats for Text Analysis*

Data Source	Data Format	Data Structure Type
News articles	TXT, HTML, or Scanned PDF	Unstructured
Literature	TXT, DOC, HTML, or PDF	Unstructured
E-mail	TXT, MSG, or EML	Unstructured
Web pages	HTML	Semi-structured
Server logs	LOG or TXT	Semi-structured or Quasi-structured
Social network API firehoses	XML, JSON, or RSS	Semi-structured
Call center transcripts	TXT	Unstructured

9.1 Text Analysis Steps

A text analysis problem usually consists of three important steps: parsing, search and retrieval, and text mining. Note that a text analysis problem may also consist of other subtasks (such as discourse and segmentation) that are outside the scope of this book.

Parsing is the process that takes unstructured text and imposes a structure for further analysis. The unstructured text could be a plain text file, a weblog, an Extensible Markup Language (XML) file, a HyperText Markup Language (HTML) file, or a Word document. Parsing deconstructs the provided text and renders it in a more structured way for the subsequent steps.

Search and retrieval is the identification of the documents in a corpus that contain search items such as specific words, phrases, topics, or entities like people or organizations. These search items are generally called ***key terms.*** Search and retrieval originated from the field of library science and is now used extensively by web search engines.

Text mining uses the terms and indexes produced by the prior two steps to discover meaningful insights pertaining to domains or problems of interest. With the proper representation of the text, many of the techniques mentioned in the previous chapters, such as clustering and classification, can be adapted to text mining. For example, the *k*-means from Chapter 4, "Advanced Analytical Theory and Methods: Clustering," can be modified to cluster text documents into groups, where each group represents a collection of documents with a similar topic [2]. The distance of a document to a centroid represents how closely the document talks about that topic. Classification tasks such as sentiment analysis and spam filtering are prominent use

cases for the naïve Bayes classifier (Chapter 7, "Advanced Analytical Theory and Methods: Classification"). Text mining may utilize methods and techniques from various fields of study, such as statistical analysis, information retrieval, data mining, and natural language processing.

Note that, in reality, all three steps do not have to be present in a text analysis project. If the goal is to construct a corpus or provide a catalog service, for example, the focus would be the parsing task using one or more text preprocessing techniques, such as part-of-speech (POS) tagging, named entity recognition, lemmatization, or stemming. Furthermore, the three tasks do not have to be sequential. Sometimes their orders might even look like a tree. For example, one could use parsing to build a data store and choose to either search and retrieve the related documents or use text mining on the entire data store to gain insights.

Part-of-Speech (POS) Tagging, Lemmatization, and Stemming

The goal of ***POS tagging*** is to build a model whose input is a sentence, such as:

he saw a fox

and whose output is a tag sequence. Each tag marks the POS for the corresponding word, such as:

PRP VBD DT NN

according to the Penn Treebank POS tags [3]. Therefore, the four words are mapped to pronoun (personal), verb (past tense), determiner, and noun (singular), respectively.

Both lemmatization and stemming are techniques to reduce the number of dimensions and reduce inflections or variant forms to the base form to more accurately measure the number of times each word appears.

With the use of a given dictionary, ***lemmatization*** finds the correct dictionary base form of a word. For example, given the sentence:

obesity causes many problems

the output of lemmatization would be:

obesity cause many problem

Different from lemmatization, ***stemming*** does not need a dictionary, and it usually refers to a crude process of stripping affixes based on a set of heuristics with the hope of correctly achieving the goal to reduce inflections or variant forms. After the process, words are stripped to become *stems*. A stem is not necessarily an actual word defined in the natural language, but it is sufficient to differentiate itself from the stems of other words. A well-known rule-based stemming algorithm is ***Porter's stemming algorithm***. It defines a set of production rules to iteratively transform words into their stems. For the sentence shown previously:

obesity causes many problems

the output of Porter's stemming algorithm is:

obes caus mani problem

9.2 A Text Analysis Example

To further describe the three text analysis steps, consider the fictitious company ACME, maker of two products: `bPhone` and `bEbook`. ACME is in strong competition with other companies that manufacture and sell similar products. To succeed, ACME needs to produce excellent phones and eBook readers and increase sales.

One of the ways the company does this is to monitor what is being said about ACME products in social media. In other words, what is the buzz on its products? ACME wants to search all that is said about ACME products in social media sites, such as Twitter and Facebook, and popular review sites, such as Amazon and ConsumerReports. It wants to answer questions such as these.

- Are people mentioning its products?
- What is being said? Are the products seen as good or bad? If people think an ACME product is bad, why? For example, are they complaining about the battery life of the `bPhone`, or the response time in their `bEbook`?

ACME can monitor the social media buzz using a simple process based on the three steps outlined in Section 9.1. This process is illustrated in Figure 9-1, and it includes the modules in the next list.

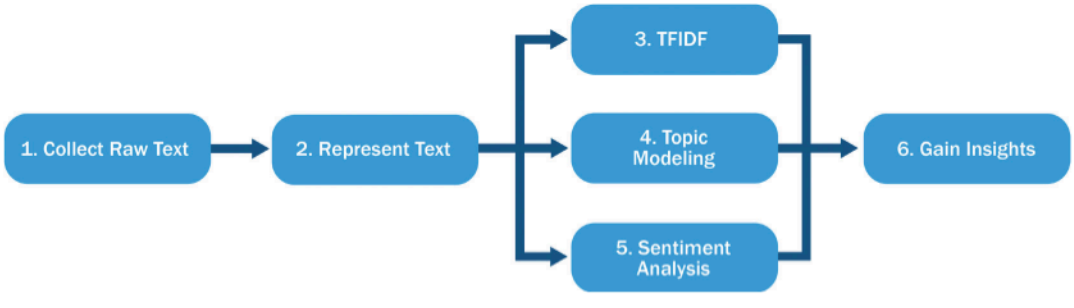


FIGURE 9-1 *ACME's Text Analysis Process*

1. Collect raw text (Section 9.3). This corresponds to Phase 1 and Phase 2 of the Data Analytic Lifecycle. In this step, the Data Science team at ACME monitors websites for references to specific products. The websites may include social media and review sites. The team could interact with social network application programming interfaces (APIs) process data feeds, or scrape pages and use product names as keywords to get the raw data. Regular expressions are commonly used in this case to identify text that matches certain patterns. Additional filters can be applied to the raw data for a more focused study. For example, only retrieving the reviews originating in New York instead of the entire United States would allow ACME to conduct regional studies on its products. Generally, it is a good practice to apply filters during the data collection phase. They can reduce I/O workloads and minimize the storage requirements.
2. Represent text (Section 9.4). Convert each review into a suitable document representation with proper indices, and build a corpus based on these indexed reviews. This step corresponds to Phases 2 and 3 of the Data Analytic Lifecycle.

3. Compute the usefulness of each word in the reviews using methods such as TFIDF (Section 9.5). This and the following two steps correspond to Phases 3 through 5 of the Data Analytic Lifecycle.
4. Categorize documents by topics (Section 9.6). This can be achieved through topic models (such as latent Dirichlet allocation).
5. Determine sentiments of the reviews (Section 9.7). Identify whether the reviews are positive or negative. Many product review sites provide ratings of a product with each review. If such information is not available, techniques like sentiment analysis can be used on the textual data to infer the underlying sentiments. People can express many emotions. To keep the process simple, ACME considers sentiments as positive, neutral, or negative.
6. Review the results and gain greater insights (Section 9.8). This step corresponds to Phase 5 and 6 of the Data Analytic Lifecycle. Marketing gathers the results from the previous steps. Find out what exactly makes people love or hate a product. Use one or more visualization techniques to report the findings. Test the soundness of the conclusions and operationalize the findings if applicable.

This process organizes the topics presented in the rest of the chapter and calls out some of the difficulties that are unique to text analysis.

9.3 Collecting Raw Text

Recall that in the Data Analytic Lifecycle seen in Chapter 2, "Data Analytics Lifecycle," discovery is the first phase. In it, the Data Science team investigates the problem, understands the necessary data sources, and formulates initial hypotheses. Correspondingly, for text analysis, data must be collected before anything can happen. The Data Science team starts by actively monitoring various websites for user-generated contents. The user-generated contents being collected could be related articles from news portals and blogs, comments on ACME's products from online shops or reviews sites, or social media posts that contain keywords `bPhone` or `bEbook`. Regardless of where the data comes from, it's likely that the team would deal with semi-structured data such as HTML web pages, Really Simple Syndication (RSS) feeds, XML, or JavaScript Object Notation (JSON) files. Enough structure needs to be imposed to find the part of the raw text that the team really cares about. In the brand management example, ACME is interested in what the reviews say about `bPhone` or `bEbook` and when the reviews are posted. Therefore, the team will actively collect such information.

Many websites and services offer public APIs [4, 5] for third-party developers to access their data. For example, the Twitter API [6] allows developers to choose from the Streaming API or the REST API to retrieve public Twitter posts that contain the keywords `bPhone` or `bEbook`. Developers can also read tweets in real time from a specific user or tweets posted near a specific venue. The fetched tweets are in the JSON format.

As an example, a sample tweet that contains the keyword `bPhone` fetched using the Twitter Streaming API version 1.1 is shown next.

```
{
  "created_at": "Thu Aug 15 20:06:48 +0000 2013",
  "coordinates": {
      "type": "Point",
      "coordinates": [
```

```
            -157.81538521787621,
            21.3002578885766
        ]
    },
        "favorite_count": 0,
        "id": 368101488276824010,
        "id_str": "368101488276824014",
        "lang": "en",
        "metadata": {
            "iso_language_code": "en",
            "result_type": "recent"
        },
        "retweet_count": 0,
        "retweeted": false,
        "source": "<a href=\"http://www.twitter.com\"
                    rel=\"nofollow\">Twitter for bPhone</a>",
        "text": "I once had a gf back in the day. Then the bPhone
                    came out lol",
        "truncated": false,
        "user": {
            "contributors_enabled": false,
            "created_at": „Mon Jun 24 09:15:54 +0000 2013",
            "default_profile": false,
            "default_profile_image": false,
            "description": "Love Life and Live Good",
            "favourites_count": 23,
            "follow_request_sent": false,
            "followers_count": 96,
            "following": false,
            "friends_count": 347,
            "geo_enabled": false,
            "id": 2542887414,
            "id_str": "2542887414",
            "is_translator": false,
            "lang": "en-gb",
            "listed_count": 0,
            "location": "Beautiful Hawaii",
            "name": "The Original DJ Ice",
            "notifications": false,
            "profile_background_color": "C0DEED",
            "profile_background_image_url":
 "http://a0.twimg.com/profile_bg_imgs/378800000/b12e56725ee.jpeg",
            "profile_background_tile": true,
            "profile_image_url":
 "http://a0.twimg.com/profile_imgs/378800010/2d55a4388bcffd5.jpeg",
            "profile_link_color": "0084B4",
            "profile_sidebar_border_color": "FFFFFF",
```

```
53            "profile_sidebar_fill_color": "DDEEF6",
54            "profile_text_color": "333333",
55            "profile_use_background_image": true,
56            "protected": false,
57            "screen_name": "DJ_Ice",
58            "statuses_count": 186,
59            "time_zone": "Hawaii",
60            "url": null,
61            "utc_offset": -36000,
62            "verified": false
63        }
64 }
```

Fields *created_at* at line 2 and *text* at line 22 in the previous tweet provide the information that interests ACME. The *created_at* entry stores the timestamp that the tweet was published, and the *text* field stores the main content of the Twitter post. Other fields could be useful, too. For example, utilizing fields such as *coordinates* (line 3 to 9), user's local language (*lang*, line 40), user's *location* (line 42), *time_zone* (line 59), and *utc_offset* (line 61) allows the analysis to focus on tweets from a specific region. Therefore, the team can research what people say about ACME's products at a more granular level.

Many news portals and blogs provide data feeds that are in an open standard format, such as RSS or XML. As an example, an RSS feed for a phone review blog is shown next.

```
01 <channel>
02     <title>All about Phones</title>
03     <description>My Phone Review Site</description>
04     <link>http://www.phones.com/link.htm</link>
05
06     <item>
07         <title>bPhone: The best!</title>
08         <description>I love LOVE my bPhone!</description>
09         <link>http://www.phones.com/link.htm</link>
10         <guid isPermaLink="false">1102345</guid>
11         <pubDate>Tue, 29 Aug 2011 09:00:00 -0400</pubDate>
12     <item>
13 </channel>
```

The content from the *title* (line 7), the *description* (line 8), and the published date (*pubDate*, line 11) is what ACME is interested in.

If the plan is to collect user comments on ACME's products from online shops and review sites where APIs or data feeds are not provided, the team may have to write web scrapers to parse web pages and automatically extract the interesting data from those HTML files. A ***web scraper*** is a software program (bot) that systematically browses the World Wide Web, downloads web pages, extracts useful information, and stores it somewhere for further study.

Unfortunately, it is nearly impossible to write a one-size-fits-all web scraper. This is because websites like online shops and review sites have different structures. It is common to customize a web scraper for a specific website. In addition, the website formats can change over time, which requires the web scraper to

be updated every now and then. To build a web scraper for a specific website, one must study the HTML source code of its web pages to find patterns before extracting any useful content. For example, the team may find out that each user comment in the HTML is enclosed by a DIV element inside another DIV with the ID `usrcommt`, or it might be enclosed by a DIV element with the CLASS `commtcls`.

The team can then construct the web scraper based on the identified patterns. The scraper can use the `curl` tool [7] to fetch HTML source code given specific URLs, use XPath [8] and regular expressions to select and extract the data that match the patterns, and write them into a data store.

Regular expressions can find words and strings that match particular patterns in the text effectively and efficiently. Table 9-3 shows some regular expressions. The general idea is that once text from the fields of interest is obtained, regular expressions can help identify if the text is really interesting for the project. In this case, do those fields mention `bPhone`, `bEbook`, or `ACME`? When matching the text, regular expressions can also take into account capitalizations, common misspellings, common abbreviations, and special formats for e-mail addresses, dates, and telephone numbers.

TABLE 9-3 *Example Regular Expressions*

Regular Expression	Matches	Note
`b(P|p)hone`	bPhone, bphone	Pipe "|" means "or"
`bEbo*k`	bEbk, bEbok, bEbook, bEboook, bEbooook, bEboooook, …	"*" matches zero or more occurrences of the preceding letter
`bEbo+k`	bEbok, bEbook, bEboook, bEbooook, bEboooook, …	"+" matches one or more occurrences of the preceding letter
`bEbo{2,4}k`	bEbook, bEboook, bEbooook	"{2,4}" matches from two to four repetitions of the preceding letter "o"
`^I love`	Text starting with "I love"	"^" matches the start of a string
`ACME$`	Text ending with "ACME"	"$" matches the end of a string

This section has discussed three different sources where raw data may come from: tweets that contain keywords `bPhone` or `bEbook`, related articles from news portals and blogs, and comments on ACME's products from online shops or reviews sites.

If one chooses not to build a data collector from scratch, many companies such as GNIP [9] and DataSift [10] can provide data collection or data reselling services.

Depending on how the fetched raw data will be used, the Data Science team needs to be careful not to violate the rights of the owner of the information and user agreements about use of websites during the data collection. Many websites place a file called `robots.txt` in the root directory—that is, `http://.../robots.txt` (for example, `http://www.amazon.com/robots.txt`). It lists the directories and files that are allowed or disallowed to be visited so that web scrapers or web crawlers know how to treat the website correctly.

9.4 Representing Text

After the previous step, the team now has some raw text to start with. In this data representation step, raw text is first transformed with text normalization techniques such as tokenization and case folding. Then it is represented in a more structured way for analysis.

Tokenization is the task of separating (also called tokenizing) words from the body of text. Raw text is converted into collections of tokens after the tokenization, where each token is generally a word.

A common approach is tokenizing on spaces. For example, with the tweet shown previously:

```
I once had a gf back in the day. Then the bPhone came out lol
```

tokenization based on spaces would output a list of tokens.

```
{I, once, had, a, gf, back, in, the, day.,
Then, the, bPhone, came, out, lol}
```

Note that token "`day.`" contains a period. This is the result of only using space as the separator. Therefore, tokens "`day.`" and "`day`" would be considered different terms in the downstream analysis unless an additional lookup table is provided. One way to fix the problem without the use of a lookup table is to remove the period if it appears at the end of a sentence. Another way is to tokenize the text based on punctuation marks and spaces. In this case, the previous tweet would become:

```
{I, once, had, a, gf, back, in, the, day, .,
Then, the, bPhone, came, out, lol}
```

However, tokenizing based on punctuation marks might not be well suited to certain scenarios. For example, if the text contains contractions such as `we'll`, tokenizing based on punctuation will split them into separated words `we` and `ll`. For words such as `can't`, the output would be `can` and `t`. It would be more preferable either not to tokenize them or to tokenize `we'll` into `we` and `'ll`, and `can't` into `can` and `'t`. The `'t` token is more recognizable as negative than the `t` token. If the team is dealing with certain tasks such as information extraction or sentiment analysis, tokenizing solely based on punctuation marks and spaces may obscure or even distort meanings in the text.

Tokenization is a much more difficult task than one may expect. For example, should words like `state-of-the-art`, `Wi-Fi`, and `San Francisco` be considered one token or more? Should words like `Résumé`, `résumé`, and `resume` all map to the same token? Tokenization is even more difficult beyond English. In German, for example, there are many unsegmented compound nouns. In Chinese, there are no spaces between words. Japanese has several alphabets intermingled. This list can go on.

It's safe to say that there is no single tokenizer that will work in every scenario. The team needs to decide what counts as a token depending on the domain of the task and select an appropriate tokenization technique that fits most situations well. In reality, it's common to pair a standard tokenization technique with a lookup table to address the contractions and terms that should not be tokenized. Sometimes it may not be a bad idea to develop one's own tokenization from scratch.

Another text normalization technique is called ***case folding***, which reduces all letters to lowercase (or the opposite if applicable). For the previous tweet, after case folding the text would become this:

```
i once had a gf back in the day. then the bphone came out lol
```

One needs to be cautious applying case folding to tasks such as information extraction, sentiment analysis, and machine translation. If implemented incorrectly, case folding may reduce or change the meaning of the text and create additional noise. For example, when `General Motors` becomes `general` and `motors`, the downstream analysis may very likely consider them as separated words rather than the name of a company. When the abbreviation of the World Health Organization `WHO` or the rock band `The Who` become `who`, they may both be interpreted as the pronoun `who`.

If case folding must be present, one way to reduce such problems is to create a lookup table of words not to be case folded. Alternatively, the team can come up with some heuristics or rules-based strategies for the case folding. For example, the program can be taught to ignore words that have uppercase in the middle of a sentence.

After normalizing the text by tokenization and case folding, it needs to be represented in a more structured way. A simple yet widely used approach to represent text is called ***bag-of-words***. Given a document, bag-of-words represents the document as a set of terms, ignoring information such as order, context, inferences, and discourse. Each word is considered a term or token (which is often the smallest unit for the analysis). In many cases, bag-of-words additionally assumes every term in the document is independent. The document then becomes a vector with one dimension for every distinct term in the space, and the terms are unordered. The permutation D* of a document D contains the same words exactly the same number of times but in a different order. Therefore, using the bag-of-words representation, document D and its permutation D* would share the same representation.

Bag-of-words takes quite a naïve approach, as order plays an important role in the semantics of text. With bag-of-words, many texts with different meanings are combined into one form. For example, the texts "a dog bites a man" and "a man bites a dog" have very different meanings, but they would share the same representation with bag-of-words.

Although the bag-of-words technique oversimplifies the problem, it is still considered a good approach to start with, and it is widely used for text analysis. A paper by Salton and Buckley [11] states the effectiveness of using single words as identifiers as opposed to multiple-term identifiers, which retain the order of the words:

> ***In reviewing the extensive literature accumulated during the past 25 years in the area of retrieval system evaluation, the overwhelming evidence is that the judicious use of single-term identifiers is preferable to the incorporation of more complex entities extracted from the texts themselves or obtained from available vocabulary schedules.***

Although the work by Salton and Buckley was published in 1988, there has been little, if any, substantial evidence to discredit the claim. Bag-of-words uses single-term identifiers, which are usually sufficient for the text analysis in place of multiple-term identifiers.

Using single words as identifiers with the bag-of-words representation, the ***term frequency*** (TF) of each word can be calculated. Term frequency represents the weight of each term in a document, and it is proportional to the number of occurrences of the term in that document. Figure 9-2 shows the 50 most frequent words and the numbers of occurrences from Shakespeare's *Hamlet*. The word frequency distribution roughly follows ***Zipf's Law*** [12, 13]—that is, the i-th most common word occurs approximately $1/i$ as

often as the most frequent term. In other words, the frequency of a word is inversely proportional to its rank in the frequency table. Term frequency is revisited later in this chapter.

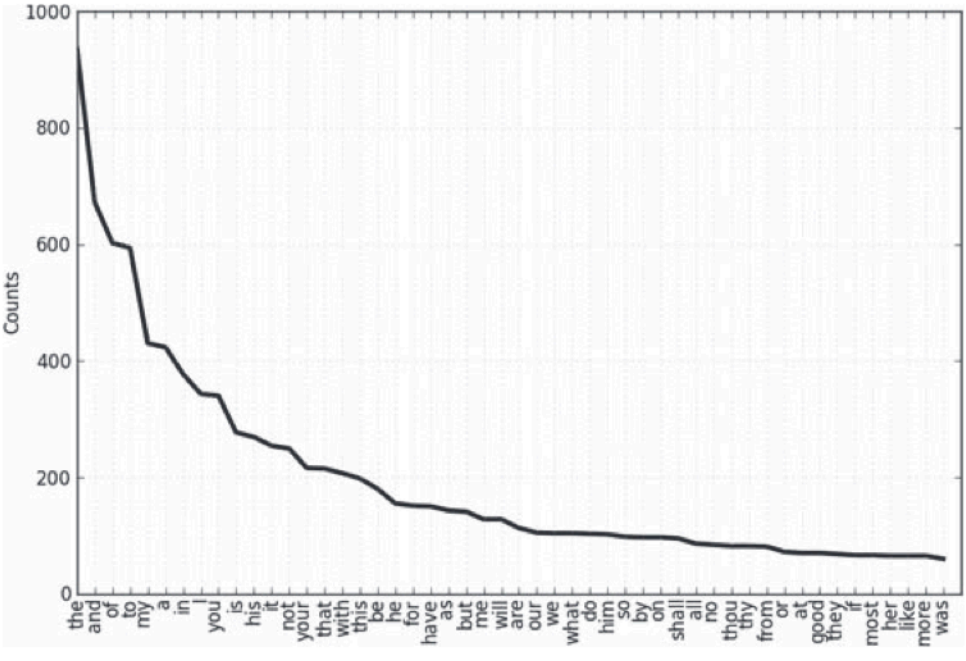


FIGURE 9-2 *The 50 most frequent words in Shakespeare's Hamlet*

What's Beyond Bag-of-Words?

Bag-of-words is a common technique to start with. But sometimes the Data Science team prefers other methods of text representation that are more sophisticated. These more advanced methods consider factors such as word order, context, inferences, and discourse. For example, one such method can keep track of the word order of every document and compare the normalized differences of the word orders [14]. These advanced techniques are outside the scope of this book.

Besides extracting the terms, their morphological ***features*** may need to be included. The morphological features specify additional information about the terms, which may include root words, affixes, part-of-speech tags, named entities, or intonation (variations of spoken pitch). The features from this step contribute to the downstream analysis in classification or sentiment analysis.

The set of features that need to be extracted and stored highly depends on the specific task to be performed. If the task is to label and distinguish the part of speech, for example, the features will include all the words in the text and their corresponding part-of-speech tags. If the task is to annotate the named entities

like names and organizations, the features highlight such information appearing in the text. Constructing the features is no trivial task; quite often this is done entirely manually, and sometimes it requires domain expertise.

Sometimes creating features is a text analysis task all to itself. One such example is ***topic modeling.*** Topic modeling provides a way to quickly analyze large volumes of raw text and identify the latent topics. Topic modeling may not require the documents to be labeled or annotated. It can discover topics directly from an analysis of the raw text. A topic consists of a cluster of words that frequently occur together and that share the same theme. Probabilistic topic modeling, discussed in greater detail later in Section 9.6, is a suite of algorithms that aim to parse large archives of documents and discover and annotate the topics.

It is important not only to create a representation of a document but also to create a representation of a corpus. As introduced earlier in the chapter, a corpus is a collection of documents. A corpus could be so large that it includes all the documents in one or more languages, or it could be smaller or limited to a specific domain, such as technology, medicine, or law. For a web search engine, the entire World Wide Web is the relevant corpus. Most corpora are much smaller. The Brown Corpus [15] was the first million-word electronic corpus of English, created in 1961 at Brown University. It includes text from around 500 sources, and the source has been categorized into 15 genres, such as news, editorial, fiction, and so on. Table 9-4 lists the genres of the Brown Corpus as an example of how to organize information in a corpus.

TABLE 9-4 *Categories of the Brown Corpus*

Category	Number of Sources	Example Source
A. Reportage	44	*Chicago Tribune*
B. Editorial	27	*Christian Science Monitor*
C. Reviews	17	*Life*
D. Religion	17	*William Pollard: Physicist and Christian*
E. Skills and Hobbies	36	*Joseph E. Choate: The American Boating Scene*
F. Popular Lore	48	*David Boroff: Jewish Teen-Age Culture*
G. Belles Lettres, Biography, Memoirs, and so on	75	*Selma J. Cohen: Avant-Garde Choreography*
H. Miscellaneous	30	*U. S. Dep't of Defense: Medicine in National Defense*
J. Learned	80	*J. F. Vedder: Micrometeorites*
K. General Fiction	29	*David Stacton: The Judges of the Secret Court*

(*continues*)

TABLE 9-4 *Categories of the Brown Corpus (Continued)*

Category	Number of Sources	Example Source
L. Mystery and Detective Fiction	24	*S. L. M. Barlow: Monologue of Murder*
M. Science Fiction	6	*Jim Harmon: The Planet with No Nightmare*
N. Adventure and Western Fiction	29	*Paul Brock: Toughest Lawman in the Old West*
P. Romance and Love Story	29	*Morley Callaghan: A Passion in Rome*
R. Humor	9	*Evan Esar: Humorous English*

Many corpora focus on specific domains. For example, the BioCreative corpora [16] are from biology, the Switchboard corpus [17] contains telephone conversations, and the European Parliament Proceedings Parallel Corpus [18] was extracted from the proceedings of the European Parliament in 21 European languages.

Most corpora come with metadata, such as the size of the corpus and the domains from which the text is extracted. Some corpora (such as the Brown Corpus) include the information content of every word appearing in the text. ***Information content*** (IC) is a metric to denote the importance of a term in a corpus. The conventional way [19] of measuring the IC of a term is to combine the knowledge of its hierarchical structure from an ontology with statistics on its actual usage in text derived from a corpus. Terms with higher IC values are considered more important than terms with lower IC values. For example, the word *necklace* generally has a higher IC value than the word *jewelry* in an English corpus because *jewelry* is more general and is likely to appear more often than *necklace*. Research shows that IC can help measure the semantic similarity of terms [20]. In addition, such measures do not require an annotated corpus, and they generally achieve strong correlations with human judgment [21, 20].

In the brand management example, the team has collected the ACME product reviews and turned them into the proper representation with the techniques discussed earlier. Next, the reviews and the representation need to be stored in a searchable archive for future reference and research. This archive could be a SQL database, XML or JSON files, or plain text files from one or more directories.

Corpus statistics such as IC can help identify the importance of a term from the documents being analyzed. However, IC values included in the metadata of a traditional corpus (such as Brown corpus) sitting externally as a knowledge base cannot satisfy the need to analyze the dynamically changed, unstructured data from the web. The problem is twofold. First, both traditional corpora and IC metadata do not change over time. Any term not existing in the corpus text and any newly invented words would automatically receive a zero IC value. Second, the corpus represents the entire knowledge base for the algorithm being used in the downstream analysis. The nature of the unstructured text determines that the data being analyzed can contain any topics, many of which may be absent in the given knowledge base. For example, if the task is to research people's attitudes on musicians, a traditional corpus constructed 50 years ago would not know that the term *U2* is a band; therefore, it would receive a zero on IC, which means it's not an

important term. A better approach would go through all the fetched documents and find out that most of them are related to music, with *U2* appearing too often to be an unimportant term. Therefore, it is necessary to come up with a metric that can easily adapt to the context and nature of the text instead of relying on a traditional corpus. The next section discusses such a metric. It's known as Term Frequency—Inverse Document Frequency (TFIDF), which is based entirely on all the fetched documents and which keeps track of the importance of terms occurring in each of the documents.

Note that the fetched documents may change constantly over time. Consider the case of a web search engine, in which each fetched document corresponds to a matching web page in a search result. The documents are added, modified, or removed and, as a result, the metrics and indices must be updated correspondingly. Additionally, word distributions can change over time, which reduces the effectiveness of classifiers and filters (such as spam filters) unless they are retrained.

9.5 Term Frequency—Inverse Document Frequency (TFIDF)

This section presents TFIDF, a measure widely used in information retrieval and text analysis. Instead of using a traditional corpus as a knowledge base, TFIDF directly works on top of the fetched documents and treats these documents as the "corpus." TFIDF is robust and efficient on dynamic content, because document changes require only the update of frequency counts.

Given a term t and a document $d = \{t_1, t_2, t_3, \ldots t_n\}$ containing n terms, the simplest form of term frequency of t in d can be defined as the number of times t appears in d, as shown in Equation 9-1.

$$TF_1(t,d) = \sum_{i=1}^{n} f(t,t_i) \qquad t_i \in d; |d| = n$$

where

$$f(t,t') = \begin{cases} 1, & \text{if } t = t' \\ 0, & \text{otherwise} \end{cases} \tag{9-1}$$

To understand how the term frequency is computed, consider a bag-of-words vector space of 10 words: `i`, `love`, `acme`, `my`, `bebook`, `bphone`, `fantastic`, `slow`, `terrible`, and `terrific`. Given the text `I love LOVE my bPhone` extracted from the RSS feed in Section 9.3, Table 9-5 shows its corresponding term frequency vector after case folding and tokenization.

TABLE 9-5 *A Sample Term Frequency Vector*

Term	Frequency
i	1
love	2
acme	0

(*continues*)

TABLE 9-5 *A Sample Term Frequency Vector (Continued)*

Term	Frequency
my	1
bebook	0
bphone	1
fantastic	0
slow	0
terrible	0
terrific	0

The term frequency function can be logarithmically scaled. Recall that in Figure 3-11 and Figure 3-12 of Chapter 3, "Review of Basic Data Analytic Methods Using R," it shows the logarithm can be applied to distribution with a long tail to enable more data detail. Similarly, the logarithm can be applied to word frequencies whose distribution also contains a long tail, as shown in Equation 9-2.

$$TF_2(t,d) = \log[TF_1(t,d)+1] \tag{9-2}$$

Because longer documents contain more terms, they tend to have higher term frequency values. They also tend to contain more distinct terms. These factors can conspire to raise the term frequency values of longer documents and lead to undesirable bias favoring longer documents. To address this problem, the term frequency can be normalized. For example, the term frequency of term t in document d can be normalized based on the number of terms in d as shown in Equation 9-3.

$$TF_3(t,d) = \frac{TF_1(t,d)}{n} \qquad |d| = n \tag{9-3}$$

Besides the three common definitions mentioned earlier, there are other less common variations [22] of term frequency. In practice, one needs to choose the term frequency definition that is the most suitable to the data and the problem to be solved.

A term frequency vector (shown in Table 9-5) can become very high dimensional because the bag-of-words vector space can grow substantially to include all the words in English. The high dimensionality makes it difficult to store and parse the text and contribute to performance issues related to text analysis.

For the purpose of reducing dimensionality, not all the words from a given language need to be included in the term frequency vector. In English, for example, it is common to remove words such as `the`, `a`, `of`, `and`, `to`, and other articles that are not likely to contribute to semantic understanding. These common words are called ***stop words***. Lists of stop words are available in various languages for automating the identification of stop words. Among them is the *Snowball's stop words list* [23] that contains stop words in more than ten languages.

Another simple yet effective way to reduce dimensionality is to store a term and its frequency only if the term appears at least once in a document. Any term not existing in the term frequency vector by default will have a frequency of 0. Therefore, the previous term frequency vector would be simplified to what is shown in Table 9-6.

TABLE 9-6 *A Simpler Form of the Term Frequency Vector*

Term	Frequency
i	1
love	2
my	1
bphone	1

Some NLP techniques such as lemmatization and stemming can also reduce high dimensionality. Lemmatization and stemming are two different techniques that combine various forms of a word. With these techniques, words such as `play`, `plays`, `played`, and `playing` can be mapped to the same term.

It has been shown that the term frequency is based on the raw count of a term occurring in a stand-alone document. Term frequency by itself suffers a critical problem: It regards that stand-alone document as the entire world. The importance of a term is solely based on its presence in this particular document. Stop words such as `the`, `and`, and `a` could be inappropriately considered the most important because they have the highest frequencies in every document. For example, the top three most frequent words in Shakespeare's *Hamlet* are all stop words (`the`, `and`, and `of`, as shown in Figure 9-2). Besides stop words, words that are more general in meaning tend to appear more often, thus having higher term frequencies. In an article about consumer telecommunications, the word `phone` would be likely to receive a high term frequency. As a result, the important keywords such as `bPhone` and `bEbook` and their related words could appear to be less important. Consider a search engine that responds to a search query and fetches relevant documents. Using term frequency alone, the search engine would not properly assess how relevant each document is in relation to the search query.

A quick fix for the problem is to introduce an additional variable that has a broader view of the world—considering the importance of a term not only in a single document but in a collection of documents, or in a corpus. The additional variable should reduce the effect of the term frequency as the term appears in more documents.

Indeed, that is the intention of the ***inverted document frequency*** (IDF). The IDF inversely corresponds to the ***document frequency*** (DF), which is defined to be the number of documents in the corpus that contain a term. Let a corpus D contain N documents. The document frequency of a term t in corpus $D = \{d_1, d_2, \ldots d_N\}$ is defined as shown in Equation 9-4.

$$DF(t) = \sum_{i=1}^{N} f'(t, d_i) \qquad d_i \in D; |D| = N$$

where

$$f'(t, d') = \begin{cases} 1, & if\ \ t \in d' \\ 0, & otherwise \end{cases} \tag{9-4}$$

The Inverse document frequency of a term t is obtained by dividing N by the document frequency of the term and then taking the logarithm of that quotient, as shown in Equation 9-5.

$$IDF_1(t) = \log \frac{N}{DF(t)} \tag{9-5}$$

If the term is not in the corpus, it leads to a division-by-zero. A quick fix is to add 1 to the denominator, as demonstrated in Equation 9-6.

$$IDF_2(t) = \log \frac{N}{DF(t) + 1} \tag{9-6}$$

The precise base of the logarithm is not material to the ranking of a term. Mathematically, the base constitutes a constant multiplicative factor towards the overall result.

Figure 9-3 shows 50 words with (a) the highest corpus-wide term frequencies (TF), (b) the highest document frequencies (DF), and (c) the highest Inverse document frequencies (IDF) from the news category of the Brown Corpus. Stop words tend to have higher TF and DF because they are likely to appear more often in most documents.

Words with higher IDF tend to be more meaningful over the entire corpus. In other words, the IDF of a rare term would be high, and the IDF of a frequent term would be low. For example, if a corpus contains 1,000 documents, 1,000 of them might contain the word `the`, and 10 of them might contain the word `bPhone`. With Equation 9-5, the IDF of `the` would be 0, and the IDF of `bPhone` would be log100, which is greater than the IDF of `the`. If a corpus consists of mostly phone reviews, the word `phone` would probably have high TF and DF but low IDF.

Despite the fact that IDF encourages words that are more meaningful, it comes with a caveat. Because the total document count of a corpus (N) remains a constant, IDF solely depends on the DF. All words having the same DF value therefore receive the same IDF value. IDF scores words higher that occur less frequently across the documents. Those words that score the lowest DF receive the same highest IDF. In Figure 9-3 (c), for example, `sunbonnet` and `narcotic` appeared in an equal number of documents in the Brown corpus; therefore, they received the same IDF values. In many cases, it is useful to distinguish between two words that appear in an equal number of documents. Methods to further weight words should be considered to refine the IDF score.

The TFIDF (or TF-IDF) is a measure that considers both the prevalence of a term within a document (TF) and the scarcity of the term over the entire corpus (IDF). The TFIDF of a term t in a document d is defined as the term frequency of t in d multiplying the document frequency of t in the corpus as shown in Equation 9-7:

$$TFIDF(t, d) = TF(t, d) \times IDF(t) \tag{9-7}$$

TFIDF scores words higher that appear more often in a document but occur less often across all documents in the corpus. Note that TFIDF applies to a term in a specific document, so the same term is likely to receive different TFIDF scores in different documents (because the TF values may be different).

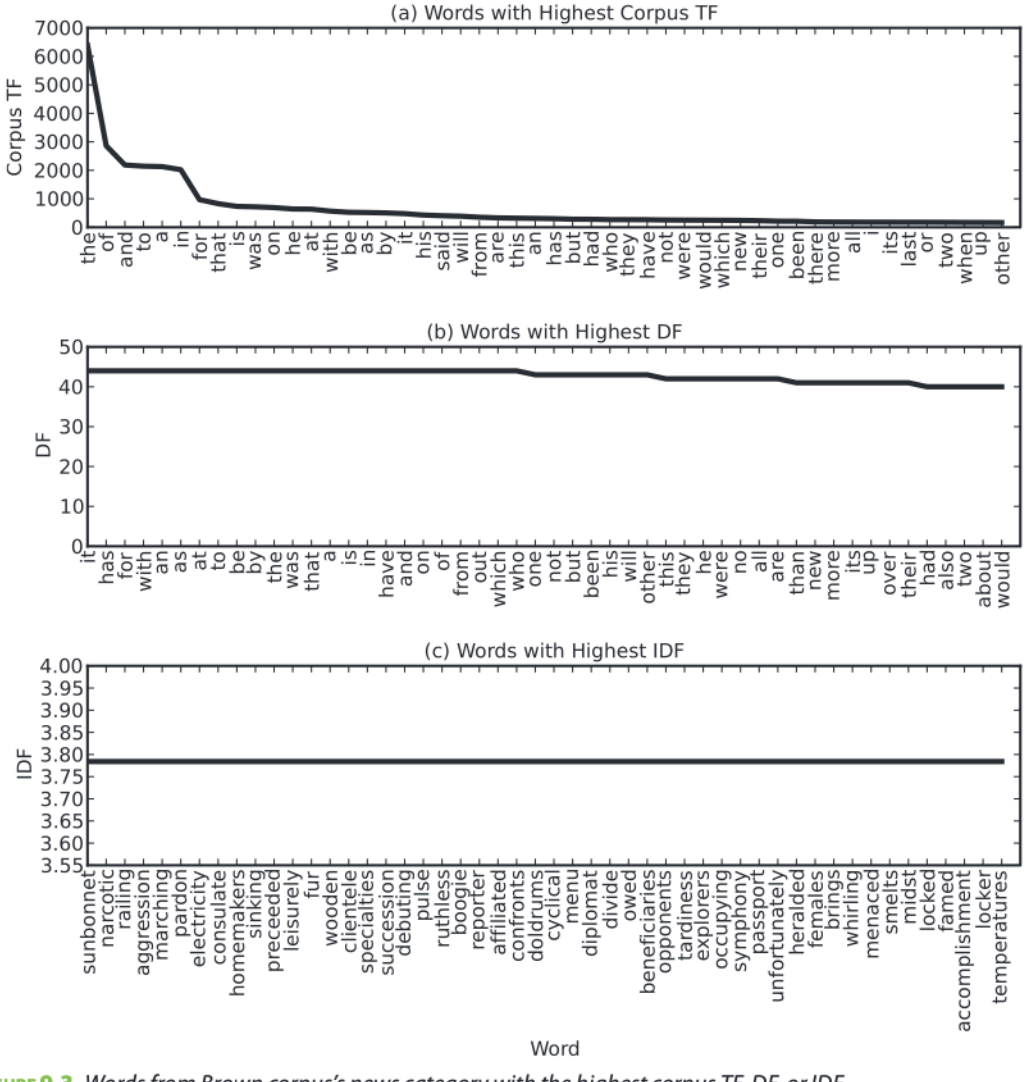


FIGURE 9-3 *Words from Brown corpus's news category with the highest corpus TF, DF, or IDF*

TFIDF is efficient in that the calculations are simple and straightforward, and it does not require knowledge of the underlying meanings of the text. But this approach also reveals little of the inter-document or intra-document statistical structure. The next section shows how topic models can address this shortcoming of TFIDF.

9.6 Categorizing Documents by Topics

With the reviews collected and represented, the data science team at ACME wants to categorize the reviews by topics. As discussed earlier in the chapter, a topic consists of a cluster of words that frequently occur together and share the same theme.

The topics of a document are not as straightforward as they might initially appear. Consider these two reviews:

1. The bPhone5x has coverage everywhere. It's much less flaky than my old bPhone4G.
2. While I love ACME's bPhone series, I've been quite disappointed by the bEbook. The text is illegible, and it makes even my old NBook look blazingly fast.

Is the first review about bPhone5x or bPhone4G? Is the second review about bPhone, bEbook, or NBook? For machines, these questions can be difficult to answer.

Intuitively, if a review is talking about bPhone5x, the term `bPhone5x` and related terms (such as `phone` and `ACME`) are likely to appear frequently. A document typically consists of multiple themes running through the text in different proportions—for example, 30% on a topic related to `phones`, 15% on a topic related to `appearance`, 10% on a topic related to `shipping`, 5% on a topic related to `service`, and so on.

Document grouping can be achieved with clustering methods such as *k*-means clustering [24] or classification methods such as support vector machines [25], *k*-nearest neighbors [26], or naïve Bayes [27]. However, a more feasible and prevalent approach is to use ***topic modeling***. Topic modeling provides tools to automatically organize, search, understand, and summarize from vast amounts of information. ***Topic models*** [28, 29] are statistical models that examine words from a set of documents, determine the themes over the text, and discover how the themes are associated or change over time. The process of topic modeling can be simplified to the following.

1. Uncover the hidden topical patterns within a corpus.
2. Annotate documents according to these topics.
3. Use annotations to organize, search, and summarize texts.

A ***topic*** is formally defined as a distribution over a fixed vocabulary of words [29]. Different topics would have different distributions over the same vocabulary. A topic can be viewed as a cluster of words with related meanings, and each word has a corresponding weight inside this topic. Note that a word from the vocabulary can reside in multiple topics with different weights. Topic models do not necessarily require prior knowledge of the texts. The topics can emerge solely based on analyzing the text.

The simplest topic model is ***latent Dirichlet allocation*** (LDA) [29], a generative probabilistic model of a corpus proposed by David M. Blei and two other researchers. In generative probabilistic modeling, data

is treated as the result of a generative process that includes hidden variables. LDA assumes that there is a fixed vocabulary of words, and the number of the latent topics is predefined and remains constant. LDA assumes that each latent topic follows a Dirichlet distribution [30] over the vocabulary, and each document is represented as a random mixture of latent topics.

Figure 9-4 illustrates the intuitions behind LDA. The left side of the figure shows four topics built from a corpus, where each topic contains a list of the most important words from the vocabulary. The four example topics are related to problem, policy, neural, and report. For each document, a distribution over the topics is chosen, as shown in the histogram on the right. Next, a topic assignment is picked for each word in the document, and the word from the corresponding topic (colored discs) is chosen. In reality, only the documents (as shown in the middle of the figure) are available. The goal of LDA is to infer the underlying topics, topic proportions, and topic assignments for every document.

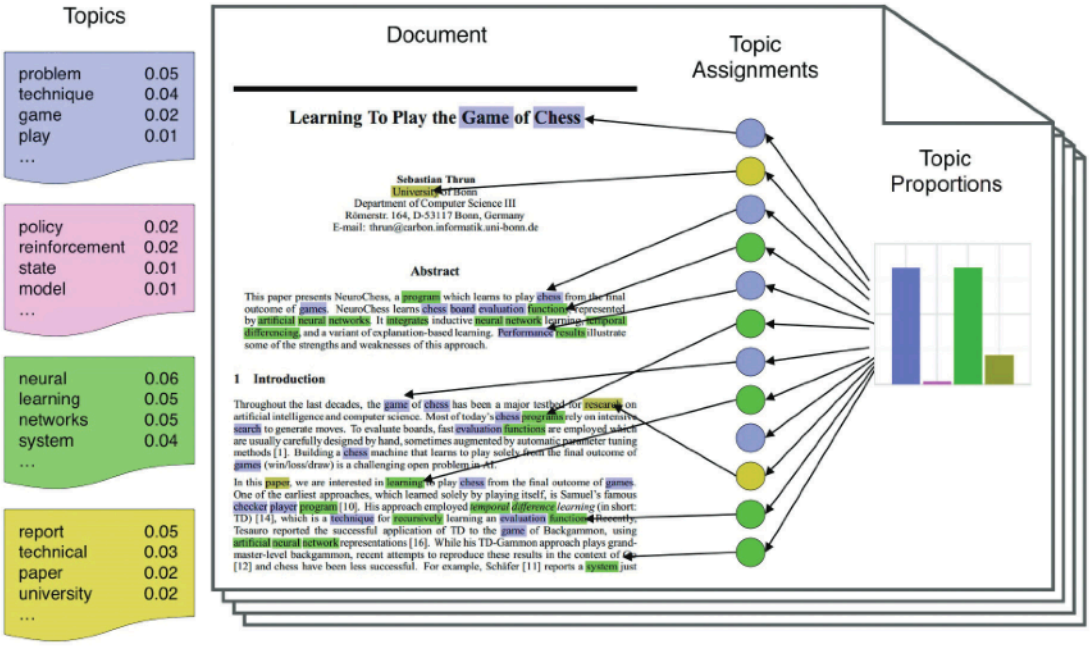


FIGURE 9-4 *The intuitions behind LDA*

The reader can refer to the original paper [29] for the mathematical detail of LDA. Basically, LDA can be viewed as a case of hierarchical Bayesian estimation with a posterior distribution to group data such as documents with similar topics.

Many programming tools provide software packages that can perform LDA over datasets. R comes with an `lda` package [31] that has built-in functions and sample datasets. The `lda` package was developed by David M. Blei's research group [32]. Figure 9-5 shows the distributions of ten topics on nine scientific documents randomly drawn from the `cora` dataset of the `lda` package. The `cora` dataset is a collection of 2,410 scientific documents extracted from the Cora search engine [33].

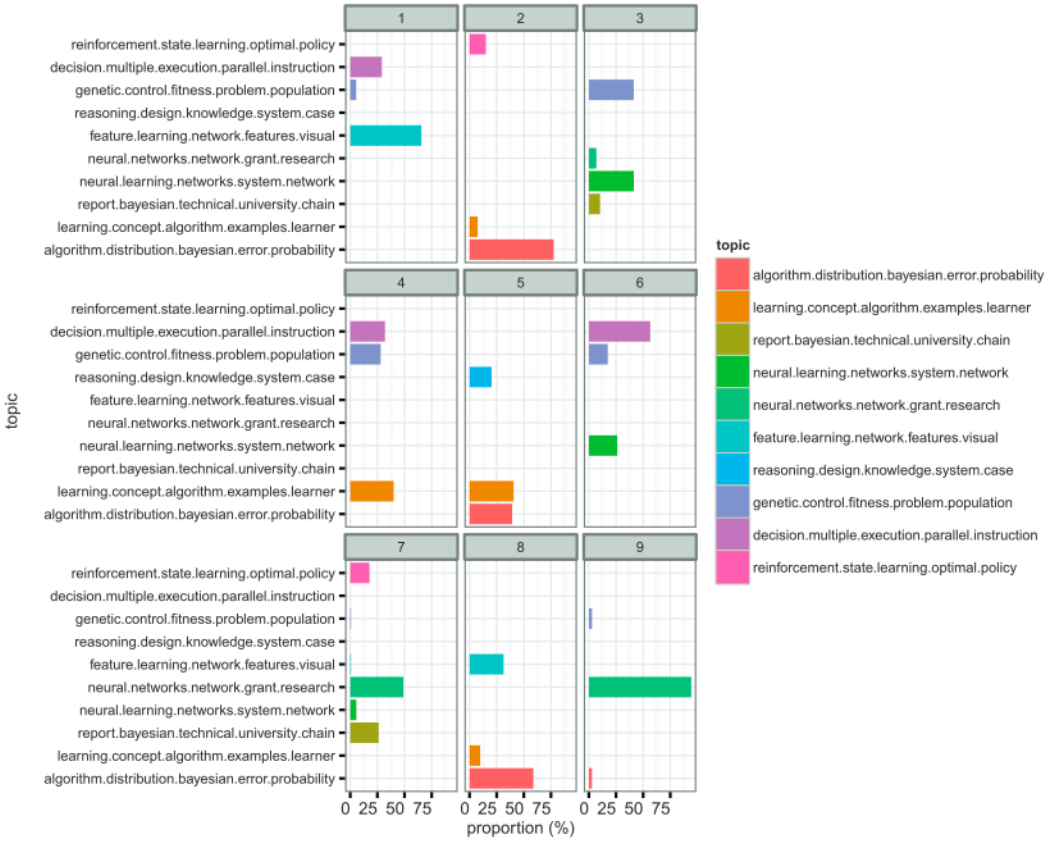


FIGURE 9-5 *Distributions of ten topics over nine scientific documents from the Cora dataset*

The code that follows shows how to generate a graph similar to Figure 9-5 using R and add-on packages such as `lda` and `ggplot`.

```
require("ggplot2")
require("reshape2")
require("lda")

# load documents and vocabulary
data(cora.documents)
data(cora.vocab)

theme_set(theme_bw())

# Number of topic clusters to display
K <- 10

# Number of documents to display
N <- 9
```

```
result <- lda.collapsed.gibbs.sampler(cora.documents,
                                      K,  ## Num clusters
                                      cora.vocab,
                                      25,  ## Num iterations
                                      0.1,
                                      0.1,
                                      compute.log.likelihood=TRUE)

# Get the top words in the cluster
top.words <- top.topic.words(result$topics, 5, by.score=TRUE)

# build topic proportions
topic.props <- t(result$document_sums) / colSums(result$document_sums)

document.samples <- sample(1:dim(topic.props)[1], N)
topic.props <- topic.props[document.samples,]

topic.props[is.na(topic.props)] <-  1 / K

colnames(topic.props) <- apply(top.words, 2, paste, collapse=" ")

topic.props.df <- melt(cbind(data.frame(topic.props),
                             document=factor(1:N)),
                       variable.name="topic",
                       id.vars = "document")

qplot(topic, value*100, fill=topic, stat="identity",
      ylab="proportion (%)", data=topic.props.df,
      geom="histogram") +
  theme(axis.text.x = element_text(angle=0, hjust=1, size=12)) +
  coord_flip() +
  facet_wrap(~ document, ncol=3)
```

Topic models can be used in document modeling, document classification, and collaborative filtering [29]. Topic models not only can be applied to textual data, they can also help annotate images. Just as a document can be considered a collection of topics, images can be considered a collection of image features.

9.7 Determining Sentiments

In addition to the TFIDF and topic models, the Data Science team may want to identify the sentiments in user comments and reviews of the ACME products. ***Sentiment analysis*** refers to a group of tasks that use statistics and natural language processing to mine opinions to identify and extract subjective information from texts.

Early work on sentiment analysis focused on detecting the polarity of product reviews from Epinions [34] and movie reviews from the Internet Movie Database (IMDb) [35] at the document level. Later work handles sentiment analysis at the sentence level [36]. More recently, the focus has shifted to phrase-level [37] and short-text forms in response to the popularity of micro-blogging services such as Twitter [38, 39, 40, 41, 42].

Intuitively, to conduct sentiment analysis, one can manually construct lists of words with positive sentiments (such as *brilliant*, *awesome*, and *spectacular*) and negative sentiments (such as *awful*, *stupid*, and *hideous*). Related work has pointed out that such an approach can be expected to achieve accuracy around 60% [35], and it is likely to be outperformed by examination of corpus statistics [43].

Classification methods such as naïve Bayes as introduced in Chapter 7, maximum entropy (MaxEnt), and support vector machines (SVM) are often used to extract corpus statistics for sentiment analysis. Related research has found out that these classifiers can score around 80% accuracy [35, 41, 42] on sentiment analysis over unstructured data. One or more of such classifiers can be applied to unstructured data, such as movie reviews or even tweets.

The movie review corpus by Pang et al. [35] includes 2,000 movie reviews collected from an IMDb archive of the rec.arts.movies.reviews newsgroup [43]. These movie reviews have been manually tagged into 1,000 positive reviews and 1,000 negative reviews.

Depending on the classifier, the data may need to be split into training and testing sets. As seen previously in Chapter 7, a useful rule of the thumb for splitting data is to produce a training set much bigger than the testing set. For example, an 80/20 split would produce 80% of the data as the training set and 20% as the testing set.

Next, one or more classifiers are trained over the training set to learn the characteristics or patterns residing in the data. The sentiment tags in the testing data are hidden away from the classifiers. After the training, classifiers are tested over the testing set to infer the sentiment tags. Finally, the result is compared against the original sentiment tags to evaluate the overall performance of the classifier.

The code that follows is written in Python using the Natural Language Processing Toolkit (NLTK) library (`http://nltk.org/`). It shows how to perform sentiment analysis using the naïve Bayes classifier over the movie review corpus.

The code splits the 2,000 reviews into 1,600 reviews as the training set and 400 reviews as the testing set. The naïve Bayes classifier learns from the training set. The sentiments in the testing set are hidden away from the classifier. For each review in the training set, the classifier learns how each feature impacts the outcome sentiment. Next, the classifier is given the testing set. For each review in the set, it predicts what the corresponding sentiment should be, given the features in the current review.

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews
from collections import defaultdict
import numpy as np

# define an 80/20 split for train/test
SPLIT = 0.8

def word_feats(words):
    feats = defaultdict(lambda: False)
    for word in words:
        feats[word] = True
    return feats

posids = movie_reviews.fileids('pos')
```

```
negids = movie_reviews.fileids('neg')

posfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'pos')
            for f in posids]
negfeats = [(word_feats(movie_reviews.words(fileids=[f])), 'neg')
            for f in negids]

cutoff = int(len(posfeats) * SPLIT)

trainfeats = negfeats[:cutoff] + posfeats[:cutoff]
testfeats = negfeats[cutoff:] + posfeats[cutoff:]

print 'Train on %d instances\nTest on %d instances' % (len(trainfeats),
                                                        len(testfeats))

classifier = NaiveBayesClassifier.train(trainfeats)
print 'Accuracy:', nltk.classify.util.accuracy(classifier, testfeats)

classifier.show_most_informative_features()

# prepare confusion matrix

pos = [classifier.classify(fs) for (fs,l) in posfeats[cutoff:]]
pos = np.array(pos)
neg = [classifier.classify(fs) for (fs,l) in negfeats[cutoff:]]
neg = np.array(neg)

print 'Confusion matrix:'
print '\t'*2, 'Predicted class'
print '-'*40
print '|\t %d (TP) \t|\t %d (FN) \t| Actual class' % (
          (pos == 'pos').sum(), (pos == 'neg').sum()
print '-'*40
print '|\t %d (FP) \t|\t %d (TN) \t|' % (
          (neg == 'pos').sum(), (neg == 'neg').sum())
print '-'*40
```

The output that follows shows that the naïve Bayes classifier is trained on 1,600 instances and tested on 400 instances from the movie corpus. The classifier achieves an accuracy of 73.5%. Most information features for positive reviews from the corpus include words such as `outstanding`, `vulnerable`, and `astounding`; and words such as `insulting`, `ludicrous`, and `uninvolving` are the most informative features for negative reviews. At the end, the output also shows the confusion matrix corresponding to the classifier to further evaluate the performance.

```
Train on 1600 instances
Test on 400 instances
Accuracy: 0.735
Most Informative Features
```

```
            outstanding = True          pos : neg    =     13.9 : 1.0
              insulting = True          neg : pos    =     13.7 : 1.0
             vulnerable = True          pos : neg    =     13.0 : 1.0
              ludicrous = True          neg : pos    =     12.6 : 1.0
            uninvolving = True          neg : pos    =     12.3 : 1.0
             astounding = True          pos : neg    =     11.7 : 1.0
                 avoids = True          pos : neg    =     11.7 : 1.0
            fascination = True          pos : neg    =     11.0 : 1.0
              animators = True          pos : neg    =     10.3 : 1.0
                 symbol = True          pos : neg    =     10.3 : 1.0
Confusion matrix:
              Predicted class
---------------------------------------
|       195 (TP)      |        5 (FN)      | Actual class
---------------------------------------
|       101 (FP)      |       99 (TN)      |
---------------------------------------
```

As discussed earlier in Chapter 7, a ***confusion matrix*** is a specific table layout that allows visualization of the performance of a model over the testing set. Every row and column corresponds to a possible class in the dataset. Each cell in the matrix shows the number of test examples for which the actual class is the row and the predicted class is the column. Good results correspond to large numbers down the main diagonal (TP and TN) and small, ideally zero, off-diagonal elements (FP and FN). Table 9-7 shows the confusion matrix from the previous program output for the testing set of 400 reviews. Because a well-performed classifier should have a confusion matrix with large numbers for TP and TN and ideally near zero numbers for FP and FN, it can be concluded that the naïve Bayes classifier has many false negatives, and it does not perform very well on this testing set.

TABLE 9-7 *Confusion Matrix for the Example Testing Set*

		Predicted Class	
		Positive	Negative
Actual Class	Positive	195 (TP)	5 (FN)
	Negative	101 (FP)	99 (TN)

Chapter 7 has introduced a few measures to evaluate the performance of a classifier beyond the confusion matrix. Precision and recall are two measures commonly used to evaluate tasks related to text analysis. Definitions of precision and recall are given in Equations 9-8 and 9-9.

$$Precision = \frac{TP}{TP + FP} \tag{9-8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9-9}$$

Precision is defined as the percentage of documents in the results that are relevant. If by entering keyword `bPhone`, the search engine returns 100 documents, and 70 of them are relevant, the precision of the search engine result is 0.7%.

Recall is the percentage of returned documents among all relevant documents in the corpus. If by entering keyword `bPhone`, the search engine returns 100 documents, only 70 of which are relevant while failing to return 10 additional, relevant documents, the recall is $70 / (70 + 10) = 0.875$.

Therefore, the naïve Bayes classifier from Table 9-7 receives a recall of $195 / (195 + 5) = 0.975$ and a precision of $195/(195 + 101) \approx 0.659$.

Precision and recall are important concepts, whether the task is about information retrieval of a search engine or text analysis over a finite corpus. A good classifier ideally should achieve both precision and recall close to 1.0. In information retrieval, a perfect precision score of 1.0 means that every result retrieved by a search was relevant (but says nothing about whether all relevant documents were retrieved), whereas a perfect recall score of 1.0 means that all relevant documents were retrieved by the search (but says nothing about how many irrelevant documents were also retrieved). Both precision and recall are therefore based on an understanding and measure of relevance. In reality, it is difficult for a classifier to achieve both high precision and high recall. For the example in Table 9-7, the naïve Bayes classifier has a high recall but a low precision. Therefore, the Data Science team needs to check the cleanliness of the data, optimize the classifier, and find if there are ways to improve the precision while retaining the high recall.

Classifiers determine sentiments solely based on the datasets on which they are trained. The domain of the datasets and the characteristics of the features determine what the knowledge classifiers can learn. For example, *lightweight* is a positive feature for reviews on laptops but not necessarily for reviews on wheelbarrows or textbooks. In addition, the training and the testing sets should share similar traits for classifiers to perform well. For example, classifiers trained on movie reviews generally should not be tested on tweets or blog comments.

Note that an absolute sentiment level is not necessarily very informative. Instead, a baseline should be established and then compared against the latest observed values. For example, a ratio of 40% positive tweets on a topic versus 60% negative might not be considered a sign that a product is unsuccessful if other similar successful products have a similar ratio based on the psychology of when people tweet.

The previous example demonstrates how to use naïve Bayes to perform sentiment analysis. The example can be applied to tweets on ACME's `bPhone` and `bEbook` simply by replacing the movie review corpus with the pretagged tweets. Other classifiers can also be used in place of naïve Bayes.

The movie review corpus contains only 2,000 reviews; therefore, it is relatively easy to manually tag each review. For sentiment analysis based on larger amounts of streaming data such as millions or billions of tweets, it is less feasible to collect and construct datasets of tweets that are big enough or manually tag each of the tweets to train and test one or more classifiers. There are two popular ways to cope with this problem. The first way to construct pretagged data, as illustrated in recent work by Go et al. [41] and Pak and Paroubek [42], is to apply supervision and use emoticons such as `:)` and `:(` to indicate if a tweet contains positive or negative sentiments. Words from these tweets can in turn be used as clues to classify the sentiments of future tweets. Go et al. [41] use classification methods including naïve Bayes, MaxEnt, and SVM over the training and testing datasets to perform sentiment classifications. Their demo is available at `http://www.sentiment140.com`. Figure 9-6 shows the sentiments resulting from a query against the term "Boston weather" on a set of tweets. Viewers can mark the result as accurate or inaccurate, and such feedback can be incorporated in future training of the algorithm.

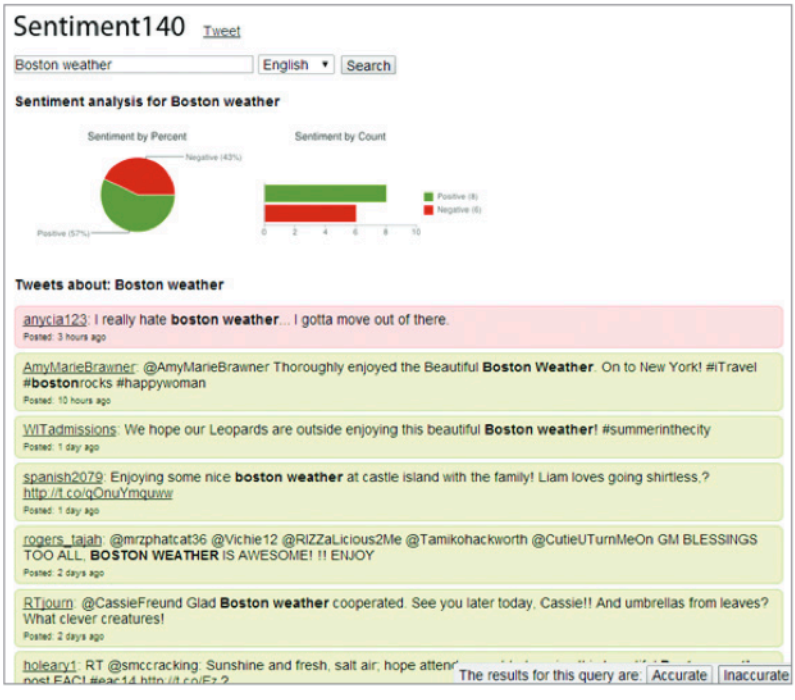


FIGURE 9-6 *Sentiment140 [41], an online tool for Twitter sentiment analysis*

Emoticons make it easy and fast to detect sentiments of millions or billions of tweets. However, using emoticons as the sole indicator of sentiments sometimes can be misleading, as emoticons may not necessarily correspond to the sentiments in the accompanied text. For example, the sample tweet shown in Figure 9-7 contains the :) emoticon, but the text does not express a positive sentiment.

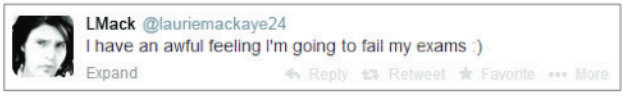


FIGURE 9-7 *Tweet with the :) emoticon does not necessarily correspond to a positive sentiment*

To address this problem, related research usually uses Amazon Mechanical Turk (MTurk) [44] to collect human-tagged reviews. MTurk is a crowdsourcing Internet marketplace that enables individuals or businesses to coordinate the use of human intelligence to perform tasks that are difficult for computers to do. In many cases, MTurk has been shown to collect human input much faster compared to traditional channels such as door-to-door surveys. For the example sentiment analysis task, the Data Science team can publish the tweets collected from Section 9.3 to MTurk as Human Intelligence Tasks (HITs). The team can then ask human workers to tag each tweet as positive, neutral, or negative. The result can be used to train one or more classifiers or test the performances of classifiers. Figure 9-8 shows a sample task on MTurk related to sentiment analysis.

FIGURE 9-8 *Amazon Mechanical Turk*

9.8 Gaining Insights

So far this chapter has discussed several text analysis tasks including text collection, text representation, TFIDF, topic models, and sentiment analysis. This section shows how ACME uses these techniques to gain insights into customer opinions about its products. To keep the example simple, this section only uses `bPhone` to illustrate the steps.

Corresponding to the data collection phase, the Data Science team has used `bPhone` as the keyword to collect more than 300 reviews from a popular technical review website.

The 300 reviews are visualized as a word cloud after removing stop words. A ***word cloud*** (or ***tag cloud***) is a visual representation of textual data. Tags are generally single words, and the importance of each word is shown with font size or color. Figure 9-9 shows the word cloud built from the 300 reviews. The reviews have been previously case folded and tokenized into lowercased words, and stop words have been removed from the text. A more frequently appearing word in Figure 9-9 is shown with a larger font size. The orientation of each word is only for the aesthetical purpose. Most of the graph is taken up by the words `phone` and `bphone`, which occur frequently but are not very informative. Overall, the graph reveals little information. The team needs to conduct further analyses on the data.

FIGURE 9-9 *Word cloud on all 300 reviews on bPhone*

Fortunately, the popular technical review website allows users to provide ratings on a scale from one to five when they post reviews. The team can divide the reviews into subgroups using those ratings.

To reveal more information, the team can remove words such as `phone`, `bPhone`, and `ACME`, which are not very useful for the study. Related research often refers to these words as ***domain-specific stop words***. Figure 9-10 shows the word cloud corresponding to 50 five-star reviews extracted from the data. Note that the shades of gray are only for the aesthetical purpose. The result suggests that customers are satisfied with the `seller`, the `brand`, and the `product`, and they `recommend` bPhone to their friends and families.

Figure 9-11 shows the word cloud of 70 one-star reviews. The words `sim` and `button` occur frequently enough that it would be advisable to sample the reviews that contain these terms and determine what is being said about buttons and SIM cards. Word clouds can reveal useful information beyond the most prominent terms. For example, the graph in Figure 9-11 oddly contains words like `stolen` and `Venezuela`. As the Data Science team investigates the stories behind these words, it finds that these words appear in 1-star reviews because there are a few unauthorized sellers from Venezuela that sell stolen

bPhones. ACME can take further actions from this point. This is an example of how text analysis and even simple visualizations can help gain insights.

FIGURE 9-10 *Word cloud on five-star reviews*

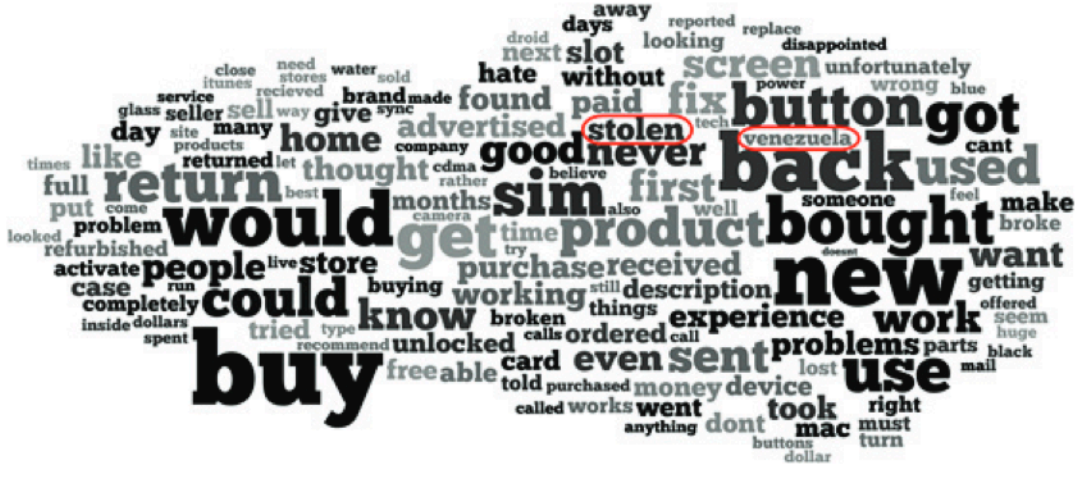


FIGURE 9-11 *Word cloud on one-star reviews*

TFIDF can be used to highlight the informative words in the reviews. Figure 9-12 shows a subset of the reviews in which each word with a larger font size corresponds to a higher TFIDF value. Each review is considered a document. With TFIDF, data analysts can quickly go through the reviews and identify what aspects are perceived to make bPhone a good product or a bad product.

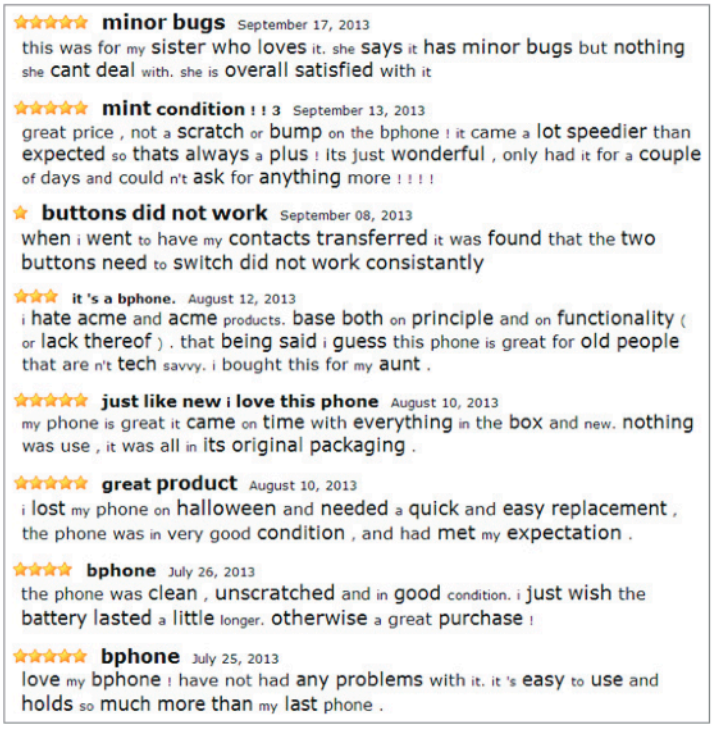


FIGURE 9-12 *Reviews highlighted by TFIDF values*

Topic models such as LDA can categorize the reviews into topics. Figures 9-13 and 9-14 show circular graphs of topics as results of the LDA. These figures are produced with tools and technologies such as Python, NoSQL, and D3.js. Figure 9-13 visualizes ten topics built from the five-star reviews. Each topic focuses on a different aspect that can characterize the reviews. The disc size represents the weight of a word. In an interactive environment, hovering the mouse over a topic displays the full words and their corresponding weights.

Figure 9-14 visualizes ten topics from one-star reviews. For example, the bottom-right topic contains words such as `button`, `power`, and `broken`, which may indicate that bPhone has problems related to button and power supply. The Data Science team can track down these reviews and find out if that's really the case.

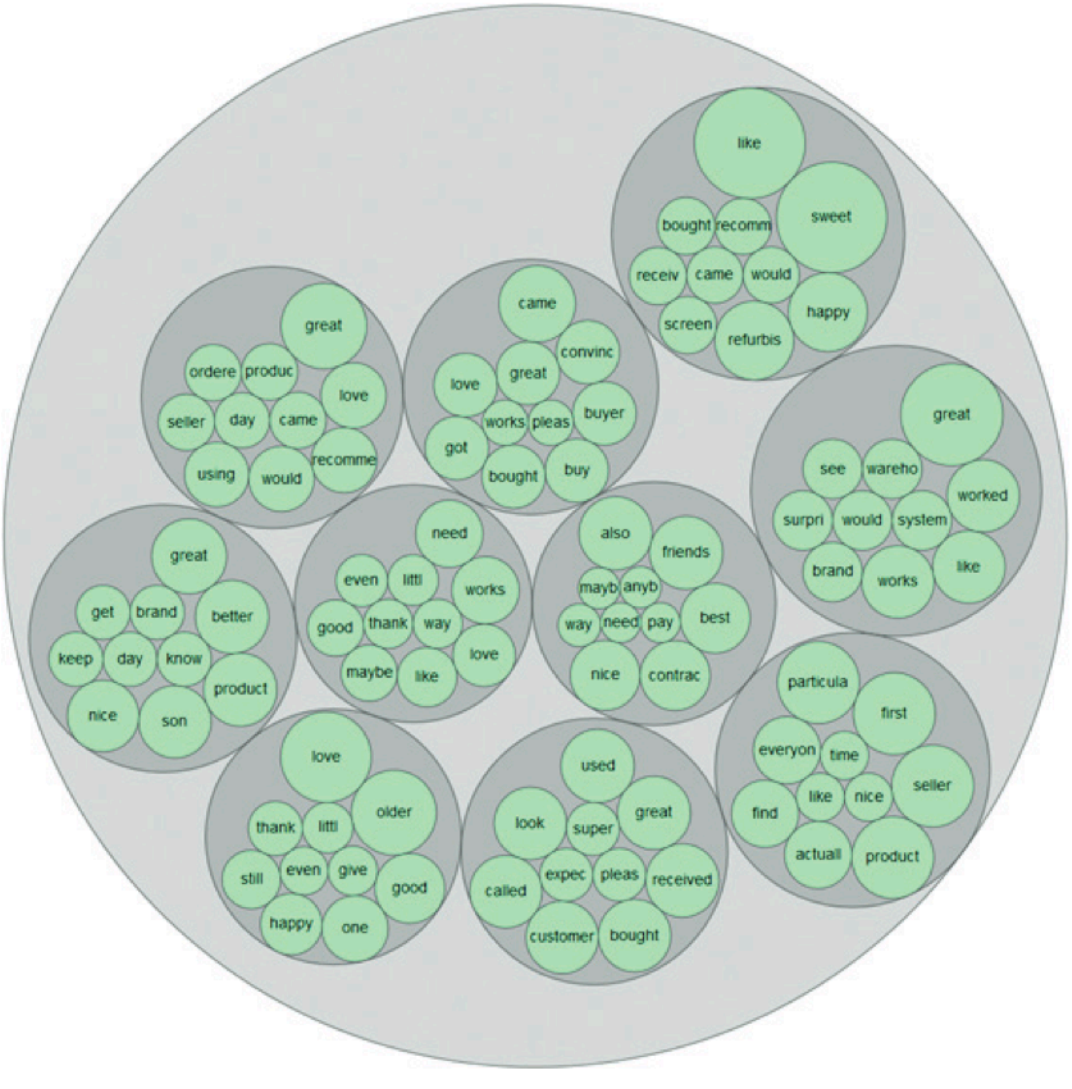


FIGURE 9-13 *Ten topics on five-star reviews*

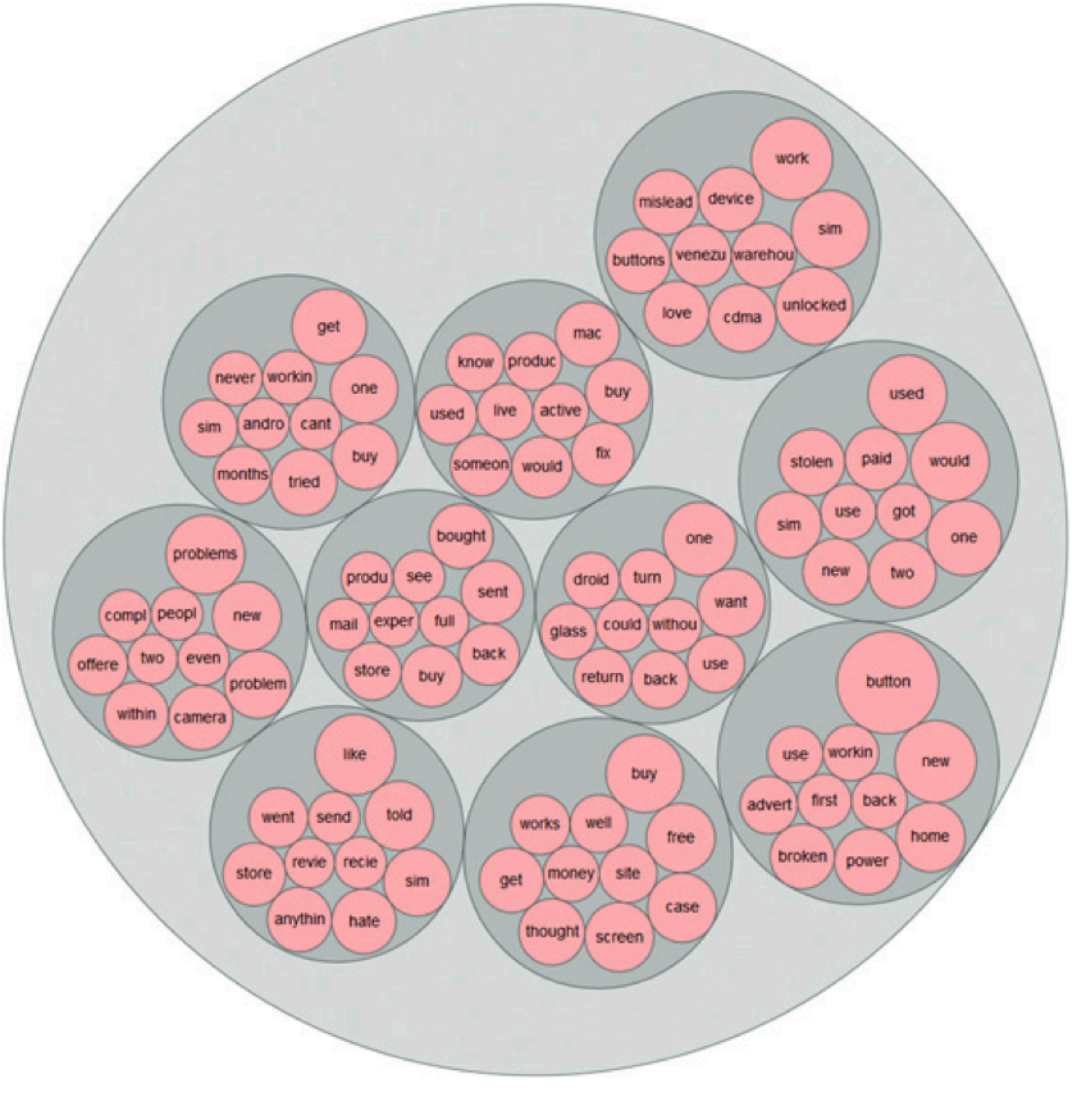


FIGURE 9-14 *Ten topics on one-star reviews*

Figure 9-15 provides a different way to visualize the topics. Five topics are extracted from five-star reviews and one-star reviews, respectively. In an interactive environment, hovering the mouse on a topic highlights the corresponding words in this topic. The screenshots in Figure 9-15 were taken when `Topic 4` is highlighted for both groups. The weight of a word in a topic is indicated by the disc size.

FIGURE 9-15 *Five topics on five-star reviews (left) and 1-star reviews (right)*

The Data Science team has also conducted sentiment analysis over 100 tweets from the popular micro-blogging site Twitter. The result is shown in Figure 9-16. The left side represents negative sentiments, and the right side represents positive sentiments. Vertically, the tweets have been randomly placed for aesthetic purposes. Each tweet is shown as a disc, where the size represents the number of followers of the user who made the original tweet. The color shade of a disc represents how frequently this tweet has been retweeted. The figure indicates that most customers are satisfied with ACME's bPhone.

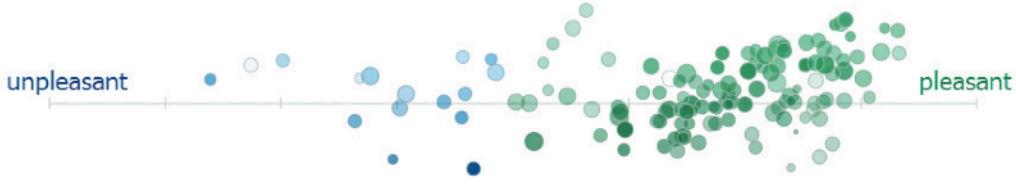


FIGURE 9-16 *Sentiment analysis on Tweets related to bPhone*

Summary

This chapter has discussed several subtasks of text analysis, including parsing, search and retrieval, and text mining. With a brand management example, the chapter talks about a typical text analysis process: (1) collecting raw text, (2) representing text, (3) using TFIDF to compute the usefulness of each word in the texts, (4) categorizing documents by topics using topic modeling, (5) sentiment analysis, and (6) gaining greater insights.

Overall text analysis is no trivial task. Corresponding to the Data Analytic Lifecycle, the most time-consuming parts of a text analysis project often are not performing the statistics or implementing algorithms. Chances are the team would spend most of the time formulating the problem, getting the data, and preparing the data.

Exercises

1. What are the main challenges of text analysis?
2. What is a corpus?
3. What are common words (such as *a, and, of*) called?
4. Why can't we use TF alone to measure the usefulness of the words?
5. What is a caveat of IDF? How does TFIDF address the problem?
6. Name three benefits of using the TFIDF.
7. What methods can be used for sentiment analysis?
8. What is the definition of *topic* in topic models?
9. Explain the trade-offs for precision and recall.
10. Perform LDA topic modeling on the Reuters-21578 corpus using Python and LDA. The NLTK has already come with the Reuters-21578 corpus. To import this corpus, enter the following comment in the Python prompt:

```
from nltk.corpus import reuters
```

The LDA has already been implemented by several Python libraries such as `gensim` [45]. Either use one such library or implement your own LDA to perform topic modeling on the Reuters-21578 corpus.

11. Choose a topic of your interest, such as a movie, a celebrity, or any buzz word. Then collect 100 tweets related to this topic. Hand-tag them as positive, neutral, or negative. Next, split them into 80 tweets as the training set and the remaining 20 as the testing set. Run one or more classifiers over these tweets to perform sentiment analysis. What are the precision and recall of these classifiers? Which classifier performs better than the others?

Bibliography

[1] Dr. Seuss, "Green Eggs and Ham," New York, NY, USA, Random House, 1960.

[2] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," ***KDD Workshop on Text Mining,*** 2000.

[3] "The Penn Treebank Project," University of Pennsylvania [Online]. Available: `http://www.cis.upenn.edu/~treebank/home.html`. [Accessed 26 March 2014].

[4] Wikipedia, "List of Open APIs" [Online]. Available: `http://en.wikipedia.org/wiki/List_of_open_APIs`. [Accessed 27 March 2014].

[5] ProgrammableWeb, "API Directory" [Online]. Available: `http://www.programmableweb.com/apis/directory`. [Accessed 27 March 2014].

[6] Twitter, "Twitter Developers Site" [Online]. Available: `https://dev.twitter.com/`. [Accessed 27 March 2014].

[7] "Curl and libcurl Tools" [Online]. Available: `http://curl.haxx.se/`. [Accessed 27 March 2014].

[8] "XML Path Language (XPath) 2.0," World Wide Web Consortium, 14 December 2010. [Online]. Available: `http://www.w3.org/TR/xpath20/`. [Accessed 27 March 2014].

[9] "Gnip: The Source for Social Data," GNIP [Online]. Available: `http://gnip.com/`. [Accessed 12 June 2014].

[10] "DataSift: Power Decisions with Social Data," DataSift [Online]. Available: `http://datasift.com/`. [Accessed 12 June 2014].

[11] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," in *Information Processing and Management*, 1988, pp. 513–523.

[12] G. K. Zipf, *Human Behavior and the Principle of Least Effort*, Reading, MA: Addison-Wesley, 1949.

[13] M. E. Newman, "Power Laws, Pareto Distributions, and Zipf's Law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, 2005.

[14] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett, "Sentence Similarity Based on Semantic Nets and Corpus Statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138–1150, 2006.

[15] W. N. Francis and H. Kucera, "Brown Corpus Manual," 1979. [Online]. Available: `http://icame.uib.no/brown/bcm.html`.

[16] "Critical Assessment of Information Extraction in Biology (BioCreative)" [Online]. Available: `http://www.biocreative.org/`. [Accessed 2 April 2014].

[17] J. J. Godfrey and E. Holliman, "Switchboard-1 Release 2," Linguistic Data Consortium, Philadelphia, 1997. [Online]. Available: `http://catalog.ldc.upenn.edu/LDC97S62`. [Accessed 2 April 2014].

[18] P. Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation," *MT Summit,* 2005.
[19] N. Seco, T. Veale, and J. Hayes, "An Intrinsic Information Content Metric for Semantic Similarity in WordNet," *ECAI,* vol. 16, pp. 1089–1090, 2004.
[20] P. Resnik, "Using Information Content to Evaluate Semantic Similarity in a Taxonomy," In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95),* vol. 1, pp. 448–453, 1995.
[21] T. Pedersen, "Information Content Measures of Semantic Similarity Perform Better Without Sense-Tagged Text," *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics,* pp. 329–332, June 2010.
[22] C. D. Manning, P. Raghavan, and H. Schütze, "Document and Query Weighting Schemes," in *Introduction to Information Retrieval,* Cambridge, United Kingdom, Cambridge University Press, 2008, p. 128.
[23] M. Porter, "Porter's English Stop Word List," 12 February 2007. [Online]. Available: http://snowball.tartarus.org/algorithms/english/stop.txt. [Accessed 2 April 2014].
[24] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," *KDD workshop on text mining,* vol. 400, no. 1, 2000.
[25] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," *ICML,* vol. 99, pp. 200–209, 1999.
[26] P. Soucy and G. W. Mineau, "A Simple KNN Algorithm for Text Categorization," *ICDM,* pp. 647–648, 2001.
[27] B. Liu, X. Li, W. S. Lee, and P. S. Yu, "Text Classification by Labeling Words," *AAAI,* vol. 4, pp. 425–430, 2004.
[28] D. M. Blei, "Probabilistic Topic Models," *Communications of the ACM,* vol. 55, no. 4, pp. 77–84, 2012.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research,* vol. 3, pp. 993–1022, 2003.
[30] T. Minka, "Estimating a Dirichlet Distribution," 2000.
[31] J. Chang, "lda: Collapsed Gibbs Sampling Methods for Topic Models," *CRAN,* 14 October 2012. [Online]. Available: `http://cran.r-project.org/web/packages/lda/`. [Accessed 3 April 2014].
[32] D. M. Blei, "Topic Modeling Software" [Online]. Available: http://www.cs.princeton.edu/~blei/topicmodeling.html. [Accessed 11 June 2014].
[33] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "A Machine Learning Approach to Building Domain-Specific Search Engines," *IJCAI,* vol. 99, 1999.
[34] P. D. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," *Proceedings of the Association for Computational Linguistics,* pp. 417–424, 2002.
[35] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs Up? Sentiment Classification Using Machine Learning Techniques," *Proceedings of EMNLP,* pp. 79–86, 2002.
[36] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 168–177, 2004.
[37] A. Agarwal, F. Biadsy, and K. R. Mckeown, "Contextual Phrase-Level Polarity Analysis Using Lexical Affect Scoring and Syntactic N-Grams," *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics,* pp. 24–32, 2009.

[38] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, "From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series," *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM '10,* pp. 122–129, 2010.

[39] A. Agarwal, B. Xie, I. Vovsha, O. Rambow and R. Passonneau, "Sentiment Analysis of Twitter Data," *In Proceedings of the Workshop on Languages in Social Media,* pp. 30–38, 2011.

[40] H. Saif, Y. He, and H. Alani, "Semantic Sentiment Analysis of Twitter," *Proceedings of the 11th International Conference on The Semantic Web (ISWC'12),* pp. 508–524, 2012.

[41] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification Using Distant Supervision," *CS224N Project Report, Stanford,* pp. 1–12, 2009.

[42] A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10),* pp. 19–21, 2010.

[43] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval,* vol. 2, no. 1–2, pp. 1–135, 2008.

[44] "Amazon Mechanical Turk" [Online]. Available: http://www.mturk.com/. [Accessed 7 April 2014].

[45] R. Řehůřek, "Python Gensim Library" [Online]. Available: http://radimrehurek.com/gensim/. [Accessed 8 April 2014].

10

Advanced Analytics—Technology and Tools: MapReduce and Hadoop

Key Concepts

Hadoop
Hadoop Ecosystem
MapReduce
NoSQL

Chapter 4, "Advanced Analytical Theory and Methods: Clustering," through Chapter 9, "Advanced Analytical Theory and Methods: Text Analysis," covered several useful analytical methods to classify, predict, and examine relationships within the data. This chapter and Chapter 11, "Advanced Analytics—Technology and Tools: In-Database Analytics," address several aspects of collecting, storing, and processing unstructured and structured data, respectively. This chapter presents some key technologies and tools related to the Apache Hadoop software library, "a framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models" [1].

This chapter focuses on how Hadoop stores data in a distributed system and how Hadoop implements a simple programming paradigm known as MapReduce. Although this chapter makes some Java-specific references, the only intended prerequisite knowledge is a basic understanding of programming. Furthermore, the Java-specific details of writing a MapReduce program for Apache Hadoop are beyond the scope of this text. This omission may appear troublesome, but tools in the Hadoop ecosystem, such as Apache Pig and Apache Hive, can often eliminate the need to explicitly code a MapReduce program. Along with other Hadoop-related tools, Pig and Hive are covered in a portion of this chapter dealing with the Hadoop ecosystem.

To illustrate the power of Hadoop in handling unstructured data, the following discussion provides several Hadoop use cases.

10.1 Analytics for Unstructured Data

Prior to conducting data analysis, the required data must be collected and processed to extract the useful information. The degree of initial processing and data preparation depends on the volume of data, as well as how straightforward it is to understand the structure of the data.

Recall the four types of data structures discussed in Chapter 1, "Introduction to Big Data Analytics":

- **Structured:** A specific and consistent format (for example, a data table)
- **Semi-structured:** A self-describing format (for example, an XML file)
- **Quasi-structured:** A somewhat inconsistent format (for example, a hyperlink)
- **Unstructured:** An inconsistent format (for example, text or video)

Structured data, such as relational database management system (RDBMS) tables, is typically the easiest data format to interpret. However, in practice it is still necessary to understand the various values that may appear in a certain column and what these values represent in different situations (based, for example, on the contents of the other columns for the same record). Also, some columns may contain unstructured text or stored objects, such as pictures or videos. Although the tools presented in this chapter focus on unstructured data, these tools can also be utilized for more structured datasets.

10.1.1 Use Cases

The following material provides several use cases for MapReduce. The MapReduce paradigm offers the means to break a large task into smaller tasks, run tasks in parallel, and consolidate the outputs of the individual tasks into the final output. Apache Hadoop includes a software implementation of MapReduce. More details on MapReduce and Hadoop are provided later in this chapter.

IBM Watson

In 2011, IBM's computer system Watson participated in the U.S. television game show *Jeopardy* against two of the best *Jeopardy* champions in the show's history. In the game, the contestants are provided a clue such as "He likes his martinis shaken, not stirred" and the correct response, phrased in the form of a question, would be, "Who is James Bond?" Over the three-day tournament, Watson was able to defeat the two human contestants.

To educate Watson, Hadoop was utilized to process various data sources such as encyclopedias, dictionaries, news wire feeds, literature, and the entire contents of Wikipedia [2]. For each clue provided during the game, Watson had to perform the following tasks in less than three seconds [3]:

- Deconstruct the provided clue into words and phrases
- Establish the grammatical relationship between the words and the phrases
- Create a set of similar terms to use in Watson's search for a response
- Use Hadoop to coordinate the search for a response across terabytes of data
- Determine possible responses and assign their likelihood of being correct
- Actuate the buzzer
- Provide a syntactically correct response in English

Among other applications, Watson is being used in the medical profession to diagnose patients and provide treatment recommendations [4].

LinkedIn

LinkedIn is an online professional network of 250 million users in 200 countries as of early 2014 [5]. LinkedIn provides several free and subscription-based services, such as company information pages, job postings, talent searches, social graphs of one's contacts, personally tailored news feeds, and access to discussion groups, including a Hadoop users group. LinkedIn utilizes Hadoop for the following purposes [6]:

- Process daily production database transaction logs
- Examine the users' activities such as views and clicks
- Feed the extracted data back to the production systems
- Restructure the data to add to an analytical database
- Develop and test analytical models

Yahoo!

As of 2012, Yahoo! has one of the largest publicly announced Hadoop deployments at 42,000 nodes across several clusters utilizing 350 petabytes of raw storage [7]. Yahoo!'s Hadoop applications include the following [8]:

- Search index creation and maintenance
- Web page content optimization

- Web ad placement optimization
- Spam filters
- Ad-hoc analysis and analytic model development

Prior to deploying Hadoop, it took 26 days to process three years' worth of log data. With Hadoop, the processing time was reduced to 20 minutes.

10.1.2 MapReduce

As mentioned earlier, the MapReduce paradigm provides the means to break a large task into smaller tasks, run the tasks in parallel, and consolidate the outputs of the individual tasks into the final output. As its name implies, MapReduce consists of two basic parts—a map step and a reduce step—detailed as follows:

Map:

- Applies an operation to a piece of data
- Provides some intermediate output

Reduce:

- Consolidates the intermediate outputs from the map steps
- Provides the final output

Each step uses key/value pairs, denoted as `<key, value>`, as input and output. It is useful to think of the key/value pairs as a simple ordered pair. However, the pairs can take fairly complex forms. For example, the key could be a filename, and the value could be the entire contents of the file.

The simplest illustration of MapReduce is a word count example in which the task is to simply count the number of times each word appears in a collection of documents. In practice, the objective of such an exercise is to establish a list of words and their frequency for purposes of search or establishing the relative importance of certain words. Chapter 9 provides more details on text analytics. Figure 10-1 illustrates the MapReduce processing for a single input—in this case, a line of text.

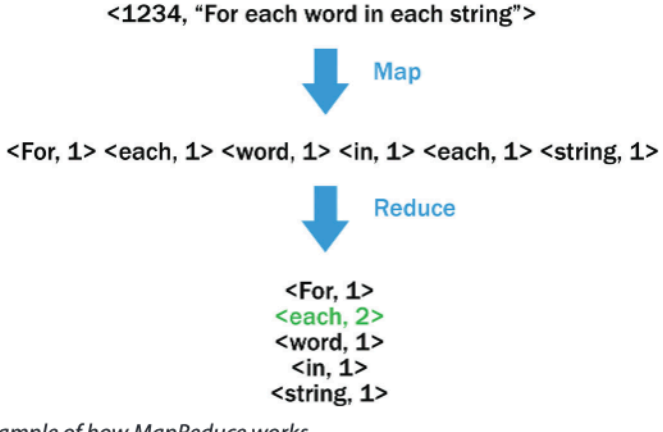


FIGURE 10-1 *Example of how MapReduce works*

In this example, the map step parses the provided text string into individual words and emits a set of key/value pairs of the form `<word, 1>`. For each unique key—in this example, `word`—the reduce step sums the `1` values and outputs the `<word, count>` key/value pairs. Because the word `each` appeared twice in the given line of text, the reduce step provides a corresponding key/value pair of `<each, 2>`.

It should be noted that, in this example, the original key, `1234`, is ignored in the processing. In a typical word count application, the map step may be applied to millions of lines of text, and the reduce step will summarize the key/value pairs generated by all the map steps.

Expanding on the word count example, the final output of a MapReduce process applied to a set of documents might have the `key` as an ordered pair and the `value` as an ordered tuple of length 2n. A possible representation of such a key/value pair follows:

```
<(filename, datetime),(word1,5, word2,7,... , wordn,6)>
```

In this construction, the key is the ordered pair `filename` and `datetime`. The value consists of the n pairs of the words and their individual counts in the corresponding file.

Of course, a word count problem could be addressed in many ways other than MapReduce. However, MapReduce has the advantage of being able to distribute the workload over a cluster of computers and run the tasks in parallel. In a word count, the documents, or even pieces of the documents, could be processed simultaneously during the map step. A key characteristic of MapReduce is that the processing of one portion of the input can be carried out independently of the processing of the other inputs. Thus, the workload can be easily distributed over a cluster of machines.

U.S. Navy rear admiral Grace Hopper (1906–1992), who was a pioneer in the field of computers, provided one of the best explanations of the need for using a group of computers. She commented that during preindustrial times, oxen were used for heavy pulling, but when one ox couldn't budge a log, people didn't try to raise a larger ox; they added more oxen. Her point was that as computational problems grow, instead of building a bigger, more powerful, and more expensive computer, a better alternative is to build a system of computers to share the workload. Thus, in the MapReduce context, a large processing task would be distributed across many computers.

Although the concept of MapReduce has existed for decades, Google led the resurgence in its interest and adoption starting in 2004 with the published work by Dean and Ghemawat [9]. This paper described Google's approach for crawling the web and building Google's search engine. As the paper describes, MapReduce has been used in functional programming languages such as Lisp, which obtained its name from being readily able to process lists (List processing).

In 2007, a well-publicized MapReduce use case was the conversion of 11 million *New York Times* newspaper articles from 1851 to 1980 into PDF files. The intent was to make the PDF files openly available to users on the Internet. After some development and testing of the MapReduce code on a local machine, the 11 million PDF files were generated on a 100-node cluster in about 24 hours [10].

What allowed the development of the MapReduce code and its execution to proceed easily was that the MapReduce paradigm had already been implemented in Apache Hadoop.

10.1.3 Apache Hadoop

Although MapReduce is a simple paradigm to understand, it is not as easy to implement, especially in a distributed system. Executing a MapReduce job (the MapReduce code run against some specified data) requires the management and coordination of several activities:

- MapReduce jobs need to be scheduled based on the system's workload.
- Jobs need to be monitored and managed to ensure that any encountered errors are properly handled so that the job continues to execute if the system partially fails.
- Input data needs to be spread across the cluster.
- Map step processing of the input needs to be conducted across the distributed system, preferably on the same machines where the data resides.
- Intermediate outputs from the numerous map steps need to be collected and provided to the proper machines for the reduce step execution.
- Final output needs to be made available for use by another user, another application, or perhaps another MapReduce job.

Fortunately, Apache Hadoop handles these activities and more. Furthermore, many of these activities are transparent to the developer/user. The following material examines the implementation of MapReduce in Hadoop, an open source project managed and licensed by the Apache Software Foundation [11].

The origins of Hadoop began as a search engine called Nutch, developed by Doug Cutting and Mike Cafarella. Based on two Google papers [9] [12], versions of MapReduce and the Google File System were added to Nutch in 2004. In 2006, Yahoo! hired Cutting, who helped to develop Hadoop based on the code in Nutch [13]. The name "Hadoop" came from the name of Cutting's child's stuffed toy elephant that also inspired the well-recognized symbol for the Hadoop project.

Next, an overview of how data is stored in a Hadoop environment is presented.

Hadoop Distributed File System (HDFS)

Based on the Google File System [12], the Hadoop Distributed File System (HDFS) is a file system that provides the capability to distribute data across a cluster to take advantage of the parallel processing of MapReduce. HDFS is not an alternative to common file systems, such as ext3, ext4, and XFS. In fact, HDFS depends on each disk drive's file system to manage the data being stored to the drive media. The Hadoop Wiki [14] provides more details on disk configuration options and considerations.

For a given file, HDFS breaks the file, say, into 64 MB blocks and stores the blocks across the cluster. So, if a file size is 300 MB, the file is stored in five blocks: four 64 MB blocks and one 44 MB block. If a file size is smaller than 64 MB, the block is assigned the size of the file.

Whenever possible, HDFS attempts to store the blocks for a file on different machines so the map step can operate on each block of a file in parallel. Also, by default, HDFS creates three copies of each block across the cluster to provide the necessary redundancy in case of a failure. If a machine fails, HDFS replicates an accessible copy of the relevant data blocks to another available machine. HDFS is also rack aware, which means that it distributes the blocks across several equipment racks to prevent an entire rack failure from causing a data unavailable event. Additionally, the three copies of each block allow Hadoop some flexibility in determining which machine to use for the map step on a particular block. For example,

an idle or underutilized machine that contains a data block to be processed can be scheduled to process that data block.

To manage the data access, HDFS utilizes three Java daemons (background processes): NameNode, DataNode, and Secondary NameNode. Running on a single machine, the ***NameNode*** daemon determines and tracks where the various blocks of a data file are stored. The ***DataNode*** daemon manages the data stored on each machine. If a client application wants to access a particular file stored in HDFS, the application contacts the NameNode, and the NameNode provides the application with the locations of the various blocks for that file. The application then communicates with the appropriate DataNodes to access the file.

Each DataNode periodically builds a report about the blocks stored on the DataNode and sends the report to the NameNode. If one or more blocks are not accessible on a DataNode, the NameNode ensures that an accessible copy of an inaccessible data block is replicated to another machine. For performance reasons, the NameNode resides in a machine's memory. Because the NameNode is critical to the operation of HDFS, any unavailability or corruption of the NameNode results in a data unavailability event on the cluster. Thus, the NameNode is viewed as a single point of failure in the Hadoop environment [15]. To minimize the chance of a NameNode failure and to improve performance, the NameNode is typically run on a dedicated machine.

A third daemon, the ***Secondary NameNode***, provides the capability to perform some of the NameNode tasks to reduce the load on the NameNode. Such tasks include updating the file system image with the contents of the file system edit logs. It is important to note that the Secondary NameNode is not a backup or redundant NameNode. In the event of a NameNode outage, the NameNode must be restarted and initialized with the last file system image file and the contents of the edits logs. The latest versions of Hadoop provide an HDFS High Availability (HA) feature. This feature enables the use of two NameNodes: one in an active state, and the other in a standby state. If an active NameNode fails, the standby NameNode takes over. When using the HDFS HA feature, a Secondary NameNode is unnecessary [16].

Figure 10-2 illustrates a Hadoop cluster with ten machines and the storage of one large file requiring three HDFS data blocks. Furthermore, this file is stored using triple replication. The machines running the NameNode and the Secondary NameNode are considered ***master nodes***. Because the DataNodes take their instructions from the master nodes, the machines running the DataNodes are referred to as ***worker nodes***.

Structuring a MapReduce Job in Hadoop

Hadoop provides the ability to run MapReduce jobs as described, at a high level, in Section 10.1.2. This section offers specific details on how a MapReduce job is run in Hadoop. A typical MapReduce program in Java consists of three classes: the driver, the mapper, and the reducer.

The ***driver*** provides details such as input file locations, the provisions for adding the input file to the map task, the names of the mapper and reducer Java classes, and the location of the reduce task output. Various job configuration options can also be specified in the driver. For example, the number of reducers can be manually specified in the driver. Such options are useful depending on how the MapReduce job output will be used in later downstream processing.

The ***mapper*** provides the logic to be processed on each data block corresponding to the specified input files in the driver code. For example, in the word count MapReduce example provided earlier, a map task is instantiated on a worker node where a data block resides. Each map task processes a fragment of the text, line by line, parses a line into words, and emits `<word, 1>` for each word, regardless of how many

times `word` appears in the line of text. The key/value pairs are stored temporarily in the worker node's memory (or cached to the node's disk).

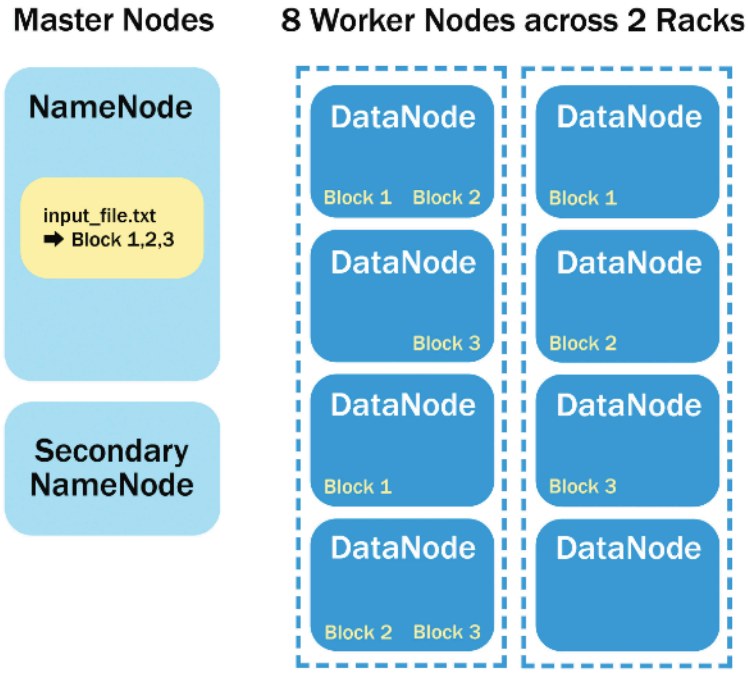


FIGURE 10-2 *A file stored in HDFS*

Next, the key/value pairs are processed by the built-in ***shuffle and sort*** functionality based on the number of reducers to be executed. In this simple example, there is only one reducer. So, all the intermediate data is passed to it. From the various map task outputs, for each unique key, arrays (lists in Java) of the associated values in the key/value pairs are constructed. Also, Hadoop ensures that the keys are passed to each reducer in sorted order. In Figure 10-3, `<each, (1,1)>` is the first key/value pair processed, followed alphabetically by `<For, (1)>` and the rest of the key/value pairs until the last key/value pair is passed to the reducer. The `( )` denotes a list of values which, in this case, is just an array of ones.

In general, each reducer processes the values for each key and emits a key/value pair as defined by the reduce logic. The output is then stored in HDFS like any other file in, say, 64 MB blocks replicated three times across the nodes.

Additional Considerations in Structuring a MapReduce Job

The preceding discussion presented the basics of structuring and running a MapReduce job on a Hadoop cluster. Several Hadoop features provide additional functionality to a MapReduce job.

First, a ***combiner*** is a useful option to apply, when possible, between the map task and the shuffle and sort. Typically, the combiner applies the same logic used in the reducer, but it also applies this logic on the output of each map task. In the word count example, a combiner sums up the number of occurrences of

each word from a mapper's output. Figure 10-4 illustrates how a combiner processes a single string in the simple word count example.

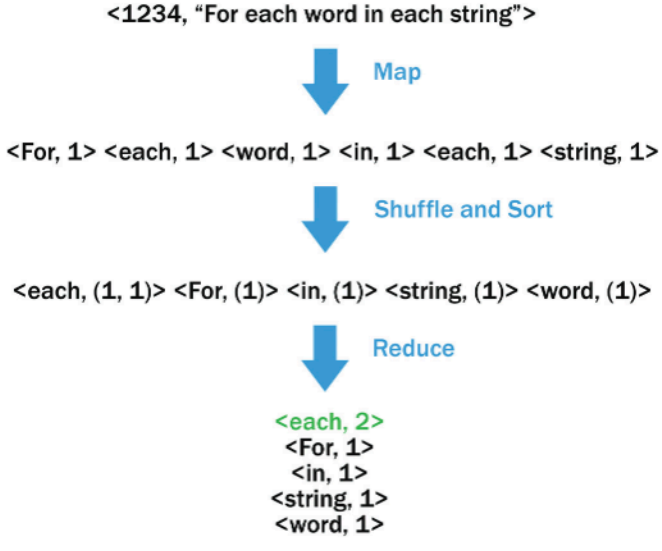


FIGURE 10-3 *Shuffle and sort*

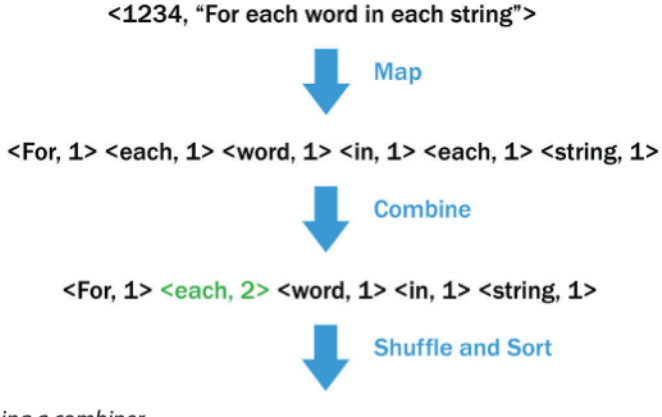


FIGURE 10-4 *Using a combiner*

Thus, in a production setting, instead of ten thousand possible `<the, 1>` key/value pairs being emitted from the map task to the Shuffle and Sort, the combiner emits one `<the, 10000>` key/value pair. The reduce step still obtains a list of values for each word, but instead of receiving a list of up to a million ones `list(1,1,. . .,1)` for a key, the reduce step obtains a list, such as `list(10000,964,. . .,8345)`, which might be as long as the number of map tasks that were run. The use of a combiner minimizes the amount of intermediate map output that the reducer must store, transfer over the network, and process.

Another useful option is the ***partitioner***. It determines the reducers that receive keys and the corresponding list of values. Using the simple word count example, Figure 10-5 shows that a partitioner can send every word that begins with a vowel to one reducer and the other words that begin with a consonant to another reducer.

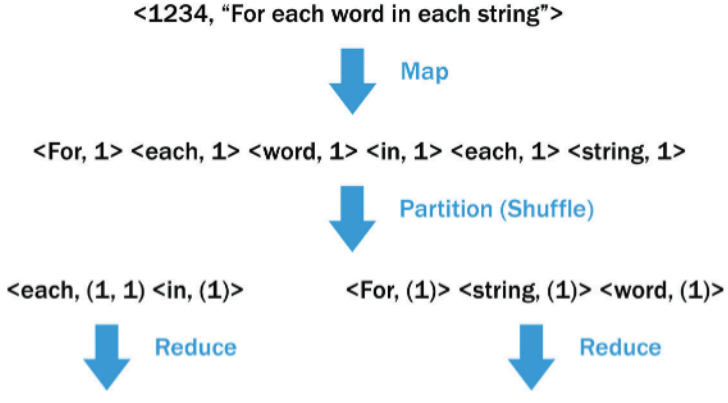


FIGURE 10-5 *Using a custom partitioner*

As a more practical example, a user could use a partitioner to separate the output into separate files for each calendar year for subsequent analysis. Also, a partitioner could be used to ensure that the workload is evenly distributed across the reducers. For example, if a few keys are known to be associated with a large majority of the data, it may be useful to ensure that these keys go to separate reducers to achieve better overall performance. Otherwise, one reducer might be assigned the majority of the data, and the MapReduce job will not complete until that one long-running reduce task completes.

Developing and Executing a Hadoop MapReduce Program

A common approach to develop a Hadoop MapReduce program is to write Java code using an Interactive Development Environment (IDE) tool such as Eclipse [17]. Compared to a plaintext editor or a command-line interface (CLI), IDE tools offer a better experience to write, compile, test, and debug code. A typical MapReduce program consists of three Java files: one each for the driver code, map code, and reduce code. Additional, Java files can be written for the combiner or the custom partitioner, if applicable. The Java code is compiled and stored as a Java Archive (JAR) file. This JAR file is then executed against the specified HDFS input files.

Beyond learning the mechanics of submitting a MapReduce job, three key challenges to a new Hadoop developer are defining the logic of the code to use the MapReduce paradigm; learning the Apache Hadoop Java classes, methods, and interfaces; and implementing the driver, map, and reduce functionality in Java. Some prior experience with Java makes it easier for a new Hadoop developer to focus on learning Hadoop and writing the MapReduce job.

For users who prefer to use a programming language other than Java, there are some other options. One option is to use the ***Hadoop Streaming API***, which allows the user to write and run Hadoop jobs with no direct knowledge of Java [18]. However, knowledge of some other programming language, such as Python, C, or Ruby, is necessary. Apache Hadoop provides the `Hadoop-streaming.jar` file that

accepts the HDFS paths for the input/output files and the paths for the files that implement the map and reduce functionality.

Here are some important considerations when preparing and running a Hadoop streaming job:

- Although the shuffle and sort output are provided to the reducer in key sorted order, the reducer does not receive the corresponding values as a list; rather, it receives individual key/value pairs. The reduce code has to monitor for changes in the value of the key and appropriately handle the new key.
- The map and reduce code must already be in an executable form, or the necessary interpreter must already be installed on each worker node.
- The map and reduce code must already reside on each worker node, or the location of the code must be provided when the job is submitted. In the latter case, the code is copied to each worker node.
- Some functionality, such as a partitioner, still needs to be written in Java.
- The inputs and outputs are handled through stdin and stdout. Stderr is also available to track the status of the tasks, implement counter functionality, and report execution issues to the display [18].
- The streaming API may not perform as well as similar functionality written in Java.

A second alternative is to use ***Hadoop pipes***, a mechanism that uses compiled C++ code for the map and reduced functionality. An advantage of using C++ is the extensive numerical libraries available to include in the code [19].

To work directly with data in HDFS, one option is to use the C API (libhdfs) or the Java API provided with Apache Hadoop. These APIs allow reads and writes to HDFS data files outside the typical MapReduce paradigm [20]. Such an approach may be useful when attempting to debug a MapReduce job by examining the input data or when the objective is to transform the HDFS data prior to running a MapReduce job.

Yet Another Resource Negotiator (YARN)

Apache Hadoop continues to undergo further development and frequent updates. An important change was to separate the MapReduce functionality from the functionality that manages the running of the jobs and the associated responsibilities in a distributed environment. This rewrite is sometimes called MapReduce 2.0, or Yet Another Resource Negotiator (YARN). YARN separates the resource management of the cluster from the scheduling and monitoring of jobs running on the cluster. The YARN implementation makes it possible for paradigms other than MapReduce to be utilized in Hadoop environments. For example, a Bulk Synchronous Parallel (BSP) [21] model may be more appropriate for graph processing than MapReduce [22] is. Apache Hama, which implements the BSP model, is one of several applications being modified to utilize the power of YARN [23].

YARN replaces the functionality previously provided by the JobTracker and TaskTracker daemons. In earlier releases of Hadoop, a MapReduce job is submitted to the JobTracker daemon. The JobTracker communicates with the NameNode to determine which worker nodes store the required data blocks for the MapReduce job. The JobTracker then assigns individual map and reduce tasks to the TaskTracker running on worker nodes. To optimize performance, each task is preferably assigned to a worker node that is storing an input data block. The TaskTracker periodically communicates with the JobTracker on the status of its executing tasks. If a task appears to have failed, the JobTracker can assign the task to a different TaskTracker.

10.2 The Hadoop Ecosystem

So far, this chapter has provided an overview of Apache Hadoop relative to its implementation of HDFS and the MapReduce paradigm. Hadoop's popularity has spawned proprietary and open source tools to make Apache Hadoop easier to use and provide additional functionality and features. This portion of the chapter examines the following Hadoop-related Apache projects:

- **Pig:** Provides a high-level data-flow programming language
- **Hive:** Provides SQL-like access
- **Mahout:** Provides analytical tools
- **HBase:** Provides real-time reads and writes

By masking the details necessary to develop a MapReduce program, Pig and Hive each enable a developer to write high-level code that is later translated into one or more MapReduce programs. Because MapReduce is intended for batch processing, Pig and Hive are also intended for batch processing use cases.

Once Hadoop processes a dataset, Mahout provides several tools that can analyze the data in a Hadoop environment. For example, a k-means clustering analysis, as described in Chapter 4, can be conducted using Mahout.

Differentiating itself from Pig and Hive batch processing, HBase provides the ability to perform real-time reads and writes of data stored in a Hadoop environment. This real-time access is accomplished partly by storing data in memory as well as in HDFS. Also, HBase does not rely on MapReduce to access the HBase data. Because the design and operation of HBase are significantly different from relational databases and the other Hadoop tools examined, a detailed description of HBase will be presented.

10.2.1 Pig

Apache Pig consists of a data flow language, Pig Latin, and an environment to execute the Pig code. The main benefit of using Pig is to utilize the power of MapReduce in a distributed system, while simplifying the tasks of developing and executing a MapReduce job. In most cases, it is transparent to the user that a MapReduce job is running in the background when Pig commands are executed. This abstraction layer on top of Hadoop simplifies the development of code against data in HDFS and makes MapReduce more accessible to a larger audience.

Like Hadoop, Pig's origin began at Yahoo! in 2006. Pig was transferred to the Apache Software Foundation in 2007 and had its first release as an Apache Hadoop subproject in 2008. As Pig evolves over time, three main characteristics persist: ease of programming, behind-the-scenes code optimization, and extensibility of capabilities [24].

With Apache Hadoop and Pig already installed, the basics of using Pig include entering the Pig execution environment by typing **`pig`** at the command prompt and then entering a sequence of Pig instruction lines at the `grunt` prompt.

An example of Pig-specific commands is shown here:

```
$ pig
grunt> records  =  LOAD '/user/customer.txt' AS
                        (cust_id:INT, first_name:CHARARRAY,
```

```
                          last_name:CHARARRAY,
                          email_address:CHARARRAY);
grunt> filtered_records = FILTER records
                          BY email_address matches '.*@isp.com';
grunt> STORE filtered_records INTO '/user/isp_customers';
grunt> quit
$
```

At the first `grunt` prompt, a text file is designated by the Pig variable `records` with four defined fields: `cust_id`, `first_name`, `last_name`, and `email_address`. Next, the variable `filtered_records` is assigned those records where the `email_address` ends with `@isp.com` to extract the customers whose e-mail address is from a particular Internet service provider (ISP). Using the `STORE` command, the filtered records are written to an HDFS folder, `isp_customers`. Finally, to exit the interactive Pig environment, execute the `QUIT` command. Alternatively, these individual Pig commands could be written to the file `filter_script.pig` and submit them at the command prompt as follows:

```
$ pig   filter_script.pig
```

Such Pig instructions are translated, behind the scenes, into one or more MapReduce jobs. Thus, Pig simplifies the coding of a MapReduce job and enables the user to quickly develop, test, and debug the Pig code. In this particular example, the MapReduce job would be initiated after the `STORE` command is processed. Prior to the `STORE` command, Pig had begun to build an execution plan but had not yet initiated MapReduce processing.

Pig provides for the execution of several common data manipulations, such as inner and outer joins between two or more files (tables), as would be expected in a typical relational database. Writing these joins explicitly in MapReduce using Hadoop would be quite involved and complex. Pig also provides a `GROUP BY` functionality that is similar to the `Group By` functionality offered in SQL. Chapter 11 has more details on using `Group By` and other SQL statements.

An additional feature of Pig is that it provides many built-in functions that are easily utilized in Pig code. Table 10-1 includes several useful functions by category.

TABLE 10-1 *Built-In Pig Functions*

Eval	Load/Store	Math	String	DateTime
AVG	BinStorage()	ABS	INDEXOF	AddDuration
CONCAT	JsonLoader	CEIL	LAST_ INDEX_OF	CurrentTime
COUNT	JsonStorage	COS, ACOS	LCFORST	DaysBetween
COUNT_STAR	PigDump	EXP	LOWER	GetDay
DIFF	PigStorage	FLOOR	REGEX_ EXTRACT	GetHour

(*continues*)

TABLE 10-1 *Built-In Pig Functions (Continued)*

Eval	Load/Store	Math	String	DateTime
IsEmpty	TextLoader	LOG, LOG10	REPLACE	GetMinute
MAX	HBaseStorage	RANDOM	STRSPLIT	GetMonth
MIN		ROUND	SUBSTRING	GetWeek
SIZE		SIN, ASIN	TRIM	GetWeekYear
SUM		SQRT	UCFIRST	GetYear
TOKENIZE		TAN, ATAN	UPPER	MinutesBetween
				SubtractDuration
				ToDate

Other functions and the details of these built-in functions can be found at the `pig.apache.org` website [25].

In terms of extensibility, Pig allows the execution of user-defined functions (UDFs) in its environment. Thus, some complex operations can be coded in the user's language of choice and executed in the Pig environment. Users can share their UDFs in a repository called the Piggybank hosted on the Apache site [26]. Over time, the most useful UDFs may be included as built-in functions in Pig.

10.2.2 Hive

Similar to Pig, Apache Hive enables users to process data without explicitly writing MapReduce code. One key difference to Pig is that the Hive language, HiveQL (Hive Query Language), resembles Structured Query Language (SQL) rather than a scripting language.

A Hive table structure consists of rows and columns. The rows typically correspond to some record, transaction, or particular entity (for example, customer) detail. The values of the corresponding columns represent the various attributes or characteristics for each row. Hadoop and its ecosystem are used to apply some structure to unstructured data. Therefore, if a table structure is an appropriate way to view the restructured data, Hive may be a good tool to use.

Additionally, a user may consider using Hive if the user has experience with SQL and the data is already in HDFS. Another consideration in using Hive may be how data will be updated or added to the Hive tables. If data will simply be added to a table periodically, Hive works well, but if there is a need to update data in place, it may be beneficial to consider another tool, such as HBase, which will be discussed in the next section.

Although Hive's performance may be better in certain applications than a conventional SQL database, Hive is not intended for real-time querying. A Hive query is first translated into a MapReduce job, which is then submitted to the Hadoop cluster. Thus, the execution of the query has to compete for resources with any other submitted job. Like Pig, Hive is intended for batch processing. Again, HBase may be a better choice for real-time query needs.

To summarize the preceding discussion, consider using Hive when the following conditions exist:

- Data easily fits into a table structure.
- Data is already in HDFS. (Note: Non-HDFS files can be loaded into a Hive table.)
- Developers are comfortable with SQL programming and queries.
- There is a desire to partition datasets based on time. (For example, daily updates are added to the Hive table.)
- Batch processing is acceptable.

The remainder of the Hive discussion covers some HiveQL basics. From the command prompt, a user enters the interactive Hive environment by simply entering **`hive`**:

```
$ hive
hive>
```

From this environment, a user can define new tables, query them, or summarize their contents. To illustrate how to use HiveQL, the following example defines a new Hive table to hold customer data, load existing HDFS data into the Hive table, and query the table.

The first step is to create a table called `customer` to store customer details. Because the table will be populated from an existing tab ('\t')-delimited HDFS file, this format is specified in the table creation query.

```
hive> create table customer (
                   cust_id bigint,
                   first_name string,
                   last_name  string,
                   email_address  string)
                   row format delimited
                   fields terminated by '\t';
```

The following HiveQL query is executed to count the number of records in the newly created table, `customer`. Because the table is currently empty, the query returns a result of zero, the last line of the provided output. The query is converted and run as a MapReduce job, which results in one map task and one reduce task being executed.

```
hive> select count(*) from customer;
Total MapReduce jobs = 1

Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
Starting Job = job_1394125045435_0001, Tracking URL =
     http://pivhdsne:8088/proxy/application_1394125045435_0001/
Kill Command = /usr/lib/gphd/hadoop/bin/hadoop job
     -kill job_1394125045435_0001
Hadoop job information for Stage-1: number of mappers: 1;
     number of reducers: 1
2014-03-06 12:30:23,542 Stage-1 map = 0%,  reduce = 0%
2014-03-06 12:30:36,586 Stage-1 map = 100%,  reduce = 0%,
     Cumulative CPU 1.71 sec
2014-03-06 12:30:48,500 Stage-1 map = 100%,  reduce = 100%,
     Cumulative CPU 3.76 sec
```

```
MapReduce Total cumulative CPU time: 3 seconds 760 msec
Ended Job = job_1394125045435_0001
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 3.76 sec   HDFS Read: 242
     HDFS Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 760 msec
OK
0
```

When querying large tables, Hive outperforms and scales better than most conventional database queries. As stated earlier, Hive translates HiveQL queries into MapReduce jobs that process pieces of large datasets in parallel.

To load the customer table with the contents of HDFS file, `customer.txt`, it is only necessary to provide the HDFS directory path to the file.

```
hive>  load data inpath '/user/customer.txt' into table customer;
```

The following query displays three rows from the `customer` table.

```
hive>  select * from customer limit 3;
34567678     Mary   Jones        mary.jones@isp.com
897572388    Harry  Schmidt      harry.schmidt@isp.com
89976576     Tom    Smith        thomas.smith@another_isp.com
```

It is often necessary to join one or more Hive tables based on one or more columns. The following example provides the mechanism to join the `customer` table with another table, `orders`, which stores the details about the customer's orders. Instead of placing all the customer details in the order table, only the corresponding `cust_id` appears in the `orders` table.

```
hive>  select o.order_number, o.order_date, c.*
              from orders o inner join customer c
              on o.cust_id = c.cust_id
              where c.email_address = 'mary.jones@isp.com';

Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
Starting Job = job_1394125045435_0002, Tracking URL =
     http://pivhdsne:8088/proxy/application_1394125045435_0002/
Kill Command = /usr/lib/gphd/hadoop/bin/hadoop job
     -kill job_1394125045435_0002
Hadoop job information for Stage-1: number of mappers: 2;
     number of reducers: 1
2014-03-06 13:26:20,277 Stage-1 map = 0%,  reduce = 0%
2014-03-06 13:26:42,568 Stage-1 map = 50%, reduce = 0%,
     Cumulative CPU 4.23 sec
2014-03-06 13:26:43,637 Stage-1 map = 100%,reduce = 0%,
     Cumulative CPU 4.79 sec
2014-03-06 13:26:52,658 Stage-1 map = 100%,reduce = 100%,
     Cumulative CPU 7.07 sec
MapReduce Total cumulative CPU time: 7 seconds 70 msec
```

```
Ended Job = job_1394125045435_0002
MapReduce Jobs Launched:
Job 0: Map: 2  Reduce: 1   Cumulative CPU: 7.07 sec   HDFS Read: 602
     HDFS Write: 140 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 70 msec
OK
X234825811 2013-11-15 17:08:43 34567678  Mary Jones mary.jones@isp.com
X234823904 2013-11-04 12:53:19 34567678  Mary Jones mary.jones@isp.com
```

The use of joins and SQL in general will be covered in Chapter 11. To exit the Hive interactive environment, use `quit`.

```
hive> quit;
$
```

An alternative to running in the interactive environment is to collect the HiveQL statements in a script (for example, `my_script.sql`) and then execute the file as follows:

```
$ hive -f my_script.sql
```

This introduction to Hive provided some of the basic HiveQL commands and statements. The reader is encouraged to research and utilize, when appropriate, other Hive functionality such as external tables, explain plans, partitions, and the `INSERT INTO` command to append data to the existing content of a Hive table.

Following are some Hive use cases:

- **Exploratory or ad-hoc analysis of HDFS data:** Data can be queried, transformed, and exported to analytical tools, such as R.
- **Extracts or data feeds to reporting systems, dashboards, or data repositories such as HBase:** Hive queries can be scheduled to provide such periodic feeds.
- **Combining external structured data to data already residing in HDFS:** Hadoop is excellent for processing unstructured data, but often there is structured data residing in an RDBMS, such as Oracle or SQL Server, that needs to be joined with the data residing in HDFS. The data from an RDBMS can be periodically added to Hive tables for querying with existing data in HDFS.

10.2.3 HBase

Unlike Pig and Hive, which are intended for batch applications, Apache HBase is capable of providing real-time read and write access to datasets with billions of rows and millions of columns. To illustrate the differences between HBase and a relational database, this section presents considerable details about the implementation and use of HBase.

The HBase design is based on Google's 2006 paper on Bigtable. This paper described Bigtable as a "distributed storage system for managing structured data." Google used Bigtable to store Google product–specific data for sites such as Google Earth, which provides satellite images of the world. Bigtable was also used to store web crawler results, data for personalized search optimization, and website clickstream data. Bigtable was built on top of the Google File System. MapReduce was also utilized to process

data into or out of a Bigtable. For example, the raw clickstream data was stored in a Bigtable. Periodically, a scheduled MapReduce job would run that would process and summarize the newly added clickstream data and append the results to a second Bigtable [27].

The development of HBase began in 2006. HBase was included as part of a Hadoop distribution at the end of 2007. In May 2010, HBase became an Apache Top Level Project. Later in 2010, Facebook began to use HBase for its user messaging infrastructure, which accommodated 350 million users sending 15 billion messages per month [28].

HBase Architecture and Data Model

HBase is a data store that is intended to be distributed across a cluster of nodes. Like Hadoop and many of its related Apache projects, HBase is built upon HDFS and achieves its real-time access speeds by sharing the workload over a large number of nodes in a distributed cluster. An HBase table consists of rows and columns. However, an HBase table also has a third dimension, version, to maintain the different values of a row and column intersection over time.

To illustrate this third dimension, a simple example would be that for any given online customer, several shipping addresses could be stored. So, the row would be indicated by a customer number. One column would provide the shipping address. The value of the shipping address would be added at the intersection of the customer number and the shipping address column, along with a timestamp corresponding to when the customer last used this shipping address.

During a customer's checkout process from an online retailer, a website might use such a table to retrieve and display the customer's previous shipping addresses. As shown in Figure 10-6, the customer can then select the appropriate address, add a new address, or delete any addresses that are no longer relevant.

FIGURE 10-6 *Choosing a shipping address at checkout*

Of course, in addition to a customer's shipping address, other customer information, such as billing address, preferences, billing credits/debits, and customer benefits (for example, free shipping) must be stored. For this type of application, real-time access is required. Thus, the use of the batch processing of Pig, Hive, or Hadoop's MapReduce is not a reasonable implementation approach. The following discussion examines how HBase stores the data and provides real-time read and write access.

As mentioned, HBase is built on top of HDFS. HBase uses a key/value structure to store the contents of an HBase table. Each value is the data to be stored at the intersection of the row, column, and version. Each key consists of the following elements [29]:

- Row length
- Row (sometimes called the row key)
- Column family length
- Column family
- Column qualifier
- Version
- Key type

The ***row*** is used as the primary attribute to access the contents of an HBase table. The row is the basis for how the data is distributed across the cluster and allows a query of an HBase table to quickly retrieve the desired elements. Thus, the structure or layout of the row has to be specifically designed based on how the data will be accessed. In this respect, an HBase table is purpose built and is not intended for general ad-hoc querying and analysis. In other words, it is important to know how the HBase table will be used; this understanding of the table's usage helps to optimally define the construction of the row and the table.

For example, if an HBase table is to store the content of e-mails, the row may be constructed as the concatenation of an e-mail address and the date sent. Because the HBase table will be stored based on the row, the retrieval of the e-mails by a given e-mail address will be fairly efficient, but the retrieval of all e-mails in a certain date range will take much longer. The later discussion on regions provides more details on how data is stored in HBase.

A column in an HBase table is designated by the combination of the ***column family*** and the ***column qualifier***. The column family provides a high-level grouping for the column qualifiers. In the earlier shipping address example, the row could contain the `order_number`, and the order details could be stored under the column family `orders`, using the column qualifiers such as `shipping_address`, `billing_address`, `order_date`. In HBase, a column is specified as column family:column qualifier. In the example, the column `orders:shipping_address` refers to an order's shipping address.

A ***cell*** is the intersection of a row and a column in a table. The ***version***, sometimes called the ***timestamp***, provides the ability to maintain different values for a cell's contents in HBase. Although the user can define a custom value for the version when writing an entry to the table, a typical HBase implementation uses HBase's default, the current system time. In Java, this timestamp is obtained with `System.getCurrentTimeMillis()`, the number of milliseconds since January 1, 1970. Because it is likely that only the most recent version of a cell may be required, the cells are stored in descending order of the version. If the application requires the cells to be stored and retrieved in ascending order of their creation time, the approach is to use `Long.MAX_VALUE - System.getCurrentTimeMillis()` in Java as the version number. `Long.MAX_VALUE` corresponds to the maximum value that a long integer can be in Java. In this case, the storing and sorting is still in descending order of the version values.

Key type is used to identify whether a particular key corresponds to a write operation to the HBase table or a delete operation from the table. Technically, a delete from an HBase table is accomplished with a write to the table. The key type indicates the purpose of the write. For deletes, a tombstone marker is

written to the table to indicate that all cell versions equal to or older than the specified timestamp should be deleted for the corresponding row and column `family:column` qualifier.

Once an HBase environment is installed, the user can enter the HBase shell environment by entering **`hbase shell`** at the command prompt. An HBase table, `my_table`, can then be created as follows:

```
$ hbase shell
hbase> create 'my_table', 'cf1', 'cf2',
              {SPLITS =>['250000','500000','750000']}
```

Two column families, `cf1` and `cf2`, are defined in the table. The `SPLITS` option specifies how the table will be divided based on the row portion of the key. In this example, the table is split into four parts, called ***regions***. Rows less than `250000` are added to the first region; rows from `250000` to less than `500000` are added to the second region, and likewise for the remaining splits. These splits provide the primary mechanism for achieving the real-time read and write access. In this example, `my_table` is split into four regions, each on its own worker node in the Hadoop cluster. Thus, as the table size increases or the user load increases, additional worker nodes and region splits can be added to scale the cluster appropriately. The reads and writes are based on the contents of the row. HBase can quickly determine the appropriate region to direct a read or write command. More about regions and their implementation will be discussed later.

Only column families, not column qualifiers, need to be defined during HBase table creation. New column qualifiers can be defined whenever data is written to the HBase table. Unlike most relational databases, in which a database administrator needs to add a column and define the data type, columns can be added to an HBase table as the need arises. Such flexibility is one of the strengths of HBase and is certainly desirable when dealing with unstructured data. Over time, the unstructured data will likely change. Thus, the new content with new column qualifiers must be extracted and added to the HBase table.

Column families help to define how the table will be physically stored. An HBase table is split into regions, but each region is split into column families that are stored separately in HDFS. From the Linux command prompt, running `hadoop fs -ls -R /hbase` shows how the HBase table, *`my_table`*, is stored in HBase.

```
$ hadoop fs -ls -R /hbase

  0 2014-02-28 16:40 /hbase/my_table/028ed22e02ad07d2d73344cd53a11fb4
243 2014-02-28 16:40 /hbase/my_table/028ed22e02ad07d2d73344cd53a11fb4/
                     .regioninfo
  0 2014-02-28 16:40 /hbase/my_table/028ed22e02ad07d2d73344cd53a11fb4/
                     cf1
  0 2014-02-28 16:40 /hbase/my_table/028ed22e02ad07d2d73344cd53a11fb4/
                     cf2
  0 2014-02-28 16:40 /hbase/my_table/2327b09784889e6198909d8b8f342289
255 2014-02-28 16:40 /hbase/my_table/2327b09784889e6198909d8b8f342289/
                     .regioninfo
  0 2014-02-28 16:40 /hbase/my_table/2327b09784889e6198909d8b8f342289/
                     cf1
  0 2014-02-28 16:40 /hbase/my_table/2327b09784889e6198909d8b8f342289/
                     cf2
  0 2014-02-28 16:40 /hbase/my_table/4b4fc9ad951297efe2b9b38640f7a5fd
267 2014-02-28 16:40 /hbase/my_table/4b4fc9ad951297efe2b9b38640f7a5fd/
                     .regioninfo
```

```
  0 2014-02-28 16:40 /hbase/my_table/4b4fc9ad951297efe2b9b38640f7a5fd/
                     cf1
  0 2014-02-28 16:40 /hbase/my_table/4b4fc9ad951297efe2b9b38640f7a5fd/
                     cf2
  0 2014-02-28 16:40 /hbase/my_table/e40be0371f43135e36ea67edec6e31e3
267 2014-02-28 16:40 /hbase/my_table/e40be0371f43135e36ea67edec6e31e3/
                     .regioninfo
  0 2014-02-28 16:40 /hbase/my_table/e40be0371f43135e36ea67edec6e31e3/
                     cf1
  0 2014-02-28 16:40 /hbase/my_table/e40be0371f43135e36ea67edec6e31e3/
                     cf2
```

As can be seen, four subdirectories have been created under `/hbase/mytable`. Each subdirectory is named by taking the hash of its respective region name, which includes the start and end rows. Under each of these directories are the directories for the column families, `cf1` and `cf2` in the example, and the `.regioninfo` file, which contains several options and attributes for how the regions will be maintained. The column family directories store keys and values for the corresponding column qualifiers. The column qualifiers from one column family should seldom be read with the column qualifiers from another column family. The reason for the separate column families is to minimize the amount of unnecessary data that HBase has to sift through within a region to find the requested data. Requesting data from two column families means that multiple directories have to be scanned to pull all the desired columns, which defeats the purpose of creating the column families in the first place. In such cases, the table design may be better off with just one column family. In practice, the number of column families should be no more than two or three. Otherwise, performance issues may arise [30].

The following operations add data to the table using the `put` command. From these three `put` operations, `data1` and `data2` are entered into column qualifiers, `cq1` and `cq2`, respectively, in column family `cf1`. The value `data3` is entered into column qualifier `cq3` in column family `cf2`. The row is designated by row key `000700` in each operation.

```
hbase> put 'my_table', '000700', 'cf1:cq1', 'data1'

0 row(s) in 0.0030 seconds

hbase> put 'my_table', '000700', 'cf1:cq2', 'data2'

0 row(s) in 0.0030 seconds

hbase> put 'my_table', '000700', 'cf2:cq3', 'data3'

0 row(s) in 0.0040 seconds
```

Data can be retrieved from the HBase table by using the `get` command. As mentioned earlier, the timestamp defaults to the milliseconds since January 1, 1970.

```
hbase> get 'my_table', '000700', 'cf2:cq3'

COLUMN       CELL
 cf2:cq3     timestamp=1393866138714, value=data3
1 row(s) in 0.0350 seconds
```

By default, the `get` command returns the most recent version. To illustrate, after executing a second `put` operation in the same row and column, a subsequent `get` provides the most recently added value of `data4`.

```
hbase> put 'my_table', '000700', 'cf2:cq3', 'data4'

0 row(s) in 0.0040 seconds

hbase> get 'my_table', '000700', 'cf2:cq3'

COLUMN       CELL
 cf2:cq3     timestamp=1393866431669, value=data4
1 row(s) in 0.0080 seconds
```

The `get` operation can provide multiple versions by specifying the number of versions to retrieve. This example illustrates that the cells are presented in descending version order.

```
hbase> get 'my_table', '000700', {COLUMN => 'cf2:cq3', VERSIONS => 2}

COLUMN       CELL
 cf2:cq3     timestamp=1393866431669, value=data4
 cf2:cq3     timestamp=1393866138714, value=data3
2 row(s) in 1.0200 seconds
```

A similar operation to the `get` command is `scan`. A `scan` retrieves all the rows between a specified `STARTROW` and a `STOPROW`, but excluding the `STOPROW`. Note: if the `STOPROW` was set to `000700`, only row `000600` would have been returned.

```
hbase> scan 'my_table', {STARTROW => '000600', STOPROW =>'000800'}

ROW           COLUMN+CELL
 000600       column=cf1:cq2, timestamp=1393866792008, value=data5
 000700       column=cf1:cq1, timestamp=1393866105687, value=data1
 000700       column=cf1:cq2, timestamp=1393866122073, value=data2
 000700       column=cf2:cq3, timestamp=1393866431669, value=data4
2 row(s) in 0.0400 seconds
```

The next operation deletes the oldest entry for column `cf2:cq3` for row `000700` by specifying the timestamp.

```
hbase> delete 'my_table', '000700', 'cf2:cq3', 1393866138714
0 row(s) in 0.0110 seconds
```

Repeating the earlier `get` operation to obtain both versions only provides the last version for that cell. After all, the older version was deleted.

```
hbase> get 'my_table', '000700', {COLUMN => 'cf2:cq3', VERSIONS => 2}

COLUMN       CELL
 cf2:cq3     timestamp=1393866431669, value=data4
1 row(s) in 0.0130 seconds
```

However, running a `scan` operation, with the RAW option set to `true`, reveals that the deleted entry actually remains. The highlighted line illustrates the creation of a tombstone marker, which informs the default `get` and `scan` operations to ignore all older cell versions of the particular row and column.

```
hbase> scan 'my_table', {RAW => true, VERSIONS => 2,
            STARTROW => '000700'}

ROW           COLUMN+CELL
 000700       column=cf1:cq1, timestamp=1393866105687, value=data1
 000700       column=cf1:cq2, timestamp=1393866122073, value=data2
 000700       column=cf2:cq3, timestamp=1393866431669, value=data4
 000700     column=cf2:cq3, timestamp=1393866138714, type=DeleteColumn
 000700       column=cf2:cq3, timestamp=1393866138714, value=data3
1 row(s) in 0.0370 seconds
```

When will the deleted entries be permanently removed? To understand this process, it is necessary to understand how HBase processes operations and achieves the real-time read and write access. As mentioned earlier, an HBase table is split into regions based on the row. Each region is maintained by a worker node. During a `put` or `delete` operation against a particular region, the worker node first writes the command to a Write Ahead Log (WAL) file for the region. The WAL ensures that the operations are not lost if a system fails. Next, the results of the operation are stored within the worker node's RAM in a repository called MemStore [31].

Writing the entry to the MemStore provides the real-time access required. Any client can access the entries in the MemStore as soon as they are written. As the MemStore increases in size or at predetermined time intervals, the sorted MemStore is then written (flushed) to a file, known as an HFile, in HDFS on the same worker node. A typical HBase implementation flushes the MemStore when its contents are slightly less than the HDFS block size. Over time, these flushed files accumulate, and the worker node performs a ***minor compaction*** that performs a sorted merge of the various flushed files.

Meanwhile, any `get` or `scan` requests that the worker node receives examine these possible storage locations:

- MemStore
- HFiles resulting from MemStore flushes
- HFiles from minor compactions

Thus, in the case of a `delete` operation followed relatively quickly by a `get` operation on the same row, the tombstone marker is found in the MemStore and the corresponding previous versions in the smaller HFiles or previously merged HFiles. The `get` command is instantaneously processed and the appropriate data returned to the client.

Over time, as the smaller HFiles accumulate, the worker node runs a ***major compaction*** that merges the smaller HFiles into one large HFile. During the major compaction, the deleted entries and the tombstone markers are permanently removed from the files.

Use Cases for HBase

As described in Google's Bigtable paper, a common use case for a data store such as HBase is to store the results from a web crawler. Using this paper's example, the row `com.cnn.www`, for example, corresponds

to a website URL, `www.cnn.com`. A column family, called `anchor`, is defined to capture the website URLs that provide links to the row's website. What may not be an obvious implementation is that those anchoring website URLs are used as the column qualifiers. For example, if `sportsillustrated.cnn.com` provides a link to `www.cnn.com`, the column qualifier is `sportsillustrated.cnn.com`. Additional websites that provide links to `www.cnn.com` appear as additional column qualifiers. The value stored in the cell is simply the text on the website that provides the link. Here is how the CNN example may look in HBase following a `get` operation.

```
hbase> get 'web_table', 'com.cnn.www', {VERSIONS => 2}

COLUMN                                CELL
 anchor:sportsillustrated.cnn.com     timestamp=1380224620597, value=cnn
 anchor:sportsillustrated.cnn.com     timestamp=1380224000001, value=cnn.com
 anchor:edition.cnn.com               timestamp=1380224620597, value=cnn
```

Additional results are returned for each corresponding website that provides a link to `www.cnn.com`. Finally, an explanation is required for using `com.cnn.www` for the row instead of `www.cnn.com`. By reversing the URLs, the various suffixes (`.com`, `.gov`, or `.net`) that correspond to the Internet's top-level domains are stored in order. Also, the next part of the domain name (`cnn`) is stored in order. So, all of the `cnn.com` websites could be retrieved by a scan with the `STARTROW` of `com.cnn` and the appropriate `STOPROW`.

This simple use case illustrates several important points. First, it is possible to get to a billion rows and millions of columns in an HBase table. As of February 2014, more than 920 million websites have been identified [32]. Second, the row needs to be defined based on how the data will be accessed. An HBase table needs to be designed with a specific purpose in mind and a well-reasoned plan for how data will be read and written. Finally, it may be advantageous to use the column qualifiers to actually store the data of interest, rather than simply storing it in a cell. In the example, as new hosting websites are established, they become new column qualifiers.

A second use case is the storage and search access of messages. In 2010, Facebook implemented such a system using HBase. At the time, Facebook's system was handling more than 15 billion user-to-user messages per month and 120 billion chat messages per month [33]. The following describes Facebook's approach to building a search index for user inboxes. Using each word in each user's message, an HBase table was designed as follows:

- The row was defined to be the user ID.
- The column qualifier was set to a word that appears in the message.
- The version was the message ID.
- The cell's content was the offset of the word in the message.

This implementation allowed Facebook to provide auto-complete capability in the search box and to return the results of the query quickly, with the most recent messages at the top. As long as the message IDs increase over time, the versions, stored in descending order, ensure that the most recent e-mails are returned first to the user [34].

These two use cases help illustrate the importance of the upfront design of the HBase table based on how the data will be accessed. Also, these examples illustrate the power of being able to add new columns

by adding new column qualifiers, on demand. In a typical RDBMS implementation, new columns require the involvement of a DBA to alter the structure of the table.

Other HBase Usage Considerations

In addition to the HBase design aspects presented in the use case discussions, the following considerations are important for a successful implementation.

- **Java API:** Previously, several HBase shell commands and operations were presented. The shell commands are useful for exploring the data in an HBase environment and illustrating their use. However, in a production environment, the HBase Java API could be used to program the desired operations and the conditions in which to execute the operations.
- **Column family and column qualifier names:** It is important to keep the name lengths of the column families and column qualifiers as short as possible. Although short names tend to go against conventional wisdom about using meaningful, descriptive names, the names of column family name and the column qualifier are stored as part of the key of each key/value pair. Thus, every additional byte added to a name over each row can quickly add up. Also, by default, three copies of each HDFS block are replicated across the Hadoop cluster, which triples the storage requirement.
- **Defining rows:** The definition of the row is one of the most important aspects of the HBase table design. In general, this is the main mechanism to perform read/write operations on an HBase table. The row needs to be constructed in such a way that the requested columns can be easily and quickly retrieved.
- **Avoid creating sequential rows:** A natural tendency is to create rows sequentially. For example, if the row key is to have the customer identification number, and the customer identification numbers are created sequentially, HBase may run into a situation in which all the new users and their data are being written to just one region, which is not distributing the workload across the cluster as intended [35]. An approach to resolve such a problem is to randomly assign a prefix to the sequential number.
- **Versioning control:** HBase table options that can be defined during table creation or altered later control how long a version of a cell's contents will exist. There are options for TimeToLive (TTL) after which any older versions will be deleted. Also, there are options for the minimum and maximum number of versions to maintain.
- **Zookeeper:** HBase uses Apache Zookeeper to coordinate and manage the various regions running on the distributed cluster. In general, Zookeeper is "a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications." [36] Instead of building its own coordination service, HBase uses Zookeeper. Relative to HBase, there are some Zookeeper configuration considerations [37].

10.2.4 Mahout

The majority of this chapter has focused on processing, structuring, and storing large datasets using Apache Hadoop and various parts of its ecosystem. After a dataset is available in HDFS, the next step may be to apply an analytical technique presented in Chapters 4 through 9. Tools such as R are useful for analyzing relatively small datasets, but they may suffer from performance issues with the large datasets stored in Hadoop. To apply the analytical techniques within the Hadoop environment, an option is to use Apache

Mahout. This Apache project provides executable Java libraries to apply analytical techniques in a scalable manner to Big Data. In general, a mahout is a person who controls an elephant. Apache Mahout is the toolset that directs Hadoop, the elephant in this case, to yield meaningful analytic results.

Mahout provides Java code that implements the algorithms for several techniques in the following three categories [38]:

Classification:

- Logistic regression
- Naïve Bayes
- Random forests
- Hidden Markov models

Clustering:

- Canopy clustering
- K-means clustering
- Fuzzy k-means
- Expectation maximization (EM)

Recommenders/collaborative filtering:

- Nondistributed recommenders
- Distributed item-based collaborative filtering

Pivotal HD Enterprise with HAWQ

Users can download and install Apache Hadoop and the described ecosystem tools directly from the `www.apache.org` website. Another installation option is downloading commercially packaged distributions of the various Apache Hadoop projects. These distributions often include additional user functionality as well as cluster management utilities. Pivotal is a company that provides a distribution called Pivotal HD Enterprise, as illustrated in Figure 10-7.

Pivotal HD Enterprise includes several Apache software components that have been presented in this chapter. Additional Apache software includes the following:

- **Oozie:** Manages Apache Hadoop jobs by acting as a workflow scheduler system
- **Sqoop:** Efficiently moves data between Hadoop and relational databases
- **Flume:** Collects and aggregates streaming data (for example, log data)

Additional functionality provided by Pivotal includes [39] the following:

- **Command Center** is a robust cluster management tool that allows users to install, configure, monitor, and manage Hadoop components and services through a web graphical interface. It simplifies Hadoop cluster installation, upgrades, and expansion using a comprehensive dashboard with instant views of the health of the cluster and key performance metrics. Users can view live and historical information about the host, application,

and job-level metrics across the entire Pivotal HD cluster. Command Center also provides CLI and web services APIs for integration into enterprise monitoring services.

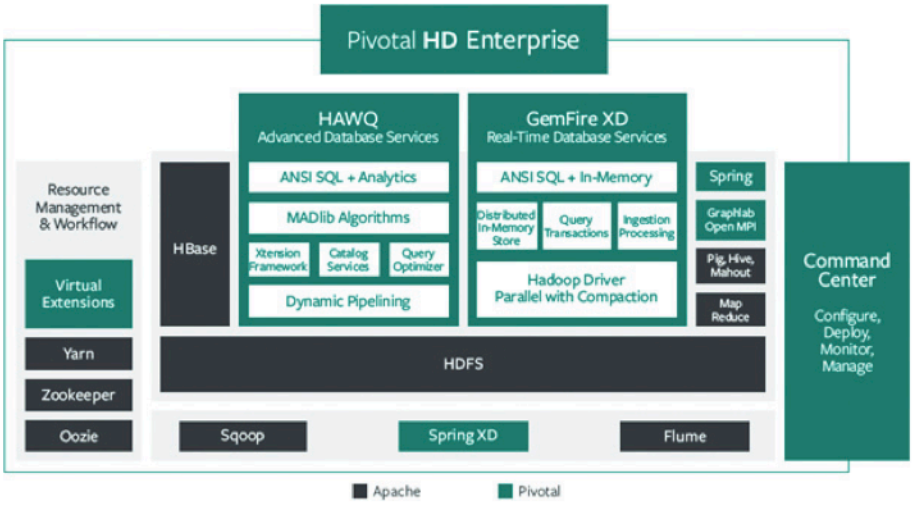


FIGURE 10-7 *Components of Pivotal HD Enterprise*

- **Graphlab on Open MPI (Message Passing Interface)** is a highly used and mature graph-based, high-performing, distributed computation framework that easily scales to graphs with billions of vertices and edges. It is now able to run natively within an existing Hadoop cluster, eliminating costly data movement. This allows data scientists and analysts to leverage popular algorithms such as page rank, collaborative filtering, and computer vision natively in Hadoop rather than copying the data somewhere else to run the analytics, which would lengthen data science cycles. Combined with MADlib's machine learning algorithms for relational data, Pivotal HD becomes the leading advanced analytical platform for machine learning in the world.
- **Hadoop Virtualization Extensions (HVE)** plug-ins make Hadoop aware of the virtual topology and scale Hadoop nodes dynamically in a virtual environment. Pivotal HD is the first Hadoop distribution to include HVE plug-ins, enabling easy deployment of Hadoop in an enterprise environment. With HVE, Pivotal HD can deliver truly elastic scalability in the cloud, augmenting on-premises deployment options.
- **HAWQ (HAdoop With Query)** adds SQL's expressive power to Hadoop to accelerate data analytics projects, simplify development while increasing productivity, expand Hadoop's capabilities, and cut costs. HAWQ can help render Hadoop queries faster than any Hadoop-based query interface on the market by adding rich, proven, parallel SQL processing facilities. HAWQ leverages existing business intelligence and analytics products and a workforce's existing SQL skills to bring more than 100 times performance improvement to a wide range of query types and workloads.

10.3 NoSQL

NoSQL (Not only Structured Query Language) is a term used to describe those data stores that are applied to unstructured data. As described earlier, HBase is such a tool that is ideal for storing key/values in column families. In general, the power of NoSQL data stores is that as the size of the data grows, the implemented solution can scale by simply adding additional machines to the distributed system. Four major categories of NoSQL tools and a few examples are provided next [40].

Key/value stores contain data (the value) that can be simply accessed by a given identifier (the key). As described in the MapReduce discussion, the values can be complex. In a key/value store, there is no stored structure of how to use the data; the client that reads and writes to a key/value store needs to maintain and utilize the logic of how to meaningfully extract the useful elements from the key and the value. Here are some uses for key/value stores:

- Using a customer's login ID as the key, the value contains the customer's preferences.
- Using a web session ID as the key, the value contains everything that was captured during the session.

Document stores are useful when the value of the key/value pair is a file and the file itself is self-describing (for example, JSON or XML). The underlying structure of the documents can be used to query and customize the display of the documents' content. Because the document is self-describing, the document store can provide additional functionality over a key/value store. For example, a document store may provide the ability to create indexes to speed the searching of the documents. Otherwise, every document in the data store would have to be examined. Document stores may be useful for the following:

- Content management of web pages
- Web analytics of stored log data

Column family stores are useful for sparse datasets, records with thousands of columns but only a few columns have entries. The key/value concept still applies, but in this case a key is associated with a collection of columns. In this collection, related columns are grouped into column families. For example, columns for age, gender, income, and education may be grouped into a demographic family. Column family data stores are useful in the following instances:

- To store and render blog entries, tags, and viewers' feedback
- To store and update various web page metrics and counters

Graph databases are intended for use cases such as networks, where there are items (people or web page links) and relationships between these items. While it is possible to store graphs such as trees in a relational database, it often becomes cumbersome to navigate, scale, and add new relationships. Graph databases help to overcome these possible obstacles and can be optimized to quickly traverse a graph (move from one item in the network to another item in the network). Following are examples of graph database implementations:

- Social networks such as Facebook and LinkedIn
- Geospatial applications such as delivery and traffic systems to optimize the time to reach one or more destinations

Table 10-2 provides a few examples of NoSQL data stores. As is often the case, the choice of a specific data store should be made based on the functional and performance requirements. A particular data store may provide exceptional functionality in one aspect, but that functionality may come at a loss of other functionality or performance.

TABLE 10-2 *Examples of NoSQL Data Stores*

Category	Data Store	Website
Key/Value	Redis	`redis.io`
	Voldemort	`www.project-voldemort.com/voldemort`
Document	CouchDB	`couchdb.apache.org`
	MongoDB	`www.mongodb.org`
Column family	Cassandra	`cassandra.apache.org`
	HBase	`hbase.apache.org/`
Graph	FlockDB	`github.com/twitter/flockdb`
	Neo4j	`www.neo4j.org`

Summary

This chapter examined the MapReduce paradigm and its application in Big Data analytics. Specifically, it examined the implementation of MapReduce in Apache Hadoop. The power of MapReduce is realized with the use of the Hadoop Distributed File System (HDFS) to store data in a distributed system. The ability to run a MapReduce job on the data stored across a cluster of machines enables the parallel processing of petabytes or exabytes of data. Furthermore, by adding additional machines to the cluster, Hadoop can scale as the data volumes grow.

This chapter examined several Apache projects within the Hadoop ecosystem. By providing a higher-level programming language, Apache Pig and Hive simplify the code development by masking the underlying MapReduce logic to perform common data processing tasks such as filtering, joining datasets, and restructuring data. Once the data is properly conditioned within the Hadoop cluster, Apache Mahout can be used to conduct data analyses such as clustering, classification, and collaborative filtering.

The strength of MapReduce in Apache Hadoop and the so far mentioned projects in the Hadoop ecosystem are in batch processing environments. When real-time processing, including read and writes, are required, Apache HBase is an option. HBase uses HDFS to store large volumes of data across the cluster, but it also maintains recent changes within memory to ensure the real-time availability of the latest data. Whereas MapReduce in Hadoop, Pig, and Hive are more general-purpose tools that can address a wide range of tasks, HBase is a somewhat more purpose-specific tool. Data will be retrieved from and written to the HBase in a well-understood manner.

HBase is one example of the NoSQL (Not only Structured Query Language) data stores that are being developed to address specific Big Data use cases. Maintaining and traversing social network graphs are examples of relational databases not being the best choice as a data store. However, relational databases and SQL remain powerful and common tools and will be examined in more detail in Chapter 11.

Exercises

1. Research and document additional use cases and actual implementations for Hadoop.
2. Compare and contrast Hadoop, Pig, Hive, and HBase. List strengths and weaknesses of each tool set. Research and summarize three published use cases for each tool set.

   Exercises 3 through 5 require some programming background and a working Hadoop environment. The text of the novel *War and Peace* can be downloaded from `http://onlinebooks.library.upenn.edu/` and used as the dataset for these exercises. However, other datasets can easily be substituted. Document all processing steps applied to the data.

3. Use MapReduce in Hadoop to perform a word count on the specified dataset.
4. Use Pig to perform a word count on the specified dataset.
5. Use Hive to perform a word count on the specified dataset.

Bibliography

[1] Apache, "Apache Hadoop," [Online]. Available: `http://hadoop.apache.org/`. [Accessed 8 May 2014].

[2] Wikipedia, "IBM Watson," [Online]. Available: `http://en.wikipedia.org/wiki/IBM_Watson`. [Accessed 11 Februry 2014].

[3] D. Davidian, "IBM.com," 14 February 2011. [Online]. Available: `https://www-304.ibm.com/connections/blogs/davidian/tags/hadoop?lang=en_us`. [Accessed 11 February 2014].

[4] IBM, "IBM.com," [Online]. Available: `http://www-03.ibm.com/innovation/us/watson/watson_in_healthcare.shtml`. [Accessed 11 February 2014].

[5] Linkedin, "LinkedIn," [Online]. Available: `http://www.linkedin.com/about-us`. [Accessed 11 February 2014].

[6] LinkedIn, "Hadoop," [Online]. Available: `http://data.linkedin.com/projects/hadoop`. [Accessed 11 February 2014].

[7] S. Singh, "http://developer.yahoo.com/," [Online]. Available: `http://developer.yahoo.com/blogs/hadoop/apache-hbase-yahoo-multi-tenancy-helm-again-171710422.html`. [Accessed 11 February 2014].

[8] E. Baldeschwieler, "http://www.slideshare.net," [Online]. Available: http://www.slideshare.net/ydn/hadoop-yahoo-internet-scale-data-processing. [Accessed 11 February 2014].

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," [Online]. Available: http://research.google.com/archive/mapreduce.html. [Accessed 11 February 2014].

[10] D. Gottfrid, "Self-Service, Prorated Supercomputing Fun!," 01 November 2007. [Online]. Available: http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/. [Accessed 11 February 2014].

[11] "apache.org," [Online]. Available: http://www.apache.org/. [Accessed 11 February 2014].

[12] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," [Online]. Available: http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf. [Accessed 11 February 2014].

[13] D. Cutting, "Free Search: Rambilings About Lucene, Nutch, Hadoop and Other Stuff," [Online]. Available: http://cutting.wordpress.com. [Accessed 11 February 2014].

[14] "Hadoop Wiki Disk Setup," [Online]. Available: http://wiki.apache.org/hadoop/DiskSetup. [Accessed 20 February 2014].

[15] "wiki.apache.org/hadoop," [Online]. Available: http://wiki.apache.org/hadoop/NameNode. [Accessed 11 February 2014].

[16] "HDFS High Availability," [Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailabilityWithNFS.html. [Accessed 8 May 2014].

[17] Eclipse. [Online]. Available: https://www.eclipse.org/downloads/. [Accessed 27 February 2014].

[18] Apache, "Hadoop Streaming," [Online]. Available: https://wiki.apache.org/hadoop/HadoopStreaming. [Accessed 8 May 2014].

[19] "Hadoop Pipes," [Online]. Available: http://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/pipes/package-summary.html. [Accessed 19 February 2014].

[20] "HDFS Design," [Online]. Available: http://hadoop.apache.org/docs/stable1/hdfs_design.html. [Accessed 19 February 2014].

[21] "BSP Tutorial," [Online]. Available: http://hama.apache.org/hama_bsp_tutorial.html. [Accessed 20 February 2014].

[22] "Hama," [Online]. Available: http://hama.apache.org/. [Accessed 20 February 2014].

[23] "PoweredByYarn," [Online]. Available: http://wiki.apache.org/hadoop/PoweredByYarn. [Accessed 20 February 2014].

[24] "pig.apache.org," [Online]. Available: http://pig.apache.org/.

[25] "Pig," [Online]. Available: http://pig.apache.org/. [Accessed 11 Feb 2014].

[26] "Piggybank," [Online]. Available: https://cwiki.apache.org/confluence/display/PIG/PiggyBank. [Accessed 28 February 2014].

[27] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber Fay Chang, "Bigtable: A Distributed Storage System for Structured Data," [Online]. Available: http://research.google.com/archive/bigtable.html. [Accessed 11 February 2014].

[28] K. Muthukkaruppan, "The Underlying Technology of Messages," 15 November 2010. [Online]. Available: http://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919. [Accessed 11 February 2014].

[29] "HBase Key Value," [Online]. Available: http://hbase.apache.org/book/regions.arch.html. [Accessed 28 February 2014].

[30] "Number of Column Families," [Online]. Available: http://hbase.apache.org/book/number.of.cfs.html.

[31] "HBase Regionserver," [Online]. Available: http://hbase.apache.org/book/regionserver.arch.html. [Accessed 3 March 2014].

[32] "Netcraft," [Online]. Available: http://news.netcraft.com/archives/2014/02/03/february-2014-web-server-survey.html. [Accessed 21 February 2014].

[33] K. Muthukkaruppan, "The Underlying Technology of Messages," 15 November 2010. [Online]. Available: http://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919. [Accessed 2011 February 2014].

[34] N. Spiegelberg. [Online]. Available: http://www.slideshare.net/brizzzdotcom/facebook-messages-hbase. [Accessed 11 February 2014].

[35] "HBase Rowkey," [Online]. Available: http://hbase.apache.org/book/rowkey.design.html. [Accessed 4 March 2014].

[36] "Zookeeper," [Online]. Available: http://zookeeper.apache.org/. [Accessed 11 Feb 2014].

[37] "Zookeeper," [Online]. Available: http://hbase.apache.org/book/zookeeper.html. [Accessed 21 February 2014].

[38] "Mahout," [Online]. Available: http://mahout.apache.org/users/basics/algorithms.html. [Accessed 19 February 2014].

[39] "Pivotal HD," [Online]. Available: http://www.gopivotal.com/big-data/pivotal-hd. [Accessed 8 May 2014].

[40] P. J. Sadalage and M. Fowler, *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot*, Upper Saddle River, NJ: Addison Wesley, 2013.

11

Advanced Analytics—Technology and Tools: In-Database Analytics

Key Concepts

MADlib
Regular expressions
SQL
User-defined functions
Window functions

In-database analytics is a broad term that describes the processing of data within its repository. In many of the earlier R examples, data was extracted from a data source and loaded into R. One advantage of in-database analytics is that the need for movement of the data into an analytic tool is eliminated. Also, by performing the analysis within the database, it is possible to obtain almost real-time results. Applications of in-database analytics include credit card transaction fraud detection, product recommendations, and web advertisement selection tailored for a particular user.

A popular open-source database is PostgreSQL. This name references an important in-database analytic language known as ***Structured Query Language (SQL)***. This chapter examines basic as well as advanced topics in SQL. The provided examples of SQL code were tested against Greenplum database 4.1.1.1, which is based on PostgreSQL 8.2.15. However, the presented concepts are applicable to other SQL environments.

11.1 SQL Essentials

A relational database, part of a Relational Database Management System (RDBMS), organizes data in tables with established relationships between the tables. Figure 11-1 shows the relationships between five tables used to store details about orders placed at an e-commerce retailer.

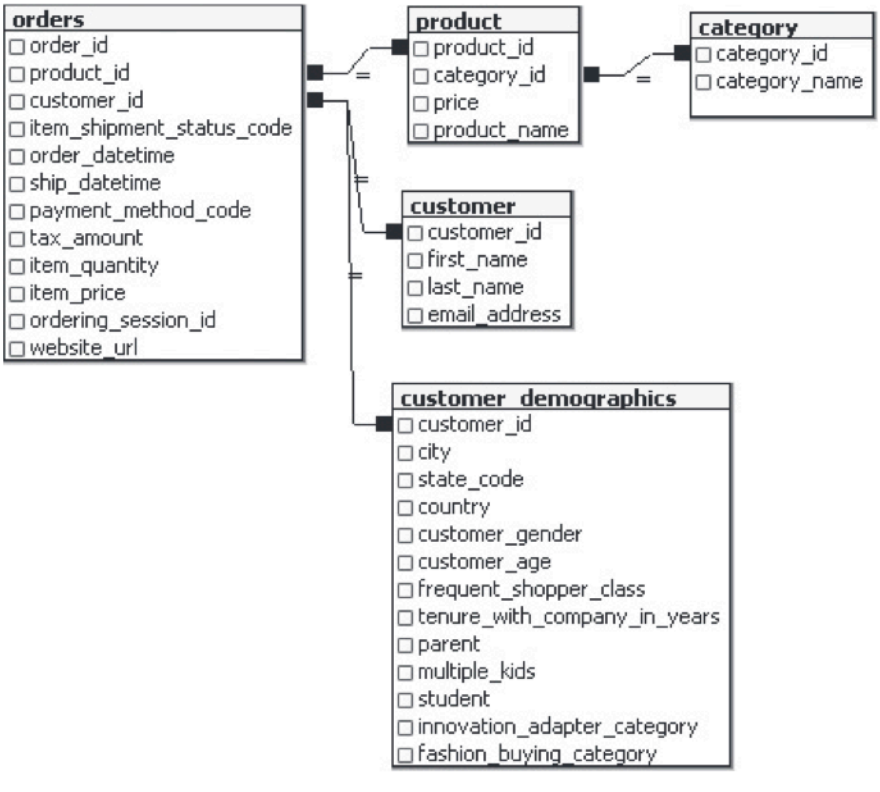


FIGURE 11-1 *Relationship diagram*

The table *orders* contains records for each order transaction. Each record contains data elements such as the *product_id* ordered, the *customer_id* for the customer who placed the order, the *order_datetime*, and so on. The other four tables provide additional details about the ordered items and the customer. The lines between the tables in Figure 11-1 illustrate the relationships between the tables. For example, a customer's first name, last name, and gender from the *customer* table can be associated with an *orders* record based on equality of the *customer_id* in these two tables.

Although it is possible to build one large table to hold all the order and customer details, the use of five tables has its advantages. The first advantage is disk storage savings. Instead of storing the product name, which can be several hundred characters in length, in the *orders* table, a much shorter *product_id*, of perhaps a few bytes, can be used and stored in place of the product's name.

Another advantage is that changes and corrections are easily made. In this example, the table *category* is used to categorize each product. If it is discovered that an incorrect category was assigned to a particular product item, only the *category_id* in the *product* table needs to be updated. Without the *product* and *category* tables, it may be necessary to update hundreds of thousands of records in the *orders* table.

A third advantage is that products can be added to the database prior to any orders being placed. Similarly, new categories can be created in anticipation of entirely new product lines being added to the online retailer's offerings later.

In a relational database design, the preference is not to duplicate pieces of data such as the customer's name across multiple records. The process of reducing such duplication is known as ***normalization***. It is important to recognize that a database that is designed to process transactions may not necessarily be optimally designed for analytical purposes. Transactional databases are often optimized to handle the insertion of new records or updates to existing records, but not optimally tuned to perform ad-hoc querying. Therefore, in designing analytical data warehouses, it is common to combine several of the tables and create one larger table, even though some pieces of data may be duplicated.

Regardless of a database's purpose, SQL is typically used to query the contents of the relational database tables as well as to insert, update, and delete data. A basic SQL query against the *customer* table may look like this.

```
SELECT first_name,
       last_name
FROM   customer
WHERE  customer_id = 666730

first_name    last_name
Mason         Hu
```

This query returns the customer information for the customer with a *customer_id* of 666730. This SQL query consists of three key parts:

- **SELECT:** Specifies the table columns to be displayed
- **FROM:** Specifies the name of the table to be queried
- **WHERE:** Specifies the criterion or filter to be applied

In a relational database, it is often necessary to access related data from multiple tables at once. To accomplish this task, the SQL query uses `JOIN` statements to specify the relationships between the multiple tables.

11.1.1 Joins

Joins enable a database user to appropriately select columns from two or more tables. Based on the relationship diagram in Figure 11-1, the following SQL query provides an example of the most common type of join: an inner join.

```
SELECT c.customer_id,
       o.order_id,
       o.product_id,
       o.item_quantity AS qty
FROM   orders o
       INNER JOIN customer c
               ON o.customer_id = c.customer_id
WHERE  c.first_name = 'Mason'
       AND c.last_name = 'Hu'
```

customer_id	order_id	product_id	qty
666730	51965-1172-6384-6923	33611	5
666730	79487-2349-4233-6891	34098	1
666730	39489-4031-0789-6076	33928	1
666730	29892-1218-2722-3191	33625	1
666730	07751-7728-7969-3140	34140	4
666730	85394-8022-6681-4716	33571	1

This query returns details of the orders placed by customer Mason Hu. The SQL query joins the two tables in the `FROM` clause based on the equality of the *`customer_id`* values. In this query, the specific *`customer_id`* value for Mason Hu does not need to be known by the programmer; only the customer's full name needs to be known.

Some additional functionality beyond the use of the `INNER JOIN` is introduced in this SQL query. Aliases `o` and `c` are assigned to tables *`orders`* and *`customer`*, respectively. Aliases are used in place of the full table names to improve the readability of the query. By design, the column names specified in the `SELECT` clause are also provided in the output. However, the outputted column name can be modified with the `AS` keyword. In the SQL query, the values of *`item_quantity`* are displayed, but this outputted column is now called *`qty`*.

The `INNER JOIN` returns those rows from the two tables where the `ON` criterion is met. From the earlier query on the *`customer`* table, there is only one row in the table for customer Mason Hu. Because the corresponding *`customer_id`* for Mason Hu appears six times in the *`orders`* table, the `INNER JOIN` query returns six records. If the `WHERE` clause was not included, the query would have returned millions of rows for all the orders that had a matching customer.

Suppose an analyst wants to know which customers have created an online account but have not yet placed an order. The next query uses a `RIGHT OUTER JOIN` to identify the first five

customers, alphabetically, who have not placed an order. The sorting of the records is accomplished with the `ORDER BY` clause.

```
SELECT c.customer_id,
       c.first_name,
       c.last_name,
       o.order_id
FROM   orders o
       RIGHT OUTER JOIN customer c
                     ON o.customer_id = c.customer_id
WHERE  o.order_id IS NULL
ORDER  BY c.last_name,
          c.first_name
LIMIT 5
```

```
customer_id first_name  last_name   order_id
143915      Abigail     Aaron
965886      Audrey      Aaron
982042      Carter      Aaron
125302      Daniel      Aaron
103964      Emily       Aaron
```

In the SQL query, a `RIGHT OUTER JOIN` is used to specify that all rows from the table *customer*, on the right-hand side (RHS) of the join, should be returned, regardless of whether there is a matching *customer_id* in the *orders* table. In this query, the `WHERE` clause restricts the results to only those joined customer records where there is no matching *order_id*. `NULL` is a special SQL keyword that denotes an unknown value. Without the `WHERE` clause, the output also would have included all the records that had a matching *customer_id* in the *orders* table, as seen in the following SQL query.

```
SELECT c.customer_id,
       c.first_name,
       c.last_name,
       o.order_id
FROM   orders o
       RIGHT OUTER JOIN customer c
                     ON o.customer_id = c.customer_id
ORDER  BY c.last_name,
          c.first_name
LIMIT 5
```

```
customer_id  first_name    last_name    order_id
143915       Abigail       Aaron
222599       Addison       Aaron        50314-7576-3355-6960
222599       Addison       Aaron        21007-7541-1255-3531
222599       Addison       Aaron        19396-4363-4499-8582
222599       Addison       Aaron        69225-1638-2944-0264
```

In the query results, the first customer, Abigail Aaron, had not placed an order, but the next customer, Addison Aaron, has placed at least four orders.

There are several other types of join statements. The `LEFT OUTER JOIN` performs the same functionality as the `RIGHT OUTER JOIN` except that all records from the table on the left-hand side (LHS) of the join are considered. A `FULL OUTER JOIN` includes all records from both tables regardless of whether there is a matching record in the other table. A `CROSS JOIN` combines two tables by matching every row of the first table with every row of the second table. If the two tables have 100 and 1,000 rows, respectively, then the resulting `CROSS JOIN` of these tables will have 100,000 rows.

The actual records returned from any join operation depend on the criteria stated in the `WHERE` clause. Thus, careful consideration needs to be taken in using a `WHERE` clause, especially with outer joins. Otherwise, the intended use of the outer join may be undone.

11.1.2 Set Operations

SQL provides the ability to perform set operations, such as unions and intersections, on rows of data. For example, suppose all the records in the *orders* table are split into two tables. The *orders_arch* table, short for orders archived, contains the orders entered prior to January 2013. The orders transacted in or after January 2013 are stored in the *orders_recent* table. However, all the orders for *product_id* 33611 are required for an analysis. One approach would be to write and run two separate queries against the two tables. The results from the two queries could then be merged later into a separate file or table. Alternatively, one query could be written using the `UNION ALL` operator as follows:

```
SELECT customer_id,
       order_id,
       order_datetime,
       product_id,
       item_quantity AS qty
FROM   orders_arch
WHERE  product_id = 33611
UNION ALL
SELECT customer_id,
       order_id,
       order_datetime,
       product_id,
       item_quantity AS qty
FROM   orders_recent
WHERE  product_id = 33611
ORDER  BY order_datetime
```

```
customer_id  order_id              order_datetime      product_id  qty
643126       13501-6446-6326-0182  2005-01-02 19:28:08 33611       1
725940       70738-4014-1618-2531  2005-01-08 06:16:31 33611       1
742448       03107-1712-8668-9967  2005-01-08 16:11:39 33611       1
.
.
.
```

```
640847        73619-0127-0657-7016  2013-01-05 14:53:27 33611        1
660446        55160-7129-2408-9181  2013-01-07 03:59:36 33611        1
647335        75014-7339-1214-6447  2013-01-27 13:02:10 33611        1
.
.
.
```

The first three records from each table are shown in the output. Because the resulting records from both tables are appended together in the output, it is important that the columns are specified in the same order and that the data types of the columns are compatible. `UNION ALL` merges the results of the two `SELECT` statements regardless of any duplicate records appearing in both `SELECT` statements. If only `UNION` was used, any duplicate records, based on all the specified columns, would be eliminated.

The `INTERSECT` operator determines any identical records that are returned by two `SELECT` statements. For example, if one wanted to know what items were purchased prior to 2013 as well as later, the SQL query using the `INTERSECT` operator would be this.

```
SELECT product_id
FROM   orders_arch
INTERSECT
SELECT product_id
FROM   orders_recent
```

```
product_id
22
30
31
.
.
.
```

It is important to note that the intersection only returns a *`product_id`* if it appears in both tables and returns exactly one instance of such a *`product_id`*. Thus, only a list of distinct product IDs is returned by the query.

To count the number of products that were ordered prior to 2013 but not after that point in time, the `EXCEPT` operator can be used to exclude the product IDs in the *`orders_recent`* table from the product IDs in the *`orders_arch`* table, as shown in the following SQL query.

```
SELECT COUNT(e.*)
FROM   (SELECT product_id
        FROM   orders_arch
        EXCEPT
        SELECT product_id
        FROM   orders_recent) e
```

```
13569
```

The preceding query uses the `COUNT` aggregate function to determine the number of returned rows from a second SQL query that includes the `EXCEPT` operator. This SQL query within a query is sometimes

called a ***subquery*** or a ***nested query***. Subqueries enable the construction of fairly complex queries without having to first execute the pieces, dump the rows to temporary tables, and then execute another SQL query to process those temporary tables. Subqueries can be used in place of a table within the `FROM` clause or can be used in the `WHERE` clause.

11.1.3 Grouping Extensions

Previously, the `COUNT()` aggregate function was used to count the number of returned rows from a query. Such aggregate functions often summarize a dataset after applying some grouping operation to it. For example, it may be desired to know the revenue by year or shipments per week. The following SQL query uses the `SUM()` aggregate function along with the `GROUP BY` operator to provide the top three ordered items based on *`item_quantity`*.

```
SELECT i.product_id,
       SUM(i.item_quantity) AS total
FROM   orders_recent i
GROUP  BY i.product_id
ORDER  BY SUM(i.item_quantity) DESC
LIMIT 3

product_id    total
15072         6089
15066         6082
15060         6053
```

`GROUP BY` can use the `ROLLUP()` operator to calculate subtotals and grand totals. The following SQL query employs the previous query as a subquery in the `WHERE` clause to supply the number of items ordered by year for the top three items ordered overall. The `ROLLUP` operator provides the subtotals, which match the previous output for each *`product_id`*, as well as the grand total.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total
FROM   orders_recent r
WHERE  r.product_id IN (SELECT o.product_id
                        FROM   orders_recent o
                        GROUP  BY o.product_id
                        ORDER  BY SUM(o.item_quantity) DESC
                        LIMIT 3)
GROUP  BY ROLLUP( r.product_id, DATE_PART('year', r.order_datetime) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime)

product_id  year    total
15060       2013    5996
15060       2014    57
15060               6053
15066       2013    6030
```

```
15066       2014    52
15066               6082
15072       2013    6023
15072       2014    66
15072               6089
                    18224
```

The CUBE operator expands on the functionality of the ROLLUP operator by providing subtotals for each column specified in the CUBE statement. Modifying the prior query by replacing the ROLLUP operator with the CUBE operator results in the same output with the addition of the subtotals for each year.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total
FROM   orders_recent r
WHERE  r.product_id IN (SELECT o.product_id
                        FROM   orders_recent o
                        GROUP  BY o.product_id
                        ORDER  BY SUM(o.item_quantity) DESC
                        LIMIT 3)
GROUP  BY CUBE( r.product_id, DATE_PART('year', r.order_datetime) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime
```

```
product_id  year    total
15060       2013    5996
15060       2014    57
15060               6053
15066       2013    6030
15066       2014    52
15066               6082
15072       2013    6023
15072       2014    66
15072               6089
            2013    18049          ← additional row
            2014    175            ← additional row
                    18224
```

Because null values in the output indicate the subtotal and grand total rows, care must be taken when null values appear in the columns being grouped. For example, null values may be part of the dataset being analyzed. The GROUPING() function can identify which rows with null values are used for the subtotals or grand totals.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime)           AS year,
       SUM(r.item_quantity)                          AS total,
       GROUPING(r.product_id)                        AS group_id,
       GROUPING(DATE_PART('year', r.order_datetime)) AS group_year
FROM   orders_recent r
```

```
WHERE  r.product_id IN (SELECT o.product_id
                        FROM   orders_recent o
                        GROUP  BY o.product_id
                        ORDER  BY SUM(o.item_quantity) DESC
                        LIMIT 3)
GROUP  BY CUBE( r.product_id, DATE_PART('year', r.order_datetime) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime)
```

product_id	year	total	group_id	group_year
15060	2013	5996	0	0
15060	2014	57	0	0
15060		6053	0	1
15066	2013	6030	0	0
15066	2014	52	0	0
15066		6082	0	1
15072	2013	6023	0	0
15072	2014	66	0	0
15072		6089	0	1
	2013	18049	1	0
	2014	175	1	0
		18224	1	1

In the preceding query, *group_year* is set to 1 when a total is calculated across the values of *year*. Similarly, *group_id* is set to 1 when a total is calculated across the values of *product_id*.

The functionality of ROLLUP and CUBE can be customized via GROUPING SETS. The SQL query using the CUBE operator can be replaced with the following query that employs GROUPING SETS to provide the same results.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total
FROM   orders_recent r
WHERE  r.product_id IN (SELECT o.product_id
                        FROM   orders_recent o
                        GROUP  BY o.product_id
                        ORDER  BY SUM(o.item_quantity) DESC
                        LIMIT 3)
GROUP  BY GROUPING SETS( ( r.product_id,
                           DATE_PART('year', r.order_datetime) ),
                         ( r.product_id ),
                         ( DATE_PART('year', r.order_datetime) ),
                         ( ) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime)
```

The listed grouping sets define the columns for which subtotals will be provided. The last grouping set, ( ), specifies that the overall total is supplied in the query results. For example, if only the grand total was desired, the following SQL query using GROUPING SETS could be used.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total
FROM   orders_recent r
WHERE  r.product_id IN (SELECT o.product_id
                        FROM   orders_recent o
                        GROUP  BY o.product_id
                        ORDER  BY SUM(o.item_quantity) DESC
                        LIMIT 3)
GROUP  BY GROUPING SETS( ( r.product_id,
                           DATE_PART('year', r.order_datetime) ),
                         ( ) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime)
```

product_id	year	total
15060	2013	5996
15060	2014	57
15066	2013	6030
15066	2014	52
15072	2013	6023
15072	2014	66
		18224

Because the GROUP BY clause can contain multiple CUBE, ROLLUP, or column specifications, duplicate grouping sets might occur. The GROUP_ID() function identifies the unique rows with a 0 and the redundant rows with a 1, 2, ... To illustrate the function GROUP_ID(), both ROLLUP and CUBE are used when only one specific ***product_id*** is being examined.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total,
       GROUP_ID()                          AS group_id
FROM   orders_recent r
WHERE  r.product_id IN ( 15060 )
GROUP  BY ROLLUP( r.product_id, DATE_PART('year', r.order_datetime) ),
          CUBE( r.product_id, DATE_PART('year', r.order_datetime) )
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime),
          GROUP_ID()
```

```
product_id  year    total   group_id
15060       2013    5996    0
15060       2013    5996    1
15060       2013    5996    3
15060       2013    5996    4
15060       2013    5996    5
15060       2013    5996    6
15060       2014    57      0
15060       2014    57      1
15060       2014    57      2
15060       2014    57      3
15060       2014    57      4
15060       2014    57      5
15060       2014    57      6
15060               6053    0
15060               6053    1
15060               6053    2
            2013    5996    0
            2014    57      0
                    6053    0
```

Filtering on the *group_id* values equal to zero yields unique records. This filtering can be accomplished with the HAVING clause, as illustrated in the next SQL query.

```
SELECT r.product_id,
       DATE_PART('year', r.order_datetime) AS year,
       SUM(r.item_quantity)                AS total,
       GROUP_ID()                          AS group_id
FROM   orders_recent r
WHERE  r.product_id IN ( 15060 )
GROUP  BY ROLLUP( r.product_id, DATE_PART('year', r.order_datetime) ),
          CUBE( r.product_id, DATE_PART('year', r.order_datetime) )
HAVING GROUP_ID() = 0
ORDER  BY r.product_id,
          DATE_PART('year', r.order_datetime),
          GROUP_ID()
```

```
product_id   year   total   group_id
15060        2013   5996    0
15060        2014   57      0
15060               6053    0
             2013   5996    0
             2014   57      0
                    6053    0
```

11.2 In-Database Text Analysis

SQL offers several basic text string functions as well as wildcard search functionality. Related SELECT statements and their results enclosed in the SQL comment delimiters, /**/, include the following:

```
SELECT SUBSTRING('1234567890', 3,2)   /* returns '34'  */
SELECT '1234567890' LIKE '%7%'        /* returns True  */
SELECT '1234567890' LIKE '7%'         /* returns False */
SELECT '1234567890' LIKE '_2%'        /* returns True  */
SELECT '1234567890' LIKE '_3%'        /* returns False */
SELECT '1234567890' LIKE '__3%'       /* returns True  */
```

This section examines more dynamic and flexible tools for text analysis, called ***regular expressions,*** and their use in SQL queries to perform pattern matching. Table 11-1 includes several forms of the comparison operator used with regular expressions and related SQL examples that produce a `True` result.

TABLE 11-1 *Regular Expression Operators*

Operator	Description	Example
~	Contains the regular expression (case sensitive)	`'123a567' ~ 'a'`
~*	Contains the regular expression (case insensitive)	`'123a567' ~* 'A'`
!~	Does not contain the regular expression (case sensitive)	`'123a567' !~ 'A'`
!~*	Does not contain the regular expression (case insensitive)	`'123a567' !~* 'b'`

More complex forms of the patterns that are specified at the RHS of the comparison operator can be constructed by using the elements in Table 11-2.

TABLE 11-2 *Regular Expression Elements*

Element	Description
|	Matches item a or b (a|b)
^	Looks for matches at the beginning of the string
$	Looks for matches at the end of the string
.	Matches any single character
*	Matches preceding item zero or more times
+	Matches preceding item one or more times
?	Makes the preceding item optional
{n}	Matches the preceding item exactly n times

(*continues*)

TABLE 11-2 *Regular Expression Elements (Continued)*

Element	Description
( )	Matches the contents exactly
[ ]	Matches any of the characters in the content, such as [0–9]
\\x	Matches a nonalphanumeric character named x
\\y	Matches an escape string \y

To illustrate the use of these elements, the following `SELECT` statements include examples in which the comparisons are `True` or `False`.

```
/* matches x or y ('x|y')*/
SELECT  '123a567' ~ '23|b'       /* returns True  */
SELECT  '123a567' ~ '32|b'       /* returns False */

/* matches the beginning of the string  */
SELECT '123a567' ~ '^123a'       /* returns True  */
SELECT '123a567' ~ '^123a7'      /* returns False */

/* matches the end of the string  */
SELECT '123a567' ~ 'a567$'       /* returns True  */
SELECT '123a567' ~ '27$'         /* returns False */

/* matches any single character */
SELECT '123a567' ~ '2.a'         /* returns True  */
SELECT '123a567' ~ '2..5'        /* returns True  */
SELECT '123a567' ~ '2...5'       /* returns False */

/* matches preceding character zero or more times */
SELECT '123a567' ~ '2*'          /* returns True  */
SELECT '123a567' ~ '2*a'         /* returns True  */
SELECT '123a567' ~ '7*a'         /* returns True  */
SELECT '123a567' ~ '37*'         /* returns True  */
SELECT '123a567' ~ '87*'         /* returns False */

/* matches preceding character one or more times */
SELECT '123a567' ~ '2+'          /* returns True  */
SELECT '123a567' ~ '2+a'         /* returns False */
SELECT '123a567' ~ '7+a'         /* returns False */
SELECT '123a567' ~ '37+'         /* returns False */
SELECT '123a567' ~ '87+'         /* returns False */

/* makes the preceding character optional */
SELECT '123a567' ~ '2?'          /* returns True  */
```

```
SELECT '123a567' ~ '2?a'          /* returns True  */
SELECT '123a567' ~ '7?a'          /* returns True  */
SELECT '123a567' ~ '37?'          /* returns True  */
SELECT '123a567' ~ '87?'          /* returns False */

/* Matches the preceding item exactly {n} times */

SELECT '123a567' ~ '5{0}'         /* returns True  */
SELECT '123a567' ~ '5{1}'         /* returns True  */
SELECT '123a567' ~ '5{2}'         /* returns False */
SELECT '1235567' ~ '5{2}'         /* returns True  */
SELECT '123a567' ~ '8{0}'         /* returns True  */
SELECT '123a567' ~ '8{1}'         /* returns False */

/* Matches the contents exactly */
SELECT '123a567' ~ '(23a5)'       /* returns True  */
SELECT '123a567' ~ '(13a5)'       /* returns False */
SELECT '123a567' ~ '(23a5)7*'     /* returns True  */
SELECT '123a567' ~ '(23a5)7+'     /* returns False */

/* Matches any of the contents */
SELECT '123a567' ~ '[23a8]'       /* returns True  */
SELECT '123a567' ~ '[8a32]'       /* returns True  */
SELECT '123a567' ~ '[(13a5)]'     /* returns True  */
SELECT '123a567' ~ '[xyz9]'       /* returns False */
SELECT '123a567' ~ '[a-z]'        /* returns True  */
SELECT '123a567' ~ '[b-z]'        /* returns False */

/* Matches a nonalphanumeric */
SELECT '$50K+' ~ '\\$'            /* returns True  */
SELECT '$50K+' ~ '\\+'            /* returns True  */
SELECT '$50K+' ~ '\\$\\+'         /* returns False */

/* Use of the backslash for escape clauses                        */
/* \\w denotes the characters 0-9, a-z, A-Z, or the underscore(_) */
SELECT '123a567'  ~ '\\w'         /* returns True  */
SELECT '123a567+' ~ '\\w'         /* returns True  */
SELECT '++++++++' ~ '\\w'         /* returns False */
SELECT '_' ~ '\\w'                /* returns True  */
SELECT '+' ~ '\\w'                /* returns False */
```

Regular expressions can be developed to identify mailing addresses, e-mail addresses, phone numbers, or currency amounts.

```
/* use of more complex regular expressions */
SELECT '$50K+' ~ '\\$[0-9]*K\\+'          /* returns True  */
SELECT '$50K+' ~ '\\$[0-9]K\\+'           /* returns False */
SELECT '$50M+' ~ '\\$[0-9]*K\\+'          /* returns False */
```

```
SELECT '$50M+' ~ '\\$[0-9]*(K|M)\\+'       /* returns True  */

/* check for ZIP code of form #####-#### */
SELECT '02038-2531' ~ '[0-9]{5}-[0-9]{4}'  /* returns True  */
SELECT '02038-253'  ~ '[0-9]{5}-[0-9]{4}'  /* returns False */
SELECT '02038'      ~ '[0-9]{5}-[0-9]{4}'  /* returns False */
```

So far, the application of regular expressions has been illustrated by including the Boolean comparison in a `SELECT` statement as if the result of the comparison was to be returned as a column. In practice, these comparisons are used in a `SELECT` statement's `WHERE` clause against a table column to identify specific records of interest. For example, the following SQL query identifies those ZIP codes in a table of customer addresses that do not match the form #####-####. Once the invalid ZIP codes are identified, corrections can be made by manual or automated means.

```
SELECT address_id,
       customer_id,
       city,
       state,
       zip,
       country
FROM   customer_addresses
WHERE  zip !~ '^[0-9]{5}-[0-9]{4}$'

address_id  customer_id   city          state    zip           country
7           13            SINAI         SD       57061-o236    USA
18          27            SHELL ROCK    IA       S0670-0480    USA
24          37            NASHVILLE     TN       37228-219     USA
.
.
.
```

SQL functions enable the use of regular expressions to extract the matching text, such as `SUBSTRING()`, as well as update the text, such as `REGEXP_REPL()`.

```
/* extract ZIP code from text string */
SELECT SUBSTRING('4321A Main Street Franklin, MA 02038-2531'
FROM '[0-9]{5}-[0-9]{4}')

02038-2531

/* replace long format zip code with short format ZIP code */
SELECT REGEXP_REPLACE('4321A Main Street Franklin, MA 02038-2531',
       '[0-9]{5}-[0-9]{4}',
       SUBSTRING(SUBSTRING('4321A Main Street Franklin, MA 02038-2531'
FROM '[0-9]{5}-[0-9]{4}'),1,5)    )

4321A Main Street Franklin, MA 02038
```

Regular expressions provide considerable flexibility in searching and modifying text strings. However, it is quite easy to build a regular expression that does not work entirely as intended. For example, a particular operation may work properly with a given dataset, but future datasets may contain new cases to be handled. Thus, it is important to thoroughly test any SQL code using regular expressions.

11.3 Advanced SQL

Building upon the foundation provided in the earlier parts of this chapter, this section presents advanced SQL techniques that can simplify in-database analytics.

11.3.1 Window Functions

In Section 11.1.3, several SQL examples using aggregate functions and grouping options to summarize a dataset were provided. A ***window function*** enables aggregation to occur but still provides the entire dataset with the summary results. For example, the `RANK()` function can be used to order a set of rows based on some attribute. Based on the SQL table, *`orders_recent`*, introduced in Section 11.1.2, the following SQL query provides a ranking of customers based on their total expenditures.

```
SELECT s.customer_id,
       s.sales,
       RANK()
         OVER (
           ORDER BY s.sales DESC ) AS sales_rank
FROM   (SELECT r.customer_id,
               SUM(r.item_quantity * r.item_price) AS sales
        FROM   orders_recent r
        GROUP  BY r.customer_id) s
```

```
customer_id   sales       sales_rank
683377        27840.00    1
238107        19983.65    2
661519        18134.11    3
628278        17965.44    4
619660        17944.20    5
.
.
.
```

The subquery in the `FROM` clause computes the total sales for each customer. In the outermost `SELECT` clause, the sales are ranked in descending order. Window functions, such as `RANK()`, are followed by an `OVER` clause that specifies how the function should be applied. Additionally, the window function can be applied to groupings of a given dataset using the `PARTITION BY` clause. The following SQL query provides the customer rankings based on sales within product categories.

```
SELECT s.category_name,
       s.customer_id,
```

```
       s.sales,
       RANK()
         OVER (
           PARTITION BY s.category_name
           ORDER BY s.sales DESC ) AS sales_rank
FROM   (SELECT c.category_name,
               r.customer_id,
               SUM(r.item_quantity * r.item_price) AS sales
        FROM   orders_recent r
               LEFT OUTER JOIN product p
                            ON r.product_id = p.product_id
               LEFT OUTER JOIN category c
                            ON p.category_id = c.category_id
        GROUP  BY c.category_name,
                  r.customer_id) s
ORDER  BY s.category_name,
          sales_rank
```

```
category_name                      customer_id sales       sales_rank
Apparel                            596396      4899.93     1
Apparel                            319036      2799.96     2
Apparel                            455683      2799.96     2
Apparel                            468209      2700.00     4
Apparel                            456107      2118.00     5
.
.
.
Apparel                            430126      2.20        78731
Automotive Parts and Accessories   362572      5706.48     1
Automotive Parts and Accessories   587564      5109.12     2
Automotive Parts and Accessories   377616      4279.86     3
Automotive Parts and Accessories   443618      4279.86     3
Automotive Parts and Accessories   590658      3668.55     5
.
.
.
```

In this case, the subquery determines each customer's sales in the respective product category. The outer `SELECT` clause then ranks the customer's sales within each category. The provided portions of the SQL query output illustrate that the ranking begins at 1 for each category and demonstrate how the rankings are affected by ties in the amount of *`sales`*.

A second use of windowing functions is to perform calculations over a sliding window in time. For example, moving averages can be used to smooth weekly sales figures that may exhibit large week-to-week variation, as shown in the plot in Figure 11-2.

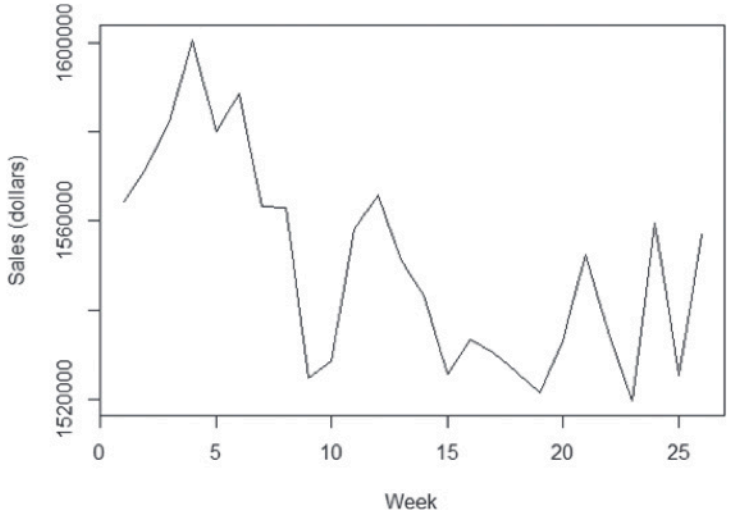


FIGURE 11-2 *Weekly sales for an online retailer*

The following SQL query illustrates how moving averages can be implemented using window functions:

```
SELECT year,
       week,
       sales,
       AVG(sales)
         OVER (
           ORDER BY year, week
           ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING) AS moving_avg
FROM   sales_by_week
WHERE  year = 2014
       AND week <= 26
ORDER  BY year,
          week
```

year	week	sales	moving_avg	
2014	1	1564539	1572999.333	←average of weeks 1, 2, 3
2014	2	1572128	1579941.75	←average of weeks 1, 2, 3, 4
2014	3	1582331	1579982.6	←average of weeks 1, 2, 3, 4, 5
2014	4	1600769	1584834.4	←average of weeks 2, 3, 4, 5, 6
2014	5	1580146	1583037.2	←average of weeks 3, 4, 5, 6, 7
2014	6	1588798	1579179.6	
2014	7	1563142	1563975.6	
2014	8	1563043	1553665	
2014	9	1524749	1547534.8	

```
2014    10      1528593     1548051.6
2014    11      1558147     1545714.2
2014    12      1565726     1549404
2014    13      1551356     1548812.6
2014    14      1543198     1543820.2
2014    15      1525636     1536767.6
2014    16      1533185     1531662.2
2014    17      1530463     1527313.6
2014    18      1525829     1528787.8
2014    19      1521455     1532649
2014    20      1533007     1533370
2014    21      1552491     1532116
2014    22      1534068     1539713.6
2014    23      1519559     1538199.6
2014    24      1559443     1539086.2  ←average of weeks 22,23,24,25,26
2014    25      1525437     1540340.75 ←average of weeks 23,24,25,26
2014    26      1556924     1547268    ←average of weeks 24,25,26
```

The windowing function uses the built-in aggregate function `AVG()`, which computes the arithmetic average of a set of values. The `ORDER BY` clause sorts the records in chronological order and specifies which rows should be included in the averaging process with the current row. In this SQL query, the moving average is based on the current row, the preceding two rows, and the following two rows. Because the dataset does not include the last two weeks of 2013, the first moving average value of 1,572,999.333 is the average of the first three weeks of 2014: the current week and the two subsequent weeks. The moving average value for the second week, 1,579,941.75, is the sales value for week 2 averaged with the prior week and the two subsequent weeks. For weeks 3 through 24, the moving average is based on the sales from 5-week periods, centered on the current week. At week 25, the window begins to include fewer weeks because the following weeks are unavailable. Figure 11-3 illustrates the applied smoothing process against the weekly sales figures.

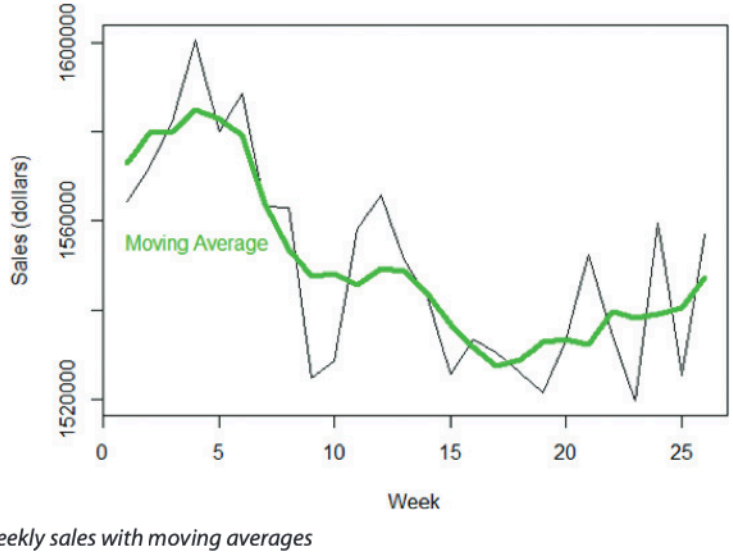


FIGURE 11-3 *Weekly sales with moving averages*

Built-in window functions may vary by SQL implementation. Table 11-3 [1] from the PostgreSQL documentation includes the list of general-purpose window functions.

TABLE 11-3 *Window Functions*

Function	Description
`row_number()`	Number of the current row within its partition, counting from 1.
`rank()`	Rank of the current row with gaps; same as `row_number` of its first peer.
`dense_rank()`	Rank of the current row without gaps; this function counts peer groups.
`percent_rank()`	Relative rank of the current row: (rank – 1) / (total rows – 1).
`cume_dist()`	Relative rank of the current row: (number of rows preceding or peer with current row) / (total rows).
`ntile(num_buckets integer)`	Integer ranging from 1 to the argument value, dividing the partition as equally as possible.
`lag(value any [, offset integer [, default any ]])`	Returns the value evaluated at the row that is offset rows before the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to `null`.
`lead(value any [, offset integer [, default any ]])`	Returns the value evaluated at the row that is offset rows after the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, the offset defaults to 1 and the default to `null`.
`first_value(value any)`	Returns the value evaluated at the first row of the window frame.
`last_value(value any)`	Returns the value evaluated at the last row of the window frame.
`nth_value(value any, nth integer)`	Returns the value evaluated at the nth row of the window frame (counting from 1); `null` if no such row.

http://www.postgresql.org/docs/9.3/static/functions-window.html

11.3.2 User-Defined Functions and Aggregates

When the built-in SQL functions are insufficient for a particular task or analysis, SQL enables the user to create user-defined functions and aggregates. This custom functionality can be incorporated into SQL queries in the same ways that the built-in functions and aggregates are used. User-defined functions can also be created to simplify processing tasks that a user may commonly encounter.

For example, a user-defined function can be written to translate text strings for female (F) and male (M) to 0 and 1, respectively. Such a function may be helpful when formatting data for use in a regression analysis. Such a function, `fm_convert()`, could be implemented as follows:

```
CREATE FUNCTION fm_convert(text) RETURNS integer AS
'SELECT CASE
         WHEN $1 = ''F'' THEN 0
         WHEN $1 = ''M'' THEN 1
         ELSE NULL
        END'
LANGUAGE SQL
IMMUTABLE
RETURNS NULL ON NULL INPUT
```

In declaring the function, the SQL query is placed within single quotes. The first and only passed value is referenced by *$1*. The SQL query is followed by a `LANGUAGE` statement that explicitly states that the preceding statement is written in SQL. Another option is to write the code in C. `IMMUTABLE` indicates that the function does not update the database and does not use the database for lookups. The `IMMUTABLE` declaration informs the database's query optimizer how best to implement the function. The `RETURNS NULL ON NULL INPUT` statement specifies how the function addresses the case when any of the inputs are `null` values.

In the online retail example, the `fm_convert()` function can be applied to the *customer_gender* column in the *customer_demographics* table as follows.

```
SELECT customer_gender,
       fm_convert(customer_gender) as male
FROM   customer_demographics
LIMIT 5

customer_gender     male
M                   1
F                   0
F                   0
M                   1
M                   1
```

Built-in and user-defined functions can be incorporated into user-defined aggregates, which can then be used as a window function. In Section 11.3.1, a window function is used to calculate moving averages to smooth a data series. In this section, a user-defined aggregate is created to calculate an ***Exponentially Weighted Moving Average (EWMA)***. For a given time series, the EWMA series is defined as shown in Equation 11-1.

$$EWMA_t = \begin{cases} y_t & for\ t = 1 \\ \alpha . y_t + (1-\alpha) . EWMA_{t-1} & for\ t \geq 2 \end{cases} \tag{11-1}$$

where $0 \leq \alpha \leq 1$

The smoothing factor, determines how much weight to place on the latest point in a given time series. By repeatedly substituting into Equation 11-1 for the prior value of the EWMA series, it can be shown that the weights against the original series are exponentially decaying backward in time.

To implement EWMA smoothing as a user-defined aggregate in SQL, the functionality in Equation 11-1 needs to be implemented first as a user-defined function.

```
CREATE FUNCTION ewma_calc(numeric, numeric, numeric) RETURNS numeric as
/* $1 = prior value of EWMA         */
/* $2 = current value of series     */
/* $3 = alpha, the smoothing factor */
'SELECT CASE
         WHEN $3 IS NULL                        /* bad alpha */
         OR $3 < 0
         OR $3 > 1 THEN NULL
         WHEN $1 IS NULL THEN $2                /* t = 1        */
         WHEN $2 IS NULL THEN $1                /* y is unknown */
         ELSE ($3 * $2) + (1-$3) *$1            /* t >= 2       */
        END'
LANGUAGE SQL
IMMUTABLE
```

Accepting three numeric inputs as defined in the comments, the `ewma_calc()` function addresses possible bad values of the smoothing factor as well as the special case in which the other inputs are `null`. The `ELSE` statement performs the usual EWMA calculation. Once this function is created, it can be referenced in the user-defined aggregate, `ewma()`.

```
CREATE AGGREGATE ewma(numeric, numeric)
       (SFUNC = ewma_calc,
        STYPE = numeric,
        PREFUNC = dummy_function)
```

In the `CREATE AGGREGATE` statement for `ewma()`, `SFUNC` assigns the state transition function (`ewma_calc` in this example) and `STYPE` assigns the data type of the variable to store the current state of the aggregate. The variable for the current state is made available to the `ewma_calc()` function as the first variable, *$1*. In this case, because the `ewma_calc()` function requires three inputs, the `ewma()` aggregate requires only two inputs; the state variable is always internally available to the aggregate. The `PREFUNC` assignment is required in the Greenplum database for use in a massively parallel processing (MPP) environment. For some aggregates, it is necessary to perform some preliminary functionality on the current state variables for a couple of servers in the MPP environment. In this example, the assigned `PREFUNC` function is added as a placeholder and is not utilized in the proper execution of the `ewma()` aggregate function.

As a window function, the `ewma()` aggregate, with a smoothing factor of 0.1, can be applied to the weekly sales data as follows.

```
SELECT year,
       week,
       sales,
       ewma(sales, .1)
         OVER (
           ORDER BY year, week)
FROM   sales_by_week
WHERE  year = 2014
```

```
       AND week <= 26
ORDER  BY year,
          week

year    week    sales       ewma
2014    1       1564539     1564539.00
2014    2       1572128     1565297.90
2014    3       1582331     1567001.21
2014    4       1600769     1570377.99
2014    5       1580146     1571354.79
.
.
.
2014    23      1519559     1542043.47
2014    24      1559443     1543783.42
2014    25      1525437     1541948.78
2014    26      1556924     1543446.30
```

Figure 11-4 includes the EWMA smoothed series to the plot from Figure 11-3.

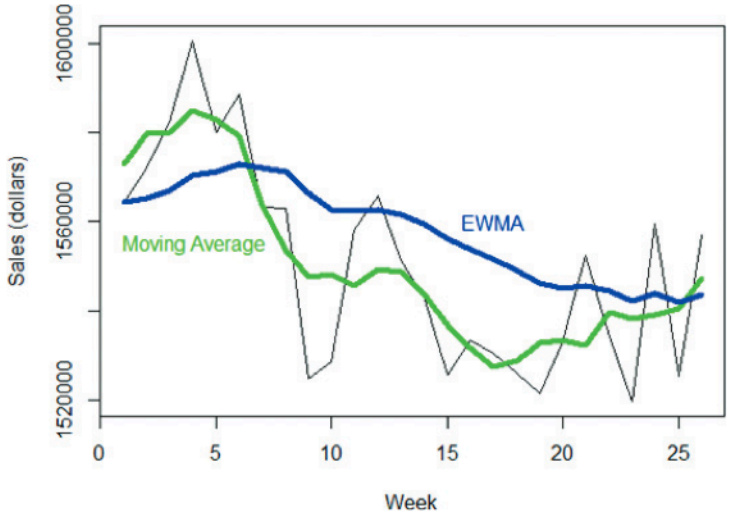


FIGURE 11-4 *Weekly sales with moving average and EWMA*

Increasing the value of the smoothing factor from 0.1 causes the EWMA to follow the actual data better, but the trade-off is that large fluctuations in the data cause larger fluctuations in the smoothed series. The user-defined aggregate, `ewma()`, is used in the SQL query in the same manner as any other window function with the specification of the `OVER` clause.

11.3.3 Ordered Aggregates

Sometimes the value of an aggregate may depend on an ordered set of values. For example, to determine the median of a set of values, it is common to first sort the values from smallest to largest and identify the median from the center of the sorted values. The sorting can be accomplished by using the function `array_agg()`. The following SQL query calculates the median of the weekly sales data.

```
SELECT (d.ord_sales[ d.n/2 + 1 ] +
                    d.ord_sales[ (d.n + 1)/2 ]) / 2.0 as median
FROM   (SELECT ARRAY_AGG(s.sales ORDER BY s.sales) AS ord_sales,
               COUNT(*)     AS n
        FROM   sales_by_week s
        WHERE  s.year = 2014
               AND s.week <= 26) d
```

```
median
1551923.5
```

In general, the function `ARRAY_AGG()` builds an array from a table column. Executing the subquery from the previous SQL query for just the first five weeks illustrates the creation of the array, denoted by the braces, and the sorted weekly sales within the array.

```
SELECT ARRAY_AGG(s.sales ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
FROM   sales_by_week s
WHERE  s.year = 2014
       AND s.week <= 5
```

```
ord_sales                                        n
{1564539,1572128,1580146,1582331,1600769}        5
```

Besides creating an array, the values can be concatenated together into one text string using the `string_agg()` function.

```
SELECT STRING_AGG(s.sales ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
FROM   sales_by_week s
WHERE  s.year = 2014
```

```
       AND s.week <= 5

ord_sales                                        n
1564539157212815801461582331160076 9             5
```

However, in this particular example, it may be useful to separate the values with a delimiter, such as a comma.

```
SELECT STRING_AGG(s.sales, ',' ORDER BY s.sales) AS ord_sales,
       COUNT(*) AS n
FROM   sales_by_week s
WHERE  s.year = 2014
       AND s.week <= 5

ord_sales                                        n
1564539,1572128,1580146,1582331,1600769          5
```

Although the sorted sales appear to be an array, there are no braces around the output. So the displayed ordered sales are a text string.

11.3.4 MADlib

SQL implementations include many basic analytical and statistical built-in functions, such as means and variances. As illustrated in this chapter, SQL also enables the development of user-defined functions and aggregates to provide additional functionality. Furthermore, SQL databases can utilize an external library of functions. One such library is known as ***MADlib***. The description file [2] included with the MADlib library download states the following:

> *MADlib is an open-source library for scalable in-database analytics. It offers data-parallel implementations of mathematical, statistical, and machine learning methods for structured and unstructured data.*

The concept of Magnetic/Agile/Deep (MAD) analysis skills was introduced in a 2009 paper by Cohen, et al. [3]. This paper describes the components of MAD as follows:

- **Magnetic:** Traditional Enterprise Data Warehouse (EDW) approaches "repel" new data sources, discouraging their incorporation until they are carefully cleansed and integrated. Given the ubiquity of data in modern organizations, a data warehouse can keep pace today only by being "magnetic": attracting all the data sources that crop up within an organization regardless of data quality niceties.
- **Agile:** Data Warehousing orthodoxy is based on long-range and careful design and planning. Given growing numbers of data sources and increasingly sophisticated and mission-critical data analyses, a modern warehouse must instead allow analysts to easily ingest, digest, produce, and adapt data rapidly. This requires a database whose physical and logical contents can be in continuous rapid evolution.
- **Deep:** Modern data analyses involve increasingly sophisticated statistical methods that go well beyond the rollups and drilldowns of traditional business intelligence (BI). Moreover, analysts often need to see both the forest and the trees in running these algorithms; they want to study enormous datasets without resorting to samples and extracts. The modern data warehouse should serve both as a deep data repository and as a sophisticated algorithmic runtime engine.

In response to the inability of a traditional EDW to readily accommodate new data sources, the concept of a ***data lake*** has emerged. A data lake represents an environment that collects and stores large volumes of structured and unstructured datasets, typically in their original, unaltered forms. More than a data depository, the data lake architecture enables the various users and data science teams to conduct data exploration and related analytical activities. Apache Hadoop is often considered a key component of building a data lake [4].

Because MADlib is designed and built to accommodate massive parallel processing of data, MADlib is ideal for Big Data in-database analytics. MADlib supports the open-source database PostgreSQL as well as the Pivotal Greenplum Database and Pivotal HAWQ. HAWQ is a SQL query engine for data stored in the Hadoop Distributed File System (HDFS). Apache Hadoop and the Pivotal products were described in Chapter 10, "Advanced Analytics—Technology and Tools: MapReduce and Hadoop."

MADlib version 1.6 modules [5] are described in Table 11-4.

TABLE 11-4 *MADlib Modules*

Module	Description
Generalized Linear Models	Includes linear regression, logistic regression, and multinomial logistic regression
Cross Validation	Evaluates the predictive power of a fitted model
Linear Systems	Solves dense and sparse linear system problems
Matrix Factorization	Performs low-rank matrix factorization and singular value decomposition
Association Rules	Implements the Apriori algorithm to identify frequent item sets
Clustering	Implements k-means clustering
Topic Modeling	Provides a Latent Dirichlet Allocation predictive model for a set of documents
Descriptive Statistics	Simplifies the computation of summary statistics and correlations
Inferential Statistics	Conducts hypothesis tests
Support Modules	Provides general array and probability functions that can also be used by other MADlib modules
Dimensionality Reduction	Enables principal component analyses and projections
Time Series Analysis	Conducts ARIMA analyses

`http://doc.madlib.net/latest/modules.html`

In the following example, MADlib is used to perform a k-means clustering analysis, as described in Chapter 4, "Advanced Analytical Theory and Methods: Clustering," on the web retailer's customers. Two

customer attributes—age and total sales since 2013—have been identified as variables of interest for the purposes of the clustering analysis. The customer's age is available in the *customer_demographics* table. The total sales for each customer can be computed from the *orders_recent* table. Because it was decided to include customers who had not purchased anything, a `LEFT OUTER JOIN` is used to include all customers. The customer's age and sales are stored in an array in the *cust_age_sales* table. The MADlib k-means function expects the coordinates to be expressed as an array.

```
/* create an empty table to store the input for the k-means analysis */
CREATE TABLE cust_age_sales (
  customer_id integer,
  coordinates float8[])

/* prepare the input for the k-means analysis */
INSERT INTO cust_age_sales (customer_id, coordinates[1], coordinates[2])
  SELECT d.customer_id,
         d.customer_age,
         CASE
           WHEN s.sales IS NULL THEN 0.0
           ELSE s.sales
         END
  FROM   customer_demographics d
         LEFT OUTER JOIN (SELECT r.customer_id,
                           SUM(r.item_quantity * r.item_price) AS sales
                          FROM   orders_recent r
                          GROUP  BY r.customer_id) s
         ON d.customer_id = s.customer_id

/* examine the first 10 rows of the input */
SELECT * from cust_age_sales
order by customer_id
LIMIT 10

customer_id     coordinates
1               {32,14.98}
2               {32,51.48}
3               {33,151.89}
4               {27,88.28}
5               {31,4.85}
6               {26,54}
7               {29,63}
8               {25,101.07}
9               {32,41.05}
10              {32,0}
```

Using the MADlib function, `kmeans_random()`, the following SQL query identifies six clusters within the provided dataset. A description of the key input values is provided with the query.

```
/*
K-means analysis

cust_age_sales - SQL table containing the input data
coordinates - the column in the SQL table that contains the data points
customer_id - the column in the SQL table that contains the
              identifier for each point
km_coord - the table to store each point and its assigned cluster
km_centers - the SQL table to store the centers of each cluster
l2norm - specifies that the Euclidean distance formula is used
25 - the maximum number of iterations
0.001 - a convergence criterion
False(twice) - ignore some options
6 - build six clusters
*/

SELECT madlib.kmeans_random('cust_age_sales', 'coordinates',
                'customer_id', 'km_coord', 'km_centers',
                'l2norm', 25 ,0.001, False, False, 6)


SELECT *
FROM   km_coord
ORDER  BY pid
LIMIT 10

pid     coords              cid
1       {1,1}:{32,14.98}    6
2       {1,1}:{32,51.48}    1
3       {1,1}:{33,151.89}   4
4       {1,1}:{27,88.28}    1
5       {1,1}:{31,4.85}     6
6       {1,1}:{26,54}       1
7       {1,1}:{29,63}       1
8       {1,1}:{25,101.07}   1
9       {1,1}:{32,41.05}    1
10      {1,1}:{32,0}        6
```

The output consists of the *km_coord* table. This table contains the coordinates for each point id (*pid*), the *customer_id*, and the assigned cluster ID (*cid*). The coordinates (*coords*) are stored as sparse vectors. Sparse vectors are useful when values in an array are repeated many times. For example, {1,200,3}:{1,0,1} represents the following vector containing 204 elements, {1,0,0,…0,1,1,1}, where the zeroes are repeated 200 times.

The coordinates for each cluster center or centroid are stored in the SQL table *km_center*.

```
SELECT *
FROM   km_centers
```

```
ORDER  BY coords

cid coords
6   {1,1}:{44.1131730722154,6.31487804161302}
1   {1,1}:{39.8000419034649,61.6213603286732}
4   {1,1}:{39.2578830823738,167.758556117954}
5   {1,1}:{40.9437092852768,409.846906145043}
3   {1,1}:{42.3521947160391,1150.68858851676}
2   {1,1}:{41.2411873840445,4458.93716141001}
```

Because the age values are similar for each centroid, it appears that the sales values dominated the distance calculations. After visualizing the clusters, it is advisable to repeat the analysis after rescaling, as discussed in Chapter 4.

Summary

This chapter presented several techniques and examples illustrating how SQL can be used to perform in-database analytics. A typical SQL query involves joining several tables, filtering the returned dataset to the desired records with a `WHERE` clause, and specifying the particular columns of interest. SQL provides the set operations of `UNION` and `UNION ALL` to merge the results of two or more `SELECT` statements or `INTERSECT` to find common record elements. Other SQL queries can summarize a dataset using aggregate functions such as `COUNT()` and `SUM()` and the `GROUP BY` clause. Grouping extensions such as the `CUBE` and `ROLLUP` operators enable the computation of subtotals and grand totals.

Although SQL is most commonly associated with structured data, SQL tables often contain unstructured data such as comments, descriptions, and other freeform text content. Regular expressions and related functions can be used in SQL to examine and restructure such unstructured data for further analysis.

More complex SQL queries can utilize window functions to supply computed values such as ranks and rolling averages along with an original dataset. In addition to built-in functions, SQL offers the ability to create user-defined functions. Although it is possible to process the data within a database and extract the results into an analytical tool such as R, external libraries such as MADlib can be utilized by SQL to conduct statistical analyses within a database.

Exercises

1. Show that EWMA smoothing is equivalent to an ARIMA(0,1,1) model with no constant, as described in Chapter 8, "Advanced Analytical Theory and Methods: Time Series Analysis."
2. Referring to Equation (11-1), demonstrate that the assigned weights decay exponentially in time.
3. Develop and test a user-defined aggregate to calculate n factorial (n!), where n is an integer.
4. From a SQL table or query, randomly select 10% of the rows. Hint: Most SQL implementations have a `random()` function that provides a uniform random number between 0 and 1. Discuss possible reasons to randomly sample records from a SQL table.

Bibliography

[1] PostgreSQl.org, "Window Functions" [Online]. Available: `http://www.postgresql.org/docs/9.3/static/functions-window.html`. [Accessed 10 April 2014].

[2] MADlib, "MADlib" [Online]. Available: `http://madlib.net/download/`. [Accessed 10 April 2014].

[3] J. Cohen, B. Dolan, M. Dunlap, J. Hellerstein, and C. Welton, "MAD Skills: New Analysis Practices for Big Data," in *Proceedings of the VLDB Endowment Volume 2 Issue 2, August 2009*.

[4] E. Dumbill, "The Data Lake Dream," ***Forbes***, 14 January 2014. [Online]. Available: `http://www.forbes.com/sites/edddumbill/2014/01/14/the-data-lake-dream/`. [Accessed 4 June 2014].

[5] MADlib, "MADlib Modules" [Online]. Available: `http://doc.madlib.net/latest/modules.html`. [Accessed 10 April 2014].

12

The Endgame, or Putting It All Together

Key Concepts

Communicating and operationalizing an analytics project
Creating the final deliverables
Using a core set of material for different audiences
Comparing main focus areas for sponsors and analysts
Understanding simple data visualization principles
Cleaning up a chart or visualization

This chapter focuses on the final phase of the Data Analytics Lifecycle: operationalize. In this phase, the project team delivers final reports, code, and technical documentation. At the conclusion of this phase, the team generally attempts to set up a pilot project and implement the developed models from Phase 4 in a production environment. As stated in Chapter 2, "Data Analytics Lifecycle," teams can perform a technically accurate analysis, but if they cannot translate the results into a language that resonates with their audience, others will not see the value, and significant effort and resources will have been wasted. This chapter focuses on showing how to construct a clear narrative summary of the work and a framework for conveying the narrative to key stakeholders.

12.1 Communicating and Operationalizing an Analytics Project

As shown in Figure 12-1, the final phase in the Data Analytics Lifecycle focuses on operationalizing the project. In this phase, teams need to assess the benefits of the project work and set up a pilot to deploy the models in a controlled way before broadening the work and sharing it with a full enterprise or ecosystem of users. In this context, a pilot project can refer to a project prior to a full-scale rollout of the new algorithms or functionality. This pilot can be a project with a more limited scope and rollout to the lines of business, products, or services affected by these new models.

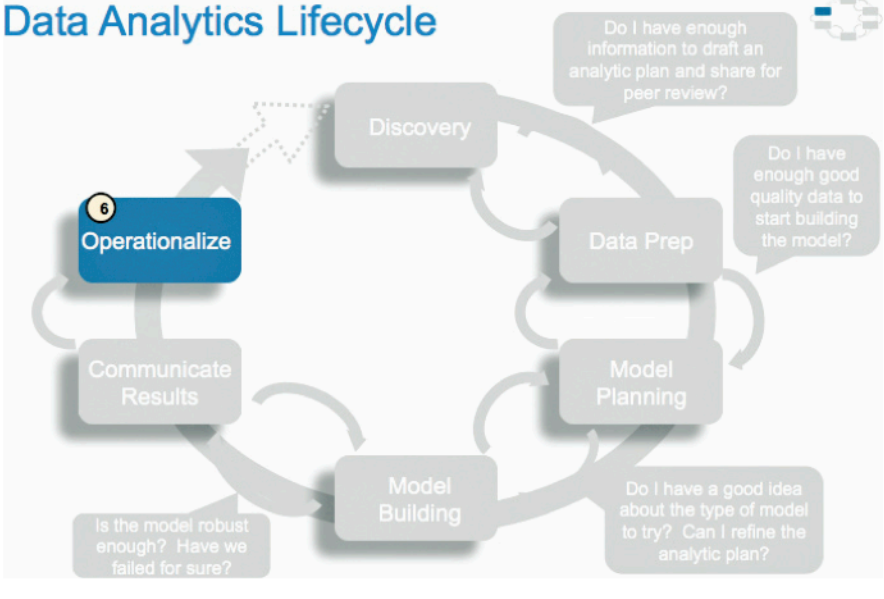


FIGURE 12-1 *Data Analytics Lifecycle, Phase 6: operationalize*

The team's ability to quantify the benefits and share them in a compelling way with the stakeholders will determine if the work will move forward into a pilot project and ultimately be run in a production environment. Therefore, it is critical to identify the benefits and state them in a clear way in the final presentations.

As the team scopes the effort involved to deploy the analytical model as a pilot project, it also needs to consider running the model in a production environment for a discrete set of products or a single line of business, which tests the model in a live setting. This allows the team to learn from the deployment and make adjustments before deploying the application or code more broadly across the enterprise. This phase can bring in a new set of team members—namely, those engineers responsible for the production environment who have a new set of issues and concerns. This group is interested in ensuring that running the model fits smoothly into the production environment and the model can be integrated into downstream processes. While executing the model in the production environment, the team should aim to detect input anomalies before they are fed to the model, assess run times, and gauge competition for resources with other processes in the production environment.

Chapter 2 included an in-depth discussion of the Data Analytics Lifecycle, including an overview of the deliverables provided in its final phase, at which time it is advisable for the team to consider the needs of each of its main stakeholders and the deliverables, illustrated in Figure 12-2, to satisfy these needs.

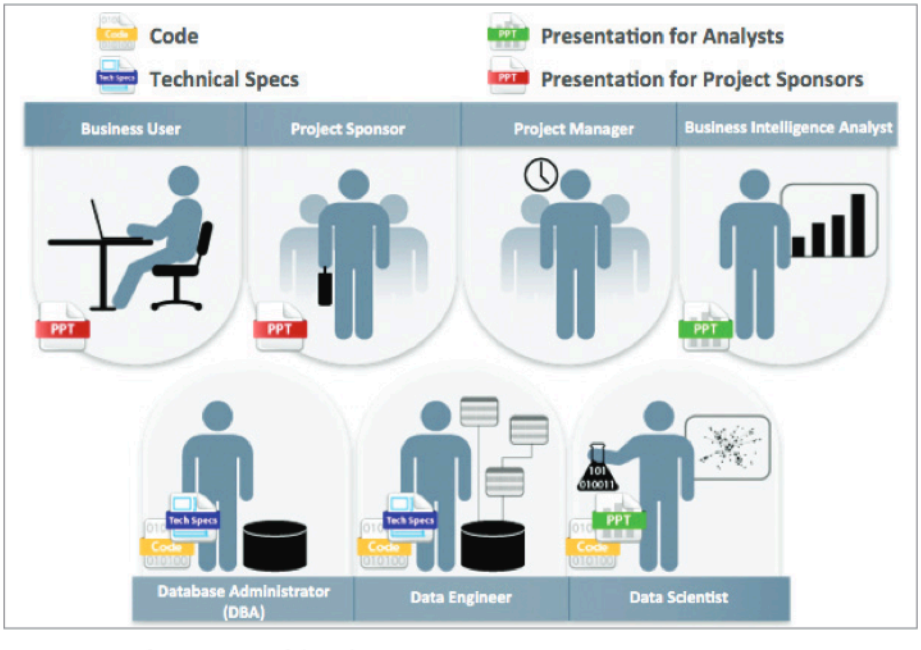


FIGURE 12-2 *Key outputs from a successful analytic project*

Following is a brief review of the key outputs for each of the main stakeholders of an analytics project and what they usually expect at the conclusion of a project:

- **Business User** typically tries to determine the benefits and implications of the findings to the business.
- **Project Sponsor** typically asks questions related to the business impact of the project, the risks and return on investment (ROI), and how the project can be evangelized within the organization and beyond.

- **Project Manager** needs to determine if the project was completed on time and within budget.
- **Business Intelligence Analyst** needs to know if the reports and dashboards he manages will be impacted and need to change.
- **Data Engineer *and* Database Administrator** (DBA) typically need to share the code from the analytical project and create technical documents that describe how to implement the code.
- **Data Scientists** need to share the code and explain the model to their peers, managers, and other stakeholders.

Although these seven roles represent many interests within a project, these interests usually overlap, and most of them can be met with four main deliverables:

- **Presentation for Project Sponsors** contains high-level takeaways for executive-level stakeholders, with a few key messages to aid their decision-making process. Focus on clean, easy visuals for the presenter to explain and for the viewer to grasp.
- **Presentation for Analysts**, which describes changes to business processes and reports. Data scientists reading this presentation are comfortable with technical graphs (such as Receiver Operating Characteristic [ROC] curves, density plots, and histograms) and will be interested in the details.
- **Code** for technical people, such as engineers and others managing the production environment
- **Technical specifications** for implementing the code

As a rule, the more executive the audience, the more succinct the presentation needs to be for project sponsors. Ensure that the presentation gets to the point quickly and frames the results in terms of value to the sponsor's organization. When presenting to other audiences with more quantitative backgrounds, focus more time on the methodology and findings. In these instances, the team can be more expansive in describing the outcomes, methodology, and analytical experiments with a peer group. This audience will be more interested in the techniques, especially if the team developed a new way of processing or analyzing data that can be reused in the future or applied to similar problems. In addition, use imagery or data visualization when possible. Although it may take more time to develop imagery, pictures are more appealing, easier to remember, and more effective to deliver key messages than long lists of bullets.

12.2 Creating the Final Deliverables

After reviewing the list of key stakeholders for data science projects and main deliverables, this section focuses on describing the deliverables in detail. To illustrate this approach, a fictional case study is used to make the examples more specific. Figure 12-3 describes a scenario of a fictional bank, YoyoDyne Bank, which would like to embark on a project to do churn prediction models of its customers. *Churn rate* in this context refers to the frequency with which customers sever their relationship as customers of YoyoDyne Bank or switch to a competing bank.

Synopsis of YoyoDyne Bank Case Study
▪ YoyoDyne Bank is a retail bank that wants to improve its Net Present Value (NPV) and its customer retention rate. ▪ It wants to establish an effective marketing campaign targeting customers to reduce the churn rate by at least five percent. ▪ The bank wants to determine whether those customers are worth retaining. In addition, the bank wants to analyze reasons for customer attrition and what it can do to keep customers from leaving. ▪ The bank wants to build a data warehouse to support marketing and other related customer care groups.

FIGURE 12-3 *Synopsis of YoyoDyne Bank case study example*

Based on this information, the data science team may create an analytics plan similar to Figure 12-4 during the project.

Components of Analytic Plan	Retail Banking: YoyoDyne Bank
Discovery **Business Problem Framed**	How can the bank identify customers with the highest likelihood for churn?
Initial Hypotheses	Transaction volume and type are key predictors of churn rates
Data and Scope	5 months of customer account history
Model Planning - Analytic Technique	Logistic regression to identify most influential factors predicting churn
Result and Key Findings	Key predictors of churn are: 1. Once customers stop using their accounts for gas and groceries, their account holdings quickly diminish and the customers churn. 2. If the customers use their debit card fewer than 5 times per month, they will leave the bank within 60 days.
Business Impact	By targeting customers who are at high risk for churn, customer attrition can be reduced by 23%. This would save $3 million in lost customer revenue and avoid $1.5 million in new customer acquisition costs each year for the bank.

FIGURE 12-4 *Analytics plan for YoyoDyne Bank case study*

In addition to guiding the model planning and methodology, the analytic plan contains components that can be used as inputs for writing about the scope, underlying assumptions, modeling techniques, initial hypotheses, and key findings in the final presentations. After spending substantial amounts of time in the modeling and performing in-depth data analysis, it is critical to reflect on the project work and consider

the context of the problems the team set out to solve. Review the work that was completed during the project, and identify observations about the model outputs, scoring, and results. Based on these observations, begin to identify the key messages and any unexpected insights.

In addition, it is important to tailor the project outputs to the audience. For a project sponsor, show that the team met the project goals. Focus on what was done, what the team accomplished, what ROI can be anticipated, and what business value can be realized. Give the project sponsor talking points to evangelize the work. Remember that the sponsor needs to relay the story to others, so make this person's job easy, and help ensure the message is accurate by providing a few talking points. Find ways to emphasize ROI and business value, and mention whether the models can be deployed within performance constraints of the sponsor's production environment.

In some organizations, the data science team may not be expected to make a full business case for future projects and implementation of the models. Instead, it needs to be able to provide guidance about the impact of the models to enable the project sponsor, or someone designated by that person, to create a business case to advocate for the pilot and subsequent deployment of this functionality. In other words, the data science team can assist in this effort by putting the results of the modeling and data science work into context to help assess the actual value and cost of implementing this work more broadly.

When presenting to a technical audience such as data scientists and analysts, focus on how the work was done. Discuss how the team accomplished the goals and the choices it made in selecting models or analyzing the data. Share analytical methods and decision-making processes so other analysts can learn from them for future projects. Describe methods, techniques, and technologies used, as this technical audience will be interested in learning about these details and considering whether the approach makes sense in this case and whether it can be extended to other, similar projects. Plan to provide specifics related to model accuracy and speed, such as how well the model will perform in a production environment.

Ideally, the team should consider starting the development of the final presentation during the project rather than at the end of the project as commonly occurs. This approach ensures that the team always has a version of the presentation with working hypotheses to show stakeholders, in case there is a need to show a work-in-process version of the project progress on short notice. In fact, many analysts write the executive summary at the outset of a project and then continually refine it over time so that at the end of the project, portions of the final presentation are already completed. This approach also reduces the chance that the team members will forget key points or insights discovered during the project. Finally, it reduces the amount of work to be done on the presentation at the conclusion of the project.

12.2.1 Developing Core Material for Multiple Audiences

Because some of the components of the projects can be used for different audiences, it can be helpful to create a core set of materials regarding the project, which can be used to create presentations for either a technical audience or an executive sponsor.

Table 12-1 depicts the main components of the final presentations for the project sponsor and an analyst audience. Notice that teams can create a core set of materials in these seven areas, which can be used for the two presentation audiences. Three areas (Project Goals, Main Findings, and Model Description), can be used as is for both presentations. Other areas need additional elaboration, such as the Approach. Still other areas, such as the Key Points, require different levels of detail for the analysts and data scientists than for the project sponsor. Each of these main components of the final presentation is discussed in subsequent sections.

TABLE 12-1 *Comparison of Materials for Sponsor and Analyst Presentations*

Presentation Component	Project Sponsor Presentation	Analyst Presentation
Project Goals	List top 3–5 agreed-upon goals.	
Main Findings	Emphasize key messages.	
Approach	High-level methodology	High-level methodology Relevant details on modeling techniques and technology
Model Description	Overview of the modeling technique	
Key Points Supported with Data	Support key points with simple charts and graphics (example: bar charts).	Show details to support the key points. Analyst-oriented charts and graphs, such as ROC curves and histograms Visuals of key variables and significance of each
Model Details	Omit this section, or discuss only at a high level.	Show the code or main logic of the model, and include model type, variables, and technology used to execute the model and score data. Identify key variables and impact of each. Describe expected model performance and any caveats. Detailed description of the modeling technique Discuss variables, scope, and predictive power.
Recommendations	Focus on business impact, including risks and ROI. Give the sponsor salient points to help her evangelize work within the organization.	Supplement recommendations with implications for the modeling or for deploying in a production environment.

12.2.2 Project Goals

The Project Goals portion of the final presentation is generally the same, or similar, for sponsors and for analysts. For each audience, the team needs to reiterate the goals of the project to lay the groundwork for

the solution and recommendations that are shared later in the presentation. In addition, the Goals slide serves to ensure there is a shared understanding between the project team and the sponsors and confirm they are aligned in moving forward in the project. Generally, the goals are agreed on early in the project. It is good practice to write them down and share them to ensure the goals and objectives are clearly understood by both the project team and the sponsors.

Figures 12-5 and 12-6 show two examples of slides for Project Goals. Figure 12-5 shows three goals for creating a predictive model to anticipate customer churn. The points on this version of the Goals slide emphasize what needs to be done, but not why, which will be included in the alternative.

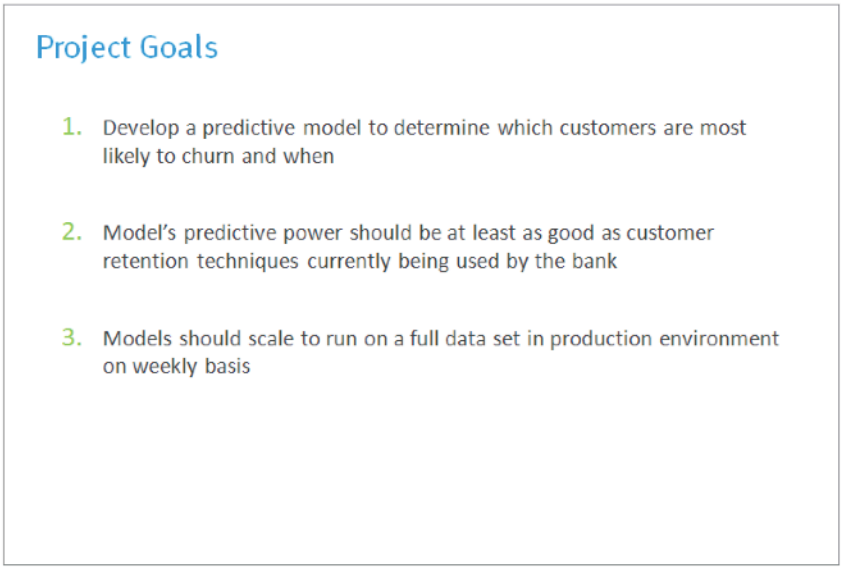


FIGURE 12-5 *Example of Project Goals slide for YoyoDyne case study*

Figure 12-6 shows a variation of the previous Project Goals slide in Figure 12-5. It is a summary of the situation prior to listing the goals. Keep in mind that when delivering final presentations, these deliverables are shared within organizations, and the original context can be lost, especially if the original sponsor leaves the group or changes roles. It is good practice to briefly recap the situation prior to showing the project goals. Keep in mind that adding a situation overview to the Goals slide does make it appear busier. The team needs to determine whether to split this into a separate slide or keep it together, depending on the audience and the team's style for delivering the final presentation.

One method for writing the situational overview in a succinct way is to summarize it in three bullets, as follows:

- **Situation:** Give a one-sentence overview of the situation that has led to the analytics project.
- **Complication:** Give a one-sentence overview of the need for addressing this now. Something has triggered the organization to decide to take action at this time. For instance, perhaps it lost 100

customers in the past two weeks and now has an executive mandate to address an issue, or perhaps it has lost five points of market share to its biggest competitor in the past three months. Usually, this sentence represents the driver for why a particular project is being initiated at this time, rather than in some vague time in the future.

- **Implication:** Give a one-sentence overview of the impact of the complication. For instance, if the bank fails to address its customer attrition problem, it stands to lose its dominant market position in three key markets. Focus on the business impact to illustrate the urgency of doing the project.

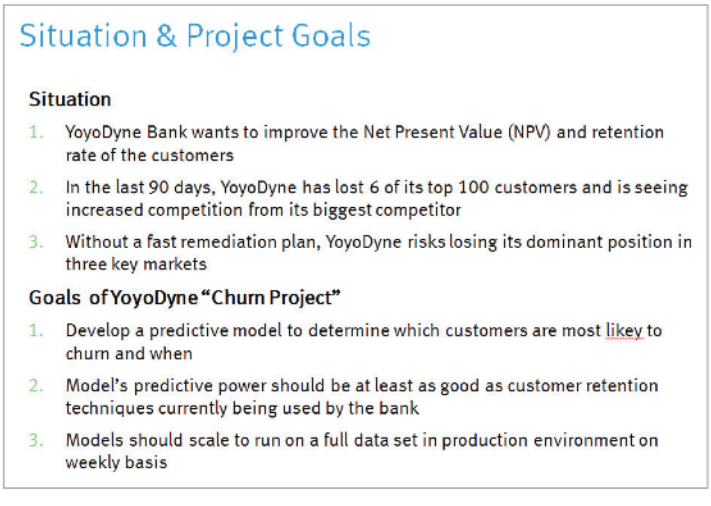


FIGURE 12-6 *Example of Situation & Project Goals slide for YoyoDyne case study*

12.2.3 Main Findings

Write a solid executive summary to portray the main findings of a project. In many cases, the summary may be the only portion of the presentation that hurried managers will read. For this reason, it is imperative to make the language clear, concise, and complete. Those reading the executive summary should be able to grasp the full story of the project and the key insights in a single slide. In addition, this is an opportunity to provide key talking points for the executive sponsor to use to evangelize the project work with others in the customer's organization. Be sure to frame the outcomes of the project in terms of both quantitative and qualitative business value. This is especially important if the presentation is for the project sponsor. The executive summary slide containing the main findings is generally the same for both sponsor and analyst audiences.

Figure 12-7 shows an example of an executive summary slide for the YoyoDyne case study. It is useful to take a closer look at the parts of the slide to make sure it is clear. Keep in mind this is not the only format for conveying the Executive Summary; it varies based on the author's style, although many of the key components are common themes in Executive Summaries.

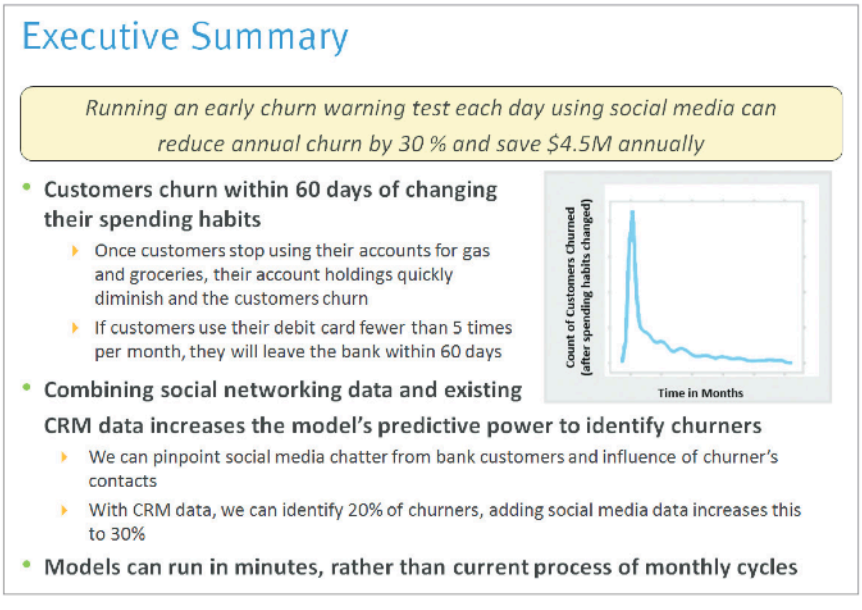


FIGURE 12-7 *Example of Executive Summary slide for YoyoDyne case study*

The key message should be clear and conspicuous at the front of the slide. It can be set apart with color or shading, as shown in Figure 12-8; other techniques can also be used to draw attention to it. The key message may become the single talking point that executives or the project sponsor take away from the project and use to support the team's recommendation for a pilot project, so it needs to be succinct and compelling. To make this message as strong as possible, measure the value of the work and quantify the cost savings, revenue, time savings, or other benefits to make the business impact concrete.

Follow the key message with three major supporting points. Although Executive Summary slides can have more than three major points, going beyond three ideas makes it difficult for people to recall the main points, so it is important to ensure that the ideas remain clear and limited to the few most impactful ideas the team wants the audience to take away from the work that was done. If the author lists ten key points, messages become diluted, and the audience may remember only one or two main points.

In addition, because this is an analytics project, be sure to make one of the key points related to if, and how well, the work will meet the sponsor's service level agreement (SLA) or expectations. Traditionally, the SLA refers to an arrangement between someone providing services, such as an information technology (IT) department or a consulting firm, and an end user or customer. In this case, the SLA refers to system performance, expected uptime of a system, and other constraints that govern an agreement. This term has become less formal and many times conveys system performance or expectations more generally related to performance or timeliness. It is in this sense that SLA is being used here. Namely, in this context, SLA

refers to the expected performance of a system and the intent that the models developed will not adversely impact the expected performance of the system into which they are integrated.

Finally, although it's not required, it is often a good idea to support the main points with a visual or graph. Visual imagery serves to make a visceral connection and helps retain the main message with the reader.

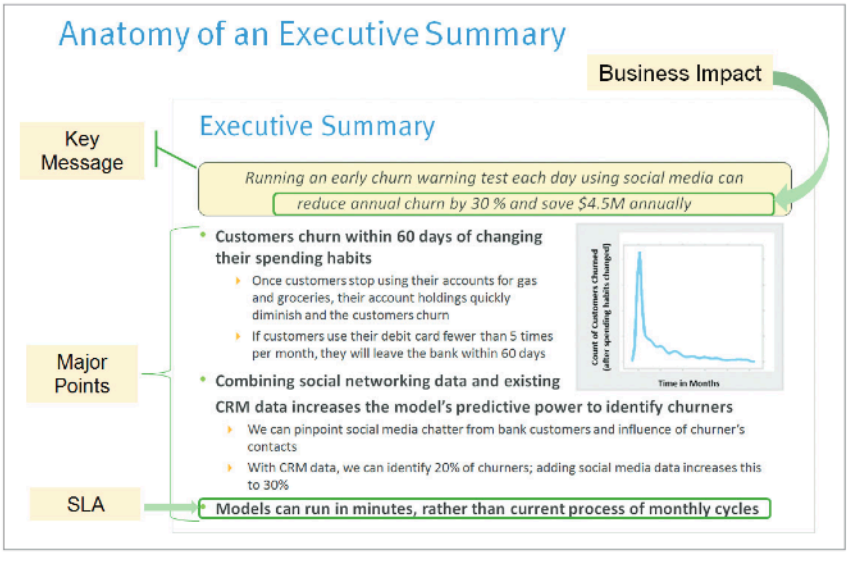


FIGURE 12-8 *Anatomy of an Executive Summary slide*

12.2.4 Approach

In the Approach portion of the presentation, the team needs to explain the methodology pursued on the project. This can include interviews with domain experts, the groups collaborating within the organization, and a few statements about the solution developed. The objective of this slide is to ensure the audience understands the course of action that was pursued well enough to explain it to others within the organization. The team should also include any additional comments related to working assumptions the team followed as it performed the work, because this can be critical in defending why they followed a specific course of action.

When explaining the solution, the discussion should remain at a high level for the project sponsors. If presenting to analysts or data scientists, provide additional detail about the type of model used, including the technology and the actual performance of the model during the tests. Finally, as part of the description of the approach, the team may want to mention constraints from systems, tools, or existing processes and any implications for how these things may need to change with this project.

Figure 12-9 shows an example of how to describe the methodology followed during a data science project to a sponsor audience.

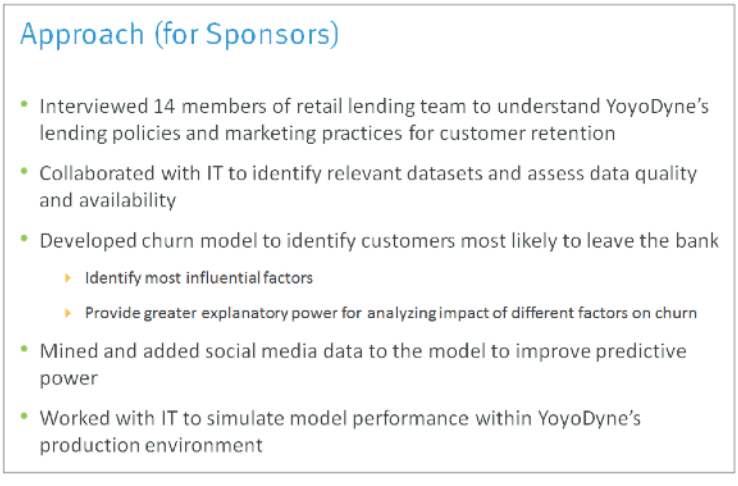


FIGURE 12-9 *Example describing the project methodology for project sponsors*

Note that the third bullet describes the churn model in general terms. Furthermore, the subbullets provide additional details in nontechnical terms. Compare this approach to the variation shown in Figure 12-10.

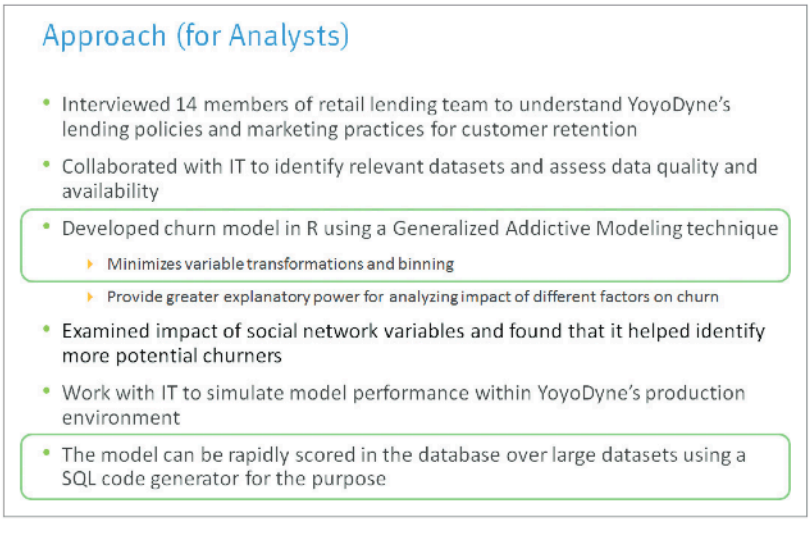


FIGURE 12-10 *Example describing the project methodology for analysts and data scientists*

Figure 12-10 shows a variation on the approach and methodology used in the data science project. In this case, most of the language and description are the same as in the example for project sponsors.

The main difference is that this version contains additional detail regarding the kind of model used and the way the model will score data quickly to meet the SLA. These differences are highlighted in the boxes shown in Figure 12-10.

12.2.5 Model Description

After describing the project approach, teams generally include a description of the model that was used. Figure 12-11 provides the model description for the Yoyodyne Bank example. Although the Model Description slide can be the same for both audiences, the interests and objectives differ for each. For the sponsor, the general methodology needs to be articulated without getting into excessive detail. Convey the basic methodology followed in the team's work to allow the sponsor to communicate this to others within the organization and provide talking points.

Mentioning the scope of the data used is critical. The purpose is to illustrate thoroughness and exude confidence that the team used an approach that accurately portrays its problem and is as free from bias as possible. A key trait of a good data scientist is the ability to be skeptical of one's own work. This is an opportunity to view the work and the deliverable critically and consider how the audience will receive the work. Try to ensure it is an unbiased view of the project and the results.

Assuming that the model will meet the agreed-upon SLAs, mention that the model will meet the SLAs based on the performance of the model within the testing or staging environment. For instance, one may want to indicate that the model processed 500,000 records in 5 minutes to give stakeholders an idea of the speed of the model during run time. Analysts will want to understand the details of the model, including the decisions made in constructing the model and the scope of the data extracts for testing and training. Be prepared to explain the team's thought process on this, as well as the speed of running the model within the test environment.

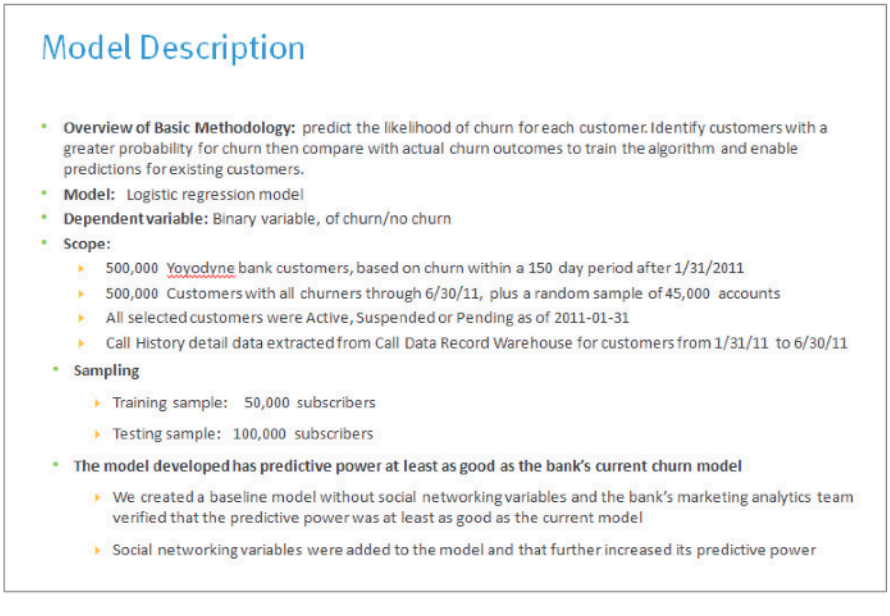


FIGURE 12-11 *Example of a model description for a data science project*

12.2.6 Key Points Supported with Data

The next step is to identify key points based on insights and observations resulting from the data and model scoring results. Find ways to illustrate the key points with charts and visualization techniques, using simpler charts for sponsors and more technical data visualization for analysts and data scientists.

Figure 12-12 shows an example of providing supporting detail regarding the rate of bank customers who would churn in various months. When developing the key points, consider the insights that will drive the biggest business impact and can be defended with data. For project sponsors, use simple charts such as bar charts, which illustrate data clearly and enable the audience to understand the value of the insights. This is also a good point to foreshadow some of the team's recommendations and begin tying together ideas to demonstrate what led to the recommendations and why. In other words, this section supplies the data and foundation for the recommendations that come later in the presentation. Creating clear, compelling slides to show the key points makes the recommendations more credible and more likely to be acted upon by the customer or sponsor.

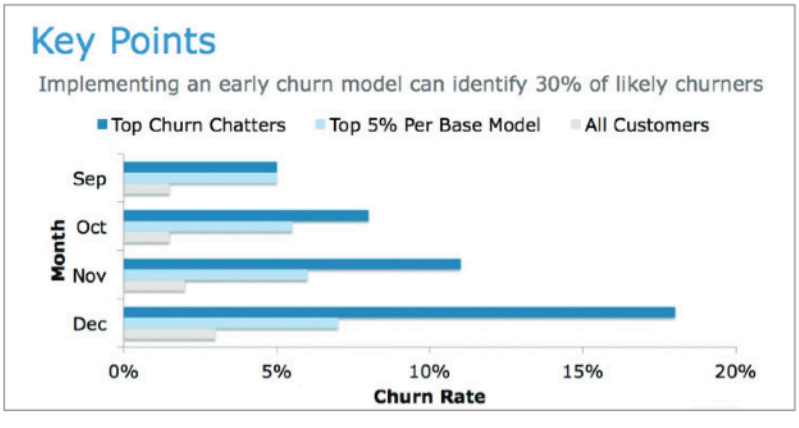


FIGURE 12-12 *Example of a presentation of key points of a data science project shown as a bar chart*

For analyst presentations, use more granular or technical charts and graphs. In this case, appropriate visualization techniques include dot charts, density plots, ROC curves, or histograms of a data distribution to support decisions made in the modeling techniques. Basic concepts of data visualization are discussed later in the chapter.

12.2.7 Model Details

Model details are typically needed by people who have a more technical understanding than the sponsors, such as those who will implement the code, or colleagues on the analytics team. Project sponsors are typically less interested in the model details; they are usually more focused on the business implications of the work rather than the details of the model. This portion of the presentation needs to show the code or main logic of the model, including the model type, variables, and technology used to execute

the model and score the data. The model details segment of the presentation should focus on describing expected model performance and any caveats related to the model performance. In addition, this portion of the presentation should provide a detailed description of the modeling technique, variables, scope, and expected effectiveness of the model.

This is where the team can provide discussion or written details related to the variables used in the model and explain how or why these variables were selected. In addition, the team should share the actual code (or at least an excerpt) developed to explain what was created and how it operates. This also serves to foster discussion related to any additional constraints or implications related to the main logic of the code. In addition, the team can use this section to illustrate details of the key variables and the predictive power of the model, using analyst-oriented charts and graphs, such as histograms, dot charts, density plots, and ROC curves.

Figure 12-13 provides a sample slide describing the data variables, and Figure 12-14 shows a sample slide with a technical graph to support the work.

Model Details

- Candidate variables: 22 from CRM, 154 from call history, and 12 social networking variables
- Through PCA and discussion with domain experts, we reduced ~190 variables to the 9 most predictive of customer churn
- General Additive Model (GAM) model built in R :

```
gam.wsn.by2 <- bam(volchurn.120.p~
s(var1, bs="cs",by=c30,k=length(custom.knots))
+s(var2 , bs="cs",by=c30)
+s(var3, bs="cs",k=5)
+s(var4 ,bs="cs",k=5,by=c30)
+s(tvar5 ,bs="cs",k=5)
+var6
+var7
+ s(var8)
+ s(var9) ,
knots=list(var1=custom.knots),
data=train.df, family=binomial, weight=weight, gamma=1.4)
```

FIGURE 12-13 *Example of model details showing model type and variables*

As part of the model detail description, guidance should be provided regarding the speed with which the model can run in the test environment; the expected performance in a live, production environment; and the technology needed. This kind of discussion addresses how well the model can meet the organization's SLA.

This section of the presentation needs to include additional caveats, assumptions, or constraints of the model and model performance, such as systems or data the model needs to interact with, performance

issues, and ways to feed the outputs of the model into existing business processes. The author of this section needs to describe the relationships of the main variables on the project objectives, such as the effects of key variables on predicting churn, and the relationship of key variables to other variables. The team may even want to make suggestions to improve the model, highlight any risks to introducing bias into the modeling technique, or describe certain segments of the data that may skew the overall predictive power of the methodology.

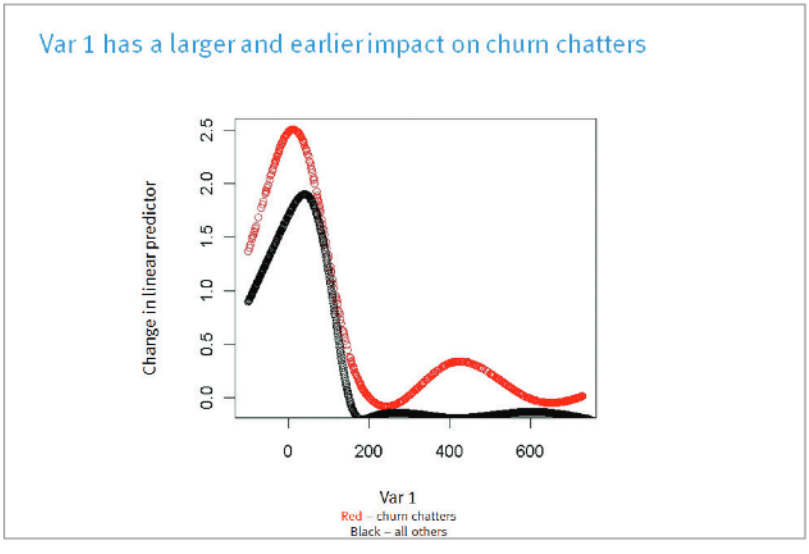


FIGURE 12-14 *Model details comparing two data variables*

12.2.8 Recommendations

The final main component of the presentation involves creating a set of recommendations that include how to deploy the model from a business perspective within the organization and any other suggestions on the rollout of the model's logic. For the Yoyodyne Bank example, Figure 12-15 provides possible recommendations from the project. In this section of the presentation, measuring the impact of the improvements and stating how to leverage that impact within the recommendations are key. For instance, the presentation might mention that every customer retained represents a time savings of six hours for one of the bank's account managers or $50,000 in savings of new account acquisitions, due to marketing costs, sales, and system-related costs.

For a presentation to a project sponsor audience, focus on the business impact of the project, including risks and ROI. Because project sponsors will be most interested in the business impact of the project, the presentation should also provide the sponsor with salient points to help evangelize the work within the organization. When preparing a presentation for analysts, supplement the main set of recommendations with any implications for the modeling or for deployment in a production environment. In either case, the

team should focus on recommending actions to operationalize the work and the benefits the customer will receive because of implementing these recommendations.

Recommendations

- **Implement the model as a pilot, before more wide-scale rollout** – test and learn from initial pilot on performance and precision
  - Addressing these promptly can potentially save more customers from churning over time and also prevent more networking that seems to drive additional churn
  - An early churn warning trigger can be set up based on this model
- **Run the predictive model daily or weekly to be proactive on customer churn**
  - In-database scorer can score large datasets in a matter of minutes and can be run daily
  - Each customer retained via early warning trigger saves 4 hours of account retention efforts & 50k in new account acquisition costs
- **Develop targeted customer surveys to investigate the causes of churn**, which will make the collection of data for investigation into the causes of churn easier

FIGURE 12-15 *Sample recommendations for a data science project*

12.2.9 Additional Tips on the Final Presentation

As a team completes a project and strives to move on to the next one, it must remember to invest adequate time in developing the final presentations. Orienting the audience to the project and providing context is important. On occasion, a team is so immersed in the project that it fails to provide sufficient context for its recommendations and the outputs of the models. A team needs to remember to spell out terminology and acronyms and avoid excessive use of jargon. It should also keep in mind that presentations may be shared extensively; therefore, recipients may not be familiar with the context and the journey the team has gone through over the course of the project.

The story may need to be told multiple times to different audiences, so the team must remain patient in repeating some of the key messages. These presentations should be viewed as opportunities to refine the key messages and evangelize the good work that was done. By this point in the process, the team has invested many hours of work and uncovered insights for the business. These presentations are an opportunity to communicate these projects and build support for future projects. As with most presentations, it is important to gauge the audience to guide shaping the message and the level of detail. Here are several more tips on developing the presentations.

- **Use imagery and visual representations:** Visuals tend to make the presentation more compelling. Also, people recall imagery better than words, because images can have a more visceral impact. These visual representations can be static and interactive data.

- **Make sure the text is mutually exclusive and collectively exhaustive (MECE):** This means having an economy of words in the presentation and making sure the key points are covered but not repeated unnecessarily.
- **Measure and quantify the benefits of the project:** This can be challenging and requires time and effort to do well. This kind of measurement should attempt to quantify benefits that have financial and other benefits in a specific way. Making the statement that a project provided "$8.5M in annual cost savings" is much more compelling than saying it has "great value."
- **Make the benefits of the project clear and conspicuous:** After calculating the benefits of the project, make sure to articulate them clearly in the presentation.

12.2.10 Providing Technical Specifications and Code

In addition to authoring the final presentations, the team needs to deliver the actual code that was developed and the technical documentation needed to support it. The team should consider how the project will affect the end users and the technical people who will need to implement the code. It is recommended that the team think through the implications of its work on the recipients of the code, the kinds of questions they will have, and their interests. For instance, indicating that the model will need to perform real-time monitoring may require extensive changes to an IT runtime environment, so the team may need to consider a compromise of nightly batch jobs to process the data. In addition, the team may need to get the technical team talking with the project sponsor to ensure the implementation and SLA will meet the business needs during the technical deployment.

The team should anticipate questions from IT related to how computationally expensive it will be to run the model in the production environment. If possible, indicate how well the model ran in the test scenarios and whether there are opportunities to tune the model or environment to optimize performance in the production environment.

Teams should approach writing technical documentation for their code as if it were an application programming interface (API). Many times, the models become encapsulated as functions that read a set of inputs in the production environment, possibly perform preprocessing on data, and create an output, including a set of post-processing results.

Consider the inputs, outputs, and other system constraints to enable a technical person to implement the analytical model, even if this person has not had a connection to the data science project up to this point. Think about the documentation as a way to introduce the data that the model needs, the logic it is using, and how other related systems need to interact with it in a production environment for it to operate well. The specifications detail the inputs the code needs and the data format and structures. For instance, it may be useful to specify whether structured data is needed or whether the expected data needs to be numeric or string formats. Describe any transformations that need to be made on the input data before the code can use it, and if scripting was created to perform these tasks. These kinds of details are important when other engineers must modify the code or utilize a different dataset or table, if and when the environment changes.

Regarding exception handling, the team must consider how the code should handle data that is outside the expected data ranges of the model parameters and how it will handle missing data values (Chapter 3, "Review of Basic Data Analytic Methods Using R"), null values, zeros, NAs, or data that is in an unexpected format or type. The technical documentation describes how to treat these exceptions and what implications may emerge on downstream processes. For the model outputs, the team must explain to what extent to post-process the output. For example, if the model returns a value representing

the probability of customer churn, additional logic may be needed to identify the scoring threshold to determine which customer accounts to flag as being at risk of churn. In addition, some provision should be made for adjusting this threshold and training the algorithm, either in an automated learning fashion or with human intervention.

Although the team must create technical documentation, many times engineers and other technical staff receive the code and may try to use it without reading through all the documentation. Therefore, it is important to add extensive comments in the code. This directs the people implementing the code on how to use it, explains what pieces of the logic are supposed to do, and guides other people through the code until they're familiar with it. If the team can do a thorough job adding comments in the code, it is much easier for someone else to maintain the code and tune it in the runtime environment. In addition, it helps the engineers edit the code when their environment changes or they need to modify processes that may be providing inputs to the code or receiving its outputs.

12.3 Data Visualization Basics

As the volume of data continues to increase, more vendors and communities are developing tools to create clear and impactful graphics for use in presentations and applications. Although not exhaustive, Table 12-2 lists some popular tools.

TABLE 12-2 *Common Tools for Data Visualization*

Open Source	Commercial Tools
R (Base package, `lattice`, `ggplot2`)	Tableau
GGobi/Rggobi	Spotfire (TIBCO)
Gnuplot	QlikView
Inkscape	Adobe Illustrator
Modest Maps	
OpenLayers	
Processing	
D3.js	
Weave	

As the volume and complexity of data has grown, users have become more reliant on using crisp visuals to illustrate key ideas and portray rich data in a simple way. Over time, the open source community has developed many libraries to offer more options for portraying graphics data visually. Although this book showed examples primarily using the base package of R, `ggplot2` provides additional options for creating professional-looking data visualization, as does the `lattice` library for R.

Gnuplot and GGobi have a command-line-driven approach to generating data visualization. The genesis of these tools mainly grew out of scientific computing and the need to express complex data visually. GGobi

also has a variant called Rggobi that enables users to access the GGobi functionality with the R software and programming language. There are many open source mapping tools available, including Modest Maps and OpenLayers, both designed for developers who would like to create interactive maps and embed them within their own development projects or on the web. The software programming language development environment, Processing, employs a Java-like language for developers to create professional-looking data visualization. Because it is based on a programming language rather than a GUI, Processing enables developers to create robust visualization and have precise control over the output. D3.js is a JavaScript library for manipulating data and creating web-based visualization with standards, such as Hypertext Markup Language (HTML), Scalable Vector Graphics (SVG), and Cascading Style Sheets (CSS). For more examples of using open source visualization tools, refer to Nathan Yau's website, flowingdata.com [1], or his book ***Visualize This*** [2], which discusses additional methods for creating data representations with open source tools.

Regarding the commercial tools shown in Table 12-2, Tableau, Spotfire (by TIBCO), and QlikView function as data visualization tools and as interactive business intelligence (BI) tools. Due to the growth of data in the past few years, organizations for the first time are beginning to place more emphasis on ease of use and visualization in BI over more traditional BI tools and databases. These tools make visualization easy and have user interfaces that are cleaner and simpler to navigate than their predecessors. Although not traditionally considered a data visualization tool, Adobe Illustrator is listed in Table 12-2 because some professionals use it to enhance visualization made in other tools. For example, some users develop a simple data visualization in R, save the image as a PDF or JPEG, and then use a tool such as Illustrator to enhance the quality of the graphic or stitch multiple visualization work into an infographic. Inkscape is an open source tool used for similar use cases, with much of Illustrator's functionality.

12.3.1 Key Points Supported with Data

It is more difficult to observe key insights when data is in tables instead of in charts. To underscore this point, in ***Say it with Charts,*** Gene Zelazny [3] mentions that to highlight data, it is best to create a visual representation out of it, such as a chart, graph, or other data visualization. The opposite is also true. Suppose an analyst chooses to downplay the data. Sharing it in a table draws less attention to it and makes it more difficult for people to digest.

The way one chooses to organize the visual in terms of the color scheme, labels, and sequence of information also influences how the viewer processes the information and what he perceives as the key message from the chart. The table shown in Figure 12-16 contains many data points. Given the layout of the information, it is difficult to identify the key points at a glance. Looking at 45 years of store opening data can be challenging, as shown in Figure 12-16.

Year	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	Total
SuperDux	1		1	1		1	5	4	4	14	13	14	20	14	17	29	24	37	33	117	42	65	79	81	90	92	82	86	106	72	62	62	40	49	22	26	33	47	78	71	67	64	91	91	33	1980
DipBox						1		1	1	1	4	5	5	5	10	10	10	6	21	33	21	22	20	29	31	50	43	45	72	91	76	94	67	80	31	34	33	33	27	35	47	32	39	27	4	1196
Total		1	1	1		2	5	5	5	15	17	19	25	19	27	39	34	43	54	150	63	87	99	110	121	142	125	131	178	163	138	156	107	129	53	60	66	80	105	106	114	96	130	118	37	3176

FIGURE 12-16 *Forty-five years of store opening data*

Even showing somewhat less data is still difficult to read through for most people. Figure 12-17 hides the first 10 years, leaving 35 years of data in the table.

Year	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	Total
SuperBox	13	14	20	14	17	29	24	37	33	117	42	65	79	81	90	92	82	86	106	72	62	62	40	49	22	26	33	47	78	71	67	64	91	91	33	1980
BigBox	4	5	5	5	10	10	10	6	21	33	21	22	20	29	31	50	43	45	72	91	76	94	67	80	31	34	33	33	27	35	47	32	39	27	4	1196
Total	17	19	25	19	27	39	34	43	54	150	63	87	99	110	121	142	125	131	178	163	138	156	107	129	53	60	66	80	105	106	114	96	130	118	37	3176

FIGURE 12-17 *Thirty-five years of store opening data*

As most readers will observe, it is challenging to make sense of data, even at relatively small scales. There are several observations in the data that one may notice, if one looks closely at the data tables:

- BigBox experienced strong growth in the 1980s and 1990s.
- By the 1980s, BigBox began adding more SuperBox stores to its mix of chain stores.
- SuperBox stores outnumber BigBox stores nearly 2 to 1 in aggregate.

Depending on the point trying to be made, the analyst must take care to organize the information in a way that intuitively enables the viewer to take away the same main point that the author intended. If the analyst fails to do this effectively, the person consuming the data must guess at the main point and may interpret something different from what was intended.

Figure 12-18 shows a map of the United States, with the points representing the geographic locations of the stores. This map is a more powerful way to depict data than a small table would be. The approach is well suited to a sponsor audience. This map shows where the BigBox store has market saturation, where the company has grown, and where it has SuperBox stores and other BigBox stores, based on the color and shading. The visualization in Figure 12-18 clearly communicates more effectively than the dense tables in Figure 12-16 and Figure 12-17. For a sponsor audience, the analytics team can also use other simple visualization techniques to portray data, such as bar charts or line charts.

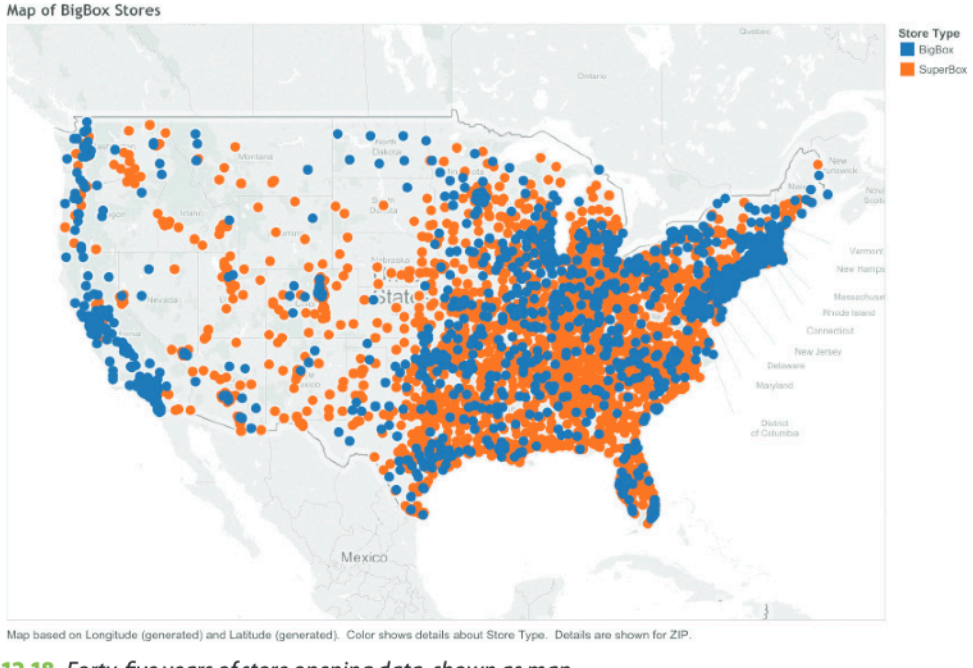


FIGURE 12-18 *Forty-five years of store opening data, shown as map*

12.3.2 Evolution of a Graph

Visualization allows people to portray data in a more compelling way than tables of data and in a way that can be understood on an intuitive, precognitive level. In addition, analysts and data scientists can use visualization to interact with and explore data. Following is an example of the steps a data scientist may go through in exploring pricing data to understand the data better, model it, and assess whether a current pricing model is working effectively. Figure 12-19 shows a distribution of pricing data as a user score reflecting price sensitivity.

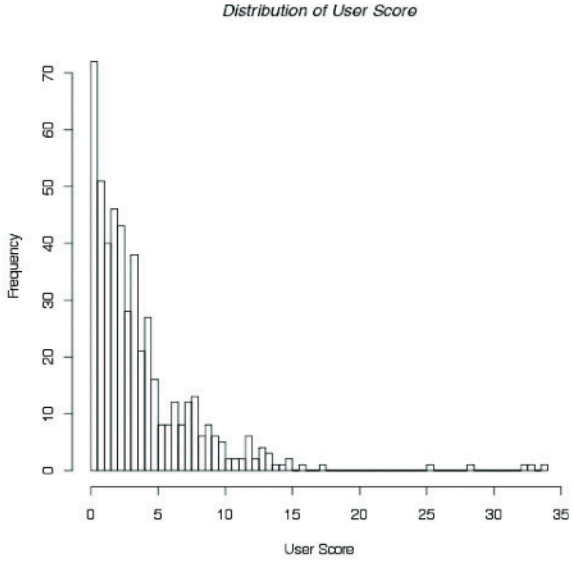


FIGURE 12-19 *Frequency distribution of user scores*

A data scientist's first step may be to view the data as a raw distribution of the pricing levels of users. Because the values have a long tail to the right, in Figure 12-19, it may be difficult to get a sense of how tightly clustered the data is between user scores of zero and five.

To understand this better, a data scientist may rerun this distribution showing a log distribution (Chapter 3) of the user score, as demonstrated in Figure 12-20.

This shows a less skewed distribution that may be easier for a data scientist to understand. Figure 12-21 illustrates a rescaled view of Figure 12-20, with the median of the distribution around 2.0. This plot provides the distribution of a new user score, or index, that may gauge the level of price sensitivity of a user when expressed in log form.

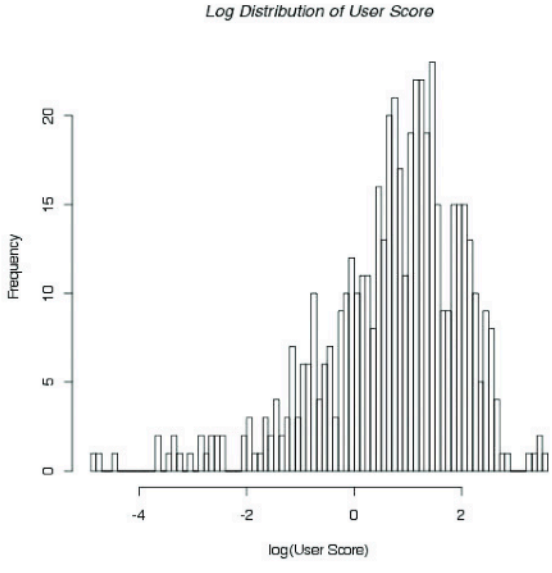


FIGURE 12-20 *Frequency distribution with log of user score*

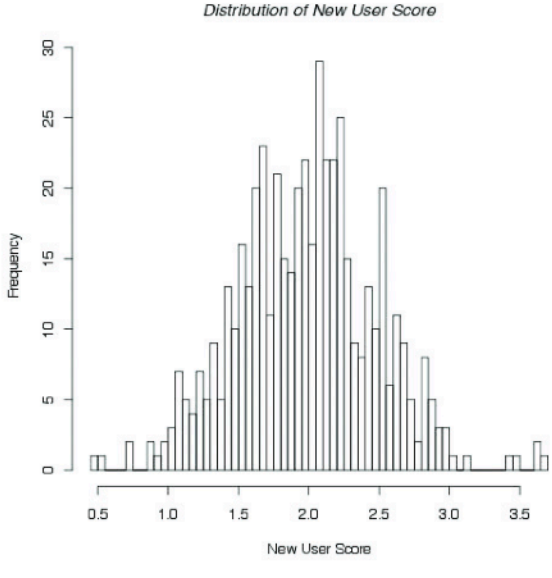


FIGURE 12-21 *Frequency distribution of new user scores*

Another idea may be to analyze the stability of price distributions over time to see if the prices offered to customers are stable or volatile. As shown in a graphic such as Figure 12-22, the prices appear to be stable. In this example, the user score of pricing remains within a tight band between two and three regardless of the time in days. In other words, the time in which a customer purchases a given product does not significantly influence the price she is willing to pay, as expressed by the user score, shown on the y-axis.

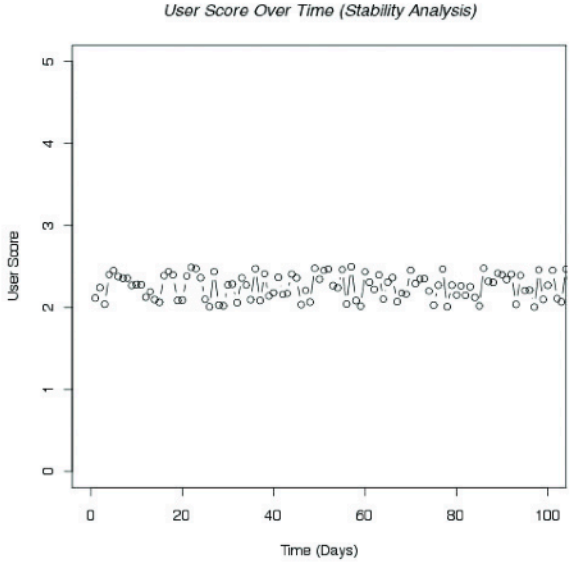


FIGURE 12-22 *Graph of stability analysis for pricing*

By this point the data scientist has learned the following about this example and made several observations about the data:

- Most user scores are between two and three in terms of their price sensitivity.
- After taking the log value of the user scores, a new user scoring index was created, which recentered the data values around the center of the distribution.
- The pricing scores appear to be stable over time, as the duration of the customer does not seem to have significant influence on the user pricing score. Instead, it appears to be relatively constant over time, within a small band of user scores.

At this point, the analysts may want to explore the range of price tiers offered to customers. Figures 12-22 and 12-23 demonstrate examples of the price tiering currently in place within the customer base.

Figure 12-23 shows the price distribution for a customer base. In this example, loyalty score and price are positively correlated; as the loyalty score increases, so do the prices that the customers are willing to pay. It may seem like a strange phenomenon that the most loyal customers in this example are willing to pay higher prices, but the reality is that customers who are very loyal tend to be less sensitive to price fluctuations or increases. The key, however, is to understand which customers are highly loyal so that appropriate pricing can be charged to the right groups of people.

Figure 12-24 shows a variation on 12-23. In this case, the new graphic portrays the same customer price tiers, but this time a rug representation (Chapter 3) has been added at the bottom to reflect the distribution of the data points.

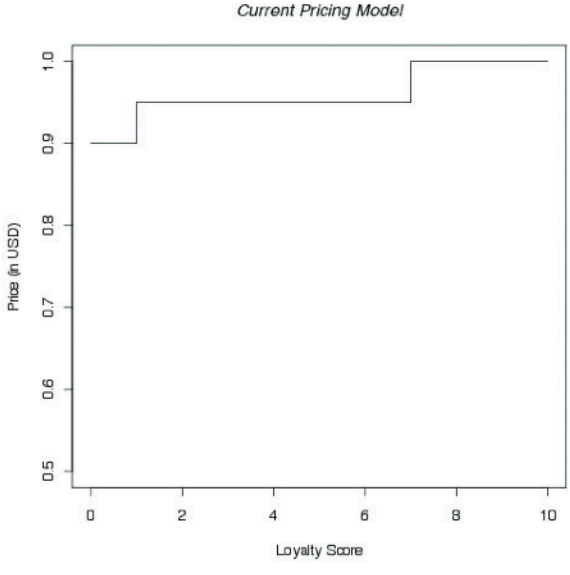


FIGURE 12-23 *Graph comparing the price in U.S. dollars with a customer loyalty score*

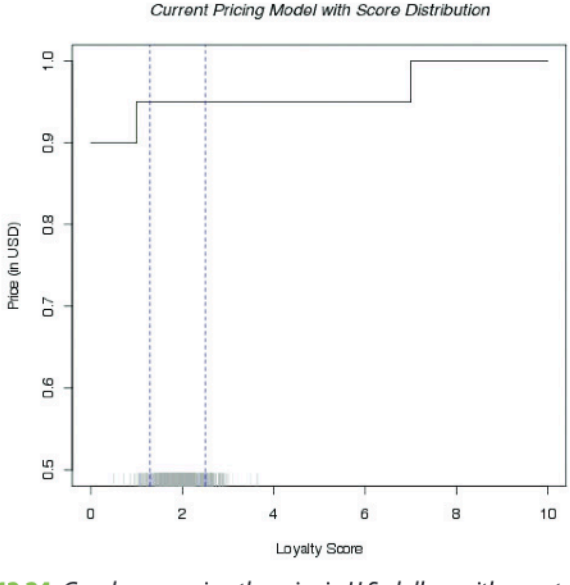


FIGURE 12-24 *Graph comparing the price in U.S. dollars with a customer loyalty score (with rug representation)*

This rug indicates that the majority of customers in this example are in a tight band of loyalty scores, between about 1 and 3 on the x-axis, all of which offered the same set of prices, which are high (between 0.9 and 1.0 on the y-axis). The y-axis in this example may represent a pricing score, or the raw value of a customer in millions of dollars. The important aspect is to recognize that the pricing is high and is offered consistently to most of the customers in this example.

Based on what was shown in Figure 12-25, the team may decide to develop a new pricing model. Rather than offering static prices to customers regardless of their level of loyalty, a new pricing model might offer more dynamic price points to customers. In this visualization, the data shows the price increases as more of a curvilinear slope relative to the customer loyalty score. The rug at the bottom of the graph indicates that most customers remain between 1 and 3 on the x-axis, but now rather than offering all these customers the same price, the proposal suggests offering progressively higher prices as customer loyalty increases. In one sense, this may seem counterintuitive. It could be argued that the best prices should be offered to the most loyal customers. However, in reality, the opposite is often the case, with the most attractive prices being offered to the least loyal customers. The rationale is that loyal customers are less price sensitive and may enjoy the product and stay with it regardless of small fluctuations in price. Conversely, customers who are not very loyal may defect unless they are offered more attractive prices to stay. In other words, less loyal customers are more price sensitive. To address this issue, a new pricing model that accounts for this may enable an organization to maximize revenue and minimize attrition by offering higher prices to more loyal customers and lower prices to less loyal customers. Creating an iterative depicting the data visually allows the viewer to see these changes in a more concrete way than by looking at tables of numbers or raw values.

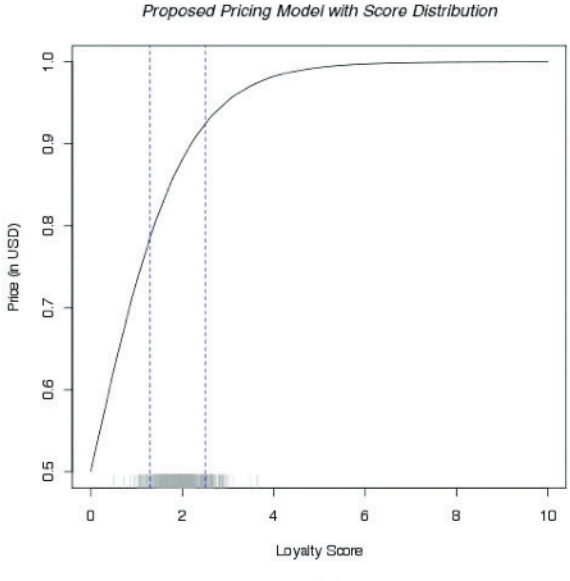


FIGURE 12-25 *New proposed pricing model compared to prices in U.S. dollars with rug*

Data scientists typically iterate and view data in many different ways, framing hypotheses, testing them, and exploring the implications of a given model. This case explores visual examples of pricing distributions, fluctuations in pricing, and the differences in price tiers before and after implementing a new model to optimize price. The visualization work illustrates how the data may look as the result of the model, and helps a data scientist understand the relationships within the data at a glance.

The resulting graph in the pricing scenario appears to be technical regarding the distribution of prices throughout a customer base and would be suitable for a technical audience composed of other data scientists. Figure 12-26 shows an example of how one may present this graphic to an audience of other data scientists or data analysts. This demonstrates a curvilinear relationship between price tiers and customer loyalty when expressed as an index. Note that the comments to the right of the graph relate to the precision of the price targeting, the amount of variability in robustness of the model, and the expectations of model speed when run in a production environment.

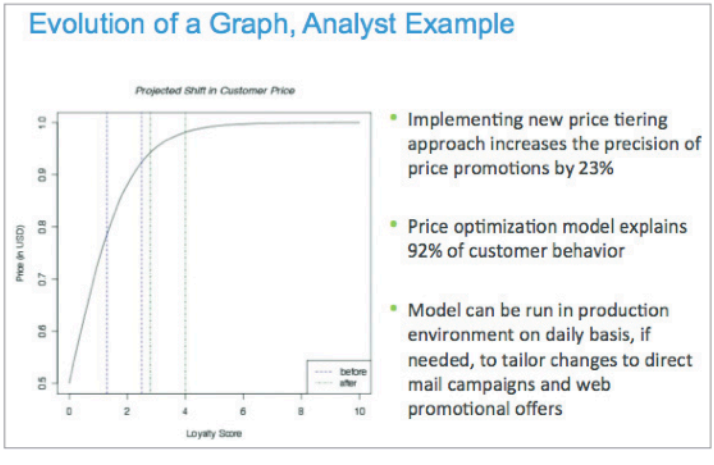


FIGURE 12-26 *Evolution of a graph, analyst example with supporting points*

Figure 12-27 portrays another example of the output from the price optimization project scenario, showing how one may present this to an audience of project sponsors. This demonstrates a simple bar chart depicting the average price per customer or user segment. Figure 12-27 shows a much simpler-looking visual than Figure 12-26. It clearly portrays that customers with lower loyalty scores tend to get lower prices due to targeting from price promotions. Note that the right side of the image focuses on the business impact and cost savings rather than the detailed characteristics of the model.

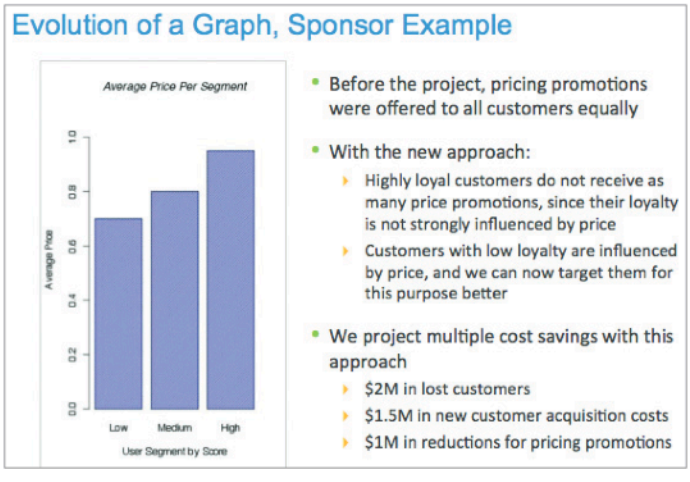


FIGURE 12-27 *Evolution of a graph, sponsor example*

The comments to the right side of the graphic in Figure 12-27 explain the impact of the model at a high level and the cost savings of implementing this approach to price optimization.

12.3.3 Common Representation Methods

Although there are many types of data visualizations, several fundamental types of charts portray data and information. It is important to know when to use a particular type of chart or graph to express a given kind of data. Table 12-3 shows some basic chart types to guide the reader in understanding that different types of charts are more suited to a situation depending on specific kinds of data and the message the team is attempting to portray. Using a type of chart for data it is not designed for may look interesting or unusual, but it generally confuses the viewer. The objective for the author is to find the best chart for expressing the data clearly so the visual does not impede the message, but rather supports the reader in taking away the intended message.

TABLE 12-3 *Common Representation Methods for Data and Charts*

Data for Visualization	Type of Chart
Components (parts of whole)	Pie chart
Item	Bar chart
Time series	Line chart
Frequency	Line chart or histogram
Correlation	Scatterplot, side-by-side bar charts

Table 12-3 shows the most fundamental and common data representations, which can be combined, embellished, and made more sophisticated depending on the situation and the audience. It is recommended

that the team consider the message it is trying to communicate and then select the appropriate type of visual to support the point. Misusing charts tends to confuse an audience, so it is important to take into account the data type and desired message when choosing a chart.

Pie charts are designed to show the components, or parts relative to a whole set of things. A pie chart is also the most commonly misused kind of chart. If the situation calls for using a pie chart, employ it only when showing only 2–3 items in a chart, and only for sponsor audiences.

Bar charts and line charts are used much more often and are useful for showing comparisons and trends over time. Even though people use vertical bar charts more often, horizontal bar charts allow an author more room to fit the text labels. Vertical bar charts tend to work well when the labels are small, such as when showing comparisons over time using years.

For frequency, histograms are useful for demonstrating the distribution of data to an analyst audience or to data scientists. As shown in the pricing example earlier in this chapter, data distributions are typically one of the first steps when visualizing data to prepare for model planning. To qualitatively evaluate correlations, scatterplots can be useful to compare relationships among variables.

As with any presentation, consider the audience and level of sophistication when selecting the chart to convey the intended message. These charts are simple examples but can easily become more complex when adding data variables, combining charts, or adding animation where appropriate.

12.3.4 How to Clean Up a Graphic

Many times software packages generate a graphic for a dataset, but the software adds too many things to the graphic. These added visual distractions can make the visual appear busy or otherwise obscure the main points that are to be made with the graphic. In general, it is a best practice to strive for simplicity when creating graphics and data visualization graphs. Knowing how to simplify graphics or clean up a messy chart is helpful for conveying the key message as clearly as possible. Figure 12-28 portrays a line chart with several design problems.

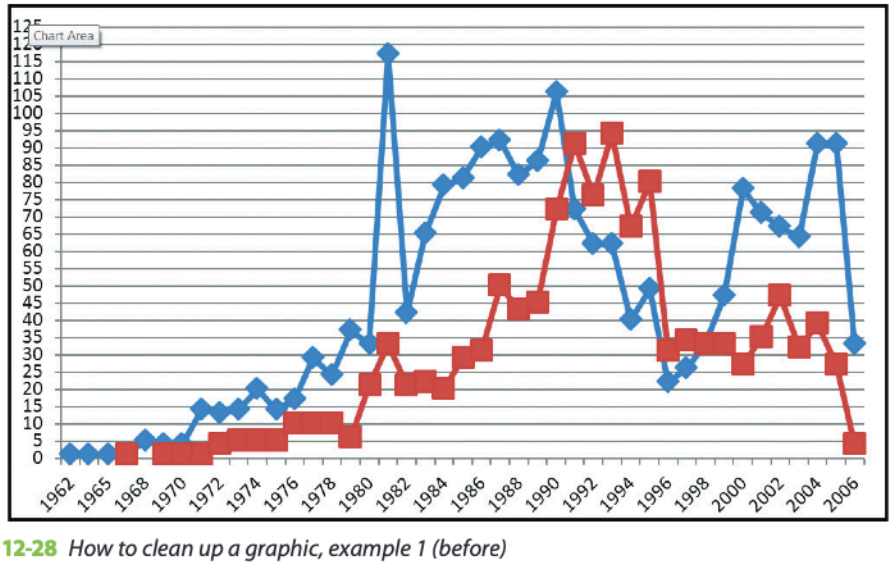


FIGURE 12-28 *How to clean up a graphic, example 1 (before)*

How to Clean Up a Graphic

The line chart shown in Figure 12-28 compares two trends over time. The chart looks busy and contains a lot of chart junk that distracts the viewer from the main message. *Chart junk* refers to elements of data visualization that provide additional materials but do not contribute to the data portion of the graphic. If chart junk were removed, the meaning and understanding of the graphic would not be diminished; it would instead be made clearer. There are five main kinds of "chart junk" in Figure 12-28:

- **Horizontal grid lines:** These serve no purpose in this graphic. They do not provide additional information for the chart.
- **Chunky data points:** These data points represented as large square blocks draw the viewer's attention to them but do not represent any specific meaning aside from the data points themselves.
- **Overuse of emphasis colors in the lines and border:** The border of the graphic is a thick, bold line. This forces the viewer's attention to the perimeter of the graphic, which contains no information value. In addition, the lines showing the trends are relatively thick.
- **No context or labels:** The chart contains no legend to provide context as to what is being shown. The lines also lack labels to explain what they represent.
- **Crowded axis labels:** There are too many axis labels, so they appear crowded. There is no need for labels on the y-axis to appear every five units or for values on the x-axis to appear every two units. Shown in this way, the axis labels distract the viewer from the actual data that is represented by the trend lines in the chart.

The five forms of chart junk in Figure 12-28 are easily corrected, as shown in Figure 12-29. Note that there is no clear message associated with the chart and no legend to provide context for what is shown in Figure 12-28.

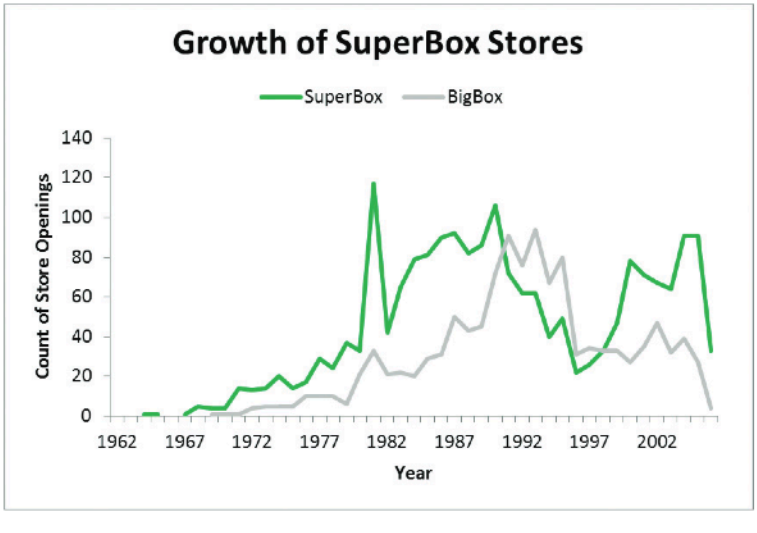


FIGURE 12-29 *How to clean up a graphic, example 1 (after)*

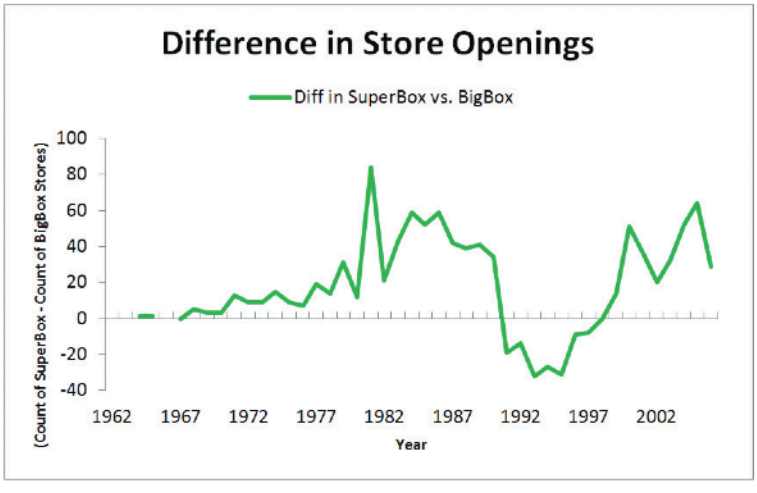


FIGURE 12-30 *How to clean up a graphic, example 1 (alternate "after" view)*

Figures 12-29 and 12-30 portray two examples of cleaned-up versions of the chart shown in Figure 12-28. Note that the problems with chart junk have been addressed. There is a clear label and title for each chart to reinforce the message, and color has been used in ways to highlight the point the author is trying to make. In Figure 12-29, a strong, green color is shown to represent the count of SuperBox stores, because this is where the viewer's focus should be drawn, whereas the count of BigBox stores is shown in a light gray color.

In addition, note the amount of white space being used in each of the two charts shown in Figures 12-29 and 12-30. Removing grid lines, excessive axes, and the visual noise within the chart allows clear contrast between the emphasis colors (the green line charts) and the standard colors (the lighter gray of the BigBox stores). When creating charts, it is best to draw most of the main visuals in standard colors, light tones, or color shades so that stronger emphasis colors can highlight the main points. In this case, the trend of BigBox stores in light gray fades into the background but does not disappear, while making the SuperBox stores trend in a darker gray (bright green in the online chart) makes it prominent to support the message the author is making about the growth of the SuperBox stores.

An alternative to Figure 12-29 is shown in Figure 12-30. If the main message is to show the difference in the growth of new stores, Figure 12-30 can be created to further simplify Figure 12-28 and graph only the difference between SuperBox stores compared to regular BigBox stores. Two examples are shown to illustrate different ways to convey the message, depending on what it is the author of these charts would like to emphasize.

How to Clean Up a Graphic, Second Example

Another example of cleaning up a chart is portrayed in Figure 12-31. This vertical bar chart suffers from more of the typical problems related to chart junk, including misuse of color schemes and lack of context.

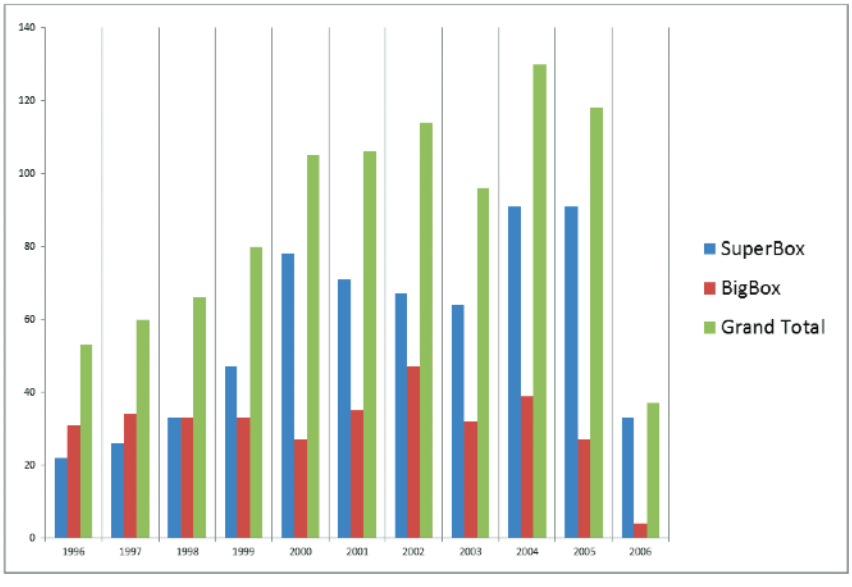


FIGURE 12-31 *How to clean up a graphic, example 2 (before)*

There are five main kinds of chart junk in Figure 12-31:

- **Vertical grid lines:** These vertical grid lines are not needed in this graphic. They provide no additional information to help the viewer understand the message in the data. Instead, these vertical grid lines only distract the viewer from looking at the data.
- **Too much emphasis color:** This bar chart uses strong colors and too much high-contrast dark gray-scale. In general, it is best to use subtle tones, with a low contrast gray as neutral color, and then emphasize the data underscoring the key message in a dark tone or strong color.
- **No chart title:** Because the graphic lacks a chart title, the viewer is not oriented to what he is viewing and does not have proper context.
- **Legend at right restricting chart space:** Although there is a legend for the chart, it is shown on the right side, which causes the vertical bar chart to be compressed horizontally. The legend would make more sense placed across the top, above the chart, where it would not interfere with the data being expressed.
- **Small labels:** The horizontal and vertical axis labels have appropriate spacing, but the font size is too small to be easily read. These should be slightly larger to be easily read, while not appearing too prominent.

Figures 12-32 and 12-33 portray two examples of cleaned-up versions of the chart shown in Figure 12-31. The problems with chart junk have been addressed. There is a clear label and title for each chart to reinforce the message, and appropriate colors have been used in ways to highlight the point the author is trying to make. Figures 12-32 and 12-33 show two options for modifying the graphic, depending on the main point the presenter is trying to make.

Figure 12-32 shows strong emphasis color (dark blue) representing the SuperBox stores to support the chart title: Growth of SuperBox Stores.

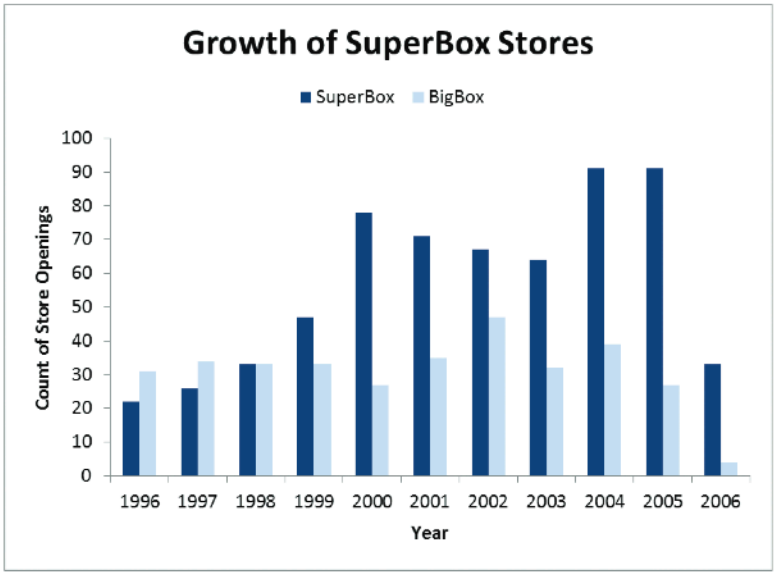


FIGURE 12-32 *How to clean up a graphic, example 2 (after)*

Suppose the presenter wanted to talk about the total growth of BigBox stores instead. A line chart showing the trends over time would be a better choice, as shown in Figure 12-33.

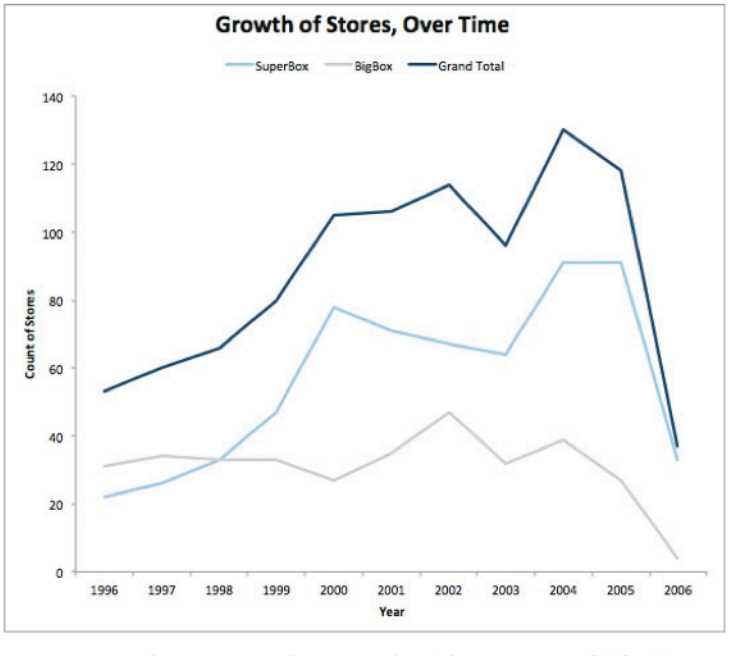


FIGURE 12-33 *How to clean up a graphic, example 2 (alternate view of "after")*

In both cases, the noise and distractions within the chart have been removed. As a result, the data in the bar chart for providing context has been deemphasized, while other data has been made more prominent because it reinforces the key point as stated in the chart's title.

12.3.5 Additional Considerations

As stated in the previous examples, the emphasis should be on simplicity when creating charts and graphs. Create graphics that are free of chart junk and utilize the simplest method for portraying graphics clearly. The goal of data visualization should be to support the key messages being made as clearly as possible and with few distractions.

Similar to the idea of removing chart junk is being cognizant of the data-ink ratio. *Data-ink* refers to the actual portion of a graphic that portrays the data, while *non-data ink* refers to labels, edges, colors, and other decoration. If one imagined the ink required to print a data visualization on paper, the data-ink ratio could be thought of as (data-ink)/(total ink used to print the graphic). In other words, the greater the ratio of data-ink in the visual, the more data rich it is and the fewer distractions it has [4].

Avoid Using Three-Dimensions in Most Graphics

One more example where people typically err is in adding unnecessary shading, depth, or dimensions to graphics. Figure 12-34 shows a vertical bar chart with two visible dimensions. This example is simple and easy to understand, and the focus is on the data, not the graphics. The author of the chart has chosen to highlight the SuperBox stores in a dark blue color, while the BigBox bars in the chart are in a lighter blue. The title is about the growth of SuperBox stores, and the SuperBox bars in the chart are in a dark, high-contrast shade that draws the viewer's attention to them.

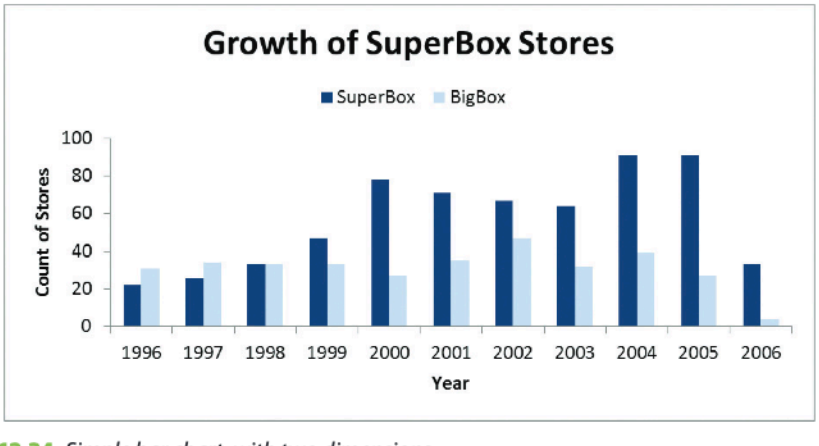


FIGURE 12-34 *Simple bar chart, with two dimensions*

Compare Figure 12-34 to Figure 12-35, which shows a three-dimensional chart. Figure 12-35 shows the original bar chart at an angle, with some attempt at showing depth. This kind of three-dimensional perspective makes it more difficult for the viewer to gauge the actual data and the scaling becomes deceptive.

Three-dimensional charts often distort scales and axes, and impede viewer cognition. Adding a third dimension for depth in Figure 12-35, does not make it fancier, just more difficult to understand.

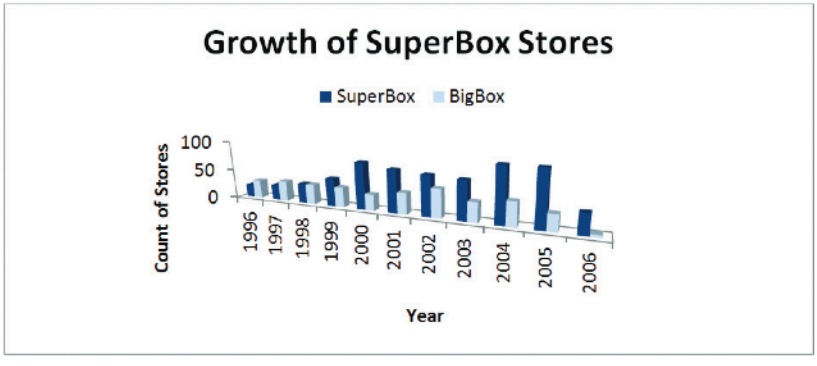


FIGURE 12-35 *Misleading bar chart, with three dimensions*

The charts in Figures 12-34 and 12-35 portray the same data, but it is more difficult to judge the actual height of the bars in Figure 12-35. Moreover, the shadowing and shape of the chart cause most viewers to spend time looking at the perspective of the chart rather than the height of the bars, which is the key message and purpose of this data visualization.

Summary

Communicating the value of analytical projects is critical for sustaining the momentum of a project and building support within organizations. This support is instrumental in turning a successful project into a system or integrating it properly into an existing production environment. Because an analytics project may need to be communicated to audiences with mixed backgrounds, this chapter recommends creating four deliverables to satisfy most of the needs of various stakeholders.

- A presentation for a project sponsor
- A presentation for an analytical audience
- Technical specification documents
- Well-annotated production code

Creating these deliverables enables the analytics project team to communicate and evangelize the work that it did, whereas the code and technical documentation assists the team that wants to implement the models within the production environment.

This chapter illustrates the importance of selecting clear and simple visual representations to support the key points in the final presentations or for portraying data. Most data representations and graphs can be improved by simply removing the visual distractions. This means minimizing or removing chart junk, which distracts the viewer from the main purpose of a chart or graph and does not add information value.

Following several common-sense principles about minimizing distractions in slides and visualizations, communicating clearly and simply, using color in a deliberate way, and taking time to provide context addresses most of the common problems in charts and slides. These few guidelines support the creation of crisp, clear visuals that convey the key messages.

In most cases, the best data visualizations use the simplest, clearest visual to illustrate the key point. Avoid unnecessary embellishment and focus on trying to find the best, simplest method for transmitting the message. Context is critical to orient the viewer to a chart or graph, because people have immediate reactions to imagery on a precognitive level. To this end, make sure to employ thoughtful use of color and orient the viewer with scales, legends, and axes.

Exercises

1. Describe four common deliverables for an analytics project.
2. What is the focus of a presentation for a project sponsor?
3. Give examples of appropriate charts to create in a presentation for other data analysts and data scientists as part of a final presentation. Explain why the charts are appropriate to show each audience.
4. Explain what types of graphs would be appropriate to show data changing over time and why.
5. As part of operationalizing an analytics project, which deliverable would you expect to provide to a Business Intelligence analyst?

References and Further Reading

Following are additional references to learn more about best practices for giving presentations.

- ***Say It with Charts*, by Gene Zelazny[3]:** Simple reference book on how to select the right graphical approach for portraying data and for ensuring the message is clearly conveyed in presentations.
- ***Pyramid Principle*, by Barbara Minto [5]:** Minto pioneered the approach for constructing logical structures for presentations in threes: three sections to the presentations, each with three main points. This teaches people how to weave a story out of the disparate pieces.
- ***Presentation Zen*, by Garr Reynolds [6]:** Teaches how to convey ideas simply and clearly and use imagery in presentations. Shows many before and after versions of graphics and slides.
- ***Now You See It*, by Stephen Few [4]:** Provides many examples for matching the appropriate kind of data visualization to a given dataset.

Bibliography

[1] N. Yau, "flowingdata.com" [Online]. Available: `http://flowingdata.com`.
[2] N. Yau, *Visualize This*, Indianapolis: Wiley, 2011.
[3] G. Zelazny, *Say It with Charts: The Executive's Guide to Visual Communication*, McGraw-Hill, 2001.
[4] S. Few, *Now You See It: Simple Visualization Techniques for Quantitative Analysis*, Analytics Press, 2009.
[5] B. Minto, *The Minto Pyramid Principle: Logic in Writing, Thinking, and Problem Solving*, Prentice Hall, 2010.
[6] G. Reynolds, *Presentation Zen: Simple Ideas on Presentation Design and Delivery*, Berkeley: New Riders, 2011.

Index

Numbers & Symbols

\ (backward slash) as separator, 69
/ (forward slash) as separator, 69
1-itemsets, 147
2-itemsets, 148–149
3 Vs (volume, variety, velocity), 2–3
3-itemsets, 149–150
4-itemsets, 150–151

A

accuracy, 225
ACF (autocorrelation function), 236–237
ACME text analysis example, 259–260
    raw text collection, 260–263
aggregates (SQL)
    ordered, 351–352
    user-defined, 347–351
aggregators of data, 18
AIE (Applied Information Economics), 28
algorithms
    clustering, 134–135
    decision trees, 197–200
        C4.5, 203–204
        CART, 204
        ID3, 203
Alphine Miner, 42
alternative hypothesis, 102–103
analytic projects
    Approach, 369–371
    BI analyst, 362
    business users, 361
    code, 362, 376–377
    communication, 360–361
    data engineer, 362
    data scientists, 362
    DBA (Database Administrator), 362
    deliverables, 362–364
        audiences, 364–365
        core material, 364–365
    key points, 372
    Main Findings, 367–369
    model description, 371
    model details, 372–374
    operationalizing, 360–361
    outputs, 361
    presentations, 362
    Project Goals, 365–367
    project manager, 362
    project sponsor, 361
    recommendations, 374–375
    stakeholders, 361–362
    technical specifications, 376–377
analytic sandboxes. *See* sandboxes
analytical architecture, 13–15
analytics
    business drivers, 11
    examples, 22–23
    new approaches, 16–19
ANOVA, 110–114
Anscombe's quartet, 82–83
`aov( )` function, 78
Apache Hadoop. *See* Hadoop
APIs (application programming interfaces), Hadoop, 304–305
`apriori( )` function, 146, 152–157
Apriori algorithm, 139
    grocery store example, 143
        `Groceries` dataset, 144–146
        itemset generation, 146–151
        rule generation, 152–157
    itemsets, 139, 140–141
        counting, 158
    partitioning and, 158
    sampling and, 158
    transaction reduction and, 158
architecture, analytical, 13–15
`arima( )` function, 246
ARIMA (Autoregressive Integrated Moving Average) model, 236
    ACF, 236–237
    ARMA model, 241–244
    autoregressive models, 238–239
    building, 244–252
    cautions, 252–253
    constant variance, 250–251
    evaluating, 244–252
    fitted time series models, 249–250
    forecasting, 251–252
    moving average models, 239–241
    normality, 250–251
    PACF, 238–239
    reasons to choose, 252–253
    seasonal autoregressive integrated moving average model, 243–244
    VARIMA, 253
ARMA (Autoregressive Moving Average) model, 241–244
`array( )` function, 74
arrays
    matrices, 74
    R, 74–75
association rules, 138–139
    application, 143
    candidate rules, 141–142
    diagnostics, 158

testing and, 157–158
validation, 157–158
attributes
objects, k-means, 130–131
R, 71–72
AUC (area under the curve), 227
autoregressive models, 238–239
averages, moving average models, 239–241

B

bagging, 228
bag-of-words in text analysis, 265–266
banking, 18
`barplot( )` function, 88
barplots, 93–94
Bayes' Theorem, 212–214. *See also* naïve Bayes
conditional probability, 212
BI (business intelligence)
analytical tools, 10
versus Data Science, 12–13
Big Data
3 Vs, 2–3
analytics, examples, 22–23
characteristics, 2
definitions, 2–3
drivers, 15–16
ecosystem, 16–19
key roles, 19–22
McKinsey & Co. on, 3
volume, 2–3
boosting, 228–229
bootstrap aggregation, 228
box-and-whisker plots, 95–96
Box-Jenkins methodology, 235–236
ARIMA model, 236
branches (decision trees), 193
Brown Corpus, 267–268
business drivers for analytics, 11
Business Intelligence Analyst, Operationalize phase, 52
Business Intelligence Analyst role, 27
Business User, Operationalize phase, 52
Business User role, 27
buyers of data, 18

C

C4.5 algorithm, 203–204
cable TV providers, 17
candidate rules, 141–142
CART (Classification And Regression Trees), 204
case folding in text analysis, 264–265
categorical algorithms, 205
categorical variables, 170–171
`cbind( )` function, 78
centroids, 120–122
starting positions, 134
character data types, R, 72
charts, 386–387
churn rate (customers), 120
logistic regression, 180–181
`class( )` function, 72
classification
bagging, 228
boosting, 228–229
bootstrap aggregation, 228
decision trees, 192–193
algorithms, 197–200, 203–204
binary decisions, 206
branches, 193
categorical attributes, 205
classification trees, 193
correlated variables, 206
decision stump, 194
evaluating, 204–206
greedy algorithm, 204
internal nodes, 193
irrelevant variables, 205
nodes, 193
numerical attributes, 205
R and, 206–211
redundant variables, 206
regions, 205
regression trees, 193
root, 193
short trees, 194
splits, 193, 194, 197, 200–203
structure, 205
uses, 194
naïve Bayes, 211–212
Bayes' theorem, 212–214
diagnostics, 217–218
naïve Bayes classifier, 214–217
R and, 218–224
smoothing, 217
classification trees, 193
classifiers
accuracy, 225
diagnostics, 224–228
recall, 225
clickstream, 9
clustering, 118
algorithms, 134–135
centroids, 120–122

starting positions, 134
diagnostics, 128–129
k-means, 118–119
algorithm, 120–122
customer segmentation, 120
image processing and, 119
medical uses, 119
reasons to choose, 130–134
rescaling, 133–134
units of measure, 132–133
labels, 127
number of clusters, 123–127
code, technical specifications in project, 376–377
coefficients, linear regression, 169
combiners, 302–303
Communicate Results phase of lifecycle, 30, 49–50
components, short trees as, 194
conditional entropy, 199
conditional probability, 212
naïve Bayes classifier, 215–216
confidence, 141–142
outcome, 172
parameters, 171
confidence interval, 107
`confint( )` function, 171
confusion matrix, 224, 280
contingency tables, 79
continuous variables, discretization, 211
corpora
Brown Corpus, 267–268
corpora in Natural Language Processing, 256
IC (information content), 268–269
sentiment analysis and, 278
correlated variables, 206
credit card companies, 2
CRISP-DM, 28
crowdsourcing, 17
CSV (comma-separated-value) files, 64–65
importing, 64–65
customer segmentation
k-means, 120
logistic regression, 180–181
CVS files, 6
cyclic components of time series analysis, 235

D

data
growth needs, 9–10
sources, 15–16
`data( )` function, 84
data aggregators, 17–18
data analysis, exploratory, 80–82
visualization and, 82–85
Data Analytics Lifecycle
Business Intelligence Analyst role, 27
Business User role, 27
Communicate Results phase, 30, 49–50
GINA case study, 58–59
Data Engineer role, 27–28
Data preparation phase, 29, 36–37
Alpine Miner, 42
data conditioning, 40–41
data visualization, 41–42
Data Wrangler, 42
dataset inventory, 39–40
ETLT, 38–39
GINA case study, 55–56
Hadoop, 42
OpenRefine, 42
sandbox preparation, 37–38
tools, 42
Data Scientist role, 28
DBA (Database Administrator) role, 27
Discovery phase, 29
business domain, 30–31
data source identification, 35–36
framing, 32–33
GINA case study, 54–55
hypothesis development, 35
resources, 31–32
sponsor interview, 33–34
stakeholder identification, 33
GINA case study, 53–60
Model Building phase, 30, 46–48
Alpine Miner, 48
GINA case study, 56–58
Mathematica, 48
Matlab, 48
Octave, 48
PL/R, 48
Python, 48
R, 48
SAS Enterprise Miner, 48
SPSS Modeler, 48
SQL, 48
STATISTICA, 48
WEKA, 48
Model Planning phase, 29–30, 42–44
data exploration, 44–45
GINA case study, 56
model selection, 45
R, 45–46

SAS/ACCESS, 46
SQL Analysis services, 46
variable selection, 44–45
Operationalize phase, 30, 50–53, 360
Business Intelligence Analyst and, 52
Business User and, 52
Data Engineer and, 52
Data Scientist and, 52
DBA (Database Administrator) and, 52
GINA case study, 59–60
Project Manager and, 52
Project Sponsor and, 52
processes, 28
Project Manager role, 27
Project Sponsor role, 27
roles, 26–28
data buyers, 18
data cleansing, 86
data collectors, 17
data conditioning, 40–41
data creation rate, 3
data devices, 17
Data Engineer, Operationalize phase, 52
Data Engineer role, 27–28
data formats, text analysis, 257
data frames, 75–76
data marts, 10
Data preparation phase of lifecycle, 29, 36–37
data conditioning, 40–41
data visualization, 41–42
dataset inventory, 39–40
ETLT, 38–39
sandbox preparation, 37–38
data repositories, 9–11
types, 10–11
Data Savvy Professionals, 20
Data Science *versus* BI, 12–13
Data Scientists, 28
activities, 20–21
business challenges, 20
characteristics, 21–22
Operationalize phase and, 52
recommendations and, 21
statistical models and, 20–21
data sources
Discovery phase, 35–36
text analysis, 257
data structures, 5–9
quasi-structured data, 6, 7
semi-structured data, 6
structured data, 6
unstructured data, 6
data types in R, 71–72
character, 72
logical, 72
numeric, 72
vectors, 73–74
data users, 18
data visualization, 41–42, 377–378
CSS and, 378
GGobi, 377–378
Gnuplot, 377–378
graphs, 380–386
clean up, 387–392
three-dimensional, 392–393
HTML and, 378
key points with support, 378–379
representation methods, 386–387
SVG and, 378
data warehouses, 11
Data Wrangler, 42
datasets
exporting, R and, 69–71
importing, R and, 69–71
inventory, 39–40
Davenport, Tom, 28
DBA (Database Administrator), 10, 27
Operational phase and, 52
decision trees, 192–193
algorithms, 197–200
C4.5, 203–204
CART, 204
categorical, 205
greedy, 204
ID3, 203
numerical, 205
binary decisions, 206
branches, 193
classification trees, 193
correlated variables, 206
evaluating, 204–206
greedy algorithms, 204
internal nodes, 193
irrelevant variables, 205
nodes
depth, 193
leaf, 193
R and, 206–211
redundant variables, 206
regions, 205
regression trees, 193
root, 193
short trees, 194
decision stump, 194

splits, 193, 197
detecting, 200–203
limiting, 194
structure, 205
uses, 194
Deep Analytical Talent, 19–20
DELTA framework, 28
demand forecasting, linear regression and, 162
density plots, exploratory data analysis, 88–91
dependent variables, 162
descriptive statistics, 79–80
deviance, 183–184
devices, 17
mobile, 16
nontraditional, 16
smart devices, 16
DF (document frequency), 271–272
diagnostic imaging, 16
diagnostics
association rules, 158
classifiers, 224–228
linear regression
linearity assumption, 173
N-fold cross-validation, 177–178
normality assumption, 174–177
residuals, 173–174
logistic regression
deviance, 183–184
histogram of probabilities, 188
log-likelihood test, 184–185
pseudo-R^2, 183
ROC curve, 185–187
naïve Bayes, 217–218
`diff( )` function, 245
difference in means, 104
confidence interval, 107
student's t-testing, 104–106
Welch's t-test, 106–108
differencing, 241–242
dirty data, 85–87
Discovery phase of lifecycle, 29
data source identification, 35–36
framing, 32–33
hypothesis development, 35
sponsor interview, 33–34
stakeholder identification, 33
discretization of continuous variables, 211
documents, categorization, 274–277
`dotchart( )` function, 88

E

Eclipse, 304
ecosystem of Big Data, 16–19
Data Savvy Professionals, 20
Deep Analytical Talent, 19–20
key roles, 19–22
Technology and Data Enablers, 20
EDWs (Enterprise Data Warehouses), 10
effect size, 110
EMC Google search example, 7–9
emoticons, 282
engineering, logistic regression and, 179
ensemble methods, decision trees, 194
error distribution
linear regression model, 165–166
residual standard error, 170
ETLT, 38–39
EXCEPT operator (SQL), 333–3334
exploratory data analysis, 80–82
density plot, 88–91
dirty data, 85–87
histograms, 88–91
multiple variables, 91–92
analysis over time, 99
barplots, 93–94
box-and-whisker plots, 95–96
dotcharts, 93–94
hexbinplots, 96–97
versus presentation, 99–101
scatterplot matrix, 97–99
visualization and, 82–85
single variable, 88–91
exporting datasets in R, 69–71
expressions, regular, 263

F

Facebook, 2, 3–4
factors, 77–78
financial information, logistic regression and, 179
FNR (false negative rate), 225
forecasting
ARIMA (Autoregressive Integrated Moving Average) model, 251–252
linear regression and, 162
FP (false positives), confusion matrix, 224
FPR (false positive rate), 225
framing in Discovery phase, 32–33
functions
`aov( )`, 78
`apriori( )`, 146, 152–157
`arima( )`, 246
`array( )`, 74
`barplot( )`, 88
`cbind( )`, 78
`class( )`, 72
`confint( )`, 171

data( ), 84
diff( ), 245
dotchart( ), 88
gl( ), 84
glm( ), 183
hclust( ), 135
head( ), 65
inspect( ), 147, 154–155
integer( ), 72
IQR( ), 80
is.data.frame( ), 75
is.na( ), 86
is.vector( ), 73
jpeg( ), 71
kmeans( ), 134
kmode( ), 134–135
length( ), 72
library( ), 70
lm( ), 66
load.image( ), 68–69
matrix.inverse( ), 74
mean( ), 86
my_range( ), 80
na.exclude( ), 86
pamk( ), 135
Pig, 307–308
plot( ), 65, 153–154, 245
predict( ), 172
rbind( ), 78
read.csv( ), 64–65, 75
read.csv2( ), 70
read.delim2( ), 70
rpart, 207
SQL, 347–351
sqlQuery( ), 70
str( ), 75
summary( ), 65, 66–67, 79, 80–82
t( ), 74
ts( ), 245
typeof( ), 72
wilcox.test( ), 109
window functions (SQL), 343–347
write.csv( ), 70
write.csv2( ), 70
write.table( ), 70

G

Generalized Linear Model function, 182
genetic sequencing, 3, 4
genomics, 4, 16
genotyping, 4
GGobi, 377–378
GINA (Global Innovation Network and Analysis), Data Analytics Lifecycle case study, 53–60
gl( ) function, 84
glm( ) function, 183
Gnuplot, 377–378
GPS systems, 16
Graph Search (Facebook), 3–4
graphs, 380–386
    clean up, 387–392
    three-dimensional, 392–393
greedy algorithms, 204
Green Eggs and Ham, text analysis and, 256
grocery store example of Apriori algorithm, 143
    Groceries dataset, 144–146
    itemsets, frequent generation, 146–151
    rules, generating, 152–157
growth needs of data, 9–10
GUIs (graphical user interfaces), R and, 67–69

H

Hadoop
    Data preparation phase, 42
    Hadoop Streaming API, 304–305
    HBase, 311–312
        architecture, 312–317
        column family names, 319
        column qualifier names, 319
        data model, 312–317
        Java API and, 319
        rows, 319
        use cases, 317–319
        versioning, 319
        Zookeeper, 319
    HDFS, 300–301
    Hive, 308–311
    LinkedIn, 297
    Mahout, 319–320
    MapReduce, 22
        combiners, 302–303
        development, 304–305
        drivers, 301
        execution, 304–305
        mappers, 301–302
        partitioners, 304
        structuring, 301–304
    natural language processing, 18
    Pig, 306–308
    pipes, 305
    Watson (IBM), 297
    Yahoo!, 297–298
    YARN (Yet Another Resource Negotiator), 305
hash-based itemsets, Apriori algorithm and, 158

HAWQ (HAdoop With Query), 321
HBase, 311–312
architecture, 312–317
column family names, 319
column qualifier names, 319
data model, 312–317
Java API and, 319
rows, 319
use cases, 317–319
versioning, 319
Zookeeper, 319
`hclust( )` function, 135
HDFS (Hadoop Distributed File System), 300–301
`head( )` function, 65
hexbinplots, 96–97
histograms
exploratory data analysis, 88–91
logistic regression, 188
Hive, 308–311
HiveQL (Hive Query Language), 308
Hopper, Grace, 299
Hubbard, Doug, 28
HVE (Hadoop Virtualization Extensions), 321
hypotheses
alternative hypothesis, 102–103
Discovery phase, 35
null hypothesis, 102
hypothesis testing, 102–104
two-sided hypothesis testing, 105
type I errors, 109–110
type II errors, 109–110

I

IBM Watson, 297
ID3 algorithm, 203
IDE (Interactive Development Environment), 304
IDF (inverted document frequency), 271–272
importing datasets in R, 69–71
in-database analytics
SQL, 328–338
text analysis, 338–339
independent variables, 162
input variables, 192
`inspect( )` function, 147, 154–155
`integer( )` function, 72
internal nodes (decision trees), 193
Internet of Things, 17–18
INTERSECT operator (SQL), 333
`IQR( )` function, 80
`is.data.frame( )` function, 75
`is.na( )` function, 86
`is.vector( )` function, 73
itemsets, 139
1-itemsets, 147
2-itemsets, 148–149
3-itemsets, 149–150
4-itemsets, 150–151
Apriori algorithm, 139
Apriori property, 139
downward closure property, 139
dynamic counting, Apriori algorithm and, 158
frequent itemset, 139
generation, frequent, 146–151
hash-based, Apriori algorithm and, 158
k-itemset, 139, 140–141

J

joins (SQL), 330–332
`jpeg( )` function, 71

K

k clusters
finding, 120–122
number of, 123–127
k-itemset, 139, 140–141
k-means, 118–119
customer segmentation, 120
image processing and, 119
k clusters
finding, 120–122
number of, 123–127
medical uses, 119
objects, attributes, 130–131
R and, 123–127
reasons to choose, 130–134
rescaling, 133–134
units of measure, 132–133
`kmeans( )` function, 134
`kmode( )` function, 134–135

L

lag, 237
Laplace smoothing, 217
lasso regression, 189
LDA (latent Dirichlet allocation), 274–275
leaf nodes, 192, 193
lemmatization, text analysis and, 258
`length( )` function, 72
leverage, 142
`library( )` function, 70

lifecycle. *See also* Data Analytics Lifecycle
lift, 142
linear regression, 162
    coefficients, 169
    diagnostics
        linearity assumption, 173
        N-fold cross-validation, 177–178
        normality assumption, 174–177
        residuals, 173–174
    model, 163–165
        categorical variables, 170–171
        normally distributed errors, 165–166
        outcome confidence intervals, 172
        parameter confidence intervals, 171
        prediction interval on outcome, 172
        R, 166–170
    p-values, 169–170
    use cases, 162–163
LinkedIn, 2, 22–23, 297
lists in R, 76–77
`lm( )` function, 66
`load.image( )` function, 68–69
logical data types, R, 72
logistic regression, 178
    cautions, 188–189
    diagnostics, 181–182
        deviance, 183–184
        histogram of probabilities, 188
        log-likelihood test, 184–185
        pseudo-R^2, 183
        ROC curve, 185–187
    Generalized Linear Model function, 182
    model, 179–181
    multinomial, 190
    reasons to choose, 188–189
    use cases, 179
log-likelihood test, 184–185
loyalty cards, 17

M

MAD (Magnetic/Agile/Deep) skills, 28, 352–356
MADlib, 352–356
Mahout, 319–320
MapReduce, 22, 298–299
    combiners, 302–303
    development, 304–305
    drivers, 301–302
    execution, 304–305
    mappers, 301–302
    partitioners, 304
    structuring, 301–304
market basket analysis, 139
    association rules, 143
marketing, logistic regression and, 179
master nodes, 301
matrices
    confusion matrix, 224
    R, 74–75
    scatterplot matrices, 97–99
`matrix.inverse( )` function, 74
MaxEnt (maximum entropy), 278
McKinsey & Co. definition of Big Data, 3
`mean( )` function, 86
medical information, 16
    k-means and, 119
    linear regression and, 162
    logistic regression and, 179
minimum confidence, 141
missing data, 86
mobile devices, 16
mobile phone companies, 2
Model Building phase of lifecycle, 30, 46–48
    Alpine Miner, 48
    Mathematica, 48
    Matlab, 48
    Octave, 48
    PL/R, 48
    Python, 48
    R, 48
    SAS Enterprise Miner, 48
    SPSS Modeler, 48
    SQL, 48
    STATISTICA, 48
    WEKA, 48
Model Planning phase of lifecycle, 29–30, 42–44
    data exploration, 44–45
    model selection, 45
    R, 45–46
    SAS/ACCESS, 46
    SQL Analysis services, 46
    variables, selecting, 44–45
morphological features in text analysis, 266–267
moving average models, 239–241
MPP (massively parallel processing), 5
MTurk (Mechanical Turk), 282
multinomial logistic regression, 190
multivariate time series analysis, 253
`my_range( )` function, 80

N

`na.exclude( )` function, 86
naïve Bayes, 211–212
    Bayes' theorem, 212–214
    diagnostics, 217–218

naïve Bayes classifier, 214–217
R and, 218–224
sentiment analysis and, 278
smoothing, 217
natural language processing, 18
N-fold cross-validation, 177–178
NLP (Natural Language Processing), 256
nodes
master, 301
worker, 301
nodes (decision trees), 192
depth, 193
leaf, 193
leaf nodes, 192, 193
nonparametric tests, 108–109
nontraditional devices, 16
normality
ARIMA model, 250–251
linear regression, 174–177
normalization, data conditioning, 40–41
NoSQL, 322–323
null deviance, 183
null hypothesis, 102
numeric data types, R, 72
numerical algorithms, 205
numerical underflow, 216–217

O

objects, k-means, attributes, 130–131
OLAP (online analytical processing), 6
cubes, 10
OpenRefine, 42
Operationalize phase of lifecycle, 30, 50–53, 360
Business Intelligence Analyst and, 52
Business User and, 52
Data Engineer and, 52
Data Scientist and, 52
DBA (Database Administrator) and, 52
Project Manager and, 52
Project Sponsor and, 52
operators, subsetting, 75
outcome
confidence intervals, 172
prediction interval, 172

P

PACF (partial autocorrelation function), 238–239
`pamk( )` function, 135
parameters, confidence intervals, 171
parametric tests, 108–109
parsing, text analysis and, 257
partitioning
Apriori algorithm and, 158
MapReduce, 304
photographs, 16
Pig, 306–308
Pivotal HD Enterprise, 320–321
`plot( )` function, 65, 153–154, 245
POS (part-of-speech) tagging, 258
power of a test, 110
precision in sentiment analysis, 281
`predict( )` function, 172
prediction trees. *See* decision trees
presentation *versus* data exploration, 99–101
probability, conditional, 212
naïve Bayes classifier, 215–216
Project Manager, Operationalize phase, 52
Project Manager role, 27
Project Sponsor, Operationalize phase, 52
Project Sponsor role, 27
pseudo-R^2, 183
p-values, linear regression, 169–170

Q

quasi-structured data, 6, 7
queries, SQL, 329–330
nested, 3334
subqueries, 3334

R

arrays, 74–75
attributes, types, 71–72
data frames, 75–76
data types, 71–72
character, 72
logical, 72
numeric, 72
vectors, 73–74
decision trees, 206–211
descriptive statistics, 79–80
exploratory data analysis, 80–82
density plot, 88–91
dirty data, 85–87
histograms, 88–91
multiple variables, 91–99
versus presentation, 99–101
visualization and, 82–85, 88–91
factors, 77–78
functions

aov( ), 78
array( ), 74
barplot( ), 88
cbind( ), 78
class( ), 72
data( ), 84
dotchart( ), 88
gl( ), 84
head( ), 65
import function defaults, 70
integer( ), 72
IQR( ), 80
is.data.frame( ), 75
is.na( ), 86
is.vector( ), 73
jpeg( ), 71
length( ), 72
library( ), 70
lm( ), 66
load.image( ), 68–69
my_range( ), 80
plot( ) function, 65
rbind( ), 78
read.csv( ), 65, 75
read.csv2( ), 70
read.delim( ), 69
read.delim2( ), 70
read.table( ), 69
str( ), 75
summary( ), 65, 66–67, 79
t( ), 74
typeof( ), 72
visualizing single variable, 88
write.csv( ), 70
write.csv2( ), 70
write.table( ), 70
GUIs, 67–69
import/export, 69–71
k-means analysis, 123–127
linear regression model, 166–170
lists, 76–77
matrices, 74–75
model planning and, 45–46
naïve Bayes and, 218–224
operators, subsetting, 75
overview, 64–67
statistical techniques, 101–102
ANOVA, 110–114
difference in means, 104–108
effect size, 110
hypothesis testing, 102–104
power of test, 110
sample size, 110
type I errors, 109–110
type II errors, 109–110
tables, contingency tables, 79
R commander GUI, 67
random components of time series analysis, 235
Rattle GUI, 67
raw text
collection, 260–263
tokenization, 264
rbind( ) function, 78
RDBMS, 6
read.csv( ) function, 64–65, 75
read.csv2( ) function, 70
read.delim( ) function, 69
read.delim2( ) function, 70
read.table( ) function, 69
real estate, linear regression and, 162
recall in sentiment analysis, 281
redundant variables, 206
regression
lasso, 189
linear, 162
coefficients, 169
diagnostics, 173–178
model, 163–172
p-values, 169–170
use cases, 162–163
logistic, 178
cautions, 188–189
diagnostics, 181–188
model, 179–181
multinomial logistic, 190
reasons to choose, 188–189
use cases, 179
multinomial logistic, 190
ridge, 189
variables
dependent, 162
independent, 162
regression trees, 193
regular expressions, 263, 339–340
relationships, 141
repositories, 9–11
types, 10–11
representation methods, 386–387
rescaling, k-means, 133–134
residual deviance, 183
residual standard error, 170

residuals, linear regression, 173–174
resources, Discovery phase of lifecycle, 31–32
RFID readers, 16
ridge regression, 189
ROC (receiver operating characteristic) curve, 185–187, 225
roots (decision trees), 193
`rpart` function, 207
RStudio GUI, 67–68
rules
  association rules, 138–139
    application, 143
    candidate rules, 141–142
    diagnostics, 158
    testing and, 157–158
    validation, 157–158
  generating, grocery store example (Apriori), 152–157

S

sales, time series analysis and, 234
sample size, 110
sampling, Apriori algorithm and, 158
sandboxes, 10, 11. *See also* work spaces
  Data preparation phase, 37–38
SAS/ACCESS, model planning, 46
scatterplot matrix, 97–99
scatterplots, 81
  Anscombe's quartet, 83
  multiple variables, 91–92
scientific method, 28
searches, text analysis and, 257
seasonal autoregressive integrated moving average model, 243–244
seasonality components of time series analysis, 235
seismic processing, 16
semi-structured data, 6
SensorNet, 17–18
sentiment analysis in text analysis, 277–283
  confusion matrix, 280
  precision, 281
  recall, 281
shopping
  loyalty cards, 17
  RFID chips in carts, 17
short trees, 194
smart devices, 16
smartphones, 17
smoothing, 217
social media, 3–4
sources of data, 15–16
spart parts planning, time series analysis and, 234–235
splits (decision trees), 193
  detecting, 200–203
sponsor interview, Discovery phase, 33
spreadmarts, 10
spreadsheets, 6, 9, 10
SQL (Structured Query Language), 328–329
  aggregates
    ordered, 351–352
    user-defined, 347–351
  EXCEPT operator, 333–3334
  functions, user-defined, 347–351
  grouping, 334–338
  INTERSECT operator, 333
  joins, 330–332
  MADlib, 352–356
  queries, 329–330
    nested, 3334
    subqueries, 3334
  set operations, 332–334
  UNION ALL operator, 332–333
  window functions, 343–347
SQL Analysis services, model planning and, 46
`sqlQuery( )` function, 70
stakeholders, Discovery phase of lifecycle, 33
stationary time series, 236
statistical techniques, 101–102
  ANOVA, 110–114
  difference in means, 104
    student's t-test, 104–106
    Welch's t-test, 106–108
  effect size, 110
  hypothesis testing, 102–104
  power of test, 110
  sample size, 110
  type I errors, 109–110
  type II errors, 109–110
  Wilcoxon rank-sum test, 108–109
statistics
  Anscombe's quartet, 82–83
  descriptive, 79–80
stemming, text analysis and, 258
stock trading, time series analysis and, 235
stop words, 270–271
`str( )` function, 75
structured data, 6
subsetting operators, 75
`summary( )` function, 65, 66–67, 79, 80–82
SVM (support vector machines), 278

T

`t( )` function, 74
tables, contengency tables, 79
Target stores, 22
t-distribution

ANOVA, 110–114
student's t-test, 104–106
Welch's t-test, 106–108
technical specifications in project, 376–377
Technology and Data Enablers, 20
testing, association rules and, 157–158
text analysis, 256
ACME example, 259–263
bag-of-words, 265–266
corpora, 264–265
Brown Corpus, 267–268
corpora in Natural Language Processing, 256
IC (information corpora), 268–269
data formats, 257
data sources, 257
document categorization, 274–277
Green Eggs and Ham, 256
in-database, 338–339
lemmatization, 258
morphological features, 266–267
NLP (Natural Language Processing), 256
parsing, 257
POS (part-of-speech) tagging, 258
raw text, collection, 260–263
search and retrieval, 257
sentiment analysis, 277–283
stemming, 258
stop words, 270–271
text mining, 257–258
TF (term frequency) of words, 265–266
DF, 271–272
IDF, 271–272
lemmatization, 271
stemming, 271
stop words, 270–271
TFIDF, 269–274
tokenization, 264
topic modeling, 267, 274
LDA (latent Dirichlet allocation), 274–275
web scraper, 262–263
word clouds, 284
Zipf's Law, 265–266
text mining, 257
textual data files, 6
TF (term frequency) of words, 265–266
DF (document frequency), 271–272
IDF (inverted document frequency), 271–272
lemmatization, 271
stemming, 271
stop words, 270–271
TFIDF, 269–274
TFIDF (Term Frequency-Inverse Document Frequency), 269–274, 285–286
time series analysis
ARIMA model, 236
ACF, 236–237
ARMA model, 241–244
autoregressive models, 238–239
building, 244–252
cautions, 252–253
constant variance, 250–251
evaluating, 244–252
fitted models, 249–250
forecasting, 251–252
moving average models, 239–241
normality, 250–251
PACF, 238–239
reasons to choose, 252–253
seasonal autogregressive integrated moving average model, 243–244
ARMAX (Autoregressive Moving Average with Exogenous inputs), 253
Box-Jenkins methodology, 235–236
cyclic components, 235
differencing, 241–242
fitted models, 249–250
GARCH (Generalized Autoregressive Conditionally Heteroscedastic), 253
Kalman filtering, 253
multivariate time series analysis, 253
random components, 235
seasonal autoregressive integrated moving average model, 243–244
seasonality, 235
spectral analysis, 253
stationary time series, 236
trends, 235
use cases, 234–235
white noise process, 239
tokenization in text analysis, 264
topic modeling in text analysis, 267, 274
LDA (latent Dirichlet allocation), 274–275
TP (true positives), confusion matrix, 224
TPR (true positive rate), 225
transaction data, 6
transaction reduction, Apriori algorithm and, 158
trends, time series analysis, 235
TRP (True Positive Rate), 185–187
`ts( )` function, 245
two-sided hypothesis test, 105
type I errors, 109–110
type II errors, 109–110
`typeof( )` function, 72

U

UNION ALL operator (SQL), 332–333
units of measure, k-means, 132–133
unstructured data, 6

Apache Hadoop, HDFS, 300–301
LinkedIn, 297
MapReduce, 298–299
natural language processing, 18
use cases, 296–298
Watson (IBM), 297
Yahoo!, 297–298
unsupervised techniques. *See* clustering
users of data, 18

V

validation, association rules and, 157–158
variables
categorical, 170–171
continuous, discretization, 211
correlated, 206
decision trees, 205
dependent, 162
factors, 77–78
independent, 162
input, 192
redundant, 206
VARIMA (Vector ARIMA), 253
vectors, R, 73–74
video footage, 16
k-means and, 119
video surveillance, 16
visualization, 41–42. *See also* data visualization
exploratory data analysis, 82–85
single variable, 88–91
grocery store example (Apriori), 152–157
volume, variety, velocity. *See* 3 Vs (volume, variety, velocity)

W

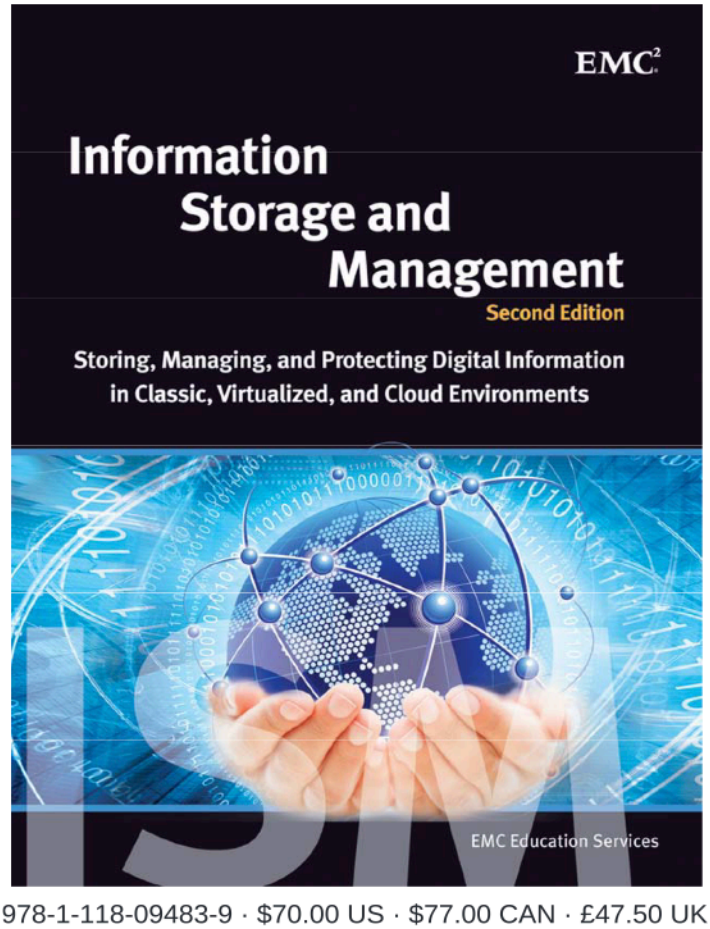
Watson (IBM), 297
web scraper, 262–263
white noise process, 239
Wilcoxan rank-sum test, 108–109
`wilcox.test( )` function, 109
window functions (SQL), 343–347
word clouds, 284
work spaces, 10, 11. *See also* sandboxes
Data preparation phase, 37–38
worker nodes, 301
`write.csv( )` function, 70
`write.csv2( )` function, 70
`write.table( )` function, 70
WSS (Within Sum of Squares), 123–127

X-Z

XML (eXtensible Markup Language), 6
Yahoo!, 297–298
YARN (Yet Another Resource Negotiator), 305
Zipf's Law, 265–266

A comprehensive book on information storage and management by the EMC's Education Services, a global technology leader.

More than ever, the IT industry is challenged with employing and developing highly skilled technical professionals with storage technology expertise across classic, virtualized, and cloud environments. This book covers concepts, principles, and deployment considerations across technologies that are used for storing and managing information.



Key Technology Strategies for Classic, Virtualized, and Cloud Environments:

- Challenges and Solutions for Data Storage and Management
- Intelligent Storage, Object-Based Storage, and Unified Storage
- Storage Networking, Federation, and Protocols
- Backup, Recovery, Deduplication, and Archive
- Business Continuity and Disaster Recovery
- Cloud Computing and Converged Infrastructure
- Storage Security and Managing Storage Infrastructure

978-1-118-09483-9 · $70.00 US · $77.00 CAN · £47.50 UK

WILEY

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.