

PROBABILITY WITH R

AN INTRODUCTION WITH
COMPUTER SCIENCE APPLICATIONS

SECOND EDITION

JANE M. HORGAN



WILEY

PROBABILITY WITH *R*

PROBABILITY with *R*

AN INTRODUCTION with COMPUTER SCIENCE APPLICATIONS

Second Edition

JANE M. HORGAN

WILEY

This second edition first published 2020
© 2020 John Wiley & Sons, Inc. All rights reserved.

Edition History

John Wiley & Sons, Inc. (1e, 2009)

All rights reserved. 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 or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at <http://www.wiley.com/go/permissions>.

The right of Jane M. Horgan to be identified as the author of this work has been asserted in accordance with law.

Registered Office

John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA

Editorial Office

111 River Street, Hoboken, NJ 07030, USA

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Limit of Liability/Disclaimer of Warranty

While the publisher and authors have used their best efforts in preparing this work, they 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 any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication Data

Names: Horgan, Jane M., author.

Title: Probability with R : an introduction with computer science applications / Jane Mary Horgan.

Description: Second edition. | Hoboken, NJ, USA : Wiley, 2020. | Includes bibliographical references and index.

Identifiers: LCCN 2019032520 (print) | LCCN 2019032521 (ebook) | ISBN 9781119536949 (hardback) | ISBN 9781119536925 (adobe pdf) | ISBN 9781119536987 (epub)

Subjects: LCSH: Computer science--Mathematics. | Probabilities. | R (Computer program language)

Classification: LCC QA76.9.M35 H863 2020 (print) | LCC QA76.9.M35 (ebook) | DDC 004.01/5113--dc23

LC record available at <https://lccn.loc.gov/2019032520>

LC ebook record available at <https://lccn.loc.gov/2019032521>

Cover design by Wiley

Set in 10/12pt TimesLTStd by SPi Global, Chennai, India

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To the memory of Willie, referee, and father of all the Horgans

CONTENTS

PREFACE TO THE SECOND EDITION	xiii
PREFACE TO THE FIRST EDITION	xvii
ACKNOWLEDGMENTS	xxi
ABOUT THE COMPANION WEBSITE	xxiii
I THE <i>R</i> LANGUAGE	1
1 Basics of <i>R</i>	3
1.1 What Is <i>R</i> ?	3
1.2 Installing <i>R</i>	4
1.3 <i>R</i> Documentation	4
1.4 Basics	5
1.5 Getting Help	6
1.6 Data Entry	7
1.7 Missing Values	11
1.8 Editing	12
1.9 Tidying Up	12
1.10 Saving and Retrieving	13
1.11 Packages	13
1.12 Interfaces	14
1.13 Project	16

2 Summarizing Statistical Data	17
2.1 Measures of Central Tendency	17
2.2 Measures of Dispersion	21
2.3 Overall Summary Statistics	24
2.4 Programming in <i>R</i>	25
2.5 Project	30
3 Graphical Displays	31
3.1 Boxplots	31
3.2 Histograms	36
3.3 Stem and Leaf	40
3.4 Scatter Plots	40
3.5 The Line of Best Fit	43
3.6 Machine Learning and the Line of Best Fit	44
3.7 Graphical Displays Versus Summary Statistics	49
3.8 Projects	53
II FUNDAMENTALS OF PROBABILITY	55
4 Probability Basics	57
4.1 Experiments, Sample Spaces, and Events	58
4.2 Classical Approach to Probability	61
4.3 Permutations and Combinations	64
4.4 The Birthday Problem	71
4.5 Balls and Bins	76
4.6 <i>R</i> Functions for Allocation	79
4.7 Allocation Overload	81
4.8 Relative Frequency Approach to Probability	83
4.9 Simulating Probabilities	84
4.10 Projects	89
5 Rules of Probability	91
5.1 Probability and Sets	91
5.2 Mutually Exclusive Events	92
5.3 Complementary Events	93
5.4 Axioms of Probability	94
5.5 Properties of Probability	96
6 Conditional Probability	104
6.1 Multiplication Law of Probability	107
6.2 Independent Events	108

6.3	Independence of More than Two Events	110
6.4	The Intel Fiasco	113
6.5	Law of Total Probability	115
6.6	Trees	118
6.7	Project	123
7	Posterior Probability and Bayes	124
7.1	Bayes' Rule	124
7.2	Hardware Fault Diagnosis	131
7.3	Machine Learning and Classification	132
7.4	Spam Filtering	135
7.5	Machine Translation	137
8	Reliability	142
8.1	Series Systems	142
8.2	Parallel Systems	143
8.3	Reliability of a System	143
8.4	Series–Parallel Systems	150
8.5	The Design of Systems	153
8.6	The General System	158
III	DISCRETE DISTRIBUTIONS	161
9	Introduction to Discrete Distributions	163
9.1	Discrete Random Variables	163
9.2	Cumulative Distribution Function	168
9.3	Some Simple Discrete Distributions	170
9.4	Benford's Law	174
9.5	Summarizing Random Variables: Expectation	175
9.6	Properties of Expectations	180
9.7	Simulating Discrete Random Variables and Expectations	183
9.8	Bivariate Distributions	187
9.9	Marginal Distributions	189
9.10	Conditional Distributions	190
9.11	Project	194
10	The Geometric Distribution	196
10.1	Geometric Random Variables	198
10.2	Cumulative Distribution Function	203
10.3	The Quantile Function	207
10.4	Geometric Expectations	209
10.5	Simulating Geometric Probabilities and Expectations	210

10.6	Amnesia	217
10.7	Simulating Markov	219
10.8	Projects	224
11	The Binomial Distribution	226
11.1	Binomial Probabilities	227
11.2	Binomial Random Variables	229
11.3	Cumulative Distribution Function	233
11.4	The Quantile Function	235
11.5	Reliability: The General System	238
11.6	Machine Learning	241
11.7	Binomial Expectations	245
11.8	Simulating Binomial Probabilities and Expectations	248
11.9	Projects	254
12	The Hypergeometric Distribution	255
12.1	Hypergeometric Random Variables	257
12.2	Cumulative Distribution Function	260
12.3	The Lottery	262
12.4	Hypergeometric or Binomial?	266
12.5	Projects	273
13	The Poisson Distribution	274
13.1	Death by Horse Kick	274
13.2	Limiting Binomial Distribution	275
13.3	Random Events in Time and Space	281
13.4	Probability Density Function	283
13.5	Cumulative Distribution Function	287
13.6	The Quantile Function	289
13.7	Estimating Software Reliability	290
13.8	Modeling Defects in Integrated Circuits	292
13.9	Simulating Poisson Probabilities	293
13.10	Projects	298
14	Sampling Inspection Schemes	299
14.1	Introduction	299
14.2	Single Sampling Inspection Schemes	300
14.3	Acceptance Probabilities	301
14.4	Simulating Sampling Inspection Schemes	303
14.5	Operating Characteristic Curve	308
14.6	Producer's and Consumer's Risks	310
14.7	Design of Sampling Schemes	311

14.8 Rectifying Sampling Inspection Schemes	315
14.9 Average Outgoing Quality	316
14.10 Double Sampling Inspection Schemes	318
14.11 Average Sample Size	319
14.12 Single Versus Double Schemes	320
14.13 Projects	324
IV CONTINUOUS DISTRIBUTIONS	325
15 Introduction to Continuous Distributions	327
15.1 Introduction to Continuous Random Variables	328
15.2 Probability Density Function	328
15.3 Cumulative Distribution Function	331
15.4 The Uniform Distribution	332
15.5 Expectation of a Continuous Random Variable	336
15.6 Simulating Continuous Variables	338
16 The Exponential Distribution	341
16.1 Modeling Waiting Times	341
16.2 Probability Density Function of Waiting Times	342
16.3 Cumulative Distribution Function	344
16.4 Modeling Lifetimes	347
16.5 Quantiles	349
16.6 Exponential Expectations	351
16.7 Simulating Exponential Probabilities and Expectations	353
16.8 Amnesia	356
16.9 Simulating Markov	360
16.10 Project	369
17 Queues	370
17.1 The Single Server Queue	370
17.2 Traffic Intensity	371
17.3 Queue Length	372
17.4 Average Response Time	376
17.5 Extensions of the M/M/1 Queue	378
17.6 Project	382
18 The Normal Distribution	383
18.1 The Normal Probability Density Function	385
18.2 The Cumulative Distribution Function	387

18.3	Quantiles	389
18.4	The Standard Normal Distribution	391
18.5	Achieving Normality: Limiting Distributions	394
18.6	Projects	405
19	Process Control	407
19.1	Control Charts	407
19.2	Cusum Charts	411
19.3	Charts for Defective Rates	412
19.4	Project	416
V	TAILING OFF	417
20	The Inequalities of Markov and Chebyshev	419
20.1	Markov's Inequality	420
20.2	Algorithm Runtime	426
20.3	Chebyshev's Inequality	427
Appendix A:	Data: Examination Results	433
Appendix B:	The Line of Best Fit: Coefficient Derivations	437
Appendix C:	Variance Derivations	440
Appendix D:	Binomial Approximation to the Hypergeometric	446
Appendix E:	Normal Tables	448
Appendix F:	The Inequalities of Markov and Chebyshev	450
INDEX TO R COMMANDS		453
INDEX		457
POSTFACE		

PREFACE TO THE SECOND EDITION

It is now over 10 years since the publication of the first edition of “*Probability with R*.” Back then we had just begun to hear of smartphones, fitbits, apps, and Bluetooth; machine learning was in its infancy. It is timely to address how probability applies to new developments in computing. The applications and examples of the first edition are beginning to look somewhat passé and old fashioned. Here, therefore, we offer an updated and extended version of that first edition.

This second edition is still intended to be a first course in probability, addressed to students of computing and related disciplines. As in the first edition, we favor experimentation and simulation rather than the traditional mathematical approach. We continue to rely on the freely downloadable language *R*, which has of course evolved over the past 10 years.

Our *R* programs are integrated throughout the text, to illustrate the concepts of probability, to simulate distributions, and to explore new problems. We have been mindful to avoid as far as is possible mathematical details, instead encouraging students to investigate for themselves, through experimentation and simulation in *R*. Algebraic derivations, when deemed necessary, are developed in the appendices.

In this second edition, all chapters have been revised and updated. Examples and applications of probability in new areas of computing, as well as exercises and projects, have been added to most chapters. The *R* code has been improved and expanded, by using procedures and functions that have become available in recent years. Extended use of loops and curve facilities to generate graphs with differing parameters have tidied up our approach to limiting distributions.

Briefly the changes in this second edition are

1. Part I, “The *R* Language” now contains:
 - new and improved *R* procedures, and an introduction to packages and interfaces (Chapter 1);
 - examples on apps to illustrate outliers, to calculate statistics in a data frame and statistics appropriate to skewed data (Chapter 2);
 - an introduction to linear regression, with a discussion of its importance as a tool in machine learning. We show how to obtain the line of best fit with the training set, and how to use the testing set to examine the suitability of the model. We also include extra graphing facilities (Chapter 3).
2. In Part II, “Fundamentals of Probability”:
 - Chapter 4 has been extended with extra examples on password recognition and new *R* functions to address hash table collision, server overload and the general allocation problem;
 - The concept of “independence” has now been extended from pairs to multiply variables (Chapter 6);
 - Chapter 7 contains new material on machine learning, notably the use of Bayes’ theorem to develop spam filters.
3. Part III “Discrete Distributions” now includes:
 - an introduction to bivariate discrete distributions, and programming techniques to handle large conditional matrices (Chapter 9);
 - an algorithm to simulate the Markov property of the geometric distribution (Chapter 10);
 - an extension of the reliability model of Chapter 8 to the general reliability model (Chapter 11);
 - an update of the lottery rules (Chapter 12);
 - an extended range of Poisson applications such as network failures, website hits, and virus attacks (Chapter 13).
4. In Part IV “Continuous Distributions”:
 - Chapters 16 and 17 have been reorganized. Chapter 17 now concentrates entirely on queues while Chapter 16 is extended to deal with the applications of the exponential distribution to lifetime models.
5. Part V “Tailing Off”
 - has extra exercises on recent applications of computing.
6. We have added three new appendices: Appendix A gives the data set used in Part I, Appendix B derives the coefficients of the line of best fit and Appendix F contains new proofs of the Markov and Chebyshev inequalities. The original appendices A, B, and C have been relabeled C, D, and E.

7. A separate index containing *R* commands and functions has been added.

All errors in the first edition have hopefully been corrected. I apologize in advance for any new errors that may escape my notice in this edition; should they arise, they will be corrected in the companion website.

Jane M. Horgan

*Dublin City University
Ireland
2019*

PREFACE TO THE FIRST EDITION

This book is offered as a first introduction to probability, and its application to computer disciplines. It has grown from a one-semester course delivered over the past several years to students reading for a degree in computing at Dublin City University. Students of computing seem to be able happily to think about Database, Computer Architecture, Language Design, Software Engineering, Operating Systems, and then to freeze up when it comes to “Probability,” and to wonder what it might have to do with computing. Convincing undergraduates of the relevance of probability to computing is one of the objectives of this book.

One reason for writing this has been my inability to find a good text in which probability is applied to problems in computing at the appropriate level. Most existing texts on probability seem to be overly rigorous, too mathematical for the typical computing student. While some computer students may be adept at mathematics, there are many who resist the subject. In this book, we have largely replaced the mathematical approach to probability by one of simulation and experimentation, taking advantage of the powerful graphical and simulation facilities of the statistical system *R*, which is freely available, and downloadable, from the web. The text is designed for students who have taken a first course in mathematics, involving just a little calculus, as is usual in most degree courses in computing. Mathematical derivations in the main text are kept to a minimum: when we think it necessary, algebraic details are provided in the appendices. To emphasize our attitude to the simulation and experimentation approach, we have chosen to incorporate instructions in *R* throughout the text, rather than put them back to an appendix.

Features of the book which distinguish it from other texts in probability include

- R is used not only as a tool for calculation and data analysis, but also to illustrate the concepts of probability, to simulate distributions, and to explore by experimentation different scenarios in decision-making. The R books currently available skim over the concepts of probability, and concentrate on using it for statistical inference and modelling.
- Recognizing that the student better understands definitions, generalizations and abstractions after seeing the applications, almost all new ideas are introduced and illustrated by real, computer-related, examples, covering a wide range of computer science applications.

Although we have addressed in the first instance computer scientists, we believe that this book should also be suitable for students of engineering and mathematics.

There are in all five parts to the book, starting in Part I with an introduction to R . This presents the procedures of R needed to summarize and provide graphical displays of statistical data. An introduction to programming in R is also included. Not meant to be a manual, this part is intended only to get the student started. As we progress, more procedures of R are introduced as the need arises.

Part II sets the foundations of probability, and introduces the functions available in R for examining them. R is used not only for calculating probabilities involving unwieldy computations but also for obtaining probabilities through simulation. Probability events and sample spaces are illustrated with the usual gambling experiments, as well as inspection of integrated-circuit chips, and observation of randomness in computer programming. A discussion of the “Intel Chip Fiasco” leads on to the “balls and bins” problem, which in turn is applied to assigning jobs to processors. It is shown how Bayes’ Theorem has important applications in modern-day computer science such as machine learning and machine translation. Methods to assess reliability of a computer containing many systems, which in turn contain many components, are considered.

Part III deals with discrete random variables. Nearly every chapter opens with a sequence of examples, designed to motivate the detail that follows. Techniques are developed for examining discrete variables by simulation in R . The objective is to empower students to be able to approximate parameters without having sufficient mathematical knowledge to derive them exactly. The Bernoulli, geometric, binomial, hypergeometric and Poisson distributions are each dealt with in a similar fashion, beginning with a set of examples with different parameters and using the graphical facilities in R to examine their distributions. Limiting distributions are exhibited through simulation, and the students use R to obtain rules of thumb to establish when these approximations are valid. R is also used to design single- and double-sampling inspection schemes.

Part IV deals with continuous random variables. The exponential distribution is introduced as the waiting time between Poisson occurrences, and the graphical facilities of R illustrate the models. The Markov memoryless property is simulated using R . Some applications of the exponential distribution are investigated, notably in the

areas of reliability and queues. *R* is used to model response times with varying traffic intensities. We have examined models for server queue lengths without using any of the formulae typical in a traditional approach. The normal distribution and some of its applications are discussed. It is shown how *R* can be used, both to illustrate limiting distributions and as a set of statistical tables.

Part V addresses the problem of obtaining probability bounds on the runtime of new algorithms when the distribution is unknown. Here Markov and Chebyshev inequalities provide estimates of probability when the information about the random variable is limited.

The exercises and projects at the end of each chapter are an integral part of the exposition. Many of them require the use of *R*, in some cases to perform routine calculations and in others to conduct experiments on simulated data. The projects are designed to improve the students' understanding of how probability interacts with computing.

This is a self-contained book with no need for ancillary material, other than, of course, the programming language *R*. There is a freely downloadable manual (*Venables, W.N., Smith, D.M., and the R Development Core Team (2004). An Introduction to R: A Programming Environment for Data Analysis and Graphics, Version 2.6.2*).

Students should be encouraged to use this in conjunction with the text. One of the big attractions of *R* is that it is open source. Most other systems, such as Matlab and Mathematica, require a license; apart from the expense, this makes access more difficult, and the student more likely not to use them.

Jane M. Horgan

*Dublin City University
2008*

ACKNOWLEDGMENTS

The generous contributions of James Power from Maynooth University and Charlie Daly from Dublin City University have much improved this second edition. I am deeply appreciative of their advice and help, and their provision of many relevant examples in areas of computing that have evolved since the first edition was published. They also read much of the new material and supplied valuable feedback. As I write this, we are in a state of shock at the sudden and untimely death of our close friend and colleague James Power. Maynooth University, Computer Science in Ireland and the Academic World generally are greatly diminished by his departure. It is difficult to accept his absence. We had so much more to learn from him.

I owe a huge debt of gratitude to Helen Fallon, Deputy Librarian at Maynooth University, who provided library facilities and ongoing assistance with the tracking down of material, relevant articles, and books.

A special word of thanks to my technical advisors: Systems analyst Rosaleen McGrath provided invaluable advice and help with file management. Jim Doyle and Eugene Curran continued to ensure that I had the best hardware and best software. Mathematician Robin Harte checked the mathematical derivations in the appendices and helped with the final edit.

As teachers, we should always remember that not only do we teach but also we are taught. Throughout my working life, I have been privileged to work with students, both in Ireland and abroad, who never ceased to inspire me. I am indebted to all who took this course over the years, and provided invaluable comments and constructive suggestions. I also appreciate the feedback from instructors who used the original version of the book, as well as those who reviewed it. Thank you to those who sent me words of appreciation, alerted me to errata and made useful suggestions. I hope you will continue to do so. My email address is

janemhorgan@gmail.com

I continue to be indebted to the developers and the core team of *R* for maintaining and extending the *R* system which is so important for teaching and research. As always, it has been a pleasure to work with the production team of John Wiley & Sons.

Thanks to my colleague and neighbor Margaret Byrne for her ongoing catchphrase “will that book of yours ever get finished?”; to all my friends in Kuşadası for putting up with me; apologies for those cancelled Turkish classes, missed appointments, lunch dates, and nights out.

Finally, let me thank my gambling trio. Jane, the former princess, is now a queen of hearts, Amy, once a pirate, is now the club ace, and the precious diamond Ava, not around in time for the first edition, is the youngest member of the club, and a force to be reckoned with.

Jane M. Horgan

2019

ABOUT THE COMPANION WEBSITE

This book is accompanied by a companion website:



www.wiley.com/go/Horgan/probabilitywithr2e

The website includes the following materials:

- the data file in electronic form;
- solutions to selected exercises for students;
- full set of solutions for instructors.

PART I

THE *R* LANGUAGE

1

BASICS OF *R*

This chapter introduces *R* and describes some of its basic operations and editing procedures. The syntax used to read and edit statistical data and to perform some basic statistical calculations is given.

It is not the intention to provide a complete set of features of the language, but rather to give a flavor of the structure of *R* and how to get help to proceed further. Additional features are introduced in later chapters as the need for them arises.

1.1 WHAT IS *R*?

R is a data-analysis system, a sophisticated calculator and an object-oriented programming language. It provides an environment for statistical analysis and graphics that differs from standard statistical packages such as SPSS and Minitab; these provide point-and-click graphical-user interfaces (GUIs), while *R* is command-driven. Users type commands at a prompt, and the *R* interpreter responds.

It is possible to get quite far using *R* to execute simple expressions from the command line, and some users may never need to go beyond this. At a more advanced level, users write their own functions, either to systematize repetitive work or to develop add-on packages for new functionality.

1.2 INSTALLING R

R is obtained from the website called CRAN (Comprehensive *R* Archive Network) and is downloaded by proceeding as follows:

- Go to the CRAN website at <http://cran.r-project.org/>;
- Choose an operating system from Linux, (Mac) OS X, and Windows appropriate to your computer. In this book, we work in the Windows environment, click “Download R for Windows”;
- Choose the “base” package;
- Click “Download R 3.6.1”, which is the current version at the time of writing.
- Press the option “Run.”

R is now installed, and you should see an “R” icon on your computer. Clicking on this will start up the standard *R* package.

1.3 R DOCUMENTATION

R documentation is available at <http://cran.r-project.org/manuals>, where you will find the following manuals:

- *An Introduction to R* gives an introduction to the language and how to use *R* for doing statistical analysis and graphics;
- *The R Language Definition* documents the language per se, that is, the objects that it works on, and the details of the expression evaluation process, which are useful to know when programming *R* functions;
- *Writing R Extensions* covers how to create your own packages, how to write *R* help files, and the foreign language (C, C++, Fortran, etc.) interfaces;
- *R Data Import/Export* describes the import and export facilities available either in *R* itself or via packages that are available from CRAN;
- *R Installation and Administration*;
- *R Internals* is a guide to the internal structures of *R* and coding standards for the core team working on *R* itself;
- *The R Reference Index* contains all help files of the *R* standard and recommended packages in printable form.

The documentation may be downloaded or browsed. We suggest that you download and obtain a hard copy of *An Introduction to R* by Venables et al. (2019) Version 3.6.1 (2019-07-05) and access the others as you require.

1.4 BASICS

To familiarize yourself with *R*, you should work through the commands given below at a workstation to which *R* has been downloaded.

To start, either click on the *R* icon (if you have created a short cut on your screen) or go to “Programs,” select *R*, and then click on the *R* icon. When the *R* program is started, and after it prints an introductory message on the screen, the interpreter prompts for input with the command prompt “>.”

Expressions that are typed at the command prompt are evaluated and printed. For example,

```
6+7*3/2
```

returns

```
[1] 16.5
```

To assign or store data, write

```
x <- 1:4
```

Here, the integers 1, 2, 3, 4 are assigned to the vector *x*. To check the contents of *x*, type

```
x
```

which returns

```
[1] 1 2 3 4
```

To square the elements of *x*, write

```
x2 <- x**2
```

or equivalently

```
x2 <- x^2
```

that causes each element in the vector *x* to be squared and stored in the vector *x2*. To examine the contents of *x2*, write

```
x2
```

which gives

```
[1] 1 4 9 16
```

To illustrate how *R* is case sensitive, consider

```
X <- 10
prod1 <- X*x
prod1
[1] 10 20 30 40
```

Here, the integer 10 is stored in *X*. *X * x* causes each element of the vector *x* to be multiplied by 10.

Some points to note:

- `<-` is the assignment operator; in the illustration “*x <- 1:4*, the integers 1, 2, 3, 4 are assigned to the vector *x*;
- *R* is case sensitive; for example, *x* and *X* represent different variables;
- Variable names can consist of any combination of lower and upper case letters, numerals, periods, and underscores, but cannot begin with a numeral or underscore;
- All the above examples of variables are numeric, but we shall see that *R* supports many other types of data.

The entities that *R* creates and manipulates are called *objects*. These include variables, arrays of numbers, strings, or functions.

All objects created in *R* are stored in what is known as the *workspace*.

1.5 GETTING HELP

The easiest way of getting help when you are working in the *R* environment is to click the *Help* button on the toolbar.

Alternatively, you can type

```
help()
```

for online help, or

```
help.start()
```

for an HTML browser interface.

It could be helpful to look at some demonstrations of *R* by typing

```
demo()
```

which gives a list of all available demonstrations.

Demonstrations on specific topics can be obtained by inserting an argument. For example,

```
demo(plotmath)
```

gives some examples of the use of mathematical notation.

A more specific way of getting help, when working in the *R* environment, is to type the name of the function you require. For example,

```
help(read.table)
```

will provide details on the exact syntactic structure of the instruction `read.table`.

An alternative is

```
?read.table
```

To obtain all that is available on a particular topic, use `apropos`.

```
apropos ("boxplot")
```

returns

```
"boxplot", "boxplot.default", "boxplot.stats"
```

which are all of the objects that contain the word “boxplot.”

1.6 DATA ENTRY

Before carrying out a statistical analysis, it is necessary to get the data into the computer. How you do this varies depending on the amount of data involved.

1.6.1 Reading and Displaying Data on Screen

A small data set, for example, a small set of repeated measurements on a single variable, may be entered directly from the screen. It is usually stored as a vector, which is essentially a list of numbers.

Example 1.1 Entering data from the screen to a vector

The total downtime occurring in the last month of 23 workstations in a computer laboratory was observed (in minutes) as follows:

```
0, 1, 2, 12, 12, 14, 18, 21, 21, 23, 24, 25, 28, 29, 30, 30, 30, 33, 36, 44, 45, 47, 51
```

To input these data from the screen environment of *R*, write

```
downtime <- c(0, 1, 2, 12, 12, 14, 18, 21, 21, 23, 24, 25,
28, 29, 30, 30, 30, 33, 36, 44, 45, 47, 51)
```

The construct $c(\dots)$ is used to define a vector containing the 23 data points. These data are then assigned to a vector called *downtime*.

To view the contents of the vector, type

```
downtime
```

which will display all the values in the vector *downtime*. △

R handles a vector as a single object. Calculations can be done with vectors like ordinary numbers provided they are the same length.

1.6.2 Reading Data from a File to a Data Frame

When the data set is large, it is better to set up a text file to store the data than to enter them directly from the screen.

A large data set is usually stored as a matrix, which consists of columns and rows. The columns denote the variables, while the rows are the observations on the variables. In *R*, this type of data set is stored in what is referred to as a *data frame*.

Definition 1.1 *Data frame*

A data frame is an object with rows and columns or equivalently it is a list of vectors of the same length. Each vector consists of repeated observations of some variable. The variables may be numbers, strings or factors. □

Example 1.2 *Reading data from a file into a data frame*

The examination results for a class of 119 students pursuing a computing degree are given on our companion website (www.wiley.com/go/Horgan/probabilitywithr2e) as a text file called *results.txt*. The complete data set is also given in Appendix A.

gender	arch1	prog1	arch2	prog2
m	99	98	83	94
m	NA	NA	86	77
m	97	97	92	93
m	99	97	95	96
m	89	92	86	94
m	91	97	91	97
m	100	88	96	85
f	86	82	89	87
m	89	88	65	84
m	85	90	83	85
m	50	91	84	93

m	96	71	56	83
f	98	80	81	94
m	96	76	59	84
			

The first row of the file contains the headings, *gender* and *arch1*, *prog1*, *arch2*, *prog2*, which are abbreviations for Architecture and Programming from Semester 1 and Semester 2, respectively. The remaining rows are the marks (%) obtained for each student. NA denotes that the marks are not available in this particular case.

The construct for reading this type of data into a data frame is `read.table`.

```
results <- read.table ("F:/data/results.txt", header = T)
```

assuming that your data file *results.txt* is stored in the *data* folder on the F drive. This command causes the data to be assigned to a data frame called *results*. Here `header = T` or equivalently `header = TRUE` specifies that the first line is a header, in this case containing the names of the variables. Notice that the forward slash (/) is used in the filename, not the backslash (\) which would be expected in the windows environment. The backslash has itself a meaning within R, and cannot be used in this context: / or \\ are used instead. Thus, we could have written

```
results <- read.table ("F:\\data\\results.txt", header = TRUE)
```

with the same effect. ◀

The contents of the file *results* may be listed on screen by typing

```
results
```

which gives

	gender	arch1	prog1	arch2	prog2
1	m	99	98	83	94
2	m	NA	NA	86	77
3	m	97	97	92	93
4	m	99	97	95	96
5	m	89	92	86	94
6	m	91	97	91	97
7	m	100	88	96	85
8	f	86	82	89	87
9	m	89	88	65	84
10	m	85	90	83	85
11	m	50	91	84	93
12	m	96	71	56	83

```
13      f    98    80    81    94
14      m    96    76    59    84
....
```

Notice that the *gender* variable is a factor with two levels “f” and “m,” while the remaining four variables are numeric. The figures in the first column on the left are the row numbers, and allows us to access individual elements in the *data frame*.

While we could list the entire data frame on the screen, this is inconvenient for all but the smallest data sets. R provides facilities for listing the first few rows and the last few rows.

```
head(results, n = 4)
```

gives the first four rows of the data set.

	gender	arch1	prog1	arch2	prog2
1	m	99	98	83	94
2	m	NA	NA	86	77
3	m	97	97	92	93
4	m	99	97	95	96

and

```
tail(results, n = 4)
```

gives the last four lines of the data set.

	gender	arch1	prog1	arch2	prog2
116	m	16	27	25	7
117	m	73	51	48	23
118	m	56	54	49	25
119	m	46	64	13	19

The convention for accessing the column variables is to use the name of the *data frame* followed by the name of the relevant column. For example,

```
results$arch1[5]
```

returns

```
[1] 89
```

which is the fifth observation in the column labeled *arch1*.

Usually, when a new data frame is created, the following two commands are issued.

```
attach(results)
```

```
names(results)
```

which give

```
[1] "gender" "arch1"  "prog1"  "arch2"  "prog2"
```

indicating that the column variables can be accessed without the prefix `results`. For example,

```
arch1[5]
```

gives

```
[1] 89
```

The command `read.table` assumes that the data in the text file are separated by spaces. Other forms include:

`read.csv`, used when the data points are separated by commas;

`read.csv2`, used when the data are separated by semicolons.

It is also possible to enter data into a spreadsheet and store it in a data frame, by writing

```
newdata <- data.frame()  
fix(newdata)
```

which brings up a blank spreadsheet called `newdata`, and the user may then enter the variable labels and the variable values.

Right click and *close* creates a data frame `newdata` in which the new information is stored.

If you subsequently need to amend or add to this data frame write

```
fix(newdata)
```

which retrieves the spreadsheet with the data. You can then edit the data as required.

Right click and *close* saves the amended data frame.

1.7 MISSING VALUES

R allows vectors to contain a special *NA* value to indicate that the data point is not available. In the second record in `results.txt`, notice that *NA* appears for `arch1` and `prog1`. This means that the marks for this student are not available in Architecture and Programming in the first semester; the student may not have sat these examinations. The absent marks are referred to as *missing values*, and are not included at the analysis stage.

1.8 EDITING

It is possible to edit data that have been already entered and to edit and invoke commands that have been previously used.

1.8.1 Data Editing

The data you have read and stored may be edited and changed interactively during your *R* session. Simply click on *Edit* on the toolbar to get access to the *Data Editor*, which allows you to bring up any data frame as a spreadsheet. You can edit its entries as you wish.

It is also possible to change particular entries of a data frame. For example,

```
arch1[7] <- 10
```

changes the mark for the seventh student in *arch1* in the data frame *results* from 100 to 10. It may have been entered as 100 in error.

1.8.2 Command Editing

The command

```
history()
```

brings up the previous 25 commands on a separate screen. These can be edited and/or used again as you wish.

```
history(max.show = Inf)
```

retrieves all previous commands that you have used.

1.9 TIDYING UP

As your *R* session continues, you may find that the set of objects you have used has become unwieldy, and you may want to remove some. To see what the workspace contains write

```
ls()
```

or equivalently

```
objects()
```

which causes all objects in the workspace to appear on the screen. If you have run the preceding examples in this chapter, the following should appear.

```
[1] "downtime"      "newdata"       "prod1"      "results"      "x"  
[6] "x"           "x2"
```

The content of the workspace can also be examined from the toolbar; go to *Misc* and choose *List Objects*.

To tidy up, you might want to remove some.

```
rm(x2)
```

removes the object x2.

To remove the complete workspace, write

```
rm(list = ls())
```

1.10 SAVING AND RETRIEVING

To save the entire workspace, click *File/Save Workspace* on the tool bar. You will then be given the opportunity to specify the location where you want to save the workspace. The workspace is saved to a file with *Rdata* attachment.

A saved workspace may be retrieved at *File* on the toolbar by clicking on *Load Workspace*, and specifying its location.

In a similar fashion, a file containing the commands may be saved and loaded by clicking on *Save History* or *Load History*, and specifying its location. The commands are stored in a file with *RHistory* attachment.

1.11 PACKAGES

A key feature of *R* is that it is a library of packages as much as it is a programming language. Packages are a collection of stored functions, data, and code that can be loaded for specific needs. The base installation of *R* contains a set of packages, which are sufficient for most users' needs. To see what packages are available with the base installation, type

```
search()
```

which gives

```
[1] ".GlobalEnv"      "package:stats"      "package:graphics"  
[4] "package:grDevices" "package:utils"      "package:datasets"
```

```
[7] "package:methods"     "Autoloads"           "package:base"
```

The `library` function enables you to see what is available in any particular package. For example, by writing

```
library(help = "stats")
```

you will get a list of statistical procedures available in the “stats” package.

By clicking on `packages` on the toolbar, you will be given a list of available packages. If you want to install a specific package, “Matrix” say, just click on “Matrix.” Alternatively, write

```
install.packages("Matrix")
```

in the command window.

```
library(help = "Matrix")
```

tells us what procedures are in the package “Matrix.”

```
installed.packages()
```

details all installed packages.

Currently, there are thousands of packages available. It is unlikely that you will need any of these at this stage of your learning, as the packages available in the base package are usually adequate for the general users’ needs.

1.12 INTERFACES

Over the last decade, many attempts have been made to develop interfaces for *R* to make it more user-friendly. Some interfaces are enhanced code editors that interact with *R* in order to make it easier to use; *RStudio* and *R tools for Visual Studio* are two such examples, both of which provide an integrated development environment (IDE) for working with *R*.

There are also full-blown GUI such as *R Commander* (*Rcmdr*), *Rattle* and *RKWard*, all of which contain menus and dialog boxes with a point-and-click approach.

We will look at two of the most popular of these interfaces, *RStudio*, and *R Commander*.

1.12.1 RStudio

RStudio is the most well known of the code editors currently available to interface with *R*. Written in C++, Version 1 was released in 2016, and since then many updates and revisions have been made. Like *R*, it is open source, and, after you have installed *R*, it can be downloaded from the *RStudio* website at <http://www.rstudio.org> by proceeding as follows:

- Click “Download RStudio”;

- Choose the Desktop version;
- Select the setup program suitable to your operating system;
- Press the option “Run”.

RStudio is now installed, and you should see an icon on your screen. Clicking on this will start up and launch the *RStudio* interface. A screen divided into four windows will appear, each window representing a set of integrated tools designed to help you to be more productive in *R*.

- The top left quadrant is the *Editor Window*.
- The lower left quadrant is the *Command Window* where you write and execute commands after the prompt “>” similar to what you do in the main *R*. This is where you will first work, and when it is correct, copy it into the *Editor Window*.
- The top right quadrant is the *Workspace and History Window*. This keeps a record of the commands you have used.
- The lower right quadrant is referred to as the *File/Plots/Packages Window*. Here, you can open files, view plots and load packages. This window also contains the help function.

To see for yourselves, click on the *RStudio* icon. Similar to *R*, *RStudio* can be run on a desktop using Windows, (Mac) OS X, and Linux or in a browser connected to *RStudio*.

1.12.2 *R Commander*

Of the GUI available, *R Commander*, also known as *Rcmdr*, is the most popular with a point-and-click interface to many statistical tasks. It is called “commander” because every time you make a selection from the menu, the *R* commands corresponding to the task is listed in the output window. You can save this code to be used again. *Rcmdr* is obtained by going into *R*, clicking on packages on the tool bar, and downloading *Rcmdr*.

EXERCISES 1.1

1. In a class of 50 students of computing, 23 are female and 27 are male. The results of their first-year Java programming examination are given as follows:

Females: 57, 59, 78, 79, 60, 65, 68, 71, 75, 48, 51, 55, 56, 41, 43,
44, 75, 78, 80, 81, 83, 83, 85

Males: 48, 49, 49, 30, 30, 31, 32, 35, 37, 41, 86, 42, 51, 53, 56,
42, 44, 50, 51, 65, 67, 51, 56, 58, 64, 64, 75

- (a) Read these data into *R* by storing them in the following ways:

- As two vectors, one for the females and one for the males;
 - As one vector, with a factor vector designating the gender.
- (b) If it was discovered that the mark for the 34th student was entered incorrectly and should have been 46 instead of 86, use an appropriate editing procedure to change this.
- (c) Save the workspace in a file in a suitable directory for access later.

1.13 PROJECT

Download *RStudio* and familiarize yourself with its workings, by using it to do Exercise 1. Decide, at this point, whether you prefer using *RStudio* for your data analysis rather than using *R* directly. Follow your preference throughout the rest of the book. It is up to you!

REFERENCE

Venables, W.N., Smith, D.M. and the *R* Core Team (2018), *An Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics*, Version 3.6.1 (2019-07-05).

2

SUMMARIZING STATISTICAL DATA

In this chapter, we explore some of the procedures available in *R* to summarize statistical data, and we give some examples of writing programs.

2.1 MEASURES OF CENTRAL TENDENCY

Measures of central tendency are typical or central points in the data. The most commonly used are the mean and the median.

Mean: The mean is the sum of all values divided by the number of cases, excluding the missing values.

To obtain the mean of the data in Example 1.1 stored in *downtime* write

```
mean(downtime)
```

```
[1] 25.04348
```

So the average downtime of all the computers in the laboratory is just over 25 minutes.

Going back to the original data in Exercise 1.1 stored in *marks*, to obtain the mean, write

```
mean(marks)
```

which gives

```
[1] 57.44
```

To obtain the mean marks for females, write

```
mean(marks[1:23])  
[1] 65.86957
```

For males,

```
mean(marks[24:50])  
[1] 50.25926
```

illustrating that the female average is substantially higher than the male average.

To obtain the mean of the corrected data in Exercise 1.1, recall that the mark of 86 for the 34th student on the list was an error, and that it should have been 46. We changed it with

```
marks[34] <- 46
```

The new overall average is

```
mean(marks)  
56.64
```

and the new male average is

```
mean(marks[24:50])  
[1] 48.77778
```

increasing the gap between the male and female averages even further.

If we perform a similar operation for the variables in the examination data given in Example 1.2, we run into trouble. Suppose we want the mean mark for Architecture in Semester 1. In *R*

```
mean(arch1)
```

gives

```
[1] NA
```

Recall that, in the results file, we recorded the missing marks with the special value *NA* to indicate that these marks were “not available”. *R* will not perform arithmetic

operations on objects containing *NA*, unless specifically mandated to skip/remove missing values. To do this, you need to insert the argument `na.rm = T` or `na.rm = TRUE`, (not available, remove) into the function.

For `arch1`, writing

```
mean(arch1, na.rm = TRUE)
```

yields

```
[1] 63.56897
```

To obtain the mean of all the variables in *results* file, we use the *R* function `sapply`.

```
sapply(results, mean, na.rm = T)
```

yields

gender	arch1	prog1	arch2	prog2
NA	63.56897	59.01709	51.97391	53.78378

Notice that a *NA* message is returned for gender. The reason for this is that the gender variable is nonnumeric, and *R* cannot calculate its mean. We could, instead specify the columns that we want to work on.

```
sapply(results[2:5], mean, na.rm = TRUE)
```

gives

arch1	prog1	arch2	prog2
63.56897	59.01709	51.97391	53.78378

Median: The median is the middle value of the data set; 50% of the observations is less and 50% is more than this value.

In *R*

```
median(downtime)
```

yields

```
[1] 25
```

which means that 50% of the computers experienced less than 25 minutes of downtime, while 50% experienced more than 25 minutes of downtime.

Also,

```
median(marks)
[1] 55.5
```

In both of these examples (*downtime* and *marks*), you will observe that the medians are not too far away from their respective means.

The median is particularly useful when there are extreme values in the data. Let us look at another example.

Example 2.1 Apps Usage

Examining the nine apps with greatest usage on your smartphone, you may find the usage statistics (in MB) are

App	Usage (MB)
Facebook	39.72
Chrome	35.37
WhatsApp	5.73
Google	5.60
System Account	3.30
Instagram	3.22
Gmail	2.52
Messenger	1.71
Maps	1.55

To enter the data, write

```
usage <- c(39.72, 35.27, 5.73, 5.6, 3.3, 3.22, 2.52, 1.71, 1.55)
```

The mean is

```
mean(usage)
[1] 10.95778
```

while the median is

```
median(usage)
[1] 3.3
```

Unlike the previous examples, where the mean and median were similar, here the mean is more than three times the median. Looking at the data again, you will notice that the usage of the first two apps, Facebook and Chrome, is much larger than the usages of the other apps in the data set. These values are the cause of the mean being

so high. Such values are often designated as outliers and are analyzed separately. Omitting them and calculating the mean and median once more, we get

```
mean(usage[3:9])
[1] 3.375714

median(usage[3:9])
[1] 3.22
```

Now, we see that there is not much difference between the mean and median. \triangleleft

When there are extremely high values in the data, using the mean as a measure of central tendency gives the wrong impression. A classic example of this is wage statistics where there may be a few instances of very high salaries, which will grossly inflate the average, giving the impression that salaries are higher than they actually are.

2.2 MEASURES OF DISPERSION

Measures of dispersion, as the name suggests, estimate the spread or variation in a data set. There are many ways of measuring spread, and we consider some of the most common.

Range: The simplest measure of spread of data is the range, which is the difference between the maximum and the minimum values.

```
rangedown <- max(downtime) - min(downtime)
rangedown
[1] 51
```

tells us that the range in the downtime data is 51 minutes.

```
rangearch1 <- max(arch1, na.rm = T) - min(arch1, na.rm = T)
rangearch1
[1] 97
```

gives the range of the marks awarded in Architecture in Semester 1.

The *R* function `range` may also be used.

```
range(arch1, na.rm = TRUE)
[1] 3 100
```

which gives the minimum (3) and the maximum (100) of the marks obtained in Architecture in Semester 1.

Note that, since `arch1` contains missing values, the declaration of `na.rm = T` or equivalently `na.rm = TRUE` needs to be used.

To get the range for all the examination subjects in `results`, we use the function `sapply`.

```
sapply(results[2:5], range, na.rm = TRUE)
```

gives the minimum and maximum of each subject.

	arch1	prog1	arch2	prog2
[1,]	3	12	6	5
[2,]	100	98	98	97

Standard deviation: The standard deviation (`sd`) measures how much the data values deviate from their average. It is the square root of the average squared deviations from the mean. A small standard deviation implies most values are near the mean. A large standard deviation indicates that values are widely spread above and below the mean.

In *R*

```
sd(downtime)
```

yields

```
[1] 14.27164.
```

Recall that we calculated the mean to be 25.04 minutes. We might loosely describe the downtime as being “25 minutes on average give or take 14 minutes.”

For the data in `results`

```
sapply(results[2:5], sd, na.rm = TRUE)
```

gives the standard deviation of each examination subject in `results`:

	arch1	prog1	arch2	prog2
	24.37469	23.24012	21.99061	27.08082

Quantiles: The quantiles divide the data into proportions, usually into quarters called quartiles, tenths called deciles, and percentages called percentiles. The default calculation in *R* is quartiles.

```
quantile(downtime)
```

gives

0%	25%	50%	75%	100%
0.0	16.0	25.0	31.5	51.0

The first quartile (16.0) is the value that breaks the data so that 25% is below this value and 75% is above.

The second quartile (25.0) is the value that breaks the data so that 50% is below and 50% is above (notice that the 2nd quartile is the median).

The third quartile (31.5) is the value that breaks the data so that 75% is below and 25% is above.

We could say that 25% of the computer systems in the laboratory experienced less than 16 minutes of downtime, another 25% of them were down for between 16 and 25 minutes, and so on.

Interquartile range: The difference between the first and third quartiles is called the *interquartile range* and is sometimes used as a rough estimate of the standard deviation. In downtime it is $31.5 - 16.0 = 15.5$, not too far away from 14.27, which we calculated to be the standard deviation.

Deciles: Deciles divide the data into tenths. To get the deciles in R, first define the required break points

```
deciles <- seq(0, 1, 0.1)
```

The function `seq` creates a vector consisting of an equidistant series of numbers. In this case, `seq` assigns values in [0, 1] in intervals of 0.1 to the vector called `deciles`. Writing in R

```
deciles
```

shows what the vector contains

```
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Adding this extra argument to the quantile function

```
quantile(downtime, deciles)
```

yields

0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0.0	4.0	12.8	19.8	22.6	25.0	29.2	30.0	34.8	44.8	51.0

Interpreting this output, we could say that 90% of the computer systems in the laboratory experienced less than 45 minutes of downtime.

Similarly, for the percentiles, use

```
percentiles <- seq(0, 1, 0.01)
```

as an argument in the quantile function, and write

```
quantile(downtime, percentiles)
```

2.3 OVERALL SUMMARY STATISTICS

A quicker way of summarizing the data is to use the `summary` function.

```
summary(downtime)
```

returns

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	16.00	25.00	25.04	31.50	51.00

which are the minimum the first quartile, the median, the mean, the third quartile, and the maximum, respectively.

For `arch1`, we might write

```
summary(arch1)
```

which gives

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
3.00	46.75	68.50	63.57	83.25	100.00	3.00

An entire data frame may be summarized by using the `summary` command. Let us do this in the data frame `results`. First, it is wise to make a declaration about the categorical variable `gender`.

```
gender <- factor(gender)
```

designates the variable `gender` as a factor, and ensures that it is treated as such in the `summary` function.

```
summary(results)
gender      arch1          prog1          arch2          prog2
f: 19  Min.   : 3.00  Min.   :12.00  Min.   : 6.00  Min.   : 5.00
m:100  1st Qu.:46.75  1st Qu.:40.00  1st Qu.:40.00  1st Qu.:30.00
        Median :68.50  Median :64.00  Median :48.00  Median :57.00
        Mean   :63.57  Mean   :59.02  Mean   :51.97  Mean   :53.78
        3rd Qu.:83.25  3rd Qu.:78.00  3rd Qu.:61.00  3rd Qu.:76.50
        Max.   :100.00  Max.   :98.00  Max.   :98.00  Max.   :97.00
        NA's   : 3.00  NA's   : 2.00  NA's   : 4.00  NA's   : 8.00
```

Notice how the display for `gender` is different than that for the other variables; we are simply given the frequency for each gender.

2.4 PROGRAMMING IN R

One of the great benefits of *R* is that it is possible to write your own programs and use them as functions in your analysis. Programming is extremely simple in *R* because of the way it handles vectors and data frames. To illustrate, let us write a program to calculate the mean of *downtime*. The formula for the mean of a variable *x* with *n* values is given by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

In standard programming languages, implementing this formula would necessitate initialization and loops, but with *R*, statistical calculations such as these are much easier to implement. For example,

```
sum(downtime)
```

gives

576

which is the sum of the elements in *downtime*

```
length(downtime)
```

gives

23

gives the number of elements in *downtime*.

To calculate the mean, write

```
meandown <- sum(downtime)/length(downtime)
meandown
[1] 25.04348
```

Let us also look at how to calculate the standard deviation of the data in *downtime*.

The formula for the standard deviation of *n* data points stored in an *x* vector is

$$sd = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

We illustrate step by step how this is calculated for *downtime*.

First, subtract the mean from each data point.

```
downtime - meandown
[1] -25.04347826 -24.04347826 -23.04347826 -13.04347826 -13.04347826
[6] -11.04347826 -7.04347826 -4.04347826 -4.04347826 -2.04347826
[11] -1.04347826 -0.04347826 3.95652174 2.95652174 4.95652174
[16] 4.95652174 4.95652174 7.95652174 10.95652174 18.95652174
[21] 19.95652174 21.95652174 25.95652174
```

Then, obtain the squares of these differences.

```
(downtime - meandown) ^ 2
[1] 6.271758e+02 5.780888e+02 5.310019e+02 1.701323e+02 1.701323e+02
[6] 1.219584e+02 4.961059e+01 1.634972e+01 1.634972e+01 4.175803e+00
[11] 1.088847e+00 1.890359e-03 1.565406e+01 8.741021e+00 2.456711e+01
[16] 2.456711e+01 2.456711e+01 6.330624e+01 1.200454e+02 3.593497e+02
[21] 3.982628e+02 4.820888e+02 6.737410e+02
```

Sum the squared differences.

```
sum( (downtime - meandown) ^ 2)
[1] 4480.957
```

Finally, divide this sum by `length(downtime) - 1` and take the square root.

```
sqrt( sum( (downtime -meandown) ^ 2) / (length(downtime) -1) )
[1] 14.27164
```

You will recall that *R* has built-in functions to calculate the most commonly used statistical measures. You will also recall that the mean and the standard deviation can be obtained directly with

```
mean(downtime)
[1] 25.04348
sd(downtime)
[1] 14.27164
```

We took you through the calculations to illustrate how easy it is to program in *R*.

2.4.1 Creating Functions

Occasionally, you might require some statistical functions that are not available in R. You will need to create your own function. Let us take, as an example, the skewness coefficient, which measures how much the data differ from symmetry.

The skewness coefficient is defined as

$$\text{skew} = \frac{\sqrt{n} \sum_{i=1}^n (x_i - \bar{x})^3}{\left(\sum_{i=1}^n (x_i - \bar{x})^2 \right)^{3/2}}. \quad (2.1)$$

A perfectly symmetrical set of data will have a skewness of 0; when the skewness coefficient is substantially greater than 0, the data are positively asymmetric with a long tail to the right, and a negative skewness coefficient means that data are negatively asymmetric with a long tail to the left. As a rule of thumb, if the skewness is outside the interval $(-1, 1)$, the data are considered to be highly skewed. If it is between -1 and -0.5 or 0.5 and 1 , the data are moderately skewed.

Example 2.2 A program to calculate skewness

The following syntax calculates the skewness coefficient of a set of data and assigns it to a function called *skew* that has one argument (*x*).

```
skew <- function(x) {
  xbar <- mean(x)
  sum2 <- sum((x-xbar)^2, na.rm = T)
  sum3 <- sum((x-xbar)^3, na.rm = T)
  skew <- (sqrt(length(x)) * sum3) / (sum2^(1.5))
  skew}
```

You will agree that the conventions of vector calculations make it very easy to calculate statistical functions.

When *skew* has been defined, you can calculate the skewness on any data set. For example,

```
skew(downtime)
```

gives

```
[1] -0.04818095
```

which indicates that the *downtime* data is slightly negatively skewed.

Looking again at the data given Example 2.1, let us calculate the skewness coefficient

```
skew(usage)
[1] 1.322147
```

which illustrates that the data is highly skewed. Recall that the first two values are outliers in the sense that they are very much larger than the other values in the data set. If we calculate the skewness with those values removed, we get

```
skew(usage[3:9])
[1] 0.4651059
```

which is very much smaller than that obtained with the full set. \triangleleft

2.4.2 Scripts

There are various ways of developing programs in *R*.

The most useful way of writing programs is by means of *R*'s own built-in editor called *Scripts*. From *File* at the toolbar click on *New Script* (*File/New Script*). You are then presented with a blank screen to develop your program. When done, you may save and retrieve this program as you wish. *File/Save* causes the file to be saved. You may designate the name you want to call it, and it will be given a *.R* extension. In subsequent sessions, *File/Open Script* brings up all the *.R* files that you have saved. You can select the one you wish to use.

When you want to execute a line or group of lines, highlight them and press *Ctrl/R*, that is, *Ctrl* and the letter *R* simultaneously. The commands are then transferred to the control window and executed.

Alternatively, if the program is short, it may be developed interactively while working at your computer.

Programs may also be developed in a text editor, like Notepad, saved with the *.R* extension and retrieved using the *source* statement.

```
source("C:\\test")
```

retrieves the program named *test.R* from the C directory. Another way of doing this, while working in *R*, is to click on *File* on the tool bar where you will be given the option to *Source R code*, and then you can browse and retrieve the program you require.

EXERCISES 2.1

1. For the class of 50 students of computing detailed in Exercise 1.1, use *R* to:
 - (a) obtain the summary statistics for each gender, and for the entire class;

- (b) calculate the deciles for each gender and for the entire class;
(c) obtain the skewness coefficient for the females and for the males.
2. It is required to estimate the number of message buffers in use in the main memory of the computer system at Power Products Ltd. To do this, 20 programs were run, and the number of message buffers in use were found to be

141, 146, 157, 151, 152, 140, 142, 156, 150, 140,
139, 135, 143, 146, 146, 152, 140, 136, 149, 148

Calculate the average number of buffers used. What is the standard deviation? Would you say these data are skewed?

3. To get an idea of the runtime of a particular server, 20 jobs were processed and their execution times (in seconds) were observed as follows:

0.1, 0.9, 1.1, 2.3, 3.4, 16.1, 2.7, 3.2, 2.0, 1.5,
2.2, 2.0, 1.3, 1.4, 0.6, 0.7, 2.5, 1.2, 3.7, 2.7

Examine these data and calculate appropriate measures of central tendency and dispersion.

4. Ten data sets were used to run a program and measure the execution time. The results (in milliseconds) were observed as follows:

43, 42, 55, 67, 101, 200, 58, 52, 48, 62

Use appropriate measures of central tendency and dispersion to describe these data.

5. The following data give the amount of time (in minutes) in one day spent on Facebook by each of 15 students.

65, 240, 70, 35, 10, 360, 45, 50, 20, 15, 40, 60, 65, 55, 60

Obtain appropriate measures of central tendency and measures of dispersion for these data.

2.5 PROJECT

Write the skewness program, and use it to calculate the skewness coefficient of the four examination subjects in *results.txt*. What can you say about these data?

Pearson has given an approximate formula for the skewness that is easier to calculate than the exact formula given in Equation 2.1.

$$\text{skew} \approx \frac{3(\text{mean} - \text{median})}{\text{standard deviation}}$$

Write a program to calculate this, and apply it to the data in *results.txt*. Is it a reasonable approximation?

3

GRAPHICAL DISPLAYS

In addition to numerical summaries of statistical data, there are various pictorial representations and graphical displays available in *R* that have a more dramatic impact and make for a better understanding of the data. The ease and speed with which graphical displays can be produced is one of the important features of *R*. By writing

```
demo(graphics)
```

you will see examples of the many graphical procedures of *R*, along with the code needed to implement them. In this chapter, we will examine some of the most common of these.

3.1 BOXPLOTS

A boxplot is a graphical summary based on the median, quartiles, and extreme values. To display the *downtime* data given in Example 1.1 using a boxplot, write

```
boxplot(downtime)
```

which gives Fig. 3.1. Often called the *Box and Whiskers Plot*, the box represents the interquartile range that contains 50% of cases. The whiskers are the lines that extend

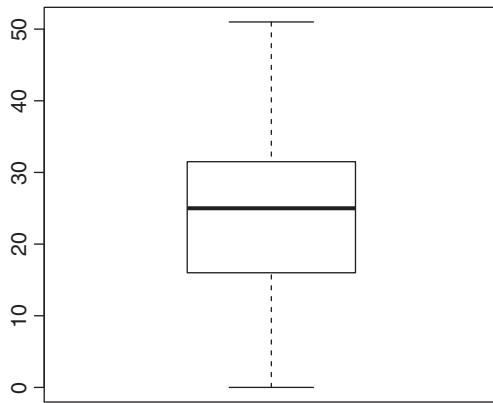


FIGURE 3.1 A Simple Boxplot

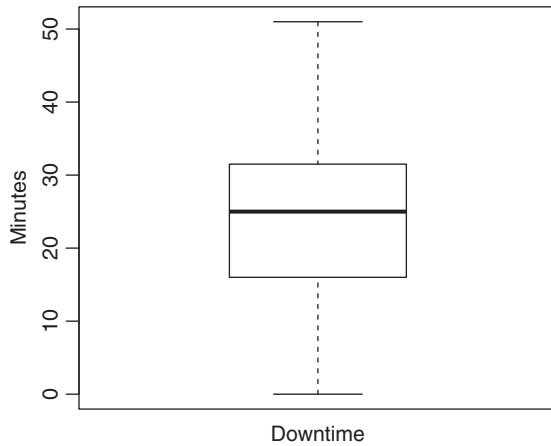


FIGURE 3.2 A Boxplot with Axis Labels

from the box to the highest and lowest values. The line across the box indicates the median.

To improve the look of the graph, we could label the axes as follows:

```
boxplot(downtime, xlab = "Downtime", ylab = "Minutes")
```

which gives Fig. 3.2.

Multiple boxplots can be displayed on the same axis, by adding extra arguments to the boxplot function. For example,

```
boxplot(results$arch1, results$arch2,
        xlab = "Architecture Semesters 1 and 2")
```

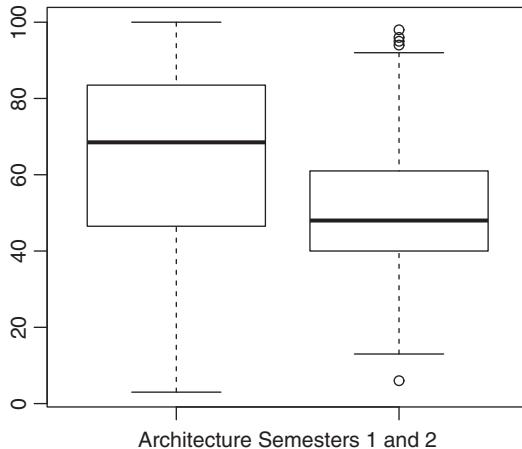


FIGURE 3.3 Multiple Boxplots

or simply

```
boxplot(arch1, arch2,
       xlab = "Architecture Semesters 1 and 2")
```

gives Fig. 3.3.

Figure 3.3 allows us to compare the performance of the students in Architecture in the two semesters. It shows, for example, that the marks are lower in Architecture in Semester 2 and the range of marks is narrower than those obtained in Architecture in Semester 1.

Notice also in Fig. 3.3 that there are points outside the whiskers of the boxplot in Architecture in Semester 2. These points represent cases over 1.5 box lengths from the upper or lower end of the box and are called *outliers*. They are considered atypical of the data in general, being either extremely low or extremely high compared to the rest of the data.

Looking at Exercise 1.1 with the uncorrected data, Fig. 3.4 is obtained using

```
boxplot(marks~gendermarks)
```

Notice the outlier in Fig. 3.4 in the male boxplot, a value that appears large compared to the rest of the data. You will recall that a check on the examination results indicated that this value should have been 46, not 86, and we corrected it using

```
marks[34] <- 46
```

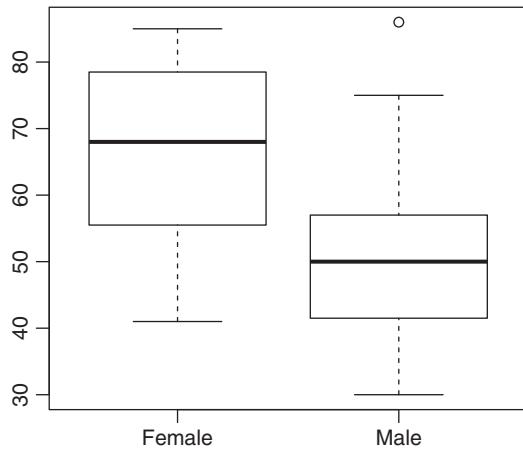


FIGURE 3.4 A Gender Comparison

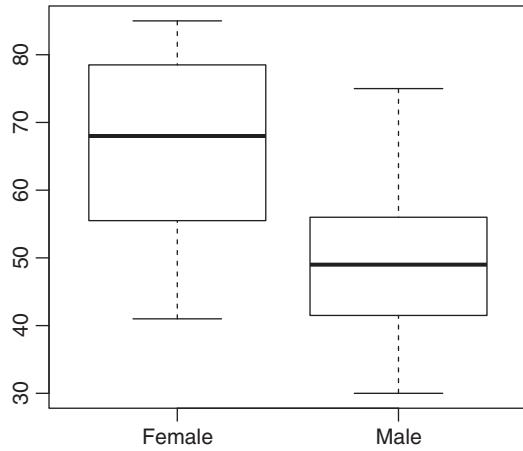


FIGURE 3.5 A Gender Comparison (corrected)

Repeating the analysis, after making this correction

```
boxplot(marks~gendermarks)
```

gives Fig. 3.5.

You will now observe from Fig. 3.5 that there are no outliers in the male or female data. In this way, a boxplot may be used as a data validation tool. Of course, it is possible that the mark of 86 may in fact be valid, and that a male student did indeed

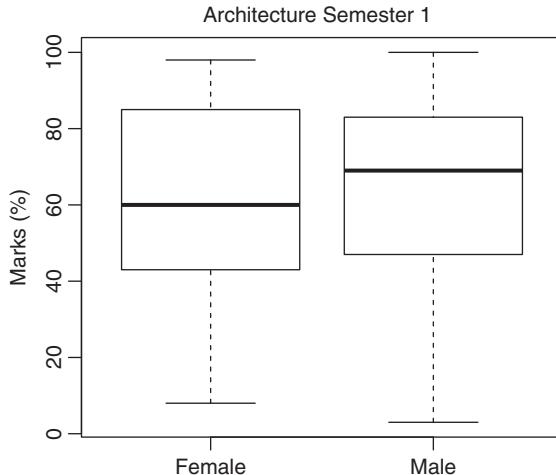


FIGURE 3.6 A Gender Comparison

obtain a mark that was much higher than his classmates. A boxplot highlights this and alerts us to the possibility of an error.

To compare the performance of females and males in Architecture in Semester 1, write

```
gender <- factor(gender, levels = c("f", "m"),
                  labels = c("Female", "Male"))
```

which changes the labels from “f” and “m” to “Female” and “Male,” respectively. Then

```
boxplot(arch1~gender, ylab = "Marks (%)",
        main = "Architecture Semester 1", font.main = 1)
```

outputs Fig. 3.6.

Notice the effect of using `main = "Architecture Semester 1"` that puts the title on the diagram. Also, the use of `font.main = 1` ensures that the main title is in plain font.

We can display plots as a matrix using the `par` function: `par(mfrow = c(2, 2))` causes the outputs to be displayed in a 2×2 array.

```
par(mfrow = c(2, 2))
boxplot(arch1~gender, main = "Architecture Semester 1", font.main = 1)
boxplot(arch2~gender, main = "Architecture Semester 2", font.main = 1)
boxplot(prog1~gender, main = "Programming Semester 1", font.main = 1)
boxplot(prog2~gender, main = "Programming Semester 2", font.main = 1)
```

produces Fig. 3.7.

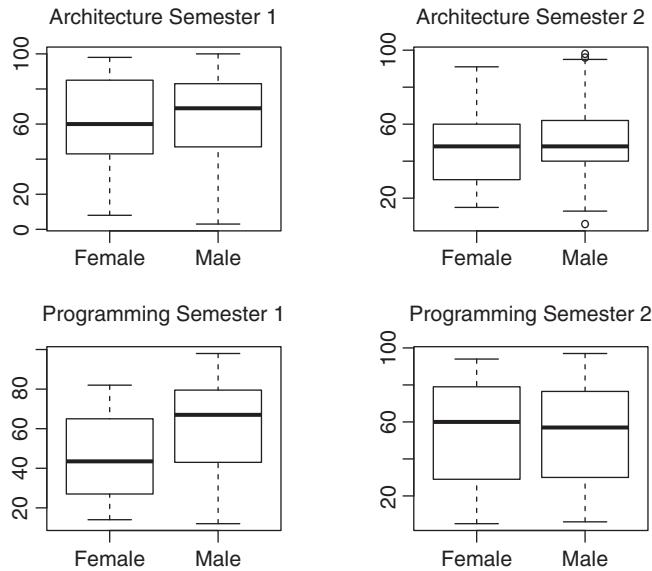


FIGURE 3.7 A Lattice of Boxplots

We see from Fig. 3.7 that female students seem to do less well than their male counterparts in Programming in Semester 1, where the median mark of the females is considerably lower than that of the males: it is lower even than the first quartile of the male marks. In the other subjects, there do not appear to be any substantial differences.

To undo a matrix-type output, write

```
par(mfrow = c(1, 1))
```

which restores the graphics output to the full screen.

3.2 HISTOGRAMS

A histogram is a graphical display of frequencies in categories of a variable and is the traditional way of examining the “shape” of the data.

```
hist(prog1, xlab ="Marks (%)",
      main = "Programming Semester 1")
```

yields Fig. 3.8.

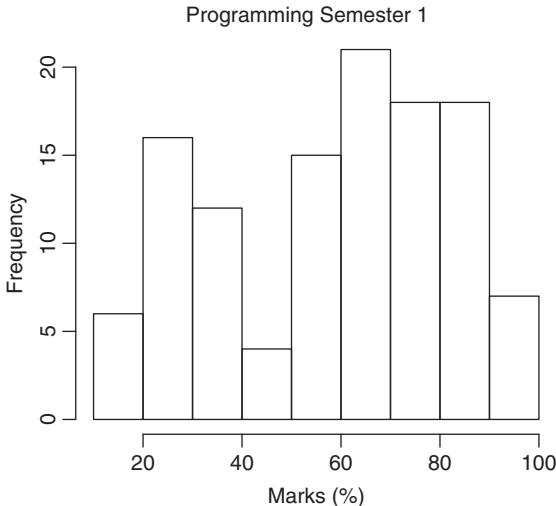


FIGURE 3.8 A Histogram with Default Breaks

As we can see from Fig. 3.8, `hist` gives the count of the observations that fall within the categories or “bins” as they are sometimes called. *R* chooses a “suitable” number of categories, unless otherwise specified. Alternatively, `breaks` may be used as an argument in `hist` to determine the number of categories. For example, to get five categories of equal width, you need to include `breaks = 5` as an argument.

```
hist(prog1, xlab = "Marks (%)",
      main = "Programming Semester 1", breaks = 5)
```

gives Fig. 3.9

Recall that `par` can be used to represent all the subjects in one diagram. Type

```
par (mfrow = c(2,2))
hist(arch1, xlab = "Architecture",
      main = "Semester 1", ylim = c(0, 35))
hist(arch2, xlab = "Architecture",
      main = "Semester 2", ylim = c(0, 35))
hist(prog1, xlab = "Programming",
      main = " ", ylim = c(0, 35))
hist(prog2, xlab = "Programming",
      main = " ", ylim = c(0, 35))
```

to get Fig. 3.10. The `ylim = c(0, 35)` ensures that the y-axis is the same scale for all the four subjects.

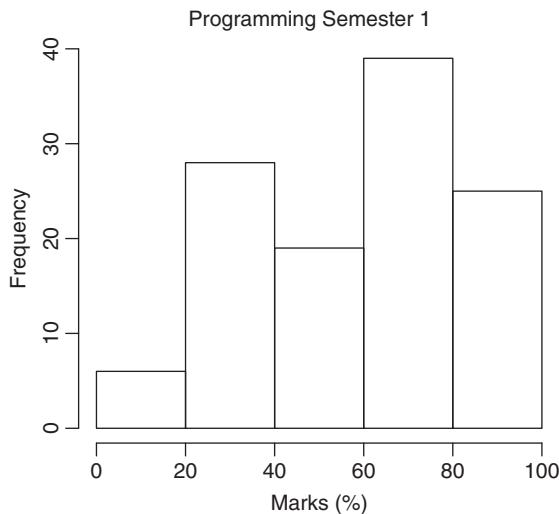


FIGURE 3.9 A Histogram with Five Breaks of Equal Width

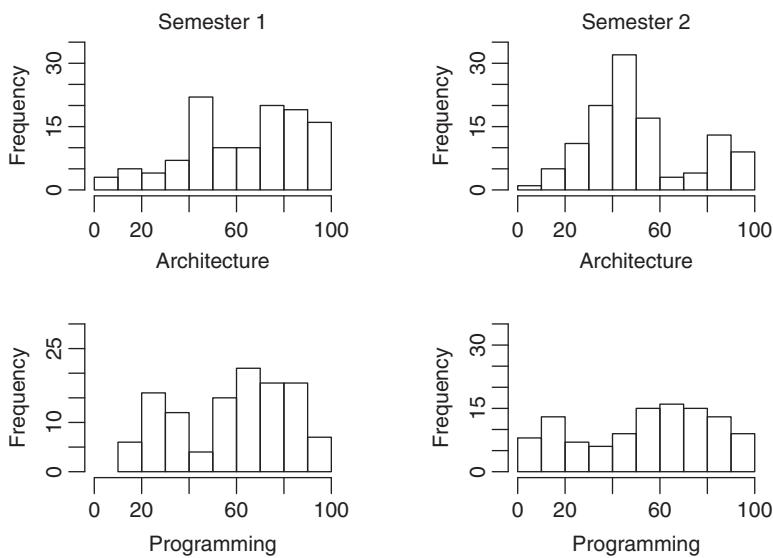


FIGURE 3.10 Histogram of Each Subject in Each Semester

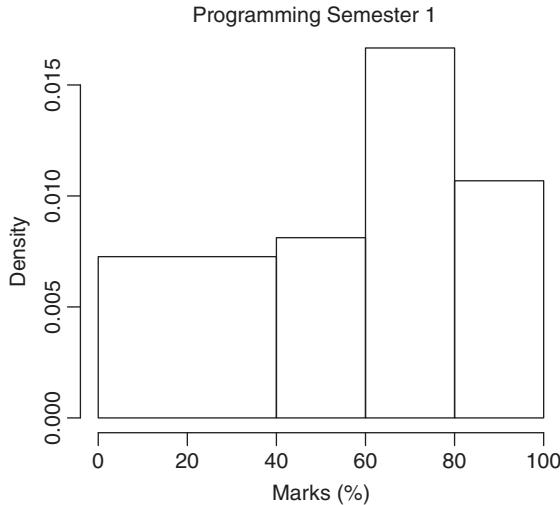


FIGURE 3.11 A Histogram with Breaks of a Specified Width

Up until now, we have invoked the default parameters of the histogram, notably the bin widths are equal and the frequency in each bin is calculated. These parameters may be changed as appropriate. For example, you may want to specify the bin break-points to represent the failures and the various classes of passes and honors.

```
bins <- c(0, 40, 60, 80, 100)

hist(prog1, xlab = "Marks (%)",
      main = "Programming Semester 1", breaks = bins)
```

yields Fig. 3.11.

In Fig. 3.11, observe that the y-axis now represents the density. When the bins are not of equal length, *R* returns a normalized histogram, so that its total area is equal to one.

To get a histogram of percentages, write in *R*

```
h <- hist(prog1, plot = FALSE, breaks = 5) #this
      postpones the plot display
h$density <- h$counts/sum(h$counts)*100 #this calculates
      percentages
plot(h, xlab = "Marks (%)", freq = FALSE,
      ylab = "Percentage", main = "Programming Semester 1")
```

The output is given in Fig. 3.12. The # allows for a comment. Anything written after # is ignored.

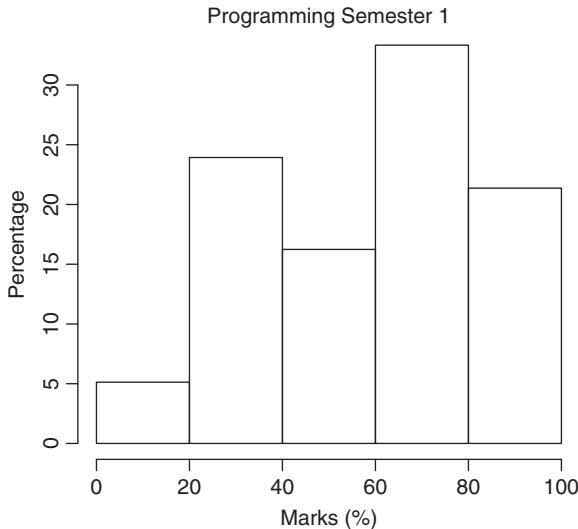


FIGURE 3.12 Histogram with Percentages

3.3 STEM AND LEAF

The stem and leaf diagram is a more modern way of displaying data than the histogram. It is a depiction of the shape of the data using the actual numbers observed. Similar to the histogram, the stem and leaf gives the frequencies of categories of the variable, but it goes further than that and gives the actual values in each category.

The marks obtained in Programming in Semester 1 are depicted as a stem and leaf diagram using

```
stem(prog1)
```

which yields Fig. 3.13.

From Fig. 3.13, we are able to see the individual observations, as well as the shape of the data as a whole. Notice that there are many marks of exactly 40, whereas just one student obtains a mark between 35 and 40. One wonders if this has anything to do with the fact that 40 is a pass, and that the examiner has been generous to borderline students. This point would go unnoticed with a histogram.

3.4 SCATTER PLOTS

Plots of data are useful to investigate relationships between variables. To examine, for example, the relationship between the performance of students in Programming in Semesters 1 and 2, we could write

```
The decimal point is 1 digit(s) to the right of the |
1 | 2344
1 | 59
2 | 11
2 | 5556777889999
3 | 0113
3 | 6
4 | 00000000
4 | 6779
5 | 12223344
5 | 56679
6 | 0011123444
6 | 566777888999
7 | 0112344
7 | 5666666899
8 | 001112222334
8 | 5678899
9 | 0122
9 | 7778
```

FIGURE 3.13 A Stem and Leaf Diagram

```
plot(prog1, prog2,
      xlab = "Programming Semester 1",
      ylab = "Programming Semester 2")
```

to obtain Fig. 3.14.

When more than two variables are involved, *R* provides a facility for producing scatter plots of all possible pairs.

To do this, first create a data frame of all the variables that you want to compare.

```
courses <- results[2:5]
```

This creates a data frame *courses* containing the second to the fifth variables in *results*, that is, *arch1*, *prog1*, *arch2*, and *prog2*. Writing

```
pairs(courses)
```

or equivalently

```
pairs(results[2:5])
```

will generate Fig. 3.15, which, as you can see, gives scatter plots for all possible pairs.

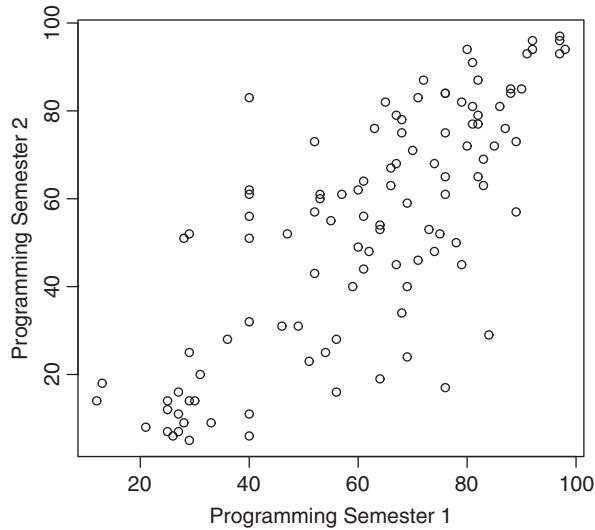


FIGURE 3.14 A Scatter Plot

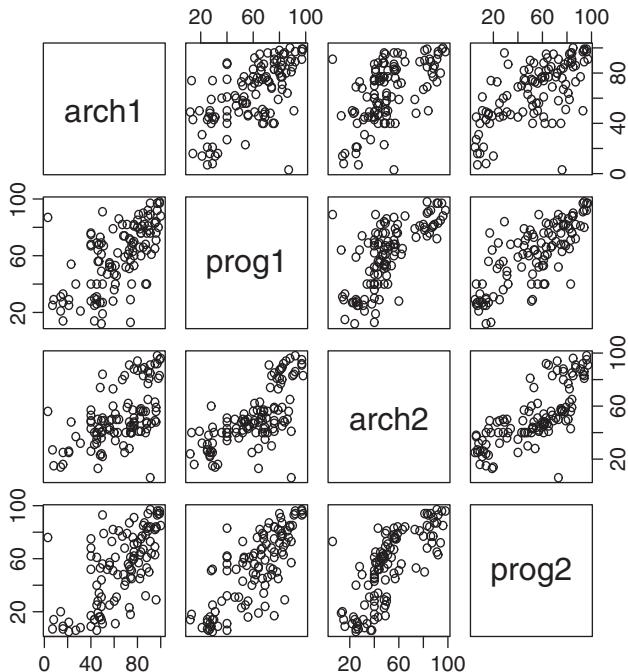


FIGURE 3.15 Use of the *Pairs* Function

3.5 THE LINE OF BEST FIT

Returning to Fig. 3.14, we can see that there is a *linear trend* in these data. One variable increases with the other; not surprisingly, students doing well in Programming in Semester 1 are likely to do well also in Programming in Semester 2, and those doing badly in Semester 1 will tend to do badly in Semester 2. We might ask, if it is possible to estimate the Semester 2 results from those obtained in Semester 1.

In the case of the Programming subjects, we have a set of points (*prog1*, *prog2*), and having established, from the scatter plot, that a linear trend exists, we attempt to fit a line that best fits the data. In *R*

```
lm(prog2~prog1)
```

calculates what is referred to as the linear model (*lm*) of *prog2* on *prog1*, or simply the line

$$\text{prog2} = a + b(\text{prog1})$$

that best fits the data.

The output is

```
Call:
lm(formula = prog2~prog1)

Coefficients:
(Intercept)      prog1
-5.455          0.960
```

Therefore, the line that best fits these data is

$$\text{prog2} = -5.455 + 0.96(\text{prog1})$$

To draw this line on the scatter diagram, write

```
plot(prog2, prog1)
abline(lm(prog2~prog1))
```

which gives Fig. 3.16.

The line of best fit may be used to make predictions. For example, we might be able to predict how students will do in Semester 2 from the results that they obtained in Semester 1. If the mark on Programming 1 for a particular student is 70, that student would be expected to do well also in Programming 2, estimated to obtain $-5.455 + 0.960 * 70 \approx 62$. A student doing badly in Programming 1, 30 say, would also be expected to do badly in Programming 2. $-5.455 + 0.960 * 30 \approx 24$. These

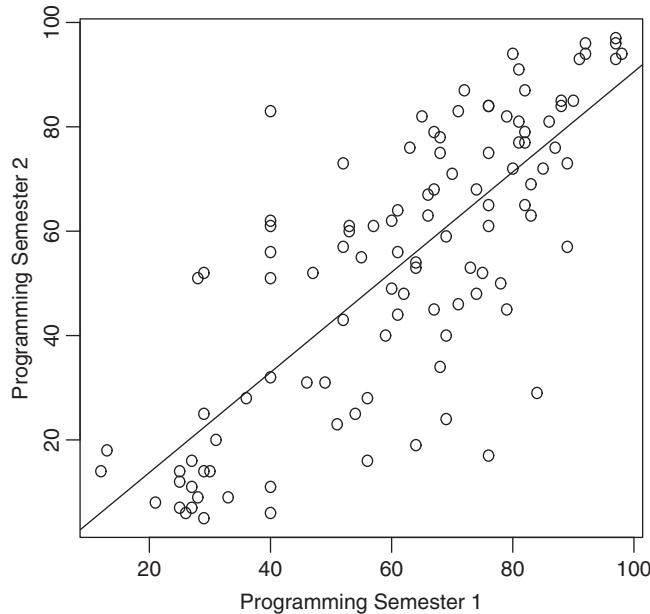


FIGURE 3.16 The Line of Best Fit

predictions may not be exact but, if the linear trend is strong and past trends continue, they will be reasonably close.

A word of warning is appropriate here. The estimated values are based on the assumption that the past trend continues. This may not always be the case. For example, students who do badly in Semester 1, may get such a shock that they work harder in Semester 2, and change the pattern. Similarly, students getting high marks in Semester 1 may be lulled into a sense of false security and take it easy in Semester 2. Consequently, they may not do as well as expected. Hence, the Semester 1 trends may not continue, and the model may no longer be valid.

3.6 MACHINE LEARNING AND THE LINE OF BEST FIT

Machine learning is the science of getting computer systems to use algorithms and statistical models to study patterns and learn from data. Supervised learning is the machine learning task of using past data to learn a function in order to predict a future output.

The *line of best fit* is one of the many techniques that machine learning has borrowed from the field of Probability and Statistics to “train” the machine to make predictions. In this case of what is also known as the *simple linear regression line* in statistics, a set of pairs (x_i, y_i) of data is obtained, x is referred to as the independent

variable, and y is the dependent variable. The objective is to estimate y from x . The line of best fit, $y = a + bx$, is obtained by choosing the intercept a and slope b so that the sum of the squared distances from the observed y_i to the estimated \hat{y}_i is minimized. The algebraic details of the derivations of a and b are given in Appendix B.

Often, the data for supervised learning are randomly divided into two parts, one for training and the other for testing. In machine learning, we derive the line of best fit from the training set

$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$$

The testing set is used to see how well the line actually fits. Usually, an 80 : 20 breakdown of the data is made, the 80% is used for “training,” that is, to obtain the line, and the 20% is used to decide if the line really fits the data, and to ascertain if the model is appropriate for future predictions. The model is updated as new data become available.

Example 3.1

Suppose there are 50 pairs (x_i, y_i) of observations available for obtaining the line that best fits the data in order to predict y from x . The data are randomly divided into the training set and testing set, using 40 observations for training (Table 3.1), and 10 for testing (Table 3.2).

Use the training set to obtain the line of best fit of y on x and the testing set to examine how well the line fits the data.

First, read the training set (x_train , y_train) into R.

```
x_train <- c(11.8, 10.8, 8.6, ..., 8.9)
y_train <- c(31.3, 59.9, 27.6, ..., 38.5)
```

and the testing set (x_test , y_test)

```
x_test <- c(8.5, 9.4, 5.4, ..., 2.8)
y_test <- c(49.4, 43.0, 19.3, ..., 13.6)
```

Then, plot the training set, to establish if a linear trend exists.

```
plot(x_train, y_train, main = "Training Data", font.main = 1)
```

gives Fig. 3.17.

Since Fig. 3.17 shows a linear trend, we obtain the line of best fit of y_train on x_train , and superimpose it on the scatter diagram in Fig. 3.17. In R, write

```
abline(lm(y_train ~ x_train))
```

to get Fig. 3.18.

TABLE 3.1 The Training Set

Observation Numbers	<i>x_train</i>	<i>y_train</i>	Observation Numbers	<i>x_train</i>	<i>y_train</i>
1	11.8	31.3	21	15.1	80.1
2	10.8	59.9	22	14.7	66.9
3	8.6	27.6	23	10.5	42.0
4	10.3	57.7	24	10.9	72.9
5	8.5	50.2	25	11.6	67.8
6	11.6	52.1	26	9.1	45.3
7	14.4	79.1	27	5.4	30.2
8	8.6	32.3	28	8.8	49.6
9	12.4	58.8	29	11.2	44.3
10	14.9	79.5	30	7.4	46.1
11	8.9	57.0	31	7.9	45.1
12	8.7	35.1	32	12.2	46.5
13	11.7	68.2	33	8.5	42.7
14	11.4	60.1	34	9.3	56.3
15	8.8	44.5	35	10.0	27.4
16	5.9	28.9	36	3.8	20.2
17	13.5	75.8	37	14.9	68.5
18	8.7	48.7	38	12.4	72.6
19	11.0	54.7	39	11.1	54.3
20	8.3	32.8	40	8.9	38.5

TABLE 3.2 The Testing Set

Observation Numbers	1	2	3	4	5	6	7	8	9	10
<i>x_test</i>	8.5	9.4	5.4	11.7	6.5	10.3	12.7	11.0	15.4	2.8
<i>y_test</i>	49.4	43.0	19.3	56.4	28.3	53.7	58.1	28.7	80.7	13.6

Next, we use the testing data to decide on the suitability of the line.

The coefficients of the line are obtained in *R* with

```
lm(formula = y_train ~ x_train)
```

Coefficients:

(Intercept)	<i>x_train</i>
-0.9764	4.9959

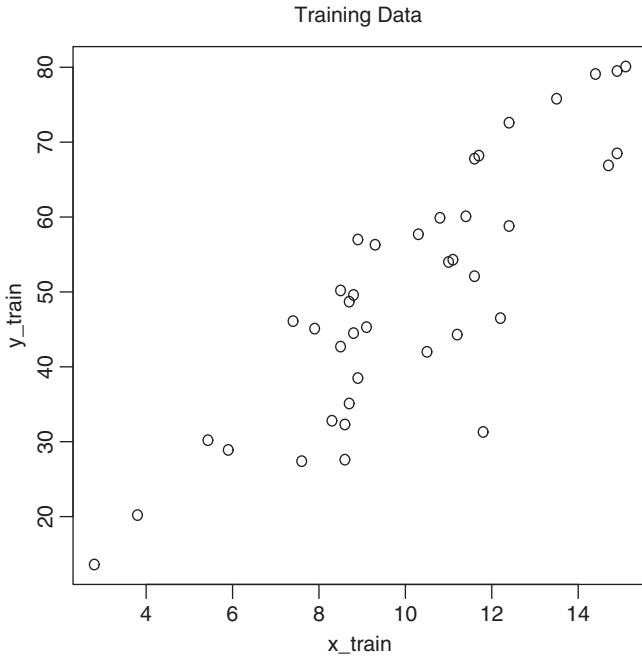


FIGURE 3.17 The Scatter of the Training Data

The estimated values y_{est} are calculated in R as follows:

```
y_est <- - 0.9764 + 4.9959 * x_test
round(y_est, 1)
```

which gives

```
y_est
41.5 46.0 26.0 57.5 31.5 50.5 62.5 54.0 76.0 13.0
```

We now compare these estimated values with the observed values.

```
y_test
49.4 43.0 19.3 56.4 28.7 53.7 58.1 54.0 80.7 13.6
```

```
plot(x_test, y_test, main = "Testing Data", font.main = 1)
abline(lm(y_train ~ x_train)) # plot the line of best fit
segments(x_test, y_test, x_test, y_est)
```

gives Fig. 3.19. Here, segments plots vertical lines between (x_{test}, y_{test}) and (x_{test}, y_{est})

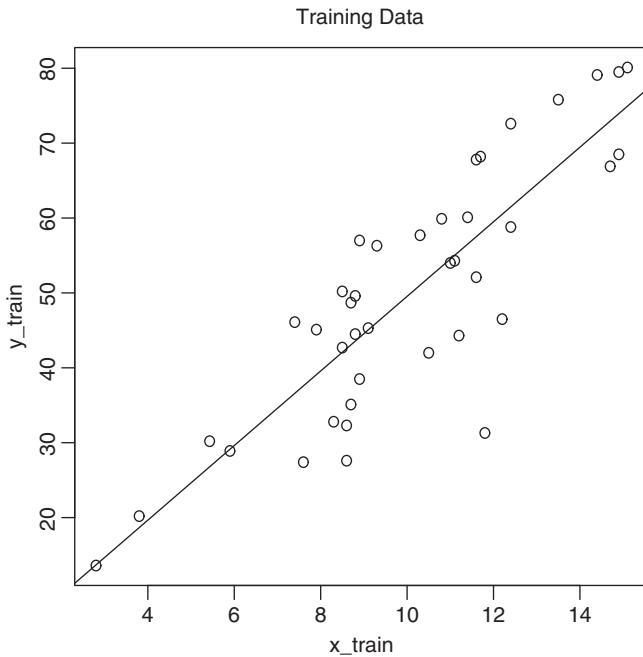


FIGURE 3.18 The Line of Best Fit for the Training Data

Figure 3.19 shows the observed values, y_{test} , along with the values estimated from the line, y_{est} . The vertical lines illustrate the differences between them. A decision has to be made then as to whether or not the line is a “good fit” or whether an alternative model should be investigated. \triangleleft

The *line of best fit* is the simplest regression model; it uses just one independent variable for prediction. In real-life situations, many more independent variables or other models, such as, for example a quadratic, may be required, but for supervised learning, the approach is always the same:

- Determine if there is a relationship between the dependent variable and the independent variables;
- Fit the model to the training data;
- Test the suitability of the model by predicting the y -values in the testing data from the model and by comparing the observed and predicted y -values.

The predictions from these models assumes that the trend, based on the data analyzed, continues to exist. Should the trend change, for example, when a house pricing model is estimated from data before an economic crash, the predictions will not be valid.

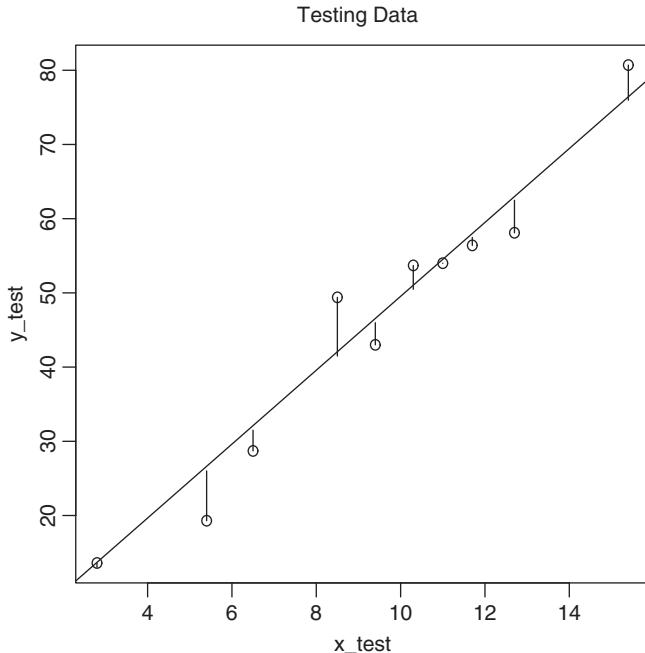


FIGURE 3.19 Differences Between Observed and Estimated y Values in the Testing Set

Regression analysis is just one of the many techniques from the area of Probability and Statistics that machine learning invokes. We will encounter more in later chapters. Should you wish to go into this topic more deeply, we recommend the book, *A First Course in Machine Learning* by Girolami (2015).

3.7 GRAPHICAL DISPLAYS VERSUS SUMMARY STATISTICS

Before we finish, let us look at a simple, classic example of the importance of using graphical displays to provide insight into the data. The example is that of Anscombe (1973), who provides four data sets, given in Table 3.3 and often referred to as the *Anscombe Quartet*. Each data set consists of two variables on which there are 11 observations.

First, read the data into separate vectors.

```
x1 <- c(10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5)
y1 <- c(8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26,
      10.84, 4.82, 5.68)
```

TABLE 3.3 The Anscombe Quartet

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

and so on for x_2 , y_2 , x_3 , y_3 , x_4 , and y_4 . Then, for convenience, group the data into data frames as follows:

```
dataset1 <- data.frame(x1,y1)
dataset2 <- data.frame(x2,y2)
dataset3 <- data.frame(x3,y3)
dataset4 <- data.frame(x4,y4)
```

When presented with data such as these, it is usual to obtain summary statistics. Let us do this using *R*.

To obtain the means of the variables in each data set, write

```
mean(dataset1)
      x1      y1
9.000000 7.500909

mean(dataset2)
      x2      y2
9.000000 7.497273

mean(dataset3)
      x3      y3
9.0 7.5

mean(dataset4)
      x4      y4
9.000000 7.500909
```

The means for the x variables, as you can see, are practically identical as are the means for the y variables.

Let us look at the standard deviations.

```
sd(dataset1)
  x1      y1
3.316625 2.031568

sd(dataset2)
  x2      y2
3.316625 2.028463

sd(dataset3)
  x3      y3
3.316625 2.030424

sd(dataset4)
  x4      y4
3.316625 2.030579
```

The standard deviations, as you can see, are also practically identical for the four x variables, and also for the y variables.

Calculating the mean and standard deviation is the usual way to summarize data. With these data, if this was all that we did, we would conclude naively that the four data sets are “equivalent,” since that is what the statistics say. But what do the statistics not say?

Investigating further, using graphical displays, gives a different picture. Pairwise plots would be the obvious exploratory technique to use with paired data.

```
par(mfrow = c(2, 2))
plot(x1,y1, xlim = c(0, 20), ylim = c(0, 13))
plot(x2,y2, xlim = c(0, 20), ylim = c(0, 13))
plot(x3,y3, xlim = c(0, 20), ylim = c(0, 13))
plot(x4,y4, xlim = c(0, 20), ylim = c(0, 13))
```

gives Fig. 3.20. Notice again the use of `xlim` and `ylim` to ensure that the scales on the axes are the same in the four plots, in order that a valid comparison can be made.

Examining Fig. 3.20, we see that there are very great differences in the data sets:

1. Data set 1 is linear with some scatter;
2. Data set 2 is quadratic;
3. Data set 3 has an outlier. If the outlier were removed the data would be linear;

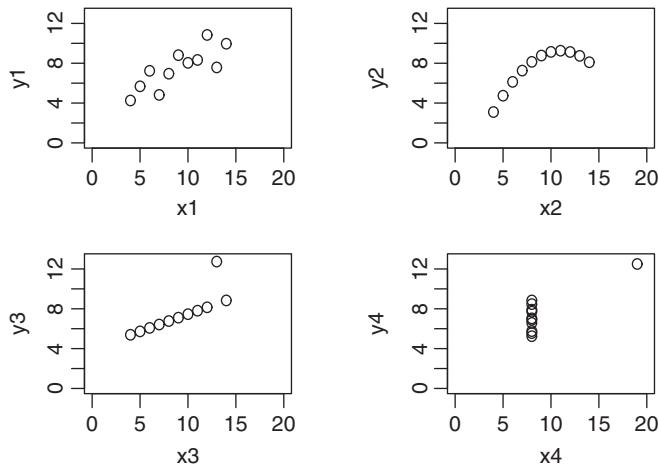


FIGURE 3.20 Plots of Four Data Sets with Same Means and Standard Deviations

4. Data set 4 contains x values that are equal except for one outlier. If the outlier were removed, the data would be vertical.

Graphical displays are the core of getting “insight/feel” for the data. Such “insight/feel” does not come from the quantitative statistics; on the contrary, calculations of quantitative statistics should come after the exploratory data analysis using graphical displays.

EXERCISES 3.1

1. Use the data in “results.txt” to develop boxplots of all the subjects on the same graph.
2. Obtain a stem and leaf of each subject in “results.txt.” Are there patterns emerging?
3. For the class of 50 students of computing detailed in Exercise 1.1, use *R* to
 - (a) form the stem-and-leaf display for each gender, and discuss the advantages of this representation compared to the traditional histogram;
 - (b) construct a box-plot for each gender and discuss the findings.
4. Plot the marks in Architecture 1 against those in Architecture 2 and obtain the line of best fit. In your opinion, is it a suitable model for predicting the results obtained in Architecture 2 from those obtained in Architecture 1?
5. The following table gives the number of hours spent studying for the probability examination and the result obtained (%) by each of 10 students.

Study hours	5	4	8	7	10	6	10	4	0	0
Exam results	73	64	80	70	85	50	86	50	20	25

Plot the data and decide if there is a linear trend. If there is, use *R* to obtain the line of best fit.

6. The percentage of households with access to the Internet in Ireland in each of the years 2010–2017 is given in the following table:

Year	2010	2011	2012	2013	2014	2015	2016	2017
Internet access	72	78	81	82	82	85	87	89

This set of data is to be used as a training set to estimate Internet access in the future.

- (a) Plot the data and decide if there is a linear trend.
- (b) If there is, obtain the line of best fit.
- (c) Can you predict what the Internet access will be in 2019?

3.8 PROJECTS

1. In Appendix B, we show that the line of best fit $y = a + bx$ is obtained when

$$b = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

and

$$a = \bar{y} - b\bar{x}$$

Write a program in *R* to calculate a and b and use it to obtain the line that best fits the data in Exercise 5 above. Check your results using the `lm(y~x)` function given in *R*.

2. When plotting in Fig. 3.10, we used `font.main = 1` to ensure the main titles are in plain font.

Alternative fonts available are

2 = bold,

3 = italic,

4 = bold italic

5 = symbol.

Fonts may also be changed on the x - and y -axis labels, with `font.lab`. Explore the effect of changing the fonts in Fig. 3.7.

REFERENCES

- Anscombe, F.J. (1973), Graphs in statistical analysis, *American Statistician*, 27, 1721.
Girolami, M. (2015), *A First Course in Machine Learning*, CRC Press.

PART II

FUNDAMENTALS OF PROBABILITY

4

PROBABILITY BASICS

“Probably not,” “they probably will,” “in all probability,” etc.: in the English language, we think we know what we mean when we say something will *probably* happen. Mathematical “probability” is something a bit more technical than this intuitive feeling.

Probability evolved in the sixteenth and seventeenth centuries from gambling and games of chance. Tossing a “fair” die, and pulling a card from a “well-shuffled deck,” would have been the “experiments” that originally led to the formal theory of probability. Of course, probability today has developed way beyond this and has become relevant in a wide range of areas.

The relevance of probability to computing stems from the fact that many computing problems contain elements of uncertainty or randomness. For example, the results of experiments to measure, say, algorithm efficiency, CPU (central processing unit) time or response time, may depend on a number of factors including, how busy the computer is at the time of running the program, how many other jobs are running, how many computers are on the network, and these factors vary from experiment to experiment.

In this chapter, we outline the basic concepts of probability, and introduce the functions available in *R* for examining them.

4.1 EXPERIMENTS, SAMPLE SPACES, AND EVENTS

Any activity for which the outcome is uncertain can be thought of as an “experiment.” An experiment could be as simple as pulling a card from a deck to observe what card occurs, or tossing a coin to observe which face turns up, or it could consist of running a computer program to observe how long it takes to execute.

The set of all possible outcomes of an experiment is called the “sample space,” usually denoted by S .

An “event,” denoted by E , is a subset of S . Any possible subset of the sample space S constitutes an “event” E . Thus, the sample space S itself is technically an event, as is the empty subset \emptyset , which has no points at all. Those subsets $\{x\}$ of S , which consist of single points $x \in S$, can be classified as “outcomes,” so that every possible event is just a set of possible outcomes.

Example 4.1

When tossing a coin, there are two possible outcomes, “head” or “tail,” and we choose to be interested in getting a head.

Experiment: Toss a coin

Sample space: $S = \{H, T\}$

Event: $E = \{H\}$

△

Example 4.2

Alternatively, we can toss two coins and look for at least one head.

Experiment: Toss two coins

Sample space: $S = \{(H, H), (H, T), (T, H), (T, T)\}$

Event: $E = \{(H, T), (T, H), (H, H)\}$

△

Example 4.3

We can draw a card from a pack and look for a spade.

Experiment: Draw a card from a deck

Sample space: $S = \{\spadesuit A, \spadesuit 2, \dots, \spadesuit K, \heartsuit A, \dots, \heartsuit 10, \clubsuit A, \dots, \clubsuit K\}$ (the usual 52 cards)

Event: $E = \{\spadesuit A, \spadesuit 2, \dots, \spadesuit K\}$ (the 13 spades)

△

Example 4.4

We can roll a die and look for an even number.

Experiment: Roll a die

Sample space: $S = \{1, 2, 3, 4, 5, 6\}$ (the six faces)

Event: $E = \{2, 4, 6\}$

△

Example 4.5

We can roll two dice and look for a double.

Experiment: Roll two dice

Sample space: $S = \{(1, 1), (1, 2), \dots, (1, 6), (2, 1), \dots, (6, 6)\}$

Event: $E = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6)\}$

△

Example 4.6

We can run a computer program and see whether it compiles.

Experiment: Run a computer program

Sample space: $S = \{\text{compiles, does not compile}\}$

Event: $E = \{\text{compiles}\}$

△

Example 4.7

We can carry out a web search to find a particular record (keyword).

Experiment: Search the web for a particular record

Sample space: $S = \{\text{record found, record not found}\}$

Event: $E = \{\text{record found}\}$

△

Example 4.8

We can inspect integrated circuit (IC) chips from a manufacturing process repeatedly

Experiment: Inspect chips from a manufacturing process until the first defective is discovered

Sample space: $S = \{d, gd, ggd, gggd, \dots\}$: here d denotes a defective chip in the first inspection, gd denotes a good chip followed by a defective chip, and so on.

Event: $E = \{d, gd, ggd\}$ (a defective chip occurs in the first three inspections)

△

Example 4.9

The “response time” of an inquiry is the time that elapses from the instant the inquiry is submitted until the response has been received and displayed on the screen.

Experiment: Type the inquiry

Sample space: $S = \mathbb{R}^+ = \{t \in \mathbb{R} | t > 0\}$ (the response time in milliseconds). Here, \mathbb{R} represents the usual “number line,” and therefore \mathbb{R}^+ represents the positive number line.

Event: $E = \{t \in \mathbb{R} | 0 < t \leq 1\}$ (the response time is less than or equal to a millisecond)

△

Example 4.10

We can again run a computer program, this time to measure its “CPU time,” which is the amount of time the program takes to be processed in the CPU.

Experiment: Run a program to measure its CPU time

Sample space: $S = \mathbb{R}^+ = \{t \in \mathbb{R} | t > 0\}$ (CPU time in milliseconds)

Event: $E = \{t \in \mathbb{R} | t > 5\}$ (CPU time exceeds five milliseconds) \triangleleft

Example 4.11

Browse the web requesting a particular item of information, and measure how long it takes to respond to your request.

Experiment: Browse the web requesting a particular item of information

Sample space: $S = \mathbb{R}^+ = \{t \in \mathbb{R} | t > 0\}$ (time in milliseconds)

Event: $E = \{t \in \mathbb{R} | t \leq 200\}$ (time taken to respond is no more than 200 milliseconds) \triangleleft

4.1.1 Types of Sample Spaces

The sample spaces that we encountered above fall into three categories, *finite*, *infinite discrete*, and *continuous*.

- “Finite” sample spaces are exactly what the name suggests: there is only a finite number of possible outcomes.
- “Infinite discrete” sample spaces will be countably infinite, like the positive integers, or the integers.
- “Continuous” sample spaces will correspond to finite or infinite intervals of real numbers.

Examples 4.1–4.7 have finite sample spaces.

Example 4.8 is infinite discrete.

The sample spaces in Examples 4.9–4.11 are continuous.

EXERCISES 4.1

1. Four bits are transmitted over a digital communication channel. Each bit is either distorted or received without distortion. List the elements of the sample space S , and the event of interest E that at most one bit will be distorted.
2. An order for a computer system can specify memory of 16, 64, 128 gigabytes, and disk storage of one or two terabytes. Describe the set of all designs.
3. In the design of a website, you are allowed to use one of three colors, black, red, and white, and one of two fonts, normal or italic. In addition, an image may be put in one of three positions. How many designs are possible?

4. Computer chips coming off an assembly line are tested for quality, and are rated defective (d) or good (g). A quality control inspection, carried out every hour, tests the chips until two consecutive chips are defective or until four chips have been tested, whichever occurs first. List the elements of the sample space S for this experiment, and the event E that four chips will be inspected.
5. A student can repeat an examination until it is passed, but is allowed to attempt the examination at most four times. List the elements of the sample space S , and the event E that the student passes.

4.2 CLASSICAL APPROACH TO PROBABILITY

There is a simple approach to probability for finite sample spaces: count the number $\mathbf{n}(E)$ of outcomes favorable to an event E and divide it by the total number of outcomes.

Definition 4.1 *Classical Probability*

$$P(E) = \frac{\text{number of points in } E}{\text{number of points in } S} = \frac{\mathbf{n}(E)}{\mathbf{n}(S)}$$

□

With this definition of probability, it is assumed that each individual outcome $\{x\}$ has exactly the same probability.

$$x \in S \implies P(\{x\}) = \frac{1}{\mathbf{n}(S)}$$

This is, of course, what we mean when we describe a coin or a die as “fair.”

In Example 4.1, tossing a coin, with $E = \{H\}$, Definition 4.1 gives

$$P(E) = \frac{1}{2}$$

In Example 4.2, tossing two coins, we remember $S = \{(H, H), (H, T), (T, H), (T, T)\}$, so that $\mathbf{n}(S) = 4$. If the event of interest $E = \{(H, T), (T, H), (H, H)\}$ is to get at least one head, then $\mathbf{n}(E) = 3$. Now our definition gives

$$P(E) = \frac{3}{4}$$

In Example 4.3, pulling a card from a deck, $\mathbf{n}(S) = 52$, if the event of interest is a spade, then $\mathbf{n}(E) = 13$ giving

$$P(E) = \frac{13}{52} = \frac{1}{4}$$

In Example 4.4, there are in all six possible outcomes of the experiment of rolling a die, $S = \{1, 2, 3, 4, 5, 6\}$, of which three are favorable to the event $E = \{2, 4, 6\}$ of getting an even number. Our definition gives

$$P(E) = \frac{3}{6} = \frac{1}{2}$$

In Example 4.5, there are 36 possible outcomes of the experiment of rolling two dice, six of which are “doubles.”

$$P(E) = \frac{6}{36} = \frac{1}{6}$$

Definition 4.1 assumes that each outcome is equally likely, so that the coin or the die is “fair.” You would be entitled to wonder, when embarking on a coin-tossing or card-drawing experiment, how you would be supposed to know that the coin was fair, or the pack well-shuffled, and therefore how you could trust this definition. We will return to this issue later in the chapter.

Example 4.12

A group of four IC chips consists of two good chips and two defective chips. The experiment consists of selecting three chips at random from this group. Write down the sample space. What is the probability that two are defective?

A natural sample space for this problem consists of all possible three-chip selections from the group of four chips:

$$S = \{\{g_1, g_2, d_1\}, \{g_1, g_2, d_2\}, \{g_1, d_1, d_2\}, \{g_2, d_1, d_2\}\}$$

where g denotes a good chip and d denotes a defective chip. The sample space S consists of four points, or possible outcomes $\mathbf{n}(S) = 4$.

We interpret the phrase “selected at random” as implying that every point in the sample space is equally likely, and therefore, we can use our classical formula given in Definition 4.1.

Let E be the event that two of the three selected chips are defective.

$$E = \{\{g_1, d_1, d_2\}, \{g_2, d_1, d_2\}\}$$

Since E contains two sample points, that is, $\mathbf{n}(E) = 2$, we calculate

$$P(E) = \frac{\mathbf{n}(E)}{\mathbf{n}(S)} = \frac{2}{4} = \frac{1}{2}$$

□

Example 4.13

Consider the following **if** statement in a program:

if B **then** s_1 **else** s_2

The experiment consists of “observing” two successive executions of this **if** statement. When this program is executed twice, there are four possible outcomes.

$$S = \{(s_1, s_1), (s_1, s_2), (s_2, s_1), (s_2, s_2)\}$$

Once again, we assume that all these possibilities are equally likely.

Suppose E is the event that there is at least one execution of the statement s_1 .

$$E = \{(s_1, s_1), (s_1, s_2), (s_2, s_1)\}$$

So, in this case, $\mathbf{n}(E) = 3$ and $\mathbf{n}(S) = 4$. Hence,

$$P(E) = \frac{3}{4}$$

Suppose instead, that we are interested in the event that the statement s_2 is executed first. Then

$$E = \{s_2 \text{ first}\} = \{(s_2, s_1), (s_2, s_2)\}$$

so that $\mathbf{n}(E) = 2$. Hence,

$$P(E) = \frac{2}{4} = \frac{1}{2}$$

△

Example 4.14

In a software development company, 200 new programs are written each week, 120 in C++ and 80 in Java. Sixty percent of the programs written in C++ and 80% of the Java programs compile on the first run. The actual numbers are given in Table 4.1.

A program is chosen at random from the week’s supply. What is the probability that the program compiles on the first run?

The sample space S consists of programs written each week, that is, $\mathbf{n}(S) = 200$. Let E be the event that a program, selected at random, compiles on the first run. There are 136 programs that compile on the first run, so $\mathbf{n}(E) = 136$. Hence,

$$P(E) = \frac{136}{200} = 0.68$$

TABLE 4.1 Programs

	Compiles on First Run	Does Not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

What this means is that there is a 68% chance that a program, selected at random from the week's supply, will be one that compiles on the first run.

Suppose instead that we are interested in the probability that a program selected at random from the week's supply is written in C++ and compiles on the first run.

The event E now consists of those programs that are written in C++ and compile on the first run. Reading from Table 4.1, we see that the number of programs in E is 72, that is, $n(E) = 72$. The sample space S is the same as before, that is, $n(S) = 200$, giving

$$P(E) = \frac{72}{200} = 0.36$$

which means that if a program is selected at random from the week's supply, there is a 36% chance that it will have been written in C++ and compiles on the first run.

We could ask what the probability is that a program, selected at random, is written in Java and does not compile on the first run.

This time E consists of all those programs that are written in Java that did not compile on the first run. Reading from Table 4.1, we see that the number of such programs is 16, that is, $n(E) = 16$. Thus,

$$P(E) = \frac{16}{200} = 0.08$$

△

4.3 PERMUTATIONS AND COMBINATIONS

In the classical approach to the probability $P(E)$ of an event E , we have to count the outcomes in the sample space S , and count the outcomes of E , and divide one into the other. For large sample spaces, these counts may be a formidable undertaking. Fortunately, there are certain standard techniques from combinatorial analysis that can come to our aid. The most famous of these are known as “permutations” and “combinations,” and they provide concise methods of counting the number in S and E without having to write down the outcomes explicitly. Let us investigate.

4.3.1 Permutations

Example 4.15

If passwords can consist of three letters, find the probability that a randomly chosen password will not have any repeated letters.

Let $A = \{a, b, \dots, z\}$ be the usual alphabet of 26 letters.

The sample space is all possible sets of three letters including repeats and is given by

$$S = A^3 = \{(\alpha, \beta, \gamma) : \{\alpha, \beta, \gamma\} \subseteq A\}$$

To count the number of possible passwords (α, β, γ) , notice again that there are 26 possible choices for α , and that for each α , there are 26 possible choices of (α, β) ,

and then finally for each (α, β) there are 26 possible choices of (α, β, γ) giving in total $\mathbf{n}(S) = 26^3 = 17,576$ possible passwords.

The event of interest is

$$E = \{(\alpha, \beta, \gamma) : \alpha \neq \beta \neq \gamma \neq \alpha\}$$

To count the number of passwords (α, β, γ) in E , notice that there are again 26 choices for α , and now for each α only 25 possible choices of (α, β) , and then finally for each (α, β) there are only 24 possible choices of (α, β, γ) : this gives in total $\mathbf{n}(E) = 26 \times 25 \times 24 = 15,600$. Hence,

$$P(E) = \frac{26 \times 25 \times 24}{26 \times 26 \times 26} = \frac{15,600}{17,576}$$

We could calculate this probability in R using

```
prod(26:24)/26^3
0.887574
```

There is therefore almost an 89% chance that three-letter passwords will consist of distinct letters.

As you may have noticed, the above example is unrealistic, in that passwords rarely consist of so few letters. Let us take an example with a more realistic number of letters. \triangleleft

Example 4.16

If passwords can consist of six letters, find the probability that a randomly chosen password will not have any repeated letters.

Again, the event of interest E is that a password consisting of six letters will not have any repeated letters. In this case, there are 26^6 possible passwords, and the number of these without repeated letters is $26 \times 25 \times 24 \times 23 \times 22 \times 21$. Therefore,

$$P(E) = \frac{26 \times 25 \times 24 \times 23 \times 22 \times 21}{26^6}$$

Calculating the probability in R

```
prod(26:21)/26^6
0.5366045
```

which is the probability that six-letter passwords will consist of distinct letters. \triangleleft

We used `prod(26:24)` to calculate the product $26 \times 25 \times 24$, and `prod(26:21)` to calculate the product $26 \times 25 \times 24 \times 23 \times 22 \times 21$. These products belong to a famous family, usually written $P(n, k) = {}^n P_k$. With $n = 26$

and $k = 3$, ${}^{26}P_3$ is described as “the number of permutations of 3 out of 26,” and ${}^{26}P_6$ is described as “the number of permutations of 6 out of 26.”

Definition 4.2 *Permutations*

Permutations are ordered samples, or sequences of a particular size that can be chosen, without replacement, from a population. \square

For example,

1. If the set is of size 3, we may write the points as $\{1, 2, 3\}$. The possible sequences of size 2 are
 $(1, 2); (1, 3); (2, 1); (2, 3); (3, 1); (3, 2)$
that is, six ordered sequences altogether.
2. With four elements $\{1, 2, 3, 4\}$, there are 12 possible sequences of size 2.
 $(1, 2); (1, 3); (1, 4); (2, 1); (2, 3); (2, 4)$
 $(3, 1); (3, 2); (3, 4); (4, 1); (4, 2); (4, 3)$
3. With four elements $\{1, 2, 3, 4\}$, if we were to choose three at random and without replacement, there are 24 possibilities.
 $(1, 2, 3); (1, 3, 2); (1, 2, 4); (1, 4, 2); (1, 3, 4); (1, 4, 3)$
 $(2, 1, 3); (2, 3, 1); (2, 1, 4); (2, 4, 1); (2, 3, 4); (2, 4, 3)$
 $(3, 1, 2); (3, 2, 1); (3, 2, 4); (3, 4, 2); (3, 1, 4); (3, 4, 1)$
 $(4, 1, 3); (4, 3, 1); (4, 1, 2); (4, 2, 1); (4, 3, 2); (4, 2, 3)$
that is, there are 24 ordered sequences.

Generally, the number of ways of choosing ordered samples of size k from n is given by

$$P(n, k) = {}^n P_k = n(n - 1) \cdots (n - (k - 1))$$

The calculation is the same as for 3 out of 26 and 6 out of 26 considered in the previous examples. There are n possible choices of the first element of the sequence, $n - 1$ possible choices of the second, and finally $n - k + 1$ choices of the last; multiply together for the total number of choices.

${}^n P_k$ can be calculated easily in R using

```
prod(n:n-(k-1)) ≡ prod(n:n-k+1)
```

1. ${}^3 P_2 = 3 \times 2 = 6$

In R

```
prod(3:2)
[1] 6
```

2. ${}^4 P_2 = 4 \times 3 = 12$

In R

```
prod (4:3)
[1] 12
```

3. ${}^4P_3 = 4 \times 3 \times 2 = 24$

In R

```
prod (4:2)
[1] 24
```

Example 4.17

1. Passwords consist of six characters from the 10 digits and the 26 letters of the alphabet, and they are case sensitive.
 - (a) How many different passwords are possible?
 - (b) What is the probability that a password, chosen at random, consists of different characters?
 - (c) Using password cracking tools, it is claimed that passwords can be tested at the rate of a hundred million passwords in one second. How long would it take to try all of the passwords?
2. Suppose a valid password must begin with a letter, and contain at least one digit; how many passwords are now possible, and how long would it take to try all of the passwords?

Solution

1. (a) There are 10 digits, 26 lower-case letters, and 26 upper-case letters giving a total of 62 characters from which to choose. The number of six-character passwords is $62^6 = 56,800,235,584$.
- (b) The proportion of passwords consisting of six different characters is

```
prod(62:57)/62^6
[1] 0.7792512
```

which means that nearly 78% of six-character passwords, chosen from the 10 digits, 26 upper-case, and 26 lower-case letters consist of different characters.

- (c) In (a) we calculated the number of six-character passwords to be 56,800,235,584. If the cracking tool can check 100 million a second, then it will take just over 568 seconds to check all the passwords, 10 minutes approximately.
2. In order to calculate the number of times at least one digit occurs in the last five characters of the password, it is easier to get the exact opposite, that is, the number of times there is no digit in the last five characters, which is 52^5 . Now, the total number of possibilities for the last five characters is 62^5 . Therefore,

the total number of passwords containing at least one digit in the last five characters is $62^5 - 52^5$.

Since, the first character must be a letter, there are 52 possibilities. Therefore, the number of valid passwords is $52(62^5 - 52^5)$, which can be calculated in R as

```
52 * (62^5 - 52^5)
[1] 27868297600
```

just under 28,000 million.

If the cracking tool can check 100 million a second, then it will take less than 280 seconds to check all the passwords, less than 5 minutes.

The fact that password cracking tools can operate so fast to crack your password illustrates why, in most systems, a limit is put on the number of passwords (usually three) that you are allowed to try to gain access to a system. \triangleleft

4.3.2 Combinations

The difference between “combinations” and “permutations” lies in the fact that, with combinations, we neglect the ordering of the selected elements. If we return to the passwords, reinterpreted as a kind of lottery game, the game is to choose, in any order, three distinct letters out of 26.

Example 4.18

In how many ways can we select three distinct letters $\{\alpha, \beta, \gamma\}$ from the usual alphabet $A = \{a, b, \dots, z\}$?

The calculation rides on the back of the earlier “permutation” count. There are, as we have seen, $26 \times 25 \times 24$ ordered passwords (α, β, γ) . Now, any unordered set $\{\alpha, \beta, \gamma\}$ can, by the same calculation, be ordered in $3 \times 2 \times 1 = 6$ different ways: each unordered selection comes from exactly six different ordered selections.

Taking the three letters a, b, c, the ordered sets are (a, b, c), (a, c, b), (b, a, c), (b, c, a), (c, a, b), (c, b, a), six in all. However, if we ignore the order, there is just one combination.

The total number of unordered selections is therefore,

$${}^{26}C_3 = \frac{{}^{26}P_3}{{}^3P_3} = \frac{26 \times 25 \times 24}{3 \times 2 \times 1} = 2600$$

\triangleleft

Generally,

$${}^nC_k = \frac{{}^nP_k}{{}^kP_k}$$

The number kP_k is also famous in its own right:

$${}^kP_k = k! = k \times (k-1) \times \cdots \times 1$$

which is known as “ k factorial,” or “factorial k .” With this notation we can write

$${}^n P_k = \frac{n!}{(n-k)!} \quad ; \quad {}^n C_k = \frac{n!}{(n-k)!k!}$$

${}^n C_k$ is also denoted by $\binom{n}{k}$ or $C(n, k)$ and known as the *binomial coefficient*:

$$(a+b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} = \sum_{k=0}^n {}^n C_k a^k b^{n-k}$$

It is always curious when one single formula appears in two different places! It will reoccur in later chapters.

Definition 4.3 Combinations

Combinations are unordered sets of a particular size that can be chosen, without replacement, from a population. \square

For example,

1. If a set is of size $n = 3$, we may write the elements as $\{1, 2, 3\}$.

Recall that, for permutations (ordered sequences), there are, in all, 6 of size 2:

$(1, 2); (1, 3); (2, 1); (2, 3); (3, 1); (3, 2)$

For combinations $(1, 2)$ and $(2, 1)$ are equivalent, as are $(1, 3)$ and $(3, 1)$, and also $(2, 3)$ and $(3, 2)$.

Therefore, the number of unordered sets of size 2 is 3: $\{1, 2\}; \{1, 3\}; \{2, 3\}$

2. For four elements $\{1, 2, 3, 4\}$, the combinations of two elements from four are $\{1, 2\}; \{1, 3\}; \{1, 4\}; \{2, 3\}; \{2, 4\}; \{3, 4\}$
that is, there are six unordered sets altogether.
3. The combinations of three elements from a total of four $\{1, 2, 3, 4\}$ are $\{1, 2, 3\}; \{1, 2, 4\}; \{2, 3, 4\}; \{3, 1, 4\}$
that is, just four unordered sets.

In R, the command `choose (n, k)` calculates the number of combinations of k in n .

$$1. \quad {}^3 C_2 = \frac{{}^3 P_2}{2!} = \frac{3 \times 2}{2 \times 1} = 3$$

In R

```
choose(3, 2)
[1] 3
```

$$2. \quad {}^4 C_2 = \frac{{}^4 P_2}{2!} = \frac{4 \times 3}{2 \times 1} = 6$$

In R

```
choose(4, 2)
```

```
[1] 6
```

$$3. {}^4C_3 = \frac{{}^4P_3}{3!} = \frac{4 \times 3 \times 2}{3 \times 2 \times 1} = 4$$

In R

```
choose(4, 3)
```

```
[1] 4
```

Example 4.19

If a box contains 75 good IC chips and 25 defective chips from which 12 chips are selected at random, find the probability that all selected chips are good.

Here, the sample space S consists of all possible samples of 12 that can be drawn without replacement from the total of 100 in the box.

E is the event that all the chips are good. The points in E are those samples of size 12 that can be chosen from the 75 good chips. Clearly, it would be near impossible to list and count S and E . Instead, we use combinations.

The number of possible samples of 12 chips that can be drawn without replacement from 100 is $\binom{100}{12}$, that is, $n(S) = \binom{100}{12}$.

With R we calculate $n(S)$.

```
choose(100, 12)
```

```
1.050421e+15
```

Therefore, $n(S) = 1.050421e + 15 = 1.050421 \times 10^{15}$. As you can see, the number of samples in S is very large indeed. We would not have been able to count them without the aid of these counting techniques.

Let us look now at the event of interest.

E = “All chips are good”

Since there are 75 good chips, there are $\binom{75}{12}$ possible ways of choosing a sample of 12 good chips at random without replacement from the box.

From R

```
choose(75, 12)
```

```
[1] 2.612389e+13
```

$n(E) = 2.612389e + 13 \equiv 2.612389 \times 10^{13}$, again a very large number.

The probability of getting 12 good chips in a sample of 12, drawn at random without replacement from the box containing 100, is

$$P(E) = \frac{\binom{75}{12}}{\binom{100}{12}}$$

$P(E)$ can be calculated directly from R with

```
choose(75,12) / choose(100,12)
[1] 0.02486992
```

So, if a sample of 12 is drawn at random from a box containing 100 chips of which 75 are good and the other 25 are defective, there is less than a 2.5% chance that every one of the 12 would be good. \triangleleft

4.4 THE BIRTHDAY PROBLEM

Example 4.20 *The Birthday Paradox*

The so-called “birthday problem” or “birthday paradox” asks,

What is the probability that at least two people in a room have the same birthday?

Here, we mean that they will celebrate their birthday on the same day of the year, without necessarily being the same age. To simplify the discussion, we will pretend that there is no such thing as a leap year, so that there are 365 days from which to choose.

We might also ask,

How large should the class be so that the chance of at least two students having the same birthday is greater than the probability that all have different birthdays?

Let us investigate.

The event of interest is

$$E = \text{“At least two students have the same birthday”}$$

Here, it is easier to work with the complementary event \bar{E} , which is the event that E does not occur.

$$\bar{E} = \text{“all the students have different birthdays”}$$

The number of ways that k students have different birthdays is the number of ways of choosing k different days from 365, that is, choose k without replacement from 365, which is ${}^{365}P_k$. This means that

$$\mathbf{n}(\bar{E}) = {}^{365}P_k$$

which can be calculated in R with

```
prod(365:365-k+1)
```

Now, for the sample space S , we note that each student can have a birthday on any of the 365 days. So the number of possible ways than these k students can have birthdays is 365^k , that is,

$$n(S) = 365^k$$

It follows that the probability that all the k students in the class have different birthdays is

$$P(\text{All different birthdays}) = \frac{^{365}P_k}{365^k}$$

and the probability of the occurrence of the exact opposite event, that is, that at least two will have the same birthday is

$$P(\text{At least 2 birthdays are the same}) = 1 - \frac{^{365}P_k}{365^k}$$

We can calculate this probability in R for various values of k .

```

k <- 2 #class size = 2
prod(365:(365-k+1))/365^k  #all different
[1] 0.9972603
1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.002739726

k <- 10 #class size = 10
prod(365:(365-k+1))/365^k  #all different
[1] 0.8830518
1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.1169482

k <- 20 #class size = 20
prod(365:(365-k+1))/365^k  #all different
[1] 0.5885616
1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.4114384

k <- 30 #class size = 30
prod(365:(365-k+1))/365^k  #all different
[1] 0.2936838
1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.7063162

k <- 40 #class size = 40
prod(365:(365-k+1))/365^k  #all different
[1] 0.1087682

```

```

1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.8912318

k <- 50 #class size = 50
prod(365:(365-k+1))/365^k  #all different
[1] 0.02962642
1 - prod(365:(365-k+1))/365^k  #at least 2 the same
[1] 0.9703736

```

Recall that the `#` allows for a comment. Anything written after `#` is ignored by *R*. These results are summarized in Table 4.2.

We can see from Table 4.2 that the probability of at least two students having the same birthday increases rapidly with the class size. It is practically certain that, in a class size of 50, at least two students will have the same birthday, and there is nearly a 50:50 chance with a class size of 20. The crossover point, where the probability that two students have the same birthday is greater than the probability that all are different, turns out to be $k = 23$, rather smaller than what most people would expect. \triangleleft

The following *R* code calculates and plots the probability that at least two of the k students have the same birthday for values of k ranging from 2 to 50.

4.4.1 A Program to Calculate the Birthday Probabilities

```

diffbirthday <- 1
for (k in 2:50) {
  diffbirthday[k] <- diffbirthday[k-1] * (365 - (k-1)) / 365
}
samebirthday <- 1 - diffbirthday
k <- seq(1, 50, 1) #creates a vector containing
# integers from 1 to 50
plot (k, samebirthday, main = "Birthday Paradox",
      xlab = "Number in room",
      ylab = "Probability of at least 2 with same
              birthday")
text(23, 0.5, "Number = 23,     Probability > 0.5")

```

From this, we obtain Fig. 4.1.

Note that `text(23, 0.5, "Number = 23, Probability > 0.5")` writes the text centered at the point $x = 23$ and $y = 0.5$.

TABLE 4.2 Birthday Probabilities

Number of Students in a Class	Probability that All Birthdays are Different	Probability that at Least Two Have the Same Birthday
02	0.9972603	0.0027397
10	0.8830518	0.1169482
20	0.5885616	0.4114384
30	0.2936838	0.7063162
40	0.1087682	0.8912318
50	0.0296264	0.9703736

4.4.2 R Functions for the Birthday Problem

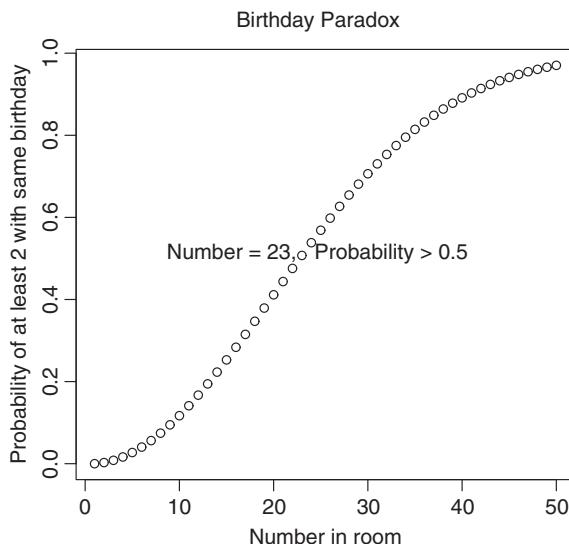
The base package in *R* now contains two functions in *R* to calculate the birthday probabilities.

- The function `pbirthday` computes the probability of birthday coincidences. For example,

```
pbirthday(n = 23, classes = 365, coincident = 2)
```

gives

```
[1] 0.5072972
```

**FIGURE 4.1** The Birthday Probabilities

which is the probability that, in a group of 23, at least 2 will have the same birthday, which we know already from previous deliberations.

In the `pbirthday` function, `classes = 365` and `coincident = 2` are the defaults. So, by simply writing

```
pbirthday(n = 23)
```

we get

```
[1] 0.5072972
```

- The function `qbirthday` calculates the minimum number of people necessary to ensure that there is at least a given probability that at least two of them have the same birthday. For example, to obtain the minimum number of people needed to have at least a 50% chance (`prob = 0.5`) that at least two have the same birthday (`coincident = 2`), assuming that there are 365 days (`classes = 365`) days in the year, write

```
qbirthday(prob = 0.5, classes = 365, coincident = 2)
[1] 23
```

which confirms what we previously derived; a minimum of 23 people ensures that at least two of them have the same birthday with a 50% probability or more. Similar to `pbirthday`, in this case, `classes = 365` and `coincident = 2` are the defaults. By simply writing in *R*

```
qbirthday(prob = 0.5)
```

we get

```
[1] 23
```

By changing the parameters in `pbirthday` and `qbirthday`, we can solve other versions of birthday coincidences. For example,

- `pbirthday(n = 5, classes = 12, coincident = 2)`
`[1] 0.6180556`

is the probability that, with $n = 5$ people, at least two of them will be born in the same month (`classes = 12`).

- `qbirthday(prob = 0.9, classes = 12, coincident = 2)`
`[1] 8`

means that a minimum of eight people are needed to have at least a 90% chance that two of them have a birthday in the same month (`classes = 12`).

- `pbirthday(n = 10, classes = 52, coincident = 2)`
`[1] 0.6028663`

is the probability that, with $n = 10$ people, at least two will be born in the same week (`classes = 52`).

- `qbirthday(prob = 0.9, classes = 52, coincident = 2)`
`[1] 16`

means that, a minimum of 16 people are needed to have at least a 90% chance that two people have a birthday in the same week.

The functions `qbirthday` and `pbirthday` can also calculate the probability of more than two birthday coincidences. For example, we might be interested in the probability that at least three have the same birthday in a class of 100 say. In *R*

`pbirthday(100, classes = 365, coincident = 3)`

gives

`[1] 0.6398217`

which is the probability that, with 100 people, at least three have the same birthday.

`qbirthday(0.5, 365, 3)`
`[1] 88`

which means that a minimum of 88 people is necessary for there to be at least a 50% chance that three will have the same birthday.

4.5 BALLS AND BINS

The birthday problem fits into a more general mathematical framework that is often formulated in terms of “balls and bins.” We throw m balls into n bins and assess the probabilities of the distribution of the balls among the bins. We might seek the probability that two balls will be in any one bin, which is similar to finding the probability that two people have the same birthday discussed in the previous section. There are several other questions we could ask about this process. What is the probability that a bin will be empty? If a bin can contain a maximum of j balls, what is the probability that it will be overloaded?

In the area of computer science, we might examine how m jobs are allocated to n processors.

Example 4.21

A series of 10 jobs arrives at a server farm with 10 processors. Assume that each of the jobs is equally likely to go through any of the processors. Find the probability that all processors are occupied.

In all, there are a total of 10^{10} ways of assigning the 10 jobs to the 10 processors.

$$n(S) = 10^{10}$$

Let E be the event that all the processors are occupied.

In this case, all the 10 jobs will have to go to different processors. As with the birthday problem, the number of ways that each processor receives one job is

$$n(E) = 10! = {}^{10}P_{10}$$

Therefore, the probability that all the jobs go to different processors is

$$P(\text{All different processors}) = \frac{{}^{10}P_{10}}{10^{10}}$$

Calculating this in R gives

```
prod(10:1)/10^10
[1] 0.00036288
```

△

As you can see, there is a very small chance, less than 0.04%, that each processor will receive exactly one job. Consequently, there is a very large probability that the opposite will occur, that is, that at least one processor will receive at least two jobs.

$$\begin{aligned} P(\text{At least two jobs assigned to at least one processor}) &= 1 - 0.00036288 \\ &= 0.9996372 \end{aligned}$$

With almost 100% certainty, at least one processor will have more than one job assigned to it.

Another question we could ask is: How many processors would be needed to be fairly confident that none of the processors will receive more than one job?

We might define “fairly confident” as 90% sure. Then, the problem is to choose n , the number of processors, so that

$$P(\text{All different processors}) = \frac{{}^nP_{10}}{n^{10}} \geq 0.90$$

After some experimentation in R , we have

```
n <- 400
prod(n: (n-9))/n^10
[1] 0.8927923
```

```
n <- 500
prod(n: (n-9))/n^10
[1] 0.9134054
```

You would need over 400 processors to be 90% sure that no processor would receive more than one job!

The following *R* code calculates and plots the probability that the 10 jobs are allocated to 10 different processors for varying numbers of available processors.

```
for (n in (10: 500)) {
  diffprocessor[n] <- prod(n: (n-9))/n^10
  # probability all different processors
}
k <- seq(10, 500, 10) #creates a vector containing
# integers from 10 to 500 in intervals of 10
plot (k, diffprocessor[k], ylim = c(0, 1),
      xlab = "Number of processors available",
      ylab = "Probability that no processor receives
              more than one job")
abline(h = 0.9)
```

The output is given in Fig. 4.2.

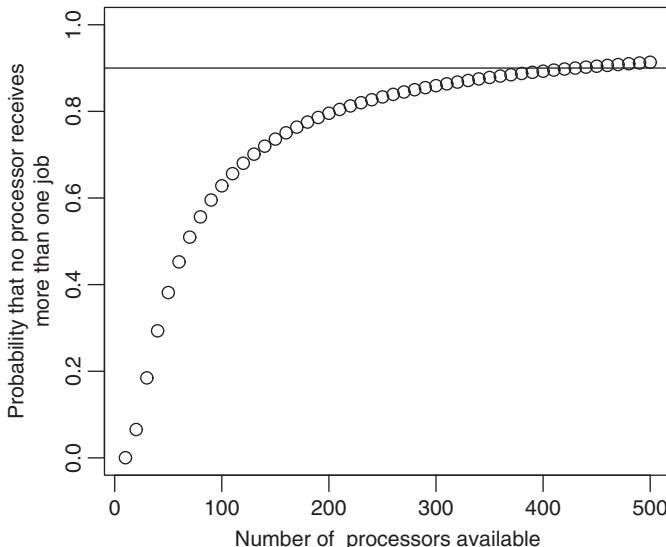


FIGURE 4.2 Allocation of 10 Jobs

The argument `ylim = c(0, 1)` in the plot function ensures that the extremes of the y-axis are 0 and 1. The horizontal line is drawn at the 90% probability point, and it occurs at just more than 400 processors. In fact, it takes a minimum of 431 to exceed this probability.

```
diffprocessor[431]
[1] 0.9001589
```

Notice that, with 100 processors, there is just over a 60% chance that the 10 jobs will be allocated to different processors.

```
diffprocessor[100]
[1] 0.6281565
```

With 500 processors:

```
diffprocessor[500]
[1] 0.9134054
```

which is the probability that the 10 jobs will be allocated to 10 different processors. Then $1 - 0.9134054 = 0.0865946$ is the probability that at least one processor will have two of the jobs allocated to it, that is, that even with 500 available processors, there is still more than an 8% chance that at least one processor will have two of the 10 jobs allocated to it.

4.6 R FUNCTIONS FOR ALLOCATION

We now show that the functions `pbirthday` and `pbirthday` have applications way beyond that of calculating birthday coincidences and can also be used to tackle the general allocation problem in computer science.

Looking again at Example 4.21 where there are 10 jobs to be allocated to 10 processors, we might repeat the calculations using the `pbirthday` function with $n = 10$ and $\text{classes} = 10$.

```
pbirthday(n = 10, classes = 10, coincident = 2)
[1] 0.9996371
```

which is the probability that with 10 jobs, at least two will go into the same processor. Conversely, the probability that all 10 will go into different processors is

```
1 - pbirthday(n = 10, classes = 10, coincident = 2)
[1] 0.00036288.
```

which is what we calculated it to be in Example 4.21

The probability of a coincidence with 400 processors can be calculated with

```
pbirthday(10, 400, 2)
[1] 0.1072077
```

and the probability of a coincidence with 500 processors is

```
pbirthday(10, 500, 2)
[1] 0.0865946
```

similar to what we obtained above.

Example 4.22

A PIN has four digits. Hence, there are 10^4 possible PINS. What are the chances of people choosing the same 4-digit PIN? What is the minimum number of people that will ensure that there is at least a 50% probability that two or more will have the same PIN?

```
qbirthday(prob = 0.5, classes = 10^4, coincident = 2)
[1] 119
```

which means that with 119 people each with a 4-digit PIN, there is at least a 50% chance that two will have the same PIN.

Remembering that `prob = 0.5` and `coincident = 2` are the defaults in the function `qbirthday`, then

```
qbirthday(classes = 10^4)
[1] 119
```

gives the same result. □

Example 4.23

A hash table contains n slots, to which values are assigned by a hash function. A collision is said to occur if more than one value hashes into any particular slot. Suppose a particular table contains 100 slots, and 10 values are inserted, calculate the probability of a collision.

```
pbirthday(n = 10, classes = 100, coincident = 2)
[1] 0.3718435
```

which is the probability that at least two will go into the same slot, that is, the probability of a collision.

Equivalently,

```
qbirthday(prob = 0.37, classes = 100, coincident = 2)
[1] 10
```

□

We might want to know: what is the maximum number of values that can be stored, in order to be at least 90% sure that there will not be a collision? Equivalently, we want that value for which there is at most a 10% chance of a collision.

```
qbirthday(prob = 0.10, classes = 100, coincident = 2)
[1] 6
```

means that with six values stored, there is at least a 10% chance of a collision, and therefore, the probability of no collision is less than 90%. We could also calculate the probability of a collision with

```
pbirthday(n = 6, classes = 100, coincident = 2)
[1] 0.1417223
```

which is the probability of a collision with six slots.

With five slots, the probability of a collision is

```
pbirthday(n = 5, classes = 100, coincident = 2)
[1] 0.09654976
```

So, $n = 5$ is the maximum number of values to be at least 90% sure that there will not be a collision.

4.7 ALLOCATION OVERLOAD

The real power of the functions `pbirthday` and `qbirthday` is their ability to deal with more than two coincidences, and hence to deal with the problems of server overload.

Returning to Example 4.21, suppose that, if any processor gets three or more jobs allocated to it, it becomes overloaded. Calculate

- (a) the probability of overloading.
- (b) the number of processors that would be needed to be 90% certain that there is no overload, if 10 jobs arrived together.

Solution

(a) `pbirthday(n = 10, classes = 10, coincident = 3)`
`[1] 0.5584702`

which is the probability that at least one processor will be overloaded.

- (b) For the probability of an overload to be less than 10%, we need to increase the number of processors. Even if we double the number of processors,

```
pbirthday(n = 10, classes = 20, coincident = 3)
[1] 0.250856
```

there is still more than a 25% chance that at least one processor is overloaded, which means that there is less than a 75% chance that none will be overloaded.

For various numbers of processors, the probabilities of an overload can be calculated empirically as follows:

```
pbirthday(n = 10, classes = 30, coincident = 3)
[1] 0.1347647
```

```
pbirthday(n = 10, classes = 40, coincident = 3)
[1] 0.0828951
```

```
pbirthday(n = 10, classes = 35, coincident = 3)
[1] 0.1042613
```

```
pbirthday(n = 10, classes = 36, coincident = 3)
[1] 0.09938741
```

Therefore, with 10 jobs arriving, it is necessary to have 36 processors to ensure that the probability of an overload is less than 10%. Conversely, with 36 processors, there is a 90% chance of no overload.

We can check this result with qbirthday,

```
qbirthday(0.0993874, classes = 36, coincident = 3)
[1] 10
```

If you want to increase this probability, you will need to control the job arrivals (if possible) or increase the number of processors.

Example 4.24

A web server is considered to be overworked and slows down, if it receives 50 or more requests simultaneously. If 700 requests arrive, and there are 20 servers to which they will randomly be allocated, calculate the probability that any one of these servers will be overloaded. How many servers are necessary to be at least 90% sure that an overload does not occur?

Solution

Write in R

```
pbirthday(n = 700, classes = 20, coincident = 50)
[1] 0.1903118
```

which means that, when 700 requests arrive simultaneously, the probability that at least one server will receive 50 or more of these is over 19%.

To decrease the probability of an overload, we need to increase the number of servers. Suppose we add another server.

```
pbirthday(n = 700, classes = 21, coincident = 50)
[1] 0.08852041
```

So with 21 servers, the probability of an overload is less than 9%, which means that the probability that there will be no overload is greater than 90%. \triangleleft

4.8 RELATIVE FREQUENCY APPROACH TO PROBABILITY

There is an “experimental” approach to calculating probabilities, which can, to a certain extent, validate or otherwise the “classical” approach discussed in Section 4.2. The idea is to repeat over and over again the relevant experiment, and then count how many times an event occurs. Now the “relative frequency” of an event E will in theory get closer and closer to $P(E)$, the “probability of E .”

Formally, if the sample space of an experiment is S , then the sample space for an n -fold repetition of the experiment will be $S^n = S \times S \times \cdots \times S$, and we shall write

$$N_n(E) = \text{the number of times } E \text{ occurs in } n \text{ repetitions}$$

Thus, $N_1(E)$ is either 1 or 0, according as E does or does not occur, $N_2(E)$ can be 0, 1 or 2, and in general

$$N_n(E) \in \{1, 2, \dots, n\}$$

Our aspiration is enshrined in the following definition.

Definition 4.4

$$P(E) = \lim_{n \rightarrow \infty} \frac{N_n(E)}{n}$$

\square

This is, of course, a rather subtle statement: “limits” are not to be taken lightly and do not always exist. In practice, you will believe that you have arrived at a “limit” when your fraction $N_n(E)/n$ settles down, and stops changing as n increases.

In Example 4.1, we might toss a coin a large number of times and observe how many heads occur. The outcomes of the series of tosses indicate how “fair” the coin is. You might expect in over 1,000 tosses that there would be between 450 and 550 heads, which would suggest a fair coin. If however you found that heads occurred 750 times say, you would see the coin as biased, with the probability of a head about 3/4. In the long run, the relative frequency of heads to the total number of tosses should

approach 0.5 as the number of tosses increases. If it does not, then there is a reason to be suspicious about this so-called “fair” coin: gamblers beware! Try it yourselves, start tossing.

$$P(E) \approx \frac{\text{number of heads}}{\text{number of tosses}}$$

In Example 4.3, we might pull a card from a deck a large number of times, returning and shuffling the pack each time, and observe the number of spades.

$$P(E) \approx \frac{\text{number of spades}}{\text{number of draws}}$$

In Example 4.4, if you were to throw a die repeatedly you would expect each face 1, 2, … , 6 to appear approximately equally often, indicative of a fair die. If, in over 1000 throws, you were to find 700 evens say, you would again see bias, with the probability of an even at around 70%.

$$P(E) \approx \frac{\text{number of even numbers}}{\text{number of times the die is rolled}}$$

If $P(E)$ is not close to 0.5, there is reason to believe that the die is “loaded.”

4.9 SIMULATING PROBABILITIES

In *R*, we can simulate the relative frequency approach by using the *sample* function as follows:

Returning to Example 4.1 (*Tossing a coin*)

```
x <- sample(c("H", "T"), 10, replace = TRUE)
```

generates 10 items from the vector `c("H", "T")` which contains *H* and *T* and stores them in *x*. The `replace = TRUE` indicates that the items are replaced after each selection which, of course, means that items can repeatedly occur in the sample. To see what *x* contains, write

```
x
[1] "T" "T" "H" "H" "T" "T" "T" "H" "H" "T"
```

To count the number of *H*'s and *T*'s, use

```
table(x)
```

which gives

```
x
H T
4 6
```

To obtain the estimated probabilities, that is, the relative frequencies,

```
table(x) /10
```

gives

x	
H	T
0.40	0.60

You can check and observe that the estimated probabilities approach the true value as the number of tosses increases. To simulate 100 tosses, write

```
x <- sample(c("H", "T"), 100, replace = T)
table(x)
```

which gives

x	
H	T
47	53

and

```
table(x) /100
```

gives

x	
H	T
0.47	0.53

Here, we see that we are nearer the 50 : 50 expected ratio than we were with 10 tosses.

When we increase the number of tosses to 1,000, we are nearer still.

```
x<- sample(c("H", "T"), 1000, replace = T)
table(x)/1000
```

x	
H	T
0.493	0.507

and with 10,000 tosses,

```
x <- sample(c("H", "T"), 10000, replace = T)
```

```
table(x)/10000
x
      H         T
0.4983 0.5017
```

the probabilities are very near the “true value” of 0.5.

Example 4.25

A manufacturing process is known to produce 20% defective items.

To simulate the process in *R*, write

```
sample( c("d", "g"), 10, replace = T, prob = c(0.2, 0.8))
[1] "g" "g" "g" "g" "d" "d" "g" "d" "g" "g"
```

Notice here that the first vector `c("d", "g")` contains the sample space, and the vector at the end of the command `prob = c(0.2, 0.8)` contains the probabilities. \triangleleft

In the above examples, we already know the probability of the event occurring on each empirical experiment. The real power of simulation comes from its ability to estimate probabilities when they are not known ahead of time.

In the next example, we consider a problem where the exact answer is difficult to obtain but for which simulation gives an insight into the process.

Example 4.26

Jane and Amy are gambling together. A fair coin is tossed repeatedly. Each time a head comes up, Jane wins a euro from Amy, and each time a tail comes up, Jane loses a euro to Amy.

Carry out this experiment 50 times, and estimate the number of times that Jane is ahead in these 50 tosses. How much has Jane won or lost?

To generate the 50 tosses, we could write

```
x <- sample (c("H", "T"), 50, replace = TRUE)
```

Equivalently, a head means +1 euro for Jane and a tail means -1 euro; so, we might generate

```
x <- sample (c(1, -1), 50, replace = T)
```

which simply translates the amount Jane stands to gain or lose at each toss. Note that `replace = TRUE` may be written as `replace = T`. The results for the 50 tosses are

```
x
[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
[26] -1  1  1 -1  1 -1  1  1 -1 -1 -1 -1  1  1  1 -1 -1  1 -1 -1 -1  1  1  1
```

To calculate the accumulated losses and gains at each toss, write in *R*

```
winnings <- cumsum (x)
```

The *R* function `cumsum` obtains the partial sums of the vector *x* and, in this case, it represents the accumulated amount won or lost at each toss.

To see what Jane has in hand at each toss

```
winnings
[1] 1 2 1 2 3 4 3 4 5 4 3 2 3 4 3 2 1 2 3 2 3 4 3 2 1 0 -1 -2 -1 0 1 0 -1 0 -1 -2 -3 -2 -1 0
[26] 0 1 2 1 2 1 2 3 2 1 0 -1 -2 -1 0 1 0 -1 0 -1 -2 -3 -2 -1 0
```

A plot of this is obtained with

```
num <- 1:50
plot(num, winnings, type = "o",
      xlab = "Toss number", ylab = "Winnings")
abline(0,0)
```

`type = "o"` joins the points, `abline(0,0)` draws a line based on the intercept and slope *a* and *b* respectively; here *a* = 0 and *b* = 0. We could alternatively use `abline(h=0)` with the same effect.

Figure 4.3 shows that, in this simulation of the game, Jane is winning up to the 35th toss, and then dips below zero but never as much as the rise for the gains.

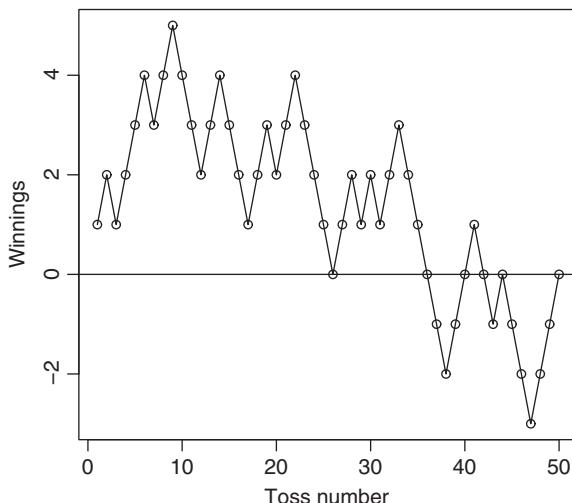


FIGURE 4.3 Jane's Winnings at Each Toss

Despite this, Jane's overall winnings amount to zero as can be seen by using
winnings [50]

or

sum (x)

both yield

0

indicating that at the end of the game, Jane just broke even, in spite of being ahead for most of the time. \triangleleft

Simulation is invaluable to estimate situations like these, where, on the one hand, the results are counterintuitive, and on the other, it is not clear how to calculate the theoretical probabilities.

EXERCISES 4.2

1. If passwords can consist of eight letters, find the probability that a randomly chosen password will not have any repeated letters.
2. A seven-letter password consists of case sensitive letters (52 in all) and the 10 digits. If a valid password must begin with a letter, and contain at least one digit, how many passwords are possible?
3. Selma Solutions Ltd claims that its product has the ability to test up to 300,000,000 passwords in a second on a standard smartphone. If you lost your smartphone, and the finder uses this system, how long will it take to break into your phone if the password consists of
 - (a) eight case sensitive letters without repeats?
 - (b) ten case sensitive letters?
 - (c) eight characters chosen from the alphabet, case sensitive and 10 digits?
4. Using password cracking tools, it is claimed that passwords can be tested at the rate of one hundred million passwords in one second. If passwords consist of eight characters from the 10 digits, and the 26 letters of the alphabet, and are case sensitive, and if passwords are not allowed to have repeated characters, how long would it take to try all of these passwords?
5. A sample of size 10 is chosen at random from a class of 100 consisting of 60 females and 40 males. Obtain the probability of getting 10 females.

6. A box with 15 silicon chips contains five defectives. If a random sample of three chips is drawn at random, what is the probability that all three are defective?
7. A batch of 50 CPU chips contains 10 that are defective. Two chips are selected at random, without replacement.
 - (a) What is the probability that the first one selected is defective?
 - (b) What is the probability that the second one selected is defective?
 - (c) What is the probability that both are defective?
 - (d) How would the probability in (b) change if chips selected were replaced before the next selection?
8. In a party of five students, compute the probability that at least two have the same birthday (month/day), assuming a 365-day year.
9. The probability that two students in a class have the same birthday is at least 75%. What is the minimum size of the class?
10. A series of 20 jobs arrive at a computing center with 50 processors. Assume that each of the jobs is equally likely to go through any of the processors.
 - (a) Find the probability that a processor is used at least twice.
 - (b) What is the probability that at least one processor is idle?
 - (c) If any processor can handle at most four jobs without being overloaded, what is the probability of an overload?

4.10 PROJECTS

1. Use R to illustrate that the probability of getting:
 - (a) a head is 0.5 if a fair coin is tossed repeatedly;
 - (b) a red card is 0.5 if cards are drawn repeatedly with replacement from a well-shuffled deck;
 - (c) an even number is 0.5 if a fair die is rolled repeatedly.
2. An experiment consists of tossing two fair coins. Use R to simulate this experiment 100 times and obtain the relative frequency of each possible outcome. Hence, estimate the probability of getting one head and one tail in any order.
3. An experiment consists of rolling a die. Use R to simulate this experiment 600 times and obtain the relative frequency of each possible outcome. Hence, estimate the probability of getting each of 1, 2, 3, 4, 5, and 6.
4. Amy and Jane are gambling together. A fair coin is tossed repeatedly. Each time a head comes up, Amy wins two euro from Jane, and each time a tail comes up, Amy loses two euro to Jane. Use R to simulate this game 100 times, and estimate:
 - (a) the number of times that Amy is ahead in these 100 tosses;

- (b) how much Amy has won or lost after 100 tosses.
5. A series of 20 jobs arrive at a computing center with 50 processors. Assume that each of the jobs is equally likely to go through any of the processors.
- Simulate the allocation problem a large number of times, and estimate how many times the most used processor was used. How many of the processors were not used at all?
 - In Example 4.21, you will recall that we calculated that over 431 processors are needed to be at least 90% sure that no processor will receive more than one of the 10 jobs to be allocated. You will agree that this appears excessively large. To assure yourself that the answer is correct, simulate the experiment a large number of times, and record the usage patterns of the processors.
6. When plotting the graph in Fig. 4.3, we used vertical lines to attach probabilities, that is, “type = o” in the plot command, which joined the points.

Alternative characterizations of a plot may be obtained by using different options. For example,

“p” for points

“l” for lines

“b” for both

“c” for the lines part alone of “b”

“h” for “histogram-like (or high-density) vertical lines

“s” for stair steps

“n” for no plotting.

Explore the possibilities of different types of plots by varying the `type` symbols.

RECOMMENDED READING

Olofsson, P. (2015), *Probabilities: The Little Numbers That Rule Our Lives*, Wiley.

5

RULES OF PROBABILITY

So far, we have defined the issues relating to probability in an ad hoc way. Our approach to solving probability problems has been somewhat intuitive, treating each problem as it arrived as something new. We now formalize the definitions, and define the assumptions that we have made, or need to make, to develop the probability system further, by introducing the axioms of probability. We first provide some further notation on events.

5.1 PROBABILITY AND SETS

In considering probability, it is enlightening to use the language and notation of sets.

If E_1 and E_2 are events in the sample space, then the union $E_1 \cup E_2$ denotes the event “ E_1 or E_2 .”

Let us look at the simple experiment in Example 4.3 of pulling a card from a deck. Let E_1 be the event that we draw a spade and E_2 be the event that we draw an ace from a deck.

$$E_1 = \text{Spade}, \quad E_2 = \text{Ace}$$

$$E_1 \cup E_2 = \text{Spade or Ace}$$

TABLE 5.1 Sets and Events

Sets	Events
$E_1 \cup E_2$	E_1 or E_2
$E_1 \cap E_2$	E_1 and E_2

Counting the number of spades (13) and the remaining aces (3), we have

$$P(E_1 \cup E_2) = \frac{16}{52}$$

Similarly, $E_1 \cap E_2$ denotes the event “ E_1 and E_2 .”

Suppose, when pulling a card from a deck, we are interested in getting a spade and an ace.

Again

$$E_1 = \text{Spade}, \quad E_2 = \text{Ace}$$

Now, we denote

$$E_1 \cap E_2 = \text{Spade and Ace}$$

There is only one ace of spades.

$$P(E_1 \cap E_2) = P(\text{Spade and Ace}) = \frac{1}{52}$$

Table 5.1 summarizes the and/or relationship of sets and events.

5.2 MUTUALLY EXCLUSIVE EVENTS

Two events E_1 and E_2 are said to be mutually exclusive if they cannot occur together.

For example, if we pull a card from a deck, let us consider

$$E_1 = \text{Spade}, \quad E_2 = \text{Heart}$$

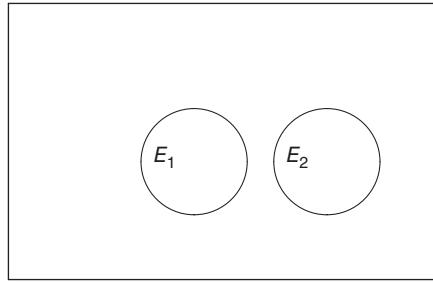
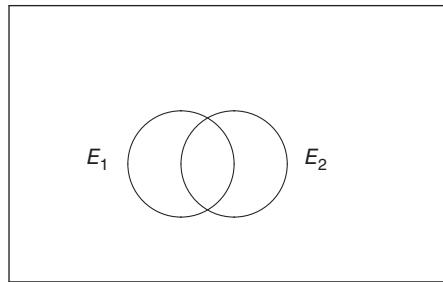
With one card selected from this deck, it is impossible to get both a heart and a spade. We may get one or the other but not both. We say that the intersection $E_1 \cap E_2$ is empty and denote it by

$$E_1 \cap E_2 = \emptyset$$

and the joint probability is 0.

$$P(E_1 \cap E_2) = 0$$

E_1 and E_2 are said to be mutually exclusive.

**FIGURE 5.1** Mutually Exclusive Events**FIGURE 5.2** Non Mutually Exclusive Events

We have shown, in the previous section, that when

$$E_1 = \text{Spade}, \quad E_2 = \text{Ace}$$

then

$$P(E_1 \cap E_2) = \frac{1}{52}$$

In this case,

$$P(E_1 \cap E_2) \neq 0$$

E_1 and E_2 are said to be non mutually exclusive.

Mutually exclusive and non mutually exclusive events are illustrated in Figs. 5.1 and 5.2, respectively.

5.3 COMPLEMENTARY EVENTS

The nonoccurrence of an event E is said to be the complementary event and denoted by \bar{E} . It is that part of the sample space S that is not contained in E (Fig. 5.3). For

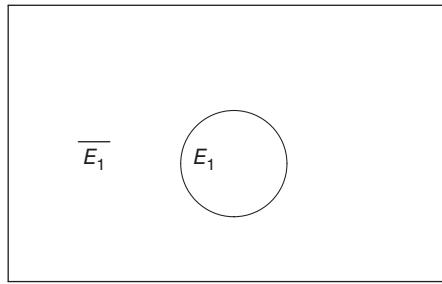


FIGURE 5.3 Complementary Events

example, when drawing a card from a deck, if E consists of an ace, then \bar{E} consists of all those cards that are not aces.

5.4 AXIOMS OF PROBABILITY

Officially, axioms are defined by mathematicians as self-evident truths. In practice, we are a little more pragmatic. Here, for probability, axioms are to be the rules that any random experiment must satisfy.

A probability function P , defined on subsets of the sample space S , satisfies the following axioms.

1. *Positivity*: For all $E \subseteq S$, then

$$P(E) \geq 0$$

2. *Additivity*: If E_1 and E_2 are mutually exclusive events, that is, $E_1 \cap E_2 = \emptyset$, then

$$P(E_1 \cup E_2) = P(E_1) + P(E_2)$$

3. *Certainty*: For the sample space S

$$P(S) = 1$$

Axiom 1 asserts that all probabilities are nonnegative numbers.

Axiom 2 states that the probability of the union of two mutually exclusive events is the sum of the individual event probabilities; this can be seen to be “self-evident” if we examine Fig. 5.1.

Axiom 3 requires that the entire sample space S has a probability of 1. This also is reasonably “self-evident” if we think that something must happen, whenever we carry out an experiment.

These axioms enable us to derive other properties of probability. But before doing so, let us look at Example 4.13 in the previous chapter, where we considered the following **if** statement in a program:

if B **then** s_1 **else** s_2

The random experiment consists of “observing” two successive executions of the **if** statement. The sample space consists of the four possible outcomes:

$$S = \{(s_1, s_1), (s_1, s_2), (s_2, s_1), (s_2, s_2)\}$$

We assumed earlier that all these points in the sample space are equally likely. Suppose we now know that, on the basis of strong experimental evidence, the following probability assignment is justified:

$$P(s_1, s_1) = 0.34, \quad P(s_1, s_2) = 0.26, \quad P(s_2, s_1) = 0.26, \quad P(s_2, s_2) = 0.14$$

As before, we let E be the event of at least one execution of the statement s_1 . Then E consists of three possibilities,

$$E_1 = \{(s_1, s_1)\}$$

$$E_2 = \{(s_1, s_2)\}$$

$$E_3 = \{(s_2, s_1)\}$$

and

$$E = E_1 \cup E_2 \cup E_3.$$

Since, in any execution of this program, just one of these events is possible, then we can say that the three events are mutually exclusive, and apply Axiom 2 repeatedly to get

$$\begin{aligned} P(E) &= P(E_1) + P(E_2) + P(E_3) \\ &= 0.34 + 0.26 + 0.26 \\ &= 0.86 \end{aligned}$$

Suppose we are also interested in the event that statement s_2 is executed first. This event consists of two mutually exclusive events,

$$E_3 = \{(s_2, s_1)\}$$

and

$$E_4 = \{(s_2, s_2)\}$$

with

$$E = E_3 \cup E_4$$

Since E_3 and E_4 are mutually exclusive

$$\begin{aligned} P(E) &= P(E_3) + P(E_4) \\ &= 0.26 + 0.14 \\ &= 0.40 \end{aligned}$$

5.5 PROPERTIES OF PROBABILITY

Axioms are supposed to form an economical list, but they should also have interesting consequences. This is true for the axioms of probability.

Property 1 *Complementary events*

For each $E \subseteq S$,

$$P(\bar{E}) = 1 - P(E)$$

Proof

$$S = E \cup \bar{E}$$

Now, E and \bar{E} are mutually exclusive, that is,

$$E \cap \bar{E} = \emptyset$$

Hence,

$$P(S) = P(E \cup \bar{E}) = P(E) + P(\bar{E}) \quad \text{from Axiom 2}$$

Also,

$$P(S) = 1 \quad \text{from Axiom 3}$$

that is,

$$1 = P(E) + P(\bar{E})$$

So,

$$P(\bar{E}) = 1 - P(E)$$

□

Property 1, the complementary approach, is used in cases where the complement of an event of interest is easier to handle than the original event. You will recall that this property was anticipated earlier when we used it to solve the birthday problem (Example 4.20). We calculated the probability that all the students in the class had different birthdays, and by subtracting this from 1, we obtained the probability that at least two students have the same birthday. It would have been very difficult to

obtain this result directly. The probability that at least two have the same birthday is a compound event consisting of very many simple events, for example, exactly two, exactly three, exactly four the same, and so on.

Property 2 *The empty set*

$$P(\emptyset) = 0 \quad \text{where } \emptyset \text{ is the empty set}$$

Proof

$$S = S \cup \emptyset$$

Now, S and \emptyset are mutually exclusive events, that is,

$$S \cap \emptyset = \emptyset$$

Hence,

$$P(S) = P(S \cup \emptyset) = P(S) + P(\emptyset) \quad \text{from Axiom 2}$$

Also,

$$P(S) = 1 \quad \text{from Axiom 3}$$

Therefore,

$$1 = 1 + P(\emptyset)$$

Then,

$$P(\emptyset) = 0$$

□

What we are saying here is that the *impossible event* “has probability zero.”

Property 3 *Monotonicity*

If E_1 and E_2 are subsets of S such that $E_1 \subseteq E_2$,

$$P(E_1) \leq P(E_2)$$

Proof

$$E_2 = E_1 \cup (E_2 \cap \bar{E}_1)$$

Since E_1 and $E_2 \cap \bar{E}_1$ are mutually exclusive, then from Axiom 2

$$\begin{aligned} P(E_2) &= P(E_1) + P(E_2 \cap \bar{E}_1) \\ &\geq P(E_1) \end{aligned}$$

as $P(E_2 \cap \bar{E}_1) \geq 0$ from Axiom 1.

□

This is an intuitively obvious property, which simply states that if an event E_2 contains all the outcomes of an event E_1 and possibly more, then E_2 has a higher probability of occurring than E_1 .

Property 4 *Probability between 0 and 1*

For each $E \subseteq S$,

$$0 \leq P(E) \leq 1$$

Proof

Since

$$\emptyset \subseteq E \subseteq S$$

then from Property 3

$$P(\emptyset) \leq P(E) \leq P(S)$$

$$0 \leq P(E) \leq 1$$

□

This property, together with Axiom 1, confirms that every event has its probability somewhere between 0 and 1.

Property 5 *The addition law of probability*

If E_1 and E_2 are subsets of S , then

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$$

Proof

$$E_1 \cup E_2 = E_1 \cup (E_2 \cap \bar{E}_1)$$

Now, since E_1 and $E_2 \cap \bar{E}_1$ are mutually exclusive,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2 \cap \bar{E}_1) \quad \text{from Axiom 2} \tag{5.1}$$

Now, E_2 may be written as the union of two mutually exclusive events as follows:

$$E_2 = (E_2 \cap E_1) \cup (E_2 \cap \bar{E}_1)$$

So,

$$P(E_2) = P(E_2 \cap E_1) + P(E_2 \cap \bar{E}_1) \quad \text{from Axiom 2}$$

Thus,

$$P(E_2 \cap \bar{E}_1) = P(E_2) - P(E_2 \cap E_1) \tag{5.2}$$

Inserting Equation 5.2 into Equation 5.1, we get

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2) \quad (5.3)$$

□

The addition law of probability is best illustrated with the “pulling cards from a deck experiment.”

Example 5.1

A card is drawn from a well-shuffled deck. What is the probability that the card is an ace or a heart?

Let A be the event that the card is an ace.

$$P(A) = \frac{4}{52}$$

Let B be the event that the card is a heart.

$$P(B) = \frac{13}{52}$$

Then, from Equation 5.3,

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Now $A \cap B$ is the event that the selected card is an ace and a heart. Since there is just one ace of hearts, then

$$P(A \cap B) = \frac{1}{52}$$

So,

$$P(A \cup B) = \frac{4}{52} + \frac{13}{52} - \frac{1}{52} = \frac{16}{52}$$

△

Example 5.2

Suppose that, after five years of use, 30% of tablet computers have problems with the camera, 20% have problems with the flash memory, and 10% have problems with both. What is the proportion of tablets that experience no problems with the camera or the flash memory after five years?

Let E_1 be the event of a camera problem.

$$P(E_1) = 0.3$$

Let E_2 be the event of a flash memory problem.

$$P(E_2) = 0.2$$

Then $E_1 \cup E_2$ is the event that there is a problem with either the camera or the memory or both. From Equation 5.3,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$$

Now $E_1 \cap E_2$ is the event that there is a problem with both the camera and the memory.

$$P(E_1 \cap E_2) = 0.1$$

So,

$$P(E_1 \cup E_2) = 0.3 + 0.2 - 0.1 = 0.4$$

is the probability that there is a problem with either the camera or the memory or both. The complementary event

$$1 - P(E_1 \cup E_2) = 1 - 0.4 = 0.6$$

is the probability that there are no problems with the camera or the flash memory, which means that, after five years, 60% of tablet computers will still have problem-free cameras and flash memory. \triangleleft

Example 5.3

Let us examine once again the data given in Table 4.1 in Example 4.14.

	Compiles on First Run	Does Not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

Suppose we are now interested in the probability that a program is written in C++ or compiles on first run. We denote E as the event that a program drawn at random, compiles on the first run, and let C++ that the program is written in C++.

Then we might ask

$$P(\text{C++} \cup E) = ?$$

Notice that the two events C++ and E are not mutually exclusive because it is obviously possible to select a C++ program that compiles on the first run.

From the addition law of probability

$$P(\text{C++} \cup E) = P(\text{C++}) + P(E) - P(\text{C++} \cap E)$$

We can read from the table that

$$P(\text{C++}) = 120/200, \quad P(E) = 136/200, \quad P(\text{C++} \cap E) = 72/200$$

Therefore,

$$P(\text{C++} \cup E) = \frac{120}{200} + \frac{136}{200} - \frac{72}{200}$$

Calculate in R

$$120/200 + 136/200 - 72/200$$

[1] 0.92

Similarly, for the probability that a randomly selected program is written in Java or does not compile on the first run, we are looking for

$$P(\text{Java} \cup \bar{E}) = P(\text{Java}) + P(\bar{E}) - P(\text{Java} \cap \bar{E})$$

$$P(\text{Java} \cup \bar{E}) = \frac{80}{200} + \frac{64}{200} - \frac{16}{200}$$

Calculate in R

$$80/200 + 64/200 - 16/200$$

[1] 0.64

△

Example 5.4

A university library system uses passwords that are five characters in length, and each character is one of the 26 letters (a–z) or 10 digits (0–9). The first character has to be a letter.

Suppose a hacker selects a password at random. Determine the probabilities that the selected password:

1. begins with a vowel (a, e, i, o, u);
2. ends with an odd number (1, 3, 5, 7, 9);
3. either begins with a vowel or ends with an odd number.

You may assume that passwords are not case sensitive.

Solution

1. Let A denote the event that the password begins with a vowel (a, e, i, o, u). Then

$$P(A) = \frac{5}{26}$$

Here, the denominator is 26, all letters of the alphabet are possible to start with.

2. Let B denote the event that a password ends with an odd number (1, 3, 5, 7, 9). Then,

$$P(B) = \frac{5}{36}$$

Here, the denominator 36 is calculated to be the number of possible endings, all the 26 letters of the alphabet and the 10 digits.

3. The addition law of probability (Equation 5.3) tells us that the probability of A or B is

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

To calculate $P(A \cap B)$, the probability that a password starts with a vowel and ends with an odd number, we first calculate the number of possible passwords.

The first character can be any of the 26 letters of the alphabet, and each of the next characters can be any of the 26 letters as well as any of the 10 digits, that is, 36 possibilities in all. So the number of possible passwords is 26×36^4 .

As there are five ways to start with a vowel, and five ways to end with an odd number, then, the number of possible passwords that start with a vowel and end with an odd number is $5 \times 36^3 \times 5$. Obviously, the intervening three characters in the middle can assume any of the 26 letters and 10 digits.

So,

$$P(A \cap B) = \frac{5 \times 36^3 \times 5}{26 \times 36^4} = \frac{5}{26} \times \frac{5}{36}$$

Using the addition law, the probability that a password begins with a vowel or ends with an odd number is

$$P(A \cup B) = \frac{5}{26} + \frac{5}{36} - \left(\frac{5}{26} \times \frac{5}{36} \right)$$

Calculate in R

```
5/26 + 5/36 - (5/26)*(5/36)
[1] 0.3044872
```

which means that there is over a 30% chance of selecting a password beginning with a vowel and ending with an odd number. \triangleleft

EXERCISES 5.1

1. A virus enters the system either through email or through the Internet. There is a 30% chance of receiving the virus through the Internet and a 40% chance of receiving it through email. There is a 15% chance that the virus enters the

- system simultaneously through the Internet and email. Calculate the probability that the virus enters the system.
2. Among employees at Daly Developers Ltd, 80% use *R*, 60% use Python, and 50% use both. What proportion of programmers use either *R* or Python?
 3. A computer program is subjected to two independent tests. These tests will discover an error with probabilities 0.8 and 0.7 respectively. Suppose a program contains an error, what is the probability that it will be discovered?
 4. Orders for a computer are summarized by the optional features that are requested as follows: 50% of the purchases chose the base model with no extra features, 30% had one optional feature, and 20% had two.
 - (a) What is the probability that an order requests at least one optional feature?
 - (b) What is the probability that an order does not request more than one optional feature?
 5. An Internet banking system uses passwords that are six characters, and each character is one of the 26 letters (a–z) or 10 digits (0–9). The first character has to be a letter. Determine the proportion of passwords that:
 - (a) begins with a consonant;
 - (b) ends with an even number (0, 2, 4, 6, 8);
 - (c) begins with a consonant or ends with an even number.You may assume that passwords are not case sensitive.

6. If E_1 , E_2 , and E_3 are subsets of S show that

$$\begin{aligned}P(E_1 \cup E_2 \cup E_3) &= P(E_1) + P(E_2) + P(E_3) \\&\quad - P(E_1 \cap E_2) - P(E_2 \cap E_3) - P(E_1 \cap E_3) \\&\quad + P(E_1 \cap E_2 \cap E_3)\end{aligned}$$

7. A computer program is subjected to three independent tests. These tests will discover an error with probabilities 0.8, 0.7, and 0.6, respectively. Suppose a program contains an error, what is the probability that it will be discovered?

SUPPLEMENTARY READING

Nahin, P. J. (2012), *Dueling Idiots and Other Probability Puzzles*, Princeton University Press.

6

CONDITIONAL PROBABILITY

Intuitively, the likelihood of one event occurring may well be influenced by the occurrence or otherwise of other events: for example, the probability that a student will not turn up to a lecture will be influenced by some other happening, such as maybe whether or not it is raining, or whether or not the freshers' ball was held the night before. This is *conditional* probability.

To motivate the idea, let us return to Table 4.1, where we considered data from a software development company, wherein 200 programs were written each week in either C++ or Java, with the following compiling frequencies:

	Compiles on First Run	Does Not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

Suppose, we now ask what the probability is that a program, chosen at random from the week's supply, compiles on the first run, given that we know it has been written in C++. In other words, we know that the program, which may or may not compile on the first run, has been selected from the 120 programs that were written

in C++. In effect, the sample space reduces to the 120 programs written in C++, of which 72 compiled on the first run; thus the probability that the selected program will compile on the first run is 72/120.

Let us examine this further. Denote by C++ and Java, the events that a program is written in C++ and Java respectively, and let E be the event that a program selected compiles on the first run and \bar{E} the event that it does not.

For conditional probability, $P(E|C++)$ is the probability that the program will compile on the first run given that it has been written in C++. It is the probability that the program compiles and is written in C++, relative to the probability that it is a C++ program.

$$P(E|C++) = \frac{P(E \cap C++)}{P(C++)} \quad (6.1)$$

Now $E \cap C++$ is the event that a program from the week's supply both compiles on the first run E and is written in C++. There are 200 programs in the week's supply, and of these a total of 72 are written in C++ and compile on the first run, which means

$$P(E \cap C++) = \frac{72}{200} \quad (6.2)$$

Also, 120 of the 200 programs are written in C++, which gives

$$P(C++) = \frac{120}{200} \quad (6.3)$$

Putting Equations 6.2 and 6.3 into Equation 6.1, we get

$$P(E|C++) = \frac{72/200}{120/200} = \frac{72}{120} \quad (6.4)$$

which is what we already deduced above.

Similarly, if we know that a program selected at random is written in Java, the probability that it compiles on the first run is

$$\begin{aligned} P(E|Java) &= \frac{P(E \cap Java)}{P(Java)} \\ &= \frac{64/200}{80/200} = \frac{64}{80} \end{aligned}$$

$P(E|C++)$ and $P(E|Java)$ are called conditional probabilities, and they may differ from the probability of E without any imposed condition, which is that a program compiles on the first run.

$$P(E) = \frac{136}{200} = 0.68, \text{ while } P(E|C++) = \frac{72}{120} = 0.6 \text{ and } P(E|Java) = \frac{64}{80} = 0.8$$

Generally, for any two events E_1 and E_2 , we write

$$P(E_2|E_1) = \text{the probability of } E_2 \text{ assuming the occurrence of } E_1$$

which we can refer to as “the probability of E_2 given E_1 .”

Definition 6.1 *Conditional probability*

The conditional probability of an event E_2 given an event E_1 is defined as

$$P(E_2|E_1) = \frac{P(E_2 \cap E_1)}{P(E_1)} \quad (6.5)$$

□

As we have said, this is the probability of the occurrence of the event $E_2 \cap E_1$ in the sample space E_1 .

Example 6.1

Consider four computer firms A, B, C, D bidding for a certain contract. A survey of past bidding success of these firms shows the following probabilities of winning:

$$P(A) = 0.35, \quad P(B) = 0.15, \quad P(C) = 0.3, \quad P(D) = 0.2$$

Before the decision is made to award the contract, firm B withdraws the bid. Find the new probabilities of winning the bid for A, C , and D .

Here, we need the conditional probabilities of A, C , and D given the complementary event \bar{B} : $P(A|\bar{B})$, $P(C|\bar{B})$, and $P(D|\bar{B})$.

$$P(\bar{B}) = 1 - P(B) = 1 - 0.15 = 0.85.$$

To obtain the other probabilities given that B has withdrawn, we normalize them with respect to \bar{B} :

$$P(A|\bar{B}) = \frac{P(A \cap \bar{B})}{P(\bar{B})} = \frac{0.35}{0.85} = 0.413$$

$$P(C|\bar{B}) = \frac{P(C \cap \bar{B})}{P(\bar{B})} = \frac{0.3}{0.85} = 0.352$$

$$P(D|\bar{B}) = \frac{P(D \cap \bar{B})}{P(\bar{B})} = \frac{0.2}{0.85} = 0.235$$

Note here that $A \cap \bar{B}$, the event that firm A gets the contract and firm B does not, is the same as the event that A gets the contract, that is,

$$P(A \cap \bar{B}) = P(A)$$

Similarly

$$P(C \cap \bar{B}) = P(C)$$

and

$$P(D \cap \bar{B}) = P(D)$$

You should check that

$$P(A|\bar{B}) + P(C|\bar{B}) + P(D|\bar{B}) = 1$$

◻

6.1 MULTIPLICATION LAW OF PROBABILITY

A rearrangement of Equation 6.5 yields the *multiplication law of probability*.

$$P(E_1 \cap E_2) = P(E_1)P(E_2|E_1) \quad (6.6)$$

For example, if we were to pull two cards from a deck without replacement, we might let B_1 be the event of a black on the first draw, and B_2 a black on the second draw.

To calculate the probability of two blacks, write

$$\begin{aligned} P(B_1 \cap B_2) &= P(B_1) \cdot P(B_2|B_1) \\ &= \frac{26}{52} \times \frac{25}{51} \end{aligned}$$

This multiplication law of probability can be generalized to more than two events.

$$\begin{aligned} P(E_1 \cap E_2 \cap E_3 \cap \cdots \cap E_k) &= P(E_1)P(E_2|E_1)P(E_3|E_1 \cap E_2) \\ &\quad \cdots P(E_k|E_1 \cap E_2 \cap \cdots \cap E_{k-1}) \end{aligned}$$

Thus, if we were to draw three cards from a deck without replacement, then the probability of three blacks is

$$\begin{aligned} P(B_1 \cap B_2 \cap B_3) &= P(B_1) \cdot P(B_2|B_1) \cdot P(B_3|(B_1 \cap B_2)) \\ &= \frac{26}{52} \times \frac{25}{51} \times \frac{24}{50} \end{aligned}$$

The birthday problem (revisited): Returning to the birthday problem of Example 4.20, instead of using permutations and counting, we could view it as a series of k events and apply the multiplication law of probability.

Let B_i be the event that the birthday of the i th student selected does not match any of the birthdays of the previous students selected.

E is the event that all students have different birthdays.

$$E = B_1 \cap B_2 \cap \cdots \cap B_k$$

Clearly, the first student selected can have a birthday on any of the 365 days, so we set $P(B_1) = 1$. With two students, the probability of different birthdays is that the second student has a birthday different from that of the first.

$$P(E) = P(B_2|B_1) = \frac{354}{365}$$

that is, the second student can have a birthday on any of the days of the year except the birthday of the first student.

With three students, the probability that the third is different from the previous two is

$$P(B_3|B_1 \cap B_2) = \frac{353}{365}$$

that is, the third student can have a birthday on any of the days of the year, except the two of the previous two students.

Generally, with k students,

$$P(B_k|B_1 \cap B_2 \cap \cdots \cap B_{k-1}) = \frac{365 - (k - 1)}{365}$$

Using the multiplication law of probability for k events, the probability that all of the k students have different birthdays is

$$P(E) = \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365 - (k - 1)}{365}$$

and the probability that at least two students have the same birthday,

$$P(\bar{E}) = 1 - \left(\frac{364}{365} \times \cdots \times \frac{365 - (k - 1)}{365} \right)$$

which is the same result as we got in Example 4.20. It is also the approach that we used in the *R* program that we developed for calculating the birthday probabilities in Section 4.4.

6.2 INDEPENDENT EVENTS

Sometimes, the occurrence of one event has no effect on the occurrence of another. For example, when tossing a fair coin, the probability of a head is $1/2$ in every toss,

so the probability of a second head, given that the first toss is a head, remains at 1/2. With the obvious notation,

$$P(H_2|H_1) = P(H_2) = \frac{1}{2}$$

The outcome of the second toss is *independent* of the outcome of the first.

6.2.1 Independence of Two Events

Definition 6.2 *Independence of two events*

E_2 is said to be independent of E_1 if

$$P(E_2|E_1) = P(E_2)$$

Therefore, it follows that the multiplication law, given in Equation 6.6, reduces to

$$P(E_1 \cap E_2) = P(E_1)P(E_2) \quad (6.7)$$

when E_1 and E_2 are independent events. \square

Clearly, “ E_2 is independent of E_1 ” means exactly the same as “ E_1 is independent of E_2 ,” and we can simply say “ E_1 and E_2 are independent.” If E_1 and E_2 do not satisfy Equation 6.7, they are said to be dependent.

Example 6.2

As part of a promotional campaign, a store is offering a free Fitbit with the purchase of the new iPhone model. If 5% of the iPhones are faulty and 10% of the Fitbits are faulty, what is the probability that a customer gets both a faulty iPhone and a faulty Fitbit?

Let E_1 be the event that the iPhone is faulty, and E_2 be the event that the Fitbit is faulty. Then,

$$P(E_1) = 0.05 \quad \text{and} \quad P(E_2) = 0.1$$

Therefore,

$$\begin{aligned} P(E_1 \cap E_2) &= P(E_1)P(E_2) \\ &= 0.05 \times 0.1 = 0.005 \end{aligned} \quad \square$$

Example 6.3

Ava keeps a backup copy of her programs on the hard disk of her laptop and on a memory stick. There is a 10% chance that the hard disk will crash and a 50% chance

that the memory stick will be lost. What is the probability that Ava's work will get lost?

Suppose E is the event that Ava's work will get lost.

Let E_1 be the event that the hard disk will crash, and E_2 be the event that the memory stick will be lost. We can write $E = E_1 \cap E_2$.

Ava will lose her work if the computer crashes and the memory stick containing the backup copy of the work gets lost.

$$P(E) = P(E_1 \cap E_2)$$

Since the hard disk and the memory stick are independent units,

$$\begin{aligned} P(E) &= P(E_1) \times P(E_2) \\ &= 0.1 \times 0.5 \\ &= 0.05 \end{aligned}$$

There is a 5% chance that Ava's work will be lost. Perhaps a better backup system should be considered. The cloud maybe? \triangleleft

We have, of course, neglected the not impossible situation in which Ava is robbed of both her laptop and memory stick.¹

6.3 INDEPENDENCE OF MORE THAN TWO EVENTS

Defining the independence of more than two events is a little bit trickier than defining independence of two events.

Definition 6.3 *Pairwise independence of more than two events*

For $k > 2$ independent events, E_1, E_2, \dots, E_k are pairwise independent when any pair E_i, E_j is independent, that is,

$$P(E_j | E_i) = P(E_j) \quad \forall i \neq j$$

\square

Definition 6.4 *Mutual Independence of more than two events*

E_1, E_2, \dots, E_k are mutually independent if and only if, for any subset of these k events, the product law for independent events holds, that is, for any subset i_1, i_2, \dots, i_n ($2 < n \leq k$) that are elements of $(1, 2, \dots, k)$

$$P(E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_n}) = P(E_{i_1})P(E_{i_2}) \cdots P(E_{i_n})$$

¹We are indebted to Raimu Hügli for this observation.

It follows that, for mutually independent events, E_1, E_2, \dots, E_n ,

$$P(E_1 \cap E_2 \cap \dots \cap E_n) = P(E_1)P(E_2) \cdots P(E_n), \quad n \leq k$$

□

Example 6.4

Draw three cards from a deck with replacement. What is the probability that all are black?

In this case, because the cards are replaced each time one is drawn, the probability of getting a black is the same from draw to draw. Let

E_1 be the event that the first is black. $P(E_1) = 26/52$

E_2 be the event that the second is black. $P(E_2) = 26/52$

E_3 be the event that the third is black. $P(E_3) = 26/52$

Therefore,

$$P(3 \text{ black cards}) = P(E_1 \cap E_2 \cap E_3) = \frac{26}{52} \times \frac{26}{52} \times \frac{26}{52}$$

▫

Notice that

$$P(E_1 \cap E_2 \cap E_3) = P(E_1)P(E_2)P(E_3)$$

and

$$P(E_1 \cap E_2) = P(E_1)P(E_2)$$

$$P(E_1 \cap E_3) = P(E_1)P(E_3)$$

$$P(E_2 \cap E_3) = P(E_2)P(E_3)$$

So, as well as being pairwise independent, these events are also mutually independent.

However, events that are pairwise independent are not necessarily mutually independent. Let us look at another example.

Example 6.5

Draw three cards from a deck with replacement. This time, let:

E_1 be the event that the first and second cards are the same suit;

$$P(E_1) = 4 \times \left(\frac{13}{52}\right)^2 = \frac{1}{4}$$

E_2 be the event that the second and third cards are the same suit;

$$P(E_2) = 4 \times \left(\frac{13}{52}\right)^2 = \frac{1}{4}$$

E_3 be the event that the first and third cards are the same suit.

$$P(E_3) = 4 \times \left(\frac{13}{52}\right)^2 = \frac{1}{4}$$

We can show that these events are pairwise independent but not mutually independent.

First, let us look at E_1 and E_2 .

Obviously,

$$P(E_1)P(E_2) = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$$

Also, $E_1 \cap E_2$ is the event that the first, second, and third cards are the same suit.

$$\begin{aligned} P(E_1 \cap E_2) &= P(\text{the first, second, and third cards are the same suit}) \\ &= 4 \times \left(\frac{13}{52}\right)^3 \\ &= \frac{1}{4} \times \frac{1}{4} = \frac{1}{16} \end{aligned}$$

Therefore, $P(E_1 \cap E_2) = P(E_1)P(E_2)$

Similarly, it can be shown that

$$P(E_1 \cap E_3) = P(E_1)P(E_3)$$

and

$$P(E_2 \cap E_3) = P(E_2)P(E_3)$$

which means that E_1 , E_2 , and E_3 are pairwise independent.

However, they are not mutually independent:

$$\begin{aligned} P(E_1 \cap E_2 \cap E_3) &= P(\text{all three cards are the same suit}) \\ &= 4 \times \frac{13}{52} \times \frac{13}{52} \times \frac{13}{52} \\ &= \frac{1}{16} \end{aligned}$$

But

$$P(E_1)P(E_2)P(E_3) = \frac{1}{4} \times \frac{1}{4} \times \frac{1}{4} = \frac{1}{64}$$

which is not equal to $P(E_1 \cap E_2 \cap E_3)$. \triangle

So, these three events are not mutually independent.

6.4 THE INTEL FIASCO

In October 1994, a flaw was discovered in the Pentium chip installed in many new personal computers. The chip produced an incorrect result when dividing two numbers. Intel, the manufacturer of the Pentium chip, initially announced that such an error would occur once in 9 billion divisions, or “once in 27,000 years” for a typical user. Consequently, it did not immediately offer to replace the chip.

They soon discovered that it did not take anywhere near 27,000 years for an error to occur. Within weeks of the release, clients were reporting errors in their calculations.

Let us now investigate how Intel got their probability calculations badly wrong.

If an error occurs “once in 9 billion divisions” as Intel maintained, then for one division performed using the flawed chip, the probability of E the event of an error is

$$P(E) = 1/(9000000000) = 1/(9 \text{ billion})$$

and the probability that no error will occur in one division

$$P(\bar{E}) = 1 - \frac{1}{9 \text{ billion}} \approx 1$$

No error in one division is a practical certainty. Confirming this in *R*

```
error <- 1/9000000000
error
```

returns

```
[1] 1.111111e-10
```

which is practically 0. Also

```
noerror <- 1 - error
noerror
```

returns

```
[1] 1
```

We now consider two successive divisions performed using the flawed chip. Assuming that any one division has no impact on any other division, that is, successive divisions are independent, we can write the probability that no error occurs in two divisions

$$P(E) = \left(1 - \frac{1}{9 \text{ billion}}\right)^2$$

Write in *R*

```
noerror2 <- (1-error)**2
noerror2

returns

[1] 1
```

again illustrating that the probability of getting an error in two divisions is negligible.

But, what Intel had not anticipated was that, depending on the procedure, statistical software packages may have to perform an extremely large number of divisions to produce the required output. For heavy users of the software, 1 billion divisions over a short time period is not unusual.

The probability that 1 billion divisions, performed using the flawed Pentium chip, will result in no errors is

$$P(E) = \left(1 - \frac{1}{9 \text{ billion}}\right)^{1 \text{ billion}}$$

and the probability of at least one error in a billion divisions is

$$P(\bar{E}) = 1 - \left(1 - \frac{1}{9 \text{ billion}}\right)^{1 \text{ billion}}$$

We use *R* to calculate the value.

```
bill<-1000000000 #number of divisions
noerrorbill <- noerror**bill
noerrorbill

returns

[1] 0.8948393
```

The probability that no error occurs in 1 billion divisions is just under 90%, which means that the probability that at least one error occurs in the 1 billion divisions is over 10%.

```
atleastonebill <- 1 - noerrorbill
atleastonebill
[1] 0.1051607
```

Two months after the flaw was discovered, Intel agreed to replace all Pentium chips free of charge.

Let us go to *R* to observe the pattern of errors with increasing numbers of divisions.

```
computations <- seq(0, 10000000000, 100000000) #0-10 billion divisions
error <- 1/9000000000
noerror <- 1 - error
noerrortotal <- noerror^computations # probability of no error
atleastone <- 1 - noerrortotal # probability of at least
    one error
plot(computations, atleastone, xlab = "Number of divisions",
      ylab = "Probability of at least one error",
      ylim = c(0, 1))
```

yields Fig. 6.1.

Figure 6.1 confirms that, with 1 billion divisions, the probability of an error is 10% approximately. With 2 billion ($2e+09$) divisions, it is 20%, and continues to increase. Calculations of this order of magnitude are not unusual when carrying out a statistical modeling problem, so it is no surprise, to statisticians at least, that the chip was withdrawn.

6.5 LAW OF TOTAL PROBABILITY

Often, an event can emanate from different sources, and we may want to calculate the total probability of the occurrence of the event, taking into account all the sources from which it came.

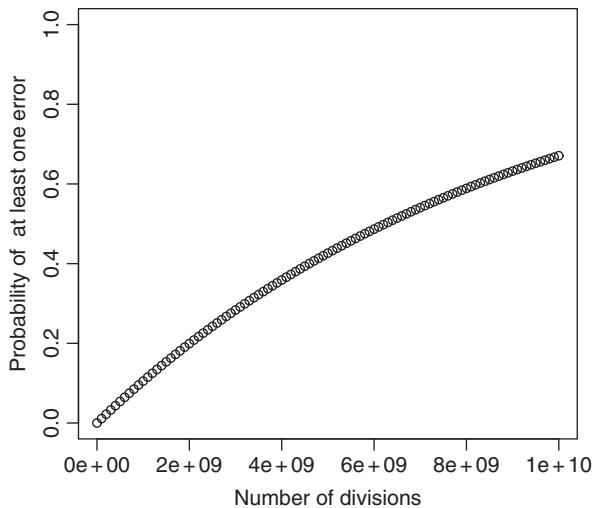


FIGURE 6.1 Error Probabilities with the Intel Chip

Let us look again at the data given in Table 4.1

	Compiles on First Run	Does Not Compile on First Run	
C++	72	48	120
Java	64	16	80
	136	64	200

Suppose we ask, what is the probability that a program compiles on the first run?

Equivalently, what is the overall proportion of programs that compile on the first run?

Reading from the table, we could deduce that the probability is 136/200, but such a table is not often at hand, so let us approach this simple problem in a more formal manner.

The proportion of programs that compile on the first run consists of the proportion of Java programs that compile on the first run, added to the proportion of C++ programs that compile on the first run.

In set notation, if E is the event that a program compiles on the first run, then

$$E = (\text{C++} \cap E) \cup (\text{Java} \cap E)$$

Since $\text{C++} \cap E$ and $\text{Java} \cap E$ are mutually exclusive, then

$$P(E) = P(\text{C++} \cap E) + P(\text{Java} \cap E)$$

This is called the *total* probability of E .

$$\begin{aligned} P(E) &= P(\text{C++})P(E|\text{C++}) + P(\text{Java})P(E|\text{Java}) \\ &= \left(\frac{120}{200}\right)\left(\frac{72}{80}\right) + \left(\frac{80}{200}\right)\left(\frac{64}{120}\right) = \frac{136}{200} \end{aligned}$$

which is what we already anticipated.

This approach illustrates how to calculate the total probability of an event E that can emanate from different sources. In the aforementioned example, the sources are the two different programming languages, C++ and Java. There are different probabilities associated with each of these languages, that is, the proportion of programs written in C++ and Java are 0.6 and 0.4, respectively. These are called the *prior* probabilities.

The total probability is the sum of the conditional probabilities that the program compiles on the first run given that it is written in a specific language, weighted by the prior probability of being written in that language in the first place.

Example 6.6

In a university department, 50% of documents are written in Word, 30% in Latex, and 20% in Html. From past experience it is known that:

40% of the Word documents exceed 10 pages;

20% of the Latex documents exceed 10 pages;

20% of the Html documents exceed 10 pages.

What is the overall proportion of documents containing more than 10 pages?

Assuming E to be the event that a document selected at random is more than 10 pages long, we may write

$$E = (\text{Word} \cap E) \cup (\text{Latex} \cap E) \cup (\text{Html} \cap E)$$

The total probability

$$\begin{aligned} P(E) &= P(\text{Word})P(E|\text{Word}) + P(\text{Latex})P(E|\text{Latex}) + P(\text{Html})P(E|\text{Html}) \\ &= (0.5 \times 0.4) + (0.3 \times 0.2) + (0.2 \times 0.2) = 0.3 \end{aligned}$$

Thirty percent of the documents exceed 10 pages. Here, the sources of the document are the three different word processors, Word, Latex, and Html, each with different prior probabilities: $P(\text{Word}) = 0.5$, $P(\text{Latex}) = 0.3$, $P(\text{Html}) = 0.2$.

The total probability is the sum of the conditional probabilities that the document exceeds 10 pages given that it written with a specific package, weighted by the prior probability of being written with that package in the first place. \triangleleft

We can generalize this approach as follows:

Theorem 6.1 Law of total probability

If a sample space S can be partitioned into k mutually exclusive and exhaustive events, $A_1, A_2, A_3, \dots, A_k$, then for any event E

$$P(E) = P(A_1)P(E|A_1) + P(A_2)P(E|A_2) + \dots + P(A_k)P(E|A_k)$$

Proof

We can write E as $E \cap S$, and since the entire sample space $S = A_1 \cup A_2 \cup \dots \cup A_k$, we have

$$\begin{aligned} E &= E \cap S \\ &= E \cap (A_1 \cup A_2 \cup \dots \cup A_k) \\ &= (E \cap A_1) \cup (E \cap A_2) \cup \dots \cup (E \cap A_k) \end{aligned}$$

The events $E \cap A_1, E \cap A_2, \dots, E \cap A_k$ are mutually exclusive because A_1, A_2, \dots, A_k are.

Hence,

$$\begin{aligned} P(E) &= P(E \cap A_1) + P(E \cap A_2) + \dots + P(E \cap A_k) \\ &= P(A_1)P(E|A_1) + P(A_2)P(E|A_2) + \dots + P(A_k)P(E|A_k) \end{aligned}$$

□

6.6 TREES

A helpful device to clarify the use of the law of total probability, especially when the number of partitions of the sample space is large, is to display the possible outcomes at each stage of the experiment as branches on a tree. A tree diagram for the compiling probabilities of programs written in C++ and Java, would look like Fig. 6.2.

The main branches on the left represent the initial or prior probabilities, and the branches emanating from these are the conditional probabilities. To obtain the total probability, work along the branches to the outcome desired, and sum the probabilities in each route.

Figure 6.2 shows that one route of the compile probabilities is 0.6×0.6 and the second route has probability 0.4×0.8 . So the total probability is

$$P(E) = (0.6 \times 0.6) + (0.4 \times 0.8)$$

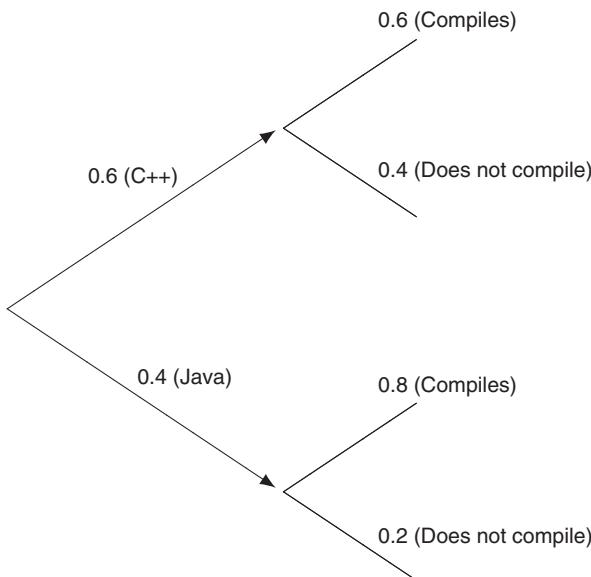


FIGURE 6.2 Compiling Probabilities of C++ and Java Programs

Similarly, we can obtain $P(\bar{E})$, the probability that a program does not compile on the first run, by working along the branches to the desired outcomes of \bar{E} the event that the program does not compile.

$$P(\bar{E}) = (0.6 \times 0.4) + (0.4 \times 0.2)$$

Let us look at another example.

Example 6.7

Inquiries to an online computer system arrive on five communication lines. The percentage of messages received through the different lines are

Line	1	2	3	4	5
Percentage received	20	30	10	15	25

From past experience, it is known that the percentage of messages exceeding 100 characters on the different lines are

Line	1	2	3	4	5
Percentage exceeding 100 characters	40	60	20	80	90

Figure 6.3 represents this problem in the form of a tree.

We calculate $P(E)$, the overall or total probability of messages exceeding 100 characters, by proceeding across each route and adding.

$$P(E) = (0.2 \times 0.4) + (0.3 \times 0.6) + (0.1 \times 0.2) + (0.15 \times 0.8) + (0.25 \times 0.9)$$

In this example, the sources are the five different lines. There are different probabilities associated with each of these lines. The proportions of programs arriving on lines 1, 2, 3, 4, and 5 are, respectively, 0.20, 0.30, 0.10, 0.15 and 0.25, which are the prior probabilities.

The total probability is the sum of the conditional probability that the message exceeds 100 characters, given that it comes through a specific line, weighted by the prior probability of coming through that line in the first place. \triangleleft

Example 6.8

A binary communication channel carries data as one of two sets of signals denoted by 0 and 1. Owing to noise, a transmitted 0 is sometimes received as a 1, and a transmitted 1 is sometimes received as a 0.

For a given channel, it can be assumed that a transmitted 0 is correctly received with probability 0.95 and a transmitted 1 is correctly received with probability 0.75.

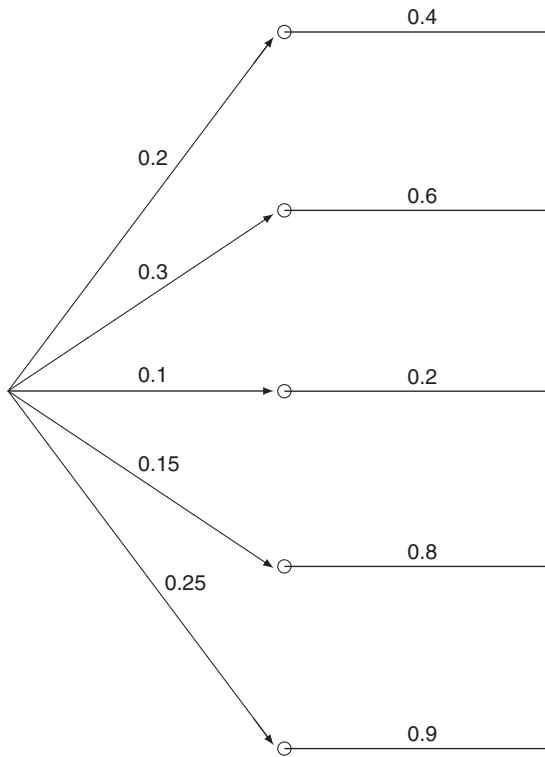


FIGURE 6.3 Inquiries to Computer System with Five Communications Lines

It is also known that 70% of all signals are transmitted as a 0. If a signal is sent, determine the probability that

- (a) a zero is received;
- (b) an error occurs.

Solution

Let R_0 be the event that a zero is received. Let T_0 be the event that a zero is transmitted.

Let R_1 be the event that a one is received. Let T_1 be the event that a one is transmitted.

For (a), the probability that a zero was received, we note

$$R_0 = (T_0 \cap R_0) \cup (T_1 \cap R_0)$$

So

$$P(R_0) = P(T_0)P(R_0|T_0) + P(T_1)P(R_0|T_1)$$

Therefore,

$$P(R_0) = (0.7 \times 0.95) + (0.3 \times 0.25) = 0.74$$

A simpler way of doing this is by examining the tree in Fig. 6.4:

To calculate the probability of receiving a zero, we trace the origins of all the zeros received, and work along the branches.

For (b), the probability that an error occurs, we note that an error occurs when a signal arrives that is different from what was transmitted, that is, a message sent as a zero arrives as a one or a message transmitted as a one arrives as a zero.

If we let E be the event of an error, then we may write

$$E = (T_0 \cap R_1) \cup (T_1 \cap R_0)$$

and

$$\begin{aligned} P(E) &= P(T_0)P(R_1|T_0) + P(T_1)P(R_0|T_1) \\ &= (0.7 \times 0.05) + (0.3 \times 0.25) = 0.11 \end{aligned}$$
▫

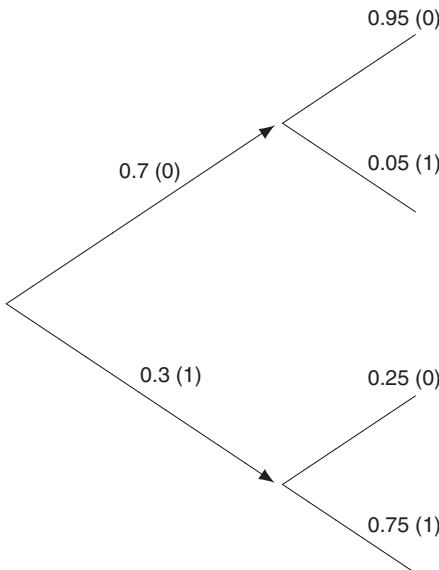


FIGURE 6.4 Transmit and Receive Probabilities of a Communication Channel

EXERCISES 6.1

- Before a software package is launched, it is subject to four independent error tests. If there is an error, the first test will find it with probability 0.4, the second test will find it with probability 0.3, and the third and fourth tests will discover it each with probability 0.2. If the program contains an error, what is the probability that it will be found?
- A survey of Android Apps bought in the App store, indicated that 50% were related to music, 30% related to games, and 20% related to movies. Twenty percent of the music apps, 40% of the game apps, and 10% of the movie apps are free. What is the proportion of free apps available in the App store?
- In Example 6.8, calculate the probability that a one is received.
- An email message can travel through one of three server routes. The percentage of errors in each of the servers and the percentage of messages that travel each route are shown in the following table. Assume that the servers are independent.

	Percentage of Messages	Percentage of Errors
Server 1	40	1
Server 2	25	2
Server 3	35	1.5

Determine the percentage of emails that contain an error.

- A geospatial analysis system has four sensors supplying images. The percentage of images supplied by each sensor and the percentage of images relevant to a query are shown in the following table.

Sensor	Percentage of Images Supplied	Percentage of Relevant Images
1	15	50
2	20	60
3	25	80
4	40	85

What is the overall percentage of relevant images?

- A fair coin is tossed twice.

Let E_1 be the event that both tosses have the same outcome, that is, $E_1 = \{HH, TT\}$.

Let E_2 be the event that the first toss is a head, that is, $E_2 = \{HH, HT\}$.

Let E_3 be the event that the second toss is a head, that is, $E_3 = \{TH, HH\}$.

Show that E_1 , E_2 , and E_3 are pairwise independent but not mutually independent.

6.7 PROJECT

In Section 6.4, we wrote a program in *R* to illustrate the Intel problem. Redo this program, and plot the number of divisions against the probabilities that at least one error will occur in the calculations. Mark on your graph the number of divisions that will make it more than 10% likely that an error will occur.

7

POSTERIOR PROBABILITY AND BAYES

In the eighteenth century, a Presbyterian minister called Thomas Bayes was attempting to prove that God existed. Although he was not successful in his efforts, he left us with a result that is of considerable use in modern-day probability and statistics. It is called *Bayes' rule*. In this chapter, we study this rule, and show how it has some important applications in computer science.

7.1 BAYES' RULE

7.1.1 Bayes' Rule for Two Events

Consider two events A and B . From the multiplication law given in Equation 6.6

$$P(A \cap B) = P(A)P(B|A)$$

and

$$P(B \cap A) = P(B)P(A|B)$$

Since $P(A \cap B) = P(B \cap A)$, then it follows that

$$P(B)P(A|B) = P(A)P(B|A)$$

Therefore,

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

This is called Bayes' rule for two events. It enables us to compute the probability of A after B has occurred.

Returning to the compiling probabilities given in Table 4.1, we might ask, what is the probability that a program has been written in C++ when we know that it has compiled in the first run?

Here, we are seeking the *posterior* probability $P(\text{C++}|E)$. We may write

$$P(\text{C++}|E) = \frac{P(\text{C++})P(E|\text{C++})}{P(E)}$$

The denominator is the total probability

$$P(E) = P(\text{C++})P(E|\text{C++}) + P(\text{Java})P(E|\text{Java})$$

So, the posterior probability of the program having been written in C++, after it has been discovered that it compiled on the first run, is

$$P(\text{C++}|E) = \frac{P(\text{C++})P(E|\text{C++})}{P(\text{C++})P(E|\text{C++}) + P(\text{Java})P(E|\text{Java})}$$

Analogously, the posterior probability of the program being written in Java, after it has been discovered to compile on the first run, is

$$P(\text{Java}|E) = \frac{P(\text{Java})P(E|\text{Java})}{P(\text{C++})P(E|\text{C++}) + P(\text{Java})P(E|\text{Java})}$$

Looking back at the compiling probabilities given in Table 4.1, a quick calculation in *R* gives

```
totalprob <- (120/200)*(72/120)+(80/200)*(64/80) # total probability
postc <- (120/200)*(72/120)/totalprob #posterior probability of C++
postc
[1] 0.5294118
postjava <- (80/200)*(64/80)/totalprob #posterior probability of Java
postjava
[1] 0.4705882
```

The posterior probabilities of 0.53 for C++ and 0.47 for Java are not the same as the prior probabilities, which are 0.60 for C++ and 0.40 for Java. What this means is that, if a program selected at random compiles on the first run, the probability that it was written in C++ decreases, while the probability that it was written in Java increases.

Example 7.1

A binary communication channel carries data as one of two sets of signals denoted by 0 and 1. Owing to noise, a transmitted 0 is sometimes received as a 1, and a transmitted 1 is sometimes received as a 0.

For a given channel, it can be assumed that a transmitted 0 is correctly received with probability 0.95 and a transmitted 1 is correctly received with probability 0.75. It is also known that 70% of all messages are transmitted as a 0.

If a signal is received as a zero, what is the probability that it was transmitted as a zero?

Solution

Let R_0 be the event that a zero is received. Let T_0 be the event that a zero is transmitted.

Let R_1 be the event that a one is received. Let T_1 be the event that a one is transmitted.

We use Bayes' rule and write

$$P(T_0|R_0) = \frac{P(T_0)P(R_0|T_0)}{P(R_0)}$$

Now,

$$P(T_0)P(R_0|T_0) = 0.7 \times 0.95$$

and

$$R_0 = (T_0 \cap R_0) \cup (T_1 \cap R_0)$$

Now,

$$P(R_0) = P(T_0)P(R_0|T_0) + P(T_1)P(R_0|T_1)$$

Therefore,

$$P(R_0) = (0.7 \times 0.95) + (0.3 \times 0.25) = 0.74$$

So,

$$P(T_0|R_0) = \frac{0.7 \times 0.95}{(0.7 \times 0.95) + (0.3 \times 0.25)} = \frac{0.665}{0.74} = 0.90$$

Similarly

$$P(T_1|R_0) = \frac{0.3 \times 0.25}{(0.7 \times 0.95) + (0.3 \times 0.25)} = \frac{0.075}{0.74} = 0.10$$

△

7.1.2 Bayes' Rule for More Than Two Events

Bayes' rule is easily generalized to k events as follows:

Theorem 7.1 Bayes' rule

If a sample space S can be partitioned into k mutually exclusive and exhaustive events A_1, A_2, \dots, A_k , then

$$P(A_i|E) = \frac{P(A_i)P(E|A_i)}{P(A_1)P(E|A_1) + P(A_2)P(E|A_2) + \dots + P(A_k)P(E|A_k)}$$

Proof

For any i , $1 \leq i \leq k$,

$$E \cap A_i = A_i \cap E$$

$$P(E)P(A_i|E) = P(A_i)P(E|A_i)$$

which implies that

$$P(A_i|E) = \frac{P(A_i)P(E|A_i)}{P(E)} \quad (7.1)$$

Using the law of total probability, we can write

$$P(E) = P(A_1)P(E|A_1) + P(A_2)P(E|A_2) + \dots + P(A_k)P(E|A_k)$$

Equation 7.1 may be written as

$$P(A_i|E) = \frac{P(A_i)P(E|A_i)}{P(A_1)P(E|A_1) + P(A_2)P(E|A_2) + \dots + P(A_k)P(E|A_k)}$$

$P(A_i|E)$ is called the posterior probability. \square

In Example 6.6, the prior probabilities are the proportion of documents that are written in Word, Latex, and Html.

50% of the documents are written in Word, that is, $P(\text{Word}) = 0.5$

30% of the documents are written in Latex, that is, $P(\text{Latex}) = 0.3$

20% of the documents are written in Html, that is, $P(\text{Html}) = 0.2$

If a document that was chosen at random was found to exceed 10 pages, we might ask, what the probability now is that it has been written in Latex.

Let E be the event that a document, chosen at random, contains more than 10 pages. The probability that it has been written in Latex is calculated using Bayes' rule.

By Bayes' rule

$$P(\text{Latex}|E) = \frac{P(\text{Latex})P(E|\text{Latex})}{P(E)}$$

By the law of total probability

$$P(E) = P(\text{Word})P(E|\text{Word}) + P(\text{Latex})P(E|\text{Latex}) + P(\text{Html})P(E|\text{Html})$$

So,

$$\begin{aligned} P(\text{Latex}|E) &= \frac{P(\text{Latex})P(E|\text{Latex})}{P(\text{Word})P(E|\text{Word}) + P(\text{Latex})P(E|\text{Latex}) + P(\text{Html})P(E|\text{Html})} \\ &= \frac{0.3 \times 0.2}{(0.5 \times 0.4) + (0.3 \times 0.2) + (0.2 \times 0.2)} = \frac{0.06}{0.3} = 0.2 \end{aligned}$$

Similarly,

$$P(\text{Word}|E) = \frac{0.5 \times 0.4}{0.3} = 0.67$$

and

$$P(\text{Html}|E) = \frac{0.2 \times 0.2}{0.3} = 0.13$$

In Example 6.7, the prior probabilities that messages came through communications lines 1–5 were 0.2, 0.3, 0.1, 0.15, and 0.25, respectively. If a message is received and exceeds 100 characters (E), we might be interested in reassessing these probabilities, $P(L_1|E)$, $P(L_2|E)$, $P(L_3|E)$, $P(L_4|E)$, and $P(L_5|E)$.

Let E be the event of the message exceeding 100 characters, and let L_1, L_2, \dots, L_5 be the events of the messages going through the different lines. We seek the posterior probabilities.

Now,

$$P(L_1|E) = \frac{P(L_1 \cap E)}{P(E)}$$

and

$$\begin{aligned} E &= (E \cap L_1) \cup (E \cap L_2) \cup (E \cap L_3) \cup (E \cap L_4) \cup (E \cap L_5) \\ P(E) &= P(E \cap L_1) + P(E \cap L_2) + P(E \cap L_3) + P(E \cap L_4) + P(E \cap L_5) \\ &= P(L_1)P(E|L_1) + P(L_2)P(E|L_2) + \dots + P(L_5)P(E|L_5) \end{aligned}$$

Therefore,

$$\begin{aligned} P(L_1|E) &= \frac{P(L_1)P(E|L_1)}{P(L_1)P(E|L_1) + P(L_2)P(E|L_2) + \dots + P(L_5)P(E|L_5)} \\ &= \frac{0.2 \times 0.4}{(0.2 \times 0.4) + (0.3 \times 0.6) + (0.1 \times 0.2) + (0.15 \times 0.8) + (0.25 \times 0.9)} \end{aligned}$$

In R,

```
totalprob <- 0.2*0.4 + 0.3*0.6 + 0.1*0.2 + 0.15*0.8 + 0.25*0.9
L1 <- (0.2 * 0.4)/totalprob
L1
[1] 0.128
```

that is, $P(L_1|E) = 0.128$

Calculating the remaining posterior probability in *R*, we have

```
L2 <- (0.3 * 0.6) /totalprob
L2
[1] 0.288

L3 <- (0.1 * 0.2) /totalprob
L3
[1] 0.032

L4 <- (0.15 * 0.8) /totalprob
L4
[1] 0.192

L5 <- (0.25 * 0.9) /totalprob
L5
[1] 0.36
```

A quick check shows that the sum of the posterior probabilities is 1.

```
L1 + L2 + L3 + L4 + L5
[1] 1
```

Let us look at how the prior and posterior probabilities differ in Example 6.7.

Lines	1	2	3	4	5
Prior probabilities	0.20	0.30	0.10	0.15	0.25
Posterior probabilities	0.126	0.288	0.032	0.192	0.36

We see that the first three of the posterior probabilities are less than the prior probabilities and, in the other two lines 4 and 5, the prior probabilities are less than the posterior probabilities. With line 4, the prior probability is 0.15 and the posterior is 0.192, whereas with line 5, the prior probability is 0.25, and the posterior is 0.36 when the message exceeds 100 characters.

Let us look again at the messages that exceed 100 characters.

Line	1	2	3	4	5
% of messages exceeding 100 characters	40	60	20	80	90

So the prior probabilities that a message exceeds 100 characters for lines 4 and 5 are 0.8 and 0.9, respectively. What the posterior probabilities are saying is that, if

a message is received that exceeds 100 characters, then the likelihood of it coming from line 4 has increased, as also has the likelihood of it coming from line 5, whereas there is a decreased likelihood of the coming from lines 1, 2, and 3.

Example 7.2 *The game problem (with apologies to Marilyn, Monty Hall and Let's Make a Deal)*

You are a contestant in a game which allows you to choose one out of three doors. One of these doors conceals a tablet computer while the other two are empty. When you have made your choice, the host of the show opens one of the remaining doors and shows you that it is one of the empty ones. You are now given the opportunity to change your door. Should you do so, or should you stick with your original choice?

The question we address is that if you change, what are your chances of winning the tablet computer?

When you randomly choose a door originally, there is a probability of $1/3$ that the computer will be behind the door that you choose, and a probability of $2/3$ that it will be behind one of the other two doors. These probabilities do not change when a door is opened and revealed empty. Therefore, if you switch you will have a probability of $2/3$ of winning the tablet computer.

Most people do not believe this, thinking instead that when one door is opened and revealed empty, the choice is now between the other two doors, and so the probability is $1/2$. The flaw in this reasoning is that the host, who can see what is behind all three doors, sometimes has a choice between 2 doors, and other times has just one empty door to reveal.

We can use conditional probability to derive the exact probability of getting a tablet computer after switching.

Let D_1 , D_2 , and D_3 be the events that the computer is behind door 1, door 2, and door 3, respectively. We assume that

$$P(D_1) = P(D_2) = P(D_3) = \frac{1}{3}$$

and hence,

$$P(\bar{D}_1) = P(\bar{D}_2) = P(\bar{D}_3) = \frac{2}{3}$$

Suppose you choose door 1.

Let E be the event that the host opens door 2 and reveals it to be empty. If you change after this, you will be choosing door 3. So we need to know $P(D_3|E)$. From Bayes' rule,

$$P(D_3|E) = \frac{P(D_3)P(E|D_3)}{P(E)}$$

Now, from the law of total probability,

$$E = (D_1 \cap E) \cup (D_2 \cap E) \cup (D_3 \cap E)$$

Therefore,

$$\begin{aligned} P(E) &= P(D_1 \cap E) + P(D_2 \cap E) + P(D_3 \cap E) \\ &= P(D_1)P(E|D_1) + P(D_2)P(E|D_2) + P(D_3)P(E|D_3) \end{aligned}$$

So we can write

$$P(D_3|E) = \frac{P(D_3)P(E|D_3)}{P(D_1)P(E|D_1) + P(D_2)P(E|D_2) + P(D_3)P(E|D_3)} \quad (7.2)$$

We need to calculate $P(E|D_1)$, $P(E|D_2)$, and $P(E|D_3)$.

$P(E|D_1)$ is the probability that the host opens door 2 and reveals it to be empty, given that it is behind door 1. When the computer is behind door 1, the door selected by you, the host has two doors to select from. We assume that the host selects one of these at random, that is, $P(E|D_1) = 1/2$.

$P(E|D_2)$ is the probability that the host opens door 2 and reveals it to be empty, given that it is behind door 2. This is an impossibility, so it has a probability of zero, that is, $P(E|D_2) = 0$.

Finally, $P(E|D_3)$ is the probability that the host opens door 2 and reveals it to be empty, given that it is behind door 3. When the computer is behind door 3, there is just one way of revealing an empty door; the host must open door 2. This is a certainty, so it has a probability of one, that is, $P(E|D_3) = 1$.

Putting these values into Equation (7.2), we have

$$P(D_3|E) = \frac{\frac{1}{3} \times 1}{\left(\frac{1}{3} \times \frac{1}{2}\right) + \left(\frac{1}{3} \times 0\right) + \left(\frac{1}{3} \times 1\right)} = \frac{2}{3}$$

So changing would increase your probability of winning the tablet computer from $1/3$ to $2/3$. You would double your chance by changing. The prior probability of winning the computer was $1/3$, while the posterior probability that the computer is behind the door you have not chosen is $2/3$. \triangleleft

When this problem was first enunciated in the TV show called “Let’s make a deal,” which was popular in the 1970s, it caused great controversy. Even today, it is still being written about and discussed extensively. A book which deals with this problem, along with other counter intuitive propositions, is *The Curious Incident of the Dog in the Night-Time*, Haddon (2003), which is worth reading.

7.2 HARDWARE FAULT DIAGNOSIS

An important application of Bayesian probabilities is in the area of fault diagnosis in hardware. When you call a manufacturer’s helpline or log in to the website, you are asked some questions, and then Bayes’ rule is applied to determine the most likely cause of your problem.

We look at a very simple example.

Example 7.3

Printer failures are associated with three types of problems: hardware, software, and electrical connections. A printer manufacturer obtained the following probabilities from a database of tests results. The probabilities of problems concerning hardware, software, or electrical connections are 0.1, 0.6, and 0.3, respectively. The probability of a printer problem, given that there is a hardware problem is 0.9, given that there is a software problem is 0.2, and given that there is an electrical problem is 0.5. If a customer seeks help to diagnose the printer failure, what is the most likely cause of the problem?

Solution

Let H , S , and E denote the events of hardware, software, or electrical problems, respectively. Then the prior probabilities are $P(H) = 0.1$, $P(S) = 0.6$, and $P(E) = 0.3$.

Let F denote the event of a printer failure. We need to know the posterior probabilities $P(H|F)$, $P(S|F)$, and $P(E|F)$, the likelihood that, given that a printer failure has been reported, it will be due to faulty hardware, software, or electrical. We calculate these probabilities using Bayes' rule.

First, we calculate the total probability of a failure occurring.

$$\begin{aligned} F &= (H \cap F) \cup (S \cap F) \cup (E \cap F) \\ P(F) &= P(H)P(F|H) + P(S)P(F|S) + P(E)P(F|E) \\ &= (0.1 \times 0.9) + (0.6 \times 0.2) + (0.3 \times 0.5) = 0.36 \end{aligned}$$

Then, the posterior probabilities are

$$\begin{aligned} P(H|F) &= \frac{P(H)P(F|H)}{P(F)} = \frac{0.1 \times 0.9}{0.36} = 0.25 \\ P(S|F) &= \frac{P(S)P(F|S)}{P(F)} = \frac{0.6 \times 0.2}{0.36} = 0.333 \\ P(E|F) &= \frac{P(E)P(F|E)}{P(F)} = \frac{0.3 \times 0.5}{0.36} = 0.417 \end{aligned}$$

Because $P(E|F)$ is the largest, the most likely cause of the problem is electrical. A help desk or website dialogue to diagnose the problem should check into the electrical problem first. \triangleleft

7.3 MACHINE LEARNING AND CLASSIFICATION

Bayes' theorem is sometimes used in supervised learning to classify items into two or more groups. A training set of examples is used to "learn" the prior and conditional probabilities of being in each group, and these probabilities are used to classify new examples.

7.3.1 Two-Case Classification

Suppose that there are two classes, $y = 1$ and $y = 2$ by which we can classify a new value of f .

By Bayes' rule, we can write

$$P(y = 1|f) = \frac{P((y = 1) \cap f)}{P(f)} = \frac{P(y = 1)P(f|y = 1)}{P(f)} \quad (7.3)$$

$$P(y = 2|f) = \frac{P((y = 2) \cap f)}{P(f)} = \frac{P(y = 2)P(f|y = 2)}{P(f)} \quad (7.4)$$

Dividing Equation 7.3 by Equation 7.4 we get

$$\frac{P(y = 1|f)}{P(y = 2|f)} = \frac{P(y = 1)P(f|y = 1)}{P(y = 2)P(f|y = 2)} \quad (7.5)$$

Our decision is to classify a new example into class 1 if

$$\frac{P(y = 1|f)}{P(y = 2|f)} > 1$$

or equivalently if

$$\frac{P(y = 1)P(f|y = 1)}{P(y = 2)P(f|y = 2)} > 1$$

which means that f goes into class 1 if

$$P(y = 1)P(f|y = 1) > P(y = 2)P(f|y = 2)$$

and f goes into class 2 if

$$P(y = 1)P(f|y = 1) < P(y = 2)P(f|y = 2)$$

When

$$P(y = 1)P(f|y = 1) = P(y = 2)P(f|y = 2)$$

the result is inconclusive.

The conditional probabilities of $P(f|y = 1)$ and $P(f|y = 2)$ are assumed to be already learned from the training set as are the prior probabilities $P(y = 1)$ and $P(y = 2)$. If the prior and conditional probabilities are accurately estimated, then the classifications will have a high probability of being correct.

Example 7.4

From a training set the prior probabilities of being in each of two classes are estimated to be $P(y = 1) = 0.4$ and $P(y = 2) = 0.6$. Also the conditional probabilities for a new

example f are $P(f|y = 1) = 0.5$ and $P(f|y = 2) = 0.3$. Into what class should you classify the new example?

$$P(y = 1)P(f|y = 1)) = 0.4 \times 0.5 = 0.20$$

and

$$P(y = 2)P(f|y = 2) = 0.6 \times 0.3 = 0.18$$

and since

$$P(y = 1)P(f|y = 1) > P(y = 2)P(f|y = 2)$$

the new example goes into class 1. \triangleleft

7.3.2 Multi-case Classification

When the number of classes, k , by which we can classify a new value f of the item is more than 2, we denote $y = 1, y = 2, \dots, y = k$, as the k classes, and use Bayes to calculate the posterior probabilities.

By Bayes' rule for any $1 \leq i \leq k$

$$P(y = i|f) = \frac{P((y = i) \cap f)}{P(f)} = \frac{P(y = i)P(f|y = i)}{P(f)}$$

Our decision is to classify into the class with the maximum posterior probability, that is, choose class j so that

$$P(y = j|f) = \max_i \frac{P(y = i)P(f|y = i)}{P(f)}, \quad 1 \leq i \leq k$$

or equivalently

$$P(y = j|f) = \max_i (P(y = i)P(f|y = i)), \quad 1 \leq i \leq k$$

since $P(f)$ is constant regardless of i .

Example 7.5

Suppose the prior probabilities of being in any of three classes have been learned from the training set and are estimated to be

$$P(y = 1) = 0.2, \quad P(y = 2) = 0.3, \quad \text{and} \quad P(y = 3) = 0.5$$

Also, the conditional probabilities for the new example f have been estimated from the training set as

$$P(f|y = 1) = 0.5, \quad P(f|y = 2) = 0.3, \quad \text{and} \quad P(f|y = 3) = 0.6$$

Into what class should you classify the new example?

Now

$$P(y = 1)P(f|y = 1)) = 0.2 \times 0.5 = 0.10$$

$$P(y = 2)P(f|y = 2) = 0.3 \times 0.3 = 0.09$$

and

$$P(y = 3)P(f|y = 3) = 0.5 \times 0.6 = 0.30$$

Therefore, the new example goes into class 3, which has the maximum posterior probability. \triangleleft

7.4 SPAM FILTERING

Email spam filtering is a way of processing incoming email messages to decide if they are spam or legitimate. Spam detection is an example of the two-case classification problem discussed in the previous section. Particular words have particular probabilities of occurring in spam email. The system does not know these words in advance and must first be trained so it can build them up. A training set of examples, where the spam and legitimate emails are known, is examined. The proportions of spam and legitimate emails found in the training set are used as estimates of the prior probabilities. Then, the spam messages are examined to see if they contain a particular word, say w , and the proportion containing w is calculated, as is the proportion of legitimate emails that contain w . These proportions of spam and legitimate emails that contain w are used as estimates of the conditional probabilities.

If we designate $y = 1$ for the spam class, and $y = 2$ for the legitimate class, then, from the training set, we get estimates of:

- the prior probabilities;

$P(y = 1)$ the proportion of spam emails

$P(y = 2)$ the proportion of legitimate emails

- the conditional probabilities;

$P(w|y = 1)$, the proportion of spam emails that contain the word w

$P(w|y = 2)$, the proportion of legitimate emails that contain the word w

For these probabilities to make sense, the training set of messages must be large and representative. With this information, we can use Bayes' theorem to estimate the posterior probabilities, that is, the probabilities that an email which is received as spam, really is spam, or is in fact legitimate. Then we choose the class with the highest posterior probability. Let us look at an example.

Example 7.6

Suppose the training set consists of 30% spam emails and 70% legitimate emails. Also, 60% of the spam emails and 25% of the legitimate emails contain the word “finance.” A new email contains the word “finance.” How should it be classified?

Solution

Denote $y = 1$ as the spam class, and $y = 2$ as the legitimate class. Then we can write the prior probabilities $P(y = 1) = 0.3$ and $P(y = 2) = 0.7$. Also, the proportion of emails in each class that contain the word “finance,” are used as estimates of the conditional probabilities. If we let w be the event that an email contains the word “finance,” then we can write the conditional probabilities as

$$P(w|y = 1) = 0.6 \text{ and } P(w|y = 2) = 0.25$$

We use Bayes’ theorem to obtain the posterior probabilities of the email being spam or legitimate, given that it contains the word.

The probability that it is spam, given that it contains the word, is

$$P(y = 1|w) = \frac{P((y = 1) \cap w)}{P(w)} = \frac{P(y = 1)P(w|y = 1)}{P(w)}$$

and the probability that it is legitimate, given that it contains the word, is

$$P(y = 2|w) = \frac{P((y = 2) \cap w)}{P(w)} = \frac{P(y = 2)P(w|y = 2)}{P(w)}$$

We choose the maximum of these, or equivalently, the maximum of

$$P(y = 1)P(w|y = 1) \text{ and } P(y = 2)P(w|y = 2)$$

since the denominators are the same. Now,

$$P(y = 1)P(w|y = 1) = 0.3 \times 0.6 = 0.18$$

and

$$P(y = 2)P(w|y = 2) = 0.7 \times 0.25 = 0.175$$

Since

$$P(y = 1)P(w|y = 1) > P(y = 2)P(w|y = 2)$$

then the new email is assigned to the class $y = 1$, that is, it is classified as spam. \blacktriangleleft

It should be pointed out that rarely is a message designated as spam or legitimate based on one word. Usually, several words and phrases are considered in order to determine the probabilities. Also, the filter adjusts the prior and conditional

probabilities in its database as new information becomes available. Spam filtering is an ongoing process.

7.5 MACHINE TRANSLATION

Bayes' rule is sometimes used to help translate text from one natural language to another.

A string of Turkish words t can be translated into a string of English words e in many different ways. Often, knowing the broader context in which t occurs may narrow the set of acceptable English translations, but even so, many acceptable translations remain. This is an example of a multiclass classification discussed in Section 7.3.

Bayes' rule can be used to make a choice between them.

To every pair (e, t) a number $P(t|e)$ is assigned which is interpreted as the probability that a translator when given e to translate will return t as the translation.

Given a string of Turkish words t , the job of the translation system is to find the string e that the native speaker had in mind, when t was produced. We seek $P(e|t)$.

From Bayes' rule

$$P(e|t) = \frac{P(e)P(t|e)}{P(t)}$$

We seek that \hat{e} to maximize $P(e|t)$, that is, choose \hat{e} so that

$$P(\hat{e}|t) = \max_e \frac{P(e)P(t|e)}{P(t)}$$

which is equivalent to $\max_e(P(e)P(t|e))$ since $P(t)$ is constant regardless of e . This is known as the *fundamental equation of machine translation*, and is usually written as

$$\hat{e} = \operatorname{argmax} P(e)P(t|e)$$

As a simple example, suppose there are three possibilities for e :

$$P(e_1) = 0.2, \quad P(e_2) = 0.5, \quad P(e_3) = 0.3$$

One of these has been translated into t . A passage t has the following conditional probabilities:

$$P(t|e_1) = 0.4, \quad P(t|e_2) = 0.2, \quad P(t|e_3) = 0.4$$

We can use Bayes' theorem to decide which is the most likely e .

$$P(e_1)P(t|e_1) = 0.2 \times 0.4 = 0.08$$

$$P(e_2)P(t|e_2) = 0.5 \times 0.2 = 0.1$$

$$P(e_3)P(t|e_3) = 0.3 \times 0.4 = 0.12$$

The best translation in this case is e_3 , since it has the highest posterior probability.

In real translation problems, there are many more than three possibilities: the conditional probability $P(t|e)$ is just an enormous table of conditional probabilities that associates a real number between 0 and 1 with every possible pairing of a Turkish passage and an English passage.

It is necessary to estimate from a training set:

- (i) the prior probabilities of the passage e ;
- (ii) the conditional probabilities of the Turkish passage given the English passage.

With the proper choices for these probabilities, translations of a high quality can be made.

The essential question of machine translation is an empirical one: can approximations to the distributions $P(e)$ and $P(t|e)$ be found that are good enough to achieve an acceptable quality of translation. If so, then Bayes' rule will provide the best translation.

EXERCISES 7.1

Use R as appropriate in the following exercises.

1. Obtain the posterior probabilities in Example 6.7, if a message has been received which does not exceed 100 characters.
2. A binary communication channel carries data as one of two sets of signals denoted by 0 and 1. Owing to noise, a transmitted 0 is sometimes received as a 1, and a transmitted 1 is sometimes received as a 0. For a given channel, it can be assumed that a transmitted 0 is correctly received with probability 0.95, and a transmitted 1 is correctly received with probability 0.75. Also, 70% of all messages are transmitted as a 0. If a signal is sent, determine the probability that:
 - (a) a 1 was received;
 - (b) a 1 was transmitted given than a 1 was received.
3. Look again at the game show problem in Example 7.2. This time, assume you choose the second door, and the host opens door 3 and reveals it to be empty. Derive the posterior probability that the tablet computer is behind door 1, that is, the probability of winning the tablet computer if you change from your original choice of door 2.
4. A batch of transistors contains 7% defectives. Each transistor is subjected to a test that correctly identifies a defective, but also misidentifies as defective about 10% of the nondefective transistors. Given that a randomly chosen transistor is declared defective by the tester, compute the probability that it is actually defective.

5. The 10 most common passwords in 2018 were found to be

123456	2.79%
password	1.78%
123456789	0.82%
12345678	0.76%
12345	0.63%
111111	0.41%
1234567	0.39%
sunshine	0.10%
qwerty	0.11%
iloveyou	0.10%

- (a) What is the proportion of users with one of these passwords?
- (b) If a hacker knows that your password is on this list and contains eight characters, what is the probability yours is selected?
6. An email message can travel through one of three server routes. The percentage of errors in each of the servers and the percentage of messages that travel through each route are shown in the following table. Assume that the servers are independent.
- | | Percentage of Messages | Percentage of Errors |
|----------|------------------------|----------------------|
| Server 1 | 40 | 1 |
| Server 2 | 25 | 2 |
| Server 3 | 35 | 1.5 |
- (a) What is the probability of receiving an email containing an error?
- (b) What is the probability that a message will arrive without error?
- (c) If a message arrives without error, what is the probability that it was sent through server 1?
7. There are three employees working at an IT company: Jane, Amy, and Ava, doing 10%, 30%, and 60% of the programming, respectively. 8% of Jane's work, 5% of Amy's work, and just 1% of Ava's work is in error. What is the overall percentage of error? If a program is found with an error, who is the most likely person to have written it?
8. A geospatial analysis system has four sensors supplying images. The percentage of images supplied by each sensor and the percentage of images relevant to a query are shown in the following table.

Sensor	Percentage of Images Supplied	Percentage of Relevant Images
1	15	50
2	20	60
3	25	80
4	40	85

If an image is selected and found to be relevant, what is the probability that it came from (i) sensor 1, (ii) sensor 2, (iii) sensor 3, (iv) sensor 4?

9. A software company surveyed managers to determine the probability that they would buy a new graphics package that includes three-dimensional graphics. About 20% of office managers were certain that they would not buy the package, 70% claimed that they would buy, and the others were undecided. Of those who said that they would not buy the package, only 10% said that they were interested in upgrading their computer hardware. Of those interested in buying the graphics package, 40% were also interested in upgrading their computer hardware. Of the undecided, 20% were interested in upgrading their computer hardware.

Let A denote the intention of not buying, B the intention of buying, C the undecided, and G the intention of upgrading the computer hardware.

- (a) Calculate the probability that a manager chosen at random will not upgrade the computer hardware ($P(\bar{G})$).
 - (b) Explain what is meant by the posterior probability of B given G , $P(B|G)$.
 - (c) Construct a tree diagram and use it to calculate the following probabilities: $P(G)$, $P(B|\bar{G})$, $P(B|G)$, $P(C|G)$, $P(\bar{C}|\bar{G})$.
10. An email login system uses six-character passwords, and each character is one of the 26 letters (a–z) or the 10 digits (0–9). Uppercase letters are not used.
- (a) Suppose a hacker selects a password at random. What is the probability that your password is selected?
 - (b) Suppose a hacker knows your password contains only letters and selects a set of six letters at random. What is the probability that your password is selected?
 - (c) Suppose a hacker has seen you type in the first and last characters in your password. What is the probability that she will get the other four characters correct?
11. A warranty period of one year is given with a computer laptop. From past experience, it is known that in 20% of cases, the electric connectors are not kept dry during the warranty period. The probability is 5% that, when these electric connectors are not kept dry, the laptop will fail during the warranty period. On the other hand, if the connector is kept dry, the chance that the laptop will fail is just 1%.

- (a) What proportion of connectors fail during the warranty period?
 - (b) If the connector is presented which has failed during the warranty period, what is the probability that it has not been kept dry?
12. A manufacturer of smartphones buys integrated circuits from suppliers A, B, and C; 50% from A, 30% from B, and 20% from C. In the past, 1% of circuits supplied by A, 3% supplied by B, and 4% supplied by C have been defective. A circuit is selected at random and found to be defective. What is the probability that it came from supplier B?
13. A malicious spyware can infect a computer system through the Internet or through email. The spyware comes through the Internet 70% of the time and 30% of the time, it gets in through email. If it enters via the Internet the anti-virus detector will detect it with probability 0.6, and via email, it is detected with probability 0.8.
- (a) What is the probability that this spyware infects the system?
 - (b) If the spyware is detected, what is the probability that it came through the Internet?
14. Of the emails you receive 20% are spam on average. Your spam filter is able to detect 90% of them but also misclassifies as spam 15% of the genuine emails.
- (a) If an email arrives and is marked spam, what is the probability that it really is spam?
 - (b) If an email arrives and is not marked spam, what is the probability that it is legitimate?
15. A large set of emails are examined. It was found that 40% of these were spam. 75% of the spam emails contain the word “money,” and 50% of the legitimate emails contain the word “money.” If a new email arrives and contains the word “money,” what is its most likely designation?

REFERENCE

Haddon, M. (2003), *The Curious Incident of the Dog in the Night-Time*, Jonathan Cape.

8

RELIABILITY

Reliability theory is concerned with the probability that a system will function. When such a system consists of separate components, which may or may not function independently then the reliability of the system depends in various ways on the reliability of these components. Systems with connected components have been proposed for reliability evaluation in the design of integrated circuits, microwave technology, telecommunications, pipeline systems, computer ring networks, and spacecraft relay systems. Such systems are characterized by logical or physical connections among components. A computer, for example, is comprised of hundreds of systems, each of which may have hundreds or thousands of components. The components are, of course, subject to possible failure, and these failures may cause individual systems to fail, and ultimately cause the entire system to fail. Whether or not a system is functioning is determined solely by the knowledge of which of its components are functioning. There are various types of configurations of the components in different systems. In this chapter, we look at some of these configurations and their reliabilities.

8.1 SERIES SYSTEMS

Definition 8.1 *Series systems*

A series system is a system in which all the components are in series so that the system will function if and only if all of the components of the system are functioning. If just one component of a series system fails, then the whole system fails. \square

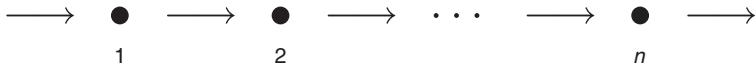


FIGURE 8.1 A Series System

The structure/topology of a series system with n components is illustrated in Fig. 8.1.

8.2 PARALLEL SYSTEMS

Definition 8.2 *Parallel systems*

A parallel system is a system in which the components are in parallel, and the system will work provided at least one of these components works. It will fail only if all the components fail. \square

The structure of a parallel system with n components is illustrated in Fig. 8.2.

Unlike systems in a series, systems with a parallel structure offer built-in redundancy in that some of the components may fail without affecting the successful functioning of the system.

8.3 RELIABILITY OF A SYSTEM

In general, we refer to the reliability of a component as the probability that the component will function for a given period of time. The reliability of the system depends on the reliability of its component parts and the manner in which they are connected. We consider systems that have components connected in series, in parallel, and a combination of both.

8.3.1 Reliability of a Series System

Suppose the system contains n components in series as in Fig. 8.1. Suppose also that the components operate independently, in that the failure or non-failure of one does not affect the operation of another.

Let p_i be the reliability of the i th component, that is, the probability that the i th component functions for some fixed time period.

$$\text{Rel}(i) = p_i = P(\text{the } i\text{th component survives throughout this time period})$$

For the present we assume that p_i is known. Later in Chapter 16, we will show how it can be estimated.

Since series systems work only if all the components work, the reliability of the system containing n components in series is the product of the individual component reliabilities.

$$\text{Rel} = p_1 p_2 \cdots p_n$$

Here, we have used the multiplication law of probability for independent events.

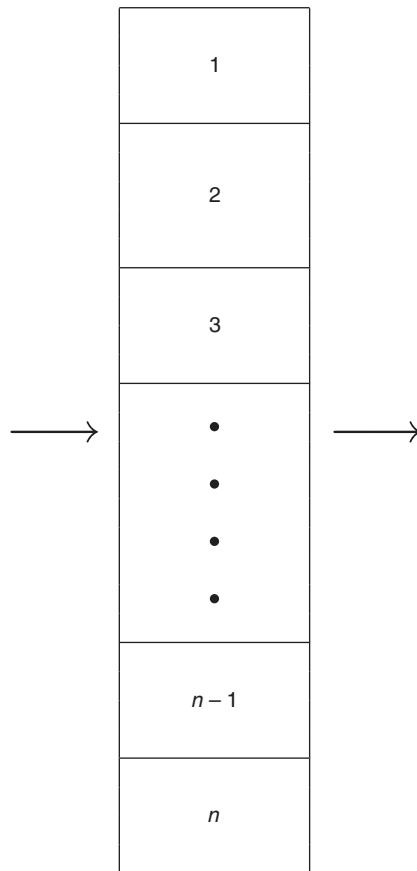
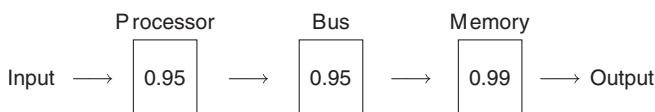


FIGURE 8.2 A Parallel System

Example 8.1

A simple computer consists of a processor, a bus, and a memory. The computer will work only if all three are functioning correctly. The probability that the processor is functioning is 0.95, that the bus is functioning 0.95, and that the memory is functioning is 0.99.

This can be viewed as a series interconnection of three components.



The probability that the computer will work is

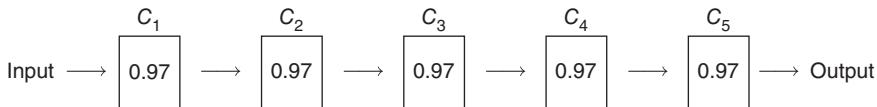
$$\text{Rel} = 0.95 \times 0.95 \times 0.99 = 0.893475$$

△

So, even though each component has a reliability of 95% or more, the overall reliability of the computer is less than 90%.

Example 8.2

A system consists of five components in series, each having a reliability of 0.97. What is the reliability of the system?



As the five components are in series, the system will function if and only if all of the five components function.

$$\text{Rel} = (0.97)^5 = 0.86 \quad \triangleleft$$

The problem with series systems is that the reliability quickly decreases as the number of components increases. In this example, we see that while all of the components have a high reliability of 0.97, the series system reliability is just 0.86.

With six components in series, each with reliability 0.97, the reliability of the system is

$$\text{Rel} = (0.97)^6 = 0.83$$

For seven components, the reliability of a series systems is

$$\text{Rel} = (0.97)^7 = 0.81$$

This means that when seven components, each with a reliability of 0.97, are connected in series, the reliability of the system is just 0.81, or equivalently, there is nearly a 20% chance that the system will fail. This example illustrates that, even with very high component reliability, the overall reliability of the series system will be much lower, depending on the number of components.

Figure 8.3 contains the reliability of series systems with differing numbers of components; the components in the first graph have reliability $p = 0.7$, in the second $p = 0.8$, in the third $p = 0.9$, and in the final $p = 0.99$.

We see from Fig. 8.3, that, with component reliability $p = 0.7$, the system reliability is as low as 0.2 with just five components. This means, that if there are five components in series, each with reliability of 0.7, the chance that the series system will fail is almost 80%.

Similarly, with $p = 0.8$, the reliability is about 30% when the number of components is five, that is, there is a 70% chance that the system will fail when it contains five components in series.

Even when the component reliability of 0.9, the system reliability is less than 0.6 when the number of components is five.

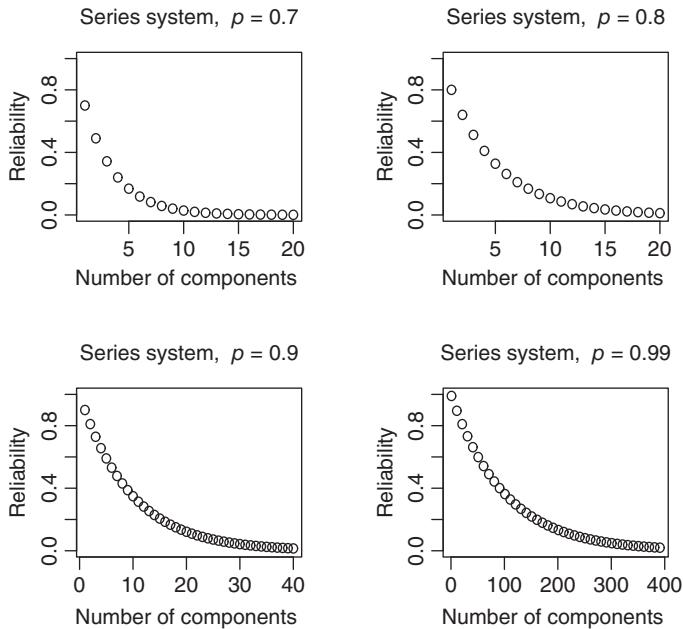


FIGURE 8.3 Reliability of a Series System with Differing Component Reliability p , for Increasing Numbers of Components

Finally, when the component reliability is 0.99, the system reliability decreases more slowly as the number of components increases. Eventually however, the system reliability drops to dangerously low levels. Here, with 100 components, the system reliability is less than 0.4.

Figure 8.3 has been generated using the following *R* code.

```
par(mfrow = c(2,2))

k <- 1:20 # Number of Components
p <- 0.7
plot (k, p^k, xlab = "Number of components ", ylab = "Reliability",
      ylim = c(0, 1), main = "Series system, p = 0.7", font.main = 1)

p <- 0.8
plot (k, p^k, xlab = "Number of components ", ylab = "Reliability",
      ylim = c(0, 1), main = "Series system, p = 0.8", font.main = 1)

k <- 1:40 # Number of Components
p <- 0.9
plot (k, p^k, xlab = "Number of components ", ylab = "Reliability",
      ylim = c(0, 1), main = "Series system, p = 0.9", font.main = 1)
```

```

k <- seq(1, 400, 10) # Number of Components 1, 11, 21, and so on.
p <- 0.99
plot (k, p^k, xlab = "Number of components ", ylab = "Reliability",
      ylim = c(0, 1), main = "Series system, p = 0.99", font.main = 1)

```

Example 8.3

An electronic product contains 100 integrated circuits. The probability that any integrated circuit is defective is 0.001. The integrated circuits are independent. The product operates only if all the integrated circuits are operational. What is the probability that the product is operational?

Solution

Each component has a probability of 0.999 of functioning. Since the product operates only if all 100 components are operational, the probability that the 100 components are functioning is

$$\text{Rel} = (0.999)^{100}$$

obtained in R with

```

0.999**100
[1] 0.9047921

```

The reliability is just over 90%, even though each component has a reliability of 99.9%. \triangleleft

Bearing in mind that computing and electrical systems have hundreds or thousands of components, a series formation on its own will never be sufficiently reliable, no matter how high the individual component reliability is. The components need to be backed up in parallel.

8.3.2 Reliability of a Parallel System

Systems with parallel structure have built-in redundancy. Components are backed up, and the system will work even if some of these cease to function. A parallel system is depicted in Fig. 8.2. If it is assumed that the system works if just one of the n components works, the reliability is

$$\text{Rel} = P(\text{at least one of the components is working})$$

Adopting the complementary approach,

$$\text{Rel} = 1 - P(\text{all components have failed})$$

$$P(\text{all components fail}) = P((1\text{st fails}) \cap (2\text{nd fails}) \cap \dots \cap (n\text{th fails}))$$

Assuming again that the components are independent, the probability that they all fail is, by the multiplication law,

$$P(\text{all components fail}) = P(\text{1st fails})P(\text{2nd fails}) \cdots P(\text{nth fails})$$

Now, the probability that the i th component fails is $1 - p_i$. So,

$$P(\text{all components fail}) = (1 - p_1) \times (1 - p_2) \times \cdots \times (1 - p_n) = \prod_{i=1}^n (1 - p_i)$$

Therefore, the reliability of a parallel system is

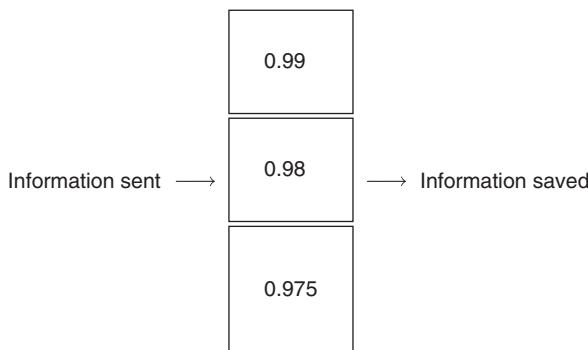
$$\text{Rel} = 1 - \prod_{i=1}^n (1 - p_i)$$

Example 8.4

A hard drive has a 1% chance of crashing. To reduce the probability of losing the stored information, two backups are used, one with a reliability of 98% and the other with a reliability of 97.5%. All three hard drives are independent of each other. The information is lost only if the three drives crash. Calculate the probability that the information is not lost.

Solution

This is a parallel system, where the information is stored on the three hard disks, and the information will be lost if an only if all three disks fail.



Since the components are in parallel, the information is saved provided at least one of the three hard drives does not crash.

$$\text{Rel} = P(\text{at least one hard drive does not crash})$$

Taking the complementary approach,

$$P(\text{at least one hard drive functioning}) = 1 - P(\text{all hard drives crash})$$

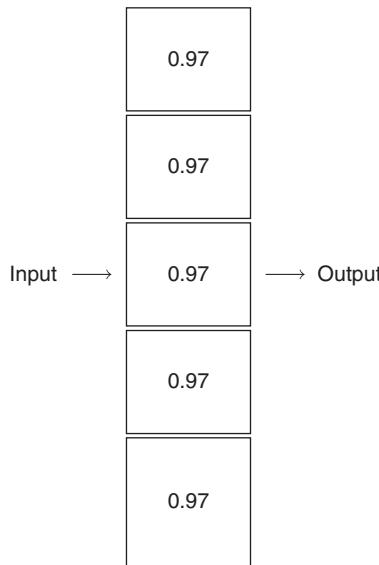
Therefore,

$$\text{Rel} = 1 - (0.01 \times 0.02 \times 0.025) = 0.999995$$

△

Example 8.5

A system consists of five components in parallel. If each component has a reliability of 0.97, what is the overall reliability of the system?



Since the components are in parallel, the system will function provided at least one of the five components works.

$$\text{Rel} = P(\text{at least one component is functioning})$$

Taking the complementary approach,

$$P(\text{at least one component functioning}) = 1 - P(\text{all components fail})$$

Therefore,

$$\text{Rel} = 1 - (0.03)^5 \approx 1.00000$$

△

The problem with parallel systems is that the “law of diminishing returns” operates. The rate of increase in reliability with each additional component decreases as the number of components increases. In Example 8.5, it may have been unnecessary to have five components since the individual components have such a high reliability. Let us investigate the reliability of the system with fewer components.

With two components, $\text{Rel} = 1 - (0.03)^2 = 0.9991$

With three components, $\text{Rel} = 1 - (0.03)^3 = 0.999973$

With four components, $\text{Rel} = 1 - (0.03)^4 = 0.9999992$.

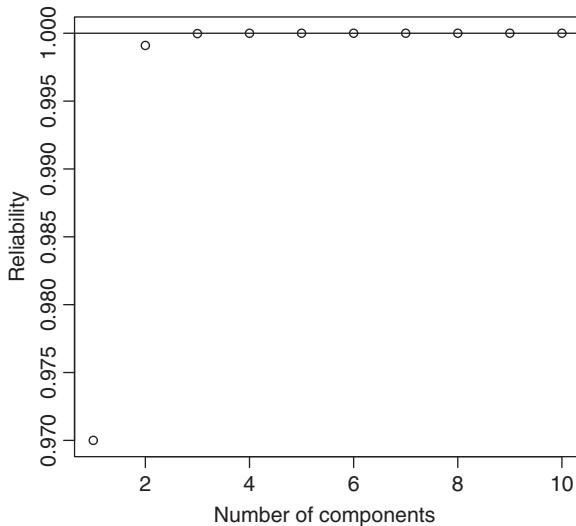


FIGURE 8.4 Reliability of a Parallel System with Increasing Number of Components: Component Reliability = 0.97

You will agree that, in this system, there is not much to be gained, in terms of increased reliability, with more than three components. We can see this more clearly using R .

```
k <- 1:10
p <- 0.97
plot (k, 1-(1-p)^k , xlab = "Number of components",
      ylab = "Reliability")
abline(1,0)
```

gives Fig. 8.4.

The horizontal line in Fig. 8.4 represents the 100% reliability mark, and you can see that this is approximately achieved with three components; so attaching any more than three components in parallel is unnecessary in this case.

8.4 SERIES-PARALLEL SYSTEMS

Most systems are combinations of series and parallel systems, where some of the components in series are replicated in parallel.

Example 8.6

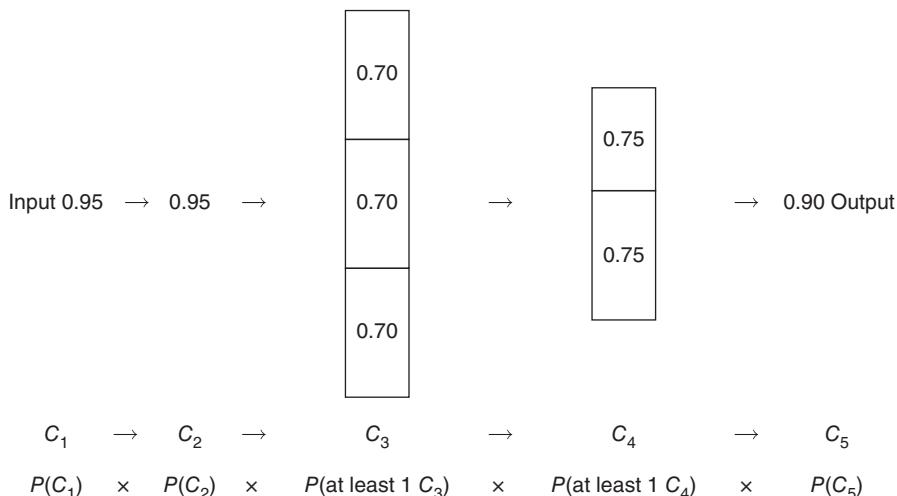
Consider a system with five kinds of component, with reliabilities as follows:

- Component 1: 0.95
- Component 2: 0.95

- Component 3: 0.70
- Component 4: 0.75
- Component 5: 0.90

Because of the low reliability of the third and fourth components, they are replicated; the system contains three of the third component and two of the fourth component.

The system



using as the convention that “ C_i ” means “ C_i works.”

Clearly,

$$P(C_1) = 0.95$$

$$P(C_2) = 0.95$$

Since C_3 is backed up so that there are three of them, then to “get through,” just one of them needs to work.

$$P(\text{at least 1 } C_3) = 1 - P(0 \text{ } C_3)$$

Since $P(C_3) = 0.70$, then the probability that it does not work is 0.3.

Therefore,

$$P(0 \text{ } C_3) = (0.3)^3$$

and so

$$P(\text{at least one } C_3) = 1 - (0.3)^3$$

Similarly, as there are two C_4 components, each of which has a reliability of 0.75,
 $P(\text{at least one of } C_4) = 1 - P(0 \text{ } C_4) = 1 - (0.25)^2$

Finally, $P(C_5) = 0.9$

Since the components are independent, we can use the multiplication law to get the system reliability.

$$\text{Rel} = P(C_1) \times P(C_2) \times P(\text{at least one } C_3) \times P(\text{at least one } C_4) \times P(C_5)$$

Therefore,

$$\text{Rel} = 0.95 \times 0.95 \times (1 - (0.30)^3) \times (1 - (0.25)^2) \times 0.90$$

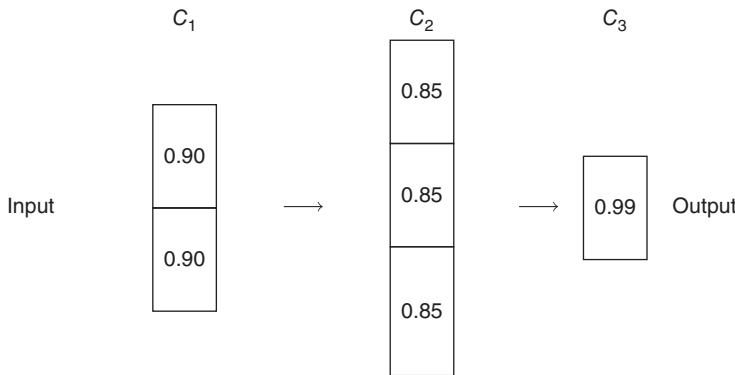
which can be calculated in *R* with

```
0.95 * 0.95 * (1 - 0.3^3) * (1 - 0.25^2) * 0.9
[1] 0.7409243
```

The reliability of the system is just over 74%, which means that there is a 26% chance that the system will fail. \triangleleft

Example 8.7

The following system operates only if there is a path of functional devices from left to right. The probability that each device functions is shown on the graph. Assume the devices fail independently.



$$\begin{aligned} \text{Reliability: } & P(\text{at least 1 } C_1) \times P(\text{at least 2 } C_2) \times P(C_3) \\ & 1 - (0.1)^2 \times 1 - (0.15)^3 \times 0.99 \end{aligned}$$

In *R*

```
(1 - 0.1^2) * (1 - 0.15^3) * 0.99
[1] 0.9767922
```

This system has a reliability of nearly 98%. \triangleleft

8.5 THE DESIGN OF SYSTEMS

Systems may be designed so that reliabilities are at a certain level. The design of a system may involve deciding on the maximum number of components that may need to be aligned in series in order to maintain a given reliability. Also, the design of a system may involve deciding on the minimum number of components to align in parallel to achieve a given reliability. For example, how much redundancy is needed to ensure a given reliability? If there is too much redundancy, the system is not cost effective, and if there is not enough, the system will not meet the specifications.

Example 8.8

A series system consists of components each with a reliability of 0.99. What is the maximum number of these components that may be put in series to ensure that the systems reliability is at least 90%?

Solution

Algebraically, we need k to be small enough so that

$$\text{Rel} = (0.99)^k \geq 0.9$$

We can examine the reliability with differing numbers of components using R .

```
k <- 1:20
plot(k, 0.99^k, xlab = "Number of components in series",
      ylab = "System reliability")
abline(0.9, 0)
```

gives Fig. 8.5.

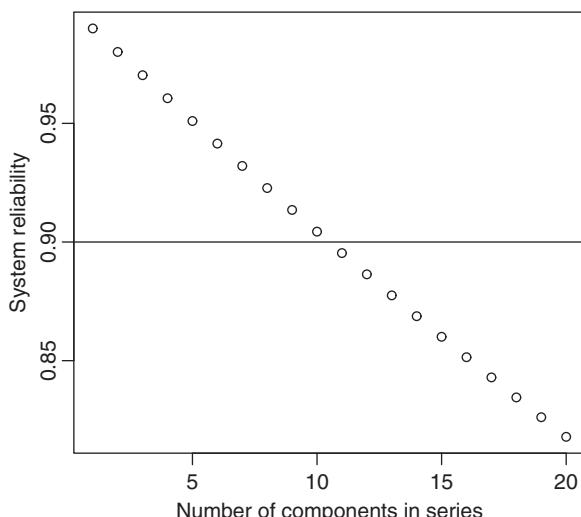


FIGURE 8.5 Reliability of a Series System with k Components each with Reliability 0.9

Figure 8.5 shows how the reliability decreases with increasing numbers of components in series. We can also see that no more than 10 components in series are allowed if we are to maintain a reliability level of 90%; any more would bring the reliability below 0.9. \triangleleft

Example 8.9

A component with a reliability of 0.75 is to be backed up in parallel. What is the minimum number of components needed to ensure that the system has more than 99% reliability?

Solution

The probability that any component will not function is $1 - 0.75 = 0.25$. With k components, the reliability of the system is the probability that at least one of the components will function.

$$\text{Rel} = P(\text{At least one component functions}) = 1 - P(\text{all components fail})$$

that is,

$$\text{Rel} = 1 - (0.25)^k$$

We must choose k so that

$$\text{Rel} = 1 - (0.25)^k \geq 0.99$$

This equation can be solved algebraically. Our approach is to invoke *R* to examine Rel for various values of k components.

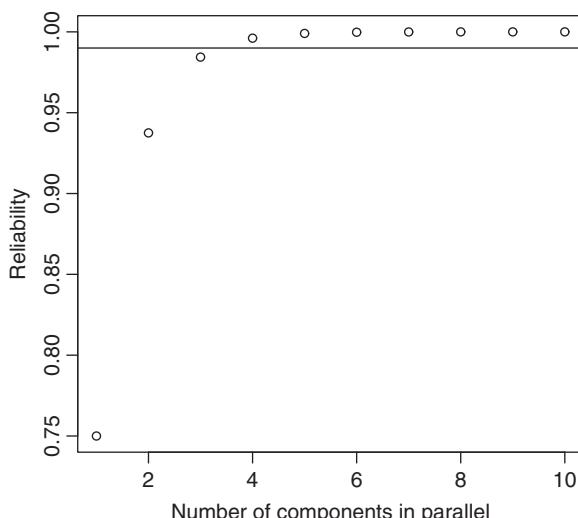


FIGURE 8.6 Reliability of a Parallel System with k Components

```

k <- seq(1:10)
r <- 1 - 0.25^k
plot(k, r, xlab = "Number of components in parallel",
      ylab = "Reliability")
abline(0.99, 0)

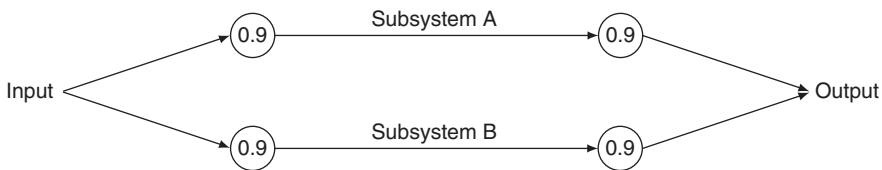
```

gives Fig. 8.6.

Figure 8.6 shows that it is necessary to have four components in parallel to achieve a reliability level of 0.99; any fewer would not meet the specifications, while any more would be unnecessarily large. Also, notice that there is not much to be gained from having more than five components in parallel. \triangleleft

Example 8.10

The following diagram is a representation of a system that comprises two subsystems operating in parallel. Each subsystem has two components that operate in series. The system will operate properly as long as at least one of the subsystem functions properly. The probability of failure of each component is 0.1. Assume that the components operate independently of each other.



How many subsystems in parallel, like the two shown, would be required to guarantee that the system would operate properly at least 99% of the time?

Solution

Subsystem A is a series system with two components. It will operate only if both components operate. The probability that both components are working is $(0.9)^2$. Since these components are in series, the system fails if any one of these fails.

$$P(\text{Subsystem A fails}) = P(\text{at least one component fails}) = 1 - P(\text{both work})$$

$$P(\text{Subsystem A fails}) = 1 - (0.9)^2$$

Similarly, for Subsystem B,

$$P(\text{Subsystem B fails}) = 1 - (0.9)^2$$

Since these subsystems are arranged in parallel, the overall system will operate if at least one of these subsystems works.

$$\text{Rel} = P(\text{at least one subsystem works})$$

Equivalently,

$$\text{Rel} = 1 - P(\text{both subsystems fail})$$

$$\text{Rel} = 1 - (1 - (0.9)^2)^2 = 0.9639$$

So the overall reliability of the system is 96.39%. If we want to increase the reliability to 99.9%, we need to add some more subsystems.

Generally with k subsystems, the reliability is

$$\text{Rel} = 1 - (1 - (0.9)^2)^k$$

We can examine this with different values of k .

```
k <- 1:10
plot(k, 1 - (1 - 0.9^2)^k, xlab = "Number of subsystems",
      ylab = "Reliability")
abline(0.99, 0)
```

gives Fig. 8.7, which is the reliability for various levels of backup.

Observe from Fig. 8.7 that three systems working in parallel will give a reliability of over 99%.

Figure 8.7 also shows that there is little to be gained from having more than four subsystems in parallel, since at this stage the reliability is practically 100%. More than four systems would provide unnecessary redundancy, possibly increasing cost. \triangle

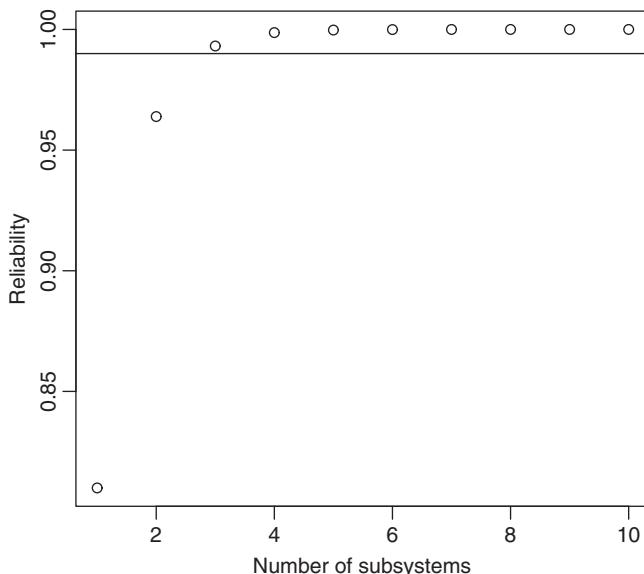


FIGURE 8.7 Reliability of k Subsystems in Series with Component Reliability of 0.9

Example 8.11

Suppose the reliability of the components in the subsystems in Example 8.10 is just 0.7. How many parallel subsystems would be required to guarantee that the system would operate properly at least 99% of the time?

Here we would expect to require more subsystems than the minimum of three required in the last example because now the component reliability is 0.7, whereas previously it was 0.9.

As before, we can deduce that with k subsystems, the reliability is

$$\text{Rel} = 1 - (1 - (0.7)^2)^k$$

We can again use *R* to examine the reliability with different values of k .

```
k <- 1:10
plot(k, 1 - (1 - 0.7^2)^k, xlab = "Number of Subsystems",
      ylab = "Reliability")
abline(0.99, 0)
```

gives Fig. 8.8.

From Fig. 8.8, we see that more than six subsystems in parallel are needed to obtain a sufficiently high reliability for the overall system. For example with four subsystems, the reliability is 0.92, and it would take seven subsystems to bring the reliability

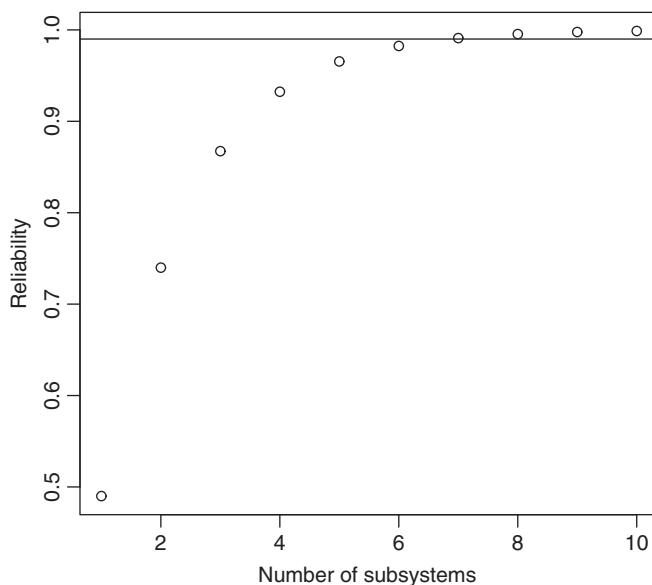


FIGURE 8.8 Reliability of k Subsystems in Series with Component Reliability of 0.7

to 0.99. Observe also in Fig. 8.8, that the reliability rises steeply with the addition of the earlier systems, but the rate of improvement decreases when six subsystems have been added. \triangleleft

8.6 THE GENERAL SYSTEM

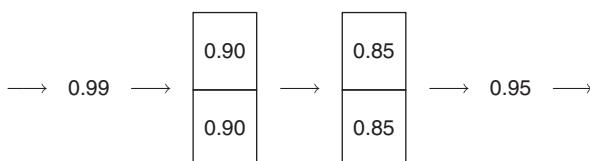
A generalization of the series and parallel systems is to consider consecutive k out of n systems. This means that the components are linearly connected in such a way that the system fails if and only if at least k consecutive components fail. This type of structure is called “ k out of n : F ” system, denoted by

$$\text{Con}/k/n : F$$

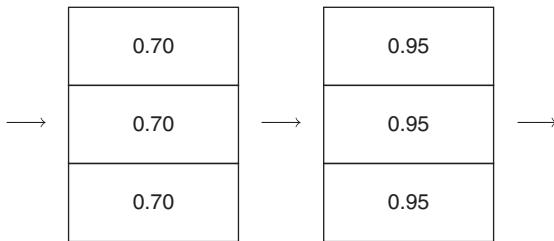
This strategy has often been proposed for system reliability evaluation in the design of integrated circuits. The consecutive k out of n system includes series system when $k = 1$ and parallel systems with $k = n$. We shall return to this in Chapter 11.

EXERCISES 8.1

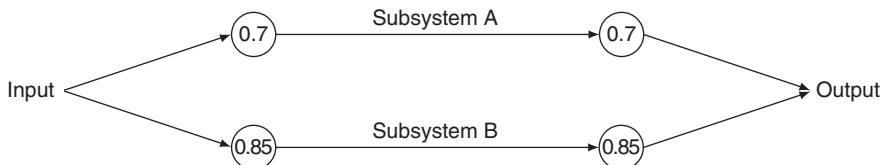
1. A robotic tool is given a one-year guarantee after purchase. The tool contains four components each of which has a probability of 0.01 of failing during the warranty period. The tool fails if any one of the components fails. Assuming that the components fail independently, what is the probability that the tool will fail during the warranty period?
2. Hard drives on a computing system have a 5% chance of crashing. Therefore, three backups have been installed. The system will operate if at least one of the hard drives is working. Calculate the reliability of the system, that is, the probability that the system is operational.
3. The following system operates only if there is a path of functional devices from left to right. The probability that each device functions is shown on the graph. Assume the devices are independent. Calculate the reliability of the system.



4. The following diagram is a representation of a system that comprises two parallel systems in series.



- (a) Calculate the reliability of the overall system.
 - (b) As you can see in the first parallel system, the component reliability is 0.7. To improve the overall reliability you decide that more components in parallel are needed. Use R to decide how many more components should be added to bring this gateway up to a reliability level of 99%, and calculate the reliability of the system, when the components are increased.
5. The following diagram is a representation of a system that comprises two subsystems operating in parallel. Each subsystem has two components that operate in series. The system will operate as long as at least one of the subsystems functions. The reliability of the components in the first subsystem is 0.7, and the reliability of the components in the second system is 0.85. Assume that the components operate independently of each other:



- (a) Calculate the reliability of the system.
- (b) What is the probability that exactly one subsystem fails?
- (c) How many parallel Subsystems A would be required to guarantee that the system would operate at least 95% of the time?
- (d) How many parallel Subsystems B would be required to guarantee that the system would operate at least 95% of the time?
- (e) Use R to investigate a possible design so that the overall reliability is 0.99, that is, how many Subsystems A or B or both should be added?

PART III

DISCRETE DISTRIBUTIONS

9

INTRODUCTION TO DISCRETE DISTRIBUTIONS

In this chapter, we introduce the concepts of discrete random variables and expectation and go on to develop basic techniques for examining them by simulation in *R*. Along the way, we introduce the Bernoulli and uniform discrete distributions. We end the chapter with an introduction to discrete bivariate distributions.

9.1 DISCRETE RANDOM VARIABLES

A random variable is a rule that assigns a numerical value to each possible outcome of an experiment, that is, a mapping from the sample space S to the number line. It is said to be discrete if its values assume integer points on the number line, that is, the outcomes are finite or countably infinite. Discrete random variables are in effect numerical representations of discrete sample spaces.

Definition 9.1 *Discrete distributions*

The set of all possible values of a discrete variable, together with their associated probabilities, is called a discrete probability distribution. □

Let us look at some of the examples considered in Chapter 4 and show how to obtain their probability distributions.

Example 9.1 *Toss a fair coin*

Let $X = 1$ if a head occurs, and $X = 0$ if a tail occurs. Since the coin is fair, $P(X = 1) = P(X = 0) = 1/2$. The distribution is

Outcome (x)	0	1
Probability ($P(X = x)$)	0.5	0.5

△

Example 9.2 *Tossing two fair coins*

Let X be the number of heads. Then the values of X are 0, 1, or 2. The sample space consists of the following equally likely outcomes. $\{(T, T), (H, T), (T, H), \text{ and } (H, H)\}$. So $P(X = 0) = 1/4$, $P(X = 1) = 2/4$, and $P(X = 2) = 1/4$. The distribution is

Outcome (x)	0	1	2
Probability ($P(X = x)$)	0.25	0.5	0.25

△

Example 9.3 *Draw a card from a deck*

Let X equal to 1 if a spade is drawn and 0 otherwise. $P(X = 1) = 13/52$ and $P(X = 0) = 39/52$. The distribution is

Outcome (x)	0	1
Probability ($P(X = x)$)	0.75	0.25

△

Example 9.4 *Roll a fair die*

Let X equal the face number, then $X = 1, 2, 3, 4, 5, 6$. $P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = P(X = 5) = P(X = 6) = 1/6$.

Outcome (x)	1	2	3	4	5	6
Probability ($P(X = x)$)	1/6	1/6	1/6	1/6	1/6	1/6

△

Notice that, in all the above examples, the probabilities sum to one.

In these experiments, tossing a fair coin, rolling a fair die, pulling cards from a well-shuffled deck, the probabilities are known before carrying out the experiment. This is not always the case. Sometimes, to obtain the probabilities, we may need to adopt what is known as a relative frequency approach. Indeed, we may ask, how do we know that a particular coin or die is “fair?” Let us look at another example.

Example 9.5

In order to obtain a model for hardware failures in a computer system, the number of crashes occurring each week was observed over a period of one year. It was found that

- 0 failures occurred in each of 9 weeks
- 1 failure occurred in each of 14 weeks
- 2 failures occurred in each of 13 weeks
- 3 failures occurred in each of 9 weeks
- 4 failures occurred in each of 4 weeks
- 5 failures occurred in each of 2 weeks
- 6 failures occurred in 1 week.

By obtaining the relative frequency of the occurrence of hardware failures, we can estimate the probabilities. Use *R*

```
weeks <- c(9, 14, 13, 9, 4, 2, 1)
sum(weeks)
[1] 52
probability <- weeks/52
probability
[1] 0.17307692 0.26923077 0.25000000 0.17307692 0.07692308 0.03846154 0.01923077
round((probability, 2) #rounding the probabilities to two decimal places
[1] 0.17 0.27 0.25 0.17 0.08 0.04 0.02
```

The probability distribution of the hardware failures in a week is approximated to be

Number of failures (x)	0	1	2	3	4	5	6
Probability ($P(X = x)$)	0.17	0.27	0.25	0.17	0.08	0.04	0.02

In this case, we are estimating the probabilities from the observed frequencies of hardware failures over a year. We hope that the estimated probabilities are not too far away from the true probabilities. The greater the number of observations, the more accurate the estimated probabilities are. In real life, the estimated probability distribution is often the only option open to us.

A graphical display is obtained with

```
failures <- 0:6
plot(failures, probability, xlab = "Number of hardware
failures in a week", ylab = "Probability", type = "h")
```

which gives Fig. 9.1.

Here `type = "h"` ensures that vertical lines are drawn from the *x* axis to the *x*, *y* co-ordinates. \triangleleft

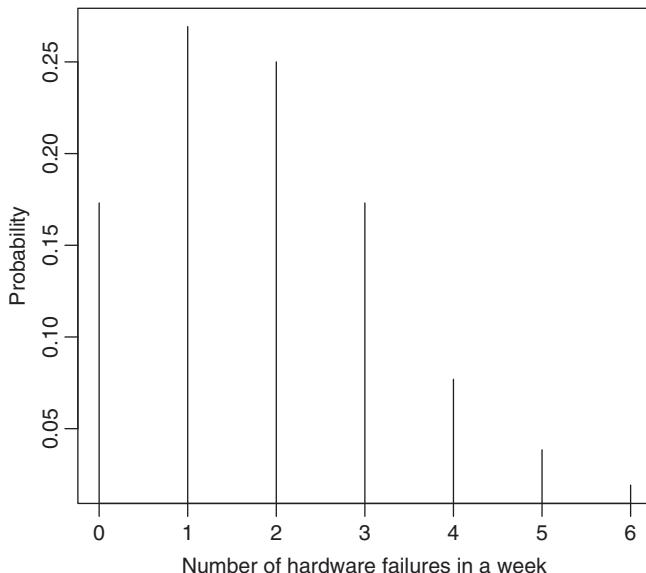


FIGURE 9.1 Probability Density Function of the Hardware Failures

Example 9.6

A particular Java assembler interface was used 2,000 times, and the operand stack size was observed. It was found that

- a zero stack size occurred 100 times
- a stack size of 1 occurred 200 times
- a stack size of 2 occurred 500 times
- a stack size of 3 occurred 500 times
- a stack size of 4 occurred 400 times
- a stack size of 5 occurred 200 times
- a stack size of 6 occurred 80 times
- a stack size of 7 occurred 20 times

By obtaining the relative frequency of each stack size, we can estimate the following probability distribution:

Stack size (x)	0	1	2	3	4	5	6	7
Probability ($P(X = x)$)	0.05	0.10	0.25	0.25	0.20	0.10	0.04	0.01

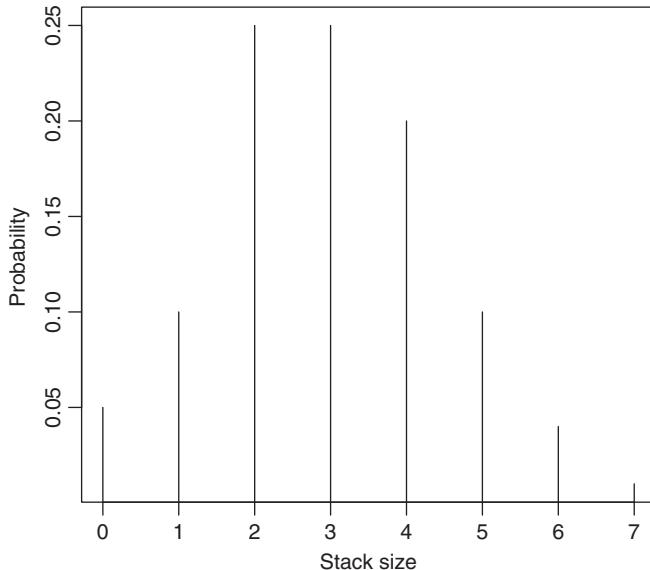


FIGURE 9.2 Operand Stack Size of a Java Assembler Interface

A plot of the distribution of the stack size can be obtained from *R* with

```
stacksize <- 0:7
probability <- c(0.05, 0.10, 0.25, 0.25, 0.20, 0.10, 0.04, 0.01)
plot(stacksize, probability, xlab = "Stack size",
     ylab = "Probability", type = "h")
```

which gives Fig. 9.2. \triangleleft

Definition 9.2 Probability density function (pdf)

For a discrete random variable X , the probability $P(X = x)$ is denoted by $p(x)$ and referred to as the probability density function (pdf), or probability mass function. It satisfies

- $P(X = x) \geq 0$, or equivalently $p(x) \geq 0$
- $\sum_x P(X = x) = 1$, equivalently $\sum_x p(x) = 1$

Obviously, it follows that

- $0 \leq p(x) \leq 1$. \square

9.2 CUMULATIVE DISTRIBUTION FUNCTION

Often, instead of the probability, $P(X = x)$, of exactly one outcome, we might be interested in a group of outcomes of the sample space. In Example 9.5, the manager of the computer system may need to be assured that more than two hardware failures are unlikely to occur in a week, or equivalently that the probability that no more than two hardware failures occur in a week is large ($P(X \leq 2)$). Let us calculate it.

The event $E\{X \leq 2\}$ is the union of the events $\{X = 0\}$, $\{X = 1\}$, $\{X = 2\}$, which are mutually exclusive. That is,

$$E = \{X \leq 2\} = \{X = 0\} \cup \{X = 1\} \cup \{X = 2\}$$

Therefore,

$$\begin{aligned} P(X \leq 2) &= P(X = 0) + P(X = 1) + P(X = 2) \\ &= 0.17 + 0.27 + 0.25 = 0.69 \end{aligned}$$

This is called the cumulative probability, which is obtained by adding the probabilities up to and including two.

Definition 9.3 *Cumulative distribution function (cdf)*

The cumulative distribution function, $F(x)$, of a discrete random variable X is defined by

$$F(x) = \sum_{k \leq x} p(k)$$

where $p(k) = P(X = k)$. □

Let us look again at Example 9.5 and examine the cdf of the hardware failures in a week. We can calculate the cumulative probabilities using the `cumsum` function in *R*.

```
probability <- c(0.17, 0.27, 0.25, 0.17, 0.08, 0.04, 0.02)
cumprob <- cumsum(probability)
cumprob
```

gives

```
[1] 0.17 0.44 0.69 0.86 0.94 0.98 1.00
```

The cdf is

Number of failures (x)	0	1	2	3	4	5	6
$F(x) \equiv (P(X \leq x))$	0.17	0.44	0.69	0.86	0.94	0.98	1.00

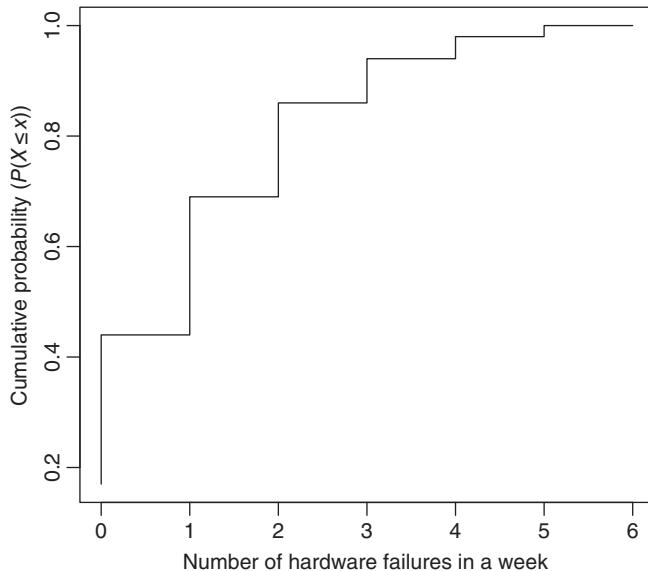


FIGURE 9.3 Cumulative Distribution Function of Hardware Failures

A graphical display is obtained with

```
plot(failures, cumprob, xlab = "Number of hardware failures in a week",
     ylab = "Cumulative probability (P(X<=x))", type = "S")
```

gives Fig. 9.3.

Notice that the expression `type = "S"` in the `plot` function creates the stair steps.

Example 9.7

Errors in eight-bit bytes are detected by a certifier that detects missing pulses in an experimental transition channel. The number of bits in error in each byte ranges from two to six with the following probabilities:

Number of bits in error (x)	2	3	4	5	6
Probability ($p(x)$)	0.4	0.2	0.2	0.1	0.1

We can calculate the cumulative probabilities as

Number of bits in error (x)	2	3	4	5	6
Cumulative probability ($F(x)$)	0.4	0.6	0.8	0.9	1

which means that the probability of

- less than or equal to 2 errors = $P(X \leq 2) = 0.4$
- less than or equal to 3 errors = $P(X \leq 3) = 0.6$
- less than or equal to 4 errors = $P(X \leq 4) = 0.8$
- less than or equal to 5 errors = $P(X \leq 5) = 0.9$
- less than or equal to 6 errors = $P(X \leq 6) = 1.0$

△

9.3 SOME SIMPLE DISCRETE DISTRIBUTIONS

9.3.1 The Bernoulli Distribution

Example 9.8

PCs in a laboratory are attached over the Internet to a central computing system. The probability that any PC is ready to transmit is 0.95. At the lab session, a student chooses a PC at random.

Let $X = 1$ if the PC is ready and $X = 0$ if the PC is not ready.

△

Example 9.9

An unfair coin is tossed once. Success and failure are “heads” and “tails,” respectively, and the probability of a head is 0.7.

Let $X = 1$ if the toss yields a head and $X = 0$ if the toss yields a tail.

△

Example 9.10

It is known that 20% of products on a production line are defective. A product is selected at random from the production line.

Let $X = 1$ if the product is defective, and $X = 0$ if the product is nondefective.

△

Example 9.11

One percent of bits transmitted through a digital transmission are received in error.

Let $X = 1$ if a transmitted bit is in error, and $X = 0$ if a transmitted bit is not in error.

△

Each of the above examples consists of an experiment that has two possible outcomes, a “success” and a “failure.” The only difference in the examples is the probability p of the occurrence of the event of interest, usually designated a success.

$p = 0.95$ in Example 9.8,

$p = 0.70$ in Example 9.9,

$p = 0.20$ in Example 9.10,

$p = 0.01$ in Example 9.11.

They are examples of a probability model known as the *Bernoulli distribution*, the simplest discrete distribution, whose variable takes just two values, 0 and 1.

Definition 9.4 *The Bernoulli random variable*

An experiment consists of two outcomes, a “success” or a “failure.” If we let X equal 1 when the outcome is a success, and 0 when it is a failure, and p the probability of a “success,” then the pdf of X is given by

$$P(X = 0) = 1 - p$$

$$P(X = 1) = p$$

where $0 \leq p \leq 1$.

The cdf is

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - p, & 0 \leq x < 1 \\ 1, & x \geq 1 \end{cases}$$

X is said to be a *Bernoulli* random variable. □

The Bernoulli distribution is the simplest discrete distribution taking just two values 0 and 1, with probabilities $1 - p$ and p , respectively.

9.3.2 Discrete Uniform Distribution

Example 9.12

In the coin tossing experiment, the probability distribution function is

x	0	1
$P(X = x)$	0.5	0.5

△

Example 9.13

When rolling a fair die the random variable X can take values 1, 2, 3, 4, 5, 6, and the probabilities are

x	1	2	3	4	5	6
$P(X = x)$	1/6	1/6	1/6	1/6	1/6	1/6

△

Example 9.14

It is assumed that the first digit of the serial number of laptops is equally likely to be any of the digits one to nine. If a laptop is selected at random from a large batch, and X

is the first digit of the serial number, then X has a uniform distribution. Its values can take 1, 2, 3, 4, 5, 6, 7, 8, 9, each with probability 1/9.

x	1	2	3	4	5	6	7	8	9
$P(X = x)$	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9

△

In each of the above examples, an experiment has n equiprobable outcomes, and each outcome has a probability of $p = 1/n$ of occurrence.

Note

$p = 1/2$ in Example 9.12

$p = 1/6$ in Example 9.13

$p = 1/9$ in Example 9.14

Definition 9.5 Uniform discrete random variable

A random variable X has a uniform discrete distribution if each of the values in the range x_1, x_2, \dots, x_n has equal probability of

$$P(X = x_i) = \frac{1}{n}, \quad i = 1, 2, \dots, n$$

□

To depict the distribution of the experiment in Example 9.13, write in R

```
x <- c(1:6) # assigns the values 1, 2, 3, 4, 5, 6 to
the vector x
probability <- c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
plot (x, probability, type = "h",
      xlab = "Rolling a die", ylab = "Probability")
```

to obtain Fig. 9.4.

Example 9.15

The cdf of the die rolling experiment is obtained in R with the code

```
plot (x, (1:6)/6, type = "s", xlab = " Rolling a die",
      ylab = "Cumulative probability")
```

Here $(1 : 6)/6$ is the vector containing the cumulative probabilities. The *cdf* is given in Fig. 9.5. △

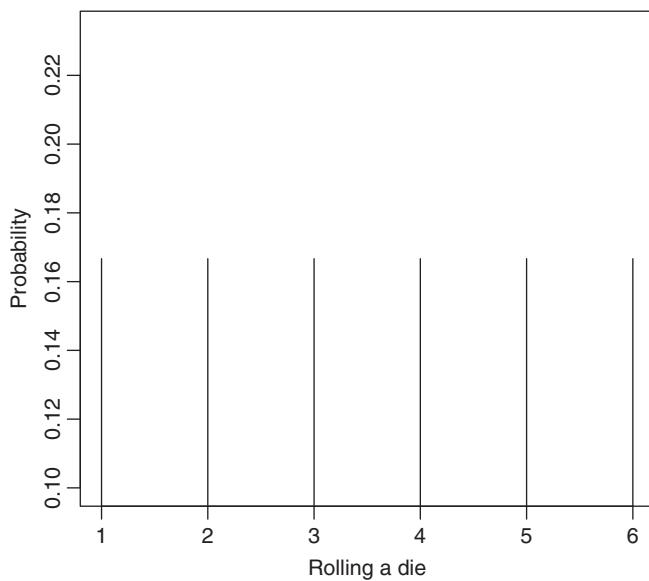


FIGURE 9.4 Probability Density Function

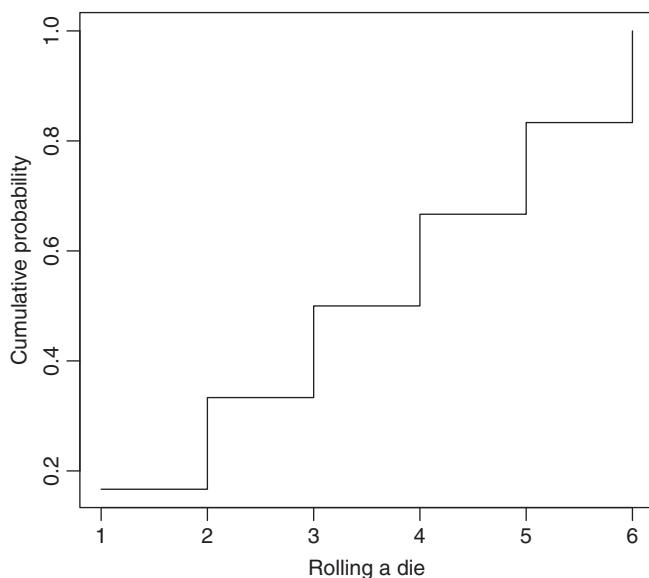


FIGURE 9.5 Cumulative Distribution Function

9.4 BENFORD'S LAW

Looking back at Example 9.14, it was assumed that all first digits of serial numbers of laptops ranged from one to nine and were all equally likely. While this seems plausible, the reality is different. Benford (1938) showed that the number one occurs more frequently as a first digit than all the others. Examining diverse data sets such as rivers, American League baseball statistics, atomic weights of elements, pressure, X-ray volts, and numbers appearing in Reader's Digest articles, he found that the number one appeared as a first digit 30% of the time in all cases.

Much earlier, in 1881, Newcomb, an astronomer and mathematician, noticed that books of logarithms in the library were quite dirty at the beginning and progressively cleaner throughout, which indicated that students looked up numbers starting with one more often than two, two more often than three, and so on. So instead of a uniform distribution assumed in Example 9.14, the first digit would have the highest probability, with probabilities decreasing as the digits increase to nine.

Taking data from different sources, Benford's observations indicated that a likely distribution of first digits would look like the one shown in Table 9.1, and not at all like the uniform distribution assumed in Example 9.14.

In R

```
x <- 1:9
prob <- c(0.306, 0.185, 0.124, 0.094, 0.080, 0.064,
        0.051, 0.049, 0.047)
plot (x, prob, type = "h", xlab = "First digit",
      ylab = "Probability")
```

gives the distribution shown in Fig. 9.6.

Although Benford himself provided evidence that this pattern applied to data from disparate sources back in 1936, it took nearly 60 years before a model was derived to depict this pattern. In 1996, T.P. Hill showed that

$$f(x) = \log_{10} \left(1 + \frac{1}{x} \right), \quad x = 1, 2, \dots, 9$$

approximates the first digit frequencies.

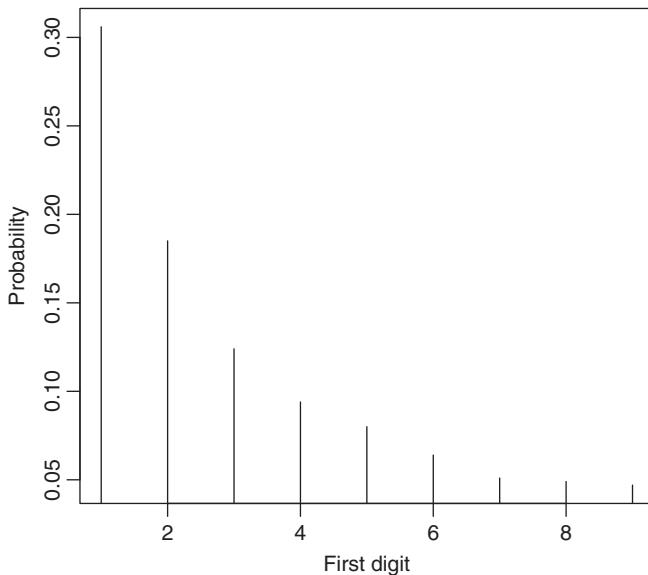
We can check this in R.

```
x <- 1:9
prob <- log10(1 + 1/x)
```

obtains the logarithm of $(1 + 1/x)$ to the base 10 for integer values of x from 1 to 9.

TABLE 9.1 Benford's First Digits

First digit (x)	1	2	3	4	5	6	7	8	9
Probability ($p(x)$)	0.306	0.185	0.124	0.094	0.080	0.064	0.051	0.049	0.047

**FIGURE 9.6** Benford's First Digit

Rounding the output to three decimal places,

```
round(prob, 3)
[1] 0.301 0.176 0.125 0.097 0.079 0.067 0.058 0.051 0.046
```

which, you will agree, are reasonably close to Benford's data given in Table 9.1.

Benford's law has an application to the detection of fraud. Fake data do not usually obey the first digit law. All this shows that finding the correct distribution for a variable may be more subtle than one would think.

9.5 SUMMARIZING RANDOM VARIABLES: EXPECTATION

In Chapter 2, we explored some of the procedures available in *R* to summarize statistical data, and looked, in particular, at measures of central tendency, which are typical or central values of the data set, and measures of spread which, as the name suggests, measure the spread or range of the data.

In a similar fashion, we can summarize random variables by obtaining the average or central values of their distribution, and values which measure their dispersion or variability. This leads us to what is commonly known as the *Expectation*, which is a way of representing important features and attributes of a random variable.

9.5.1 Expected Value of a Discrete Random Variable

In Chapter 2, we defined the *mean* of a set of data as the sum of all values divided by the number of cases, and calculated the average downtime of 23 computers in a computer laboratory.

```
downtime <- c(0, 1, 2, 12, 12, 14, 18, 21, 21, 23, 24,
25, 28, 29, 30, 30, 30, 33, 36, 44, 45, 47, 51)
```

by using

```
mean(downtime)
[1] 25.04348.
```

So the average downtime of all the computers in the laboratory is just over 25 minutes.

Recall that we calculated the mean explicitly in Section 2.4 in Chapter 2 using

```
meandown <- sum(downtime)/length(downtime)
meandown
[1] 25.04348
```

Example 9.16

If 30.3, 24.9, 28.9, 31.7, 26.3, 27.7 represent the salaries in thousands of euro of six recent computer science graduates, then the average is calculated as

$$\text{Average} = \frac{30.3 + 24.9 + 28.9 + 31.7 + 26.3 + 27.7}{6} = 28.3$$

Equivalently,

$$\begin{aligned} \text{Average} &= \left(30.3 \times \frac{1}{6}\right) + \left(24.9 \times \frac{1}{6}\right) + \left(28.9 \times \frac{1}{6}\right) \\ &\quad + \left(31.7 \times \frac{1}{6}\right) + \left(26.3 \times \frac{1}{6}\right) + \left(27.7 \times \frac{1}{6}\right) \\ &= 28.3 \end{aligned}$$

□

In Example 9.5, we might ask, what is the average number of hardware failures?

The number of hardware failures range from 0 to 7, but to calculate the simple arithmetic average of $(0 + 1 + 2 + 3 + 4 + 5 + 6)/7 = 3$ would not be correct, since the values are not all equally likely to occur.

Recall that the number of hardware failures observed over a year revealed that

9 of the 52 weeks had no failures

14 of the 52 weeks had 1 failure

- 13 of the 52 weeks had 2 failures
- 9 of the 52 weeks had 3 failures
- 4 of the 52 weeks had 4 failures
- 2 of the 52 weeks had 5 failures
- 1 of the 52 weeks had 6 failures

The logical thing to do is to get the average by using the relative frequencies of occurrences.

$$\begin{aligned}\text{Average} = & \left(0 \times \frac{9}{52}\right) + \left(1 \times \frac{14}{52}\right) + \left(2 \times \frac{13}{52}\right) + \left(3 \times \frac{9}{52}\right) \\ & + \left(4 \times \frac{4}{52}\right) + \left(5 \times \frac{2}{52}\right) + \left(6 \times \frac{1}{52}\right)\end{aligned}$$

We can check our calculations in *R*.

```
x <- c(0:6)
probability <- c(9/52, 14/52, 13/52, 9/52, 4/52, 2/52, 1/52)
sum(x*probability)
[1] 1.904
```

When dealing with discrete random variables, this weighted average is known as the expected value or the mean of X .

Definition 9.6 Mean of a discrete random variable

The mean of a discrete random variable is defined as the weighted average of all possible values. The weights are the probabilities of respective values of the random variable

$$\mu = E(X) = \sum_x xp(x) \quad (9.1)$$

More generally, the expectation of any function $g(X)$ of a random variable X is the weighted sum of its values.

$$E(g(X)) = \sum_x g(x)p(x) \quad (9.2)$$

where $p(x)$ are again the probabilities $P(X = x)$. □

Example 9.17

Each day, a programmer writes five programs. Over a period of time, it was observed that the number of these programs that compile range from zero to five in any one day with the following probabilities:

Number that compiles (x)	0	1	2	3	4	5
Probability ($p(x)$)	0.237	0.396	0.264	0.088	0.014	0.001

What is the average number of programs that compile per day?

Again

$$\begin{aligned} E(X) &= (0 \times 0.237) + (1 \times 0.396) + (2 \times 0.264) + (3 \times 0.088) \\ &\quad + (4 \times 0.014) + (5 \times 0.001) \end{aligned}$$

In R

```
x <- 0:5
prob <- c(0.237, 0.396, 0.264, 0.088, 0.014, 0.001)
expectation <- sum(x*prob)
expectation
[1] 1.249
```

This indicates that, in the long run, the average number of programs that compile in a day is 1.249. \triangleleft

Returning to Example 9.7, we can get the mean number of errors in the eight-bit bytes.

```
errors <- 2:6
prob <- c(0.4, 0.2, 0.2, 0.1, 0.1)
mean <- sum(errors*prob)
mean
[1] 3.3
```

On average 3.3 bits are in error in an eight-bit byte.

9.5.2 Measuring the Spread of a Discrete Random Variable

One possible way of measuring the spread of a variable is to see how each of its values differs from its mean $\mu = \sum_x xp(x)$ and get the weighted average of all these differences.

$$E(X - \mu) = \sum_x (x - \mu)p(x) \tag{9.3}$$

This is an example of Equation 9.2 with $g(X) = X - \mu$. There is a problem with Equation 9.3, however, as some of the $x - \mu$ are negative, and others are positive,

and it turns out that the positives and negatives cancel out so that Equation 9.3 is zero.

$$\begin{aligned} E(X - \mu) &= \sum_x (x - \mu)p(x) \\ &= \sum_x xp(x) - \mu \sum_x p(x) \\ &= \mu - \mu = 0 \end{aligned}$$

since $\sum_x xp(x) = \mu$ and the sum of the probabilities $\sum_x p(x) = 1$.

One way to overcome this would be to take the absolute values.

$$E(|X - \mu|) = \sum_x |x - \mu|p(x) \quad (9.4)$$

This is called the *mean deviation*, and although it gets over the problem of negative deviations canceling with positive deviations, it is not commonly used as a measure of spread.

Definition 9.7 Variance of a discrete random variable

The usual way of overcoming the problem of negative and positive deviations is to square them.

$$V(X) = \sigma^2 = E(X - \mu)^2 = \sum_x (x - \mu)^2 p(x)$$

This is called the variance and is the most commonly used measure of spread. It is the weighted average of the squared differences between each possible value of the random variable and the mean, the weights being again the probabilities of the respective outcomes. \square

The square root of the variance, σ , is the subtle alternative to Equation 9.4, and there are technical mathematical reasons for preferring what appears to be a more complicated alternative.

Example 9.18

Returning to the number of hardware failures occurring over a year, given in Example 9.5, we note again that the values of the random variable have different probabilities of occurrence. It would make sense that, when calculating the variance, the deviations should be weighted by their relative frequencies, just as we did when obtaining the mean, $\mu = 1.904$

$$\begin{aligned} E(X - \mu)^2 &= (0 - 1.904)^2 \frac{9}{52} + (1 - 1.904)^2 \frac{14}{52} + (2 - 1.904)^2 \frac{13}{52} + (3 - 1.904)^2 \frac{9}{52} \\ &\quad + (4 - 1.904)^2 \frac{4}{52} + (5 - 1.904)^2 \frac{2}{52} + (6 - 1.904)^2 \frac{1}{52} \end{aligned}$$

In R

```
x <- 0:6
probability <- c(9/52, 14/52, 13/52, 9/52, 4/52, 2/52, 1/52)
mean <- sum(x*probability)
mean
variance <- sum((x-mean)^2 *probability)
variance
[1] 2.0869
```

◇

Example 9.19

We can get the variance of the number of errors in the eight-bit bytes described in Example 9.7. Recall that the mean was calculated to be

$$E(X) = 3.3$$

So the variance is

$$\begin{aligned} E(X - 3.3)^2 &= (2 - 3.3)^2 * 0.4 + (3 - 3.3)^2 * 0.2 + (4 - 3.3)^2 * 0.2 \\ &\quad +(5 - 3.3)^2 * 0.1 + (6 - 3.3)^2 * 0.1 \end{aligned}$$

In R

```
errors <- 2:6
prob <- c(0.4, 0.2, 0.2, 0.1, 0.1)
mean <- sum(errors*prob)
var <- sum((errors - mean)**2*prob)
var
[1] 1.81
```

◇

9.6 PROPERTIES OF EXPECTATIONS

Let us look again at Example 9.5, the number of hardware failures, X , of a computer system in a week of operation, which has the following distribution:

Number of failures (x)	0	1	2	3	4	5	6
Probability ($p(x)$)	0.17	0.27	0.25	0.17	0.08	0.04	0.02

Suppose the monetary loss due to failures can be estimated in euro using the following linear function:

$$\text{Loss} = 10X + 200$$

Let us estimate:

- the expected weekly loss;
- the variance of the expected weekly loss.

In this case, we need to know

$$E(10X + 200)$$

$$V(10X + 200)$$

From the definition of *expectation* given above, it is easy to deduce the following properties for any constant c :

1. $E(X + c) = E(X) + c$

That is, *adding a constant to each value has the effect of increasing the average by the constant amount*

$$\begin{aligned} E(X + c) &= \sum_x (x + c)p(x) \\ &= \sum_x xp(x) + c \sum_x p(x) \\ &= E(X) + c \end{aligned}$$

since $\sum_x p(x) = 1$ and $\sum_x xp(x) = E(X)$

2. $E(cX) = cE(X)$

That is, *multiplying each value by a constant amount increases the average by the constant amount*

$$\begin{aligned} E(cX) &= \sum_x cxp(x) \\ &= c \sum_x xp(x) \\ &= cE(X) \end{aligned}$$

3. $V(X + c) = V(X)$

That is, *adding a constant to each value of the variable does not affect the variance*

$$\begin{aligned} V(X + c) &= E((X + c) - E(X + c))^2 \\ &= E(X - E(X))^2 \\ &= V(X) \end{aligned}$$

4. $V(cX) = c^2V(X)$

That is, multiplying each value of the variable by a constant c causes the variance to be multiplied by the square of that constant

$$\begin{aligned} V(cX) &= E(cX - E(cX))^2 \\ &= E(cX - cE(X))^2 \\ &= E(c(X - E(X)))^2 \\ &= \sum_x (c(x - E(X)))^2 p(x) \\ &= c^2 \sum_x (x - E(X))^2 p(x) \\ &= c^2 E(X - E(X))^2 \\ &= c^2 V(X) \end{aligned}$$

The first two results, concerning the mean values, are intuitively obvious, but 3 and 4, concerning the variances are not. We will illustrate the results on variances using simulation in the next section. Meanwhile, returning to assessing the monetary loss due to failures in Example 9.5

$$E(10X + 200) = 10E(X) + 200$$

and

$$V(10X + 200) = 10^2 V(X)$$

We showed previously that $E(X) = 1.904$ and $V(X) = 2.0869$.

Therefore,

$$E(10X + 200) = 10(1.904) + 200 = 219.04$$

and

$$V(10X + 200) = 10^2(2.0869) = 208.69$$

Example 9.20

The average salary of new employees in a computer firm is €37,500 with a variance of 400.

After negotiations with the trade union, it was agreed that employees would each get a rise of €200, in addition to a 10% increase in their basic salaries. What is the new average salary?

Let X = old salary; Y = new salary.

$$Y = 200 + 1.1X$$

Then

$$E(Y) = 200 + 1.1E(X) = 200 + 1.1(37,500) = €41,450$$

since $E(X) = €37,500$.

Also,

$$V(Y) = 1.1^2 V(X) = 1.1^2(400) = 484$$

since $V(X) = 400$. \(\triangleleft\)

9.7 SIMULATING DISCRETE RANDOM VARIABLES AND EXPECTATIONS

Let us look again at Example 9.4, rolling a die. Let $X = 1, 2, 3, 4, 5, 6$, the face number, and each can occur with equal probability $1/6$.

The average

$$E(X) = \left(1 \times \frac{1}{6}\right) + \left(2 \times \frac{1}{6}\right) + \left(3 \times \frac{1}{6}\right) + \left(4 \times \frac{1}{6}\right) + \left(5 \times \frac{1}{6}\right) + \left(6 \times \frac{1}{6}\right) = 3.5$$

The average is 3.5, although it is not possible to get 3.5 showing on the die in any roll. So what does 3.5 mean in the context of rolling a die?

If we throw a die a large number of times, we would expect the average of all the outcomes to be 3.5. Let us investigate with R .

```
roll <- sample(c(1:6), 60, replace = T)
```

The `sample` function choose 60 numbers between 1 and 6 with replacement (`replace = T`), and with equal probability. To see what is in `roll`

```
roll
[1] 4 2 6 1 2 1 2 2 5 2 5 3 1 1 6 4 5 5 3 2 3 3 5 1 2 6
[27] 5 6 3 3 5 1 4 4 4 6 4 5 4 2 6 4 1 2 1 5 6 4 4 2 4 1
[53] 6 4 5 3 1 4 1 4
```

This is a simulation of rolling a fair die 60 times. The average face value is obtained in R with

```
mean(roll)
[1] 3.433333
```

which, as you can see, is not too far away from $E(X) = 3.5$.

You will recall that the variance is obtained by subtracting the mean from each value, squaring and averaging.

```
variance <- sum((roll - mean(roll))**2)/60
variance
[1] 2.812222
```

Obviously, the greater the number of simulations, the more accurate the estimates are. Let us repeat this experiment 6000 times.

```
roll = sample(c(1:6), 6000, replace = T)
```

The `table` function gives the simulated frequencies of each die face.

```
table(roll)
roll
  1     2     3     4     5     6
 970 1007 1047  999 1004  973
```

We can estimate the probabilities by calculating the relative frequencies and rounding to two decimal places.

```
round(table(roll)/6000, 2)
  1     2     3     4     5     6
 0.16  0.16  0.17  0.16  0.16  0.16
```

Notice that the estimated probabilities are near equal. They may be graphed:

```
plot(table(roll)/6000, xlab = "Die face",
     ylab = "Relative frequency")
```

and compared with the actual distribution:

```
probability <- c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
plot (1:6, probability, type = "h", xlab = "Die face",
      ylab = "Probability", ylim = c(0, 1/6))
```

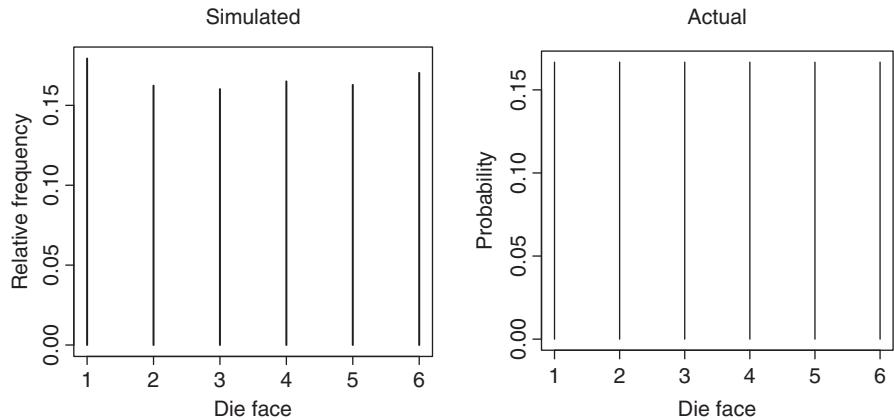
Figure 9.7 gives the distribution the simulated distribution obtained by rolling a die 6000 times, along with actual uniform distribution.

You can see from Table 9.7 that the simulated distribution is near uniform, with the relative frequencies near the actual $1/6$ in all cases.

The mean of the simulated distribution is

```
mean(roll)
[1] 3.4965
```

similar to the actual mean of 3.5.

**FIGURE 9.7** Rolling a Die

Returning to Example 9.5, we can simulate 1,000 hardware failures as follows:

```
failures <- sample(c(0:6), 1000, replace = T,
                    prob = c(0.17, 0.27, 0.25, 0.17, 0.08, 0.04, 0.02))
```

To obtain the simulated distribution of hardware failures, write in *R*

```
plot(table(failures)/1000, xlab = "Number of hardware
failures in a week", ylab = "Relative frequency")
```

Figure 9.8 gives the simulated distribution along with the actual distribution.

In Fig. 9.8, we see that the simulated distribution approximates the actual quite closely.

To obtain the mean and variance of the simulated data, write in *R*

```
mean(failures)
```

which gives

```
[1] 1.934
```

and

```
var(failures)
```

gives

```
[1] 2.103054
```

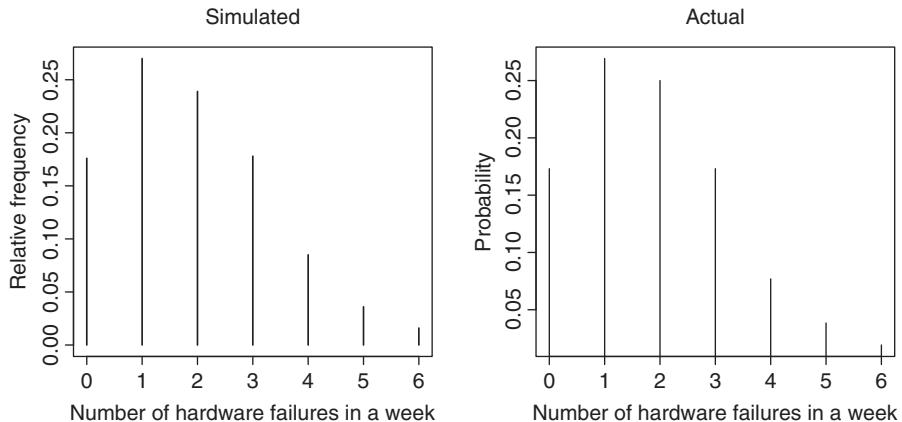


FIGURE 9.8 Hardware Failures

each of which is near the respective theoretical parameters $\mu = 1.904$ and $V(X) = 2.0869$.

Now, let us multiply the simulated data by 10 and calculate the mean and variance.

```
failures10 <- 10*failures
mean(failures10)
[1] 19.34
var(failures10)
[1] 210.3054
```

Here, the mean has increased by 10, and the variance by 100, as we have anticipated from the theoretical results. Finally, if we add 200

```
failures10plus200 <- 10*failures + 200
```

and obtain the mean and variance of the transformed data.

```
mean(failures10plus200)
[1] 219.94
var(failures10plus200)
[1] 210.3054
```

We can see that the mean has increased by 200, while the variance stays the same. Adding 200 to each value increases the mean by 200 but does not change the variance. What we are doing in effect is moving the x -values along the axis to the right while maintaining the spread.

9.8 BIVARIATE DISTRIBUTIONS

Up until now, we have examined distributions with one variable. Often, we may be interested in studying two or more random variables defined on a given sample space. For example, the cost of a laptop and its capacity, the size of RAM and the speed of the CPU, or the errors in two modules of a program. In what follows, we take a brief look at probability distributions with two variables.

Let us look at some examples.

Example 9.21

A program consists of two modules.

X = the number of errors in the first module.

Y = the number of errors in the second module.

When X and Y are discrete, it is usual to depict the bivariate distribution as a table or matrix, as shown in Table 9.2. \triangleleft

Example 9.22

Roll two fair dice. Let X be the outcome of first roll and let Y be the outcome of the second roll. This bivariate distribution is given in Table 9.3. \triangleleft

TABLE 9.2 Joint Distribution of Errors in Programs

		Module 2 (Y)			
		0	1	2	3
Module 1 (X)	0	0.2	0.2	0.05	0.05
	1	0.2	0.1	0.1	0.1

TABLE 9.3 Rolling Two Dice

		Die 2 (Y)					
		1	2	3	4	5	6
Die 1 (X)	1	1/36	1/36	1/36	1/36	1/36	1/36
	2	1/36	1/36	1/36	1/36	1/36	1/36
	3	1/36	1/36	1/36	1/36	1/36	1/36
	4	1/36	1/36	1/36	1/36	1/36	1/36
	5	1/36	1/36	1/36	1/36	1/36	1/36
	6	1/36	1/36	1/36	1/36	1/36	1/36

TABLE 9.4 Joint Distribution of Hardware Failures

		Laboratory 1 (Y)		
		1	2	3
Laboratory 1 (X)		1	0.52	0.20
		2	0.14	0.02
		3	0.06	0.01
				0

Example 9.23

Students use two laboratories for their practical tutorials. Let X be the number of hardware failures in laboratory 1 and Y be the number of hardware failures in laboratory 2 in a given week during the semester. Table 9.4 gives the distribution of X and Y . \triangleleft

Definition 9.8 *Probability density function of a bivariate random variable*

With two random variables X and Y , the probability that X takes the value x and Y takes the value y , that is $P((X = x) \cap (Y = y))$, is written as $P(X = x, Y = y)$ or equivalently $p(x, y)$, and is called the joint probability of X and Y . \square

A bivariate discrete distribution satisfies the following:

- $P(X = x, Y = y) \geq 0$ or equivalently $p(x, y) \geq 0$
- $\sum_x \sum_y P(X = x, Y = y) = 1$ or equivalently $\sum_x \sum_y p(x, y) = 1$

Obviously,

$$0 \leq p(x, y) \leq 1$$

It is easy to check that these conditions are satisfied in the aforementioned examples.

Let us examine Example 9.23, which deals with hardware failures in two laboratories. To read these data in R , we use the matrix function.

```
prob <- c(0.52, 0.14, 0.06, 0.20, 0.02, 0.01, 0.04, 0.01, 0.00)
```

Then form a 3×3 matrix.

```
hardware <- matrix(prob, 3, 3)
```

```
hardware
```

[,1]	[,2]	[,3]
[1,] 0.52	0.20	0.04
[2,] 0.14	0.02	0.01
[3,] 0.06	0.01	0.00

Let us check that this satisfies the conditions of a joint probability distribution,

- Clearly, each entry in the matrix is less than 1
- To check that $\sum_x \sum_y (p(x, y)) = 1$, write in R

```
sum(hardware)
```

which returns

```
[1] 1
```

We can also calculate probabilities of various events from this matrix. For example,

- `hardware[2, 3]`
`[1] 0.01`

which is $P(X = 2, Y = 3)$

- $P(X = 1) = 0.52 + 0.20 + 0.04 = 0.76$. In R, sum row 1 over the columns

```
sum(hardware[1, ])  

[1] 0.76
```

- $P(Y = 3) = 0.04 + 0.01 + 0 = 0.05$. In R, sum column 3 over the rows.

```
sum(hardware[, 3])  

[1] 0.05
```

9.9 MARGINAL DISTRIBUTIONS

Definition 9.9 *Marginal distribution of a random variable*

The marginal distribution of X is written as $p(x)$ and obtained as follows:

$$p(x) = P(X = x) = \sum_y p(x, y)$$

which is the probability that X takes on a particular value, regardless of the values of Y . In other words, Y has been *marginalized* out. □

Similarly, the marginal distribution of Y is

$$p(y) = P(Y = y) = \sum_x p(x, y)$$

To get the marginal distribution of X in Example 9.23 concerning hardware failures in two laboratories, simply sum the columns. In R

```
sum(hardware[1, 1:3])
[1] 0.76
```

gives $P(X = 1)$

```
sum(hardware[2, 1:3])
[1] 0.17
```

gives $P(X = 2)$ and

```
sum(hardware[3, 1:3])
[1] 0.07
```

gives $P(X = 3)$.

The marginal distribution of Y is obtained by summing the rows. In R

```
sum(hardware[1:3, 1])
[1] 0.72
```

gives $P(Y = 1)$

```
sum(hardware[1:3, 2])
[1] 0.23
```

gives $P(Y = 2)$ and

```
sum(hardware[1:3, 3])
[1] 0.0
```

gives $P(Y = 3)$.

Table 9.5 gives the bivariate probabilities along with the marginal distribution of X (row totals) and the marginal distribution of Y (column totals).

So we can see from Table 9.5 that, by obtaining marginal totals, we can retrieve the univariate distributions of the variables X and Y .

9.10 CONDITIONAL DISTRIBUTIONS

Definition 9.10 *The conditional distribution of a random variable*

If $P(X, Y)$ is the joint distribution function of X and Y , then the conditional distribution of X is the set probabilities of X for specified values of Y , and the conditional distribution of Y is these probabilities of Y for specified values of X . \square

TABLE 9.5 Marginal Distributions of Hardware Failures

$P(X = x, Y = y)$ $p(x, y)$		Laboratory 2 (Y)			Marginal (X)
		1	2	3	$P(X = x)$
Laboratory 1 (X)	1	0.52	0.20	0.04	0.76
	2	0.14	0.02	0.01	0.17
	3	0.06	0.01	0	0.07
Marginal (Y)	$P(Y = y)$	0.72	0.23	0.05	1

In Example 9.23, concerning hardware failures in two laboratories, we calculate the conditional distributions of X for given values of Y .

- With $Y = 1$

$$P(X = 1|Y = 1) = \frac{P((X = 1) \cap (Y = 1))}{P(Y = 1)} = \frac{0.52}{0.72} = 0.72$$

$$P(X = 2|Y = 1) = \frac{P((X = 2) \cap (Y = 1))}{P(Y = 1)} = \frac{0.14}{0.72} = 0.20$$

$$P(X = 3|Y = 1) = \frac{P((X = 3) \cap (Y = 1))}{P(Y = 1)} = \frac{0.06}{0.72} = 0.08$$

gives the conditional distribution of X for $Y = 1$.

- With $Y = 2$

$$P(X = 1|Y = 2) = \frac{P((X = 1) \cap (Y = 2))}{P(Y = 2)} = \frac{0.20}{0.23} = 0.87$$

$$P(X = 2|Y = 2) = \frac{P((X = 2) \cap (Y = 2))}{P(Y = 2)} = \frac{0.02}{0.23} = 0.09$$

$$P(X = 3|Y = 2) = \frac{P((X = 3) \cap (Y = 2))}{P(Y = 2)} = \frac{0.01}{0.23} = 0.04$$

gives the conditional distribution of X for $Y = 2$.

- With $Y = 3$

$$P(X = 1|Y = 3) = \frac{P((X = 1) \cap (Y = 3))}{P(Y = 3)} = \frac{0.04}{0.05} = 0.8$$

$$P(X = 2|Y = 3) = \frac{P((X = 2) \cap (Y = 3))}{P(Y = 3)} = \frac{0.01}{0.05} = 0.2$$

$$P(X = 3|Y = 3) = \frac{P((X = 3) \cap (Y = 3))}{P(Y = 3)} = \frac{0}{0.05} = 0.0$$

gives the condition distribution of X for $Y = 3$.

TABLE 9.6 Conditional Distributions of Hardware Failures in Lab 1 (X)

X	$P(X Y = 1)$	$P(X Y = 2)$	$P(X Y = 3)$
1	0.72	0.87	0.8
2	0.19	0.09	0.2
3	0.08	0.04	0.0

Notice that

$$\text{Conditional} = \frac{\text{joint}}{\text{marginal}}$$

The conditional distributions of X are summarized in Table 9.6 for each value of Y . In the same fashion, we obtain the conditional distributions of Y , for each value of X .

- With $X = 1$

$$P(Y = 1|X = 1) = \frac{P((X = 1) \cap (Y = 1))}{P(X = 1)} = \frac{0.52}{0.76} = 0.69$$

$$P(Y = 2|X = 1) = \frac{P((X = 1) \cap (Y = 2))}{P(X = 1)} = \frac{0.20}{0.76} = 0.26$$

$$P(Y = 3|X = 1) = \frac{P((X = 1) \cap (Y = 3))}{P(X = 1)} = \frac{0.04}{0.76} = 0.05$$

gives the conditional distribution of Y for $X = 1$.

- With $X = 2$

$$P(Y = 1|X = 2) = \frac{P((X = 2) \cap (Y = 1))}{P(X = 2)} = \frac{0.14}{0.17} = 0.82$$

$$P(Y = 2|X = 2) = \frac{P((X = 2) \cap (Y = 2))}{P(X = 2)} = \frac{0.02}{0.17} = 0.12$$

$$P(Y = 3|X = 2) = \frac{P((X = 2) \cap (Y = 3))}{P(X = 2)} = \frac{0.01}{0.17} = 0.06$$

gives the conditional distribution of Y for $X = 2$.

- With $X = 3$

$$P(Y = 1|X = 3) = \frac{P((X = 3) \cap (Y = 1))}{P(X = 3)} = \frac{0.06}{0.07} = 0.86$$

$$P(Y = 2|X = 3) = \frac{P((X = 3) \cap (Y = 2))}{P(X = 3)} = \frac{0.01}{0.07} = 0.14$$

$$P(Y = 3|X = 3) = \frac{P((X = 3) \cap (Y = 3))}{P(X = 3)} = \frac{0}{0.07} = 0.0$$

gives the condition distribution of Y for $X = 3$.

The conditional distributions of Y are summarized in Table 9.7.

TABLE 9.7 Conditional Distributions of Hardware Failures in Lab 2 (Y)

Y	1	2	3
$P(Y X = 1)$	0.69	0.26	0.05
$P(Y X = 2)$	0.82	0.12	0.06
$P(Y X = 3)$	0.86	0.14	0.0

These calculations can be obtained in R . For example, to obtain the conditional distribution $P(X|Y = 2)$, first obtain the marginal $P(Y = 2)$.

```
y2 <- sum(hardware[1:3, 2])
y2
[1] 0.23
```

Then the conditional probabilities $P(X|Y = 2)$ can be obtained with

```
round((hardware[1, 2]/y2), 2)
[1] 0.87
round((hardware[2, 2]/y2), 2)
[1] 0.09
round((hardware[3, 2]/y2), 2)
[1] 0.04
```

or more neatly,

```
round((hardware[1:3, 2]/y2), 2)
[1] 0.87 0.09 0.04
```

Notice that we have rounded the probabilities to two decimal places.

Of course, using R with these small tables is not essential, but often in practical situations, the matrix can be of much higher dimensions. For example, in machine translation, which we discussed in Chapter 7, matrices of conditional probabilities can be of enormous dimensions.

Definition 9.11 *Independence*

Random variables X and Y are independent if for every x and y

$$p(x, y) = p(x)p(y)$$

□

Let us examine Example 9.4 and investigate if hardware failures in the two laboratories are independent.

- $P(X = 1) = 0.72$
- $P(Y = 1) = 0.17$
- $P((X = 1) \cap (Y = 1)) = 0.52 \neq 0.72 \times 0.17$

Therefore, X and Y are not independent. Not surprisingly, hardware failures in the two laboratories are not independent.

EXERCISES 9.1

1. For a particular Java assembler interface, it is known that the operand stack size has the following probability distribution function:

Stack size	0	1	2	3	4	5	6	7
Probability	0.05	0.15	0.25	0.20	0.15	0.15	0.04	0.01

Use R to:

- (a) plot this distribution;
 - (b) calculate the expected stack size;
 - (c) calculate the variance of the stack size.
2. In Example 9.21, obtain the marginal distributions of errors for module 1 and module 2, and use R to calculate the matrix of conditional distributions for each of the modules.

9.11 PROJECT

With your own computing experience, develop a front end to R that allows the user

- to input the values of a univariate discrete random variable and the associated probabilities and to obtain the mean and variance.
- to input the values of a bivariate discrete random variable and the associated probabilities and to obtain the marginal and conditional distributions. Your program should provide a facility to calculate the mean and variance of each distribution, and to plot the pdf and cdf.

In each program, do validity checks that the probabilities are in the interval $[0, 1]$, and that they sum to one.

REFERENCES

- Benford, F. (1938), The law of anomalous numbers, *Proceedings of the American Philosophical Society*, 76, 551572.
- Hill, T.P. (1996), The first digit phenomenon, *American Mathematical Monthly*, 102, 322–327.
- Newcomb, S. (1881), Note on the frequency of the use of digits in natural numbers, *American Journal of Mathematics*, 4, 39–40.

10

THE GEOMETRIC DISTRIBUTION

Example 10.1

Workstations attached to a server on a network each have a probability of 0.95 of being operational. Workstations are polled until the first operational workstation is found.

Let X denote the number of workstations polled until the first operational workstation is found.

$$P(X = 1) = P(\text{operational workstation in first poll})$$

$$P(X = 2) = P(\text{first workstation not operational followed by an operational workstation})$$

$$P(X = 3) = P(\text{first two workstations not operational followed by an operational workstation}) \quad \triangleleft$$

Example 10.2

A fair coin is tossed repeatedly until the first head is obtained.

Let X denote the number of tosses to first head.

$$P(X = 1) = P(\text{head in first toss})$$

$$P(X = 2) = P(\text{tail first followed by a head})$$

$$P(X = 3) = P(\text{two tails followed by a head}) \quad \triangleleft$$

Example 10.3

It is known that 20% of products on a production line are defective. Products are inspected until the first defective is obtained.

Let X denote the number of inspections to obtain first defective.

$$P(X = 1) = P(\text{defective in first inspection})$$

$$P(X = 2) = P(\text{nondefective first followed by a defective})$$

$$P(X = 3) = P(\text{two nondefectives followed by a defective}) \quad \triangleleft$$

Example 10.4

One percent of bits transmitted through a digital transmission are received in error.

Let X denote the number of bits transmitted until the first error.

$$P(X = 1) = P(\text{error in first transmission})$$

$$P(X = 2) = P(\text{one correct bit followed by an error in second transmission})$$

$$P(X = 3) = P(\text{two correct bits followed by an error in third transmission}) \quad \triangleleft$$

In each of the aforementioned examples, an experiment is carried out repeatedly until the first occurrence of the event of interest. We will call such an event a “success,” although this may not always be a good thing; in Example 10.3, for example, a “success” is a defective.

If we denote the probability of a success in any one trial as p , then

$$p = 0.95 \text{ in Example 10.1}$$

$$p = 0.50 \text{ in Example 10.2}$$

$$p = 0.20 \text{ in Example 10.3}$$

$$p = 0.01 \text{ in Example 10.4}$$

The geometric distribution consists of a sequence of Bernoulli trials carried out until the first success. The probabilities can be visualized in Fig. 10.1. As one proceeds along the baseline, at each intersection, there is a probability p of turning left (“success”) and a probability $q = 1 - p$ of continuing straight on (“failure”).

$P(X = 1)$	success in first trial	p
$P(X = 2)$	failure followed by a success	qp
$P(X = 3)$	two failures followed by a success	q^2p
$P(X = 4)$	three failures followed by a success	q^3p

and generally

$$P(X = x) \quad x - 1 \text{ failures followed by first success} \quad q^{x-1}p$$

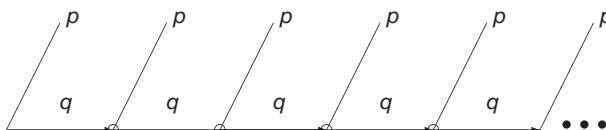


FIGURE 10.1 Geometric Probabilities

This model has what is known as a *geometric distribution*, so called because the successive probabilities are in geometric progression.

10.1 GEOMETRIC RANDOM VARIABLES

Definition 10.1 Geometric random variable

A geometric random variable, X , is one that can be represented as the number of trials to the first success in an experiment in which

1. Each trial has two possible outcomes;
 - A success with probability p
 - A failure with probability $q = 1 - p$
2. Repeated trials are independent. □

10.1.1 Probability Density Function

The probability density function (pdf) is defined as

$$P(X = x) = q^{x-1}p; \quad x = 1, 2, 3, \dots \quad (10.1)$$

which is the probability that the first success will occur in the x th trial.

The geometric random variable is discrete because its values are integers, and infinite because, theoretically, we could be waiting forever to arrive at a success.

We can easily check that $\sum_x P(X = x) = 1$.

$$\begin{aligned} \sum_x P(X = x) &= p + qp + q^2p + q^3p + \dots \\ &= p(1 + q + q^2 + q^3 + \dots) = p \times \frac{1}{1-q} = 1 \end{aligned}$$

remembering that the sum of the infinite geometric series with ratio $q < 1$ is

$$1 + q + q^2 + q^3 + \dots = \frac{1}{1-q} = \frac{1}{p}$$

Example 10.5

Products produced by a machine have a 3% defective rate.

What is the probability that the first defective occurs in the fifth item inspected?

$$\begin{aligned} P(X = 5) &= P(\text{first four nondefective}) \quad P(\text{fifth defective}) \\ &= (0.97)^4 \times 0.03 \\ &= 0.0266 \quad \text{□} \end{aligned}$$

Example 10.6

Jane and Ava are rolling a die. Whoever gets a six first wins. As Ava is the younger, she starts first. What is the probability that

- (a) Ava wins?
- (b) Jane wins?

Now, Ava will win if the first six occurs in the first, third, fifth, and so on, roll of the die, and Jane will win if the first six occurs on the second, fourth, sixth, and so on, roll of the die.

Therefore,

$$P(\text{Ava wins}) = P(\text{six first or first six on 3rd toss or first six on 5th toss} + \dots)$$

$$= \frac{1}{6} + \left(\frac{5}{6}\right)^2 \frac{1}{6} + \left(\frac{5}{6}\right)^4 \frac{1}{6} + \dots$$

which is an infinite geometric series with first term $1/6$ and ratio $(5/6)^2$. So,

$$P(\text{Ava wins}) = \frac{1}{6} \left(\frac{1}{1 - (\frac{5}{6})^2} \right) = \frac{6}{11}$$

Obviously, if Ava does not win, Jane will. Therefore,

$$P(\text{Jane wins}) = 1 - P(A) = \frac{5}{11}$$

Not surprisingly, Ava, who starts first, has a higher probability of winning than Jane. \triangleleft

Example 10.7

Suppose there are three players, Jane, Amy, and Ava. Again, Ava the youngest starts first, Amy, the second youngest plays second, and Jane, the eldest, plays third. Whoever rolls the first six wins the game. What is the probability that

- (i) Ava wins?
- (ii) Amy wins?
- (iii) Jane wins?

Ava will win if the first six occurs on first or fourth or seventh or tenth roll

Amy will win if the first six occurs in second or fifth or eighth or eleventh roll

Jane, who goes third, will win if the first six occurs on third or sixth or ninth

By reasoning similar to what we used in the previous example,

$$P(\text{Ava wins}) = P(\text{six first or first six on fourth toss or first six on seventh toss or ...})$$

$$= \frac{1}{6} + \left(\frac{5}{6}\right)^3 \frac{1}{6} + \left(\frac{5}{6}\right)^6 \frac{1}{6} + \dots$$

which is an infinite geometric series with first term $1/6$ and ratio $(5/6)^3$.

Therefore,

$$P(\text{Ava wins}) = \left(\frac{1}{6}\right) \left(\frac{1}{1 - \left(\frac{5}{6}\right)^3} \right)$$

Calculating in *R*

```
(1/6) * (1 / (1 - (5/6) ** 3))
```

gives

```
[1] 0.3956044
```

$P(\text{Amy wins}) = P(\text{first six on 2nd toss or first six on 5th toss or first six on 8th toss or } \dots)$

$$= \left(\frac{5}{6}\right) \frac{1}{6} + \left(\frac{5}{6}\right)^4 \frac{1}{6} + \left(\frac{5}{6}\right)^7 \frac{1}{6} + \dots$$

which is an infinite geometric series with first term $(5/6)(1/6)$ and ratio $(5/6)^3$.

Therefore,

$$P(\text{Amy wins}) = \frac{5}{36} \left(\frac{1}{1 - \left(\frac{5}{6}\right)^3} \right)$$

Calculating in *R*

```
(5/36) * (1 / (1 - (5/6) ** 3))
```

gives

```
[1] 0.3296703
```

Finally, the probability that Jane wins is the complementary event.

$$P(\text{Jane wins}) = 1 - (P(\text{Ava wins}) + P(\text{Amy wins})) = 0.2747253$$

Clearly, the order of play is important in determining the likelihood of winning:

- Ava, who starts first, has nearly a 40% chance of winning;
- Amy, who starts second, has less than a 33% chance of winning;
- Jane, the last player, has less than a 28% chance of winning.

▫

10.1.2 Calculating Geometric pdfs in *R*

Functions are provided in *R* to calculate and plot the pdf. For the geometric distribution, we prefix the shortened name *geom* with “d”, for the pdf; the function *dgeom*,

along with the parameter p , the probability of a success, as an argument will calculate the probabilities.

The result in Example 10.3, finding the probability that the first defective will occur in the fifth defective, can be checked in R .

```
dgeom (x = 4, prob = 0.03)
```

or

```
dgeom (4, 0.03)
```

gives

```
[1] 0.02655878
```

The convention in R is to record X as the number of failures that occur *before* the first success. It is more usual to define the geometric random variable X as the number of trials to the first success, as we did in the previous section. This does not cause a problem provided we remember that the X calculations in R correspond to our $X + 1$ calculations above. For example, $X = 0$ in R means zero failures before first success, which is equivalent to $X = 1$, the first success occurs in the first trial, in our model. Similarly, $X = 1$ in R means one failure before first success, which is equivalent to the first success occurring in the second trial.

It is easy to calculate a set of probabilities in R by putting the x values in a vector and using `dgeom`. In Example 10.1, to calculate the probabilities of getting an operational workstation in first poll, second poll, third poll, and fourth poll, write in R

```
x <- 0:3
dgeom(x, 0.95)
```

gives

```
[1] 0.9500000 0.0475000 0.0023750 00011875
```

Clearly, the number of places after the decimal points is excessive here, use

```
round(dgeom(x, 0.95), 4)
[1] 0.9500 0.0475 0.0024 0.0001
```

to round the data to four decimal places.

In Example 10.2, to calculate the probability of getting the first head in the first, second, third toss, and so on,

```
x <- 0:9
prob <- dgeom(x, 0.5)
```

To round the output to four decimal places, write

```
round(prob, 4)
[1] 0.5000 0.2500 0.1250 0.0625 0.0313 0.0156 0.0078
[8] 0.0039 0.0020 0.0010
```

In Example 10.3, to calculate the probability of getting the first defective in first, second inspection, and so on,

```
x <- 0:9
prob <- dgeom(x, 0.2)
round(prob, 4)
```

gives

```
[1] 0.2000 0.1600 0.1280 0.1024 0.0819 0.0655 0.0524
[8] 0.0419 0.0336 0.0268
```

In Example 10.4, to calculate the probability of getting the first bit in error in first bit received, second bit received, and so on,

```
x <- 0:9
prob <- dgeom(x, 0.01)
round(prob, 4)
```

gives

```
[1] 0.0100 0.0099 0.0098 0.0097 0.0096 0.0095 0.0094
[8] 0.0093 0.0092 0.0091
```

Notice that, in all these examples, the geometric probabilities tail off to zero as the number of trials increases.

In Example 10.1, the probability of obtaining an operational workstation in any one trial is very high at 0.95. Not surprisingly, the probability of having to go to the second trial for the first operational workstation drops sharply from 0.95 to 0.0475. After four trials, the probabilities are negligible. It is practically certain that you will get an operational workstation before you have to go to a fifth trial.

In Example 10.3, the distribution takes a little longer for the probabilities to become negligible, since the probability of a defective in any one trial is $p = 0.2$, substantially less than the $p = 0.95$ in Example 10.1. Even so, by the tenth inspection, there is just a 2.68% chance that you will still be waiting for the first defective. So while the geometric random variable is theoretically infinite, in practice, the probabilities do eventually become negligible.

A clearer picture emerges if we plot the pdfs. The *R* code

```

par (mfrow = c(2, 2))

x <- 0:4
plot(x+1, dgeom(x, prob = 0.95),
      xlab = "X = Number of trials", ylab = "P(X=x)",
      type = "h", main = "First workstation, p = 0.95",
      font.main = 1)

x <- 0:9
plot(x+1, dgeom(x, prob = 0.5),
      xlab = "X = Number of trials", ylab = "P(X=x)",
      type = "h", main = "First head, p = 0.5",
      font.main = 1)

x <- 0:19
plot(x+1, dgeom(x, prob = 0.2),
      xlab = "X = Number of trials", ylab = "P(X=x)",
      type = "h", main = "First defective, p = 0.2",
      font.main = 1)

x <- seq(0, 400, 50)
plot(x+1, dgeom(x, prob = 0.01),
      xlab = "X = Number of trials", ylab = "P(X=x)",
      type = "h", main = "First bit in error",
      p = 0.01, font.main = 1)

```

generates Fig. 10.2.

You will remember that `par(mfrow = c(2, 2))` writes the output in 2×2 matrix form, and `type = "h"` draws a horizontal line from the x axis to the probabilities. Should you need to join the points use `type = "o"`.

Observe from Fig. 10.2 that the geometric distribution is positively skewed, in all cases tailing off as X increases. When p is large, the probabilities tend to zero very quickly. As p decreases, the descent is not so steep. With $p = 0.5$, the probability is practically zero after $X = 10$; with $p = 0.2$, it takes longer to tail off and with $p = 0.01$, even at $X = 300$ there is still a nonnegligible probability of obtaining the first success.

10.2 CUMULATIVE DISTRIBUTION FUNCTION

Recall that the cumulative distribution function (cdf) is defined as

$$P(X \leq x) = P(X = 1) + P(X = 2) + \cdots + P(X = x)$$

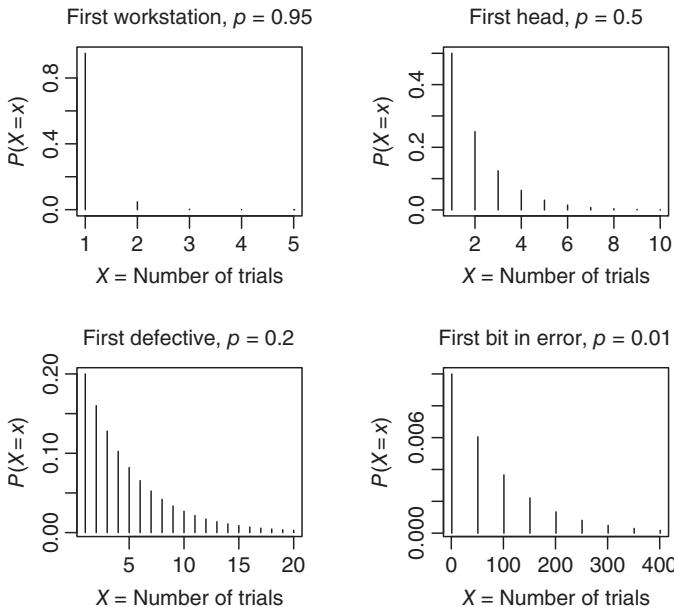


FIGURE 10.2 Geometric pdfs

Now, for the geometric random variable,

$$\begin{aligned} P(X = 1) + P(X = 2) + \cdots + P(X = x) &= p + qp + q^2p + \cdots + q^{x-1}p \\ &= p(1 + q + q^2 + \cdots + q^{x-1}) \end{aligned}$$

Now

$$1 + q + q^2 + \cdots + q^{x-1} = \frac{1 - q^x}{1 - q}$$

which is the sum of the first x terms of the geometric series with ratio q . Therefore,

$$\begin{aligned} P(X = 1) + P(X = 2) + \cdots + P(X = x) &= p \left(\frac{1 - q^x}{1 - q} \right) \\ &= p \left(\frac{1 - q^x}{p} \right) \\ &= 1 - q^x \end{aligned}$$

The *cdf* of the geometric distribution can, therefore, be written as

$$P(X \leq x) = 1 - q^x, \quad x = 1, 2, 3, \dots$$

which represents the probability that the first success occurs somewhere between 1 and x .

Example 10.8

If the defective rate of products is 0.03, what is the probability that the first defective occurs in the first five inspections?

$$\begin{aligned} P(X \leq 5) &= 1 - q^5 \\ &= 1 - (0.97)^5 \\ &= 0.141 \end{aligned}$$

△

10.2.1 Calculating Geometric cdfs in R

The function in *R* to calculate and plot the cdf is *pgeom* with the parameter *p* as an argument.

To check the calculations in Example 10.8.

```
pgeom(4, 0.03)
```

```
[1] 0.1412660
```

Let us now use *R* to calculate the cdfs of Examples 10.1–10.4.

Example 10.1, the probability of getting an operational workstation (*p* = 0.95)

```
x <- 0:4
prob <- pgeom(x, 0.95)
round(prob, 5)
```

calculates the cumulative probabilities, rounded to five decimal places.

```
[1] 0.95000 0.99750 0.99988 0.99999 1.00000
```

Example 10.2, the probability of getting the first head (*p* = 0.5)

```
x <- 0:9
round(pgeom(x, 0.5), 3)
```

gives

```
[1] 0.500 0.7500 0.875 0.938 0.969 0.984 0.992
[8] 0.996 0.998 0.999
```

Example 10.3, the probability of getting the first defective (*p* = 0.2)

```
x <- 0:9
round(pgeom(x, 0.2), 3)
```

gives

```
[1] 0.200 0.360 0.488 0.590 0.672 0.738 0.790
[8] 0.832 0.866 0.893
```

Example 10.4, the probability of getting the first bit in error ($p = 0.01$)

```
x <- 0:9
round(pgeom(x, 0.01), 3)
```

gives

```
[1] 0.010 0.020 0.030 0.039 0.049 0.059 0.068
[8] 0.077 0.086 0.096
```

Examining these figures, we see that when p is large, the probability is high that the first success occurs early on in the range of x . In Example 10.1, where $p = 0.95$, it is almost certain that the first success will occur in the first two trials, while in Example 10.4, where $p = 0.01$, there is less than a 10% chance that the first defective will occur in the first ten trials. We can look at the graphical displays of these cdfs using *R*.

```
par (mfrow = c(2,2))
x <- 0:4
plot(x+1, pgeom(x, prob = 0.95),
      xlab = "X = Number of trials", ylab = "P(X<=x)",
      type = "s", main = "First workstation, p = 0.95",
      font.main = 1)

x <- 0:9
plot(x+1, pgeom(x, prob = 0.5),
      xlab = "X = Number of trials", ylab = "P(X<=x)",
      type = "s", main = "First head, p = 0.5",
      font.main = 1)

x <- 0:19
plot(x+1, pgeom(x, prob = 0.2),
      xlab = "X = Number of trials", ylab = "P(X<=x)",
      type = "s", main = "First defective, p = 0.2",
      font.main = 1)

x <- seq(0, 399)
plot(x+1, pgeom(x, prob = 0.01),
      xlab = "X = Number of trials", ylab = "P(X<=x)",
      type = "s", main = "First bit in error",
      p = 0.01, font.main = 1)
```

generates Fig. 10.3

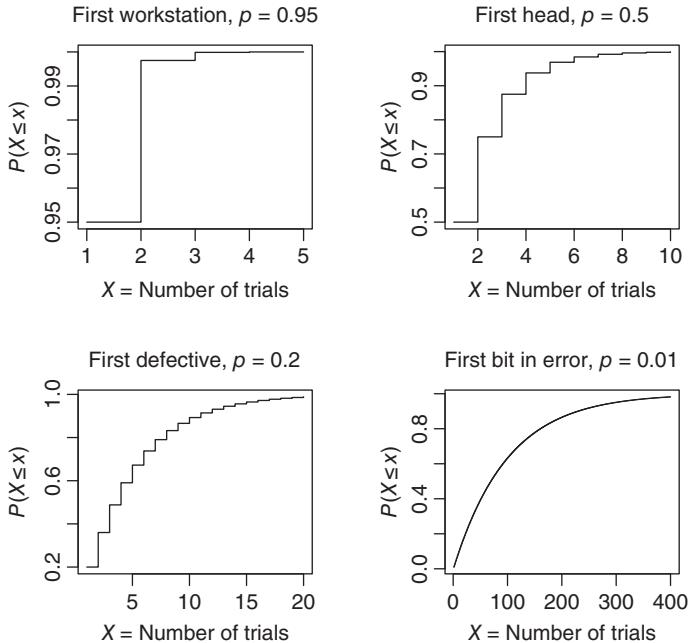


FIGURE 10.3 Geometric cdfs

Recall that `type = "s"` creates the “stair steps” effect, which is suitable for graphing discrete cdfs. Figure 10.3 shows how the rate at which $P(X \leq x)$ tends to 1 for various values of p , quickly with $p = 0.95$, and the rate decreases as the probability p of a success in any one trial decreases. In Example 10.4, where the probability of obtaining a communication bit in error is just 0.01, the cumulative probabilities increase very slowly; even after 100 trials there is just over a 60% that a bit is received which is in error.

10.3 THE QUANTILE FUNCTION

As well as calculating specific probabilities or cumulative probabilities, we might want to know how many inspections are necessary to be 75% certain say, that a success will occur.

In Example 10.3, the production line that has a 20% defective rate, we might ask, what is the minimum number of inspections, k say, that would be necessary so that the probability of observing a defective is at least 75%?

Choose k so that

$$P(X \leq k) \geq 0.75.$$

k is said to be the 75% quantile.

Generally, the $u\%$ quantile is the minimum k such that $P(X \leq k) \geq u/100$.

The quantile function is the inverse of the cdf. The 50% quantile is the median, and the 25% and 75% quantiles are the first and third quartiles, respectively.

For the geometric distribution, the $u\%$ quantile is the minimum k so that

$$P(X \leq k) = 1 - q^k \geq u/100$$

For example, to calculate the 75% quantile, we take $u = 75$ as given, and choose k so that

$$1 - q^k \geq 0.75$$

Let us look at Example 10.3, the production line that has a 20% defective rate, and obtain the 75% quantile, that is, the minimum number of inspections necessary so that the probability of observing a defective is at least 75%.

We can do this in *R*. The function `qgeom` obtains the quantiles. For the 75% quantile, where the defective rate $p = 0.2$, write

```
qgeom(0.75, 0.2)
```

to obtain

```
[1] 6
```

Notice that the first argument in `qgeom` is the required quantile, and the second argument is the probability of a success.

Recall that *R* gives the number of failures before the first success, that is, there are six failures before first success. This coincides with $X = 7$, the number of trials to the first success. There is at least a 75% chance of obtaining the first defective in the first seven inspections.

You can also do it by examining the cdf in *R*.

```
pgeom(6, 0.2)
[1] 0.7902848
```

means that there is approximately an 80% chance of getting the first defective by the seventh inspection.

```
pgeom(5, 0.2)
[1] 0.737856
```

means that there is approximately a 74% chance of getting a defective by the sixth inspection. So the minimum number of inspections necessary to be at least 75% sure that a defective will occur is seven, as we discovered by using `qgeom`.

10.4 GEOMETRIC EXPECTATIONS

It is not unreasonable to ask how long, on average, it will take to obtain the first success. Intuitively, it will depend on the probability of a success in any one trial; if this is high, then the average number of trials for the first success will be low, or early in the sequence of trials, and if p is small, we might have to wait a long time for the first success.

The expected number of trials to the first success is

$$\begin{aligned}\mu = E(X) &= \sum_{x=1}^{\infty} xp(x) \quad \text{denoting } P(X = x) \text{ by } p(x) \\ &= \sum_{x=1}^{\infty} xq^{x-1}p \\ &= p + 2qp + 3q^2p + 4q^3p + \dots\end{aligned}$$

Multiply this by q

$$q E(X) = qp + 2q^2p + 3q^3p + 4q^4p + \dots$$

Subtract

$$\begin{aligned}E(X) - q E(X) &= p + qp + q^2p + q^3p + q^4p + \dots \\ &= p(1 + q + q^2 + q^3 + \dots) \\ &= \frac{p}{1 - q} = 1\end{aligned}$$

recognizing $1 + q + q^2 + q^3 + \dots = 1/(1 - q)$ as the infinite geometric series with the first term one and with ratio $q = 1 - p$.

From this, we have

$$(1 - q)E(X) = 1$$

So the mean of the geometric distribution is

$$\mu = E(X) = \frac{1}{1 - q} = \frac{1}{p}$$

When $p = 0.5$, as in the coin tossing example (Example 10.2), the expected number of tosses to the first head is $1/(0.5) = 2$.

In the production line with 20% defective, that is, $p = 0.2$ (Example 10.3), an average of $1/(0.2) = 5$ inspections is necessary to get the first defective.

In the digital transmission system (Example 10.4), with 1% transmission error, an average of $1/(0.01) = 100$ bits is necessary to obtain one defective.

The average may be of help in quality control. In Example 10.3, the manufacturer might claim to produce at most 20% defective. We would expect that when testing the quality of the products, repeated testing, hourly say, would yield an average run of around five inspections to find the first defective; some runs will be less than five, others will be equal to, and others could be greater than five, but in the long term, the average number of inspections to finding the first defective should be very near five. If it is substantially greater than five, then we become suspicious of the manufacturer's claim. If it is substantially less, then the production quality is higher than the manufacturer is claiming.

Similarly, for bit transmission in Example 10.4, with a claim of 1% error in transmission, we would expect on the average, in the long run, to inspect about 100 bits until we find an error.

It is also reasonable to calculate how variable the number of trials to the first success is. For the geometric distribution

$$\begin{aligned}\sigma^2 &= E(X - \mu)^2 = \sum_{x=1}^{\infty} (x - \mu)^2 p(x) \\ &= \sum_{x=1}^{\infty} \left(x - \frac{1}{p} \right)^2 q^{x-1} p\end{aligned}$$

It can be shown that

$$\sigma^2 = \frac{q}{p^2}$$

The algebra to derive the variance is similar to what we used to derive the mean, and is given in Appendix C. Here, in the next section, we will show how an approximation may be obtained by simulation.

10.5 SIMULATING GEOMETRIC PROBABILITIES AND EXPECTATIONS

In this section, we show how to simulate a geometric distribution and to use the simulated results to obtain estimates of the mean and variance.

We look again at Example 10.2, tossing a coin to the first head. This time we estimate by simulation in *R*, the number of tosses to obtain the first head. We then calculate the average number of tosses that would be carried out if tosses continued until the first head occurred. We do this by generating random numbers from the appropriate geometric distribution using the *R* function

$$rgeom(n, p)$$

The first argument specifies the number of simulations to carry out, and the second argument is the parameter *p*. For example,

```
beforefirsthead <- rgeom(100, 0.5)
```

generates 100 random numbers from a geometric distribution with a parameter $p = 0.5$. This can be taken as simulating tossing a fair coin until a head occurs.

Since R gives the number of tosses before the first head, we add one to it to get the number of tosses to the first head.

```
tofirsthead <- beforefirsthead + 1
```

To see the output, type

```
tofirsthead
```

which returns

```
[1] 2 1 4 2 2 2 1 2 1 2 1 2 1 2 1 2 2 1 1 3 2 1 1 1 3 1 1 1 2 1 2 3 3 1 1 2 5
[38] 2 1 2 1 3 4 2 1 4 2 1 1 1 1 2 3 1 2 3 1 1 1 1 4 2 2 1 3 2 1 3 1 3 1 2 1 1
[75] 3 1 1 1 3 2 2 2 1 1 5 1 1 5 1 1 6 2 1 1 1 2 1 1 2 1 1
```

For clarity, obtain the frequency table.

```
table(tofirsthead)
```

gives

	1	2	3	4	5	6
50	30	12	4	3	1	

which shows that the first head occurs with varying frequencies between the first and sixth toss.

```
plot(table(tofirsthead)/100, xlab = "Number of tosses to
first head", ylab = "Relative frequency")
```

gives Fig. 10.4, which is the simulated distribution of tossing a coin until the first head.

The actual distribution is obtained in R with

```
x <- 0:6
plot(x+1, dgeom(x, 0.5), xlab = "First head",
ylab = "Probability", type = "h")
```

Figure 10.5 gives the actual distribution along with the simulated distribution.

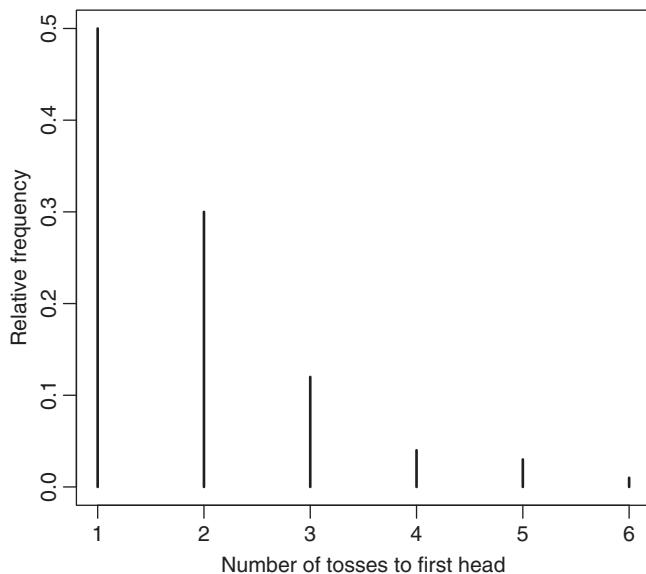


FIGURE 10.4 Simulated Number of Tosses to First Head

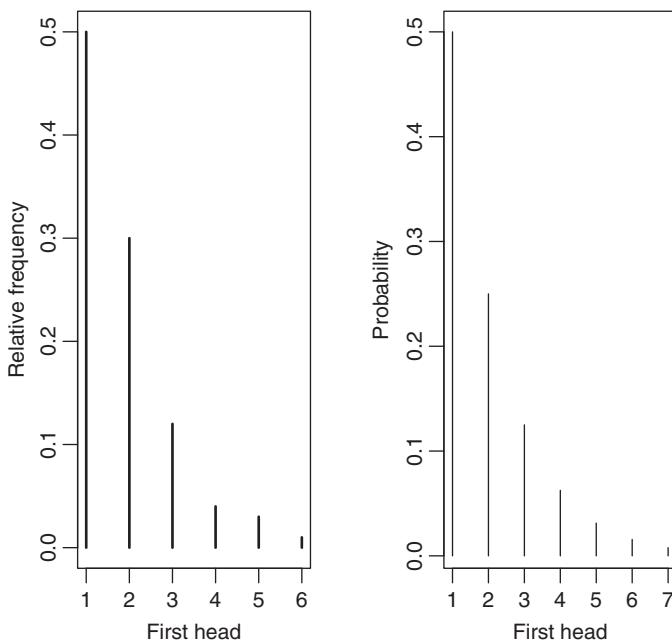


FIGURE 10.5 Simulated and Theoretical Geometric Distributions with $p = 0.5$

As you can see from Fig. 10.5, the simulated relative frequencies are close to the actual probabilities. The mean of the simulated number of tosses to first head is obtained with

```
mean(tofirsthead)
```

giving

```
[1] 1.83
```

The variance of the simulated tosses is

```
var(x)
```

which gives

```
1.19303
```

which are used to estimate the true mean $\mu = 1/p = 1/(0.5) = 2$ and the true variance $q/p^2 = 2$.

With a greater number of simulations, we would expect the estimated values of the parameters to be closer to the true values. It is no problem to carry out a very large number of simulations with modern computing facilities. Let us redo the analysis with 1,000 simulations.

```
beforefirsthead <- rgeom(1000, 0.5)
tobefirsthead <- beforefirsthead +1
```

generates 1,000 simulated results from the coin tossing experiment.

Again, to get the mean of the simulated data, write

```
mean(tobefirsthead)
```

which gives

```
[1] 1.999
```

The variance is obtained with

```
var(tobefirsthead)
```

which gives

```
[1] 2.025024
```

Notice that the mean and variance of the simulated values are much closer to the true parameters $\mu = 2$ and $\sigma^2 = 2$ than those obtained with 100 simulations. You will

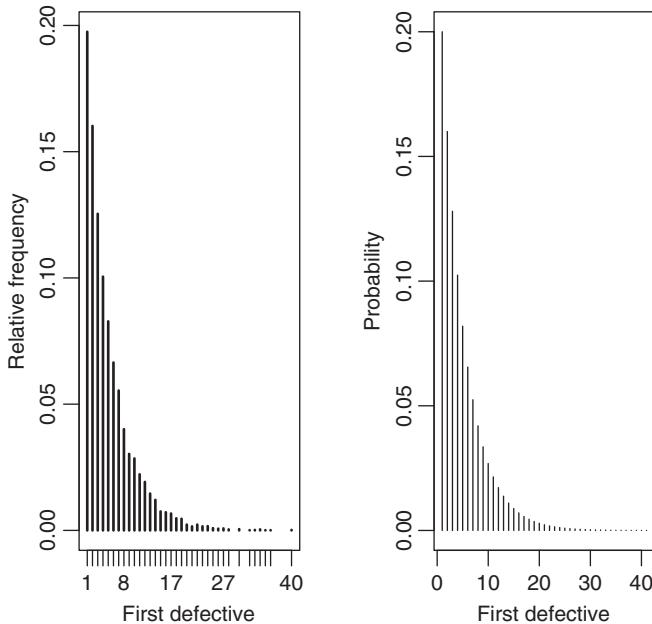


FIGURE 10.6 Simulated and Theoretical Geometric Distributions with $p = 0.2$

also have observed that 1,000 simulations did not take much longer than generating 100 – a tribute to the power of computers today.

Let us now return to Example 10.3, where a production line has a 20% defective rate, and carry out a similar experiment to estimate the number of inspections to the first defective. This time we simulate the experiment 10,000 times.

```
tofirstdefective <- rgeom(10000, 0.2) + 1
par(mfrow = c(1,2)) #multiframe row-wise
plot(table(tofirstdefective)/length(tofirstdefective),
     xlab = "First defective",
     ylab = "Relative frequency", type = "h")
x <- 0:40
plot(x+1, dgeom(x, 0.2), xlab = "First defective",
     ylab = "Probability", type = "h")
```

gives Fig. 10.6

As you can see, the simulated relative frequencies are close to the actual probabilities.

The mean of the simulated number of inspections to the first defective is obtained in R

```
mean(tofirstdefective)
```

giving

```
[1] 5.0594
```

which is not too far away from the theoretical mean $\mu = 1/p = 1/(0.2) = 5$.

The variance is obtained with

```
var(tofirstdefective)
```

which gives

```
[1] 20.3031
```

which again is close to the true variance $q/p^2 = 20$.

The standard deviation may be obtained with

```
sd(tofirstdefective)
```

gives

```
[1] 4.505896
```

Example 10.9

Amy is thinking of playing a game. The rules are as follows: A fair coin is tossed until the first head. If this occurs on k th toss, Amy gets 2^k euro. Simulate this game, and try to establish how much Amy should pay to play it.

The first head can occur in any toss, so Amy stands to win 2, or 2^2 , or 2^3 , or 2^4 , ... depending on the value of k , the number of tosses to the first head.

We simulate the process 50 times.

```
k <- rgeom(50, 0.5)
```

which gives

```
k
[1] 0 2 2 0 1 0 2 0 2 1 1 0 3 2 0 0 0 0 1 0 3 2 3 3 0 0 0 1 0 2 0 2 0 2 3 3 0 1
[39] 1 0 1 0 2 0 2 0 1 0 0 1
```

These figures represent the number of tosses before the first head. To obtain the number of tosses to the first head and to summarize, write

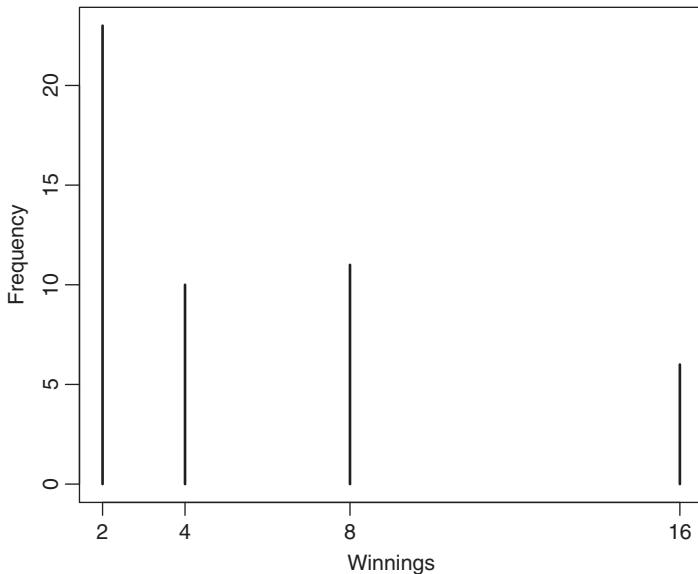


FIGURE 10.7 Amy's Winnings in 50 Plays

```
k <- k+1
table(k)
```

```
k
1 2 3 4
23 10 11 6
```

To calculate the actual winnings,

```
winnings <- 2**k
table(winnings)
2 4 8 16
23 10 11 6
```

To graph the winnings, write in *R*

```
plot(table(winnings), xlab = "Winnings",
     ylab = "Frequency")
```

which gives Fig. 10.7.

The average of Amy's winnings is

```
mean(winnings)
[1] 5.4
```

Thus, on average, Amy wins €5.4 per game. If she pays more than this to play, she will lose in the long run. \triangleleft

10.6 AMNESIA

There is a property of the geometric distribution which is indirectly responsible for a certain amount of financial hardship.

Example 10.10

Amy’s “partner in crime”, Ava, is betting on heads in the tossing of a fair coin, with an initial stake of €1. The odds are evens. After each successive tail, Ava doubles the stake, hoping to recoup her losses. After 10 tails in succession, what is her next stake, and what is the probability of losing it?

Powers of 2 increase faster than punters realize. After 10 tails in succession, Ava would have lost

$$1 + 2 + 2^2 + \cdots + 2^9 = 1,023$$

Her last stake was $2^9 = 512$, and she is now about to add another €1,024 to the losses. There is a 50% chance of losing at the 11th toss. The probability of a head in the next toss remains at 1/2, regardless of what has gone before.

After 10 tosses, all tails, Ava is down €1,023. If the next toss is a head, the winnings will be €1,024 giving an overall win of just €1. If the next toss is a tail, a loss of €1,024 will be incurred, bringing Ava’s overall loss to €2,047. \triangleleft

Coins have no memory. Unfortunately, we have a tendency to feel that, given that heads and tails are 50: 50, long sequences of tails are increasingly likely to be followed by heads. Gamblers notoriously up their stakes in such situations. Beware!

The memoryless property of the geometric distribution is known as the “Markov property,” in honor of the Russian mathematician Markov who was the first to notice it.

Theorem 10.1 The Markov property of the geometric distribution
If X is a geometric random variable, then for each integer n

$$P(X = x + n | X > n) = P(X = x) \quad (10.2)$$

□

Example 10.11

Products are inspected until the first defective is found. The first 10 inspections have been found to be free of defectives. What is the probability that the first defective will occur in the 15th inspection?

Let E_1 be the event that first 10 inspections are free of defectives.

Let E_2 be the event that that first defective will occur on the 15th inspection.

$$\begin{aligned}
 P(X = 15|X > 10) &= P(E_2|E_1) \\
 &= \frac{P(E_1 \cap E_2)}{P(E_1)} \\
 &= \frac{P((X = 15) \cap (X > 10))}{P(X > 10)} \\
 &= \frac{P(X = 15)}{P(X > 10)} \\
 &= \frac{q^{14}p}{q^{10}} \\
 &= q^4p \\
 &= P(X = 5)
 \end{aligned}$$

If the first 10 inspections were free of defectives, the probability that the first defective will occur in the next five is the same as the probability that the first defective will occur in the first five. It makes no difference to the probability that we have already had 10 inspections free of defectives. \triangleleft

Proof of Markov's property

Let

$$E_1 = \{X > n\}$$

$$E_2 = \{X = x + n\}$$

We may write

$$P(X = x + n|X > n) = P(E_2|E_1)$$

But

$$P(E_2|E_1) = \frac{P(E_1 \cap E_2)}{P(E_1)}$$

Now

$$P(E_1 \cap E_2) = P(X = x + n) = q^{x+n-1}p$$

and

$$P(E_1) = P(X > n) = q^n$$

Thus

$$\begin{aligned}
 P(E_2|E_1) &= \frac{q^{x+n-1}p}{q^n} \\
 &= q^{x-1}p
 \end{aligned}$$

But

$$P(X = x) = q^{x-1}p$$

giving Equation 10.2. \square

Example 10.12

The probability that Ava gains access to the Internet using her phone while traveling by train is 0.6 at each attempt. She keeps trying until access is achieved.

- (a) What is the probability that it takes no more than four attempts to get in?
- (b) If two attempts been made without gaining access, what is the probability that she will gain access in any of the next four attempts?
- (c) What is the minimum number of attempts necessary to be 95% certain that she gets in?

Solution

Let X be the number of attempts necessary to get in.

- (a) $P(X \leq 4)$ can be obtained in R with

```
pgeom(3, 0.6)
[1] 0.9744
```

remembering that R calculates the number of attempts before the first success.

- (b) $P(X \leq 6|X > 2) = P(X \leq 4) = 0.9744$ using the Markov property.
- (c) Choose k so that $P(x \leq k) \geq 0.95$ which is obtained in R with

```
qgeom(0.95, 0.6)
[1] 3
```

which means three attempts before a successful entry. Equivalently, with four attempts, Ava will be 95% certain of gaining entry. \triangleleft

10.7 SIMULATING MARKOV

You will recall that, in Section 10.5, we simulated a production process with 20% defectives, and we examined the pattern of defective occurrences. Let us return to these simulated data now, and investigate the Markov property.

This time, we will generate 10,000 random numbers from the geometric distribution with parameter $p = 0.2$

```
beforefirstdefective <- rgeom(10000, 0.2)
```

These numbers represent the number of inspections before a defective is found.

As usual, we add one and consider the number of inspections to find the first defective.

```
tofirstdefective <- beforefirstdefective + 1
```

The vector `tofirstdefective` contains 10,000 simulated number of inspections to obtain the first defective.

As before, we use X as the number of inspections to find the first a defective. To obtain an approximation of $P(X = 15|X > 10)$, we select all the values of X such that $X > 10$.

```
tofirstdefective10plus =
  subset(tofirstdefective, tofirstdefective > 10)
length(tofirstdefective10plus)
[1] 1128
```

There are 1,128 instances for which the first 10 items are found to be free of defectives. The number of inspections necessary to the next defective is obtained in *R* by creating a new variable.

```
y <- tofirstdefective10plus - 10

mean(y)
[1] 4.929078
var(y)
[1] 18.93374
sd(y)
[1] 4.351292
```

Comparing these to the parameters of the full set

```
mean(tofirstdefective)
[1] 5.036
var(tofirstdefective)
[1] 19.9999
sd(tofirstdefective)
[1] 4.472125
```

You will agree that there is not much difference between the two sets of parameters. These estimated parameters, in turn, are not too far away from the theoretical mean $\mu = 1/(0.2) = 5$ and variance $(0.8)/(0.2)^2 = 20$.

Figure 10.8 gives three distributions, the theoretical pdf $P(X = x)$, the simulated $P(X = x)$, and the simulated pdf after 10 inspections without a defective $P(X = x + 10|X > 10)$. For clarity, we limit the x -axis range to 20.

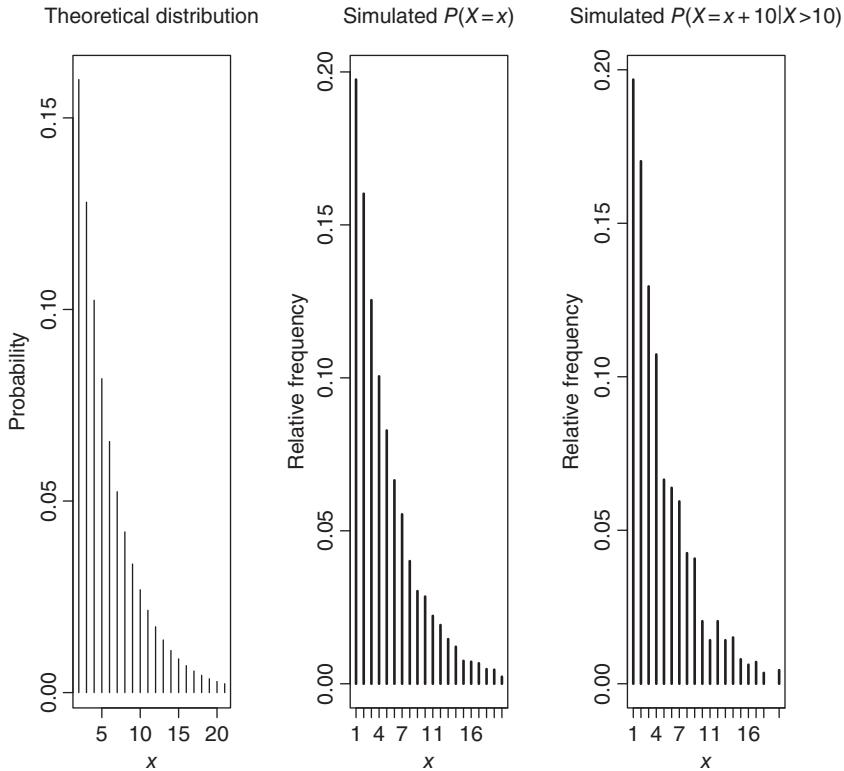


FIGURE 10.8 Geometric Distribution: Number of Inspections (X) to the First Defective

Figure 10.8 is obtained with the following *R* code.

```
par(mfrow = c(1, 3)) #multiframe  row-wise)
x <- seq(0:19)
plot(x+1, dgeom(x, prob = 0.2),
     ylab = "probability", xlab = "x",
     main = "Theoretical dist", type = "h", font.main = 1)
plot(table(tofirstdefective)/length(tofirstdefective),
     xlim = c(1,20),
     ylab = "Relative frequency", xlab = "x",
     main = "Simulated P(X=x)", type = "h", font.main = 1)
plot(table(tofirstdefective10plus-10)/length(tofirstdefective10plus),
     xlim = c(1,20),
     ylab = "Relative frequency", xlab = "x",
     main = "Simulated P(X=x+10|X>10)", type = "h",
     font.main = 1)
```

TABLE 10.1 Simulation Comparisons of the Geometric pdf

x	Theoretical $P(X = x)$	Simulated $P(X = x)$	x	Simulated $P(X = x X > 10)$
1	0.200	0.192	11	0.207
2	0.160	0.162	12	0.162
3	0.128	0.129	13	0.126
4	0.102	0.107	14	0.111
5	0.082	0.083	15	0.078
6	0.066	0.068	16	0.066
7	0.052	0.052	17	0.046
8	0.042	0.041	18	0.044
9	0.034	0.033	19	0.035
10	0.027	0.028	20	0.028

Examining Fig. 10.8, you will agree that the three diagrams are practically identical, which is no surprise now that we know about Markov.

Table 10.1 gives the first 10 terms of each distribution.

Table 10.1 confirms how close the distributions are. We could also obtain estimates of the cdf probabilities using the simulated data as follows:

- $P(X < 15|T > 10)$ may be approximated in R by first counting the numbers less than 15, in the conditional distribution stored in `submissions10plus`.

```
favorable <- length(subset(tofirstdefective10plus, tofirstdefective10plus < 15))
```

and expressing this amount as a proportion of the total number in `tostartdefective10plus`.

```
total <- length(tofirstdefective10plus)
```

to get an estimate of $P(x < 15|X > 10)$:

```
probability <- favorable/total
[1] 0.673
```

Compare this to $P(X < 5)$ estimated from the complete simulated population,

```
length(subset(tofirstdefective, tofirstdefective < 5))/10000
[1] 0.684
```

which is not too far away the theoretical $P(X < 15 | X > 10) = 0.742$.

The exact probability is

```
pgeom(4, 0.2)  
[1] 0.67232
```

EXERCISES 10.1

1. An industrial process manufacture components with a 10% defective rate. Each day the process is subjected to a quality control inspection; products are inspected sequentially until the first defective is obtained.
 - (a) What is the probability that the first defective will occur in one of the first three components inspected?
 - (b) What is the probability that more than three components will be inspected before a defective is found?
 - (c) If three products have been inspected without finding a defective, what is the probability that the next three inspections will be free of defectives?
 - (d) What is number of inspections expected until first defective?
2. Consider the number of blocks on the master file that are read before the first block is updated. Each day, the transaction file is run against the master file and approximately 4% of master blocks are updated.
 - (a) What is the probability that more than 10 blocks will be read until the first block is found that must be updated?
 - (b) If 10 blocks were read without needing to be updated, what is the probability that the first update will occur in the 14th block?
3. The probability of a successful optical alignment in an assembly of an optical data storage product is 0.9.
 - (a) What is the probability that the first successful alignment requires exactly four trials?
 - (b) What is the probability that the first successful alignment requires at most four trials?
 - (c) What is the probability that the first successful alignment requires at least four trials?
 - (d) What is the average number of trials necessary to obtain the first successful alignment?
4. A search engine goes through a list of sites for a given key phrase, and terminates as soon as the key phrase is found. Each site has a 60% chance of containing the key phrase.
 - (a) What is the probability of finding the key phrase in the third site?

- (b) How many sites, on average, will need to be searched in order to find the key phrase?
 - (c) What is the minimum number of sites that will need to be searched to be 99% sure of finding the key phrase?
5. The probability of being able to log on to a computer system from a remote workstation during a busy working day is 0.8. What is the probability
- (a) of first getting in on the second attempt?
 - (b) that it will take more than three attempts to get in?
6. Jane and Ava are tossing a fair coin. Whoever gets the first head wins. As Ava is the younger, she starts first. Calculate their respective probabilities of winning.

10.8 PROJECTS

1. With your own computing experience, develop a front end to *R*, which provides the user with an opportunity to invoke the geometric distribution by clicking an icon on screen, to input the relevant value of the parameter p and the values of the variable x , and to obtain plots of the pdf and cdf.

Be sure to provide validity checks, for example, that p is in the interval (0, 1), and that x is a positive integer.

2. By generating 10,000 searches in *R*, carry out a simulation experiment for a search engine going through a list of sites for a given key phrase, until the key phrase is found. You should allow your program to input the probability p that any site will contain the key phrase.

(a) Plot the simulated pdf and calculate its mean and variance.

(b) Obtain the simulated conditional distribution of searches when three searches have been carried out without success. Calculate its mean and variance, and satisfy yourself that they are equivalent to the simulated distribution of the complete set.

As test data assume each site has a 60% chance of containing the key phrase.

To satisfy yourself that the Markov memoryless property holds, obtain estimates of

- (a) $P(X = 4|X > 3)$ and $P(X = 1)$
- (b) $P(X = 5|X > 3)$ and $P(X = 2)$

where X is the number of searches to the first success

3. If all families have children until the first girl, estimate by simulation, the average number of children per family.
4. In Examples 10.1–10.4:

- (a) Calculate the first 10 values of the pdf and cdf using `dgeom` and `pgeom`, and use the *round* function to write the output to three decimal places;

- (b) Use the `plot` function to provide a graphical display of the probability distribution function (pdf) and the cumulative density function (cdf);
- (c) Simulate the pdfs and cdfs using `rgeom`;
- (d) Compare the simulated pdfs and cdfs with their theoretical equivalents;
- (e) Estimate, by simulation, the average and standard deviation of each distribution.

11

THE BINOMIAL DISTRIBUTION

Example 11.1

Five workstations are attached to a server on a network. The probability that any workstation is operational is 0.95.

Let X denote the number of operational workstations.

$$P(X = 0) = P(\text{no operational workstation in 5})$$

$$P(X = 1) = P(1 \text{ operational workstation in 5})$$

$$P(X = 2) = P(2 \text{ operational workstations in 5})$$

□

Example 11.2

A fair coin is tossed 10 times.

Let X be the number of heads obtained in 10 tosses of the coin.

$$P(X = 0) = P(\text{no head in 10 tosses})$$

$$P(X = 1) = P(1 \text{ head in 10 tosses})$$

$$P(X = 2) = P(2 \text{ heads in 10 tosses})$$

□

Example 11.3

It is known that 20% of integrated circuit chips on a production line are defective. To maintain and monitor the quality of the chips, a sample of 20 chips is selected at regular intervals for inspection.

Let X denote the number of defectives found in the sample.

$$P(X = 0) = P(\text{no defective in a sample of size } 20)$$

$$P(X = 1) = P(1 \text{ defective in a sample of size } 20)$$

$$P(X = 2) = P(2 \text{ defectives in a sample of size } 20)$$

△

Example 11.4

It is known that 1% of bits transmitted through a digital transmission are received in error. One hundred bits are transmitted each day.

Let X denote the number of bits found in error each day.

$$P(X = 0) = P(\text{no error in the } 100 \text{ transmissions})$$

$$P(X = 1) = P(1 \text{ errors in the } 100 \text{ transmissions})$$

$$P(X = 2) = P(2 \text{ errors in the } 100 \text{ transmissions})$$

△

In each of the aforementioned examples, an experiment is carried out a number of times to observe the number of occurrences of the event of interest which, as with the geometric distribution, is called a “success.”

Observe that these experiments are the same as those that we considered in the examples at the beginning of Chapter 10, where we investigated the number of trials necessary to arrive at the first success. Now we examine the number of successes in a fixed number of trials. If we denote the number of trials by n and, as before, the probability of a success in one trial by p , then

$$n = 5 \text{ and } p = 0.95 \text{ in Example 11.1}$$

$$n = 10 \text{ and } p = 0.50 \text{ in Example 11.2}$$

$$n = 20 \text{ and } p = 0.20 \text{ in Example 11.3}$$

$$n = 100 \text{ and } p = 0.01 \text{ in Example 11.4}$$

11.1 BINOMIAL PROBABILITIES

Let us work out the probabilities for Example 11.1, where the experiment is to poll the five workstations to establish if they are operational; each workstation has a probability of 0.95 of being operational, that is, the probability of a success (operational) with any workstation is 0.95, and the probability of a failure (not operational) is 0.05.

$P(X = 0)$ is the probability of finding none of the workstations operational, that is, five failures.

$$FFFFF$$

Since the probability of a failure is 0.05 for each workstation, then

$$P(X = 0) = (0.05)^5$$

For $P(X = 1)$ is the probability of exactly one of the five workstations being operational. There are five possible ways of this happening.

$$SFFFF, FSFFF, FFSFF, FFFSF, FFFFS$$

Here, *SFFFF* means that the first workstation is operational and the other four are not operational, *FFSFF* means the first two workstations are not operational, the third is operational, and the last two are not operational, and so on.

Each of the five different ways of getting one operational workstation has a probability

$$(0.95)^1(0.05)^4$$

and they are mutually exclusive. So, $P(X = 1)$ is obtained by adding each possible way of getting exactly one operational workstation.

$$P(X = 1) = 5(0.95)^1(0.05)^4 \quad (11.1)$$

Note that the number of different ways of getting one operational workstation in five, is the number of combinations of one from five, that is, $\binom{5}{1} = 5$. So, we could write Equation 11.1 as

$$P(\text{exactly 1 operational workstation in 5}) = \binom{5}{1} (0.95)^1(0.05)^4$$

For $P(X = 2)$, the probability of finding exactly two of the five workstations operational:

SSFFF, SFSFF, SFFSF, SFFFS, FSSFF,

FSFSF, FSFFS, FFSSF, FFSFS, FFFSS

Clearly, there are 10 different ways of finding two operational workstations in the five, and each of these has probability

$$(0.95)^2(0.05)^3$$

Since the different ways are mutually exclusive, the probabilities of occurrence can be added together to give

$$P(X = 2) = 10(0.95)^2(0.05)^3 \quad (11.2)$$

Note that the number of different ways of getting two operational workstations in five is the number of combinations of two from five, that is, $\binom{5}{2} = 10$. So we can write Equation 11.2 as

$$P(\text{exactly 2 operational workstations in 5}) = \binom{5}{2} (0.95)^2(0.05)^3$$

In a similar fashion, we calculate

$$\begin{aligned} P(X = 3) &= P(\text{exactly 3 of the 5 workstations operational}) = \binom{5}{3} (0.95)^3 (0.05)^2 \\ P(X = 4) &= P(\text{exactly 4 of the 5 workstations operational}) = \binom{5}{4} (0.95)^4 (0.05)^1 \\ P(X = 5) &= P(\text{all 5 workstations operational}) = (0.95)^5 \end{aligned}$$

This is an example of a *binomial distribution*, so called because the successive probabilities are the terms of the binomial expansion of $(p + q)^5$, where $q = 1 - p$.

$$\begin{aligned} (p + q)^5 &= q^5 + \binom{5}{1} pq^4 + \binom{5}{2} p^2 q^3 + \binom{5}{3} p^3 q^2 + \binom{5}{4} p^4 q + p^5 \\ &= P(X = 0) + P(X = 1) + P(X = 2) + P(X = 3) + P(X = 4) + P(X = 5) \end{aligned}$$

11.2 BINOMIAL RANDOM VARIABLES

Definition 11.1 *Binomial random variable*

A binomial random variable X is one which is represented as the number of successes in n trials of an experiment in which

1. Each trial has two possible outcomes:
 - A success with probability p ;
 - A failure with probability $q = 1 - p$.
2. Repeated trials are independent.

The probability of getting exactly x successes in n trials is

$$P(X = x) = \binom{n}{x} p^x q^{n-x}$$

since there are $\binom{n}{x}$ mutually exclusive ways of getting exactly x successes in n , and each of these ways has a probability of $p^x q^{n-x}$.

The probability density function (pdf) is

$$p(x) = \binom{n}{x} p^x q^{n-x}, \quad x = 0, 1, 2, \dots, n$$

which is the probability that exactly x successes will occur in the n trials. □

Explicitly,

$$\begin{aligned}
 P(X = 0) &= P(0 \text{ successes in } n \text{ trials}) &= q^n \\
 P(X = 1) &= P(\text{exactly 1 success in } n \text{ trials}) &= \binom{n}{1} p^1 q^{n-1} \\
 P(X = 2) &= P(\text{exactly 2 successes in } n \text{ trials}) &= \binom{n}{2} p^2 q^{n-2} \\
 P(X = 3) &= P(\text{exactly 3 successes in } n \text{ trials}) &= \binom{n}{3} p^3 q^{n-3} \\
 &\vdots \\
 P(X = n) &= P(\text{All } n \text{ successes}) &= p^n
 \end{aligned}$$

Since X , the number of successes in n trials, can take any integer value in the range $[0, n]$, this random variable is said to be discrete and finite.

Observe again that

$$q^n, \quad \binom{n}{1} p^1 q^{n-1}, \quad \binom{n}{2} p^2 q^{n-2}, \dots, \quad \binom{n}{n-1} p^{n-1} \cdot q, \quad p^n$$

are the terms of the binomial expansion of $(p + q)^n$.

$$\begin{aligned}
 \sum_{i=0}^n \binom{n}{i} p^i q^{n-i} &= (p + q)^n \\
 &= 1, \quad \text{since } p + q = 1
 \end{aligned}$$

which indicates that it is a valid pdf.

11.2.1 Calculating Binomial pdfs with R

It is easy to calculate the binomial probabilities in R. Simply prefix the shortened name *binom*, with “d” for the pdf. The function `dbinom`, along with the parameters n and p as arguments, is used to calculate the probabilities.

In Example 11.1, the probability of getting exactly three operational workstations from five is

```
dbinom(x = 3, size = 5, prob = 0.95)
```

or simply

```
dbinom(3, 5, 0.95)
```

which gives

```
[1] 0.02143438
```

The probabilities of finding 0, 1, 2, 3, 4, 5 operational workstations can be obtained by putting the integers into a vector and using `dbinom`.

```
x <- 0:5
```

and

```
dbinom(x, size = 5, prob = 0.95)
[1] 0.0000003125 0.0000296875 0.0011281250 0.0214343750 0.2036265625
[6] 0.773780937
```

In Example 11.2, the probabilities of getting 0, 1, 2, ..., 10 heads in 10 tosses of a coin are

```
x <- 0:10
dbinom(x, 10, 0.5)
[1] 0.0009765625 0.0097656250 0.0439453125 0.1171875000 0.2050781250
[6] 0.2460937500 0.2050781250 0.1171875000 0.0439453125 0.0097656250
[11] 0.0009765625
```

For a neater output, we could round to four decimal places using

```
round(dbinom(x, 10, 0.5), 4)
[1] 0.0010 0.0098 0.0439 0.1172 0.2051 0.2461 0.2051 0.1172 0.0439 0.0098
[11] 0.0010
```

Similarly, for Examples 11.3 and 11.4, the probabilities can be generated using respectively,

```
dbinom(x, 20, 0.2)
```

and

```
dbinom(x, 100, 0.01)
```

11.2.2 Plotting Binomial pdfs in R

To examine the pattern of the binomial probabilities with different parameters, we plot the pdfs using the following R code.

```
par(mfrow = c(2,2)) # multiframe

x <- 0:5 #Example 11.1
plot(x, dbinom(x, size = 5, prob = 0.95),
      xlab = "Number of operational workstations",
      ylab = "P(X = x)", type = "h",
      main = "Workstations, n = 5, p = 0.95", font.main = 1)
```

```

x <- 0:10 #Example 11.2
plot(x, dbinom(x, size = 10, prob = 0.5),
      xlab = "Number of heads",
      ylab = "P(X = x)", type = "h",
      main = "Heads, n = 10, p = 0.5", font.main= 1)

x <- 0:20 #Example 11.3
plot(x, dbinom(x, size = 20, prob = 0.2),
      xlab = "Number of defectives",
      ylab = "P(X = x)", type = "h",
      main = "Defectives, n = 20, p = 0.2", font.main = 1)

x <- 0:20 #Example 11.4. No need for the full range of 100
plot(x, dbinom(x, size = 100, prob = 0.01),
      xlab = "Number of bits in error",
      ylab = "P(X = x)", type = "h",
      main = "Bits in error, n = 100, p = 0.01", font.main = 1)

```

generates Fig. 11.1.

Figure 11.1 shows that the first pdf is negatively skewed ($p = 0.95$), while the second one in the top row appears to be perfectly symmetric and bell-shaped

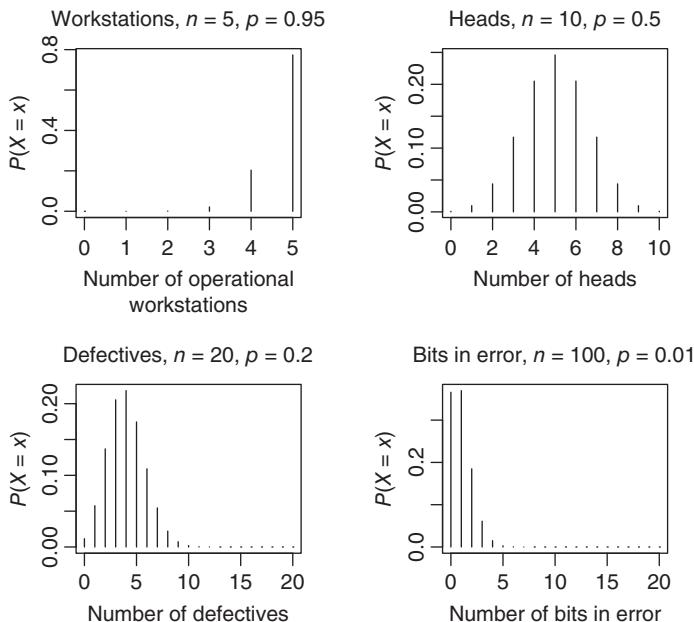


FIGURE 11.1 Binomial pdfs

($p = 0.5$). The first pdf on the second row is near symmetric ($p = 0.2$), while the last pdf is positively skewed ($p = 0.01$). These differences are due, among other things, to the varying values of p . We will discuss this further in later chapters.

11.3 CUMULATIVE DISTRIBUTION FUNCTION

Recall that the cumulative distribution function (cdf) is defined as

$$P(X \leq x) = P(X = 0) + P(X = 1) + P(X = 2) + \cdots + P(X = x)$$

which for the binomial distribution is

$$P(X \leq x) = q^n + \binom{n}{1} p^1 q^{n-1} + \cdots + \binom{n}{x} p^x q^{n-x}$$

Looking at Example 11.1 again, the workstations on a network, we might ask:
 What is the probability that at most three workstations will be operational?
 Let X be the number of workstations operational, and recall that the probability that any workstation is operational is 0.95.

$$\begin{aligned} & P(\text{less than or equal to 3 workstations will be operational}) \\ &= P(X = 0) + P(X = 1) + P(X = 2) + P(X = 3) \\ &= (0.05)^5 + \binom{5}{1} (0.95)^1 (0.05)^4 + \binom{5}{2} (0.95)^2 (0.05)^3 + \binom{5}{3} (0.95)^3 (0.05)^2 \\ &= 0.0000003 + 0.00003 + 0.00113 + 0.02143 \\ &\approx 0.0226 \end{aligned}$$

Also,

$$\begin{aligned} & P(4 \text{ or more workstations will be operational}) = P(X \geq 4) \\ &= 1 - P(X \leq 3) \\ &\approx 1 - 0.0226 \\ &= 0.9774 \end{aligned}$$

11.3.1 Calculating Binomial cdfs in R

The function in R used to calculate the cumulative distribution is *pbinom*.

Let us quickly solve the previous example with *R*.
 $P(X \leq 3)$ is obtained with

```
pbinary(3, 5, 0.95)
[1] 0.0225925
```

and $P(X > 3)$ is

```
1 - pbinary(3, 5, 0.95)
[1] 0.9774075
```

Returning to Example 11.3, where 20% of integrated circuit chips made at a plant are defective. Samples of size 20 of these chips are selected at regular intervals.

```
pbinary(4, size = 20, prob = 0.2)
0.6296483
```

calculates the probability that a sample will contain at most four defective chips. There is almost a 63% chance that a sample chosen at random contains four or less defectives. Complementarily, the probability that a sample will contain more than four defective chips is

$$P(X > 4) = 1 - P(X \leq 4)$$

In *R*

```
1 - pbinary(4, size = 20, prob = 0.2)
[1] 0.3703517
```

There is just over a 37% chance that a sample will contain five or more defectives.

To obtain the cumulative probabilities, that is, $P(X \leq x)$ for all x between 0 and 20, write

```
x <- 0:20
cumprob <- pbinary(x, size = 20, prob = 0.2)
```

To round the output to four decimal places, write

```
round(cumprob, 4)
[1] 0.0115 0.0692 0.2061 0.4114 0.6296 0.8042 0.9133 0.9679 0.9900
[10] 0.9974 0.9994 0.9999 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
[19] 1.0000 1.0000 1.0000
```

11.3.2 Plotting cdfs in R

To plot the cdfs of Examples 11.1–11.4 write in R

```

par(mfrow = c(2,2)) #multiframe
x <- 0:5
plot(x, pbinom(x, size = 5, prob = 0.95),
      xlab = "Number of operational workstations",
      ylab = "P(X <= x)", ylim = c(0, 1), type = "s",
      main = "Workstations, n = 5, p = 0.95", font.main = 1)

x <- 0:10
plot(x, pbinom(x, size = 10, prob = 0.5),
      xlab = "Number of heads",
      ylab = "P(X< = x)", ylim = c(0, 1), type = "s",
      main = "Heads, n = 10, p = 0.5", font.main = 1)

x <- 0:20
plot(x, pbinom(x, size = 20, prob = 0.2),
      xlab = "Number of defectives",
      ylab = "P(X <= x)", ylim = c(0, 1), type = "s",
      main = "Defectives, n = 20, p = 0.2", font.main = 1)

x <- 0:100
plot(x, pbinom(x, size = 100, prob = 0.01),
      xlab = "Number of bits in error",
      ylab = "P(X <= x)", ylim = c(0, 1), type = "s",
      main = "Bits in error, n = 100, p = 0.01", font.main = 1)

```

This generates Fig. 11.2.

Notice from Fig. 11.2 that there is approximately a 40% chance that there will be no error in 100 bits transmitted. This is not surprising since the probability than any bit being in error is just 1%. On the other hand, concerning the workstations, each of which has a 0.95 chance of being operational, it is very unlikely that less than three of the five will be operational.

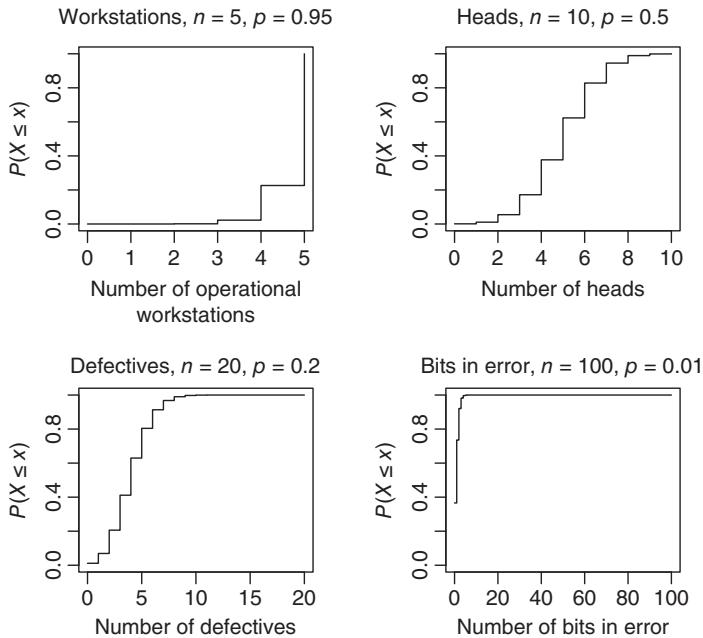
11.4 THE QUANTILE FUNCTION

In Example 11.3, the samples of integrated chips, we might ask:

Up to how many defectives will the samples contain with at least 95% certainty?

We must choose k so that

$$P(X \leq k) \geq 0.95$$

**FIGURE 11.2** Binomial cdfs

In *R*

```
qbinom(0.95, 20, 0.2)
[1] 7
```

This means that there is at least a 95% certainty that the samples contain less than or equal to seven defective chips. Put another way, if chips from this production process were packed in batches of size 20, then at least 95% of these batches would contain less than or equal to seven defectives.

This can be confirmed from the cdf. In *R*,

```
pbinom(6, 20, 0.2)
[1] 0.9133075
```

which means that $P(X \leq 6) < 0.95$.

Also from *R*,

```
pbinom(7, 20, 0.2)
[1] 0.9678573
```

which means that $P(X \leq 7) > 0.95$.

Therefore, $k = 7$ is the minimum value to satisfy $P(X \leq k) \geq 0.95$.

Example 11.5

A student laboratory consists of 50 desktop computers on a network. Each computer has a probability of 0.01 of failing in any week. How many students can be accommodated in a lab session in order to be 95% sure that there will be a computer available for every student?

Let X be the number of breakdowns. We want to choose k so that

$$P(X \leq k) \geq 0.95$$

Using the quantile function in R ,

```
qbinom(0.95, 50, 0.01)
[1] 2
```

which means that there is more than a 95% chance that no more than two computers will break down each week. So, if 48 students attend a lab session, they will all get a computer with 95% certainty. \triangleleft

Example 11.6

Suppose there are n frames per packet, and the probability that a frame gets through without an error is 0.999. What is the maximum size that a packet can be so that the probability that it contains no frame in error is at least 0.9?

Let X be the number of frames in error in a packet of size n . The probability that there are no frames in error in a packet is

$$P(X = 0) = (0.999)^n$$

For this to be greater than 0.9, we need to choose n so that

$$(0.999)^n > 0.9$$

Let us plot $P(X = 0)$ for values of n in the range [1:200]. Write in R

```
n <- 1:200
plot(n, 0.999^n, type = "l",
      xlab = "Number of frames in a packet",
      ylab = "P(Frames in error = 0)",
      ylim = c(0.9, 1))
```

yields Fig. 11.3

Here, `ylim = c(0.9, 1)` plots the probabilities from 0.9 to 1. Notice in Fig. 11.3 that when the packet size is just over 100, the probability of it containing no frames in error is approximately 0.9. To check further, we have

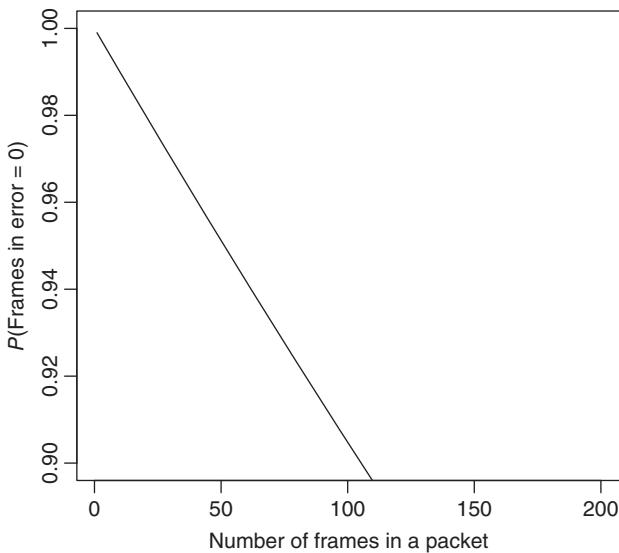


FIGURE 11.3 Probabilities of Error-Free Packets Containing Different Numbers of Frames

```

0.999^103
[1] 0.9020805
0.999^104
[1] 0.9011784
0.999^105
[1] 0.9002772
0.999^106
[1] 0.899377

```

When the packet size reaches 106, the probability of it being error free is below 90%. \triangle

11.5 RELIABILITY: THE GENERAL SYSTEM

In Chapter 8, we considered the reliability of series and parallel systems with connected components assuming that the system fails only if all the components in a gateway fails and works if at least one component works, that is, the Con/k:n:F system with $k = 1$. We are able to extend this now to the general system with $k > 1$ using the binomial distribution.

Example 11.7

A system consists of five components and will operate only if two components or more are working. If each component operates independently and has a probability of 0.1 of failing, what is the reliability of the system?

The reliability of the system is

$$\text{Rel} = P(\text{at least 2 components work})$$

This can be calculated in *R* using the `pbinom` function.

```
1 - pbinom(1, 5, 0.9)
[1] 0.99144
```

Generally, for any system consisting of n components operating independently, with p the reliability of each component and k the required number of working components:

- $1 - \text{pbinom}(k-1, n, p)$ is the reliability of the system;
- $\text{pbinom}(k-1, n, p)$ is the probability that the system will fail. △

Example 11.8

A system consists of 100 components operating independently, and each component has a 40% chance of failing. The system will work if at least 50 of the components are working. Calculate:

- the reliability of the system;
- the probability that the system will fail.

The system works provided 50 or more components work, Using *R*, the reliability can be calculated to be

```
1 - pbinom(49, 100, 0.6)
0.9832383
```

Conversely, the probability that the system will fail is

```
pbinom(49, 100, 0.6)
[1] 0.01676169
```

△

11.5.1 The Series–Parallel General System

As we saw in Chapter 8, most practical systems consists of more than one set of components arranged in series.

Example 11.9

On an integrated circuit board, there are three types of IC chips replicated and arranged in series.

The first type of chip has a reliability of 0.5 and is backed up 100 times.
 The second has a reliability of 0.6 and is backed up 30 times.
 The third has a reliability of 0.8 and is backed up 10 times.
 The circuit board will work provided, at most 29 of the first, at most four of the second and at most one of the third type of chip, fail.
 If it can be assumed that the chips fail independently, obtain the reliability of the circuit board.

Solution

Let C_1 be the event that the first chip works. $P(C_1) = 0.5$
 Let C_2 be the event that the second chip works. $P(C_2) = 0.6$
 Let C_3 be the event that the third chip works. $P(C_3) = 0.8$
 The reliability of the system is

$$\begin{aligned} \text{Rel} &= P(\text{at least } 30 \text{ of } C_1) \times P(\text{at least five of } C_2) \times P(\text{at least two of } C_3) \\ &= (1 - P(C_1 \leq 29)) \times (1 - P(C_2 \leq 4)) \times (1 - P(C_3 \leq 1)) \end{aligned}$$

which can be calculated in R .

```
(1 - pbinom(29, 100, 0.5)) * (1 - pbinom(4, 30, 0.6)) * (1 - pbinom(1, 10, 0.8))
[1] 0.9999795
```

The integrated circuit board has a reliability of over 99.99%. \triangleleft

Example 11.10

Suppose again that there are three types of IC chips replicated and arranged in series. This time, the first type of chip has a reliability of just 0.3, the second chip has a reliability of 0.4, and the third chip has a reliability of 0.7. For the system to work, at least 30 of the first chip, at least 5 of the second chip and at least 2 of the third chip must function.

Use R to design, by experimentation, the system so that there is an overall reliability of at least 99%.

If we used the same backup system as in the previous example, that is, 100, 30, and 10 of the first, second, and third chips, respectively, the overall reliability would be very low. Let us check in R .

```
(1 - pbinom(29, 100, 0.3)) * (1 - pbinom(4, 30, 0.4)) * (1 - pbinom(1, 10, 0.7))
```

gives

```
[1] 0.5367712
```

which is far too low. We need to backup more or improve the reliability of individual chips.

For the first chip, the reliability is

```
(1 - pbinom(29, 100, 0.3))
[1] 0.5376603
```

which is too low. Let us increase the number of chips to 200 (assuming of course that this is possible).

```
1 - pbinom(29, 200, 0.3)
[1] 0.9999997
```

which is probably unnecessarily high. Let us try 150.

```
1 - pbinom(29, 150, 0.3)
[1] 0.9978915
```

Looking at the second chip

```
1 - pbinom(3, 30, 0.4)
[1] 0.9996867
```

which appears to be fine.

Finally, with the third chip

```
1 - pbinom(1, 10, 0.7)
[1] 0.9998563
```

So, if we add more of the first chip; increase the number of them from 100 to 150, and calculate the overall reliability, we get

```
(1 - pbinom(29, 150, 0.3)) * (1 - pbinom(3, 30, 0.4)) * (1 - pbinom(1, 10, 0.7))
[1] 0.9974355
```

a reliability of 99.74355 which more than satisfies the design specifications. ◀

11.6 MACHINE LEARNING

In Chapter 6, we showed how Bayes' theorem may be used in supervised learning in which a learning algorithm uses a training set of examples and outputs a classifier to classify new examples. The classifier is used to classify into predefined classes. Sometimes, the individual decisions of an ensemble of classifiers are combined to classify new examples in order to improve the classification accuracy.

Example 11.11

Suppose there are three independent classifiers used to classify a new example. The probability that any of these classifiers correctly classifies the new example is 0.7, and therefore 0.3 of making an error. If a majority decision is made, what is the probability that the new case will be correctly classified?

Let X be the number of correct classifications made by the three classifiers. Then a majority vote means $X \geq 2$. Because the classifiers are independent, X follows a binomial distribution, with parameters $n = 3$ and $p = 0.7$.

The probability of correctly classifying a new example is

$$\begin{aligned} P(X \geq 2) &= P(X = 2) + P(X = 3) \\ &= \binom{3}{2} (0.7)^2 (0.3) + (0.7)^3 \end{aligned}$$

```
1 - pbinom(1, 3, 0.7)
0.784
```

We have improved the probability of a correct classification from 0.7 with one classifier to 0.784 with three. Conversely, we have decreased the probability of an incorrect classification from 0.3 with one classifier to 0.216 with three. \triangleleft

It would appear that by increasing the number of classifiers, we can improve the classification accuracy further. Let us use R to investigate. With five classifiers, the probability of three or more correct classifications is

```
1 - pbinom(2, 5, 0.7)
[1] 0.83692
```

meaning that there is an 83.69% chance that a correct classification will be made. Conversely, the probability of three or more classifiers being simultaneously wrong is 0.163, which is much less than the error rate of 0.3 of the individual classifiers.

The following R code calculates the probabilities of correct classifications when the number of classifiers is three, five, seven, and so on up to 29.

```
n <- seq(3, 29, 2)
majority <- (n+1)/2
probmajority <- 1 - pbinom(majority-1, n, 0.7)
```

To view the output, write

```
round(probmajority, 4)
[1] 0.7840 0.8369 0.8740 0.9012 0.9218 0.9376 0.9500 0.9597 0.9674 0.9736
[11] 0.9786 0.9825 0.9857 0.9883
```

Clearly, the probability of a correct classification increases with the number of classifiers. To get a clear picture, we could graph the probabilities.

```
plot(n, probmajority, xlab = "Number of classifiers",
      ylab = "Probability of a correct classification",
      ylim = c (0.7, 1), type = "h")
```

generates Fig. 11.4.

Notice that in Fig. 11.4 the probabilities increase with the number of classifiers, but notice also that the “law of diminishing returns” operates. After about $n = 21$ or so classifiers, the increase in the probabilities is very little.

Of course, if the individual classifiers make errors at rates exceeding 0.5, the error rate of the voted ensemble will increase as a result of the voting. Let us use *R* to illustrate.

For $n = 21$ classifiers, let us calculate the probability that a majority decision is in error for various values of p , where p is the probability that any one classifier is in error.

```
p <- seq(0.1, 0.9, 0.1)
```

puts values into p in the range from 0.1 to 0.9 in intervals of 0.1, that is

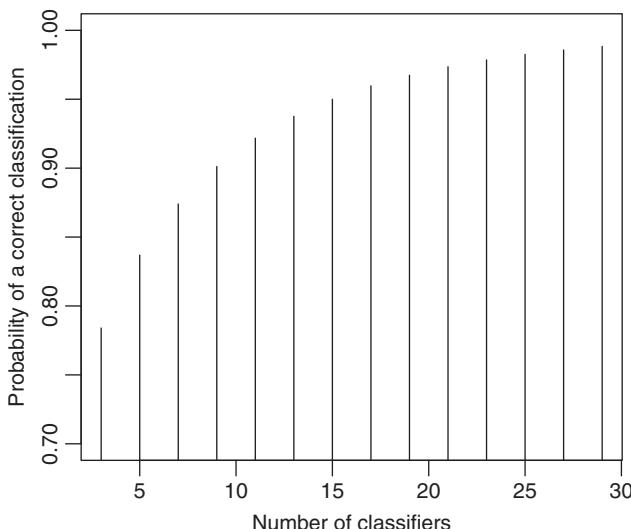


FIGURE 11.4 Correct Classification Probabilities with Varying Numbers of Independent Classifiers Each with a Probability of 0.7 of Correct Classification, and a Majority Decision is Made

```
p
[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

The probability of being wrong based on a majority decision is calculated and plotted with the following *R* code.

```
plot(p, 1-pbinom(10, 21, p),
      xlab = "Probability of an error in any one classifier",
      ylab = "Probability that more than 10 classifiers are in error",
      type = "h")
lines(p, p)
```

This generates Fig. 11.5.

The line in Fig. 11.5 indicates the probability p that an individual classifier will make an error. We see that, when $p < 0.5$, the majority probability is below this line, which means that the probability of making an incorrect classification is less than the error probability of an individual. When $p = 0.5$, the majority probability is the same as the individual probability, but when $p > 0.5$, the probability of making an error, based on a majority decision, is above the line, which means that the majority probability is greater than the error probability of an individual. Thus, the key to successful ensemble methods is to construct individual independent classifiers with error rates below 0.5.

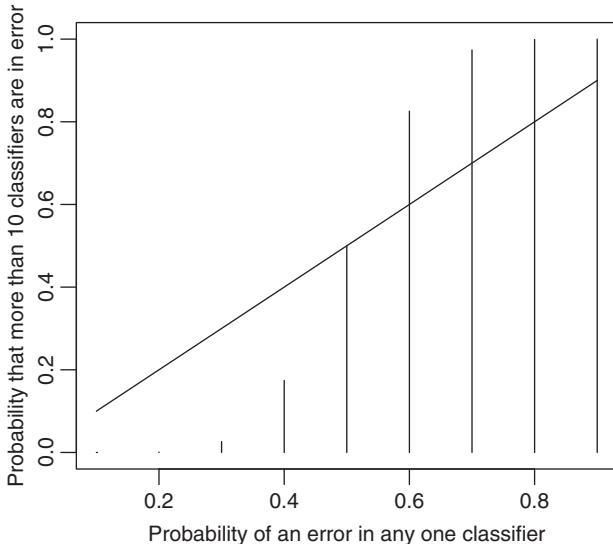


FIGURE 11.5 The Probability that More than 10 of 21 Independent Classifiers will Make an Error

11.7 BINOMIAL EXPECTATIONS

Looking again at the examples given at the beginning of the chapter we might ask:

In a laboratory with five workstations, how many workstations are operational on average, and how will this vary from day to day? (Example 11.1)

How many heads on average do we get when we toss a coin 10 times, and how will this vary from set to set? (Example 11.2)

How many defectives on average do we get in samples of size 20, and how will this vary from sample to sample? (Example 11.3)

How many bits in error would we expect in the 100 daily transmissions, and how will the number of errors vary from day to day? (Example 11.4)

This brings us to the mean and variance of the binomial distribution.

11.7.1 Mean of the Binomial Distribution

Recall that the mean of a discrete variable is defined as

$$\mu = E(X) = \sum_{x=0}^n xp(x) \quad (11.3)$$

In the binomial distribution, $p(x) = \binom{n}{x} p^x q^{n-x}$. Therefore,

$$\mu = E(X) = \sum_{x=0}^n x \binom{n}{x} p^x q^{n-x} \quad (11.4)$$

Example 11.12

From past experience, it is known that there is a 25% chance that a source program compiles successfully. Five such programs are written each day.

The number of programs that compile each day ranges from zero to five, and the probabilities are

$$P(X = x) = \binom{5}{x} (0.25)^x (0.75)^{5-x} \quad \text{for } x = 0, 1, 2, \dots, 5$$

These probabilities may be obtained in R.

```
x <- 0:5
dbinom(x, size = 5, prob = 0.25)
```

We round to three decimal places.

```
round(dbinom(x, 5, 0.25), 3)
[1] 0.237 0.396 0.264 0.088 0.015 0.001
```

TABLE 11.1 Binomial Probabilities with $n = 5$ and $p = 0.25$

Number that compiles, x	0	1	2	3	4	5
Probability, $p(x)$	0.237	0.396	0.264	0.088	0.015	0.001

So, we can summarize the binomial pdf with $n = 5$ and $p = 0.25$ as shown in Table 11.1.

To calculate the mean,

$$\begin{aligned} E(X) &= (0 \times 0.237) + (1 \times 0.396) + (2 \times 0.264) + (3 \times 0.088) \\ &\quad + (4 \times 0.015) + (5 \times 0.001) \\ &= 1.25 \end{aligned}$$

which means that the average number of programs that compile each day is 1.25. Some days, there might be zero, other days one, or two, or even three, but over a long number of days, there will be an average of 1.25 programs compiling per day. \triangleleft

While this direct approach to calculating the average is possible when n is small, it becomes tedious for large n . In what follows, we provide a simple and quick way of calculating the mean of the binomial distribution.

$$\begin{aligned} E(X) &= \sum_{x=0}^n x \binom{n}{x} p^x q^{n-x} \\ &= \sum_{x=1}^n x \binom{n}{x} p^x q^{n-x} \quad \text{since the first term is 0} \\ &= \sum_{x=1}^n n \binom{n-1}{x-1} p^x q^{n-x} \\ &= np \sum_{x-1=0}^{n-1} \binom{n-1}{x-1} p^{x-1} q^{(n-1)-(x-1)} \\ &= np(p+q)^{n-1} \\ &= np \quad \text{since } p+q=1 \end{aligned}$$

A binomial distribution with parameters n and p has a mean of np .

$$\mu = E(X) = np$$

Applying this to Example 11.12, the programs written in a day where $n = 5$ and $p = 0.25$, we may obtain the mean simply by calculating

$$E(X) = np = 5 \times 0.25 = 1.25$$

There is no need to go through the specific probability calculations as we did previously.

For Examples 11.1–11.4, we can calculate the means as follows.

Example 11.1: In the network system with five workstations, each with a probability of 0.95 of being operational, the average number of operational workstations is $\mu = 5 \times 0.95 = 4.75$.

Example 11.2: Tossing a coin 10 times, an average of $\mu = 10 \times 0.5 = 5$ heads would be expected.

Example 11.3: In the samples of 20 products from a production line with 20% defective, an average of $\mu = 20 \times 0.2 = 4$ defectives are expected per sample.

Example 11.4: In the digital transmission system, with 1% transmission error, there will be an average of $\mu = 100 \times 0.01 = 1$ bit in error per day's transmissions of 100.

11.7.2 Variance of the Binomial Distribution

Recall that the variance is defined as

$$\sigma^2 = E(X - \mu)^2 = \sum_x (x - \mu)^2 p(x)$$

For a binomial distribution with parameters n and p , this translates to

$$\sigma^2 = \sum_x (x - np)^2 \binom{n}{x} p^x q^{n-x} \quad (11.5)$$

To calculate the variance in Example 11.12, recall that the mean number of programs that compile is $\mu = 1.25$, and the compiling probabilities are given in Table 11.1. Putting these values into Equation 11.5, we have

$$\begin{aligned} \sigma^2 &= (0 - 1.25)^2(0.237) + (1 - 1.25)^2(0.396) + (2 - 1.25)^2(0.264) \\ &\quad + (3 - 1.25)^2(0.088) + (4 - 1.25)^2(0.015) + (5 - 1.25)^2(0.001) \\ &= 0.9375 \end{aligned}$$

The standard deviation

$$\sigma = \sqrt{0.9375} = 0.9683$$

Similar to the formula for the mean, mathematics provides us with a quick way of calculating the variance of a binomial distribution. By a derivation similar to that of $E(X) = np$, it can be shown that

$$\sigma^2 = \sum_x (x - np)^2 \binom{n}{x} p^x q^{n-x} = npq$$

The algebraic details of the binomial variance derivations are given in Appendix C.

The variances of Examples 11.1–11.4 may be calculated as follows:

Example 11.1: In the network system with $n = 5$ workstations attached, each with a probability of $p = 0.95$ of being operational,

$$\sigma^2 = 5 \times 0.95 \times 0.05 = 0.2375$$

Example 11.2: Tossing a coin $n = 10$ times,

$$\sigma^2 = 10 \times 0.5 \times 0.5 = 2.5$$

Example 11.3: In the samples of $n = 20$ products from a production line with $p = 20\%$ defective,

$$\sigma^2 = 20 \times 0.2 \times 0.8 = 3.2$$

Example 11.4: In the digital transmission system with $n = 100$ transmissions and $p = 0.01$ transmission error,

$$\sigma^2 = 100 \times 0.01 \times 0.99 = 0.99$$

To help us understand the meaning of σ^2 in the practical context, we carry out a simulation experiment in the next section.

11.8 SIMULATING BINOMIAL PROBABILITIES AND EXPECTATIONS

In this section, we show how to simulate a binomial distribution, and how to use the simulated results to obtain an estimate of the mean and variance. Let us look again at Example 11.3, where a manufacturing process produces integrated circuit chips with a 20% defective rate, and samples of size 20 are taken at regular intervals.

To simulate this, we use

`rbinom(k, n, p)`

which generates k values from a binomial distribution with parameters n and p .

For example, the function

```
defectives <- rbinom(50, 20, 0.2)
```

generates 50 random numbers from a binomial distribution with parameter $n = 20$ and $p = 0.2$. This can be taken as simulating the number of defectives of 50 samples of size 20 drawn from this production line of integrated chips in Example 11.3.

To see the output type

```
defectives
[1] 8 4 2 3 1 6 6 6 5 3 3 2 5 6 3 3 9 1 5 4 4 3 8 4 6 4
[27] 1 5 3 6 2 3 4 4 1 5 6 7 7 8 2 2 4 5 3 2 4 3 4 5
```

This output represents the number of defectives in each of the 50 samples; the first sample has eight defectives, the second sample has four defectives, and so on. To summarize, write

```
table(defectives)
```

which gives

```
defectives
 1 2 3 4 5 6 7 8 9
 4 6 10 10 7 7 2 3 1
```

The simulated probabilities or relative frequencies are obtained with

```
table(defectives)/length(defectives)
 1      2      3      4      5      6      7      8      9
 0.08  0.12  0.20  0.20  0.14  0.14  0.04  0.06  0.02
```

We can compare the simulated probabilities with the theoretical probabilities.

The following *R* code gives Fig. 11.6.

```
par(mfrow= c(1, 2))
plot(table(defectives)/length(defectives),
     xlab = "Number of defectives",
     ylab = "Relative frequency", type = "h")
x <- 1:9
plot(x, dbinom(x, 20, 0.2),
     xlab = "Number of defectives",
     ylab = "Probability" , type = "h")
```

Figure 11.6 shows that the simulated relative frequencies are close to the actual probabilities. Also note that the number of defectives per sample can be as high as nine, but more often between two and six.

11.8.1 Simulated Mean and Variance

The mean of the simulated number of defectives per sample is

```
mean(defectives)
```

giving

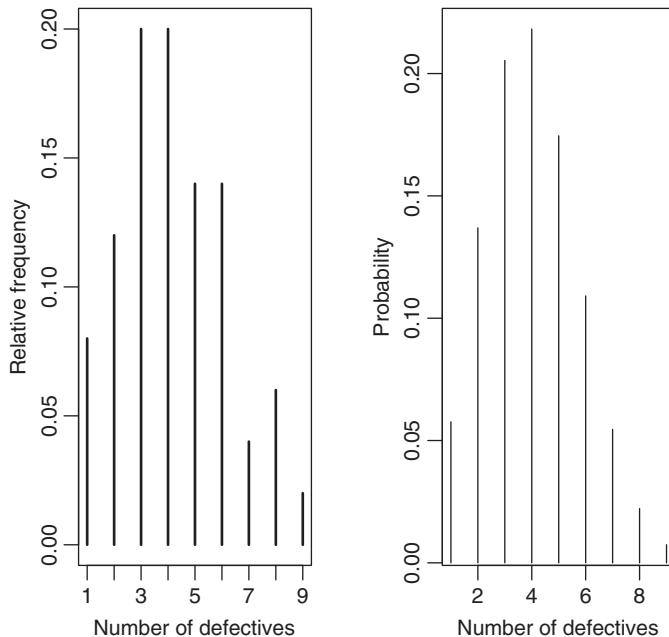


FIGURE 11.6 Simulated and Theoretical Binomial Distributions with $n = 20$ and $p = 0.2$

```
[1] 4.2
```

which is not too far away from the theoretical mean of $np = 20 \times 0.2 = 4$.

The simulated variance is obtained with

```
var(defectives)
```

which gives

```
[1] 3.959184
```

which is a little larger than the variance $20 \times 0.2 \times 0.8 = 3.2$. The reason for this is that we carried out just 50 simulations. For more accurate results, it would be necessary to increase the number of simulations, which as you know is a very easy thing to do in R. We keep it small here for illustration purposes.

The standard deviation may be obtained with

```
sd(defectives)
```

which gives

```
[1] 1.989770
```

which compares favorably with the true standard deviation.

```
sqrt(20*0.2*0.8)
[1] 1.788854
```

Statistical theory tells us that most of the data (more than 95%) lies within two standard deviations from the mean. In Example 11.3, with $n = 20$ and $p = 0.2$, we calculated the average number of defectives per sample to be $\mu = 4$ and the standard deviation $\sigma \approx 2$. So, the number of defectives in the samples should be in the interval 4 ± 4 , that is, between 0 and 8. Let us investigate with R.

The following R code gives a scatter diagram of the number of defectives in each of the 50 samples.

```
x <- 1:50 #sample number
plot(x, defectives,
      xlab = "Sample number"
      ylab = "Number of defectives")
abline(h = 4) #horizontal line for the mean
```

The output is given in Fig. 11.7.

Examining Fig. 11.7, we can say “with confidence” that the average number of defective chips per sample is four “give or take four.” Most samples should have

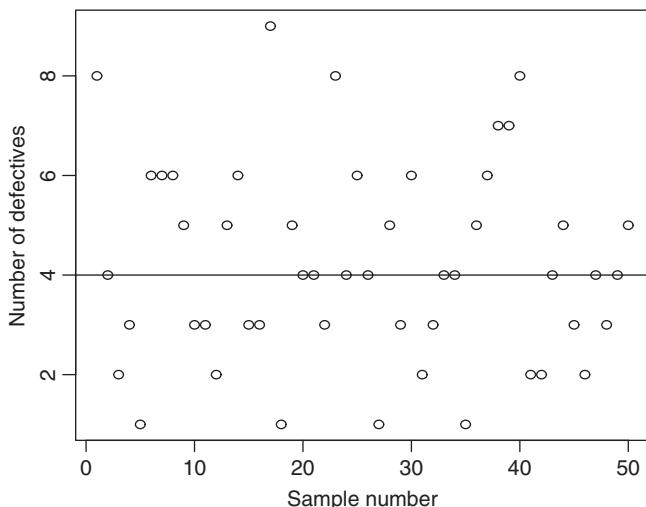


FIGURE 11.7 Number of Defectives per Sample of 20 with $p = 0.2$

less than eight defectives, while once in a while, there may be a sample containing more than eight defectives. Should it happen often, we would become suspicious about the claims regarding the quality of the production process.

EXERCISES 11.1

To answer the following exercises, use the `dbinom`, `pbinom`, and `qbinom` *R* functions appropriately.

1. As part of a sales strategy, 20% of randomly chosen new internet service subscribers receive a getaway break prize from the subscriber. If all 20 households in a particular housing estate subscribe, what is the probability that
 - (a) none of the households get the break;
 - (b) less than half of the households get the break;
 - (c) at least four of them get the break.
2. Approximately 40% of students are known to use Wikipedia when they are doing their projects. In a class of 10, what is the probability that more than half of them have used Wikipedia?
3. A central office in a company has eight computers operating independently. The probability that any one of them fails in a day is 0.20. Computers are repaired in the evening of each day and made operational for the next day. What is the probability that, in any one day,
 - (a) all eight will fail?
 - (b) there will always be a computer available?
4. Five users are reading the same webpage for hotel reservations. The probability that any user will click the “BUY” link is 0.95. What is the probability that
 - (a) none of them will buy?
 - (b) all of them will buy?
 - (c) more than half of them will buy?
5. It is estimated that 75% of adults use Twitter. If a sample of 20 adults is selected at random to answer a survey on Twitter, what is the probability that
 - (a) 75% of them use Twitter?
 - (b) more than half of them use Twitter;
 - (c) all of them use Twitter?
6. Five electronic components are wired to a controller that can switch from a failed component to one of the remaining spares. The probability that any component fails is 0.05. What is the probability that the controller will be unable to switch to a spare?

7. The probability of being able to log on to a computer system from a remote workstation during a busy working day is 0.8. Suppose that 10 independent attempts are made. What is the probability that
 - (a) less than five attempts will be successful?
 - (b) all attempts will be successful?
8. Consider a plant manufacturing chips of which 10% are expected to be defective. A sample of 10 chips is randomly selected, and the number of defectives X is observed. Write down the pdf and cdf of X . Hence find the probability of obtaining:
 - (a) exactly three defectives;
 - (b) at most three defectives;
 - (c) more than three defectives.
9. An Internet search engine is searching sites for a given keyword. This keyword occurs in about 10% of sites. Calculate the probability that:
 - (a) it will be found in at least three of the first 10 sites searched;
 - (b) it will not be found in any of the 10 sites.How many sites will need to be searched to be 90% sure of finding the keyword at least once?
10. An electronic product contains five integrated circuits. The probability that any one of the integrated circuits is defective is 0.05, and they are independent. The product operates only if at least two of the integrated circuits are nondefective.
 - (a) What is the probability that the product operates?
 - (b) How many integrated circuits should the product contain, in order to be 95% certain that the product will operate?
11. Five percent of memory sticks, produced by a hardware manufacturing company, are defective. If these are packed in boxes of $n = 20$, obtain:
 - (a) the proportion of boxes that have more than 5% defective;
 - (b) the average number of defectives per box.
12. A laboratory network consisting of 25 computers was attacked by a computer virus. The virus enters each computer with probability of 0.3 independently of other computers. Find the probability that more than half the computers are affected.
13. A folder contains 20 executable files. When a computer virus attacks the system, each file has a 25% chance of being affected. What is the probability that more than five of the executable files will be affected during a virus attack?
14. In a particular public school, it is known that 80% of students have access to the Internet at home. In a class of 20, what is the probability that all of them have access to the Internet?

15. In Example 11.9, suppose the system reliability requirement is 99%. You will recall that with the design given in this example, the system reliability was found to be 99.9. Investigate using R , if you can develop a more cost-effective design, by reducing the number of chips of each type while maintaining 99% reliability.
16. In Example 11.10, is it possible to use fewer components while maintaining a reliability of 99%?
17. In supervised learning, an ensemble of classifiers contains 17 independent classifiers whose individual decisions are combined to classify a new example. A classifier has a 0.8 chance of making a correct decision, and a 0.2 chance of being in error. To improve the accuracy, a majority vote is taken. What is the probability of an incorrect classification?

11.9 PROJECTS

1. Using your own computing experience, develop a front end to R , which provides the user with an opportunity to invoke the binomial distribution by clicking an icon on screen, to input the relevant values of the parameters n and p and to obtain plots of the pdf and cdf.
Be sure to provide validity checks that, for example, p is in the interval $(0, 1)$, n is a nonnegative integer, and $x \leq n$.
2. Use R to simulate a binomial distribution with, $n = 20$ and $p = 0.4$, a thousand times. Obtain the simulated pdf and satisfy yourself that it compares favorably to the theoretical pdf. Obtain the mean and variance of the simulated distribution, and compare them to the theoretical mean and variance.

12

THE HYPERGEOMETRIC DISTRIBUTION

Example 12.1

Five cards are chosen from a well-shuffled deck.

Let X be the number of diamonds selected.

$$P(X = 0) = P(0 \text{ diamonds in the selected 5})$$

$$P(X = 1) = P(1 \text{ diamond in the selected 5})$$

$$P(X = 2) = P(2 \text{ diamonds in the selected 5})$$

▫

Example 12.2

An audio amplifier consists of six transistors. It has been ascertained that three of the transistors are faulty, but it is not known which three. Amy removes three transistors at random and inspects them.

Let X be the number of defective transistors that Amy finds.

$$P(X = 0) = P(\text{the sample will catch none of the defective transistors})$$

$$P(X = 1) = P(\text{the sample will catch one of the defective transistors})$$

$$P(X = 2) = P(\text{the sample will catch two of the defective transistors})$$

$$P(X = 3) = P(\text{the sample will catch three of the defective transistors})$$

▫

Example 12.3

A batch of 20 integrated circuit chips contains 20% defectives. A sample of 10 is drawn at random.

Probability with R: An Introduction with Computer Science Applications, Second Edition. Jane M. Horgan.

© 2020 John Wiley & Sons, Inc. Published 2020 by John Wiley & Sons, Inc.

Companion website: www.wiley.com/go/Horgan/probabilitywithr2e

Let X denote the number of defective chips in the sample.

$$P(X = 0) = P(0 \text{ defectives in the sample of size } 10)$$

$$P(X = 1) = P(1 \text{ defective in the sample of size } 10)$$

$$P(X = 2) = P(2 \text{ defectives in the sample of size } 10)$$

△

Example 12.4

A batch of 100 printed circuit boards contains 30 that are defective. A sample of 20 is selected without replacement from the batch for function testing.

Let X denote the number of defective boards in the sample.

$$P(X = 0) = P(0 \text{ defectives in the sample of size } 20)$$

$$P(X = 1) = P(1 \text{ defective in the sample of size } 20)$$

$$P(X = 2) = P(2 \text{ defectives in the sample of size } 20)$$

△

In each of these examples, a selection of size n is made from a larger group of size N , in order to observe the number of occurrences of the event of interest.

As in previous chapters, we call the event of interest a “success.” The proportion of successes in the whole group is p .

$$N = 52, n = 5, p = 0.25 \text{ in Example 12.1}$$

$$N = 6, n = 3, p = 0.5 \text{ in Example 12.2}$$

$$N = 20, n = 10, p = 0.2 \text{ in Example 12.3}$$

$$N = 100, n = 20, p = 0.3 \text{ in Example 12.4}$$

This model is called the *hypergeometric distribution*. Its parameters are n the number chosen, N the total number, and p the proportion of successes in the group to start with. This gives $M = Np$, the number of successes in the group to start with.

Let us look at Example 12.1. The deck of cards can be visualized as

13	39
Diamonds	Non-diamonds

Five cards are selected at random and without replacement from the deck. The number of different sets of five cards to be chosen from the full deck of 52 is $\binom{52}{5}$.

$P(X = 0)$ is the probability of getting no diamonds in the sample of $n = 5$. This means that the five cards are selected from the other 39 non-diamonds. The number of ways of doing this is $\binom{39}{5}$, which is therefore the number of ways of choosing a set of cards favorable to the event of $X = 0$

Going back to our basic definition of probability in Chapter 4 where, for any event E , we defined

$$P(E) = \frac{\text{Number of occurrences favorable to } E}{\text{Total number of occurrences}}$$

So

$$P(X = 0) = \frac{\text{Number of favorable selections}}{\text{Total number of possible selections}} = \frac{\binom{39}{5}}{\binom{52}{5}}$$

For $P(X = 1)$, the number of favorable selections consists of the number of ways of choosing 1 diamond from the 13, which is $\binom{13}{1}$, and the number of ways of choosing 4 cards from the other 39 cards, which is $\binom{39}{4}$. So the total number of favorable selections is $\binom{13}{1} \binom{39}{4}$. Therefore,

$$P(X = 1) = \frac{\binom{13}{1} \binom{39}{4}}{\binom{52}{5}}$$

Similarly, $P(X = 2)$ means exactly two diamonds, which can be achieved by selecting 2 from the 13 diamonds, in $\binom{13}{2}$ ways, and exactly 3 from the other suits, in $\binom{39}{3}$ ways, giving a total of $\binom{13}{2} \binom{39}{3}$ favorable to the event of $X = 2$. So,

$$P(X = 2) = \frac{\binom{13}{2} \binom{39}{3}}{\binom{52}{5}}$$

For the remaining probabilities, we can write

$$\begin{aligned} P(X = 3) &= P(\text{exactly 3 of the 5 cards are diamonds}) = \binom{13}{3} \binom{39}{2} / \binom{52}{5} \\ P(X = 4) &= P(\text{exactly 4 of the 5 cards are diamonds}) = \binom{13}{4} \binom{39}{1} / \binom{52}{5} \\ P(X = 5) &= P(\text{all 5 cards are diamonds}) = \binom{13}{5} \binom{39}{0} / \binom{52}{5} \end{aligned}$$

12.1 HYPERGEOMETRIC RANDOM VARIABLES

Definition 12.1 *Hypergeometric random variable*

A hypergeometric random variable X is one which is represented as the number of successes in a sample of size n drawn without replacement from a population containing N elements. The proportion of successes in the population initially is p , so the number of successes in the population to start with can be calculated to be $M = Np$.

The population may be depicted as

$Np = M$ Successes	$N(1 - p) = N - M$ Non-successes
-----------------------	-------------------------------------

To calculate the probability of getting exactly x successes when drawing n elements without replacement from N , note that

- there are $\binom{N}{n}$ possible ways of choosing a sample of n without replacement from N .
- there are $\binom{M}{x}$ ways of choosing exactly x successes, and $\binom{N-M}{n-x}$ ways of choosing exactly $n - x$ non-successes, giving a total of $\binom{M}{x} \binom{N-M}{n-x}$ ways of getting exactly x successes in the sample of size n .

So, the probability density function is

$$P(X = x) = \frac{\binom{M}{x} \binom{N-M}{n-x}}{\binom{N}{n}} \quad x = 0, 1, \dots, n$$

□

In Example 12.2, we might ask,

What is the probability that Amy finds the three defective transistors when she selects three of the six?

$$P(X = 3) = \frac{\binom{3}{3}}{\binom{6}{3}} = \frac{3 \times 2 \times 1}{6 \times 5 \times 4} = \frac{1}{20} = 0.05$$

12.1.1 Calculating Hypergeometric Probabilities with R

In R, the function *hyper* with arguments M , $N - M$, and n will calculate the probabilities. Simply prefix the shortened name *hyper*, with “d” for the probability density function (pdf). The function *dhyper*, along with the parameters, will calculate the probabilities.

In Example 12.1, the probabilities of finding 0, 1, 2, 3, 4, 5 diamonds in the selected five cards may be obtained with

```
x <- 0:5
dhyper(x, 13, 39, 5)
```

gives

```
[1] 0.2215336134 0.4114195678 0.2742797119 0.0815426170 0.0107292917
[6] 0.0004951981
```

In Example 12.2, the probabilities of getting 0, 1, 2, 3 defective transistors in three selected from the six are

```
x <- 0:3
dhyper(x, 3, 3, 3)
[1] 0.05 0.45 0.45 0.05
```

In Example 12.3, the batch of 20 chips with 20% defective, there are 4 defective and 16 nondefective chips. In a sample of 10 selected without replacement from the batch, it is possible to get 0, 1, 2, 3, or 4 defectives. To calculate the probabilities of these occurring, write

```
x <- 0:4
dhyper(x, 4, 16, 10)
```

which gives

```
[1] 0.04334365 0.24767802 0.41795666 0.24767802 0.04334365
```

In Example 12.4, a sample of 20 is drawn without replacement from a batch of size 100 printed circuit boards with 30 defectives. To calculate the probabilities, use

```
x <- 0:10
dhyper(x, 30, 70, 20)
```

12.1.2 Plotting the Hypergeometric Function

To examine the pattern of the hypergeometric probabilities, we plot the pdfs of Examples 12.1–12.4 using *R*.

```
par(mfrow = c(2,2))
x <- 0:5 #Example 12.1
plot(x, dhyper(x, 13, 39, 5),
      xlab = "Number of diamonds", type = "h",
      ylab = "P(X = x)",
      main = "N = 52, M = 13, n = 5", font.main = 1)

x <- 0:3 #Example 12.2.
plot(x, dhyper(x, 3, 3, 3),
      xlab = "Number of defective transistors", type = "h",
      ylab = "P(X = x)",
      main = "N = 6, M = 3, n = 3", font.main = 1)
```

```

x <- 0:4 #Example 12.3
plot(x, dhyper(x, 4, 16, 10),
      xlab = "Number of defective IC chips", type = "h",
      ylab = "P(X = x)",
      main = "N = 20, M = 4, n = 10", font.main = 1)

x <- 0:20 #Example 12.4
plot(x, dhyper(x, 30, 70, 20),
      xlab = "Number of defective PC boards", type = "h",
      ylab = "P(X = x)",
      main = "N = 100, M = 30, n = 20", font.main = 1)

```

This generates Fig. 12.1.

Note the bell-shaped nature of the last plot, with $N = 100$ and $n = 20$.

12.2 CUMULATIVE DISTRIBUTION FUNCTION

The cumulative distribution function (cdf) of the hypergeometric distribution, $P(X \leq x)$, represents the probability that the selected items will contain up to, and including, x successes. In Example 12.1, we might want to know the probability

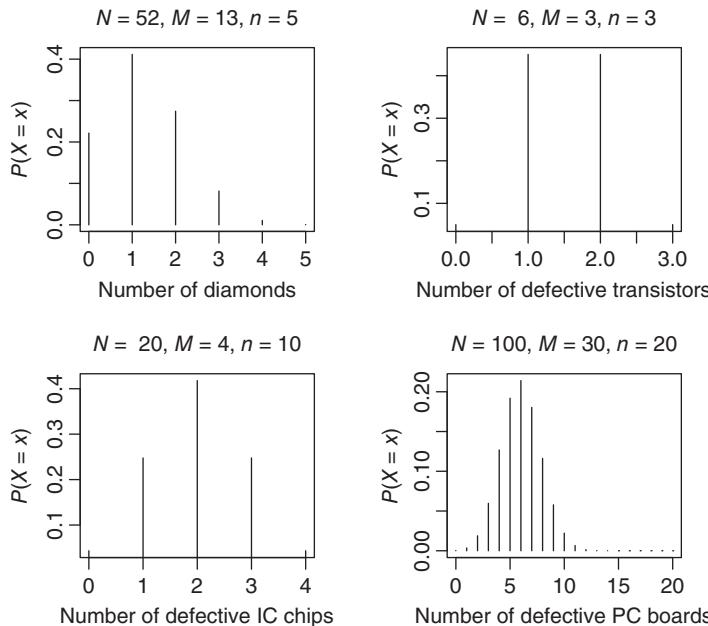


FIGURE 12.1 Hypergeometric pdfs

of getting at most two diamonds in the five selected without replacement from a well-shuffled deck.

$$\begin{aligned}
 P(X \leq 2) &= P(X = 0) + P(X = 1) + P(X = 2) \\
 &= \frac{\binom{39}{5}}{\binom{52}{5}} + \frac{\binom{13}{1}\binom{39}{4}}{\binom{52}{5}} + \frac{\binom{13}{2}\binom{39}{3}}{\binom{52}{5}} \\
 &= 0.222 + 0.411 + 0.274 = 0.907
 \end{aligned}$$

This can be calculated in *R* with

```
phyper(2, 13, 39, 5)
[1] 0.9072329
```

To plot the cdfs of Examples 12.1–12.4, write

```

par(mfrow = c(2,2))
x <- 0:5 #Example 12.1
plot(x, phyper(x, 13, 39, 5),
      xlab = "Number of diamonds", type = "s",
      ylab = "P(X <= x)",
      main = "N = 52, M = 13, n = 5", font.main =1)

x <- 0:3 #Example 12.2
plot(x, phyper(x, 3, 3, 3),
      xlab = "Number of defective transistors", type = "s",
      ylab = "P(X <= x)",
      main = "N = 6, M = 3, n = 3", font.main =1)

x <- 0:10 #Example 12.3
plot(x, phyper(x, 4, 16, 10),
      xlab = "Number of defective IC chips", type = "s",
      ylab = "P(X <= x)",
      main = "N = 20, M = 4, n = 10", font.main =1)

x <- 0:20 #Example 12.4
plot(x, phyper(x, 30, 70, 20),
      xlab = "Number of defective PC boards", type = "s",
      ylab = "P(X <= x)",
      main = "N = 100, M = 30, n = 20", font.main =1)
```

This generates Fig. 12.2.

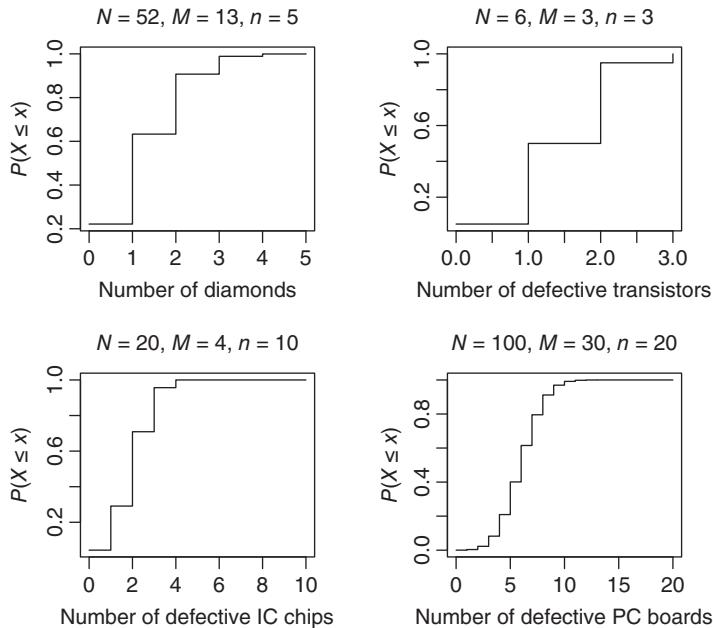


FIGURE 12.2 Hypergeometric cdfs

12.3 THE LOTTERY

“The National Lottery” is a scheme by which the government raises money for selected charities by dangling enormous amounts of money in front of the populace. The format is such that to hit the jackpot, you need to correctly chose n different numbers from N .

This is a classic example of the hypergeometric distribution. In the draw, n random numbers are selected from the N , and your winnings depend on how many of these match your selections.

Let X be the number of drawn numbers that match your selection.

$P(X = 0)$ is the probability that none match your selection

$P(X = 1)$ is the probability that 1 matches your selection

$P(X = 2)$ is the probability that 2 match your selection

and so on.

Usually $n = 6$, and, in Ireland, N used to be 36, but over the years, it has increased to 47 which, as we shall see, makes it even more difficult to win. Let us first consider the situation when $N = 36$.

The population consists of

6 Favorable Your chosen 6 numbers	30 Non-favorable The other 30 numbers
--------------------------------------	--

$$P(X = x) = \frac{\binom{6}{x} \binom{30}{n-x}}{\binom{36}{6}}, \quad x = 0, 1, 2, 3, 4, 5, 6$$

We can calculate this directly from R using

```
x <- 0:6
round(dhyper(x, 6, 30, 6), 7)
```

which gives

```
[1] 0.3048452 0.4389771 0.2110467 0.0416882 0.0033499
[6] 0.0000924 0.0000005
```

Here, we note the high probability of matching none is 30%, of matching one is 44%, and of matching two is 21%. These matches (0, 1, and 2) do not command a prize in the Irish lottery. The probabilities of matching more than two decreases very quickly to, 4% for three matches, 0.3% for four matches, 0.009% for five matches, and a minuscule 0.00005% for the jackpot.

There is only one way of winning the jackpot; that is, by matching all six out of a total of $\binom{36}{6} = 1,947,792$. And yet people believe they are going to win.

To understand how unlikely it is to win the jackpot, suppose you buy one set of numbers each week without fail. Then the length of time you would expect to have to wait to win the jackpot would be $1,947,792/52 \approx 37,458$ years.

To calculate this

```
x <- choose(36, 6)
[1] 1947792
x/52
[1] 37457.54
```

If you bought one ticket a week, you would expect to wait over 37,000 years to win the jackpot.

If you increased to two tickets a week, you would still have to wait

```
x/(2*52)
[1] 18728.77
```

which means that, even if you bought 104 tickets a year, the expected waiting time would be over 18,728 years.

Increasing your tickets to 100 per week means that you would expect to wait

```
x / (100 * 52)
[1] 374.5754
```

more than 374 years on average for a win.

These calculations also assume that you are the sole winner. If other punters should happen to choose the same numbers, then your winnings are divided among the whole group.

The above expected waiting times and the chances of winning were calculated using 36 numbers. In Ireland, as indeed in other countries, the amount of numbers have increased steadily over the years. In 1992, the numbers increased to 39, in 1994, they went to 42, in 2006, they went to 45, and in 2015, they went to 47. Let us look at the effect of these changes on the chances of winning.

The pdfs given in Fig. 12.3 were obtained from *R* using `dhyper(x, 6, 33, 6)`, `dhyper(x, 6, 36, 6)`, `dhyper(x, 6, 39, 6)`, and `dhyper(x, 6, 41, 6)` which represent the matching probabilities respectively when $N = 39$, 42, 45, and 47.

At first sight, Fig. 12.3 suggests that not much has changed as the numbers increase. The chance of winning the jackpot was always minuscule. Notice though, that the chance of matching zero increases as the numbers increase.

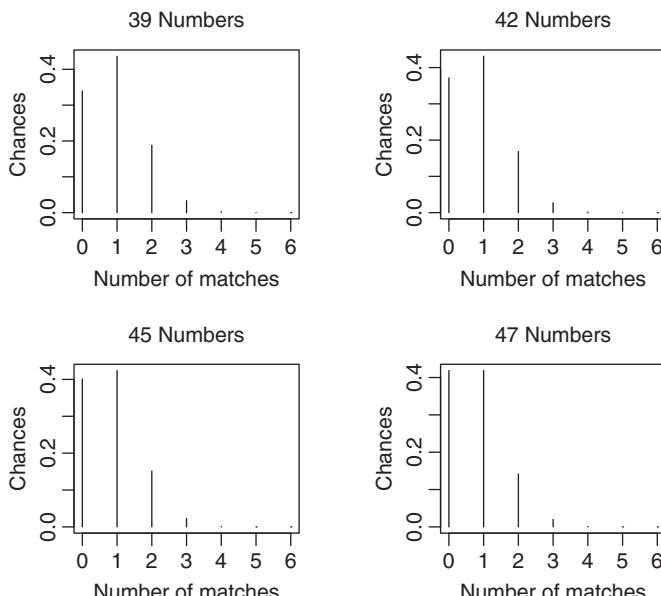


FIGURE 12.3 The Lottery with Differing Total Number Sets

12.3.1 Winning Something

It might be more informative to examine the probability of winning some sort of consolation prize. In the Irish Lottery, any match less than or equal to 2 gets nothing. Let us therefore investigate $P(X \leq 2)$ for each set, the probability of winning nothing at all on your bet.

$$P(X \leq 2) = P(\text{match 0}) + P(\text{match 1}) + P(\text{match 2})$$

Alternatively, we might want to know what is the probability of winning something, no matter how small.

$$P(X \geq 3) = 1 - P(X \leq 2)$$

We can calculate this using the cdf of the hypergeometric distribution.

With 36 numbers,

```
1 - phyper(2, 6, 30, 6)
[1] 0.0451311
```

there is a 4.5% chance of getting some return from your bet with 36 numbers.

With 39 numbers,

```
1 - phyper(2, 6, 33, 6)
[1] 0.03593397
```

the chance of some return reduces to 3.6%.

With 42 numbers,

```
1 - phyper(2, 6, 36, 6)
[1] 0.02906466
```

there is less than a 3% chance of winning something.

With 45 numbers,

```
1 - phyper(2, 6, 39, 6)
[1] 0.02383408
```

the chances of winning something has come down to 2.4%.

With 47 numbers,

```
1 - phyper(2, 6, 41, 6)
[1] 0.02102402
```

the chance of winning something is just 2.1%.

The increase from 36 to 47 numbers has had the effect of more than halving the chance of getting any return whatsoever on your bet.

12.3.2 Winning the Jackpot

As for winning the jackpot,

With 36 numbers, choose $(36, 6)$ gives 1,947,792 – a chance of about 1 in 2 million.

With 39 numbers, choose $(39, 6)$ gives 3,262,623 – a chance of less than 1 in 3 million.

With 42 numbers, choose $(42, 6)$ gives 5,245,786 – a chance of less than 1 in 5 million.

With 45 numbers, choose $(45, 6)$ gives 8,145,060 – a chance of less than 1 in 8 million.

And now the current situation of 47 numbers choose $(47, 6)$ gives 10,737,573 – a chance of nearly as low as 1 in 11 million!

Punters dream on.

12.4 HYPERGEOMETRIC OR BINOMIAL?

Example 12.5

Suppose 10% of items coming off a production line are defective. They are packed in boxes of 20. To ensure a level of quality among the delivered products, a sample of size 10 is selected at random from each box, and the number, X , of defective items occurring in the sample is ascertained.

Suppose the sampling is done with replacement, that is, when an item is selected and examined, it is returned to the box. In this case:

1. Each selection has two possible outcomes;
 - A defective with probability $p = 0.1$
 - A nondefective with probability $q = 0.9$
2. Repeated selections are independent, since the items are replaced, and so the probability of selecting a defective item is independent of whether or not a defective item was selected previously; the probability of a defective item remains at p .

If you look back at the definition of the binomial distribution, given in the previous chapter, you will see that X is a binomial random variable, and the probabilities can be calculated as follows.

$$P(X = 0) = (0.9)^{10} = 0.3487$$

$$P(X = 1) = \binom{10}{1} (0.1)^1 (0.9)^9 = 0.3874$$

$$P(X = 2) = \binom{10}{2} (0.1)^2 (0.9)^8 = 0.1937$$

In practice, sampling is never done “with replacement”. After inspection, the inspected item is not returned to the batch.

Instead, we have a hypergeometric distribution. Let us calculate the probabilities.

$$P(X = 0) = \frac{\binom{18}{10}}{\binom{20}{10}} = 0.2368$$

$$P(X = 1) = \frac{\binom{2}{1} \binom{18}{9}}{\binom{20}{10}} = 0.5263$$

$$P(X = 2) = \frac{\binom{2}{2} \binom{18}{8}}{\binom{20}{10}} = 0.2368$$

Clearly, the probabilities obtained using the binomial pdf differ considerably from those obtained using the hypergeometric pdf. \triangleleft

Let us now look at a larger batch containing $N = 200$ items, again with 10% defective, from which a sample of size $n = 10$ is to be taken.

Calculating the probabilities with the hypergeometric

$$P(X = 0) = \frac{\binom{180}{10}}{\binom{200}{10}} = 0.3398$$

$$P(X = 1) = \frac{\binom{20}{1} \binom{180}{9}}{\binom{200}{10}} = 0.3974$$

$$P(X = 2) = \frac{\binom{20}{2} \binom{180}{8}}{\binom{200}{10}} = 0.1975$$

TABLE 12.1 Hypergeometric and Binomial Probabilities with $n = 10$ and $p = 0.1$

Batch Size	20	200	2000	Bin. Approx
$P(X = 0)$	0.2368	0.3398	0.3478	0.3487
$P(X = 1)$	0.5263	0.3974	0.3884	0.3874
$P(X = 2)$	0.2368	0.1975	0.1940	0.1937
$P(X \leq 2)$	0.999	0.9347	0.9302	0.9298

Notice that these probabilities appear to be getting nearer to the probabilities that we calculated using the binomial distribution.

Suppose we take an even larger batch of size $N = 2,000$ items of which 10% are defective.

Then

$$P(X = 0) = \frac{\binom{1800}{10}}{\binom{2000}{10}} = 0.3478$$

$$P(X = 1) = \frac{\binom{200}{1} \binom{1800}{9}}{\binom{2000}{10}} = 0.3884$$

$$P(X = 2) = \frac{\binom{200}{2} \binom{1800}{8}}{\binom{2000}{10}} = 0.1941$$

Now, these are very close to the binomial probabilities obtained previously.

$$P(X = 0) = 0.3487, \quad P(X = 1) = 0.3874, \quad \text{and} \quad P(X = 2) = 0.1937$$

What we are attempting to illustrate is that, when the batch size N is large, the hypergeometric distribution may be approximated with the binomial. The results for $N = 20, 200, 2000$ are summarized in Table 12.1.

From Table 12.1, it is clear that the hypergeometric probabilities approach the binomial as the batch size N increases.

We can illustrate further using plots in *R*.

```
x <- 0:4
N <- 20
M <- 0.1 * N
L <- 0.9 * N
y <- dhyper(x, M, L, 10)
yround <- round(y, 3)
plot(x, y, type = "h", ylim = c(0, .51), xlab = " ",
     main = "Hypr(N = 20)", axes = FALSE)
text(x, dhyper(x, M, L, 10), yround)
axis(1)
```

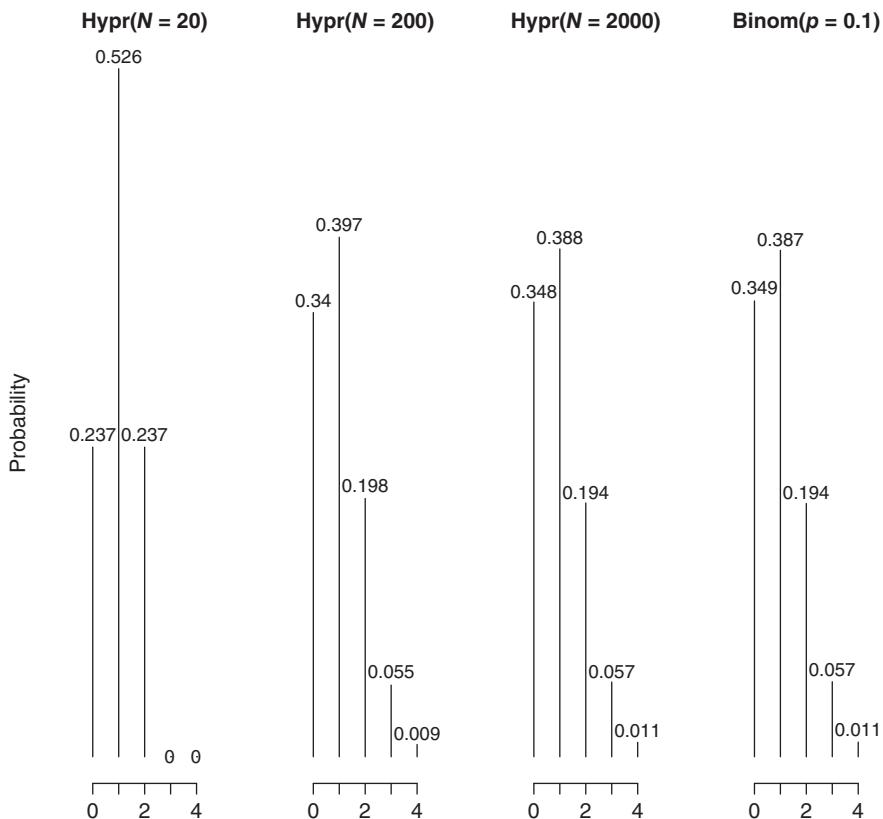


FIGURE 12.4 Hypergeometric and Binomial pdfs with $n = 10, p = 0.1$

This code gives the first plot in Fig. 12.4. The command `axes = FALSE` suppresses all axes, including the boundary axes, `axis(1)` restores the x -axis. In the `text` command, `labels` plots the point `(x, dhyper(x, M, L, 10))`.

The first three hypergeometric graphs in Fig. 12.4 can be generated using a `for` loop in R.

```
par(mfrow = c(1, 4))
x <- 0:4
size <- c(20, 200, 2000)
for (N in size) {
  M <- 0.1 * N
  L <- 0.9 * N
  y <- dhyper(x, M, L, 10)
  yround <- round(y, 3)
  plot(x, y, type = "h", ylim = c(0, 0.51), xlab = " ",
        main = bquote(Hyper(N == .(N))), axes = FALSE)
  text(x, dhyper(x, M, L, 10), yround)
  axis(1)
}
```

Here, `bquote(Hyper(N == .(N)))` writes the argument except the expression in `.()` which is evaluated.

The binomial plot is added to Fig. 12.4 with

```
y <- dbinom(x, 10, 0.1)
yround <- round(y, 3)
plot(x, y, type = "h", ylim = c(0, 0.51),
      xlab = " ", ylab = " ",
      main = "Binom(p = 0.1)", axes = FALSE)
text(x, dbinom(x, 10, 0.1), yround)
axis(1)
```

It is clear from Fig. 12.4 that there is very little difference between the binomial probabilities and the hypergeometric when N is large.

We could examine the differences in more detail using `matplot` and `segments`.

```
x <- 0:10
N <- 20
M <- 0.1*N
L <- 0.9*N
matplot(x, cbind(dbinom(x, 10, 0.1), dhyper(x, M, L, 10)),
         xlab = "Number of occurrences", ylab = "Probability",
         main = "Hyper (N = 20)", pch = 25, font.main = 1)
segments(x, dbinom(x, 10, 0.1), x, dhyper(x, M, L, 10))
```

creates the first diagram in Fig. 12.5. Note the use of:

- `cbind` which combines R objects by columns. In this case, it combines `dbinom(x, 10, 0.01)` and `dhyper(x, M, L, 10)`;
- `matplot` which plots columns of matrices. In this case, it plots `(x, xdbinom(x, 10, 0.01))` and `(x, dhyper(x, M, L, 10))` for each x ;
- `pch` which plots the points with triangles;
- `segments` which joins the points

$$(x, dbinom(x, 10, 0.1)) \text{ and } (x, dhyper(x, M, L, 10)).$$

We could generate the six plots in Fig. 12.5 using the `for` command.

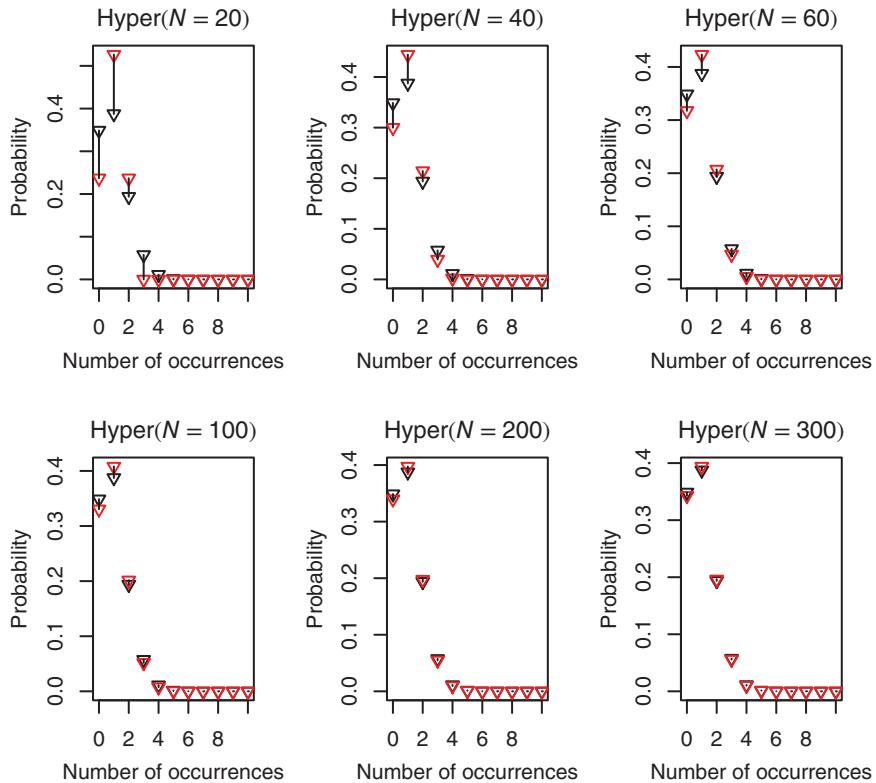


FIGURE 12.5 Hypergeometric and Binomial pdfs with $n = 10, p = 0.1$

```

par(mfrow = c(2,3))
x <- 0:10
size <- c(20, 40, 60, 100, 200, 300)
for (N in size) {
  M <- 0.1*N
  L <- 0.9*N
  matplot(x, cbind(dbinom(x, 10, 0.1), dhyper(x, M, L, 10)),
    xlab = "Number of occurrences", ylab = "Probability",
    main = bquote(Hyper(N == .(N))), pch= 25, font.main = 1)
  segments(x, dbinom(x, 10, 0.1), x, dhyper(x, M, L, 10))
}

```

Figure 12.5 illustrates that, when the batch size N is large, there is very little difference between the probabilities obtained using the hypergeometric distribution, and those obtained with the binomial probabilities. The “sampling-without-replacement” hypergeometric model is equivalent to the “sampling-with-replacement” binomial model.

What this means is that, when a sample of size n is chosen from a batch of size N without replacement, if N is large relative to n , so that the sampling fraction n/N is negligible, then sampling without replacement is practically the same as sampling with replacement.

It is shown in Appendix D that

$$\begin{aligned}
 P(X = x) &= \frac{\binom{Np}{x} \binom{N(1-p)}{n-x}}{\binom{N}{x}} \\
 &\rightarrow \binom{n}{x} p^x q^{n-x} \text{ as } N \rightarrow \infty.
 \end{aligned}$$

This approximation is valid when n is small relative to N so that $n/N \approx 0$.

EXERCISES 12.1

- Consider a plant manufacturing IC chips of which 10% are expected to be defective. The chips are packed in boxes for export. Before transportation, a sample is drawn from each box. Estimate the probability that the sample contains more than 10% defectives, when:
 - A sample of 10 is selected from a box of 40;
 - A sample of 10 is selected from a box of 5000.
- On a circuit board, there are 50 chips of which 5 are defective. If 15 chips are selected at random, what is the probability that at most 2 are defective?

3. The Irish lottery has 47 numbers from which you can choose 6. Ava buys one lotto ticket and selects six numbers at random from the 47 possibilities. She will get some sort of monetary prize, if she matches three or more. What is the probability that she will win a prize?
4. In a shipment of 200 Pentium processors, there are 10 defectives. A quality control inspector selects a sample of five processors at random for inspection.
 - (a) Obtain the probability that the sample will contain 0, 1, 2, 3, 4, or 5 defectives
 - (b) The batch is rejected if it contains one or more defectives. What is this probability?

12.5 PROJECTS

1. We have illustrated that the hypergeometric distribution can be approximated by the binomial distribution.

$$\frac{\binom{Np}{x} \binom{N-Np}{n-x}}{\binom{N}{n}} \rightarrow \binom{n}{x} p^x (1-p)^{n-x} \quad x = 0, 1, 2, \dots, n .$$

as $N \rightarrow \infty$ so that $n/N \approx 0$.

Use R to derive a rule of thumb as to when best to move from the hypergeometric distribution to the binomial distribution and to decide for what values of n and N is this approximation is valid?

Hint: Assume $p = 0.1$ to start. Then with $N = 100$ and $n = 10$, calculate the probabilities with both pdfs. If the approximation is inaccurate, increase N ($N = 200$, say) and keep going until you have a good approximation. Repeat the process for $n = 20$.

Then, you might take $p = 0.2$ and do it all over again.

The objective is to obtain a rule of thumb, based perhaps on the size of n relative to N , for deciding when to use the binomial as an approximation for the hypergeometric. In other words, when sampling without replacement may be treated as sampling with replacement.

2. When plotting Fig. 12.5 we used the plotting character `pch` = 25 to create the triangles. R uses a range of point symbols. Experiment yourself by changing the number in `pch`.

13

THE POISSON DISTRIBUTION

In Chapter 11, we considered the binomial random variable for assessing the probability of X occurrences in a sample of size n .

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x} \quad (13.1)$$

where p is the probability of a success.

We look at this again now, and consider the binomial distribution when p is small and n is large. The probability of the occurrence of the event is so low that it is called a “rare event.”

13.1 DEATH BY HORSE KICK

The rare event phenomenon was first investigated by Siméon Denis Poisson in 1837 who applied it to the verdicts of juries, but it was to be another 60 years before it began to come to public attention (Poisson, 1837). In 1898, von Bortkiewicz examined the chance of a Prussian cavalryman being killed by a kick of a horse (von Bortkiewicz, 1898). He examined the number of recorded fatalities by horse kicks sustained in 10 army corps of the Prussian cavalry over 20 years, giving a total of 200 corps recorded. The data are summarized in Table 13.1.

TABLE 13.1 Horse Kick Fatalities in the Prussian Cavalry

Number of Deaths (X)	Frequency (f)
0	109
1	65
2	22
3	3
4	1
>4	0
Total	200

From the data given in Table 13.1, we can estimate the average occurrence of a fatal horse kick.

$$\lambda = \frac{(0 * 109) + (1 * 65) + (2 * 22) + (3 * 3) + (4 * 1)}{200} = 0.61$$

von Bortkiewicz reasoned that it is meaningless to ask how many times per year a cavalry man was not kicked to death by a horse. Maybe it could be said that there are an infinite number of moments when people were not kicked by horses. He reasoned that the probability of the occurrence of the event of this kind is minutely small, while the number of times it did not occur is infinitely large, and deduced that deaths by horse kicks might be modeled by the binomial distribution as $n \rightarrow \infty$ and $p \rightarrow 0$.

13.2 LIMITING BINOMIAL DISTRIBUTION

Poisson and von Bortkiewicz looked at

$$P(X = x) = \lim \binom{n}{x} p^x (1-p)^{n-x} \text{ as } (n, p) \rightarrow (\infty, 0) \text{ with } np \text{ constant.}$$

The limiting distribution is obtained as follows: with $np = \lambda$ so that $p = \lambda/n$

$$\begin{aligned} P(X = x) &= \binom{n}{x} p^x (1-p)^{n-x} \\ &= \binom{n}{x} \left(\frac{\lambda}{n}\right)^x \left(1 - \frac{\lambda}{n}\right)^{n-x} \\ &= \binom{n}{x} \frac{\lambda^x}{n^x} \left(\frac{1 - \frac{\lambda}{n}}{1 - \frac{\lambda}{n}}\right)^x \end{aligned}$$

Now,

$$\left(1 - \frac{\lambda}{n}\right)^n \rightarrow e^{-\lambda} \text{ as } n \rightarrow \infty$$

$$\left(1 - \frac{\lambda}{n}\right)^x \rightarrow 1 \text{ as } n \rightarrow \infty$$

$$\begin{aligned} \binom{n}{x} \frac{1}{n^x} &= \frac{1}{x!} \frac{n(n-1) \dots (n-(x-1))}{n^x} \\ &= \frac{1}{x!} 1 \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \dots \left(1 - \frac{x-1}{n}\right) \rightarrow \frac{1}{x!} \quad \text{as } n \rightarrow \infty. \end{aligned}$$

Thus,

$$P(X = x) \rightarrow \frac{\lambda^x}{x!} e^{-\lambda} \text{ as } n \rightarrow \infty \text{ and } p \rightarrow 0 \text{ with } np = \lambda \quad (13.2)$$

This limit defines what is now known as the Poisson distribution.

Let us use *R* to calculate the probabilities of being kicked to death by a horse with the binomial density, given in Equation (13.1), and with the Poisson density given in Equation (13.2), and then compare these with the actual data given in Table 13.1.

Using the binomial distribution with $p = \lambda/200$

```
lambda <- 0.61
x <- 0:5
prob <- dbinom(x, 200, lambda/200)
[1] 0.5428446228 0.3321482721 0.1011070735 0.0204151200 0.0030759880
[6] 0.0003688902
```

gives the estimated binomial probabilities of the occurrences of 0, 1, 2, 3, 4, 5 horse kicks.

To see what to expect with 200, write

```
round(200*dbinom(x, 200, lambda/200), 2)
[1] 108.57 66.43 20.22 4.08 0.62 0.07
```

which gives the number of horse kicks expected with the binomial model (rounded to two decimal places).

Performing a similar analysis with the Poisson densities, use the function *dpois* in *R*. The Poisson probabilities are calculated by prefixing the shortened name *pois* with “d” and attaching λ as an argument.

```
dpois(x, lambda)
[1] 0.5433508691 0.3314440301 0.1010904292 0.0205550539 0.0031346457
[6] 0.0003824268
```

gives the estimated Poisson probabilities of the occurrences of 0, 1, 2, 3, 4, 5 horse kicks.

```
round(200*dpois(x, lambda), 2)
[1] 108.67 66.29 20.22 4.11 0.63 0.08
```

gives the number of horse kicks expected with the limiting Poisson model (rounded to two decimal places).

TABLE 13.2 Estimating Fatal Horse Kicks

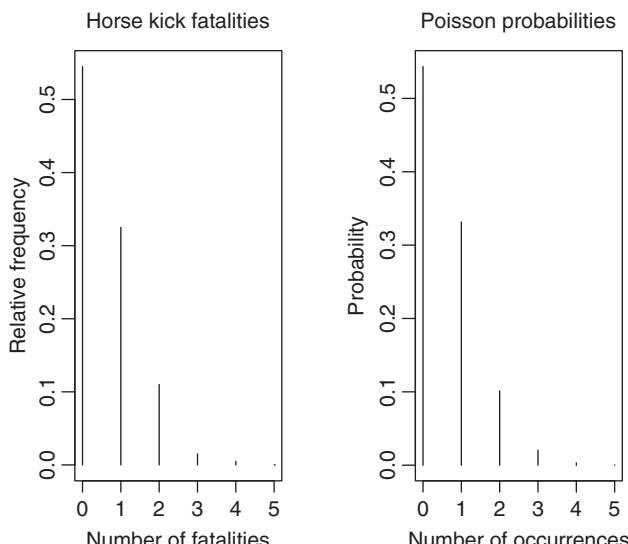
Fatal Horse Kicks	Actual	Binomial ($n = 200, p = \lambda/200$)	Poisson ($\lambda = 0.61$)
0	109	108.67	108.57
1	65	66.43	66.29
2	22	20.22	20.22
3	3	4.08	4.11
4	1	0.62	0.63
>4	0	0.07	0.08

Table 13.2 gives the actual incidences of horse kicks, along with those that would occur under the assumption of a binomial density and a Poisson density.

We make two important observations from Table 13.2.

1. The frequencies obtained with the binomial and Poisson are practically identical, confirming the Poisson distribution as a limiting binomial distribution when $n \rightarrow \infty$ and $p \rightarrow 0$ so that np is constant.
2. The actual frequencies are quite close to those estimated with the binomial and Poisson models.

In Fig. 13.1 the data are plotted, along with a Poisson distribution, using the following *R* code.

**FIGURE 13.1** Horse Kick Fatalities in the Prussian Army

```

relfreq <- c(109/200, 65/200, 22/200, 3/200, 1/200, 0)
x<- 0:5
par(mfrow = c(1, 2))
plot(x, relfreq, type = "h",
      xlab = "Number of fatalities",
      ylab = "Relative frequency",
      main = "Horse kick fatalities",
      font.main = 1)
plot(x, dpois(x, 0.61), type = "h",
      xlab = "Number of occurrences",
      ylab = "Probability",
      main = "Poisson probabilities",
      font.main = 1)

```

Example 13.1

Let us now examine bits being transmitted over a digital communication channel.

- (i) Consider the transmission of 10 bits over a digital communication channel. Let the random variable X be the number of bits in error. Suppose the probability that a bit is in error is 0.2 and the transmissions are independent. Then X has a binomial distribution.

In R

```

x <- 0:10
n <- 10
prob <- 0.2
plot (x, dbinom(x, n, prob), xlab= "X",
      ylab = "P(X = x)", type = "h")

```

will plot the the probabilities as given in Fig. 13.2.

- (ii) Consider the transmission of 50 bits over a digital communication channel. Let the random variable X be the number of bits in error. Suppose the probability that a bit is in error is 0.04 and the transmissions are independent. This time $n = 50$, and $p = 0.04$ (Fig. 13.3).
- (iii) Consider the transmission of 100 bits over a digital communication channel. Let the random variable X be the number of bits in error. Suppose the probability that a bit is in error is 0.02 and the transmissions are independent. Now $n = 100$ and $p = 0.02$ (Fig. 13.4).

What we are doing in Example 13.1 is examining the number of bits in error as n gets bigger and p gets smaller in such a way that np remains constant. These assumptions allow us to concentrate on events that are rare (Fig. 13.5).

We summarize the first few probabilities for the different binomials with the Poisson limit in Table 13.3.

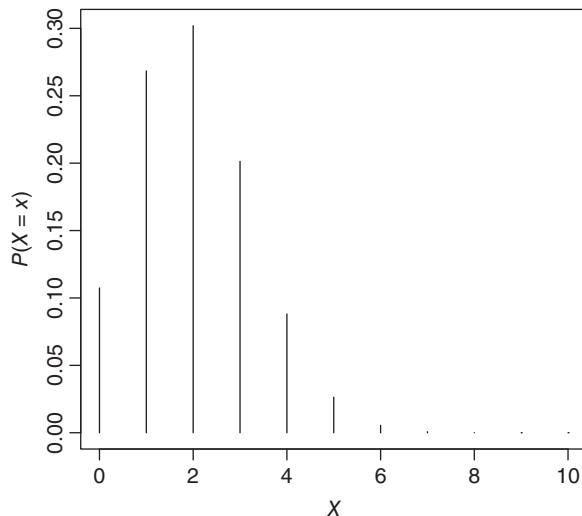


FIGURE 13.2 Binomial pdf $n = 10, p = 0.2$

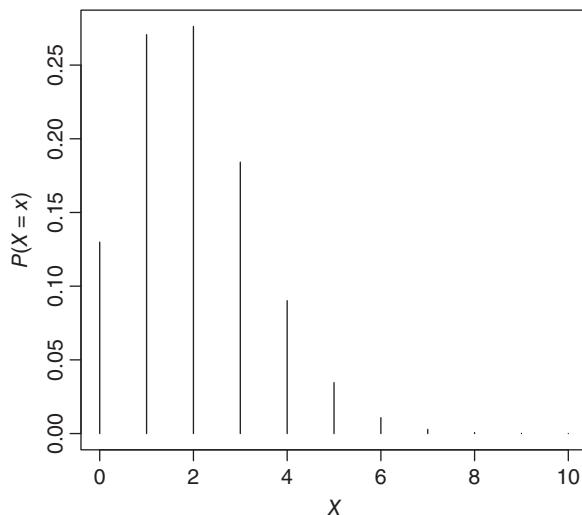


FIGURE 13.3 Binomial pdf $n = 50, p = 0.04$

It is clear from the Table 13.3 that the Poisson probabilities are accurate approximations of those of the binomial when the number of trials, n , is large, and the probability, p , of a success in any one trial is small. \triangleleft

Example 13.2

An Internet provider receives an average of 50 new customers a day in a new housing development. The number of potential Internet users in this area is approximately

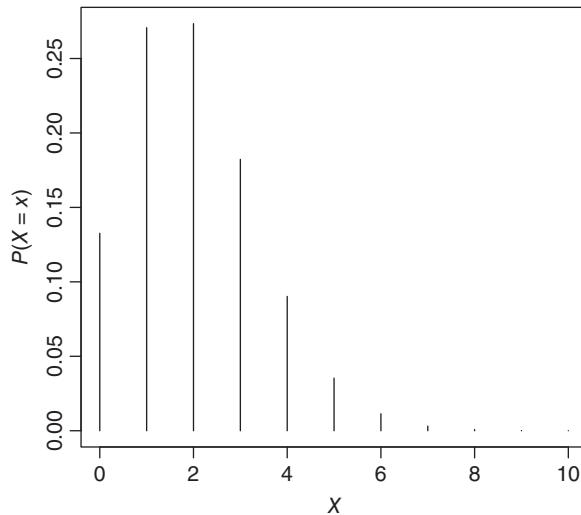


FIGURE 13.4 Binomial pdf $n = 100$, $p = 0.02$

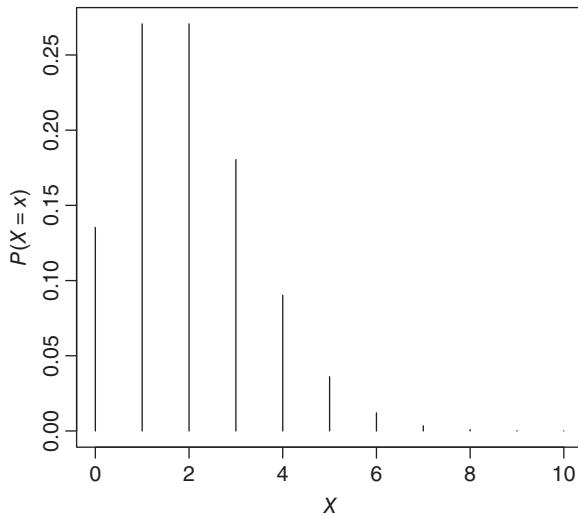


FIGURE 13.5 Poisson pdf $\lambda = 2$

half a million. What is the probability that, in any day, the number of customers will exceed 40?

An average of 50 new customers a day from 500,000 means the probability of a new customer is $50/500,000 = 0.0001$. We want $P(X > 40)$, where X is a binomial distribution with $n = 500,000$ and $p = 0.0001$.

Obviously,

$$P(X > 40) = 1 - P(X \leq 40)$$

TABLE 13.3 Binomial and Poisson Probabilities

Number of Bits in Error	Binomial $n = 10, p = 0.2$	Binomial $n = 50, p = 0.04$	Binomial $n = 100, p = 0.02$	Poisson $\lambda = 2$
$P(X = 0)$	0.10737	0.12989	0.13262	0.13534
$P(X = 1)$	0.26844	0.27060	0.27065	0.27067
$P(X = 2)$	0.30199	0.27623	0.27341	0.27067
$P(X = 3)$	0.20133	0.18416	0.18228	0.18045
$P(X \leq 3)$	0.87913	0.86088	0.85896	0.85713

which can be calculated in R using

```
1- pbinom(40, 500000, 0.0001)
```

which gives

```
[1] 0.9139407
```

Also, the Poisson approximation to the binomial, with $\lambda = 50$, is

```
1- ppois(40, 50)
```

```
[1] 0.91393
```

which is practically the same. There is a 91.4% chance that the number of new customers in any day will exceed 40. \triangleleft

The Poisson probabilities are easier to calculate than the binomial probabilities, which made the approximation useful in the past. However, advances in electronics could be said to have ended the usefulness of this Poisson approximation. Today's computers or even smartphones can calculate the binomial probabilities without difficulty, and accurately no matter how large n is or how small p is.

13.3 RANDOM EVENTS IN TIME AND SPACE

Today, the Poisson distribution is more likely to be seen as a distribution in its own right than as a limit of the binomial distribution. It has become fundamental in the analysis of problems relating to random occurrences of events in time or space and has been found to be of particular applicability in computer science. It has been used, for example, to estimate:

- the number of bugs in software;
- the number of faulty chips on wafers;
- the number of defects that occur on LSI (large scale intervention) chips.

It is expected that the bugs, faults, or defects occur rarely but are important when they do. Happenings such as these are often referred to as “events occurring independently in time.”

Other examples of the Poisson distribution include:

- the number of network failures per day;
- the number of visitors to a website per minute;
- the number of file server virus infections at a data center per week;
- the number of service requests to a web server accessing the cloud.

Let us look at some examples.

Example 13.3

You have observed that hits to your website occur at a rate of 2 a day on average.

Let X be the number of hits in a day.

$$P(X = 0) = P(\text{no hits in a day})$$

$$P(X = 1) = P(1 \text{ hit in a day})$$

$$P(X = 2) = P(2 \text{ hits in a day})$$

△

Example 13.4

You observe the number of telephone calls that arrive each day on your phone over a period of a year, and note that the average is 3.

Let X be the number of calls that arrive in any one day.

$$P(X = 0) = P(\text{no telephone calls arrive in any one day})$$

$$P(X = 1) = P(1 \text{ telephone call arrives in any one day})$$

$$P(X = 2) = P(2 \text{ telephone calls arrive in any one day})$$

△

Example 13.5

Records show that the average rate of job submissions in a busy computer center is four per minute.

Let X be the number of jobs arriving in any one minute.

$$P(X = 0) = P(\text{no submissions in a minute})$$

$$P(X = 1) = P(1 \text{ submission in a minute})$$

$$P(X = 2) = P(2 \text{ submissions in a minute})$$

△

Example 13.6

Records indicate that messages arrive to a computer server at the rate of 6 per hour.

Let X be the number of messages arriving in any one hour.

$$P(X = 0) = P(\text{no arrivals in an hour})$$

$$P(X = 1) = P(1 \text{ arrival in an hour})$$

$$P(X = 2) = P(2 \text{ arrivals in an hour})$$

△

Each of the above examples involves the occurrences of an event in a period of time. The average occurrence rate in the time period is labeled λ .

- $\lambda = 2$ in a day in Example 13.3
- $\lambda = 3$ in a day in Example 13.4
- $\lambda = 4$ in a minute in Example 13.5
- $\lambda = 6$ in an hour in Example 13.6

13.4 PROBABILITY DENSITY FUNCTION

The probability of x occurrences in a unit time interval is

$$P(X = x) = e^{-\lambda} \frac{\lambda^x}{x!}, \quad x = 0, 1, 2, \dots \quad (13.3)$$

where λ is the average occurrence in the time interval. Notice X takes values from 0 to ∞ , so the Poisson variable is another example of a discrete infinite variable. (You will recall that the geometric random variable from Chapter 10 is also countably infinite.)

Explicitly

$$\begin{aligned} P(X = 0) &= P(0 \text{ occurrences in the unit time interval}) = e^{-\lambda} \\ P(X = 1) &= P(\text{exactly 1 occurrence in the unit time interval}) = \lambda e^{-\lambda} \\ P(X = 2) &= P(\text{exactly 2 occurrences in the unit time interval}) = \frac{\lambda^2}{2!} e^{-\lambda} \\ P(X = 3) &= P(\text{exactly 3 occurrences in the unit time interval}) = \frac{\lambda^3}{3!} e^{-\lambda} \\ &\quad \dots \\ &\quad \dots \end{aligned}$$

Note that

$$\sum_{x=0}^{\infty} p(x) = \sum_{x=0}^{\infty} \frac{\lambda^x}{x!} e^{-\lambda} = e^{-\lambda} \sum_{x=0}^{\infty} \frac{\lambda^x}{x!}$$

Since

$$1, \quad \lambda, \quad \frac{\lambda^2}{2!}, \quad \frac{\lambda^3}{3!}, \quad \dots, \quad \frac{\lambda^x}{x!}$$

are the terms of the Taylor expansion of e^λ , then

$$\sum_{x=0}^{\infty} \frac{\lambda^x}{x!} = e^\lambda$$

which means that

$$e^{-\lambda} \sum_{x=0}^{\infty} \frac{\lambda^x}{x!} = e^{-\lambda} e^\lambda = 1$$

Therefore, Equation 13.3 represents a valid probability density function (pdf).

Also, we can confirm that λ is actually the mean.

$$E(X) = \sum_{x=0}^{\infty} xe^{-\lambda} \frac{\lambda^x}{x!} = e^{-\lambda} \sum_{x=0}^{\infty} x \frac{\lambda^x}{x!} \quad (13.4)$$

Examining

$$\begin{aligned} \sum_{x=0}^{\infty} x \frac{\lambda^x}{x!} &= \sum_{x=1}^{\infty} x \frac{\lambda^x}{x!} \text{ dropping the first term which is zero} \\ &= \sum_{x=1}^{\infty} \frac{\lambda^x}{(x-1)!}, \text{ cancelling } x \\ &= \lambda \sum_{x=1}^{\infty} \frac{\lambda^{x-1}}{(x-1)!} \\ &= \lambda \sum_{y=0}^{\infty} \frac{\lambda^y}{y!} = \lambda e^{\lambda} \text{ putting } x-1=y. \end{aligned}$$

Therefore,

$$\sum_{x=0}^{\infty} x \frac{\lambda^x}{x!} = \lambda e^{\lambda}$$

which gives

$$E(X) = e^{-\lambda} \sum_{x=0}^{\infty} x \frac{\lambda^x}{x!} = e^{-\lambda} \lambda e^{\lambda} = \lambda$$

By a similar derivation, it can be shown that

$$\sigma^2 = \sum_{x=0}^{\infty} (x - \lambda)^2 e^{-\lambda} \frac{\lambda^x}{x!} = \lambda$$

This means that

$$\sigma^2 = \lambda$$

Details of the necessary algebra are given in Appendix C.

Example 13.7

Consider a computer system with a Poisson job-arrival stream at an average of 2 per minute.

The probability that in any one-minute interval there will be zero job arrivals is

$$P(X = 0) = e^{-2} = 0.135$$

The probability that in any one-minute interval there will be three job arrivals is

$$P(X = 3) = e^{-2} \frac{2^3}{3!} = 0.18$$

△

13.4.1 Calculating Poisson pdfs with *R*

You will recall that in Section 13.2, we calculated the Poisson probabilities in *R* by prefixing the shortened name `pois` with “d.” We used the function `dpois`, along with the parameter λ as an argument, to calculate the probabilities.

To check our results in the previous example, write

```
dpois(0, 2)
[1] 0.1353353
```

which gives the probability of zero occurrences for a Poisson distribution with $\lambda = 2$.

```
dpois(3, 2)
[1] 0.1804470
```

which gives the probability of three occurrences for a Poisson distribution with $\lambda = 2$.

In Example 13.3, website hits, $\lambda = 2$

```
x <- 0:6
dpois(x, 2)
[1] 0.13533528 0.27067057 0.27067057 0.18044704
[5] 0.09022352 0.03608941 0.01202980
```

In Example 13.4, with $\lambda = 3$, the probabilities of getting 0, 1, 2, 3, ... calls in a day are

```
x <- 0:6
dpois(x, 3)
[1] 0.04978707 0.14936121 0.22404181 0.22404181
[5] 0.16803136 0.10081881 0.05040941
```

Similarly, for Examples 13.5 and 13.6, the probabilities can be generated using respectively

```
dpois(x, 4)
```

and

```
dpois(x, 6)
```

13.4.2 Plotting Poisson pdfs with R

To examine the pattern of the Poisson probabilities with different parameters, we plot the pdfs of Examples 13.3–13.6.

```
par(mfrow = c(2, 2)) # multiframe
x <- 0:12 #look at the first 12 probabilities
plot (x, dpois(x, 2), xlab = "Number of hits",
      ylab = "P(X = x)", type = "h",
      main = "Website hits: Poisson(2)", font.main = 1)

plot (x, dpois(x, 3), xlab = "Number of calls",
      ylab = "P(X = x)", type = "h",
      main = "Telephone calls: Poisson(3)", font.main = 1)

plot (x, dpois(x, 4), xlab = "Number of submissions",
      ylab = "P(X = x)", type = "h",
      main = "Job submissions: Poisson(4)", font.main = 1)

plot (x, dpois(x, 6), xlab = "Number of messages",
      ylab = "P(X = x)", type = "h",
      main= "Messages to server: Poisson(6)", font.main = 1)
```

generates Fig. 13.6.

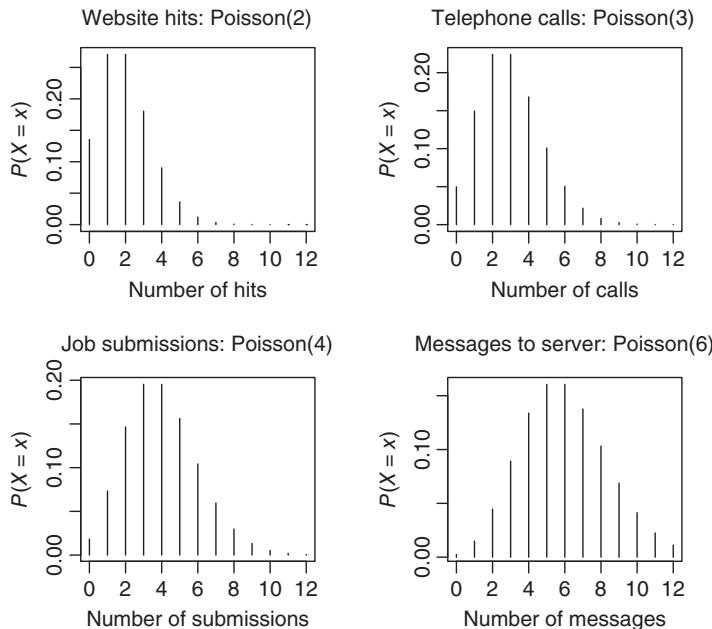


FIGURE 13.6 Poisson pdfs

We see in Fig. 13.6 that the pdf is negatively skewed when $\lambda = 2$. The skewness decreases as λ increases. At $\lambda = 6$, the pdf appears symmetrical and bell-shaped.

13.5 CUMULATIVE DISTRIBUTION FUNCTION

For the Poisson distribution, the cumulative distribution function (cdf) is

$$P(X \leq x) = e^{-\lambda} + e^{-\lambda} \frac{\lambda}{1} + e^{-\lambda} \frac{\lambda^2}{2} + \cdots + e^{-\lambda} \frac{\lambda^x}{x!}$$

Looking at Example 13.7 again, job arrivals in a minute, we might ask: What is the probability that there will be at most three job arrivals in a minute?

$$\begin{aligned} P(X \leq 3) &= P(0) + P(1) + P(2) + P(3) \\ &= e^{-2} + e^{-2} \frac{2}{1} + e^{-2} \frac{2^2}{2!} + e^{-2} \frac{2^3}{3!} \\ &= 0.1353 + 0.2707 + 0.2707 + 0.1805 \\ &= 0.8571 \end{aligned}$$

There is more than an 85% chance that the number of jobs that arrive will be at most three in any one minute.

To calculate the probability that there will be more than three arrivals, write

$$\begin{aligned} P(X > 3) &= 1 - P(X \leq 3) \\ &= 1 - 0.8571 \\ &= 0.1429 \end{aligned}$$

These are less than a 15% chance that there will be more than three job arrivals in a minute.

13.5.1 Calculating Poisson cdfs in R

The function in R for calculating the cumulative distribution is `ppois`.

To calculate the first few terms of the cdf of job arrivals with $\lambda = 2$, write

```
x <- 0:6
round(ppois(x, 2), 4)
```

to get

```
[1] 0.1353 0.4060 0.6767 0.8571 0.9473 0.9834 0.9955
```

13.5.2 Plotting Poisson cdfs

To examine the pattern of the Poisson probabilities with different parameters, we plot the cdfs of Examples 13.3–13.6.

```
par(mfrow = c(2, 2)) # multiframe
x <- 0:12
plot (x, ppois(x, 2), xlab = "Number of hits",
      ylab = "P(X = x)", type = "s",
      main = "Website hits: Poisson(2)", font.main = 1)

plot (x, ppois(x, 3), xlab = "Number of calls",
      ylab = "P(X = x)", type = "s",
      main = "Telephone calls: Poisson(3)", font.main = 1)

plot (x, ppois(x, 4), xlab = "Number of submissions",
      ylab = "P(X = x)", type = "s",
      main = "Job submissions: Poisson(4)", font.main = 1)

plot (x, ppois(x, 6), xlab = "Number of messages",
      ylab = "P(X = x)", type = "s",
      main = "Server messages: Poisson(6)", font.main = 1)
```

generates Fig. 13.7.

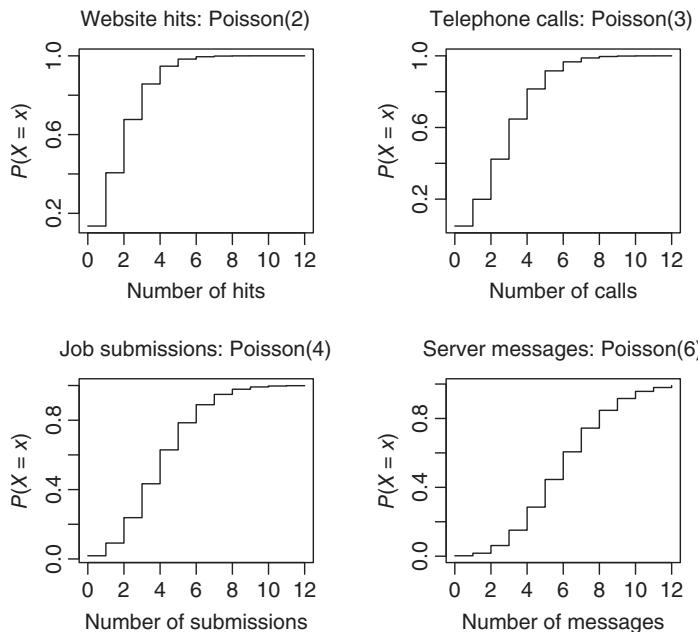


FIGURE 13.7 Poisson cdfs

13.6 THE QUANTILE FUNCTION

Looking again at Example 13.3, concerning the number of hits to a website, you might want to know the maximum number of hits that will arrive in any one day. While it is not possible to be 100% certain about the maximum, we are able to choose k so that you are, for example, 99% sure that the number of hits in a day never exceeds k .

We must choose k so that

$$P(X \leq k) \geq 0.99$$

You will recognize this as the quantile of the Poisson distribution.

In R

```
qpois(0.99, 2)
[1] 6
```

You are at least 99% sure that there will be less than or equal to six hits in any one day. Conversely, there is less than a 1% chance that you will get more than six hits in any one day.

In Example 13.4, calls to your phone, what is the maximum number k so that the probability is at least 0.9 that the number of calls in a day is at most equal to k ?

We must choose k so that

$$P(X \leq k) \geq 0.9$$

In R

```
qpois(0.90, 3)
[1] 5
```

You are 90% sure that the number of telephone calls that you get on your phone in any one day is between 0 and 5.

In Example 13.5, what is the maximum number k so that the probability that the number of job submissions in a minute is less than or equal to this value is 0.9?

```
qpois(0.9, 4)
[1] 7
```

There is at least a 90% chance that the number of job submissions in any minute does not exceed seven. Put another way, there is at most a 10% chance that there will be more than seven job submissions in any one minute.

In Example 13.6, what is the maximum number k so that the probability that the number of messages to the server in an hour is at least 0.75?

```
qpois(0.75, 6)
[1] 8
```

There is at least a 75% chance that the number of messages to the server is less than or equal to eight. Complementarily, this means that there is less than a 25%

chance that the number of messages exceeds 8; it is the third quartile of this Poisson distribution, that value which divides the distribution so that there is 25% above and 75% below.

To get the first quartile, that is, that value which divides the distribution so that there is 25% below and 75% above, write

```
qpois(0.25, 6)
[1] 4
```

Therefore, the interquartile range is [4, 8], which means that 50% of the time, the number of messages arriving will be between 4 and 8.

Example 13.8

A web server receives an average of 1,000 queries per day. How much excess capacity should it build in to be 95% sure that it can meet the demand?

This is the Poisson distribution with $\lambda = 1000$

Choose k so that

$$P(X \leq k) \geq 0.95$$

```
qpois(0.95, 1000)
[1] 1052
```

and to double check

```
ppois(1052, 1000)
[1] 0.9506515
```

The extra capacity needed is 52. △

13.7 ESTIMATING SOFTWARE RELIABILITY

When a computer software package is developed, testing procedures are often put in place to eliminate the bugs in the package. One common procedure is to try the package on a set of well-known problems to see if any error occurs. This goes on for a fixed period of time with all the errors being noted. The testing then stops and the package is checked to determine and remove the bugs that caused the errors.

Example 13.9

The mean number of errors due to a particular bug occurring in a minute is 0.0001. The questions the Poisson distribution can answer are

- What is the probability that no error will occur in 20 minutes?

- How long would the program need to run, to ensure that there will be a 99.95% chance that an error will show up to highlight this bug?

It is usual to use a Poisson distribution to model the occurrences of bugs in this software, in this case with mean $\lambda = 0.0001$ per minute or equivalently $\lambda = 20(0.0001) = 0.002$ per 20-minute interval.

The probability that no error will occur in a 20-minute interval is

$$P(0) = e^{-0.002} = 0.998002$$

There is a 99.8% chance that no error will occur in 20 minutes. There is just a 0.2% chance that an error will show up in the first 20 minutes.

Obviously, the program will need to run a lot longer than 20 minutes, if we are to find the bug with a high degree of certainty.

If we want to be 99.95% sure of finding this bug, we need to let it run so that

$$P(X \geq 1) \geq 0.9995$$

or equivalently

$$P(X = 0) < 0.0005$$

Now, the probability of no occurrence in a time interval of length k minutes is

$$P(\text{no occurrence in } k \text{ minutes}) = e^{-(0.0001)k}$$

So, we must choose k so that

$$e^{-(0.0001)k} < 0.0005$$

or equivalently

$$1 - e^{-(0.0001)k} \geq 0.9995$$

We go to R to examine the probabilities of at least one error in differing time spans.

```

k <- seq(50000, 100000, 5000) #Running time 50,000
      minutes upwards
y <- 1 - exp(-(0.0001)*k) #At least one in k minutes
plot(k, y,
      xlab = "Running time of the package in minutes",
      ylab = "Probability of at least one bug")
abline(h = 0.9995)

```

This generates Fig. 13.8.

The horizontal line is drawn at the 0.9995 probability. Reading from the graph, we see that it will take approximately $7.5e + 04 = 75,000$ minutes, that is, 1,250 hours or 52 days, to be 99.95% confident that an error will show up to highlight this bug. \triangleleft

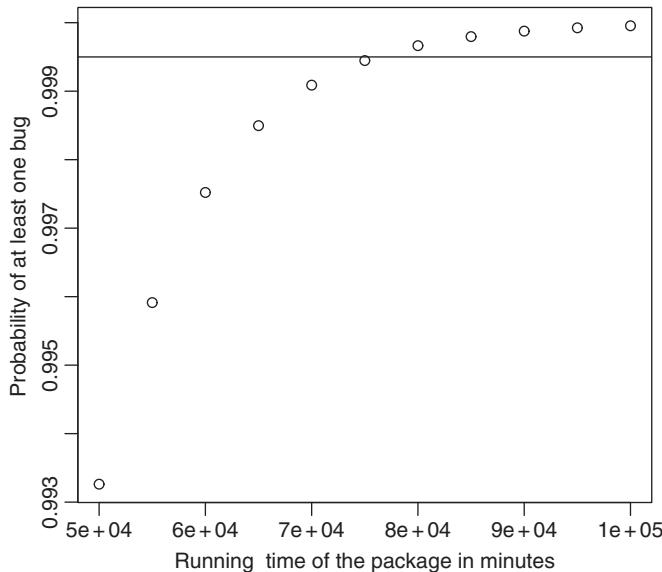


FIGURE 13.8 Bug Detection

13.8 MODELING DEFECTS IN INTEGRATED CIRCUITS

The key to the success of any integrated circuit manufacturing operation is the manufactured semiconductor chips that can be sold to customers. The chips have to survive a battery of device tests and are deemed to be defective if they contain a fault. A fault is defined as a defect that causes an integrated circuit to fail when it is tested for a specific application. Only chips that have survived these tests are sold to the customer. This fraction is known as the yield.

In integrated circuit fabrication, it is important to correctly predict the yield of the designed device and to develop models for the distribution of defects in wafers. It is necessary to determine the average defect density which is the distribution of defect counts in tiny areas on wafers.

The simplest model of defect spatial distributions is to assume that the defects are randomly distributed and to use this to predict the yield of chips of the proposed area. For a given defect density (average number of defects per wafer), the Poisson distribution is used to predict the number of chips with 0, 1, 2, and so on defects.

Example 13.10

The average number of defects per wafer (defect density) is 3. The redundancy built into the design allows for up to 4 defects per wafer. What is the probability that the redundancy will not be sufficient, if the defects follow a Poisson distribution?

$$P(X = k) = e^{-3} \frac{3^k}{k!}$$

The redundancy will not be sufficient when the number of defects exceeds 4. Let X be the number of defects per wafer.

$$\begin{aligned} P(X > 4) &= 1 - (P(X = 0) + P(X = 1) + P(X = 2) + P(X = 3) + P(X = 4)) \\ &= 1 - e^{-3} \left(1 + \frac{3^1}{1!} + \frac{3^2}{2!} + \frac{3^3}{3!} + \frac{3^4}{4!} \right) \end{aligned}$$

In R, we obtain $P(X > 4)$ using `ppois`.

```
1-ppois(4, 3)
[1] 0.1847368
```

There is over an 18% chance that the defects will exceed redundancy. \triangleleft

13.9 SIMULATING POISSON PROBABILITIES

In this section, we show how to simulate a Poisson distribution and observe the pattern of occurrences. We use

```
rpois(k, λ)
```

which generates k values from a Poisson distribution with parameter λ . The first argument specifies the number of simulations to carry out, and the second argument is the parameter.

Let us look again at Example 13.3, the number of hits to a website with an average of 2 per day.

```
hits <- rpois(100, 2)
```

generates 100 random numbers from a Poisson distribution with parameter $\lambda = 2$. This can be taken as simulating the number of hits to a website considered in Example 13.3.

This output represents the number of hits in each of the 100 days. The following R code generates Fig. 13.9.

```
hits <- rpois(100, 2)
x <- 1:100
plot(x, hits, xlab = "Day",
      ylab = "Number of hits", type = "l")
```

The argument `type = "l"` causes the points to be joined.

We see from Fig. 13.9 that on the first day, there was just one hit, no hit the next day followed by three hits on the third day, and so on. The highest number of hits was 7 and this occurred on just one day. The lowest number of hits was 0.

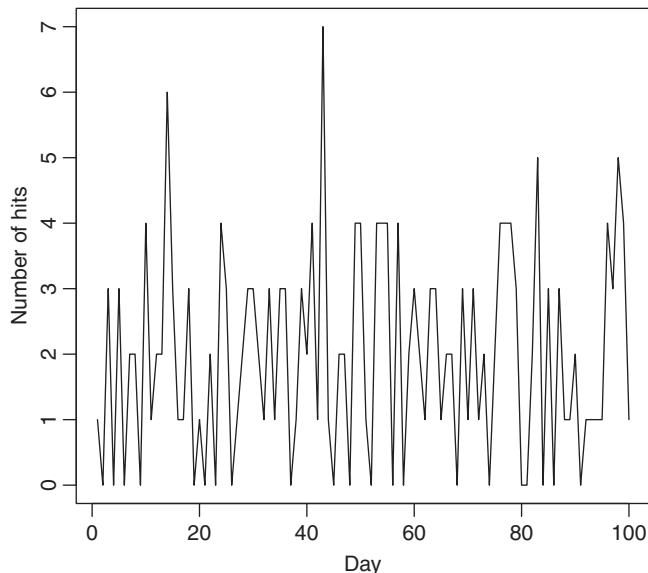


FIGURE 13.9 Simulated Web Hits: Poisson Distribution, $\lambda = 2$

To summarize the output in table form write in R,

```
table(hits)
```

which gives

hits	0	1	2	3	4	5	6	7
21	23	18	20	14	2	1	1	

Here, we see that on 21 of the 100 days, there were no hits at all. Note again that the maximum number of hits was 7 which occurred on just one day. Hits of 1, 2, or 3 per day occurred most often, on 61 of the 100 days in total.

We can also express our simulations as a table of relative frequencies which are estimates of the Poisson probabilities.

```
table(hits) / length(hits)
```

gives

hits	0	1	2	3	4	5	6	7
	0.21	0.23	0.18	0.20	0.14	0.02	0.01	0.01

Depicting this graphically using *R*

```
plot(table(hits)/length(hits),
     xlab = "Number of hits to a website in a day",
     ylab = "Relative frequency", type = "h")
```

gives Fig. 13.10.

Notice that the relative frequencies in Fig. 13.10 are similar to the the actual probabilities obtained with

```
round(dpois(x, 2), 2)
[1] 0.27 0.27 0.18 0.09 0.04 0.01 0.00
```

Also, the mean of the simulated data is

```
mean(hits)
[1] 1.98
```

which is not too far away from the true $\lambda = 2$.

The variance of the simulated data is obtained in *R* with

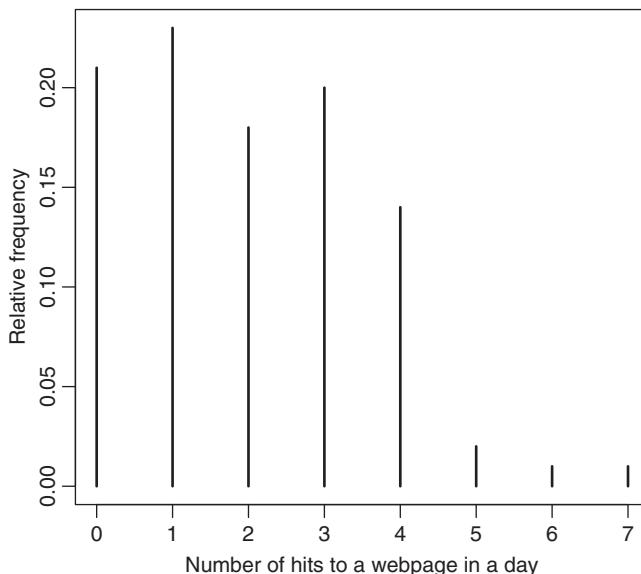


FIGURE 13.10 Simulated Web Hits: Poisson Distribution, $\lambda = 2$

```
var(hits)
[1] 2.44404
```

The standard deviation is

```
sd(hits)
[1] 1.563343
```

Notice that the simulated variance is greater than the true variance of $\lambda = 2$. Maybe we should have carried out a greater number of simulations. Try it yourselves with 1,000 simulations.

EXERCISES 13.1

1. The manufacturer of disk drives in one of the well-known brands of computers expects 1% of the disk drives to malfunction during the warranty period. Calculate the probability that, in a sample of 100 disk drives, not more than three will malfunction.
2. The average rate of job submissions in a computer center is four per minute. If it can be assumed that the number of submissions per minute interval is Poisson distributed, calculate the probability that:
 - (a) at least two jobs will be submitted in any minute;
 - (b) no job will be submitted in any minute;
 - (c) no more than one job will be submitted in any one-minute interval.
3. A new user of a smartphone receives an average of three messages per day. If it can be assumed that the arrival pattern of these messages is Poisson, calculate the following probabilities:
 - (a) exactly three messages will be received in any day;
 - (b) more than three messages will arrive in a day;
 - (c) a day will pass without a message;
 - (d) if one day has passed without a message, six messages will arrive the next day.
4. Contamination is a problem in the manufacture of optical storage disks (CDs). The number of particles of contamination that occur on an optical disk has a Poisson distribution, and the average number of particles per square centimeter of media surface is 0.1. The area of a disk under study is 100 cm^2 . Find the probability that 12 particles occur in this area.
5. When computer disk manufacturers test a disk, they write to the disk and then test it using a certifier. The certifier counts the number of missing pulses or errors. If it can be assumed that the number of errors in a test area on a disk has a Poisson distribution with $\lambda = 0.2$,

- (a) what is the expected number of errors per test area?
 - (b) what percentage of test areas have two or fewer errors?
6. The number of messages sent to a computer bulletin board is a Poisson variable with a mean of six messages per hour. What is the probability that
- (a) 5 messages are received in an hour?
 - (b) 10 messages are received in an hour and a half?
 - (c) less than 2 messages are received in one half hour?
7. Assume that the number of errors along a magnetic recording surface is a Poisson random variable with a mean of one error in every 10^5 bits. A sector of data consists of 4,096 eight-bit bytes.
- (a) What is the mean number of errors per sector?
 - (b) What is the probability of more than one error in a sector?
8. Messages arrive to a computer server according to a Poisson distribution with a mean rate of one per five-minute interval. Determine the length of an interval of time such that the probability that no messages arrive during this interval is 0.90.
9. The number of machine failures per day in a certain plant has a Poisson distribution with parameter 4. Present maintenance facilities can repair three machines per day, otherwise a contractor is called out. On any given day, what is the likelihood of having machines repaired by a contractor?
10. The probability that an algorithm fails to give the correct answer is 1/100. One thousand different data sets are run using this algorithm.
- (a) What is the probability that:
 - i. the algorithm will give the correct answer using all data sets?
 - ii. at least one data set will get the wrong answer?
 - iii. the algorithm will give the wrong answer in no more than five data sets?
 - (b) What is the most likely number of data sets for which the algorithm gives the wrong answer?
11. In an Internet trading system, the network is known to go down once a week on average.
- (a) Compute the probability that:
 - i. there will be more than two breakdowns in any one week;
 - ii. a week will go by without a breakdown.
 - (b) What is the maximum number of breakdowns that you can expect in a week with 95% confidence?
 - (c) If each network breakdown results in a loss of €1,000 revenue, calculate the average weekly loss due to network failures. What is the variance?

12. It is estimated that Google receives an average of 40,000 search queries every second. In the last two seconds, Google received 50,000 and 55,000 queries, respectively. Is there a need to suspect that maybe the average has increased? You may assume query arrivals follow a Poisson distribution.
13. When a virus attack occurs, an average of 2% of files on a disk are infected. If it can be assumed that files are attacked independently of one another, calculate the probability that in a disk containing 200 files:
 - (a) no files will be infected;
 - (b) less than two files will be infected.
14. A database server is designed to handle a maximum of 100 transactions per second. If transactions arrive at the rate of 75 per second, what is the probability that the server is overloaded? You may assume that the arrivals follow a Poisson distribution.
15. When a fault occurs in your laptop, it can be cleared by a reboot in 95% of cases, and in the other 5% of cases, a repair technician needs to be called. What is the probability that in 100 faults, all can be cleared by a reboot?

13.10 PROJECTS

1. Develop a front end to *R*, which provides the user with an opportunity to invoke the Poisson distribution by clicking an icon on screen, to input the relevant values of the parameter λ and to obtain plots of the pdf and cdf.
2. We have proved that the binomial tends to the Poisson as p tends to 0 and n tends to infinity. Specifically

$$\binom{n}{x} p^x (1-p)^{n-x} \rightarrow e^{-np} \frac{(np)^x}{x!} \quad x = 0, 1, 2, \dots, n$$

as n increases and p decreases with constant np . Use *R* to determine how big n should be and how small p should be for this approximation to be valid?

Hint: Start with $p = 0.1$ and $n = 10$, then start increasing n ($n = 20, 30, \dots$) until the approximation is valid. Then take a smaller p and repeat the process. Perhaps your “rule of thumb” might give a minimum value of n and a maximum value of p .

3. Write a program in *R* to generate Figs. 13.6 and 13.7, using `for` loops and `bquote` functions.

REFERENCES

- Poisson, S.D. (1937), *Research on the Probability of Criminal and Civil Verdicts*, Bachelor (in French).
- von Bortkiewicz, L. (1898) *The Law of Small Numbers*, Lapsiz, Germany, Teubner (in German).

14

SAMPLING INSPECTION SCHEMES

14.1 INTRODUCTION

When products are manufactured, they are often packed in batches and subjected to a quality test before delivery to the customers. A sample of products is taken from each batch and examined for defectives. A decision to reject or accept the batch is made on the basis of the number of defective items in the sample. If too many defectives are found in a sample, the batch is rejected. After that, it may be sold off cheaply or subjected to a 100% inspection.

Sampling inspection schemes may differ with respect to the sample size taken from each batch, the allowable number of defectives, and the post inspection action. In this chapter, we look at some such inspection schemes.

Samples are usually taken, rather than a complete inspection of the products, in order to save time and money. Of course, in the case of destructive testing, in which the inspection process involves damage to the product so that it cannot be used again, sampling is essential, and it is important to keep the sample size as small as possible, while maintaining adequate monitoring of the quality of the products.

In contrast, in precision engineering and dependable systems, it is more common to inspect every item, in which case there are few statistical problems, and most of the following remarks do not apply.

14.2 SINGLE SAMPLING INSPECTION SCHEMES

Example 14.1

A certain item is packed 100 to a box. Before distributing to the retailers, each box is subjected to a quality control test whereby a sample of 10 items is randomly drawn from each box. The box is acceptable if it contains less than or equal to one defective item, otherwise the box is rejected. \triangleleft

Example 14.2

Random access memory chips are packed in batches of 1,000, and 20 chips are randomly selected from each batch for inspection. If more than two chips are found to be defective, the complete batch is rejected. \triangleleft

Example 14.3

A manufacturer of microchips packs them in batches of 10,000. One hundred chips are chosen randomly from each batch and subjected to a quality test. If three or fewer chips are found to be defective in the sample, the batch is accepted, otherwise it is rejected. \triangleleft

These are examples of what is known as a *single sampling inspection scheme*, where a decision to accept or reject the batch is based on an inspection of a single sample.

If we denote the size of a batch by N , the sample size taken from each batch by n , and the maximum number of defectives allowed in the sample in order to accept the batch by c , then

$N = 100, n = 10$, and $c = 1$ in Example 14.1

$N = 1,000, n = 20$, and $c = 2$ in Example 14.2

$N = 10,000, n = 100$, and $c = 3$ in Example 14.3

Generally with p , the proportion defective in the batch, and $q = 1 - p$, the proportion nondefective, the batch looks like

Np	$N(1 - p)$
Defective	Good

The proportion of defectives p , sometimes called the *batch quality*, will differ from batch to batch.

Let X be the number of defectives found in a sample of size n drawn from a batch. A batch is accepted provided $X \leq c$. The probability of acceptance of batches containing the proportion p defective is

$$P(\text{acceptance}, p) = P(X \leq c) = P(X = 0) + P(X = 1) + \cdots + P(X = c)$$

Sampling is done without replacement, so the number of defectives in a sample follows a hypergeometric distribution. We can write

$$P(\text{acceptance}, p) = \frac{\binom{Np}{0} \binom{N(1-p)}{n}}{\binom{N}{n}} + \frac{\binom{Np}{1} \binom{N(1-p)}{n-1}}{\binom{N}{n}} + \cdots + \frac{\binom{Np}{c} \binom{N(1-p)}{n-c}}{\binom{N}{n}} \quad (14.1)$$

The objective of sampling inspection is to reject batches with low quality, that is, high p , and to accept those with high quality, that is, low p .

14.3 ACCEPTANCE PROBABILITIES

In Example 14.1, the rule is

Allow up to one defective in samples of size 10 drawn without replacement from batches of size 100.

A batch is accepted provided the number of defectives in the sample $X \leq 1$. We can write

$$P(\text{acceptance}, p) = \frac{\binom{100p}{0} \binom{100(1-p)}{10}}{\binom{100}{10}} + \frac{\binom{100p}{1} \binom{100(1-p)}{9}}{\binom{100}{10}}$$

For example, with $p = 0.1$

$$\begin{aligned} P(\text{acceptance}, 0.1) &= \frac{\binom{10}{0} \binom{90}{10}}{\binom{100}{10}} + \frac{\binom{10}{1} \binom{90}{9}}{\binom{100}{10}} \\ &= 0.3304762 + 0.4079953 \\ &= 0.7384715 \end{aligned}$$

which is the acceptance rate of batches containing 10% defective.

To calculate the acceptance probabilities for batches with varying defective rates in R , for $p = 0.05, 0.1$, and 0.15 say, first put the defective rates into a vector.

```
p <- c(0.05, 0.1, 0.15)
```

Then, use `phyper` to calculate $P(X \leq 1)$, the cdf of the hypergeometric distribution.

```
phyper(1, 100*p, 100*(1-p), 10)
[1] 0.9231433 0.7384715 0.5375491
```

The probability of accepting batches:

- with $p = 0.05$, that is, 5 defectives and 95 good, is 0.9231433;
- with $p = 0.10$, that is, 10 defectives and 90 good, is 0.7384715;
- with $p = 0.15$, that is, 15 defectives and 85 good is 0.5375491.

Looking at Example 14.2, the rule is

Allow up to two defectives in a sample of size 20 taken from a batch containing 1,000 components.

This means that a batch is accepted if the sample contains at most 2 defectives, otherwise it is rejected. In previous chapters, we showed that the binomial may be used as an approximation to the hypergeometric when the batch size N is large. In this case, since $N = 1,000$, sampling 20 items without replacement has a minuscule effect on the proportion of defectives in the batch; therefore, we can use the binomial distribution to approximate the hypergeometric. The acceptance probability, given in Equation 14.1 may be approximated by

$$P(\text{acceptance}, p) \approx (1-p)^{20} + \binom{20}{1} p^1 (1-p)^{19} + \binom{20}{2} p^2 (1-p)^{18}$$

It should be pointed out however, that calculating probabilities with R is an easy matter, so the limiting distributions are not as important as they would be if we were doing them manually. We will use them as a matter of preference because, on the one hand, they are more elegant and, on the other hand, they involve fewer calculations than would be needed for the original hypergeometric distribution, and hence are likely to contain fewer round-off errors when N is large.

If a batch contains 10% defectives, the probability that it will pass this quality test and will be accepted is

$$\begin{aligned} P(\text{acceptance}, 0.1) &\approx (0.9)^{20} + \binom{20}{1} (0.1)^1 (0.9)^{19} + \binom{20}{2} (0.1)^2 (0.9)^{18} \\ &= 0.122 + 0.270 + 0.285 \\ &= 0.677 \end{aligned}$$

There is a 67.7% chance that batches containing 10% defectives will be accepted.

Let us examine the acceptance rates of batches containing $p = 0.05$, 0.1, and 0.15 in R .

```
p <- c(0.05, 0.1, 0.15)
pbinom(2, 20, p)
[1] 0.9245163 0.6769268 0.4048963
```

meaning that there is approximately a 92%, 68%, and 40% chance of accepting batches with 5%, 10%, and 15% quality levels, respectively.

Finally, we examine Example 14.3 where the rule is
Allow up to three defectives in a sample of size 100 drawn from batches of size 10,000.

With the sample size n large, and the proportion defective p small, we saw in Chapter 13 that the Poisson distribution can be used to approximate the probabilities using $\lambda = np$. Therefore, the acceptance probability may be approximated by

$$P(\text{acceptance}, p) \approx e^{-\lambda} + e^{-\lambda}\lambda + e^{-\lambda}\frac{\lambda^2}{2!} + e^{-\lambda}\frac{\lambda^3}{3!} \quad \text{where } \lambda = 100p$$

If a batch contains 10% defectives, $\lambda = 100 \times 0.1 = 10$

$$\begin{aligned} P(\text{acceptance}, 0.1) &\approx e^{-10} + e^{-10}\frac{10^1}{1} + e^{-10}\frac{10^2}{2!} + e^{-10}\frac{10^3}{3!} \\ &= 0.00005 + 0.00045 + 0.00227 + 0.00757 \\ &= 0.01034 \end{aligned}$$

There is just over a 1% chance of accepting batches with 10% defectives.

We can examine the acceptance rates of batches containing $p = 0.05, 0.1$, and 0.15 . From R , we get

```
p <- c(0.05, 0.1, 0.15)
round(ppois(3, 100*p), 4)
```

calculates $P(X \leq 3)$ when $p = 0.05, 0.1$, and 0.15 , rounded to four decimal places. The output is

```
[1] 0.2650 0.0103 0.0002
```

meaning that there is approximately 26%, 1%, and practically a 0% chance of accepting batches with 5%, 10%, and 15% quality levels, respectively.

To understand better what these probabilities mean, we use R to simulate the schemes in the next section.

14.4 SIMULATING SAMPLING INSPECTION SCHEMES

Let us first look at Example 14.1.

Recall `rhyper` selects random samples from a hypergeometric distribution.

```
rhyper(100, 5, 95, 10)
```

generates 100 simulated values from a hypergeometric distribution. The output represents the number of defectives in samples of size 10 chosen from batches of size 100 containing 5 defectives.

Obviously,

```
rhyper(100, 10, 90, 10)
```

and

```
rhyper(100, 15, 85, 10)
```

simulate defective rates in samples drawn from batches with 10% and 15% quality levels, respectively.

To plot the output for $p = 0.05$, write in *R*

```
p <- 0.05
plot(rhyper(100, 100*p, 100*(1-p), 10), ylim = c(0,5),
      xlab = "Batch number", ylab = "Number of defectives",
      main = bquote(p ==.(p) ), font.main = 1)
abline(h = 1.2)
```

Here, `bquote(p ==.(p))` writes the argument except the expression in `.` which is evaluated, and `abline(h = 1.2)` draws a horizontal line just above 1, the allowable number of defectives. A sample containing more than one defective causes the corresponding batches to be rejected.

To generate Fig. 14.1, set

```
prob <- c(0.05, 0.10, 0.15)
```

and use a `for` loop

```
par(mfrow = c(1,3))
for (p in prob)
{
  plot(rhyper(100, 100*p, 100*(1-p), 10), ylim = c(0,5),
        xlab = "Batch number", ylab = "Number of defectives",
        main = bquote(p ==.(p) ), font.main = 1)
  abline(h = 1.2)
}
```

Figure 14.1 gives the simulated schemes when $p = 0.05, 0.1$, and 0.15 .

The horizontal line in Fig. 14.1 is drawn just above the allowable number of defectives. Samples containing numbers of defectives above this lead to rejection of the batches from which they were drawn. Notice that the majority of batches are accepted when the batch quality is 5%. There are in fact just five samples containing more than one defective. The batches from which these samples were drawn will be rejected.

Not surprisingly, with a quality level of $p = 0.1$, there is a greater number of defectives in the samples than we obtained when $p = 0.05$. Hence, many more batches are rejected based on the sample results.

As one would hope, with a high 15% rate of defectives, approximately half of the samples contain more than one defective, leading to rejection of the corresponding batches.

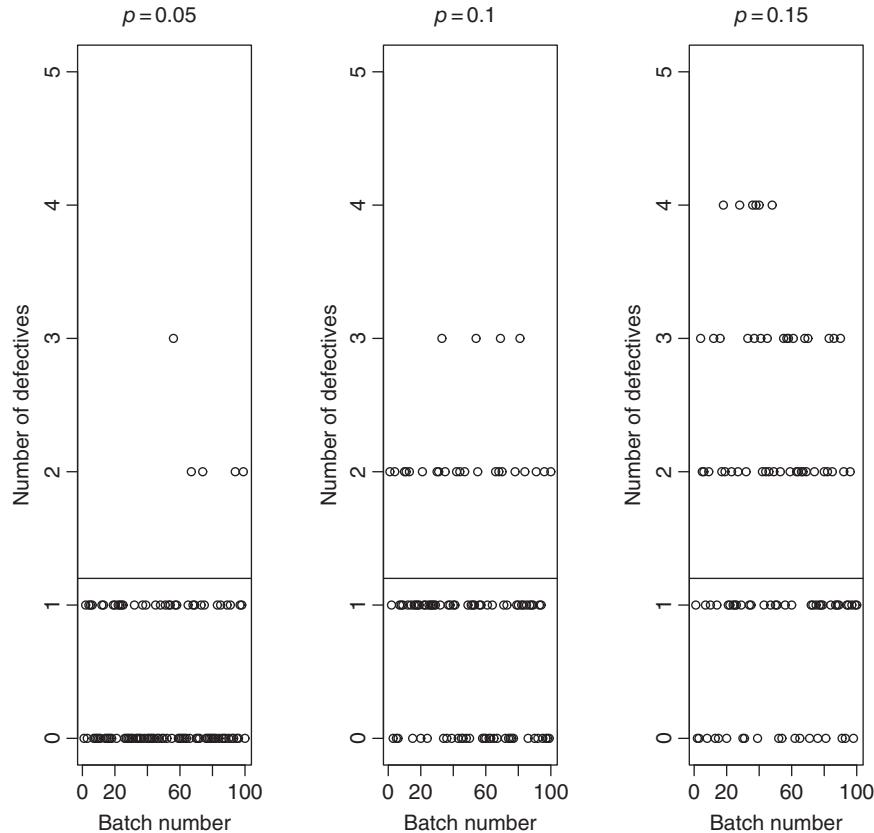


FIGURE 14.1 Simulated Number of Defectives in Samples of $n = 10$ from Batches of Size 100 with Defective Rates of 0.05, 0.1, and 0.15

In Example 14.2, samples of size $n = 20$ are drawn from batches containing 1,000 components, and up to 2 defectives are allowed before the batch is rejected. As before, we use R to simulate the performance of this scheme. If we let

```
prob <- c(0.05, 0.10, 0.15)
```

we can use a `for` loop to generate Fig. 14.2.

```
par(mfrow = c(1,3))
for (p in prob)
{
  plot(rbinom(100, 20, p), ylim = c(0,8),
    xlab = "Batch number", ylab = "Number of defectives",
    main = bquote(p == .(p)), font.main = 1)
  abline(h = 2.2)
}
```

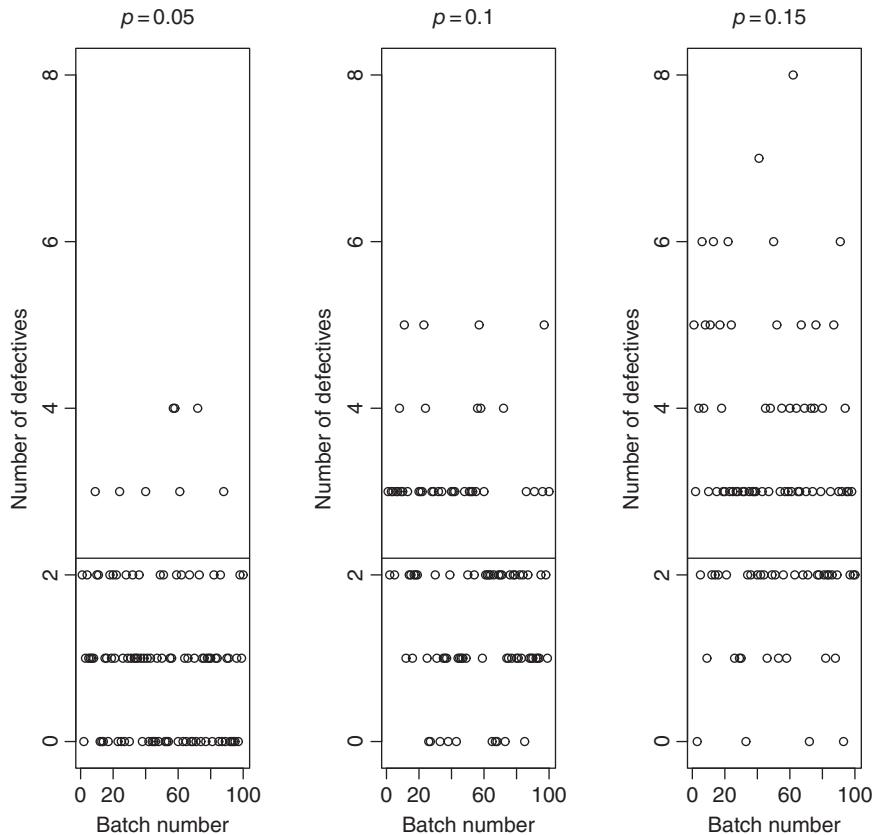


FIGURE 14.2 Simulated Number of Defectives in Samples of $n = 20$ drawn from Large Batches with Defective Rates of $p = 0.05$, 0.1 , and 0.15

Notice in Fig. 14.2 that, when $p = 0.05$, all but seven batches are accepted. When $p = 0.1$, there are many batches which yield samples with 3, 4, and even 5 defectives. When it comes to batches with 15% defective, we can see that a large proportion of batches contain 3 or more, even reaching 8 in one case. All of these batches will be rejected with this scheme.

Example 14.3 allows up to 3 defectives in samples of size $n = 100$ drawn from batches containing 10,000 microchips.

If we let

```
prob = c(0.05, 0.10, 0.15)
```

and use a `for` loop,

```
par(mfrow = c(1, 3))
for (p in prob)
```

```
{
plot(rpois(100, 100*p), ylim = c(0, 20),
      xlab = "Batch number", ylab = "Number of defectives",
      main = bquote(p ==.(p) ), font.main = 1)
abline(h = 3.2)
}
```

we get Fig. 14.3.

You will observe in Fig. 14.3 that no batch is accepted with a quality level of 15% or 10%. This scheme is therefore more successful than the previous two in rejecting batches with these quality levels. This is not surprising since it involves a higher level of inspections, and a relatively lower allowable number of defectives, “3 in 100” compared to “1 in 10” in Example 14.1, and “2 in 20” in Example 14.2.

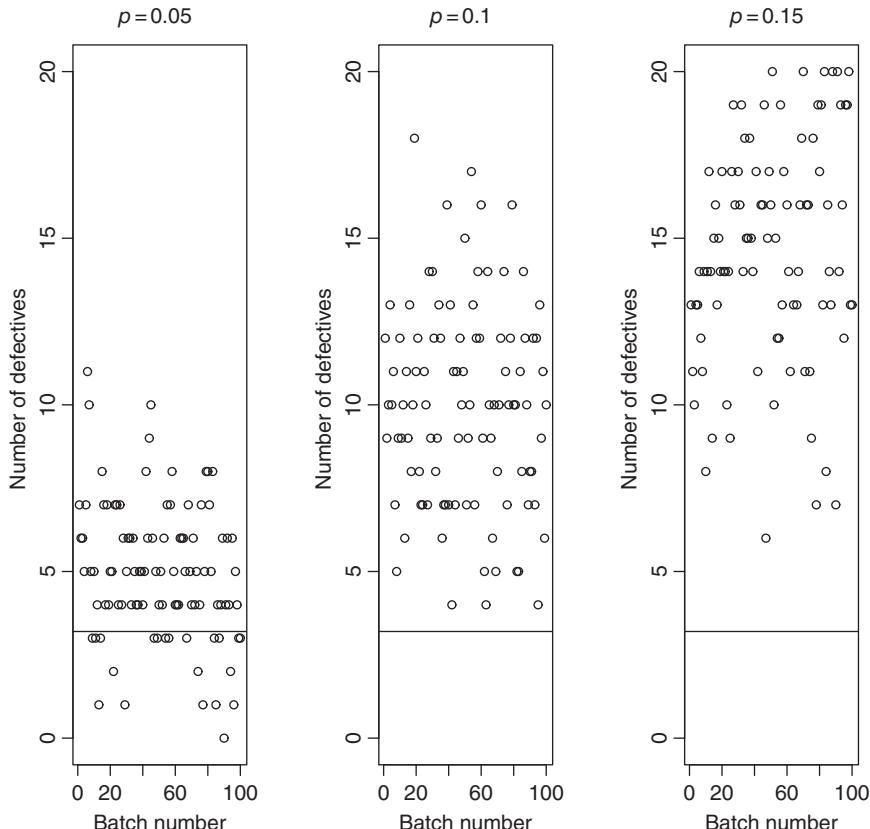


FIGURE 14.3 Simulated Number of Defectives in Samples of $n = 100$ drawn from Large Batches with Defective Rates of 0.05, 0.1, and 0.15

14.5 OPERATING CHARACTERISTIC CURVE

In the previous section, we illustrated how the acceptance or rejection of batches varies according to:

- the rules of the sampling inspection scheme;
- the actual proportion p of defectives in the batches.

The acceptance of batches with different quality levels can be examined in more detail by means of what is called the *operating characteristic* (OC) curve which is simply a plot of p , the batch quality, against P , its probability of acceptance, calculated for all values of p . For the three schemes discussed in Examples 14.2.1–14.2.3, we use R to obtain their *operating characteristic curves*.

```

par(mfrow = c(1,3))
p <- seq(0, 1, 0.01)
P <- phyper (1, 100*p, 100*(1-p), 10)
plot(p, P, type = "l", xlab = "Proportion defective",
      ylab = "Acceptance probability", xlim = c(0, 0.5), font.main = 1)

p <- seq(0, 1, 0.001)
P <- pbinom(2, 20, p)
plot(p, P, type = "l", xlab = "Proportion defective",
      ylab = "Acceptance probability", xlim = c(0, 0.5), font.main = 1)

p <- seq(0, 1, 0.0001)
P <- ppois(3, 100*p)
plot(p, P, type = "l", xlab = "Proportion defective",
      ylab = "Acceptance probability", xlim = c(0, 0.5), font.main = 1)

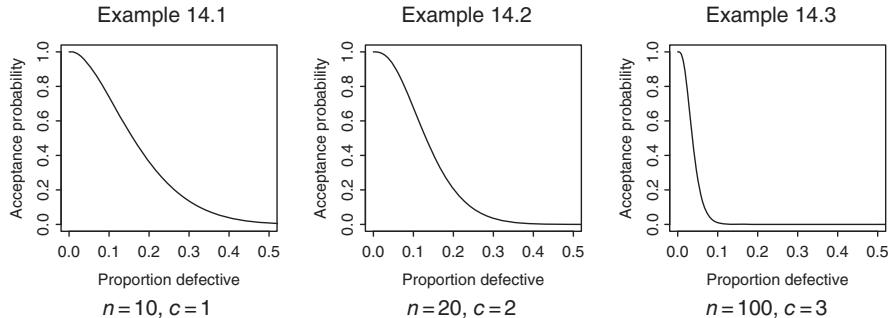
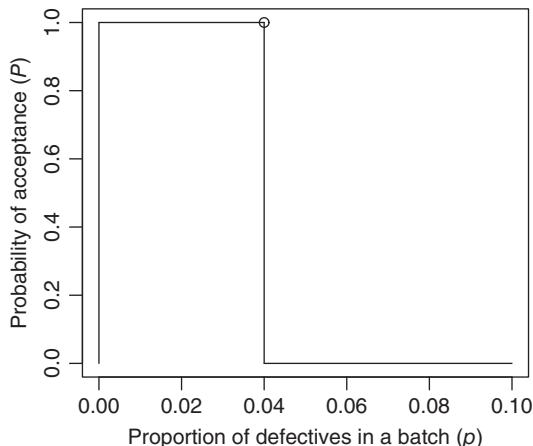
```

which generates Fig. 14.4.

You will notice, from Fig. 14.4, that each of the three OC curves fall steeply at a certain stage, meaning that there is a high probability of accepting batches with low proportions defective, and a low probability of accepting batches with high proportions defective. Notice that the OC curve for Example 14.3 falls the most steeply. Example 14.3 has the largest sample size $n = 100$, and the lowest allowable proportion defective, 3 out of 100. Example 14.1 is the worst of all since it has the smallest sample size, and allows 10% defectives. Example 14.2 lies in between; while this scheme, similar to Example 14.1, allows 10% defective, it inspects more items.

Ideally, we should design a scheme so that acceptable batches would be accepted with certainty, and unacceptable batches would have a zero chance of acceptance. Figure 14.5 is an example of the ideal OC curve, when batches containing up to 4% defective are acceptable.

When a sample is inspected rather than the complete batch, this ideal is never achieved. With less than a 100% inspection, there will always be some risk of not

**FIGURE 14.4** Operating Characteristic Curves**FIGURE 14.5** Ideal Operating Characteristic Curve which Rejects Batches containing more than the Acceptable 4% Defectives

accepting “good” batches, and not rejecting “bad” batches. The most we can hope for is that there will be a high probability of accepting batches with low defective rates and of rejecting batches with high defective rates. Figure 14.5 can be generated in R with

```
x <- 0.04
y <- 1
plot(x, y, xlim = c(0, 0.1), ylim = c(0, 1),
      xlab = "Proportion of defectives in a batch (p)",
      ylab = "Probability of acceptance (P)")
lines(c(0, 0.04), c(1,1), lty = 1)
lines(c(0.04, 0.04), c(0,1), lty = 1)
```

```
lines(c(0, 0), c(0,1), lty = 1)
lines(c(0.04, 0.1), c(0,0), lty = 1)
```

The argument `lty = 1` in the `lines` function has the effect of drawing a continuous line between the designated points

14.6 PRODUCER'S AND CONSUMER'S RISKS

The probability of rejecting “good” batches, those with low proportions defective, and accepting “bad” batches, those with high proportions defective, are known respectively as the producer’s and consumer’s risks. They measure how the OC curve differs from the ideal.

- *The producer’s risk, rejecting batches with acceptable defective rates:*
For example, if the producer gives a guarantee that batches contain no more than 4% defective, then batches with less than or equal to 4% defective are deemed “good” batches and should be accepted. Rejecting these involve unnecessary extra costs to the producer. This proportion of defective (4%) is called the acceptable quality level (AQL).
- *The consumer’s risk, accepting batches with unacceptably high proportions defective:*
For example, batches containing 10% defective may be deemed “bad” and should be rejected. Accepting a bad batch means that the consumer is given a batch with an unacceptably high level of defectives. This proportion of defectives is called the limiting quality (LQ).

Suppose, in Examples 14.1–14.3, “good” batches are defined to be those with less than or equal to 4% defective, and “bad” batches are those with more than 10% defectives, we can calculate the producer’s risk and consumer’s risk in *R* as follows.

For the producer’s risks, write

```
p <- 0.04 #Good batches
1-phyper (1, 100*p, 100*(1-p), 10)
[1] 0.0487692 #Example 14.1
1-pbinom(2, 20, p)
[1] 0.04386279 #Example 14.2
1-ppois(3, 100*p)
[1] 0.5665299 #Example 14.3
```

For the consumer’s risks, write

```
p <- 0.10 #Bad batches
phyper (1, 100*p, 100*(1-p), 10)
[1] 0.7384715 #Example 14.1
```

```
pbinom(2, 20, p)
[1] 0.6769268 #Example 14.2
ppois(3, 100*p)
[1] 0.01033605 #Example 14.3
```

The risks are summarized in Table 14.1 for Examples 14.1–14.3

We see in Table 14.1 that there is a 4.9% chance of rejecting batches containing 4% defective (producer’s risk), while the consumer’s risk, that of accepting batches containing 10% defective, is a massive 73.8%. This scheme favors the producer.

Example 14.2 has similar risks. This is to be expected, since this scheme allows the same proportion defective in the sample as Example 14.1. The risks are a little lower, at 4.4% for the producer and 67.7% for the consumer, because of an increased sample size.

Example 14.3 is the opposite to the previous two examples in that it has a very high probability of rejecting good batches (56.7%), and a very low probability of accepting bad batches (1%). You will recall that this allows just 3 defectives in a sample of 100, that is, 3% of the sample items are allowed to be defective, while Examples 14.1 and 14.2 allow 10% of the sample to be defective before rejecting the batch. This is the reason why the scheme in Example 14.3 rejects so many good batches and accepts very few bad batches. This scheme favors the consumer.

In Examples 14.1 and 14.2, the consumer’s risk is too high, while, with Example 14.3, the producer’s risk is too high. Is it possible to reduce the consumer’s risk in Examples 14.1 and 14.2, while maybe allowing an increase in the producer’s risk? In Example 14.3, we might seek to reduce the producer’s risk, while allowing a little increase in the consumer’s risk. In the next section, we show how to design sampling schemes with prescribed risks.

14.7 DESIGN OF SAMPLING SCHEMES

The design of sampling inspection schemes involves a balancing act between the producer’s and the consumer’s risks. A sampling scheme consists of designating

- the sample size to be drawn from each batch (n)
- the allowable number of defectives in each sample (c)

TABLE 14.1 Producer’s and Consumer’s Risks

	Example 14.1 $n = 10, c = 1$	Example 14.2 $n = 20, c = 2$	Example 14.3 $n = 100, c = 3$
Producer’s risk Reject “good” batches ($p = 0.04$)	0.049	0.044	0.567
Consumer’s risk Accept “bad” batches ($p = 0.1$)	0.738	0.677	0.010

Often, n and c are chosen when the consumer's and producer's risks have been specified, and the objective is to get as near as possible to the ideal OC curve by choosing them appropriately.

For example, it may be desired to reject just 5% of batches containing 1% defectives (producer's risk) and to accept 8% of batches containing 10% defectives (consumer's risk).

We might seek to choose n and c so that

$$\text{Producer's risk} = P(X > c) = 1 - \frac{\sum_{x=0}^c \binom{Np}{x} \binom{N(1-p)}{n-x}}{\binom{N}{n}} = 0.05 \text{ when } p = 0.01$$

$$\text{Consumer's risk} = P(X \leq c) = \frac{\sum_{x=0}^c \binom{Np}{x} \binom{N(1-p)}{n-x}}{\binom{N}{n}} = 0.08 \text{ when } p = 0.1$$

This amounts to two simultaneous equations involving two unknowns n and c , which can be solved algebraically. We do not give the algebraic solution here; instead, we use R to arrive at the results iteratively.

14.7.1 Choosing c

Let us first look at the case where the sample size is specified and we need to know how many defectives should be allowed to occur in the sample. This situation arises when budget constraints determine the number of items to be inspected.

Example 14.4

In inspecting very large lots, it is intended to take a sample of 100 items from each batch. Acceptable batches are those with less than or equal to 3% defective, and batches are unacceptable if they contain 10% or more defectives. How many defectives should you allow per sample so that the producer's risk is approximately 8%, and the consumer's risk approximately 6%?

We look first at the producer's risk, for various values of c .

```
1-pbinom(4, 100, 0.03) # c = 4
[1] 0.1821452
1-pbinom(5, 100, 0.03) # c = 5
[1] 0.08083713
1-pbinom(6, 100, 0.03) # c = 6
[1] 0.03122751
```

We see that the nearest we can get to the specified producer's risk is 0.0808 with $c = 5$.

Next calculate the consumer's risk with $n = 100$, $c = 5$, and $p = 0.1$

```
pbinom(5, 100, 0.1)
0.05757689
```

The consumer's risk is 5.7%.

Note that, since n and c have to be integers, it is rarely possible to achieve the specified risks exactly.

Suggested Sampling Scheme:

Allow up to 5 defectives in a sample of $n = 100$ before rejecting the batch

▫

14.7.2 0/1 Inspection Schemes

Another approach used to design sampling schemes is to keep the allowable number of defectives fixed, and then to decide on the sample size necessary to comply with the risk constraints. The approach frequently used is to accept a batch only if the sample is free of defectives. As soon as one defective is found in the sample, the batch is rejected. Such a plan is called an 0/1 sampling inspection scheme: the allowable number of defectives in the sample $c = 0$. These 0/1 schemes have the great advantage of simplicity, and in particular appeal to the layperson, who may be unhappy when batches found to contain a defective are allowed through.

Example 14.5

It is desired to accept batches with 1% defectives with probability 0.95, and to accept batches with 20% defectives with probability 0.05. The batch is accepted only if the sample contains no defective items. What sample size is necessary?

First, we examine the acceptance probability of accepting good batches. We need to choose a sample size so that the probability will be close to the desired 0.95 .

```
pbinom(0, 10, 0.01) #n = 10
[1] 0.904382
pbinom(0, 8, 0.01) #n = 8
[1] 0.9227447
pbinom(0, 6, 0.01) #n = 6
[1] 0.9414801
```

So, with a sample of size $n = 6$, there is a 94% chance of accepting lots with 1% defective.

The probability of rejecting batches, those with 20% defective, is

```
pbinom(0, 6, 0.2) #Accepting "bad" batches: Consumer's risk
[1] 0.262144
```

With $n = 6$ and $c = 0$, we have a 26% chance of accepting batches with 20% defective, more than the desired 5%, but it is the best we could do for the stated producer's risk.

We could approach the problem by getting the consumer's risk right, and calculating the corresponding producer's risk.

```
pbinom(0, 13, 0.2) #Accepting bad batches: Consumer's risk
[1] 0.05497558
```

Taking samples of size 13, and rejecting batches for which samples have one or more defectives, results in a consumer's risk of just over 5%.

The producer's risk with $n = 13$ is

```
1- pbinom(0, 13, 0.01) # Rejecting good batches
[1] 0.1224790
```

just over 12%, more than double the desired 5%. \triangleleft

Example 14.5 illustrates that 0/1 sampling schemes have the disadvantage of a badly shaped operating characteristic curve, meaning that a good producer's risk is achieved at the expense of a poor consumer's risk, and vice versa.

A compromise could be obtained by allowing some flexibility in the risks. If, for example, we decided to increase the producer's risk to 10% say, then

```
1-pbinom(0, 10, 0.01) #Rejecting good batches; Producer's risk
0.09561792
pbinom(0, 10, 0.2) #Accepting bad batches; Consumer's risk
[1] 0.1073742
```

This means that a plan with a sample size of 10 has a producer's risk of 10% and a consumer's risk of 10% approximately.

Example 14.6

A producer inspects large batches and wants batches containing 1% defective to be rejected with a probability of 0.9. The consumer wants the probability of accepting batches containing 5% defective to be 0.60. Suppose a batch is accepted only if the sample contains no defective items. Find the sample size so that the sampling plan meets these requirements.

After some experimentation with R , we arrive at

```
pbinom(0, 10, 0.01)
[1] 0.904382
pbinom(0, 10, 0.05)
[1] 0.5987369
```

A sample size of 10 will meet these specifications pretty closely. \triangleleft

14.7.3 Deciding on n and c Simultaneously

Example 14.7

If good batches are those with less than or equal to 1% defective, and bad batches are those with 10% or more defectives, choose n and c so that the producer's risk is 5% and the consumer's risks is 8%.

We first check an 0/1 scheme.

```
1 - pbinom(0, 6, 0.01) #rejecting "good" batches
0.05851985
pbinom(0, 6, 0.1) # accepting "bad" batches
[1] 0.531441
```

which means that the producer's risk is approximately 6%, and the consumer's risk more than 53%! We clearly need to try another scheme. \triangleleft

Let us increase the sample size and the allowable number of defectives and observe the results in R . After some experimentation, we arrive at

```
1- pbinom(1, 40, 0.01) #Producer's Risk
[1] 0.06073662
pbinom(1, 40, 0.1) #Consumer's Risk
[1] 0.0804737
```

A sampling scheme which allows up to 1 defective in a sample of size 40, has a producer risk of 6%, and a consumer's risk of 8%.

One of the advantages of designing schemes experimentally is that you can see for yourself how the risks change as the sample size and/or the allowable number of defectives are varied.

14.8 RECTIFYING SAMPLING INSPECTION SCHEMES

When defectives are found in a sample, they are often replaced before being returned to the batch. When batches are rejected, they may be subjected to 100% inspection and all the defectives replaced. These represent what are known as *rectifying sampling schemes*.

In general, with rectifying schemes:

- accepted batches are accepted without change except that the defectives found in the sample are replaced;
- rejected batches are subjected to 100% inspection, and all the defectives are replaced.

With a rectifying scheme, the consumer will receive two types of batches:

- Those that have been accepted by the inspection scheme but may still contain some defective items;
- Those that have been rejected by the inspection scheme, subjected to 100% inspection, and all defectives have been replaced.

Obviously, rectifying schemes lead to an improvement in batch quality.

14.9 AVERAGE OUTGOING QUALITY

The *average outgoing quality* (AOQ) is the average proportion of defectives in batches received by the consumer.

For batches containing a proportion p defectives, $N - n$ items remain after the sample has been selected. These defective items are either

- inspected 100%, if the batch is rejected and all defective items are replaced or
- accepted without further inspection, if the batch is accepted so that some defective items remain.

Therefore, the number of defectives, X , remaining in the batch after inspection is either zero if the batch is rejected or $p(N - n)$ if the batch is accepted.

Writing P as the probability of accepting batches containing a proportion p defectives, we have

$$X = 0 \text{ with probability of rejection, } 1 - P$$

$$X = p(N - n) \text{ with probability of acceptance, } P$$

The average number of defectives items remaining in batches with proportion p defective is

$$E(X) = 0 \times (1 - P) + p(N - n) \times P = (N - n)pP$$

Therefore, batches which start out with a proportion p defective end up, on average, containing $(N - n)pP$ defectives.

Expressing this in terms of the number of items N in the batches, we have the average proportion defective in the batches, which is called the *average outgoing quality* (AOQ).

$$\text{AOQ} = (N - n)pP/N$$

When N is large relative to n , $N - n \approx N$, giving

$$\text{AOQ} \approx pP.$$

Let us return to Example 14.3 where the sampling plan is to allow up to 3 defectives in a sample of size 100 drawn from batches of size 10,000 and examine the average outgoing quality for some values of p .

TABLE 14.2 Average Outgoing Quality

p	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07
$AOQ(pP)$	0	0.0098	0.0171	0.0194	0.0172	0.033	0.0091	0.0057

In R, write

```
p <- seq(0, 1, 0.01)
AOQ <- p*ppois(3, 100*p)
round(AOQ, 4) # rounds to 4 decimal places.
```

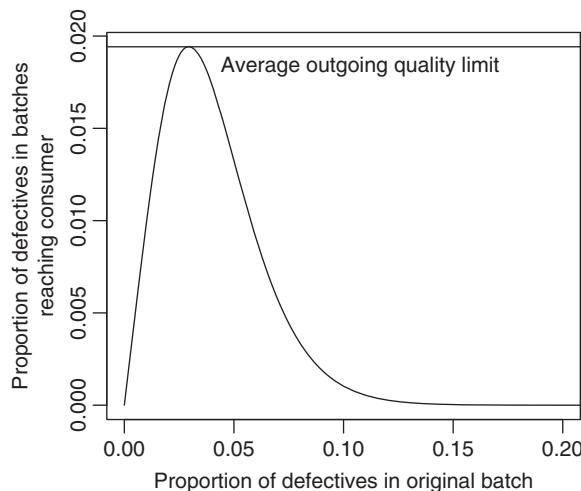
Table 14.2 gives some of the output.

A graphical illustration is obtained in R with

```
max(AOQ)
[1] 0.01941908
plot(p, AOQ, type = "l", ylim = c(0, max(AOQ)), xlim = c(0, 0.2),
xlab= "Proportion of defectives in original batch",
ylab = "Proportion of defectives in batches reaching consumer")
abline(h = max(AOQ))
text(0.03, 0.0194, "Average outgoing quality limit")
```

and presented in Fig. 14.6.

In Fig. 14.6, we see that the maximum average outgoing quality is 0.0194, and this occurs when the proportion of defectives in the batch is $p = 0.03$. The worst

**FIGURE 14.6** Average Outgoing Quality

quality batches, those with the highest average defective rate, are those which originally contain 3% defective. This is called the average outgoing quality limit AOQL. Batches with defective rates higher than that are more likely to be rejected and subjected to a complete inspection so that all defective items are replaced.

Obviously, rectifying sampling inspection improve the quality level. The amount of improvement depends on the proportion of defectives, and the consequent probability of acceptance. The maximum improvement in quality occurs when the batches are rejected; in this case, all the defectives in the batch are replaced.

14.10 DOUBLE SAMPLING INSPECTION SCHEMES

Example 14.8

A manufacturer of microchips packs them in batches of 10,000. Fifty chips are chosen randomly from each batch and subjected to a quality test.

Stage 1: Take a sample of 50 chips.

1. If no defective occurs, the batch is accepted.
2. If 1, 2, or 3 defectives occur, a further 50 items are selected for inspection.
3. If 4 or more defectives occur, the batch is rejected.

Stage 2: When 1, 2, or 3 defectives occur in the first sample, select a further 50 chips.

1. If 4 or more defectives occur in the combined sample, the batch is rejected.
2. If less than 4 defectives are found, the batch is accepted.

This is an example of schemes that are called *double sampling inspection schemes* where, after the initial sample inspection, a second sample may be taken before a final decision to reject or accept the batch is made. The first sample can lead to one of three decisions.

1. Accept the batch.
2. Take a second sample.
3. Reject the batch.

When a second sample is taken, the decision to reject or accept the batch depends on the total number of defectives in the combined sample.

Let

E = event of accepting the batch

A = event of accepting the batch at the first stage

B = event of accepting the batch at the second stage

TABLE 14.3 Acceptance Probabilities

Defective rate (p)	0	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16
Acceptance probability (P)	1	0.865	0.448	0.159	0.046	0.012	0.003	0.001	0.000

The acceptance probability is the probability that either the batch is accepted at the first stage, or the inspection process goes to the second stage and is then accepted.

$$P(E) = P(A \cup B) = P(A) + P(B)$$

$$P(A) = P(0 \text{ defectives})$$

$$\begin{aligned} P(B) &= P(1 \text{ defective in 1st sample})P(\leq 2 \text{ defectives in 2nd sample}) \\ &\quad + P(2 \text{ defectives in 1st sample})P(\leq 1 \text{ defectives in 2nd sample}) \\ &\quad + P(3 \text{ defectives in 1st sample})P(0 \text{ defectives in 2nd sample}) \end{aligned}$$

We use R to calculate the acceptance probability, $P(A) + P(B)$, for various values of p , and to plot the operating characteristic curve.

```
p <- seq(0, 1, 0.02)
PA <- pbinom(0, 50, p) # acceptance at first stage
PB <- dbinom(1, 50, p)*pbinom(2, 50, p) +
      dbinom(2, 50, p)*pbinom(1, 50, p) +
      dbinom(3, 50,p)*pbinom(0, 50, p) #acceptance at second stage
P <- PA + PB
```

Table 14.3 gives the first few acceptance probabilities.

The operating characteristic curve is obtained with

```
plot(p, P, ylab = "Acceptance probability",
     xlab = "Proportion defective",
     xlim = c(0, 0.2), type = "l")
```

which gives Fig. 14.7. △

14.11 AVERAGE SAMPLE SIZE

With a double scheme, the sample size is variable. In the Example 14.8, the sample is either 50 or 100. It is 50 if accepted or rejected at the first stage, that is, when the number of defectives in the batch is 0 or more than 3. It is 100 if it is necessary to go to the second stage, that is, when the sample is found to contain 1, 2, or 3 defectives.

$$\begin{aligned} n &= 50 \text{ with } X = 0 \text{ or } X > 3 \\ &= 100 \text{ with } X = 1, 2, \text{ or } 3 \end{aligned}$$

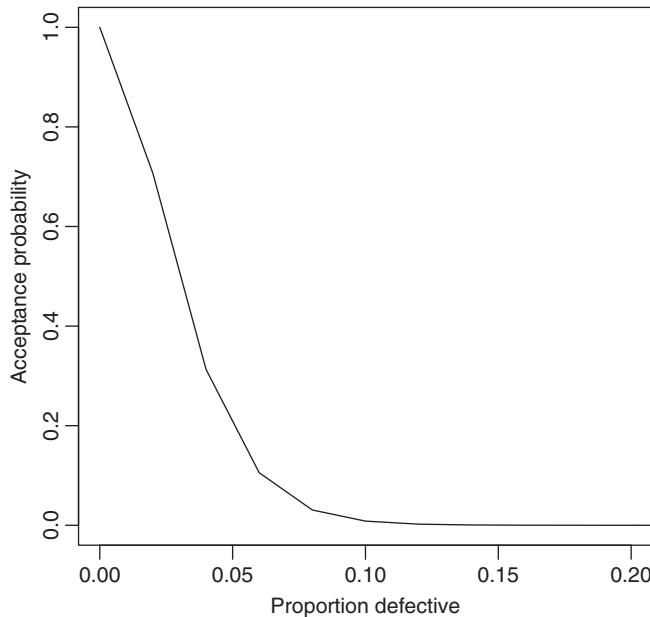


FIGURE 14.7 Operating Characteristic Curve

The expected sample size is

$$50 + 50P(X = 1 \text{ or } 2 \text{ or } 3)$$

We can calculate the expected sample size in R using

```
n <- 50 + 50 * (dbinom(1, 50, p) + dbinom(2, 50, p) + dbinom(3, 50, p))
```

Table 14.4 gives the values average sample size, \bar{n} , for some values of p .

A plot of the average sample size is given in Fig. 14.8.

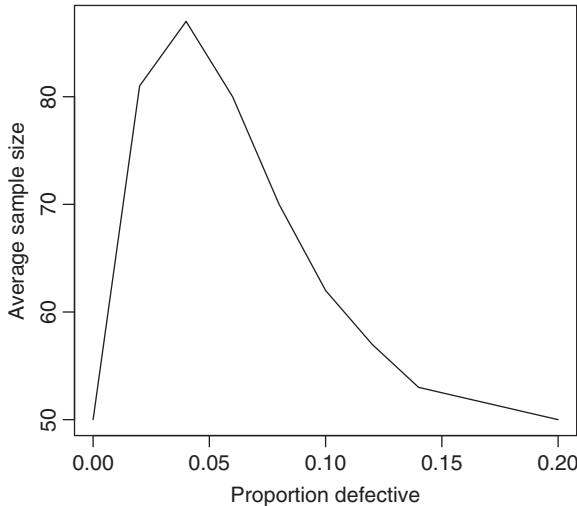
The maximum average sample size is 87 and occurs with batches containing 4% defective. This defective rate is most likely to necessitate going to the second stage.

14.12 SINGLE VERSUS DOUBLE SCHEMES

Let us compare the single sample scheme outlined in Example 14.3 and the double sample scheme outlined in Example 14.8.

TABLE 14.4 Average Sample Size

p	0.00	0.02	0.04	0.06	0.08	0.10	0.12	0.14	0.16	0.18	0.20
\bar{n}	50	81	87	80	70	62	57	53	52	51	50

**FIGURE 14.8** Average Sample Size

In both of these examples, the batches are rejected if a sample of 100 yields 4 or more defectives. Example 14.3 inspects the 100 items together, while Example 14.8 takes an initial sample of 50 and goes to the second stage if the initial sample in the first sample contains 1, 2, or 3 defectives.

With Example 14.3, a sample of 100 is taken from each batch while with Example 14.8, samples of size 50 are taken from some batches, and samples of size 100 are taken from others. The sample size differs from batch to batch, but we were able to calculate the average sample size required from batches with differing proportions defective and found that the greatest average sample size required was 87, and this occurred in batches containing 4% defectives. The double sample schemes involves a minimum reduction in sample size of 13%.

For the single sampling scheme in Example 14.3, the average outgoing quality is calculated in R

```
p <- seq(0, 0.07, 0.01)
P <- ppois(3, 100*p)
AOQ1<- P*p
```

The AOQ for the double sampling schemes in Example 14.8 is calculated as follows:

```
PA <- pbinom(0, 50, p) # acceptance at first stage
PB <- dbinom(1, 50, p)*pbinom(2, 50, p) +
      dbinom(2, 50, p)*pbinom(1, 50, p) +
      dbinom(3, 50,p)*pbinom(0, 50, p) # acceptance at second stage
P2 <- PA + PB
AOQ2 <- P2*p
```

Table 14.5 gives some of the output.

TABLE 14.5 Average Outgoing Quality

<i>p</i>	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07
Single sampling	0	0.0098	0.0171	0.0194	0.0173	0.0136	0.0091	0.0057
Double sampling	0	0.0098	0.0173	0.0198	0.0179	0.0138	0.0095	0.0061

We see from Table 14.5 that the average outgoing quality is just slightly better with single sampling schemes, which is not surprising in view of the greater sampling intensity.

This example illustrates that, while double sampling schemes are not as easy to implement as their single counterparts, they have more flexibility than a single sampling plan. Double sampling schemes enable very good and very bad batches to be detected with smaller sample sizes and doubtful batches to be subjected to further investigation. A double sampling scheme is economical in terms of the average number of items inspected, a fact which may be of paramount importance when the test is destructive or expensive or time consuming. Thus, double sampling enables sampling costs to be reduced, as well as providing the psychological advantage of giving a batch a second chance.

EXERCISES 14.1

Use *R* to work the following examples.

1. A manufacturer of microchips packs them in batches of 1,000 and randomly selects 10 chips from each batch for inspection. If more than two chips are found to be defective, the complete batch is rejected.
 - (a) Calculate the probabilities of rejecting batches for various proportions defective.
 - (b) Draw the operating characteristic curve.
 - (c) Determine the producer's and consumer's risks if batches containing 5% defective should be accepted, and batches containing 10% defective should be rejected.
2. Random access memory chips are packed in batches of 1,000. A sample of size 15 is randomly selected from each batch and subjected to tests of reliability. The batch is accepted if the sample contains no more than one defective, otherwise it is rejected.
 - (a) Calculate the probabilities of accepting batches with various proportions defective and draw the operating characteristic curve.
 - (b) Calculate the producer's risk if batches containing up to 5% defective are acceptable.
3. Consider a plant manufacturing chips of which 10% are expected to be defective. The chips are packed 30 to a box for distribution. A sample of size 10 is

drawn without replacement from each box. If more than one defective is found in the sample, the box is rejected and subjected to a complete inspection. What is the probability that a box will be rejected?

4. A single sampling plan allows up to and including 2 defectives in a sample of size 20, drawn from large batches.
 - (a) Draw its operating characteristic curve.
 - (b) What batches have more than 95% chance of acceptance?
 - (c) What batches have less than a 10% chance of acceptance?
5. In inspecting batches of 1,000 items, it is desired to accept batches containing 1% defective item with probability 0.90, and it is desired to reject batches containing 20% defective items with probability 0.90. Suppose the batch is accepted only if the sample is free of defective items. What sample size is necessary?
6. If unacceptable batches are rectified in Examples 14.1 and 14.2, draw the average outgoing quality curve (AOQ), and obtain the average outgoing quality limit (AOQL).
7. Batches of 1,000 memory sticks are subjected to the following sampling inspection scheme: a sample of 10 is selected, and the batch is accepted if this sample contains no unacceptable items; if this sample contains 3 or more unacceptable items, the batch is rejected. If the sample contains either 1 or 2 unacceptable items, a second sample of 20 is drawn; the batch is then accepted if the total number of unacceptable items in the combined samples is at most 3. Suppose that, at any stage, an unacceptable item is replaced by a good item.
 - (a) Find the probability that a batch containing 15 unacceptable items is accepted.
 - (b) Find the AOQL for this double sampling plan, if unacceptable batches are rectified.
8. The IC manufacturer has a quality control unit that sample tests completed devices. The manufacturer must guarantee that the device will meet its specifications. The devices are packed in batches of 5,000, from which a random sample of 100 is chosen at random for inspection. If a bad device is detected, the batch is rejected, and all the devices in the batch are verified by the production system, and replaced.

What guarantee can the manufacturer give regarding batches that contain 1% defective?
9. A batch of 200 items, which contains 15 defective items, is subjected to an inspection scheme that calls for a sample of size 6 to be drawn. The sample is accepted if the sample contains at most one defective item.
 - (a) Find the probability that the batch is accepted.
 - (b) If defective items in the sample are replaced by good items, find the average outgoing quality.

10. A day's production of 2,000 memory sticks is inspected as follows. If an initial sample of 15 shows at most two defective sticks, the day's production is accepted, and subject to no further sampling. If, however, the first sample shows 3 or more defective sticks, then a second sample of 20 sticks is chosen, and the day's production is accepted if the total number of defectives in the two samples is no more than 4.
 - (a) Find the probability that the day's production is accepted if it contains 10 defective sticks.
 - (b) Find the average outgoing quality.
 - (c) Graph the operating characteristic curve.

14.13 PROJECTS

1. Develop a system in *R*, which designs a sampling scheme based on the producer's and consumer's risks. Your program should allow the risks to be input, and iteratively to get the best sample size n and c to arrive, as near as possible, at the given risks. It should also allow the user to specify the sample size if desired, or instead to specify the allowable number of defectives, as is done in 0/1 schemes.
2. In Fig. 14.5, we used line type `lty = 1` to draw a continuous line between the designated points. By varying the number assigned to `lty`, explore the possibilities of getting different types of lines.

PART IV

CONTINUOUS DISTRIBUTIONS

15

INTRODUCTION TO CONTINUOUS DISTRIBUTIONS

So far, the random variables that we have considered have been discrete. They resulted from situations where we counted, for example, the number of errors in a program, the number of defectives in a batch, or the number of job arrivals in a day. In many practical situations, our experiment will consist of measuring rather than counting, and the results will be continuous rather than discrete.

Consider, for example, the experiment of submitting a program and measuring the time, X , that it takes to be executed. Clearly, X is a time interval. We may know from past experience that the response time will never be greater than five seconds, so it would be natural to take the sample space S to be the interval from 0 to 5, that is

$$S = [0, 5] = \{x \in \mathbb{R} | 0 \leq x \leq 5\}$$

In this case, although S is a “finite interval,” the number of actual points in S is infinite and not even countably infinite. This random variable is said to be continuous.

While the features of continuous variables are similar in many ways to those of discrete variables, they are treated somewhat differently.

15.1 INTRODUCTION TO CONTINUOUS RANDOM VARIABLES

Definition 15.1 *Continuous random variables*

If a random variable X takes values in a finite or infinite interval, then X is said to be a *continuous* random variable. \square

We look again now at some examples of continuous measurements which were given in Chapter 4.

Example 4.9 measured the “response time,” X , of an submitted inquiry. Here, $0 < X < \infty$.

Example 4.10 measured the “CPU time” of a submitted program, the amount of time, X , the program takes to be processed in the central processing unit. Here, $0 < X < \infty$.

Example 4.11 browsed the web requesting a particular item of information, and measured how long, X , it takes to respond. Again $0 < X < \infty$.

15.2 PROBABILITY DENSITY FUNCTION

A probability density function $f(x)$ of a continuous variable X is a real-valued function defined on the range of X such that:

1. the total area under the curve of $f(x)$ is 1;
2. the area under the curve of $f(x)$ between two points a and b is the probability that the random variable X lies between a and b ;
3. $f(x)$ is positive or zero. It is zero in any range where the random variable never falls.

Such $f(x)$ is called the *pdf* of X . Equivalently and more concisely, $f(x)$ has the following properties:

1. $\int_{-\infty}^{\infty} f(x)dx = 1$;
2. $\int_a^b f(x)dx = P(a \leq X \leq b)$;
3. $f(x) \geq 0$.

This is similar to the pdf of the discrete variable but there are some important differences:

- It is not meaningful to consider the probability that X assumes a particular value. For a continuous variable, $P(X = k) = 0$ for each and every k . There is no such thing as point probability with continuous variables. Instead, we have the probability *density*, which is the probability of hitting a small region around k divided by the size of the region.

- In many of the problems associated with discrete random variables, the pdf can be derived from the assumptions and physical description of the data. For example, examining products until the first defective leads to a geometric distribution. With continuous variables, the form of the distribution is not usually derived from some physical model. While it may sometimes be based on past observations of similar data, at other times, it is simply assumed to be of a certain form.

Example 15.1

In an electronic engineering laboratory, current in a circuit is measured using an ammeter. Due to several random factors, the measurement X varies. Past records indicate that current varies along the interval $[2, 6]$ but not uniformly. Higher values of X have higher probabilities of occurrence. It has been found that a good model for the data is represented by the following pdf:

$$\begin{aligned} f(x) &= 0.025x + 0.15, \quad 2 < x < 6 \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

In R, this pdf can be plotted using the function `curve`.

```
curve(0.025*x + 0.15, 2, 6, xlim = c(0, 8))
```

This generates Fig. 15.1.

We see from Fig. 15.1 that the pdf increases with X . Let us check that the integral under the curve over the entire range of X is one, that is,

$$\int_2^6 (0.025x + 0.15)dx = 1$$

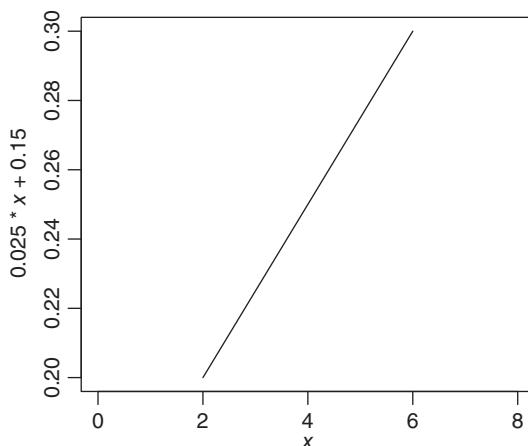


FIGURE 15.1 Probability Density Function of Current in a Circuit

$$\begin{aligned}
 \int_2^6 (0.025x + 0.15)dx &= \left[0.025 \frac{x^2}{2} + 0.15x \right]_2^6 \\
 &= (0.025 * 36/2 + 0.15 * 6) - (0.025 * 4/2 + 0.15 * 2) \\
 &= 1.35 - 0.35 \\
 &= 1
 \end{aligned}$$

This can be done in *R* by first defining the function

```
integrand <- function(x) (0.025*x + 0.15)
```

and then integrating that function in the range of *X*:

```
integrate(integrand, 2, 6)
```

which gives the output

```
1 with absolute error < 1.1e-14
```

Notice that the value of the integral is given with an absolute error. The reason for this is that *R* performs the integration numerically, and so the integral may not be exact. In this case, *R* is saying that the error is less than 1.1×10^{-14} , which, you will agree, is negligible.

Suppose we want the probability that *X* lies between 2 and 3.

$$\begin{aligned}
 P(2 < X < 3) &= \int_2^3 (0.025x + 0.15)dx \\
 &= \left[\frac{0.025x^2}{2} + 0.15x \right]_2^3 \\
 &= (0.025 * 9/2 + 0.15 * 3) - (0.025 * 4/2 + 0.15 * 2) \\
 &= 0.5625 - 0.35 \\
 &= 0.2125
 \end{aligned}$$

The probability can be obtained quickly in *R*, with

```
integrate(integrand, 2, 3)
```

which gives

```
0.2125 with absolute error < 2.4e-15
```



15.3 CUMULATIVE DISTRIBUTION FUNCTION

As in the discrete case, the cumulative distribution function (cdf) is defined as the probability that the random variable X is less than or equal to some specified x , $P(X \leq x)$.

Definition 15.2 *Cumulative distribution function*

The cdf of a continuous variable X is usually denoted by $F(x)$ and defined by

$$F(x) = P(X \leq x) = \int_{-\infty}^x f(t)dt$$

□

Example 15.2

Returning to Example 15.1, let us find the probability that the measurement of the current is less than or equal to 4.

Graphically with R

```
curve(0.025*x + 0.15, 2, 6, xlim = c(0,8))
x <- seq(2, 4, 0.01)
lines (x, 0.025*x + 0.15, type = "h", col = "grey")
```

gives Fig. 15.2.

In R, we invoke the previously defined function `integrand`, and integrate it in the relevant interval.

```
integrate(integrand, 2, 4)
0.45 with absolute error < 5e-15
```

Notice again that the value of the integral is given with an absolute error. ▲

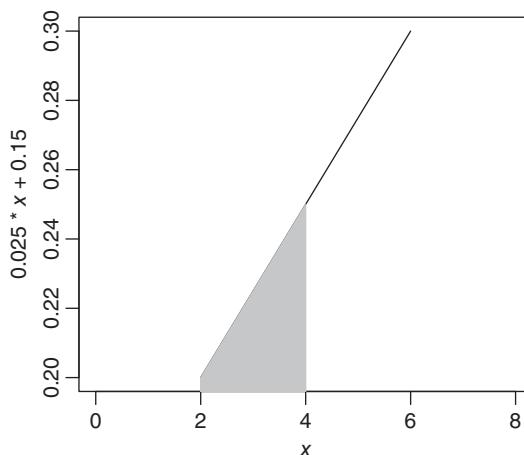


FIGURE 15.2 $P(X \leq 4)$: Area Under the Curve

You will remember that the area of a line is zero,

$$P(X = x) = 0$$

that is, the point probability is zero, and hence, for continuous variables,

$$P(X < x) = P(X \leq x).$$

15.4 THE UNIFORM DISTRIBUTION

A continuous random variable that appears to have equally likely outcomes along its range of values is said to have a uniform distribution. Generally, the uniform distribution describes the occurrence of a variable, X , uniformly along a finite interval $[a, b]$.

Definition 15.3 *Continuous uniform distribution*

The pdf of a uniform distribution is given by

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases}$$

□

The parameters are a and b , the interval end points. Clearly, the area under the curve is equal to one. There is no need to integrate since it is a rectangle with width of the base $b - a$ and height $1/(b - a)$, giving the area under the general uniform curve $(b - a) \times 1/(b - a) = 1$.

In R, the shortened name *unif* is prefixed by *d* for the pdf, and *p* for the cdf of the uniform distribution. For example, to plot the pdf and cdf of a random variable distributed uniformly in the interval [2, 5], write

```
curve(dunif(x, 2, 5), xlim = c(0, 7))
curve(punif(x, 2, 5), xlim = c(0, 7))
```

which gives Fig. 15.3

Example 15.3

It was reported in the International Journal of Circuit Theory and Applications, May-June 1990, that a switched-capacitor circuit for generating random signals had been designed, built, and tested by researchers at the University of California, Berkeley. The circuit's trajectory was shown to be uniformly distributed in the interval (0,1].

The pdf and cdf are obtained using *dunif* and *punif*, respectively.

```
curve(dunif(x, 0, 1), xlim = c(0, 2))
curve(punif(x, 0, 1), xlim = c(0, 2))
```

which gives Fig. 15.4

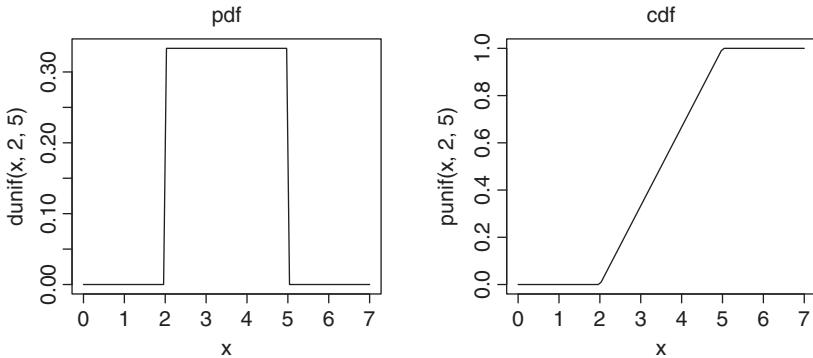


FIGURE 15.3 The pdf and cdf of the Uniform [2, 5] Random Variable

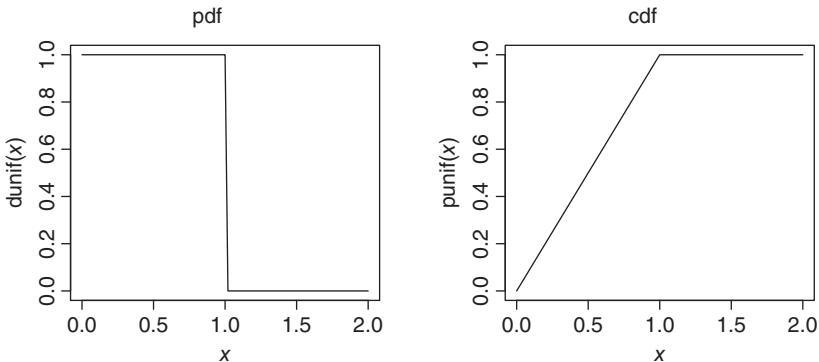


FIGURE 15.4 The Uniform [0,1] pdf and cdf

Suppose we want the probability that the trajectory falls between 0.2 and 0.5 as indicated in Fig. 15.5.

$P(0.2 < X < 0.5)$ is the area of the rectangle with base width $0.5 - 0.2 = 0.3$, and height 1. So, the probability is got by the simple calculation

$$P(0.2 < X < 0.5) = (0.5 - 0.2) \times 1 = 0.3$$

Figure 15.5 is generated in R with the following code:

```
curve(dunif(x, 0, 1))
x <- seq(0.2, 0.5, 0.001)
lines(x, dunif(x), type = "h", col = "grey")
text(0.35, 1.05, "P(0.2 < X < 0.5)")
```

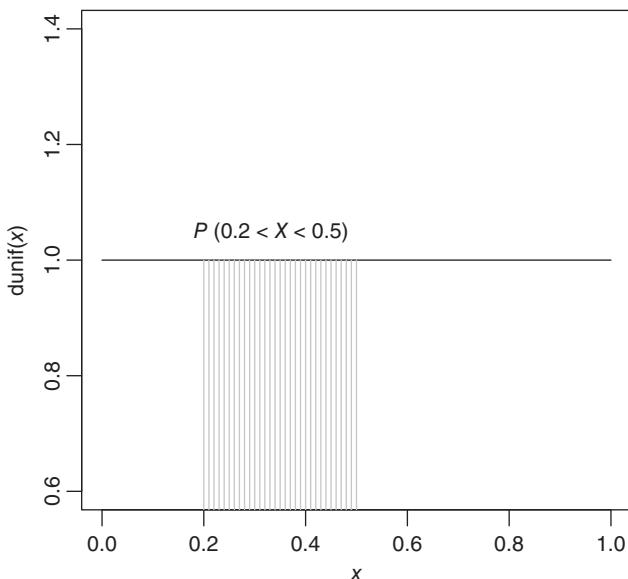


FIGURE 15.5 Uniform Probabilities: $P(0.2 < X < 0.5)$

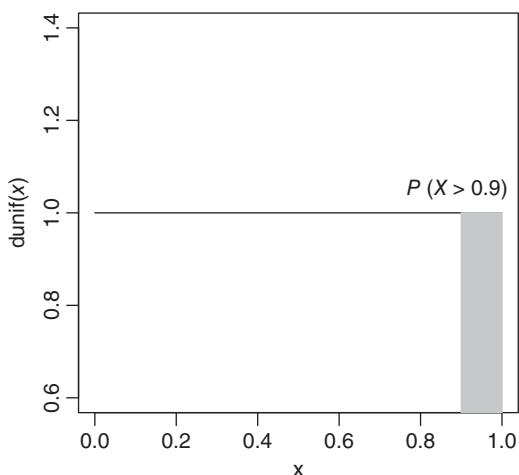


FIGURE 15.6 Uniform Probabilities: $P(X > 0.9)$

The cumulative probability, $P(X \leq x)$, is obtained in *R* using `punif`. For example,

$P(X \leq 0.5)$ is calculated using `punif(0.5, 0, 1)`

$P(X \leq 0.2)$ is calculated using `punif(0.2, 0, 1)`

So, $P(0.2 < X < 0.5)$ is calculated by subtraction.

```
punif(0.5, 0, 1) - punif(0.2, 0, 1)
```

which gives

```
[1] 0.3
```

To calculate the probability that a trajectory is observed and exceeds 0.9, $P(X > 0.9)$ as shown in Figure 15.6, take the area of the rectangle $0.1 \times 1 = 0.1$, or in *R* with

```
1 - punif(0.9, 0, 1)
[1] 0.1
```

Figure 15.6 was generated with the following code.

```
curve(dunif(x, 0, 1))
x <- seq(0.9, 1.0, 0.001)
lines(x, dunif(x), type = "h", col="grey")
text(0.90, 1.05, "P(X > 0.9)")
```

Suppose we want the maximum value of the trajectory, for which there is a 95% chance that it will not be exceeded.

We want k so that

$$P(X \leq k) \geq 0.95.$$

This can be obtained in *R* using the quantile function `qunif`.

```
qunif(0.95, 0, 1)
[1] 0.95
```

△

Clearly, these probabilities are easy to calculate manually. We have used *R* here to illustrate its facilities. In later chapters, we will see how useful *R* is for obtaining probabilities with more complicated pdfs.

15.5 EXPECTATION OF A CONTINUOUS RANDOM VARIABLE

Recall, when X is a discrete random variable, the mean is

$$\mu = E(X) = \sum_x xp(x)$$

and the variance is

$$\sigma^2 = E(X - \mu)^2 = \sum_x (x - \mu)^2 p(x)$$

Analogous to the discrete case, when X is continuous, the mean and variance are defined as

$$\mu = E(X) = \int_{-\infty}^{\infty} xf(x)dx \quad (15.1)$$

and

$$\sigma^2 = E(X - \mu)^2 = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx \quad (15.2)$$

respectively. For this to make sense, of course the integrals have to “converge.”

Returning to Example 15.1 where $f(x) = 0.025x + 0.15$, let us obtain $E(X)$ using R. First, define as a function, the argument of the mean integral function given in Equation 15.1.

```
xfx <- function(x) (x * (0.025*x + 0.15))
```

and integrate it over the range of x

```
integrate(xfx, 2, 6)
```

to get

```
4.133333 with absolute error < 4.6e-14
```

Similarly, for the variance given in Equation 15.2, define as a function the argument of the variance integral function.

```
deviations <- function(x) ((x - 4.13333)^2 * (0.025*x + 0.15))
```

Integrate

```
integrate(deviations, 2, 6)
```

to get

```
1.315556 with absolute error < 1.5e-14
```

15.5.1 Mean and Variance of a Uniform Distribution

If X is uniformly distributed in $[a, b]$, the mean is

$$\begin{aligned} E(X) &= \frac{1}{b-a} \int_{x=a}^b x dx \\ &= \frac{1}{b-a} \left(\frac{b^2 - a^2}{2} \right) \\ &= \frac{b+a}{2} \end{aligned}$$

By similar algebraic manipulations, it can be shown that the variance

$$\begin{aligned} E(X - \mu)^2 &= \frac{1}{b-a} \int_{x=a}^b (x - \frac{b+a}{2})^2 dx \\ &= \frac{(b-a)^2}{12} \end{aligned}$$

The details are given in Appendix C.

In Example 15.3 which has a uniform distribution in $(0, 1]$, the mean of the circuit's trajectory is $1/2$, and the variance is $1/12 = 0.0833$.

15.5.2 Properties of Expectations

More generally, the expectation of any function $g(X)$ of a continuous random variable X is

$$E(g(X)) = \int_x g(x)f(x)dx$$

where $f(x)$ is the pdf.

It is easy to deduce the following properties, for any constant c .

$$E(X + c) = E(X) + c.$$

$$\begin{aligned} E(X + c) &= \int_x (x + c)f(x)dx \\ &= \int_x xf(x)dx + c \int_x f(x)dx \\ &= E(X) + c \end{aligned}$$

since $\int_x f(x)dx = 1$ and $\int_x xf(x)dx = E(X)$

$$E(cX) = cE(X)$$

$$\begin{aligned} E(cX) &= \int_x cx f(x)dx \\ &= c \int_x xf(x)dx \\ &= cE(X) \end{aligned}$$

$$V(X + c) = V(X)$$

$$\begin{aligned} V(X + c) &= E(X + c - E(X + c))^2 \\ &= E(X - E(X))^2 \\ &= V(X) \end{aligned}$$

$$V(cX) = c^2 V(X)$$

$$\begin{aligned} V(cX) &= E(cX - E(cX))^2 \\ &= E(c(X - E(X)))^2 \\ &= c^2 E(X - E(X))^2 \\ &= c^2 V(X) \end{aligned}$$

You will notice that these are the same results that we obtained for the discrete random variable in Chapter 9.

15.6 SIMULATING CONTINUOUS VARIABLES

Let us look again at the current in a circuit introduced in Example 15.1. To generate an empirical distribution write

```
x <- seq(2, 6, 0.001) #creates a vector of
  equidistant values between 2 and 6
fx <- 0.025*x + 0.15 #evaluates f(x) for each x
```

Plotting the values of x and fx gives the empirical pdf.

```
plot(x, fx, ylab = "f(x) = 0.025 x + 0.15", xlim = c(0, 8))
```

gives Fig. 15.7.

You will agree that this empirical distribution is similar to the actual, given in Fig. 15.1.

We can take a sample from this population as follows:

```
samplevalues <- sample(x, 50, replace = T, prob = fx)
```

selects 50 values between 2 and 6 with the density function given in Example 15.1. To see what they are, write

```
samplevalues
[1] 3.345 2.688 3.941 4.243 5.402 5.215 5.275 2.648 4.754 2.599
[11] 3.947 4.012 4.264 5.707 5.331 5.723 5.458 2.028 3.042 4.858
[21] 2.333 2.472 2.394 3.441 5.456 5.695 3.887 2.476 5.493 2.176
[31] 5.460 5.944 4.149 4.923 5.275 3.105 2.479 2.279 5.592 2.214
[41] 5.031 5.138 3.877 3.549 3.929 4.982 2.286 5.000 2.770 4.368
```

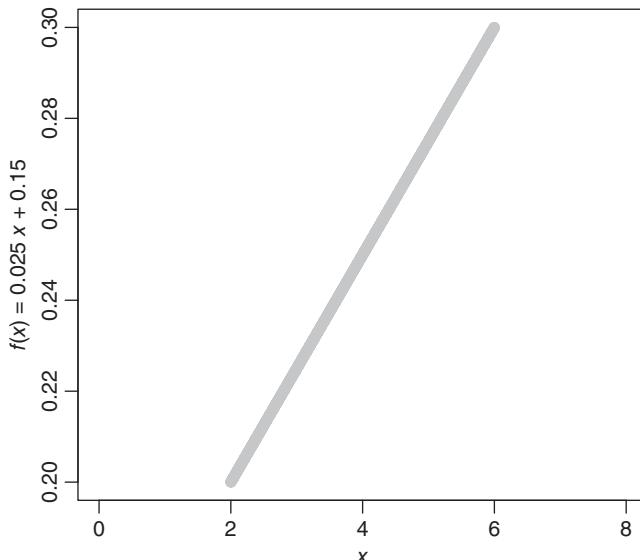


FIGURE 15.7 Empirical Probability Density Function

To obtain the empirical mean, write

```
mean(samplevalues)
[1] 4.05306
```

and variance

```
var(samplevalues)
[1] 1.574018
```

These values are not too far away from the exact mean of 4.13 and variance 1.315 derived in Section 15.5. Obviously, these estimates would be closer to the true values if we increased the number of simulations. Do this yourself.

15.6.1 Simulating the Uniform Distribution

For generating random sample values from the uniform distribution, the function `runif` is used. To simulate 100 trajectories in Example 15.3 which has a uniform distribution in $(0, 1]$, write

```
traject <- runif(100, 0, 1)
```

which generates 100 random numbers from a uniform distribution with parameters 0 and 1.

The empirical mean is

```
mean(trajект)
[1] 0.5040182
```

and the variance is

```
var(trajект)
[1] 0.08500762
```

which are not too far away from the true values of $1/2$ and $1/12$ obtained previously.

EXERCISES 15.1

Use R to solve the following.

1. A machine produces memory sticks of varying lengths, distributed uniformly between 2 and 12 mm. Memory sticks longer than 10 mm do not meet the design criterion, and must be scrapped.
 - (a) Calculate the probability that a memory stick will be scrapped.
 - (b) Simulate 50 memory stick lengths and obtain a histogram of the simulated values. Calculate the simulated mean and variance.
2. Students in an electronic engineering laboratory measure current in using an ammeter. Due to several random factors, the measurement X follows the pdf

$$f(x) = 0.025x + b, \quad 2 < x < 6$$

- (a) Show that, for this to be a genuine probability density, $b = 0.15$.
 - (b) Find the probability that the measurement of the current exceeds 3 A.
 - (c) Find $E(X)$.
3. In a radio communications system, the phase difference X between the transmitter and receiver is modeled as having a uniform density in $[-\pi, +\pi]$. Find $P(X \leq 0)$ and $P(X \leq \pi/2)$.

16

THE EXPONENTIAL DISTRIBUTION

16.1 MODELING WAITING TIMES

Example 16.1

You have observed that the number of hits to your website follow a Poisson distribution at a rate of 2 per day on average.

Let T be the time (in days) between hits.

$$P(T < 1) = P(\text{the waiting time between hits is less than one day})$$

$$P(T > 2) = P(\text{two days will elapse without a hit})$$

$$P(1 \leq T \leq 2) = P(\text{the time between hits is between one and two days})$$

▫

Example 16.2

You observe the number of telephone calls that arrive each day on your phone over a period of a year, and note that the arrivals follow a Poisson distribution with an average of three per day.

Let T be the waiting time between calls.

$$P(T < 0.5) = P(\text{you will be waiting less than half a day for a call})$$

$$P(T > 1) = P(\text{you will be waiting more than a day for a call})$$

$$P(0.5 \leq T < 1) = P(\text{you will wait at least half a day and less than one day for a call})$$

▫

Example 16.3

Records show that job submissions to a computer center have a Poisson distribution with an average of four per minute.

Let T be the time in minutes between submissions.

$$P(T < 0.25) = P(\text{the time between submissions is less than 0.25 minute (15 seconds)})$$

$$P(T > 0.5) = P(\text{the time between submissions is greater than 0.5 minute (30 seconds)})$$

$$P(0.25 \leq T \leq 1) = P(\text{the time between submissions is between 0.25 and 1 minute})$$

△

Example 16.4

Records indicate that messages arrive to a computer server following a Poisson distribution at the rate of 6 per hour.

Let T be the time in hours that elapses between messages.

$$P(T < 1/6) = P(\text{less than 10 minutes will elapse between arrivals})$$

$$P(T > 1/2) = P(\text{more than half an hour will elapse between arrivals})$$

$$P(1/6 \leq T \leq 1/2) = P(\text{the time between arrivals is between 10 and 30 minutes})$$

△

Notice that these examples are the same as Examples 13.3–13.6 that we used in Chapter 13 when introducing the Poisson distribution. Now, we are measuring the elapsed time between the arrivals for various arrival rates λ .

$$\lambda = 2 \text{ per day in Example 16.1}$$

$$\lambda = 3 \text{ per day in Example 16.2}$$

$$\lambda = 4 \text{ per minute in Example 16.3}$$

$$\lambda = 6 \text{ per hour in Example 16.4}$$

16.2 PROBABILITY DENSITY FUNCTION OF WAITING TIMES

Let T be the time that elapses after a Poisson event.

If λ is the occurrence rate in a unit time interval, then λt is the occurrence rate in a time interval of length t .

Now, the probability that no Poisson event occurred in the time interval $[0, t]$ is given by

$$P(0, t) = \frac{e^{-\lambda t} (\lambda t)^0}{0!} = e^{-\lambda t}$$

Obviously, “no Poisson event occurs in the interval $[0, t]$ ” means that the waiting time for the next occurrence is greater than t .

$$P(T > t) = P(0, t) = e^{-\lambda t}.$$

Hence, the *cumulative distribution function* (cdf) of the waiting time is

$$F(t) = P(T \leq t) = 1 - P(T > t) = 1 - e^{-\lambda t} \quad (16.1)$$

and, working backwards, we have the *probability density function (pdf)* of the waiting time

$$\begin{aligned} f(t) &= F'(t) \\ &= \frac{d(1 - e^{-\lambda t})}{dt} \\ &= \lambda e^{-\lambda t} \end{aligned}$$

This is called the *exponential distribution* and, as we have just shown, measures the waiting time between Poisson occurrences.

Definition 16.1 *The exponential random variable*

A continuous random variable T is said to have an exponential distribution with parameter $\lambda > 0$ if the probability density function $f(t)$ takes the form

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

□

Now

$$\begin{aligned} \int_0^\infty f(t) dt &= \int_0^\infty \lambda e^{-\lambda t} dt \\ &= \left(\frac{\lambda e^{-\lambda t}}{-\lambda} \right)_0^\infty \\ &= (-e^{-\lambda t})_0^\infty \\ &= -e^{-\infty} + e^0 = 1 \end{aligned}$$

which means that $f(t)$ is a valid pdf.

In R, as is usual, d followed by a shortened version of the name of the distribution, in this case `dexp`, gives the pdf. The pdf of Examples 16.1–16.4 are generated respectively with

```
par(mfrow = c(2, 2))
curve(dexp(x, 2), 0, 3, main = expression(paste(lambda, " = 2")),
      font.main = 1, xlab = "t", ylab = "f(t)")
curve(dexp(x, 3), 0, 3, main = expression(paste(lambda, " = 3")),
      font.main = 1, xlab = "t", ylab = "f(t)")
curve(dexp(x, 4), 0, 3, main = expression(paste(lambda, " = 4")),
      font.main = 1, xlab = "t", ylab = "f(t)")
curve(dexp(x, 6), 0, 3, main = expression(paste(lambda, " = 6")),
      font.main = 1, xlab = "t", ylab = "f(t)")
```

which gives Fig. 16.1. Here, `main = expression(paste(lambda, " = 2"))` causes the Greek letter $\lambda = 2$ to be written in the title of the first curve, and similarly for the others.

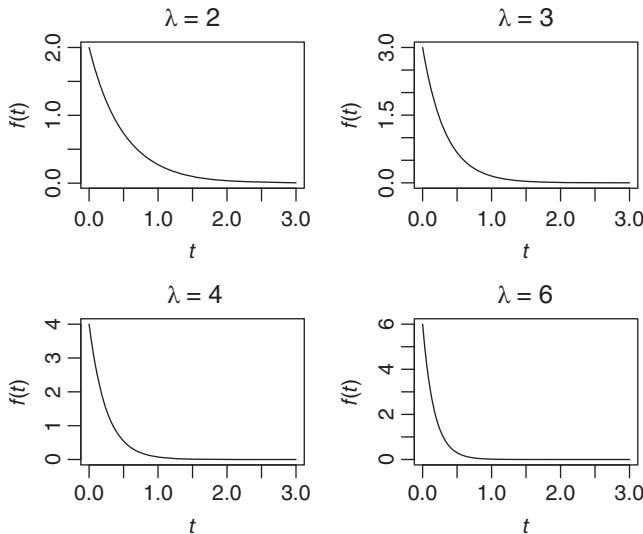


FIGURE 16.1 Density of the Exponential Distribution with $\lambda = 2, 3, 4$, and 6

Alternatively, using `for` loop with `bquote`, we could generate Fig. 16.1.

```
par(mfrow = c(2, 2))
parameter <- c(2, 3, 4, 6)
for(rate in parameter) {
  curve(dexp(x, rate), 0, 3, main = bquote(lambda ==.(rate)),
    font.main = 1, xlab = "t", ylab = "f(t)")
}
```

Notice that the densities in Fig. 16.1 are all skewed to the right, indicating that shorter waiting times are more likely than longer waiting times. Also note that the drop in the density is more acute and occurs earlier in the t -axis as λ increases. Recalling that λ is the average rate of the Poisson occurrences in a unit time interval, it is obvious that the higher this is, the more frequent the occurrences and the shorter the waiting time becomes. \square

16.3 CUMULATIVE DISTRIBUTION FUNCTION

The cdf is $F(t) = P(T \leq t)$ and measures the probability that the variable is less than or equal to t . From Equation 16.1, we have

$$F(t) = \begin{cases} 1 - e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

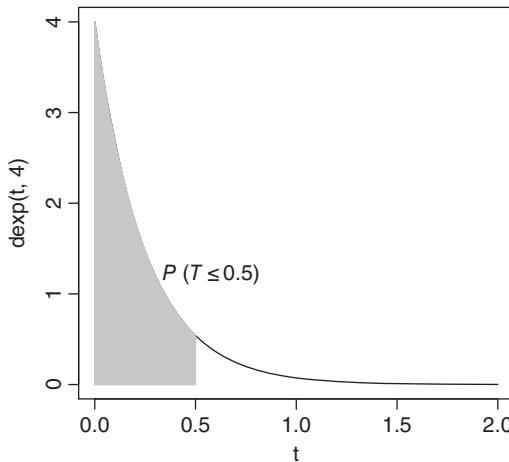


FIGURE 16.2 Area under the Exponential Curve

In Example 16.3, where jobs arrive every 15 seconds on average, $\lambda = 4$ per minute. Suppose we are interested in assessing the probability of waiting less than or equal to 30 seconds, that is, half a minute, $P(T \leq 0.5)$. Let us first draw the curve in R, and shade the area that we want.

```
curve(dexp(x, 4), 0, 2, xlab = "t", ylab = "dexp(t, 4)")
x <- seq(0, 0.5, 0.001)
lines(x, dexp(x, 4), type = "h") #shaded area
text(0.5, dexp(0.3, 4), expression(P(T <= 0.5)))
```

gives Fig. 16.2.

Here, `lines` is used to shade the area in the range $[0, 0.5]$ of x , and `text` writes the expression centered at the point $x = 0.5$ and $y = \text{dexp}(0.3, 4)$.

We can calculate $P(T \leq 0.5)$, the area under the curve from 0 to 0.5, directly by using integration.

$$\begin{aligned} P(T \leq 0.5) &= \int_0^{0.5} 4e^{-4t} dt \\ &= [-e^{-4t}]_{t=0}^{0.5} \\ &= 1 - e^{-2} \\ &= 0.86 \end{aligned}$$

Alternatively, we could use the cdf, in this case `pexp`, to get $P(T \leq 0.5)$, by writing

```
pexp(0.5, 4)
[1] 0.8646647
```

There is approximately an 86% chance that the waiting time between arrivals is less than or equal to 0.5 minute or 30 seconds. Put another way, 86% of all interarrival times will be less than or equal to 30 seconds.

Let us now use `pexp` to calculate the probabilities considered in Examples 16.1 – 16.4.

In Example 16.1 regarding website hits, where the rate average λ is two per day, let T be the time (in days) between hits.

$$P(T < 1) = P(\text{the waiting time is less than a day})$$

```
pexp(1, 2)
[1] 0.8646647
```

$$P(T > 2) = P(2 \text{ days will elapse without a hit})$$

```
1 - pexp(2, 2)
[1] 0.01831564
```

$$P(1 \leq T \leq 2) = P(\text{the waiting time is between one and two days})$$

```
pexp(2, 2) - pexp(1, 2)
[1] 0.1170196
```

In Example 16.2, where calls occur at a rate of 3 per day, let T be the waiting time between calls.

$$P(T < 0.5) = P(\text{you will be waiting less than half a day for a call})$$

```
pexp(0.5, 3)
[1] 0.7768698
```

$$P(T > 1) = P(\text{you will be waiting more than a day for a call})$$

```
1 - pexp(1, 3)
[1] 0.04978707
```

$$P(0.5 \leq T < 1) = P(\text{you will wait at least half a day and less than one whole day for a call})$$

```
pexp(1, 3) - pexp(0.5, 3)
[1] 0.1733431
```

16.4 MODELING LIFETIMES

So far, we have considered the exponential distribution as a model of the waiting time between Poisson occurrences. But this is not the only situation where the exponential distribution applies. Studies have shown, for example, that the lifetime of a computer monitor is often exponentially distributed. In fact the lifetime of many types of electronic components that do not typically experience wear-out type failures have been found to be exponentially distributed. The exponential distribution was the first distribution widely used to model lifetimes of components. In this section, we will look at some applications which relate to electronic components used in the area of computing.

In general, if the lifetime of a component can be modeled by an exponential distribution of the form

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

then λ is called the *failure (hazard) rate* of the machine. The probability that the component will last more than t units of time,

$$P(T > t) = e^{-\lambda t}$$

is called the *reliability* of the component at time t and is denoted by $\text{Rel}(t)$.

Example 16.5

The length of life in years, T , of a heavily used workstation in a student laboratory is exponentially distributed with $\lambda = 0.5$ year, that is,

$$f(t) = \begin{cases} (0.5) e^{-(0.5)t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

The probability that the workstation will last more than t years, $P(T > t)$,

$$\text{Rel}(t) = P(T > t) = e^{-(0.5)t}$$

The probability that the workstation will last more than one year is

$$\text{Rel}(1) = e^{-(0.5)} = 0.607$$

The probability that the workstation will last more than two years is

$$\text{Rel}(2) = e^{-1} = 0.368$$

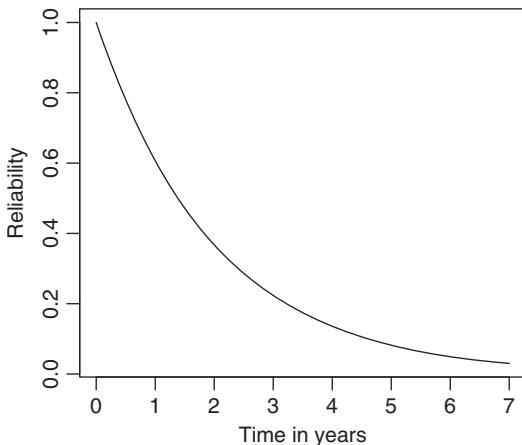


FIGURE 16.3 Reliability of a workstation with Failure Rate $\lambda = 0.5$

We can use R to examine the reliability as t increases.

```
curve(exp(-0.5 * x), 0, 7, xlab = "Time in years",
      ylab = "Reliability", type = "l")
```

gives Fig. 16.3. Δ

We see from Fig. 16.3 that the reliability decreases with time.

You will recall that in Chapter 8, we looked at the reliability of systems of components arranged in series and parallel and defined the reliability of a component as the probability that the component survives for some fixed period. Often the reliability of components is estimated from the exponential distribution.

Let us return to Example 8.5 concerning a system consisting of five components arranged in series. Suppose we know that the distribution of the lifetime of each component is exponential with $\lambda = 0.1$:

$$f(t) = \begin{cases} (0.1) e^{-(0.1)t}, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

Now we might seek to establish the reliability of a component at time t , that is, the probability that the component lasts beyond t .

$$\text{Rel}(t) = P(T > t) = e^{-(0.1)t}$$

The probability that a component will last more than one year is

$$\text{Rel}(1) = e^{-(0.1)} = 0.9048374$$

As the components are independent and arranged in series, then the reliability of the system at the end of one year is $(e^{-(0.1)})^5$ which can be calculated in R with

```
(exp(-0.1)) ** 5
[1] 0.6065307
```

Example 16.6

A workstation contains ten chips each operating independently. Each has an average lifetime of 1,000 hours, and the lifetime is exponential. All chips must work for the workstation to work. What is the probability that the workstation will last more than 1,000?

Here, the mean $E(T)$ is given as 1,000, and we know that the mean of an exponential distribution is $1/\lambda$. So $1/\lambda = 1,000$, giving $\lambda = 0.001$.

Let E be the event that the workstation will last more than 1,000 hours. Then,

$$\begin{aligned} P(E) &= P(\text{all 10 chips last more than 1000 hours}) \\ &= (P(T > 1000))^{10} \\ &= (e^{-\lambda \cdot 1000})^{10} \\ &= (e^{-1})^{10}, \text{ since } \lambda = 0.001 \\ &= e^{-10} \end{aligned}$$

Calculating in R

```
exp(-10)
```

gives

```
[1] 4.539993e-05
```

a very small probability indeed. \triangleleft

16.5 QUANTILES

As is the convention, q followed by the shortened version exp of the exponential name, $qexp$ calculates the quantiles of the exponential distribution. In this case, the quantile function deals with the maximum waiting time between two occurrences.

We might ask in Example 16.3: What is the maximum waiting time between two job submissions with 95% confidence?

We need to find k so that

$$P(T \leq k) = 0.95$$

You will recognize this as the quantile function.

```
qexp(0.95, 4)
[1] 0.748933
```

This means that the probability that there will be at most a waiting time of 0.75 minute, about 45 seconds, between two job submissions is 0.95.

In Example 16.5, we might seek to establish when the reliability of the workstation is just 5%.

Choose k so that the probability that

$$P(T > k) = 0.05$$

or equivalently, choose k so that

$$P(T \leq k) = 0.95$$

You will recognize this as the quantile function. From R

```
qexp(0.95, 0.5)
[1] 5.991465
```

which means that the chance that the machine will last longer than six years is just 5%. Equivalently, the reliability of the workstation after six years is just 5%.

$$\text{Rel}(6) = 0.05$$

Example 16.7

Suppose that there are 50 workstations in the laboratory, we might ask how long will it take to have more than 90% of them still working, or equivalently at most 10% to have broken down?

We need to find k so that the probability that the machine will last longer than k is 90%

$$P(T > k) = 0.9$$

or equivalently

$$P(T \leq k) = 0.1$$

which is the quantile function, and is obtained in R with

```
qexp(0.1, 0.5)
[1] 0.2107210
```

This means

$$P(T \leq 0.21) = 0.1$$

There is a 10% chance that the workstation will last just 0.21 year, and conversely, a 90% chance that the lifetime will be greater than that. So in just under 3 months (0.21 year), one would expect to find about 5 workstations broken, and 45 still working.

To check it in *R*

```
pexp(0.21, 0.5)
[1] 0.09967548
```

△

16.6 EXPONENTIAL EXPECTATIONS

Regarding the website hits discussed in Example 16.1, you might want to know how long, on average, you will have to wait for a hit.

For the exponential distribution, the average is

$$E(T) = \int_0^\infty t \lambda e^{-\lambda t} dt \quad (16.2)$$

This is done using “integration by parts”:

$$\int_0^\infty u dv = [uv]_0^\infty - \int_0^\infty v du$$

The trick is to spot the u and v in Equation 16.2. If

$$t \lambda e^{-\lambda t} dt = u dv$$

we could let $u = t$ and $dv = \lambda e^{-\lambda t} dt$. Therefore, $du = dt$ and $v = -e^{-\lambda t}$. Then,

$$\begin{aligned} \int_0^\infty t \lambda e^{-\lambda t} dt &= [-te^{-\lambda t}]_0^\infty + \int_0^\infty e^{-\lambda t} dt \\ &= \left[-te^{-\lambda t} - \frac{1}{\lambda} e^{-\lambda t} \right]_0^\infty \\ &= \frac{1}{\lambda} \end{aligned}$$

So, if website hits arrive at the rate of $\lambda = 2$ per day, the average waiting time, in days, between any two hits is

$$E(T) = \frac{1}{\lambda} = \frac{1}{2}$$

It is also reasonable to ask if the interarrival times will vary.

The variance of the exponential distribution is

$$\sigma^2 = \int_0^\infty \left(t - \frac{1}{\lambda} \right)^2 \lambda e^{-\lambda t} dt$$

It can be shown, using integration by parts again, that

$$\sigma^2 = \frac{1}{\lambda^2} \quad (16.3)$$

The derivation of the variance in Equation 16.3 is given in Appendix C. We will show, in the next section, how to obtain an approximation of it using simulation in *R*.

In Example 16.1, concerning website hits, which occur at a rate of $\lambda = 2$ per day, the average waiting time, in days, between hits is

$$E(T) = \frac{1}{2}$$

and the variance is

$$V(T) = \frac{1}{2^2}$$

In Example 16.2, concerning calls arriving at the rate of 3 per day, the average waiting time, in days, between calls is

$$E(T) = \frac{1}{3}$$

and the variance is

$$V(T) = \frac{1}{3^2}$$

For job submissions, in Example 16.3, which arrive at the rate of 4 per minute, the average waiting time, in minutes, between jobs is

$$E(T) = \frac{1}{4}$$

while the variance is

$$V(T) = \frac{1}{4^2} = 0.0625$$

In Example 16.4, concerning messages arriving to a computer server at the rate of 6 per hour, the average waiting time, in hours, between message arrivals is

$$E(T) = \frac{1}{6}$$

and the variance is

$$V(T) = \frac{1}{6^2}$$

Obviously, the greater the arrival rate λ , the shorter the average waiting time and smaller the variance.

Looking at Example 16.5, concerning the lifetime of workstations, with $\lambda = 0.5$,

$$E(T) = 1/0.5 = 2$$

which means the expected lifetime of the workstation is two years.

The variance is

$$V(T) = 1/(0.5)^2 = 4$$

16.7 SIMULATING EXPONENTIAL PROBABILITIES AND EXPECTATIONS

In this section, we show how to simulate values from the exponential distribution and to use the simulated values to obtain estimates of the mean and variance.

16.7.1 Simulating Job Submissions

We return to Example 16.3, the job submissions to a server, and examine the pattern of job submissions by generating random numbers from the appropriate exponential distribution using the *R* function

```
rexp(n, lambda)
```

The first argument specifies the number of simulations to be carried out, and the second argument is the parameter λ of the exponential distribution. For example, the function

```
submissions <- rexp(10000, 4)
```

generates 10,000 random numbers from an exponential distribution with a parameter $\lambda = 4$ and stores them as a vector called `submissions`. This experiment can be taken as simulating the waiting time between 10,000 job submissions.

To understand what the vector `submissions` contains, let us look at the first two terms.

```
round(submissions[1:2], 2)
[1] 0.61 0.18
```

approximately 0.61 and 0.18 minutes or equivalently 37 and 11 seconds waiting for the first and second job, respectively. This means that the first job arrived after 37 seconds, and the second job arrived 11 seconds later, at the 48th second.

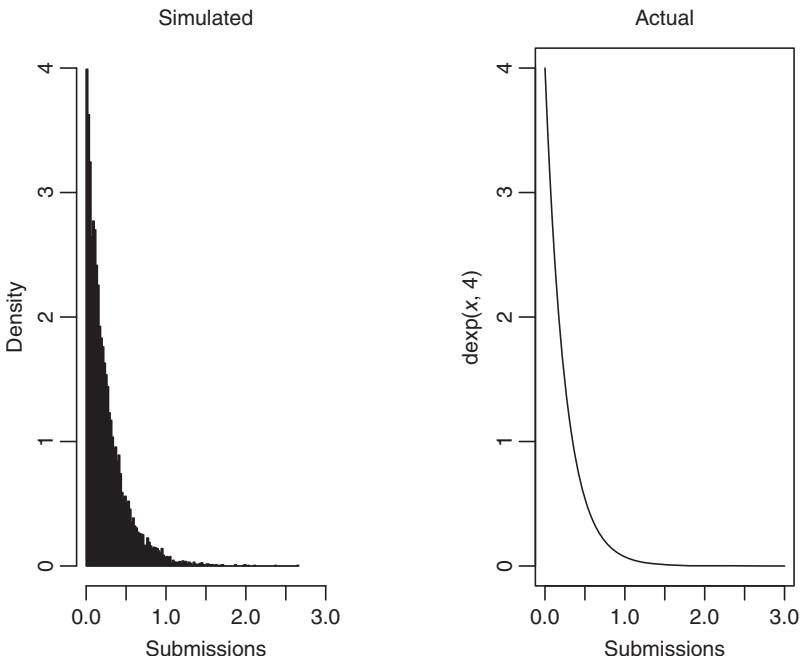


FIGURE 16.4 Simulated and Actual Distributions of Job Submissions to a Server, Assuming Interarrival Times are Exponentially Distributed with a Rate of 4 Per Minute

To summarize the output, we use the `hist` function

```
hist(submissions, freq = FALSE, xlim= c(0,3), ylim= c(0,4),
     breaks = 100, main = "Simulated", font.main = 1)
```

Here, `freq = FALSE` normalizes the histogram so that the total area is equal to one. The output is given in Figure 16.4 alongside the actual distribution, which, you will recall, is obtained using `curve(dexp(x, 4))`.

Notice how the simulated normalized histogram resembles in shape the actual pdf.

The mean of the simulated submissions is obtained with

```
mean(submissions)
```

which gives

```
[1] 0.2477416
```

which is not too far away from the theoretical mean $\mu = 1/4 = 0.25$.

The variance is obtained with

```
var(submissions)
```

which gives

```
[1] 0.06405638
```

This also is a close approximation of the true value of the variance $1/4^2 = 0.0625$.

To obtain the maximum, use

```
max(submissions)
```

gives

```
[1] 2.613249
```

which indicates that the longest time between submissions is likely to be just over 2.6 minutes.

16.7.2 Simulating Lifetimes

We could use *R* to simulate the probabilities in Example 16.5 concerning the lifetime of workstations with $\lambda = 0.5$.

```
lifetimes <- rexp(10000, 0.5)
```

generates 10,000 values.

To estimate the probability that the workstation will last more than six years, write

```
computersgreater6 <- subset(lifetimes, lifetimes > 6)
length(computersgreater6)/length(lifetimes)
0.0465
```

which is an approximation of $\text{Rel}(6) = e^{-(0.5)6}$ obtained in *R* with

```
exp(-0.5*6)
[1] 0.04978707
```

It could also be calculated with

```
1 - pexp(6, 0.5)
[1] 0.04978707
```

Similarly, we could estimate the probability that the workstation will last more than one year.

```
computersgreaterone <- subset(lifetimes, lifetimes > 1)
```

selects those workstations that have lifetimes greater than one year.

```
length(computersgreaterone)/length(lifetimes)
[1] 0.6023
```

obtains the number of workstations that have lifetimes greater than one year as a proportion of the original number of workstations, and returns 0.6023, which is an estimate of the reliability at one year.

$\text{Rel}(1) \approx 0.6023$.

The mean of the simulated lifetimes is obtained with

```
mean(lifetimes)
```

which gives

```
[1] 1.978845
```

which is not too far away from the theoretical mean $\mu = 1/(0.5) = 2$.

The estimated variance is obtained with

```
var(lifetimes)
```

which gives

```
[1] 4.01051
```

This also is a close approximation of the true value of the variance $1/(0.5)^2 = 4$.

16.8 AMNESIA

You will recall that, in Chapter 10, we proved that the Markov or “memoryless” property holds for the geometric distribution. We will now show that this property also holds for the exponential distribution.

In Example 16.3, where job submissions arrive at the rate of 4 per minute, we might ask:

What is the probability that a submission will arrive in the next 15 seconds given that the last 15 seconds passed without a submission?

Equivalently: What is the probability that, after a wait of 0.25 minute, a job will arrive in the next 0.25 minute?

We seek

$$P(T \leq 0.5 | T > 0.25)$$

Now

$$\begin{aligned} P(T \leq 0.5 | T > 0.25) &= \frac{P((T > 0.25) \cap (T \leq 0.5))}{P(T > 0.25)} \\ &= \frac{P(0.25 < T \leq 0.5)}{P(T > 0.25)} \end{aligned}$$

For the numerator

$$\begin{aligned} P(0.25 < T \leq 0.5) &= \int_{0.25}^{0.5} 4e^{-4t} dt \\ &= -e^{-4(0.5)} + e^{-4(0.25)} \\ &= -e^{-2} + e^{-1} \\ &= e^{-1}(1 - e^{-1}) \end{aligned}$$

The denominator is

$$\begin{aligned} P(T > 0.25) &= \int_{0.25}^{\infty} 4e^{-4t} dt \\ &= (-e^{-4t})_{0.25}^{\infty} \\ &= e^{-1} \end{aligned}$$

So,

$$\begin{aligned} P(0.25 < T \leq 0.5 | T > 0.25) &= \frac{e^{-1}(1 - e^{-1})}{e^{-1}} \\ &= 1 - e^{-1} \\ &= P(T \leq 0.25) \end{aligned}$$

which means that the probability that a job will arrive in the next 0.25 minute, after the gap of 0.25 minute, is just the initial probability of waiting less than 0.25 minute. The probability is unaffected by the time that has passed without a job submission, that is the Markov or “memoryless property” holds.

Theorem 16.1 The Markov property of the exponential distribution
If T is an exponential random variable, then

$$P(T \leq u + t | T > t) = P(T \leq u)$$

Proof

$$\begin{aligned} P(T \leq u + t | T > t) &= \frac{P((T > t) \cap (T \leq u + t))}{P(T > t)} \\ &= \frac{P(t < T \leq u + t)}{P(T > t)} \end{aligned}$$

The numerator is

$$\begin{aligned} P(t < T \leq u + t) &= \int_t^{u+t} \lambda e^{-\lambda x} dx \\ &= (-e^{-\lambda x})_t^{u+t} \\ &= -e^{-\lambda(u+t)} + e^{-\lambda t} \\ &= e^{-\lambda t}(1 - e^{-\lambda u}) \end{aligned}$$

The denominator is

$$\begin{aligned} P(T > t) &= \int_t^{\infty} \lambda e^{-\lambda x} dx \\ &= (-e^{-\lambda x})_t^{\infty} \\ &= e^{-\lambda t} \end{aligned}$$

Thus,

$$\begin{aligned} P(T \leq u + t | T > t) &= \frac{e^{-\lambda t}(1 - e^{-\lambda u})}{e^{-\lambda t}} \\ &= 1 - e^{-\lambda u} \\ &= P(T \leq u) \end{aligned}$$

So

$$P(T \leq u + t | T > t) = 1 - e^{-\lambda u} = P(T \leq u)$$

and conversely

$$P(T > u + t | T > t) = e^{-\lambda u} = P(T > u)$$

□

Example 16.8

WhatsApp texts arrive on your smartphone at an average rate of 12 per hour. Find the probability that a text will arrive in the next five minutes, given that 10 minutes has elapsed since the last text.

We need to calculate $P(T \leq 15 | T > 10)$, where T is the waiting time.

From the Markov property,

$$P(T \leq 15 | T > 10) = P(T \leq 5)$$

Now

$$P(T \leq 5) = 1 - e^{-5\lambda}$$

12 per hour means $\lambda = 12/60 = 0.2$ per minute.

$$P(T \leq 5) = 1 - e^{-(5)(0.2)} = 1 - e^{-1} = 0.632$$

A check in R gives

```
pexp(5, 0.2)
[1] 0.6321206
```

There is approximately a 63% chance that a text will arrive in the next five minutes, regardless of the time you have already been waiting.

Conversely,

$$P(T > 15 | T > 10) = P(T > 5) = 0.37$$

The probability that you will have to wait at least a further five minutes for a text is 0.37 regardless of how much time that you have been waiting already. \triangleleft

Returning to Example 16.5, we might ask: what is the probability that a workstation which has lasted 2 years will last at least a further three years?

From the Markov Property,

$$P(T > 5 | T > 2) = P(T > 3)$$

which means that the probability that the workstation will last three years more, after lasting for two years, is the same as the probability of lasting three years from the start. However long it lasts, it is still as good as new. Its breakdown will be as a result of some sudden failure, not wear and tear.

Example 16.9

Studies of a single-machine-tool system showed that the time that the machine operates before breaking down is exponentially distributed with a mean of 10 hours.

- (a) Determine the failure rate and the reliability.
- (b) Find the probability that the machine operates for at least 12 hours before breaking down.
- (c) If the machine has already been operating 8 hours, what is the probability that it will last another 4 hours?

Solution

Now, the mean $E(T)$ is given as 10 hours, and we know that the mean of an exponential distribution is $1/\lambda$. So $1/\lambda = 10$, giving $\lambda = 0.1$. Then, for this example, the specific pdf is

$$f(t) = \begin{cases} (0.1) e^{-(0.1)t}, & t \geq 0, \\ 0, & t < 0 \end{cases}$$

- (a) The failure rate is 0.1 hour, and the reliability $\text{Rel}(t) = P(T > t) = e^{-(0.1)t}$.
- (b) The reliability at $T = 12$,

$$\text{Rel}(12) = e^{-(1.2)} = 0.30$$

which means that there is a 30% chance that the machine will last longer than 12 hours.

- (c) Here, we are asking about the likelihood that the machine will last up to 12 hours given that it has already lasted 8 hours.

We seek

$$P(T > 12 | T > 8)$$

From Markov,

$$P(T > 12 | T > 8) = P(T > 4) = e^{-4(0.1)} = 0.67032$$

If the machine has lasted 8 hours, its reliability is the same as if it were new.

$$\text{Rel}(12 | T > 8) = \text{Rel}(4)$$

□

16.9 SIMULATING MARKOV

You will recall that, in Section 16.7.1, we simulated job submissions to a server. Let us return to these simulated data now and investigate the Markov property.

The vector `submissions` contains 10,000 simulated interarrival times between jobs with arrival rate of $\lambda = 4$.

To obtain an approximation of $P(T > 0.5 | T > 0.25)$, we first select all the submissions greater than 0.25.

```
submissions25plus <- subset(submissions, submissions > 0.25)
length(submissions25plus)
[1] 3643
```

There are 3,643 interarrival times with waiting times greater than 0.25 minute (15 seconds). Now, having waited 15 seconds, the time we have to wait for the next submissions is obtained in *R* by creating a new variable

```
y <- submissions25plus - 0.25
```

Also

```
mean(y)
[1] 0.2500272
var(y)
[1] 0.06719528
```

Comparing these to the mean and variance of the full set of interarrival times,

```
mean(submissions)
[1] 0.2477416
var(submissions)
[1] 0.06405638,
```

you will agree that there is not much difference between the two sets of parameters.

These estimated parameters, in turn, are not too far away from the theoretical mean $\mu = 1/4 = 0.25$ and variance $1/4^2 = 0.0625$.

The simulated pdf of the distribution of waiting times from 15 seconds is depicted as a histogram using

```
hist(y, freq = FALSE, xlab = "t - 0.25 | t > 0.25",
      main = "f(t - 0.25 | t > 0.25)", font.main = 1)
```

and compared with the simulated pdf of the original waiting times also summarized as a histogram.

```
hist(submissions, freq = FALSE, xlab = "t",
      main = "f(t)", font.main = 1)
```

The outputs are given in Fig. 16.5.

You will notice in Fig. 16.5 that the two histograms are practically identical, which is no surprise, now that we know about Markov.

$P(T < 0.5 | T > 0.25)$ may be approximated in *R* by first counting the numbers less than 0.5, in the conditional distribution stored in *submissions25plus*

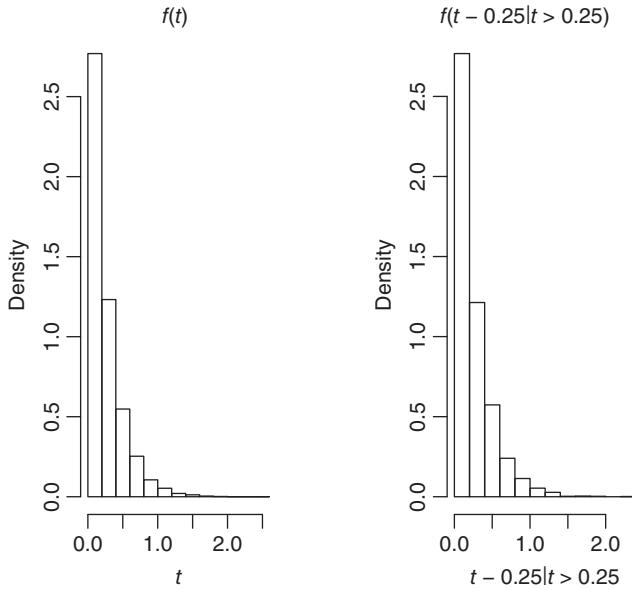


FIGURE 16.5 Simulated Exponential Distribution for $t > 0.25$ Compared with the Simulated Distribution for the Full Range of $t > 0$

```
favourable <- length(subset(submissions25plus, submissions25plus < 0.5))
```

and expressing this amount as a proportion of the total number in `submissions25plus`

```
total <- length(submissions25plus)
```

to get an estimate of $P(T < 0.5|T > 0.25)$

```
probability <- favourable/total
[1] 0.6324441
```

Compare this to $P(X < 0.25)$ estimated from the complete simulated population,

```
length(subset(submissions, submissions < 0.25))/10000
[1] 0.6289
```

which is not much different from 0.6324441 what we estimated $P(T < 0.5|T > 0.25)$ to be.

16.9.1 When Does Markov Not Hold?

So far, we have shown that the Markov property holds for the geometric discrete distribution (Chapter 10), and now the exponential distribution. We will next look at an example when the Markov property does not hold.

Let us repeat the simulation with a uniform rather than an exponential distribution. We now assume that the time T between job submissions is uniformly distributed in the range $[0, 0.5]$. Recall from the Chapter 15 that, when T is uniform in $[a, b]$,

$$E(T) = \frac{a + b}{2}$$

and

$$V(T) = \frac{(b - a)^2}{12}$$

In this case, with $a = 0$ and $b = 0.5$

$$E(T) = \frac{0 + 0.5}{2} = 0.25.$$

and

$$V(T) = \frac{(0.5 - 0)^2}{12} = 0.02083333$$

The jobs are arriving uniformly with an average of 0.25 minute between them. We can use R to generate a sequence of simulated values.

```
submissions<- runif(10000, 0, 0.5)
```

To get a feel for how jobs arrive with uniform arrival rate, let us look at the first 10 arrival times.

```
round(submissions[1:10], 2)
[1] 0.04 0.27 0.29 0.09 0.06 0.34 0.27 0.22 0.02 0.49
```

The first arrival occurs after a wait of 0.04 minute, the next job arrives 0.27 minute later at $0.04 + 0.27 = 0.31$ minute, the next one after a 0.29 minute gap, at $0.31 + 0.29 = 0.6$ minute, and so on.

We can summarize the simulated data.

```
hist(submissions, freq = F, xlab = "Interarrival times",
     main = " ", font.main = 1)
```

giving Fig. 16.6.

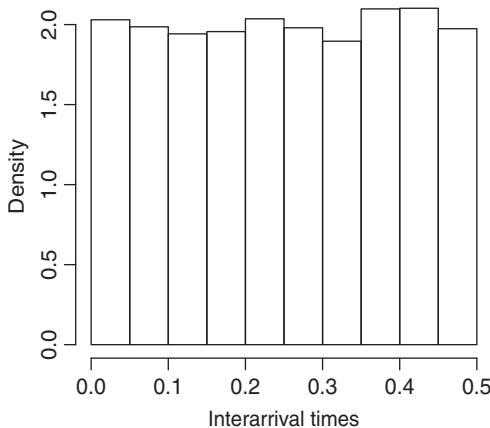


FIGURE 16.6 Interarrival Times Uniform in $[0, 0.5]$

The mean and variance of the simulated distribution are

```
mean(submissions)
[1] 0.2509014
var(submissions)
[1] 0.02096188
```

which are close to the theoretical mean 0.25 and variance 0.02083333.

Now suppose that no job has arrived in the last 0.25 minute. How long on average should we expect to wait for the next job submission?

To answer this question, we examine the behaviour of interarrival times which are greater than 0.25 minute. We return to the 10,000 simulated job submissions and eliminate those for which the interarrival times are less than or equal to 0.25 minute.

```
submissions25plus <- subset(submissions, submissions > 0.25)
length(submissions25plus)
[1] 5053
```

There are 5,053 submissions greater than 0.25 minute. This is to be expected with a uniform distribution, 0.25 is the mid-range of the interval $(0, 0.5)$, so we would expect about half of the 10,000 observations to be above the mid-range and half below.

Now we will examine the situation where we have waited 0.25 minute, and we want to know the probabilities of waiting times beyond this. To measure the remaining waiting time after 0.25 minute, create a new variable consisting of waiting times beyond 0.25 minute.

```
y <- submissions25plus - 0.25
```

To examine the characteristics of this variable

```
mean(y)
[1] 0.1243997
var(y)
[1] 0.005112792
```

which are not at all close to the theoretical distribution, where

$$E(T) = 0.25 \text{ and } V(T) = 0.02083333$$

The pdf of the distribution of waiting times beyond 0.25 minute is obtained with

```
hist(y, freq = F, main = "f(t - 0.25 | t > 0.25)",
      xlim = c(0, 0.5), ylim = c(0, 4), xlab = " ")
```

We compare this with a histogram of the original simulated data using

```
hist(submissions, freq = F, xlab = " ", main = "f(t)",
      xlim = c(0, 0.5), ylim = c(0, 4))
```

The two distributions are given in Fig. 16.7.

While both of the simulated distributions in Fig. 16.7 appear uniform, the waiting time after 0.25 minute is now less than the original. What this means is that, with uniform waiting times, after waiting a certain amount of time, the remaining wait is dependent on how long you have been waiting already.

To compare some empirical probabilities

- $P(T < 0.25)$

```
length(subset(submissions, submissions < 0.25)) / 10000
[1] 0.127
```

- $P(T < 0.5 | T > 0.25)$

```
length(subset(submissions25plus, submissions25plus < 0.5))
/length(submissions25plus)
[1] 0.4387171
```

or equivalently

```
length(subset(y, y < 0.25)) / length(y)
[1] 0.4387171
```

which is not at all close to 0.127 as obtained from the entire simulated population. The Markov property does not hold for the uniform distribution.

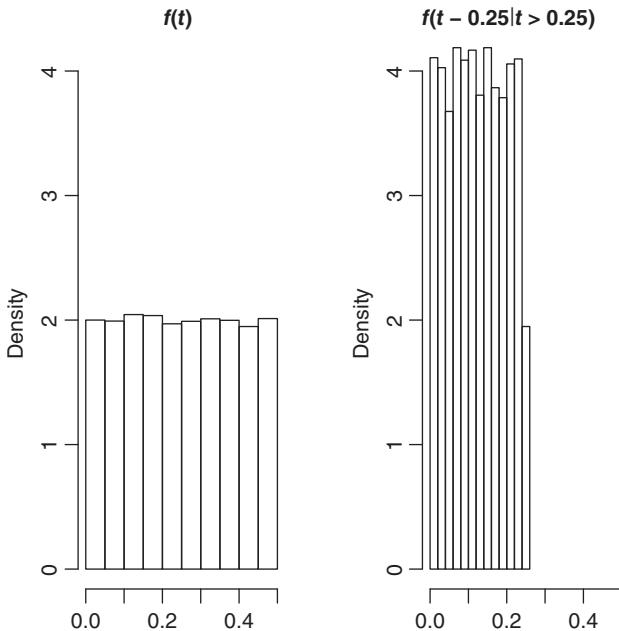


FIGURE 16.7 Simulated Uniform Waiting Time after $t > 0.25$ Compared with the Original

The exponential distribution is the only continuous distribution for which the Markov property holds. The geometric discrete distribution, discussed in Chapter 10, also enjoys this property; it is the only discrete distribution to do so. In a sense, the exponential distribution can be viewed as the continuous version of the geometric distribution; both measure the time until the first success. The geometric distribution, you will recall, models the time until the first success in a sequence of Bernoulli trials, while the exponential distribution models the time until the first success in a continuous time interval.

EXERCISES 16.1

1. In Example 16.3 with $\lambda = 4$ per minute, use R to obtain:
 - (a) $P(T \leq 0.25) = P(\text{time between submissions is at most 15 seconds});$
 - (b) $P(T > 0.5) = P(\text{time between submissions is greater than 30 seconds});$
 - (c) $P(0.25 < T < 1) = P(\text{time between submissions is between 15 seconds and 1 minute}).$
2. In Example 16.4, with $\lambda = 6$ per hour, use R to obtain the probabilities:
 - (a) $P(T \leq 1/6) = P(\text{at most 10 minutes (1/6 hour) elapses between arrivals});$

- (b) $P(T > 1/2) = P(\text{more than half an hour elapses between arrivals});$
- (c) $P(1/6 < T < 1/2) = P(\text{the time between arrivals is between 10 and 30 minutes}).$
3. The average rate of job submissions in a computer center is 2 per minute. If it can be assumed that the number of submissions per minute has a Poisson distribution, calculate the probability that:
- (a) more than two jobs will arrive in a minute;
 - (b) at least 30 seconds will elapse between any two jobs;
 - (c) less than 30 seconds will elapse between jobs;
 - (d) a job will arrive in the next 30 seconds, if no jobs have arrived in the last 30 seconds.
4. Assume that the number of messages input to a communication channel in an interval of duration t seconds is Poisson distributed with parameter $0.4t$. Compute the probabilities of the following events:
- (a) exactly three messages will arrive during a 10-second interval;
 - (b) more than three messages will arrive during a 10-second interval;
 - (c) 10 seconds will pass without a message;
 - (d) a message will arrive in the next 10 seconds, if 5 seconds have already gone by without a message.
5. A new user of a smartphone receives an average of two messages per day. If it can be assumed that the arrival pattern of these messages is Poisson, calculate the following probabilities:
- (a) exactly three messages will be received in any single day;
 - (b) more than three messages will arrive in a day;
 - (c) a day will pass without a message;
 - (d) if one day has passed without a message, six messages will be received the next day.
6. A software developer establishes a support hotline for customers to call in with questions about the use of the software. An average of three calls arrive every five minutes. Assuming a Poisson arrival rate, compute the probabilities of the following events:
- (a) exactly four calls will arrive during a five-minute interval;
 - (b) more than three calls will arrive during a five-minute interval;
 - (c) five minutes will pass without a call;
 - (d) a message will arrive in the next five minutes, knowing that five minutes have already gone by without a call.

7. A website receives an average of 15 visits per hour, which arrive following a Poisson distribution.
 - (a) Calculate the probability that at least 10 minutes will elapse without a visit.
 - (b) What is the probability that in any hour, there will be less than eight visits?
 - (c) Suppose that 15 minutes have elapsed since the last visit, what is the probability that a visit will occur in the next 15 minutes.
 - (d) Calculate the top quartile, and explain what it means.
8. It is suspected that the number of machine failures per day in a certain plant has a Poisson distribution with parameter $\lambda = 2$. Present maintenance facilities can repair two machines per day, otherwise a contractor is called out.
 - (a) In any given day, what is the likelihood of having machines repaired by a contractor?
 - (b) What is the probability that two days will pass without a machine failure?
 - (c) Obtain the expected time between failures.
9. The lifetime of microprocessors is known to be exponentially distributed with a mean of 5,000 hours.
 - (a) What proportion of microprocessors will last longer than 5,000 hours?
 - (b) If the manufacturer, Power Products, gives a guarantee that the microprocessors will last more than 4,000 hours, otherwise they will replace them, what proportion of microprocessors are likely to have to be replaced?
 - (c) The company wants to give a guarantee which covers 80% of the processors. What lifetime should they advertise?
10. A laptop motherboard has an average life of two years.
 - (a) What is the probability that it will fail in the first year?
 - (b) If it does last for one year, what is the probability that it will last a further two years?
11. You have bought a new iPhone. The manufacturers give a guarantee that if your battery needs a recharge before eight hours of runtime, your iPhone will be replaced. If it can be assumed that the runtime of batteries is exponentially distributed with an average of 10 hours, what is the probability that your battery will fail before the guaranteed time?
12. The length of life of an electronic component has an exponential pdf with mean 1,000 hours.
 - (a) Find the probability that a component lasts at least 1,500 hours.
 - (b) Suppose a component has been in operation for 1,000 hours, what is the probability that it will last for another 500 hours?
13. Studies of a single-machine-tool system have shown that the amount of time the machine operates before breaking down is exponentially distributed with a mean of 1,000 minutes. Find the probability that the machine operates for at least 12.5 hours before breaking down.

14. Suppose a battery has a useful life described by the exponential distribution with a mean of 500 days.
 - (a) What is the probability that it will fail before its expected lifetime?
 - (b) If a battery has a 12-month (365-day) warranty, what is the probability that it will fail during the warranty period?
 - (c) What is the probability that a battery which has lasted 365 days will operate beyond its expected lifetime?
15. The average lifetime of a certain computer part is 10 years. If it can be assumed that the lifetime is exponentially distributed:
 - (a) calculate the probability that the computer part lasts more than seven years;
 - (b) calculate the time for which you will be 80% certain that the computer part will last.

16.10 PROJECT

Carry out a simulation experiment in R of texts to a smartphone that arrive at 6 per hour, generating 10,000 exponential values (T) of time between texts, and plot the simulated distribution, estimating its mean and variance. Then consider the conditional distribution of all interarrival times greater than six minutes, obtain the conditional simulated distribution, the mean and variance, and compare these to the simulated distribution of the complete set.

To satisfy yourself that the Markov memoryless property holds, obtain estimates of

1. $P(T < 0.3|T > 0.1)$ and $P(T < 0.2)$
2. $P(T \geq 0.3|T > 0.1)$ and $P(T \geq 0.2)$
3. $P(T < 0.4|T > 0.1)$ and $P(T < 0.3)$

Repeat the experiment, assuming that the interarrival times are uniformly distributed in the interval $[3, 9]$ minutes.

17

QUEUES

Queues occur in many situations in computing:

- When a programming job is submitted, it takes its place in a queue, and has to wait until the server is free before being processed. Each job is processed when there is no other job in front;
- When a request is submitted to a website, if there are requests ahead of it, it cannot be serviced at once, so it takes its place in the queue;
- Where there are multiple users of a networked computer system with a shared printer, files submitted have to wait until the previous jobs have been printed.

In this chapter, we use *R* to investigate single server queues and their properties empirically. In particular, we look at the factors that determine the length of the queue and the likely response time.

17.1 THE SINGLE SERVER QUEUE

The simplest type of queuing system is that for which there is one server dealing with the jobs on a first-in first-out basis as depicted in Fig. 17.1.

In queuing terminology, this is known as a single-server queuing system. Empirical investigations have found that the time a server takes to process a task submitted to it

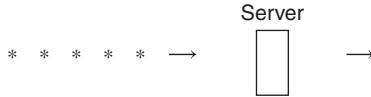


FIGURE 17.1 A Single Server Queue

in a queuing network can be modelled with the exponential distribution. Also it has been found that submissions often follow a Poisson distribution so that the interarrival time is exponential. Such a queue is denoted by M/M/1; the two Ms refer to the Markov/Memoryless property enjoyed by the interarrival and service times which are exponential, while the 1 refers to the single server.

17.2 TRAFFIC INTENSITY

The general M/M/1 queue is that where customers of any kind arrive at a single server, in accordance with a Poisson distribution. The customer is served immediately if the queue is empty, otherwise, joins the end of the queue. The service times are assumed to be exponentially distributed.

Suppose λ is the mean arrival rate of the jobs in a unit time interval. Denoting the interarrival times by T , the probability density function (pdf) of T is exponential.

$$f(t) = \lambda e^{-\lambda t}, \quad t \geq 0$$

and the cumulative distribution function (cdf) is

$$F(t) = P(T \leq t) = 1 - e^{-\lambda t}, \quad t \geq 0$$

The mean interarrival time is $1/\lambda$.

Denote by μ the average number of jobs processed in a unit time interval, and by S the processing time of the jobs. If S can be assumed to be exponential, the pdf of S is

$$f(s) = \mu e^{-\mu s}, \quad s \geq 0$$

and the cdf is

$$F(s) = P(S \leq s) = 1 - e^{-\mu s}, \quad s \geq 0$$

The average processing time is $1/\mu$.

The arrival rate relative to the service rate is called the *traffic intensity* and is denoted by I :

$$I = \frac{\lambda}{\mu}$$

There are three scenarios with respect to traffic intensity I :

1. $I > 1$ which means $\lambda > \mu$: the average arrival rate exceeds the average service rate. The jobs are arriving faster than they are being processed;

2. $I = 1$ which means $\lambda = \mu$: the average arrival rate equals the average service rate. The jobs are arriving on average at the same rate as they are being processed;
3. $I < 1$ which means $\lambda < \mu$: the average arrival rate is exceeded by the average service rate. The jobs are being processed faster than they arrive.

17.3 QUEUE LENGTH

Clearly, the queue length depends on the traffic intensity, that is the arrival rate λ relative to the service rate μ .

Let us investigate the queue lengths for differing traffic intensities.

Example 17.1

Supposing that jobs arrive at the rate of four jobs per second, use *R* to examine the queue length when the service rate is

- (a) 3.8 jobs per second;
- (b) 4 jobs per second;
- (c) 4.2 jobs per second.

You can assume that both the service times and the interarrival times are exponentially distributed. \triangleleft

17.3.1 Queue Length with Traffic Intensity > 1

In (a), the service rate $\mu = 3.8$ is less than the arrival rate $\lambda = 4$. Therefore, the traffic intensity $I = \lambda/\mu = 4/3.8 \approx 1.05 > 1$. In this case, we would expect the queue to increase indefinitely. The following *R* code¹ generates 10,000 Poisson arrivals, and 10,000 Poisson services, and calculates the queue length at each time interval.

```
arrivals <- rpois(10000, 4)
service <- rpois(10000, 3.8)
queue[1] <- max(arrivals[1] - service[1], 0)
for (t in 2:10000) queue[t] =
  max(queue[t-1] + arrivals[t] - service[t], 0)
plot(queue, xlab = "Time", ylab = "Queue length")
```

generates Fig. 17.2

It is clear from Fig. 17.2 that when $I > 1$, the length of the queue keeps increasing. Some action needs to be taken to halt the impetus. With a single server, either the service rate needs to be improved, or the arrival rate should be slowed down. Otherwise, the queue will keep getting bigger.

¹Thanks to Gary Keogh for these suggestions.

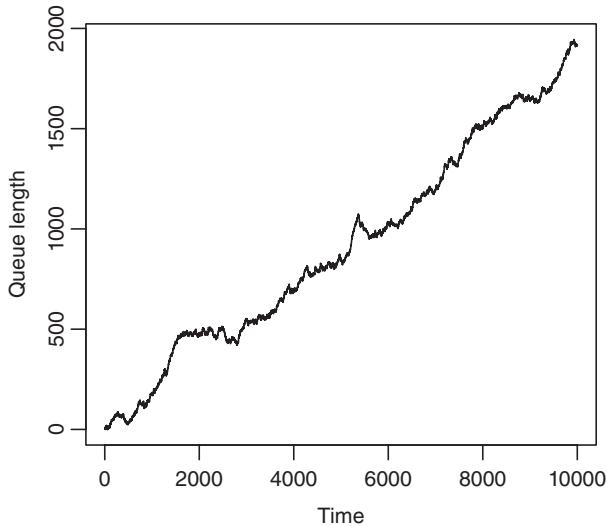


FIGURE 17.2 Queue Length with Traffic Intensity > 1

17.3.2 Queue Length when the Traffic Intensity = 1

In (b), the service rate $\mu = 4$ is the same as the arrival rate $\lambda = 4$. Therefore, the traffic intensity $I = \lambda/\mu = 1$. Let us investigate what happens to the queue length in this situation.

```
queue <- 0
arrivals <- rpois(10000, 4)
service <- rpois(10000, 4)
queue[1] <- max(arrivals[1] - service[1], 0)
for (t in 2:10000)
  queue[t] = max(queue[t-1] + arrivals[t] - service[t], 0)
plot(queue, xlab = "Time", ylab = "Queue length")
```

This code generates Fig. 17.3.

It is clear from Fig. 17.3 that there is still a problem when the service rate has been improved so that it is now the same as the arrival rate. The queue does not stabilize, and the long-term trend is an increasing queue.

17.3.3 Queue Length with Traffic Intensity < 1

In (c), the service rate $\mu = 4.2$, which is greater than the arrival rate $\lambda = 4$. Therefore, the traffic intensity $I = \lambda / \mu = 4/4.2 \approx 0.95 < 1$. Let us see what happens in this case.

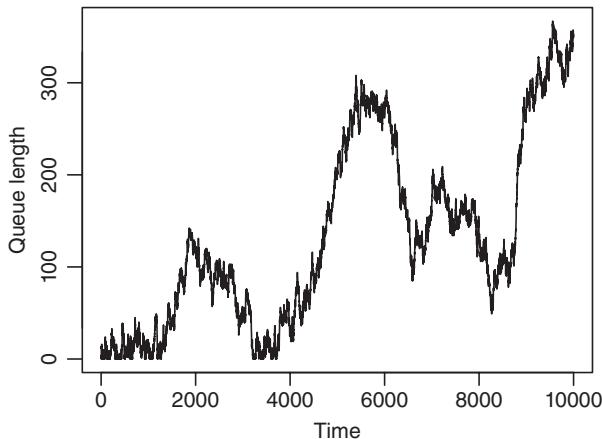


FIGURE 17.3 Queue Pattern with Traffic Intensity = 1

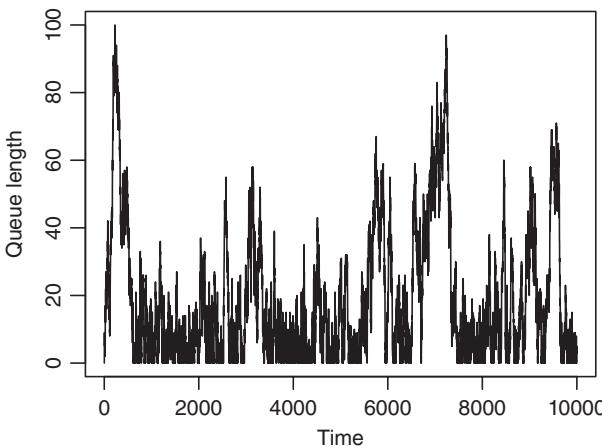


FIGURE 17.4 Queue Pattern when the Traffic Intensity $I < 1$

In R, we generate the queue pattern with service rate $\mu = 4.2$

```
service <- rpois(10000, 4.2)
```

Everything else is the same and we get Fig. 17.4

The important thing to notice in Fig. 17.4 is that the queue is not increasing over time. It seems to have stabilized. It makes sense, in this case, to calculate the average queue size.

```
mean(queue)
[1] 18.4218
sd(queue)
[1] 19.51606
```

What this means is that an average of 18.4218 jobs are in the queue, and Fig. 17.4 indicates that this will continue to be the case no matter how long the process continues. Therefore, when a job enters the queue, the average time that it will spend there is the average length of the queue multiplied by the average interarrival time, which is $1/\lambda = 1/4$. In this example, the average queuing time is calculated as

```
mean(queue) * (1/4)
[1] 4.6
```

The average time that a job spends in the queue is 4.6 seconds.

We can also estimate the worst-case scenario, the longest wait that is likely. The longest queue size in the simulated data is obtained with

```
max(queue)
[1] 103
```

Again, since interarrival time is 1/4 second, the longest wait in the queue is

```
max(queue) * (1/4)
[1] 25.75
```

nearly 26 seconds.

To look at the best case, we seek the minimum queue length.

```
min(queue)
[1] 0
```

It was found using the `table` function in *R*, that this occurred over 1,000 times, which means that over 10% of the jobs did not have to wait at all; these jobs went straight to the processing unit.

We take a last look at the queue with different traffic intensities in Fig. 17.5.

Figure 17.5 highlights that, when the traffic intensity $I > 1$, there is a severe queuing problem. With $I = 1$, the problem is not as severe, but it does exist, and we see that, in the long run, it will become serious. The only tenable solution to the queuing problem is to keep $I < 1$.

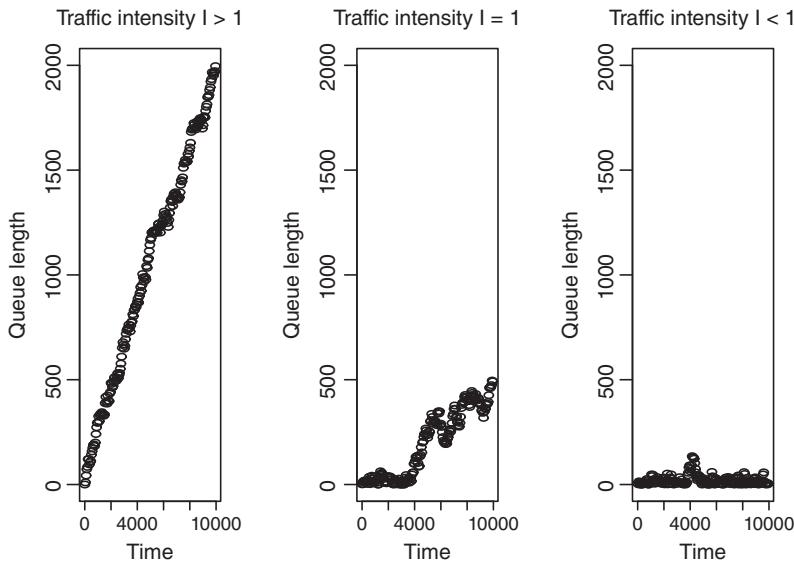


FIGURE 17.5 Queue Length with $\lambda = 4$ and $\mu = 3.8, 4$ and 4.2

17.4 AVERAGE RESPONSE TIME

Definition 17.1 *Average response time (ART)*

The average response time (ART) is the average time to job completion. Equivalently, it is the average turnaround time between the submission of the job and its return. \square

ART is the average time the job spends in the queue plus the average time it takes to be serviced.

We now calculate ART in Example 17.1 (c). In this case, the average service time is $1/\mu = 1/4.2 = 0.24$. Alternatively, we could estimate this from the simulated values.

```
mean(service)
[1] 4.183
```

which is the estimated service rate, and

```
1/mean(service)
[1] 0.2390629
```

is the estimated average service time.

We have shown that the average time a job spends waiting to be serviced is 4.6 seconds. Therefore, the ART is

$$\text{ART} = 4.6 + 0.24 = 4.86 \text{ s.}$$

So, with an arrival rate of 4 per second and a service rate at 4.2 per second, the ART is approximately 4.86 seconds.

Suppose that the manager of the center wishes to upgrade the equipment to improve the response time. A new system, which would increase the average service rate by 10%, is being considered. What effect would this have on the response time?

Let us investigate with R .

An increase of 10% in the average service rate means that the average service rate, μ , goes from 4.2 to 4.62 per second. Now, if we generate service rates using

```
service <- rpois(10000, 4.62)
```

and calculate and plot the queue length as we did before, we get Fig. 17.6.

The average queue length

```
mean(queue)
[1] 5.7595
```

Since the arrival rate is 4 per second, the average waiting time in the queue is

```
mean(queue) * (1/4)
[1] 1.439875
```

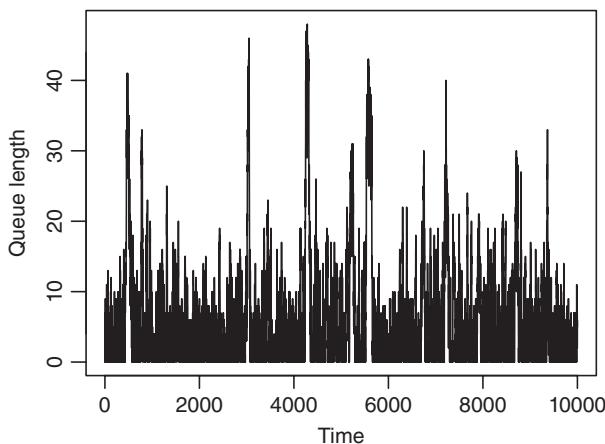


FIGURE 17.6 Queue Length with $\lambda = 4$ and $\mu = 4.62$

Adding the average service time, that has been reduced to $1/4.62 = 0.22$, to the average time in the queue, we get the ART.

$$\text{ART} = 1.44 + 0.22 = 1.66 \text{ s}$$

So, by improving the service rate by 10%, the ART has decreased from 4.86 to 1.66 seconds; it has gone down by 65%; the upgrade seems indeed worthwhile, depending on its cost of course.

17.5 EXTENSIONS OF THE M/M/1 QUEUE

The M/M/1 queue is one of the simplest types of queuing systems. It was developed to model the behavior of one server dealing with jobs arriving in one queue. However, it can be useful for studying more complicated processes in which there is more than one queue arriving to the single server. For example, suppose there are several software engineers submitting source programs independently to one server, and each submits according to a Poisson process. It is easy to show that the combined submissions are also Poisson. Similarly, requests to a website may come from thousands of users, and jobs to a printer in a computer laboratory may be submitted independently by many students.

Let us take the case of two software engineers independently submitting source programs to a single processor. The number of programs submitted by each engineer has a Poisson distribution with rates of λ_1 and λ_2 , respectively, in a unit interval of time. If X_1 is the number of programs submitted by Engineer 1, and X_2 is the number of programs submitted by Engineer 2, then the total number of programs submitted in a unit of time is $X_1 + X_2$. We seek the distribution of $X_1 + X_2$.

$$\begin{aligned} P(X_1 + X_2 = k) &= \sum_{i=0}^k P((X_1 = i) \cap (X_2 = k - i)) \\ &= \sum_{i=0}^k \left(e^{-\lambda_1} \frac{\lambda_1^i}{i!} \right) \left(e^{-\lambda_2} \frac{\lambda_2^{k-i}}{(k-i)!} \right) \\ &\quad \text{since the submissions are independent} \end{aligned}$$

$$\begin{aligned} &= e^{-(\lambda_1 + \lambda_2)} \sum_{i=0}^k \frac{1}{i!(k-i)!} \lambda_1^i \lambda_2^{k-i} \\ &= \frac{e^{-(\lambda_1 + \lambda_2)}}{k!} \sum_{i=0}^k \frac{k!}{i!(k-i)!} \lambda_1^i \lambda_2^{k-i} \\ &= \frac{e^{-(\lambda_1 + \lambda_2)}}{k!} (\lambda_1 + \lambda_2)^k \text{ using the binomial theorem} \end{aligned}$$

The last line is a Poisson probability with a mean of $\lambda_1 + \lambda_2$, which means that the sum of two independent Poisson random variables with respective means λ_1 and λ_2 is also Poisson with a mean of $\lambda_1 + \lambda_2$. By induction, the sum of three or more Poisson processes combine in a similar fashion.

From this, it follows that if there are two or more types of jobs each arriving according to a Poisson process to a single server, the queue can be modelled as a M/M/1 queue.

Example 17.2

The Daly Development Software House has configured a development server to act as a sandbox for software testing. The server processes programs submitted to it at the rate of one job per minute. There are four software engineers employed by the company, and each of these submits programs at an average rate of one job every six minutes. Use *R* to examine the queue at this current staff level.

The average service rate is one job per minute, that is, $\mu = 1$. The submission rate of each engineer is one every six minutes, that is $\lambda_i = 1/6$, $i = 1, 2, 3, 4$. If we can assume that the number of submissions of each software engineer is a Poisson random variable, and since Poisson processes combine, as we have just shown, then the sum of the submissions of the four engineers is also a Poisson random variable with mean $\lambda = 4/6 = 2/3$.

This is an M/M/1 queue with traffic intensity $I = \lambda/\mu = (2/3)/1 \approx 0.6667$. We can simulate it with the same code as we used in Section 17.3 by changing the arrival and service rates appropriately.

```
queue <- 0
arrivals <- rpois(10000, 2/3)
service <- rpois(10000, 1)
queue[1] <- max(arrivals[1] - service[1], 0)
for (t in 2:10000)
  queue[t] = max(queue[t-1] + arrivals[t] - service[t], 0)
plot(queue, xlab = "Time", ylab = "Queue length")
```

The simulated queue is given in Fig. 17.7.

The average queue length is

```
mean(queue)
[1] 1.5538
```

which means that there are less than two jobs on average in the queue. The worst-case scenario is obtained with

```
max(queue)
[1] 16
```

which means that the maximum number of jobs in the queue is 16.

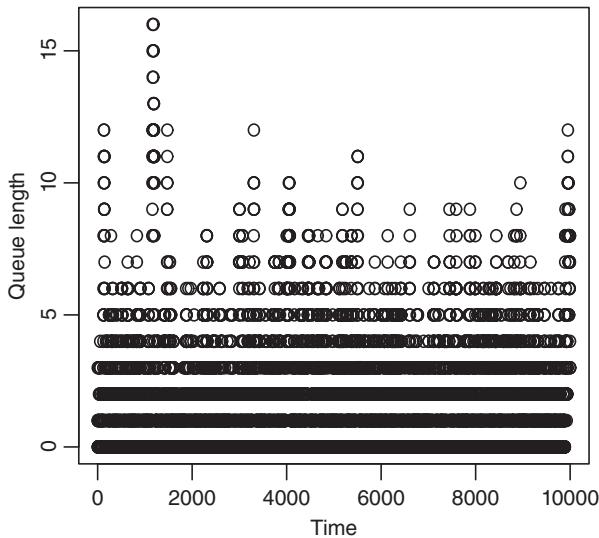


FIGURE 17.7 Queue Pattern with Four Engineers Submitting to One Processor

We can also examine how long the queue is over the 10,000 simulations.

```
plot(table(queue),
      xlab = "Length of queue",
      ylab = "Frequency")
```

gives Fig. 17.8

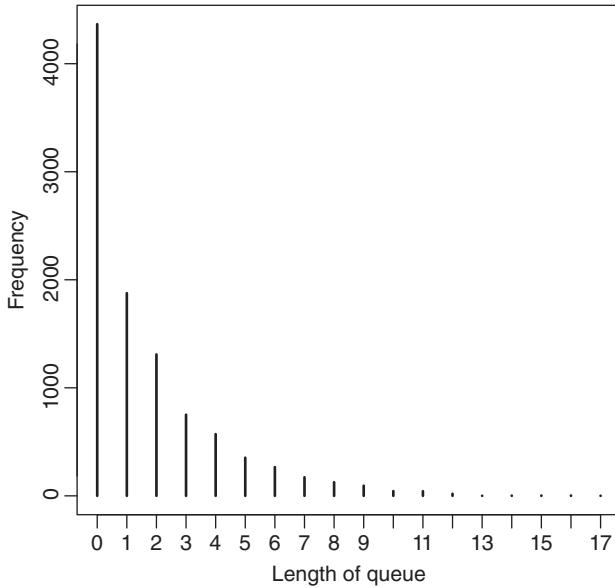
We see, from Fig. 17.8, that more than 40% of the arrivals were serviced immediately, and the queue length was greater than nine in very few cases. \triangleleft

In this chapter, we have adopted an empirical approach when investigating queues and their properties. However, there is a vast theoretical development in queuing theory. The reader is referred to Ross (2014) for more details.

EXERCISES 17.1

1. In Example 17.1, the queue size was calculated using

```
queue[1] <- max(arrivals[1] - service[1], 0)
for (t in 2:10000) queue[t] =
  max(queue[t-1] + arrivals[t] - service[t], 0)
```

**FIGURE 17.8** Queue Length over 10,000 Simulations

- (a) Redo this simulation for 1000 and 5000 iterations for each of the three traffic intensity values considered.
- (b) To examine in *R* the queue pattern at the different stages of the queue, use

```
par(mfrow = c(1, 3))
plot(queue, xlim = c(1, 1000), xlab = "Time",
     ylab = "Queue length", type = "l")

plot(queue, xlim = c(1, 5000), xlab = "Time",
     ylab = "Queue length", type = "l")
```

- (c) The following *R* code

```
for (t in 2:10000) aver[t] = mean(queue[1:t])
plot(aver, xlab = "time", type = "l")
```

calculates and plots the average queue length at all stages of the simulation process. Implement this on the data for each of the three intensity levels, and discuss the differences.

2. A geospatial analysis system processes images at the rate of one image per second. There are eight sensors supplying images, and each of these submits images at an average rate of one every 10 seconds.²
 - (a) Use R to simulate the queue with the eight sensors.
 - (b) The manager of the center is considering adding one, two, or three extra sensors to the system. Use R to investigate the effect these additions will have on the processing time and decide on the best strategy.
3. Jobs sent to a printer are held in a buffer until they can be printed sequentially on a first come, first served basis. Jobs arrive at the printer at the rate of 10 per second. The average time to print a job is five seconds. Assuming an M/M/1 system, use R to simulate the queue 10,000 times and use this to:
 - (a) find the expected value and standard deviation of the number of jobs in this system at any time.
 - (b) estimate the probability that a submitted job will be printed immediately.

17.6 PROJECT

Investigate using R how the queue length varies in a M/M/1 queue for values of the traffic intensity from 0.4 to 0.9 in intervals of 0.1, and comment on your results.

REFERENCE

Ross, S.M. (2014), *Introduction to Probability Models*, Academic Press, Amsterdam.

²Thanks to James Power for this suggestion.

18

THE NORMAL DISTRIBUTION

In many practical applications, measurements tend to cluster around a central value, with the largest proportion of measurements occurring near this central value, and decreasing as the measurements extend at either side of this central value (Figure 18.1).

This bell-shaped curve, symmetrical about the central value, is called the “normal distribution.”

Let us consider the heights of females and males. It is easy to accept that, on average, males are taller than females, but maybe the spread about their respective averages could be the same. It is credible also that most female and male heights cluster around the average, getting fewer as the height increases above and decreases below the average. The heights of females and males might well be modeled by the normal distributions depicted in Fig. 18.2.

In Fig. 18.2, we see two bell-shaped curves: the first one depicting the female heights is centered lower down the axis than the males, but the variation at either side of this central point is the same for both females and males.

Another example of normal distributions arises in product manufacturing. Suppose components are manufactured by two machines which are set to a diameter 10 cm. One machine produces components subject to an error of 1 cm, while the second machine has an error of 2 cm. Figure 18.3 shows the distributions of the diameters of the components produced by each of the two machines.

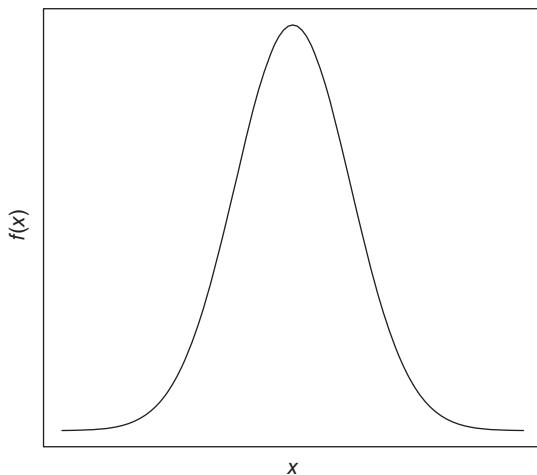


FIGURE 18.1 A Bell-Shaped and Symmetrical Curve

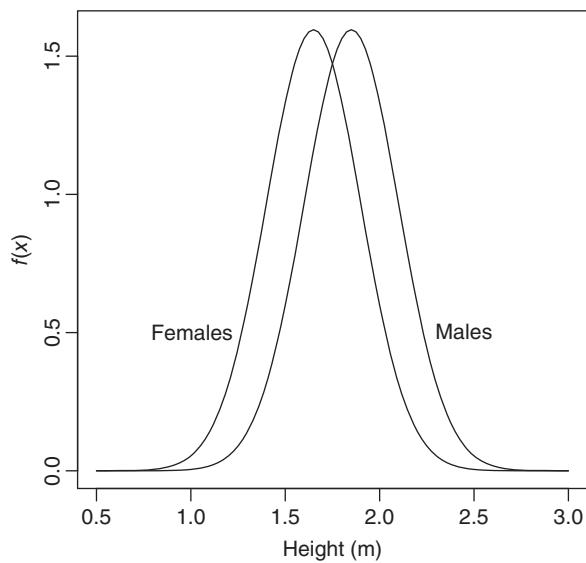


FIGURE 18.2 Two Normal Distributions with Different Means and the Same Spread

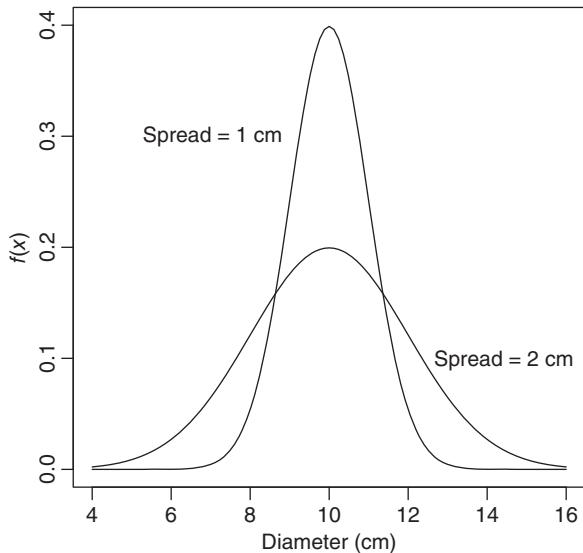


FIGURE 18.3 Two Normal Distributions with the Same Mean and Different Spreads

In Fig. 18.3, we see two bell-shaped curves, both centered at 10, one high and narrow, the other low and wide.

These are just some examples of what is arguably the most important distribution in statistics, the normal distribution. In this chapter, we will study some of its many applications.

18.1 THE NORMAL PROBABILITY DENSITY FUNCTION

Definition 18.1 *Normal distribution*

A continuous random variable X is said to have a normal distribution with parameters μ and σ if the probability density function $f(x)$ takes the form

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < X < \infty.$$

□

It can be shown that

$$E(X) = \mu$$

$$V(X) = \sigma^2$$

X is said to be normally distributed with a mean of μ and a variance of σ^2 , and therefore σ is the standard deviation. The normal distribution is often denoted by

$$X \sim N(\mu, \sigma)$$

Increasing μ moves the distribution along the x -axis, as we saw in Fig. 18.2.

$$X \sim N(1.65, 0.25) \text{ for females and } X \sim N(1.85, 0.25) \text{ for males.}$$

Increasing σ widens the distribution, as we saw in Fig. 18.3,

$$X \sim N(10, 1) \text{ and } X \sim N(10, 2)$$

18.1.1 Graphing Normal Curves in R

The probability density function (pdf) is obtained in R with `pnorm`. To plot a normal curve, use the function `curve` with `dnorm` as the argument.

```
curve (dnorm(x, 1.65, 0.25), from = 0, to = 3)
```

plots a normal curve with a mean of 1.65 and a standard deviation of 0.25 in the range 0–3.

To suppress the labels and axis ticks, you will need to include `labels = FALSE` and `ticks = FALSE`. For example

```
curve (dnorm(x, 1.65, 0.25), from = 0, to = 3,
       ylab = "f(x)", labels = FALSE, ticks = FALSE)
```

generates Fig. 18.1.

To generate normal curves on the same axis as we did in Figs. 18.2 and 18.3, write

```
plot(NA, xlab = "Height(m)", xlim = c(0.5, 3), ylab = "f(x)",
     ylim = c(0, 1.6)) # the x, y axes
curve(dnorm(x, 1.65, 0.25), add = TRUE)
curve(dnorm(x, 1.85, 0.25), add = TRUE)
text(1.15, 0.5, "Females")
text(2.35, 0.5, "Males")
```

which gives Fig. 18.2.

Similarly,

```
plot(NA, xlab = "Diameter (cm)", xlim= c(4,16), ylab = "f(x)",
     ylim = c(0, 0.4)); # the x, y axes
curve(dnorm(x, 10, 1), add = TRUE)
curve(dnorm(x, 10, 2), add = TRUE)
text(7.5, 0.3, "Spread = 1 cm")
text(15, 0.05, "Spread = 2 cm")
```

gives Fig. 18.3.

The first and second arguments in the `text` function indicates where to write the text.

Experiment yourself with changing the coordinates of the `text` function to move the text around the graph.

18.2 THE CUMULATIVE DISTRIBUTION FUNCTION

The cumulative distribution function (cdf) is obtained in *R* with `pnorm`.

$$P(X \leq x) = \text{pnorm}(x, \mu, \sigma)$$

Example 18.1

From previous records, it is known that examination results are normally distributed with mean $\mu = 45$ and standard deviation $\sigma = 4$, that is,

$$X \sim N(45, 4)$$

We can obtain probabilities very quickly in *R* using `pnorm`. Table 18.1 gives some examples.

To generate the shaded areas in the curves, first draw the appropriate normal distribution

```
curve(dnorm(x, 45, 4), 30, 60, xlab = "x", ylab = "f(x)")
```

Then define the area to be shaded, $40 < X < 50$. For example,

```
x <- seq(40, 50, 0.01) #values of x in the range 40-50 in
intervals of 0.01
```

Finally, impose the shaded areas on the original normal curve.

```
lines (x, dnorm(x, mean = 45, sd = 4), type = "h",
col = "grey") #shading
```

which has the effect of shading the area under the curve from 40 to 50. △

18.2.1 Conditional Probabilities

Example 18.2

An analogue signal received at a detector (measured in microvolts, μV) is normally distributed with a mean of 100 and a variance of 256, that is, $X \sim N(100, 16)$.

What is the probability that the signal will be less than $120 \mu\text{V}$, given that it is larger than $110 \mu\text{V}$?

We want

$$P(X < 120 | X > 110) = \frac{P((X < 120) \cap (X > 110))}{P(X > 110)} = \frac{P(110 < X < 120)}{P(X > 110)}$$

TABLE 18.1 Using R to Obtain Normal Probabilities: $X \sim N(45, 4)$

<i>R</i> Command	Probability	Area
$P(X < 45)$	<code>pnorm(45, 45, 4)</code>	0.5
$P(X > 45)$	<code>1 - pnorm(45, 45, 4)</code>	0.5
$P(X > 50)$	<code>1 - pnorm(50, 45, 4)</code>	0.1056498
$P(40 < X < 50)$	<code>pnorm(50, 45, 4) - pnorm(40, 45, 4)</code>	0.7887005
$P(X > 37)$	<code>1 - pnorm(37, 45, 4)</code>	0.9772499

$P(110 < X < 120)$ is obtained in R with

```
pnorm(120, 100, 16) - pnorm(110, 100, 16)
```

which gives 0.1603358.

To get $P(X > 110)$, write

```
1 - pnorm(110, 100, 16)
```

which gives 0.2659855

Then

$$P(X < 120 | X > 110) = \frac{0.1603358}{0.2659855} = 0.6027988 \quad \triangleleft$$

Example 18.3

From previous records, scores on an aptitude test are normally distributed with mean $\mu = 500$ and standard deviation $\sigma = 100$.

$$X \sim N(500, 100)$$

What is the probability that an individual's score is greater than 650, given that it exceeds 500?

We want

$$P(X > 650 | X > 500) = \frac{P((X > 500) \cap (X > 650))}{P(X > 500)} = \frac{P(X > 650)}{P(X > 500)}$$

To get $P(X > 650)$ in R, write

```
1 - pnorm(650, 500, 100)
```

which gives 0.0668072

To get $P(X > 500)$, write

```
1 - pnorm(500, 500, 100)
```

which gives 0.5

Then,

$$\frac{P(X > 650)}{P(X > 500)} = \frac{0.0668072}{0.5} = 0.1336144 \quad \triangleleft$$

18.3 QUANTILES

Regarding the aptitude tests in Example 18.3, you will recall that the mean $\mu = 500$ and the standard deviation $\sigma = 100$. We might want to know the cut-off point for the top 5% best test results. We seek k so that 5% of scores are above and 95% below, that is,

$$P(X \leq k) = 0.95$$

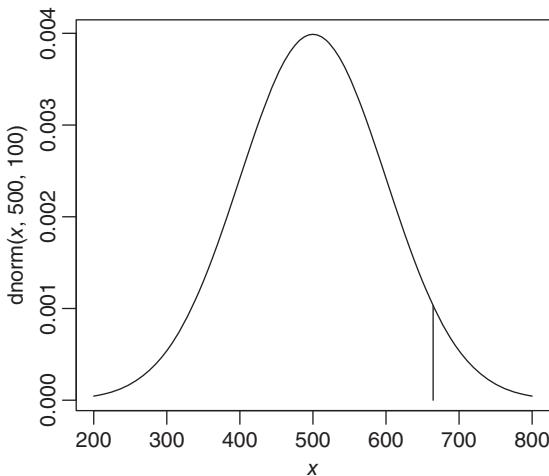


FIGURE 18.4 The Cut-off Point for 95% of the Normal Distribution with $\mu = 500$ and $\sigma = 100$

You will recognize this as the quantile, and with the same convention as for other distributions, we use `qnorm` in *R*.

```
qnorm(0.95, 500, 100)
[1] 664.4854
```

which means that 5% of the individuals scored more than 664. To illustrate in *R*, write

```
curve(dnorm(x, 500, 100), 200, 800)
x <- qnorm(0.95, 500, 100)
lines(x, dnorm(x, 500, 100), type = "h")
```

which gives Fig. 18.4.

What is the middle 40%?

We need to divide the distribution into a lower 30%, a middle 40% and an upper 30%.

To get the upper 30%, we choose k so that $P(X > k) = 0.3$ or equivalently

$$P(X \leq k) = 0.7$$

From *R*

```
qnorm(0.7, 500, 100)
```

gives $k = 552.44$.

Similarly, use the quantile function to obtain the lower 30%. Choose k so that

$$P(X \leq k) = 0.3$$

```
qnorm(0.3, 500, 100)
```

gives $k = 447.5599$.

Therefore, the interval $(447.56, 552.44)$ contains the middle 40% of the scores.

We could estimate the interquartile range, which is the middle 50%. The first quartile is obtained with

```
qnorm(0.25, 500, 100)
[1] 432.551
```

The third quartile is obtained with

```
qnorm(0.75, 500, 100)
[1] 567.449
```

Therefore, the range of the middle 50% of the data, that is, the interquartile range is $(432.55, 567.45)$.

18.4 THE STANDARD NORMAL DISTRIBUTION

The cumulative probability function (`pnorm`) together with the quantile function (`qnorm`) provide most of the probabilities for any normal distribution that we may encounter. These functions in *R* essentially replace traditional statistical tables. If for some reason *R* is not available, you will need to go back to tables.

Just one normal distribution is tabulated: the standard normal random variable Z , for which the mean $E(Z) = 0$, and the variance $V(Z) = 1$. The pdf is given by

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}, \quad -\infty < z < \infty$$

It can be shown that

$$E(Z) = 0$$

$$V(Z) = 1$$

Z is said to be normally distributed with a mean of 0 and a variance of 1, which is often denoted by

$$Z \sim N(0, 1)$$

Writing

```
curve(dnorm(x, 0, 1), from = -3, to = 3,
      xlab = "z", ylab = "f(z)")
```

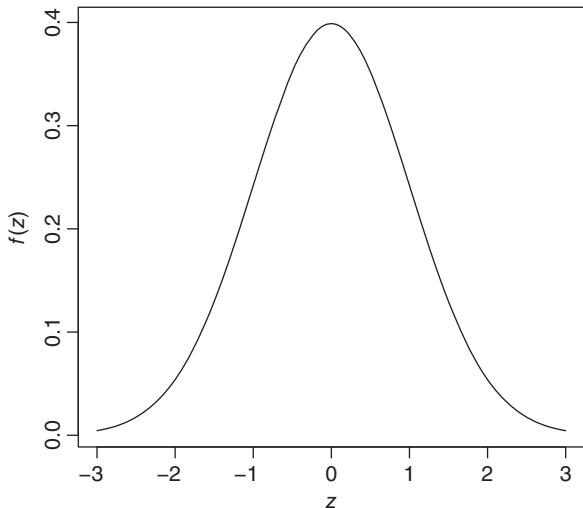


FIGURE 18.5 The Density of the Standard Normal Distribution

or, since 0 and 1 are the defaults, simply write

```
curve(dnorm(x), from = -3, to = 3, xlab = "z", ylab = "f(z)")
```

which gives Fig. 18.5.

The general normal distribution can be transformed into the standard normal distribution by a simple linear transformation. If

$$X \sim N(\mu, \sigma)$$

that is,

$$E(X) = \mu \quad \text{and} \quad V(X) = \sigma^2$$

then

$$E\left(\frac{X - \mu}{\sigma}\right) = \frac{1}{\sigma}E(X - \mu) = \frac{1}{\sigma}(E(X) - \mu) = 0$$

and

$$V\left(\frac{X - \mu}{\sigma}\right) = \frac{1}{\sigma^2}V(X - \mu) = \frac{\sigma^2}{\sigma^2} = 1$$

It can also be shown that the normality of a random variable remains after a linear transformation. So

$$Z = \frac{X - \mu}{\sigma} \sim N(0, 1)$$

We can see from Fig. 18.5 that, while in theory the range of Z is $(-\infty, \infty)$, the probabilities decrease very quickly as we move along the axis away from the mean zero. In fact,

- $P(-1.96 < Z < 1.96) = 0.95$. To check this, in R write

```
pnorm(1.96, 0, 1) - pnorm(-1.96, 0, 1)
```

- $P(-2.58 < Z < 2.58) = 0.99$.

Since $Z = (X - \mu)/\sigma$, it follows, for the general $N(\mu, \sigma)$ distribution, that

- $P(\mu - 1.96\sigma < X < \mu + 1.96\sigma) = 0.95$ which means 95% of the normal distribution lies within 1.96σ of the mean.
- $P(\mu - 2.58\sigma < X < \mu + 2.58\sigma) = 0.99$, meaning that 99% of the normal distribution lies within 2.58σ of the mean.

Returning to Example 18.1, where $X \sim N(45, 4)$, we could determine the probabilities from the table given in Appendix E using

$$Z = \frac{X - 45}{4}$$

To calculate $P(40 < X < 50)$ using the table, you will need to standardize

$$\begin{aligned} P(40 < X < 50) &= P\left(\frac{40 - 45}{4} < \frac{X - 45}{4} < \frac{50 - 45}{4}\right) \\ &= P(-1.25 < Z < 1.25) \end{aligned}$$

From the table in Appendix E, $P(Z < 1.25) = 0.8944$. It follows that $P(Z > 1.25) = 1 - 0.8944 = 0.1056$. By symmetry, $P(Z < -1.25) = 0.1056$. Therefore, $P(-1.25 < Z < 1.25) = P(Z < 1.25) - P(Z < -1.25) = 0.8944 - 0.1056 = 0.7888$.

In Example 18.2, where $X \sim N(100, 16)$, we calculate

$$\begin{aligned} P(X < 120 | X > 110) &= \frac{P(110 < X < 120)}{P(X > 110)} \\ &= \frac{P((110 - 100)/16 < Z < (120 - 100)/16)}{P(Z > (110 - 100)/16)} \\ &= \frac{P(0.625 < Z < 1.25)}{P(Z > 0.625)} \end{aligned}$$

From the table $P(Z < 1.25) = 0.8944$. To obtain $P(Z < 0.625)$ from the table, look at $P(Z < 0.62) = 0.7324$ and $P(Z < 0.63) = 0.7357$, which means $P(Z < 0.625)$ is somewhere in between, around 0.734. Therefore,

$$\begin{aligned} \frac{P(0.625 < Z < 1.25)}{P(Z > 0.625)} &\approx \frac{0.8944 - 0.7340}{1 - 0.734} \\ &= \frac{0.1604}{0.266} \\ &= 0.603 \end{aligned}$$

To calculate directly from *R*, write

```
prob <- (pnorm(1.25) - pnorm(0.625)) / (1 - pnorm(0.625))
round(prob, 3)
[1] 0.603
```

In Example 18.3, where $X \sim N(500, 100)$,

$$\begin{aligned} \frac{P(X > 650)}{P(X > 500)} &= \frac{P(Z > (650 - 500)/100)}{P(Z > (500 - 500)/100)} \\ &= \frac{P(Z > 1.5)}{P(Z > 0)} = \frac{1 - 0.9332}{0.5} = \frac{0.0668}{0.5} \\ &= 0.1336 \end{aligned}$$

In *R*

```
prob <- (1 - pnorm(650, 500, 100)) / (1 - pnorm(500, 500, 100))
round(prob, 4)
```

gives

```
[1] 0.1336
```

which, you will agree is simpler. Why bother with tables if you can use *R*?

18.5 ACHIEVING NORMALITY: LIMITING DISTRIBUTIONS

One of the reasons that the normal distribution is so important is that it is the limiting distribution of many others. Let us investigate.

Looking back at some of the applications of the binomial distribution given in Fig. 11.1, we notice again that tossing a coin 10 times, and counting the number of heads, produces a bell-shaped and symmetric pdf. Also, counting the number of defectives in a sample of size 20 had a normal-like pdf.

Again, when examining applications of the Poisson distribution in Fig. 13.6, we considered the number of messages, X , arriving at a server following a Poisson distribution with a rate $\lambda = 6$ per hour, and noticed that the pdf of X is normal-like, bell-shaped, and symmetrical.

In this section, we will use *R* to examine the limiting distributions of the binomial and Poisson distributions and investigate how and when they approach normality.

18.5.1 Normal Approximation to the Binomial

Let us return to Example 11.3, which considers a manufacturing process for integrated circuit chips of which 20% are defective. We calculate the probability of getting X defectives in n for various values of X .

To start let $n = 20$. The mean or expected number of defectives in a sample of size 20 is

$$\mu = np = 20 \times 0.2 = 4$$

The variance is

$$\sigma^2 = npq = 20 \times 0.2 \times 0.8 = 3.2$$

so that the standard deviation

$$\sigma = \sqrt{npq} = \sqrt{3.2} = 1.79 \quad \text{rounded to 2 decimal places}$$

We compare the binomial probabilities of X defectives in a sample of size 20 with the probabilities obtained from the normal distribution with the same mean and variance, that is,

$$X \sim N(4, 1.79)$$

The following *R* code generates the normal pdf of X with a mean of 4 and a variance of $npq = 3.2$

```
np <- 20*0.2
npq <- 20*0.2*0.8
curve(dnorm(x, np, sqrt(npq)), np-3*sqrt(npq), np+3*sqrt(npq),
      xlab = "Number of defectives", ylab= "pdf") #normal pdf
```

Note that we take the range of the variable to be $[np - 3\sqrt{npq}, np + 3\sqrt{npq}]$. This interval incorporates over 99% of the normal distribution.

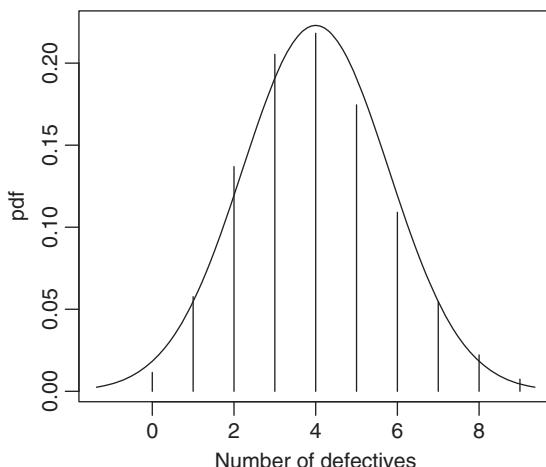


FIGURE 18.6 Normal Approximation to the Binomial $n = 20, p = 0.2$

To superimpose the binomial probabilities with parameters $n = 20$ and $p = 0.2$ write

```
x <- 0:20
lines(x, dbinom(x, 20, 0.2), type = "h")
```

which gives Fig. 18.6

It appears from Fig. 18.6 that the normal pdf fits this binomial reasonably closely. However, it must be pointed out that the binomial variable is discrete with integer values only, while the normal variable is continuous. When we are comparing probabilities, we need to calculate the normal probabilities in the range ± 0.5 above and below the binomial integer. For example, we compare $P(X = x)$ in the binomial with $P(x - 0.5 < X < x + 0.5)$ in the normal. Let us do this for the above example.

```
x <- seq(0, 20, 1)
n <- 20
p <- 0.2
x1 <- dbinom(x, n, p) #binomial probabilities
x2 <- pnorm(x + 0.5, n*p, sqrt(n*p*(1-p)))
  - pnorm(x - 0.5, n*p, sqrt(n*p*(1-p))) #normal density
    in the interval (x - 0.5, x + 0.5)
y <- data.frame(round(x1, 3), round(x2, 3)) #round to 3 decimal places
```

Table 18.2 gives some of the output. You will agree that the approximation is quite close.

Let us now examine the approximation for varying n . Figure 18.7 gives the binomial and normal densities for values of $n = 5, 10, 20, 25, 50$, and 100 . We keep p fixed at 0.2.

TABLE 18.2 Comparison of Normal and Binomial Probabilities with $n = 20$ and $p = 0.2$

Defectives x	Binomial $P(X = x)$	Normal $P(x - 0.5 < X < x + 0.5)$
0	0.012	0.019
1	0.058	0.056
2	0.137	0.120
3	0.205	0.189
4	0.218	0.220
5	0.175	0.189
6	0.109	0.120
7	0.055	0.056
8	0.022	0.019
9	0.007	0.005
10	0.002	0.001

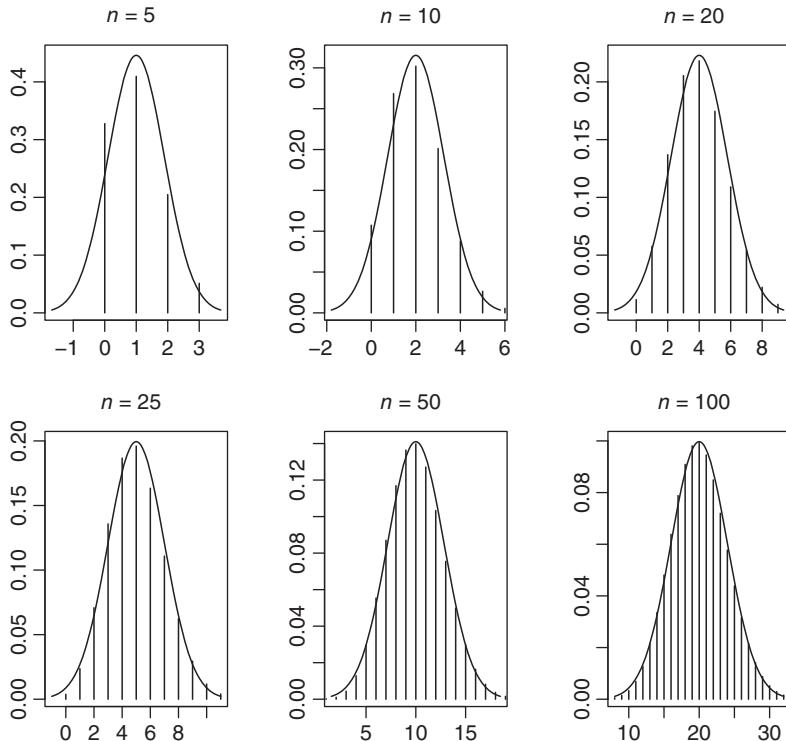


FIGURE 18.7 Binomial and Normal Densities with $p = 0.2$, for Varying Values of n

From Fig. 18.7, we are able to see the binomial distribution becoming more “normal-like” as the sample size increases. When $n = 5$ and $n = 10$, the approximations are inaccurate; some of the range of the normal distribution is even negative. The approximation improves when $n = 20$, and $n = 25$. But by the time n reaches 50 and 100, the approximation is very good indeed.

The first graph in Fig. 18.7 is generated with

```

p <- 0.2
q <- 1-p
n <- 5
curve(dnorm(x, n*p, sqrt(n*p*q)), n*p-3*sqrt(n*p*q), n*p+3*sqrt(n*p*q),
      xlab = " ", ylab = " ",
      main = bquote(n == .(n))) #normal pdf
x <- 0:n
lines(x, dbinom(x, n, 0.2), type = "h")# binomial probabilities

```

and the others may be obtained by varying values of n . More efficiently though, the six graphs may be generated using a `for` loop as follows:

```

par( mfrow = c(2,3) )
p <- 0.2
q <- 1-p
size <- c(5, 10, 20, 25, 50, 100)
for (n in size) {
curve(dnorm(x, n*p, sqrt(n*p*q)), n*p-3*sqrt(n*p*q), n*p+3*sqrt(n*p*q),
      xlab = " ", ylab = " ",
      main = bquote(n ==.(n))) #normal pdf
x <- 0:n
lines(x, dbinom(x, n, 0.2), type = "h") #binomial probabilities
}

```

The curves in Fig. 18.7 are generated for each value in the vector `size`. The `bquote` function quotes its argument and prints the value of the expression in `.()`; in this case, it writes the sample size in the main heading.

We can show that the p also has an influence on the accuracy of the convergence. Observe in Fig. 18.8 how the “shape” of the binomial pdfs become more bell-shaped and more symmetric as p goes from 0.01, for which it is skewed, to $p = 0.5$, for which it appears to be perfectly symmetrical.

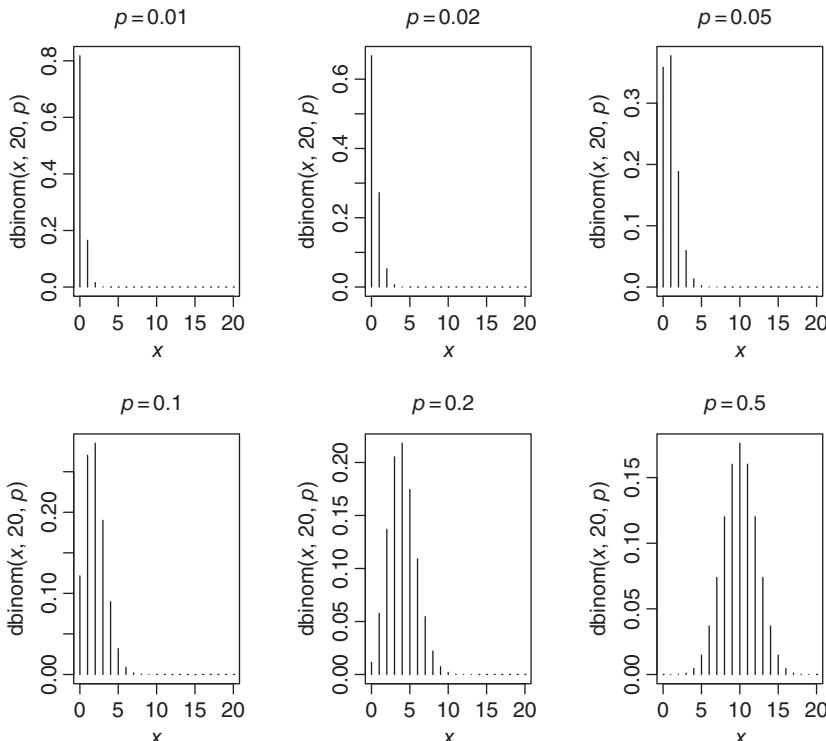


FIGURE 18.8 The Binomial Probabilities with $n = 20$, for Varying Values of p

Figure 18.8 is generated in *R* with

```
par(mfrow = c(2, 3))
x <- 0:20
prob <- c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5)
for (p in prob){
  plot(x, dbinom(x, 20, p), main = bquote(p ==.(p)), type = "h")
}
```

We can superimpose the normal curves using

```
par(mfrow = c(2, 3))
x <- 0:20
prob <- c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5)
for (p in prob){
  curve(dnorm(x, 20*p, sqrt(20*p*(1-p))), 
         20*p-3*sqrt(20*p*(1-p)), 20*p+3*sqrt(20*p*(1-p)),
         main = bquote(p ==.(p)), xlab = " ", ylab = " ")
  lines(x, dbinom(x, 20, p), type = "h")
}
```

This gives Fig. 18.9.

We see from Fig. 18.9 that the fit is not good for low values of p , with the normal distribution returning some negative possibilities. On the other hand, we see from the second row, that as p increases, the fit gets better. It is practically exact when $p = 0.5$.

The limits obtained earlier do not contradict those obtained in Chapter 13, where we showed that the binomial tends to the Poisson when p is small. With small p , the binomial will eventually get to the normal, but will tend to the Poisson first. When p is near the mid range of the $(0,1)$ interval, the binomial “skips” over the Poisson and tends quickly to the normal.

18.5.2 The Normal Approximation to the Poisson Distribution

In Fig. 13.6, we considered messages, X , arriving at a server following a Poisson distribution with a rate $\lambda = 6$ per hour. We noticed that the pdf of X is bell shaped and symmetrical. Let us examine it more closely now, and see if it can be approximated with the normal distribution.

Recall the mean $E(X) = \lambda$ and the standard deviation $\sigma = \sqrt{\lambda}$. The following *R* code plots the probabilities for the Poisson distribution with parameter $\lambda = 6$, and superimposes the normal distribution with the same mean $\mu = 6$, and standard deviation $\sigma = \sqrt{6}$.

```
curve(dnorm(x, 6, sqrt(6)), xlab = "Number of messages",
      ylab = "pdf", xlim = c(0, 20)) #normal pdf
lines(x, dpois(x, 6), type = "h")
```

This gives Fig. 18.10.

It appears from Fig. 18.10, that the normal pdf fits this Poisson pdf reasonably closely. Like the binomial, the Poisson variable is discrete with integer values only. When we are comparing probabilities, we calculate the normal probabilities in the range ± 0.5 above and below the Poisson integer, comparing $P(X = x)$ in the Poisson distribution with $P(x - 0.5 < X < x + 0.5)$ in the normal distribution. In R

```
x <- seq(0, 12, 1)
x1 <- dpois(x, 6) #Poisson probabilities for x
x2 <- pnorm(x+0.5, 6, sqrt(6))- pnorm(x - 0.5, 6, sqrt(6))
#normal densities in the interval (x - 0.5, x + 0.5)
y <- data.frame(round(x1, 3), round(x2, 3))
#round to 3 decimal places
Y
```

gives the values shown in Table 18.3

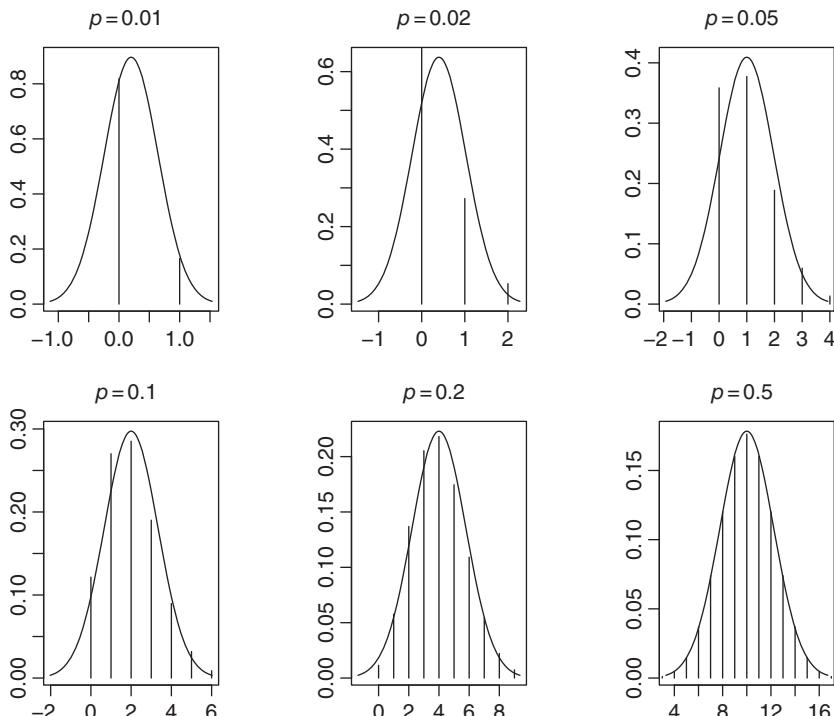


FIGURE 18.9 The Normal Approximation to the Binomial Distribution with $n = 20$ and Varying Values of p

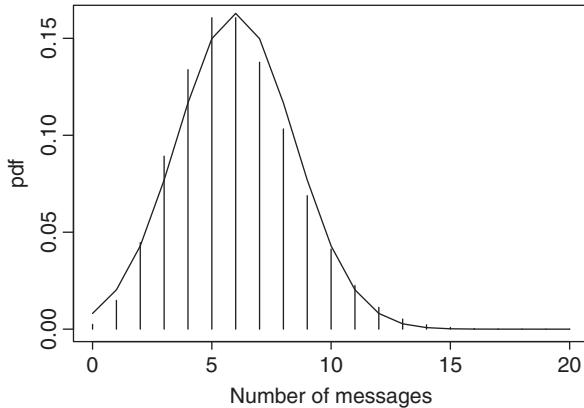


FIGURE 18.10 Normal Approximation to the Poisson distribution with $\lambda = 6$

TABLE 18.3 Comparison of Normal and Poisson Probabilities with $\lambda = 6$

Number of Messages X	Poisson $P(X = x)$	Normal $P(x - 0.5 < X < x + 0.5)$
0	0.002	0.008
1	0.015	0.021
2	0.045	0.043
3	0.089	0.077
4	0.134	0.116
5	0.161	0.149
6	0.161	0.162
7	0.138	0.149
8	0.103	0.116
9	0.069	0.077
10	0.041	0.043
11	0.024	0.021
12	0.011	0.008

Let us look at the approximation for varying values of λ . Figure 18.11 gives the Poisson pdfs with $\lambda = 2, 4, 6, 8, 10$, and 12, from which we can see that as λ increases, the normal approximation becomes closer.

Fig. 18.11 is generated in R using the `for` loop.

```
par(mfrow= c (2,3))
x <- 0:20
for(i in seq(from = 2, to = 12, by = 2)) {
```

```

lambda = i
curve(dnorm(x, lambda, sqrt(lambda)),
      lambda - 3*sqrt(lambda), lambda + 3*sqrt(lambda),
      main = bquote(lambda == .(i)),
      xlab = " ", ylab = " ")
lines(x, dpois(x, lambda), type = "h")
}

```

The `for(i in seq(from = 2, to = 12, by = 2))` statement repeats the command between the brackets {} for values of i from 2 to 12 in increments of 2, that is, for $i = 2, 4, 6, 8, 10$, and 12. The `bquote` writes the Greek letter λ , and sets it equal to the part enclosed in the brackets (). `indexgloops` in *R*

Example 18.4

Over a period of time, the number of hits to a high volume website arriving with a Poisson distribution averaged 10,000 a day.

Let us look at some probabilities calculated with the Poisson and normal distributions.

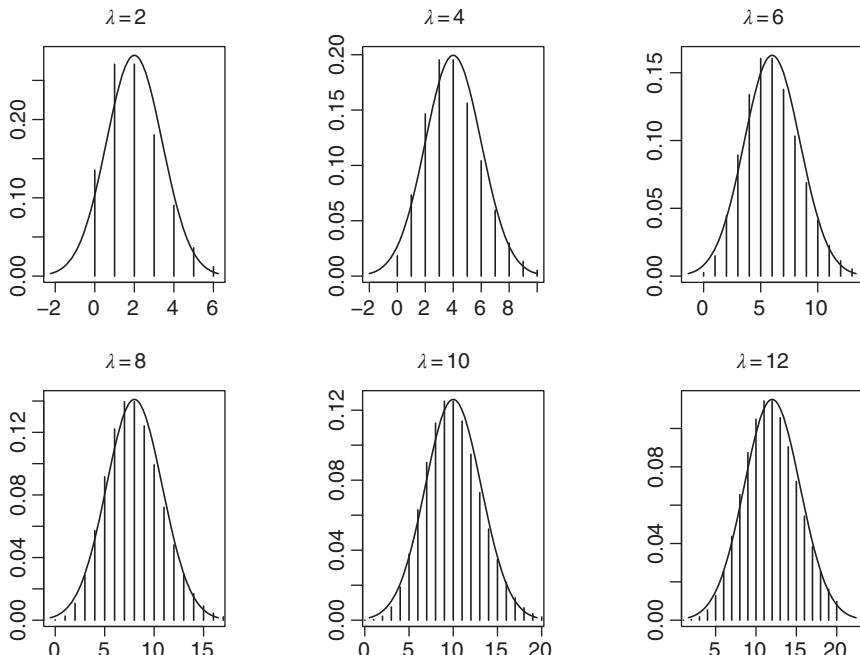


FIGURE 18.11 Normal Approximation to the Poisson Distribution with Varying Values of λ

1. The probability of more than 10,000 hits in a day

$$\begin{aligned}P(X > 10000) &= 0.4973404 \text{ using } 1\text{-ppois}(10000, 10000) \\&= 0.4980053 \text{ using } 1\text{-pnorm}(10000+0.5, 10000, 100)\end{aligned}$$

2. The probability of less than 9,900 hits in a day

$$\begin{aligned}P(X < 9900) &= 0.1598712 \text{ using ppois}(9900, 10000) \\&= 0.1598681 \text{ using pnorm}(9900+0.5, 10000, 100))\end{aligned}$$

3. The value, k , such that the probability that the number of hits in a day exceeds k is 0.01.

Choose k so that $P(X > k) = 0.01$

Equivalently, choose k so that $P(X \leq k) = 0.99$, which you will recognize as the quantile function.

$$\begin{aligned}k &= 10\,233 \text{ using qpois}(0.99, 10000) \\&= 10\,232.63 \text{ using qnorm}(0.99, 10000, 100)\end{aligned}$$

4. The expected number of days in a year in which the number of hits exceeds 10,300 hits.

$$\begin{aligned}P(X > 10\,300) &= 0.001386718 \text{ using 1-ppois}(10300, 10000) \\&= 0.001372224 \text{ using 1-pnorm}(10300-0.5, 10000, 100)\end{aligned}$$

Each rounds up to 0.14% of the 365 days, $0.0014 \times 365 \approx 0.5$, which means that, on average, a half a day each year, the number of hits to the website will exceed 10,300. In other words, hits will exceed 10,300 approximately once every two years.

Again, notice the closeness of the two sets of probabilities.

Superimposing the normal curve on the Poisson pdf with R

```
x <- seq(9600, 10400, 10)
curve (dnorm(x, 10000, 100), xlab = "Number of hits", ylab = " ")
lines(x, dpois(x, 10000), type = "h") #Poisson pdf
```

generates Fig. 18.12, what appears to be a perfect fit. △

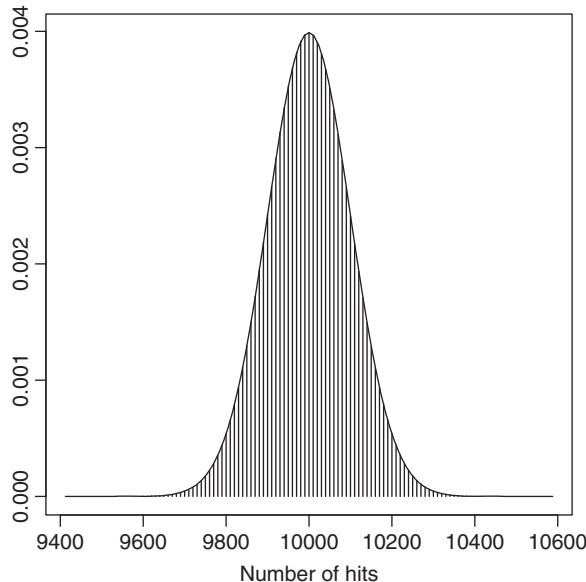


FIGURE 18.12 The $N(10,000, 100)$ pdf Superimposed on the Poisson Probabilities with $\lambda = 10,000$

18.5.3 The Central Limit Theorem

The convergence to normality of the binomial and the Poisson distributions are examples of what is known as the *central limit theorem*, which states that the distribution of the sum of n independent random variables tends to the normal distribution as $n \rightarrow \infty$.

Before the arrival of powerful computing, these approximations were essential to the calculations of binomial and Poisson probabilities, which became unwieldy when n was large. This problem no longer exists today with the easy availability of computing systems such as *R*, which, as we have seen, is able to perform the exact results as quickly as the normal approximations. However, no discussion of probability would be complete without a mention of such a famous result as the *central limit theorem*.

EXERCISES 18.1

1. An analogue signal received at a detector, measured in microvolts, is normally distributed with mean of 200 and variance of 256.
 - (a) What is the probability that the signal will exceed $224 \mu\text{V}$?
 - (b) What is the probability that it will be between 186 and $224 \mu\text{V}$?
 - (c) What is the micro voltage below which 25% of the signals will be?

- (d) What is the probability that the signal will be less than $240 \mu\text{V}$, given that it is larger than $210 \mu\text{V}$?
- (e) Estimate the interquartile range.
- (f) What is the probability that the signal will be less than $220 \mu\text{V}$, given that it is larger than $210 \mu\text{V}$?
- (g) If we know that a received signal is greater than $200 \mu\text{V}$, what is the probability that it is in fact greater than $220 \mu\text{V}$?
2. A manufacturer of a particular type of computer system is interested in improving its customer support services. As a first step, its marketing department has been charged with the responsibility of summarizing the extent of customer problems in terms of system failures. Over a period of six months, customers were surveyed and the amount of downtime (in minutes) due to system failures they had experienced during the previous month was collected. The average downtime was found to be 25 minutes and a variance of 144. If it can be assumed that downtime is normally distributed:
- (a) obtain bounds which will include 95% of the downtime of all the customers;
- (b) obtain the bound above which 10% of the downtime is included.
3. Suppose the scores, X , in an end-of-year programming examination are normally distributed with a mean of 50% and a standard deviation of 10. Students are deemed to have passed if they obtain at least 40% and less than 50%. A second class honors grade two is awarded to students obtaining at least 50% and less than 60%; a second class honors grade one is awarded to students obtaining at least 60% and less than 70%. Students obtaining 70% or more are awarded a first class honors grade. Three hundred students are about to sit the examination.
- (a) Estimate the number likely to fall into each grade, including the number who fail.
- (b) Students who exceed 75% are given free accommodation in the students' residences the following year. Estimate the number of students who are likely to be rent free.
- (c) If rents are 600 euro a month, what is the expected loss of monthly revenue due to rent-free students?
- (d) Obtain the quartiles of the distribution of marks.

18.6 PROJECTS

1. Experiment yourselves with the *dnorm* function to generate various normal distributions with differing means and standard deviations.
2. In Fig. 18.7, we compared the binomial distributions with $p = 0.2$ and different values of n with the appropriate normal distribution, and output the graphs in a

two by three matrix array. Redo this example for $p = 0.1, 0.2, 0.3, 0.4, 0.5$, as well as varying n .

3. Use R to illustrate the normal approximation to the binomial and Poisson distributions.
 - (a) In the case of the binomial, decide on how big n and p should be for this approximation to work.
 - (b) In the case of the Poisson distribution, decide on how big λ should be for this approximation to work.

In each case, develop a rule of thumb for moving from the actual distribution to the normal approximation.

19

PROCESS CONTROL

Process control is concerned with keeping a manufacturing process at a specified stable level. Products that do not meet the manufacturer's specification are unacceptable; they are produced because of changes in the manufacturing process or random variations in the process. To maintain and monitor the quality of the products, samples are taken at regular intervals for inspection. If an unsatisfactorily high number of unacceptable products are observed, some action is taken.

A process that has been running at an acceptable level may suddenly or gradually move off target. The objective of process control is to detect these changes as soon as possible so that appropriate action may be taken.

19.1 CONTROL CHARTS

The most commonly used technique for detecting changes in a manufacturing process is the *control chart*. A variable that is characteristic of the quality of the process is plotted against time. For example, a machine may be set to produce items to a certain dimension, called the target value, and usually designated by T . In any process, be it automatic or manual, there will be variations in the dimensions from item to item, even if the target value remains the same, that is, each measurement will have a random variation. The object of control charting is to detect when this variation is “more than random,” when it, in fact, is due to a shift in the target.

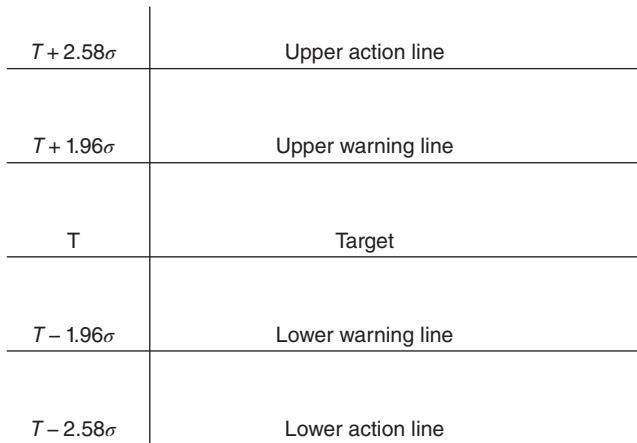


FIGURE 19.1 A Typical Control Chart

A control chart is a means of monitoring the process to ensure that it remains under control and that the nature of the product has not changed over time.

A typical control chart is given in Fig. 19.1.

The chart in Fig. 19.1 is constructed by first drawing a horizontal line through the target value T . Then, a pair of parallel lines above and below T are drawn at $T + 1.96\sigma$ and $T - 1.96\sigma$. These are called the *warning lines*, and are so placed to ensure that 95% of the components inspected should fall within these limits if the process is in control. Just 1 in 20 inspections on average is expected to fall outside these limits; if this happens we should be warned, but not required to take any action just yet.

Finally, a pair of parallel lines, 2.58 standard deviations above and below T , that is, at $T \pm 2.58\sigma$ are drawn. Just 1 in 100 observations is expected to lie outside these parallel lines. These lines are called the *action lines*; when an observation is outside these lines, there is a strong indication that the process has changed, and moved off target; some action is required. The action may consist of, for example, resetting the machine, or removing a faulty part.

To construct the control chart given in Fig. 19.1, the standard deviation σ needs to be known. If it is not, then it would be necessary to run the process for a while to obtain a measure of variability.

Obviously, control charts can only control the quality of the items within the capability of that process. For example, if the action lines $\mu \pm 2.58\sigma$ are too wide to meet design tolerances, the process needs to be tightened, otherwise, unacceptable products will inevitably get though without alert.

Example 19.1

Suppose components are manufactured to a target diameter 20 cm. An earlier process capability assessment has shown that the standard deviation of the diameter is 0.5 cm. ◀

At the end of each day, a component is chosen at random from the day's production, and the diameter is measured to ensure that the process has not moved off target. The following data were obtained on the first 30 days of production.

19.21, 19.96, 19.29, 19.21, 19.89, 21.10, 20.12, 19.88, 20.41, 19.39, 19.84, 20.42, 19.44, 20.30, 19.92, 20.84, 19.37, 19.66, 20.76, 20.13, 19.77, 20.03, 19.44, 19.63, 19.90, 19.45, 21.09, 20.42, 19.69, 19.23

To examine these data using *R*, first set the observations into a vector called *data*, that is, *data* <- c(19.21, 19.96 ···, 19.23), and then plot the data and the appropriate action and warning lines.

```
plot(data, xlab = "Day", ylab = "Diameter", ylim = c(18, 22))
abline(h = 20)
abline(h = 20 + 1.96 * 0.5)
abline(h = 20 - 1.96 * 0.5)
abline(h = 20 + 2.58 * 0.5)
abline(h = 20 - 2.58 * 0.5)
text(15, 19.7, "Target")
text(15, 21.4, "Upper action line")
text(15, 20.6, "Upper warning line")
text(15, 19.2, "Lower warning line")
text(15, 18.6, "Lower action line")
```

gives Fig. 19.2.

Notice from Fig. 19.2 that all observations are within the action lines. There were two warnings, one on day 6, and the second on day 27, but subsequent measurements

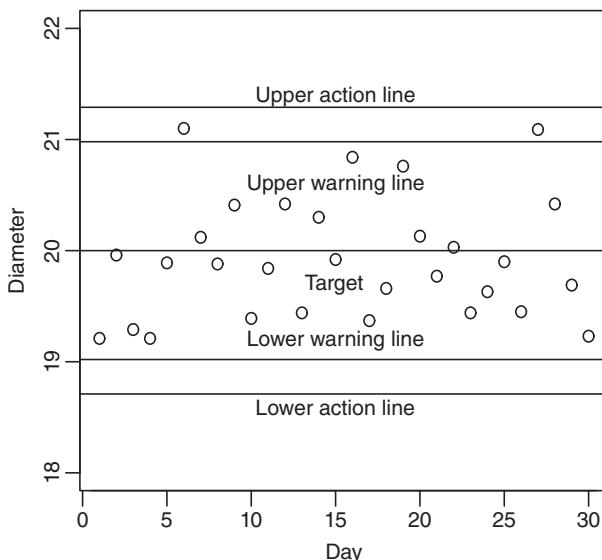


FIGURE 19.2 A Control Chart for Diameters: Day 1–30

indicated that there was nothing to be worried about. So the process can be said to be under control at this stage.

Suppose we now let this run for a further 30 days with the following results.

20.31, 20.55, 20.52, 19.77, 19.49, 19.90, 19.77, 20.26, 20.06, 20.67, 19.80, 20.91, 19.70, 19.90, 20.44, 19.80, 19.13, 19.60, 20.06, 20.19, 20.58, 20.30, 20.22, 20.74, 20.57, 20.40, 20.74, 20.82, 20.82, 20.54

Using the same procedure as before to construct a control chart, set the observations into a vector `data2 <- c(20.31, 29.55, ... , 20.54)`. This time you will also need to specify `day <- 31:60`. Then write in R,

```
plot(day, data2, xlab = "Day", ylab = "Diameter", ylim = c(18, 22))
abline(h = 20)
abline(h = 20 + 1.96 * 0.5)
abline(h = 20 - 1.96 * 0.5)
abline(h = 20 + 2.58 * 0.5)
abline(h = 20 - 2.58 * 0.5)
text(45, 21.4, "Upper action line")
text(45, 18.6, "Lower action line")
text(45, 20.6, "Upper warning line")
text(45, 19.2, "Lower warning line")
```

to get Fig. 19.3.

Now, these data appear to be in control because all of the measurements are within the action lines. Notice, however, that the measurements for the last 12 days are all

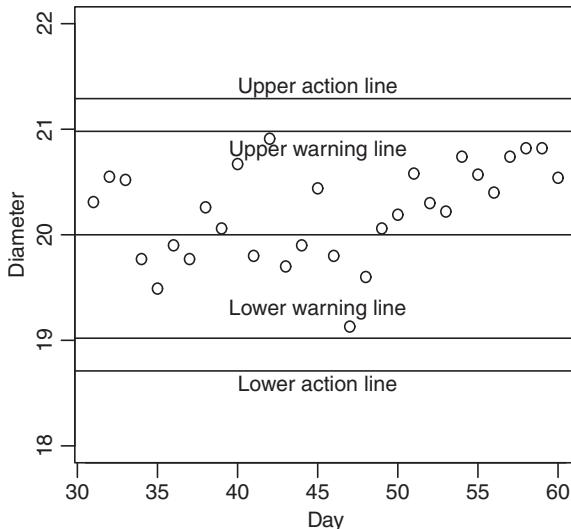


FIGURE 19.3 A Control Chart for Diameters: Day 31–60

above the target. Something is amiss, but this control chart does not alert us to the problem; in fact, it does not even give a warning.

This example illustrates that control charts are rather crude in that small changes in the target are often obscured by the residual variation. This is because each measurement is independently compared with the control limits, whereas it would be sensible to combine successive results. Because observations are treated independently, the chart is not sensitive to small changes in the product. For example, if the process moves a distance σ above the target, there is still a high probability that the measurements remain below the upper action line: a 94% chance in fact, which means that an average of over 16 results have to be taken before one observation falls above the action line, and the change is detected. Bear in mind that an observation may consist of examining the day's production, so the change may go undetected for 16 days, more than a fortnight. This means that, with this type of control chart, small changes in the process may go undetected for a long while.

An alternative is to use, what are known as *cusum charts*, which, as we shall see in the next section, accumulate the errors in the individual observations and are more sensitive to changes in the process.

19.2 CUSUM CHARTS

A cusum chart plots the sum of the deviations from the target value of the observations, up to and including time t , against t .

If we denote d_i as the diameter measurement on day i , and T is the target value, then the sum of deviations about T of each d_i is accumulated up to and including day t :

$$S_t = \sum_{i=1}^t (d_i - T)$$

A plot of S_t against t gives what is called the **cumulative sum** chart (*cusum*). In R, the function `cumsum` (*cumulative summation*) obtains the cumulative sums. Writing

```
d <- data2 - 20
S <- cumsum(d)
plot(day, S, xlab = "Day", ylab = "Cumulative sum of the deviations")
```

gives Fig. 19.4

With the *cusum chart*, it is easy to see that the change in the pattern begins to happen after day 49: from then on, the cumulative sums increase steadily. Clearly *cusum chart* has greater sensitivity than the ordinary control chart.

Repeating this analysis for the “in control” data given in Fig. 19.1, we get Fig. 19.5.

In Fig. 19.5, we see that the cusum chart gives no cause for concern. There is no pattern of successive increases or decreases.

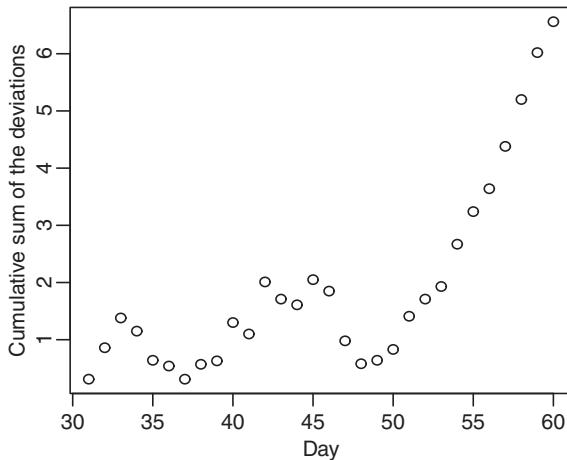


FIGURE 19.4 A Cusum Chart for Data that are Going Out of Control

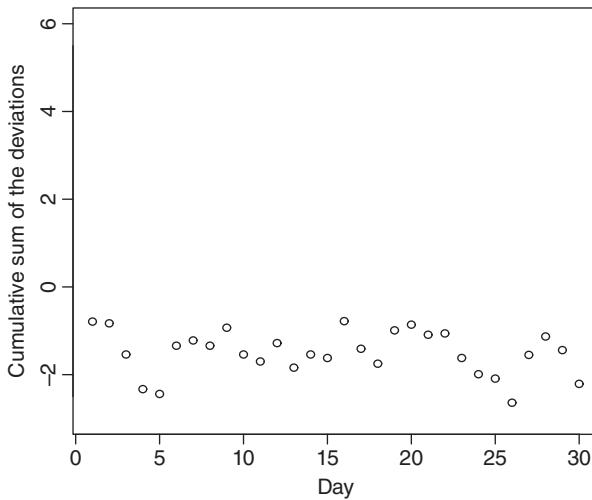


FIGURE 19.5 An ‘In-control’ Cusum Chart

19.3 CHARTS FOR DEFECTIVE RATES

When controlling the number of defectives in a batch of size n , for example, the number of defective memory sticks in a day’s production, or the number of defects per wafer, it is usual to plot only upper limits, and monitor whether or not the defective rate has increased. It is also usual to take the “1 in 1000” and “1 in 40” as the action and warning lines, respectively. These are obtained using the quantile function in *R*.

For the standard normal distribution, a 1 in a 1,000 rate is obtained with

```
qnorm(0.999, 0, 1)
[1] 3.090232
```

which means that 3.09 cuts the standard normal distribution into 99.9% below, and 0.1% above. It is common to use 3 instead of 3.09.

A 1 in 40 rate is obtained with

```
qnorm(0.975, 0, 1)
[1] 1.959964
```

and this is usually rounded to 2.

Denote the allowable tolerable error rate by ATE, then, using the normal approximation to the binomial, the upper bounds for the number of defectives in a large batch of size n is as follows:

$$\begin{aligned} n(\text{ATE}) + 3\sqrt{n(\text{ATE})(1 - \text{ATE})} &\quad \text{upper action line} \\ n(\text{ATE}) + 2\sqrt{n(\text{ATE})(1 - \text{ATE})} &\quad \text{upper warning line} \end{aligned} \quad \triangleleft$$

Example 19.2

At the end of each production run, memory sticks are checked for bad sectors. Each production run consists of 1,000 chips, and the allowable tolerable error rate is 10%. The following are the number of memory sticks with bad sectors observed in the last 10 production runs:

Run Number:	1	2	3	4	5	6	7	8	9	10
Bad Sticks:	90	101	97	96	99	105	104	92	106	95

Construct the appropriate control chart to decide if the process is in control.

In R

```
data <- c(90,101, 97, 96, 99, 105, 104, 92, 106, 95)
plot(data, xlab = "Run number",
      ylab = "Number of bad sectors", ylim = c(90, 110))
abline(h = 100)
abline(h = 100 + 2*sqrt(100* 0.1 * 0.9))
abline(h = 100 + 3*sqrt(100* 0.1 * 0.9))
text(5, 110, "Upper Action Line")
text(5, 107, "Upper Warning Line")
text(5, 101, "Allowable Tolerable Error Number")
```

which gives Fig. 19.6.

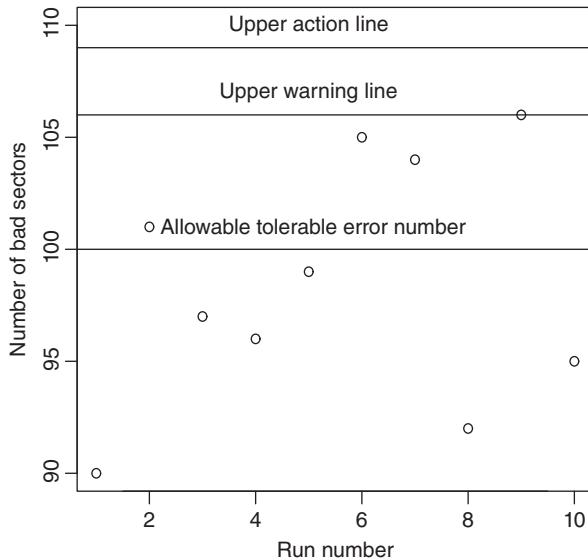


FIGURE 19.6 A Control Chart for Bad Sectors in Memory Sticks

A similar approach is used when monitoring a Poisson process to ensure its rate λ has not increased.

$$\lambda + 3\sqrt{\lambda} \quad \text{upper action line}$$

and

$$\lambda + 2\sqrt{\lambda} \quad \text{upper warning line} \quad \Delta$$

Example 19.3

Wafer yield is the number of good chips produced per wafer. It is desired to ensure that the yield is such that the number of defective chips remains at an average of five per wafer. The number of defective chips observed in the last 12 wafers produced is

Wafer Number:	1	2	3	4	5	6	7	8	9	10	11	12
Defective Chips:	4	1	7	6	7	6	4	2	6	9	10	12

To check if the process is under control, we plot the data on a control chart using *R*.

```
data <- c(4, 1, 7, 6, 7, 6, 4, 2, 6, 9, 10, 12)
plot(data, xlab = "Wafer number", ylab = "Number of defective chips")
abline(h = 5)
text(6, 5.5, "Target")
abline(h = 5 + 2*sqrt(5))
text(6, 10, "Warning")
abline(h = 5 + 3*sqrt(5))
text(6, 12, "Action")
```

which gives Fig. 19.7.

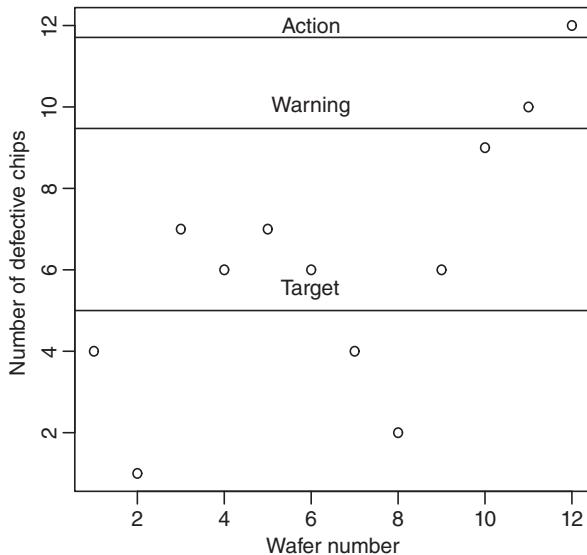


FIGURE 19.7 A Control Chart for Defective Chips in Wafers

From this chart, we see that the process has gone out of control, and action is required to improve the quality. \triangleleft

EXERCISES 19.1

1. Printed circuit boards are manufactured to a target thickness of 2 mm. The allowed variability is $\sigma = 0.3$ mm. Every 15 minutes, a circuit board is measured to ensure the process is on target.

The following data were obtained at the end of one day's production:

1.9, 2.0, 1.8, 2.4, 0.9, 1.5, 2.0, 1.8, 1.4, 2.1, 2.5, 1.6, 2.5, 2.5,
2.1, 1.3, 2.0, 2.6, 1.6, 1.8, 1.7, 2.5, 2.7, 2.5, 2.2, 2.7, 2.6, 2.4

- (a) Construct a control chart for this process.
 - (b) Plot the data on this chart, to decide whether or not the process needs to be readjusted before starting the following day's production.
2. Redo the previous example by using a CUSUM chart. Is the conclusion the same?
3. Assembled workstations are subjected to a final inspection for surface defects. The number of defects in the last 20 workstations produced are

0, 3, 5, 8, 5, 7, 4, 7, 6, 5, 5, 7, 7, 6, 4, 4, 3, 4, 2, 4

The allowable number of surface defects is 6. Is the process in control?

19.4 PROJECT

With your own computing experience, develop a front end to *R* which provides the user with an opportunity to set up a process control chart by clicking an icon on the screen. Indicate clearly the warning and action lines. The user should be able to enter new values on the screen, and an alert should be given if the process is out of control or going out of control.

PART V

TAILING OFF

20

THE INEQUALITIES OF MARKOV AND CHEBYSHEV

Our journey is tailing off. Hopefully, by now you have a good grasp of *R*, and appreciate how useful a tool it is for understanding probability.

Along the way, some important probability distributions have been introduced. In the discrete case, we examined the geometric, hypergeometric, binomial, and Poisson distributions, and, in the continuous case, the uniform, exponential, and normal distributions. With *R*, we were able to obtain probabilities, examine upper and lower bounds, calculate the quantiles and generate random numbers from these distributions using:

- the “d” function to obtain the density or point probabilities;
- the “p” function to calculate the cumulative probabilities or the distribution function;
- the “q” function to calculate the quantiles;
- the “r” function to generate random numbers (pseudo).

We remember the *R* names for these functions: for example *dbinom*, *pbinom*, *qbinom*, and *rbinom* for the binomial, and for the normal *dnorm*, *pnorm*, *qnorm*, and *rnorm*.

Up to now, probabilities were arrived at by assuming that the random variable followed a particular distribution, or by examining past data to obtain an approximate

model. But what if the distribution of the random variable is unknown and there are no past data? In this the final chapter, we introduce tools for estimating tail probabilities, when the information on the random variable is limited: the Markov and Chebyshev inequalities.

20.1 MARKOV'S INEQUALITY

The Markov inequality shows how to obtain probability bounds when the mean, but nothing else, is known.

Example 20.1

The purchase of a computer system is being considered. Regarding retrieval time, X , the only information the manufacturer can give is the expected time to retrieve a record from a direct access storage system.

$$E(X) = 100 \text{ ms}$$

What is the probability that the retrieval time will exceed 150 milliseconds?

To answer this question, we need Markov's inequality. \triangleleft

Theorem 20.1

Markov's inequality

If X is a random variable with expected value $E(X)$, then, for any $k > 0$,

$$P(X \geq k) \leq \frac{E(X)}{k} \quad (20.1)$$

Hence,

$$P(X < k) \geq 1 - \frac{E(X)}{k}. \quad (20.2)$$

\square

As you can see, this inequality gives the tail bound of a nonnegative random variable, when all we know is its expectation or mean. What it says is, that for any value $k > 0$, no matter how large, the probability that the random variable is greater than or equal to k is less than or equal to the expected value divided by k . Before offering a proof, let us apply Markov's inequality to Example 20.1, in which $E(X) = 100$.

$$P(X \geq 150) \leq \frac{100}{150} = \frac{2}{3}$$

The probability that the retrieval time is greater than or equal to 150 is less than or equal to $2/3$.

Taking $k = 200$, we have

$$P(X \geq 200) \leq \frac{100}{200} = \frac{1}{2}$$

So there is at most a 50% chance that the time to retrieve a record will be greater than or equal to 200. Equivalently, there is more than a 50% chance that the time is less than twice the mean.

By substituting $k = 300$ into Markov's inequality, we could also calculate

$$P(X \geq 300) \leq \frac{100}{300} = \frac{1}{3}$$

that is, the probability that X is greater than three times the mean is less than 1/3.

In general, for any random variable $X > 0$, for which $E(X) = \mu$, by letting $k = 2\mu$ in Markov's inequality gives

$$P(X \geq 2\mu) \leq \frac{\mu}{2\mu} = \frac{1}{2}$$

and taking $k = 3\mu$, we have

$$P(X \geq 3\mu) \leq \frac{\mu}{3\mu} = \frac{1}{3}$$

Proof of Markov's Inequality

First, break the sample space S of X into $X \geq k$ and $X < k$, and define

$$A = \{x \in S | X(x) \geq k\}$$

$$B = \{x \in S | X(x) < k\}$$

Discrete Case	Continuous Case
$E(X) = \sum_{x \in S} xp(x)$	$\int_{x \in S} xf(x)d(x)$
$= \sum_{x \in A} xp(x) + \sum_{B} xp(x)$	$\int_{x \in A} xf(x)dx + \int_{x \in B} xf(x)dx$
$\geq \sum_{x \in A} xp(x)$	$\int_{x \in A} xf(x)dx$
$\geq \sum_{x \in A} kp(x)$	$\int_{x \in A} kf(x)dx$
$= k \sum_{x \in A} p(x)$	$k \int_{x \in A} f(x)dx$
$= kP(X \geq k)$	$kP(X \geq k)$

In both the discrete and continuous cases,

$$E(X) \geq kP(X \geq k)$$

that is,

$$P(X \geq k) \leq \frac{E(X)}{k}$$

Markov's inequality gives the best bound possible for a nonnegative random variable when all that known is the mean, but by its nature it is conservative. In Appendix F, we offer an alternative pictorial illustration of Markov's inequality.

In the following examples, we compare the tail probability obtained from Markov's inequality with the tail probabilities that would be obtained if the distribution of the random variable were known. \square

Example 20.2

The average number of system crashes in a week is 2.

$$E(X) = 2$$

- (a) Estimate the probability that there will be three or more crashes in any one week.
- (b) What would the probability be, if it were known that the system crashes follow a Poisson distribution?

Solution

- (a) Applying Markov's inequality with $\mu = 2$ and $k = 3$.

$$P(X \geq 3) \leq \frac{2}{3}$$

The probability that there will be three or more crashes is less than or equal to $2/3$.

- (b) Suppose we know that crashes follow a Poisson distribution with two per week on average. The Poisson pdf is

$$P(X = x) = e^{-\lambda} \frac{\lambda^x}{x!}$$

We have shown in Chapter 13 that the mean $E(X) = \lambda$. Since, in this case, $\lambda = 2$, the pdf of the specific Poisson distribution is

$$P(X = x) = e^{-2} \frac{2^x}{x!}$$

For $P(X \geq 3) = 1 - P(X \leq 2)$, write in

```
1 - ppois(2, 2)
[1] 0.3233236
```

The exact probability is 0.3233236, while Markov's approximation at 0.66 is more than double. \triangleleft

The conservativeness of Markov's bound is not surprising since, as pointed out already, it is based on a knowledge of the mean only, nothing else.

Example 20.3

A new computer system is proposed. The manufacturer can give the information that the mean response time is

$$E(T) = 10 \text{ seconds.}$$

- (a) Estimate the probability that the response time will be more than 20 seconds.
- (b) What would the probability be, if it is known that the response time is exponentially distributed with the same mean?

Solution

- (a) With $\mu = 10$, and $k = 20$, we can say that

$$P(T \geq 20) \leq \frac{10}{20} = 0.5$$

There is at most a 50% chance that the response time will be greater than or equal to 20 seconds.

- (b) Now, suppose it were known that the response time is exponentially distributed:

$$f(t) = \lambda e^{-\lambda t}$$

Since $E(T) = 1/\lambda = 10$, then $\lambda = 0.1$. So the pdf of the specific exponential distribution that we are dealing with is

$$f(t) = (0.1)e^{-(0.1)t}$$

We can obtain $P(T \geq 20)$ in R with

```
1 - pexp(20, 0.1)
[1] 0.1353353
```

The exact probability, when we know the distribution is exponential, is 0.1353353; not surprisingly it is much less than the probability of 0.5 provided by the Markov inequality. \triangleleft

Example 20.4

A manufacturer knows that the mean diameter of a particular product is 10 mm, and wants to estimate the proportion of products that will be greater than or equal to 20 mm.

- (a) Use Markov's inequality to estimate $P(X \geq 20)$.
- (b) If the manufacturer also believes that the measurements are normally distributed with the same mean, but the variance is unknown, can the bound be tightened?

Solution

- (a) From Markov's inequality with $E(X) = 10$ and $k = 20$, we may write

$$P(X \geq 20) \leq \frac{10}{20} = \frac{1}{2}$$

- (b) Suppose that it is known that the measurements are normally distributed.

Recall `pnorm(x, mu, sigma)` calculates $P(X \leq x)$ in R. For the above example, we take $X = 20$, $\mu = 10$ and we vary σ .

```
sd <- seq(1, 200, 5)
plot(sd, 1 - pnorm(20, 10, sd), ylim = c(0, 1),
     xlab = "Standard deviation", ylab = "P(X >= 20)")
abline(h = 10/20)
text(100, 0.55, "Markov's bound")
```

Figure 20.1 gives $P(X \geq 20)$ for differing σ .

Notice from Fig. 20.1 that for the lower values of σ , $P(X \geq 20)$ is substantially less than the bound given by Markov's inequality, but when σ is large, $\sigma > 100$ for example, the probability gets near the Markov bound of 0.5. As σ increases, the tail probability tends to 0.5 but never exceeds it. What this suggests is that the Markov inequality will be accurate when applied to a random variable that deviates a lot from its expectation, and not so good when applied to data which have a small variance. \triangleleft

Suppose we want $P(X \geq 15)$, and we know that $\mu = 10$. From Markov's inequality

$$P(X \geq 15) \leq \frac{10}{15} = 0.667$$

Repeating the analysis in (b), this time with $k = 15$

```
sd <- seq(1, 200, 5)
plot(sd, 1 - pnorm(15, 10, sd), ylim = c(0, 1),
     xlab = "Standard deviation", ylab = "P(X >= 15)")
abline(h = 10/15)
text(100, 0.71, "Markov's bound")
```

we obtain Fig. 20.2.

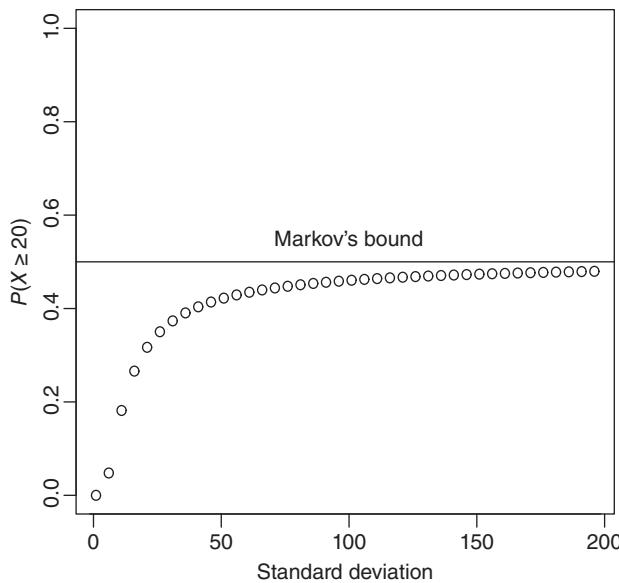


FIGURE 20.1 The Markov Bound when $E(X) = 10$, Together with the Tail Probabilities $P(X \geq 20)$ for Normal Distributions with $E(X) = 10$, and Differing Standard Deviations Ranging from 1 to 200

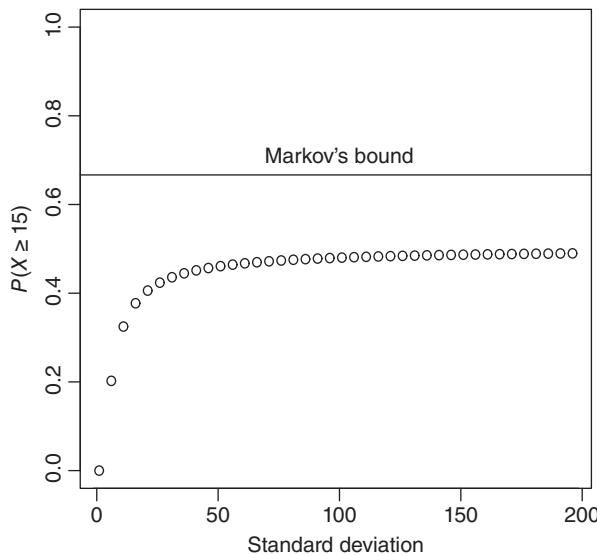


FIGURE 20.2 The Markov Bound with $E(X) = 10$, Together with the Tail Probabilities, $P(X \geq 15)$ for Normal Distributions with $E(X) = 10$, and Different Standard Deviations Ranging from 1 to 200

In Fig. 20.2, we see that the Markov bound is substantially higher than the actual probability obtained from the normal distribution, even when the standard deviation is as large as 200: $P(X \geq 15 | \sigma = 200)$ is

```
1 - pnorm(15, 10, 200)
[1] 0.4900275
```

In fact, with the normal distribution, with $E(X) = 10$, $P(X \geq 15)$ is always less than 0.5, since the normal distribution is symmetric about the mean. This example illustrates once more the conservativeness of Markov's inequality.

20.2 ALGORITHM RUNTIME

The Markov inequality is one of the major tools for establishing probability bounds on the runtime of algorithms. Suppose X is a random variable expressing the running time in seconds of an algorithm which consists of inputs of a certain size, and suppose also that we are able to compute the expected runtime $E(X) = 5$ seconds. Now this may be thought of as the order of magnitude that we would expect the time that the algorithm would take, but can we put an upper bound on the runtime? From a practical point of view, it would be important to know the likelihood that the runtime would be greater than 50, or 500, or 5,000 seconds.

From Markov's inequality

$$P(X \geq 50) \leq \frac{5}{50} = 0.1$$

$$P(X \geq 500) \leq \frac{5}{500} = 0.01$$

$$P(X \geq 5000) \leq \frac{5}{5000} = 0.001$$

Often, the running time depends on its input size, and different algorithms may have different running times. A computer program may take seconds, minutes, hours or even weeks to finish executing, depending on the number n of its inputs, and the effect n has on the runtime. If the expected runtime increases proportionally to n , we would say that the runtime is of the order of n written “runtime = $O(n)$.” Similarly, runtime of $O(n^2)$ means that it increases proportionally to n^2 . We can use Markov's inequality to tell us about the worst-case running time of the algorithm.

$$P(X \geq 2\mu) \leq \frac{1}{2}$$

There is less than a 50% chance that the runtime will be greater than twice the mean. Similarly

$$P(X \geq 3\mu) \leq \frac{1}{3}$$

and

$$P(X < 3\mu) \geq 1 - \frac{1}{3} = \frac{2}{3}$$

Runtime less than three times the expected value will occur in at least two thirds of all cases.

Example 20.5

Suppose a program looks up a specific entry in a sorted list of size n using a linear search algorithm, which has an expected runtime of $2n$ milliseconds. If the list contains 100 items, what is the probability the runtime of the algorithm will exceed 400 milliseconds?

Markov's inequality gives

$$P(X \geq 400) = P(X \geq 2\mu) \leq \frac{1}{2}$$

There is at most a 50% chance that the runtime will greater than or equal to 400 milliseconds. \triangleleft

20.3 CHEBYSHEV'S INEQUALITY

If, as well as the mean, the variance is known, a bound due to Chebyshev can be used, which is much stronger than that of Markov.

Theorem 20.2

Chebyshev's inequality

If X is a random variable with mean $E(X) = \mu$ and variance $E(X - \mu)^2 = \sigma^2$ then for any k

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \quad (20.3)$$

Hence,

$$P(|X - \mu| < k\sigma) \geq 1 - \frac{1}{k^2} \quad (20.4)$$

\square

Chebyshev's inequality gives a lower bound on the probability that a value of X is within $k\sigma$ units of the mean.

Chebyshev's inequality provides the best bound that is possible for a random variable when its mean and variance are known but its distribution is not. It is tighter than the bound obtained with the Markov inequality but obviously more conservative than the bound that would be obtained if we knew the form of the distribution that the random variable takes.

Before offering a proof, we consider some special cases. If $k = 2$, the inequality is

$$P(|X - \mu| < 2\sigma) \geq 1 - \frac{1}{2^2} = \frac{3}{4}$$

which means that three quarters of *any* probability distribution lies within two standard deviations from the mean.

With $k = 3$, the Chebyshev's equality states that

$$P(|X - \mu| < 3\sigma) \geq 1 - \frac{1}{3^2} = \frac{8}{9} = 0.89$$

showing that nearly 90% of *any* probability distribution lies within 3σ units of the mean.

Look again at Example 20.1, where a manufacturer claims that the mean retrieval time is 100 milliseconds. Suppose another manufacturer claims also that the mean retrieval time of the system is

$$\mu = E(X) = 100 \text{ milliseconds}$$

but in addition knows that the standard deviation of the retrieval time is

$$\sigma = 25 \text{ milliseconds}$$

If we want to estimate the probability that the retrieval time is within 50 milliseconds of the mean, that is,

$$P(50 < X < 150) = P(|X - 100| < 50)$$

we can apply Chebyshev's inequality with $k = 2$ and $\sigma = 25$, so that,

$$P(|X - \mu| < 2\sigma) \geq 1 - \frac{1}{2^2} = 0.75$$

which means that there is at least a 75% chance that the retrieval time is within 50 ms of the mean.

Example 20.6

Returning to Example 20.2, concerning the number of computer crashes in a week, suppose, as well as knowing that the average number of crashes in a week is 2, we also know that the standard deviation is 1.6.

$$\mu = 2$$

$$\sigma = 1.6$$

To estimate the probability that the number of crashes X in a week will be less than 4, use Chebyshev's inequality as follows:

We have

$$\begin{aligned} P(0 < X < 4) &= P(|X - 2| < 2) \\ &= P(|X - 2| < 1.25\sigma) \quad \text{since } \sigma = 1.6 \\ &\geq 1 - \frac{1}{(1.25)^2} = 0.36 \end{aligned}$$

The probability that the number of crashes in a week will be less than 4 is at least 36%. \triangleleft

Example 20.7

Memory sticks are packaged in boxes so that the average number per box is 200 with standard deviation $\sigma = 4$. Use Chebyshev's inequality to obtain a bound on the probability that boxes contain between 194 and 206 memory sticks.

We want

$$P(|X - 200| < 6) = P(|X - 200| < 1.5\sigma) \quad \text{since } \sigma = 4$$

Applying Chebyshev's inequality

$$P(|X - 200| < 1.5\sigma) \geq 1 - \frac{1}{(1.5)^2} = 1 - 0.4444 = 0.5556$$

which shows that at least 55.56% of the boxes will contain between 194 and 206 memory sticks.

Suppose we want to obtain the probability that the content of the boxes is within 10 items of the mean. We want

$$P(|X - 200| < 10) = P(|X - 200| < 2.5\sigma) \quad \text{since } \sigma = 4$$

Applying Chebyshev's inequality

$$P(|X - 200| < 2.5\sigma) \geq 1 - \frac{1}{(2.5)^2} = 1 - 0.16 = 0.84$$

shows that at least 84% of the boxes will contain between 190 and 210 memory sticks. \triangleleft

Proof of Chebyshev's Inequality

The proof of Chebyshev's inequality is similar to that of Markov's inequality. We break the sample space S into $|X - \mu| \geq k$ and $|X - \mu| < k$, and define

$$A = \{x | |X(x) - \mu| \geq k\sigma\}$$

$$B = \{x | |X(x) - \mu| < k\sigma\}$$

	Discrete Case	Continuous Case
$E(X - \mu)^2 =$	$\sum_{x \in S} (x - \mu)^2 p(x)$	$\int_{x \in S} (x - \mu)^2 f(x) dx$
$=$	$\sum_{x \in A} (x - \mu)^2 p(x) + \sum_{x \in B} (x - \mu)^2 p(x)$	$\int_{x \in A} (x - \mu)^2 f(x) dx + \int_{x \in B} (x - \mu)^2 f(x) dx$
\geq	$\sum_{x \in A} (x - \mu)^2 p(x)$	$\int_{x \in A} (x - \mu)^2 f(x) dx$
\geq	$\sum_{x \in A} k^2 \sigma^2 p(x)$	$\int_{x \in A} k^2 \sigma^2 f(x) dx$
$=$	$k^2 \sigma^2 \sum_{x \in A} p(x)$	$k^2 \sigma^2 \int_{x \in A} f(x) dx$
$=$	$k^2 \sigma^2 P(X - \mu \geq k\sigma)$	$k^2 \sigma^2 P(X - \mu \geq k\sigma)$

In both the discrete and the continuous cases,

$$\sigma^2 \geq k^2 \sigma^2 P(|X - \mu| \geq k\sigma)$$

that is,

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Equivalently,

$$P(|X - \mu| < k\sigma) \geq 1 - \frac{1}{k^2}$$

We show in Appendix F that Chebyshev's inequality is actually Markov's inequality in disguise. \square

Example 20.8

A manufacturer of computer systems surveyed customers regarding downtime of a specific system and found that the average downtime experienced was 25 minutes a month with variance 144.

- (a) How confident can the manufacturer be that the downtime will be within 15 minutes of the mean?
- (b) What would the confidence be, if it can be assumed that the downtime is normally distributed?

Solution

- (a) We want

$$P(|X - 25| < 15)$$

Using Chebyshev's inequality with $\mu = 25$ and $\sigma = \sqrt{144} = 12$, we can write

$$P(|X - 25| \geq 15) = P(|X - 25| \geq 1.25\sigma) \leq \frac{1}{(1.25)^2} = 0.64$$

Equivalently,

$$P(|X - 25| < 15) \geq 0.36$$

We are at least 36% confident that the downtime is within 15 minutes of its mean.

- (b) Now, if we know that the downtime is normally distributed, that is, $X \sim N(25, 12)$, to get the exact probability

$$P(|X - 25| < 15) = P(10 < X < 40)$$

write in R

```
pnorm(40, 25, 12) - pnorm(10, 25, 12)
[1] 0.7887005
```

\triangleleft

When the distribution is normal, there is a 78.9% probability that the downtime will be between 10 and 40 minutes, which is much greater than the 36% obtained from Chebyshev's inequality. Chebyshev's inequality gives the best estimate of a probability interval when all that is known is the mean and variance of the distribution. It can be viewed as giving a crude estimate of the probability. It is not surprising that it can be sharpened when the actual distribution becomes known.

EXERCISES 20.1

1. Daly Developers are marketing a new software. From past experience, new software from this company has on average 10 errors, that is, $\mu = 10$.
 - (a) What is the probability that the software package will contain less than 12 errors?
 - (b) If the standard deviation is known to be $\sigma = 1$, what is the probability that the software will contain between 8 and 12 errors?
2. If it takes an average of 10 seconds to download a certain file, what is the probability that the download will take at least than 12 seconds?
3. Network breakdowns occur on average once a month. Compute the probability that:
 - (a) no breakdowns will occur in any particular month;
 - (b) more than two breakdowns will occur in a month.
4. The average number of text messages received in a day on a smartphone is 15.
 - (a) Calculate the probability that, in any day, more than 20 messages will be received.
 - (b) What would the probability be, if it were known that text messages received followed a Poisson distribution?
5. The average time taken to repair a computer is 15 minutes. Use Markov's inequality to estimate the probability that the repair time will be at least 30 minutes.
6. For a particular Java assembler interface, the operand stack size is 4.
 - (a) Use Markov's inequality to estimate the probability that the stack size is at least 6.
 - (b) A different Java assembler interface is known to have the same mean, but the variance of the assembler is known to be 1.5. Its distribution is unknown. Use Chebyshev's inequality to obtain an interval that includes 95% of stack sizes of this assembler.
 - (c) Compare the results in (b) with what you would get if you knew that the distribution of the stack size were normal.

7. The mean time to retrieve a record from an online database is 200 milliseconds with a standard deviation of 58 milliseconds. The design criterion requires that at least 80% of all retrieval times must not differ from the mean by more than 75 milliseconds.
 - (a) Use Chebyshev's inequality to establish whether the design criterion is satisfied.
 - (b) Would the design criterion be satisfied if it were known that the retrieval time is normally distributed with a mean of 200 milliseconds and a standard deviation of 58 ms?

APPENDIX A

DATA: EXAMINATION RESULTS

gender	arch1	prog1	arch2	prog2
m	99	98	83	94
m	NA	NA	86	77
m	97	97	92	93
m	99	97	95	96
m	89	92	86	94
m	91	97	91	97
m	100	88	96	85
f	86	82	89	87
m	89	88	65	84
m	85	90	83	85
m	50	91	84	93
m	96	71	56	83
f	98	80	81	94
m	96	76	59	84
m	73	72	91	87
m	67	82	80	77
m	80	85	94	72
m	91	76	85	84
m	89	81	77	81
m	77	81	88	91
m	71	82	59	79
m	84	81	88	77

gender	arch1	prog1	arch2	prog2
m	95	83	92	63
m	3	87	56	76
f	95	65	63	82
f	NA	NA	91	65
m	59	79	73	82
m	95	83	49	69
m	80	80	87	72
m	97	92	98	96
m	81	89	41	57
m	77	70	51	71
m	69	74	83	68
m	82	79	57	45
f	85	66	56	67
m	87	68	56	78
m	88	76	47	61
m	83	76	41	65
m	51	67	49	79
f	76	63	57	76
m	88	64	48	53
m	61	53	54	61
m	83	60	56	49
m	90	78	81	50
m	40	67	53	68
m	92	61	47	64
m	76	69	44	59
m	72	61	62	56
f	77	53	48	60
m	58	52	50	73
m	63	62	40	48
m	48	73	74	53
m	40	75	43	52
m	40	40	48	62
m	75	67	40	45
f	49	61	49	44
m	54	47	43	52
m	56	55	44	55
m	75	40	40	51
m	64	86	50	81
f	88	40	43	83
m	82	66	51	63
m	73	64	28	54
f	59	28	60	51
m	74	57	45	61
m	45	69	35	40
m	70	52	40	43

gender	arch1	prog1	arch2	prog2
m	74	29	44	52
m	43	25	31	14
m	49	69	40	24
m	45	29	32	25
m	74	71	40	46
m	46	56	50	28
m	56	52	42	57
m	16	33	16	9
m	21	25	26	12
m	47	56	43	16
m	77	60	47	62
m	27	40	37	6
m	74	13	40	18
f	16	14	NA	NA
m	14	31	14	20
m	23	54	48	NA
m	83	76	58	75
f	NA	15	16	NA
m	45	40	40	61
m	40	28	26	9
m	48	27	23	16
m	91	89	6	73
f	50	27	22	11
m	77	82	45	65
m	49	49	36	31
m	96	84	48	29
f	21	29	25	5
m	61	40	34	11
m	50	19	41	NA
f	68	74	30	48
m	50	40	51	56
m	69	59	25	40
m	60	36	40	28
f	43	14	NA	NA
m	43	30	40	14
m	47	68	43	34
f	60	47	40	NA
m	40	68	57	75
m	45	26	38	6
m	45	31	NA	NA
f	31	21	32	8
m	49	12	24	14
m	87	40	40	32
m	40	76	49	17
f	8	29	15	14

gender	arch1	prog1	arch2	prog2
m	62	46	50	31
m	14	21	NA	NA
m	7	25	27	7
m	16	27	25	7
m	73	51	48	23
m	56	54	49	25
m	46	64	13	19

APPENDIX B

THE LINE OF BEST FIT: COEFFICIENT DERIVATIONS

We have a set of n pairs of observations $(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_n, y_n)$ and we want to find the line

$$y = a + bx$$

that “best fits” the observed data, in order to predict the value of y for new values of x . The variable x is said to be the *independent* variable, and y is called the *dependent* variable.

The goal is to find estimates of a and b that minimize the errors in predicting y from x . For any given value x_i , the predicted value of y is

$$\hat{y}_i = a + bx_i$$

Therefore, the error e_i is the difference between the observed and predicted y_i :

$$\begin{aligned} e_i &= y_i - \hat{y}_i \\ &= y_i - a - bx_i \end{aligned}$$

To get the line of best fit, choose a and b to minimize

$$S = \sum_{i=1}^n e_i^2$$

that is, minimize the sum, S , of the squared differences of each observed y_i value from its predicted value. Notice that we minimize the sum of the squared errors rather than the sum of the errors because $\sum_{i=1}^n e_i = 0$, the positive and negative e_i sum to zero.

$$\begin{aligned} S &= \sum_{i=1}^n e_i^2 \\ &= \sum_{i=1}^n (y_i - a - bx_i)^2 \end{aligned}$$

Using partial differentiation, a and b are chosen so that

$$\frac{\partial S}{\partial a} = 2 \sum_{i=1}^n (y_i - a - bx_i)(-1) = 0$$

and

$$\frac{\partial S}{\partial b} = 2 \sum_{i=1}^n (y_i - a - bx_i)(-x_i) = 0$$

from which we get two simultaneous equations:

$$\sum_{i=1}^n y_i = na + b \sum_{i=1}^n x_i \quad (\text{B.1})$$

$$\sum_{i=1}^n x_i y_i = a \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i^2 \quad (\text{B.2})$$

Multiplying Equation B.1 by $\sum_{i=1}^n x_i$, we get

$$\sum_{i=1}^n x_i \sum_{i=1}^n y_i = na \sum_{i=1}^n x_i + b \left(\sum_{i=1}^n x_i \right)^2 \quad (\text{B.3})$$

Multiplying Equation B.2 by n gives

$$n \sum_{i=1}^n x_i y_i = na \sum_{i=1}^n x_i + nb \sum_{i=1}^n x_i^2 \quad (\text{B.4})$$

By subtracting Equation B.3 from Equation B.4, we have

$$n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i = b \left(n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2 \right)$$

Therefore

$$b = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}$$

Also, by dividing Equation B.1 by n we get

$$\bar{y} = a + b\bar{x}$$

which means

$$a = \bar{y} - b\bar{x}$$

APPENDIX C

VARIANCE DERIVATIONS

Generally, for any random variable X ,

$$\begin{aligned}V(X) &= E(X - \mu)^2 = E(X^2 - 2\mu X + \mu^2) \\&= E(X^2) - 2\mu E(X) + \mu^2 \\&= E(X^2) - 2\mu^2 + \mu^2 \\&= E(X^2) - \mu^2\end{aligned}$$

where $\mu = E(X)$.

C.1 VARIANCE OF THE GEOMETRIC DISTRIBUTION (CHAPTER 10)

If X is a random variable having a geometric distribution with parameter $p = 1 - q$, then

$$V(X) = \frac{q}{p^2}$$

because

$$V(X) = E(X^2) - \frac{1}{p^2} \text{ since } \mu = \frac{1}{p}$$

Probability with R: An Introduction with Computer Science Applications, Second Edition. Jane M. Horgan.
© 2020 John Wiley & Sons, Inc. Published 2020 by John Wiley & Sons, Inc.
Companion website: www.wiley.com/go/Horgan/probabilitywithr2e

Writing $X^2 = X(X - 1) + X$, we have

$$\begin{aligned} E(X^2) &= E(X(X - 1) + X) \\ &= E(X(X - 1)) + E(X) \\ &= E(X(X - 1)) + \frac{1}{p} \end{aligned}$$

Now,

$$\begin{aligned} E(X(X - 1)) &= \sum_{x=1}^{\infty} x(x - 1)q^{x-1}p \\ &= 1 \cdot 0 \cdot p + 2 \cdot 1 \cdot qp + 3 \cdot 2 \cdot q^2p + 4 \cdot 3 \cdot q^3p + \dots \\ &= 2qp + 6q^2p + 12q^3p + \dots . \end{aligned}$$

Multiply by q to get

$$q \cdot E(X(X - 1)) = 2q^2p + 6q^3p + 12q^4p + \dots$$

Subtracting,

$$\begin{aligned} (1 - q)E(X(X - 1)) &= 2qp + 4q^2p + 6q^3p + \dots \\ &= 2q(p + 2qp + 3q^2p + \dots) \\ &= 2qE(X) \end{aligned}$$

recognizing $p + 2qp + 3q^2p + 4q^3p + \dots$ as the mean, $1/p$, of the geometric distribution.

From this, we have

$$pE(X(X - 1)) = (1 - q)E(X(X - 1)) = \frac{2q}{p}$$

So

$$E(X(X - 1)) = \frac{2q}{p^2}$$

which means that

$$E(X^2) = \frac{2q}{p^2} + \frac{1}{p}$$

and

$$V(X) = \frac{2q}{p^2} + \frac{1}{p} - \frac{1}{p^2}$$

$$\begin{aligned}
 &= \frac{2q + p - 1}{p^2} \\
 &= \frac{q}{p^2}
 \end{aligned}$$

C.2 VARIANCE OF THE BINOMIAL DISTRIBUTION (CHAPTER 11)

If X is a random variable having a binomial distribution with parameters n and $p = 1 - q$, then

$$V(X) = npq$$

because

$$V(X) = E(X^2) - (np)^2 \text{ since } \mu = np$$

$$\begin{aligned}
 E(X^2) &= E(X(X - 1) + X) \\
 &= E(X(X - 1)) + E(X) \\
 &= E(X(X - 1)) + np
 \end{aligned}$$

Now,

$$\begin{aligned}
 E(X(X - 1)) &= \sum_{x=0}^n x(x - 1) \binom{n}{x} p^x q^{n-x} \\
 &= \sum_{x=2}^n x(x - 1) \binom{n}{x} p^x q^{n-x} \\
 &= \sum_{x=2}^n n(n - 1) \binom{n - 2}{x - 2} p^x q^{n-x} \\
 &= n(n - 1)p^2 \sum_{x-2=0}^{n-2} \binom{n - 2}{x - 2} p^{x-2} q^{(n-2)-(x-2)} \\
 &= n(n - 1)p^2(p + q)^{n-2} \\
 &= n(n - 1)p^2 \\
 &= n^2p^2 - np^2
 \end{aligned}$$

which means that

$$E(X^2) = n^2p^2 - np^2 + np$$

and

$$V(X) = n^2 p^2 - np^2 + np - (np)^2 = -np^2 + np = np(1 - p) = npq$$

C.3 VARIANCE OF THE POISSON DISTRIBUTION (CHAPTER 13)

If X is a random variable having a Poisson distribution with parameter λ , then

$$V(X) = \lambda$$

because

$$V(X) = E(X^2) - \lambda^2 \text{ since } \mu = \lambda$$

$$\begin{aligned} E(X^2) &= E(X(X - 1) + X) \\ &= E(X(X - 1)) + E(X) \\ &= E(X(X - 1)) + \lambda \end{aligned}$$

Now

$$\begin{aligned} E(X(X - 1)) &= \sum_{x=0}^{\infty} x(x - 1)e^{-\lambda} \frac{\lambda^x}{x!} \\ &= e^{-\lambda} \sum_{x=2}^{\infty} x(x - 1) \frac{\lambda^x}{x!} \\ &= e^{-\lambda} \sum_{x=2}^{\infty} \frac{\lambda^x}{(x - 2)!} \\ &= e^{-\lambda} \lambda^2 \left(\sum_{x=2}^{\infty} \frac{\lambda^{x-2}}{(x - 2)!} \right) \\ &= e^{-\lambda} \lambda^2 \left(1 + \frac{\lambda}{1!} + \frac{\lambda^2}{2!} + \dots \right) \\ &= e^{-\lambda} \lambda^2 e^{\lambda} = \lambda^2 \end{aligned}$$

So

$$E(X^2) = \lambda^2 + \lambda$$

and

$$V(X) = E(X^2) - \lambda^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$$

C.4 VARIANCE OF THE UNIFORM DISTRIBUTION (CHAPTER 15)

If X is a random variable uniformly distributed in the interval $[a, b]$

$$V(X) = \frac{(b-a)^2}{12}$$

because

$$V(X) = E(X^2) - \left(\frac{a+b}{2}\right)^2$$

since $\mu = \frac{a+b}{2}$. Now

$$\begin{aligned} E(X^2) &= \frac{1}{b-a} \int_{x=a}^b x^2 dx \\ &= \frac{1}{b-a} \left(\frac{x^3}{3} \right)_a^b \\ &= \frac{b^3 - a^3}{3(b-a)} \end{aligned}$$

Then

$$\begin{aligned} V(X) &= \frac{b^3 - a^3}{3(b-a)} - \frac{(a+b)^2}{4} \\ &= \frac{b^2 + ab + a^2}{3} - \frac{b^2 + 2ab + a^2}{4} \\ &= \frac{1}{12}(4b^2 + 4ab + 4a^2 - 3b^2 - 6ab - 3a^2) \\ &= \frac{b^2 - 2ab + a^2}{12} = \frac{(b-a)^2}{12} \end{aligned}$$

C.5 VARIANCE OF THE EXPONENTIAL DISTRIBUTION (CHAPTER 16)

If X is a random variable having an exponential distribution with parameter λ , then

$$V(X) = \frac{1}{\lambda^2}$$

because

$$V(X) = E(X^2) - \frac{1}{\lambda^2} \text{ since } \mu = \frac{1}{\lambda}$$

Now, “integrating by parts,”

$$\begin{aligned} E(X^2) &= \int_0^\infty x^2 \lambda e^{-\lambda x} dx = - \int_0^\infty x^2 d(e^{-\lambda x}) \\ &= -(x^2 e^{-\lambda x})_0^\infty + \int_0^\infty e^{-\lambda x} d(x^2) \\ &= 0 + 2 \int_0^\infty x e^{-\lambda x} dx = \frac{2}{\lambda} E(X) = \frac{2}{\lambda^2} \end{aligned}$$

So

$$V(X) = \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}$$

APPENDIX D

BINOMIAL APPROXIMATION TO THE HYPERGEOMETRIC

If N is the population size, p the proportion of successes in the population, and X the number of successes in a sample of size n drawn without replacement from the population, show that as $N \rightarrow \infty$ in such a way that $n/N \rightarrow 0$, then for any $x \in [0, n]$

$$P(X = x) = \frac{\binom{Np}{x} \binom{N(1-p)}{n-x}}{\binom{N}{n}} \rightarrow \binom{n}{x} p^x q^{n-x} \quad (\text{D.1})$$

because

$$\begin{aligned}\binom{Np}{x} &= \frac{(Np)!}{x!(Np-x)!} \\ \binom{N(1-p)}{n-x} &= \frac{(N(1-p))!}{(n-x)!(N(1-p)-(n-x))!} \\ \binom{N}{n} &= \frac{N!}{n!(N-n)!}\end{aligned}$$

with $q = 1 - p$, we can write Equation D.1 as

$$\begin{aligned}
 P(X = x) &= \frac{(Np)!(Nq)!n!(N-n)!}{x!(Np-x)!(n-x)!(Nq-(n-x))!N!} \\
 &= \frac{n!}{x!(n-x)!} \frac{[Np(Np-1)\cdots(Np-x+1)][Nq(Nq-1)\cdots(Nq-(n-x)+1)]}{N(N-1)\cdots(N-n+1)} \\
 &= \frac{n!}{x!(n-x)!} \\
 &\quad \times \frac{\left[N^x p\left(p-\frac{1}{N}\right)\cdots\left(p-\frac{x-1}{N}\right)\right] \left[N^{n-x} q\left(q-\frac{1}{N}\right)\cdots\left(q-\frac{n-x-1}{N}\right)\right]}{N^n \left(1-\frac{1}{N}\right)\cdots\left(1-\frac{n-1}{N}\right)} \\
 &= \frac{n!}{x!(n-x)!} \frac{\left[p\left(p-\frac{1}{N}\right)\cdots\left(p-\frac{x-1}{N}\right)\right] \left[q\left(q-\frac{1}{N}\right)\cdots\left(q-\frac{n-x-1}{N}\right)\right]}{\left(1-\frac{1}{N}\right)\cdots\left(1-\frac{n-1}{N}\right)} \\
 &\rightarrow \binom{n}{x} p^x q^{n-x} \quad \text{as } N \rightarrow \infty
 \end{aligned}$$

which is the binomial distribution.

APPENDIX E

NORMAL TABLES



Area under the standard normal curve from $-\infty$ to z .

For example, $P(z \leq 1.42) = 0.9222$

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990

Probability with R: An Introduction with Computer Science Applications, Second Edition. Jane M. Horgan.

© 2020 John Wiley & Sons, Inc. Published 2020 by John Wiley & Sons, Inc.

Companion website: www.wiley.com/go/Horgan/probabilitywithr2e

APPENDIX F

THE INEQUALITIES OF MARKOV AND CHEBYSHEV

E.1 MARKOV'S INEQUALITY: A PROOF WITHOUT WORDS

Markov's inequality for a nonnegative random variable X is visually obvious.

In Figure F.1, the horizontal axis is the sample space S , which could be discrete, or continuous. The vertical axis is the (nonnegative) real numbers \mathbb{R} . Clearly, the area of the shaded rectangle can never be bigger than the area under the curve.

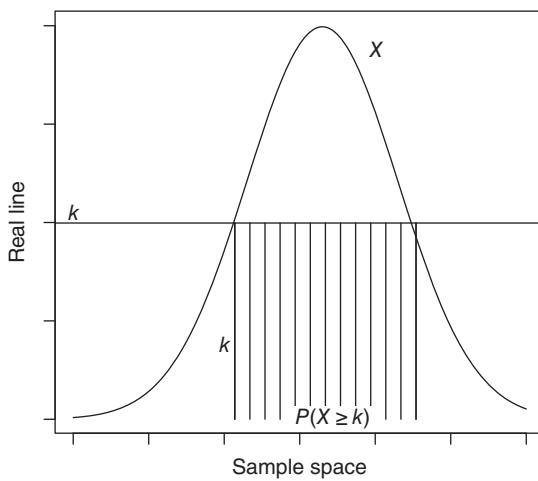


FIGURE F.1 Proof Without Words

F.2 CHEBYSHEV'S INEQUALITY: MARKOV'S IN DISGUISE

Markov's inequality states that for any $k > 0$

$$P(X \geq k) \leq \frac{E(X)}{k}$$

or equivalently

$$kP(X \geq k) \leq E(X). \quad (\text{F.1})$$

Replace k by $h = k^2$ and X by $Y = ((X - E(X))/\sigma)^2$ in Equation F.1, then we have

$$hP(Y \geq h) \leq E(Y)$$

which is

$$k^2 P\left(\left(\frac{X - E(X)}{\sigma}\right)^2 \geq k^2\right) \leq E\left(\frac{X - E(X)}{\sigma}\right)^2$$

which can be written as

$$k^2 P((X - E(X))^2 \geq k^2 \sigma^2) \leq 1 \quad \text{since} \quad E(X - E(X))^2 = \sigma^2$$

or equivalently

$$P((X - E(X))^2 \geq k^2 \sigma^2) \leq \frac{1}{k^2}$$

which is Chebyshev's inequality:

$$P(|X - E(X)| \geq k\sigma) \leq \frac{1}{k^2}.$$

Index to R Commands

* , 5
**, 5
< -, 5
>, 5
(), 402
?, 7
[], 18
#, 48, 73
\$, 32, 39
{ }, 402
abline(), 43, 45, 79, 87, 150, 153, 304, 410, 413,
 424
add, 386
apropos(), 7
attach(), 10
axes, 269, 304
axis(), 270, 304
boxplot(), 31, 33
bquote(), 270, 304, 305, 344, 397, 399, 402
breaks, 39, 354
c(), 7
cbind(), 271, 306
choose(), 69, 263
classes, 74, 79
coincident, 74, 79
col, 333, 387
cumsum(), 87, 168, 411
curve(), 329, 332, 343, 348, 390
data.frame(), 11, 50, 396
dbinom(), 266
demo(), 6, 31
dexp(), 343
dgeom(), 381
dhyper(), 258
dnorm(), 386
dpois(), 276, 285
dunif(), 332
expression(), 343
factor(), 24, 35
fix(), 11

font.lab, 54
 font.main, 35, 53
 for(), 73, 272, 304, 305, 344, 372, 402
 freq, 39
 function(), 27, 330, 336

 head(), 10
 header, 9
 help, 14
 help(), 7
 help.start(), 6
 hist(), 36, 354, 361
 breaks, 37
 freq, 39
 plot, 39
 history(), 12

 installed.packages(), 14
 integrate(), 330

 labels, 35, 270, 386
 length(), 25, 27, 220, 222, 355, 362
 levels, 35
 library(), 14
 lines(), 244, 309, 331, 333, 345, 387,
 396, 397
 list, 13
 lm(), 43, 46
 ls(), 12
 lty, 310

 main, 35
 matplot(), 271
 matrix(), 188
 max(), 21, 317, 355, 372, 373
 max.show, 12
 mean(), 17, 18, 183, 356, 374
 median(), 19
 mfrow(), 36, 203
 min(), 21, 375

 NA, 11, 18, 19
 na.rm, 19, 21, 22
 names(), 11

 objects(), 13

 pairs(), 41
 par(), 35, 37, 203
 paste(), 343
 pbinom(), 234, 302, 310
 pbirthday(), 74
 pch, 271
 pexp(), 345, 423
 pgeom(), 205
 phyper(), 261, 301, 310
 plot, 39
 plot(), 39, 43, 150, 167, 413
 pnorm(), 386, 387, 424
 ppois(), 287, 303, 310, 422
 prod(), 66, 67, 77
 punif(), 332, 335

 qbinom(), 236
 qbirthday(), 75, 82
 qexp(), 349, 350
 qgeom(), 208
 qnorm(), 390, 413
 qpois(), 289
 quantile(), 22, 403
 qunif(), 335

 range(), 21
 rbinom(), 248
 read.csv(), 11
 read.csv2(), 11
 read.table(), 9, 11
 replace, 84, 184, 338
 rexp(), 353, 355
 rgeom(), 210
 rm(), 13
 rnorm(), 419
 round(), 47, 184, 287, 184, 201
 rpois(), 293, 372
 runif(), 339

 sample(), 84, 183, 338
 sapply(), 19, 22
 sd(), 22, 215, 251
 search(), 13
 segments(), 47, 271
 seq(), 115, 146, 242, 291, 333,
 402

- source(), 28
sqrt(), 251, 391
stem(), 40
subset(), 220, 355, 360
sum(), 25, 190
summary(), 24

table(), 84, 184, 211, 380
tail(), 10
text(), 74, 270, 317, 335, 345, 386, 409, 410,
 424

ticks, 386
type, 87, 90, 165, 202, 203, 206, 270, 293,
 387, 390

var(), 355, 356

xlab, 32
xlim, 51

ylab, 32
ylim, 37, 79

INDEX

- acceptable quality limit, 310
acceptance probability, 301
accessing the cloud, 282
accessing the Internet, 219
action lines, 408
algorithm runtime, 426
algorithms, 44
allowable number of defectives, 312
allowable tolerable error (ATE), 413
amnesia, 356
android apps, 122
Anscombe Quartet, 49
AOQ, *see* average outgoing quality
AOQL, *see* average outgoing quality limit
app store, 122
apps, 20
AQL, *see* acceptable quality limit
ART, *see* average response time
ATE, *see* allowable tolerable error
average, *see* mean
average outgoing quality, 316
average outgoing quality limit, 317
average response time, 376
average sample size, 319
axis labels, 32, 33
axis limits, 37, 51
back up, 148
balls and bins, 76
batch quality, 300, 308
Bayes' Rule, 124, 126
 in classification, 133
 in fault diagnosis, 131
 in machine translation, 137
 in spam filtering, 135
Benford's law of first digits, 174
 Hill's model, 174
 graphical display of, 174
Bernouilli distribution, 170
 cumulative distribution probability, 171
 probability density function, 171
binary communication channel, 119
binomial coefficients, 69
binomial distribution
 approximation to hypergeometric, 272
 cumulative distribution function, 232
 examples of, 226
 mean, 245

- binomial distribution (*Continued*)
 - normal approximation to, 394
 - probability density function, 229
 - simulating expectations, 250
 - variance, 247
- binomial expansion, 230
- binomial theorem, 379
- birthday paradox, 71, 107
 - R* program to calculate, 73
- birthday problem, *see* birthday paradox
- bivariate distribution, 188
 - conditional, 190
 - examples of, 187
 - joint, 187
 - marginal, 189
- boxplots, 31
- buffers, 29
- bug detection, 291
- camera, 99
- cards, 99
- case sensitive, 6
- cdf, *see* cumulative distribution function
- central limit theorem, 404
- certifier, 296
- Chebyshev's bound, *see* Chebyshev's inequality
- Chebyshev's inequality, 427
 - proof of, 429, 451
- chips, 59
- classification
 - more than two cases, 134
 - two-case, 133
- classifiers, *see* machine learning
- coin tossing, 61
- combinations, 68, 69, 228
- comment # in R, 73
- communication channel transmission, 126, 278
- conditional probability, 106, 138
- consumer's risk, 310
- control charts, 407
 - action lines, 408
 - for defective rates, 412
 - for defects in wafers, 412
 - for faulty sectors in memory sticks, 413
 - for individual measurements, 409
 - program to calculate, 409
 - warning lines, 408
- counting techniques, 70
- CPU time, 60
- CRAN website, 4
- cumulative distribution function
 - Bernoulli, 171
 - binomial, 232
- continuous, 331
- discrete, 168
- exponential, 344
- geometric, 203, 204
- hypergeometric, 261
- normal, 387
- Poisson, 287
- current in a circuit, 329
- cusum charts, 411
- data editor, 12
- data entry
 - from a file, 8
 - from the screen, 7
- data frame, 8
- deciles, 23
- defect density, 292
- dependable systems, 300
- designing sampling inspection schemes, 311
 - experimentally in *R*, 314
- destructive testing, 299
- dice throwing, 183
 - pdf of, 184
 - simulating, 183
- digital communication channel, 60, 278
- discrete distribution
 - graphical display of, 165
 - univariate, 163
 - disk drives, 296
 - downloading R, 4
 - downtime, 430
- editing, 12
- editor, 28
- electrical systems, 147
- emails, 122, 139
- events, 58, 92
 - complementary, 94, 96
 - impossible, 97
 - mutually exclusive, 92
- expected value
 - of a continuous random variable, 336
 - of a discrete random variable, 176
 - of a function, 177
 - properties of, 180
- experiment, 58
- exploratory data analysis, 52
- exponential distribution, 341
 - cumulative distribution function, 344
 - examples of, 341
 - lifetimes, 342
 - mean, 351, 364

- memoryless property, *see* Markov property
- probability density function
 - simulated, 353
- simulating expectations, 364
- time between Poisson occurrences, 342
- variance, 352, 364

- Facebook, 20, 29
- factorial, 69
- failure rate, 347
- fair coin, 61
- faults in integrated circuits, 292
- finding bugs in software, 290
- first digit law, *see* Benford
 - fibit, 109
- flash memory, 99
- fraud detection, 175

- game problem, *see* Monty Hall
- games of chance, 57
- geometric distribution
 - cumulative distribution function, 203
 - examples of, 196
 - mean, 209
 - memoryless property, *see* Markov property
 - probability density function, 198
 - simulating expectations, 210
 - standard deviation, 251
 - variance, 210
- geometric series, 198, 204, 209
- geospatial analysis, 122, 139
- getting help in R, 6
- Google, 297
- graphical displays, 51
- graphical user interface (GUI), 14

- hacker, 140
- hard drive, 148
- hash table collision, 80
- heights, 383
- histogram, 36, 39
 - normalized, 39
 - percentages, 39
- horse kicks, 274
- hypergeometric distribution
 - binomial approximation, 272, 302
 - cumulative distribution function, 261
 - examples of, 255
- Poisson approximation, 303
- probability density function, 258
- sampling without replacement, 257, 272

- independence
 - mutual, 110
 - pairwise, 109, 110
- integrated circuit board
 - design of, 240
 - reliability of, 239
- integrated circuits, 147
- integrating development environment (IDE), 14
- intel, 113–114
- interarrival time, 371
- intercept, 45
- Internet access, 53
- Internet banking, 103
- interquartile range, 23, 31, 290, 391
- iPhone, 109, 368

- Java assembler interface, 431
- job submissions, 342
 - simulating, 353
 - to a printer, 378
- joint distribution, 188

- law of diminishing returns, 149, 243
- Let's Make a Deal, *see* Monty Hall
- lifetime of a component, 347
- limiting quality, 310
- line of best fit, 43, 45, 48
 - calculating the coefficients, 53
 - deriving the coefficients, 437
 - linear trend, 45
 - prediction, 43
- linear search algorithm, 427
- loops in R, 73, 304, 372, 397, 398, 401
- lottery, 262
 - calculating probabilities, 263
 - winning the jackpot, 263

- M/M/1 queue, 371
- machine learning, 44, 45, 133
 - classifiers, 242
 - regression, 44
 - spam filtering, 135
 - supervised learning, 44
 - testing set, 45, 46
 - training set, 45, 135
 - use of binomial, 242
- machine translation, 137, 193
 - conditional probabilities, 138
 - fundamental equation of, 137
 - training set, 138
- majority vote, 242
- marginal distribution, 189

- Markov property
 of the exponential, 357
 proof of, 357
 simulating, 360
 of the geometric, 217
 proof of, 217
 simulating, 219
- Markov's bound, *see* Markov's inequality
- Markov's inequality, 420
 proof of, 421
 with differing standard deviations, 424
- mean deviation, 179
- mean of, *see also* expected value
 a continuous random variable, 336
 a discrete random variable, 177, 336
 a set of values, 17
- binomial distribution, 245
- exponential, 351
- geometric, 209
- normal, 386
- Poisson, 284
- uniform
 continuous, 337
 discrete, 183
- median, 19, 32
- memory sticks, 429
- memoryless property, *see* Markov property
- Minitab, 3
- missing values, 11, 18
- Monty Hall, 130
- motherboard, 368
- mutually exclusive, *see* events
- network failures, 282, 431
- normal distribution, 383
 approximation to binomial, 394
 approximation to Poisson, 399
 cumulative distribution function, 387
 mean, 385
 plotting normal curves in *R*, 386
 probability density function, 385
 standard normal, 391
 variance, 385
- normal probabilities
 using *R*, 394
 using tables, 391, 393
- numerical integration, 330
- objects, 6
 listing, 12
 removing, 13
- OC curve, *see* operating characteristic curve
- operating characteristic curve, 308
- outlier, 20, 33, 51
- packages, 13
- parameter, 353
- password, 67, 101, 139
 cracking, 67
- pdf, *see* probability density function
- Pentium chip, 114
- percentiles, 22, 23
- permutations, 64, 66
- PIN, 80
- playing cards, 62
- plotting cdfs in *R*
 binomial, 232
 geometric, 203
 hypergeometric, 261
 Poisson, 288
- plotting multiple normal curves, 386
- plotting pdfs in *R*
 binomial, 231
 exponential, 343
 geometric, 202
 hypergeometric, 259
 normal, 386
 Poisson, 286
- Poisson distribution, 274
 approximation to binomial, 276
 cumulative distribution function, 287
 examples of, 282
 mean, 284
 simulated, 295
 normal approximation to, 399
 probability density function, 283
 simulated, 293
 variance, 284
 simulated, 295
- Poisson, S.D., 275, 298
- posterior probability, 127, 129, 131, 133
- precision engineering, 300
- printed circuit boards, 256, 415
- printer failures, 132
- prior probability, 116, 129, 131, 132
- probability
 addition law, 98
 axioms, 94
 classical approach, 61, 64
 complementary, 147
 law of total probability, 117
 multiplication law, 107, 143
 relative frequency, 83
 simulating, 84

- probability density function
 - Bernouilli, 170
 - binomial, 229
 - bivariate, 187
 - continuous, 328
 - discrete, 167
 - exponential, 343
 - geometric, 198
 - hypergeometric distribution, 258
 - normal, 385
 - Poisson distribution, 283
 - uniform
 - continuous, 332
 - discrete, 172
- processors
 - allocation of jobs, 79
 - overload, 81
- producer's risk, 310
- properties of expectation
 - continuous, 337
 - discrete, 175
- pulling cards, 58
- Python, 103
- quality control, 210, 407
- quantile, 412
 - binomial, 235
 - exponential, 349, 350
 - geometric, 208
 - normal, 389
 - Poisson, 289
- quartiles, 22
- queue length, 372
 - a program to calculate, 372
- queues
 - arrival rate, 371
 - M/M/1, 371
 - more than one queue, 378
 - queue length
 - simulating, 372
 - service rate, 371
 - service time, 371
 - single server queue, 371
- R Commander, 14, 15
 - downloading, 15
- R interfaces, 14
- random number generators in *R*
 - binomial, 248
 - continuous uniform, 339
 - exponential, 353
- geometric, 210
- normal, 419
- Poisson, 293
- random variable
 - continuous, 328
 - examples of, 328
 - discrete, 163
 - examples of, 164
- range, 21
- redundancy, 143, 147, 153
- regression, *see* line of best fit
- Rel, *see* reliability
- reliability
 - of a component, 143, 347
 - of a parallel system, 147
 - of a series system, 142, 143
 - of a series-parallel system, 150
 - of a system, 142, 349
 - of the general system, 158, 238
- response time, 59, 423
- retrieval time, 420
- rolling a die, 58, 61
- RStudio, 14
 - downloading, 14
- runtime, 427
- sample space, 58, 60, 163, 327
 - continuous, 60
 - discrete infinite, 60
 - finite, 60
- sampling
 - with replacement, 266
 - without replacement, 258, 272
- sampling inspection schemes
 - 0/1, 313
 - double, 318
 - rectifying, 315
 - single, 300
- scatter plot, 40, 45
- scripts, 28
- sensor, 122, 139
- server farm, 77
- server overload, 82, 298
- sets, 92
- simple linear regression, *see* line of best fit
- simulating
 - binomial values, 248
 - coin tossing, 84, 210
 - continuous random variables, 338
 - discrete random variables, 183
 - exponential values, 353
 - hardware failures, 185

- simulating (*Continued*)
 - Poisson values, 293
 - queue length, 372
 - reliability, 356
 - rolling a die, 183
 - sampling inspection schemes, 303
 - waiting times, 353
- skewness coefficient, 27
 - approximation, 30
 - program to calculate, 27
- slope, 45
- smartphone, 281
- software packages, 114
- spam filtering, 135
 - training set, 135
- standard deviation, 22, 25
- standard normal distribution, 413
- stem and leaf, 40
- summary statistics, 24, 50
- supervised learning, *see* machine learning
- system crash, 422
- tablet, 99
- tail probabilities, 425
- target, 408
- Taylor expansion, 283
- testing set, *see* machine learning
- tossing a coin, 58
- tossing two coins, 58
- total probability, 116, 130
- traffic intensity, 371
- training set, *see* machine learning
- tree diagram, 118, 119
- twitter, 252
- uniform discrete random variable, 172
 - cumulative distribution function, 172
 - probability density function, 172
- uniform continuous random variable, 332
 - graphs of, 332
 - mean, 337
 - simulated, 340
 - probability density function, 332
 - variance, 337
 - simulated, 340
- variance of
 - binomial, 247
 - continuous random variable, 336
 - discrete random variable, 179, 336
 - exponential, 352
 - geometric, 210
 - normal, 386
 - Poisson, 284
 - uniform
 - continuous, 337
- vector, 8
- Venables, W. N., 4, 16
- virus, 102, 103
- virus infections, 282
- von Bortkiewicz, L., 275, 298
- wafer, 281, 412
- waiting times between Poisson occurrences, 342
- warning lines, 408
- web hits, 282, 289, 367
 - simulating, 293
- wikipedia, 252
- web search, 59, 60
- web server capacity, 290
- website design, 60
- workspace, 6
 - loading, 13
 - removing, 13
- workstation, 349, 415
- writing functions in R, 27

POSTFACE

Although we have come to the end of our treatise on probability with *R*, hopefully, your journey is not completely over. The continued development of *R* depends not only on the team of core developers but also on the ever increasing community of *R* users. Users can support *R* by writing packages that other users might find useful. You now have an opportunity to do this. You can share the knowledge you have acquired with the rest of the *R* community, by converting some of what you have learned and what you believe will interest other *R* users into an *R* package and uploading it.

Instructions for doing this are available on the *R* website cran.r-project.org. There you will find the manual *Writing R Extensions* containing details, among other things, on how to create your own package.

