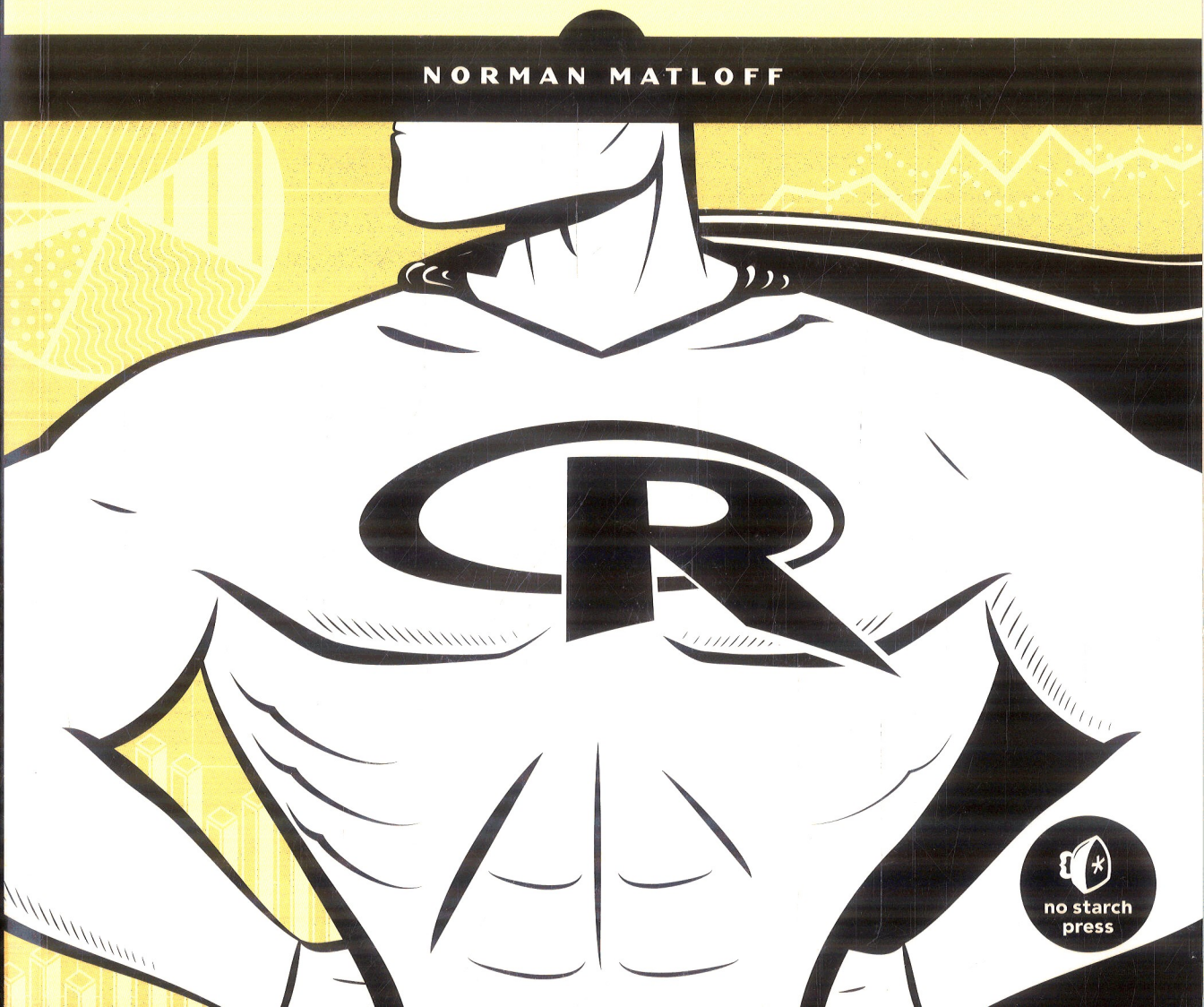


# THE ART OF R PROGRAMMING

A TOUR OF STATISTICAL SOFTWARE DESIGN

NORMAN MATLOFF



# CONTENTS IN DETAIL

## ACKNOWLEDGMENTS

xvii

## INTRODUCTION

xix

|                                            |       |
|--------------------------------------------|-------|
| Why Use R for Your Statistical Work? ..... | xix   |
| Object-Oriented Programming .....          | xvii  |
| Functional Programming .....               | xvii  |
| Whom Is This Book For? .....               | xviii |
| My Own Background .....                    | xix   |

## 1

## GETTING STARTED

1

|                                                                |    |
|----------------------------------------------------------------|----|
| 1.1 How to Run R .....                                         | 1  |
| 1.1.1 Interactive Mode .....                                   | 2  |
| 1.1.2 Batch Mode .....                                         | 3  |
| 1.2 A First R Session .....                                    | 4  |
| 1.3 Introduction to Functions .....                            | 7  |
| 1.3.1 Variable Scope .....                                     | 9  |
| 1.3.2 Default Arguments .....                                  | 9  |
| 1.4 Preview of Some Important R Data Structures .....          | 10 |
| 1.4.1 Vectors, the R Workhorse .....                           | 10 |
| 1.4.2 Character Strings .....                                  | 11 |
| 1.4.3 Matrices .....                                           | 11 |
| 1.4.4 Lists .....                                              | 12 |
| 1.4.5 Data Frames .....                                        | 14 |
| 1.4.6 Classes .....                                            | 15 |
| 1.5 Extended Example: Regression Analysis of Exam Grades ..... | 16 |
| 1.6 Startup and Shutdown .....                                 | 19 |
| 1.7 Getting Help .....                                         | 20 |
| 1.7.1 The help() Function .....                                | 20 |
| 1.7.2 The example() Function .....                             | 21 |
| 1.7.3 If You Don't Know Quite What You're Looking For .....    | 22 |
| 1.7.4 Help for Other Topics .....                              | 23 |
| 1.7.5 Help for Batch Mode .....                                | 24 |
| 1.7.6 Help on the Internet .....                               | 24 |

## 2

## VECTORS

25

|       |                                                                |    |
|-------|----------------------------------------------------------------|----|
| 2.1   | Scalars, Vectors, Arrays, and Matrices .....                   | 26 |
| 2.1.1 | Adding and Deleting Vector Elements .....                      | 26 |
| 2.1.2 | Obtaining the Length of a Vector .....                         | 27 |
| 2.1.3 | Matrices and Arrays as Vectors .....                           | 28 |
| 2.2   | Declarations .....                                             | 28 |
| 2.3   | Recycling .....                                                | 29 |
| 2.4   | Common Vector Operations .....                                 | 30 |
| 2.4.1 | Vector Arithmetic and Logical Operations .....                 | 30 |
| 2.4.2 | Vector Indexing .....                                          | 31 |
| 2.4.3 | Generating Useful Vectors with the : Operator .....            | 32 |
| 2.4.4 | Generating Vector Sequences with seq() .....                   | 33 |
| 2.4.5 | Repeating Vector Constants with rep() .....                    | 34 |
| 2.5   | Using all() and any() .....                                    | 35 |
| 2.5.1 | Extended Example: Finding Runs of Consecutive Ones .....       | 35 |
| 2.5.2 | Extended Example: Predicting Discrete-Valued Time Series ..... | 37 |
| 2.6   | Vectorized Operations .....                                    | 39 |
| 2.6.1 | Vector In, Vector Out .....                                    | 40 |
| 2.6.2 | Vector In, Matrix Out .....                                    | 42 |
| 2.7   | NA and NULL Values .....                                       | 43 |
| 2.7.1 | Using NA .....                                                 | 43 |
| 2.7.2 | Using NULL .....                                               | 44 |
| 2.8   | Filtering .....                                                | 45 |
| 2.8.1 | Generating Filtering Indices .....                             | 45 |
| 2.8.2 | Filtering with the subset() Function .....                     | 47 |
| 2.8.3 | The Selection Function which() .....                           | 47 |
| 2.9   | A Vectorized if-then-else: The ifelse() Function .....         | 48 |
| 2.9.1 | Extended Example: A Measure of Association .....               | 49 |
| 2.9.2 | Extended Example: Recoding an Abalone Data Set .....           | 51 |
| 2.10  | Testing Vector Equality .....                                  | 54 |
| 2.11  | Vector Element Names .....                                     | 56 |
| 2.12  | More on c() .....                                              | 56 |

## 3

## MATRICES AND ARRAYS

59

|       |                                                        |    |
|-------|--------------------------------------------------------|----|
| 3.1   | Creating Matrices .....                                | 59 |
| 3.2   | General Matrix Operations .....                        | 61 |
| 3.2.1 | Performing Linear Algebra Operations on Matrices ..... | 61 |
| 3.2.2 | Matrix Indexing .....                                  | 62 |
| 3.2.3 | Extended Example: Image Manipulation .....             | 63 |
| 3.2.4 | Filtering on Matrices .....                            | 66 |
| 3.2.5 | Extended Example: Generating a Covariance Matrix ..... | 69 |

|       |                                                                            |    |
|-------|----------------------------------------------------------------------------|----|
| 3.3   | Applying Functions to Matrix Rows and Columns .....                        | 70 |
| 3.3.1 | Using the <code>apply()</code> Function .....                              | 70 |
| 3.3.2 | Extended Example: Finding Outliers .....                                   | 72 |
| 3.4   | Adding and Deleting Matrix Rows and Columns .....                          | 73 |
| 3.4.1 | Changing the Size of a Matrix .....                                        | 73 |
| 3.4.2 | Extended Example: Finding the Closest Pair of Vertices in<br>a Graph ..... | 75 |
| 3.5   | More on the Vector/Matrix Distinction .....                                | 78 |
| 3.6   | Avoiding Unintended Dimension Reduction .....                              | 80 |
| 3.7   | Naming Matrix Rows and Columns .....                                       | 81 |
| 3.8   | Higher-Dimensional Arrays .....                                            | 82 |

## **4**

### **LISTS**

## **85**

|       |                                                                           |    |
|-------|---------------------------------------------------------------------------|----|
| 4.1   | Creating Lists .....                                                      | 85 |
| 4.2   | General List Operations .....                                             | 87 |
| 4.2.1 | List Indexing .....                                                       | 87 |
| 4.2.2 | Adding and Deleting List Elements .....                                   | 88 |
| 4.2.3 | Getting the Size of a List .....                                          | 90 |
| 4.2.4 | Extended Example: Text Concordance .....                                  | 90 |
| 4.3   | Accessing List Components and Values .....                                | 93 |
| 4.4   | Applying Functions to Lists .....                                         | 95 |
| 4.4.1 | Using the <code>lapply()</code> and <code>sapply()</code> Functions ..... | 95 |
| 4.4.2 | Extended Example: Text Concordance, Continued .....                       | 95 |
| 4.4.3 | Extended Example: Back to the Abalone Data .....                          | 99 |
| 4.5   | Recursive Lists .....                                                     | 99 |

## **5**

### **DATA FRAMES**

## **101**

|       |                                                                                          |     |
|-------|------------------------------------------------------------------------------------------|-----|
| 5.1   | Creating Data Frames .....                                                               | 102 |
| 5.1.1 | Accessing Data Frames .....                                                              | 102 |
| 5.1.2 | Extended Example: Regression Analysis of Exam Grades<br>Continued .....                  | 103 |
| 5.2   | Other Matrix-Like Operations .....                                                       | 104 |
| 5.2.1 | Extracting Subdata Frames .....                                                          | 104 |
| 5.2.2 | More on Treatment of NA Values .....                                                     | 105 |
| 5.2.3 | Using the <code>rbind()</code> and <code>cbind()</code> Functions and Alternatives ..... | 106 |
| 5.2.4 | Applying <code>apply()</code> .....                                                      | 107 |
| 5.2.5 | Extended Example: A Salary Study .....                                                   | 108 |
| 5.3   | Merging Data Frames .....                                                                | 109 |
| 5.3.1 | Extended Example: An Employee Database .....                                             | 111 |
| 5.4   | Applying Functions to Data Frames .....                                                  | 112 |
| 5.4.1 | Using <code>lapply()</code> and <code>sapply()</code> on Data Frames .....               | 112 |
| 5.4.2 | Extended Example: Applying Logistic Regression Models .....                              | 113 |
| 5.4.3 | Extended Example: Aids for Learning Chinese Dialects .....                               | 115 |

## **6** **FACTORS AND TABLES** **121**

|       |                                                        |     |
|-------|--------------------------------------------------------|-----|
| 6.1   | Factors and Levels                                     | 121 |
| 6.2   | Common Functions Used with Factors                     | 123 |
| 6.2.1 | The <code>tapply()</code> Function                     | 123 |
| 6.2.2 | The <code>split()</code> Function                      | 124 |
| 6.2.3 | The <code>by()</code> Function                         | 126 |
| 6.3   | Working with Tables                                    | 127 |
| 6.3.1 | Matrix/Array-Like Operations on Tables                 | 130 |
| 6.3.2 | Extended Example: Extracting a Subtable                | 131 |
| 6.3.3 | Extended Example: Finding the Largest Cells in a Table | 134 |
| 6.4   | Other Factor- and Table-Related Functions              | 136 |
| 6.4.1 | The <code>aggregate()</code> Function                  | 136 |
| 6.4.2 | The <code>cut()</code> Function                        | 136 |

## **7** **R PROGRAMMING STRUCTURES** **139**

|       |                                                                      |     |
|-------|----------------------------------------------------------------------|-----|
| 7.1   | Control Statements                                                   | 139 |
| 7.1.1 | Loops                                                                | 140 |
| 7.1.2 | Looping Over Nonvector Sets                                          | 142 |
| 7.1.3 | <code>if-else</code>                                                 | 143 |
| 7.2   | Arithmetic and Boolean Operators and Values                          | 145 |
| 7.3   | Default Values for Arguments                                         | 146 |
| 7.4   | Return Values                                                        | 147 |
| 7.4.1 | Deciding Whether to Explicitly Call <code>return()</code>            | 148 |
| 7.4.2 | Returning Complex Objects                                            | 148 |
| 7.5   | Functions Are Objects                                                | 149 |
| 7.6   | Environment and Scope Issues                                         | 151 |
| 7.6.1 | The Top-Level Environment                                            | 152 |
| 7.6.2 | The Scope Hierarchy                                                  | 152 |
| 7.6.3 | More on <code>ls()</code>                                            | 155 |
| 7.6.4 | Functions Have (Almost) No Side Effects                              | 156 |
| 7.6.5 | Extended Example: A Function to Display the Contents of a Call Frame | 157 |
| 7.7   | No Pointers in R                                                     | 159 |
| 7.8   | Writing Upstairs                                                     | 161 |
| 7.8.1 | Writing to Nonlocals with the Superassignment Operator               | 161 |
| 7.8.2 | Writing to Nonlocals with <code>assign()</code>                      | 163 |
| 7.8.3 | Extended Example: Discrete-Event Simulation in R                     | 164 |
| 7.8.4 | When Should You Use Global Variables?                                | 171 |
| 7.8.5 | Closures                                                             | 174 |
| 7.9   | Recursion                                                            | 176 |
| 7.9.1 | A Quicksort Implementation                                           | 176 |
| 7.9.2 | Extended Example: A Binary Search Tree                               | 177 |

|        |                                                            |     |
|--------|------------------------------------------------------------|-----|
| 7.10   | Replacement Functions .....                                | 182 |
| 7.10.1 | What's Considered a Replacement Function? .....            | 183 |
| 7.10.2 | Extended Example: A Self-Bookkeeping Vector Class .....    | 184 |
| 7.11   | Tools for Composing Function Code .....                    | 186 |
| 7.11.1 | Text Editors and Integrated Development Environments ..... | 186 |
| 7.11.2 | The edit() Function .....                                  | 186 |
| 7.12   | Writing Your Own Binary Operations .....                   | 187 |
| 7.13   | Anonymous Functions .....                                  | 187 |

## **8 DOING MATH AND SIMULATIONS IN R 189**

|       |                                                                           |     |
|-------|---------------------------------------------------------------------------|-----|
| 8.1   | Math Functions .....                                                      | 189 |
| 8.1.1 | Extended Example: Calculating a Probability .....                         | 190 |
| 8.1.2 | Cumulative Sums and Products .....                                        | 191 |
| 8.1.3 | Minima and Maxima .....                                                   | 191 |
| 8.1.4 | Calculus .....                                                            | 192 |
| 8.2   | Functions for Statistical Distributions .....                             | 193 |
| 8.3   | Sorting .....                                                             | 194 |
| 8.4   | Linear Algebra Operations on Vectors and Matrices .....                   | 196 |
| 8.4.1 | Extended Example: Vector Cross Product .....                              | 198 |
| 8.4.2 | Extended Example: Finding Stationary Distributions of Markov Chains ..... | 199 |
| 8.5   | Set Operations .....                                                      | 202 |
| 8.6   | Simulation Programming in R .....                                         | 204 |
| 8.6.1 | Built-In Random Variate Generators .....                                  | 204 |
| 8.6.2 | Obtaining the Same Random Stream in Repeated Runs .....                   | 205 |
| 8.6.3 | Extended Example: A Combinatorial Simulation .....                        | 205 |

## **9 OBJECT-ORIENTED PROGRAMMING 207**

|       |                                                                       |     |
|-------|-----------------------------------------------------------------------|-----|
| 9.1   | S3 Classes .....                                                      | 208 |
| 9.1.1 | S3 Generic Functions .....                                            | 208 |
| 9.1.2 | Example: OOP in the lm() Linear Model Function .....                  | 208 |
| 9.1.3 | Finding the Implementations of Generic Methods .....                  | 210 |
| 9.1.4 | Writing S3 Classes .....                                              | 212 |
| 9.1.5 | Using Inheritance .....                                               | 214 |
| 9.1.6 | Extended Example: A Class for Storing Upper-Triangular Matrices ..... | 214 |
| 9.1.7 | Extended Example: A Procedure for Polynomial Regression .....         | 219 |
| 9.2   | S4 Classes .....                                                      | 222 |
| 9.2.1 | Writing S4 Classes .....                                              | 223 |
| 9.2.2 | Implementing a Generic Function on an S4 Class .....                  | 225 |
| 9.3   | S3 Versus S4 .....                                                    | 226 |

|       |                                                                            |     |
|-------|----------------------------------------------------------------------------|-----|
| 9.4   | Managing Your Objects.....                                                 | 226 |
| 9.4.1 | Listing Your Objects with the <code>ls()</code> Function.....              | 226 |
| 9.4.2 | Removing Specific Objects with the <code>rm()</code> Function .....        | 227 |
| 9.4.3 | Saving a Collection of Objects with the <code>save()</code> Function ..... | 228 |
| 9.4.4 | "What Is This?" .....                                                      | 228 |
| 9.4.5 | The <code>exists()</code> Function .....                                   | 230 |

## 10 INPUT/OUTPUT 231

|        |                                                        |     |
|--------|--------------------------------------------------------|-----|
| 10.1   | Accessing the Keyboard and Monitor.....                | 232 |
| 10.1.1 | Using the <code>scan()</code> Function .....           | 232 |
| 10.1.2 | Using the <code>readline()</code> Function .....       | 234 |
| 10.1.3 | Printing to the Screen.....                            | 234 |
| 10.2   | Reading and Writing Files.....                         | 235 |
| 10.2.1 | Reading a Data Frame or Matrix from a File.....        | 236 |
| 10.2.2 | Reading Text Files .....                               | 237 |
| 10.2.3 | Introduction to Connections .....                      | 237 |
| 10.2.4 | Extended Example: Reading PUMS Census Files .....      | 239 |
| 10.2.5 | Accessing Files on Remote Machines via URLs .....      | 243 |
| 10.2.6 | Writing to a File .....                                | 243 |
| 10.2.7 | Getting File and Directory Information .....           | 245 |
| 10.2.8 | Extended Example: Sum the Contents of Many Files ..... | 245 |
| 10.3   | Accessing the Internet.....                            | 246 |
| 10.3.1 | Overview of TCP/IP .....                               | 247 |
| 10.3.2 | Sockets in R .....                                     | 247 |
| 10.3.3 | Extended Example: Implementing Parallel R .....        | 248 |

## 11 STRING MANIPULATION 251

|        |                                                                        |     |
|--------|------------------------------------------------------------------------|-----|
| 11.1   | An Overview of String-Manipulation Functions.....                      | 251 |
| 11.1.1 | <code>grep()</code> .....                                              | 252 |
| 11.1.2 | <code>nchar()</code> .....                                             | 252 |
| 11.1.3 | <code>paste()</code> .....                                             | 252 |
| 11.1.4 | <code>sprintf()</code> .....                                           | 253 |
| 11.1.5 | <code>substr()</code> .....                                            | 253 |
| 11.1.6 | <code>strsplit()</code> .....                                          | 253 |
| 11.1.7 | <code>regexpr()</code> .....                                           | 253 |
| 11.1.8 | <code>gregexpr()</code> .....                                          | 254 |
| 11.2   | Regular Expressions .....                                              | 254 |
| 11.2.1 | Extended Example: Testing a Filename for a Given Suffix .....          | 255 |
| 11.2.2 | Extended Example: Forming Filenames.....                               | 256 |
| 11.3   | Use of String Utilities in the <code>edtdbg</code> Debugging Tool..... | 257 |

|           |                                                                                 |            |
|-----------|---------------------------------------------------------------------------------|------------|
| <b>12</b> | <b>GRAPHICS</b>                                                                 | <b>261</b> |
| 12.1      | Creating Graphs                                                                 | 261        |
| 12.1.1    | The Workhorse of R Base Graphics: The <code>plot()</code> Function              | 262        |
| 12.1.2    | Adding Lines: The <code>abline()</code> Function                                | 263        |
| 12.1.3    | Starting a New Graph While Keeping the Old Ones                                 | 264        |
| 12.1.4    | Extended Example: Two Density Estimates on the Same Graph                       | 264        |
| 12.1.5    | Extended Example: More on the Polynomial Regression Example                     | 266        |
| 12.1.6    | Adding Points: The <code>points()</code> Function                               | 269        |
| 12.1.7    | Adding a Legend: The <code>legend()</code> Function                             | 270        |
| 12.1.8    | Adding Text: The <code>text()</code> Function                                   | 270        |
| 12.1.9    | Pinpointing Locations: The <code>locator()</code> Function                      | 271        |
| 12.1.10   | Restoring a Plot                                                                | 272        |
| 12.2      | Customizing Graphs                                                              | 272        |
| 12.2.1    | Changing Character Sizes: The <code>cex</code> Option                           | 272        |
| 12.2.2    | Changing the Range of Axes: The <code>xlim</code> and <code>ylim</code> Options | 273        |
| 12.2.3    | Adding a Polygon: The <code>polygon()</code> Function                           | 275        |
| 12.2.4    | Smoothing Points: The <code>lowess()</code> and <code>loess()</code> Functions  | 276        |
| 12.2.5    | Graphing Explicit Functions                                                     | 276        |
| 12.2.6    | Extended Example: Magnifying a Portion of a Curve                               | 277        |
| 12.3      | Saving Graphs to Files                                                          | 280        |
| 12.3.1    | R Graphics Devices                                                              | 280        |
| 12.3.2    | Saving the Displayed Graph                                                      | 281        |
| 12.3.3    | Closing an R Graphics Device                                                    | 281        |
| 12.4      | Creating Three-Dimensional Plots                                                | 282        |

|           |                                                                                                        |            |
|-----------|--------------------------------------------------------------------------------------------------------|------------|
| <b>13</b> | <b>DEBUGGING</b>                                                                                       | <b>285</b> |
| 13.1      | Fundamental Principles of Debugging                                                                    | 285        |
| 13.1.1    | The Essence of Debugging: The Principle of Confirmation                                                | 285        |
| 13.1.2    | Start Small                                                                                            | 286        |
| 13.1.3    | Debug in a Modular, Top-Down Manner                                                                    | 286        |
| 13.1.4    | Antibugging                                                                                            | 287        |
| 13.2      | Why Use a Debugging Tool?                                                                              | 287        |
| 13.3      | Using R Debugging Facilities                                                                           | 288        |
| 13.3.1    | Single-Stepping with the <code>debug()</code> and <code>browser()</code> Functions                     | 288        |
| 13.3.2    | Using Browser Commands                                                                                 | 289        |
| 13.3.3    | Setting Breakpoints                                                                                    | 289        |
| 13.3.4    | Tracking with the <code>trace()</code> Function                                                        | 291        |
| 13.3.5    | Performing Checks After a Crash with the <code>traceback()</code> and <code>debugger()</code> Function | 291        |
| 13.3.6    | Extended Example: Two Full Debugging Sessions                                                          | 292        |
| 13.4      | Moving Up in the World: More Convenient Debugging Tools                                                | 300        |

|      |                                                        |     |
|------|--------------------------------------------------------|-----|
| 13.5 | Ensuring Consistency in Debugging Simulation Code..... | 302 |
| 13.6 | Syntax and Runtime Errors.....                         | 303 |
| 13.7 | Running GDB on R Itself.....                           | 303 |

## 14

### PERFORMANCE ENHANCEMENT: SPEED AND MEMORY 305

|        |                                                                           |     |
|--------|---------------------------------------------------------------------------|-----|
| 14.1   | Writing Fast R Code.....                                                  | 306 |
| 14.2   | The Dreaded for Loop.....                                                 | 306 |
| 14.2.1 | Vectorization for Speedup.....                                            | 306 |
| 14.2.2 | Extended Example: Achieving Better Speed in a Monte Carlo Simulation..... | 308 |
| 14.2.3 | Extended Example: Generating a Powers Matrix.....                         | 312 |
| 14.3   | Functional Programming and Memory Issues.....                             | 314 |
| 14.3.1 | Vector Assignment Issues.....                                             | 314 |
| 14.3.2 | Copy-on-Change Issues.....                                                | 314 |
| 14.3.3 | Extended Example: Avoiding Memory Copy.....                               | 315 |
| 14.4   | Using Rprof() to Find Slow Spots in Your Code.....                        | 316 |
| 14.4.1 | Monitoring with Rprof().....                                              | 316 |
| 14.4.2 | How Rprof() Works.....                                                    | 318 |
| 14.5   | Byte Code Compilation.....                                                | 320 |
| 14.6   | Oh No, the Data Doesn't Fit into Memory!.....                             | 320 |
| 14.6.1 | Chunking.....                                                             | 320 |
| 14.6.2 | Using R Packages for Memory Management.....                               | 321 |

## 15

### INTERFACING R TO OTHER LANGUAGES 323

|        |                                                                  |     |
|--------|------------------------------------------------------------------|-----|
| 15.1   | Writing C/C++ Functions to Be Called from R.....                 | 323 |
| 15.1.1 | Some R-to-C/C++ Preliminaries.....                               | 324 |
| 15.1.2 | Example: Extracting Subdiagonals from a Square Matrix.....       | 324 |
| 15.1.3 | Compiling and Running Code.....                                  | 325 |
| 15.1.4 | Debugging R/C Code.....                                          | 326 |
| 15.1.5 | Extended Example: Prediction of Discrete-Valued Time Series..... | 327 |
| 15.2   | Using R from Python.....                                         | 330 |
| 15.2.1 | Installing RPy.....                                              | 330 |
| 15.2.2 | RPy Syntax.....                                                  | 330 |

## 16

### PARALLEL R 333

|        |                                           |     |
|--------|-------------------------------------------|-----|
| 16.1   | The Mutual Outlinks Problem.....          | 333 |
| 16.2   | Introducing the snow Package.....         | 334 |
| 16.2.1 | Running snow Code.....                    | 335 |
| 16.2.2 | Analyzing the snow Code.....              | 336 |
| 16.2.3 | How Much Speedup Can Be Attained?.....    | 337 |
| 16.2.4 | Extended Example: K-Means Clustering..... | 338 |

|        |                                                                                       |     |
|--------|---------------------------------------------------------------------------------------|-----|
| 16.3   | Resorting to C .....                                                                  | 340 |
| 16.3.1 | Using Multicore Machines .....                                                        | 340 |
| 16.3.2 | Extended Example: Mutual Outlinks Problem in OpenMP .....                             | 341 |
| 16.3.3 | Running the OpenMP Code .....                                                         | 342 |
| 16.3.4 | OpenMP Code Analysis .....                                                            | 343 |
| 16.3.5 | Other OpenMP Pragmas .....                                                            | 344 |
| 16.3.6 | GPU Programming .....                                                                 | 345 |
| 16.4   | General Performance Considerations .....                                              | 345 |
| 16.4.1 | Sources of Overhead .....                                                             | 346 |
| 16.4.2 | Embarrassingly Parallel Applications and Those That Aren't .....                      | 347 |
| 16.4.3 | Static Versus Dynamic Task Assignment .....                                           | 348 |
| 16.4.4 | Software Alchemy: Turning General Problems into<br>Embarrassingly Parallel Ones ..... | 350 |
| 16.5   | Debugging Parallel R Code .....                                                       | 351 |

## **A** **INSTALLING R** **353**

|     |                                               |     |
|-----|-----------------------------------------------|-----|
| A.1 | Downloading R from CRAN .....                 | 353 |
| A.2 | Installing from a Linux Package Manager ..... | 353 |
| A.3 | Installing from Source .....                  | 354 |

## **B** **INSTALLING AND USING PACKAGES** **355**

|       |                                              |     |
|-------|----------------------------------------------|-----|
| B.1   | Package Basics .....                         | 355 |
| B.2   | Loading a Package from Your Hard Drive ..... | 356 |
| B.3   | Downloading a Package from the Web .....     | 356 |
| B.3.1 | Installing Packages Automatically .....      | 356 |
| B.3.2 | Installing Packages Manually .....           | 357 |
| B.4   | Listing the Functions in a Package .....     | 358 |