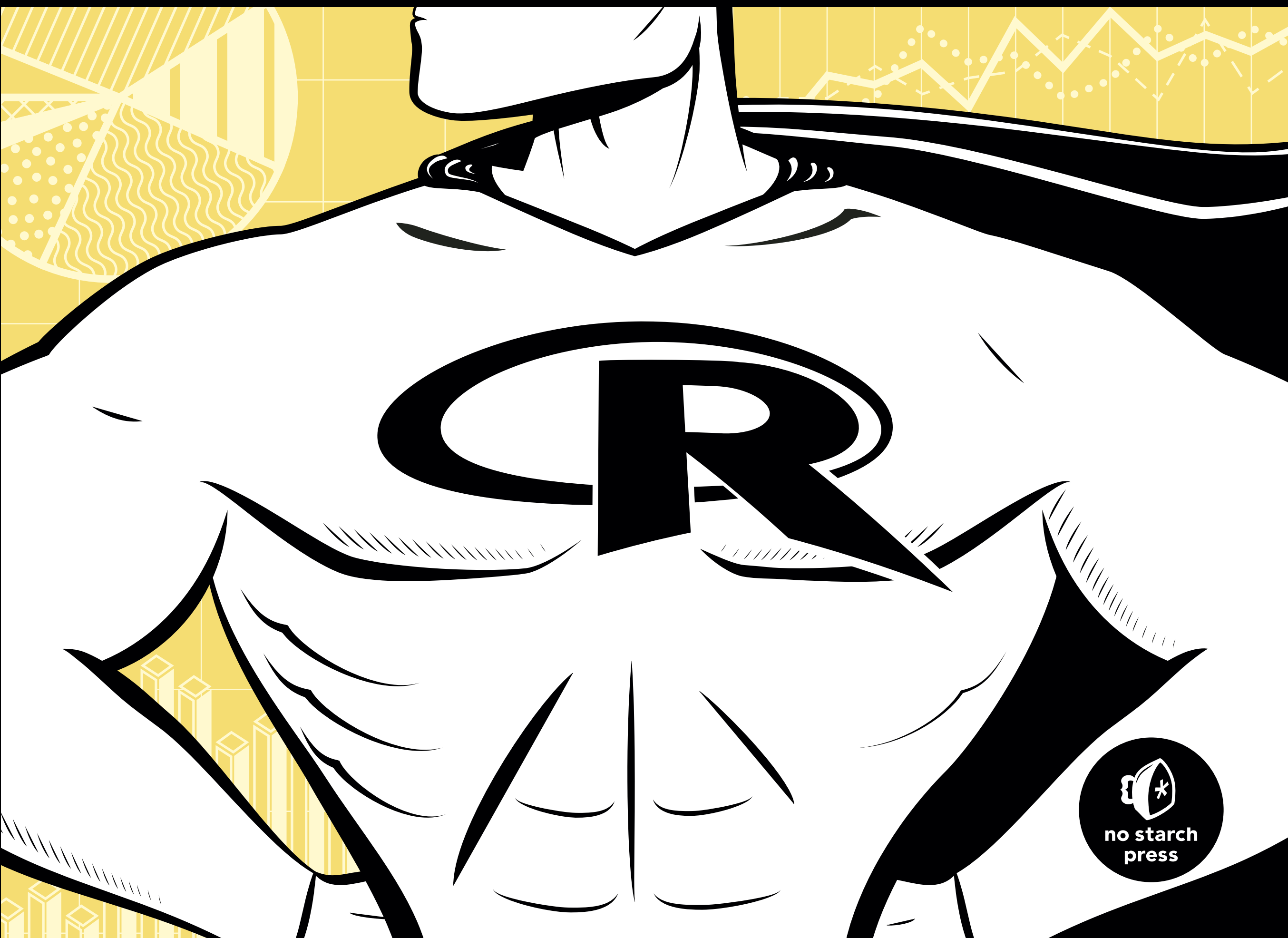
THE ART OF R PROGRAMMING

A *TOUR* OF STATISTICAL SOFTWARE DESIGN

NORMAN MATLOFF



no starch press

INDEX

Special Characters

`:` (colon operator), 32–33
`==` operator, 54–55
`>` operator, 40
`.libpaths()` function, 356–357
.Rdata file, 20
.Rhistory file, 20
.Rprofile file, 19
`<<-` (superassignment operator), 9
  simplifying code, 174
  writing to nonlocals with, 161–162
`+` operator, 31
`"%mut%"()` function, 218

A

abalone data set
  recoding, 51–54
  using `lapply()` function, 99
`abline()` graphics function, 150
`abs()` math function, 189
accessing
  data frames, 102–104
  files on remote machines via URLs, 243
  Internet, 246–250
    implementing parallel R example, 248–250
    sockets, 247–248
    TCP/IP, 247
  keyboard and monitor, 232–235
    using `print()` function, 234–235
    using `readline()` function, 234
    using `scan()` function, 232–234
  list components and values, 93–95
actual argument, 9
adding
  legends to graphs with `legend()` function, 270
  lines with `abline()` function, 263–264
  list elements, 88–90
  matrix rows and columns, 73–78
  points to graphs with `points()` function, 269–270
  text to graphs with `text()` function, 270–271
`addmargins()` function, 131
adjacency matrix, 333
`aggregate()` function, 136
`all()` function, 35–39
analogous operations, resizing matrices, 74
anonymous functions, 99, 187–188
antibugging, 287
`any()` function, 35–39
application-specific functions, 165
`apply()` function
  applying functions to matrix rows and columns, 70–72
  matrix-like operations, 107
  obtaining variable marginal values, 131
arguments. *See also specific argument by name*
  actual, 9
  default, 9–10
  default values for, 146–147
  formal, 9
arithmetic operations, 30–31, 145–146
`array()` function, 134
arrays
  higher-dimensional arrays, 82–83
  as vectors, 28
`as.matrix()` function, 81
`aspell()` function, 211
`assign()` function
  variables, 109
  writing nonlocals with, 163

atomic pragma, 343
atomic vectors, 85–86
attr() function, 212

B

batch mode, 1
  help feature, 24
  running R in, 3
Bernoulli sequence, 204
biglm package, 321
bigmemory package, 321
binary files, 237
binary search tree, 177–182
body() function, 149, 151
Boolean operators, 145–146
braces, 144
brackets, 87–88
Bravington, Mark, 300
breakpoints, setting, 289–290
  calling browser() function directly, 289–290
  using setbreakpoint() function, 290
breaks component, hist() function, 14
break statement, 141
browser commands, 289
browser() function
  setting breakpoints, 289–290
  single-stepping through code, 288
by() function, 126–127
byrow argument, matrix() function, 61, 236
byte code compilation, 320

C

c %in% y set operation, 202
cache, 346
calculus, 192–193
categorical variables, 121
cbind() function, 12, 74–75, 106–107
c browser command, 289
cdf (cumulative distribution function), 193
ceiling() math function, 190
cell counts, changing to proportions, 130
cex option, changing graph character sizes with, 272–273
c() function, 56–57
Chambers, John, 226
character strings, 251–259
  defined, 11
  regular expressions, 254–257
    forming filenames example, 256–257
    testing filename for given suffix example, 255–256
  string-manipulation functions, 251–254
    gregexpr(), 254
    grep(), 252
    nchar(), 252
    paste(), 252–253
    regexpr(), 253–254
    sprintf(), 253
    strsplit(), 253
    substr(), 253
  use of string utilities in edtdbg debugging tool, 257–259
child nodes, binary search tree, 177
Chinese dialects, aids for learning, 115–120
chi-square distribution, 193–194
chol() linear algebra function, 197
choose() set operation, 202
chunking memory, 320–321
class() function, 212
cleaner code, 172
client/server model, 247
closures, 151, 174–175
cloud() function, 282–283
cluster, snow package, 335
clusterApply() function, snow package, 72, 337, 339–340
code files, 3
code safety, 41
col() function, 69–70
colon operator (:), 32–33
color images, 63
column-major order, matrix storage, 59, 61
combinatorial simulation, 205–206
combn() function, 203
comdat$countabsamecomm component, 206
comdat$numabchosen component, 206
comdat$whosleft component, 206
comma-separated value (CSV) files, 103
comments, 3
complete.cases() function, 105–106
Comprehensive R Archive Network (CRAN), 24, 193, 353

computed mean, saving in variable, 5
concatenating, vectors, 4
connections, 237–238
constructors, 217
contingency tables, 128, 229
control statements, 139–144
  if-else function, 143–144
  looping over nonvector sets, 143
  loops, 140–142
copy-on-change policy, 314–315
cos() math function, 190
counter() function, 175
counts component
  hist() function, 14
  mapsound() function, 116
covariance matrix, generating, 69–70
CRAN (Comprehensive R Archive Network), 24, 193, 353
critical section, OpenMP, 344
crossprod() function, 196
cross-validation, 219, 222
C-style looping, 140
CSV (comma-separated value) files, 103
ct.dat file, 128
cumprod() math function, 190, 191
cumsum() math function, 39, 190–191
cumulative distribution function (cdf), 193
cumulative sums and products, 191
curve() function, 277–278
customizing graphs, 272–280
  adding polygons with polygon() function, 275–276
  changing character sizes with cex option, 272–273
  changing ranges of axes with xlim and ylim options, 273–275
  graphing explicit functions, 276–277
  magnifying portions of curve example, 277–280
  smoothing points with lowess() and loess() functions, 276
cut() function, 136–137

D

data argument, array() function, 134
data frames, 14–15, 101–102
  accessing, 102–104
  applying functions to, 112–120
    aids for learning Chinese dialects example, 115–120
    applying logistic regression models example, 113–115
    using lapply() and sapply() on data frames, 112–113
  matrix-like operations, 104–109
    apply() function, 107
    extracting subdata frames, 104–105
    NA values, 105–106
    rbind() and cbind() functions, 106–107
    salary study example, 108–109
  merging, 109–112
    employee database example, 111–112
  reading from files, 236
  regression analysis of exam grades example, 103–104
data structures, 10–16
  character strings, 11
  classes, 15–16
  data frames, 14–15
  lists, 12–14
  matrices, 11–12
  vectors, 10
debug() function, 288
debugger() function, performing checks after crash with, 291–292
debugging, 285–304
  ensuring consistency in debugging simulation code, 302
  facilities, 288–300
    browser commands, 289
    debug() and browser() functions, 288
    debugging sessions, 292–300
    setting breakpoints, 289–290
    traceback() and debugger() functions, 291–292
    trace() function, 291
  global variables and, 173
  parallel R, 351
  principles of, 285–287
    antibugging, 287
    confirmation, 285–286
    modular, top-down manner, 286
    starting small, 286
  running GDB on R, 303–304
  syntax and runtime errors, 303
  tools, 287–288, 300–302
debug package, 300–301
declarations, 28–29

default arguments, 9–10
deleting
  list elements, 88–90
  matrix rows and columns, 73–78
  a node from binary search tree, 181
density estimates, same graph, 264–266
DES (discrete-event simulation), writing, 164–171
det() linear algebra function, 197
dev.off() function, 3
df parameter, mapsound() function, 116
dgbsendeditcmd() function, 257–258
diag() linear algebra function, 197–198
diff() function, 50–51
dim argument, array() function, 134
dim attribute, matrix class, 79
dimcode argument, apply() function, 70
dimension reduction, avoiding, 80–81
dim() function, 79
dimnames argument, array() function, 134
dimnames() function, 131
dir() function, 245
discrete-event simulation (DES), writing, 164–171
discrete-valued time series, predicting, 37–39
do.call() function, 133
dosim() function, 165
double brackets, 87–88
drop argument, 68, 81
dtdbg debugging tool, use of string utilities in, 257–259
dual-core machines, 341
duplicate() function, 315
dynamic task assignment, 348–350

E

each argument, rep() function, 34
edit() function, 150, 186–187
edtdbg package, 300–302
eigen() function, 197, 201
eigenvalues, 201
eigenvectors, 201
elements
  list, adding and deleting, 88–90
  vectors
    adding and deleting, 26
    naming, 56
embarrassingly parallel applications
  defined, 347–348
  turning general problems into, 350
employee database example, 111–112
encapsulation, 207
end of file (EOF), 238
envir argument
  get() function, 159
  ls() function, 155
environment and scope, 151–159
  functions have (almost) no side effects, 156–157
  function to display contents of call frame example, 157–159
  ls() function, 155–156
  scope hierarchy, 152–155
  top-level environment, 152
EOF (end of file), 238
ess-tracebug package, 300
event list, DES, 164
event-oriented paradigm, 164
example() function, 21–22
exists() function, 230
expandut() function, 218
explicit functions, graphing, 276–277
exp() math function, 189
extracting
  subdata frames, 104–105
  subtables, 131–134

F

factorial() math function, 190
factors, 121
  functions, 123, 136
    aggregate(), 136
    by(), 126–127
    cut(), 136–137
    split(), 124–126
    tapply(), 123–124
  levels and, 121–122
fangyan, 115
fargs argument, apply() function, 70
f argument, apply() function, 70
Fedora, installing R on, 353–354
file.exists() function, 245
file.info() function, 245, 246
filetype criterion, Google, 24
filter() function, 328
filtering, 45–48
  defined, 25
  generating filtering indices, 45–47
  matrices, 66–69
  with subset() function, 47
  with which() selection function, 47–48

findud() function, 50
findwords() function, 90–91
first-class objects, 149
floor() math function, 190
for loop, 306–313
  achieving better speed in Monte Carlo simulation example, 308–311
  generating powers matrix example, 312–313
  vectorization for speedup, 306–308
formal parameters
  mapsound() function, 116
  oddcount() function, 9
formals() function, 149, 151
forming filenames, 256–257
four-element vector, adding element to, 26
fromcol parameter, mapsound() function, 116
functional programming, xxi–xxii, 314–316
  avoiding memory copy example, 315–316
  copy-on-change issues, 314–315
  vector assignment issues, 314
functions, 7–10. *See also* math functions; string-manipulation functions
  anonymous, 187–188
  applying to data frames, 112–120
    aids for learning Chinese dialects example, 115–120
    applying logistic regression models example, 113–115
    using lapply() and sapply() functions, 112–113
  applying to lists, 95–99
    abalone data example, 99
    lapply() and sapply() functions, 95
    text concordance example, 95–98
  applying to matrix rows and columns, 70–73
    apply() function, 70–72
    finding outliers example, 72–73
  default arguments, 9–10
  listing in packages, 358
  as objects, 149–151
  replacement, 182–186
  for statistical distributions, 193–194
  transcendental, 40
  variable scope, 9
  vector, 35–39, 311

G

GCC, 325
GDB (GNU debugger), 288, 327
general-purpose editors, 186
generating
  covariance matrices, 69–70
  filtering indices, 45–47
  powers matrices, 312–313
generic functions, xxi
  classes, 15
  implementing on S4 classes, 225–226
getAnywhere() function, 211
get() function, 159
  looping over nonvector sets, 142
getnextevnt() function, 165
getwd() function, 245
global variables, 9, 171–174
GNU debugger (GDB), 288, 327
GNU S language, xix
GPU programming, 171, 345
GPUs (graphics processing units), 345
gputools package, 345–346
granularity, 348
graphical user interfaces (GUIs), xx
graphics processing units (GPUs), 345
graphs, 261–283
  customizing, 272–280
    adding legends with legend() function, 270
    adding lines with abline() function, 263–264
    adding points with points() function, 269–270
    adding polygons with polygon() function, 275–276
    adding text with text() function, 270–271
    changing character sizes with cex option, 272–273
    changing ranges of axes with xlim and ylim options, 273–275
    graphing explicit functions, 276–277
    magnifying portions of curve example, 277–280
    smoothing points with lowess() and loess() functions, 276
  pinpointing locations with locator() function, 271–272
  plot() function, 262

graphs (*continued*)
  plots
    restoring, 272
    three-dimensional, 282–283
  polynomial regression example, 266–269
  saving to files, 280–281
  starting new graph while keeping old, 264
  two density estimates on same graph example, 264–266
grayscale images, 63
gregexpr() function, 254
grep() function, 109, 252
GUIs (graphical user interfaces), xx

H

hard drive, loading packages from, 356
help feature, 20–24
  additional topics, 23–24
  batch mode, 24
  example() function, 21–22
  help() function, 20–21
  help.search() function, 22–23
  online, 24
help() function, 20–21
help.search() function, 22–23
higher-dimensional arrays, 82–83
hist() function, 3, 13–14
hosts, 345
Huang, Min-Yu, 324

I

identical() function, 55
IDEs (integrated development environments), xx, 186
ifelse() function, 48–49
  assessing statistical relation of two variables example, 49–51
  control statements, 143–144
  recoding abalone data set example, 51–54
if statements, nested, 141–142
image manipulation, 63–66
images component, mapsound() function, 116
immutable objects, 314
indexing
  list, 87–88
  matrices, 62–63
  vector, 31–32
indices, filtering, 45–47
inheritance
  defined, 207
  S3 classes, 214
initglbls() function, 165
input/output (I/O). *See* I/O
installing packages. *See* packages
installing R, 353–354
  downloading base package from CRAN, 353
  from Linux package manager, 353–354
  from source, 354
install_packages() function, 356
integrated development environments (IDEs), xx, 186
intensity, pixel, 63–64
interactive mode, 2–3
interfacing R to other languages, 323–332
  using R from Python, 330–332
  writing C/C++ functions to be called from R, 323–330
    compiling and running code, 325
    debugging R/C code, 326–327
    extracting subdiagonals from square matrix example, 324–325
    prediction of discrete-valued time series example, 327–330
internal data sets, 5
internal storage, matrix, 59, 61
Internet, accessing, 246–250
  implementing parallel R example, 248–250
  sockets, 247–248
  TCP/IP, 247
Internet Protocol (IP) address, 247
intersect() set operation, 202
intextract() function, 243
I/O (input/output), 231–250
  accessing Internet, 246–250
    implementing parallel R example, 248–250
    sockets in R, 247–248
    TCP/IP, 247
  accessing keyboard and monitor, 232–235
    using print() function, 234–235
    using readline() function, 234
    using scan() function, 232–234

reading files, 235
accessing files on remote machines via URLs, 243
connections, 237–238
reading data frame or matrix from files, 236
reading PUMS census files example, 239–243
reading text files, 237
writing files
getting files and directory information, 245
sum contents of many files example, 245–246
writing to files, 243–245
IP (Internet Protocol) address, 247

J

join operation, 109

K

keyboard, accessing, 232–235
printing to screen, 234–235
using `readline()` function, 234
using `scan()` function, 232–234
KMC (k-means clustering), 338–340

L

lag operations, vector, 50–51
`lapply()` function
applying functions to lists, 95
lists, 50
looping over nonvector sets, 142
using on data frames, 112–113
latency, 346
lazy evaluation principle, 52, 147
leaving-one-out method, 219, 222
`legend()` function, 270
`length()` function
obtaining length of vector, 27
vector indexing, 32
levels, factors and, 121–122
`.libPaths()` function, 356–357
library functions, 165
linear algebra operations, on vectors and matrices, 61, 196–201
finding stationary distributions of Markov chains example, 199–201
vector cross product example, 198–199
`lines()` function, 264
Linux package manager, installing R from, 353–354
lists, 12–14, 85–100
accessing components and values, 93–95
applying functions to, 95–99
abalone data example, 99
`lapply()` and `sapply()` functions, 95
text concordance example, 95–98
general operations, 87–93
adding and deleting list elements, 88–90
getting size of list, 90
list indexing, 87–88
text concordance example, 90–93
recursive lists, 99–100
`lm()`function, 15, 208–210
load balance, 349–350
`locator()` function
determining relevant rows and columns, 64–65
pinpointing locations with, 271–272
`loess()` function, 276
`log10()` math function, 189
logical operations, 30–31
logistic regression models, applying, 113–115
`log()` math function, 189
long-run state distribution, Markov modeling, 200
loops, control statements, 140–142
`lowess()` function, 276
`ls()` function
environment and scope, 155–156
listing objects with, 226–227

M

magnifying portions of curve, 277–280
`makerow()` function, 241–242
managers, `snow` package, 335
managing objects, 226–230
determining object structure, 228–230
`exists()` function, 230
listing objects with `ls()` function, 226–227
removing specific objects with `rm()` function, 227–228
saving collection of objects with `save()` function, 228

mapsound() function, 115–116
marginal values, variable, 131
m argument, apply() function, 70
Markov chains, 199–201
MASS package, 23, 356
math functions, 189–193
  calculating probability example, 190–191
  calculus, 192–193
  cumulative sums and products, 191
  minima and maxima, 191–192
matrices, 11–12, 59–83
  adding and deleting rows and columns, 73–78
    finding closest pair of vertices in graph example, 75–78
    resizing matrix, 73–75
  applying functions to rows and columns, 70–73
    apply() function, 70–72
    finding outliers example, 72–73
  avoiding unintended dimension reduction, 80–81
  linear algebra operations on, 196–201
  naming rows and columns, 81–82
  operations, 61–70
    filtering, 66–69
    generating covariance matrix example, 69–70
    image manipulation example, 63–66
    linear algebra operations, 61
    matrix indexing, 62–63
  reading from files, 236
  vector/matrix distinction, 78–79
  as vectors, 28
matrix/array-like operations, 130–131
matrix class, 79
matrix() function, 60
matrix-inverse update method, 222
matrix-like operations, 104–109
  apply() function, 107
  extracting subdata frames, 104–105
  NA values, 105–106
  rbind() and cbind() functions, 106–107
  salary study example, 108–109
matrix-multiplication operator, 12
maxima function, 191–192
max() math function, 190, 192
mean() function, 38
memory
  chunking, 320–321
  functional programming, 314–316
    avoiding memory copy example, 315–316
    copy-on-change issues, 314–315
    vector assignment issues, 314
  using R packages for memory management, 321
merge() function, 109–110
merge sort method, numerical sorting, 347
merging data frames, 109–112
  employee database example, 111–112
metacharacters, 254
methods() function, 210
microdata, 239
minima function, 191–192
min() math function, 190, 191
M/M/1 queue, 165, 168
modes
  batch, 1, 3, 24
  defined, 26
  interactive, 2–3
modulo operator, 44
monitor, accessing, 232–235
  using print() function, 234–235
  using readline() function, 234
  using scan() function, 232–234
Monte Carlo simulation, achieving better speed in, 308–311
multicore machines, 340–341
mutlinks() function, 336
mutual outlinks, 333–334, 341–342
mvrnorm() function, MASS package, 23, 356

N

named arguments, 146–147
names() function, 56
naming
  matrix rows and columns, 81–82
  vector elements, 56
NA values
  matrix-like operations, 105–106
  vectors, 43
n browser command, 289
nchar() function, 252
ncol() function, 79

negative subscripts, 32, 63
network, defined, 247
Newton-Raphson method, 192
`next` statement, 141
`Nile` data set, 5
noise, adding to image, 65–66
nominal variables, 121
nonlocals
  writing to with superassignment operator, 161–162
  writing with `assign()` function, 163
nonvector sets, looping control statements over, 143
nonvisible functions, 211
`nreps` values, 205
`nrow()` function, 79
NULL values, 44

O

object-oriented programming. *See* OOP
objects. *See also* managing objects
  first-class, 149
  immutable, 314
`oddcount()` function, 7, 140
`omp barrier` pragma, OpenMP, 344
`omp critical` pragma, OpenMP, 344
`omp single` pragma, OpenMP, 344–345
OOP (object-oriented programming), xxi, 207–230
  managing objects. *See* managing objects
  S3 classes. *See* S3 classes
  S4 classes, 222–226
    implementing generic function on, 225–226
    vs. S3 classes, 226
    writing, 223–225
OpenMP, 344–345
  code analysis, 343
  `omp barrier` pragma, 344
  `omp critical` pragma, 344
  `omp single` pragma, 344–345
operations
  list, 87–93
    adding and deleting list elements, 88–90
    getting size of list, 90
    list indexing, 87–88
    text concordance example, 90–93
  matrix, 61–70
    filtering, 66–69
    generating covariance matrix example, 69–70
    image manipulation example, 63–66
    indexing, 62–63
    linear algebra operations, 61
  matrix/array-like, 130–131
  vector, 30–34
    arithmetic and logical operations, 30–31
    colon operator (`:`), 32–33
    generating vector sequences with `seq()` function, 33–34
    repeating vector constants with `rep()` function, 34
    vector in, matrix out, 42–43
    vector in, vector out, 40–42
    vector indexing, 31–32
operator precedence, 33
`order()` function, 97, 194–195
outliers, 49

P

packages, 355–358
  installing
    automatically, 356–357
    manually, 357–358
  listing functions in, 358
  loading from hard drive, 356
parallel R, 333–351
  debugging, 351
    embarrassingly parallel applications, 347–348
    turning general problems into, 350
  implementing, 248–250
  mutual outlinks, 333–334
  resorting to C, 340–345
    GPU programming, 345
    multicore machines, 340–341
    mutual outlinks, 341–342
    OpenMP code analysis, 343
    OpenMP pragmas, 344–345
    running OpenMP code, 342
  `snow` package, 334–340
    analyzing `snow` code, 336–337
    k-means clustering (KMC), 338–340
    running `snow` code, 335–336
    speedup, 337–338

snow package (*continued*)
  sources of overhead, 346–347
    networked systems of computers, 346–347
    shared-memory machines, 346
  static vs. dynamic task assignment, 348–350
parent.frame() function, 156
paste() function, 252–253, 257, 269
PDF devices, saving displayed graphs, 281
pdf() function, 3
Pearson product-moment correlation, 49
performance enhancement, 305–321
  byte code compilation, 320
  chunking, 320–321
  functional programming, 314–316
    avoiding memory copy example, 315–316
    copy-on-change issues, 314–315
    vector assignment issues, 314
  for loop, 306–313
    achieving better speed in a Monte Carlo simulation example, 308–311
    generating powers matrix example, 312–313
    vectorization for speedup, 306–308
  using R packages for memory management, 321
  using Rprof() function to find slow spots in code, 316–319
  writing fast R code, 306
Perron-Frobenius theorem, 201
persp() function, 22, 282
pixel intensity, 63–64
plot() function, xxi, 16, 262
plots
  restoring, 272
  three-dimensional, 282–283
plyr package, 136
pmax() math function, 190, 192
pmf (probability mass function), 193
pmin() math function, 190, 191
pointers, 159–161
points() function, 269–270
polygon() function, 275–276
polymorphism
  defined, xxi, 207
  generic functions, 208
polynomial regression, 219–222, 266–269
port number, 247
powers matrix, generating, 312–313
pragmas, OpenMP, 343–345
preda() function, 38
principle of confirmation, debugging, 285–286
print() function, 18, 234–235
print.ut() function, 218
prntrslts() function, 165
probability, calculating, 190–191
probability mass function (pmf), 193
procpairs() function, 343
prod() math function, 190
programming structures. *See* R programming structures
Public Use Microdata Samples (PUMS) census files, reading, 239
Python, using R from, 330–332

Q

Q browser command, 289
qr() linear algebra function, 197
Quicksort implementation, 176–177

R

race condition, 343
random variate generators, 204–205
rank() function, 195–196
rbind() function, 12, 106–107
  ordering events, 171
  resizing matrices, 74–75
rbinom() function, 204
R console, 2
.Rdata file, 20
Rdsm package, implementing parallel R, 249
reactevnt() function, 165
readBin() function, 248
read.csv() function, 108
reading files, 235
  accessing files on remote machines via URLs, 243
  connections, 237–238
  reading data frames or matrices from files, 236
  reading PUMS census files example, 239–243
  reading text files, 237

readline() function, 234
readLines() function, 248
reassigning matrices, 73–74
recursion, 176–182
  binary search tree example, 177–182
  Quicksort implementation, 176–177
recursive argument, concatenate function, 100
recursive vectors, 86
recycling
  defined, 25
  vectors, 29–30
reference classes, 160
regexpr() function, 253–254
regression analysis of exam grades, 16–19, 103–104
regular expressions, character string manipulation, 254–257
remote machines, accessing files on, 243
repeat loop, 241–242
repeat statement, 141
rep() function, repeating vector constants with, 34
replacement functions, 182–186
  defined, 183–184
  self-bookkeeping vector class example, 184–186
reshape package, 136
resizing matrices, 73–75
return statement, 8
return values, 147–149
  deciding whether to explicitly call return() function, 148
  returning complex objects, 148–149
REvolution Analytics, 300
rexp() function, 204
Rf_PrintValue(s) function, 304
rgamma() function, 204
.Rhistory file, 20
rm() function, 227–228
rnorm() function, 3, 204
round() function, 40–41, 190
routers, 247
row() function, 69–70
rownames() function, 82
R packages, for memory management, 321
rpois() function, 204
Rprof() function, 316–319
.Rprofile file, 19
R programming structures, 139
  anonymous functions, 187–188
  arithmetic and Boolean operators and values, 145–146
  control statements, 139–144
    if-else function, 143–144
    looping over nonvector sets, 143
    loops, 140–142
  default values for arguments, 146–147
  environment and scope issues, 151–159
    function to display contents of call frame example, 157–159
    ls() function, 155–156
    scope hierarchy, 152–155
    side effects, 156–157
    top-level environment, 152
  functions as objects, 149–151
  pointers, lack of, 159–161
  recursion, 176–182
    binary search tree example, 177–182
    Quicksort implementation, 176–177
  replacement functions, 182–186
  return values, 147–149
    deciding whether to explicitly call return() function, 148
    returning complex objects, 148–149
  tools for composing function code, 186–187
    edit() function, 186–187
    text editors and IDEs, 186
  writing, 161–175
    binary operations, 187
    closures, 174–175
    discrete-event simulation (DES) in R example, 164–171
    when to use global variables, 171–174
    writing to nonlocals with assign() function, 163
    writing to nonlocals with the super-assignment operator, 161–162
RPy module
  installing, 330
  syntax, 330–332
runif() function, 204
running
  GDB on R, 303–304
  OpenMP code, 342

running (*continued*)
  R, 1–2
    batch mode, 3
    first session, 4–7
    interactive mode, 2–3
  snow code, 335–336
runs of consecutive ones, finding, 35–37
runtime errors, 303

S

S (programming language), xix
S3 classes, 208–222
  class for storing upper-triangular matrices example, 214–219
  finding implementations of generic methods, 210–212
  generic functions, 208
  OOP in lm() function example, 208–210
  procedure for polynomial regression example, 219–222
  vs. S4 classes, 226
  using inheritance, 214
  writing, 212–213
S4 classes, 222–226
  implementing generic function on, 225–226
  vs. S3 classes, 226
  writing, 223–225
salary study, 108–109
Salzman, Pete, 285
sapply() function, 42
  applying functions to lists, 95
  using on data frames, 112–113
save() function, saving collection of objects with, 228
saving graphs to files, 280–281
scalars, 10
  Boolean operators, 145
  vectors, 26
scan() function, 142, 232–234
scatter/gather paradigm, 335–336
schedevnt() function, 165, 171
scope hierarchy, 152–155. *See also* environment and scope
sepsoundtone() function, 119
seq() function, 21, 33–34
serialize() function, 248
setbreakpoint() function, 290
setClass() function, 223
setdiff() set operation, 202
setequal() set operation, 202
setMethod() function, 225
set operations, 202–203
set.seed() function, 302
setting breakpoints, 289–290
  calling browser() function directly, 289–290
  using setbreakpoint() function, 290
setwd() function, 245
S expression pointers (SEXPs), 304
shared-memory systems, 341, 346–347
shared-memory/threads model, GPUs, 345
Sherman-Morrison-Woodbury formula, 222
shortcuts
  help() function, 20
  help.search() function, 23
showframe() function, 158
sim global variable, 172–173
simplifying code, 172
simulation programming in R, 204–206
  built-in random variate generators, 204–205
  combinatorial simulation, 205–206
  obtaining same random stream in repeated runs, 205
single brackets, 87–88
single-server queuing system, 168
sink() function, 258
sin() math function, 190
slots, S4 class, 224
snow package, 334–335
  implementing parallel R, 248–249
  k-means clustering (KMC), 338–340
  snow code
    analyzing, 336–337
    running, 335–336
  speedup, 337–338
socketConnection() function, 248
sockets, 247–248
socketSelect() function, 248
solve() function, 197
sorting, numerical, 194–196
sos package, 24
source, installing R from, 354
sourceval parameter, mapsound() function, 116
Spearman rank correlation, 49

speed
  byte code compilation, 320
  finding slow spots in code, 316–319
  for loop, 306–313
    achieving better speed in Monte Carlo simulation example, 308–311
    generating powers matrix example, 312–313
    vectorization for speedup, 306–308
  writing fast R code, 306
Spinu, Vitalie, 300
split() function, 124–126, 336
S-Plus (programming language), xix
sprintf() function, 253
sqrt() function, 42, 189
stack trace, 289
startup and shutdown, 19–20
static task assignment, 348–350
stationary distributions, Markov chains, 199–201
statistical distributions, functions for, 193–194
str() function, 14
string-manipulation functions, 11, 251–254
  gregexpr(), 254
  grep(), 252
  nchar(), 252
  paste(), 252–253
  regexpr(), 253–254
  sprintf(), 253
  strsplit(), 253
  substr(), 253
stringsAsFactors argument, data.frame() function, 102
string utilities, in edtdbg debugging tool, 257–259
strsplit() function, 253
subdeterminants, 199
submatrices, assigning values to, 62–63
subnames argument, subtable() function, 132
subscripting operations, 183
subset() function, 47, 105
subsetting, vector, 4–5
substr() function, 253
subtable() function, 132
suffix, testing filename for given, 255–256
sum() function, 190, 337
summary() function, 15, 18
summaryRprof() function, 319
summing contents of many files, 245–246
superassignment operator (<<-), 9
  simplifying code, 174
  writing to nonlocals with, 161–162
sweep() linear algebra function, 197–198
symmetric matrix, 77
syntax errors, 303

T

tabdom() function, 134
tables, 127–130
  extracting subtable example, 131–134
  finding largest cells in, 134
  functions, 136–137
    aggregate(), 136
    cut(), 136–137
  matrix/array-like operations, 130–131
tags, 86
tapply() function
  vs. by() function, 126–127
  factors, 123–124
  vs. split() function, 124
tbl argument, subtable() function, 132
tblarray array, 133
TCP/IP, 247
termination condition, 177
testing vector equality, 54–55
text, adding to graphs with text() function, 270–271
text concordance, 90–93, 95–98
text editors, 186
text files, reading, 237
text() function, adding text to graphs with, 270–271
t() function, 71, 119, 197
threaded code, 171
threads, 341
three-dimensional tables, 129–130
Tierney, Luke, 334
tocol parameter, mapsound() function, 116
tools
  for composing function code, 186–187
    edit() function, 186–187
    text editors and IDEs, 186
  debugging, 287–288, 300–302

top-level environment, 152
traceback() function, 291–292
trace() function, 291
tracemem() function, 314–315
training set, 37
transcendental functions, 40
transition probability, 200
treelike data structures, 177

U

Ubuntu, installing R on, 353–354
unclass() function, 229
union() set operation, 202
unlist() function, 93
unname() function, 94
unserialize() function, 248
upn argument, showframe() function, 158
upper-triangular matrices, class for storing, 214–219
URLs, accessing files on remote machines via, 243
u variable, 162

V

values
  assigning to submatrices, 62–63
  Boolean, 145–146
  list, accessing, 93–95
  NA, 43, 105–106
  NULL, 44
  return, 147–149
vanilla option, startup/shutdown, 20
variables
  assessing statistical relation of two, 49–51
  categorical, 121
  global, 9, 171–174
  nominal, 121
variable scope, 9
vector assignment issues, 314
vector cross product, 198–199
vector filtering, 307
vector-filtering capability, 176
vector functions, 311
vectorization
  defined, 25
  for speedup, 306–308
vectorized operations, 40
vector/matrix distinction, 78–79
vectors, 10, 25–57
  all() and any() functions, 35–39
    finding runs of consecutive ones example, 35–37
    predicting discrete-valued time series example, 37–39
  c() function, 56–57
  common operations, 30–34
    arithmetic and logical operations, 30–31
    colon operator (:), 32–33
    generating vector sequences with seq() function, 33–34
    repeating vector constants with rep() function, 34
    vector indexing, 31–32
  computing inner product of two, 196
  declarations, 28–29
  defined, 4
  elements
    adding and deleting, 26
    naming, 56
  filtering, 45–48
    generating indices for, 45–47
    with subset() function, 47
    with which() function, 47–48
  ifelse() function, 48–54
    assessing statistical relation of two variables example, 49–51
    recoding abalone data set example, 51–54
  linear algebra operations on, 196–201
  matrices and arrays as, 28
  NA value, 43
  NULL value, 44
  obtaining length of, 27
  recycling, 29–30
  scalars, 26
  testing vector equality, 54–55
  vectorized operations, 39–43
    vector in, matrix out, 42–43
    vector in, vector out, 40–42
vertices, graph, finding, 75–78

W

Web, downloading packages from, 356–358
  installing automatically, 356–357
  installing manually, 357–358

where browser command, 289
which.max() function, 73, 190
which.min() function, 190
which() function, 47–48
whitespace, 233
Wickham, Hadley, 136
wireframe() function, 282–283
wmins matrix, 77
workers, snow package, 335
working directory, 19–20
writeBin() function, 248
writeLines() function, 248
write.table() function, 244
writing, 161
- binary operations, 187
- C/C++ functions to be called from R, 323–324
  - compiling and running code, 325
  - debugging R/C code, 326–327
  - extracting subdiagonals from square matrix example, 324–325
  - prediction of discrete-valued time series example, 327–330
- closures, 174–175
- discrete-event simulation in R example, 164–171
- getting files and directory information, 245
- to nonlocals
  - with assign() function, 163
  - with superassignment operator, 161–162
- S3 classes, 212–213
- S4 classes, 223–225
- summing contents of many files example, 245–246
- when to use global variables, 171–174

X

xlim option, 273–275
x variable, 162

Y

ylim option, 273–275

Z

z variable, 162