R programming a course on theory and mechanics of statistical analysis

R is a statistical data manipulation scripting language for analysis. The R language was originally inspired by the S language developed by Bell Labs on behalf of AT&T.

The case of using R over other resources:

… Open sourced developed, supported, and extended
… Comparable and often superior to commercial enterprise products
… A high-level, general purpose language that is extensible and automatable
… Features of object-oriented, and functional programming languages
… Massive user community for support and growth

R leverages that features of object oriented programming. In general, this feature refers to the inputs and outputs of complex programmatic functions being assigned as objects for single uniform reference across the platform. R is also polymorphic, meaning a single function can be applied to different types of inputs (generic functions). An example of object-oriented programming features in R is illustrated below:

Consider the head( ) and format( ) functions in R:

```
head(x)   →   returns the first or last parts of a vector, matrix, table,
data frame or function.
x  →   is an object
format(x, . . .)   →   formats an R object for pretty printing
. . .   →   refers to additional arguments
head(format(x)) → returns the first part of the object class in the
assigned format
```

Not only can multiple independent functions be combined (polymorphic), but the objects can be applied to multiple functions without specific designation or restriction.

R leverages functional programming in ways like implicit iteration. Rather than being required to code loops, R's functional features allow expression of iterative behavior implicitly. This results in faster run times and lower computational costs. Code becomes more compact, executes faster, requires less debugging, and transitions to parallel programming in a simpler manner.

The Art of R programming A Tour of Statistical Software Design

Norman Matloff

The contents of this, and proceeding documentation is a comprehensive outline, or executive summary, written as cited from ***The Art of R Programming – A Tour of Statistical Software Design*** as written and published by Norman Matloff.

Norm Matloff's Biographical Sketch

Dr. Norm Matloff is a professor of computer science at the University of California at Davis, and was formerly a professor of statistics at that university. He is a former database software developer in Silicon Valley, and has been a statistical consultant for firms such as the Kaiser Permanente Health Plan. He was born and raised in the Los Angeles area, and has a PhD in pure mathematics from UCLA, specializing in probability/functional analysis and statistics.

Norm Matloff's Blog – Upon Closer Inspection

Norm Matloff's Blog – Mad (Data) Scientist

The Art of R Programming – A Tour of Statistical Software Design

R programming a course on theory and mechanics of statistical analysis

The following content listing outlines the topics covered in proceeding documentation:

6 functions λ in R

data structures in R

7 vectors [ ] in R

recycling in R

common vector operations = in R

vectorized operations in R

8 NA and NULL values Y in R

vector names and classes in R

9 matrices and arrays in R

10 matrix subsetting [ ] in R

11 matrix filtering in R

12 applying functions to matrix rows and columns $f(x)$ in R

13 adding and deleting matrix rows and columns [×] in R

14 matrix and vector distinction in R

15 avoiding dimension reduction in R

16 naming matrix rows and columns in R

17 higher-dimensional arrays in R

18 lists in R

19 general list operations × in R

20 adding and deleting list elements + − in R

22 accessing list components and values in R

23 applying functions to lists f( ) in R

recursive lists in R

24 data frames in R

creating data frames in R

accessing data frames in R

25 extracting sub data frames in R

data frames with NA values Y in R

26 rbind( ) and cbind( ) with data frames in R
27 apply( ) with data frames in R
merging data frames in R
28 applying functions to data frames *f*() in R
29 factors and tables in R
30 factors and functions *f*() in R
32 working with tables in R
33 matrix/array-like operations on tables in R
other factor- and table-related functions in R
34 programming structures in R
loops in R
35 looping over nonvector sets in R
36 if-else statements in R
arithmetic and boolean operators and values in R
37 default values for arguments in R
38 return values in R
functions are objects in R
39 environment and scope issues in R
40 function side-effects in R
no pointers in R
41 writing upstairs in R
42 when to use global variables in R
43 closures {} in R
44 recursion in R
45 replacement functions in R
tools for function composition in R
46 creating binary operations in R
anonymous functions in R
47 mathematics and simulations in R

calculating a probability in R
48 cumulative sums · products · minima · maxima in R
49 calculus $\int_0^1 f(x)$ in R
statistical distribution functions in R
50 sorting in R
51 linear algebra operations on vectors and matrices in R
53 set operations ∩∪ in R
54 simulation programming in R
55 object-oriented programming in R
s3 classes **S3** in R
56 implementations of generic methods in R
57 writing s3 classes in R
58 using inheritance in R
59 s4 classes **S4** in R
writing s4 classes in R
60 implementing a generic function on an S4 class in R
class comparisons **S3** *vs* **S4** in R
managing objects in R
63 input/output (i/o) in R
accessing the keyboard and monitor in R
64 reading and writing files in R
reading a dataframe or matrix from a file in R
65 reading a text file in R
66 accessing files on remote machines with urls in R
obtaining file and directory information in R
67 accessing the internet in R
sockets in R
68 string manipulation in R
69 regular expressions [T] in R
70 string utilities in the edtdbg debugging tool in R
71 graphics in R

adding lines to plots in R

72 adding points to plots in R

customizing graphs in R

73 smoothing points in R

graphing explicit functions in R

74 graphics devices in R

75 creating three-dimensional plots in R

76 debugging in R

using debugging facilities in R

77 tracking in R

performing checks after a crash in R

78 ensuring consistency while debugging in R

79 performance enhancement in R

writing fast code in R

the for loop in R

80 functional programming and memory issues in R

81 finding slow code spots in R

byte code compilations in R

conquering max memory in R

82 interfacing to other languages in R

writing C/C++ functions to be called from R

83 debugging R/C code in R

84 using R from Python

85 parallel computing in R

86 analyzing the snow code in R

87 obtainable speedup in R

resorting to C in R

89 other OpenMP programs in R

gpu programming in R

general performance considerations in R

90 debugging parallel code in R