

R Lecture Notes

Paul S.P. Cowpertwait

July 2002

Web Page for R Lecture Notes

Refer to: <http://www.massey.ac.nz/~pscawper/pub/rseries/>

1 Introduction to Statistical Computing

Broadly speaking, the aim of statistics is to understand data and the relationships between variables. Due to the growth in information technology, data sets have grown considerably as the capacity for data collection and storage has increased. In addition, computer processors are more affordable, so that most modern applied statisticians are involved in computing. To analyse data, statisticians and programmers have developed statistical software and programming languages. A good example is the S Programming Language, which was designed especially for programming with data. There are now a large number of software libraries written in S which can be used to solve almost any practical statistical problem.

1.1 Background to R

- The S Language was developed at the Bell Labs in the 1980's.
- The commercial version of S is called S-PLUS,
- The shareware (GNU) implementation of S is called R.
- In this course we shall use R.

R can be downloaded free of charge from: <http://cran.r-project.org/>

1.2 Main features

Essentially R is a suite of software facilities for data manipulation, calculation and graphical display. Amongst other things, R incorporates:

- efficient algorithms for data handling and manipulation;
- a suite of operators for calculations on arrays and matrices;
- a large collection of libraries for data analysis;
- flexible graphical facilities for displaying data;
- the S programming language;
- the flexibility to call C (or Fortran) programs.

1.3 Philosophy behind R

There is a major difference in philosophy between R and most commercial “statistical packages”. In R, a statistical analysis is usually carried out as a series of function calls, with results being stored in variables which can be interrogated by further S functions. Hence, compared with SAS or SPSS, output is minimized enabling the user to focus on the functions required to achieve the required task. These functions can then be stored in source files to enable the analyst to tailor code to their particular application.

1.4 Running R

To start R in Windows, double click the R icon. To start R in Unix or Linux, type ‘R’ at the command prompt. To get out of R, just type: `q()`.

R has an inbuilt help facility. To get more information on any specific named function, for example “boxplot”, the command is:

```
?boxplot
```

On most R installations help is available in HTML format by running

```
help.start()
```

which will launch a Web browser that allows the help pages to be browsed with hyperlinks.

The `help.search` command allows searching for help in various ways: try `?help.search` for details and examples.

The examples on a help topic can normally be run by `example(topic)`

1.5 Keyboard Input

To input the first 6 values of the speed of light data (MM Table 1.1, p.8) into a vector 'speed': `speed <- c(28, 26, 33, 24, 34, -44)`

To list what currently in the R workspace: `ls()`

To remove objects from the workspace: `rm(speed)`

1.6 Basic Syntax

S is an expression language with a simple syntax. It is case sensitive (so `A` and `a` are different symbols)

Elementary commands consist of either expressions or assignments `<-`.

If an expression is given as a command, it is evaluated, printed, and the value is lost. An assignment also evaluates an expression and passes the value to a variable but the result is not automatically printed.

Commands are separated either by a semi-colon (`;`), or by a newline. Elementary commands can be grouped together into one compound expression by braces (`{` and `}`). Comments can be put almost anywhere, starting with a hashmark (`#`); everything after `#` is a comment.

If a command is not complete at the end of a line, R will give a different prompt, by default:

`+`

on second and subsequent lines and continue to read input until the command is syntactically complete.

1.7 Some Demonstations

1.7.1 House Sales Data

Refer to Moore and McCabe Exercise 2.13, Table 2.3, p.123. The idea is to read in the House Sales data, produce a histogram, boxplot, and Normal quantile plot of the 'price' variable. Compare 'price' with 'no. of bedrooms', 'house type', etc.

```
# Reading in the data:

> sales <- read.table('sales.dat', header = T)
> sales[1:4,]
  price  type area nbeds
1 113000 TRI-LE 1350    3
2 113542 BI-LEV 1700    4
3 120000 TRI-LE 1350    4
4 120000 TRI-LE 1350    3

# some plots:

> hist(sales$price)
> hist(sales$price, xlab='Price', main='Histogram of Sales Price')
> hist(sales$price, xlab='Price', main='Histogram of Sales Price', breaks = 10)

> sales.split <- split(sales$price, factor(sales$nbeds))
> boxplot(sales.split)
> boxplot(sales.split, xlab='No. of Bedrooms', ylab='Sale prices')

# some summary statistics:

> attach(sales)
> mean(price)
[1] 177330.1
> fivenum(price)
[1] 113000.0 150500.0 174900.0 187931.5 327500.0
> fivenum(nbeds)
[1] 3 3 3 4 5
> fivenum(area)
[1] 1350 2342 2671 3051 4167
```

```

> summary(price)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
113000 150500 174900 177300 187900 327500
> sd(price)
[1] 45135.56
> cor(price, area)
[1] 0.6768706
> cor(price, nbeds)
[1] 0.2328475

# Regression and more plots:

> plot(area, price)

> price.lm <- lm(price ~ area)
# summary(price.lm) - typical regression output
# print(price.lm)   - brief

# plot with fitted regression line:
> plot(area, price)
> abline(reg=price.lm)

# residual plots:
> plot(price.lm$res)
> plot(price.lm$fit, price.lm$res)
> abline(0,0)
> qqnorm(price.lm$res)
> qline(price.lm$res)

```

1.7.2 Metabolic Rate Data

Refer to Moore and McCabe Exercise 2.10, p.121. The idea is to see how metabolic rate depends on lean body weight for males and females (this is important in nutrition studies).

```
# read in the data:
```

```
> w.dat <- read.table('weight.dat', header = T)
> w.dat[1:4,]
  sex weight rate
1  M   62.0 1792
2  M   62.9 1666
3  F   36.1  995
4  F   54.6 1425
```

```
> attach(w.dat)
```

```
# split data by 'sex' variable
> w.split <- split(weight, factor(sex))
> r.split <- split(rate, factor(sex))
```

```
# look at some plots:
> boxplot(w.split)
> boxplot(r.split)
```

```
# plot 'rate' against 'weight' with different symbols for 'male' and 'female':
```

```
> fivenum(weight)
[1] 33.1 41.6 47.4 51.5 62.9
> fivenum(rate)
[1] 913.0 1196.5 1396.0 1481.0 1867.0
> plot(w.split$F, r.split$F, pch = 1, xlim=c(30, 70), ylim = c(900, 2000), xlab=
> points(w.split$M, r.split$M, pch = 2)
```

```
# fit regression model and add line
> rate.lm <- lm(rate ~ weight)
> plot(w.split$F, r.split$F, pch = 1, xlim=c(30, 70), ylim = c(900, 2000), xlab=
> points(w.split$M, r.split$M, pch = 2)
> abline(reg=rate.lm)
```

1.8 Further Reading

“An Introduction to R”: ‘Preface’ and ‘Introduction’
Introduction to ‘Functional Programming for Data Analysis’:
<http://www.massey.ac.nz/~pscowper/161330/>

1.9 Exercises

Statistical Computing Laboratory: Practical 1