STAT300 – INTRODUCTION – 2006

Prerequisites

Mid semester assignment (Ass9)

Practical test (Ass14) – in MCL3 using linux for internals
Any scientific principle must be supported by evidence. The best evidence, especially in applied science and technology, comes from repeatable, controlled (designed) experiments.

These principles must allow verifiable, reliable predictions - hence the use of mathematical models,
Measurement involves randomness and uncertainty.
Statistics utilizes the mathematics of describing randomness and uncertainty.

(i) providing the means to summarise information (i.e. data) and to make reliable predictions,
(ii) improving understanding of scientific principles by identifying the significance of effects from different components, and
(iii) quantifying the inherent variability of a population.
LECTURE 1:

R programming (in the Linux environment)
• Learn to program so that you are not fazed by non-standard problems.

• Programming is learnt by
  1. Understanding basic principles of computer programming
  2. Belief that there must be a simple way
  3. Copying and hacking code
  4. Practice

• R meets our needs.
1. Make a directory say stat300
3. Download data (xyz.txt) and r file (xyz.r), if it exists, else create xyz.r in an editor (pico?)
4. Start R
5. source(‘‘xyz.r’’)
R software consists of a base package + libraries which contain a vast set of functions.

These functions are used to write programs which

1. read data,

2. fit models,

3. calculate predictions,

4. tabulate and plot results
Example

nconst <- sqrt(2*pi)

- The name of the object (nconst) is arbitrary
- The name of the function (sqrt) must be specified exactly.
- Names in R are case specific.
- Functions can be nested, eg. print(sqrt(2*pi)).
The order of operation is from RHS to LHS

The arrow ("<-") is the assignment operator

An object in R can be ”rich”, in the sense that it can hold much and varied information that is able to be processed by other operations, such as ”plot” and ”summary’, etc

For example

\texttt{plot(lm(y~x))}

will produce a series of diagnostic plots for the regression of y on x.
Plan of attack

An overview of R operations is given in Lecture 1 of the Notes. This will be revisited as needed throughout the unit. For the moment we will introduce functions and methods of operation as they are needed.
Methods of operation

1. [purely interactive]
   This is where you enter R commands, line by line from the terminal.

2. [executing a command file interactively]
   You execute a file of R commands from an R interactive session.

3. [BATCH]
   You execute a file of R commands from the linux OS command line prompt.
Linux tools (on *turing*):

- **pico** – an editor
- **mkdir** – make a directory
- **cd** – change directory
- **lpr** – print a file
- **soffice** – invoke StarOffice, a linux M/S-Office clone
- **ps2pdf** – convert .ps files to .pdf
- **gv** – view .ps and .pdf files
For example, to edit or create a file \texttt{xyz.r}

\texttt{pico xyz.r}

To print a data file \texttt{xyz.txt}

\texttt{lpr -Pm xyz.txt}
We will demonstrate:

The running of a simple example in all 3 modes.

How to save a graph, (and later) import it into a word processor.
Example

Data:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
</tr>
</tbody>
</table>

Use R to enter the data, print it out and plot Y vs X.
The R code: (assumed to be held in the file test.r)

```r
x <- c(1,2,3)
y <- c(10,19,29)
print(cbind(x,y))
plot(y~x)
```
All 3 methods need to be run from the linux command line prompt.

1. From the linux command line, type:

   R

   and then enter each line as shown in the file test.r above.

2. Type

   R

   as before, but now under R type

   source("test.r")

3. From the linux command line, type

   R CMD BATCH test.r
Output from all 3 methods.

Text:

> invisible(options(echo = TRUE))
> x <- c(1,2,3)
> y <- c(10,19,29)
> print(cbind(x,y))
>
>     x  y
> [1,] 1 10
> [2,] 2 19
> [3,] 3 29
> plot(y~x)
>

Graph:
Saving graphs:

1. [pure interactive]
   After the plot command, type:
   ```r
dev.print(device=pdf)
```

   This will save the graphs into a default pdf file "Rplots.pdf"
   ```r
dev.print(device=pdf,file="test.pdf")
```
   will save the graph in file test.pdf.
2. [executing a command line]

   in R enter

   source("testpdf.r")

   where testpdf.r contains

   x <- c(1,2,3)
   y <- c(10,19,29)
   print(cbind(x,y))
   pdf("test.pdf")
   plot(y~x)
   pdf()

   The graph is saved in the pdf file test.pdf, which can now be imported into a Word file using soffice.
3. For Batch operation, type at the linux command line prompt

R CMD BATCH testpdf.r

to produce the graph in test.pdf.
Note that wherever you have seen "pdf", the option of "postscript" could be used. The file extension is then ".ps". Thus "test.pdf" would become "test.ps".
Examples

Example 1

As part of a study in bioinformatics into the distribution of introns, a small simulation produced a sample distribution of 100 introns of type 0, 1 and 2. We wish to check the distribution produced, graphically and numerically.
> n <- 100
> p0 <- 0.4344
> p1 <- 0.3523
> q1 <- p0 + p1
> set.seed(9843)
> r <- runif(n)
> x <- 0 * (r <= p0) + 1 * (r < q1 & r > p0) + 2 * (r >= q1)
> print(x)
   [1] 0 2 1 0 0 2 0 0 1 1 1 2 1 0 0 1 0 2 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1
   [38] 0 0 0 1 0 1 0 1 1 1 0 0 2 1 2 0 1 1 1 2 2 0 0 0 1 2 2 0 2 1 1 0 2 1 0 0 0
   [75] 0 0 2 0 1 2 1 1 1 2 0 0 1 1 0 2 2 0 1 1 2 1 2 0 1 0
> postscript("testsim.ps", horizontal=F, width=6, height=5)
> hist(x)
> postscript()
> stem(x)

The decimal point is 1 digit(s) to the left of the |
<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>38</td>
<td>42</td>
<td>20</td>
</tr>
</tbody>
</table>

> table(x)

```r
tax(x)
```

```text
x
0 1 2
38 42 20
```
Histogram of $x$

Frequency

0.0 0.5 1.0 1.5 2.0

0 10 20 30 40
Example 2

We wish to produce a table of the distribution of the levels within a factor at 3 levels, a, b and c.

```r
> x <- c("a","a","a","a","b","b","b","c","c","c",
"c","d","d","d","d","d","a","a","a","b","b","b",
"c","c","c","c","c","c","c","d")
> print(x)
  [1] "a" "a" "a" "a" "b" "b" "b" "c" "c" "c" "d" "d" "d" "d" "d" "a" "a" "a"
[20] "b" "b" "c" "c" "c" "c" "c" "c" "c" "c" "d"
> table(x)
  x
 a b c d
7 6 9 6
> length(x)
[1] 28
```
Other
Short cuts

`seq(1,10)`

generates

1,2,..., 10

Try it!

`rep(c(1,2,3),4)`

produces

1,2,3, 1,2,3, 1,2,3, 1,2,3
Rows and columns

This simple example shows how to access elements in a data frame, such as rows, columns or individual entries.

The file x.txt contains

1 2
3 4

The R program to demonstrate the above features:


```r
> x <- read.table("x.txt")
> x
   V1 V2
1  1  2
2  3  4
> x[,1]
[1] 1 3
> x[1,]
   V1 V2
1  1  2
> x[1,1]
[1] 1
> proc.time()
[1] 0.80 0.03 0.83 0.00 0.00
>
```