STA122 Lab Session # 1: Introduction to R

Course Instructor: Prof. Merlise Clyde
Teaching Assistant: Debdeep Pati (dp55@stat.duke.edu)

January 13, 2009

1 Launching R

The following is a summary of the functions and contents of three different types of windows in R. All of the windows are accessible from the pull-down Windows menu.

- You may write R commands in the R Console window. This window also displays all the commands R has run through, the results, and error report. This window appears when you launch R. To bring this window to the foreground, click on the window or go to the Windows pull-down menu and choose the R Console. You can type and run commands in this window line by line.
- Alternatively, you can write all the commands in R Editor window and allow R to run part or all of the commands at once. You open this window by clicking the New Script option in the File menu. If you want to redo your analysis at a later time, you can save the scripts. You can also print the scripts by clicking the Print option from the File menu.
- The R Graphics window opens automatically when you create a graph. To bring a graph to the foreground, click on the graphics window or go to Windows pull-down menu and choose the R Graphics window. Graphs can be saved in various formats, such as jpg, png, bmp, ps, pdf, emf, pictex and so on.

2 Getting help in R

- R has an inbuilt help facility. The command is

  ```
  > help(solve)
  ```

  or

  ```
  > ?solve
  ```

- The examples on a help topic can normally be run by

  ```
  > example(topic)
  ```

- The two search functions **help.search()** and **apropos()** can be a huge help in finding what one wants. Examples of their use are:

```
> help.search("matrix")
```

An alternative to help.search is

```
> ??solve
```

(This lists all functions whose help pages have a title or alias in which the text string matrix appears.)

```
> apropos(matrix)
```

(This lists all function names that include the text matrix.)

3 Vectors

3.1 Allocating Vectors

- The following are assignment statements using the *function c()*

  ```
  > x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
  ```

  or

  ```
  > x = c(10.4, 5.6, 3.1, 6.4, 21.7)
  ```

- The further assignment

  ```
  > y <- c(x, 0, x)
  ```

  would create a vector y with 11 entries consisting of two copies of x with a zero in the middle place.

3.2 Vector Arithmetic

- The elementary arithmetic operators are the usual $+, -, *, /, \wedge$ for raising to a power.
- In addition all of the common arithmetic functions are available, e.g., log, exp, sin, cos, tan, sqrt, max and min(select the largest and smallest elements of a vector respectively).
- range is a function whose value is a vector of length two, namely c(min(x),max(x)), length(x) is the number of elements in x, sum(x) gives the total of the elements in x and prod(x) their product.
- Two statistical functions are mean(x) which calculates the sample mean, which is the same as sum(x)/length(x), and var(x) which gives sum$((\text{x} - \text{mean(x)})^2)$/(length(x) $-$ 1) or sample variance.
- If the argument to var() is an n-by-p matrix the value is a p-by-p sample covariance.
- sort(x) returns a vector of the same size as x with the elements arranged in increasing order.

- `> seq(-5, 5, by=.2) -> x`

  generates in x the vector c(-5.0, -4.8, -4.6,. . . , 4.6, 4.8, 5.0).

- `> y <- rep(x, times=5)`

  which will put five copies of x end-to-end in y.

- Another useful version is

  ```
  > z <- rep(x, each=5)
  ```

  which repeats each element of x five times before moving on to the next.

- `x[i]`gives the i-th element of the vector x and `[x>0]`returns a logical array of TRUE and FALSE with the i-th element TRUE if x[i] > 2.

4 Reading data and Plotting through examples

- We will read into R a file that holds population figures for Australian states and territories, and total population, at various times since 1917, then using this file to create a graph. The data in the file are:

  ```
  Year  NSW  Vic.  Qld  SA   WA   Tas. NT ACT Aust.

  1917  1904  1409 683  440  306  193   5  3  4941
  1927  2402  1727 873  565  392  211   4  8  6182
  1937  2693  1853 993  589  457  233   6 11  6836
  1947  2985  2055 1106 646  502  257  11 17  7579
  1957  3625  2656 1413 873  688  326  21 38  9640
  1967  4295  3274 1700 1110 879  375  62 103 11799
  1977  5002  3837 2130 1286 1204 415 104 214 14192
  1987  5617  4210 2675 1393 1496 449 158 265 16264
  1997  6274  4605 3401 1480 1798 474 187 310 18532
  ```

- The following reads in the data from the file austpop.txt in a local directory.

  ```
  > austpop <- read.table("data.txt", header=TRUE)
  ```

  Use of `header=TRUE` causes R to use the first line to get information on the variable names for the columns. If column headings are not included in the file, the argument can be omitted.

- Now type in austpop at the command line prompt, displaying the object on the screen:

  ```
  > austpop
  ```

- The following command prints the column names

  ```
  >names(austpop)
  ```

- For plotting the ACT with different years the following command is used. `pch` stands for the plotting character. col stands for color. The option `pch=16` sets the plotting character to a solid black dot.

```
> plot(ACT ~ Year, data=austpop, pch=16)
```

or

```
> plot(austpop$ACT ~ austpop$Year, data=austpop, pch=16)
```

or using the `attach` command the variable names can be called directly as follows.

```
> attach(austpop)
> plot(Year, ACT,type ="p",pch=1,col=1)
```

Type can be `"p"`for points, `"l"`for lines, `"o"`for overplotted points and lines, etc.

- You can actually change the plot accessories by invoking the title command.

```
title(main = "Growth of Population in the ACT county", sub = NULL,
xlab = "Year", ylab = "ACT")
```

- Suppose we have data (Year, ACT). Let $\text{ACT} = \text{a} + \text{b} * \text{Year}$ be the least squares line. Then to plot the data along with the line, the simplest R command can be

```
> l1<-min(Year)
> l2<-max(Year)
> rangeX<-seq(l1,l2,0.02)
> lines(rangeX, a+b*rangeX)
```

or just

```
> abline(a,b)
```

- We can add the population of a different place say WA in the same figure as

```
> points(Year,WA,pch=2,col=2)
```

- Sometimes in a overlaid plot, we have to adjust the scale of the figure window using the command `xlim` and `ylim` keeping in mind the range of the data.

```
> plot(Year, ACT,type ="p",pch=1,col=1,ylim=c(0,2000))
```

- We can add legend in the following manner

```
> legend(1950,1100,c("ACT","WA"),col=c(1,2),pch=c(1,2))
```