

Introduction to data analysis in R

1. Startup

When you start R, the R console appears with the prompt “>”. This is where you type commands into R; they are executed immediately when you hit Enter. You will often find it more convenient to use a Script window which allows you to type several commands and then execute them all at once, then go back and edit the commands and re-execute them, etc. This is very useful, particularly if you are doing repetitive analyses.

To open a Script window, go to **File...New script**. A new window will appear. You type your commands in the window, one per line. No “>” prompt appears; you simply type the commands in the order you would like them executed. When you are ready to execute the commands in the Script window, click **Edit...Run all**. You can also run a subset of the commands in a window by highlighting them and hitting Ctrl-R (or selecting **Edit...Run line or selection**). You can also run a single line by putting the cursor anywhere on that line and hitting Ctrl-R (the line does not need to be highlighted).

To save a script, select **File...Save** or **Save as**. R scripts should be saved with a .R extension (you will have to type this explicitly; R will not automatically append it to the script name).

To change the default directory where script files are saved or looked for, select **File...Change dir**. This will also be the directory where data files, graphs, etc. are looked for and saved.

You can add comments to a script by preceding the comment with the # sign, e.g.,

```
> median(x)      # computes median
```

2. Help

Help on R commands is available in a couple of different ways. If you know the name of the command, type it with a ? in front at the command prompt, e.g.,

```
> ?mean
```

You can also type

```
> help(mean)
```

You can also access help on commands through the **Help** menu.

3. R as a calculator

```
> 4+2
```

```
[1] 6
```

```
> (11+5)/(2^3)
```

```
[1] 2
```

```
> sqrt(11+5)
```

```
[1] 4
```

Other functions include: `log()`, `exp()`, `log10()`, `abs()`, `sin()`, `cos()`, `gamma()`, `lgamma()`, `factorial()`, `choose(,)`. Functions can be combined:

```
> log(factorial(10))
```

```
[1] 15.10441
```

4. Variables

You can store objects by assigning a name using the assignment operator “=” (you can also use “<-”: “less than” followed by “minus”; you will see this in much of the documentation, so it is good to be aware of it). R is case sensitive (A is different from a). Valid names in R can be of any length, must start with a letter, but can contain any combination of upper and lower-case letters, numbers, and periods.

```
> aa = sqrt(13*3)
> aa
[1] 6.244998
```

You can use named variables in calculations, as in the following:

```
> N = 100
> d = 2
> N/d
[1] 50
```

5. Vectors

You can create and store vectors in R using the `c()` function (concatenate or combine). If you have data on a single variable, it can be stored as a vector.

```
> x = c(2, 3, 5, 1, 4, 4)
> x
[1] 2 3 5 1 4 4
```

Most functions which operate on a single number will also operate elementwise on vectors and will yield a vector as the result. For example:

```
> x = c(1, 4, 9, 16)
> sqrt(x)
[1] 1 2 3 4
```

```
> a = c(1, 4, 12)
> b = c(1, 2, 3)
> a/b
[1] 1 2 4
```

The capability to do elementwise arithmetic with vectors means that loops can be avoided in many R calculations.

6. Data

Data are stored in two primary ways in R: single variables in vectors and sets of variables in data frames. A data frame is a spreadsheet in R with rows as cases and columns as variables. It is really just a collection of vectors representing the variables. There are several different ways to enter data into either a vector or a data frame.

Entering data

1. Into a vector using the `c()` function:

You can enter a single set of values into a vector using `c()` (which means “combine”):

```
mcgwire = c(49, 32, 33, 39, 33, 42, 39, 52, 58, 70, 65, 32, 29)
sex = c("Male", "Male", "Female", "Female", "Male")
```

2. Into a vector from a file using the `scan ()` function.

If your data consists of a single variable (that is, a single set of values like the heights of everybody in the class), then you can enter the values into a text file (using Wordpad, for example), with one or more data values per line separated by spaces. Save the file as a .txt file. Then read it in as a vector with the command

```
> name = scan("file.txt")
```

If you used commas to separate the values, then

```
> name = scan("file.txt", sep=",")
```

3. Into a data frame From a file using the `read.csv ()` function.

The easiest way to create a data frame with multiple variables per case is to first enter the data into a spreadsheet using a program such as Excel. Put the variable name at the top of each column. Variables can be numerical or character (such as Male, Female). Then save the file as a .csv file (that means "comma separated values") by choosing (in Excel) Save As...Other Formats and choosing .csv.

To read the file into a data frame in R:

```
> name = read.csv("file.csv")
```

4. From a file using the `read.table ()` function.

The `read.table ()` command is like `read.csv ()` but is more flexible and allows you to specify the separator (`sep=`), whether or not the variables names are at the top of the file (`header=T` or `F`), etc.

7. Working with data vectors

There are many statistical and graphical functions that can be applied to a single data vector, `x`.

For quantitative variables:

```
> mean(x)
> median(x)
> min(x)
> max(x)
> sum(x)
> length(x)
> summary(x)
> sd(x)
> fivenum(x)
> IQR(x)

> hist(x)
> hist(x,breaks=20) # histogram with approx. 20 intervals
> boxplot(x)
> boxplot(x,y) # side-by-side boxplots of two data vectors, x and y
```

For categorical variables:

```
> table(x)
> prop.table(table(x))
> barplot(prop.table(table(x)))
```

See below for some options on any plot.

8. Working with data frames

A data frame `d` is a spreadsheet. It can be viewed and edited with the command:

```
> fix(d)
```

(you must close the spreadsheet before you can do anything else in R).

The variables in a data frame have names; they can be viewed with

```
> names(d)
```

If the names of the variables in the data frame are, say, `sex`, `height`, `weight`, then I can refer to the variables individually as `d$sex`, `d$height`, and `d$weight`. These are vectors and I can apply the methods in the previous section to any of the variables individually, e.g., `hist(d$weight)`.

I can also get a summary of all the variables in the data frame with

```
> summary(d)
```

The number of rows in the data frame is

```
> nrow(d)
```

I can create a data frame from data vectors. For example,

```
> x = c(3, 4, 1, 9, 5)
> y = c(17, 13, 25, 11, 6)
> d = data.frame(x, y)
```

creates a data frame with variables named `x` and `y`.

Methods for pairs of categorical variables:

```
> t1 = table(d$var1, d$var2)
> prop.table(t1) # proportions out of the whole table
> prop.table(t1, 1) # proportions within rows
> prop.table(t1, 2) # proportions within columns
```

The `barplot` command can be used to make segmented or clustered bar charts (to show the relationship between two categorical variables). However, it must be done on a table with proportions within columns to work correctly.

```
> barplot(prop.table(t1, 2), legend.text=T) # segmented barplot
> barplot(prop.table(t1, 2), beside=T, legend.text=T) # clustered
barplot
```

Methods for pairs of quantitative variables:

```
> cor(d$var1, d$var2) # correlation
> plot(d$var1, d$var2) # scatterplot
```

You can create side-by-side boxplots of a quantitative variable for all the categories of a categorical variable in a data frame. If data frame `d` has variables `height` and `sex`, then

```
> boxplot(height ~ sex, data=d)
```

creates side-by-side boxplots of height for males and females. The `data=d` statement told R which data frame to use so we could specify the variables by their names rather than by `d$sex` and `d$height`.

9. Other graphics options

There are many aspects of plots that you can control; type `?par` at the command prompt for a list. Some common options which apply to most plots:

```
xlab = "Put label here"      # label for x-axis
ylab= "Put label here"      # label for y-axis
main= "Put title here"      # title for plot
xlim=c(a,b)    # force x-axis to run from a to b (a,b are numbers)
ylim=c(c,d)    # force y-axis to run from c to d
```