Introduction to R

R is a free open-source software that is available for (virtually) all operating systems. It can be downloaded at `www.r-project.org`.

The present document provides examples without a thorough description of the syntax and results. The explanations in each section should be enough to allow you to understand the logic of the examples. Try modifying the examples to see if you understood correctly and consult the help whenever needed.

1 Good Working Habits

R is a command-line program. The most convenient way to save your work is to type the commands you use in a text editor, say notepad, and to save them. You can then reproduce the operations you used on any computer whenever needed.

Before leaving R, you have to choose whether you want to save the image or not. The image is the current value of all the variables you created. When you choose to save, the values of all variables will be reloaded next time you start R. If you do not save, you will start in blank (with no variables defined yet).

2 Getting Help

If you know the name of a command and need help with it, `help(command name)` or simply `?(command name)`. Otherwise, you can browse the help document `help.start()` or from the R menu, choose `Help->Html Help`.

3 Creating Objects

The symbols `<-` , `=`, `_` are equivalent and assign a value to an object. If the object did not exist previously, it will be created; otherwise, the old value will be replaced by the new value assigned. In the examples below, `a:b` (`a`, `b` are both integers) means from `a` to `b`. So, `1:3` is equivalent `c(1,2,3)`.

```
x=2
x=c(1,4,5.6,9)
x=1:10
x=rep(1:3,c(2,5,3))
x=rep(c("A","B","C"),each=5)
```

Typing `x` will print the value of `x` on the screen. The command `c()` creates a vector from a list of elements; `rep` builds a sequence of numbers.

To list existing objects, use `ls()`. To delete the object `x`, use `rm(x)`.

4 Operations on Vectors

A special feature of R is that mathematical operations are applied to every element of a vector or matrix.

```
mass=c(100,140,98,34)
no=c(3,4,3,1)
mass/no
sqrt(no)

p=c(0.1,0.15,0.18,0.19)
y=log(p/(1-p))
exp(y)/(1+exp(y))
```

5 Simple Functions

Some simple functions can be applied to a vector. The vector is then treated as a data set. For instance :

```
x=c(100.3,102.5,104.8,132.1,124.1)
mean(x)
var(x)
sqrt(var(x))
quantile(x,c(.25,.75))
median(x)
```

Remember that `help(function name)` will give you a full description of any R function. Try `help(quantile)` for instance.

6 Types of Variables

There are different types of variables. The most common are numeric, logical and factor. Numeric variables are interpreted as numbers, logical variables as TRUE or FALSE, factor variables as categories. It is possible to convert variables from one type to another. Some functions will behave differently depending on the type of variable provided as input. In the following examples, pay attention to the results produced by `sum(x)` and `sum(y)`.

```
x=rep(1:3,each=5)
sum(x)
y=as.factor(x)
sum(y)
as.factor(rep(c("Male","Female"),c(4,5)))
```

Note that when logical variables are converted to numeric, `TRUE` is converted to `1` and `FALSE` is converted to `0`. The following example takes advantage of that conversion.

```
x=1:10
x<3
sum(x<3)
```

7 Reading Part of a Vector

Using the brackets `[]` allows you to access a part of a vector. In the following example, # is the comment symbol in R. Anything behind # will be ignored by the R complier.

```
x=11:20
x[4]
x[c(3,4)]
x[-7] # vector x, with its 7th element deleted
x>13
x[x>13]
```

In the last case, when a logical vector (of the same length as the variable) is provided, only entries corresponding to TRUEs are kept.

8 Reading Parts of a Matrix

It is possible to define matrices as well. The syntax used for vectors still applies; a comma is used to separate rows from columns. In the example below, the first argument of function `matrix` specifies the values to be stored in the matrix. The second and third arguments are the number of rows and columns of the martix. By default, R stores the values in the matrix by columns. If you want R to store the values by rows, you have to specify the option `byrow=T`. In R, `T` means `True` and `F` means `False`.

```
x=matrix(1:12,3,4)
x
y=matrix(1:12,3,4,byrow=T)
y
x[2,2]
x[-2,]
x[,-1]
x>5
x[x>5]
```

A blank means that everything (all rows or all columns) is kept. Note that omitting the comma results in treating the matrix as a vector.

Mathematical operations are made component-wise on matrices, just as with vectors. There is a special symbol for the matrix/vector product `%*%`. By default in R, a vector is a column vector. That is why we have to use the transpose function `t` in the following command.

```
t(1:3) %*% 1:3
```

9 Creating Loops

To repeat a set of command many times, you can use a loop.

```
x<-matrix(seq(0,0,length=100),10,10)
for(i in 1:10){
  x[i,i]=1
}
```

In the syntax of the `for` loop, you need to specify a counter variable and a vector of values. The counter variable will take all values of the vector sequentially.

```
x<-c(1,5,2,0.5,3)
for(i in x){
  cat("\n",i) # newlines created explicitly by ‘\n’ are printed
}
```

The function `cat` allows you to print characters on the screen.

You can also use `while` loops. Consult the help to learn their syntax.

10 Conditional Statements

Some operations may be needed only under certain conditions. The `if` statement allows you to execute something conditionally.

```
for(i in 1:10){
  if(i>5){
    cat("\nThe number",i,"is greater than 5")
  }
  else {
    cat("\nThe number",i,"is less or equal to 5")
  }
}
```

11 Reading a File

It would be tedious to enter all your data manually every time you need it. It is convenient to store data in a file that can be read when needed. Commands such as `read.csv` and `scan` can then be used to read the data.

Create a text file named *data.txt* containing

```
Brand,Weight,Color
Olymel,300,2
MapleLeaf,295,1
Olymel,305,1
MapleLeaf,308,3
Olymel,310,2
```

It is important that

- each line has the same number of elements,
- each element be separated by a comma,
- the first row contains the variable names.

In the R console, type

```
x=read.csv("data.txt")
x
x[,1]
x[,1:2]
boxplot(x[,2]~x[,1])
```

By default, the first column is converted to factor type because the first column consists of characters.

If you are experiencing trouble with this example, make sure the file `data.txt` was created and is in the current directory. You can change R's current directory in the pull-down menu `File` → `Change Dir` or you could write the full path to your file in the code above. Alternatively, you can use command `setwd(‘‘path’’)` to set the working directory. Note that you must use / instead of \ in the path name.

The function `attach` allows you to refer to the columns of a dataset without specifying the dataset name. Watch out for conflict with existing objects though!

```
x=read.csv("data.txt")
attach(x)
Weight
boxplot(Weight~Brand)
detach(x) # counterpart of attach
boxplot(Weight~Brand)
boxplot(x$Weight~x$Brand) # without attaching x, you have to use x$Weight to
                          # refer to column Weight in object x
```

The function `scan` is useful for reading from a text file containing numbers without special formatting or column names.

12 Graphics

R allows you to produce good-looking statistical graphics that can be copied and pasted to Word, or exported to a file (including the vectorial EPS format).

```
x=rnorm(30)
hist(x)
boxplot(x)
x=rnorm(1000) # a larger sample size
hist(x)
boxplot(x)
```

The function `hist` plots a histogram, the function `boxplot` a boxplot. Note the function `rnorm` which generates pseudo-random standard normal numbers. You might want to look at the histogram of a large sample of such pseudo-random numbers to see if it resembles the standard normal more closely.

To draw side-by-side boxplots, it suffices to provide a vector of factor type indicating to what group each value belongs.

```
x=read.csv("data.txt")
boxplot(x[,2]~x[,1])
```

We read `x` $\sim$ `y` as `x` explained by `y`.

A sequence of values can be plotted by the function `plot`. Refer to the help of `plot` for more details.

```
x=(1:100)/50-1
plot(x,x^2,type='l')
```

It is possible to superimpose graphics with the function `points`. It is important to fix the range of the axis (`xlim, ylim`). to avoid a wrong superposition.

```
x=(1:100)/50-1
plot(x,x^2,type='l',ylim=c(-1,1))
points(x,x^3,type='l',lty=2)
legend(.5,-.7,c("x^2","x^3"),lty=1:2) # make a legend for the plot
```