TD 1 - Initiation to R

Bertrand Iooss

Installation

R can be downloaded from one of the mirror sites in http://cran.r-project.org/mirrors.html. You should pick your nearest location.

Using External Data

R offers plenty of options for loading external data, including Excel, Minitab and SPSS files.

R Session

After R is started, there is a console awaiting for input. At the prompt (>), you can enter numbers and perform calculations.

```
> 1 + 2
[1] 3
```

Variable Assignment

We assign values to variables with the assignment operator "=". Just typing the variable by itself at the prompt will print out the value. We should note that another form of assignment operator "<-" is also in use.

```
> x = 1
> x
[1] 1
```

Functions

R functions are invoked by its name, then followed by the parenthesis, and zero or more arguments. The following apply the function c to combine three numeric values into a vector.

```
> c(1, 2, 3)
[1] 1 2 3
```

Comments

All text after the pound sign "#" within the same line is considered a comment.

```
> 1 + 1      # this is a comment
[1] 2
```

Extension Package

Sometimes we need additional functionality beyond those offered by the core R library. In order to install an extension package, you should invoke the install.packages function at the prompt and follow the instruction.

```
> install.packages()
```

Getting Help

R provides extensive documentation. For example, entering ?c or help(c) at the prompt gives documentation of the function c in R. Please give it a try.

```
> help(c)
```

If you are not sure about the name of the function you are looking for, you can perform a fuzzy search with the apropos function.

```
> apropos("nova")
[1] "anova"              "anova.glm"
   ....
```

Finally, there is an R specific Internet search engine at http://www.rseek.org for more assistance.

Data manipulation

For example, we want to assign the following sequence of numbers :

2 3 0 1 3 0 0 1

We have to type :

```
> typos = c(2,3,0,1,3,0,0,1)
> typos
[1] 2 3 0 1 3 0 0 1
```

This function stores the information inside a vector. A vector can contain information with different modes : numerical, logical (TRUE OR FALSE) or character.

```
> couleur = c("red", "blue")
> couleur
[1] "red" "blue"
```

A regular sequence gives integer numbers, for example from 1 to 8 with :

```
> pages = 1:8
> pages
[1] 1 2 3 4 5 6 7 8
```

Other manipulations are possible:

```
> pages = c(pages, 12, 13, 14)
> pages[12] = 3
```

To build a matrix, we can use the function *matrix()*, but we can also use the functions *rbind()* and *cbind()*:

```
> x = cbind ( 1 : 3 , 4 : 6 )
```

We can then have access to elements of the matrix x by:

```
> x[1,1]
> x[,1]
> x[2,]
```

Types of the data

- The vector (class *vector*) is composed with an ordered collection of elements which have the same mode;
- The factor (class *factor*) is a vector with an additional attribute, the levels, defining a categorical variable ;
- The matrix (class *matrix*) is an array of 2 dimensions of elements which have the same mode ;
- The array (class *array*) is a generalization of the matrix to *n* dimensions;

- The data structure (class *data.frame*) is an array of data composed by one or several vectors/factors with the same or different modes. For example:

```
> listing=data.frame(name=c("Pierre","Paul","Jacques"),age=c(30,45,28),eyes=c(“green","blue","brown"))
```

  You can then try:

```
> view(listing)
> mean(listing$age)
> table(listing$eye)
> summary(listing$age)
> summary(listing)
```

- …

All the objects have 2 intrinsic attributes: the mode and the length:

```
> mode(couleur) ; length(couleur)
[1] "character"
[1] 2
```

For a matrix we use the function dim.

Operators and elementary functions

R has 3 types of operators:

arithmetic	**comparison**	**logical**
+ addition	< smaller	! x NO logical
- substraction	> larger	x & y AND logical
* multiplication	<= smaller or equal	x | y OR logival
/ division	>= larger or equal	xor(x, y) OR exclusive
ˆ power	== equal	
%% modulo	!= different	
%/% integer division		
%*% matrix product		
%o% Kronecker product		

R has the following elementary functions for the treatment of vectors :

c(x,y)	concatenation
length(x)	number of elements
min(x)	minimum
max(x)	maximum
mean(x)	mean
median(x)	median
var(x)	variance
sd(x)	standard deviation
range(x)	variation domain
rank(x)	rank of the elements
sort(x)	vector of ordered elements
order(x)	vector of the indices of the ordered elements
sum(x)	sum of the elements
prod(x)	product of the elements
diff(x)	difference of the consecutive elements
cumsum(x)	cumulated sum

```
> mean(pages)
[1] 6.5
> sd(pages)
[1] 4.421024
```

Other functions are defined on matrices (*dim, nrow, ncol, diag,…*).

Some functions are generic and are applied on some objects :

- *print()* to print on the screen the object content ;
- *plot()* to realize some predefined graphical representation ;
- *summary()* to return a summary of the object content.

Writing and reading the content of an ASCII file

To export some data, we use the command *write.table()*.

```
> write.table(x=pages, file=”file.txt”, quote=F)
```

To read some data from a file, we use the command *read.tabl*e().

```
> pages1 = read.table(“file.txt”,header=T)
> pages ; pages1
```

Graphics

A basic plot:

```
> x=seq (−2 * pi , 2 *  pi , 0.1 )
> plot( x , cos( x ) )
```

To personalize the graphics, you have to look at *par*:

```
> help(par)
```

You can have a demonstration of the R graphical capabilities by typing:

```
> demo(graphics)
```

Random sequences

All probability distributions are available in R in order to obtain the density values, the distribution function, the quantile function or to generate random samples. These functions have the following form :

dfunc() for the density values
pfunc() for the distribution function
qfunc() for the quantiles
rfunc() for random generation

where *func* gives the type of the probability law.

For example, to generate random samples with different probability laws:

```
> rnorm(100, mean=0, sd = 1)              # Normal law (Gauss)
> runif(100, min=0, max=1)                # Uniform law
> rexp(100, rate=1)                       # Exponential law
> rlnorm(100, meanlog=0, sdlog=1)         # Lognormal law
```

To plot the density of the normal law between -5 and 5

```
> x=seq ( −5 ,5 , .1)
> plot( x , dnorm(x), type=”l” )
```