Tutorial to R

Why R?

R is used by a large community in life sciences and finance for purposes in context with large data sets. It is useful due to its implemented statistics functions and plotting capabilities. You can use it to let the computer do repetitive calculations while you just type in the ideas on your work. To be able to do this, you need to understand the syntax of this programming language. This summary paper is made to help you memorize the key aspects of R. You can find a lot of information on R in the internet: www.r-bloggers.com and http://cran.r-project.org/

Working with R

↑	Scroll in command history
Tab, double -Tab	Auto-completion of available variable or function names, shows argument names of functions
help(), ?, ??	question for manuals in R (type q to quit)
.R	Ending for script files
##	Comment line
;	Command separator if given in one line
=	Assignment
e-01	$*10^{-1}$
A,a	Be aware of lower and upper cases in variable and function names!
q()	Quit R console

Arithmetic

+	Addition
-	Subtraction
*	Multiplication
/	Division
%%	Modulo (remainder of division)
^	Potentiation
sqrt()	Square root
sin(), cos(), tan()	Trigonometric functions
log(), log2()	Natural logarithm, logarithm to base 2
exp()	Exponentiation of base e

Rounding

round(number, digits=integer)
→ determine decimal places
signif(number,digits=integer)
→ determine significant digits
floor() → next smaller integer
ceiling() → next larger integer

Comparison

==	equality	
>	Larger than	
<	Smaller than	
>=	Larger or equal to	
<=	Smaller or equal to	
%in%	Is the element in the set?	
TRUE/ FALSE	Boolean results (logicals)	
&	and	Connection of conditions
|	or	

Strings

“f1.txt” is yielded by:
paste(“f”,1,”.txt”,sep=””)
paste(c(“f1”,”txt”),collapse=”.”)

strsplit(“string”,“separator”)
(returns a list!)

substring(“text”, first, last)
(last is optional, predefined as last character of “text”)

Use of functions

Functions are used to save a calculation scheme which can be used repeatedly in different contexts. It takes a list of arguments and returns **one** result. Arguments can also be the result of an other function.

function declaration

```
name = function (argument1, argument2, … , optional_argument = predefined value) {
        command 1; command 2
        command 3
        result = …
        return(result)
}
```

function call

```
name(argument1, argument2, …)
```

Data types

A variable can hold
- numbers,
- characters (strings),
- logicals,
- factors,
- functions.

All variable types (except functions) can be arranged in
- vectors,
- matrices and data.frames,
- arrays and
- lists.

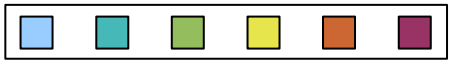
There are functions to convert some datatypes to others.

Vectors, matrices and arrays can hold only one data type, while data.frames and lists can hold mixtures.

Assignment and conversions:

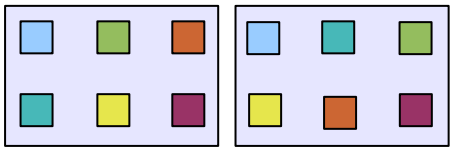
Assign character strings in quotation marks (“string“)!
Numbers assigned in quotation marks cannot be used for calculations. Convert them to numericals with the function as.numeric(). Further conversion functions are: as.integer(), as.character(), as.vector(), as.matrix(), as.data.frame(), as.list(), unlist().

Vector declaration, labeling and referencing:



empty vector: vector = vector(length=6)
sequence: vector = 1:6 (integers) or vector = seq(from, to, by=stepsize or length=integer)
general: vector = c(element1,element2,...)
as.matrix(vector) creates matrix(vector, nrow=6,ncol=1)
names(vector)=c(“blue”,”turquoise”,”green”,”yellow”,”orange”,”magenta”)
The yellow element is vector[4] or vector[“yellow”].

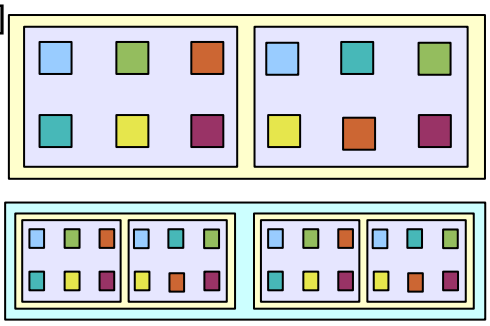
Matrix declaration, labeling and referencing:



[left] mat_l = matrix(vector,nrow=2,ncol=3)
[right] mat_r = matrix(vector,nrow=2,ncol=3,byrow=TRUE)
as.vector(mat_l) returns the vector as it was defined,
as.vector(mat_r) returns

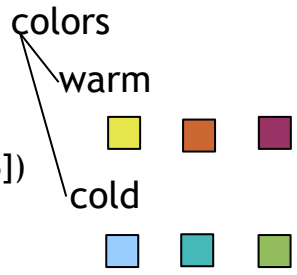
rownames(mat_r)=c(“cold”,“warm”) and colnames() can be used to give headers.
yellow is in mat_l[2,2] and mat_r[2,1] or mat_r[“warm”,1]

Array declaration, labeling and referencing:



array_3dim = array(c(...),dim=c(nrow,ncol,ndim3))
dimnames(array_3dim)[[3]]=c(“bycol”,”byrow”) [→list!]
yellow is in array_3dim[2,2,1] and array_3dim[2,1,2].
array_4dim = array(c(...),dim=c(nrow,ncol,ndim3,ndim4))
Now yellow is in array_4dim[2,2,1,1], array_4dim[2,1,2,1]
as well es in array_4dim[2,2,1,2] and array_4dim[2,1,2,2].

List declaration, labeling and referencing:



empty list: colors=list()
filling elements: colors[[1]]=vector[4:6]; colors[[2]]=vector[1:3]
naming elements: names(colors)=c(“warm”,“cold”)
list of named elements: colors=list(warm=vector[4:6],cold=vector[1:3])
Lists can be extended arbitrarily in length and with sublists.
Yellow is in colors[[“warm”]][1] or colors[[1]][1] or colors$warm[1].

Referencing data.frames

Data.frames are used like matrices, but each columns may hold a different data type.
A column of a data.frame can be referenced by data$columnname.

Variable measures

nchar() : number of characters in a string
length() : number of elements in a vector
nrow(), ncol() : number rows/columns in a matrix/array/data.frame
dim() : number of rows, columns and other dimensions in matrices/arrays/data.frames

Extending dimensions: cbind(),rbind() are used to extend a matrix by columns or rows.

Read and write to and from files

write(“text”,file=”filename.txt”)
write.table(tablevariable,file=”filename.txt”,col.names=NA)
data.frame = read.table(file=”filename.txt”,header=TRUE,sep=”\t”)

Plotting

x11() opens a new window for plotting.
plot()