Introduction to R: Overview of common concepts

Contents

1 Preliminaries

Downloading and installing R

You have to download and install R and the IDE RStudio. Make sure to use the 64bit versions.

Installing R notebooks

In order to work with the problem sets you have to install a number of packages, i.e.

- *rmarkdown*
- *formatR*

- *caTools*
- *knitr*
- *rprojroot*

via the command *install.packages("XXX")*. The installation is only needed once.

Getting help

Functions and expressions used:

- *?objectname*, e.g. *?seq* will open the help page
- *help("objectname")* does the same, e.g. *help("seq")*
- *str(objectname)*, e.g. *str(seq)* will unveil the structure of the object

2 Vectors

A vector can contain only elements of the same class, e.g. character, numeric, integer, complex or logical. Functions and expressions used:

- *c(elements)* constructs a vector with some *elements*
- *seq()* also constructs a vector using some pattern
  important arguments:
  - *from* - starting point of the sequence
  - *to* - endpoint of the sequence
  - *length* - the number of elements in the sequence
  - *by* - the difference between two elements of the sequence

  important special case: if the increment of a sequence is ± 1 you can also use *from:to*, where *from* and *to* are defined as above
- *x[a]* returns the a-th element of the vector x, a can also be a vector
- *rep(x)* repeats x, important arguments are:
  - *times* - number of duplications of x

  - *length.out* - the total number of elements of the resulting vector

- *vector()* creates a vector, important arguments:
  - *mode* - specifies the class of the elements of the vector (and thus the class of the vector itself)
  - *length* - the number of elements of the vector

- *length(x)* returns the number of elements of x

3 Functions

A function in R is another object (of type function). A user written function always looks like the following:

```
myfunction = function(arg1, arg2, arg3 = a, ... ){
statements
return(results)
}
```

- *myfunction* is the name of the user written function
- *arg1 ... arg3* are arguments of the function which have to be specified by the user (if he/she would like to use this function later), notice that the third argument *arg3* is a so called default argument, i.e. if the user does not specify it, it has the value *a*. Default arguments are used very often.
- *statements* is the so called procedure, i.e. the part of the function where something is calculated, plotted etc.
- *results* are the results returned to the user after calling the function

R is a functional language, i.e. a programming language relying heavily on functions.

4 Logical statements

Basic operators to compare statements:

- == - equality

- >, >=, < and <= - self explanatory
- *!=* - no equality
- & and && - logical and
- | and || - logical or

Functions used:

- *xor(a,b)* - exclusive or
- *all()* - tests whether all values are true
- *any()* - tests whether any values are true
- *which()* - returns the **indices** of a vector (or matrix) for which a condition is true

5 Conditional statements

Use if...else statements for conditional statements, e.g.

```
if (condition 1){statement 1}
if (condition 2){statement 2}
:
if (condition n){statement n} else{
last statement}
```

So we have always the signal word (*if* or *else*) followed by the condition (note the brackets!) and the statement, i.e. the actual calculation for this condition (note the curly brackets!). The last condition is the else tree (note that it should be in the same line as the last if tree!).

6 Matrices

Functions and expressions used:

- *matrix()* constructs a matrix, important arguments:
  - *data* - the entries of the matrix (a vector)
  - *nrow* and *ncol* - number of rows and columns

  - *byrow* - matrix is filled by rows (or columns)

- *dim()* - returns the dimension of an object
- *rbind(X, a)* and *cbind(X, a)* - adds to X the vector a either by row or column
- *attributes(a)* - returns the attributes of a, e.g. the dimension of a matrix
- *identical(a,b)* - checks whether the objects a and b are the same
- *rownames(X)* and *colnames(X)* - returns the row- or column names of a matrix
- *View(X)* - opens a new window with the data object X, which is for example a matrix
- *X[a,b]* - returns the element(s) in the a-th row and b-th column of X

7 Loops

Loops are very important, the two most popular versions are for- and while loops:

```
for (i in somevalues){
        statement
}
```

The signal word is for followed by the indexvariable (here i) and a sequence of values for this index (here somevalues, e.g. 1:5). In the curly brackets is the actual procedure which very often depends on the index. For loops are used when you know the number of executions in advance.

```
while (condition){
        statement
}
```

Here the signal word is while. In the round brackets we have some conditional statement, if this condition is true the statement is evaluated until the condition is false once. VERY IMPORTANT: The condition should not be always true - otherwise the loop will run forever. The while loop is used whenever the exact number of repetitions is not known in advance.