Applied Statistics with R

Topic 1: Introduction to R and RStudio

Kaloyan Ganev

2022/2023

Lecture contents

What is R?

The R software

- A programming environment
- An implementation of the S language
- Mostly used for statistical computing and graphs but can do other jobs, too
- Free software (GNU license)
- Available for all major operating systems, 32/64-bit versions available
- Can be downloaded from `http://cran.r-project.org/`
- You will also find many add-on packages at the same site
- Before moving to RStudio it is essential that you have R installed

RStudio

RStudio

- An open source IDE (integrated development environment) for R
- Available for MS Windows, Linux and Mac OS
- There are two versions – desktop and a browser interface to a server
- I will be using RStudio Desktop
- You can download the latest version from `http://www.rstudio.com/products/rstudio/download/`
- Make sure you all have the software installed before moving on

RStudio features

It is a regular IDE, so it possesses 'traditional' features:

- A console for executing single commands – relates directly to the 'original' R console or GUI (depending on the OS)
- Two object browsers:
  - Environment: contains defined variables and their assigned values; also, loaded packages
  - History: contains a list of already issued commands
- Some other browsers – files, plots, packages, help, and viewer; we will discuss them as appropriate
- A code editor

The code editor

- In R, you can type each time row by row in the console
- Although you can recall commands (via up and down arrows), it is not convenient when you have to repeat complex and lengthy blocks of operations
- It is much better to write down the code in a specific file and reuse it as many times as you wish by batch-running it
- You could perform this task in every available text editor (do not confuse with WYSIWYG stuff such as MS Word)
- For example, a popular editor is `Notepad++` (`http://notepad-plus-plus.org/`) but unless you strongly prefer it use the internal editor to save time

The code editor (2)

- In RStudio it is opened via `File -> New file...` and picking the respective option
- To write a batch of R commands, select `R Script`
- A new window opens where code can be typed
- You can save the code batch as a .R file

Some useful stuff

- You can set your default working directory in RStudio in `Tools -> Global options`
- Check your working directory with `getwd()`
- Getting help through the command line: `help()`. Example:

```
help(mean)
```

or

```
?mean
```

- Get function arguments:

```
args(mean)
```

- Examples of function usage:

```
example(mean)
```

R capabilities

What can R do?

- Standard arithmetic operations: addition (+), subtraction (+), multiplication (*), division (/), exponentiation (ˆ)
- Square root: `sqrt()`
- Natural and base 10 log: `log()`, `log10()`
- General-base log: `logb(x,base=q)`
- Important constants readily available: `pi`; `exp(1)`
- `expm1` calculates `exp(x) - 1`
- `log1p` calculates `log(1 + x)`
- Modulus (modulo): `a%%b`
- Trigonometric functions, absolute value, factorial, etc.

Variable assignment

- There are two ways: through `=` or `<-`
- For example,

  ```
  x = 1
  ```

  assigns the value 1 to the variable x
- It is preferable however to get used to the `<-` style
- To delete a variable: `rm()`; several variables can be deleted at once

Data types in R

- Numeric: integers and floating-point numbers
- Logical: TRUE, FALSE or NA
- Character: anything enclosed in “” or ‘’
- Complex: `2+5i`
- Check variable type: `class(x)`
- Note: These are only the so-called ‘atomic’ data types; other data types will be introduced accordingly when needed

Vectors in R

- Several ways to create:

```r
x <- c(1,2,3,4,5,6)
y <- 10:20
seq(from = 3, to = 10)
seq(from = 3, to = 10, by = 0.5)
```

- Can contain all mentioned data types
- If the data types are mixed, the vector uses the type which is the broadest
- Special vectors: `LETTERS` and `letters`
- Empty vectors, example:

```r
y <- vector(mode = "numeric")
```

Subsetting vectors

- Unlike in other languages, indexing starts from 1
- Example: let

```
x <- 10:20
```

- To take the fourth element:

```
x[4]
```

- To take the fifth and the seventh element:

```
x[c(5,7)]
```

Subsetting vectors (2)

- To take all elements except the fifth and the seventh element:

```
x[-c(5,7)]
```

- To take element from 3 to 7:

```
x[3:7]
```

- Check out what the following lines do

```
LETTERS[c(5,10)]
letters[-c(5,10)]
```

Operations with vectors

- Addition:

```
c(1,2,3,4) + c(5,6,7,8)
```

- Multiplication:

```
c(1,2,3,4) * c(5,6,7,8)
```

- Subtraction and division work in the same fashion
- Exponentiation? Check it out

```
c(1,2,3,4)^c(5,6,7,8)
```

What if vectors are of different length?

```r
c(1,2,3,4) + c(10,100)

1/c(1,2,3,4,5)

c(1,2,3,4,5) + c(10,100)
```

Some more on vectors

- `length(x)`: counts the number of elements in x
- `rep(x)`: creates a new vector containing x repeated several times; example:

```
x <- c(1,2,3)
y <- rep(x,times=3)
```

- The `names` property of a vector: has the same length as it, it provides names for each element:

```
v1 <- c(100,150,200)
names(v1) <- c("sugar","flour","milk")
```

Type v1 to see what is printed

Lists

- Recall that vectors can hold only one data type
- Unlike vectors, lists may contain multiple data types (i.e. different mode of elements is possible)
- Lists can also be containers for other data structures, e.g. other lists
- To create a list:

```r
lst1 <- list(121.5, "vacation", TRUE, c(1,2,3), abs, list("zeta"
    ,1+0.5i))
```

- To name elements:

```r
names(lst1) <- c("float","character","logical","vector","function
    ","list")
```

- ...or, you can do both simultaneously:

```r
lst1 <- list(float = 121.5, character = "vacation", logical =
    TRUE, vector = c(1,2,3), func = abs, list = list(name1="zeta"
    ,cplx=1+0.5i))
```

Lists (2)

- To select one element from a list:

```
lst1[[1]]
```

- To select a sublist (this is still a list):

```
lst1[c(2,4)]
```

- You can also select elements by their name:

```
lst1[["func"]]
```

- Multiple elements as a sublist:

```
lst1[c("func", "float")]
```

- Remove one or more elements from a list:

```
lst1[[1]] <- NULL
lst1[c("func","logical")] <- NULL
```

Matrices

- Matrices are in essence vectors
- They have in addition dimensions (shape); vectors have a dimension of NULL
- To create a matrix from a vector:

```r
vec1 <- seq(from = 1, to = 10.5, by = 0.5)
mat1 <- matrix(vec1, 4, 5)
```

- A zero matrix:

```r
mat2 <- matrix(0,5,5)
```

- An identity matrix (e.g. 5×5):

```r
diag(5)
```

Some matrix operations

- Take the two matrices A and B:

```
vec2 <- c(1,2,3,4)
vec3 <- c(5,6,7,8)
A <- matrix(vec2, 2, 2, byrow = TRUE)
B <- matrix(vec3, 2, 2, byrow = TRUE)
```

- Addition and subtraction: straightforward
- Transposition:

```
Aprime <- t(A)
```

- Inverse

```
Ainv <- solve(A)
```

- Solve the equation $\mathbf{Ax} = \mathbf{b}$

```
vec4 <- c(6,7)
b <- matrix(vec4,2,1)
x <- solve(A,b)
```

Some matrix operations (2)

- Element-by-element multiplication:

```
A * B
```

- Matrix multiplication:

```
A%*%x
```

On naming and selection

- Naming matrices' rows and columns

```r
rownames(A) <- c("row1","row2")
colnames(A) <- c("col1","col2")
```

- Take the matrix $\mathbb{Z}$:

```r
vecz <- c(1,2,3,4,5,6,7,8,9)
Z <- matrix(vecz,3,3,byrow=TRUE)
```

- Select one row or one column:

```r
Z[1,]
Z[,3]
```

- Select multiple rows or columns:

```r
Z[1:2,]
Z[,c(1,3)]
```

Arrays

- Matrices are two-dimensional arrays
- Array are essentially vectors which may have multiple dimensions; take for example:

```r
A1 <- 1:24
dim(A1) <- c(2,3,4)
class(A1)
```

- Note: Arrays can be generated from both vectors and lists!
- A natural conclusion is that in R matrices can be not only numeric
- Be careful what operations you perform with such matrices (results can be funny, and sometimes not so)!
- Selection is analogical to matrices; Naming:

```r
dimnames(A1) <- list(c("A", "B"), c("D","E", "F"), c("G","H","I",
    "J"))
```

Factors

- Similar to vectors but yet different from them in important aspects
- R enumerates the unique values in the vector, and those unique values are called *levels*
- Enumeration is done by means of integers: $1, 2, \ldots, k$, where k is the number of unique values (levels)
- Factors are predominantly used to represent categorical variables
- Another possible use is in grouping data
- We will return to factors when necessary

Data frames

- Analogical to the tabular datasets encountered in other statistical software
- Not a matrix, however
- It is a list of vectors (each column is a vector)
- Each column has a name
- All columns must be of equal length
- Selection of elements, rows, and columns is analogical to matrices
- Column selection can also be made by column name