R programming - Basic Concepts - HAX815X

Jean-Michel Marin

January 2025

Faculty of Sciences, University of Montpellier

Introduction - The R Software

- R is a statistical software freely distributed by the CRAN, created in the 90s by R. Ihaka and R. Gentleman with inspiration from S and Lisp:

`http://cran.r-project.org/`

- It is dedicated to statistical analysis and data visualization; knowing that about 80% of the analysis time is dedicated to data preparation, it is also used for data manipulation

Introduction - Structure

- R is available on many operating systems
- R consists of a core base and thematic function libraries grouped under the name *package*
- It is possible to connect R with other languages: C, Fortran, Java, Javascript, Python...
- It is possible to call R functions from Matlab, Excel, SAS, SPSS...
- Connectivity options for all types of databases: RODBC, RMySQL, ROracle, RJDBC, RMongo...

Introduction - Packages

- R was designed as an open and modular language. Many researchers use R, so new methods are often implemented; the transition from research to industry is increasingly rapid
- It is highly likely that someone else has already encountered the same problem as you: possibility of using existing packages, participating in discussions on R-bloggers...

Introduction - Installation

To install R

```
https://ftp.igh.cnrs.fr/pub/CRAN/
```

Choose your platform....

Introduction - Differences

- R is different from other software, so do not try to find analogies: R has its own way of working

- For example, you do not need to sort the data to summarize, aggregate, split, merge... functions exist for that

- Few native graphical user interfaces (GUIs)

Getting Started - Starting a Session

```
R version 4.4.2 (2024-10-31) -- ”Pile of Leaves”
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.
```

Getting Started - Starting a Session

R waits for an instruction: this is indicated by **>** at the beginning of the line; this instruction must be validated by pressing **Enter** to be **executed**

- If the instruction is correct, R executes it and returns the prompt **>**
- If the instruction is incomplete, R returns **+**, in which case you need to complete the instruction or exit with *Esc* or Ctrl+C in console mode

Getting Started - Working Directory and Paths

The place where R stores its objects and where R will write/access scripts and files

- Windows: File/Change working directory
- Linux: R is launched in a directory (via a Terminal)
- Mac: option Change working directory
- RStudio: session/set working directory

Getting Started - Objects and Session

- Useful functions **getwd**, **setwd**

```r
setwd(”/Users/marin/Desktop”)
getwd()
[1] ”/Users/marin/Desktop”
```

- To get help:

```r
help(mean)
help.start()
```

Getting Started - Packages

- Available on CRAN: `https://cran.r-project.org/`
  - Installation via the Packages tab in RStudio
  - Otherwise via the install.packages function

```r
install.packages("abcrf")
```

- Once installed, the package must be loaded

```r
library(abcrf)
```

Getting Started - Objects and Session

- To save one (or more) objects

```
save(x,X,f,file=”mydata.RData”)
```

- To save all objects (when exiting)

```
q()
```

and the question appears

```
Save workspace image? [y/n/c]: y
```

- To save all objects without exiting

```
save.image()
```

Getting Started - Text Editor

- More comfortable (undo-redo, keyboard or mouse, etc.)
- Faster: especially when repeating tasks!
- Clearer: comments can be added directly within the code

RStudio stands out as the current leader

Getting Started - RStudio Installation

RStudio is an application designed to work with R in a rich and complete development environment

`https://posit.co/download/rstudio-desktop/`

RStudio is divided into four panels:

- Text/code editor
- Workspace, history, import management
- Visualization, help panel
- Console

Getting Started - Self-Training

- Plenty of resources available on the web
- MOOCs
- Books
- The **swirl** package
- ...

The Session - Calculations

```
1 + 1
[1] 2
pi
[1] 3.141593
sin(pi / 2)
[1] 1
```

The Session - Commands

- Composed of functions and/or operators
- Operate on objects

```
help(mean)
exp(2) + 3
q()
```

Objects

- **Main data types**
  - Boolean (logical): `TRUE`, `FALSE`
  - Numeric: Integer (`1L`) or double (`3.14`)
  - Characters (character): `'hello'`, `”world”`
  - Empty (null): `NULL`
  - Complex (complex): `2+0i`, `2i`
  - Binary (raw)
- **Data structure types**
  - Monotype: All elements are of the same type (vector, matrix, array)
  - Mixed types: Lists and data frames

In statistics, the **table of individuals/variables** is the key structure: each column represents a variable, where all elements of a column are of the same type, but columns can have different types (qualitative or quantitative variables)

Objects

Creation by assignment using `=` or `<-` or `->`

```
objects()
character(0)
x <- 2
X = 4
4 -> X
objects()
[1] ”x” ”X”
print(X)  # display value
[1] 4
x; X
[1] 2
[1] 4
```

Objects

Useful functions

```
rm(x)  # delete the object
is.vector(X)
[1] TRUE
class(X)
[1] ”numeric”
length(X)
[1] 1
```

Objects

Attributes (additional properties)

```
attributes(X)
NULL
```

Major object types

Vectors

```
x <- c(1:3, 4, 5)
x
[1] 1 2 3 4 5
y <- c(”M”, ”F”, ”F”, ”M”, ”F”)
y
[1] ”M” ”F” ”F” ”M” ”F”
```

Objects

Factors qualitative variables

```
y <- factor(y)
factor(y)
[1] M F F M F
Levels: F M
```

Matrices

```
X <- matrix(c(”R”, ”T”, ”G”, ”Y”), ncol = 2, nrow = 2)
X
     [,1] [,2]
[1,] ”R”  ”G”
[2,] ”T”  ”Y”
```

Objects

Lists composite objects

```
mylist <- list(comp1 = x, comp2 = X)
mylist
$comp1
[1] 1 2 3 4 5
$comp2
     [,1] [,2]
[1,] ”R”  ”G”
[2,] ”T”  ”Y”
```

Objects

Special list, data frame

```
data.frame(var1 = x, var2 = y)
  var1 var2
1    1    M
2    2    F
3    3    F
4    4    M
5    5    F
```

Typical of data tables in statistics. After importing data with `read.table()`, character variables are converted to factors

Creating Objects - Vectors

- Concatenate/collect

```
x <- c(TRUE, FALSE, TRUE)
x
[1]  TRUE FALSE  TRUE
c(x, FALSE)
[1]  TRUE FALSE  TRUE FALSE
```

Creating Objects - Vectors

- Sequence

```
seq(1, 10, by = 2)
[1] 1 3 5 7 9
seq(1, 10, length = 4)
[1] 1 4 7 10
1:10
[1] 1 2 3 4 5 6 7 8 9 10
```

Creating Objects - Vectors

- Repeat

```
rep(c(”hello”,”bye”),2)
[1] ”hello” ”bye”   ”hello” ”bye”
rep(c(”hello”,”bye”),times=2)
[1] ”hello” ”bye”   ”hello” ”bye”
rep(c(”hello”,”bye”),each=2)
[1] ”hello” ”hello” ”bye”   ”bye”
```

Creating Objects - Matrices

- Matrix creation

```
x <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6

y <- matrix(1:2,ncol=1)
y
     [,1]
[1,]    1
[2,]    2
```

Concatenate Objects

- Concatenate

```
rbind(x,z)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    3    2    1

cbind(x,y)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    1
[2,]    4    5    6    2
```

Lists

- From an empty list

```
li <- list() ; li
list()
li[[1]] <- 1:4 ; li
[[1]]
[1] 1 2 3 4
```

- Adding an item

```
li$nouv <- matrix(1:4,nrow=2) ; li
[[1]]
[1] 1 2 3 4
$nouv
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Factors

- Character/numeric vector conversion

```
f <- factor(c(”F”,”M”,”F”,”F”))
f
[1] F M F F
Levels: F M
> levels(f)
[1] ”F” ”M”
> nlevels(f)
[1] 2
table(f)
f
F M
3 1
```

Factors

- Division into classes

```
x <- 1:10
f <- cut(x,breaks=c(1,2,4,10),include.lowest=TRUE)
f
 [1] [1,2]  [1,2]  (2,4]  (2,4]  (4,10] (4,10] (4,10]
 [8] (4,10] (4,10] (4,10]
Levels: [1,2] (2,4] (4,10]
```

Selection in objects

- either by position: in this case a position vector must be specified (it may be of a different length to the object)
- or by Booleans: in this case, the Boolean vector must be the same length as the object to be selected, and only TRUE values are retained

Selection in objects - Comparison operators

- not !
- and &
- or |
- greater (strictly) >
- greater >=
- lower (strictly) <
- lower <=
- different !=
- equal ==
- in a %in% set or match

Selection in objects - Vectors

- by coordinate number

```
x <- c(2,-1,15)
x[c(3,1,2,2,1)]
[1] 15  2 -1 -1  2
```

- by logical operators

```
x>0
[1]  TRUE FALSE  TRUE
x[x>0]
[1]  2 15
```

Selection in objects - Vectors

- by coordinate deletion

```
x[-3]
[1]  2 -1
```

Selection in objects - Matrices

- by coordinate number

```
X[indiceligne,indicecolonne]
X[indiceligne,]
X[,indicecolonne]
```

- by logical operators

- by coordinate deletion

Note that if you select a single column, R returns a vector

Selection in objects - Lists

```
maliste=list(comp1=x,comp2=X)
maliste
$comp1
[1] 1 2 3 4 5
$comp2
     [,1] [,2]
[1,] ”R”  ”G”
[2,] ”T”  ”Y”
```

Selection in objects - Lists

- by coordinate number

```
 maliste[[2]]
     [,1] [,2]
[1,] ”R”  ”G”
[2,] ”T”  ”Y”
```

- by name

```
maliste$comp1
[1] 1 2 3 4 5
maliste[[”comp1”]]
[1] 1 2 3 4 5
```

- by logical operators
- by coordinate deletion

Selection in objects - data-frame

```
df <- data.frame(var1=x,var2=y)
df
  var1 var2
1    1    M
2    2    F
3    3    F
4    4    M
5    5    F
```

- by coordinate number

Selection in objects - data-frame

- by names

```
df[,c(”var2”,”var1”)]
  var2 var1
1    M    1
2    F    2
3    F    3
4    M    4
5    F    5
df$var1
[1] 1 2 3 4 5
```

- by logical operators
- by coordinate deletion

Ordering data

- `sort` sorts a vector
- `order` returns the indices of the sorted data

```
x <- round(rnorm(8), digits =2)
sort(x)
[1] -0.57 -0.24 -0.15  0.02  0.16  0.32  0.62  1.10
sort(x,decreasing = TRUE)
[1]  1.10  0.62  0.32  0.16  0.02 -0.15 -0.24 -0.57
order(x)
[1] 3 7 6 8 4 1 2 5
```

Handling

- `unique` returns the unique values of an object
- `duplicated` returns duplicate values
- `subset` selects a subset using the condition
- `split` splits the dataset according to a criterion
- `aggregate` allows calculations to be made by groups

Merge

It is possible to merge two tables according to a key (see merging merging 2 tables in databases)

Importing data

- R can import almost any file format since there is likely a package available for it
- In RStudio, the **Import Dataset** tool offers options for different formats (Excel, SPSS, SAS, text, etc.)

the best way for all formats create a csv file and use the function ‘read.table‘

Exporting data

use the function ‘write.table‘ to create a csv file

Saving/Restoring

- Use `save()` to store one or more R objects on disk in compressed `.RData` format
- Use `load()` to restore objects, retaining their original names

Database Connections

R supports connecting to various databases via specific packages

- `RMySQL`: MySQL
- `ROracle`: Oracle
- `RPostgreSQL`: PostgreSQL
- `RSQLServer`: SQL Server
- `mongolite`: MongoDB (NoSQL)
- `RSQLite`: SQLite
- `RJDBC`: Generic databases via Java

Database Connections

- open the connection with `dbConnect()`
- query with `dbGetQuery()`
- close the connection with `dbDisconnect()`

```r
conn <- dbConnect(”RMySQL”, host = ”myserver”,
                    port = 123, dbnam = ”database”,
                    user = ”eric”, password = ”aqw”)
res <- dbGetQuery (conn, ”SELECT * FROM matable”)
dbDisconnect(conn)
```