Introduction to R*

Part 1: Overview of R

Wim R.M. Cardoen

Last updated: 10/23/2025 @ 19:12:17

Contents

1 R: What's in the name?

- Scripting language (vs. compiled language)
- Data Analysis Environment

2 Why R?

- Open Source
- Scripting language => rapid prototyping
- The same R-code can run on different OS (MS, MacOs, Linux)
- Most diverse set of statistical tools
- A lot of pre-canned packages/libraries
- Large community (and very active development)
- Job "security" (due to its popularity)
  - IEEE Top Programming Languages 2025
  - IEEE Top Programming Languages 2024
  - IEEE Top Programming Languages 2023

3 From S to R

`People are trapped in history and history is trapped in them. (James Baldwin)`

- **S** programming language: created at Bell Labs (1975-1976).
  Before, statistical computing was mainly done by calling Fortran routines.
  - Richard A. Becker
  - *John M. Chambers*
  - Allan R. Wilks
- 1988: "New S" (aka "Blue Book" by Becker, Chambers & Wilks) introduced a lot of new features:
  - functions (instead of macros)
  - double precision arithmetic
  - interface to Fortan, C[1]
  - etc.
  - became *similar* to the modern versions of S-PLUS and R.
- 1991: *Statistical Models in S* (Chambers & Hastie)
  - formula-notation (~ operator)
  - data frames
  - methods and classes (S3)
- 1998: S4 (latest version of S)
  - more advanced object-oriented features (S4 - quite different from S3) (inheritance & multiple dispatch)
- 1991: **R**oss Ihaka and **R**obert Gentleman (Auckland, New Zealand)

[1] 'Function mode gives information about the mode of an object in the sense of Becker, Chambers & Wilks (1988), and is more compatible with other implementations of the S language. Finally, the function storage.mode returns the storage mode of its argument in the sense of Becker et al. (1988). It is generally used when calling functions written in another language, such as C or FORTRAN, to ensure that R objects have the data type expected by the routine being called. (In the S language, vectors with integer or real values are both of mode "numeric", so their storage modes need to be distinguished.)' R Language Definition, v.4.1.2 (2021-11-01) p. 2

  - started their own implementation of S and named it … **R**. (*"a free implementation of something 'close to' version 3 of the S language"*)
  - borrowed features from Scheme (static/lexical scoping).
- 1995: R becomes open-source.
- 2000: Release of R v.1.0.
- 2003: Start of the R Foundation.
- **S** still exists as a commercial product (**S-PLUS**). It is currently owned by TIBCO Software Inc.

3.1 Links

- Becker, Richard A (1994). A Brief History of S.
- Chambers, John M. (2009). Software for Data Analysis: Programming with R (Statistics and Computing)
- Chambers, John M. (2020). S, R, and data science.
- The R Project for Statistical Computing
- https://www.r-bloggers.com/

4 Using R

- interactively: **R** CLI or in an IDE.
- run as a script containing R code.
- the cloud: e.g. https://rstudio.cloud/

4.1 Links

- Pre-compiled R Binaries (CRAN)
- RStudio IDE
- Latest R Source Code

5 Getting help on R

- Within R itself:
  - **help(cmd)**
  - **help(?cmd)**
  - **help(??cmd)**
  - some keywords require quotes e.g. **?'if'**, **?'for'**

```
help(mean)
?mean()
??mean()
?'if'
```

- External to the R package:
  - The Comprehensive R Archive Network
    - An Introduction to R
    - R Language Definition
  - The Art of R Programming (Normam Matloff)
  - Hadley Wickham's Website

  - R-Bloggers
  - R Mailing List
  - Stack Overflow (R)

6 General statements on R

- R is **case-sensitive** (cfr. C, C++, Python) => 'a' is different from 'A'.
- Symbol names/variables: letters, digits, ., _ are allowed.
  - **must** start with a **.** (dot) or a **letter**.
  - if symbol starts with **.** (dot), then second character **must** be a letter.
  - can't use **reserved** keywords (**?Reserved**), i.e.:
    - **if, else, repeat, while, function, for, in, next, break**
    - **TRUE, FALSE, NULL, Inf, NaN, NA**
    - **NA_integer_,NA_real_,NA_complex_,NA_character_**
  - the aforementioned rules can be overriden by the use of **backticks**, e.g.:
    - **`if`**
    - **`_myvar`**
- R commands are separated by:
  - either **;** (semi colon).
  - or newline.
- R commands can be grouped together using **{}** (curly braces).
- R comments: start with # (pound sign) (limited to the same line)
- History: use **arrow up** & **arrow down** to retrieve previous cmds.
  - .Rhistory: contains the recently used R commands
  - .RData: contains the objects stored in the Global Environment

```
Oefening baart kunst! ("Exercise gives birth to art!" - Dutch proverb)
```

7 Exercises: A taste of R

- Start R or RStudio
- Try out the following commands in R.

7.1 Check R's version, system time and working directory

```
print(R.version.string)
Sys.time()
getwd()
list.files()
```

Note:
You can invoke operating systems (OS) commands **directly**
using the **system** function to obtain the information above, i.e.:

```
system("R --version | head -n 1 | cut -d' ' -f1-4")
system("date +'%Y-%m-%d %T %Z'")
system("pwd")
system("ls")
```

7.2 Creation of variables

```
x1 <- 5
x1
typeof(x1)
ls()
```

```
x2 <- 5L
x2
typeof(x2)
ls()
```

```
x3 <- FALSE
x3
typeof(x3)
```

```
x4 <- "Programming is cool"
x4
typeof(x4)
```

```
x5 <- -15
typeof(x5)
sqrt(x5)
```

```
x6 <- -15+0i
typeof(x6)
sqrt(x6)
```

7.3 Tinkering with vectors

```
y1 <- 1:5
y1
length(y1)
```

```
y2 <- seq(10.,50,by=10)
y2
length(y2)
y2^y1
```

```
y3 <- c("Hello", "world", "!")
y3
s <- paste(y3)
s
```

```
y4 <- c(TRUE,FALSE,TRUE)
sum(y4)
```

7.4 Install the `lobstr` package within R

```
install.packages(pkgs=c("lobstr"),
                 repos=c("http://cran.us.r-project.org"), verbose= TRUE)
packageVersion("lobstr")  # Check version of the package installed
```

Info on the `lobstr` package can be found at:

- cran.r-project.org website
- lobstr website

7.5 Use the newly installated package

- approach 1:

```
x <- 1:10
lobstr::obj_addr(x)
lobstr::ref(x, character=TRUE)
cat(sprintf("Size of the object (Bytes):%s\n", lobstr::obj_size(x)))
```

- approach 2 (more common):

```
x <- 1:10
library(lobstr)
obj_addr(x)
ref(x, character=TRUE)
cat(sprintf("Size of the object (Bytes):%s\n", obj_size(x)))
```

7.6 Removing objects from your environment

```
ls()
rm(c("y1","y2","y3","y4"))
ls()
```