Introduction to R

Alberto Garfagnini

Università di Padova

AA 2023/2024 - R lecture 1

What is R ?

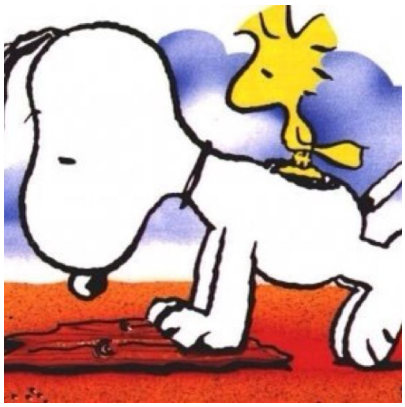
- R is a language and environment for statistical computing and graphics
- similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues.
- R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques
- R is highly extensible
- R is available as Free Software (GNU GPL) and it compiles and runs on a wide variety of UNIX platforms, Windows and MacOS
- The latest R version is `4.3.3 (Angel Food Cake)`, released on February 29, 2024

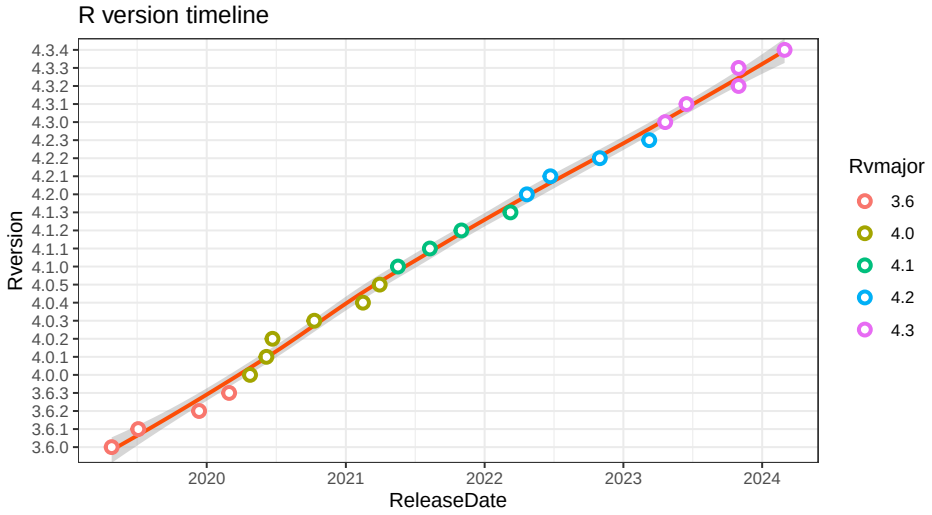
R Web Resources

- R Web Site: https://www.r-project.org/
- R source code: https://cran.r-project.org/src/base/R-4/
- R Developer Page: https://developer.r-project.org/
- A list of changes in the new version can be found here: https://cran.r-project.org/doc/manuals/r-release/NEWS.html

R versions and codenames

- R versioning number is always composed of three numbers, separated by a dot:
  - the first number is the major release version
  - the second number is a minor release code
  - the third and last number is the patch version
- R release names are taken from the Peanuts comics
  - Example: R `4.1.3 (One Push-Up)`
- see Peter Dalgaard talk at useR!2018 `https://www.youtube.com/watch?v=z1vTSdRolgI`





R Web Resources

Official Resources

- several resources available on the internet, starting from the portal of the R project for statistical computing https://www.r-project.org/
- The source code of the R version 4: https://cran.r-project.org/src/base/R-4/
- A list of all the changes among the newly released versions: https://cran.r-project.org/doc/manuals/r-release/NEWS.html
- The R developer site where the plans for new versions and links to technical paper: https://developer.r-project.org/

R blogs

- There are also several blogs that collect and link to newly written articles on R. The two most famous are
  - RWeekly
  - R Bloggers

R mailing lists

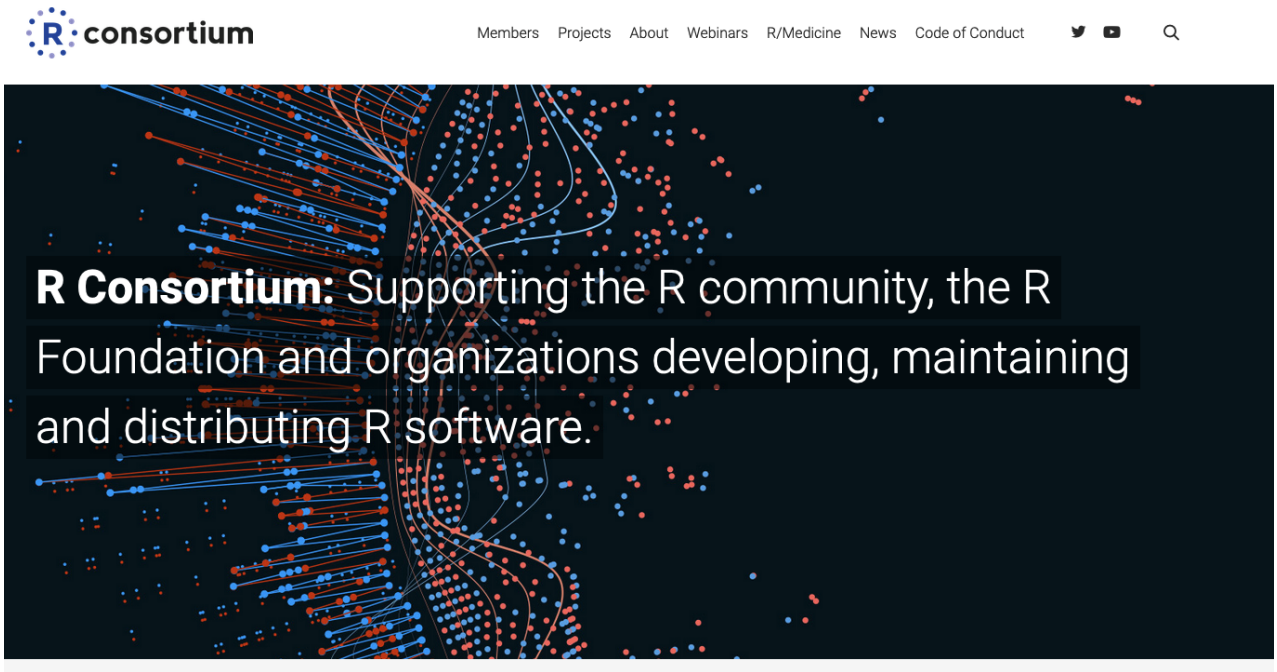
- There are five official mailing lists devoted to different aspects of the R programming environment:

- the R-help mailing list is used for discussion about problems encountered using R and the R packages
- the R-announce mailing list: which sends announcements about the release of new versions of the R programming environment
- the R-packages is a list for the announcements on the availability of new packages (usually available on CRAN
- the R-package-devel has been setup to help on package development in R
- the R-devel has been setup to help on code development in R. It is used to report and discuss on items which are too technical and specific for the R-help audience

- For further information and details, refer to the R Mailing Lists web pages: https://www.r-project.org/mail.html

The R Consortium: https://www.r-consortium.org/

The central mission of the R Consortium is to work with and provide support to the R Foundation and to the key organizations developing, maintaining, distributing and using R software through the identification, development and implementation of infrastructure projects.



Happy 24rd Birthday !

R has turned 24 this year on February 28/29! R 1.0.0 was first released on February 29, 2000, implementing a dialect of the language S, which was developed at Bell Laboratories by John Chambers. Initially, R was written by Ross Ihaka and Robert Gentleman, who were Senior Lecturers at the Department of Statistics of the University of Auckland in Auckland, New Zealand. In addition, a large group of individuals has contributed to R by testing and sending reports.
...

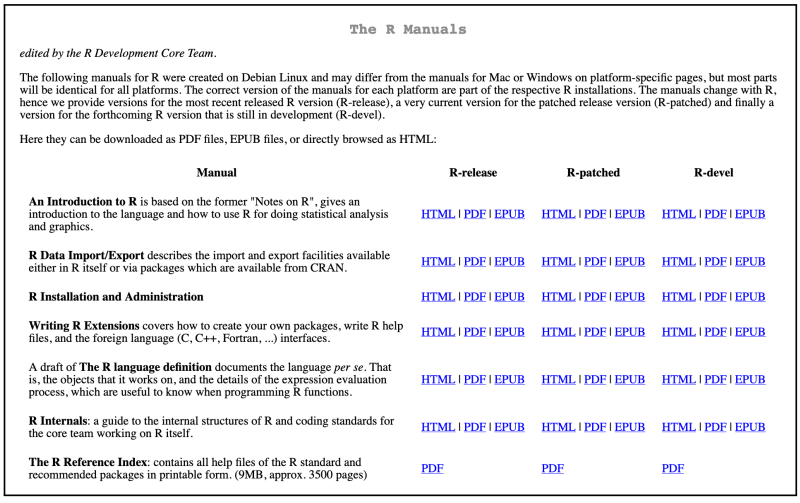
CRAN: the Comprehensive R Archive Network

- Access point to R resources: HOWTOs, FAQ, manuals, examples, ...
- CRAN Web Page: https://cran.r-project.org/
- a list of Frequently Asked Questions is available
  https://cran.r-project.org/faqs.html
- an open access R journal is published online once/twice per year:
  https://journal.r-project.org/
- and several Manuals are available on CRAN Web Page:


The R Journal

Volume 12/2, December 2020

A peer-reviewed, open-access publication of the
R Foundation for Statistical Computing


The R Manuals

edited by the R Development Core Team.

The following manuals for R were created on Debian Linux and may differ from the manuals for Mac or Windows on platform-specific pages, but most parts will be identical for all platforms. The correct version of the manuals for each platform are part of the respective R installations. The manuals change with R, hence we provide versions for the most recent released R version (R-release), a very current version for the patched release version (R-patched) and finally a version for the forthcoming R version that is still in development (R-devel).

Here they can be downloaded as PDF files, EPUB files, or directly browsed as HTML:

Manual	R-release	R-patched	R-devel
An Introduction to R is based on the former "Notes on R", gives an introduction to the language and how to use R for doing statistical analysis and graphics.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Data Import/Export describes the import and export facilities available either in R itself or via packages which are available from CRAN.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Installation and Administration	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
Writing R Extensions covers how to create your own packages, write R help files, and the foreign language (C, C++, Fortran, ...) interfaces.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
A draft of **The R language definition** documents the language *per se*. That is, the objects that it works on, and the details of the expression evaluation process, which are useful to know when programming R functions.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Internals: a guide to the internal structures of R and coding standards for the core team working on R itself.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
The R Reference Index: contains all help files of the R standard and recommended packages in printable form. (9MB, approx. 3500 pages)	PDF	PDF	PDF

R Conferences

- Several R conferences are organized and held every year. An updated and comprehensive list of conferences is maintained at the following server: https://jumpingrivers.github.io/meetingsR/
- The main R Conference is called useR! and is organized every year in different places. The conference has a wide program and has important Keynotes presentation and Tutorials on the R language. The last conferences web sites are the following:
  - https://user2022.r-project.org/
  - https://user2021.r-project.org/
  - https://user2020.r-project.org/
- The relevant presentation are recorded and available through the YouTube dedicated channel https://www.youtube.com/@useRConference_global

How to install R

- R runs on several different platforms and operating systems:
  - all the major Linux Distributions
  - the MacOS. Both Apple Silicon Arm64 and Intel 64-bit processors are supported
  - the Windows operating system
    R for Windows FAQ have been compiled for those willing to install R on Windows
- It is also possible to start form the R source code and build the binaries on specific processors and operating systems
- mooreover, there are several possibility for running R:
  - through a local installation from source code or pre-compiled binaries
  - using a virtualization environment (Docker or Singularity)
  - using the Anaconda distribution

R Local Installation

THE PREFERRED WAY

Linux

- Debian: https://cran.rstudio.com/bin/linux/debian/
- Fedora: https://cran.rstudio.com/bin/linux/fedora/
- RedHat: https://cran.rstudio.com/bin/linux/redhat/
- Suse: https://cran.rstudio.com/bin/linux/suse/README.html
- Ubuntu: https://cran.rstudio.com/bin/linux/ubuntu/

- but most Linux distributions provide specific packages to install R and the R-packages

MacOS

- https://cran.rstudio.com/bin/macosx/

Windows OS

- https://cran.rstudio.com/bin/windows/

From the Source

- It is always possible to download the R source code and compile it: https://cran.rstudio.com/src/base/R-4/

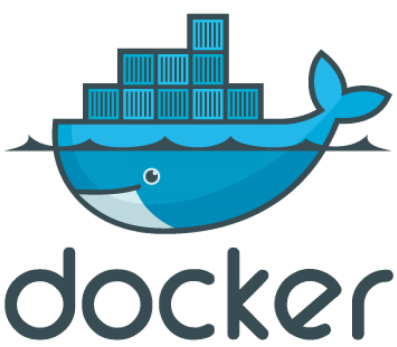
R via Docker

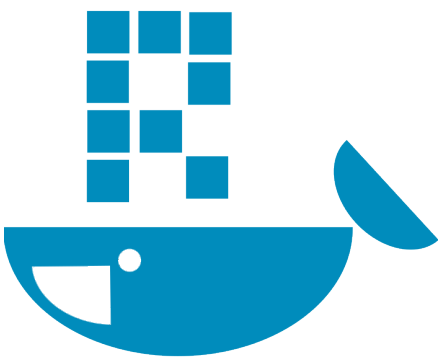
- Docker is an application that realizes what is called Platform as a Service (PaaS), encapsulating a user application and Operating System in a portable container that can run virtually anywere on anybody's infrastructure

- Official Docker containers for the R Environment are provided by the Rocker Project: a collection of Linux based containers are provided accoding to different needs.
- https://rocker-project.org/

- The following images are available:

Image	Built On	Description
rocker/r-base	debian:testing	Install R from debian repo
rocker/r-ver	ubuntu	Install R from source
rocker/rstudio	rocker/r-ver	Add RStudio Server





An example with Docker

1) you need to have Docker installed on your computer
   please refer to the instructions on how to install Docker from Docker Docs Site
   https://docs.docker.com/get-docker/

2) once Docker is installed and running, download the latest `r-base image` from the Docker hub

```
$ docker pull rocker/r-ver:4.2.2
4.2.2: Pulling from rocker/r-ver
...
d244b2a48849: Pull complete
Digest: sha256:5e67719080725a6e7ad0f10fe6e42d0cd79df60474e885cc95a57116873
Status: Downloaded image for rocker/r-ver:4.2.2
docker.io/rocker/r-ver:4.2.2
$
```

3) run the docker image:

```
$ docker run -ti --rm rocker/r-ver:4.2.2
R version 4.2.2 (2022-10-31) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
...
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
```

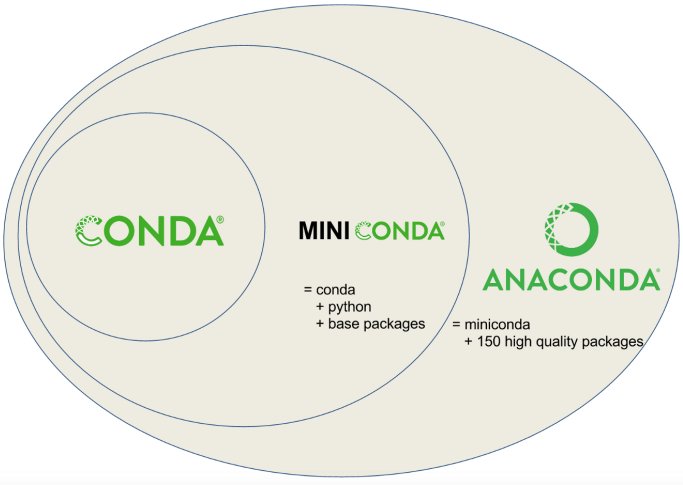

R with ANACONDA

- a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment

- it uses Conda, an open source, cross-platform, language-agnostic package manager and environment management system.

- it is available for Linux, macOS and Windows:
  https://www.anaconda.com/products/distribution

- instructions on how to install R in Anaconda:
  https://docs.anaconda.com/anaconda/user-guide/tasks/using-r-language/



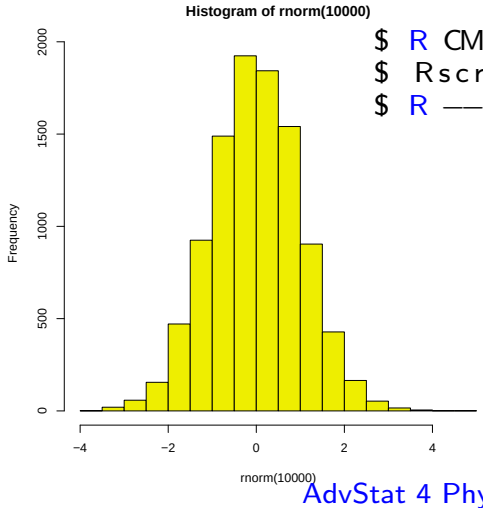
How to run R ?

Two running modes are available:

- interactive mode
- batch mode

Interactive mode R

```
$ R
> pdf("xh.pdf")
> hist(rnorm(1000),
       col="yellow")
> dev.off()
null device
          1
```



Batch mode R

```
file: xh_plot.R
pdf("xh.pdf")
hist(rnorm(1000), col="yellow")
dev.off()
```

```
$ R CMD BATCH xh_plot.R
$ Rscript xh_plot.R
$ R --no-save < xh_plot.R > xh_plot.out
```

← file: xh.pdf

Starting an interactive R session

- the R program can be invoked from the bash shell

```
$ R
```

```
R version 4.2.2 (2022-10-31) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

. . . and ending it

- exiting R can be done through the `q()` function or by typing `<CTRL>-d`

```
> q()
Save workspace image? [y/n/c]:
```

- at the end of an R session, the user can save an image of the current workspace that is automatically reloaded the next time R is started

Getting help in R

- the simplest way, if the name of the function we need help with is known, it to prefix it with the question mark symbol (?)

```
> ?plot

plot                package:graphics                R Documentation

Generic X-Y Plotting

Description:

  Generic function for plotting of R objects.  For more details
  about the graphical parameter arguments, see 'par'.

  For simple scatter plots, 'plot.default' will be used.  However,
  there are 'plot' methods for many R objects, including
  'function's, 'data.frame's, 'density' objects, etc.  Use
  'methods(plot)' and the documentation for these.

Usage:

     plot(x, y, ...)
...
```

Getting help in R

- if the name of the function is not known, but only the subject on which help is needed, the `help.search()` function can be used

```
> help.search("data input")

Help files with alias or concept or title matching 'data input'
using fuzzy matching:


utils::read.DIF          Data Input from Spreadsheet
utils::read.table        Data Input


Type '?PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO'
for entries like 'PKG::FOO-TYPE'.
```

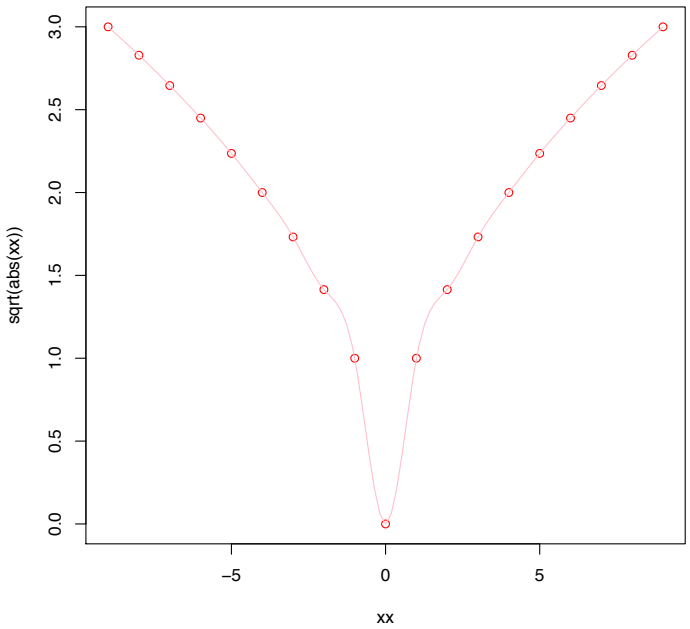
- or with the `find()` and `apropos()` functions

```
> find("read.table")
[1] "package:utils"

> apropos("lm")
 [1] ".colMeans"      ".lm.fit"        "KalmanForecast" "KalmanLike"
 [5] "KalmanRun"      "KalmanSmooth"   "colMeans"       "confint.lm"
 [9] "contr.helmert"  "dummy.coef.lm"  "getAllMethods"  "glm"
[13] "glm.control"    "glm.fit"        "kappa.lm"       "lm"
[17] "lm.fit"         "lm.influence"   "lm.wfit"        "model.matrix.lm"
[21] "nlm"            "nlminb"         "predict.glm"    "predict.lm"
[25] "residuals.glm"  "residuals.lm"   "summary.glm"    "summary.lm"
```


R worked examples

- all R functions have a set of working examples that can be invoked and examined

```
> example(sqrt)

sqrt> require(stats) # for spline

sqrt> require(graphics)

sqrt> xx <- -9:9

sqrt> plot(xx, sqrt(abs(xx)),
           col = "red")

Hit <Return> to see next plot:

sqrt> lines(spline(xx,
                   sqrt(abs(xx)),
                   n=101),
            col = "pink")
```



R packages - Help

- All functions and data sets are organized in packages
- Once R is started, some default packages are loaded by default
- to get help and list the content of a package

```
library(help = package_name)
```

- Example: list all the functions / data sets in the base package

```
> library(help=base)

                Information on package 'base'
Description:

Package:        base
Version:        4.2.2
Priority:       base
Title:          The R Base Package
Author:         R Core Team and contributors worldwide
Maintainer:     R Core Team <do-use-Contact-address@r-project.org>
Contact:        R-help mailing list <r-help@r-project.org>
Description:    Base R functions.
License:        Part of R 4.2.2
Suggests:       methods
Built:          R 4.2.2; ; 2022-10-31 22:33:08 UTC; unix

Index:
.Call                   Modern Interfaces to C/C++ code
.Device                 Lists of Open/Active Graphics Devices
...
with                    Evaluate an Expression in a Data Environment
withVisible             Return both a Value and its Visibility
write                   Write Data to a File
writeLines              Write Lines to a Connection
xtfrm                   Auxiliary Function for Sorting and Ranking
zapsmall                Rounding of Numbers: Zapping Small Ones to Zero
```


R packages - Installing

- A package can be installed from different sources:
  - from the Comprehensive R Archive Network (CRAN) repository
  - from GitHub
  - from other repositories, for instance the BioConductor, a software project for the analysis of genomic data in molocular biology
  - from a local source code tar file
- CRAN is the official repository both for R and for R packages and is organized in a network of web servers around the World
- a list of mirrors can be retrieved and used to find those more convenient
- The University of Padova mantains one mirror https://cran.stat.unipd.it/.
- a package can be installed from CRAN using the `install.packages()` function

```
install.packages("tibble")
```

- the installation from the GitHub repository can be done using the devtools package:

```
install.packages("devtools")
devtools::install_github("tibble")
```

R packages - Loading

- To load an installed package, we can use both the `library()` or `require()` R functions
- main difference: `library()` throws and error if the packages is not found; `require()` prints a message and returns FALSE
- `library()` allows to get a list of all the installed packages:

```
library()

Packages in library '/Library/Frameworks/.../4.2/Resources/library':

abind                   Combine Multidimensional Arrays
ADGofTest               Anderson-Darling GoF test
airports                Data on Airports
arrayhelpers            Convenience Functions for Arrays
ash                     David Scotts ASH Routines
askpass                 Safe Password Entry for R, Git, and SSH
assertthat              Easy Pre and Post Assertions
...
zip                     Cross-Platform 'zip' Compression
zoo                     S3 Infrastructure for Regular and Irregular
                        Time Series (Zs Ordered Observations)
```

- Example: to load the tibble package, we can use the following code:

```
library(tibble)
```


R packages - Listing

- with the command `installed.packages()` it is possible to retrieve a list of all the installed packages

```
pkg <- installed.packages()
df_pkg <- data.frame(pkg)
names(df_pkg)
#  [1] "Package"              "LibPath"              "Version"
#  [4] "Priority"             "Depends"              "Imports"
#  [7] "LinkingTo"            "Suggests"             "Enhances"
# [10] "License"              "License_is_FOSS"      "License_restricts_use"
# [13] "OS_type"              "MD5sum"               "NeedsCompilation"
# [16] "Built"

length(df_pkg[[1]])
# [1] 452

df_pkg[c(1:5,448:452),c(1,3,10,16)]
#                    Package  Version                    License Built
# boot                  boot 1.3-28.1                  Unlimited 4.2.0
# abind                abind    1.4-5                LGPL (>= 2) 4.2.0
# ADGofTest        ADGofTest      0.3                        GPL 4.2.0
# airports          airports    0.1.0                      GPL-3 4.2.0
# arrayhelpers arrayhelpers    1.1-0                        GPL 4.2.0
# xts                   xts   0.12.2                GPL (>= 2) 4.2.0
# yaml                 yaml    2.3.7 BSD_3_clause + file LICENSE 4.2.0
# zeallot           zeallot    0.1.0          MIT + file LICENSE 4.2.0
# zip                   zip    2.2.2          MIT + file LICENSE 4.2.0
# zoo                   zoo   1.8-11               GPL-2 | GPL-3 4.2.0
```

R packages - Updating

- all packages are regularly updated in the official repositories
- the `old.packages()` function allows to inspect all the installed packages and compare the version with the latest available on the packages repositories

```
old.packages()
# Package
# boot          "boot"
# class         "class"
# codetools     "codetools"
# conquer       "conquer"
...
#              LibPath
# boot          "/Library/Frameworks/.../4.2/Resources/library"
# class         "/Library/Frameworks/.../4.2/Resources/library"
# codetools     "/Library/Frameworks/.../4.2/Resources/library"
# conquer       "/Library/Frameworks/.../4.2/Resources/library"
...
#              Installed     Built    ReposVer
# boot          "1.3-28"      "4.2.2" "1.3-28.1"
# class         "7.3-20"      "4.2.2" "7.3-21"
# codetools     "0.2-18"      "4.2.2" "0.2-19"
# conquer       "1.3.1"       "4.2.0" "1.3.2"
...
#               Repository
# boot          "https://cran.stat.unipd.it/src/contrib"
# class         "https://cran.stat.unipd.it/src/contrib"
# codetools     "https://cran.stat.unipd.it/src/contrib"
# conquer       "https://cran.stat.unipd.it/src/contrib"
```

- to update a single packages use the `install.packages()` function again

```
install.packages("boot")
```


R packages - Updating All

- to update all the packages:

```
update.packages(ask = FALSE)
#--- Please select a CRAN mirror for use in this session ---
# Secure CRAN mirrors
# 1: 0-Cloud [https]
...
# 49: Italy (Padua) [https]
...
# Selection: 49

# also installing the dependencies 'httr2', 'conflicted'

#   There are binary versions available but the source versions are later:
#              binary source needs_compilation
# haven         2.5.1  2.5.2              TRUE
# packrat       0.9.0  0.9.1             FALSE
# RcppParallel  5.1.6  5.1.7              TRUE

# Do you want to install from sources the packages which need compilation? (Yes/no/cancel)
# trying URL 'https://cran.stat.unipd.it/bin/macosx/contrib/4.2/httr2_0.2.2.tgz'
# Content type 'application/octet-stream' length 376448 bytes (367 KB)
# ==================================================
# downloaded 367 KB
...
...
The downloaded source packages are in
        '/private/tmp/RtmprxXReZ/downloaded_packages'

old.packages()
# NULL
```

R packages - Remove

- to delete a package from the R library:

```
remove.packages("boot")
# Removing package from '/Library/.../4.2/Resources/library'
#        (as 'lib' is unspecified)
```

- and to reinstall the package:

```
install.packages("boot")
# trying URL 'https://cran.stat.unipd.it/bin/.../4.2/boot_1.3-28.1.tgz'
# Content type 'application/octet-stream' length 629093 bytes (614 KB)
# ==================================================
# downloaded 614 KB

# The downloaded binary packages are in
#       /tmp/RtmprxXReZ/downloaded_packages
```


R session housekeeping - 1

- to list all the objects created with the current session, use the `ls()` or `objects()` functions

```
objects()
#  [1] "Rdate"    "XXL"      "ctl"      "data"     "dc"       "diffs"
#  [7] "dl"       "duration" "group"    "hh"       "lm.D9"    "lm.D90"
# [13] "model"    "ncol"     "op"       "opar"     "r"        "res"
# [19] "st"       "t1"       "t2"       "test1"    "tf"       "times"
# [25] "trt"      "weight"   "x"        "xx"       "y"        "y1"
# [31] "y2"
```

- to list all the packages and data frames currently attached to the running R session, use `search()`

```
search()
# [1] ".GlobalEnv"        "package:lattice"   "times"
# [4] "data"              "package:stats"     "package:graphics"
# [7] "package:grDevices" "package:utils"     "package:datasets"
#[10] "package:methods"   "Autoloads"         "package:base"
```

R session housekeeping - 2

- To show the structure of an object, in a compact way, the `str()` function can be used:

```
str(fname)
#  chr "Rpackage_list_2023Jan.cvs"

str(df_pkg)
#'data.frame':  452 obs. of  16 variables:
# $ Package              : chr  "boot" "abind" "ADGofTest" "airports" ...
# $ LibPath              : chr  "/Library/Frameworks/.../4.2/Resources/library" ...
# $ Version              : chr  "1.3-28.1" "1.4-5" "0.3" "0.1.0" ...
# $ Priority             : chr  "recommended" NA NA NA ...
# $ Depends              : chr  "R (>= 3.0.0), graphics, stats" "R (>= 1.5.0)" NA "R (>= 2.10)" ...
# $ Imports              : chr  NA "methods, utils" NA NA ...
# $ LinkingTo            : chr  NA NA NA NA ...
# $ Suggests             : chr  "MASS, survival" NA NA "testthat" ...
# $ Enhances             : chr  NA NA NA NA ...
# $ License              : chr  "Unlimited" "LGPL (>= 2)" "GPL" "GPL-3" ...
# $ License_is_FOSS      : chr  NA NA NA NA ...
# $ License_restricts_use: chr  NA NA NA NA ...
# $ OS_type              : chr  NA NA NA NA ...
# $ MD5sum               : chr  NA NA NA NA ...
# $ NeedsCompilation     : chr  "no" "no" NA "no" ...
# $ Built                : chr  "4.2.0" "4.2.0" "4.2.0" "4.2.0" ...
```

- `str()` can be applied to all R objects, including functions

```
str(ls)
# function (name, pos = -1L, envir = as.environment(pos),
#           all.names = FALSE, pattern, sorted = TRUE)

str(mean)
# function (x, ...)
```


R as a calculator

- the screen prompt > invites to type commands and data
- the command line can be used as a calculator

```
log(34/5.5)
# [1] 1.821612
```

- each line can have up to 8192 characters, but can be continued on further lines if incomplete (the prompt will change from > to +)

```
log(34.7) + sqrt(12) -
 + 25 / 7 * 46^3
# [1] -347621.6
```

- two or more expressions can be placed on the same line, if are separated by ';'

```
log(10); sqrt(3.75)*4.7; 2^2
# [1] 2.302585
# [1] 9.101511
# [1] 4
```

R knows complex numbers

- complex numbers arithmetic's and elementary trigonometric, logarithmic, exponential, square root and hyperbolic functions are implemented
- a complex number has the imaginary part identified by a lower-case 'i'

```
3.5 +2i
# [1] 3.5+2i
```

- special R functions can be used with complex numbers :

```
Re(3.5 + 2i)
# [1] 3.5
Im(3.5 + 2i)
# [1] 2
Mod(3.5 + 2i)
# [1] 4.031129
Arg(3.5 + 2i)
# [1] 0.5191461
Conj(3.5 + 2i)
# [1] 3.5-2i
is.complex(3.5 + 2i)
# [1] TRUE
as.complex(3.5)
# [1] 3.5+0i
```

Function	Description
`Re(z)`	Extract the real part
`Im(z)`	Extract the imaginary part
`Mod(z)`	Calculate the modulus
`Arg(z)`	Calculate the argument : `Arg(x+yi) = atan(y/x)`
`Conj(z)`	Work out the complex conjugate
`is.complex(z)`	test for complex number membership
`as.complex(z)`	force the input as a complex number

R mathematical functions

Function	Description
`log(x)`	base e log of `x`
`exp(x)`	anti-log of `x`
`log(x,n)`	base `n` log of `x`
`log10(x)`	base 10 log of `x`
`sqrt(x)`	square root of `x`
`factorial(x)`	$x! = x(x-1)(x-2)\ldots 3\cdot 2\cdot 1$
`choose(n,x)`	binomial coefficient, $n!/(x!\cdot(n-x)!)$
`gamma(x)`	$\Gamma(x)$ for real `x`, $(x-1)!$ for integer `x`
`lgamma(x)`	natural log of $\Gamma(x)$
`abs(x)`	absolute value for `x`
`floor(x)`	greater integer less than `x`
`ceiling(x)`	smallest integer greater than `x`
`trunc(x)`	closest integer to `x` between 0 and `x`; it behaves as `floor()` for $x > 0$ and like `ceiling()` for $x < 0$

```
floor(1.6); floor(-1.6)
# [1] 1
# [1] -2
ceiling(1.6); ceiling(-1.6)
# [1] 2
# [1] -1
trunc(1.6); trunc(-1.6)
# [1] 1
# [1] -1
```

R trigonometric functions

Function	Description
`cos(x)`	cosine of `x` in radians
`sin(x)`	sine of `x` in radians
`tan(x)`	tangent of `x` in radians
`asin(x)`, `acos(x)`, `atan(x)`	inverse trigonometric functions for real or complex numbers
`asinh(x)`, `acosh(x)`, `atanh(x)`	inverse hyperbolic trigonometric functions for real or complex numbers

- all trigonometric functions measure angle in radians. R knows the value of π as `pi`

```
pi
# [1] 3.141593

sin(pi/2)
# [1] 1

cos(pi/2)
# [1] 6.123234e-17
```


R variable names and assignments

- variable names are case sensitive : `y` different from `Y`
- variable names must not begin with numbers (`4t`) or symbols (`%8`)
- variable names must not contain blank spaces (use `m.value` instead of `m value`)
- object assignment is achieved using the `'<-'`, **left arrow** operator. Do not put spaces between them or a logical test will be performed (see below)

```
x <- 5
x
# [1] 5
x < - 5
# [1] FALSE
```

- assignment can be achieved also with the `'->'`, or `'='` operators

```
sqrt(x) + x^3 -> y
y
# [1] 127.2361
z = x/y
z
# [1] 0.03929703
```

R arithmetic operators summary

`+ - * /`	sum, subtraction, multiplication, division
`%/% %% ^`	integer quotient, modulo, power
`> >= < <= == !=`	relational operators
`! & |`	logical not, and, or
`~`	model formulae ('is modelled as a function of')
`<- ->`	assignment (gets)
`$`	list indexing (the 'element name' operator)
`:`	sequence creation operator

```
119 %/% 12  # integer part of the division
# [1] 9

119 %% 12   # reminder (modulo) of the division
# [1] 11

15421 %% 7 == 0
# [1] TRUE
```

- several of these operators have different meaning inside model formulae :
  `*` indicates the main effects plus interaction (rather than multiplication),
  `:` the interaction between two variables (rather than generate a sequence), and
  `^` interactions up to the indicated power (rather than raise to the power)