

# A Gentle Introduction to R

## EXTRAS

Jonathan Whiteley

2023-08-09

1 Get 

2 About 

3 Using 

4 Names

5 Special Operators

# Section 1



# Check your Operating System

On **Linux**, R is often available in your *package management system*, and can be installed directly from there.

On **macOS**, R can be installed directly with one of:

- Homebrew (formula `r`)
- MacPorts

Otherwise, you can download and install R from CRAN (next→)

- 1 Go to [www.r-project.org](http://www.r-project.org) in your web browser
- 2 Click on 'CRAN' in the menu on left (under "**Download**")
  - ▶ CRAN = Comprehensive R Archive Network
- 3 Choose a "mirror" (server) that is close to where you are, or at an institution that you trust.  
If you're not sure, you can use one of:
  - ▶ [cloud.r-project.org](http://cloud.r-project.org)
  - ▶ [cran.r-project.org](http://cran.r-project.org)
- 4 Click on the link for your computer Operating System: Linux, macOS, or Windows (in the top section labelled "**Download and Install R**")
- 5 The next steps depend on your computer's Operating System
  - ▶ follow the instructions on the subsequent web pages

## Section 2

About 

- It's **fast & lean**
  - ▶ Load only the components you need at the time
- It works on **multiple platforms**
- It has sophisticated **graphics** capabilities
  - ▶ Produce publication-quality graphs in the same software as your analysis. No need for post-processing in Illustrator or Photoshop.
- It's **flexible, powerful, and innovative**
  - ▶ Advanced methods are often available in R before other statistical software.
- It can be used interactively, or to run pre-written *scripts*
  - ▶ Scripts provide automatic record of how an analysis was performed, that can be re-produced even years later.

# is Free/Libre Open Source Software (FLOSS)

- Free as in ‘beer’: you do not have to pay \$ for it.
- Free as in ‘speech’: You have the *freedom* to ...
  - ▶ use it for any purpose
  - ▶ study how it works and adapt it to your needs
  - ▶ redistribute copies to your friends & neighbours
  - ▶ improve it and release improvements publicly
- The source code is *open* (“open source”<sup>12</sup>)
  - ▶ the source code is publicly available
  - ▶ the license allows for anyone to copy, modify, or distribute the code.
  - ▶ open collaboration is encouraged
  - ▶ anyone can propose changes and improvements, but a *Core Team* controls what changes are integrated into the versions released & distributed by the R Project

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Open\\_source](https://en.wikipedia.org/wiki/Open_source)

<sup>2</sup><https://opensource.org/definition-annotated/>



## Section 3

Using 

# R is a programming language

- R is command-driven
  - ▶ Not “point and click”
  - ▶ No menus, pop-up windows, or wizards
- R will not tell you what to do, or guide you through the steps of an analysis or method. R provides no structure.
- R will do all the calculations for you, and it will do *exactly what you tell it* (not necessarily *what you want*).
- This means R has the flexibility and power to do *exactly what you want, exactly how you want it done*.
- **The hard part is figuring out *how to do what you want***

# Learning R

Learning any programming language is a journey. source: R-ladies Sydney, Real Python

It always feels like there's more to learn.

R is designed so that users can start by using it *interactively* (as in this workshop), and then gradually use it for more programming as their needs and skills grow.

## Section 4

### Names

# Symbolic Variables

- You can store values (*objects*) in symbolic variables (*names*) using an *assignment operator*

---

|    |   |
|----|---|
| -> | assign the <i>value</i> on the <b>left</b> to the <i>name</i> on the <b>right</b> |
| <- | assign the <i>value</i> on the <b>right</b> to the <i>name</i> on the <b>left</b> |
| =  | assign the <i>value</i> on the <b>right</b> to the <i>name</i> on the <b>left</b> |

---

- '<-' is preferred, because it is unambiguous (to people *and* to R)
- '=' is not allowed in certain situations (e.g., when surrounded by other expressions)
  - '=' is also used to set *argument values* in *function calls*, which is a different meaning and its most common use.
- You can also use the *assign function* (advanced):

```
assign('x', 3)      # assign the value 3 to the variable 'x'
```

# Variable / Object Names

- In R, all variables are *objects*
  - ▶ In R, **everything** is an *object*
- Object names can include:  
(depending on the language or *locale*)

---

|             |         |
|-------------|---------|
| letters     | a-z A-Z |
| numbers     | 0-9     |
| periods     | .       |
| underscores | _       |

---

- Names *should begin with a letter*

```
A <- 10
B = 10 * 10
log(A) -> A_log
B.seq <- 1:B
assign('x', 3)
```

# Object Names: Details

Names can start with a **letter** or a **period** (*more on this later*)

```
myvar <- T  
.myvar <- T
```

but anything else triggers an **error**

```
Omyvar <- F  
_myvar <- F  
my var <- F
```

For more information about object names in R, see:

- Section 1.8 of 'An Introduction to R'
- Section 2.1.3 of 'The R Language Definition'

# Object Names: Hidden

- Names starting with a period (.) are special and normally hidden from users.

```
ls()  
ls(all.names = TRUE)
```

- Names starting with a period are used by packages or the system for special objects that users should not interact with directly.
- Such objects may not behave as expected with common commands, such as `ls()` (above).
- Therefore, most users should avoid doing this unless they know what they are doing and have a good reason to do so.



# Object Names: Advanced

- 'Valid' names following the rules above can be referred to easily in code.
- Names with any character are actually possible, but must be quoted with backticks (`)`)
  - ▶ **This is not recommended practice**, but occasionally useful when you need to refer to an element of an object, such as list items or data frame columns, that have non-standard names.

```
`(my) [strange] {variable} 'name' "!@# $"` <- T  
print(`(my) [strange] {variable} 'name' "!@# $"`)
```

```
## [1] TRUE
```

## Section 5

### Special Operators

# Matrix math

- R can do *matrix math* — which is used in many statistical procedures
  - ▶ But the *syntax* is different from the usual math operators
- Using a regular multiplication symbol (\*) results in *element*-wise multiplication
  - ▶ each *element* (item) in matrix1 is multiplied by the corresponding *element* in matrix2, etc.

```
c(1, 2, 3) * c(3, 2, 1)
```

- *Matrix multiplication* is specified by this operator: `%%`

```
c(1, 2, 3) %% c(3, 2, 1)
```