INTRODUCTION TO ‘R’

SHORT CHECK

How would you define your proficiency level using the statistical software ‘R’?

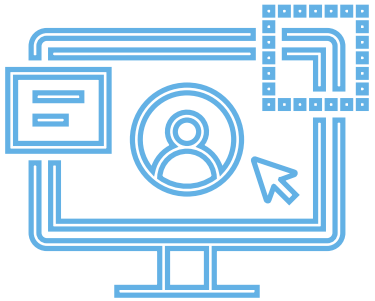
A. R… what’s that?
B. Beginner
C. Intermediate
D. R is my jam
E. I don’t use R but I am proficient in another script-based language such as Python

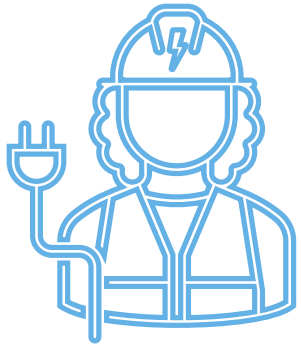
PREREQUISITE

- R & R-studio installed on your computer
  - https://learnr-examples.shinyapps.io/ex-setup-r/
  - https://www.datacamp.com/tutorial/installing-R-windows-mac-ubuntu

DISCLAIMER

I am not a software engineer!





LEARNING OBJECTIVES

- After this lecture, you should be able to:
  - Understand and apply basic R functionalities
  - Adhere to good software development practices when setting up a new R project

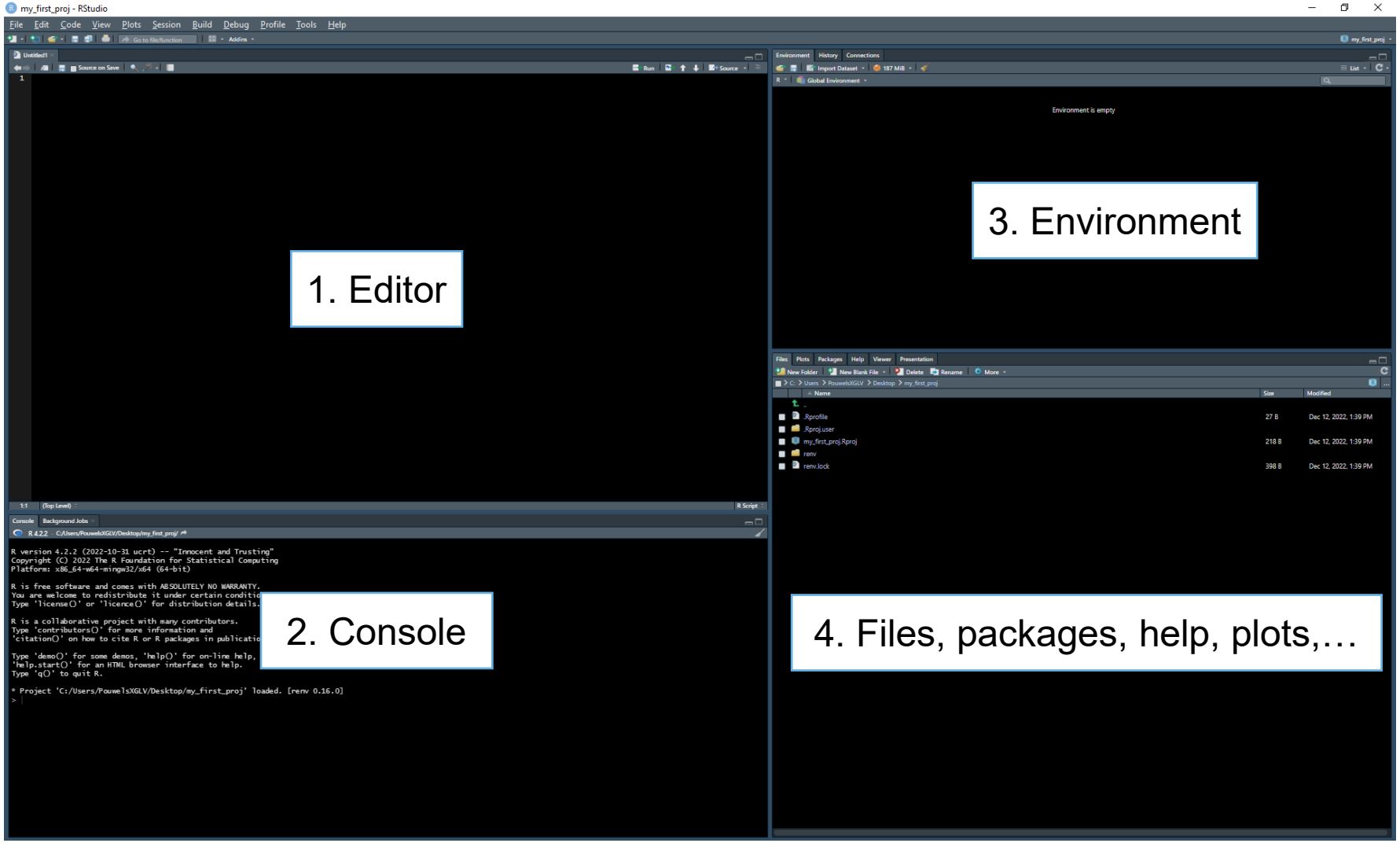
WHY DOES IT MATTER?

- It provides structure to your code
  - Easy navigation in your project
  - Easy automation of code execution
  - Easier debugging of your code
- It makes your code more readable & transparent
  - For your (future) self and others
  - Easier to share and collaborate

THE BEAUTY OF ‘R’

Source: https://www.move-lab.space/projects/roads-to-rome

RSTUDIO


my_first_proj - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
1. Editor
3. Environment
Environment is empty
2. Console
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
Type 'demo()' for some demos, 'help()' for on-line help,
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
* Project 'C:/Users/PouwelsXGLV/Desktop/my_first_proj' loaded. [renv 0.16.0]
4. Files, packages, help, plots,…

ARCHITECTURE OF R

- R packages
- Project
  - Working directory
- Scripts
  - Functions
  - Objects
    - Single object
    - Vector
    - Matrix
    - Dataframe
    - List
    - …

R PACKAGES

- Contains
  - Functions
  - Data
  - Help files
- Most are hosted on The Comprehensive R Archive Network (CRAN)
  - 22.12.12: 18,916 packages on CRAN
- Github: packages in development

R PACKAGES – INSTALLATION AND LOADING

- Use the `install.packages()` function to install the desired package.
  - NOTE: always put the package name within quotation marks.
- Use `library()` to load installed packages.

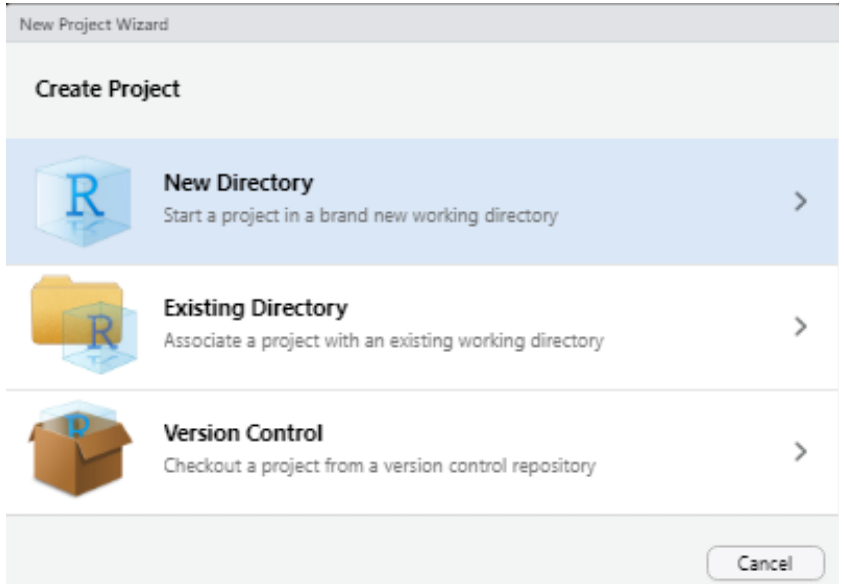
```
# install simmer
install.packages("simmer")

# load simmer
library(simmer)
```

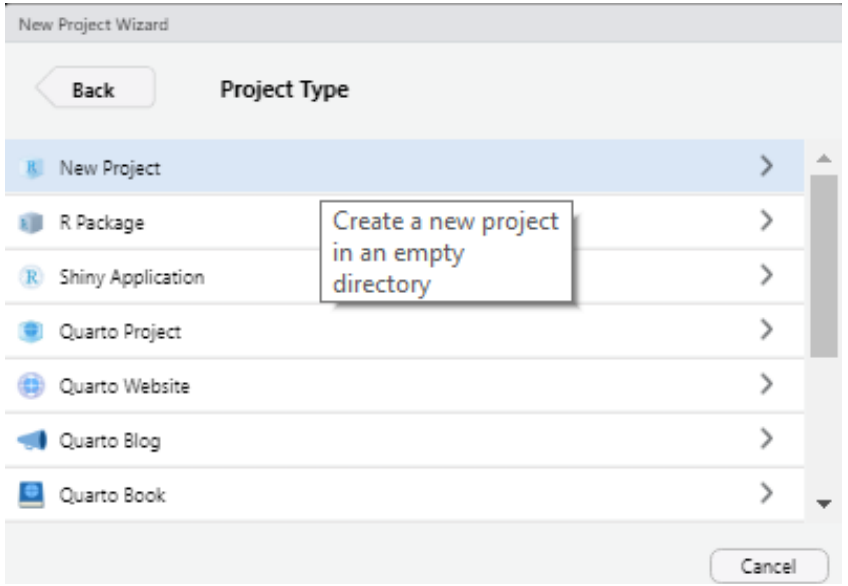
R PROJECT

- Container for
  - (Raw) data
  - R script
  - (User-defined) functions
  - Outputs
  - History of the project
  - …
  - Anything allowing to perform your analyses for a specific project
- Has its own working directory
  - Different analyses / projects = different directories!
  - Allows to work on different analyses without them interfering with each other!

CREATE NEW R PROJECT 1/3

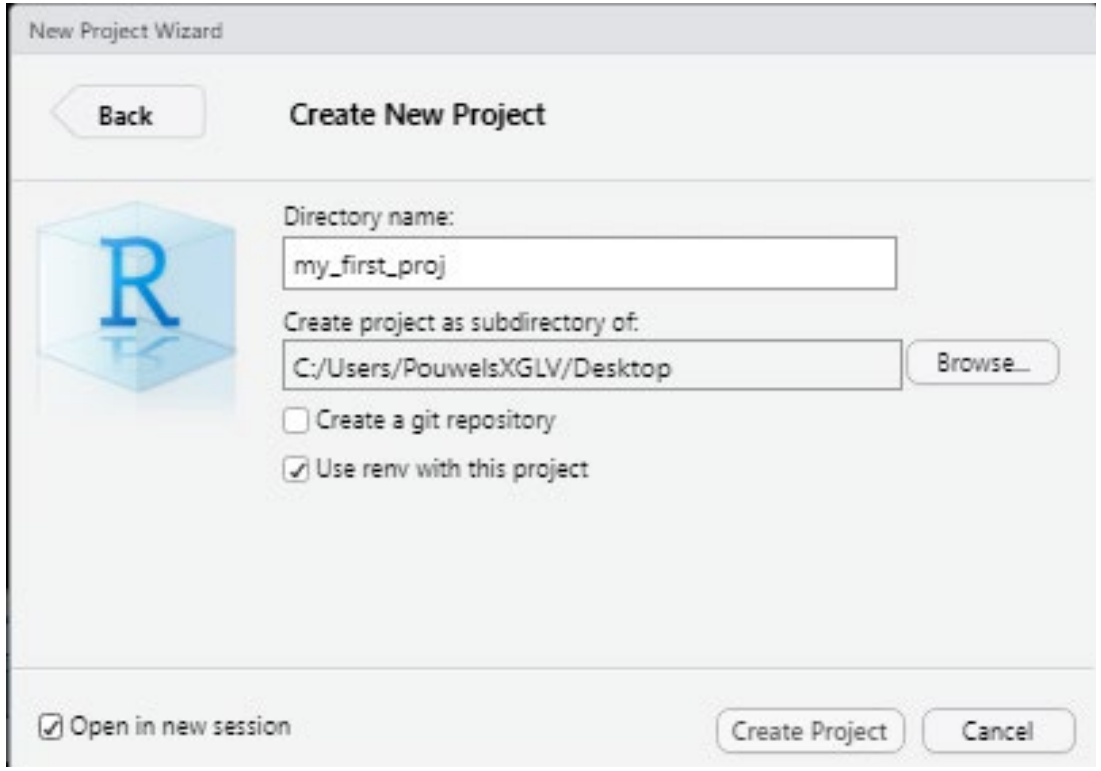
- Two ways
  - File -> New project… -> New Directory -> New Project
  - Click on ‘project’ icon and name on the top right corner -> New project… -> New Directory -> New Project





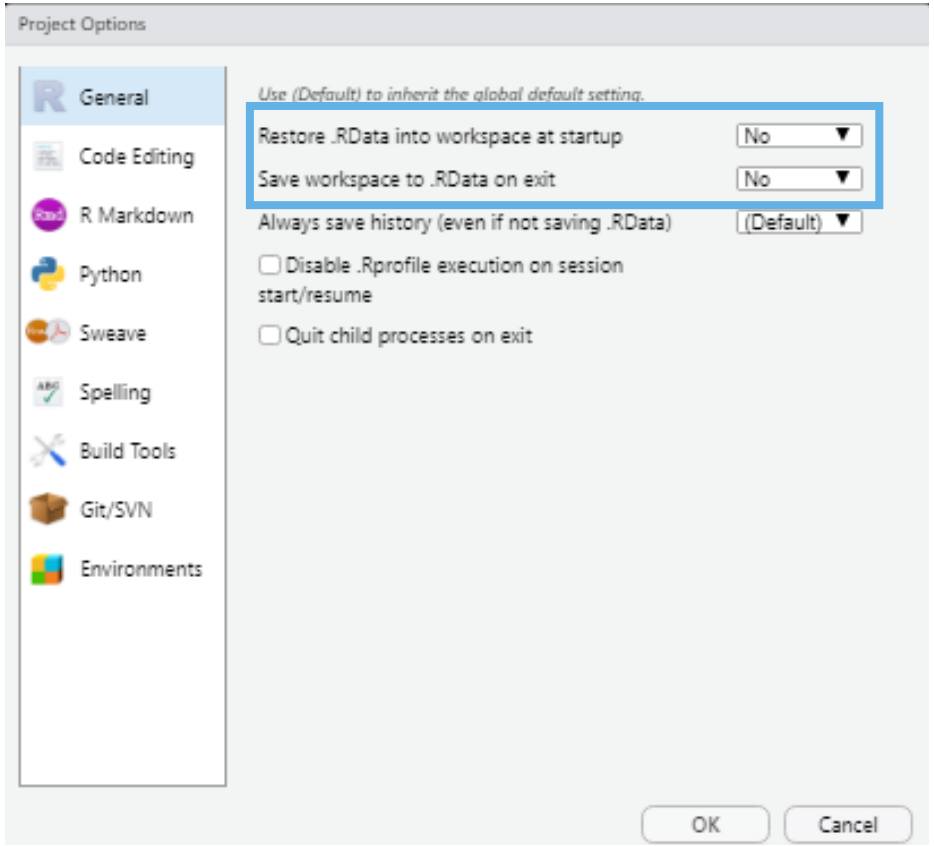
CREATE NEW R PROJECT 2/3

- Choose a location



CREATE NEW R PROJECT 3/3

- Always start with a clean sheet (Tools -> Project Options…)

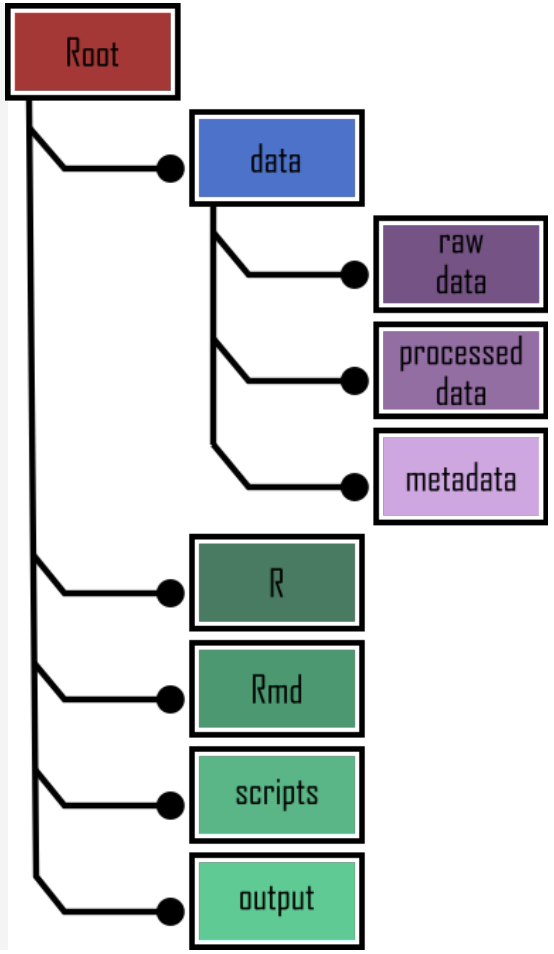


MAPS & FILES STRUCTURE - EXAMPLES

Box 3. Project layout

```
.
|-- CITATION
|-- README
|-- LICENSE
|-- requirements.txt
|-- data
|     |-- birds_count_table.csv
|-- doc
|     |-- notebook.md
|     |-- manuscript.md
|     |-- changelog.txt
|-- results
|     |-- summarized_results.csv
|-- src
|     |-- sightings_analysis.py
|     |-- runall.py
```

Wilson et al. 2017



Douglas et al. 2022

Table 1 File folder structure for organiz

Folder name	Folder function
data-raw	This is where raw data is data (<processed data parameters derived fror primary data from whic
data	This is where input data i stored in the 'data-raw' from elsewhere
R	This is where '.R' files th the analysis. The model model output to the spe the 'data' folder
analysis	This is where interactive where many operations
output	This is where output files external data files (such PSA dataset generated i having to first rerun pre running the calibration)
figs	For analyses that will inc folder, it can be helpful important for analyses t
tables	This folder includes table
report	A report folder could be data of the framework, generate a report of the model-based CEA
vignettes	A vignettes folder could work through accompa figures to integrate the
tests	A tests folder includes '.I of tests for each compli

Alarid-Escudero et al. 2019

What are the similarities???

MAPS & FILES STRUCTURE

- Separation of
  - inputs (raw data and cleaned data)
  - outputs
  - scripts
  - analysis
  - report
  - documentation
- Use a README file
  - Aim of the project
  - Which files?
  - What can you find where?
- Numerate files that needs to be run sequentially
  - Use a ‘Master’ script to run these files

CREATE A NEW MAP (FOLDER)

1. By using R command `dir.create()`

```
# create new 'output' folder
dir.create("output")

# create a 'tables' and 'figs' folder within
'output'
dir.create("output/tables")
dir.create("output/figs")
```

2. By clicking *New folder* in the lower-right panel

CREATE A NEW SCRIPT

1. Use the shortcut: Ctrl + Shift + N
2. By clicking *New Blank File* -> *R script* in the lower-right panel

R OBJECTS

1. Use `<-` to assign a value to an object
2. Use `+, - , *, /` for basic transformation

```
a <- 2 # single object
b <- a + 1
b
## [1] 3
a / b
## [1] 0.6666667
```

VECTOR

1. Use `c()` to create a vector

```
v_1 <- c(a, b) # vector
v_1 * 2 # both elements are multiplied by 2
## [1] 4 6
```

```
v_2 <- c(1:6) # : means a vector containing all
integers between 1 & 6
```

```
v_1 * v_2
## [1]  2  6  6 12 10 18
```

SOME BASIC FUNCTIONS

```
mean(v_2)
## [1] 3.5
sum(a, b, v_1)
## [1] 10
seq(from = 1, to = 2, by = 0.1)
##  [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9
2.0
rep(v_1, 3)
## [1] 2 3 2 3 2 3
```

CODE STRUCTURE & ANNOTATIONS

1. Start fresh & start with loading packages and declaring variables
2. Separate sections of code using `# headingname ----`
3. Use spaces & indent code (Ctrl + I)
   - Automatic code reformatting (Ctrl + Shift + A)
4. Annotate
   - Write why you do something, not what you do!
   - Code should *"speak for itself"*
5. Stick to a coding style

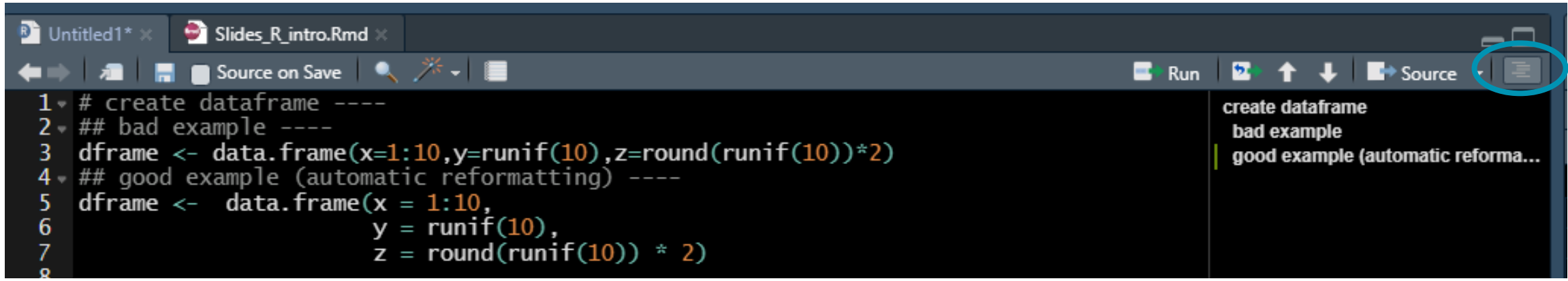
1. START FRESH

```
# bad ----
## Set up ----
dframe <- data.frame(x = 1:10,
                     y = runif(10))
set.seed(500)
library(simmer)
```

1. START FRESH

```
# good ----
rm(list = ls())
library(simmer)
set.seed(500)
dframe <- data.frame(x = 1:10,
                     y = runif(10))
```

2. SEPARATE SECTIONS



3. USE SPACES & INDENT CODE

```
# create dataframe ----
## bad example ----
dframe <-
data.frame(x=1:10,y=runif(10),z=round(runif(10))*
2)
## good example (automatic reformatting) ----
dframe <-  data.frame(x = 1:10,
                      y = runif(10),
                      z = round(runif(10)) * 2)
```

4. ANNOTATE

```r
## Bad
v_c <- c(1000, 1200, 129, 2097, 875, 982, 300,
0, 0)
v_c_pa <- rnorm(1000, mean(v_c), sd(v_c) /
sqrt(length(v_c)))

## Good
v_c <- c(1000, 1200, 129, 2097, 875, 982, 300,
0, 0) # vector of costs
v_c_pa <- rnorm(1000, mean(v_c), sd(v_c) /
sqrt(length(v_c))) # probabilistic estimates
using mean and standard error
```

5. STICK TO A CODING STYLE

- Tidyverse style guide(https://style.tidyverse.org/files.html)
- Google R coding style (https://google.github.io/styleguide/Rguide.html)
- Tilburg Science Hub (https://tilburgsciencehub.com/building-blocks/develop-your-research-skills/tips/r-code-style/)
- Decision Analysis in R for Technology in Health style
- `styler` R package
- …

Table 3 Recommended prefixes in variable names that encode data and variable type

Prefix	Data type	Prefix	Variable type
<> (no prefix)	scalar	n	Number
v	vector	p	Probability
m	matrix	r	Rate
a	array	u	Utility
df	data frame	c	Cost
dtb	data table	hr	Hazard ratio
l	list	rr	Relative risk
		ly	Life years
		q	QALYs
		se	Standard error

Source: Alarid-Escudero et al. 2019

CODING STYLE – SOME GOLDEN RULES

Try to avoid using '.' or other special characters, prefer '_'

```
# Bad
first.obj <- 2

# Good
first_obj <- 2
```

CODING STYLE – SOME GOLDEN RULES

Do not assign values or functions to common R objects (e.g. TRUE / FALSE) and functions (e.g. mean(), sum())

```
# Bad
sum <- function(x, y) {
  return(x + y)
}

# Good
sum_of_two <- function(x, y) {
  return(x + y)
}
```

CODING STYLE – SOME GOLDEN RULES

Be consistent!

```
# Bad
first_obj <- 2
SecondObj <- 45

# Good
first_obj  <- 2
second_obj <- 45
```

"There are only two hard things in Computer Science: cache invalidation and naming things."
Phil Karlton

TAKE AWAYS

What do you take away?

Write on a piece of paper / notebook what seems the most important to you when working with R (1 minute)

TAKE AWAYS

1. Work within a R project
2. Start fresh
3. Give space
4. Structure, structure, structure
   - Project folders, files, & within scripts
5. Comment
6. All that counts is your style!

ANY QUESTIONS ON R OR TUTORIAL 1?

DO IT YOURSELF!

1. Warm up exercise setting-up a R project
   - https://alexd106.github.io/BI5009/exercise_1.html

2. Take a break

3. Apply these principles when doing Tutorial 1 - Introduction to R
   - Canvas
   - Ask questions

RESOURCES

- Alarid-Escudero F, Krijkamp EM, Pechlivanoglou P, Jalal H, Kao SZ, Yang A, Enns EA. A Need for Change! A Coding Framework for Improving Transparency in Decision Modeling. Pharmacoeconomics. 2019 Nov;37(11):1329-1339. doi: 10.1007/s40273-019-00837-x. PMID: 31549359; PMCID: PMC6871515.
- Douglas et al. 2022. An introduction to R. Available at https://intro2r.com/ accessed on 15-12-2022
- Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017) Good enough practices in scientific computing. PLoS Comput Biol 13(6): e1005510. https://doi.org/10.1371/journal.pcbi.1005510