DATA MANIPULATION

LECTURE 1 INTRO TO R PROGRAMMING

MARIA MONTOYA-AGUIRRE

M1 APE @ PARIS SCHOOL OF ECONOMICS

WELCOME TO THE COURSE!

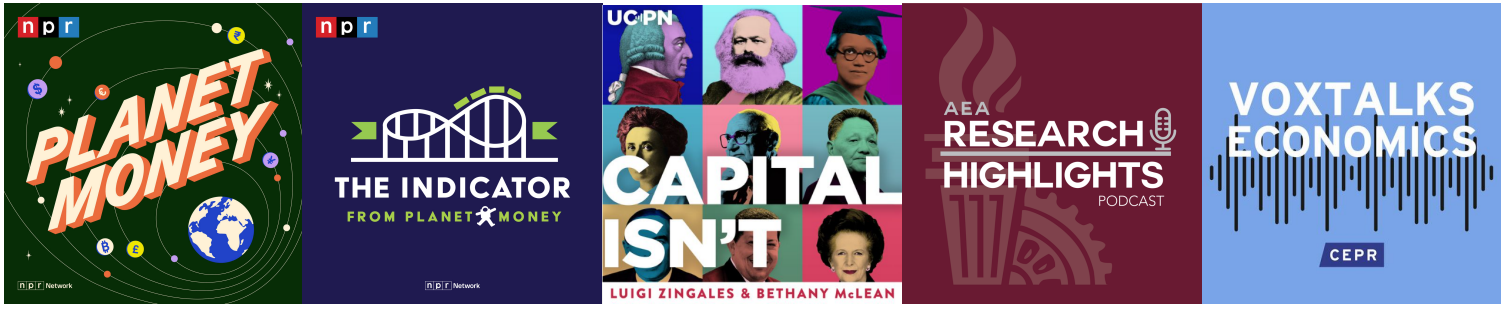
ABOUT ME:

- PhD candidate at Paris School of Economics. Started APE in 2019!
- Working on (labor ⋂ crime ⋂ development) economics
- I've used R for +5 years now.

SOME ADVICE FOR M1:

- Work consistently
- Take care of yourself
- Stay connected to the research
- Look for the questions/problems you care about

Econ podcasts I like:



ABOUT YOU

WHAT IS YOUR NAME?

WHERE ARE YOU FROM?

ONE OF YOUR FAVORITE PROBLEMS

ABOUT THE COURSE

MAIN OBJECTIVE

Learning R programming to carry out your empirical homeworks and research projects.

STRUCTURE

4 sessions (2h each) designed to introduce you to the main steps in the data analysis (~ empirical research) pipeline:

1. Data collection
2. Data cleaning and tidying **(Today!)**
3. Data analysis
    - Data visualization **(Lecture 2)**
    - Econometrics **(Lecture 4)**
4. Report results & publish code **(Lecture 3)**

One-break for you to follow the first lecture/tutorial of Econometrics I

COURSE WEBSITE

https://github.com/mariamontoyaa/2023-intro-to-R-public

🚨 Read the class policies and instructions for asking for help.

AGENDA

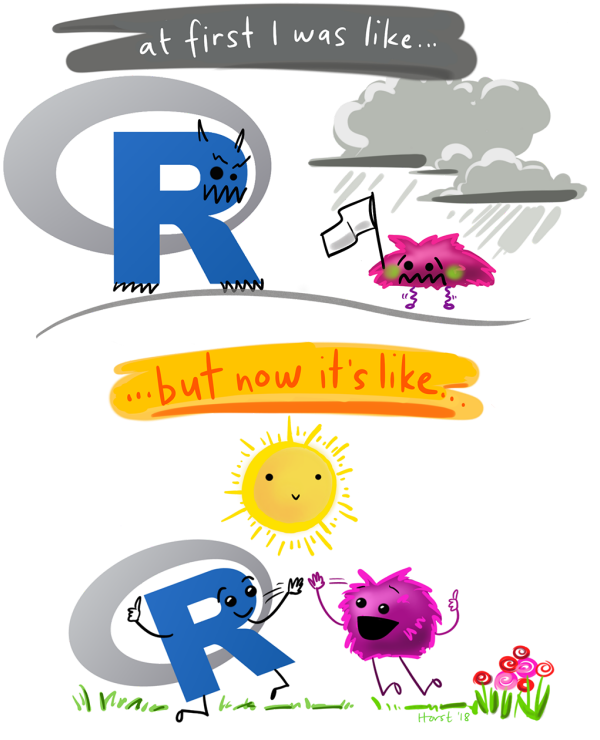
- GETTING STARTED WITH R
  - THE R STUDIO IDE
  - OBJECTS AND FUNCTIONS
  - IMPORTING DATA
- ANATOMY OF A DATA FRAME
  - SCALARS
  - VECTORS
  - DATAFRAMES
  - SUBSETTING
- MANIPULATING DATA WITH {dplyr}
  - PACKAGES
  - MAIN FUNCTIONS
  - `group_by()` vs. `summarise()`
- THE DATA ANALYSIS PIPELINE
  - A CHECKLIST FOR CLEANING DATA
  - LOOKING FOR HELP

GETTING STARTED WITH R

Programming language and software environment for statistical computing and graphics. Widely used in Economics, Statistics and Biostatistics.

WHY SHOULD WE LEARN TO CODE?

Transparent, shareable and reproducible analysis



GETTING STARTED WITH R

Programming language and software environment for statistical computing and graphics. Widely used in Economics, Statistics and Biostatistics.

WHY SHOULD WE LEARN TO CODE?

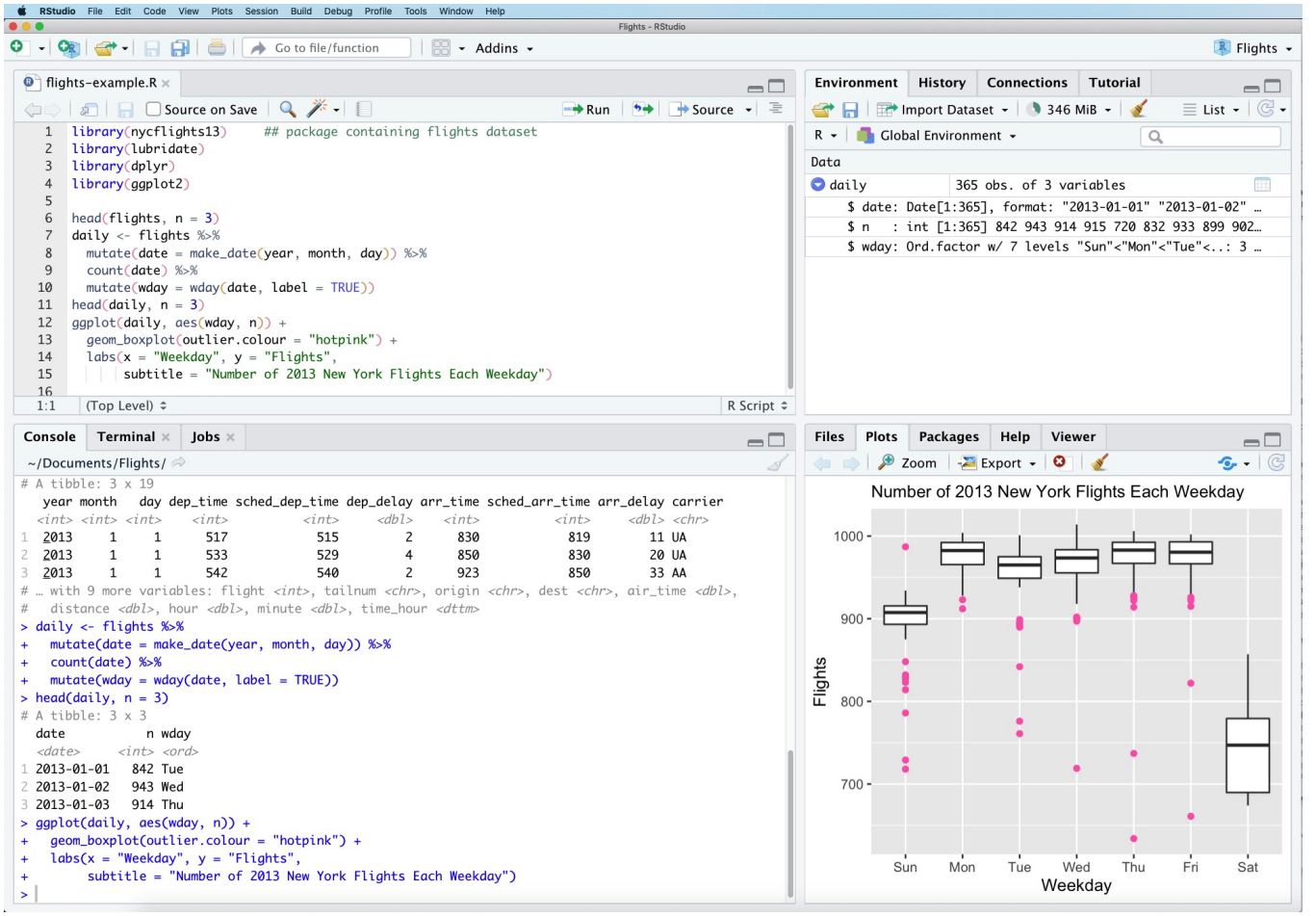
Transparent, shareable and reproducible analysis

WHY CHOOSE R?

- Free and open source. Development of features and fixes is fast
- Less specialized
- Easy to Google
- Pretty graphs (and accidental art)
- Super versatile: pretty graphs, GIS integration for maps, web scrapping, machine learning, markdown, dashboards

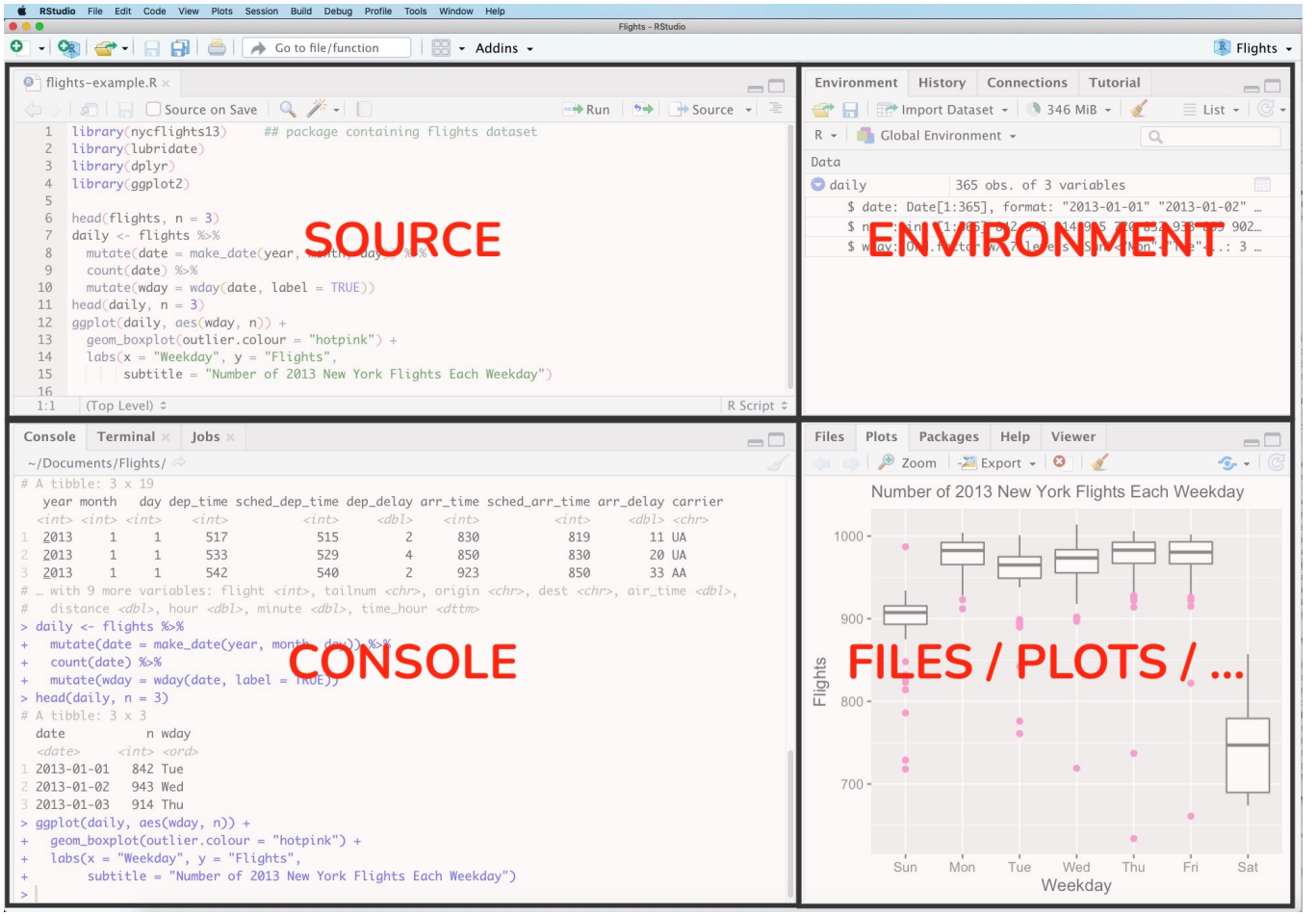
THE R STUDIO IDE

An **Integrated Development Environment** provides the comprehensive facilities for programmers in an application



THE R STUDIO IDE

An **Integrated Development Environment** provides the comprehensive facilities for programmers in an application



SOURCE

You'll spend most of you time here! Where you write and save code (File > New File > R Script) 📌 **First thing:** Save your script!

- An `#` allows to comment your code. Everything after it is ignored until a line break
- Separate different commands with a **line break** for clarity. But, note that a space doesn't break a command.
- Run and source buttons send code to the console to be executed
  - Run (from selection). Highlight the code → `Ctrl + Enter`. If no code is selected it will run the line where the cursor is.
  - Source (run everything)

```
1 + 1 # Equals two?

1 +   # Also equals two?
  1   # R recognizes that you are not done yet!

1
+ 1   # Equals one.
```

SOURCE

You'll spend most of you time here! Where you write and save code (File > New File > R Script) 📌 **First thing:** Save your script!

- An `#` allows to comment your code. Everything after it is ignored until a line break
- Separate different commands with a **line break** for clarity. But, note that a space doesn't break a command.
- Run and source buttons send code to the console to be executed
  - Run (from selection). Highlight the code → `Ctrl + Enter`. If no code is selected it will run the line where the cursor is.
  - Source (run everything)

```
1 + 1 # Equals two?
```

```
## [1] 2
```

```
1 +   # Also equals two?
  1   # R recognizes that you are not done yet!
```

```
## [1] 2
```

```
1
```

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

```
+ 1   # Equals one.
```

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

CONSOLE

The heart of the operation 

Here is where you communicate with R to get it to do things.

- You write after the `>` prompt. It means that R is ready for a new instruction. Press enter and R will execute the code.
- You can write there directly for quick calculations or fleeting requests you don't need to save.

WRITE MOST OF YOUR CODE IN THE SOURCE PANEL

ENVIRONMENT

Where we keep track of things 🖇️

- Data analysis requires manipulating different *objects*: datasets, vectors, functions, etc. Here is where we keep track of them.
- R is an **object-oriented** language. We'll talk about this in a bit. What it means is that we can see everything we are working with and things are only stored when we ask to.
- We **assign** a value to an object with `<-` the assignment operator

See what happens in the environment panel when we run this:

```
5 + 1
x <- 5    # Assigning the value
x
x + 1
x
x <- x + 1 # Assigning a new value
x
```

ENVIRONMENT

Where we keep track of things 🖇️

- Data analysis requires manipulating different *objects*: datasets, vectors, functions, etc. Here is where we keep track of them.
- R is an **object-oriented** language. We'll talk about this in a bit. What it means is that we can see everything we are working with and things are only stored when we ask to.
- We **assign** a value to an object with `<-` the assignment operator

See what happens in the environment panel when we run this:

```
5 + 1
```

```
## [1] 6
```

```
x <- 5    # Assigning the value
x
```

```
## [1] 5
```

```
x + 1
```

```
## [1] 6
```

```
x
```

```
## [1] 5
```

```
x <- x + 1 # Assigning a new value
x
```

```
## [1] 6
```

FILES/ PLOTS/ ... PANEL

Many helpful things here 🗄️

- Files shows your *working directory*, the default location of any files you read into or save out of R `getwd()`
  - We'll see more about how to work with directories and R Projects in **Lecture 3**
- Plots / Viewer
- Packages
- Help
  - Add a `?` in front of any function to get help: `?getwd()`

OBJECTS AND FUNCTIONS

R revolves around 2 things: **objects and functions** (things and procedures?)

- **Objects** are vectors, dataframes, regression models. They have attributes.
- **Functions** are procedures that typically take one or more objects as arguments (inputs), do something with them and return a new object

```
# This is not R code
window <- "This is a window"
door <- "This is a door"
one_hundred <- 100
two_thousand <- 2000

# This is not R code
open(window)
open(door)
open(book)

clean(toilet)
clean(data)
clean(conscience)
```

Programming in R is defining objects and applying functions to those objects ad infinitum

```
# This is R code
grades <- c(15, 10, 12, 13, 18)
mean(grades)
```

```
## [1] 13.6
```

IMPORTING AND INSPECTING DATA

📥 **Download the data folder from the course website**

- We are going to analyze the 250 top rated French Movies in IMDB
- Open the CSV file in your computer and inspect it: columns are variables and rows are observations

IMPORTING AND INSPECTING DATA

📥 **Download the data folder from the course website**

- We are going to analyze the 250 top rated French Movies in IMDB
- Open the CSV file in your computer and inspect it: columns are variables and rows are observations

How do we get this data into R?

- We import it using *read* functions, which take a file path as an input and give the file content as an output.

🚨 Use forward slashes `"mycomputer/folder/file.csv"` instead of backward slashes `"mycomputer\folder\file.csv"`

```
imdb <- read.csv("C:/User/Documents/01_imdb-top250-french.csv")

# Exploring the data
head(imdb, 4)   # Show first four rows
tail(imdb, 3)   # Show last three rows
View(imdb)      # Show data in a spreadsheet view
```

DID WE DO IT RIGHT?

	Name	Rank	Year	Type	Duration	Genre	Rating	MetaScore	Desc
1	Shoah	1	–1985	PG	566 min	Documentary, History, War	8.7	99	Claude Lanzmann's epic documentary recounts
2	Home	2	(I) (2009)	U	118 min	Documentary	8.5	47	With aerial footage from fifty–four countries, 'H
3	Untouchable	3	–2011	15	112 min	Biography, Comedy, Drama	8.5	57	After he becomes a quadriplegic from a paragli
4	Le Trou	4	–1960	A	131 min	Crime, Drama, Thriller	8.5	*NA*	Distrust and uncertainty arise when four long–t
5	The Man Who Planted Trees	5	–1987		30 min	Animation, Short, Drama	8.5	*NA*	The story of a shepherd's single handed quest
6	Children of Paradise	6	–1945	A	189 min	Drama, Romance	8.3	96	The theatrical life of a beautiful courtesan in 18
7	AmÃ©lie	7	–2001	15	122 min	Comedy, Romance	8.3	69	Despite being caught in her imaginative world,
8	Incendies	8	–2010	15	131 min	Drama, Mystery, War	8.3	80	Twins journey to the Middle East to discover th
9	La JetÃ©e	9	–1962	A	28 min	Short, Drama, Romance	8.2	*NA*	The story of a man forced to explore his memo
10	A Man Escaped	10	–1956	U	101 min	Drama, Thriller, War	8.2	*NA*	A captured French Resistance fighter during WW
11	A Trip to the Moon	11	–1902	U	13 min	Short, Action, Adventure	8.2	*NA*	A group of astronomers go on an expedition to
12	Z	12	–1969	A	127 min	Crime, Drama, Thriller	8.2	86	The public murder of a prominent politician an
13	Army of Shadows	13	–1969	AA	145 min	Drama, War	8.1	99	An account of underground resistance fighters
14	La haine	14	–1995	15	98 min	Crime, Drama	8.1	*NA*	24 hours in the lives of three young men in the
15	The 400 Blows	15	–1959	A	99 min	Crime, Drama	8.1	*NA*	A young boy, left without attention, delves into
16	Three Colours: Red	16	–1994	15	99 min	Drama, Mystery, Romance	8.1	100	A model discovers a retired judge is keen on in
17	Portrait of a Lady on Fire	17	–2019	15	122 min	Drama	8.1	95	On an isolated island in Brittany at the end of t
18	The Wages of Fear	18	–1953	A	131 min	Adventure, Drama, Thriller	8.2	85	In a decrepit South American village, four men
19	The Battle of Algiers	19	–1966	X	121 min	Drama, War	8.1	96	In the 1950s, fear and violence escalate as the

DID WE DO IT RIGHT?

	Name	Rank	Year	Type	Duration	Genre	Rating	MetaScore	Desc
1	Shoah	1	–1985	PG	566 min	Documentary, History, War	8.7	99	Claude Lanzmann's epic documentary recounts
2	Home	2	(I) (2009)	U	118 min	Documentary	8.5	47	With aerial footage from fifty–four countries, 'H
3	Untouchable	3	–2011	15	112 min	Biography, Comedy, Drama	8.5	57	After he becomes a quadriplegic from a paragli
4	Le Trou	4	–1960	A	131 min	Crime, Drama, Thriller	8.5	*NA*	Distrust and uncertainty arise when four long–t
5	Who Planted Trees	5	–1987		30 min	Animation, Short, Drama	8.5	*NA*	The story of a shepherd's single handed quest
	AmÃ©lie / adise	6	–1945	A	189 min	Drama, Romance	8.3	96	The theatrical life of a beautiful courtesan in 18
	Incendies	7	–2001	15	122 min	Comedy, Romance	8.3	69	Despite being caught in her imaginative world,
		8	–2010	15	131 min	Drama, Mystery, War	8.3	80	Twins journey to the Middle East to discover th
		9	–1962	A	28 min	Short, Drama, Romance	8.2	*NA*	The story of a man forced to explore his memo
	La JetÃ©e	10	–1956	U	101 min	Drama, Thriller, War	8.2	*NA*	A captured French Resistance fighter during WW
11	to the Moon	11	–1902	U	13 min	Short, Action, Adventure	8.2	*NA*	A group of astronomers go on an expedition to
12	Z	12	–1969	A	127 min	Crime, Drama, Thriller	8.2	86	The public murder of a prominent politician an
13	Army of Shadows	13	–1969	AA	145 min	Drama, War	8.1	99	An account of underground resistance fighters
14	La haine	14	–1995	15	98 min	Crime, Drama	8.1	*NA*	24 hours in the lives of three young men in the
15	The 400 Blows	15	–1959	A	99 min	Crime, Drama	8.1	*NA*	A young boy, left without attention, delves into
16	Three Colours: Red	16	–1994	15	99 min	Drama, Mystery, Romance	8.1	100	A model discovers a retired judge is keen on in
17	Portrait of a Lady on Fire	17	–2019	15	122 min	Drama	8.1	95	On an isolated island in Brittany at the end of t
18	The Wages of Fear	18	–1953	A	131 min	Adventure, Drama, Thriller	8.2	85	In a decrepit South American village, four men
19	The Battle of Algiers	19	–1966	X	121 min	Drama, War	8.1	96	In the 1950s, fear and violence escalate as the

USING FUNCTIONS

- Read the documentation! `?read.csv()`
  - Description
  - Usage
  - Arguments: with and without default values (you need to write these to work)
  - ⭐ Examples

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)
```

- You don’t need to write argument names if you write the arguments in the correct order

SO, WHAT HAPPENED?

A common problem when importing data is that the *character encoding* is handled incorrectly.

This is when we see weird characters like `Ã©` where letters should be.

All the characters we see, like `1` ,`#` or `M`, are produced by a sequence of bits (1-0), the encoding determines how these bits are translated into characters.

We need to make sure we are using the right *translation*!

```
imdb <- read.csv("C:/User/Documents/01_imdb-top250-french.csv",
                 encoding = "UTF-8") #UTF-8 should work for latin languages
```

WHEN THINGS DON'T GO WELL... DEBUGGING🐞

Go line by line making sure each command does what you expect it to.

Most of the time the problem is one of these:

- **R isn't ready:** The `>` should appear in the console. There is a `+` instead. You wrote something *incomplete* before and R is waiting for you to finish. Press `ESC`.
  `c(1, 3,`
- **Mispelled object or function** Letter reversals or capitalization are often the issue.
  `maen(1, 2, 3)`, `Mean(1, 2, 3)`
- **Punctuation problems** Mixing commas `,` and periods `.` or incorrect use of `"` `"`.
  `c(1,2.3)`
- **Wrong argument**
- **Missing arguments**

🌈 READ CAREFULLY 🌈

EXERCISE

I want to:

- Create a vector with 20 numbers drawn from a normal distribution with a mean of 5 and standard deviation of 2.
- Obtain its median.

Fix the following code.

🔍 Start with `?rnorm()`

```
v1 <- rnorm(2, 20)


Median(v1)
```

02:00

AGENDA

- GETTING STARTED WITH R
  - THE R STUDIO IDE
  - OBJECTS AND FUNCTIONS
  - IMPORTING DATA

- ANATOMY OF A DATA FRAME
  - SCALARS
  - VECTORS
  - DATAFRAMES
  - SUBSETTING

- MANIPULATING DATA WITH {dplyr}
  - PACKAGES
  - MAIN FUNCTIONS
  - `group_by()` vs. `summarise()`

- THE DATA ANALYSIS PIPELINE
  - A CHECKLIST FOR CLEANING DATA
  - LOOKING FOR HELP

ANATOMY OF A DATA FRAME

What makes up a dataset? The smallest unit of data:

SCALARS

ANATOMY OF A DATA FRAME

What makes up a dataset? The smallest unit of data:

SCALARS

Numeric

```
a <- 100
b <- 3 / 100
c <- (a + b) / b
```

Try: `class(a)`

- *int:* round numbers
  dbl: 2 decimals

Character

```
d <- "window"
e <- "4"
f <- "This is a room"
```

Try: `class(d)`

- Also called *strings*

Boolean

```
g <- TRUE
h <- (3 >= 4)
i <- (c == (a+b) / b)
```

Try: `class(g)`

- Produced with operators
  `== > < >= <= & | !`

ANATOMY OF A DATA FRAME

What makes up a dataset? The smallest unit of data:

SCALARS

Numeric

```
a <- 100
b <- 3 / 100
c <- (a + b) / b
```

Try: `class(a)`

- *int:* round numbers
  dbl: 2 decimals

Character

```
d <- "window"
e <- "4"
f <- "This is a room"
```

Try: `class(d)`

- Also called *strings*

Boolean

```
g <- TRUE
h <- (3 >= 4)
i <- (c == (a+b) / b)
```

Try: `class(g)`

- Produced with operators
  `== > < >= <= & | !`

What happens if we execute this?

```
a + b
a + e
a + as.numeric(e)
as.numeric(d)
as.character(a)
```

ANATOMY OF A DATA FRAME

What makes up a dataset? The smallest unit of data:

SCALARS

Numeric

```
a <- 100
b <- 3 / 100
c <- (a + b) / b
```

Try: `class(a)`

- *int:* round numbers
  dbl: 2 decimals

Character

```
d <- "window"
e <- "4"
f <- "This is a room"
```

Try: `class(d)`

- Also called *strings*

Boolean

```
g <- TRUE
h <- (3 >= 4)
i <- (c == (a+b) / b)
```

Try: `class(g)`

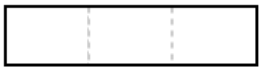
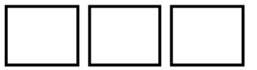
- Produced with operators
  `== > < >= <= & | !`

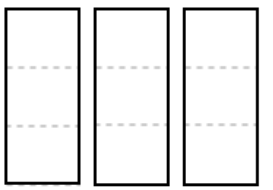
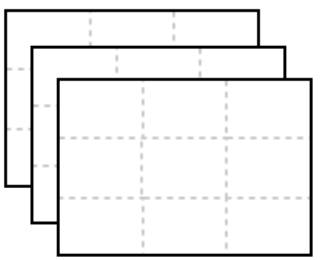
What happens if we execute this?

```
a + b
a + e              # Error in a + e : non-numeric argument to binary operator
a + as.numeric(e)
as.numeric(d)      # Character is coerced to NA (not available)
as.character(a)
```

DATA STRUCTURE

	single type	multiple types
1D	Vector	List
2D	Matrix	Data frame
nD	Array	

VECTORS

Unidimensional object that stores a sequence of values of the **same type**

```
c("Hello world", 35, FALSE)
```

```
## [1] "Hello world" "35"          "FALSE"
```

```
c("Hello world", as.character(35), as.character(FALSE))  # Equivalent to this
```

```
## [1] "Hello world" "35"          "FALSE"
```

Having the same type of values allows operations to apply to all elements in the vector

```
v1 <- c(1, 2, 3)
v1
v1 / 3
v1 * v1      # Multiply element wise
v1 %*% v1    # Matrix multiplication
```

VECTORS

Unidimensional object that stores a sequence of values of the **same type**

```
c("Hello world", 35, FALSE)
```

```
## [1] "Hello world" "35"          "FALSE"
```

```
c("Hello world", as.character(35), as.character(FALSE))  # Equivalent to this
```

```
## [1] "Hello world" "35"          "FALSE"
```

Having the same type of values allows operations to apply to all elements in the vector

```
v1 <- c(1, 2, 3)
v1
```

```
## [1] 1 2 3
```

```
v1 / 3
```

```
## [1] 0.3333333 0.6666667 1.0000000
```

```
v1 * v1      # Multiply element wise
```

```
## [1] 1 4 9
```

```
v1 %*% v1    # Matrix multiplication
```

```
##      [,1]
## [1,]   14
```

VECTORS

Unidimensional object that stores a sequence of values of the **same type**

```
c("Hello world", 35, FALSE)
```

```
## [1] "Hello world" "35"          "FALSE"
```

```
c("Hello world", as.character(35), as.character(FALSE))  # Equivalent to this
```

```
## [1] "Hello world" "35"          "FALSE"
```

Having the same type of values allows operations to apply to all elements in the vector

```
v1 <- c(1, 2, 3)
v1
```

```
## [1] 1 2 3
```

```
v1 / 3
```

```
## [1] 0.3333333 0.6666667 1.0000000
```

```
v1 * v1      # Multiply element wise
```

```
## [1] 1 4 9
```

```
v1 %*% v1    # Matrix multiplication
```

```
##      [,1]
## [1,]   14
```

```
# Creating vectors
v1 <- c(1, 1, 3, 5)    # Specify all
v1
v2 <- c(1:5)           # Evenly spaced
v2
v3 <- c(v1, v2)        # Combining
v3
```

VECTORS

Unidimensional object that stores a sequence of values of the **same type**

```
c("Hello world", 35, FALSE)
```

```
## [1] "Hello world" "35"          "FALSE"
```

```
c("Hello world", as.character(35), as.character(FALSE))  # Equivalent to this
```

```
## [1] "Hello world" "35"          "FALSE"
```

Having the same type of values allows operations to apply to all elements in the vector

```
v1 <- c(1, 2, 3)
v1
```

```
## [1] 1 2 3
```

```
v1 / 3
```

```
## [1] 0.3333333 0.6666667 1.0000000
```

```
v1 * v1      # Multiply element wise
```

```
## [1] 1 4 9
```

```
v1 %*% v1    # Matrix multiplication
```

```
##      [,1]
## [1,]   14
```

```
# Creating vectors
v1 <- c(1, 1, 3, 5)    # Specify all
v1
```

```
## [1] 1 1 3 5
```

```
v2 <- c(1:5)           # Evenly spaced
v2
```

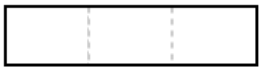
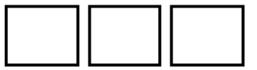
```
## [1] 1 2 3 4 5
```

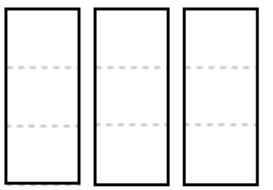
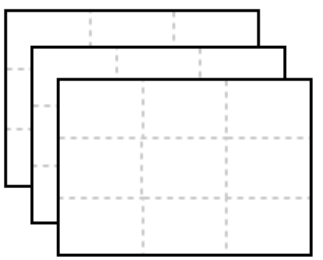
```
v3 <- c(v1, v2)        # Combining
v3
```

```
## [1] 1 1 3 5 1 2 3 4 5
```

DATA STRUCTURE

	single type	multiple types
1D	Vector	List
2D	Matrix	Data frame
nD	Array	

DATA FRAMES AND SUBSETTING

Let's explore the dataset with function `str()`

```
str(imdb)
```

```
## 'data.frame':    250 obs. of  13 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Name          : chr  "Shoah" "Home" "Untouchable" "Le Trou" ...
##  $ Rank          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Year          : chr  "-1985" "(I) (2009)" "-2011" "-1960" ...
....
```

A dataframe is a collection of vectors.
We use `$` to select a variable of the dataset:

```
imdb$Name
```

```
##   [1] "Shoah"
##   [2] "Home"
##   [3] "Untouchable"
....
```

We can also use `$` to create and replace variables

```
imdb$new_var <- c(250:1) # The vector has to be of the same length as the dataframe
```

DATA FRAMES AND SUBSETTING

We can also use the **subsetting** operator `[]` to select a portion of our data. Like this: `data[row, column]` or `vector[element]`.

We indicate what we want to select using:

- Indices (position)
  - *The second column*
- Logical operators
  - *Movies ranked above 10*
- Names:
  - *Columns "Rank" and "Duration"*

First row and cols "Name" and "Year":

```
imdb[1, c("Name", "Year")]
imdb[1, c(2, 4)]
```

DATA FRAMES AND SUBSETTING

We can also use the **subsetting** operator `[]` to select a portion of our data. Like this: `data[row, column]` or `vector[element]`.

We indicate what we want to select using:

- Indices (position)
  - *The second column*
- Logical operators
  - *Movies ranked above 10*
- Names:
  - *Columns "Rank" and "Duration"*

First row and cols "Name" and "Year":

```
imdb[1, c("Name", "Year")]
imdb[1, c(2, 4)]
```

First 10 rows, all columns:

```
imdb[1:10,]
imdb[-(11:250),]
```

DATA FRAMES AND SUBSETTING

We can also use the **subsetting** operator `[]` to select a portion of our data. Like this: `data[row, column]` or `vector[element]`.

We indicate what we want to select using:

- Indices (position)
  - *The second column*
- Logical operators
  - *Movies ranked above 10*
- Names:
  - *Columns "Rank" and "Duration"*

First row and cols "Name" and "Year":

```
imdb[1, c("Name", "Year")]
imdb[1, c(2, 4)]
```

First 10 rows, all columns:

```
imdb[1:10,]
imdb[-(11:250),]
```

Name of the movies ranked 11-14 :

```
imdb[Rank < 15 & Rank > 10, "Name" ]
  # This doesn't work. Why?
imdb[imdb$Rank < 15 &
       imdb$Rank > 10, "Name"]
```

EXERCISE

- Create the vector `c(7, 7, 8, 8, 7, 7, 8, 8)` using the function `rep()`
- Modify the code `seq(1, 6)` to get the vector c(0, 2, 4, 6)
- Get the name of the shortest movie in the data set
  🔍 Use functions `min()`, `substr()`
- There are 6 movies that last 104 minutes, what are they called?

10:00

AGENDA

- GETTING STARTED WITH R
  - THE R STUDIO IDE
  - OBJECTS AND FUNCTIONS
  - IMPORTING DATA
- ANATOMY OF A DATA FRAME
  - SCALARS
  - VECTORS
  - DATAFRAMES
  - SUBSETTING
- MANIPULATING DATA WITH {dplyr}
  - PACKAGES
  - MAIN FUNCTIONS
  - `group_by()` vs. `summarise()`
- THE DATA ANALYSIS PIPELINE
  - A CHECKLIST FOR CLEANING DATA
  - LOOKING FOR HELP

MANIPULATING DATA WITH {dplyr}

There are tooons of user-created functions, which come in *packages*

- So far we've only used functions that are already built in R, also known as **{base}**
  - Packages are centralized on the Comprehensive R Archive Newtork (CRAN)

- An R package is like a lightbulb 💡.
  First you need to install it `install.packages("package_name")`,
  and then turn it on when you need it `library("package_name")`

  - You can avoid loading the packages by specifying the package `package_name::function`
    Advantage: clear for others which package is it coming from, avoids conflicts when packages have functions with the same name.
    Disadvantage: Not practical for functions you use often.

- The 🌈`tidyverse` is an umbrella package that installs several pakages that are useful for data analysis and work well together: {tidyr}, {dplyr}, {ggplot2}, {tibble}, etc.

  - **{dplyr}**: data manipulation
  - **{tidyr}**: converting between different shapes
  - **{ggplot2}**: creating plots

MAIN FUNCTIONS IN `dplyr`

Grammar of data manipulation with very user-friendly functions for the most common tasks:

Function	Description
`mutate()`	Add / modify variables
`select()`	Keep / drop variables (columns)
`filter()`	Keep / drop observations (rows)
`arrange()`	Sort rows according to values of variables
`summarise()`	Aggregate data into descriptive statistics

MAIN FUNCTIONS IN `dplyr`

Grammar of data manipulation with very user-friendly functions for the most common tasks:

Function	Description
`mutate()`	Add / modify variables
`select()`	Keep / drop variables (columns)
`filter()`	Keep / drop observations (rows)
`arrange()`	Sort rows according to values of variables
`summarise()`	Aggregate data into descriptive statistics

The **pipe operator** `%>%` or `|>` (Ctrl + Shift + M)

- Allows to chain functions you apply to an object
- Take X and do this, and this, and this, and ...

MAIN FUNCTIONS IN dplyr

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)
```

```
imdb
```

```
##       X                                  Name Rank       Year
## 1     1                                 Shoah    1      -1985
## 2     2                                  Home    2 (I) (2009)
## 3     3                            Untouchable    3      -2011
## 4     4                               Le Trou     4      -1960
## 5     5                 The Man Who Planted Trees    5      -1987
## 6     6                      Children of Paradise    6      -1945
## 7     7                                 Amélie    7      -2001
## 8     8                               Incendies    8      -2010
## 9     9                                La Jetée    9      -1962
....
```

MAIN FUNCTIONS IN dplyr

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration)                    # Keep selected vars
```

```
##                                                  Name Rank Duration
## 1                                              Shoah    1  566 min
## 2                                               Home    2  118 min
## 3                                        Untouchable    3  112 min
## 4                                           Le Trou     4  131 min
## 5                          The Man Who Planted Trees    5   30 min
## 6                               Children of Paradise    6  189 min
## 7                                            Amélie     7  122 min
## 8                                          Incendies    8  131 min
## 9                                           La Jetée    9   28 min
....
```

MAIN FUNCTIONS IN `dplyr`

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration) %>%                # Keep selected vars
  mutate(top50 = (Rank <= 50))                    # Create a new variable
```

```
##                                                 Name Rank Duration top50
## 1                                              Shoah    1  566 min  TRUE
## 2                                               Home    2  118 min  TRUE
## 3                                        Untouchable    3  112 min  TRUE
## 4                                            Le Trou    4  131 min  TRUE
## 5                          The Man Who Planted Trees    5   30 min  TRUE
## 6                               Children of Paradise    6  189 min  TRUE
## 7                                            Amélie    7  122 min  TRUE
## 8                                          Incendies    8  131 min  TRUE
## 9                                           La Jetée    9   28 min  TRUE
....
```

MAIN FUNCTIONS IN dplyr

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration) %>%                  # Keep selected vars
  mutate(top50 = (Rank <= 50)) %>%                  # Create a new variable
  mutate(Duration =
           as.numeric(substr(Duration, 1, 3)))      # Modify existing var
```

```
##                                                  Name Rank Duration top50
## 1                                              Shoah    1      566  TRUE
## 2                                               Home    2      118  TRUE
## 3                                        Untouchable    3      112  TRUE
## 4                                           Le Trou     4      131  TRUE
## 5                             The Man Who Planted Trees    5       30  TRUE
## 6                                  Children of Paradise    6      189  TRUE
## 7                                             Amélie    7      122  TRUE
## 8                                           Incendies    8      131  TRUE
## 9                                            La Jetée    9       28  TRUE
....
```

MAIN FUNCTIONS IN `dplyr`

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration) %>%                 # Keep selected vars
  mutate(top50 = (Rank <= 50)) %>%                 # Create a new variable
  mutate(Duration =
           as.numeric(substr(Duration, 1, 3))) %>% # Modify existing var
  filter(Name != "Amélie")                         # Keep/drop certain rows
```

```
##                                                 Name Rank Duration top50
## 1                                              Shoah    1      566  TRUE
## 2                                               Home    2      118  TRUE
## 3                                        Untouchable    3      112  TRUE
## 4                                            Le Trou    4      131  TRUE
## 5                          The Man Who Planted Trees    5       30  TRUE
## 6                               Children of Paradise    6      189  TRUE
## 7                                          Incendies    8      131  TRUE
## 8                                           La Jetée    9       28  TRUE
## 9                                      A Man Escaped   10      101  TRUE
....
```

MAIN FUNCTIONS IN `dplyr`

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration) %>%                 # Keep selected vars
  mutate(top50 = (Rank <= 50)) %>%                 # Create a new variable
  mutate(Duration =
           as.numeric(substr(Duration, 1, 3))) %>% # Modify existing var
  filter(Name != "Amélie") %>%                     # Keep/drop certain rows
  arrange(-Rank)                                   # Sort rows
```

```
##                                                 Name Rank Duration top50
## 1                                  Monsieur Ibrahim  250       96 FALSE
## 2                                           Polisse  249      127 FALSE
## 3                                       Read My Lips  248      118 FALSE
## 4                                  What's in a Name?  247      109 FALSE
## 5                                          99 francs  246      100 FALSE
## 6                  The Beat That My Heart Skipped  245      108 FALSE
## 7                                     Certified Copy  244      106 FALSE
## 8                                  La Famille Bélier  243      106 FALSE
## 9                                           Pot Luck  242      122 FALSE
....
```

MAIN FUNCTIONS IN `dplyr`

```
install.packages("dplyr")      # Requires an internet connection

library(dplyr)

imdb %>%
  select(Name, Rank, Duration) %>%                # Keep selected vars
  mutate(top50 = (Rank <= 50)) %>%                # Create a new variable
  mutate(Duration =
           as.numeric(substr(Duration, 1, 3))) %>% # Modify existing var
  filter(Name != "Amélie") %>%                    # Keep/drop certain rows
  arrange(-Rank) %>%                              # Sort rows
  summarise(avg_dur = mean(Duration),             # Aggregate into statistics
            min_dur = min(Duration),
            max_dur = max(Duration))
```

```
##    avg_dur min_dur max_dur
## 1 109.1526      13     566
```

MAIN FUNCTIONS IN `dplyr`

Two useful functions for `mutate()`

`ifelse()`

```
imdb %>%
  select(Name, Type) %>%
  mutate(restricted = ifelse(Type %in% c("R", "X", "18"),
                             TRUE,
                             FALSE )) %>%
  head()
```

```
##                         Name Type restricted
## 1                      Shoah   PG      FALSE
## 2                       Home    U      FALSE
## 3                Untouchable   15      FALSE
## 4                   Le Trou    A      FALSE
## 5 The Man Who Planted Trees           FALSE
## 6      Children of Paradise    A      FALSE
```

MAIN FUNCTIONS IN `dplyr`

Two useful functions for `mutate()`

`ifelse()`

```
imdb %>%
  select(Name, Type) %>%
  mutate(restricted = ifelse(Type %in% c("R", "X", "18"),
                             TRUE,
                             FALSE )) %>%
  head()
```

`case_when()`

```
imdb <-
imdb %>%
  mutate(audience = case_when(Type %in% c("R", "X", "18") ~ "Over 17" ,
                              Type %in% c("PG", "PG-13") ~ "Parental guidance",
                              Type == "" ~ "No information",
                              .default = "General audiences"))
```

```
##                         Name Type          audience
## 1                      Shoah   PG Parental guidance
## 2                       Home    U General audiences
## 3                Untouchable   15 General audiences
## 4                    Le Trou    A General audiences
## 5 The Man Who Planted Trees          No information
## 6      Children of Paradise    A General audiences
```

group_by() AND summarise()

- `group_by()` allows you to do computations separately for different parts of the data

```
imdb %>%
  mutate(mean_all = mean(Rating, na.rm = TRUE)) %>%
  group_by(audience) %>%
  mutate(mean_aud = mean(Rating, na.rm = TRUE)) %>%
  select(Name, audience, Rating, mean_all, mean_aud) %>%
  head(5)
```

```
## # A tibble: 5 × 5
## # Groups:   audience [3]
##   Name                      audience          Rating mean_all mean_aud
##   <chr>                     <chr>              <dbl>    <dbl>    <dbl>
## 1 Shoah                     Parental guidance    8.7     7.62     7.69
## 2 Home                      General audiences    8.5     7.62     7.64
## 3 Untouchable               General audiences    8.5     7.62     7.64
## 4 Le Trou                   General audiences    8.5     7.62     7.64
## 5 The Man Who Planted Trees No information       8.5     7.62     7.48
```

- It will apply to all subsequent operations. You need to `ungroup()` to cancel its effect

```
imdb %>%
  group_by(audience) %>%
  mutate(mean_aud = mean(Rating, na.rm = TRUE)) %>%
  ungroup() %>%
  mutate(mean_all = mean(Rating, na.rm = TRUE))
```

group_by() AND summarise()

`group_by()` is also useful with `summarise()`

- Keeps the grouping variable
  - Computes statistics for each group

```
imdb %>%
  group_by(audience) %>%
  summarise(n = n(),
            avg_rating = mean(Rating))
```

```
## # A tibble: 4 × 3
##   audience              n avg_rating
##   <chr>             <int>      <dbl>
## 1 General audiences   158       7.64
## 2 No information       22       7.48
## 3 Over 17              42       7.59
## 4 Parental guidance    28       7.69
```

- `summarise()` by itself computes statistics for the whole dataset

```
imdb %>%
  summarise(n = n(),
            avg_rating = mean(Rating))
```

```
##     n avg_rating
## 1 250     7.6212
```

🚨 `mutate()` ≠ `summarise()`

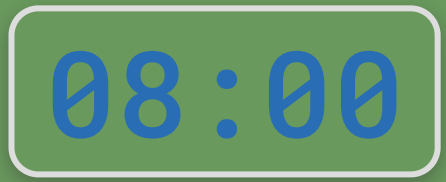
`mutate()` transforms a vector onto another **vector**

`summarise()` transforms a vector onto a **value**

EXERCISE

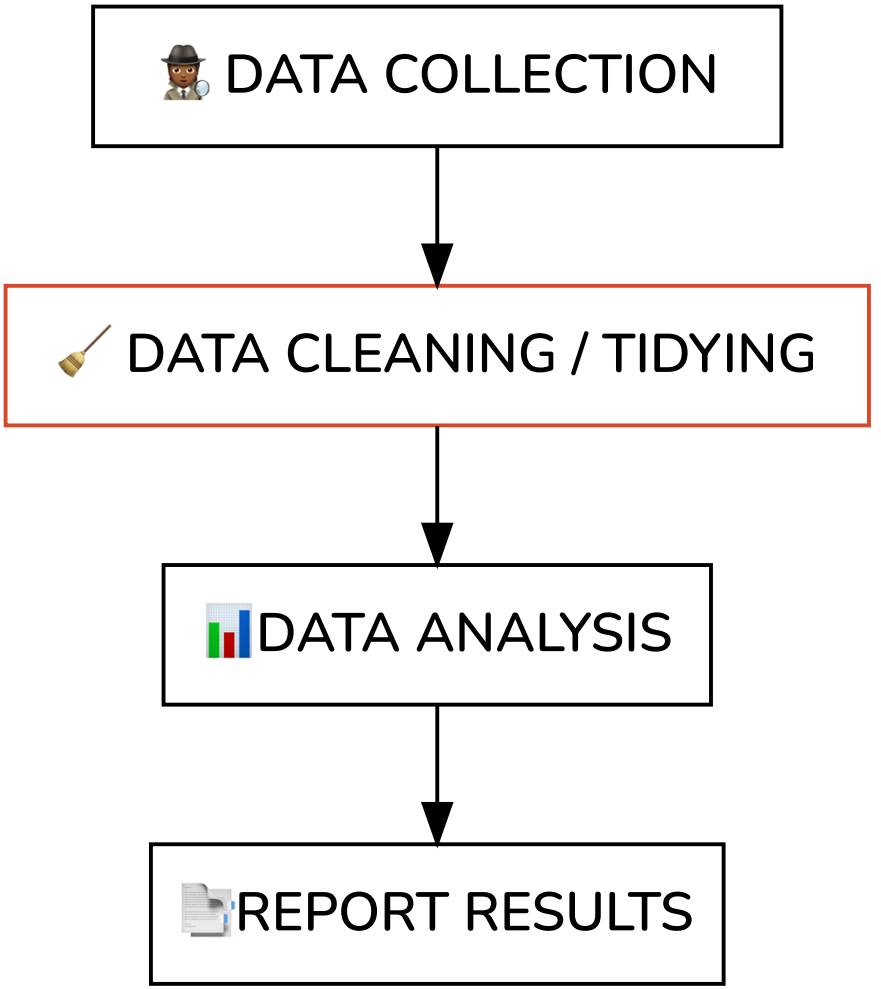
- Create a table that shows the share of comedies and dramas in the sample. 🔍 Use functions `str_detect()` from package {stringr} and `mean()`
- Create a variable *category* that classifies movies into short or long depending on whether or not they exceed 40 min., and create a separate ranking by category. 🔍 Use function `rank()` and don't forget to fix the *Duration* variable if you haven't: `as.numeric(substr(Duration, 1, 3))`

When overwhelmed, break down tasks into smaller steps!! Step away from the screen if you need to.



AGENDA

- GETTING STARTED WITH R
  - THE R STUDIO IDE
  - OBJECTS AND FUNCTIONS
  - IMPORTING DATA
- ANATOMY OF A DATA FRAME
  - SCALARS
  - VECTORS
  - DATAFRAMES
  - SUBSETTING
- MANIPULATING DATA WITH {dplyr}
  - PACKAGES
  - MAIN FUNCTIONS
  - `group_by()` vs. `summarise()`
- THE DATA ANALYSIS PIPELINE
  - A CHECKLIST FOR CLEANING DATA
  - LOOKING FOR HELP


DATA COLLECTION
DATA CLEANING / TIDYING
DATA ANALYSIS
REPORT RESULTS

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
- Data tidying: *is the structure of the data right?*
- Documentation: *is the data understandable to others?*
- Data aggregation: *do we have all we need in the same place?*

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
  - Missing values `is_na()`, `NA_real_`, `NA_character_`
    - 🏴‍☠️ 0s are replacing missing values
  - Storage types (are numbers numbers?)
  - Formats are inconsistent
  - More here: Quartz guide to bad data and how to solve them
- Data tidying: *is the structure of the data right?*
- Documentation: *is the data understandable to others?*
- Data aggregation: *do we have all we need in the same place?*

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
- Data tidying: *is the structure of the data right?* Every column is a variable, every row is an observation every cell is a single value
    - Remove duplicates
    - See examples and tips here
- Documentation: *is the data understandable to others?*
- Data aggregation: *do we have all we need in the same place?*

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
- Data tidying: *is the structure of the data right?*
- Documentation: *is the data understandable to others?*
    - Variable names
    - Variable labels
    - Metadata: describing the collection, characteristics and processing of the data.
- Data aggregation: *do we have all we need in the same place?*

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
- Data tidying: *is the structure of the data right?*
- Documentation: *is the data understandable to others?*
- Data aggregation: *do we have all we need in the same place?*
    - Merging datasets (adding columns) `left_join()`, `right_join()`
    - Appending datasets (adding rows) `rbind()`

A CHECKLIST FOR CLEANING DATA

We've learned the tools for manipulating data. What does it mean to clean and tidy our data?

- Variable management: *is the data stored in the right way?*
- Data tidying: *is the structure of the data right?*
- Documentation: *is the data understandable to others?*
- Data aggregation: *do we have all we need in the same place?*

⭐TWO RULES:

1. **Never save over raw data.** Keep the original data in a separate folder.
2. **Document everything.** `#` is your best friend.

A CHECKLIST FOR CLEANING DATA

- Variable management: *is the data stored in the right way?*
  - Missing values `is_na()`, `NA_real_`, `NA_character_`
    - 🏴‍☠️ 0s are replacing missing values
  - Storage types (are numbers numbers?)
  - Formats are inconsistent
  - More here: Quartz guide to bad data and how to solve them
- Data tidying: *is the structure of the data right?* Every column is a variable, every row is an observation every cell is a single value
  - Remove duplicates
  - See examples and tips here
- Documentation: *is the data understandable to others?*
  - Variable names
  - Variable labels
  - Metadata: describing the collection, characteristics and processing of the data.
- Data aggregation: *do we have all we need in the same place?*
  - Merging datasets (adding columns) `left_join()`, `right_join()`
  - Appending datasets (adding rows) `rbind()`

⭐TWO RULES:

1. **Never save over raw data.** Keep the original data in a separate folder.
2. **Document everything.** `#` is your best friend.

A WORD ON LEARNING R AND LOOKING FOR HELP

- We talked before about debugging and the importance of reading documentation.
- **Don't pipe blindfolded !** Check that each command does what it's expected to do.
  How is this command treating missing values? (More on *defensive programming* on Friday)
- If things are not working out, check that you specified all the arguments you need and that they are appropriate. `?function()` is free!!
- Search online. Your questions have fore sure been asked before on ***stackoverflow.***
  - Learning how to Google is a skill in itself. Think about your issue in the most abstract way.
  - If you can't understand an error message, paste the error message on Google along with the name of your command.
  - ⭐ Ask in the course website. I'll do my best to support you throughout this year.
- 💎 **Check for typos.** 70% of the time this is the issue.

HOMEWORK

🔔 Due on Wednesday before 9am .

Submit using this link and following the instructions here.

✅ Inspect the IMDB dataset [🔍`glimpse()` `table()`] and describe 5 issues we should solve in the data cleaning/tidying process that we haven't addressed in this lecture. Write down the name of at least one function that would help for this. Google if you need!

- The year variable is not numeric and has string characters. 🔍`str_length()`, `str_sub()`, `as.numeric()`

✅ Are older movies better?

1) Fix the year variable. This can be done in multiple ways! Try with 🔍`str_length()` and `str_sub()`

2) Create a variable that indicates whether a movie is older or more recent than the average year in the sample.

3) Create a table with the average rating by decade (start from the 50s).

You can **work** together, but don't copy-paste code

🖍️ Don't forget to comment your code, describe what you are doing

RESOURCES

Learning R (Manipulating data)

- YaRrr The Pirate's Guide to R
- Previous editions of this course: Louis Sirugue; Hannah Bull / Lea Dousset
- Introduction to dplyr

Data analysis pipeline (Cleaning and tidying)

- Development Research in Practice: *The DIME Analytics Data Handbook*
- DIME Data Cleaning Checklist
- Innovations for Poverty Action Data Cleaning Guide