



SESSION 1: DATA MANAGEMENT

DR. SOFIA GIL-CLAVEL

- ❖ Data types
- ❖ Dataframes
- ❖ Tidyverse

0. R & RSTUDIO

What is R?
What is RStudio?

WHAT IS R?

R is a dialect of S.

S is a language that was developed by John Chambers and others at the old Bell Telephone Laboratories, originally part of AT&T Corp. S was initiated in 1976 as an internal statistical analysis environment.

THE PHILOSOPHY OF S

The S language had its roots in data analysis, and did not come from a traditional programming language background. **Its inventors were focused on figuring out how to make data analysis easier**, first for themselves, and then eventually for others. [...] **They built a language that would be suitable for interactive data analysis** (more command-line based) as well as for writing longer programs (more traditional programming language-like).

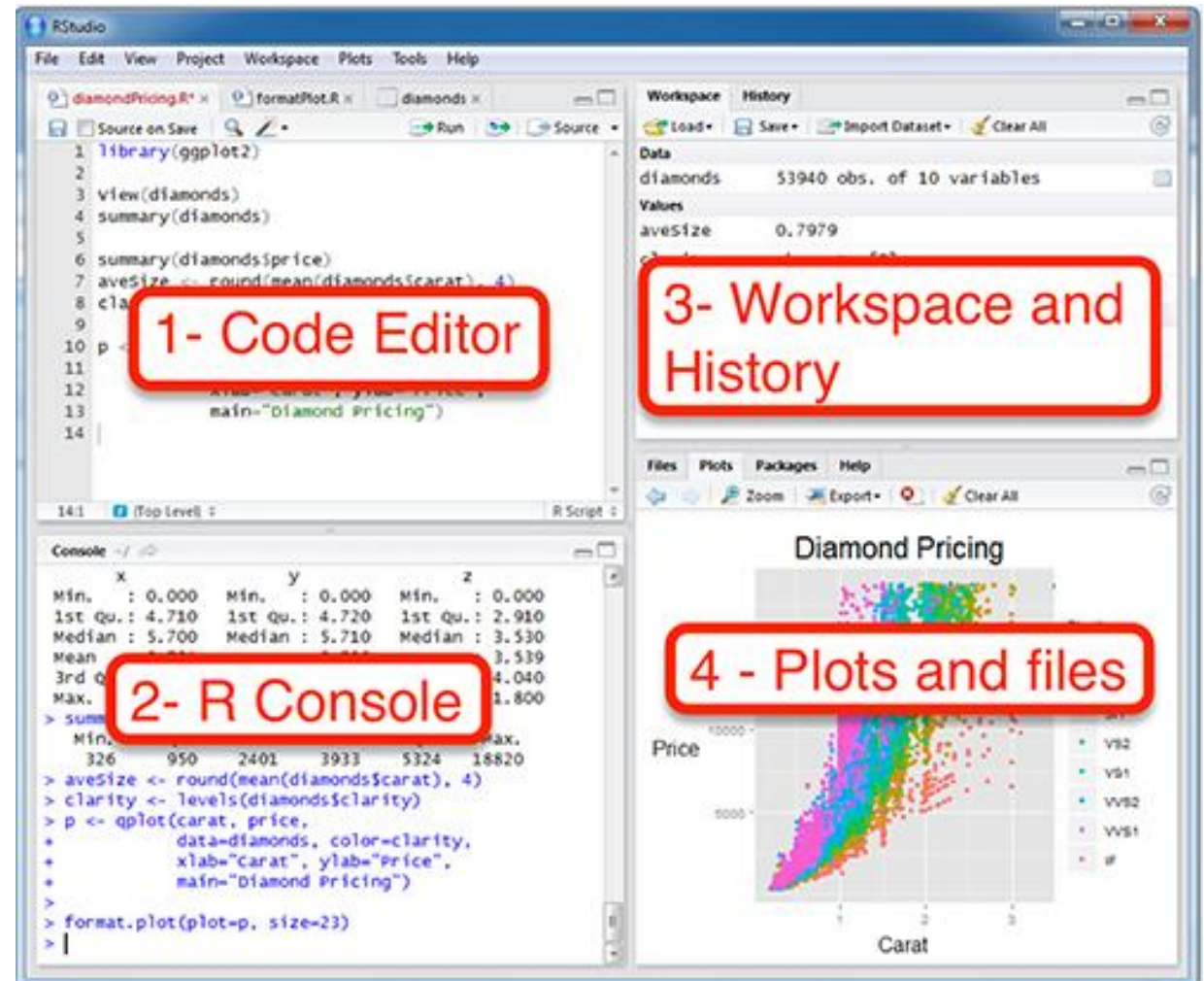
Source: Peng, Roger D. *R Programming for Data Science*. Accessed May 5, 2024. <https://bookdown.org/rdpeng/rprogdatascience/>.

WHAT IS RSTUDIO?

RStudio is an integrated development environment for R.

"RStudio." In Wikipedia, May 1, 2024.

<https://en.wikipedia.org/w/index.php?title=RStudio&oldid=1221670801>



HOMework 0

- Download R & RStudio

LET'S OPEN RSTUDIO!



1. A SHORT INTRODUCTION TO R DATA TYPES

Primitive Data Types
Vectors

❖ DATA TYPES

A data type is a set of values and operations associated with those values.

Primitive: These are the simplest types of data, because they are not constructed from other types and are unique entities that cannot be decomposed into others. These data types are defined by a set of values and a set of operations that act on those values.

- Integer
- Floating/Double
- Character and String
- Boolean

PRIMITIVE DATA

- **Numerical Type:**

- Integer
- Floating/Double

- **Character:** This type of data consists of the set of characters available for a specific language on a specific computer. The most used character code is ASCII.

- A **String** is multiple characters together

- **Boolean:** This is the simplest data type, as it only has two values (TRUE (1) and FALSE (0))

Integer	Values	-inf, ..., -2, -1, 0, 1, 2, ..., inf
	Operations	*, +, -, %, /, ...
Float	Values	..., -0.6, 0.0, 0.5, ...
	Operations	*, +, -, %, /, ...
Character/ String	Values	'\n', 'A', ..., 'Z', 'a', ..., 'z'
	Operations	<, >, ==, ...
Boolean	Values	TRUE, FALSE
	Operations	<, >, ==, ...

PRIMITIVE DATA

- **Numerical Type:**

- Integer: **Age**
- Floating/Double: **Height**

- **Character:** This type of data consists of the set of characters available for a specific language on a specific computer. The most used character code is ASCII.

- A **String** is multiple characters together: **Country names**

- **Boolean:** This is the simplest data type, as it only has two values (TRUE (1) and FALSE (0)): **Gender – Women (1), Man (0)**

Integer	Values	-inf, ..., -2, -1, 0, 1, 2, ..., inf
	Operations	*, +, -, %, /, ...
Float	Values	..., -0.6, 0.0, 0.5, ...
	Operations	*, +, -, %, /, ...
Character/ String	Values	'\n', 'A', ..., 'Z', 'a', ..., 'z'
	Operations	<, >, ==, ...
Boolean	Values	TRUE, FALSE
	Operations	<, >, ==, ...

ASSIGNMENT OPERATORS

Assignment Operators	
=	Makes a copy of the assigned object.
<-	Makes a copy of the assigned object.

> a=10	Assign the integer 10 to the variable a .
> b<-15	Assign the integer 15 to the variables b .
>	
> a+b	Perform the addition of variable a and b .
[1] 25	
>	
> c<-a+b	Assign the result of the addition of variable a and b to c .
> c	
[1] 25	
> rm(a)	Remove variable a from the environment.
> gc()	Clean the memory.

EXERCISE 1.1

Choose two operators from each category and apply them to different pairs of primitive data types. When applying them, think of possible contexts where you would use them, e.g. “+” to add the years lived by a person.

Operators					
Aritmetic		Comparative		Logical	
+	addition	<	less than	! x	logic NO
-	subtraction	>	more than	x & y	element-wise AND
*	multiplication	<=	less or equal than	x && y	single comparison AND
/	division	>=	more or equal than	x y	element-wise OR
^	power	==	equal than	x y	single comparison OR
%%	integer division	!=	different than	xor(x,y)	exclusive OR

❖ DATA TYPES

A data type is a set of values and operations associated with those values.

Primitive: These are the simplest types of data, because they are not constructed from other types and are unique entities that cannot be decomposed into others. These data types are defined by a set of values and a set of operations that act on those values.

- Integer
- Floating/Double
- Logical
- Character

Composite Data: These are types of data whose values are collections of primitive data.

- Vectors

VECTORS

A vector is a collection of observations or measurements of the same type, for example the ages of 50 people or the number of coffees we drink on different days of the week. Vectors are the simplest types of aggregated data, it is with them that more complex structures are created.

In R, the function with which a vector is created is `c()`:

```
R> myvec <- c(1,3,1,42)
R> myvec
[1] 1 3 1 42
```

VECTORS: FACTORS

As mentioned in page 10, strings and Booleans data types are the equivalent of categories (country name) and dichotomous (gender) variables. However, to make this clear to R, we need to use the function **factor()**.

Create the vector and check its class.

```
> COUNTRY=c("MX","DE","NL")
> class(COUNTRY)
[1] "character"
```

Use the **factor** to turn the vector into a categorical variable.

```
>
> COUNTRY=factor(COUNTRY,levels = c("MX","DE","NL"),labels = c("Mexico","Germany","Netherlands"))
> class(COUNTRY)
[1] "factor"
```

Use **levels** to see the categories. The first one is the reference.

```
> levels(COUNTRY)
[1] "Mexico"      "Germany"     "Netherlands"
```

EXERCISE 1.2

Create a categorical vector with four of the Dutch political parties. You can check them here:

https://en.wikipedia.org/wiki/List_of_political_parties_in_the_Netherlands

How would use their acronyms and their names?

2. DATAFRAMES

Dataframes.
From table/csv to dataframes & vice-versa.
Tidyverse.

❖ DATAFRAMES

A dataframe is a special type of R object where all the items are vectors of the same length. In a dataframe, the elements of different columns can be of different types.

In the example beneath, we are constructing a dataframe with 3 columns ('n', 's', and 'b'). 'n' is numeric. 's' is string. 'b' is Boolean.

To glue them together, we use the R function **data.frame()**.

Data Frame Construction

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b)
```

❖ DATAFRAMES

A dataframe is a special type of R object where all the items are vectors of the same length. In a dataframe, the elements of different columns can be of different types.

In the example beneath, we are constructing a dataframe with 3 columns ('n', 's', and 'b'). 'n' is numeric. 's' is string. 'b' is Boolean.

To glue them together, we use the R function **data.frame()**.

Data Frame Construction

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b)
```

R Example Data Frame

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	...
Mazda RX4	21.0	6	160	110	3.90	2.62	...
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	...
Datsun 710	22.8	4	108	93	3.85	2.32	...
.....							

❖ DATAFRAMES

The dataframe name

The columns:
`names(mtcars)`
or
`col.names(mtcars)`

The row names:
`row.names(mtcars)`

```
> mtcars
```

	mpg	cyl	disp	hp	drat	wt	...
Mazda RX4	21.0	6	160	110	3.90	2.62	...
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	...
Datsun 710	22.8	4	108	93	3.85	2.32	...
.....							

A variable:
`mtcars$disp`

A value or element:
`mtcars$wt[3]`

EXERCISE 2.1

Create a database to save the artists information:

1. Mary Leonora Carrington. British-born, naturalized Mexican. Surrealist painter. Died in May 25, 2011.
2. Remedios Varo. Spanish-born, naturalized Mexican. Surrealist painter. Died: October 8, 1963.
3. David Alfaro Siqueiros. Mexican. Social realist painter. Died: January 6, 1974.

OPEN, MODIFY, AND SAVE

With the *foreign* library you can manipulate databases other than ".csv"

Open: `read.csv()`

You can open “csv” files using this function.

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

Save: `write.csv()`

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",  
           eol = "\n", na = "NA", dec = ".", row.names = TRUE,  
           col.names = TRUE, qmethod = c("escape", "double"),  
           fileEncoding = "")
```


```
write.csv(...)
```

EXERCISE 2.2

Save the artists dataframe. Do you need the row.names parameter?

```
DIR="WRITE HERE THE PATH TO THE DATA\\"
```

```
write.csv(Artist, paste0(DIR, "Data_Name.csv"), row.names = FALSE)
```



With the command **help("paste0")** you can check what this function do.

3. TIDYVERSE

❖ TIDYVERSE: [HTTPS://WWW.TIDYVERSE.ORG/](https://www.tidyverse.org/)



R packages for data science

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

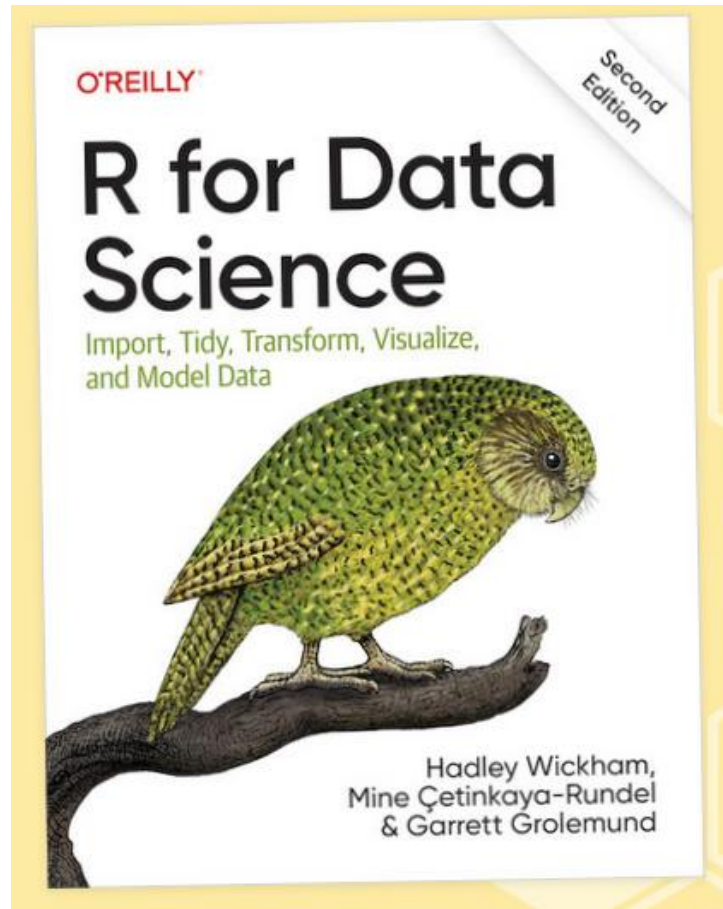
Install the complete tidyverse with:

```
install.packages("tidyverse")
```

**10 MINS BREAK
WHILE TIDYVERSE INSTALLS**



❖ TIDYVERSE: [HTTPS://WWW.TIDYVERSE.ORG/](https://www.tidyverse.org/)



Learn the tidyverse

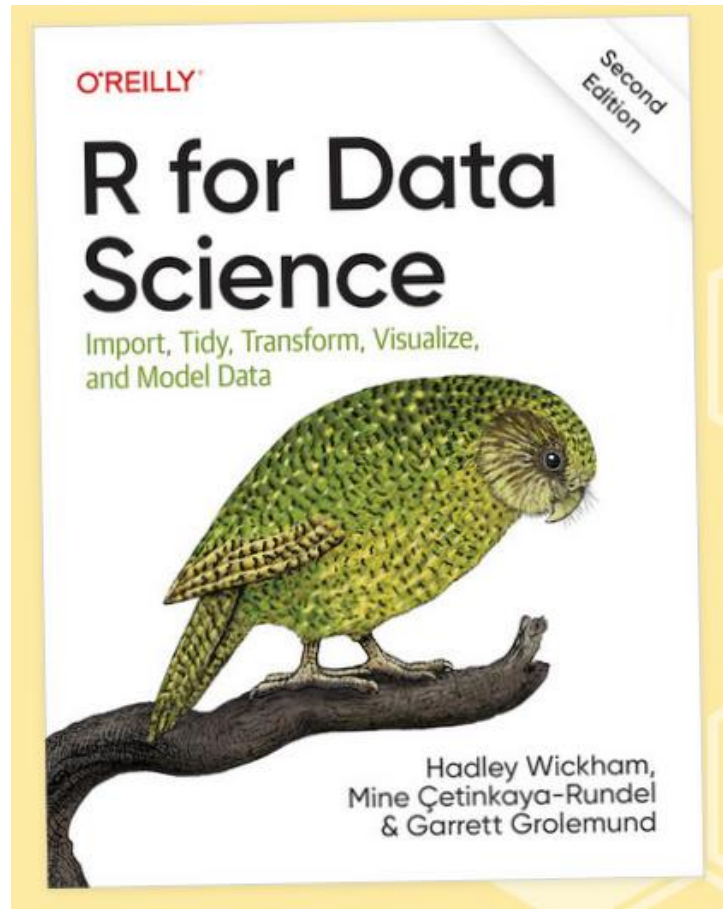
See how the tidyverse makes data science faster, easier and more fun with “R for Data Science (2e)”.

Read it **online**, buy **the book** or try another **resource** from the community.

❖ TIDYVERSE: [HTTPS://WWW.TIDYVERSE.ORG/](https://www.tidyverse.org/)

You can check the
bookdown version of the
book in:

<https://r4ds.hadley.nz/>



Learn the tidyverse

See how the tidyverse makes data science faster,
easier and more fun with “R for Data Science (2e)”.

Read it **online**, buy **the book** or try another
resource from the community.

❖ TIDYVERSE: [HTTPS://WWW.TIDYVERSE.ORG/](https://www.tidyverse.org/)

You can check the
bookdown version of the
book in:
<https://r4ds.hadley.nz/>

This is the website for the 2nd edition of “**R for Data Science**”. This book will teach you how to do data science with R: **You’ll learn how to get your data into R, get it into the most useful structure, transform it and visualize.**

Data science is a vast field, and there’s no way you can master it all by reading a single book. This book aims to give you a solid foundation in the most important tools and enough knowledge to find the resources to learn more when necessary. Our model of the steps of a typical data science project looks something like [Figure 1](#).

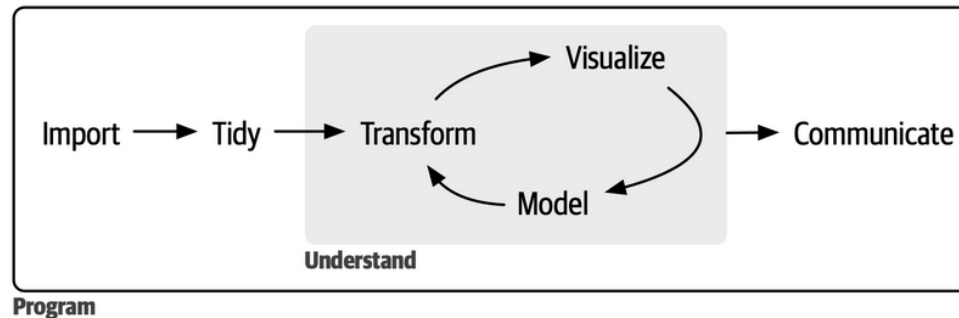
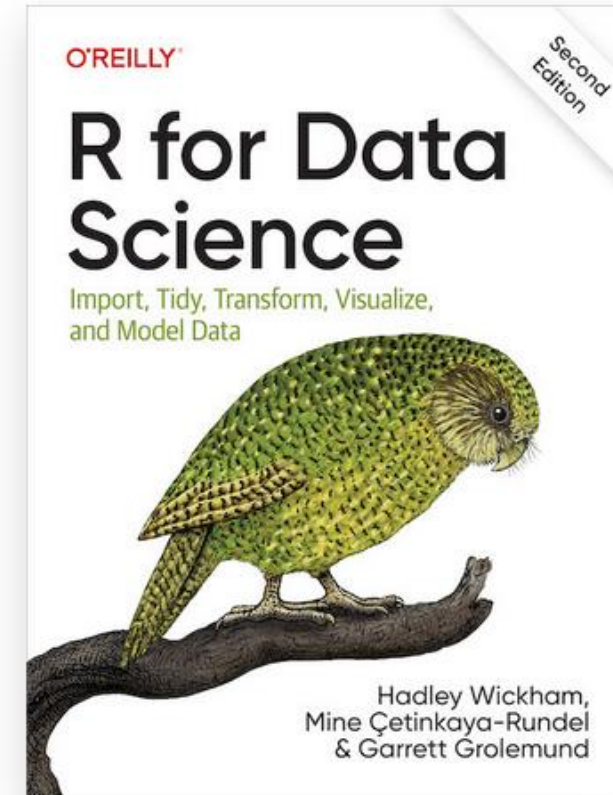


Figure 1: In our model of the data science process, you start with data import and tidying. Next, you understand your data with an iterative cycle of transforming, visualizing, and modeling. You finish the process by communicating your results to other humans.



❖ TIDYVERSE: [HTTPS://WWW.TIDYVERSE.ORG/](https://www.tidyverse.org/)

You can check the bookdown version of the book in:
<https://r4ds.hadley.nz/>

This is the website for the 2nd edition of “**R for Data Science**”. This book will teach you how to do data science with R: You’ll learn how to get your data into R, get it into the most useful structure, transform it and visualize.

Data science is a vast field, and there’s no way you can master it all by reading a single book. This book aims to give you a solid foundation in the most important tools and enough knowledge to find the resources to learn more when necessary. Our model of the steps of a typical data science project looks something like [Figure 1](#).

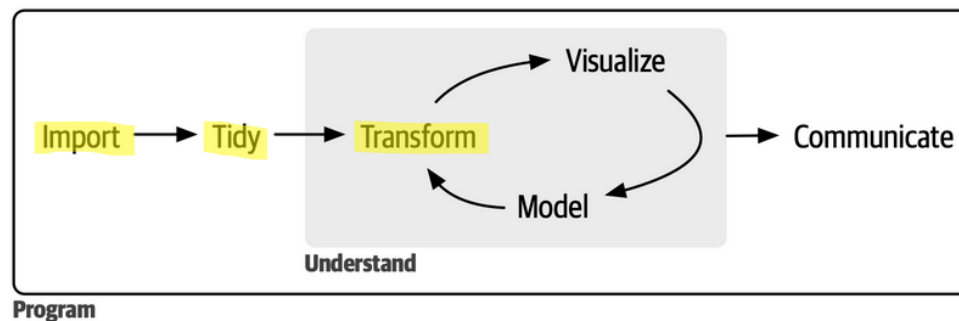
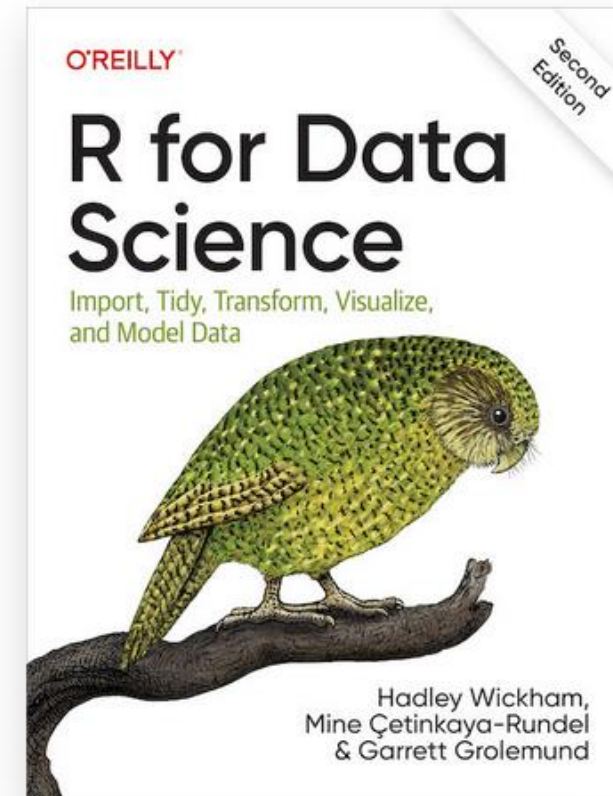


Figure 1: In our model of the data science process, you start with data import and tidying. Next, you understand your data with an iterative cycle of transforming, visualizing, and modeling. You finish the process by communicating your results to other humans.



❖ TIBBLES: [HTTPS://TIBBLE.TIDYVERSE.ORG/](https://tibble.tidyverse.org/)



```
> iris
# A tibble: 150 × 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
    <dbl>         <dbl>         <dbl>         <dbl>    <fct>
1         5.1           3.5           1.4           0.2 setosa
2         4.9           3           1.4           0.2 setosa
3         4.7           3.2           1.3           0.2 setosa
4         4.6           3.1           1.5           0.2 setosa
5          5           3.6           1.4           0.2 setosa
6         5.4           3.9           1.7           0.4 setosa
7         4.6           3.4           1.4           0.3 setosa
8          5           3.4           1.5           0.2 setosa
9         4.4           2.9           1.4           0.2 setosa
10        4.9           3.1           1.5           0.1 setosa
# i 140 more rows
# i Use `print(n = ...)` to see more rows
```

```
library(tidyverse)
?datasets

View(iris)
?iris
summary(iris)

iris
as_tibble(iris)

iris<-as_tibble(iris)
glimpse(iris)
```

❖ PIPES (%>%): [HTTPS://MAGRITTR.TIDYVERSE.ORG/](https://magrittr.tidyverse.org/)



Basic piping

- `x %>% f` is equivalent to `f(x)`
- `x %>% f(y)` is equivalent to `f(x, y)`
- `x %>% f %>% g %>% h` is equivalent to `h(g(f(x)))`

Here, “equivalent” is not technically exact: evaluation is non-standard, and the left-hand side is evaluated before passed on to the right-hand side expression. However, in most cases this has no practical implication.

❖ DPLYR: [HTTPS://DPLYR.TIDYVERSE.ORG/](https://dplyr.tidyverse.org/)



dplyr

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows.

These all combine naturally with `group_by()` which allows you to perform any operation “by group”. You can learn more about them in `vignette("dplyr")`. As well as these single-table verbs, dplyr also provides a variety of two-table verbs, which you can learn about in `vignette("two-table")`.

If you are new to dplyr, the best place to start is the [data transformation chapter](#) in R for Data Science.

EXERCISE 3.1

What do the following functions do?

pipe (%>%) <pre>iris%>% function()</pre>	select <pre>iris%>% select(Sepal.Length, Species)</pre>	mutate <pre>iris%>% mutate(SL_ = Sepal.Length <= median(Sepal.Length))</pre>
arrange <pre>iris%>% arrange(Sepal.Length)</pre>	filter <pre>iris%>% filter(Species == "setosa")</pre>	summarise <pre>iris%>% summarise(mean = mean(Sepal.Length), n = n())</pre>

EXERCISE 3.2

In the Data folder you will find HistorialCred.csv, for this exercise you will have to create the code in R to be able to **open and find the number of people who can have a loan with the bank according to the following restrictions:**

- a) The person does not have a bad credit history.
- b) The person has a monthly income greater than 1000EUR.
- c) The person is not older than 65 years old.

The bank applies the following exceptions:

- If you are an employee of the bank, you are granted.
- If you are over the age of 65, but your monthly income exceeds 5000EUR, you are awarded.

Once the people have been found, the subbase must be saved in a new ".csv". What are the dimensions of the resulting base? What kind of data did you have to work with?

Group by one or more variables



Source: [R/group-by.R](#)

Most data operations are done on groups defined by variables. `group_by()` takes an existing tbl and converts it into a grouped tbl where operations are performed "by group". `ungroup()` removes grouping.

```
group_by(.data, ..., .add = FALSE, .drop = group_by_drop_default(.data))  
  
ungroup(x, ...)
```

Source: https://dplyr.tidyverse.org/reference/group_by.html

EXERCISE 3.3

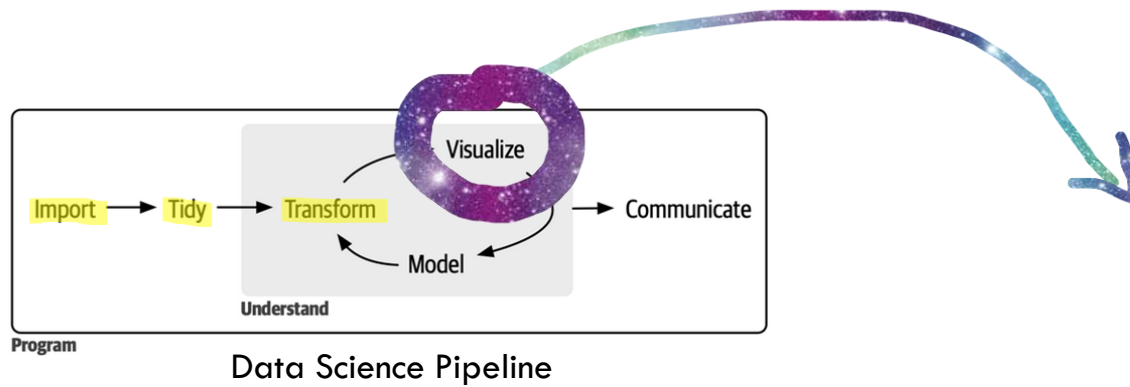
With the subbase that was found in "Exercise 3.2" find the following data:

1. Number of Men and Women Who Can Have the Credit.
2. Average Age of Men and Women.
3. Median monthly Income of Women and Men Over 25.
4. Number of Bank Female and Male employees in the database.
5. Average monthly Income of Female and Male Bank Employees.

REFERENCES

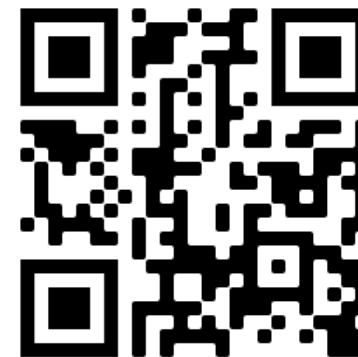
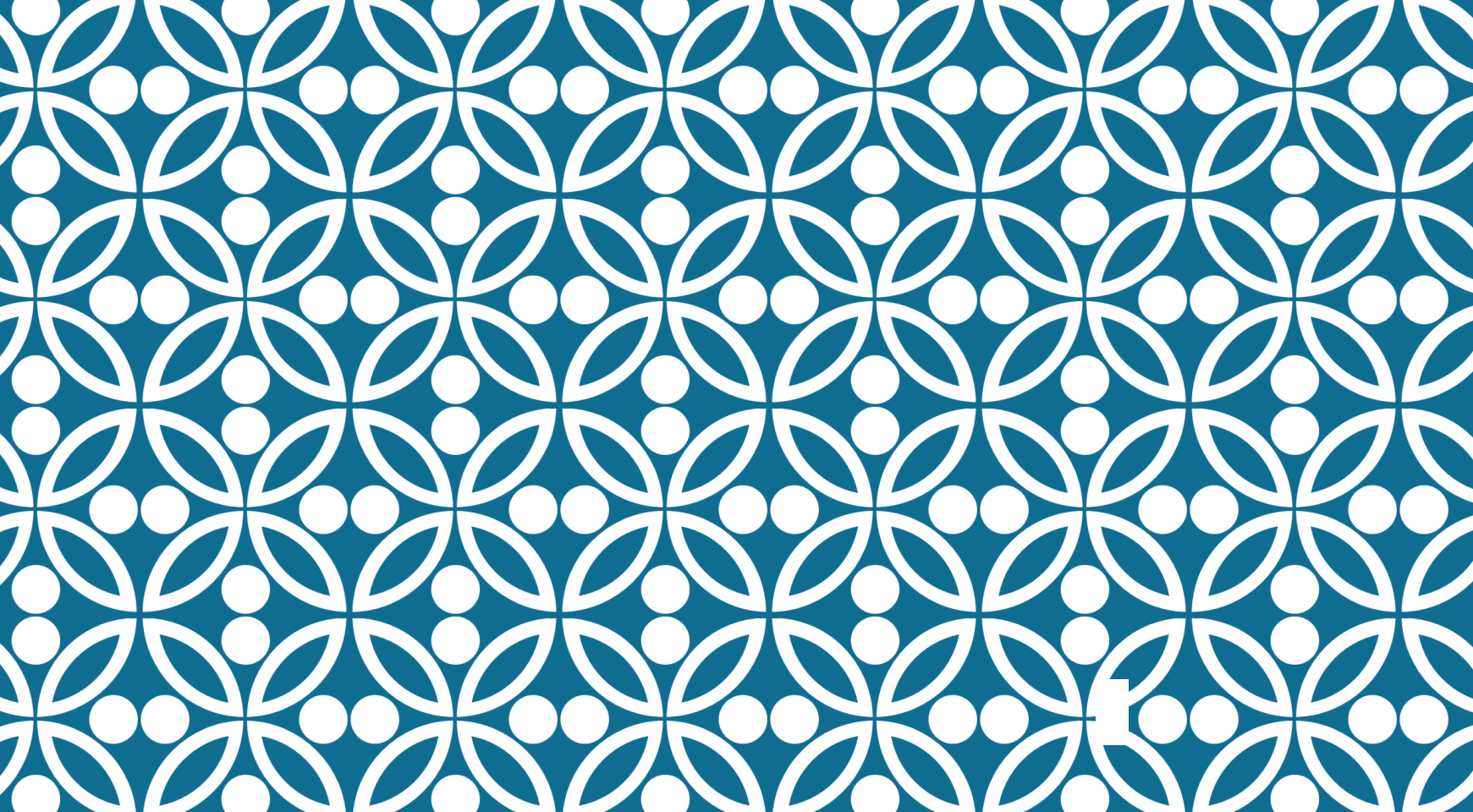
- Albert, Jim, and Maria Rizzo. *R by Example: Concepts to Code*. Use R! New York, NY: Springer New York, 2012. <https://doi.org/10.1007/978-1-4614-1365-3>.
- Davies, Tilman M. *The Book of R: A First Course in Programming and Statistics*. San Francisco: No Starch Press, 2016.
https://web.itu.edu.tr/~tokerem/The_Book_of_R.pdf
- Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. *R for data science*. "O'Reilly Media, Inc.", 2023. Accessed May 7, 2024.
<https://r4ds.hadley.nz/>.

SESSION 2: GGPLOT2 AND THE GRAMMAR OF GRAPHS



ggplot2 is a system for declaratively creating graphics, based on [The Grammar of Graphics](https://www.ggplot2.tidyverse.org/). You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Source: <https://ggplot2.tidyverse.org/>



<https://sofiag1l.github.io/>

THANKS!

Dr. Sofia Gil-Clavel