

Session 1 - Introduction to R

R training

María Reyes Retana
The World Bank | November 2024





Reproducible Research Repository

Government Analytics and R Training:

Strengthening Public Sector Reporting
and Data Analysis

November 30 - December 3, 2024



Introduction

Introduction

About this training

- This is an **introduction** to data work and statistical programming in R
- The training does not require any background in statistical programming
- A computer with R and RStudio installed is required to complete the exercises
- Internet connection is required to download training materials

Introduction

Learning objectives

By the end of the training, you will know:

- How to write **basic** R code
- A notion of how to conduct data work in R and how it differentiates from Excel

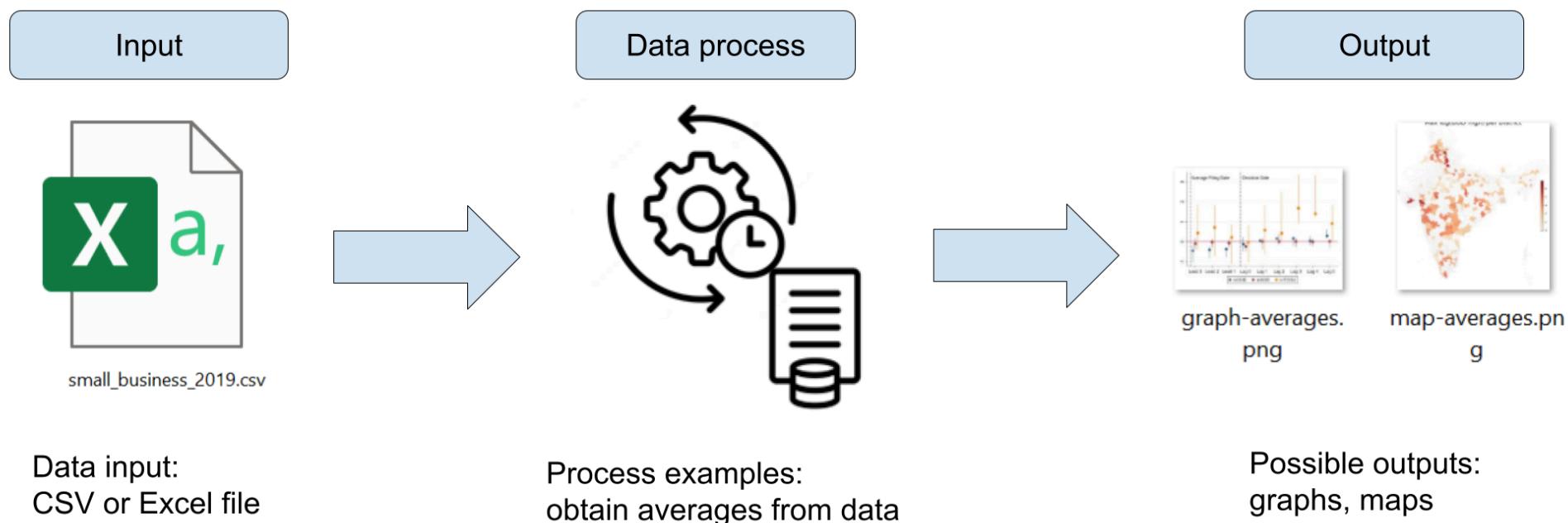


Data work and Statistical Programming

Data work

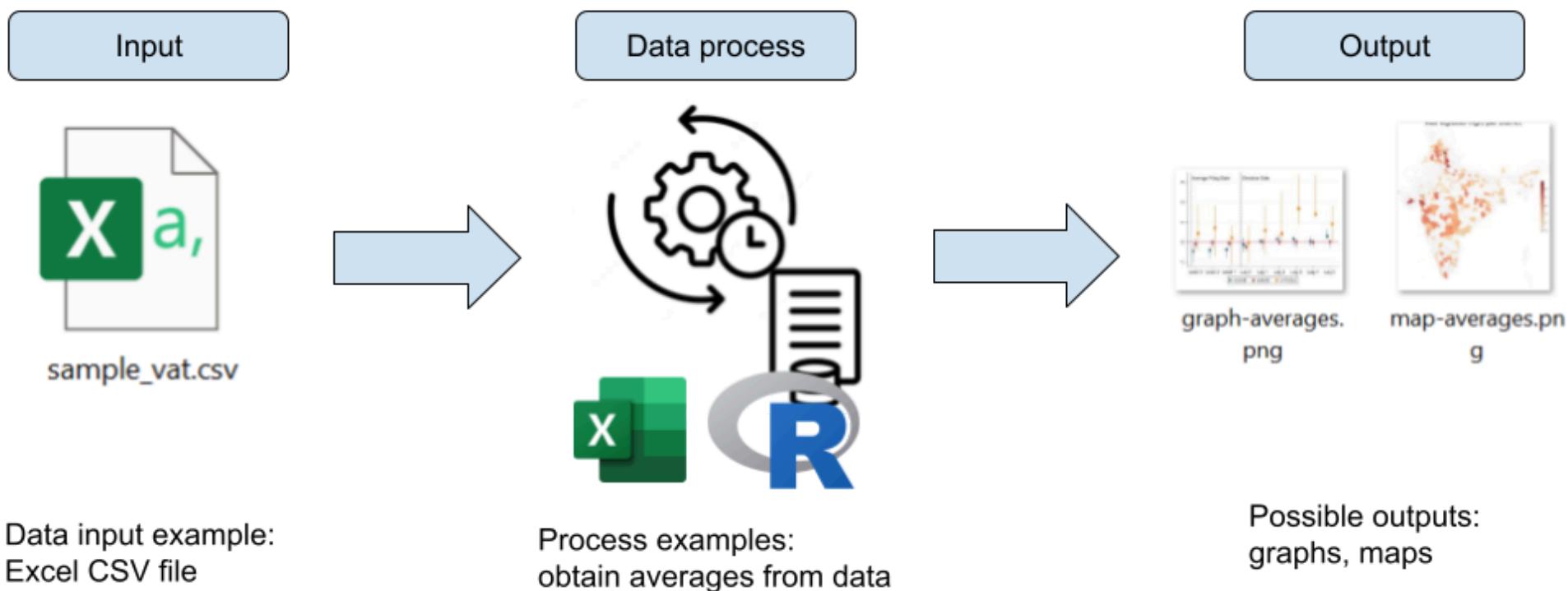
For the context of this training, we'll call data work everything that:

1. Starts with a data input
2. Runs some process with the data
3. Produces an output with the result



Data work

- It's also possible to do data work with Excel
- However, we will show in this training why using statistical programming (through R) is a better way of conducting data work



Statistical Programming

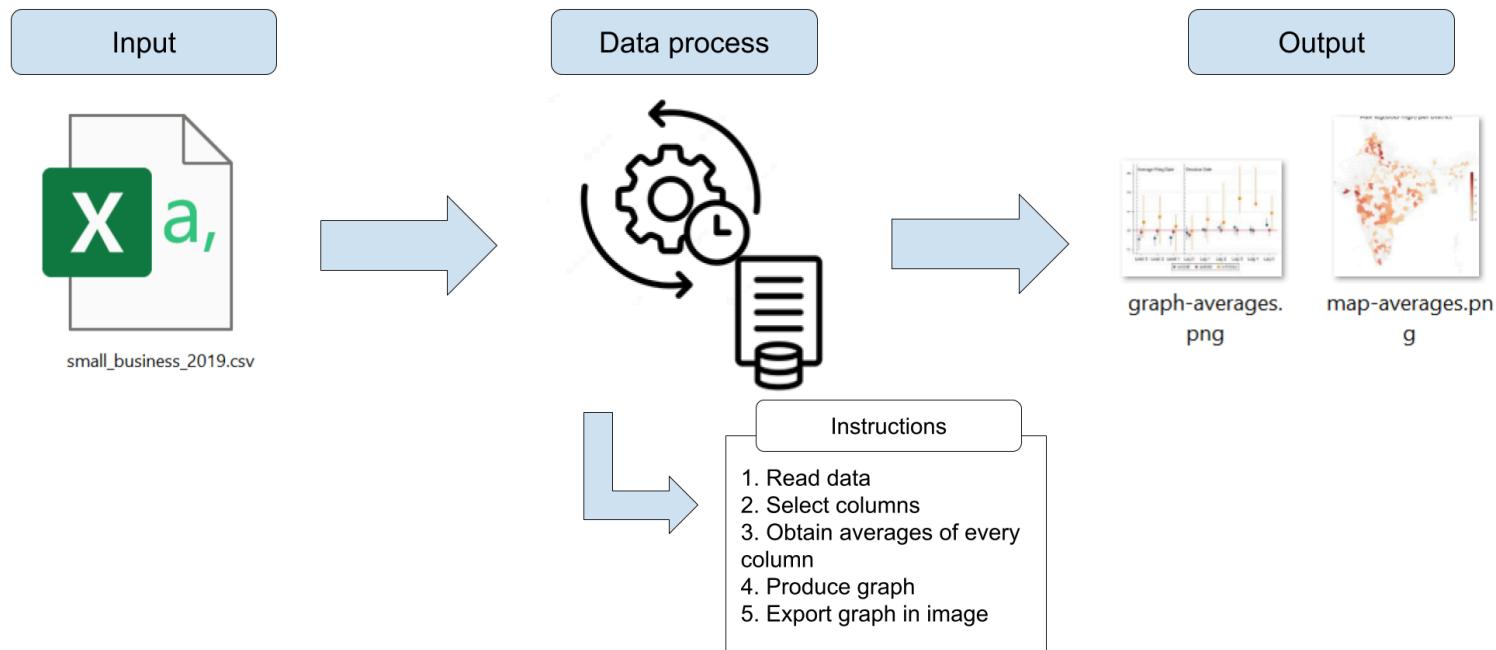
How Can it Benefit My Work?

- **Reproducibility:** Code provides an exact record of every step, making results easy to trace and verify.
- **Reusability:** Code can be reused for similar projects, saving time on future reports.
- **Transparency Over Excel:** Instructions are documented and less prone to error compared to manual steps in Excel.



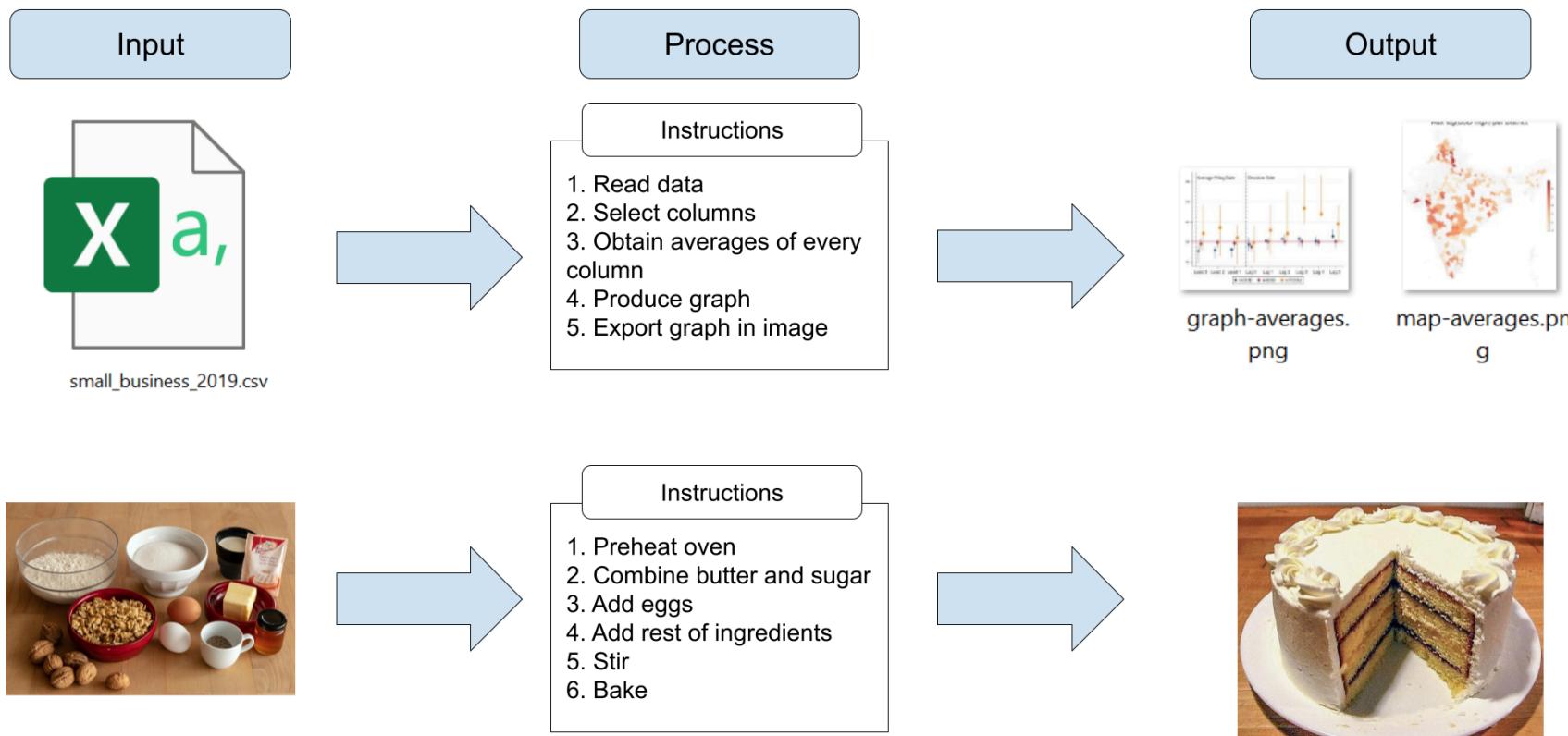
Statistical Programming

- Programming consists of producing instructions to a computer to do something
- In the context of data work, that "something" is statistical analysis or mathematical operations
- Hence, statistical programming consists of producing instructions so our computers will conduct statistical analysis on data



Statistical Programming

- You can think of statistical programming as writing a recipe



Statistical Programming

Why use R

- Statistical programming can be implemented through many different software. Other options are Stata and Python
- We recommend using R for these reasons:
 - R is free
 - R was designed specifically for statistical programming
 - There is a large worldwide community of R users. This means you can easily look for help or examples of code in the internet



Statistical Programming

It's not unusual to struggle at first but it gets better!

- By the end of this training, you will learn how to leverage your existing data to create exhibits for your annual reports using R.
- We know this may feel challenging, but you'll get there!
- Remember to ask questions, be patient with yourself, and take it step by step.



Statistical Programming

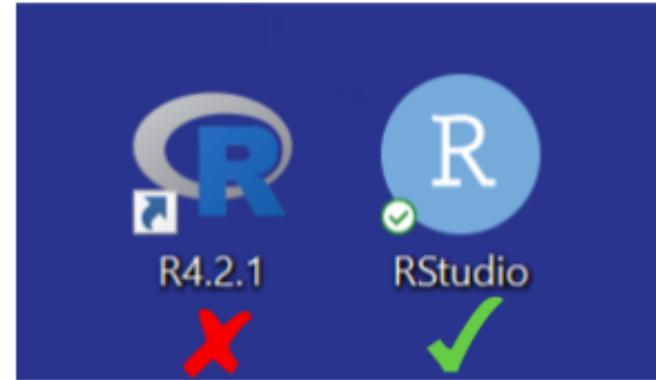
How to write R code?

- The rest of today's session focuses on the basics of writing R code
- We'll use RStudio to write R code in this training

Statistical Programming

How to write R code?

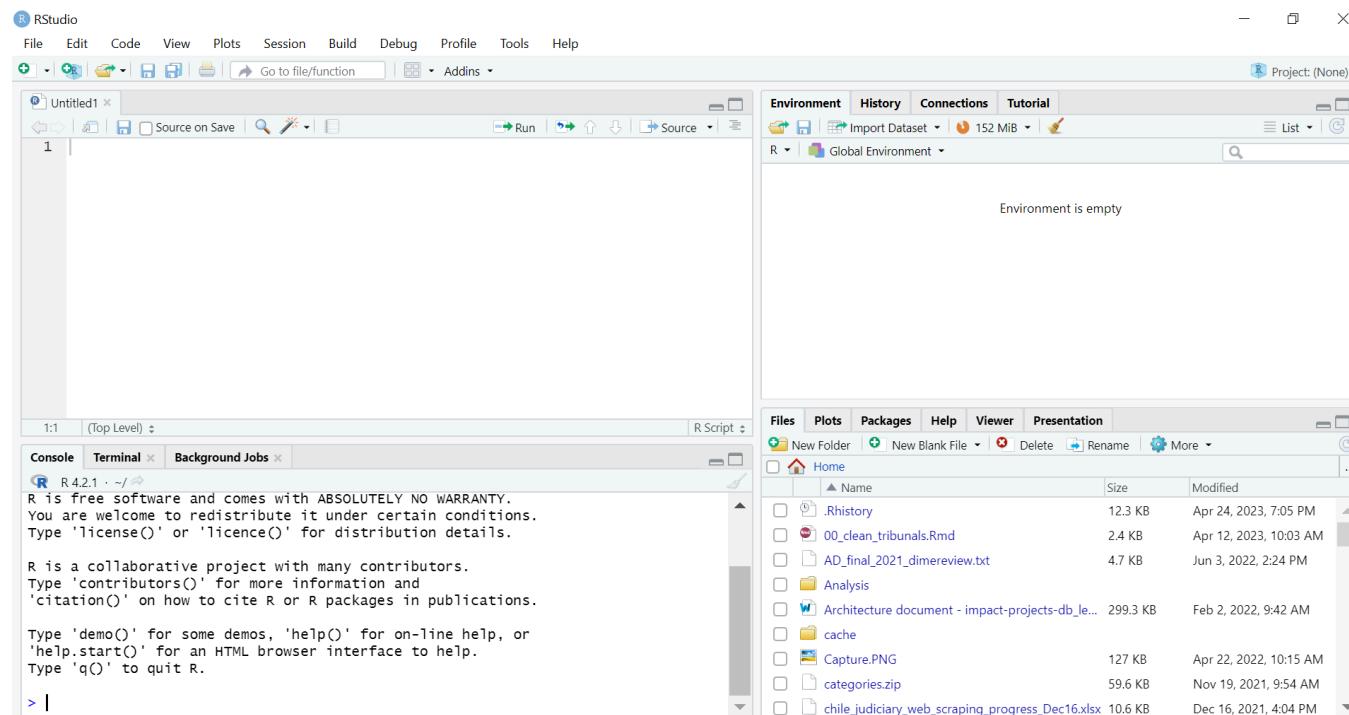
- Now open RStudio in your computer
- Please make sure you're opening RStudio and not R



Statistical Programming

How to write R code?

- Now open RStudio in your computer
- Please make sure you're opening RStudio and not R



Questions?

Writing R code

Writing R code

At its core, R can function as a (very fancy) calculator, allowing you to perform basic mathematical operations. Here are some common operators:

Mathematical Operators

- **Addition:** `+`
- **Subtraction:** `-`
- **Multiplication:** `*`
- **Division:** `/`
- **Exponentiation:** `^`

Comparison

- **Equal:** `==`
- **Not Equal:** `!=`
- **Greater than** `>`
- **Less Than** `<`

Logical Operators

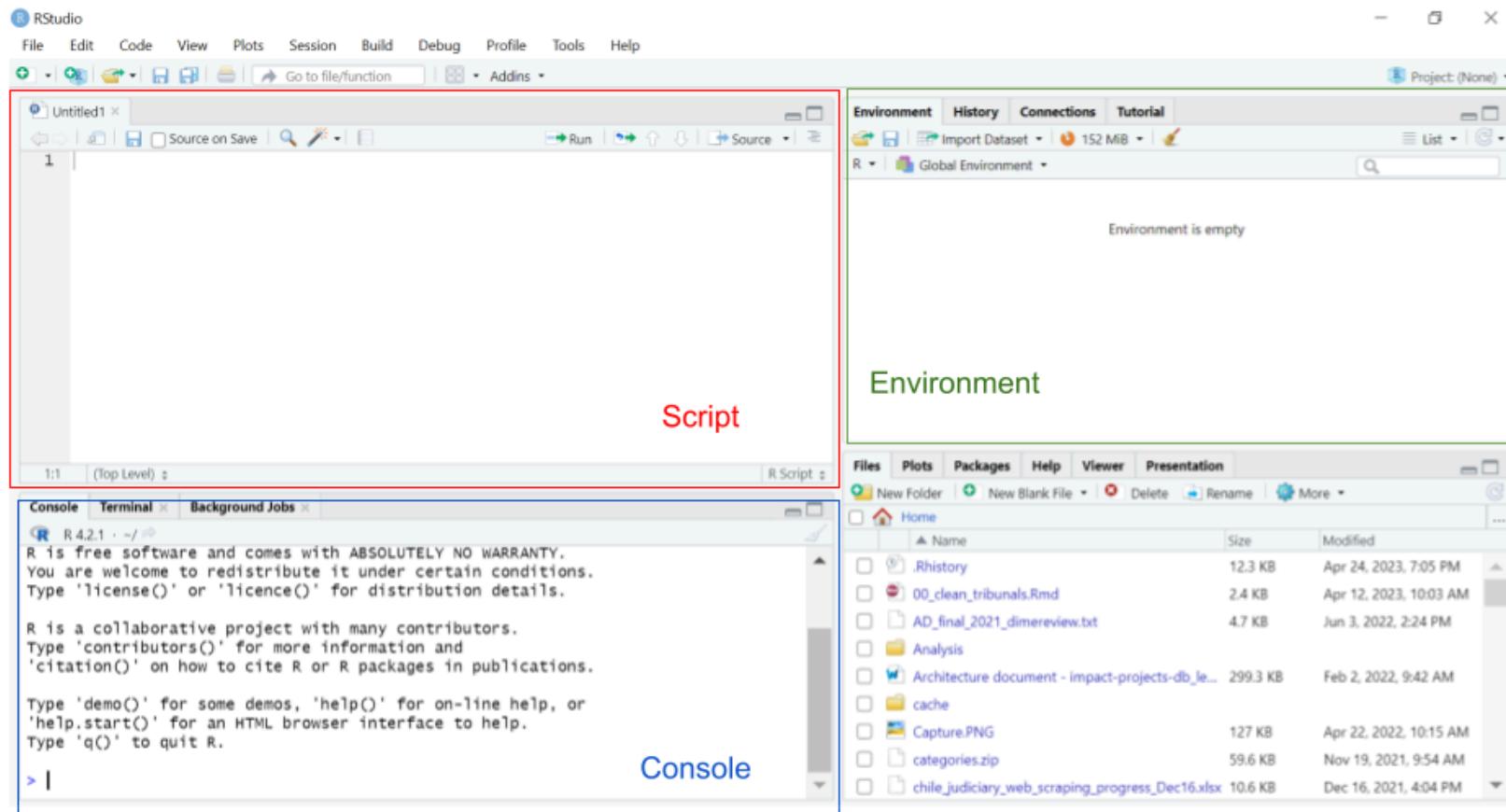
- **AND:** `&`
- **OR:** `|`

And many more ... you can see [this list](#) with a lot of examples.

Writing R code

RStudio interface

RStudio has different elements, each with a specific use.



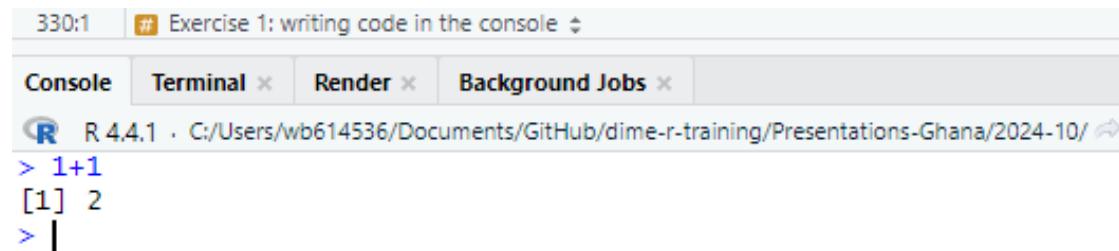
Writing R code

Exercise 1: writing code in the console

1. Write the following code in the console of RStudio

- o `1 + 1`

2. Press Enter to run the code



A screenshot of the RStudio interface showing the Console tab. The title bar says "330:1 Exercise 1: writing code in the console". The console window shows the command `> 1+1` followed by the output `[1] 2`. A cursor is visible at the end of the line.

Writing R code

R objects

- Remember we also mentioned the environment panel? that's where R keeps track of objects
- You can think of R objects as saving information, for example simple numbers or just plain text.
 - A single number can be an object
 - A word can be an object
 - Even an entire data file can be an object
- We create objects in R with the arrow operator (`<-`)
- After an object is created, we can refer to it using its name:

```
x1 <- 100 + 50
```

```
x1
```

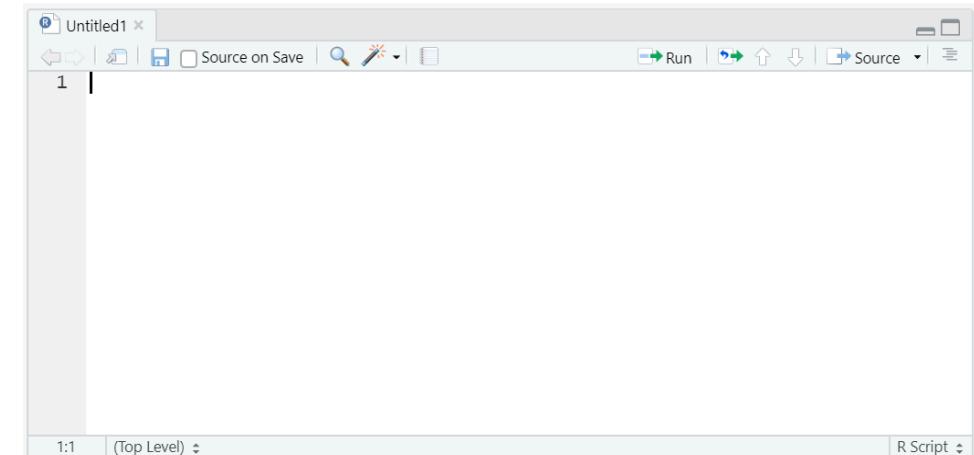
```
## [1] 150
```

Writing R code

Exercise 2: writing a short script and create objects

- 1- Write or copy the following text into the script section of RStudio

```
x1 <- 100  
x2 <- 50  
x3 <- x1 + x2  
x3
```



- 2- Select the text you introduced with your mouse

- 3- Press "Run"

Writing R code

The screenshot shows the RStudio interface. The top panel is a script editor titled "Untitled1*". It contains the following R code:

```
1 x1 <- 100
2 x2 <- 50
3 x3 <- x1 + x2
4 x3
```

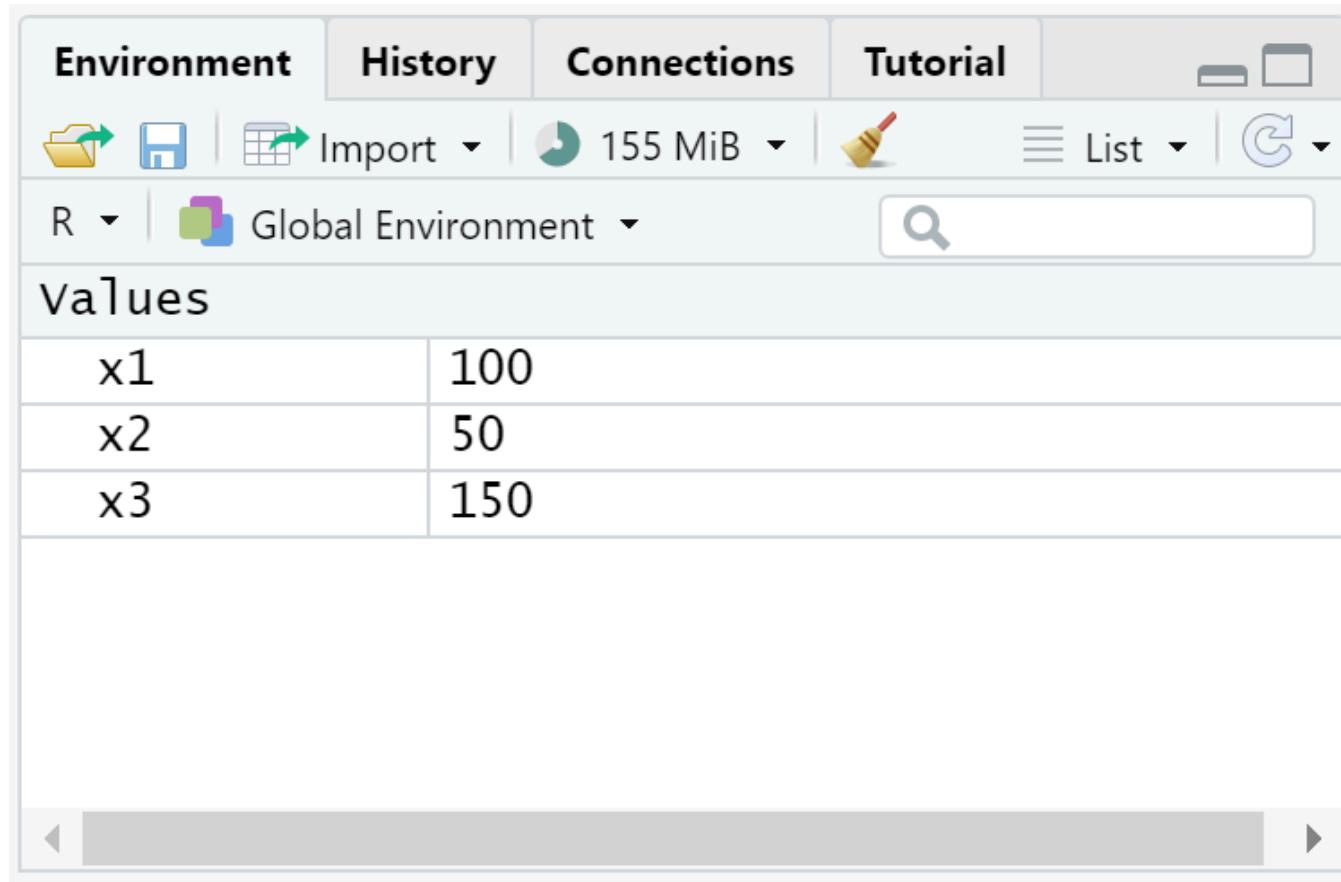
The bottom panel is a console window showing the output of the R code. The session starts with "R 4.4.1 · C:/Users/wb614536/Documents/GitHub/dime-r-training/Presentations-Ghana," followed by the executed commands and their results:

```
> x1 <- 100
> x2 <- 50
> x3 <- x1 + x2
> x3
[1] 150
>
```

Writing R code

Creating objects in R

- After any objects are created, they will show in the environment panel



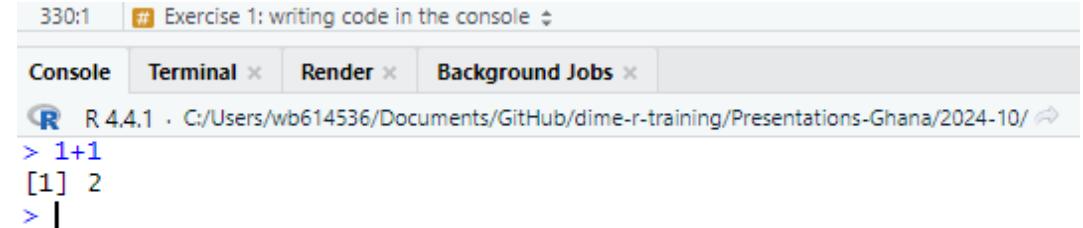
The screenshot shows the RStudio interface with the 'Environment' tab selected. The top bar includes tabs for Environment, History, Connections, and Tutorial, along with various icons for file operations like Import, Save, and Undo. A search bar is also present. The main panel displays a table titled 'values' containing three rows of data:

	values
x1	100
x2	50
x3	150

Writing R code

R scripts

- Writing and running code from the console will execute it immediately



The screenshot shows the RStudio interface with the following details:

- Title Bar:** 330:1 # Exercise 1: writing code in the console
- Tab Bar:** Console (selected), Terminal ×, Render ×, Background Jobs ×
- Environment:** R 4.4.1 · C:/Users/wb614536/Documents/GitHub/dime-r-training/Presentations-Ghana/2024-10/
- Console Output:**

```
> 1+1
[1] 2
> |
```

The console shows a simple addition operation (1+1) followed by a blank line for further input.

Writing R code

R scripts

- Writing and running code from the console will execute it immediately
- Writing code in the script panel allow us to write multiple lines of code and execute them later
 - Each line is executed in order
 - The line and the results will show in the console
- **Important:** for the rest of the training, remember to always introduce your code in the script (and not in the console) so you can keep record of what you did

The screenshot shows the RStudio interface. The top panel displays a script file titled "Untitled1" with the following code:

```
1 x1 <- 100
2 x2 <- 50
3 x3 <- x1 + x2
4 x3
```

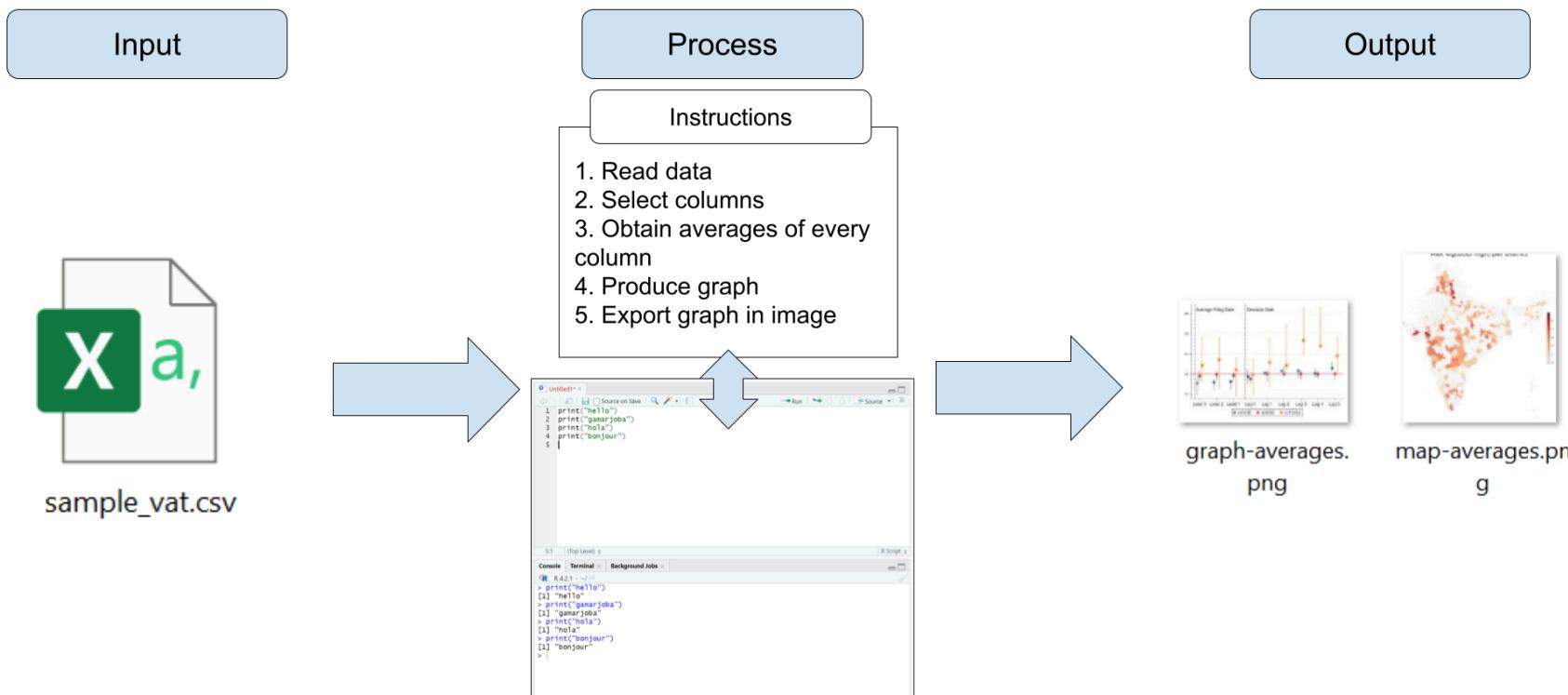
The bottom panel shows the "Console" tab with the following output:

```
R 4.4.1 - C:/Users/wb614536/Documents/GitHub/dime-r-training/Presentations-Ghana,
> x1 <- 100
> x2 <- 50
> x3 <- x1 + x2
> x3
[1] 150
>
```

Writing R code

R scripts

- In other words: scripts contain the instructions you give to your computer when doing data work



Object Types

Object Types

What Are Object Types?

- **R objects** are containers for data, and they come in different **types**.
- The type of an object determines:
 - What **operations** can be performed on it.
 - How R will treat it during analysis.

Examples of common object types:

1. **Numeric**: Used for numbers, such as `100` or `3.14`. Can be used for mathematical operations like addition, subtraction, etc.
2. **Character**: Represents text or words, like `"Hello"` or `"Region A"`.
3. **Vector**: A collection of values stored in a single dimension, like a list or a column in Excel. Example: A list of ages: `c(25, 30, 35, 40)`.
4. **Dataframe**: A collection of rows and columns, similar to a table or spreadsheet. Each column can have its own type (e.g., numeric, character).

Object Types

Let's Explore Object Types

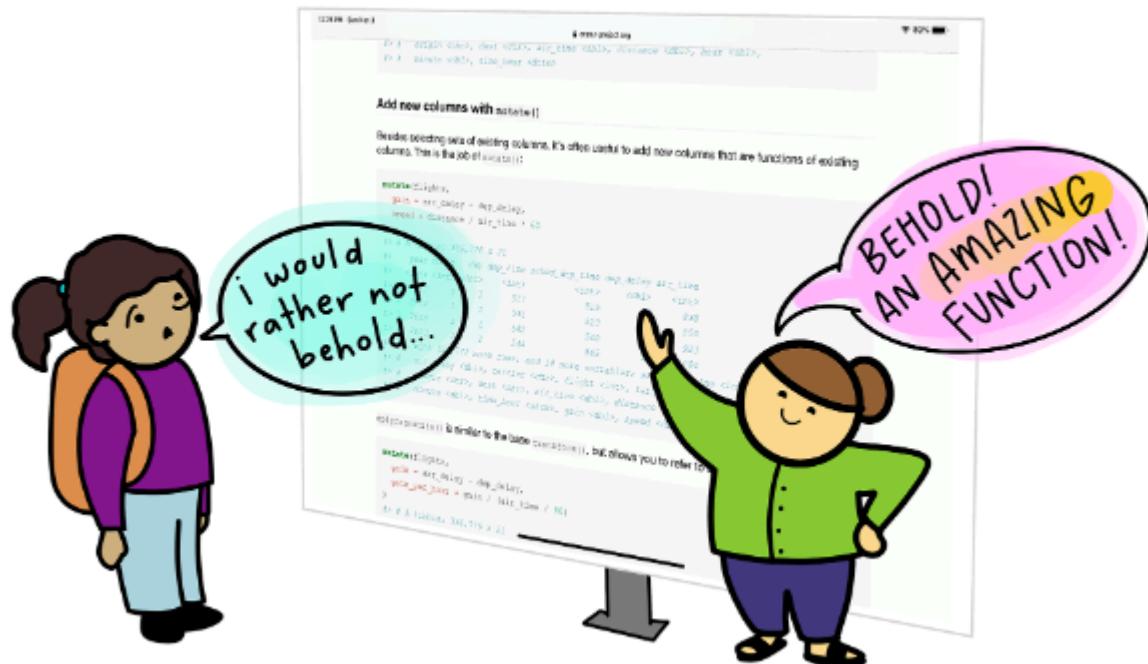
Create objects in RStudio:

```
num_object <- 42          # Numeric  
  
char_object <- "Learning R"    # Character  
  
vector_object <- c(1, 2, 3, 4) # Vector  
  
df_object <- data.frame(  
  Name = c("Alice", "Bob"),    # Dataframe  
  Age = c(25, 30)  
)
```

We will discuss more about the last type, but first we need to introduce one last concept.

Functions in R

Functions in R



Functions in R

- Functions are how we apply operations to objects in R
- You can think of functions as little machines that (in most cases) process some kind of **input** and create an **output**.
- Input is everything that goes into a function (arguments and values)

Let's take a look at an example: the star producer!

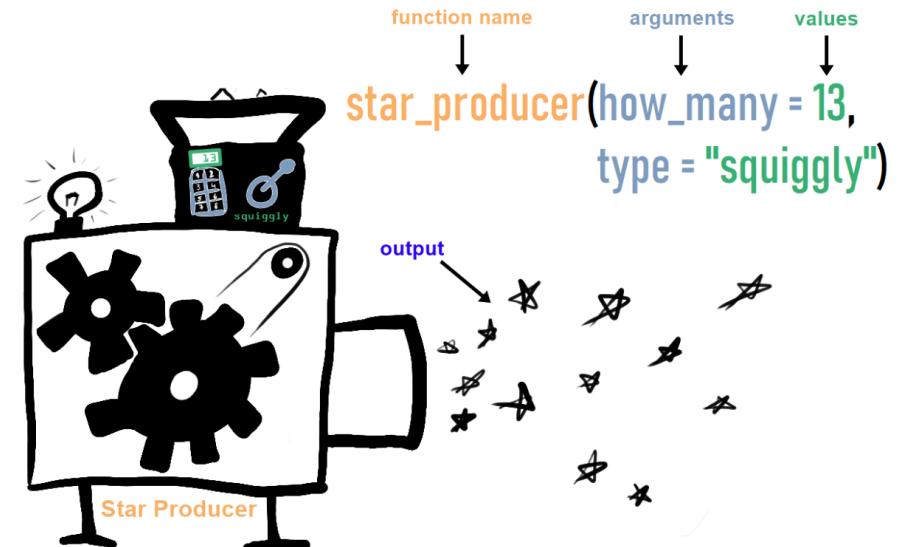
Functions in R

The Star Producer

Let's consider the following function (that does not exist unfortunately): **A star_producer!**

This little machine creates tiny hand-drawn stars depending on some input. It takes two arguments:

- `how_many` tells the machine how many stars to produce.
- `type` tells the machine how the stars should look like (in this case the machine only supports "squiggly" stars, but it could be upgraded in the future when we learn how to create our own functions later on).



Functions in R

Getting ?help

How do we know what function takes what kind of arguments?

Within R, you can always run the code:

```
?function_name
```

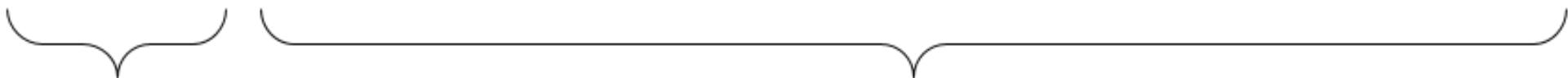
This will open up the documentation for the function, which explains how to use it.

Googling the function name (adding "R" or "rstats" to the search) will also often bring up relevant documentation or examples!

Functions in R

Now in a real function ...

```
subset(small_business_2019, region == "Guria")
```



function name arguments

- **Function name:** the name we use to call a function. It goes before the parentheses
- **Arguments:** inputs and specifications for the function to be applied.
 - Arguments go inside the parentheses
 - The first argument is the object you apply the function on

Functions in R

- The results of a function can always be stored in an object with the arrow operator (`<-`)

```
df_guria <- subset(small_business_2019, region == "Guria")
```

- The results of a function will only be printed in the console if you don't store them

Functions in R

Okay now let's try it!

Exercise 3: Using a function `sum()`

1. Compute the sum of the numbers 1 to 10 and store it in the object `sum_example`:

```
sum_example <- sum(c(1:10))
```

1. Print the stored result with `print(sum_example)`

```
print(sum_example)
```

```
## [1] 55
```

Functions in R

Note that this code is both creating a new object (with `sum_example <- sum(c(1:10))`) and printing the result in the console (with `print(sum_example)`)

The screenshot shows the RStudio interface. In the top-left, there are three tabs: 'Untitled1*', '1-introduction-to-r.Rmd*', and 'exercises-session1.R*'. The 'exercises-session1.R*' tab is active, displaying the following R code:

```
6  
7 # Exercise 3  
8  
9 sum_exercise <- sum(c(x <- 1:10))  
10 print(sum_exercise)  
11  
12
```

A yellow hand cursor is positioned over the line `sum_exercise <- sum(c(x <- 1:10))`. The 'Console' tab is selected at the bottom left, showing the output of the code:

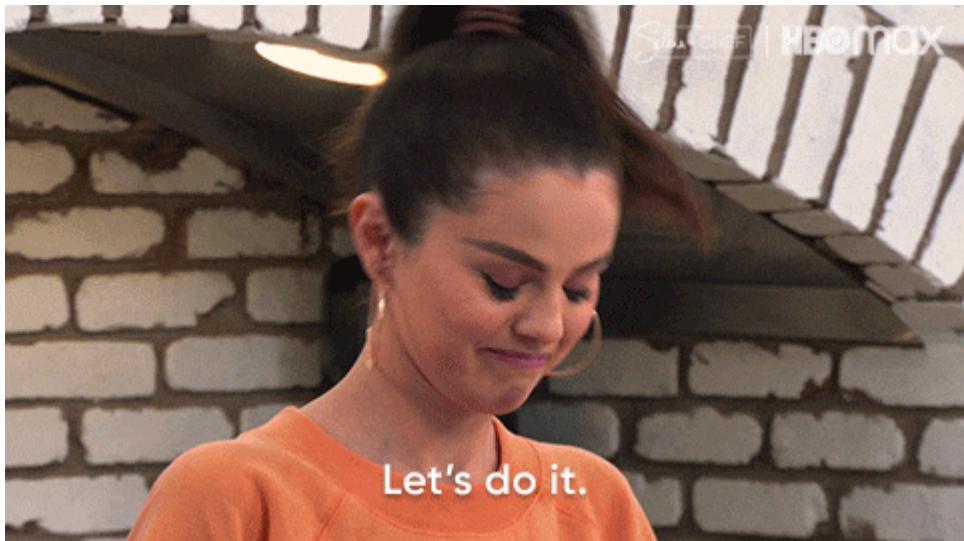
```
R 4.4.1 - C:/Users/wb614536/Documents/GitHub/dime-r-training/Presentations-Ghana/2024-10/  
> sum_exercise <- sum(c(x <- 1:10))  
> print(sum_exercise)  
[1] 55  
>
```

A yellow highlight covers the entire code block from line 9 to the final closing brace of line 12. To the right is the 'Global Environment' pane, which lists the variable `sum_exercise` with the value `55L`. Below it is the 'Files' pane, showing a directory structure:

Name	Size	Modified
..		
.Rhistory	794 B	Nov 12,
1-introduction-to-r_cache		
1-introduction-to-r.html	29.7 KB	Nov 15,
1-introduction-to-r.pdf	4 MB	Nov 6, 2
1-introduction-to-r.Rmd	24.1 KB	Nov 15,
2-data-wrangling_cache		

Writing R code

- Now we know how to use RStudio to write R code and produce scripts.
- We also know about objects and functions.
- We haven't still introduced the data to our data work. That comes next



Data in R

Data in R

Exercise 4: Loading data into R

1.- Go to this page: <https://osf.io/2apht> and download the file **department_staff_list.xlsx**

The screenshot shows the OSF interface with the following details:

- Header:** OSF HOME ▾, My Projects, Search, Support, Donate, DIME Analytics ▾.
- Project Title:** R training Georgia RS - WB September 2023
- File Name:** small_business_2019.csv
- Sheet View:** Sheet 1 displays a table with columns: modified_id, region, and income. The data includes rows for Kaxeti, Tbilisi, Guria, and Samegrelo-Z. SvaneTi.
- File Metadata:** Includes a pencil icon for edit, a download icon, and a delete icon. It also says "Press the edit button to add metadata to this file."
- Project Metadata:** Includes fields for Title (R training Georgia RS - WB September 2023), Date created (September 19, 2023), Date modified (September 19, 2023), and Contributors (DIME Analytics).

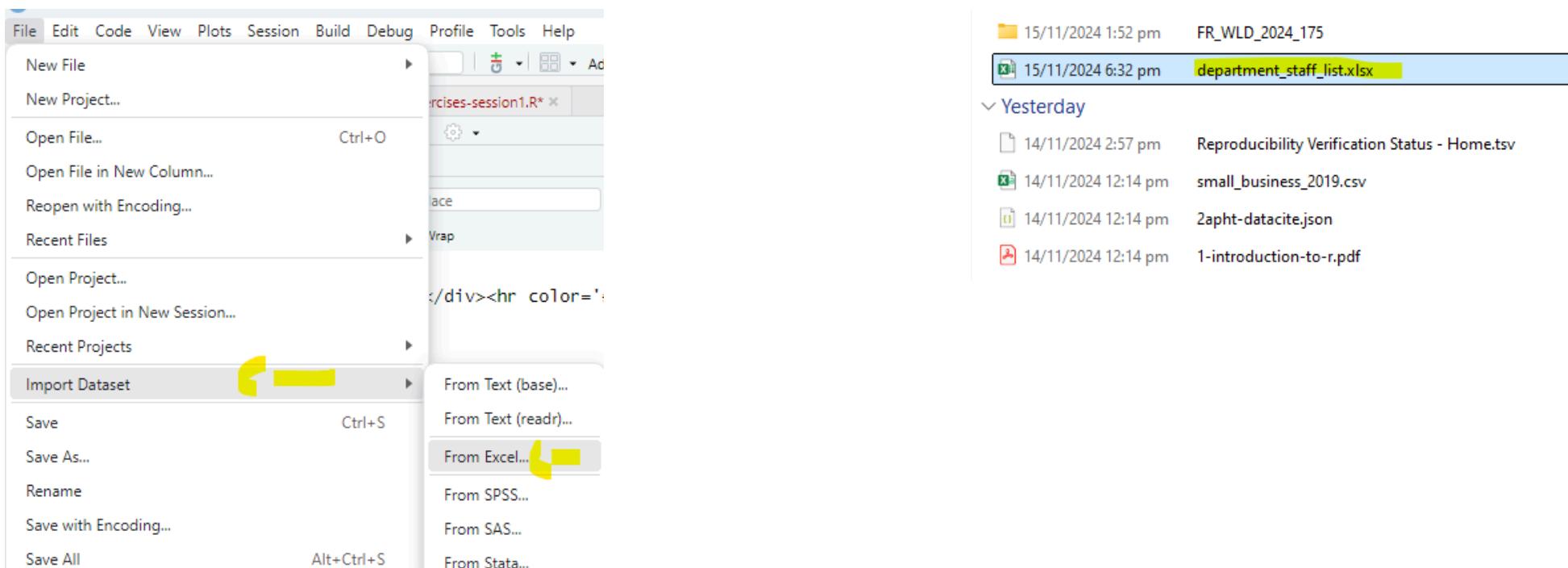
Data in R

Exercise 4: Loading data into R

There are different ways of importing data to R, one is using the point and click. Let's start with that one.

2.- In RStudio, go to **File** > **Import Dataset** > **From Excel** and select the file **department_staff_list.xlsx**

- If you don't know where the file is, check in your **Downloads** folder



Data in R

Exercise 4: Loading data into R

3 - Make sure to select **Heading** > **Yes** in the next window

4 - Select **Import**

5 - You will see that the second way to read it by code (using functions), and is what R is doing for you in the background.

Import Excel Data

File/URL:
C:/Users/wb614536/Downloads/department_staff_list.xlsx

1

3

Previewing first 50 entries.

sex	current_grade	date_of_birth	date_of_first_appointment	senior_junior_staff	department	years_of_service
Female	Director Fin. & Admin.	1964-09-25	1994-08-31	senior	State Protocol Dept	30.209446
Male	Deputy Director (Admin.)	1983-02-19	2008-11-01	senior	State Protocol Dept	16.038330
Female	Chief Exe. Officer	1965-06-15	1990-12-01	senior	State Protocol Dept	33.957563
Female	Senior Records Officer	1974-04-29	2013-05-01	senior	State Protocol Dept	11.542779
Female	Senior Procurement/supply Chain Officer	1973-08-08	2003-09-15	senior	State Protocol Dept	21.169062
Female	Sr. Private Secretary	1979-08-29	2002-01-07	senior	State Protocol Dept	22.855578
Female	Private Secretary	1982-12-26	2003-03-12	senior	State Protocol Dept	21.681040
Male	Computer Operator	1968-04-27	2002-04-02	junior	State Protocol Dept	22.622861

Import Options:

Name: department_staff_list
Sheet: Default
Range: A1:D10
Max Rows: 1000
Skip: 0
NA:
First Row as Names
Open Data Viewer

Code Preview:

```
library(readxl)  
department_staff_list <- read_excel("C:/Users/wb614536/  
/Downloads/department_staff_list.xlsx")  
View(department_staff_list)
```

Import Cancel

Data in R

- If you did this correctly, you will note that a viewer of the data now appears in RStudio
- You will also note that this appeared as a new object in your environment

The screenshot shows the RStudio interface with two main panes. The left pane is a data viewer titled "department_staff_list" containing a table of staff information. The right pane is the "Environment" tab of the global environment, listing various objects and their values.

Data Viewer (left pane):

	sex	current_grade
1	Female	Director Fin. & Admin.
2	Male	Deputy Director (Admin.)
3	Female	Chief Exe. Officer
4	Female	Senior Records Officer
5	Female	Senior Procurement/supply Chain Officer
6	Female	Snr. Private Secretary
7	Female	Private Secretary
8	Male	Computer Operator
9	Male	Asst. Director IIA
10	Female	Private Secretary
11	Male	IT/IM Officer

Environment Tab (right pane):

Object	Type
department_staff_list	8754 obs.
values	
sum_exercise	55L
x	int [1:10]
x1	100
x2	50
x3	150

Data in R

- Remember we mentioned **dataframe** objects before? For R, `department_staff_list` is an object just like `x1`, `x2`, or `x3`
- The difference is that `department_staff_list` is not a single number like `x1`, but a collection of numeric values similar to an Excel spreadsheet. In R, this type of objects are called **dataframes**
- From now, we will refer to data loaded into R as **dataframes**

The screenshot shows the RStudio interface with the 'Environment' tab selected. In the Global Environment pane, there is one dataset named 'small_business_2...'. This dataset has 984 observations and 3 variables. Below the dataset, a table shows the values for variables x1, x2, and x3:

values	
x1	100
x2	50
x3	150

Data in R

- Since dataframes are also objects, we can refer to them with their names (exm: `department_staff_list.xlsx`)
- We'll see an example of that in the next exercise

Data in R

A note about this dataframe

- Understanding the data you use is very important. For this training, `department_staff_list` is a dataframe from your department.
- Let's use another function to see what's in there

```
glimpse(department_staff_list)
```

```
## Rows: 8,754
## Columns: 7
## $ sex                  <chr> "Female", "Male", "Female", "Female", "Femal...
## $ current_grade        <chr> "Director Fin. & Admin.", "Deputy Director ...
## $ date_of_birth         <dttm> 1964-09-25, 1983-02-19, 1965-06-15, 1974-04...
## $ date_of_first_appointment <dttm> 1994-08-31, 2008-11-01, 1990-12-01, 2013-05...
## $ senior_junior_staff   <chr> "senior", "senior", "senior", "senior", "sen...
## $ department            <chr> "State Protocol Dept", "State Protocol Dept"...
## $ years_of_service       <dbl> 30.209446, 16.038330, 33.957563, 11.542779, ...
```

Data in R

Exercise 5: Subset the data

1. Use the following code to subset `department_staff_list` and leave only the observations who are "Female":

```
df_female <- subset(department_staff_list, sex == "Female")
```

- Note that we are using the arrow operator (`<-`) to store the result
- Note that there are **two equal signs** in the condition, not one
- Also note that you need to write `"Female"` enclosed in quotes and with uppercase `F`, because that's how it is in the data

1. Use `View(df_female)` to visualize the dataframe again and see how it changed (note the uppercase "V")

Data in R

The screenshot shows the RStudio interface with two panes displaying data frames.

Left Pane: Shows the `df_female` data frame.

	sex	current_grade
1	Female	Director Fin. & Admin.
2	Female	Chief Exe. Officer
3	Female	Senior Records Officer
4	Female	Senior Procurement/supply Chain Officer
5	Female	Snr. Private Secretary
6	Female	Private Secretary
7	Female	Private Secretary
8	Female	Stenographer Gd I
9	Female	Records Officer
10	Female	Principal Exe. Officer
11	Female	Steno Secretary
12	Female	senior Technical Asst.
13	Female	Principal Protocol Officer

Right Pane: Shows the `Environment` tab with the following variables listed:

Value	Type	Length
department_staff_list	8754 obs. of	
df_female	3572 obs. of	
sum_exercise	55L	
x	int [1:10] 1	
x1	100	
x2	50	
x3	150	

Below the environment pane, the `Files` tab is selected, showing the file structure:

- New Folder
- New Blank File
- Delete
- Rename

The current working directory is displayed as:

```
C: > Users > wb614536 > Documents > GitHub > dime-r-training
```

At the bottom, there is a search bar labeled "Name".

Data in R

Storing results in R

There is an important difference between using `<-` and not using it

- Not using `<-` **simply displays the result** in the console. The input dataframe will remain unchanged and the result **will not be stored**

```
subset(department_staff_list, sex == "Female")
```

```
## # A tibble: 3,572 × 7
##   sex    current_grade      date_of_birth      date_of_first_appoin...1
##   <chr>  <chr>            <dttm>           <dttm>
## 1 Female Director Fin. & Admin. 1964-09-25 00:00:00 1994-08-31 00:00:00
## 2 Female Chief Exe. Officer    1965-06-15 00:00:00 1990-12-01 00:00:00
## 3 Female Senior Records Officer 1974-04-29 00:00:00 2013-05-01 00:00:00
## 4 Female Senior Procurement/supply ... 1973-08-08 00:00:00 2003-09-15 00:00:00
## 5 Female Snr. Private Secretary 1979-08-29 00:00:00 2002-01-07 00:00:00
## 6 Female Private Secretary     1982-12-26 00:00:00 2003-03-12 00:00:00
## 7 Female Private Secretary     1986-02-13 00:00:00 2019-10-30 00:00:00
## 8 Female Stenographer Gd I     1986-11-14 00:00:00 2008-08-20 00:00:00
## 9 Female Records Officer       1981-06-17 00:00:00 2018-06-16 00:00:00
```

Storing results in R

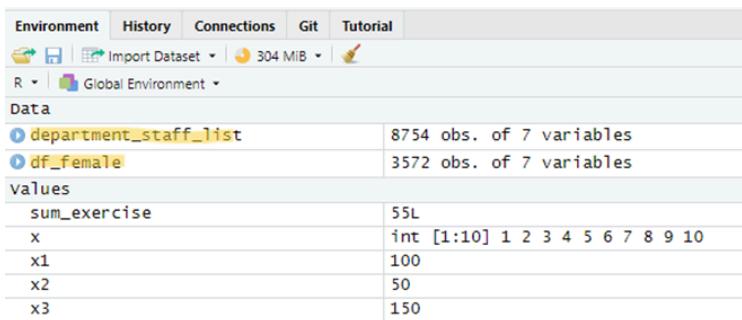
- Using `<-` tells R that we want to **store the result in a new object**, which is the object at the left side of the arrow. This time the result will not be printed in the console but the new dataframe will show in the environment panel

```
## Warning in read_fun(path = path, sheet_i = sheet, limits = limits, shim = shim,  
## : NA inserted for an unsupported date prior to 1900
```

```
df_female <- subset(department_staff_list, sex == "Female")
```

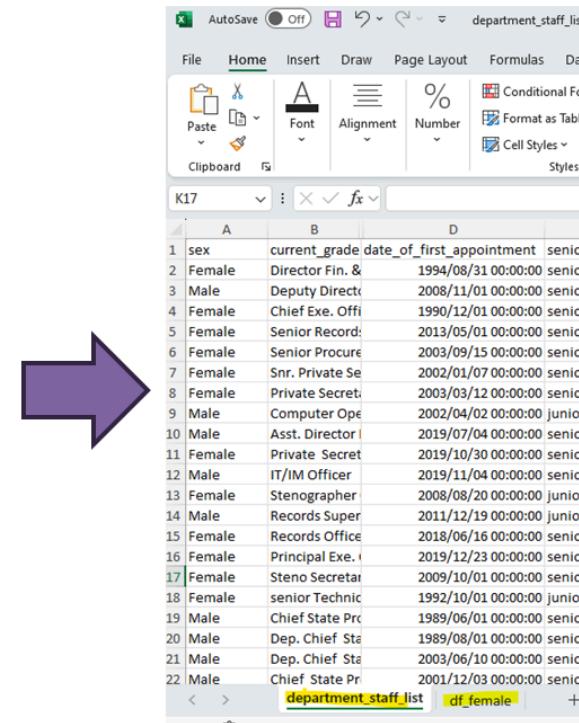
Data in R

- R can store multiple dataframes in the environment. This is analogous to having different spreadsheets in the same Excel window
- Always remember that dataframes are just objects in R. R differentiates which dataframe the code refers to with the dataframe name



Screenshot of the RStudio Environment tab. It shows the following objects:

- department_staff_list: 8754 obs. of 7 variables
- df_female: 3572 obs. of 7 variables
- values
- sum_exercise: 55L
- x: int [1:10] 1 2 3 4 5 6 7 8 9 10
- x1: 100
- x2: 50
- x3: 150



Screenshot of Microsoft Excel showing a spreadsheet titled "department_staff_list". The data consists of 22 rows of staff information:

	A	B	C	D
1	sex	current_grade	date_of_first_appointment	senio
2	Female	Director Fin. &	1994/08/31 00:00:00	senio
3	Male	Deputy Direct	2008/11/01 00:00:00	senio
4	Female	Chief Exec. Offi	1990/12/01 00:00:00	senio
5	Female	Senior Record:	2013/05/01 00:00:00	senio
6	Female	Senior Procure	2003/09/15 00:00:00	senio
7	Female	Snr. Private Se	2002/01/07 00:00:00	senio
8	Female	Private Secret	2003/03/12 00:00:00	senio
9	Male	Computer Oper	2002/04/02 00:00:00	junior
10	Male	Asst. Director I	2019/07/04 00:00:00	senio
11	Female	Private Secret	2019/10/30 00:00:00	senio
12	Male	IT/IM Officer	2019/11/04 00:00:00	senio
13	Female	Stenographer	2008/08/20 00:00:00	junior
14	Male	Records Super	2011/12/19 00:00:00	junior
15	Female	Records Office	2018/06/16 00:00:00	senio
16	Female	Principal Exec. I	2019/12/23 00:00:00	senio
17	Female	Steno Secretar	2009/10/01 00:00:00	senio
18	Female	senior Technic	1992/10/01 00:00:00	junior
19	Male	Chief State Pro	1989/06/01 00:00:00	senio
20	Male	Dep. Chief Sta	1989/08/01 00:00:00	senio
21	Male	Dep. Chief Sta	2003/06/10 00:00:00	senio
22	Male	Chief State Pr	2001/12/03 00:00:00	senio

Questions?

Wrapping up

Wrapping up

Add code comments!

- Every line of code that starts with the pound symbol (#) will be ignored when R executes the code
- This means that you can add any clarifying comment with #. These are called **code comments**
- It's always a good practice to add code comments for yourself to later remember what the code is doing or to explain your code to others if you'll share it

Wrapping up

- Try adding code comments to your script so you will remember which part corresponds to each exercise

The screenshot shows an RStudio interface with a script editor containing the following R code:

```
# Exercise 2
x1 <- 100
x2 <- 50
x3 <- x1 + x2
x3

# Exercise 3
sum_exercise <- sum(c(x <- 1:10))
print(sum_exercise)

# Exercise 4 Load data into R

# Exercise 5
df_female <- subset(department_staff_list,
                      sex == "Female")
```

The code is organized into four distinct sections, each preceded by a comment starting with '# Exercise'. The first section (# Exercise 2) contains variable assignments and a print statement. The second section (# Exercise 3) calculates the sum of integers from 1 to 10 and prints the result. The third section (# Exercise 4) is a placeholder for loading data into R. The fourth section (# Exercise 5) uses the subset function to filter a data frame for female staff.

Wrapping up

Always save your work!

- Click the floppy disk icon to save your work
- Select a location for your file and remember where you're saving it

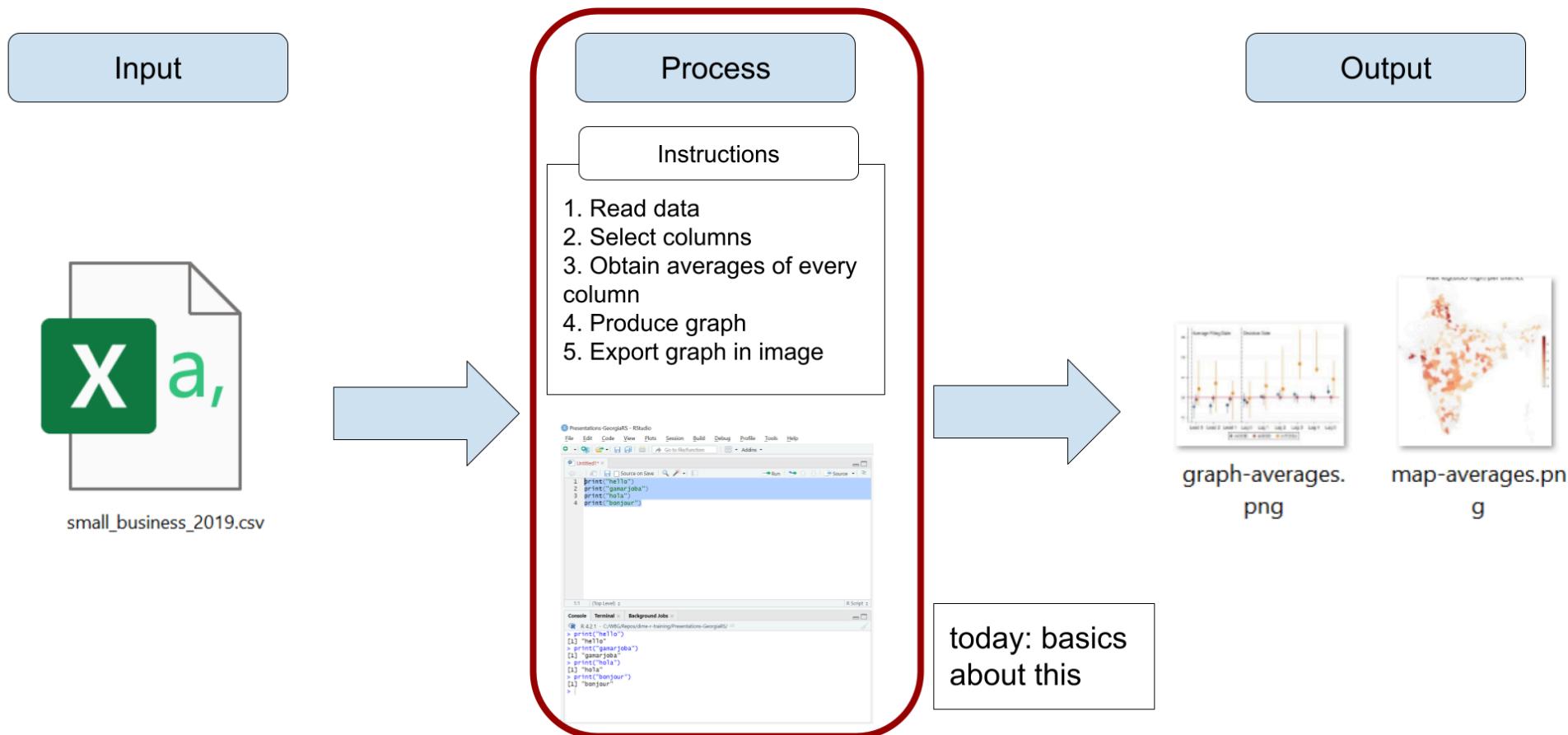


```
1 # Exercise 2
2 x1 <- 100
3 x2 <- 50
4 x3 <- x1 + x2
5 x3
6
7 # Exercise 3
8
9 sum_exercise <- sum(c(x <- 1:10))
10 print(sum_exercise)
11
12 # Exercise 4 Load data into R
13
```

Wrapping up

This session

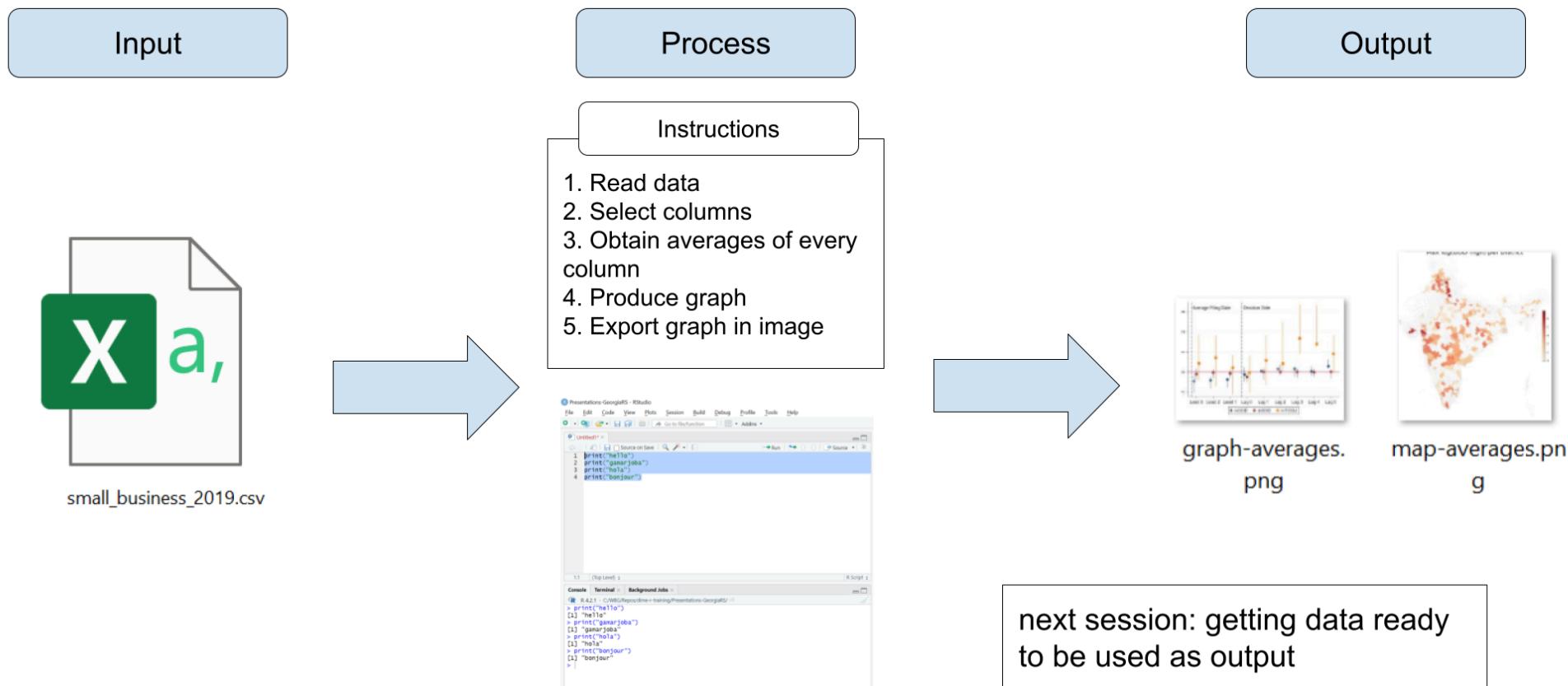
This first session focused on the basics for writing R code



Wrapping up

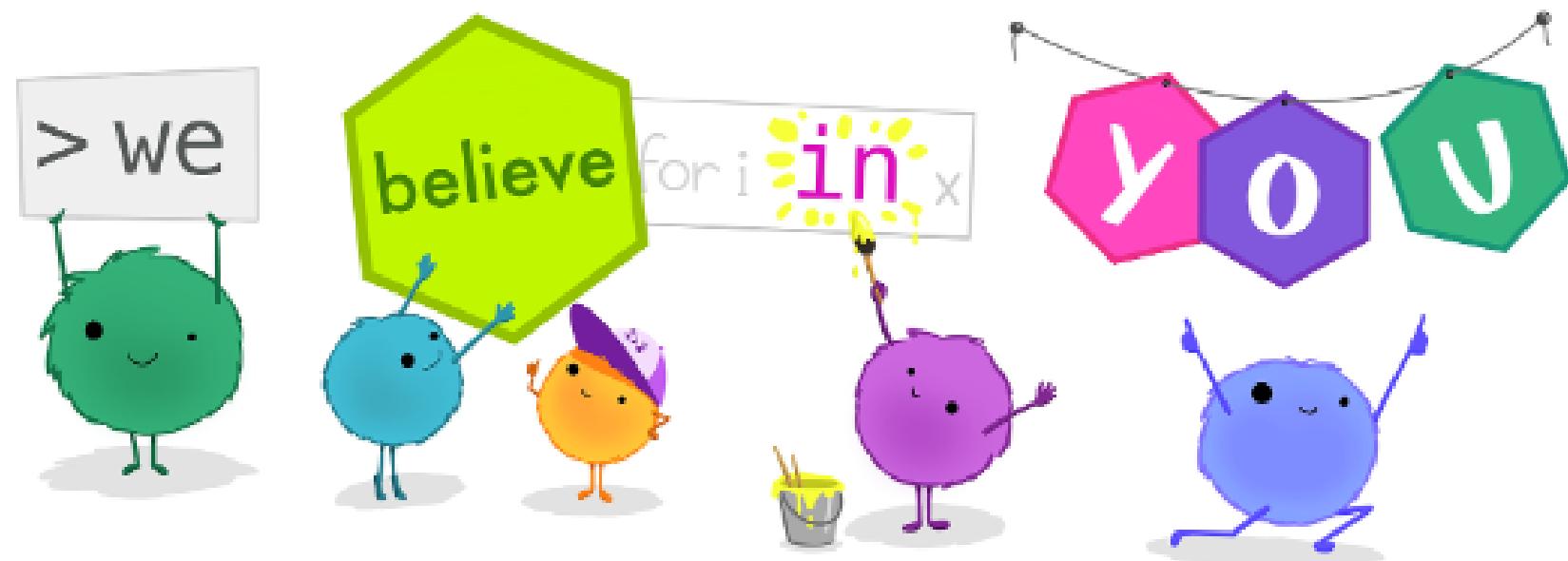
Next session

In the next session we will learn how to get data ready to be exported as outputs



Thanks! // ¡Gracias! // Obrigado!

R learners,



@allisonhorst

Appendix

Appendix

Object Types: character strings

- Character strings are collections of alphanumeric characters usually representing words or texts, or just characters in general

```
s1 <- "Hello World"  
print(s1)
```

```
## [1] "Hello World"
```

- Strings characters are **always enclosed in quotes** (" ")
- They are usually referred to as just **strings**

Appendix

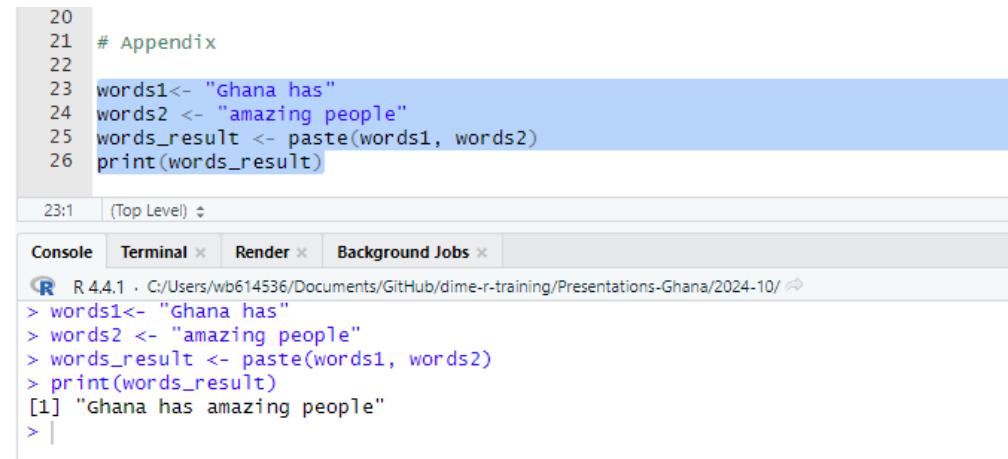
Exercise: create and operate character strings

1. Create a character string object with the words "Ghana has" and name it `words1`
2. Create a second string with the words "amazing people" and name it `words2`
 - Don't forget to use `<-` to create the string objects
 - Remember to include the quotes: `" "`
3. Use the following code to concatenate `words1` and `words2`, save the result in `words_result`, and print it:

```
words_result <- paste(words1, words2)
print(words_result)
```

Appendix

Object Types: character strings



The screenshot shows an RStudio interface. The code editor at the top contains the following R code:

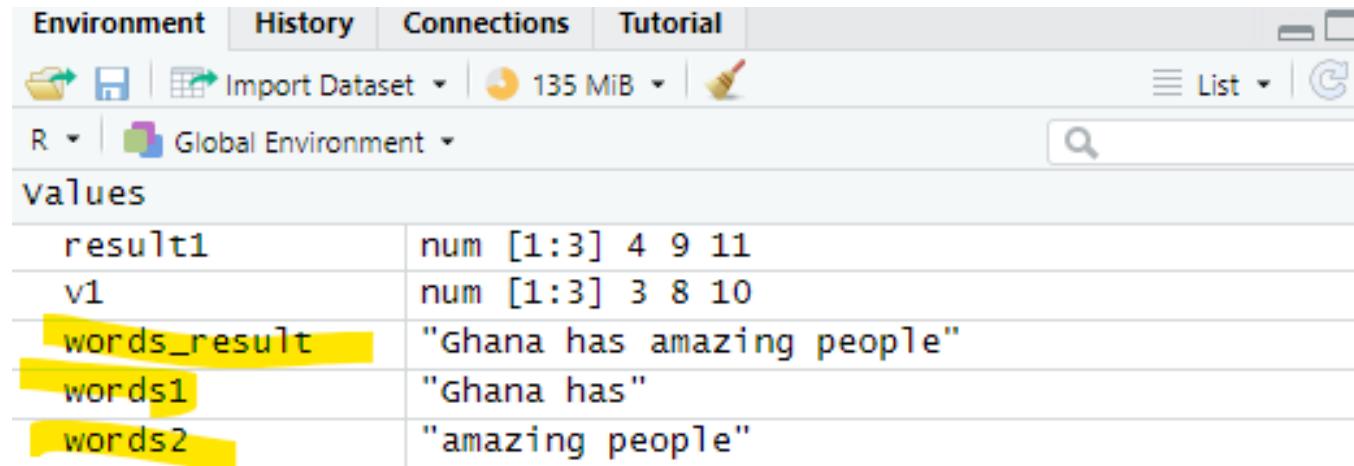
```
20
21 # Appendix
22
23 words1<- "Ghana has"
24 words2 <- "amazing people"
25 words_result <- paste(words1, words2)
26 print(words_result)
```

The code is run in the console tab, which displays the output:

```
23:1 (Top Level) ▾
Console Terminal × Render × Background Jobs ×
R 4.4.1 · C:/Users/wb614536/Documents/GitHub/dime-r-training/Presentations-Ghana/2024-10/ ↵
> words1<- "Ghana has"
> words2 <- "amazing people"
> words_result <- paste(words1, words2)
> print(words_result)
[1] "Ghana has amazing people"
> |
```

Appendix

Object Types: character strings



The screenshot shows the RStudio interface with the 'Environment' tab selected. The global environment contains the following objects:

values	
result1	num [1:3] 4 9 11
v1	num [1:3] 3 8 10
words_result	"Ghana has amazing people"
words1	"Ghana has"
words2	"amazing people"

Appendix

Object Types: Vectors

- Vectors are a collection of values **with a single dimension**, instead of being organized in rows and columns as dataframes
- You can think of a vector in R as a single column in an Excel spreadsheet or an R dataframe
- You can create vectors with the function `c()`, the vector elements are separated by commas

```
my_vector <- c(4, 8, 2, 5)
```

Appendix

Exercise: Create vectors

1- Create a vector with the elements 3, 8, and 10 and name it `v1`:

```
v1 <- c(3, 8, 10)
```

2- create a vector named `result1` with the sum of `v1` + 1.

```
result1 <- v1+1
```

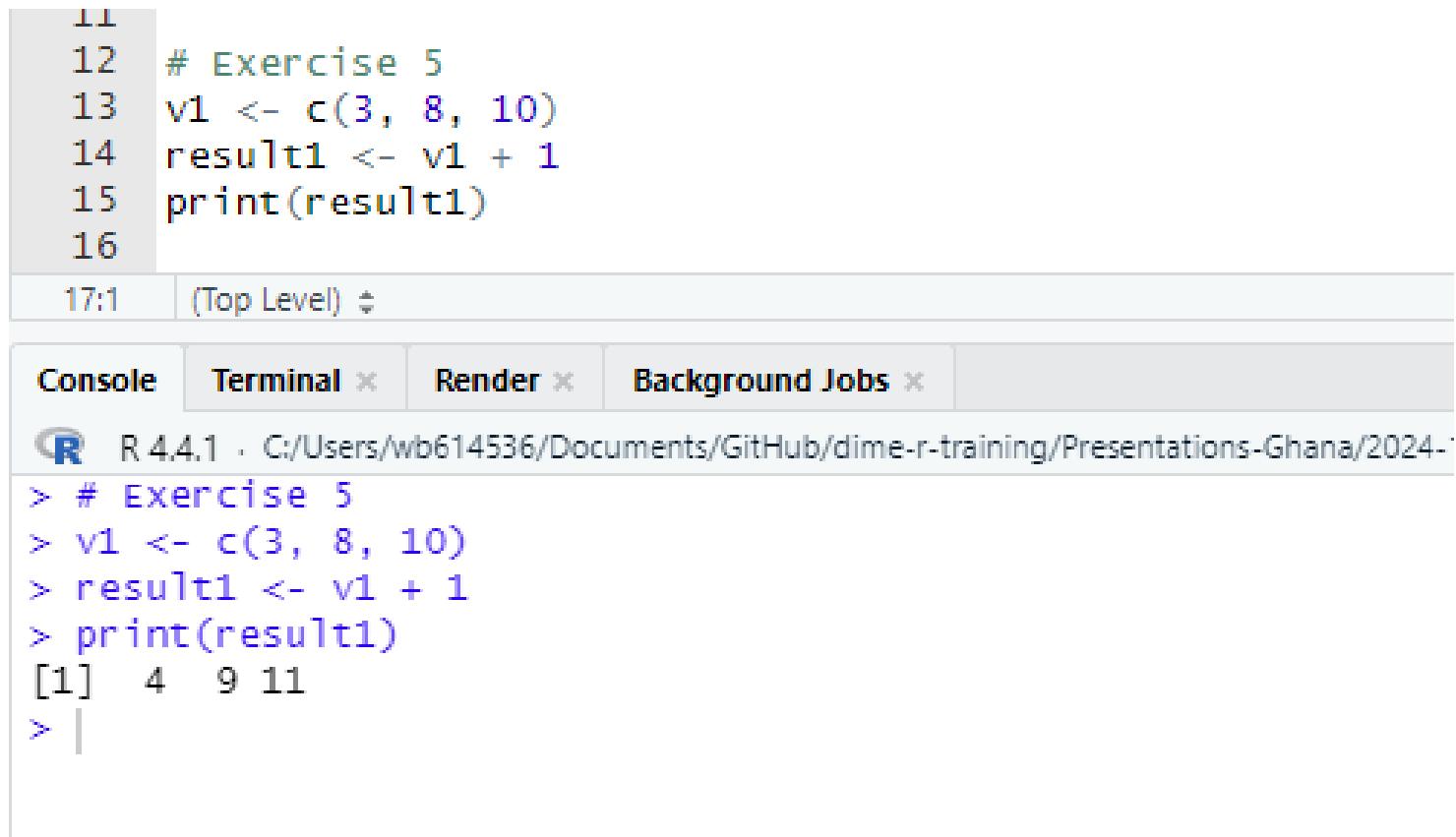
3- Print `v1` and observe the results

```
print(result1)
```

```
## [1] 4 9 11
```

Appendix

Exercise: Create vectors

```
11  
12 # Exercise 5  
13 v1 <- c(3, 8, 10)  
14 result1 <- v1 + 1  
15 print(result1)  
16  
17:1 | (Top Level)   
  


The screenshot shows the RStudio interface. The code editor at the top contains the following R code:



```
> # Exercise 5
> v1 <- c(3, 8, 10)
> result1 <- v1 + 1
> print(result1)
```



The console tab is selected, showing the output:



```
[1] 4 9 11
```

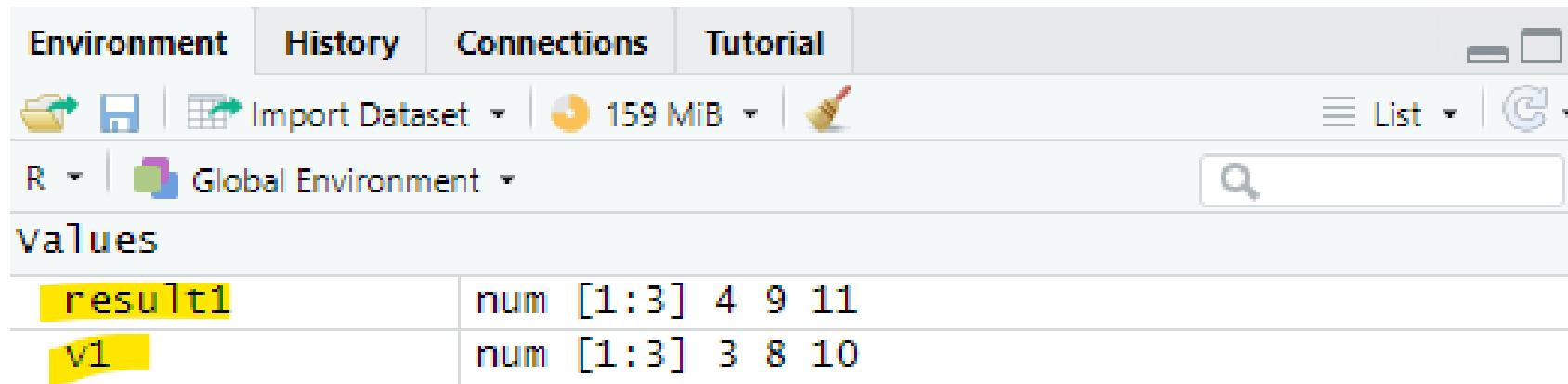


The R logo icon is visible in the top left of the interface.


```

Appendix

Exercise: Create vectors



The screenshot shows the RStudio interface with the 'Environment' tab selected. The global environment contains two vectors:

values	
result1	num [1:3] 4 9 11
v1	num [1:3] 3 8 10