Session 2 - Data wrangling

R training - Georgia RS-WB DIME

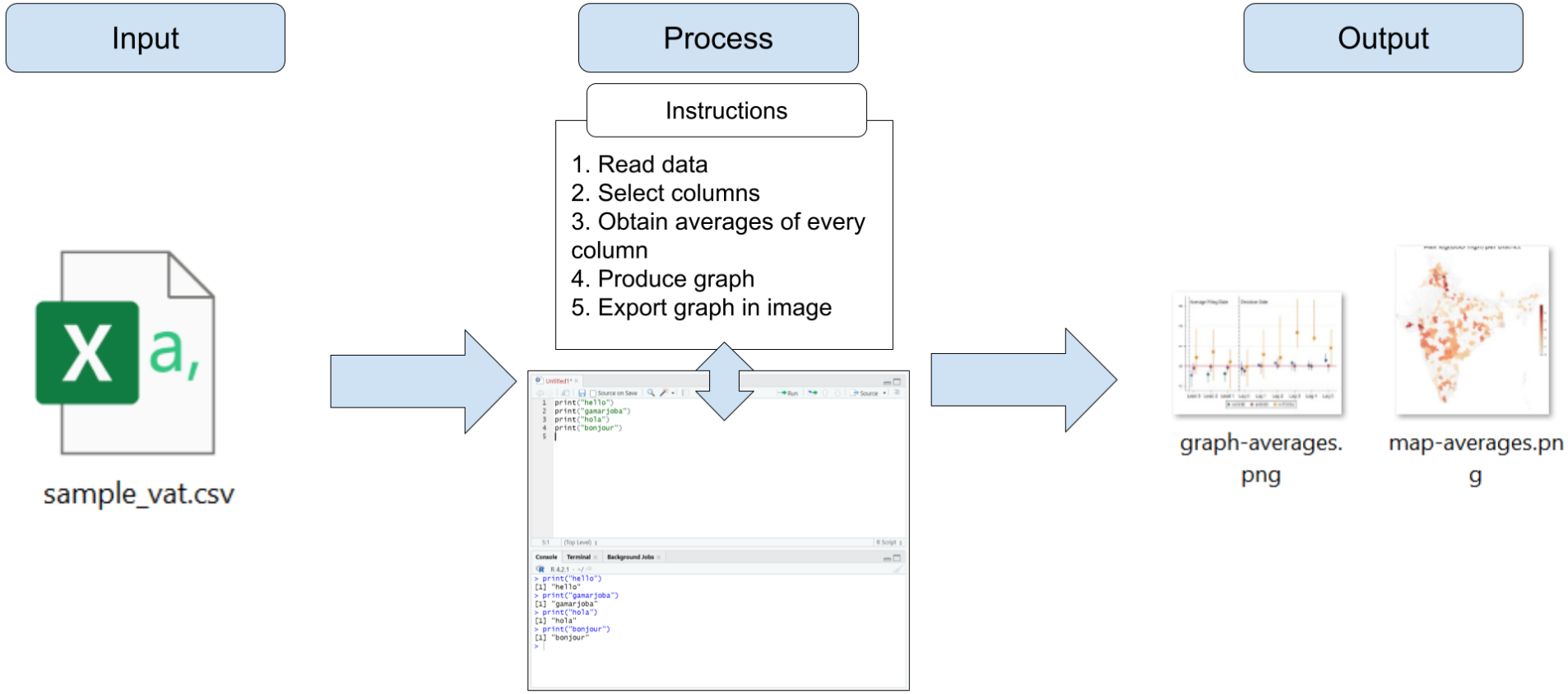
Luis Eduardo San Martin
The World Bank | September 2023

Table of contents // სარჩევი

1. About this session
2. R libraries
3. Data wrangling
4. Filtering and sorting
5. Merging dataframes
6. Exporting outputs
7. Wrapping up
8. Appendinx

About this session //

About this session // ამ სესიის შესახებ



About this session // ამ სესიის შესახებ

Input

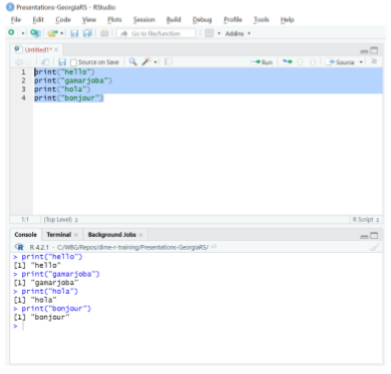
Process

Output

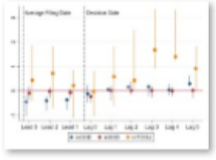
sample_vat.csv

Instructions

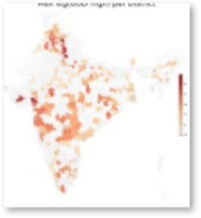
1. Read data
2. Select columns
3. Obtain averages of every column
4. Produce graph
5. Export graph in image



some more of this



graph-averages.png



map-averages.png

and basic implementation of data work pipeline

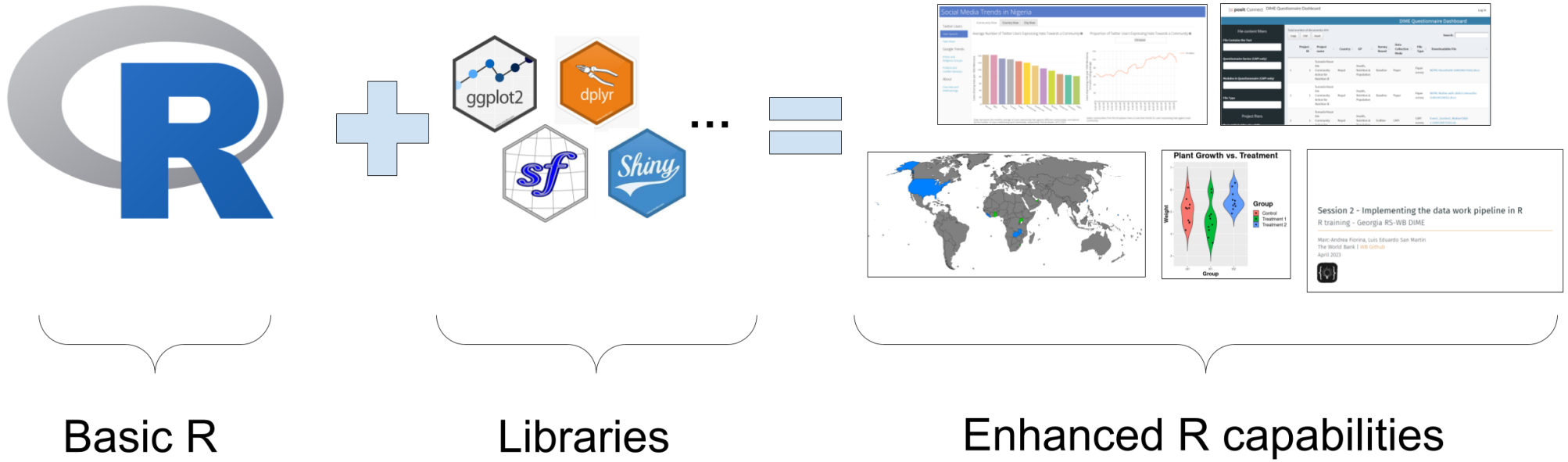
R libraries // R პაკეტები

R libraries // R პაკეტები

- Installing R in your computer gives you access to its basic functions
- Additionally, you can also install libraries. Libraries are packages of additional R functions that allow you to do:
  - Operations that basic R functions don't do (example: work with geographic data)
  - Operations that basic R functions do, but easier (example: data wrangling)

R libraries // R პაკეტები

In a nutshell:

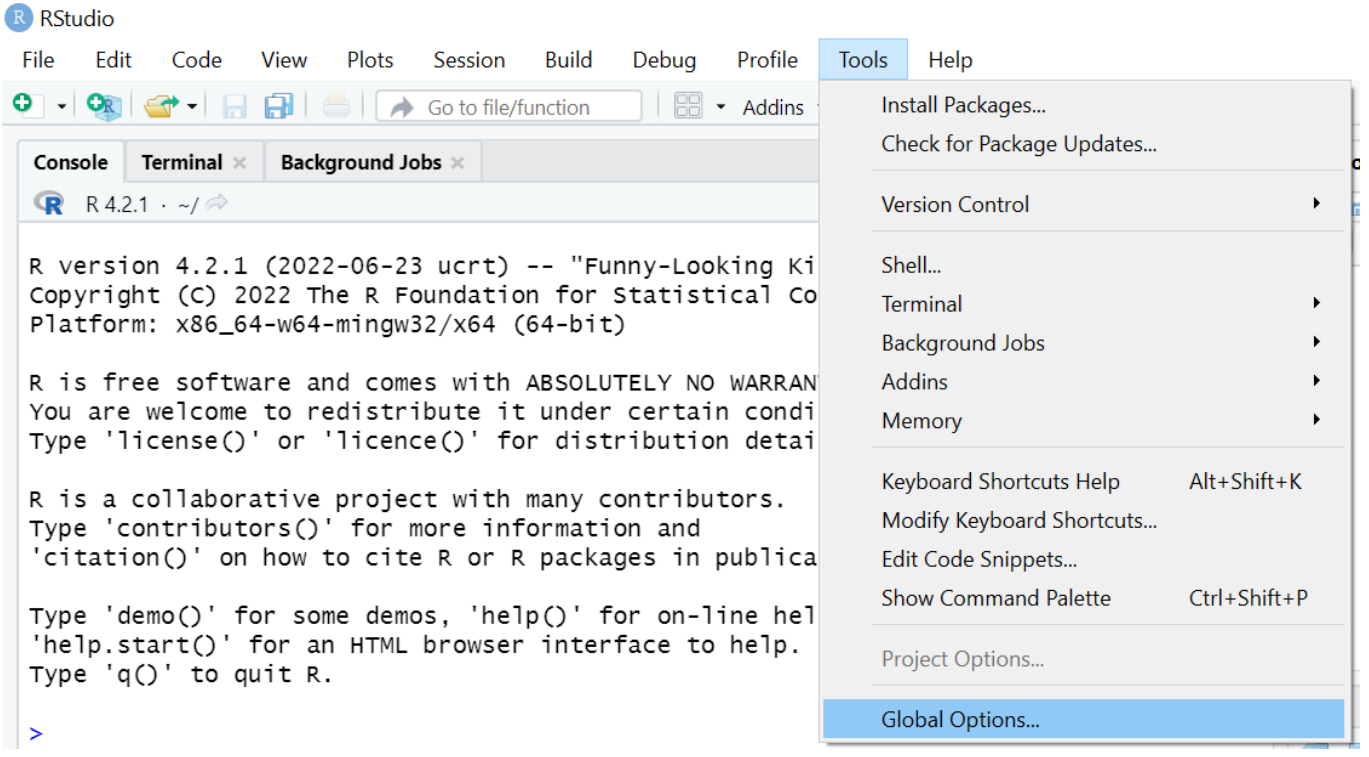


Installing R libraries

- Installing libraries is usually simple, but it can be challenging in institutional network connections such as the World Bank or the Georgia RS
- The next exercise will set up RStudio so that it can install R libraries without problems

Exercise 1: Setting up the installation of libraries

1 - In RStudio, go to `Tools` >> `Global Options...`



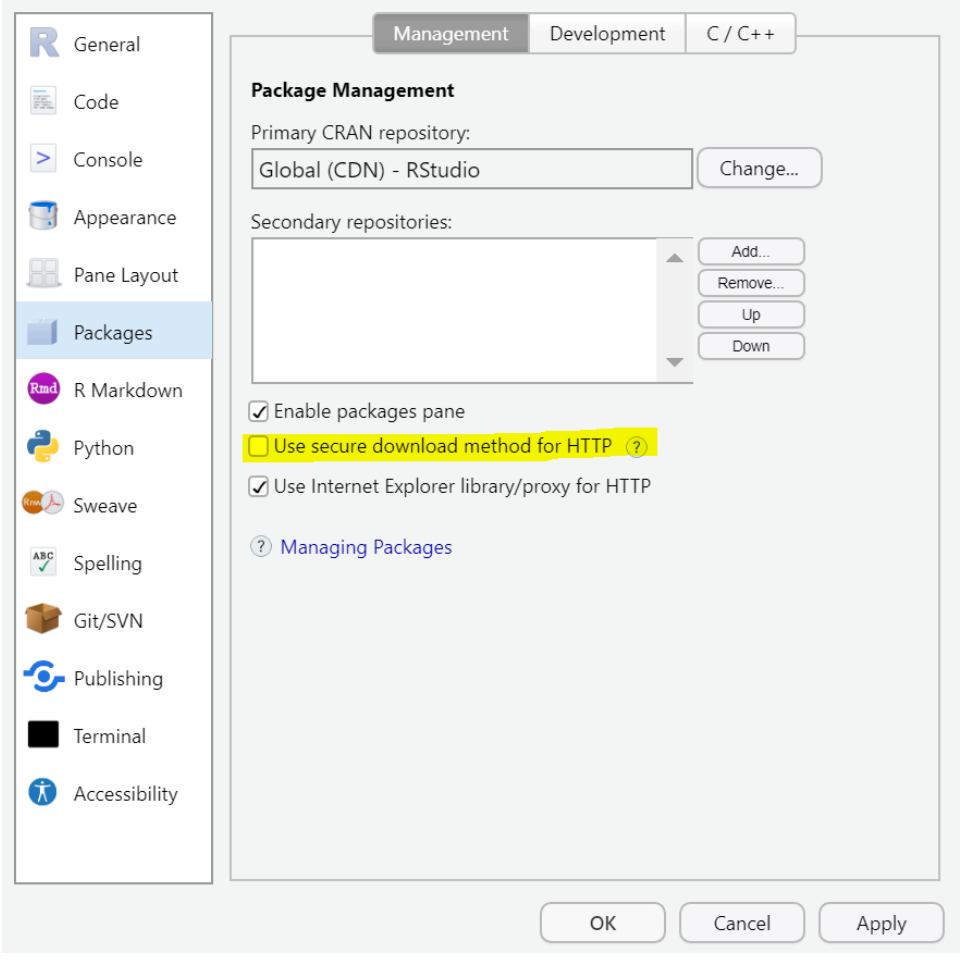
Exercise 1: Setting up the installation of libraries

2 - Select `Packages` in the left pane

3 - Uncheck `Use secure download method for HTTP`

4 - Click `OK`

You will not see any changes in your RStudio window after this, but now you'll be able to install libraries.

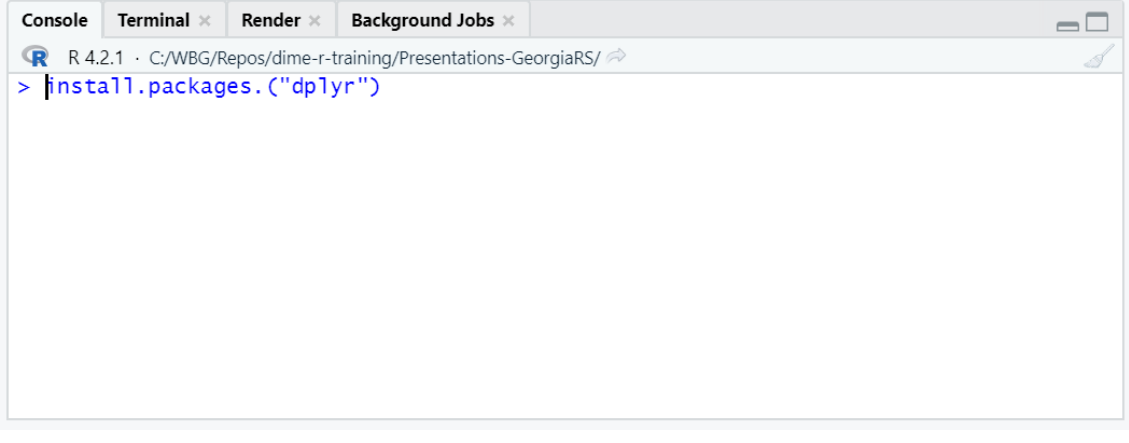


R libraries // R პაკეტები

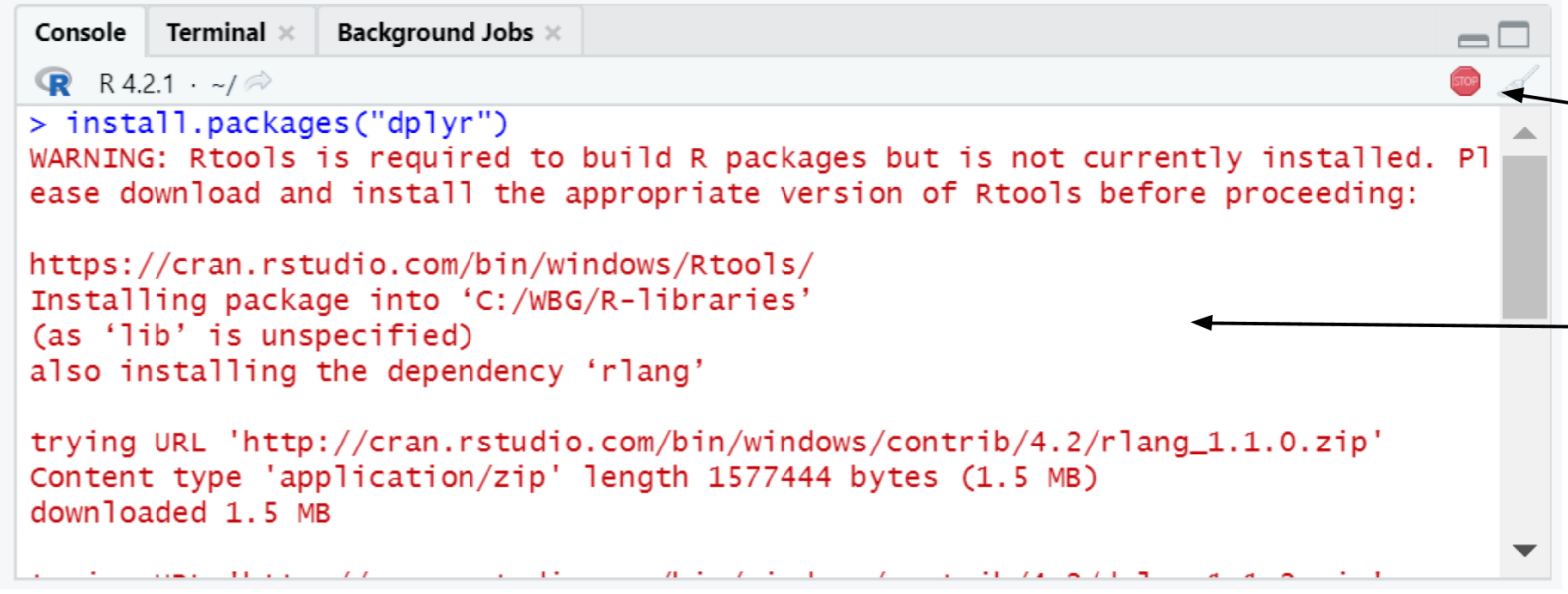
We'll use one librariy in today's session: `dplyr`

Exercise 2: Installing libraries

1. Install the libraries by using `install.packages()`
    - `install.packages("dplyr")`
    - Note the quotes (`" "`) in the packages names
    - **Introduce this code in the console**, not the script panel



Installing libraries



The "STOP" sign means that the code is still running, just wait until it finishes

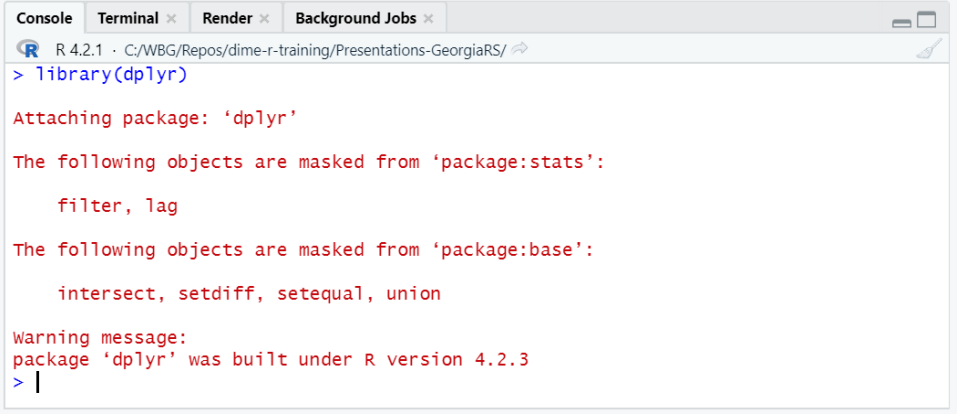
Note that this message is **not an error**

R libraries // R პაკეტები

Now that `dplyr` is installed, we only need to load them to start using the functions they have.

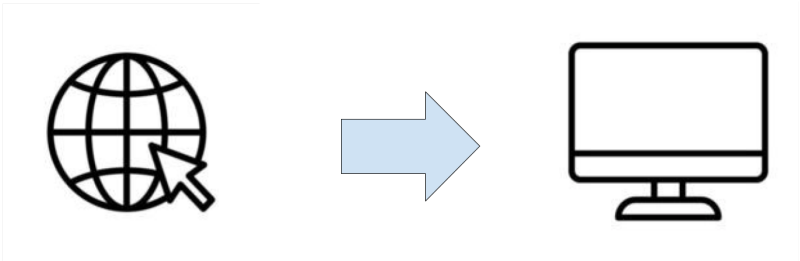
Exercise 3: Loading libraries

1. Open a new script with `File` >> `New File` >> `R Script`
2. Load `dplyr` with: `library(dplyr)`
    - Run this code from the new script you just opened
    - Notice that we don't use quotes in the library names this time

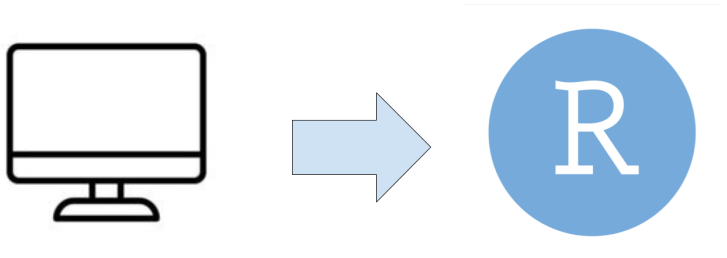


R libraries // R პაკეტები

- Library installation:



- Library loading:

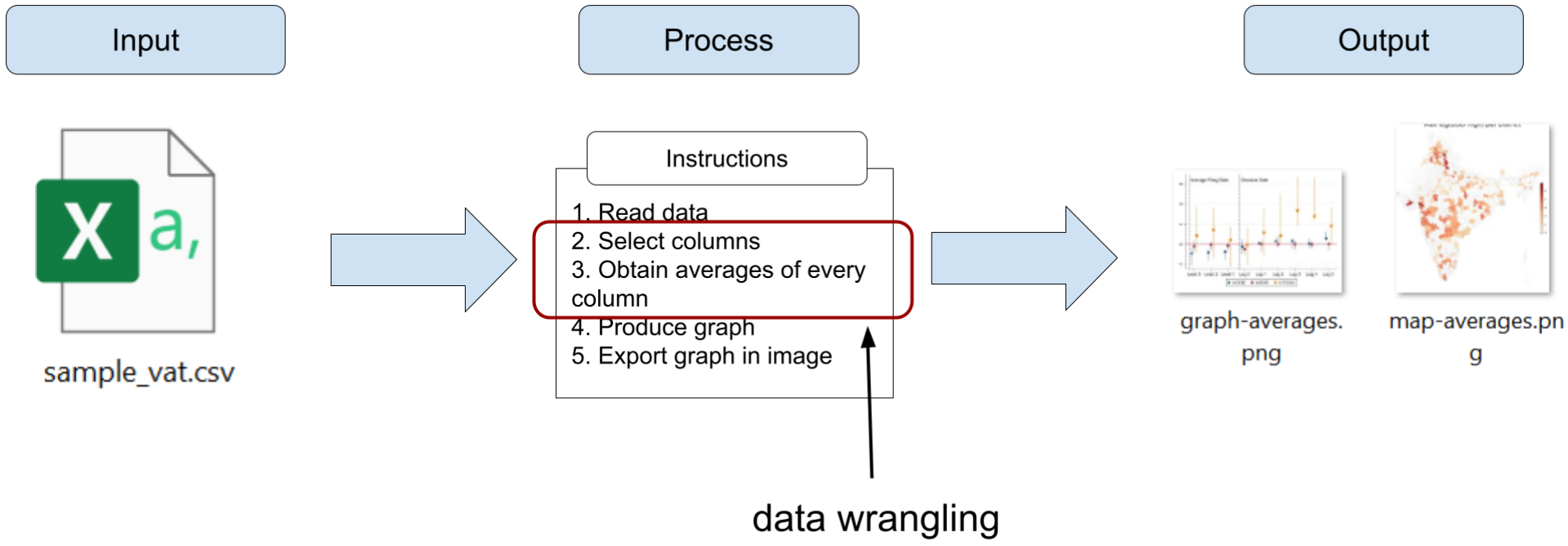


- You install R libraries only once in your computer
- You load libraries every time you open a new RStudio window (only load the libraries you will use)

Data wrangling // მონაცემთა ჩხუბი

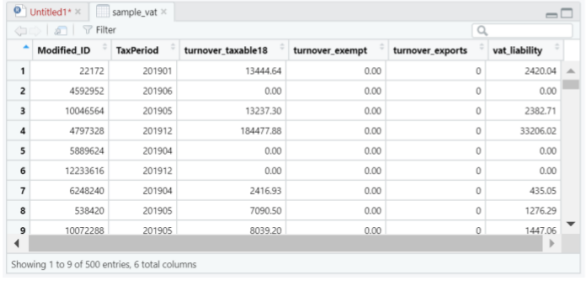
Getting your data ready

- Data is rarely in a format where it can be converted in an output right away
- In statistical programming, the process of transforming data into a condition where it's ready to be converted into an output is called **data wrangling**



Getting your data ready

- Data wrangling is one of the most crucials and time-consuming aspects of data work
- It involves not only coding, but also the mental exercise of thinking what is the shape and condition that your dataframe needs to have in order to produce your desired output



	Modified_ID	TaxPeriod	turnover_taxable18	turnover_exempt	turnover_exports	vat_liability
1	22172	201901	13444.64	0.00	0	2420.04
2	4592952	201906	0.00	0.00	0	0.00
3	10046564	201905	13237.30	0.00	0	2382.71
4	4797328	201912	184477.88	0.00	0	33206.02
5	5889624	201904	0.00	0.00	0	0.00
6	12233616	201912	0.00	0.00	0	0.00
7	6248240	201904	2416.93	0.00	0	435.05
8	538420	201905	7090.50	0.00	0	1276.29
9	10072288	201905	8039.20	0.00	0	1447.06

Showing 1 to 9 of 500 entries, 6 total columns

initial dataframe

dataframe needed for outputs

graph-averages.png

map-averages.png

outputs

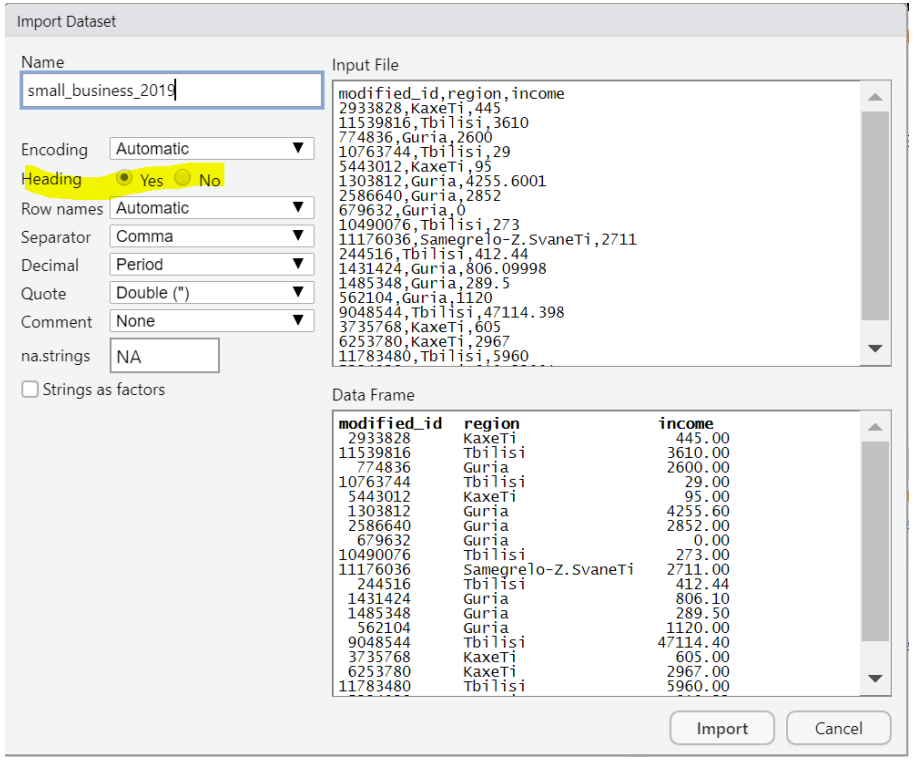
Getting your data ready

- As we said before we'll use `dplyr` and `tidyr` for data wrangling in this training
- You can also use basic R, but we recommend these libraries because its functions are easier to use

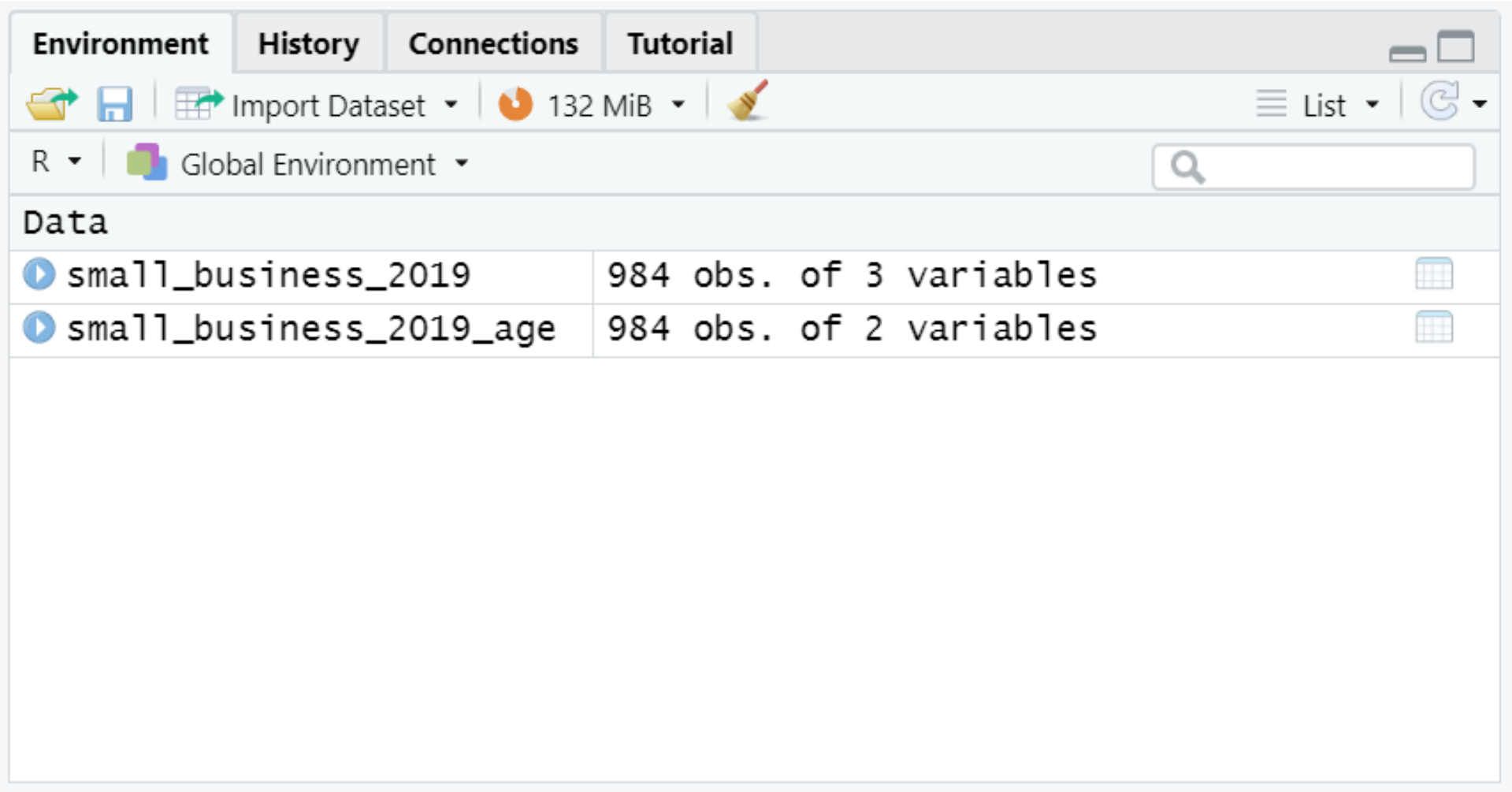
Exercise 4: Loading data

Note that this part of this is the same exercise we did in session 1, but it's okay to repeat it in order to start using a new RStudio session. **If you have RStudio open, start by closing the window and opening RStudio again**.

1. Inyou new RStudio window, go to `File` > `Import Dataset` > `From Text (base)` and select again the file `small_business_2019_age.csv`
    - if you don't know where the file is, check in the `Downloads` folder
    - if you need to download it again, it's here: https://osf.io/2apht
2. Make sure to select `Heading` > `Yes` in the next window
3. Select `Import`
4. Download this new file: https://osf.io/v6psa and repeat steps 1-3 with it



Data wrangling // მონაცემთა ჩხუბი



Note: loading data with a function

- You can also load CSV data with the function `read.csv()` instead of using this point-and-click approach
- The first argument of `read.csv()` is the path in your computer where your data is. For example

```
sample_vat <- read.csv("C:/Users/wb532468/Downloads/small_business_2019_age.csv")
```

- As usual, you need to save the result of `read.csv()` into a dataframe object with the arrow operator (`<-`) for it to be stored in the environment

Recap: knowing your data

- Dataframe `small_business_2019` is the same dataframe we used last session that contains reported income of small business in 2019 and their locations
- The new dataframe is `small_business_2019_age`
- Each row is one small business with their corresponding firm age for 2019
- Column `Modified_ID` is a taxpayer identifier
- `age` is the firm age

Data wrangling // მონაცემთა ჩხუბი

- We will only use this dataframe in one of the next exercises, but we loaded it now because it's in general a good practice to have data loaded into the memory so it's ready to be used
- For the next exercises, we will face (likely) scenarios that will show us data work operations that require data wrangling

Filtering and sorting // ფილტრაცია და დახარისხება

Data work request

Scenario 1: Imagine you're approached with the following request:

"The Georgia RS directorate is thinking of changing the criteria of what is considered a small business in Tbilisi. A critical input for this would be the list of the 50 biggest small business in Tbilisi in 2019. Can you produce such list? There is data from 2019 reported business income you can use"

Filtering and sorting // ფილტრაცია და დახარისხება

Data work request

"There is data from 2019 reported business income you can use"

exercises-session2.R* × | small_business_2019 × | df_tbilisi_50 ×

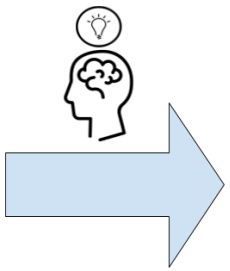
Filter

	modified_id	region	income
1	2933828	KaxeTi	445.00
2	11539816	Tbilisi	3610.00
3	774836	Guria	4500.00
4	10763744	Tbilisi	29.00
5	5443012	KaxeTi	95.00
6	1303812	Guria	1581.00
7	2586640	Guria	2852.00
8	679632	Guria	0.00
9	10490076	Tbilisi	273.00
10	11176036	Samegrelo-Z.SvaneTi	2711.00

Showing 1 to 11 of 984 entries, 3 total columns

- Observations: Business reported income of 2019

"A critical input for this would be the list of the 50 biggest small business in Tbilisi in 2019"



exercises-session2.R* × | small_business_2019 × | df_tbilisi_50 ×

Filter

	modified_id	region	income
1	1501688	Tbilisi	139394.52
2	2986564	Tbilisi	82007.90
3	11860128	Tbilisi	70000.00
4	2092784	Tbilisi	58690.76
5	9048544	Tbilisi	47114.40
6	10674748	Tbilisi	44900.68
7	7213596	Tbilisi	43429.35
8	545116	Tbilisi	37152.04
9	5937244	Tbilisi	35784.00
10	8591468	Tbilisi	35000.00

Showing 1 to 11 of 50 entries, 3 total columns

- Observations: Business reported income of 2019, only for Tbilisi, and the 50 highest
- Columns: notice that the columns are the same as in the starting dataframe

Filtering and sorting // ფილტრაცია და დახარისხება

Data work request

The data wrangling here involves a number of operations:

1. Keeping only the businesses in Tbilisi
2. Sorting by business income
3. Keeping only the 50 first businesses

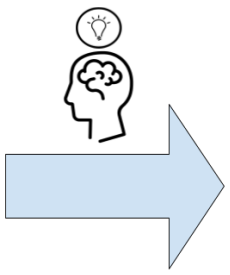
exercises-session2.R* | small_business_2019 | df_tbilisi_50

Filter

	modified_id	region	income
1	2933828	KaxeTi	445.00
2	11539816	Tbilisi	3610.00
3	774836	Guria	4500.00
4	10763744	Tbilisi	29.00
5	5443012	KaxeTi	95.00
6	1303812	Guria	1581.00
7	2586640	Guria	2852.00
8	679632	Guria	0.00
9	10490076	Tbilisi	273.00
10	11176036	Samegrelo-Z.SvaneTi	2711.00

Showing 1 to 11 of 984 entries, 3 total columns

- Observations: Business reported income of 2019



exercises-session2.R* | small_business_2019 | df_tbilisi_50

Filter

	modified_id	region	income
1	1501688	Tbilisi	139394.52
2	2986564	Tbilisi	82007.90
3	11860128	Tbilisi	70000.00
4	2092784	Tbilisi	58690.76
5	9048544	Tbilisi	47114.40
6	10674748	Tbilisi	44900.68
7	7213596	Tbilisi	43429.35
8	545116	Tbilisi	37152.04
9	5937244	Tbilisi	35784.00
10	8591468	Tbilisi	35000.00

Showing 1 to 11 of 50 entries, 3 total columns

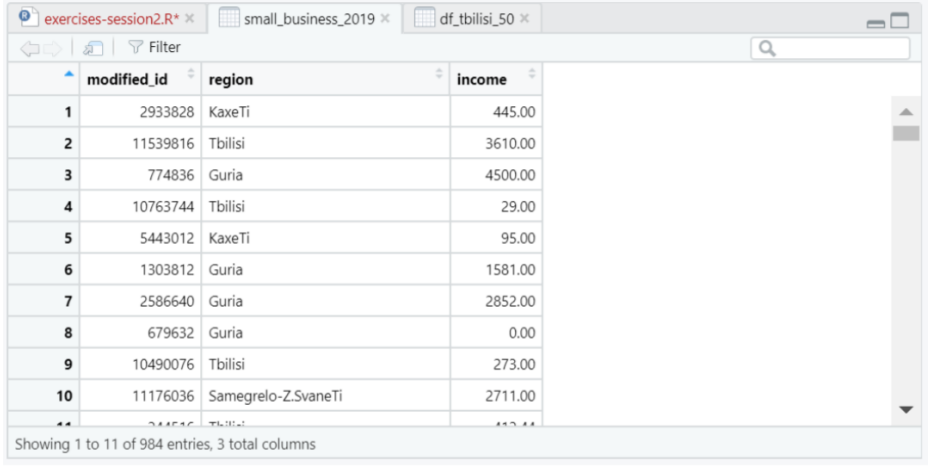
- Observations: Business reported income of 2019, only for Tbilisi, and the 50 highest
- Columns: notice that the columns are the same as in the starting dataframe

Filtering and sorting // ფილტრაცია და დახარისხება

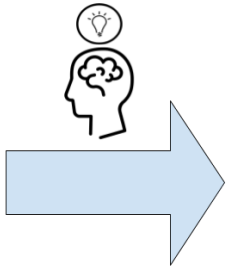
1. Keeping only the businesses in Tbilisi

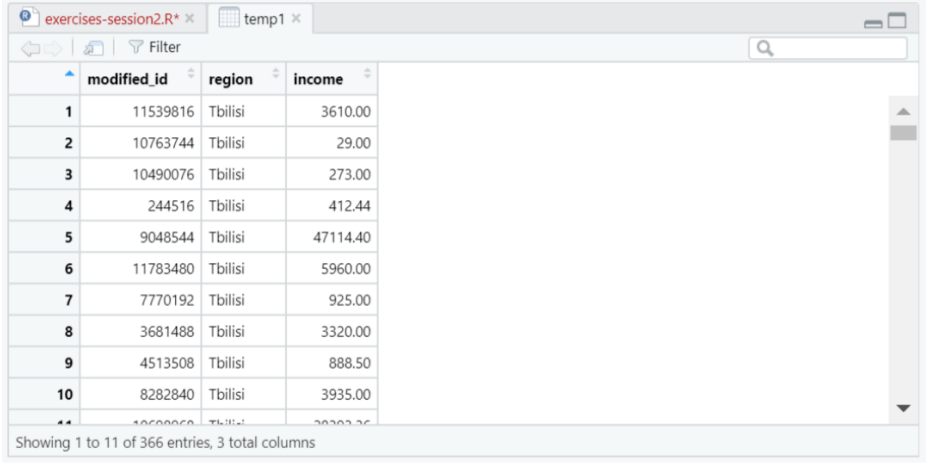
Use `filter()` for this:

```
temp1 <- filter(small_business_2019, region == "Tbilisi")
```



- Observations: Business reported income of 2019





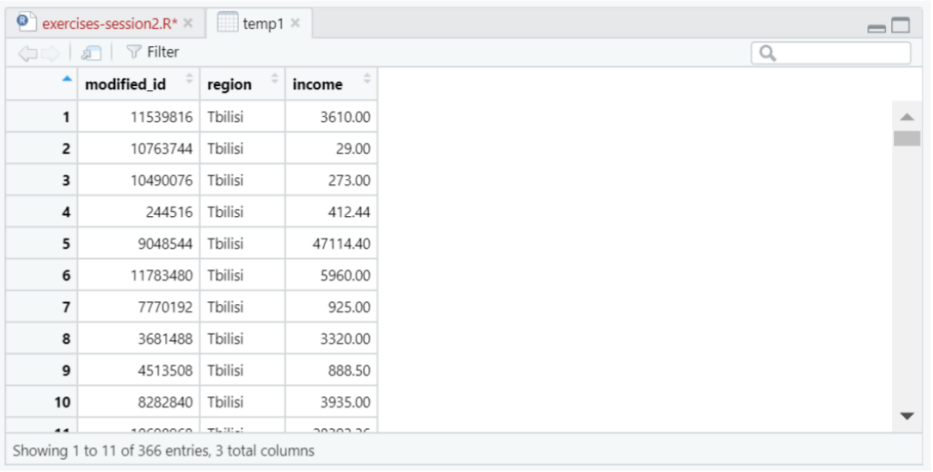
- Observations: Business reported income of 2019, only for Tbilisi

Filtering and sorting // ფილტრაცია და დახარისხება

2. Sorting by business income

Use the function `arrange()` to sort. Sortings are ascending by default in R, hence we can include the minus (`-`) symbol before `income` to tell R we want to sort descending

```
small_business_tb_sorted <- arrange(small_business_tb, -income)
```



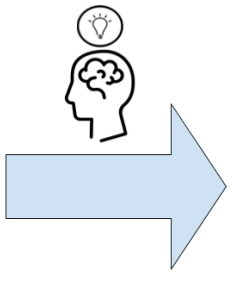
exercises-session2.R* | temp1

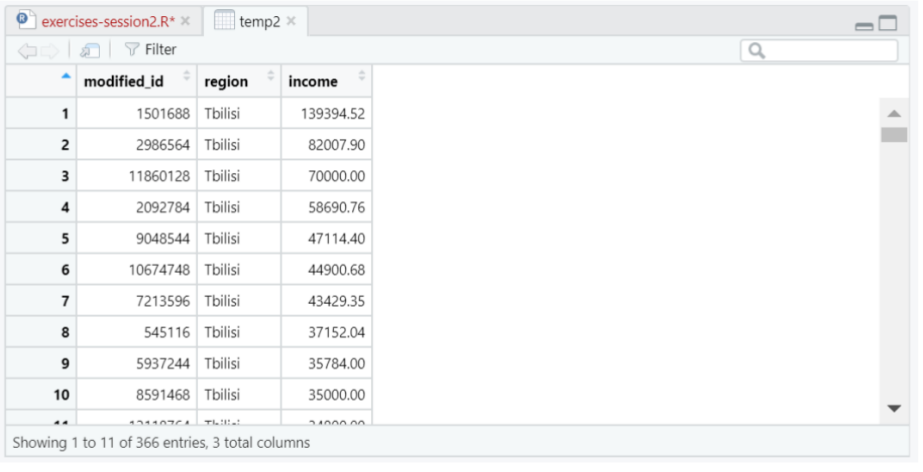
Filter

	modified_id	region	income
1	11539816	Tbilisi	3610.00
2	10763744	Tbilisi	29.00
3	10490076	Tbilisi	273.00
4	244516	Tbilisi	412.44
5	9048544	Tbilisi	47114.40
6	11783480	Tbilisi	5960.00
7	7770192	Tbilisi	925.00
8	3681488	Tbilisi	3320.00
9	4513508	Tbilisi	888.50
10	8282840	Tbilisi	3935.00

Showing 1 to 11 of 366 entries, 3 total columns

- Observations: Business reported income of 2019, only for Tbilisi





exercises-session2.R* | temp2

Filter

	modified_id	region	income
1	1501688	Tbilisi	139394.52
2	2986564	Tbilisi	82007.90
3	11860128	Tbilisi	70000.00
4	2092784	Tbilisi	58690.76
5	9048544	Tbilisi	47114.40
6	10674748	Tbilisi	44900.68
7	7213596	Tbilisi	43429.35
8	545116	Tbilisi	37152.04
9	5937244	Tbilisi	35784.00
10	8591468	Tbilisi	35000.00

Showing 1 to 11 of 366 entries, 3 total columns

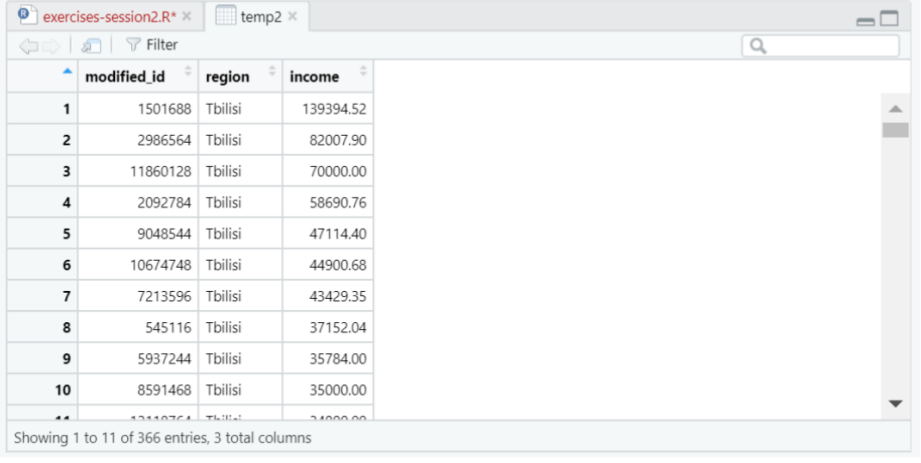
- Observations: Business reported income of 2019, only for Tbilisi, sorted descending by income

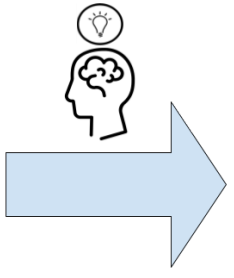
Filtering and sorting // ფილტრაცია და დახარისხება

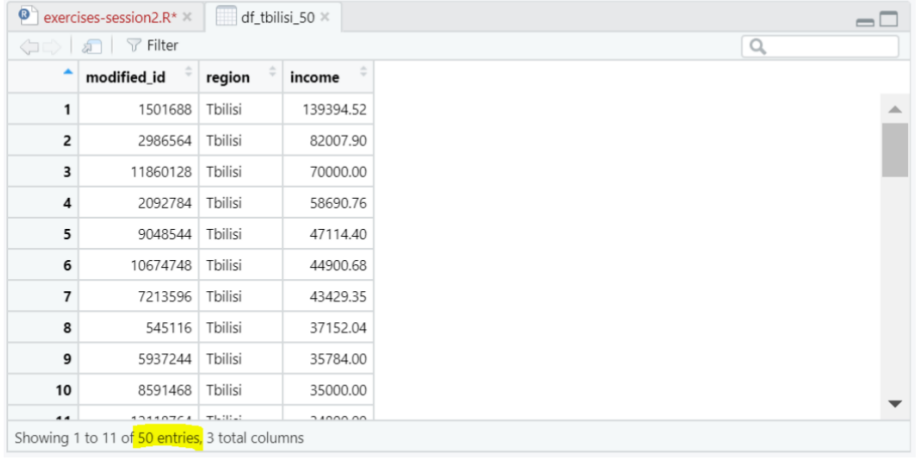
3. Keeping only the 50 first businesses after the sorting

Use `filter()` again and the helper command `nrow()`

```
result_scenario1 <- filter(small_business_tb_sorted, row_number() <= 50)
```







- Observations: Business reported income of 2019, only for Tbilisi, sorted descending by income

- Observations: Business reported income of 2019, only for Tbilisi, sorted descending by income, and the 50 highest

Exercise 5: filter and sort your data

Now that we figured out the shape the resulting dataframe needs to have and how to get it there, we can write code for this.

1.- Row selection:

```
temp1 <- filter(small_business_2019, region == "Tbilisi")
```

2.- Sort descending by income:

```
temp2 <- arrange(temp1, -income)
```

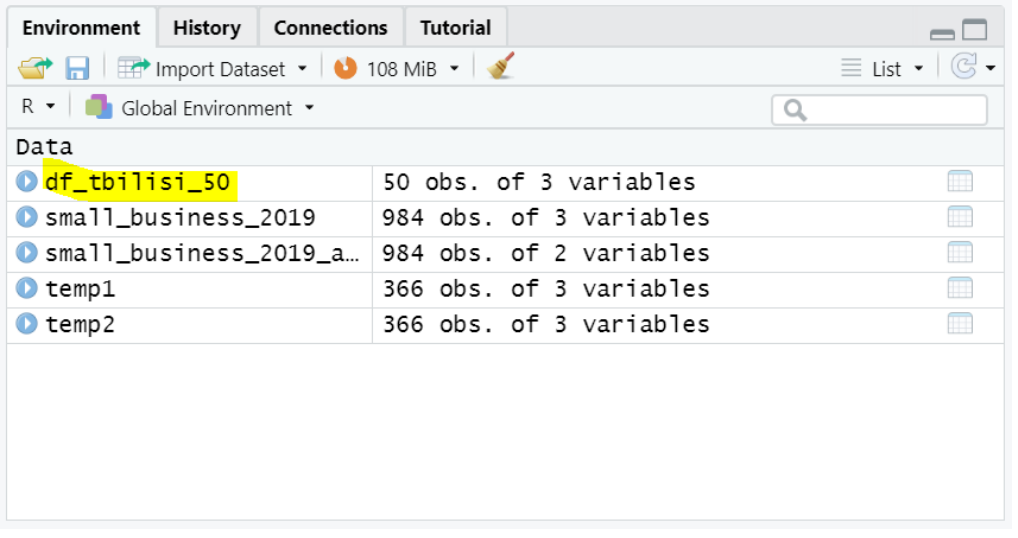
3.- Keep only the 50 first businesses after sorting:

```
df_tbilisi_50 <- filter(temp2, row_number() <= 50)
```

Filtering and sorting // ფილტრაცია და დახარისხება

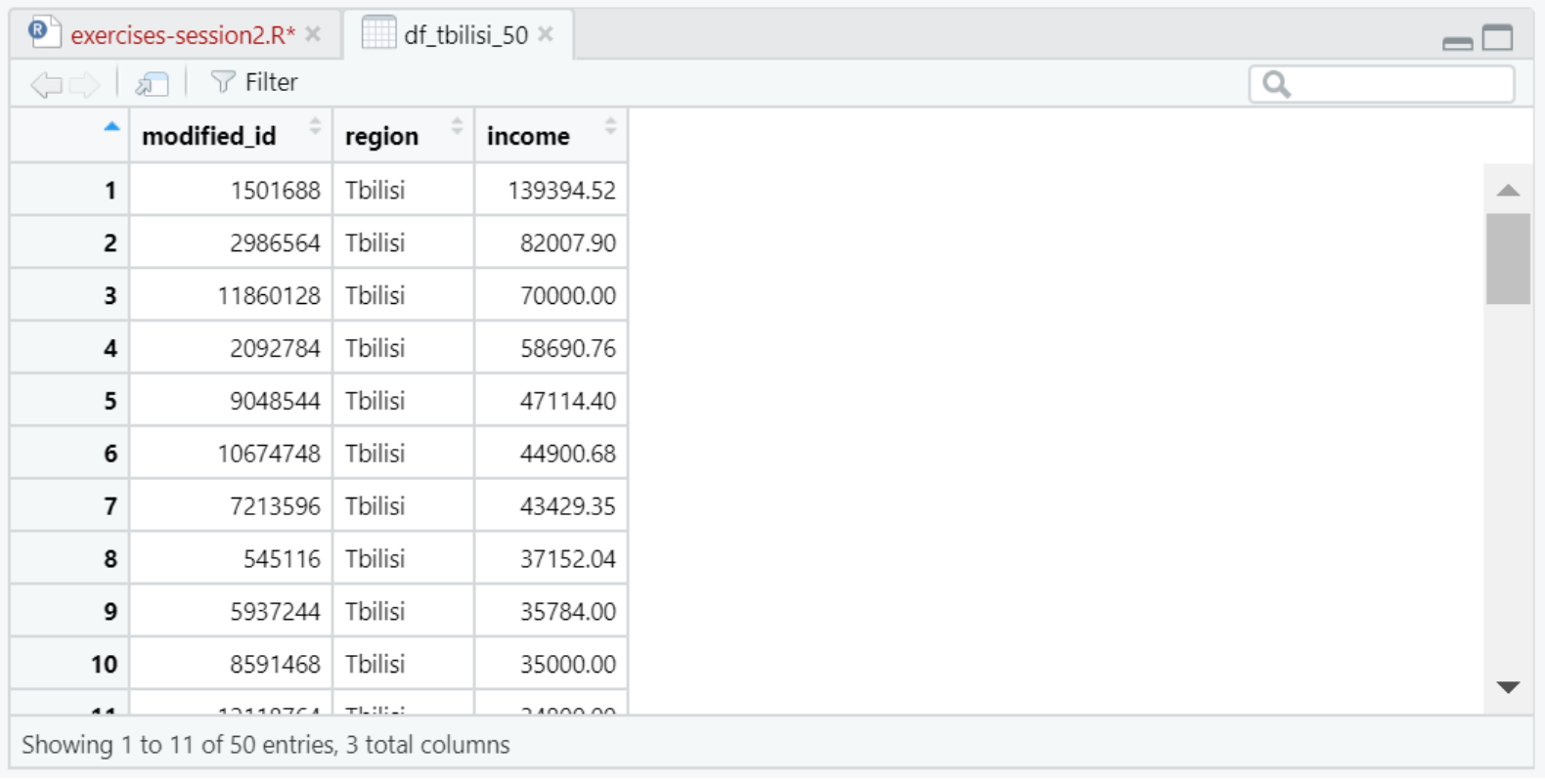
Some notes:

- `filter()`, `arrange()`, and `row_number()` are all functions from `dplyr`. Remember you have to always load `dplyr` first with `library(dplyr)` to be able to use them
- Note that we're creating two intermediate dataframes named `temp1` and `temp2` in the process
- We can avoid this by using the pipes operator (`%>%`). Pipes are very common in R programming, but we're not explaining them in this session
- The resulting dataframe is `df_tbilisi_50`



Filtering and sorting // ფილტრაცია და დახარისხება

You can check the result with `View(df_tbilisi_50)`. Now this dataframe has exactly what we wanted!



	modified_id	region	income
1	1501688	Tbilisi	139394.52
2	2986564	Tbilisi	82007.90
3	11860128	Tbilisi	70000.00
4	2092784	Tbilisi	58690.76
5	9048544	Tbilisi	47114.40
6	10674748	Tbilisi	44900.68
7	7213596	Tbilisi	43429.35
8	545116	Tbilisi	37152.04
9	5937244	Tbilisi	35784.00
10	8591468	Tbilisi	35000.00

Showing 1 to 11 of 50 entries, 3 total columns

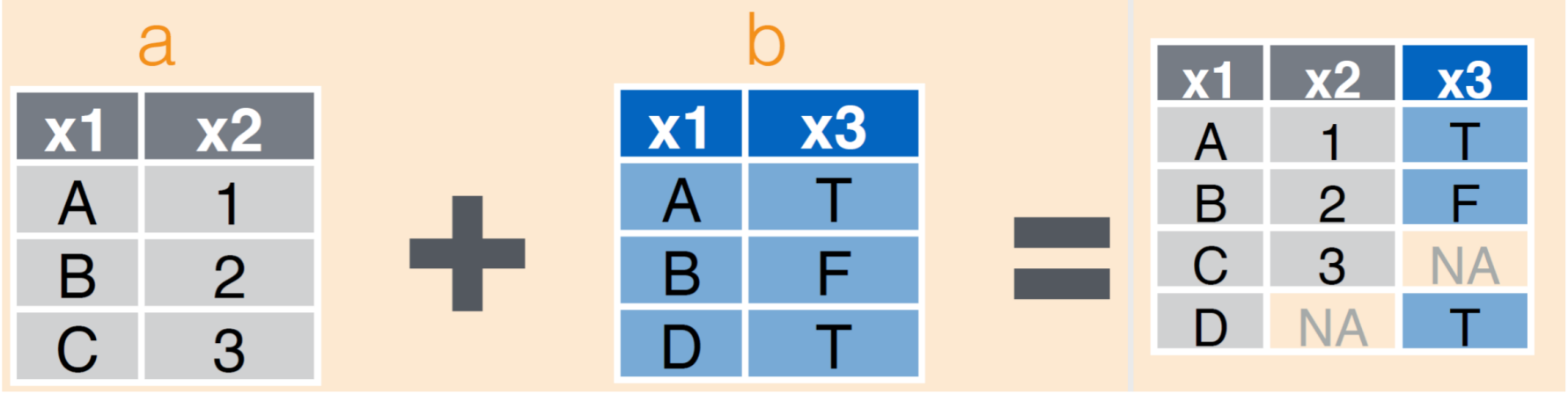
Filtering and sorting // ფილტრაცია და დახარისხება

- Filtering and sorting are two very common data wrangling operations in statistical programming
- Now we'll review a new data wrangling operation that is also quite common and useful: **merging**, also called joining

Merging dataframes // მონაცემთა ჩარჩოების შეერთება

Merging dataframes

- We'll explore one more common data wrangling operation: merging
- Merging is used when you need to bring columns from one dataframe to another
- When merging you need to use a "key" column that identifies the same units in different dataframes



Note: Image taked from RStudio's data wrangling cheat sheet

Merging dataframes

Data work request

Scenario 2:

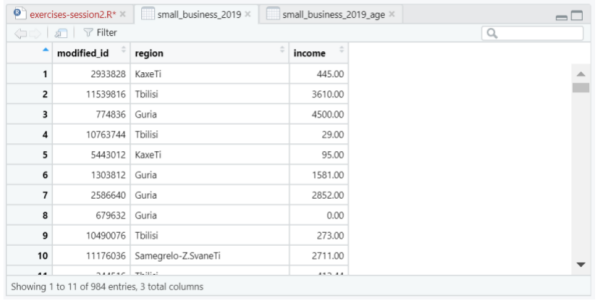
"We want to know the total business reported income for small businesses with more than 5 years in 2019. You might want to use the data of small businesses income in 2019 that you already know and another file with small businesses age"

Merging dataframes

Data work request

"Use the data of small businesses income in 2019 that you already know and another file with small businesses age"

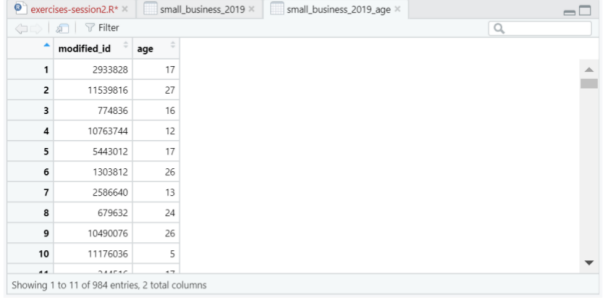
"We want to know the total business income for small businesses with more than 5 years in 2019"



	modified_id	region	income
1	2933828	KaxeTi	445.00
2	11539816	Tbilisi	3610.00
3	774836	Guria	4500.00
4	10763744	Tbilisi	29.00
5	5443012	KaxeTi	95.00
6	1303812	Guria	1581.00
7	2586640	Guria	2852.00
8	679632	Guria	0.00
9	10490076	Tbilisi	273.00
10	11176036	Samegrelo-Z.SvaneTi	2711.00

Showing 1 to 11 of 984 entries, 3 total columns

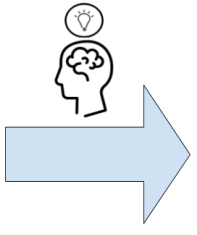
- Observations: Business reported income and region of 2019



	modified_id	age
1	2933828	17
2	11539816	27
3	774836	16
4	10763744	12
5	5443012	17
6	1303812	26
7	2586640	13
8	679632	24
9	10490076	26
10	11176036	5

Showing 1 to 11 of 984 entries, 2 total columns

- Observations: business age in 2019



2592266

- Total business income for small businesses with more than five years in 2019

Merging dataframes

Data work request

These are the steps we'd need to follow to get the data wrangled for this result:

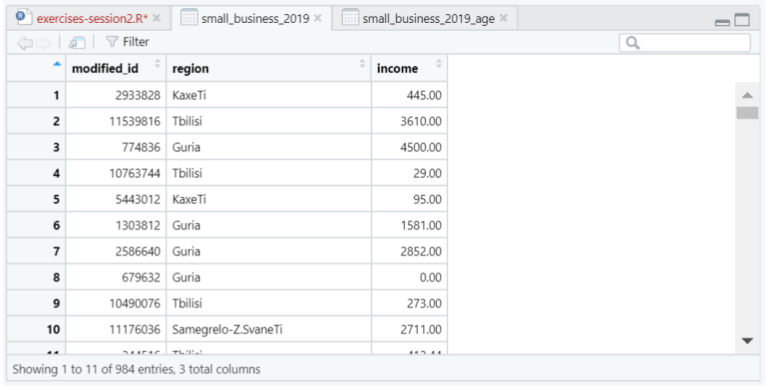
1. Select only the relevant columns from `small_business_2019`
2. Merge the dataframes
3. Filter only businesses with more than 5 years of age
4. Calculate the total income

Merging dataframes

1. Select only the relevant columns from `small_business_2019`

Use `select()` for this:

```
temp1 <- select(small_business_2019, modified_id, income)
```



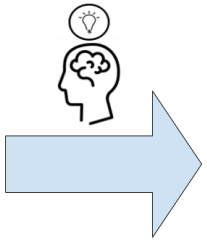
exercises-session2.R* | small_business_2019 | small_business_2019_age

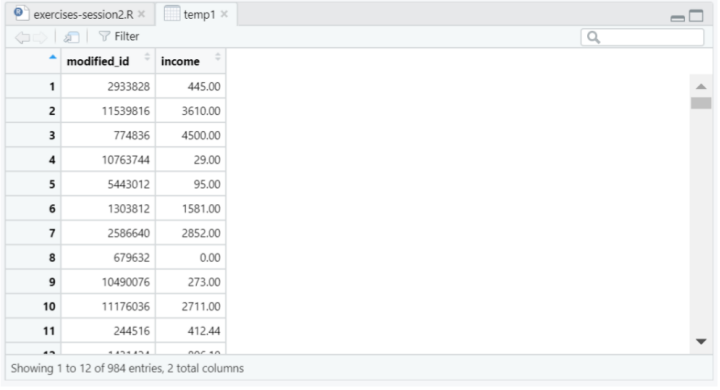
Filter

	modified_id	region	income
1	2933828	KaxeTi	445.00
2	11539816	Tbilisi	3610.00
3	774836	Guria	4500.00
4	10763744	Tbilisi	29.00
5	5443012	KaxeTi	95.00
6	1303812	Guria	1581.00
7	2586640	Guria	2852.00
8	679632	Guria	0.00
9	10490076	Tbilisi	273.00
10	11176036	Samegrelo-Z.SvaneTi	2711.00

Showing 1 to 11 of 984 entries, 3 total columns

- Observations: Business reported income and region of 2019
- Columns: modified_id, region, income





exercises-session2.R | temp1

Filter

	modified_id	income
1	2933828	445.00
2	11539816	3610.00
3	774836	4500.00
4	10763744	29.00
5	5443012	95.00
6	1303812	1581.00
7	2586640	2852.00
8	679632	0.00
9	10490076	273.00
10	11176036	2711.00
11	244516	412.44

Showing 1 to 12 of 984 entries, 2 total columns

- Observations: Business reported income in 2019
- Columns: modified_id, income

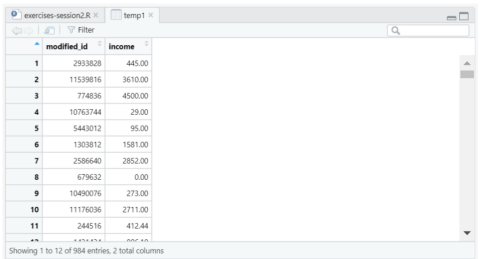
Merging dataframes

2. Merge the dataframes

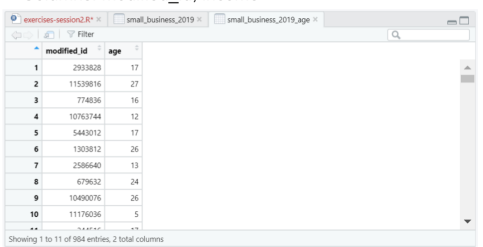
Use `inner_join()` to merge the dataframes:

```
temp2 <- inner_join(temp1, small_business_2019_age, by = "modified_id")
```

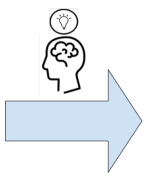
- The first two arguments are the two dataframes we want to join
- The third (named) argument is the "key" variable we merge by

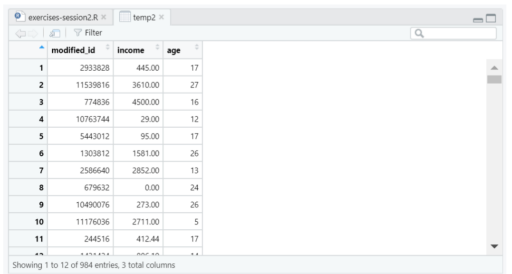


- Observations: Business reported income in 2019
- Columns: modified_id, income



- Observations: Business age in 2019
- Columns: modified_id, age





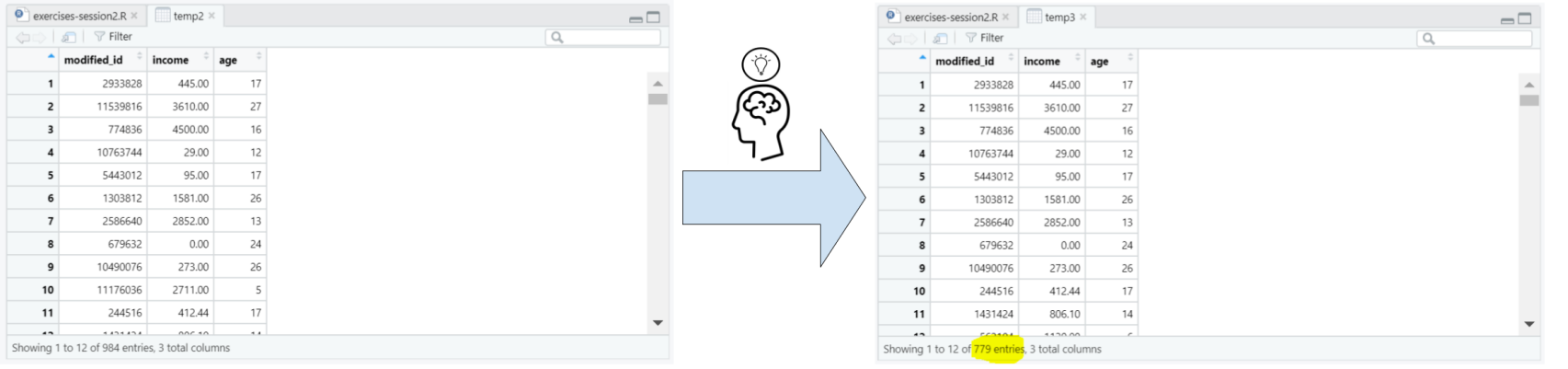
- Observations: Business age and reported income in 2019
- Columns: modified_id, income, age

Merging dataframes

3. Filter only businesses with more than 5 years of age

Use `filter()` once again:

```
temp3 <- filter(temp2, age > 5)
```



- Observations: Business age and reported income in 2019
- Columns: modified_id, income, age

- Observations: Business age and reported income in 2019, only for firms with more than five years of age
- Columns: modified_id, income, age

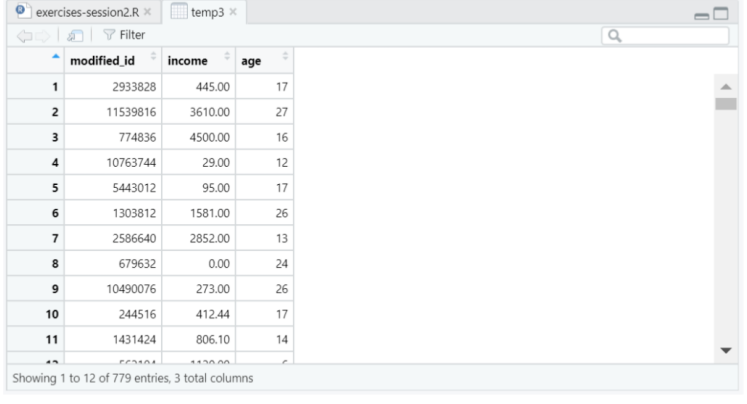
Merging dataframes

4. Calculate the total income

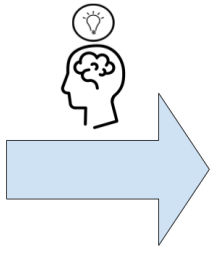
We use `colSums()` and `select()`.

- `colSums()` calculates de sum of all values of a column
- We use `select()` inside `colSums()` to select a single column from `temp3` to calculate the sum for

```
total_income <- colSums(select(temp3, income))
```



- Observations: Business age and reported income in 2019, only for firms with more than five years of age
- Columns: modified_id, income, age



2592266

- Total business income for small businesses with more than five years in 2019

Merging dataframes

Exercise 6: Merge the dataframes

Apply all the steps we reviewed in the last slides to calculate the total reported income for small businesses with more than five years of age in 2019.

1.- Select only the relevant columns of `small_business_2019`:

```
temp1 <- select(small_business_2019, modified_id, income)
```

2.- Merge the dataframes

```
temp2 <- inner_join(temp1, small_business_2019_age, by = "modified_id")
```

3.- Filter only businesses with more than five years of age

```
temp3 <- filter(temp2, age > 5)
```

4.- Calculate the total income

```
total_income <- colSums(select(temp3, income))
```

Merging dataframes

`total_income` is a numeric value with the amount we wanted to estimate.

```
print(total_income)
```

```
##  income
## 2592266
```

Merging dataframes

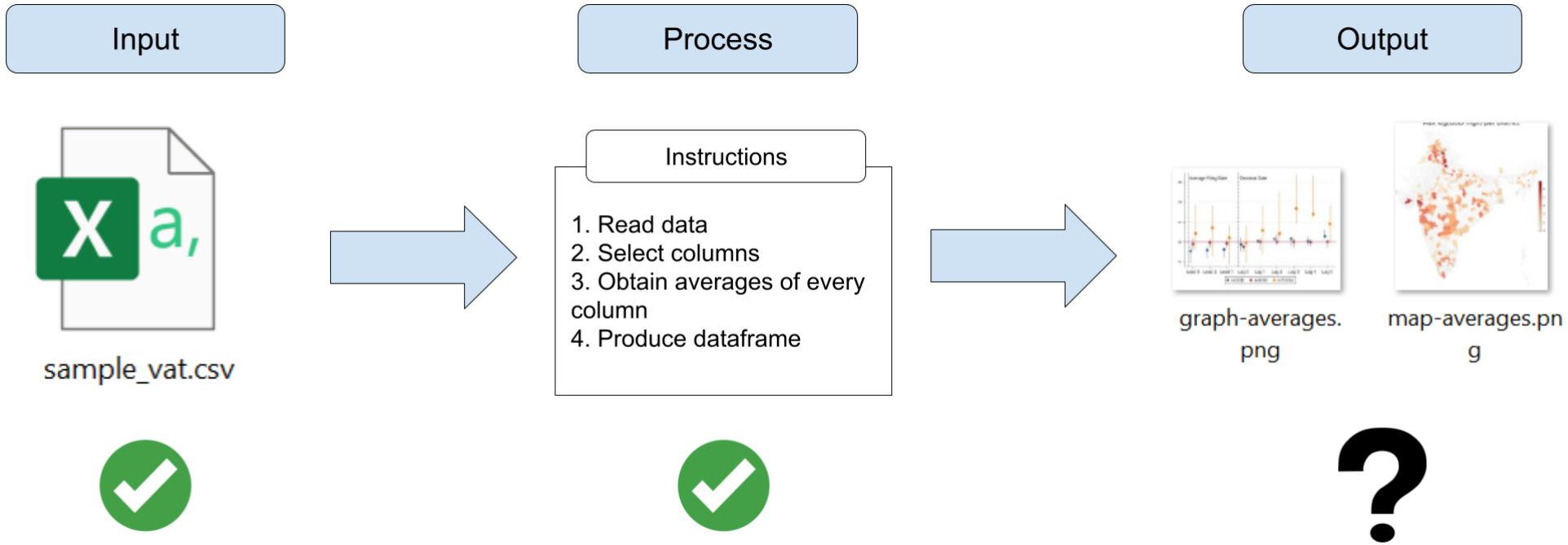
These were two examples we chose to show different possible data wrangling operations. A summary of these and other common operations are:

Operation	Function in `dplyr`
Subset columns	`select()`
Subset rows (based on condition)	`filter()`
Create new columns	`mutate()`
Create new columns based on condition	`mutate()` and `case_when()`
Create new rows	`add_row()`
Merge dataframes	`inner_join()`, `left_join()`, `right_join()`, `full_join()`
Append dataframes	`bind_rows()`
Deduplicate	`distinct()`
Collapse and create summary indicators	`group_by()`, `summarize()`
Pass a result as the first argument for the next function	`%>%` (operator, not function)

Exporting outputs // პროდუქციის ექსპორტი

Exporting outputs // პროდუქციის ექსპორტი

- Until now, we've seen full examples of part 1 and 2 of the data work pipeline
- What about exporting outputs?



- We'll see this in the next exercise

Exporting outputs // პროდუქციის ექსპორტი

Exporting dataframes

- The easiest way to export a dataframe or number is with the function `write.csv()`
- `write.csv()` creates a CSV file with the dataframe
- It takes two basic arguments:
    1. The name of the object you want to export
    2. A file path to export the object to
- `write.csv()` includes the row numbers by default. You can add the argument `row.names = FALSE` to avoid this

Exporting outputs // პროდუქციის ექსპორტი

Exercise 7: Export `df_tbilisi_50` and `total_income`

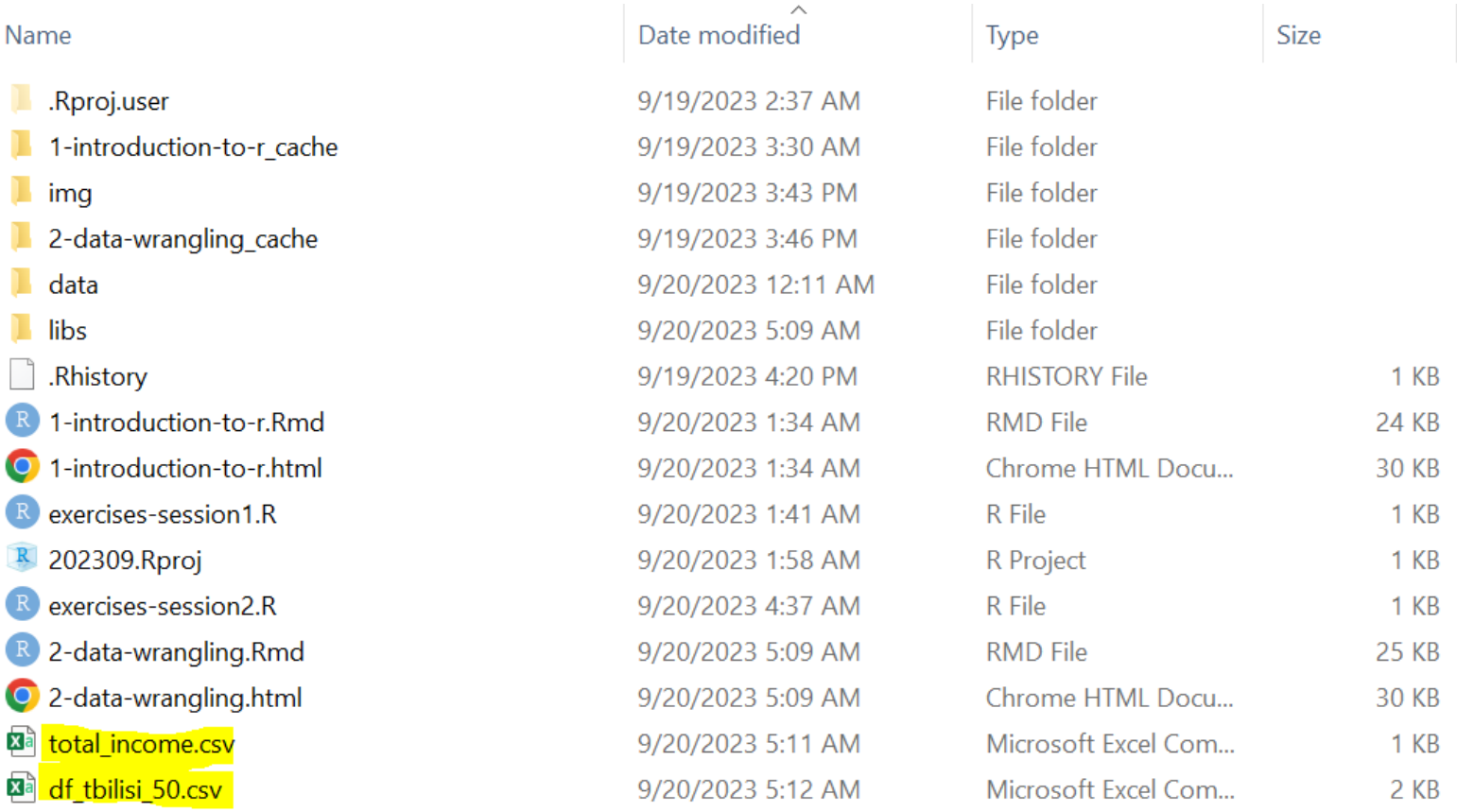
1. Use this code to export the results of the last two exercises:

```
write.csv(df_tbilisi_50,
          "df_tbilisi_50.csv",
          row.names = FALSE)

write.csv(total_income,
          "total_income.csv",
          row.names = FALSE)
```

Exporting outputs // პროდუქციის ექსპორტი

Now `df_tbilisi_50.csv` and `total_income.csv` will show in your computer (probably in your `Documents` folder).



Name	Date modified	Type	Size
.Rproj.user	9/19/2023 2:37 AM	File folder	
1-introduction-to-r_cache	9/19/2023 3:30 AM	File folder	
img	9/19/2023 3:43 PM	File folder	
2-data-wrangling_cache	9/19/2023 3:46 PM	File folder	
data	9/20/2023 12:11 AM	File folder	
libs	9/20/2023 5:09 AM	File folder	
.Rhistory	9/19/2023 4:20 PM	RHISTORY File	1 KB
1-introduction-to-r.Rmd	9/20/2023 1:34 AM	RMD File	24 KB
1-introduction-to-r.html	9/20/2023 1:34 AM	Chrome HTML Docu...	30 KB
exercises-session1.R	9/20/2023 1:41 AM	R File	1 KB
202309.Rproj	9/20/2023 1:58 AM	R Project	1 KB
exercises-session2.R	9/20/2023 4:37 AM	R File	1 KB
2-data-wrangling.Rmd	9/20/2023 5:09 AM	RMD File	25 KB
2-data-wrangling.html	9/20/2023 5:09 AM	Chrome HTML Docu...	30 KB
total_income.csv	9/20/2023 5:11 AM	Microsoft Excel Com...	1 KB
df_tbilisi_50.csv	9/20/2023 5:12 AM	Microsoft Excel Com...	2 KB

Exporting outputs // პროდუქციის ექსპორტი

Some notes on file paths

- The second argument of `write.csv()` specifies the file path we export the dataframe to

```
write.csv(df_tbilisi_50,
          "df_tbilisi_50.csv",
          row.names = FALSE)
```

- You can include any path in your computer and R will write the file in that location
    - For example: `"C:/Users/wb532468/OneDrive - WBG/Desktop"` exports the file to the desktop of my computer (this will not work in other computers)
    - Note that file paths in R use forward slashes (`/`). Back slashes (`\`) **do not work in R**

Exporting outputs // პროდუქციის ექსპორტი

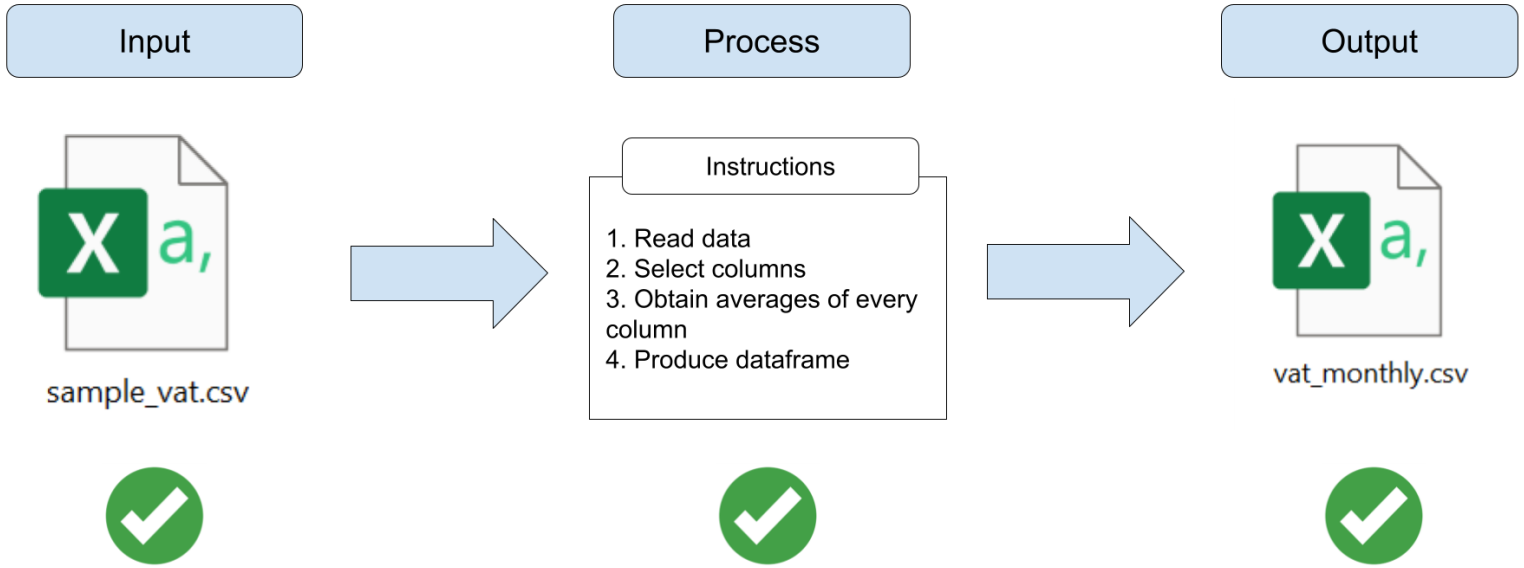
Some notes on file paths

```
write.csv(df_tbilisi_50,
          "df_tbilisi_50.csv",
          row.names = FALSE)
```

- If you only include a file name (as in `df_tbilisi_50.csv`), R will export the file to the current location of your RStudio window. This is usually the `Documents` folder in Windows
- You can check the current location of RStudio with the function `getwd()`

Exporting outputs // პროდუქციის ექსპორტი

Our data pipeline has been fully implemented at this point. Great!

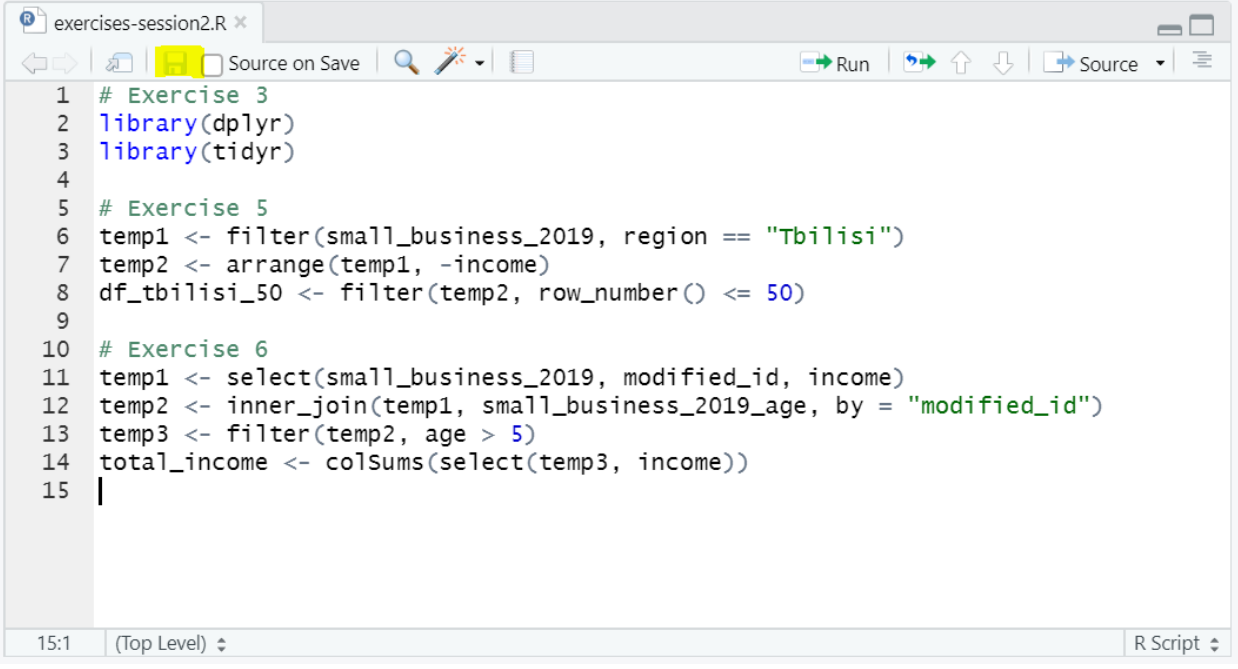


Wrapping up // შეფუთვა

Wrapping up // შეჯამება

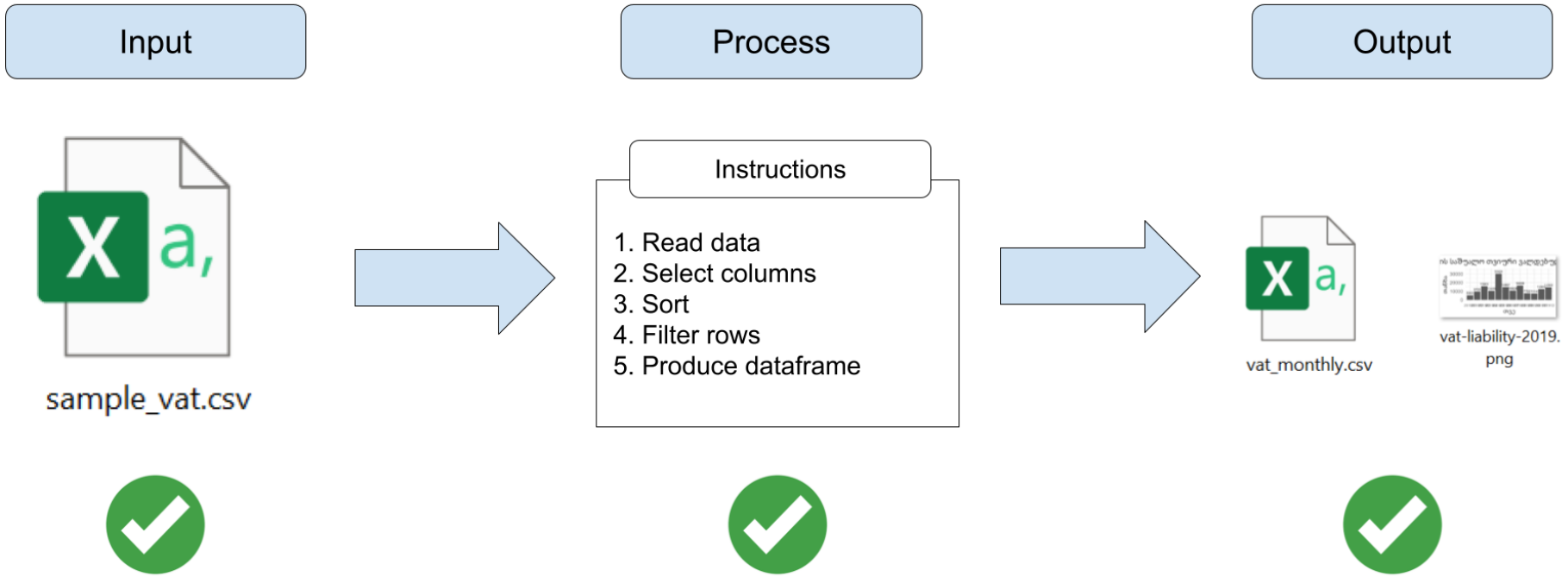
Don't forget to save your work!

- If you haven't, add code comments with `#` to differentiate your solutions for each exercise
- Click the floppy disk to save your work
- Make sure to remember where you're saving your file



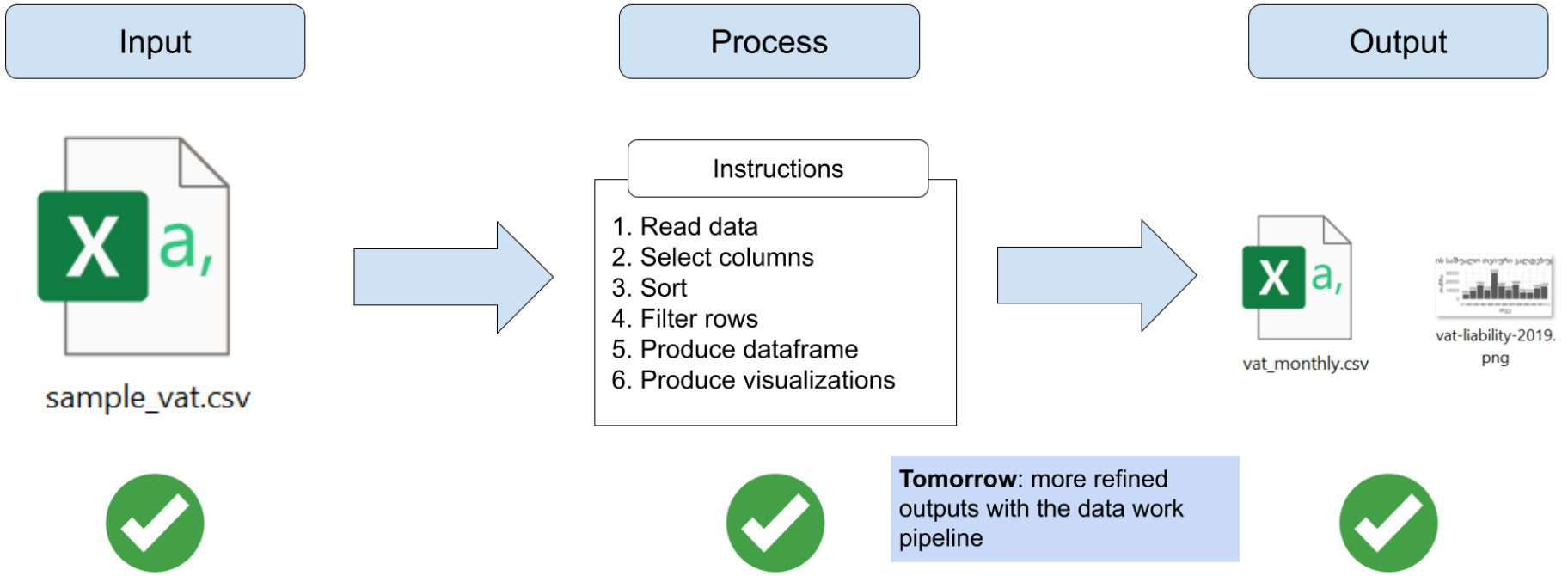
Wrapping up // შეჯამება

Data work pipeline



Wrapping up // შეჯამება

Data work pipeline



Appendix // დანართი

Collapsing

Scenario: Imagine you're approached with the following request:

"We're putting together a report where we want to include the total and average income of small businesses by region in 2019. Can you calculate these numbers? There is data for small businesses income in 2019 you can use for this."

Appendix // დანართი

Collapsing

"There is data for small businesses income in 2019 you can use for this"

"we want to include the total and average income of small businesses by region in 2019"

Collapsing

The data wrangling here involves a number of operations:

1. Keeping only the relevant columns and dropping everything else
2. Grouping the dataframe by region
3. Calculating the total income by region
4. Calculating the mean income by region

The operation of transforming a dataframe with a lower level of observations (exm: firms) to an aggregated level (exm: regions) is called **collapsing**.

Appendix // დანართი

1. Keeping only relevant columns

Use `select()` for this:

```
temp1 <- select(small_business_2019, region, income)
```

2. Grouping by region

Use `group_by()`:

```
temp2 <- group_by(temp1, region)
```

3. Calculating aggregated columns: total and average income

Use `summarize()` combined with `sum()` and `mean()`:

```
region_df <- summarize(temp2,
              total = sum(income),
              average = mean(income))
```

Exercise: Collapse your data

Now we can write code to execute the data collapsing.

1. Columns selection: `temp1 <- select(small_business_2019, region, income)`
2. Grouping by region: `temp2 <- group_by(temp1, region)`
3. Calculating the total and average by month:

```
region_df <- summarize(temp2,
              total = sum(income),
              average = mean(income))
```

Some notes:

- Once again, remember that these are all functions from `dplyr`
- There is a line break and tabulation space between each argument of `summarize()`. We use this for code clarity, R ignores line breaks and space when they are used between function arguments

Appendix // დანართი

Collapsing

You can check your result with `View(region_df)`. Now your data as collapsed as you needed!