

Session 2 - Implementing the data work pipeline in R

R training - Georgia RS-WB DIME

Marc-Andrea Fiorina, Luis Eduardo San Martin

The World Bank | [WB Github](#)

April 2023

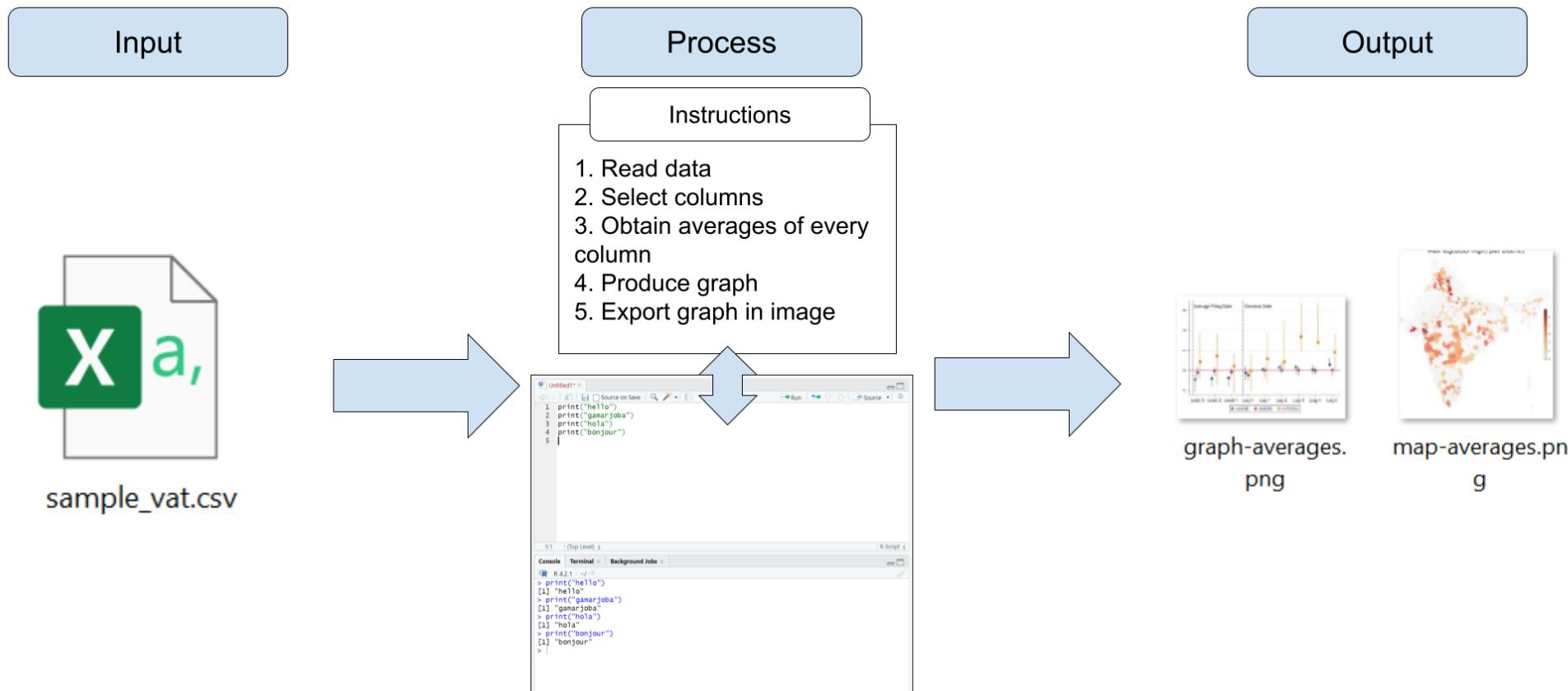


Table of contents // სარჩევი

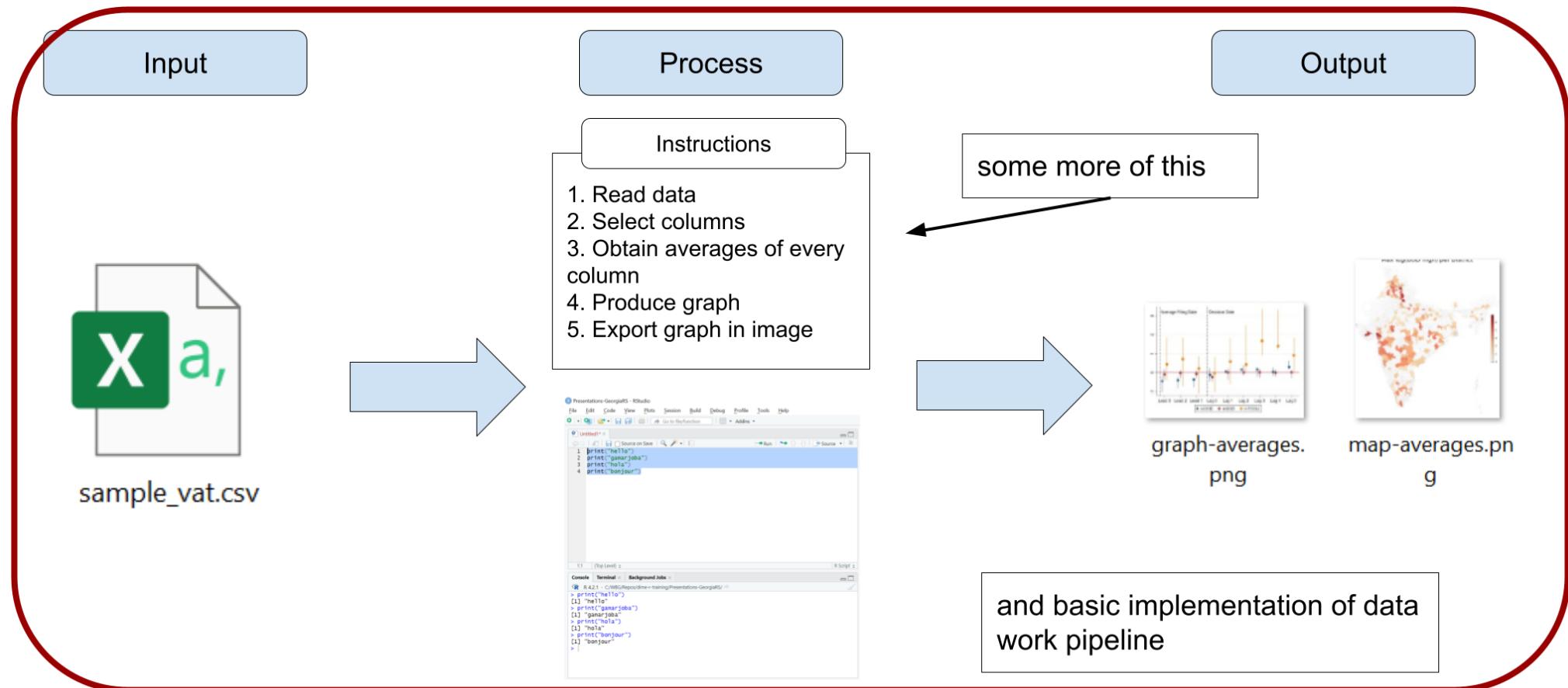
1. About this session
2. R libraries
3. Data wrangling
4. Exporting outputs
5. Data visualization
6. Wrapping up

About this session // սԺ ԱյլօօԸ Թյատրոն

About this session // սօ Այլօօն ՋյօՏՏօջ



About this session // სა სესიონის შესახებ



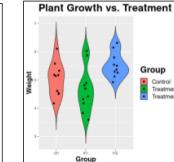
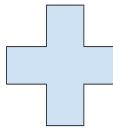
R libraries // R ბიბლიოთეკები

R libraries // R ბიბლიოთეკები

- 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 ბიბლიოთეკები

In a nutshell:



Basic R

Libraries

Enhanced R capabilities

R libraries // R ბიბლიოთეკები

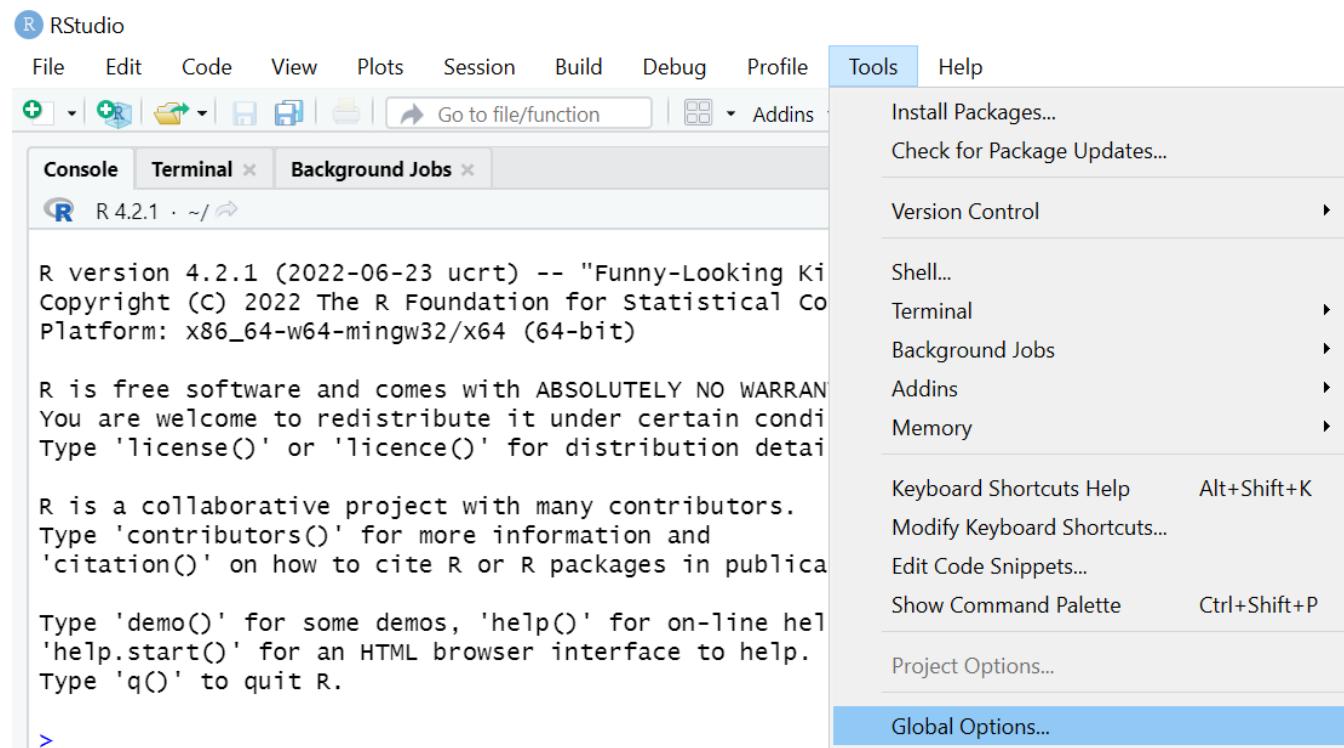
Installing R libraries

- 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

R libraries // R ბიბლიოთეკები

Exercise 1: Setting up the installation of libraries

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



R libraries // R ბიბლიოთეკები

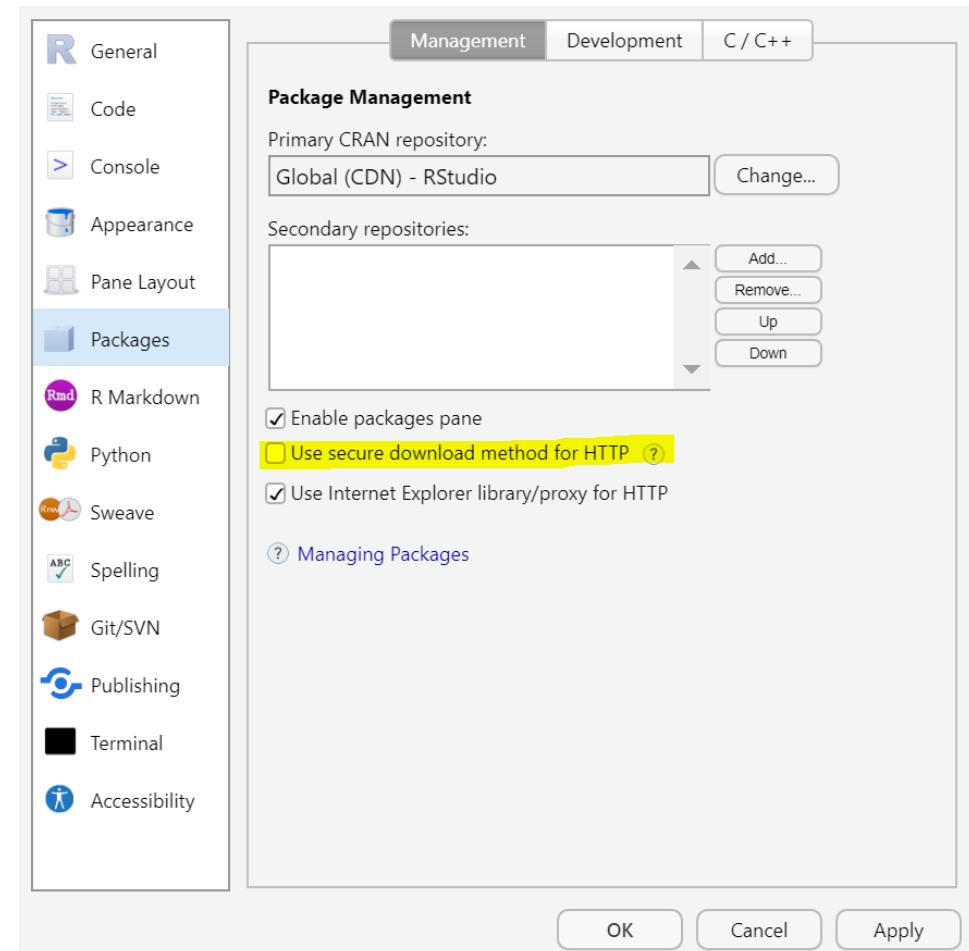
Exercise 1: Setting up the installation of libraries

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.



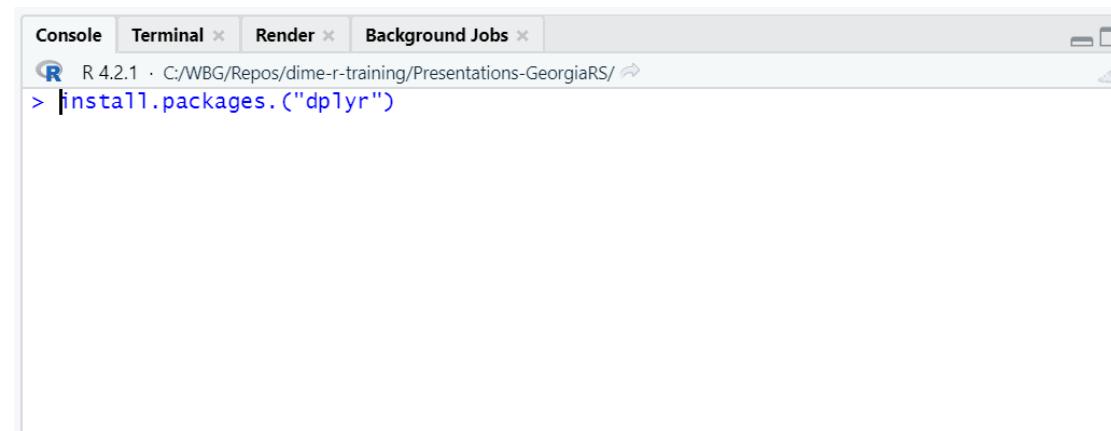
R libraries // R ბიბლიოთეკები

We'll use two libraries in today's session: `dplyr` and `ggplot2`

Exercise 2: Installing libraries

1. Install the libraries by using `install.packages()`

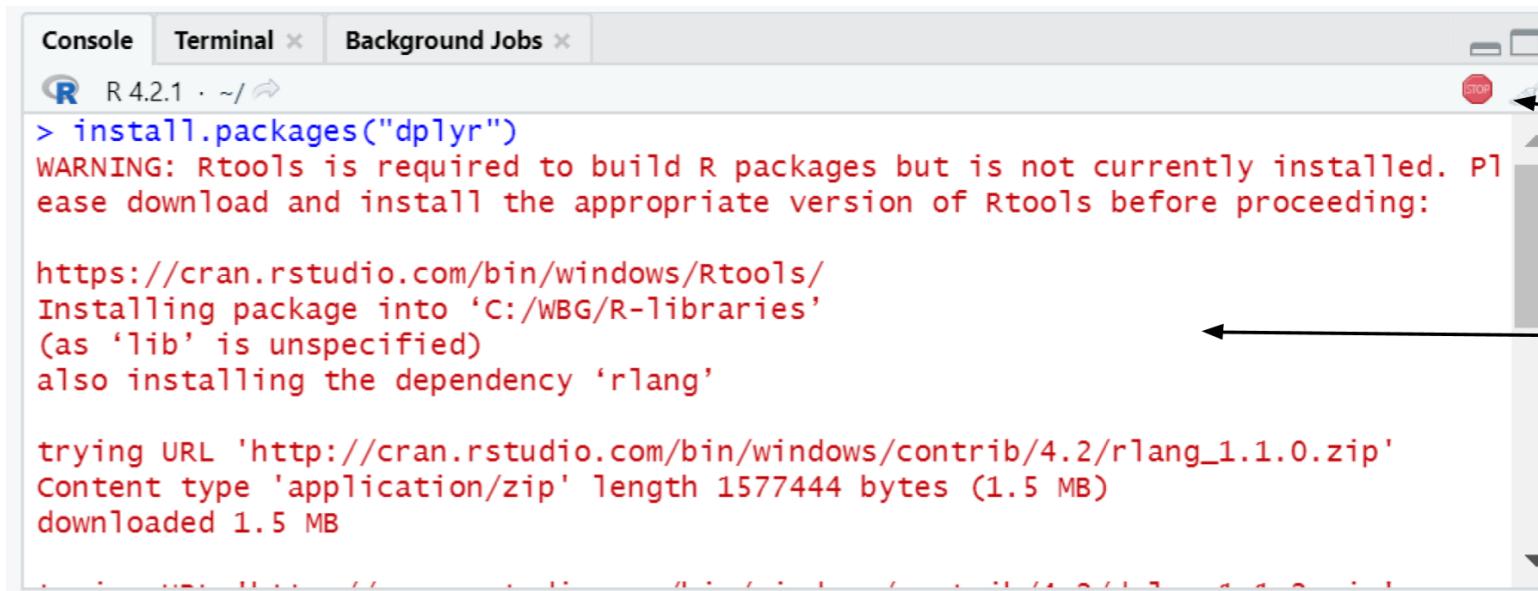
- o `install.packages("dplyr")`
- o `install.packages("ggplot2")`
- o Note the quotes (" ") in the packages names
- o **Introduce this code in the console**, not the script panel



A screenshot of the RStudio interface showing the Console tab active. The title bar includes "Console", "Terminal", "Render", and "Background Jobs". Below the title bar, the R logo and the text "R 4.2.1 · C:/WBG/Repos/dime-r-training/Presentations-GeorgiaRS/" are visible. In the main console area, the command `> install.packages("dplyr")` is typed in blue, indicating it is a user input.

R libraries // R ბიბლიოთეკები

Installing libraries



The screenshot shows the RStudio interface with the 'Console' tab selected. The R version is 4.2.1. The user has run the command `> install.packages("dplyr")`. A red warning message appears: `WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:`. Below this, the package is being installed into the directory `'C:/WBG/R-libraries'`. The message indicates that `'rlang'` is also being installed as a dependency. The download URL is shown as `trying URL 'http://cran.rstudio.com/bin/windows/contrib/4.2/rlang_1.1.0.zip'`. The content type is `application/zip`, the length is `1577444 bytes (1.5 MB)`, and it has been downloaded successfully.

```
R 4.2.1 · ~/R
> install.packages("dplyr")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/WBG/R-libraries'
(as 'lib' is unspecified)
also installing the dependency 'rlang'

trying URL 'http://cran.rstudio.com/bin/windows/contrib/4.2/rlang_1.1.0.zip'
Content type 'application/zip' length 1577444 bytes (1.5 MB)
downloaded 1.5 MB
```

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

Note that this message is not an error

Remember to do this for the installation of `ggplot2` as well.

R libraries // R ბიბლიოთეკები

Now that `dplyr` and `ggplot2` are installed, we only need to load them to start using the functions they have.

Exercise 3: Loading libraries

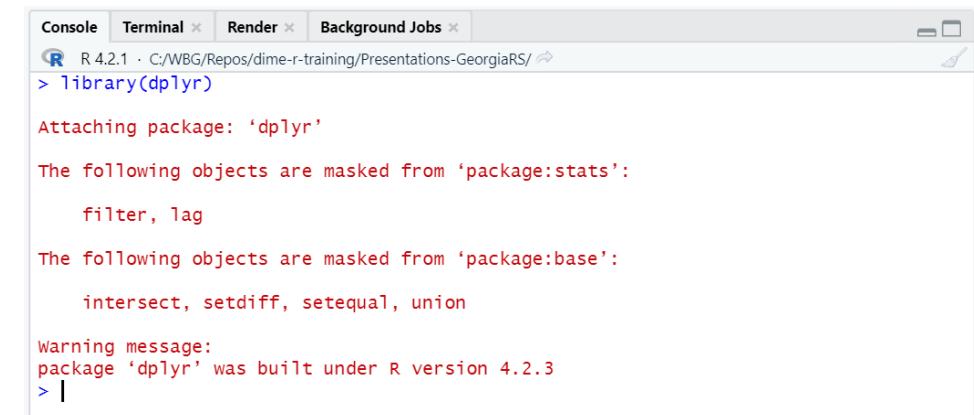
1. Open a new script with `File` >> `New File` >> `R`

`Script`

2. Load `dplyr` with: `library(dplyr)`

3. Load `ggplot2` with `library(ggplot2)`

- Run this code from the new script you just opened
- Notice that we don't use quotes in the library names this time



The screenshot shows the RStudio interface with the 'Console' tab selected. The console window displays the following text:

```
R 4.2.1 · C:/WBG/Repos/dime-r-training/Presentations-GeorgiaRS/
> library(dplyr)
Attaching package: 'dplyr'

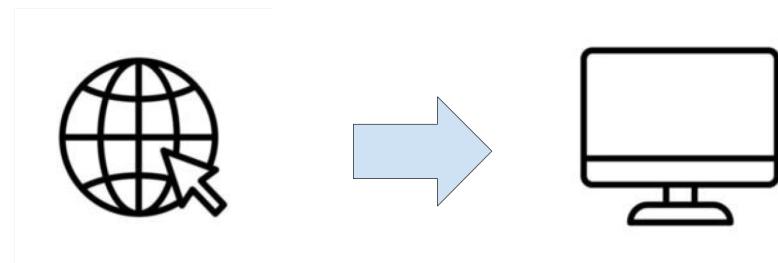
The following objects are masked from 'package:stats':
  filter, lag

The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union

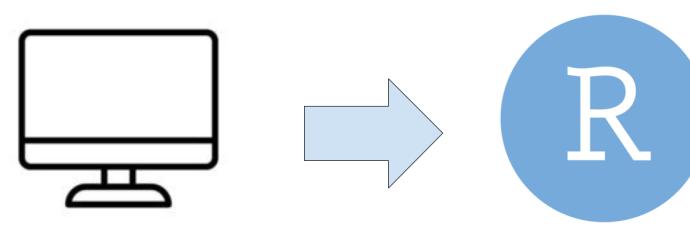
Warning message:
  package 'dplyr' was built under R version 4.2.3
> |
```

R libraries // R ბიბლიოთეკები

- Library installation:



- Library loading:



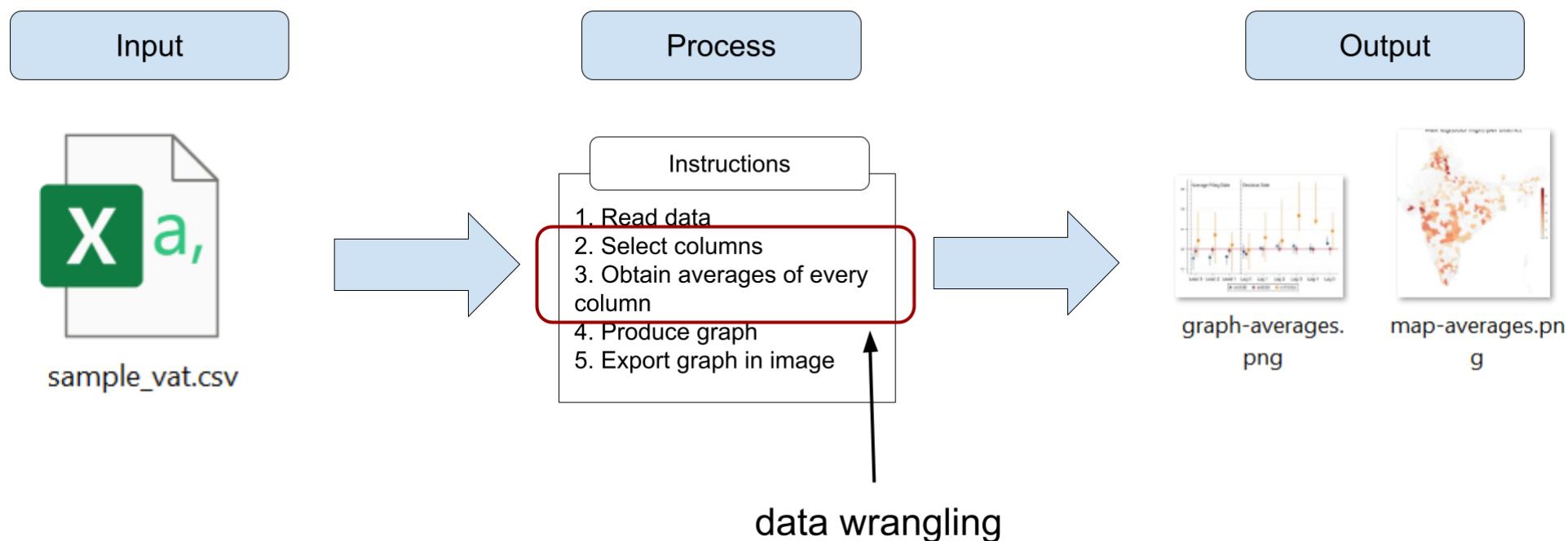
- You install R libraries only once in your computer
- You load libraries every time you open a new RStudio window (only load the libraries you will use)

Data wrangling // የወጪናመጥ኏ የሚሸፍበት

Data wrangling // የመናገድዎችን አጭዣ

Getting your data ready

- Most of times in data work, your data inputs will not be ready to be converted into outputs
- 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**



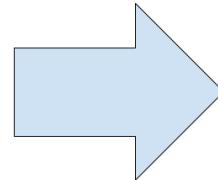
Data wrangling // የመናገዱዎችን አጭዣ

Getting your data ready

- Data wrangling is one of the most crucial aspects of data work
- It involves not only the coding aspect, 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

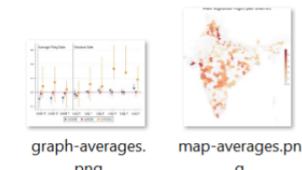
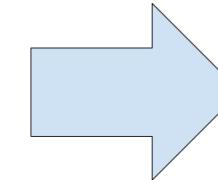
	Modified.ID	TaxPeriod	turnover_taxable18	turnover_exempt	turnover_exports	vat_liability
1	22172	201901	13444.64	0.00	0	2420.04
2	4592952	201906	0.00	0.00	0	0.00
3	10046564	201905	13237.30	0.00	0	2382.71
4	4797328	201912	184477.88	0.00	0	33206.02
5	5889624	201904	0.00	0.00	0	0.00
6	12233616	201912	0.00	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

initial dataframe



	Modified.ID	TaxPeriod	turnover_taxable18	turnover_exempt	turnover_exports	vat_liability
1	22172					2420.04
2	4592952					0.00
3	10046564					2382.71
4	4797328					33206.02
5	5889624					0.00
6	12233616					0.00
7	6248240					435.05
8	538420					1276.29
9	10072288					1447.06

dataframe needed for outputs



outputs

Data wrangling // მონაცემთა ჩბუჯი

Getting your data ready

- We'll use `dplyr` for data wrangling in this training
- You can also use basic R, but we recommend `dplyr` because its functions are easier to use



dplyr

Data wrangling // የመናገዱዎች ምንምዶ

Exercise 4: Loading data

Note that this is the same exercise we did yesterday. If you can't locate your data file, download it from here:
<https://osf.io/download/ds5w4>

1. In RStudio, go to **File** > **Import Dataset** > **From Text (base)** and select the file **sample_vat.csv**

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

2. Make sure to select **Heading** > **Yes** in the next window

3. Select **Import**

The screenshot shows the 'Import Dataset' dialog in RStudio. The 'Name' field is set to 'sample_vat'. The 'Input File' dropdown shows the path to 'sample_vat.csv'. The 'Encoding' is set to 'Automatic'. The 'Heading' option is selected, with 'Yes' being the chosen radio button. Other settings include 'Row names: Automatic', 'Separator: Comma', 'Decimal: Period', 'Quote: Double ("')', and 'Comment: None'. The 'na.strings' field contains 'NA'. A checkbox for 'Strings as factors' is unchecked. Below the dialog, a preview of the 'Data Frame' is shown, displaying the contents of the CSV file. At the bottom right of the dialog are 'Import' and 'Cancel' buttons.

Modified_ID	TaxPeriod	turnover_taxable18	turnover_exempt
22172	201901	13444.64	0
4592952	201906	0.00	0
10046564	201905	13237.30	0
4797328	201912	184477.88	0
5889624	201904	0.00	0
12233616	201912	0.00	0
6248240	201904	2416.93	0
538420	201905	7090.50	0
10072288	201905	8039.20	0
3447376	201911	0.00	0
6392344	201902	0.00	0
5545240	201903	0.00	0
2926244	201912	0.00	0
10221732	201912	857553.97	0
11341060	201911	0.00	0
5646892	201909	40621.20	0

Data wrangling // የመናገዱዎችን ክፍያዎች

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

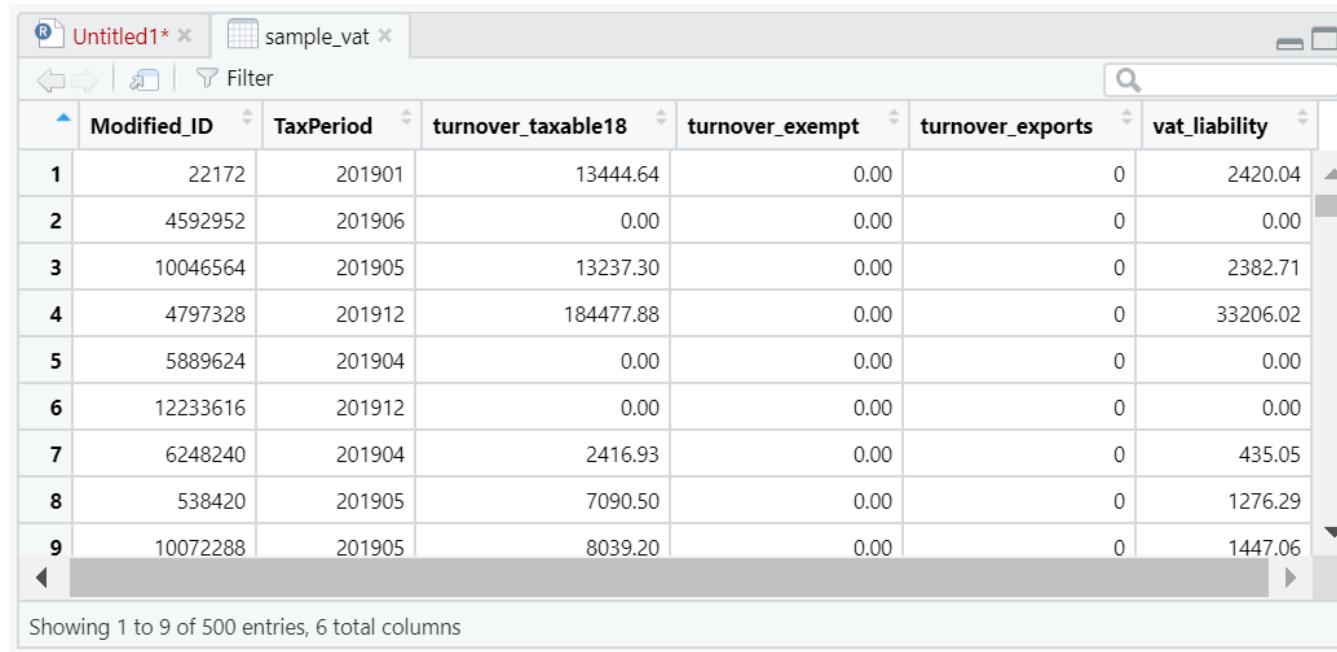
```
sample_vat <- read.csv("C:/Users/wb532468/Downloads/sample_vat.csv")
```

- 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

Data wrangling // የመናገዱዎችን ክፍያዎች

Recap: knowing your data

- The dataframe name is `sample_vat`
- Each row is one monthly tax declaration for one taxpayer
- Column `Modified_ID` is a taxpayer identifier
- `TaxPeriod` is a month variable (year + month)
- The rest are tax-related variables that we are going to use to produce outputs



The screenshot shows the RStudio environment with the 'sample_vat' data frame loaded into a grid viewer. The grid viewer has a header row with column names: 'Modified_ID', 'TaxPeriod', 'turnover_taxable18', 'turnover_exempt', 'turnover_exports', and 'vat_liability'. Below the header are 9 data rows, each containing values for these columns. Row 1 has values: 22172, 201901, 13444.64, 0.00, 0, 2420.04. Row 9 has values: 10072288, 201905, 8039.20, 0.00, 0, 1447.06. The bottom status bar of the grid viewer indicates 'Showing 1 to 9 of 500 entries, 6 total columns'.

	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

Data wrangling // የመናገድዎችን አጭዣ

Data work request

Imagine you're approached with the following request:

"We're putting together a report where we want to include the monthly totals and averages of VAT liability for 2019. Can you calculate these numbers? There is data from monthly tax declarations you can use for this."

Data wrangling // የመናገዱዎችን ክፍያዎች

Data work request

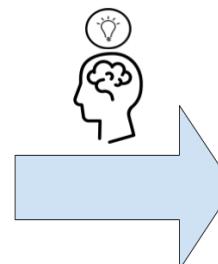
"There is data from monthly tax declarations you can use for this"

"We want to include the monthly totals and averages of VAT liability for 2019"

	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

- Observations: Monthly tax declaration (500) of 2019
- Multiple columns



	TaxPeriod	total_vat_liability	avg_vat_liability
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

Showing 1 to 10 of 12 entries, 3 total columns

- Observations: Months in 2019 (12)
- Columns: month, total VAT liability, average VAT liability

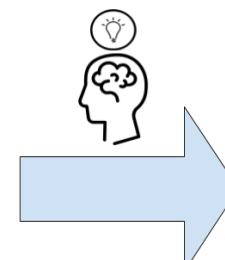
Data wrangling // የመናገዱዎችን ክፍያዎች

Data work request

The data wrangling here involves a number of operations:

1. Keeping only the relevant columns and dropping everything else
2. Collapsing the dataframe by month
3. Calculating the total VAT liability by month
4. Calculating the average VAT liability by month

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



	TaxPeriod	total_vat_liability	avg_vat_liability
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

- Observations: Monthly tax declaration (500) of 2019
- Multiple columns

- Observations: Months in 2019 (12)
- Columns: month, total VAT liability, average VAT liability

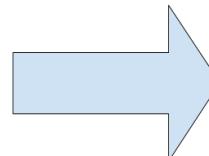
Data wrangling // ዘመናጂዬዥና ምንምዶ

1. Keeping only relevant columns

Use `select()` for this:

```
df_temp1 <- select(sample_vat, TaxPeriod, vat_liability)
```

	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



	TaxPeriod	vat_liability
1	201901	2420.04
2	201906	0.00
3	201905	2382.71
4	201912	33206.02
5	201904	0.00
6	201912	0.00
7	201904	435.05
8	201905	1276.29
9	201905	1447.06
10	201911	0.00

- Observations: Monthly tax declaration (500) of 2019
- Multiple columns

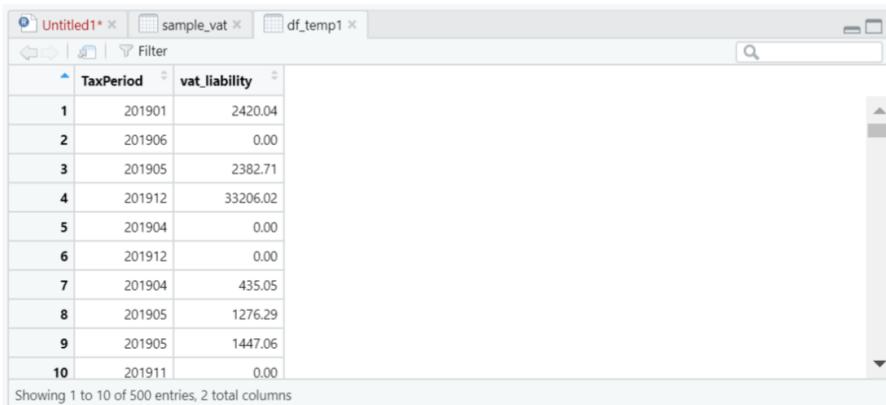
- Observations: Monthly tax declaration (500) of 2019
- Columns: month, VAT liability

Data wrangling // ዘመናጂዬዥና ምንምዶስ

2. Collapsing by month

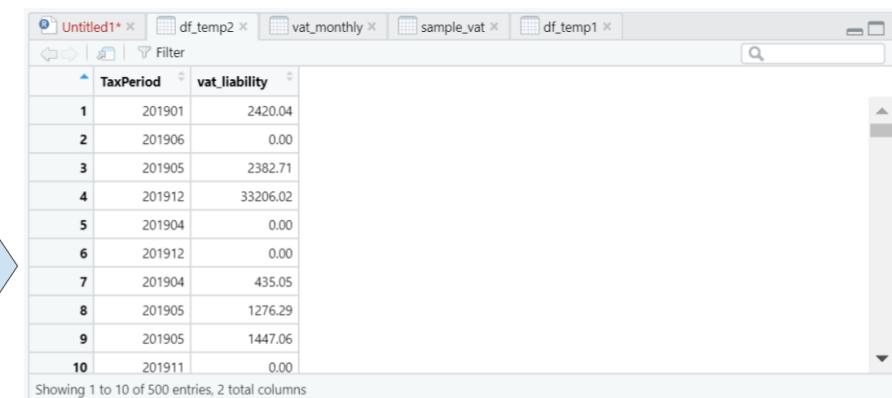
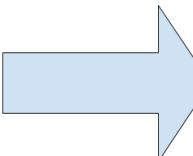
Use `group_by()`:

```
df_temp2 <- group_by(df_temp1, TaxPeriod)
```



	TaxPeriod	vat_liability
1	201901	2420.04
2	201906	0.00
3	201905	2382.71
4	201912	33206.02
5	201904	0.00
6	201912	0.00
7	201904	435.05
8	201905	1276.29
9	201905	1447.06
10	201911	0.00

Showing 1 to 10 of 500 entries, 2 total columns



	TaxPeriod	vat_liability
1	201901	2420.04
2	201906	0.00
3	201905	2382.71
4	201912	33206.02
5	201904	0.00
6	201912	0.00
7	201904	435.05
8	201905	1276.29
9	201905	1447.06
10	201911	0.00

Showing 1 to 10 of 500 entries, 2 total columns

- Observations: Monthly tax declaration (500) of 2019
- Columns: month, VAT liability

- Observations: Monthly tax declaration (500) of 2019
- Columns: month, VAT liability
- Dataframe looks similar because we haven't added the aggregated estimates yet, **but it's not the same**

Data wrangling // ዘመናጂዬዥና ምንምዶ

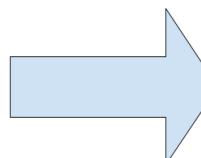
3. Calculating aggregated columns: total and average VAT liability

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

```
vat_monthly <- summarize(df_temp2,
                           total = sum(vat_liability),
                           average = mean(vat_liability))
```

	TaxPeriod	vat_liability
1	201901	2420.04
2	201906	0.00
3	201905	2382.71
4	201912	33206.02
5	201904	0.00
6	201912	0.00
7	201904	435.05
8	201905	1276.29
9	201905	1447.06
10	201911	0.00

Showing 1 to 10 of 500 entries, 2 total columns



	TaxPeriod	total	average
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

Showing 1 to 10 of 12 entries, 3 total columns

- Observations: Monthly tax declaration (500) of 2019
- Columns: month, VAT liability
- Remember that this is a grouped dataframe waiting for the creation of aggregated columns

- Observations: Months in 2019
- Columns: month, total VAT liability, average VAT liability (aggregated columns)

Data wrangling // የመናገዱዎችን ክፍያዎች

Exercise 5: Wrangle your data

Now that we figured out the shape we need the dataframe to have, we can write code to execute the data wrangling.

1. Columns selection: `df_temp1 <- select(sample_vat, TaxPeriod, vat_liability)`
2. Collapsing by month: `df_temp2 <- group_by(df_temp1, TaxPeriod)`
3. Calculating the total and average by month:

```
vat_monthly <- summarize(df_temp2,
                           total = sum(vat_liability),
                           average = mean(vat_liability))
```

Some notes:

- These are all functions from `dplyr`. Remember you have to have loaded `dplyr` first with `library(dplyr)` for this to work
- There is a line break and tabulation between each argument of `summarize()`. We use this for code clarity, R ignores line breaks when they are used between function arguments

Data wrangling // ዘመናጂዬዎች ክፍያዎች

Some more notes:

- We're creating two intermediate results named `df_temp1` and `df_temp2` in the process
 - Note that `df_temp1` is the input of the collapsing and `df_temp2` is the result
 - This can be avoided by using something called the pipes operator (`%>%`)
 - We're not covering pipes in this training, but you should know they are commonly used in work with `dplyr`
- The result is the dataframe `vat_monthly`

The screenshot shows the RStudio interface with the Environment tab selected. The Global Environment pane lists four data frames:

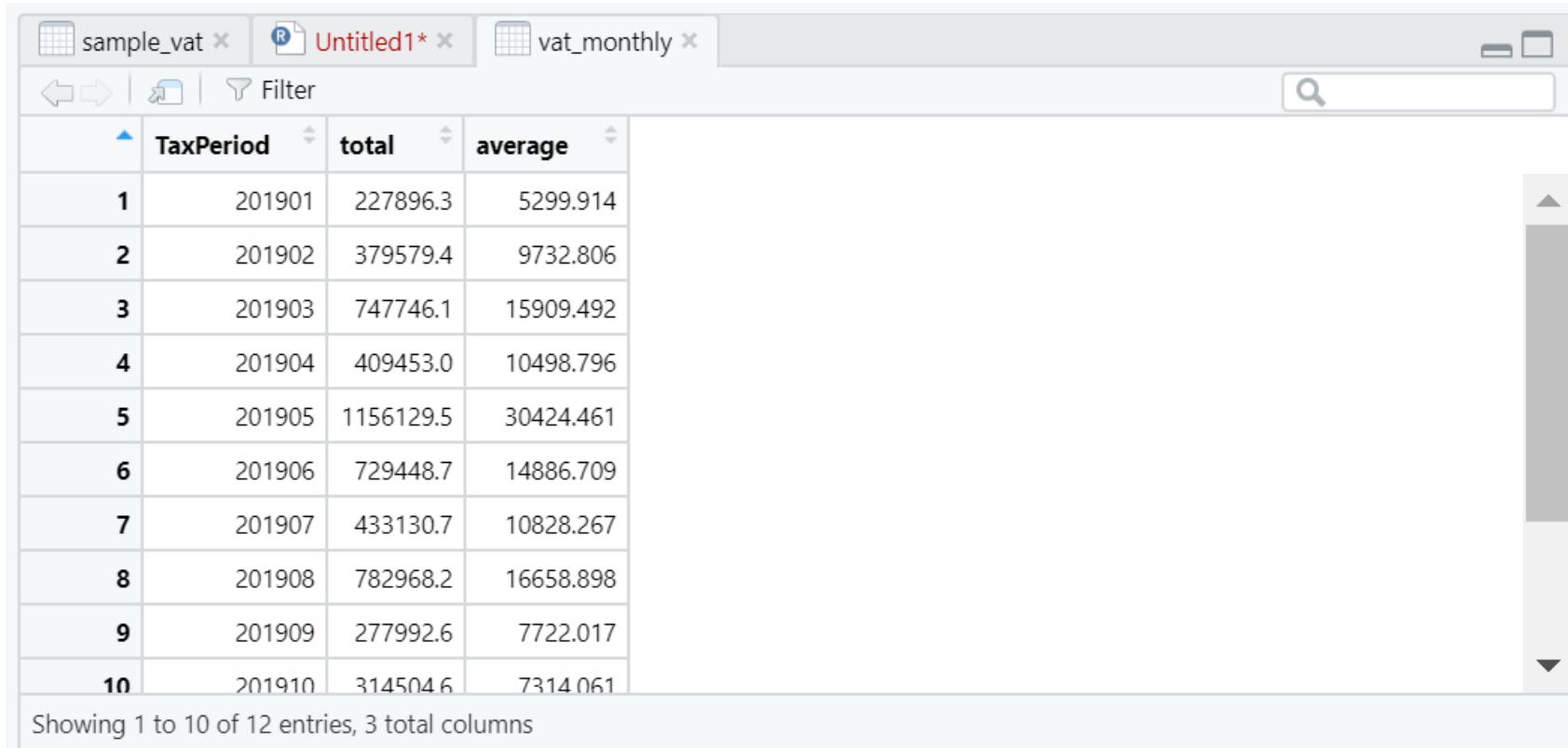
Object	Description
<code>df_temp1</code>	500 obs. of 2 variables
<code>df_temp2</code>	500 obs. of 2 variables
<code>sample_vat</code>	500 obs. of 6 variables
<code>vat_monthly</code>	12 obs. of 3 variables

Annotations with arrows point to the objects:

- An arrow points from the text "intermediate results" to the row containing `df_temp1` and `df_temp2`.
- An arrow points from the text "final result" to the row containing `vat_monthly`.

Data wrangling // ዘመናጂዬዥና ምንምዶ

You can check your result with `View(vat_monthly)`. Now your dataframe is wrangled!



	TaxPeriod	total	average
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

Showing 1 to 10 of 12 entries, 3 total columns

Data wrangling // የመናገዱዎችን ክፍያዎች

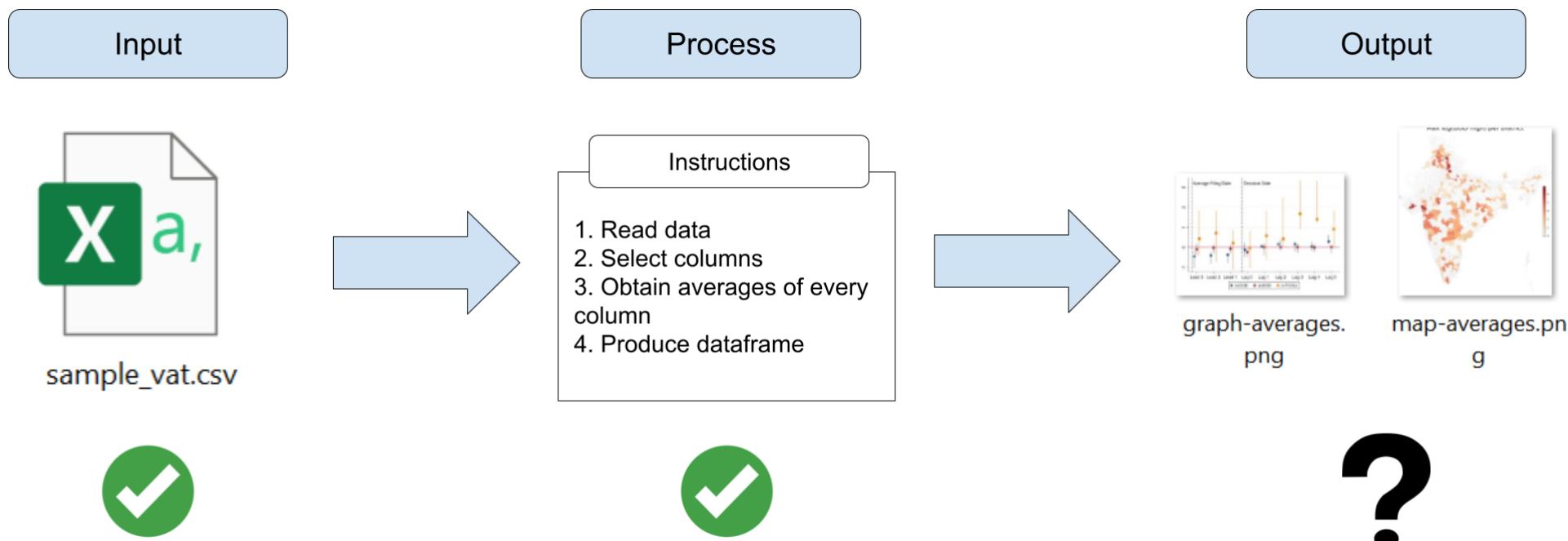
- This was a simple data wrangling example that we chose for convenience
- Data wrangling can involve lots of operations

Operation	Function in <code>dplyr</code>
Subset columns	<code>select()</code>
Subset rows (based on condition)	<code>filter()</code>
Create new columns	<code>mutate()</code>
Create new rows	<code>add_row()</code>
Merge dataframes	<code>inner_join()</code> , <code>left_join()</code> , <code>right_join()</code> , <code>full_join()</code>
Append dataframes	<code>bind_rows()</code>
Deduplicate	<code>distinct()</code>
Collapse and create summary indicators	<code>group_by()</code> , <code>summarize()</code>
Pass a result as the first argument for the next function	<code>%>%</code> (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 dataframe 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 dataframe you want to export
 2. A file path to export the dataframe 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 6: Exporting `vat_monthly`

1. Use this code to export the result dataframe:

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

Exporting outputs // პროდუქციის ექსპორტი

Now `vat_monthly.csv` will show in your computer (probably in your `Documents` folder).

Name	Date modified	Type	Size
📁 data	4/25/2023 5:40 PM	File folder	
📁 img	4/26/2023 7:57 PM	File folder	
📁 libs	4/26/2023 8:00 PM	File folder	
📁 session1_cache	4/24/2023 7:41 PM	File folder	
📁 session2_cache	4/26/2023 3:14 PM	File folder	
📄 .Rhistory	4/26/2023 4:46 PM	RHISTORY File	1 KB
[R] Presentations-GeorgiaRS.Rproj	4/25/2023 2:46 PM	R Project	1 KB
[_] README.md	4/24/2023 4:06 PM	MD File	1 KB
[R] script1.R	4/25/2023 8:39 PM	R File	1 KB
[C] session1.html	4/26/2023 4:54 PM	Chrome HTML Docu...	27 KB
[P] session1.pdf	4/25/2023 8:54 PM	Adobe Acrobat Docu...	2,961 KB
[R] session1.Rmd	4/26/2023 4:45 PM	RMD File	21 KB
[C] session2.html	4/26/2023 8:00 PM	Chrome HTML Docu...	19 KB
[R] session2.Rmd	4/26/2023 8:00 PM	RMD File	13 KB
[_] vat_monthly.csv	4/26/2023 8:08 PM	Microsoft Excel Com...	1 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(vat_monthly,  
          "vat_monthly.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

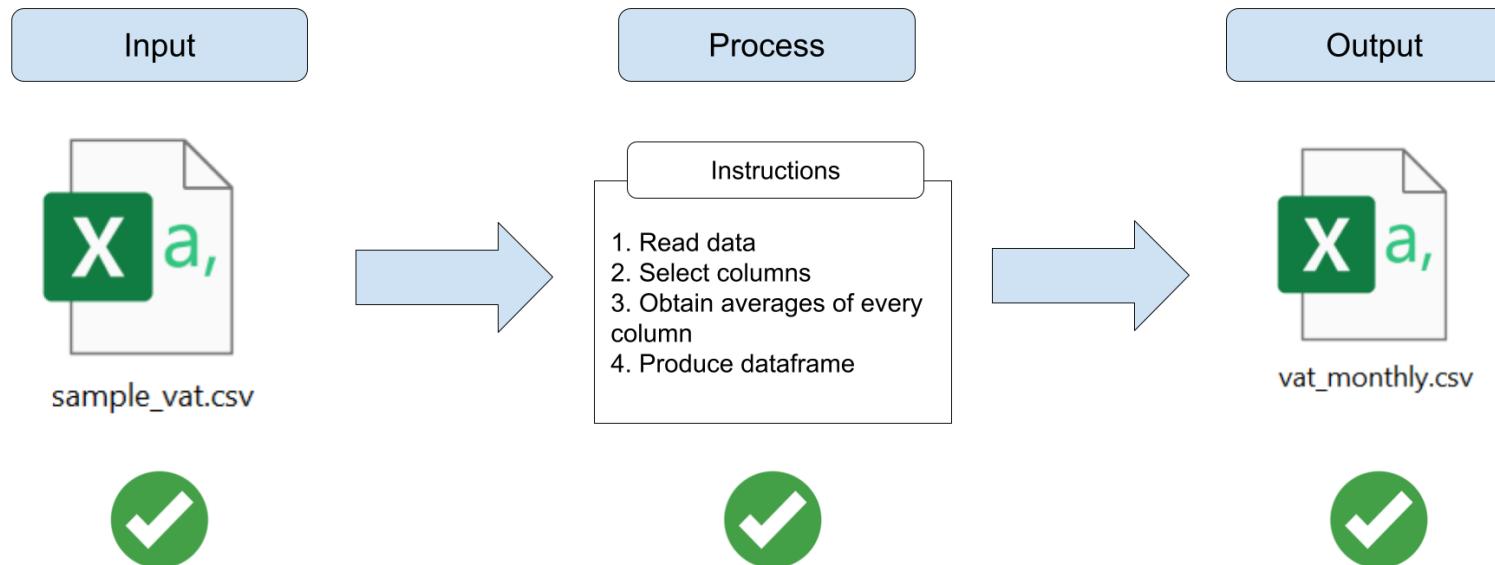
Some notes on file paths

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

- If you only include a file name (as in `vat_monthly.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!

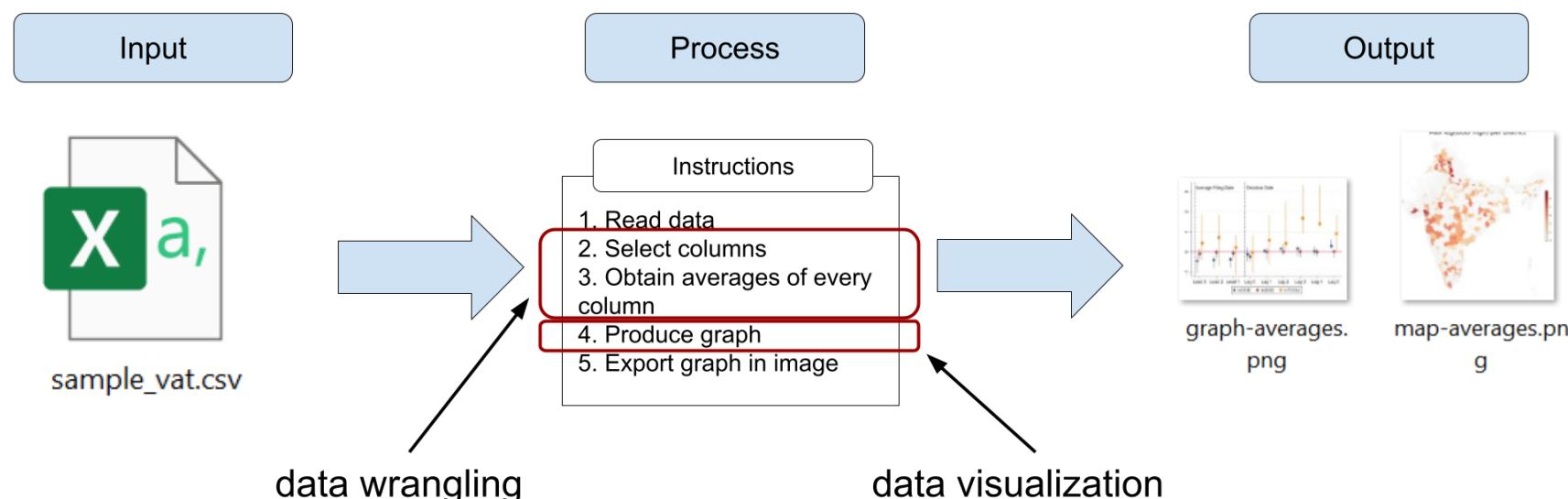


Data visualization // მონაცემთა ვიზუალიზაცია

Data visualization // მონაცემთა ვიზუალიზაცია

Data visualization in the data work pipeline

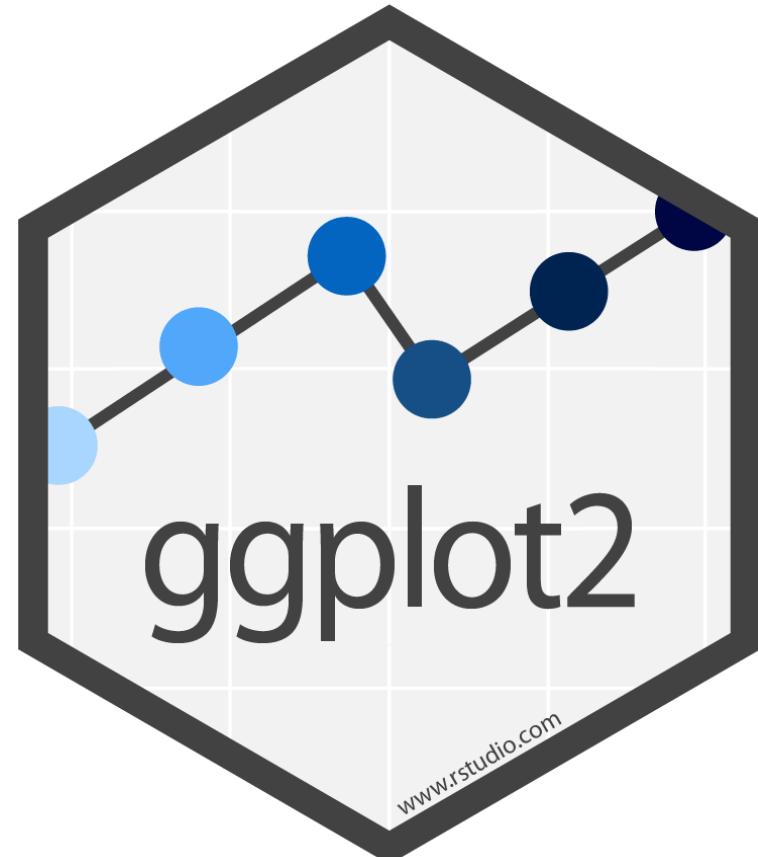
- Compared to producing a table output, data visualization involves an extra step after data wrangling: producing the visualization
- We also use R code to produce data visualizations
- The input for that code is the wrangled dataframe



Data visualization // მონაცემთა ვიზუალიზაცია

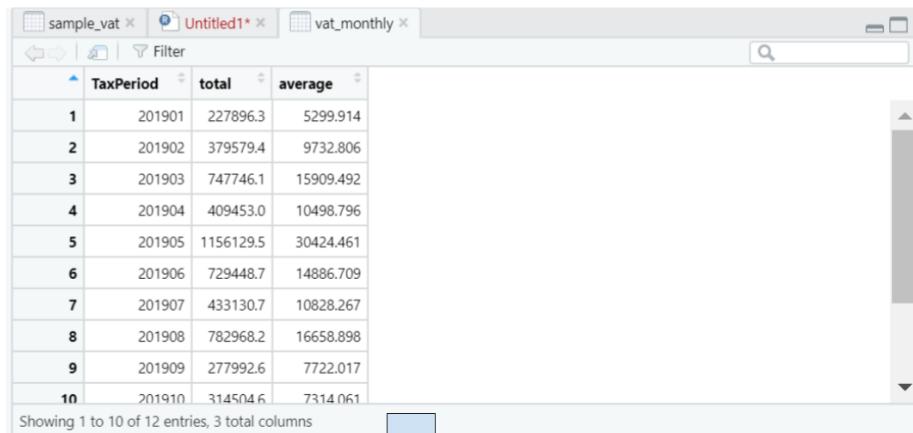
Data visualization in R

- We'll use the package `ggplot2` to create data visualizations
- `ggplot2` greatly facilitates producing plots in R
 - It follows a syntax based on a description of the plot you want to obtain
 - This syntax is called "grammar of graphics", a (somewhat) standard method of data visualization definition in statistical programming



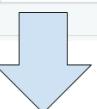
Data visualization // მონაცემთა ვიზუალიზაცია

The grammar of graphics in `ggplot2`

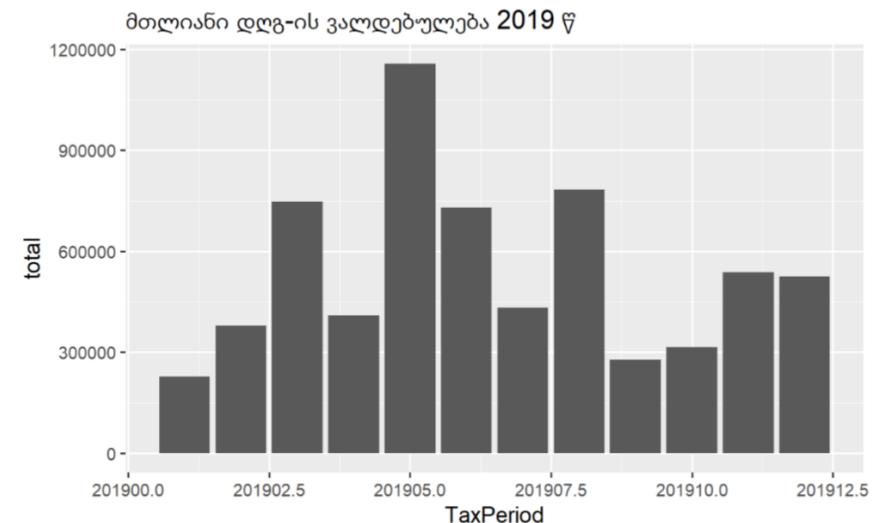


	TaxPeriod	total	average
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

Showing 1 to 10 of 12 entries, 3 total columns

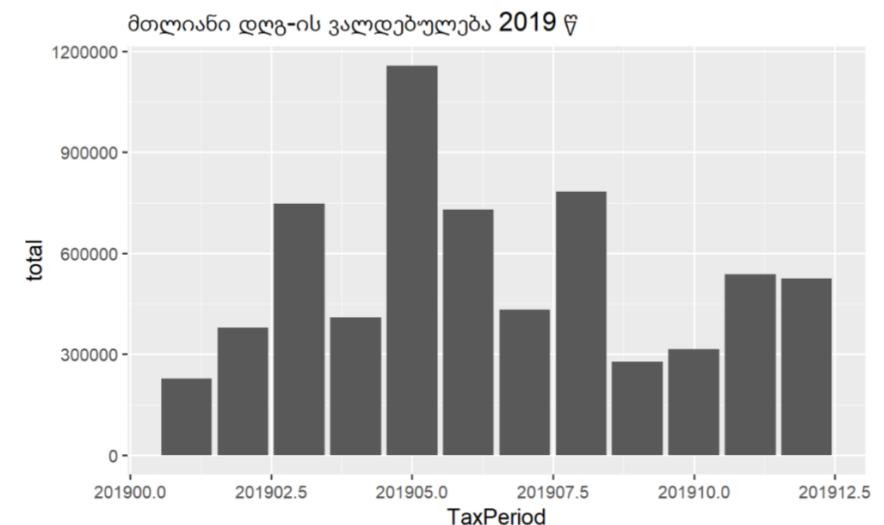
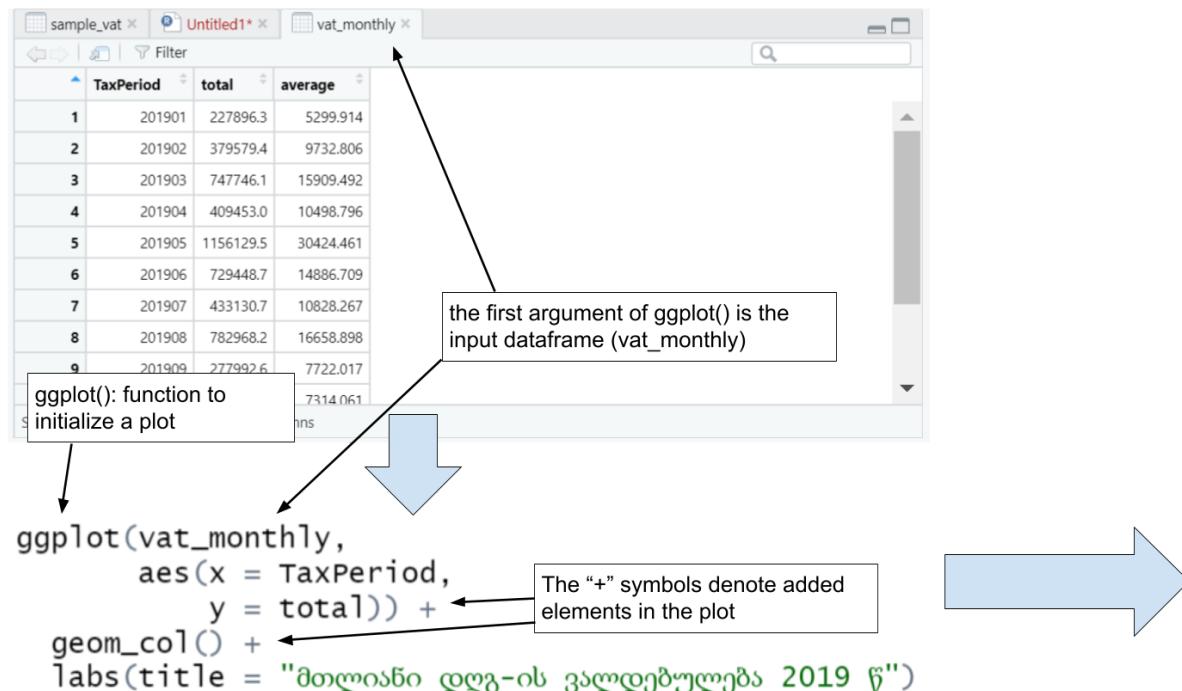


```
ggplot(vat_monthly,  
       aes(x = TaxPeriod,  
            y = total)) +  
  geom_col() +  
  labs(title = "მთლიანი დღგ-ის ვალდებულება 2019 წ")
```



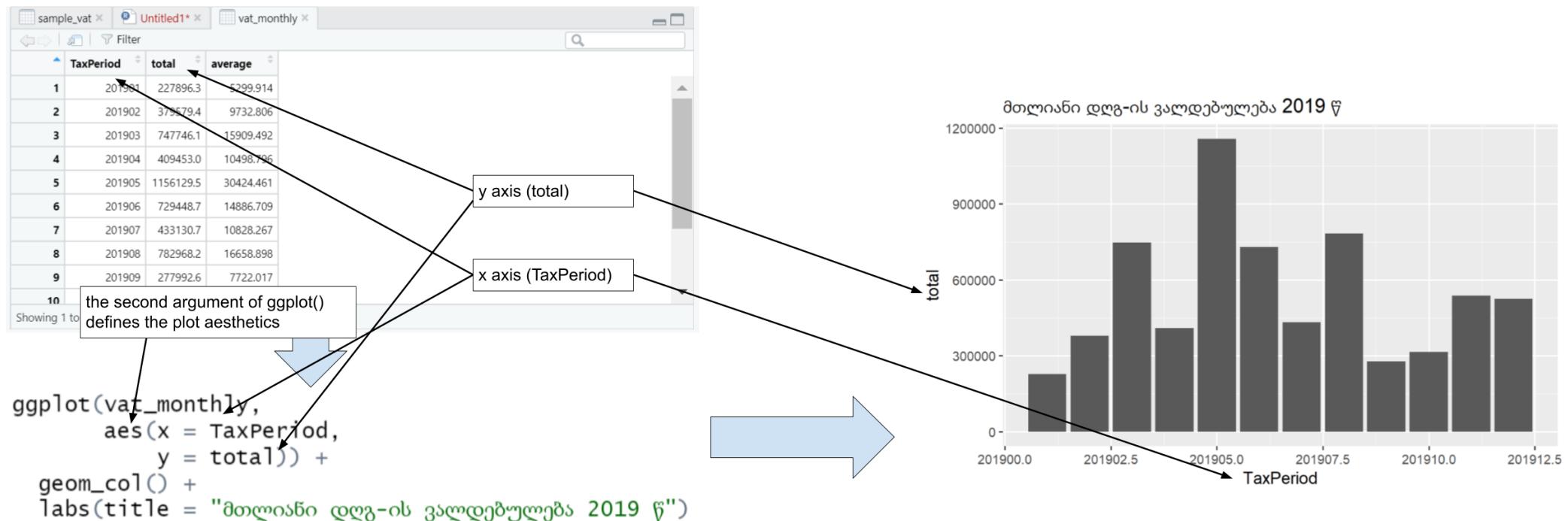
Data visualization // მონაცემთა ვიზუალიზაცია

The grammar of graphics in `ggplot2`



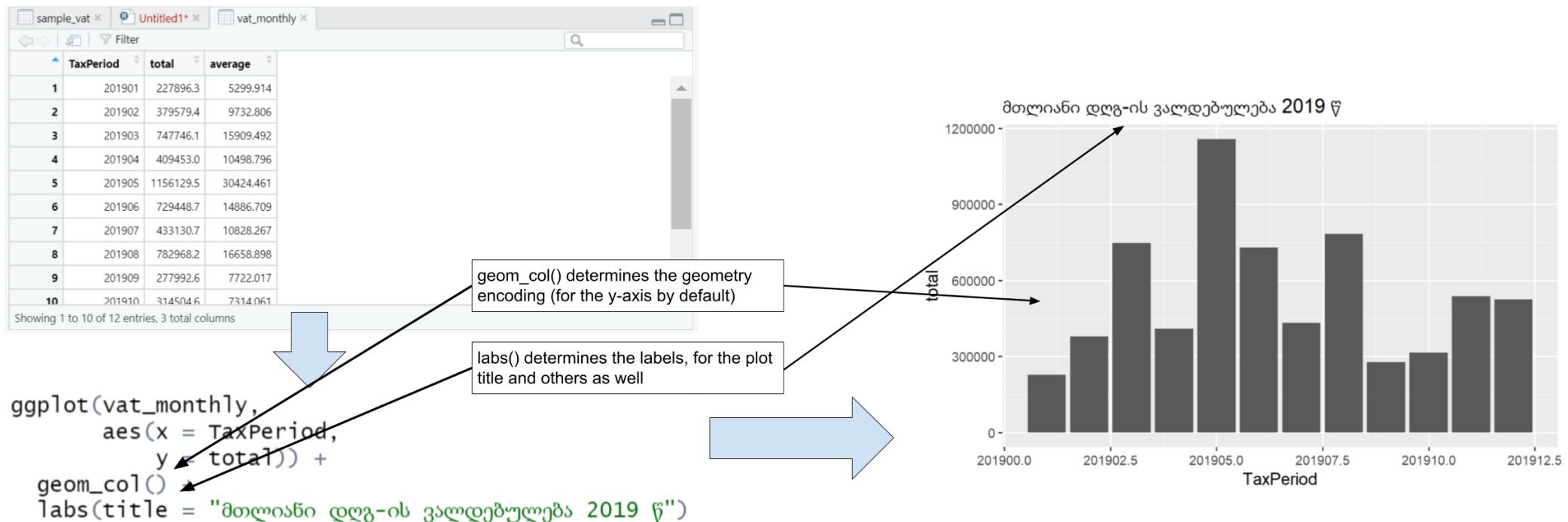
Data visualization // მონაცემთა ვიზუალიზაცია

The grammar of graphics in `ggplot2`



Data visualization // მონაცემთა ვიზუალიზაცია

The grammar of graphics in `ggplot2`



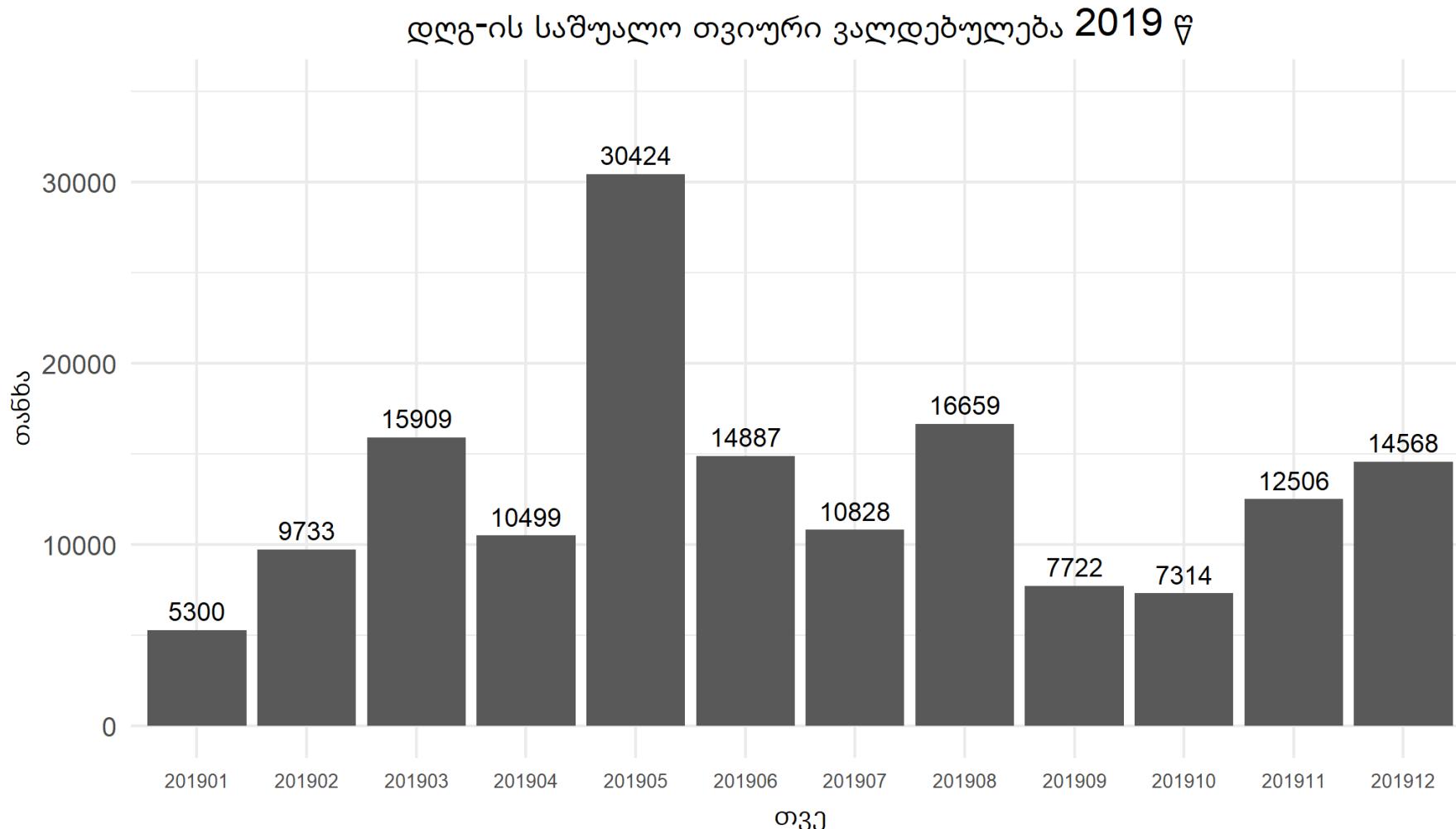
Data visualization // მონაცემთა ვიზუალიზაცია

Exercise 7

Use the dataframe `vat_monthly` to produce a bar plot (with more details) of the average VAT liability by month.

```
ggplot(vat_monthly,
       aes(x = as.factor(TaxPeriod),
           y = average)) +
  geom_col() +
  geom_text(aes(label = round(average)),
            position = position_dodge(width = 1),
            vjust = -0.5,
            size = 3) +
  labs(title = "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ",
       x = "თვე",
       y = "თანხა") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_text(size = 6.5)) +
  ylim(0, 35000)
```

Data visualization // მონაცემთა ვიზუალიზაცია



Data visualization // დონაციონალური ვიზუალიზაცია

When you produce a plot, RStudio will show it in a plot visualizer in the lower-right panel of your window.

The screenshot shows the RStudio IDE with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Addins.
- Code Editor:** Shows R script code for generating a bar chart. The title is "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ".
- Environment Tab:** Lists objects: bar_plot, df_temp1, df_temp2, sample_vat, vat_monthly.
- Plots Tab:** Active tab. Displays a bar chart titled "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ". The y-axis is "მატება" (values 0, 10000, 20000, 30000) and the x-axis is "თვე" (months 201901 to 201912). The data is as follows:

თვე	მატება
201901	5300
201902	9733
201903	15909
201904	10499
201905	30424
201906	14887
201907	10828
201908	16659
201909	7722
201910	7314
201911	12506
201912	14568

Data visualization // მონაცემთა ვიზუალიზაცია

Saving your plot in an output

- We use `ggsave()` to export the plot
- `ggsave()` by default saves the last plot you produced
- Just as `write.csv()`, `ggsave()` can use file paths to specify the location of the exported file

```
ggsave("vat-liability-2019.png")
```

Data visualization // მონაცემთა ვიზუალიზაცია

Saving your plot in an output

Name	Date modified	Type	Size
data	4/26/2023 10:26 PM	File folder	
img	4/26/2023 10:41 PM	File folder	
libs	4/26/2023 10:43 PM	File folder	
session1_cache	4/24/2023 7:41 PM	File folder	
session2_cache	4/26/2023 3:14 PM	File folder	
session2_files	4/26/2023 10:27 PM	File folder	
.Rhistory	4/26/2023 4:46 PM	RHISTORY File	1 KB
Presentations-GeorgiaRS.Rproj	4/25/2023 2:46 PM	R Project	1 KB
README.md	4/24/2023 4:06 PM	MD File	1 KB
script1.R	4/25/2023 8:39 PM	R File	1 KB
session1.html	4/26/2023 4:54 PM	Chrome HTML Docu...	27 KB
session1.pdf	4/25/2023 8:54 PM	Adobe Acrobat Docu...	2,961 KB
session1.Rmd	4/26/2023 4:45 PM	RMD File	21 KB
session2.html	4/26/2023 10:43 PM	Chrome HTML Docu...	25 KB
session2.Rmd	4/26/2023 10:43 PM	RMD File	20 KB
vat_monthly.csv	4/26/2023 8:08 PM	Microsoft Excel Com...	1 KB
vat-liability-2019.png	4/26/2023 10:36 PM	PNG File	50 KB

Data visualization // მონაცემთა ვიზუალიზაცია

Encodings in `ggplot2`

Encodings dictate how the data is represented in data visualizations. In a bar plot, the bars are the encoding. Some of the functions for encodings in `ggplot2` are:

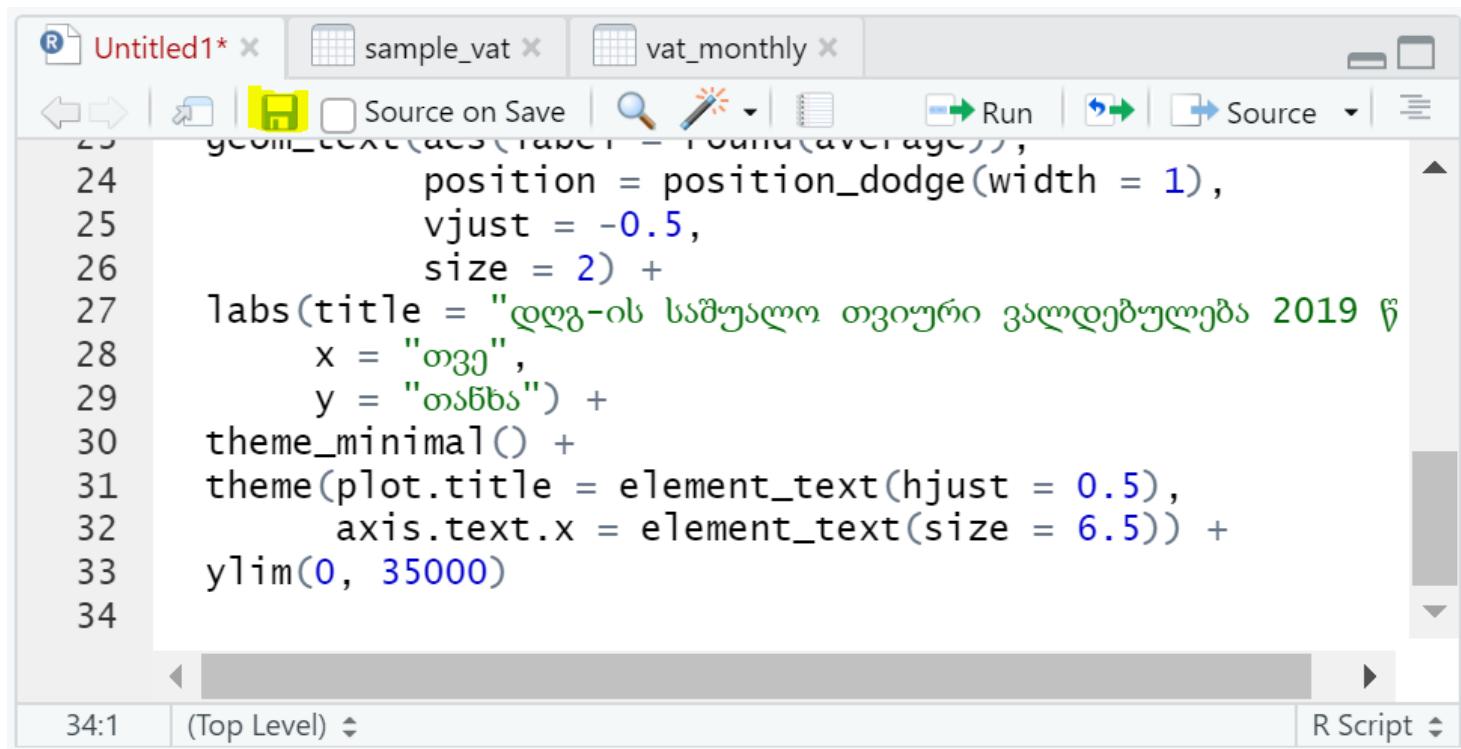
Encoding	Function in <code>ggplot2</code>
Bars	<code>geom_col()</code>
Lines	<code>geom_line()</code>
Points (scatterplot)	<code>geom_point()</code>
Area	<code>geom_area()</code>
Histogram	<code>geom_histogram()</code>
Floating labels (texts)	<code>geom_text()</code>
Box plot	<code>geom_boxplot()</code>
Pie chart	<code>geom_bar() + coord_polar()</code>
Smoothed line	<code>geom_smooth()</code>

Wrapping up // പിന്തുംതുടങ്ങാ

Wrapping up // დევილოვა

Don't forget to 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



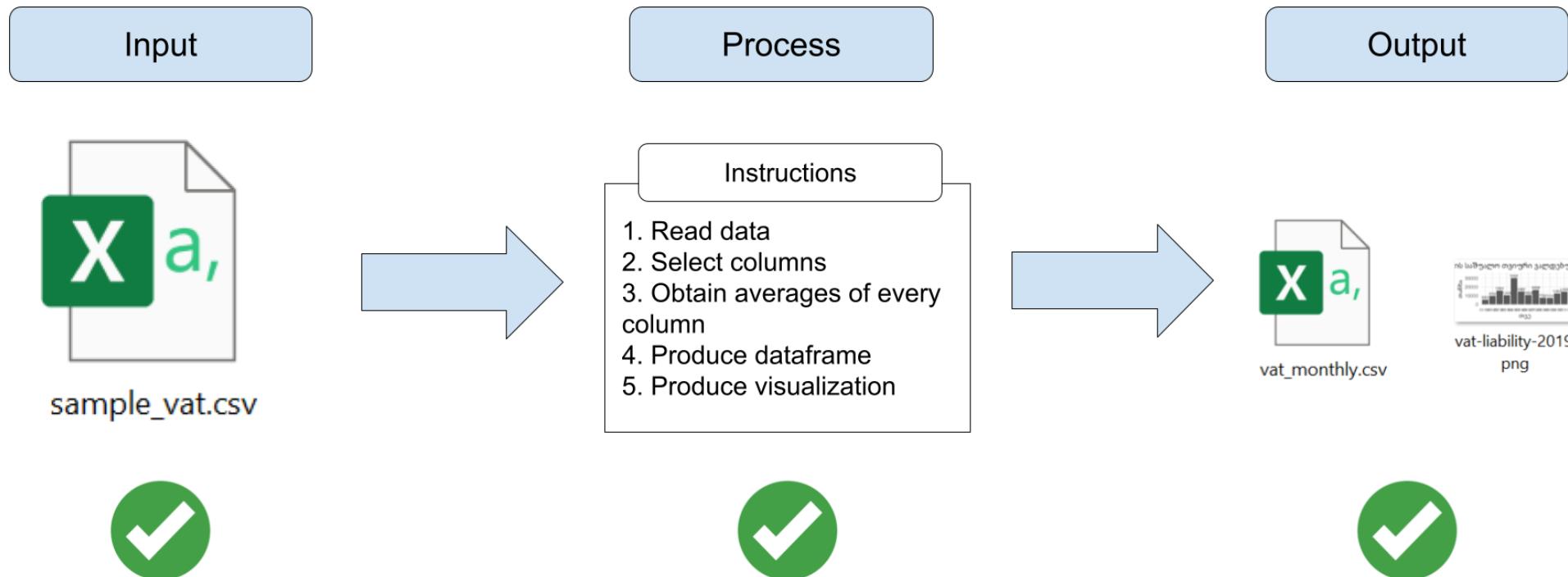
The screenshot shows the RStudio interface with an R script open. The script contains the following code:

```
23     geom_bar(stat = "identity"),
24     position = position_dodge(width = 1),
25     vjust = -0.5,
26     size = 2) +
27   labs(title = "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ",
28       x = "თვე",
29       y = "თანხმა") +
30   theme_minimal() +
31   theme(plot.title = element_text(hjust = 0.5),
32         axis.text.x = element_text(size = 6.5)) +
33   ylim(0, 35000)
34
```

The code is for creating a bar chart with dodged bars, setting the title to "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ", and adjusting the x-axis to represent months ("თვე") and the y-axis to represent values ("თანხმა"). The plot is styled with a minimalist theme and a y-axis limit of 0 to 35,000.

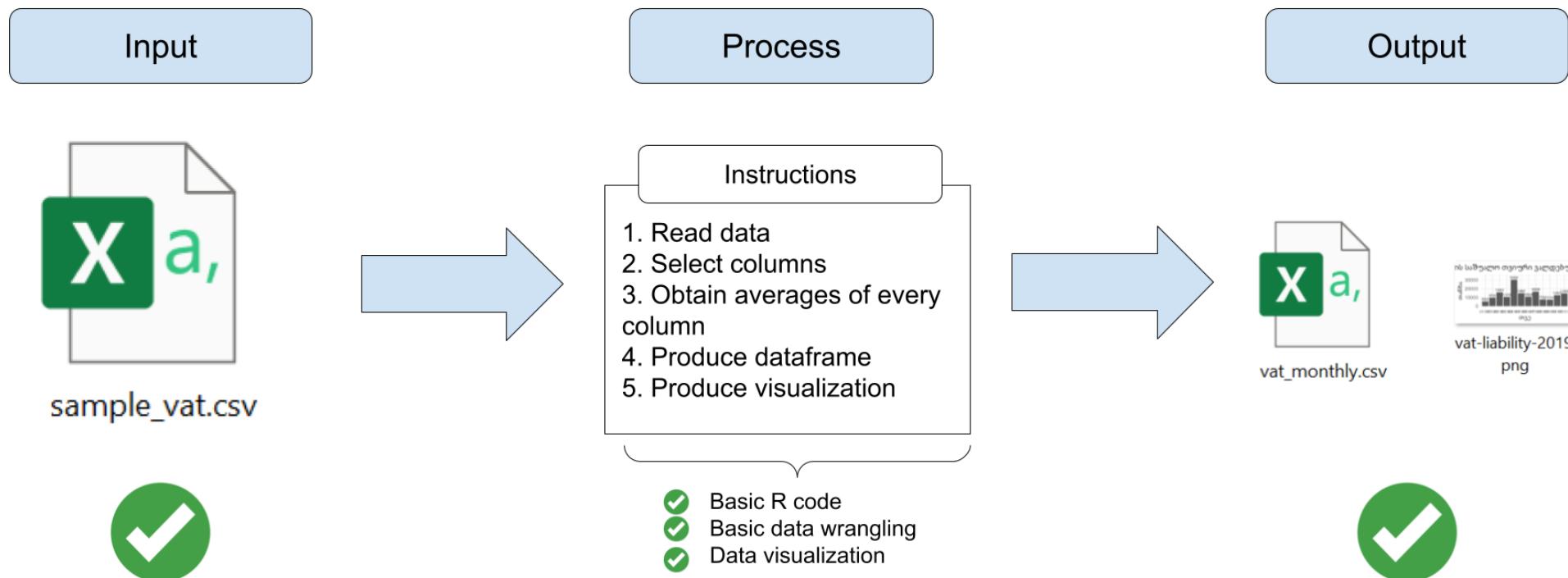
Wrapping up // მიმკვეთვა

Data work pipeline



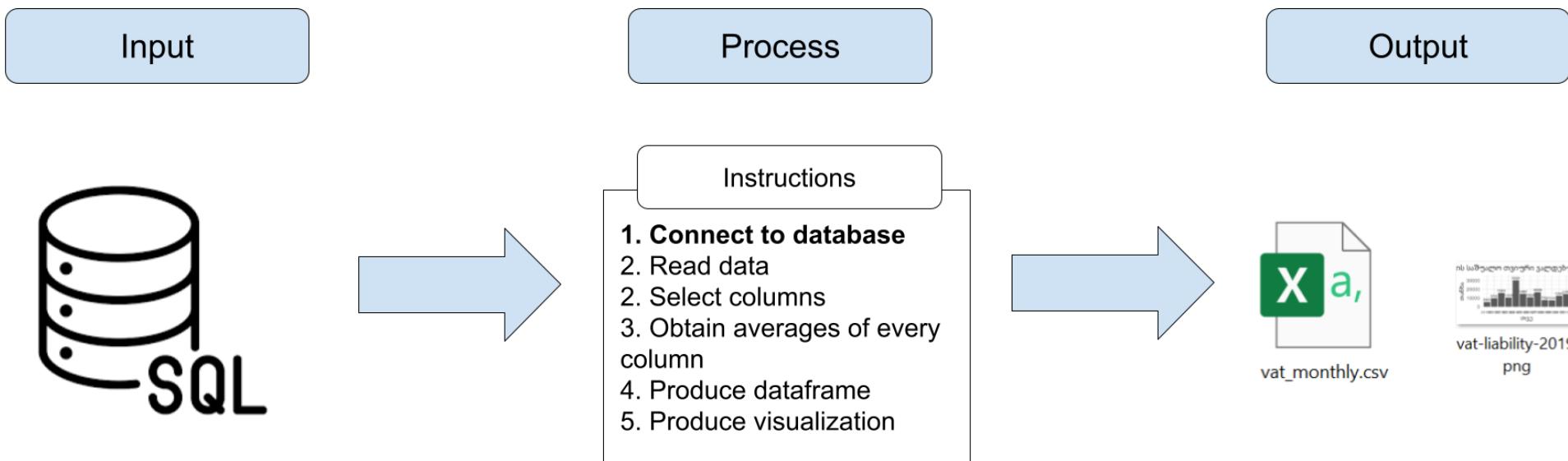
Wrapping up // მიზანთვა

Basic R code



Wrapping up // მიმკვეთვა

Looking ahead



Wrapping up // მიმკვეთვა

Looking ahead

- Connecting R with SQL databases. Some libraries:

- `dbConnect`
- `dbplyr`
- `DBI`

- More on data wrangling. More libraries:

- `tidyverse`
- `janitor`

- More on data visualization

- Geospatial data in R. Libraries:

- `sf`
- `ggmap`

Thanks! // ありがとう! // ¡Gracias! // Merci!
