Session 2 - Implementing the data work pipeline in R

R training - Georgia RS-WB DIME

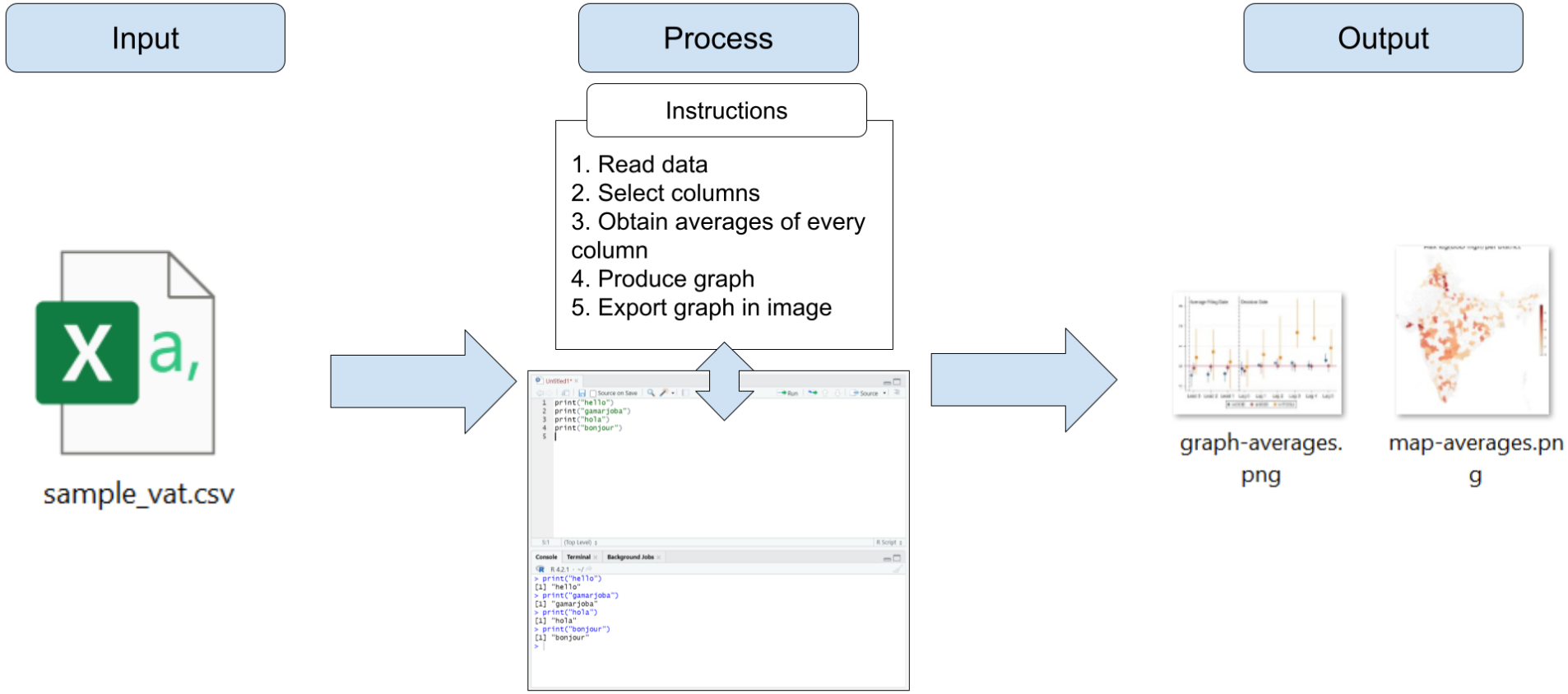
Marc-Andrea Fiorina, Luis Eduardo San Martin
The World Bank | WB Github

April 2023

Table of contents // სარჩევი

About this session // ამ სესიის შესახებ

About this session // ამ სესიის შესახებ



About this session // ამ სესიის შესახებ

Input

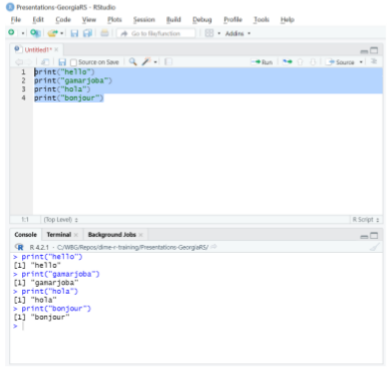
Process

Output

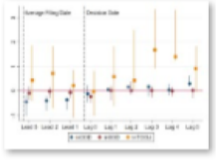
sample_vat.csv

Instructions

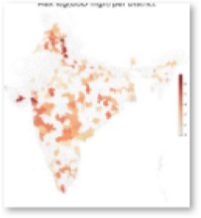
1. Read data
2. Select columns
3. Obtain averages of every column
4. Produce graph
5. Export graph in image



some more of this



graph-averages.png



map-averages.png

and basic implementation of data work pipeline

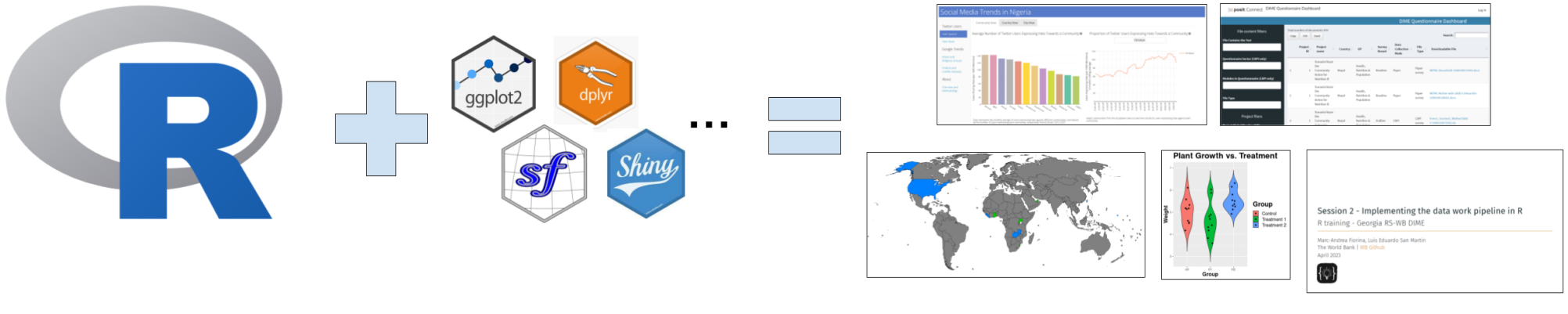
R libraries // R ბიბლიოთეკები

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:

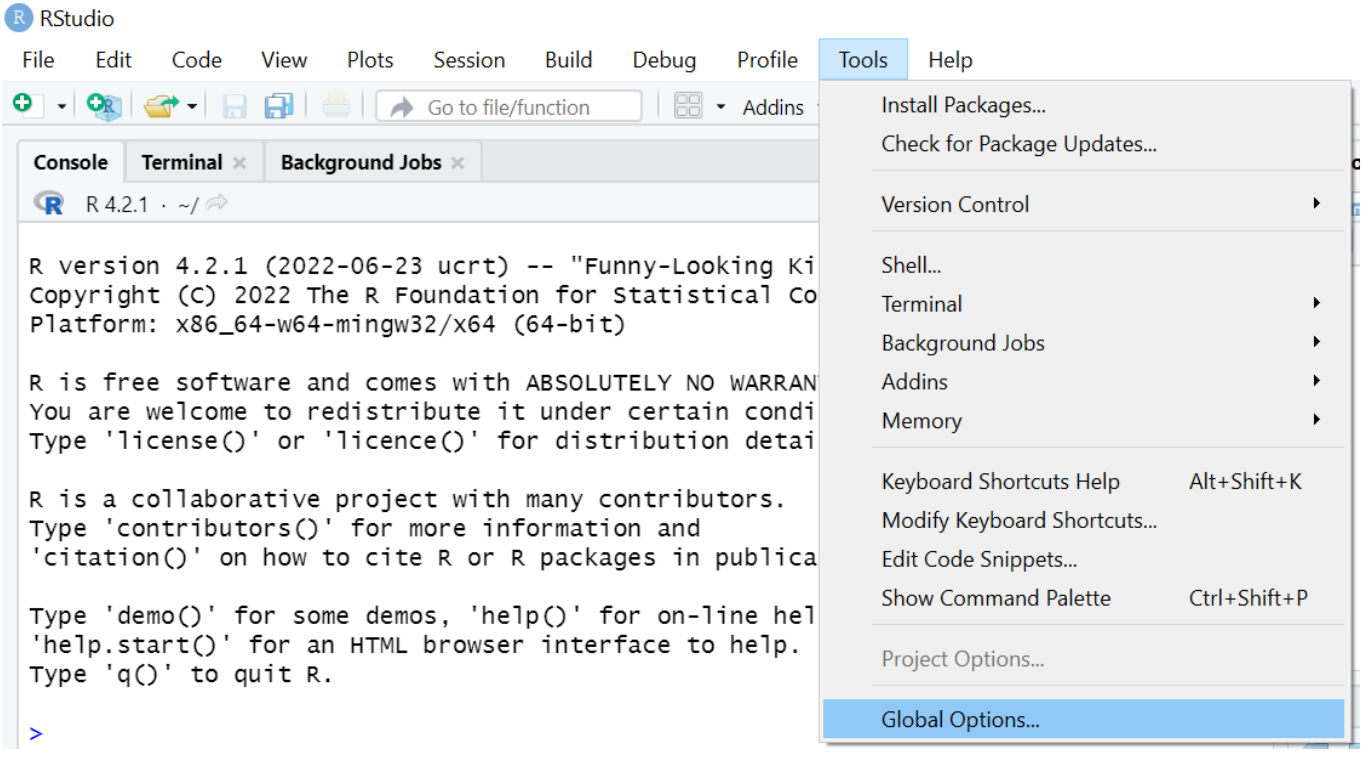


Installing R libraries

- 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...`



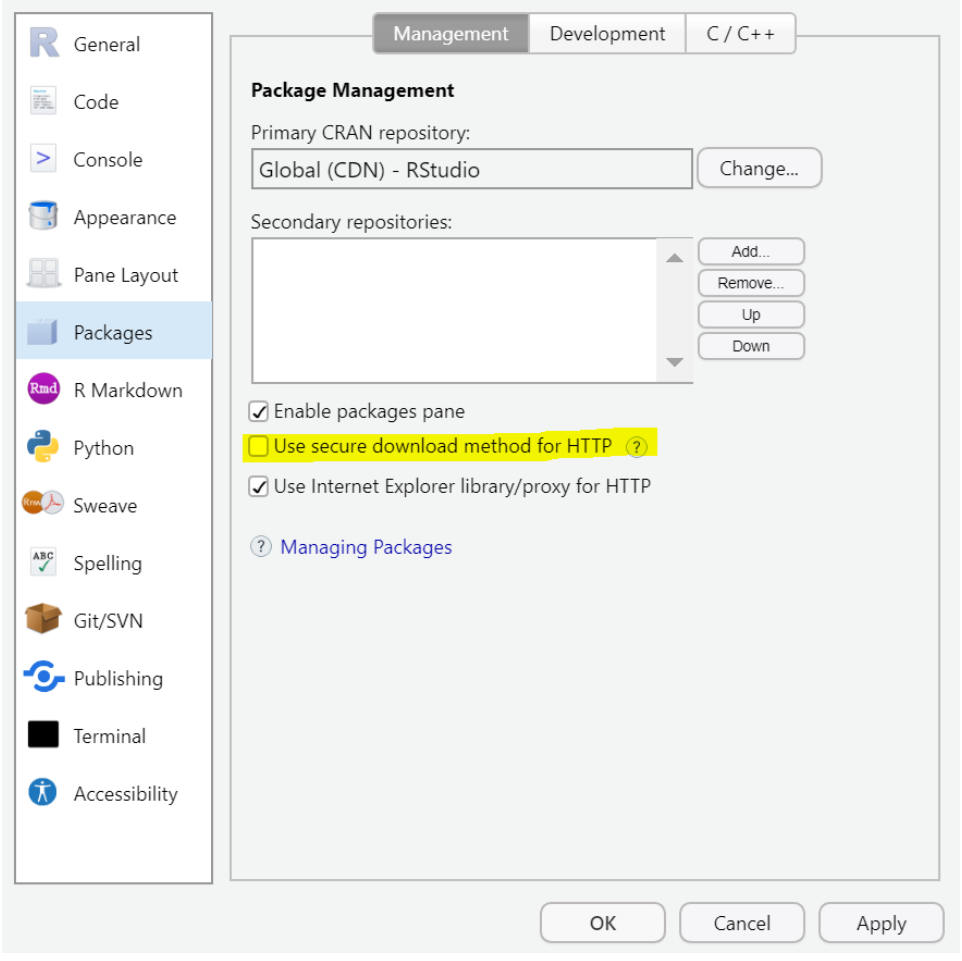
Exercise 1: Setting up the installation of libraries

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.



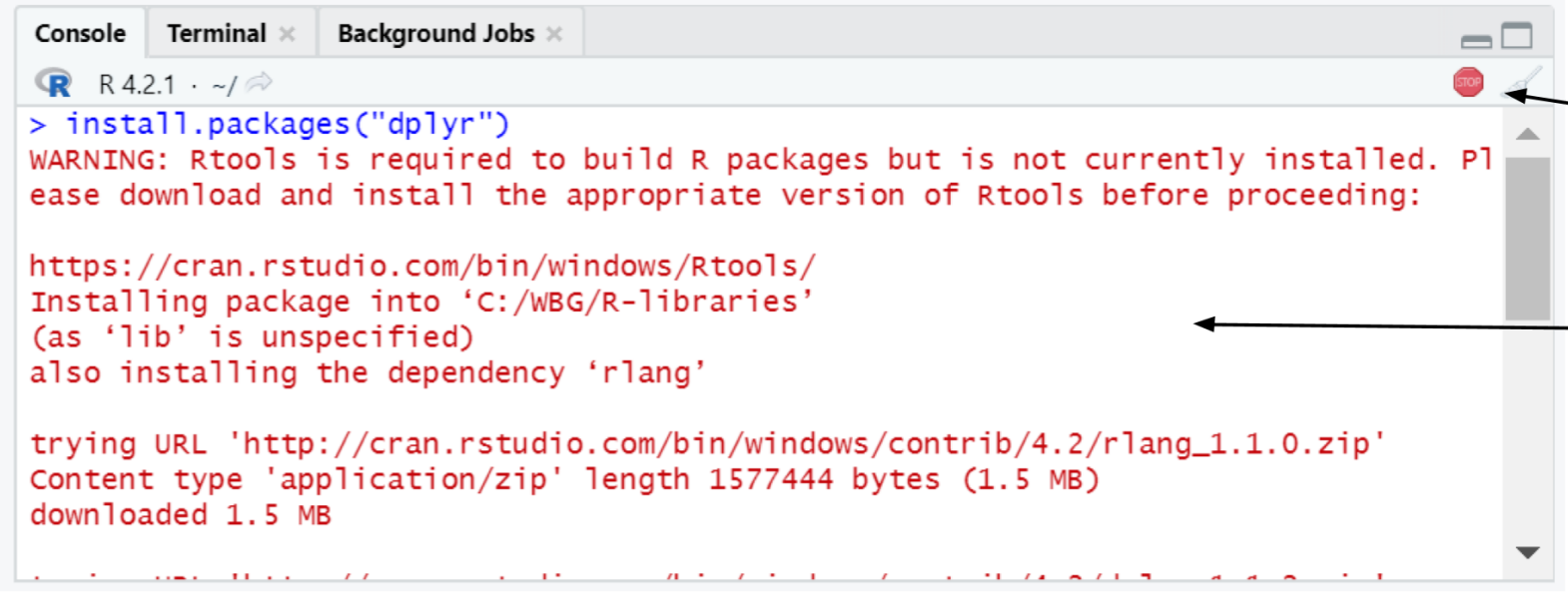
R libraries // R ბიბლიოთეკები

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

Exercise 2: Installing libraries

1. Install the libraries by using `install.packages()`
    - `install.packages("dplyr")`
    - `install.packages("ggplot2")`
    - 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**

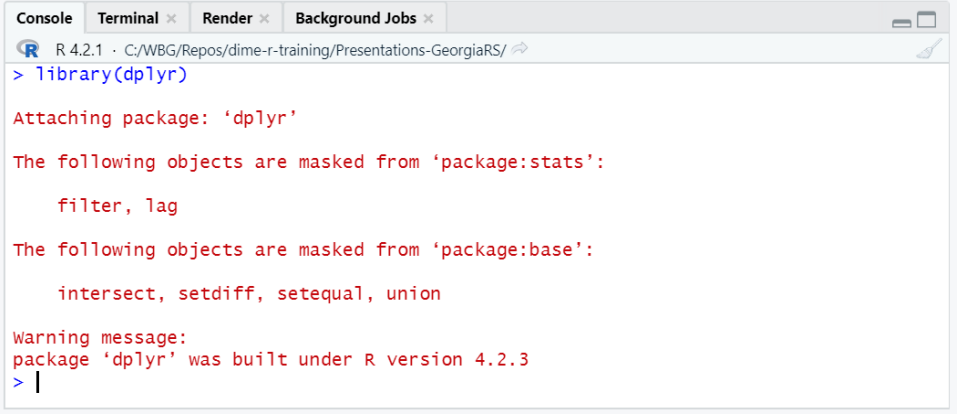
Remember to do this for the installation of `ggplot2` as well.

R libraries // R ბიბლიოთეკები

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

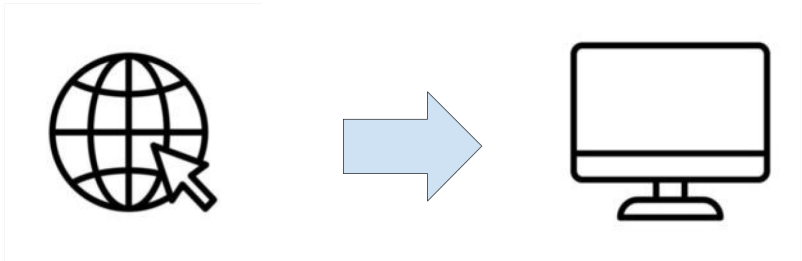
Exercise 3: Loading libraries

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

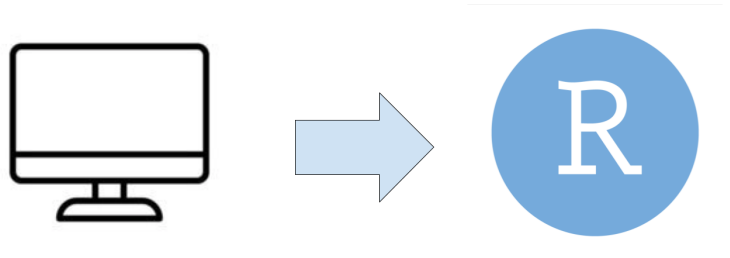


R libraries // R ბიბლიოთეკები

- Library installation:



- Library loading:



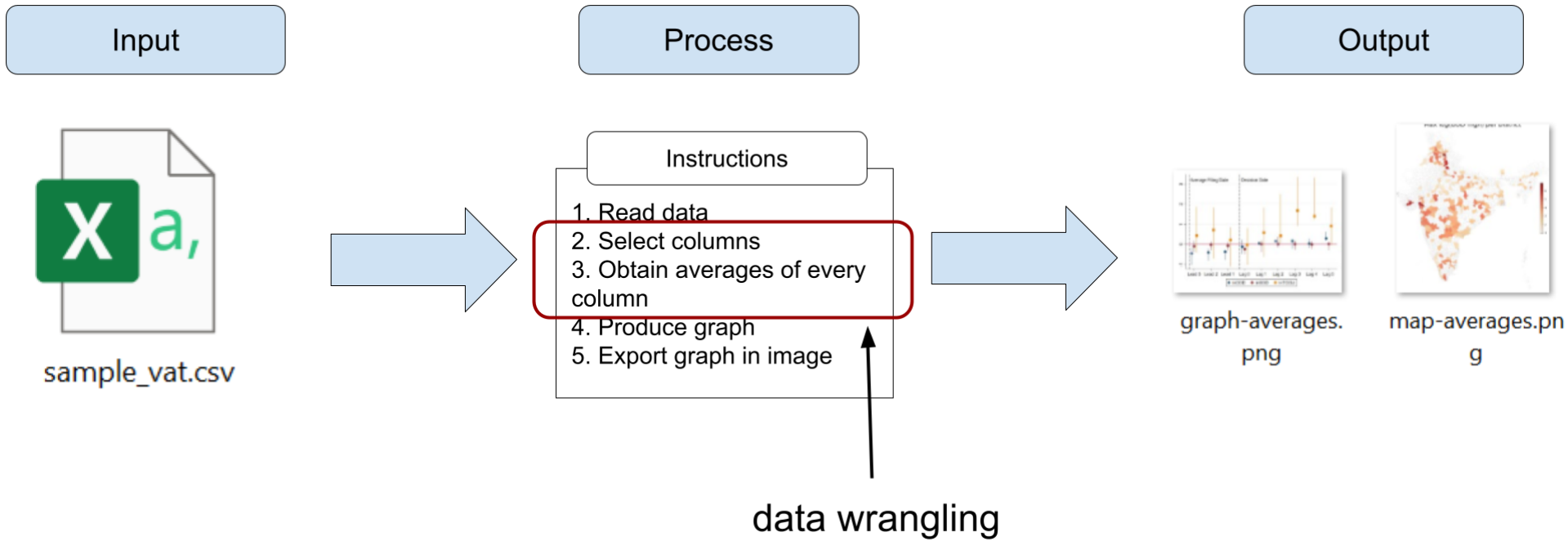
- You install R libraries only once in your computer
- You load libraries every time you open a new RStudio window (only load the libraries you will use)

Data wrangling // მონაცემთა ჩხუბი

Data wrangling // მონაცემთა ჩხუბი

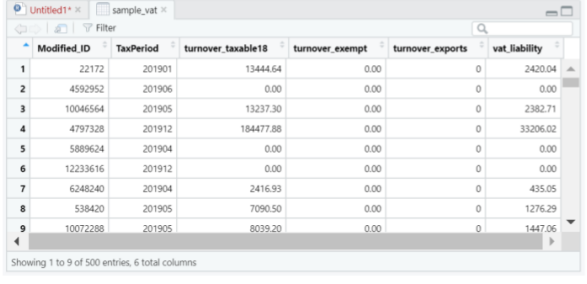
Getting your data ready

- 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**



Getting your data ready

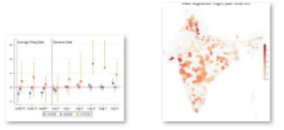
- Data wrangling is one of the most crucial aspects of data work
- It involves not only the coding aspect, but also the mental exercise of thinking what is the shape and condition that your dataframe needs to have in order to produce your desired output



	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

dataframe needed for outputs



outputs

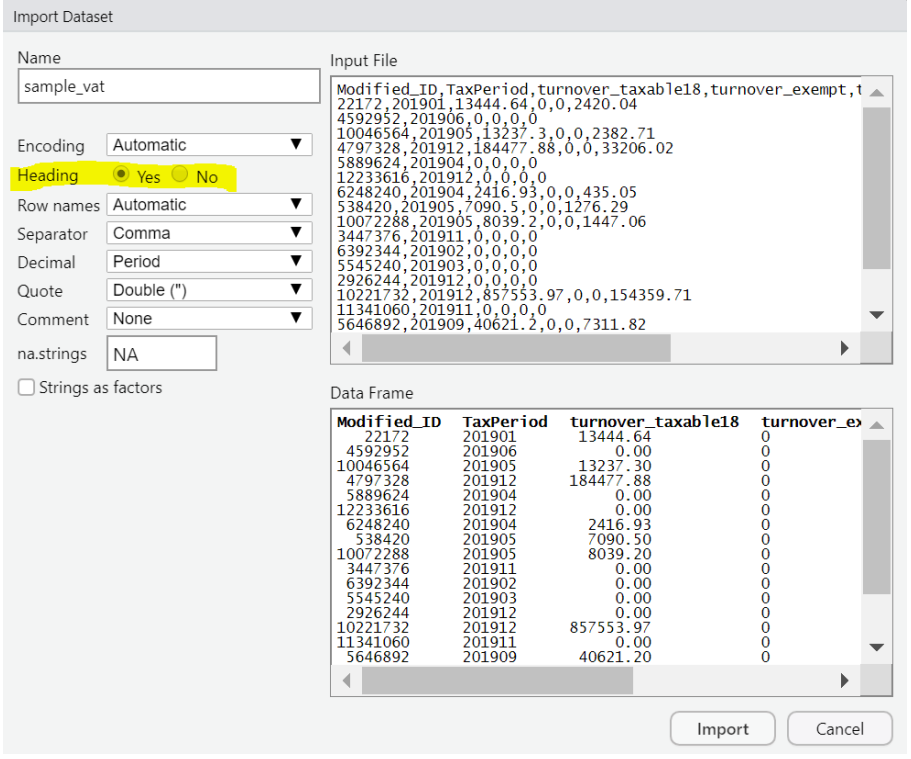
Getting your data ready

- 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

Exercise 4: Loading data

Note that this is the same exercise we did yesterday. If you 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`



Loading data with a function

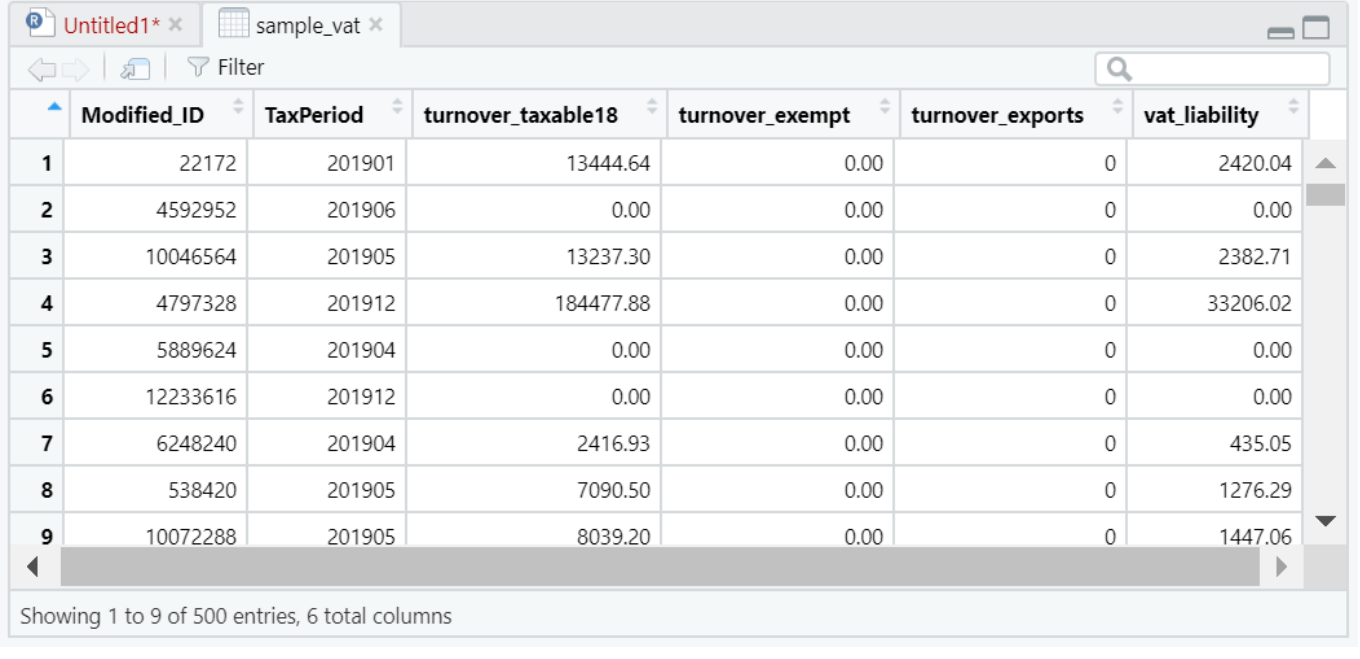
- You can also load CSV data with the function `read.csv()` instead of using this point-and-click approach
- The first argument of `read.csv()` is the path in your computer where your data is. For example

```
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

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



	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 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 work request

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

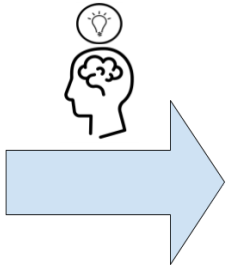
sample_vat

Filter

	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



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

sample_vat | Untitled1* | final_df

Filter

	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 work request

The data wrangling here involves a number of operations:

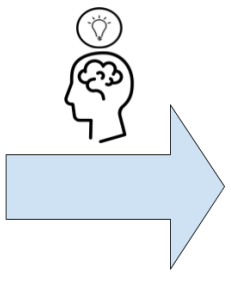
1. Keeping only the relevant columns and dropping everything else
2. Collapsing the dataframe by month
3. Calculating the total VAT liability by month
4. Calculating the average VAT liability by month

sample_vat

Filter

	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



sample_vat | Untitled1* | final_df

Filter

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

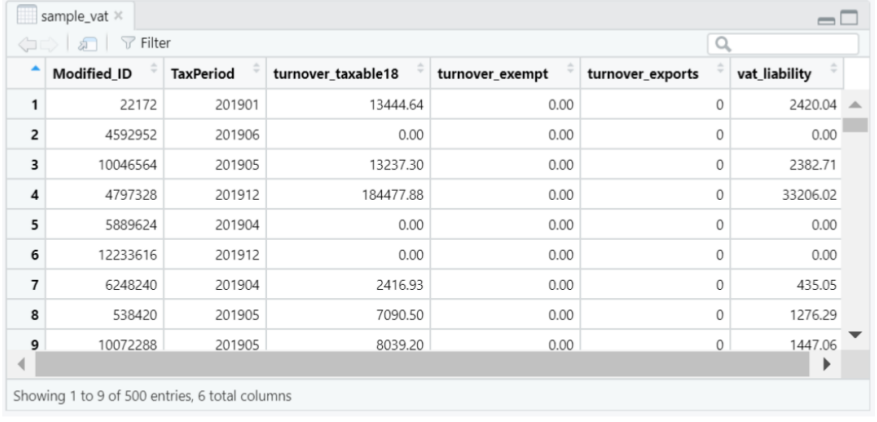
- Observations: Months in 2019 (12)
- Columns: month, total VAT liability, average VAT liability

Data wrangling // მონაცემთა ჩხუბი

1. Keeping only relevant columns

Use `select()` for this:

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



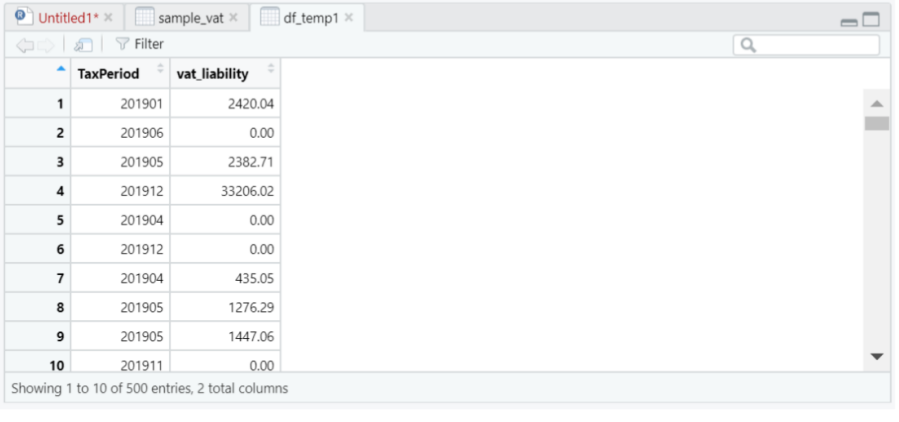
sample_vat

Filter

	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



Untitled1* sample_vat df_temp1

Filter

	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

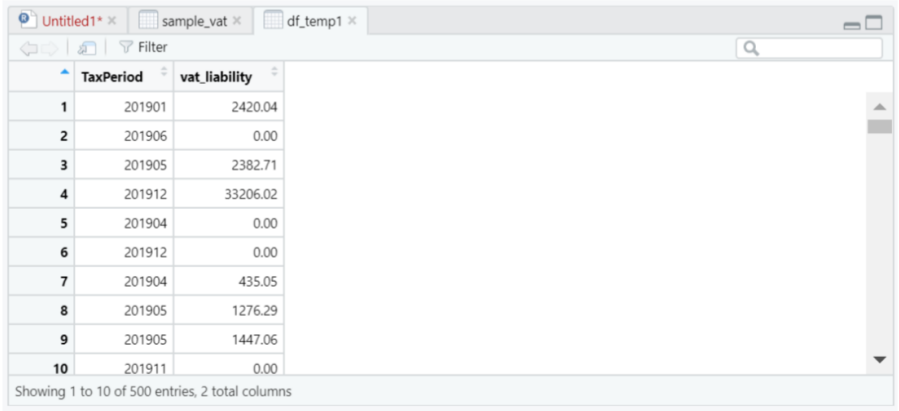
- Observations: Monthly tax declaration (500) of 2019
- Columns: month, VAT liability

Data wrangling // მონაცემთა ჩხუბი

2. Collapsing by month

Use `group_by()`:

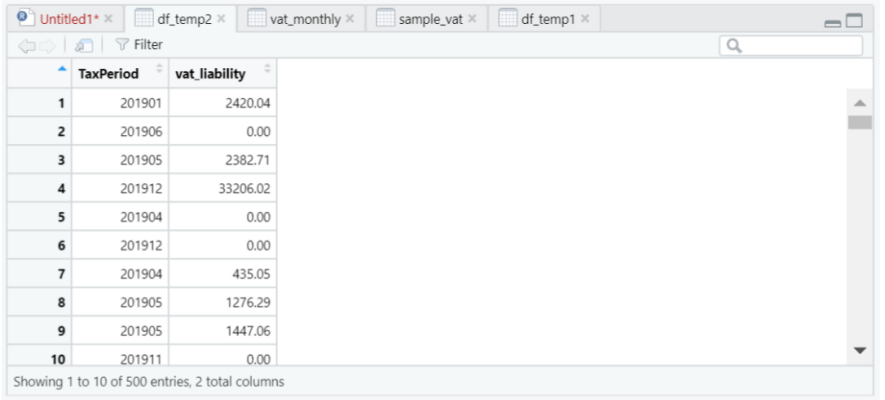
```
df_temp2 <- group_by(df_temp1, TaxPeriod)
```



	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

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

- 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**

3. Calculating aggregated columns: total and average VAT liability

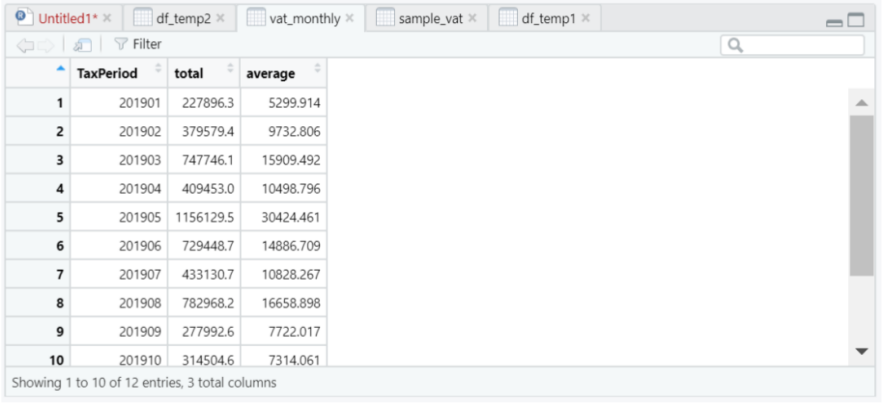
Use `summarize()` combined with `sum()` and `mean()`:

```
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

- 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



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

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))
```

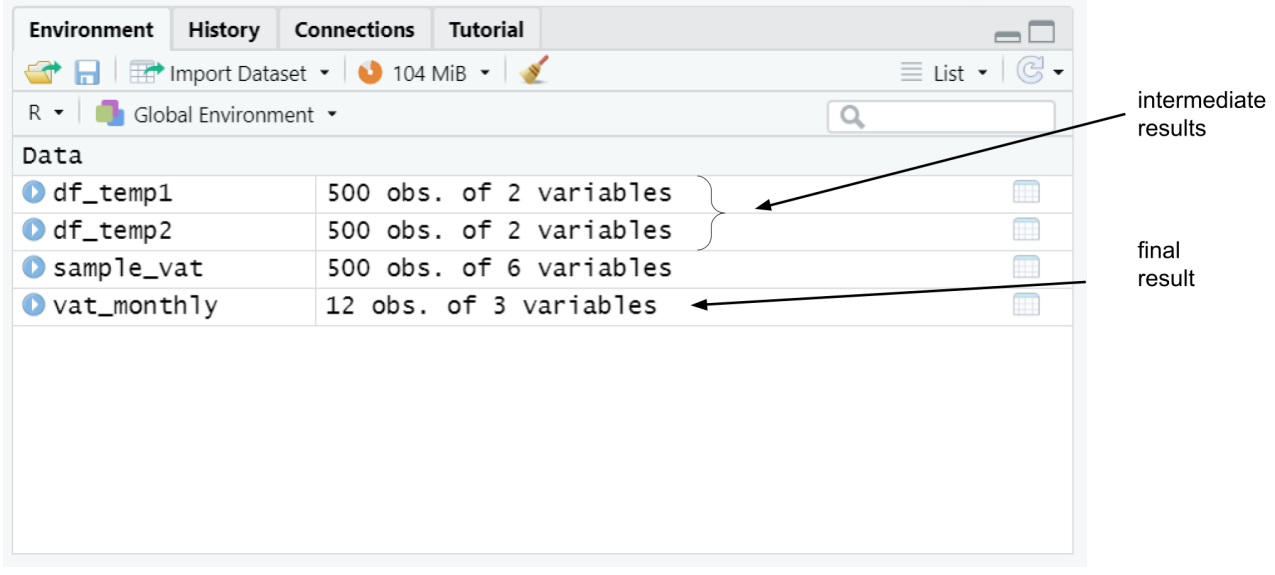
Some notes:

- 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 // მონაცემთა ჩხუბი

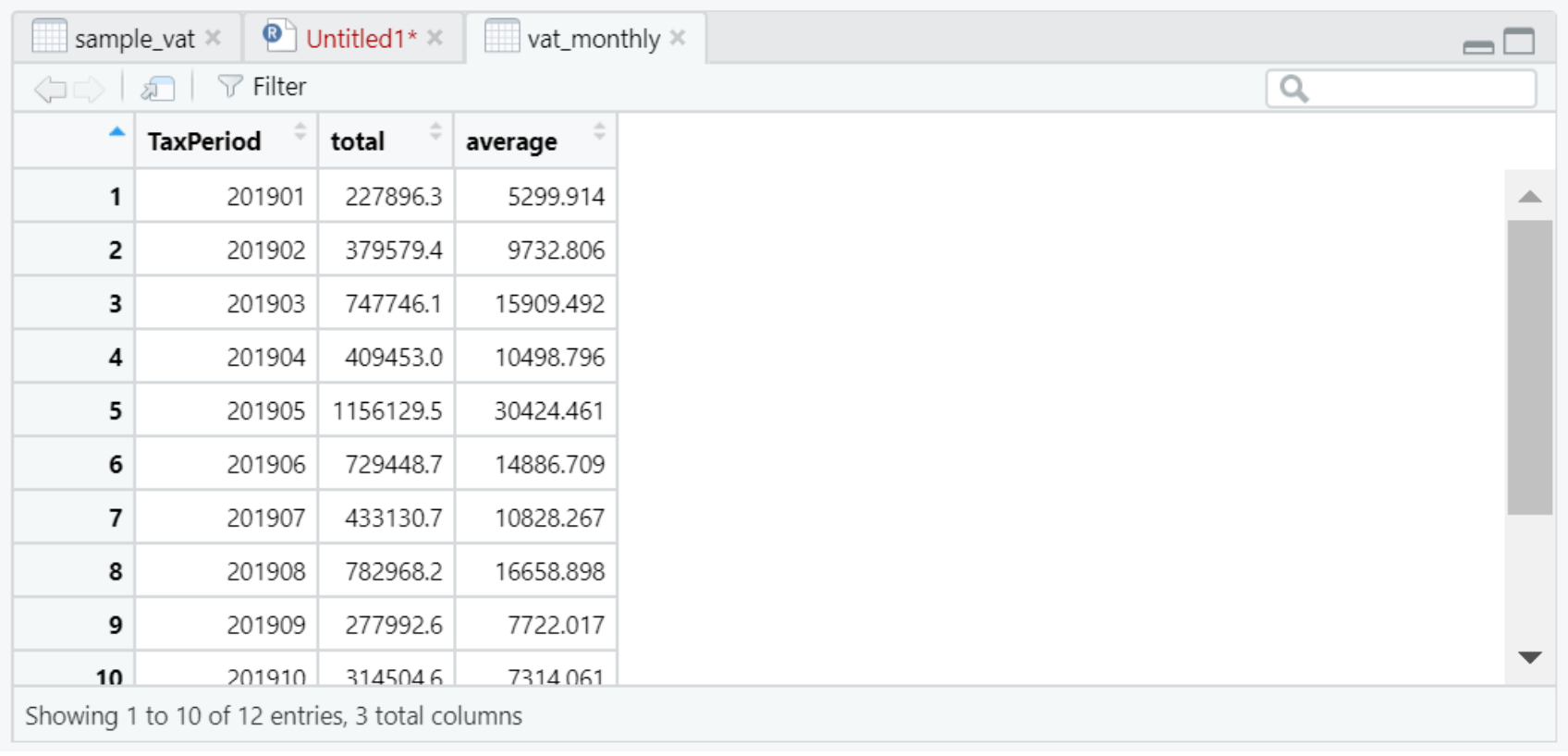
Some more notes:

- 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`



Data wrangling // მონაცემთა ჩხუბი

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 // მონაცემთა ჩხუბი

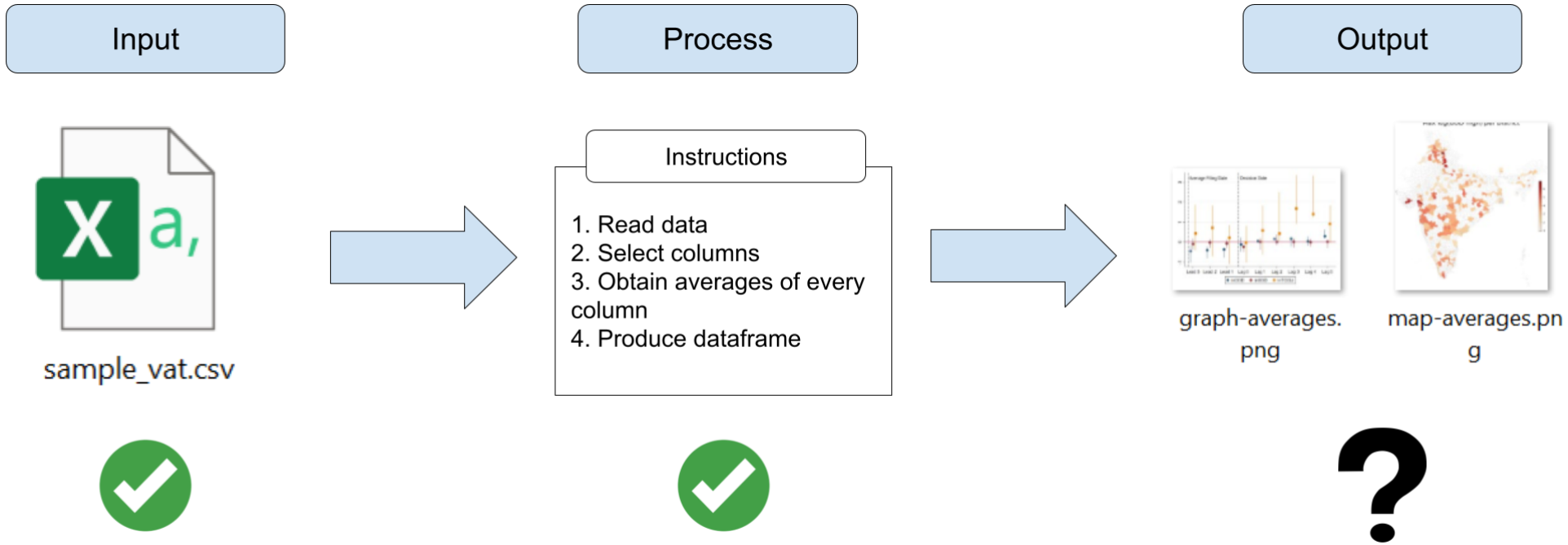
- This was a simple data wrangling example that we chose for convenience
- Data wrangling can involve lots of operations

Operation	Function in `dplyr`
Subset columns	`select()`
Subset rows (based on condition)	`filter()`
Create new columns	`mutate()`
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 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

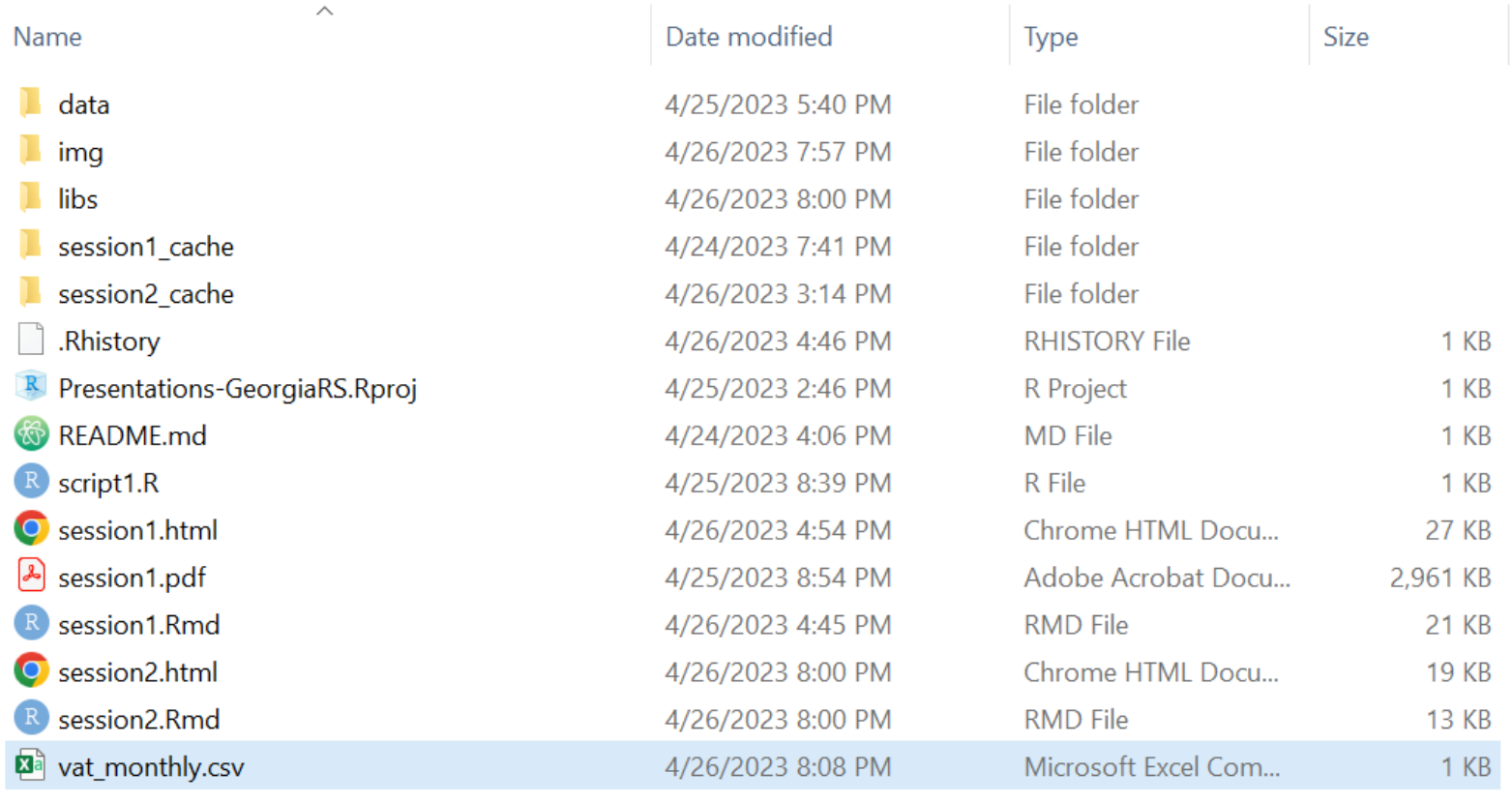
Exercise 6: Exporting `vat_monthly`

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
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 8:00 PM	Chrome HTML Docu...	19 KB
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

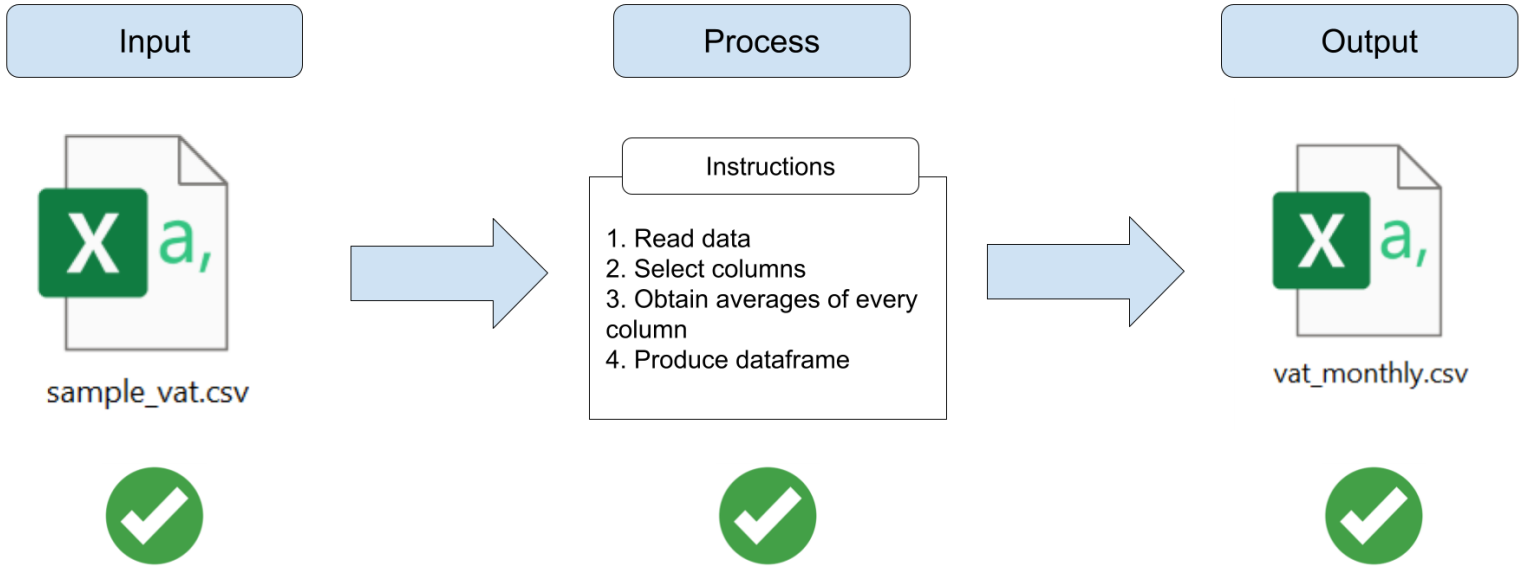
Some notes on file paths

```
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

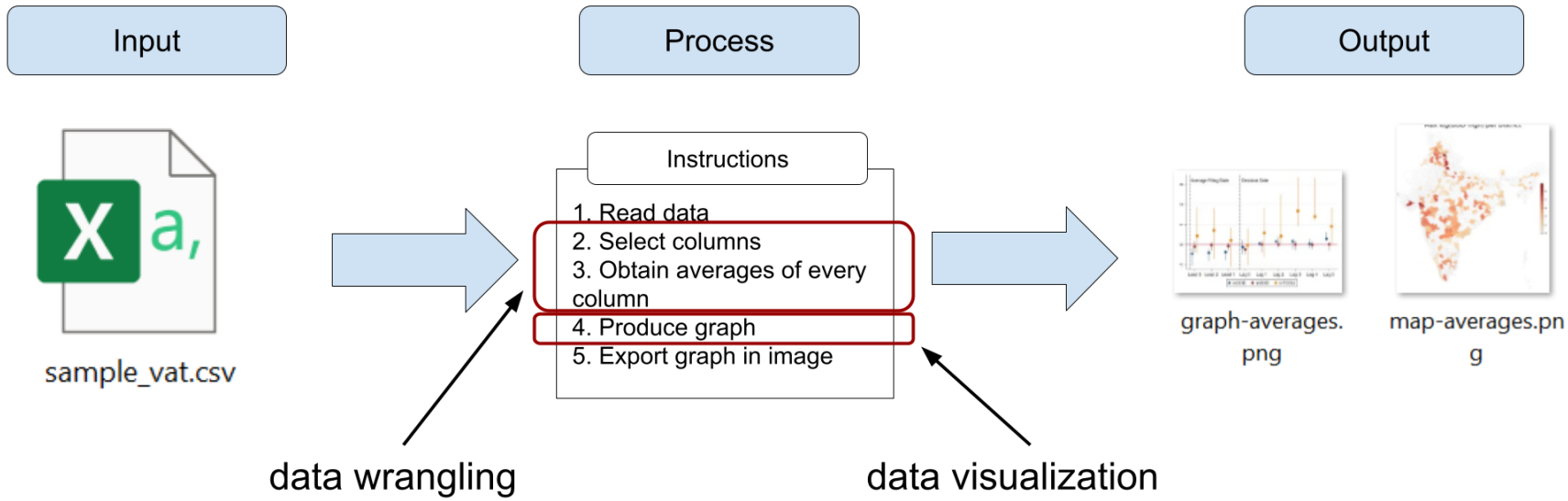
Our data pipeline has been fully implemented at this point. Great!



Data visualization // მონაცემთა ვიზუალიზაცია

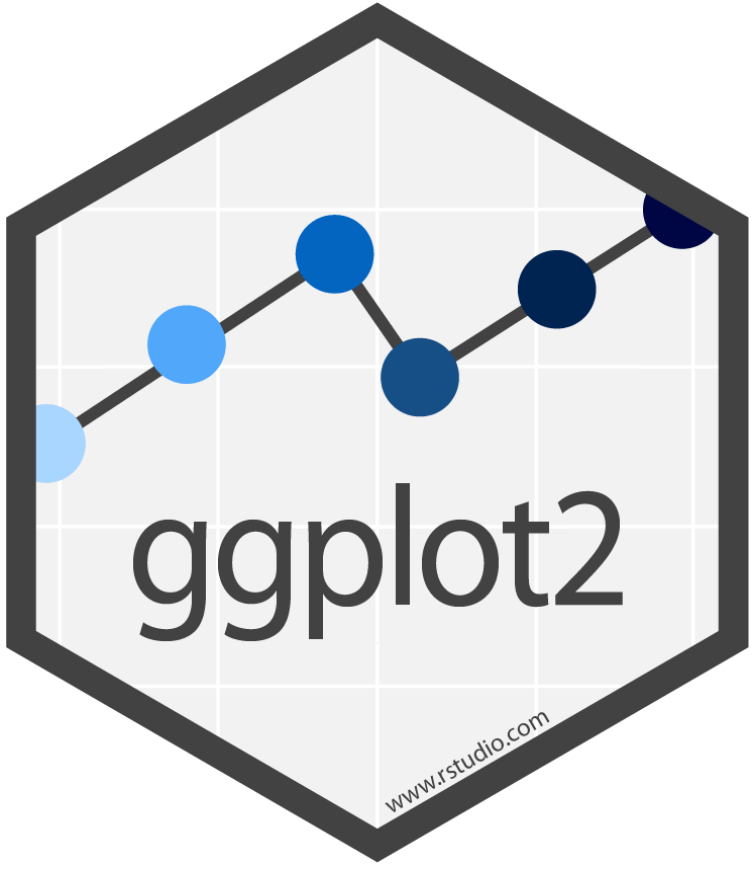
Data visualization in the data work pipeline

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



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`

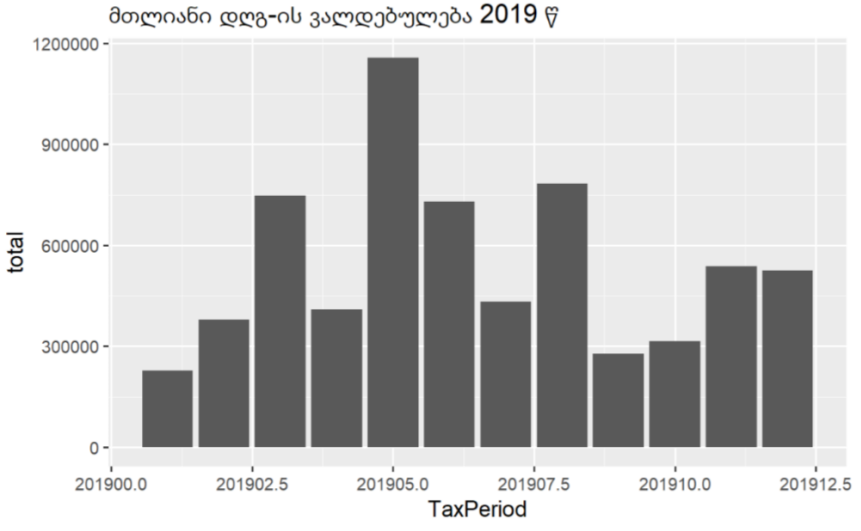
sample_vat | Untitled1* | vat_monthly

Filter

	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`

sample_vat | Untitled1* | vat_monthly

Filter

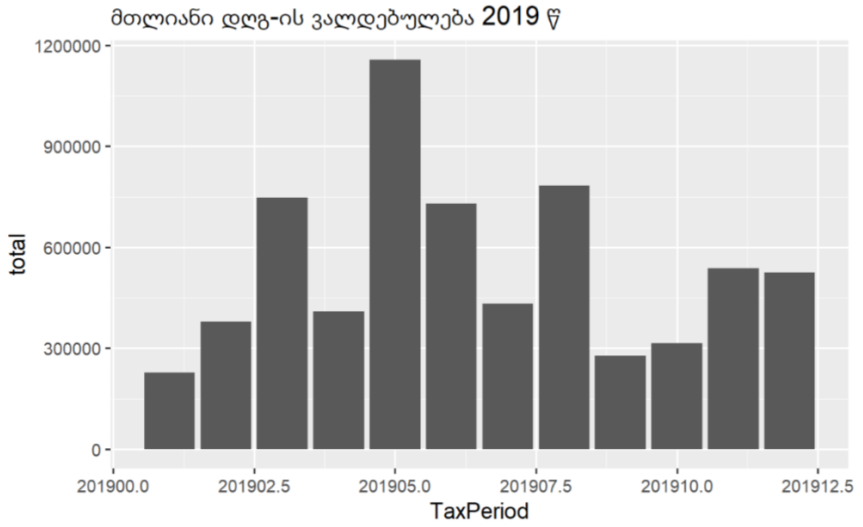
	TaxPeriod	total	average
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
			7314.061

the first argument of ggplot() is the input dataframe (vat_monthly)

ggplot(): function to initialize a plot

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

The “+” symbols denote added elements in the plot



Data visualization // მონაცემთა ვიზუალიზაცია

The grammar of graphics in `ggplot2`

sample_vat | Untitled1* | vat_monthly

Filter

	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			

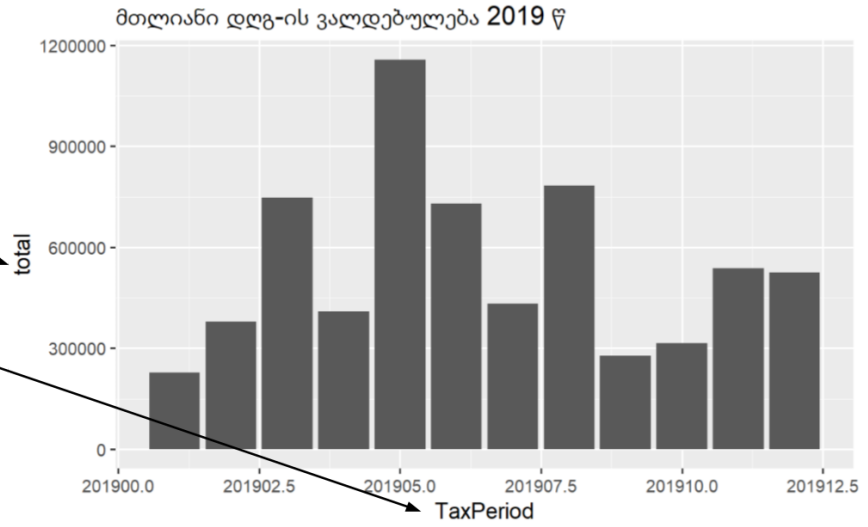
Showing 1 to

y axis (total)

x axis (TaxPeriod)

the second argument of ggplot() defines the plot aesthetics

```
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`

sample_vat | Untitled1* | vat_monthly

Filter

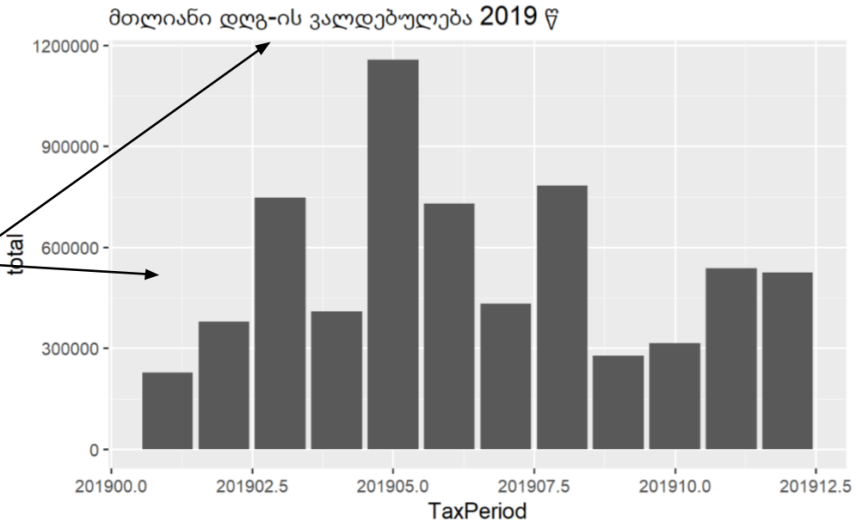
	TaxPeriod	total	average
1	201901	227896.3	5299.914
2	201902	379579.4	9732.806
3	201903	747746.1	15909.492
4	201904	409453.0	10498.796
5	201905	1156129.5	30424.461
6	201906	729448.7	14886.709
7	201907	433130.7	10828.267
8	201908	782968.2	16658.898
9	201909	277992.6	7722.017
10	201910	314504.6	7314.061

Showing 1 to 10 of 12 entries, 3 total columns

geom_col() determines the geometry encoding (for the y-axis by default)

labs() determines the labels, for the plot title and others as well

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

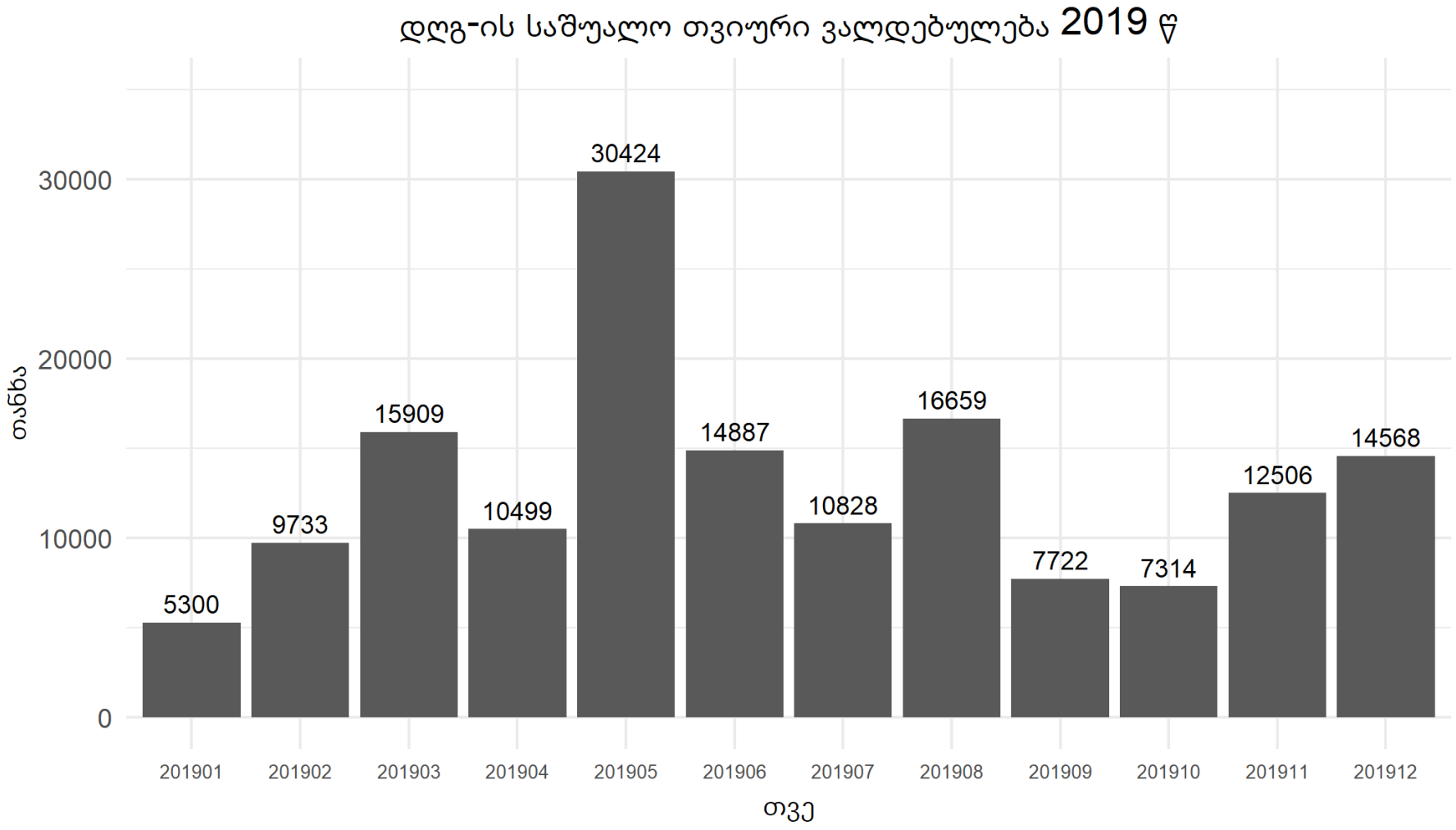


Exercise 7

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

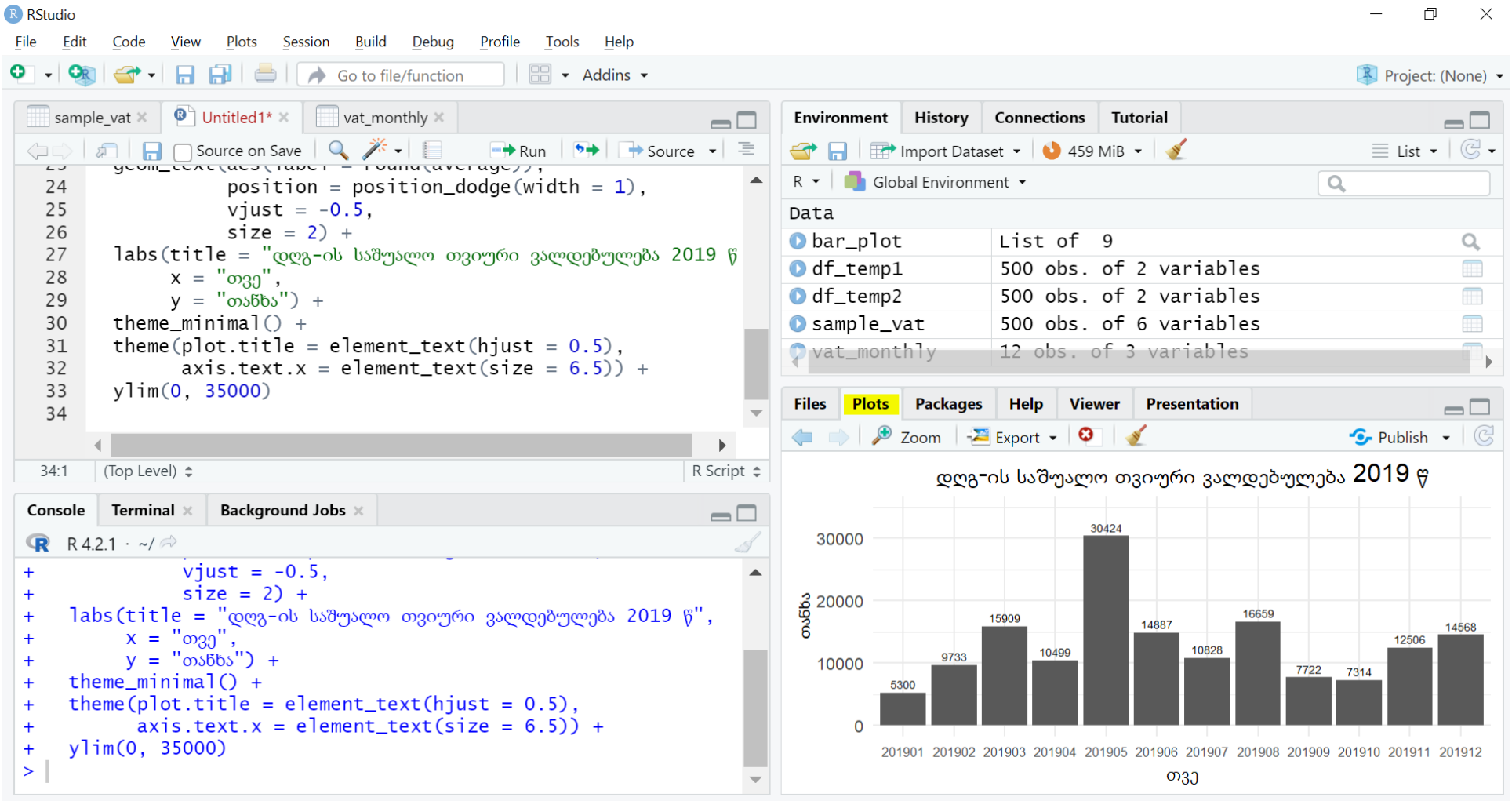
```
ggplot(vat_monthly,
       aes(x = as.factor(TaxPeriod),
           y = average)) +
  geom_col() +
  geom_text(aes(label = round(average)),
            position = position_dodge(width = 1),
            vjust = -0.5,
            size = 3) +
  labs(title = "დღგ-ის საშუალო თვიური ვალდებულება 2019 წ",
       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 // მონაცემთა ვიზუალიზაცია

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



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")
```

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

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:

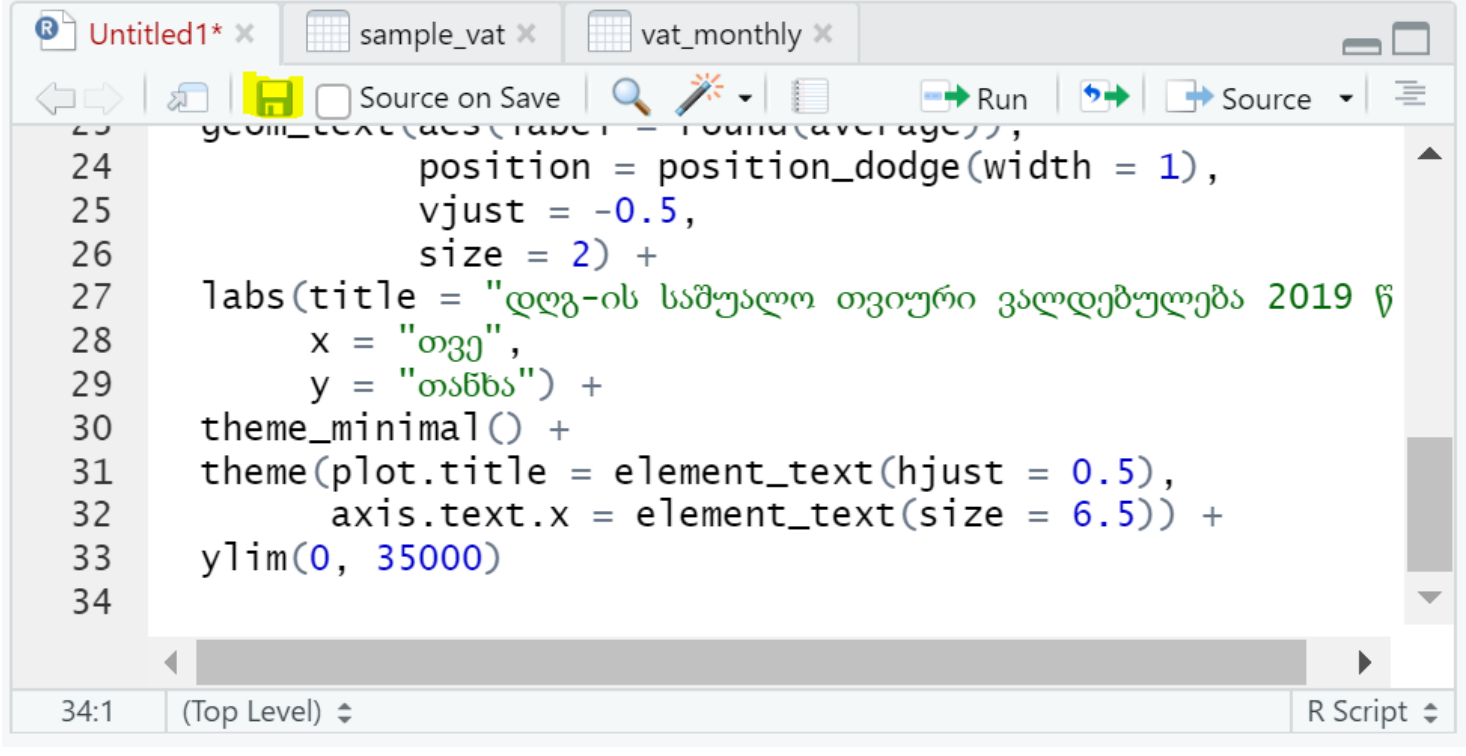
Encoding	Function in `ggplot2`
Bars	`geom_col()`
Lines	`geom_line()`
Points (scatterplot)	`geom_point()`
Area	`geom_area()`
Histogram	`geom_histogram()`
Floating labels (texts)	`geom_text()`
Box plot	`geom_boxplot()`
Pie chart	`geom_bar() + coord_polar()`
Smoothed line	`geom_smooth()`

Wrapping up // შეფუთვა

Wrapping up // შეჯუთვა

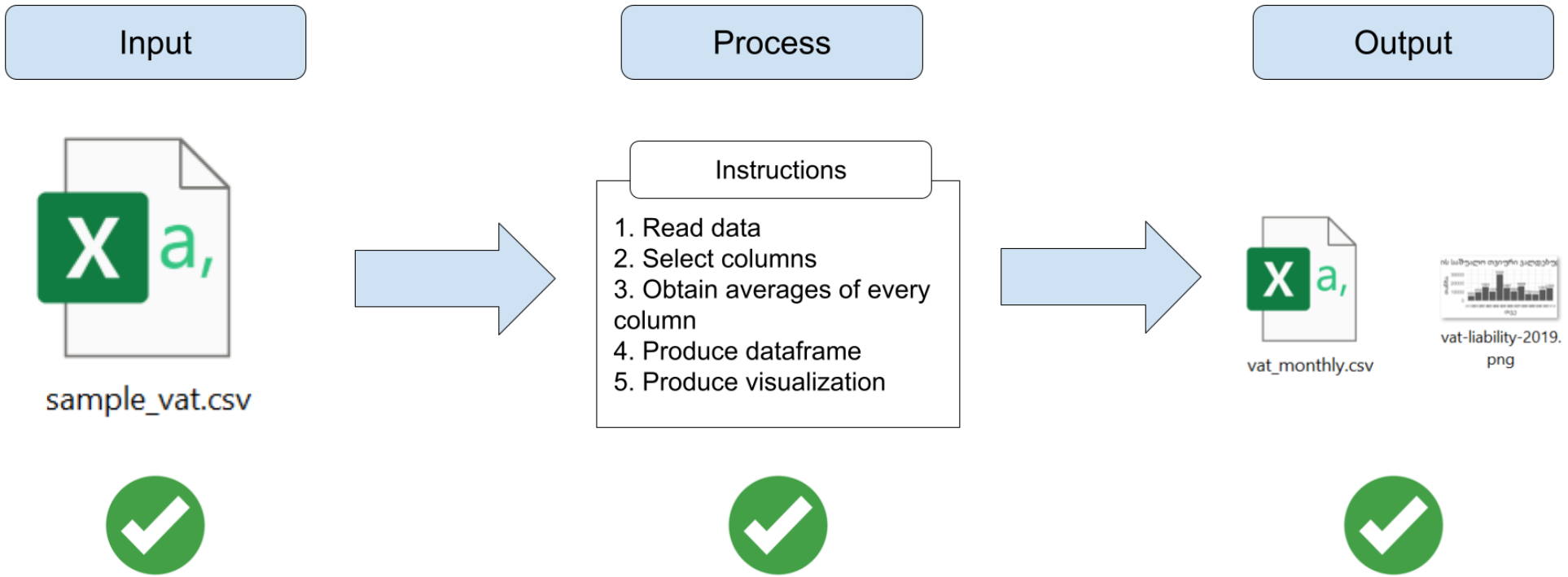
Don't forget to save your work!

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



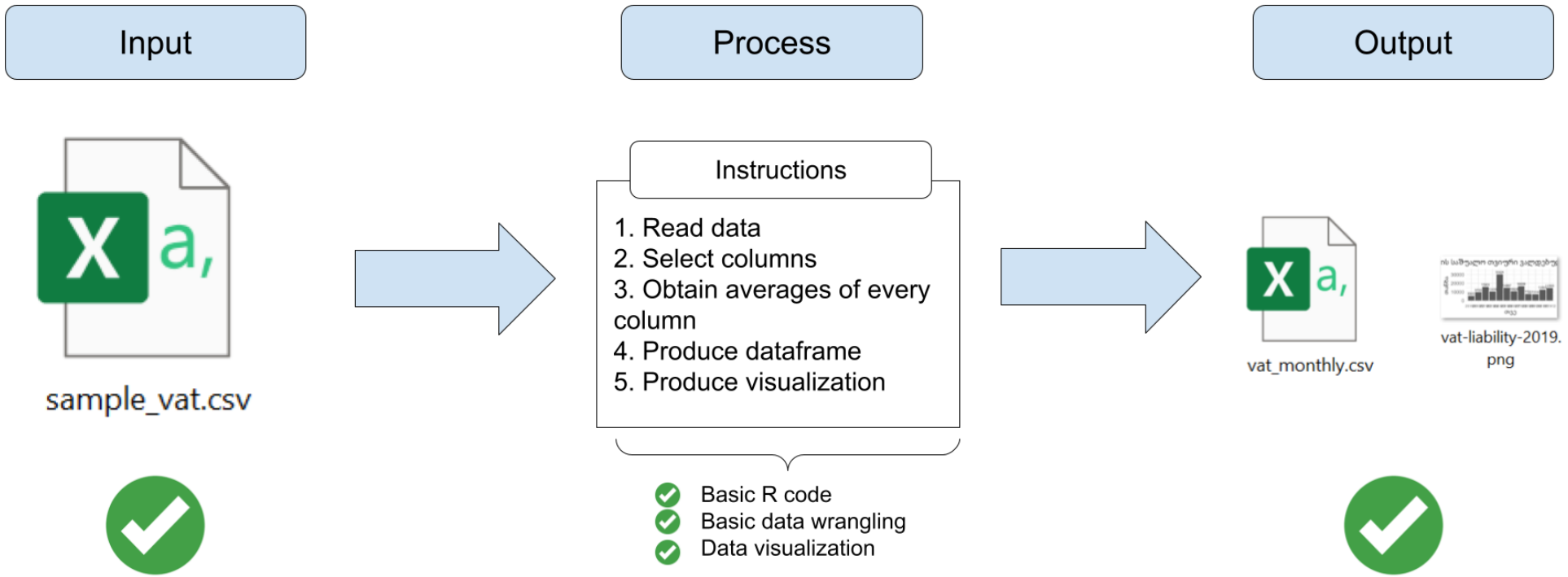
Wrapping up // შეჯამება

Data work pipeline



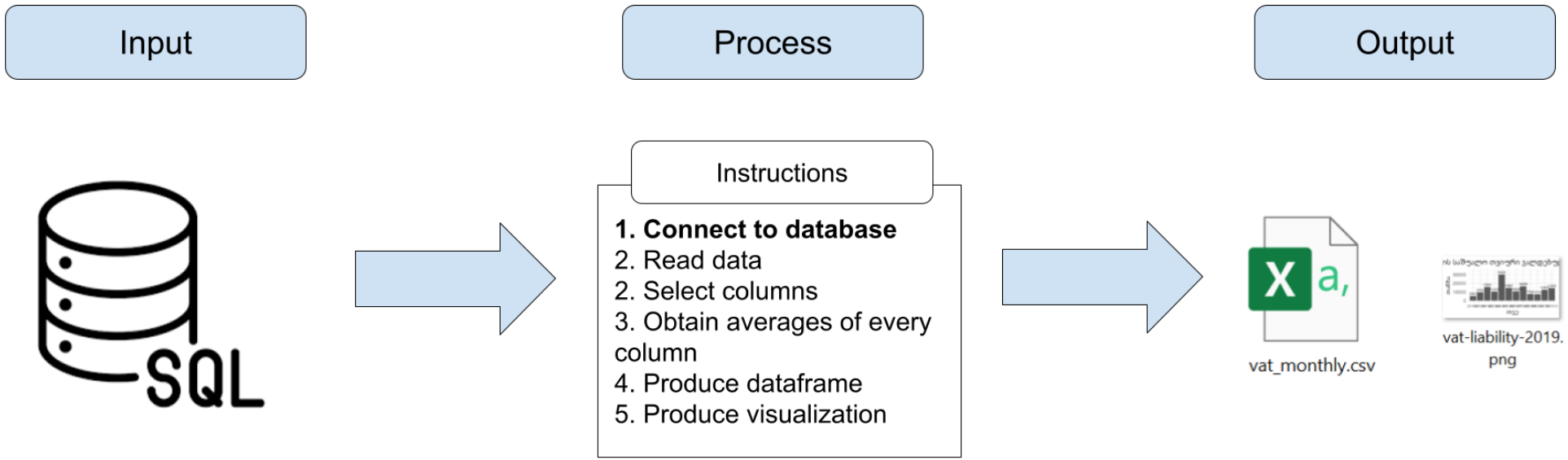
Wrapping up // შეჯამება

Basic R code



Wrapping up // შეჯამება

Looking ahead



Wrapping up // შეჯამება

Looking ahead

- Connecting R with SQL databases. Some libraries:
  - `dbConnect`
  - `dbplyr`
  - `DBI`
- More on data wrangling. More libraries:
  - `tidyr`
  - `janitor`
- More on data visualization
- Geospatial data in R. Libraries:
  - `sf`
  - `ggmap`

Thanks! // მადლობა! // ¡Gracias! // Merci!