Session 2 - Data wrangling

R training

María Reyes Retana
The World Bank | January 2025

THE WORLD BANK
IBRD • IDA | WORLD BANK GROUP
Development Economics • Impact

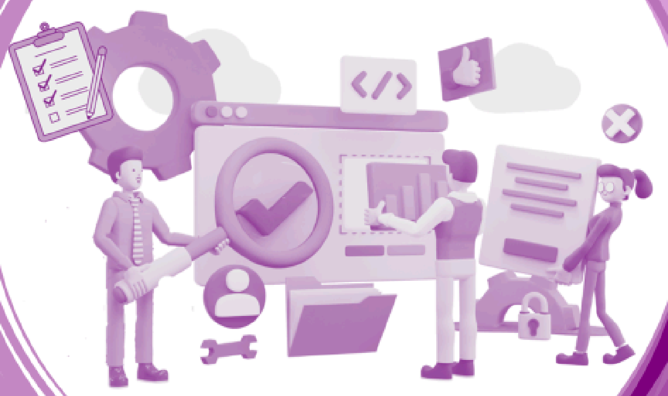
WORLD BANK GROUP
INSTITUTE FOR ECONOMIC DEVELOPMENT

Reproducible Research Repository

Government Analytics and R Training:

Strengthening Public Sector Reporting and Data Analysis

January 13 – January 17, 2025



Access to Training Materials

All the materials for this training are available at the following link:
https://osf.io/r3fn5/

Here's what you will find there:

- 📂 **Data**: All datasets we'll use throughout the training sessions.
- 📑 **Slides**: The presentation slides for each session.
- ✏️ **Solutions**: Will be added to the folder after each session.

Course Structure

This training will cover the basics of coding in R. Below is the structure of the course:

- 📅 **Day 1**: Introduction to R – Get familiar with the R environment and basic syntax.
- 🛠 **Day 2**: **Data Wrangling – Learn to clean and transform data effectively.**
- 📊 **Day 3**: Descriptive Statistics – Explore methods to summarize and analyze data.
- 📈 **Day 4**: Data Visualization – Create meaningful and impactful data visualizations.

About this session

About this session

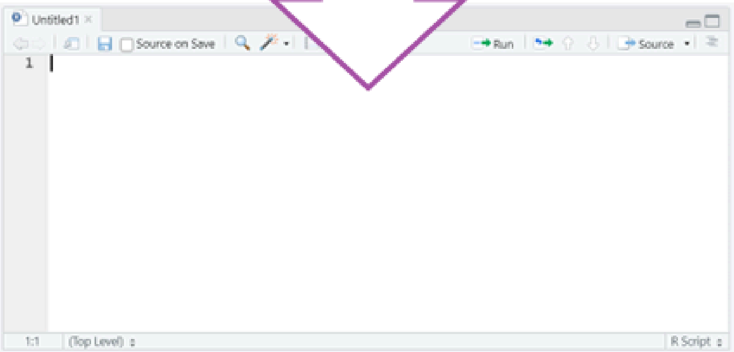
Input

department_staff_list.csv

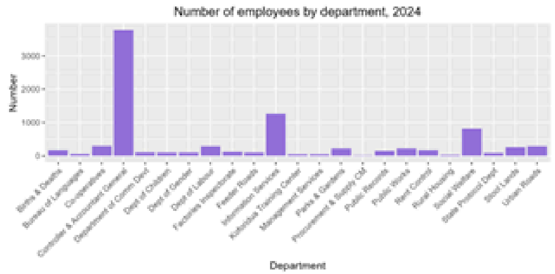
Data process

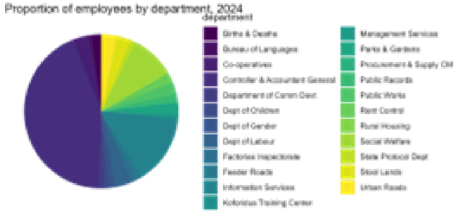
Instructions

1. Read data
2. Select columns
3. Obtain averages for all columns
4. Produce graph
5. Export graph in image



Output





About this session

Input

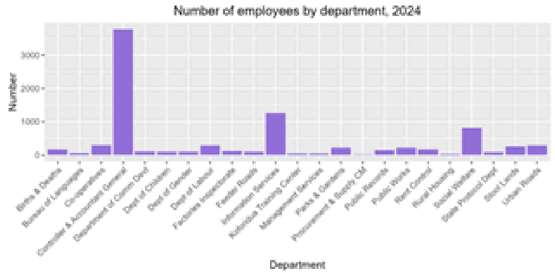
department_staff_list.csv

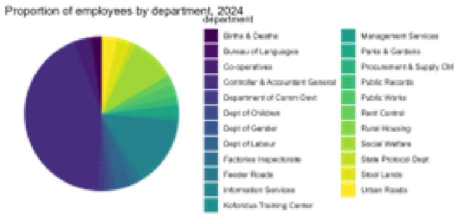
Data process

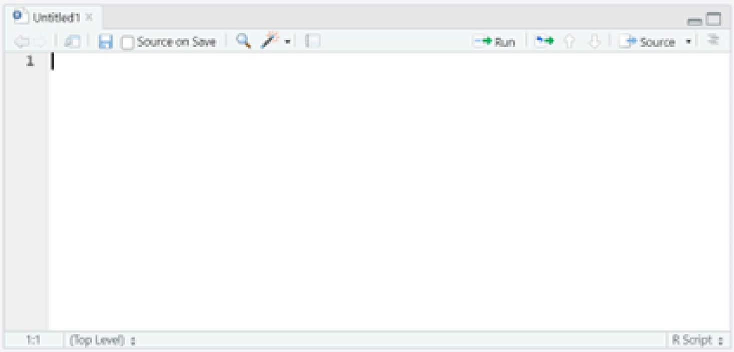
Instructions

1. Read data
2. Select columns
3. Obtain averages for all columns
4. Produce graph
5. Export graph in image

Output







This session: getting data ready to be used to create our outputs

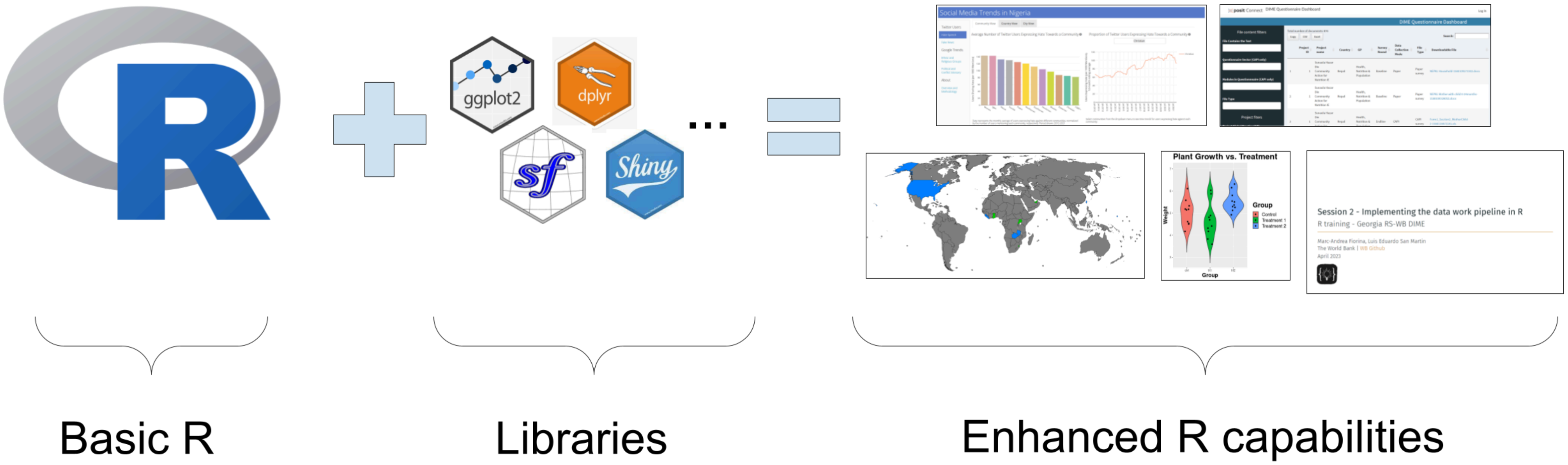
R Packages

R Packages

- Installing R in your computer gives you access to its basic functions
- Additionally, you can also install packages. Packages are a collection of 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)
- They contain code that other R users have prepared for the community.

R Packages

In a nutshell:

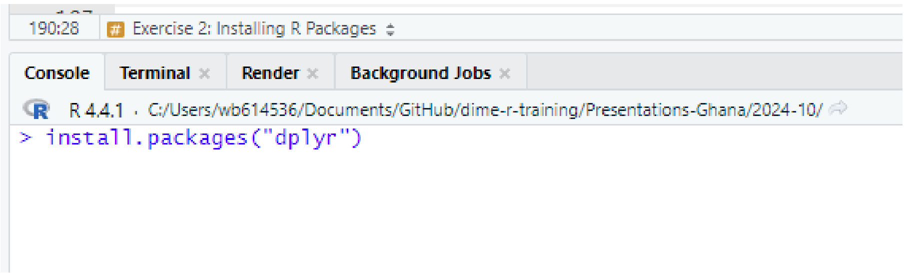


R Packages

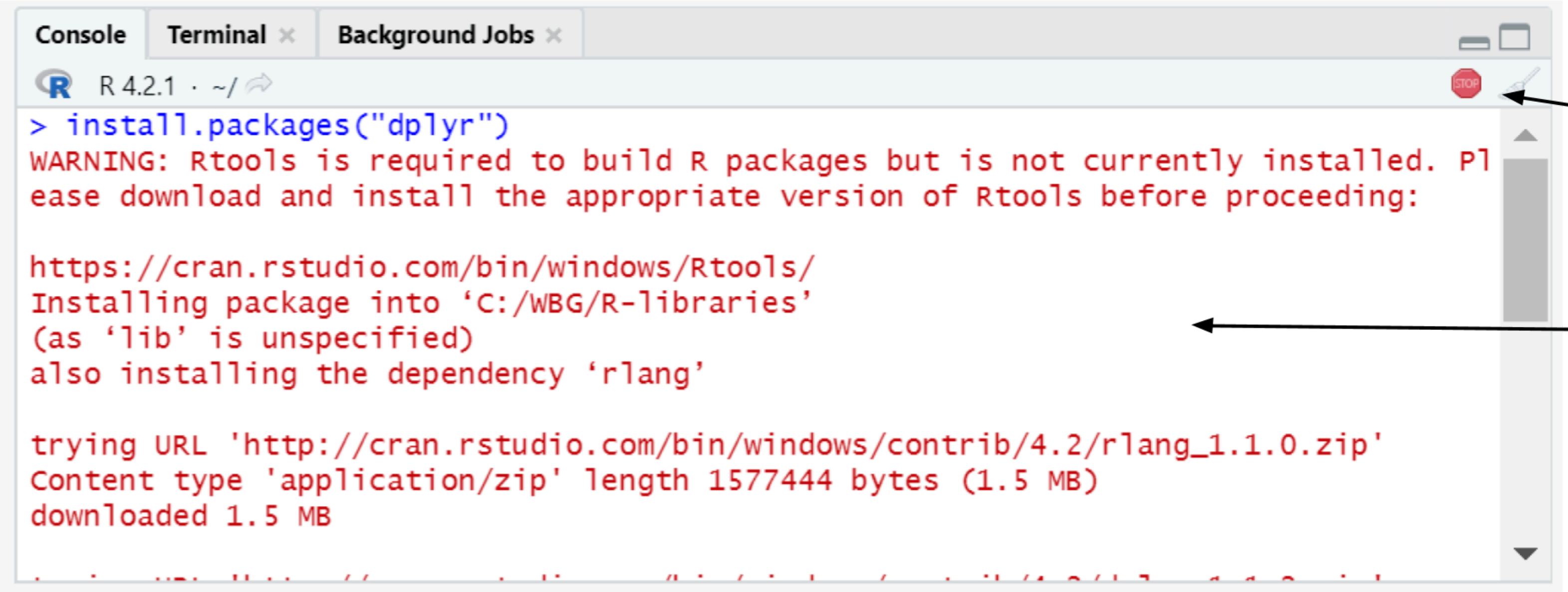
We'll use one package in today's session: `dplyr` and `tidyr` (really useful library for data cleaning and wrangling).

Exercise 1: Installing R Packages

1. Install the R Packages 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 packages



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

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

R packages

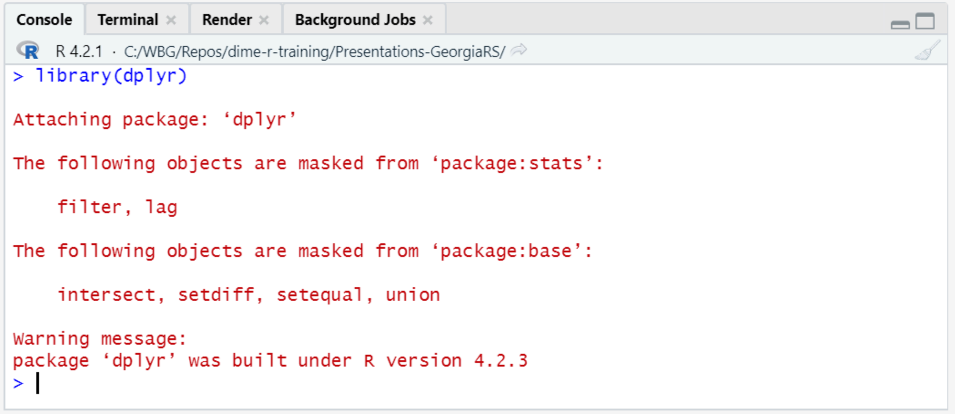
Now that `dplyr` is installed, we only need to load them to start using the functions they have.

Exercise 2: Loading packages

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

2. Load the packages with:

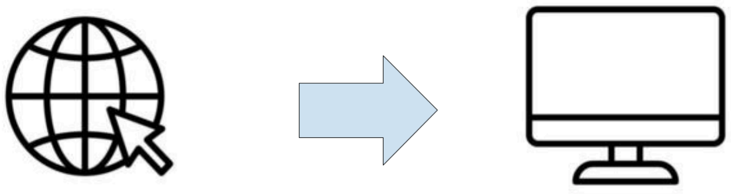
```r
library(dplyr)
```



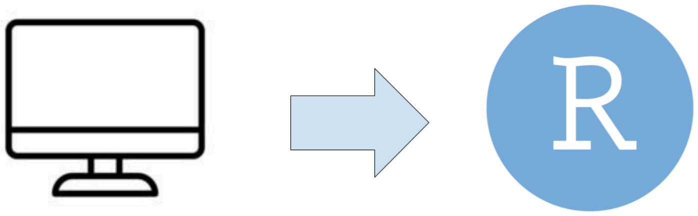
- Run this code from the new script you just opened
- Notice that we don't use quotes in the package names this time

R packages

- Library installation:



- Library loading:



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

Data wrangling

Data wrangling

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**

Input

department_staff_list.csv

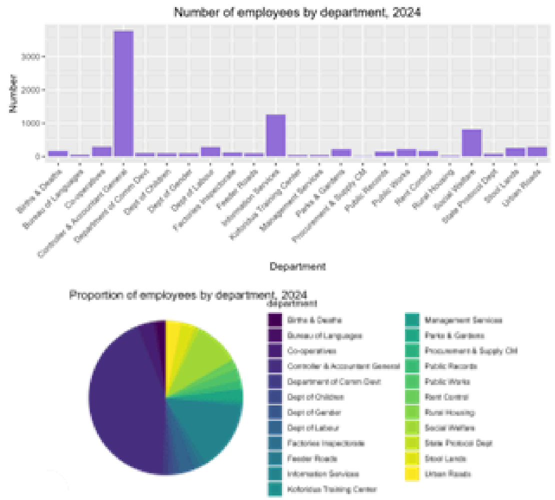
Data process

Instructions

1. Read data
2. Select columns
3. Obtain averages for all columns
4. Produce graph
5. Export graph in image

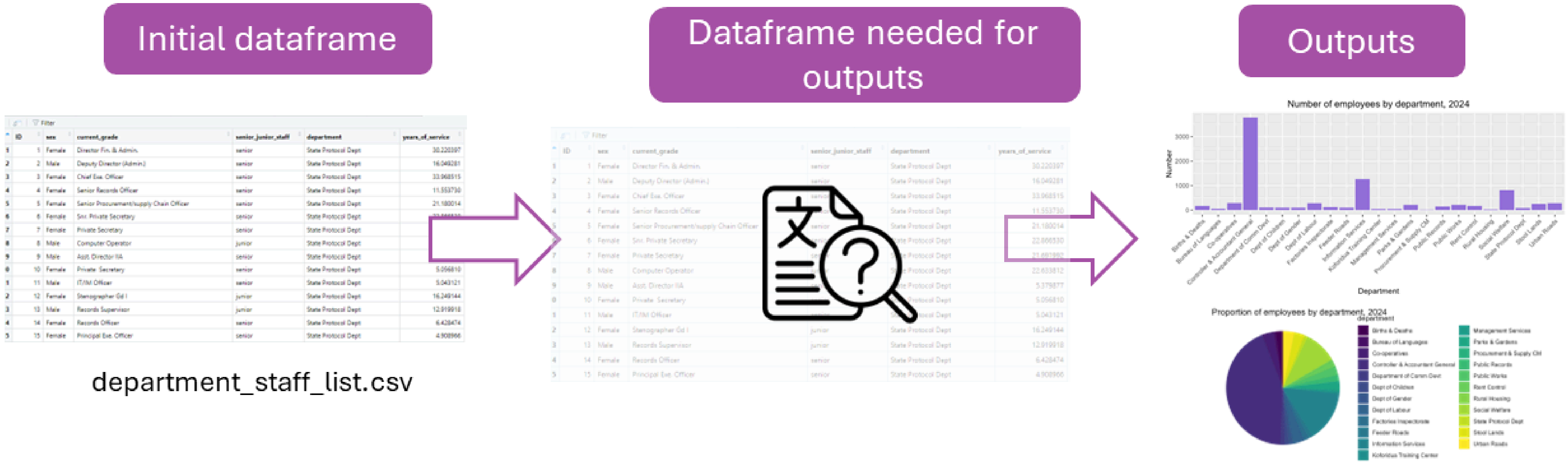
Data wrangling

Output



Getting your data ready

- Data wrangling is one of the most crucial and time-consuming aspects of analytics
- 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



Data wrangling

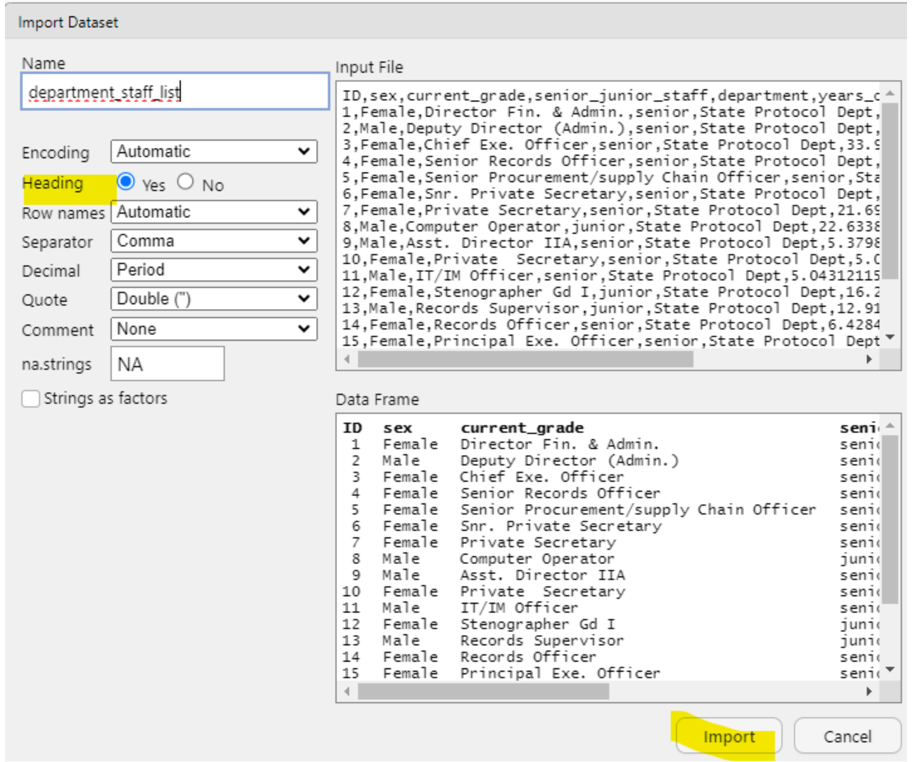
Getting your data ready

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

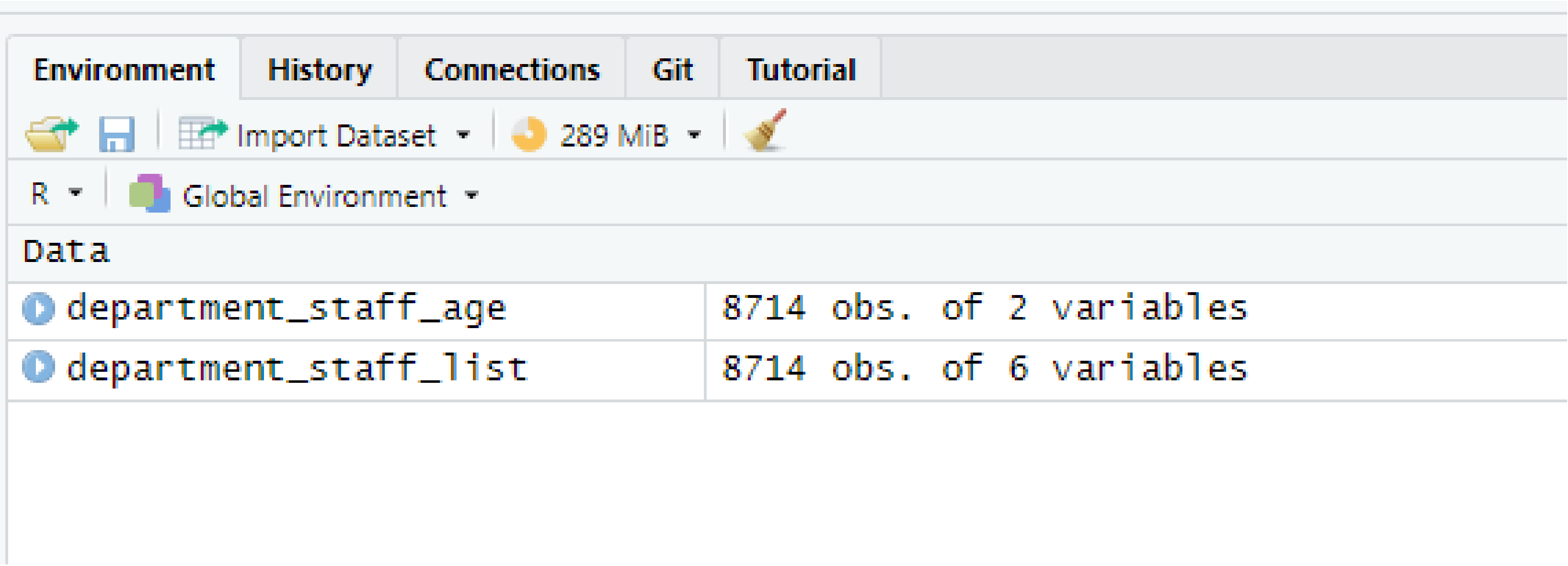
Exercise 3: Loading data

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 `department_staff_list.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/chdgj
2. Make sure to select `Heading` > `Yes` in the next window
3. Select `Import`
4. Download this new file `department_staff_age.csv`: https://osf.io/qsmke and repeat steps 1-3 with it



Data wrangling



Note: loading data with a function

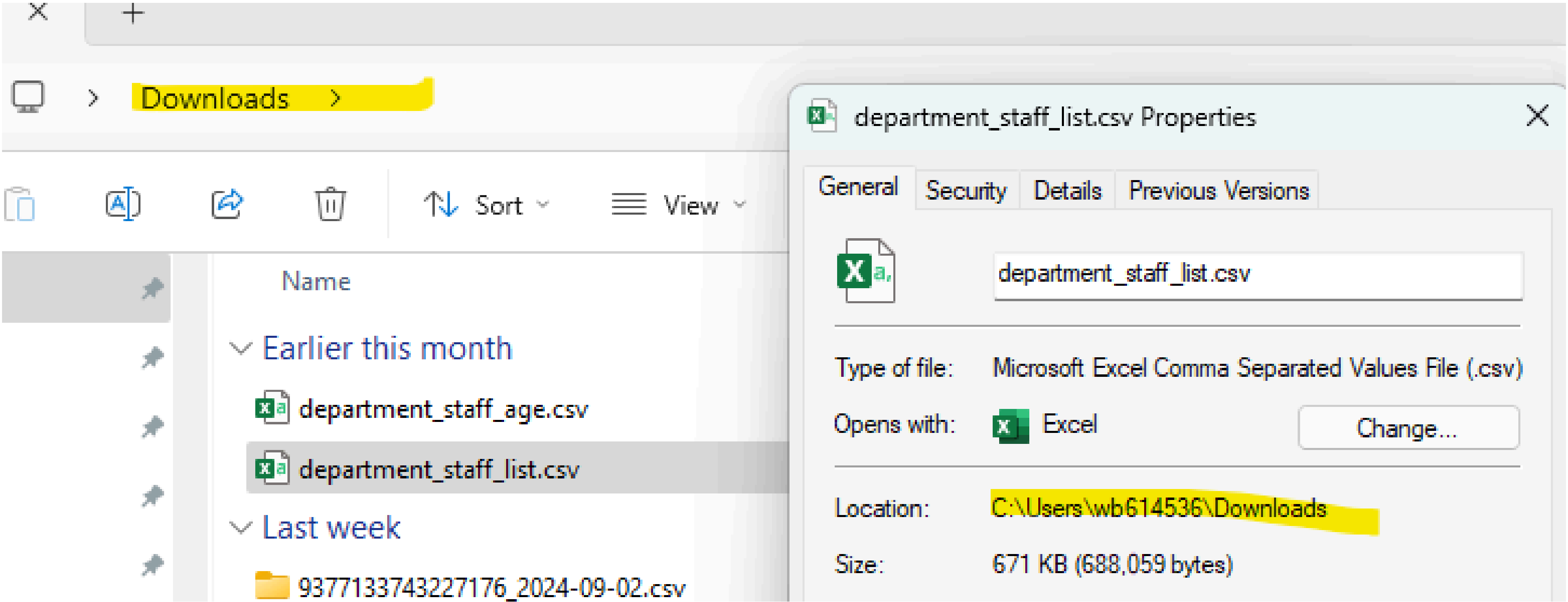
- You can also load csv files with the function `read.csv()` instead of using this point-and-click approach
- The **argument** of `read.csv()` is the path in your computer where your data is. For example

```
department_staff_list <-read.csv("C:/Users/wb614536/Downloads/department_staff_list.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

Note on file paths

- A **file path** tells R where to find your file on your computer. It is like giving R the directions to your file.
- If you downloaded the data and haven't moved it, the data path for the deparment_list dataset will be probably something like: `"C:/Users/wb614536/Downloads/department_staff_list.csv"`
- You can find the path of a file by right-clicking>properties>and seeing the location, or by clicking on the top bar (see below)



Recap: knowing your data

- Dataframe `department_staff_list` is the same dataframe we used last session that contains the data from your department.

```
glimpse(department_staff_list)
```

```
## Rows: 8,713
## Columns: 6
## $ ID                  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,…
## $ sex                 <chr> "Female", "Male", "Female", "Female", "Female", "F…
## $ current_grade       <chr> "Assist. Landscape Designer Grade II", "Co-Operati…
## $ senior_junior_staff <chr> "junior", "senior", "senior", "senior", "senior", …
## $ department          <chr> "Parks & Gardens", "Co-operatives", "Controller & …
## $ years_of_service    <dbl> 24.87, 17.83, 25.65, 2.76, 12.56, 23.75, 4.59, 3.2…
```

Recap: knowing your data

- Now we are introducing a second dataset that only has the ID number (one that I invented), and the age of that person.

```
glimpse(department_staff_age)
```

```
## Rows: 8,714
## Columns: 2
## $ ID  <int> 8603, 3695, 5563, 3529, 5032, 4391, 3069, 6048, 4680, 8551, 5107, …
## $ age <dbl> 18.00442, 18.00657, 18.00669, 18.00874, 18.01030, 18.01677, 18.018…
```

Data wrangling

- We will only use this second dataframe in one of the next exercises, but we load 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 propose scenarios that could show up while doing your annual reports or in day-to-day operations.

- We will do everything using **functions** from the `dplyr` package, that is already in our environment.

Filtering and sorting

Filtering and sorting

Filtering and sorting

Government analytics request

We will use the data from our dataframe department_staff_list

exercises-session2.R* department_staff_list

Filter

	ID	sex	current_grade
1	1	Female	Director Fin. & Admin.
2	2	Male	Deputy Director (Admin.)
3	3	Female	Chief Exe. Officer
4	4	Female	Senior Records Officer
5	5	Female	Senior Procurement/suppl
6	6	Female	Snr. Private Secretary
7	7	Female	Private Secretary
8	8	Male	Computer Operator
9	9	Male	Asst. Director IIA
10	10	Female	Private Secretary
11	11	Male	IT/IM Officer
12	12	Female	Stenographer Gd I

For this we can filter the Female rows from the sex category

Filter

	ID	sex	current_grade
1	1	Female	Director Fin. & Admin.
2	3	Female	Chief Exe. Officer
3	4	Female	Senior Records Officer
4	5	Female	Senior Procurement/supply Chain Officer
5	6	Female	Snr. Private Secretary
6	7	Female	Private Secretary
7	10	Female	Private Secretary
8	12	Female	Stenographer Gd I
9	14	Female	Records Officer
10	15	Female	Principal Exe. Officer
11	16	Female	Steno Secretary
12	17	Female	senior Technical Asst.
13	22	Female	Principal Protocol Officer

Filtering and sorting

Government analytics request

Now let's say that we are also interested at a first glance of the females that recently joined the department. We would have to do the following

1. Keeping only the female employees
2. Sorting by years of service

❎ In Excel: We would filter and then use arrange by years of service.

1. Keeping only the female employees

Use `filter()` for this:

Remember how we use functions in this case 1st argument is **data**, second argument is **filter** (sex takes the value female in this case)

```r
temp1 <- filter(department_staff_list, sex == "Female")
```

	ID	sex	current_grade	senior_junior_staff	department	years_of_service
1	1	Female	Director Fin. & Admin.	senior	State Protocol Dept	30.220397
2	3	Female	Chief Exe. Officer	senior	State Protocol Dept	33.968515
3	4	Female	Senior Records Officer	senior	State Protocol Dept	11.553730
4	5	Female	Senior Procurement/supply Chain Officer	senior	State Protocol Dept	21.180014
5	6	Female	Snr. Private Secretary	senior	State Protocol Dept	22.866530
6	7	Female	Private Secretary	senior	State Protocol Dept	21.691992
7	10	Female	Private Secretary	senior	State Protocol Dept	5.056810
8	12	Female	Stenographer Gd I	junior	State Protocol Dept	16.249144
9	14	Female	Records Officer	senior	State Protocol Dept	6.428474
10	15	Female	Principal Exe. Officer	senior	State Protocol Dept	4.908966
11	16	Female	Steno Secretary	senior	State Protocol Dept	15.134839
12	17	Female	senior Technical Asst.	junior	State Protocol Dept	32.134155
13	22	Female	Principal Protocol Officer	senior	State Protocol Dept	22.962355
14	30	Female	State Protocol Officer	senior	State Protocol Dept	5.464750
15	31	Female	State Protocol Officer	senior	State Protocol Dept	5.464750
16	32	Female	State Protocol Officer	senior	State Protocol Dept	5.464750
17	33	Female	Assistant Protocol Officer	senior	State Protocol Dept	5.286790

Showing 1 to 18 of 3,559 entries, 6 total columns

Filtering and sorting

2. Sorting by years of service

Use the function `arrange()` to sort. Sortings are ascending by default in R (this will get us from less years of service to more years of service).

```
department_female <- arrange(temp1, years_of_service)  # arrange by years of service
```

exercises-session2.R* department_female

Filter

	ID	sex	current_grade	senior_junior_staff	department	years_of_service
1	5003	Female	Records Supervisor	junior	Public Records	0.9171800
2	6497	Female	Asst. Prog Officer	senior	Dept of Gender	0.9527721
3	4969	Female	Assistant Internal Auditor	senior	Public Records	0.9993155
4	201	Female	PRINCIPAL.ACCOUNTS TECHNICIAN..CAGD	senior	Controller & Accountant General	1.1334702
5	232	Female	ASSISTANT.ACCOUNTANT..CAGD	senior	Controller & Accountant General	1.1498973
6	7555	Female	Assistant Information Officer	senior	Information Services	1.1608487
7	8568	Female	Assistant Information Officer	senior	Information Services	1.1854894
8	4890	Female	Procurement & Supply Chain Manager	senior	Procurement & Supply CM	1.1882272
9	8566	Female	Assistant Information Officer	senior	Information Services	1.1882272
10	8569	Female	Assistant Information Officer	senior	Information Services	1.1882272
11	8570	Female	Assistant Information Officer	senior	Information Services	1.1882272
12	8572	Female	Assistant Information Officer	senior	Information Services	1.1909651
13	8565	Female	Assistant Information Officer	senior	Information Services	1.2019165
14	7549	Female	Senior Information Assistant	junior	Information Services	1.2238193
15	3020	Female	ASSISTANT.ACCOUNTANT..CAGD	senior	Controller & Accountant General	1.2429843

Exercise 4: Now let's do it.. filter and sort your data

We can write the whole code for this in our exercise script. (You can copy and paste the code below)

1.- Filter by `Female`:

2.- Sort by `years_of_service`:

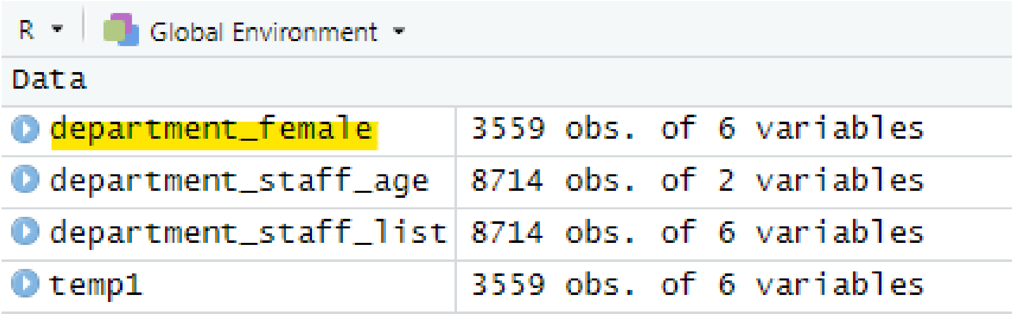
```
temp1 <-  filter(department_staff_list, sex == "Female") # filter by female

department_female <- arrange(temp1, years_of_service) # order by years of service
```

Filtering and sorting

Some notes:

- `filter()` and `arrange()` are all functions from `dplyr`. Remember you have to always load `dplyr` first with `library(dplyr)` to be able to use them
- The resulting dataframe is `department_female` (and we also have `temp1`)



Filtering and sorting

- 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: **mutate**, which basically means creating new variables

Mutate

Mutate

- `mutate` will take a statement like this:

```
mutate(variable_name = some_calculation)
```

- And attach variable_name at the end of the dataset.

Mutate

Example

Using our data frame let's say that we want to include a variable that instead of years of service we want days of service. We would do this by doing the following:

```
example_mutate <-  mutate(department_staff_list, days_of_service = years_of_service*365)
```

We will not use that variable for our government analytics examples, but this is a really useful data wrangling function.

Questions?

Merging dataframes

Merging dataframes

Merging data is a common task in data analysis, especially when working with data from multiple departments.

Let's see how it would apply to our dataframes.

Government analytics request

Scenario 2:

Let's imagine that for our annual report we are also interested in the age distribution from the employees but these are in different datasets

Government analytics request

Use the data `department_staff_list` *that you already know with the department_staff_age*

We want to include the `age` *variable to our dataframe*

ID	sex	current_grade	senior_junior_staff	department	years_of_service
1	Female	Director Fin. & Admin.	senior	State Protocol Dept	30.220397
2	Male	Deputy Director (Admin.)	senior	State Protocol Dept	16.049281
3	Female	Chief Exe. Officer	senior	State Protocol Dept	33.968515
4	Female	Senior Records Officer	senior	State Protocol Dept	11.553730
5	Female	Senior Procurement/supply Chain Officer	senior	State Protocol Dept	21.180014
6	Female	Snr. Private Secretary	senior	State Protocol Dept	22.866530
7	Female	Private Secretary	senior	State Protocol Dept	21.691992
8	Male	Computer Operator	junior	State Protocol Dept	22.633812
9	Male	Asst. Director IIA	senior	State Protocol Dept	5.379877
10	Female	Private Secretary	senior	State Protocol Dept	5.056810
11	Male	IT/IM Officer	senior	State Protocol Dept	5.043121
12	Female	Stenographer Gd I	junior	State Protocol Dept	16.249144
13	Male	Records Supervisor	junior	State Protocol Dept	12.919918
14	Female	Records Officer	senior	State Protocol Dept	6.428474
15	Female	Principal Exe. Officer	senior	State Protocol Dept	4.908966
16	Female	Steno Secretary	senior	State Protocol Dept	15.134839

+

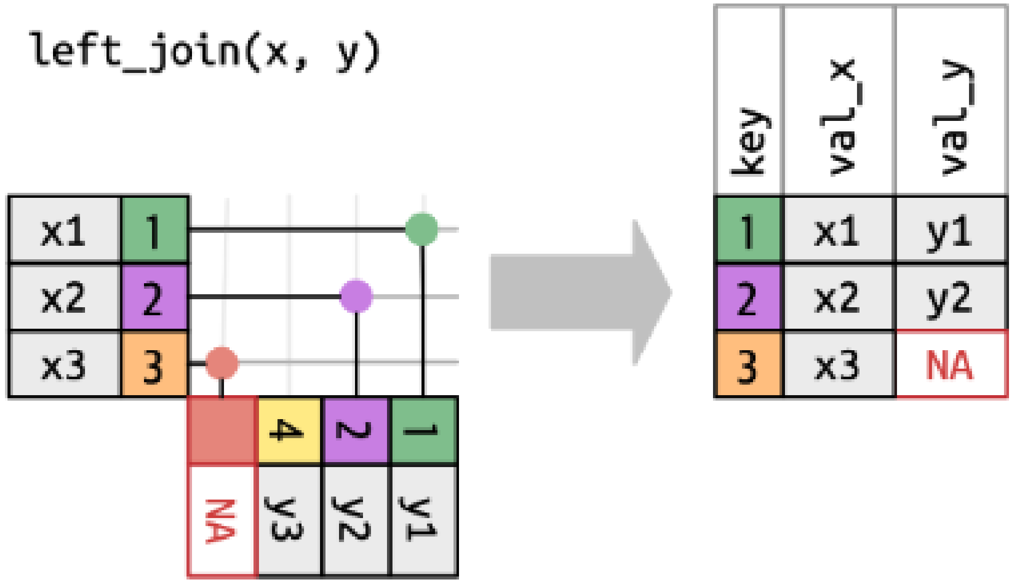
ID	age
1	60.15058
2	41.74949
3	59.43053
4	50.55989
5	51.28268
6	45.22656
7	41.90007
8	56.56400
9	29.09788
10	38.76523
11	32.14784
12	38.01506
13	34.16564
14	43.42505
15	44.99932
16	43.50171

Merging dataframes

To do this we will use `left_join()` to merge the dataframes:

- The arguments of the function are 1. the "principal" dataset and 2. the dataframe we want to merge (paste) into that one

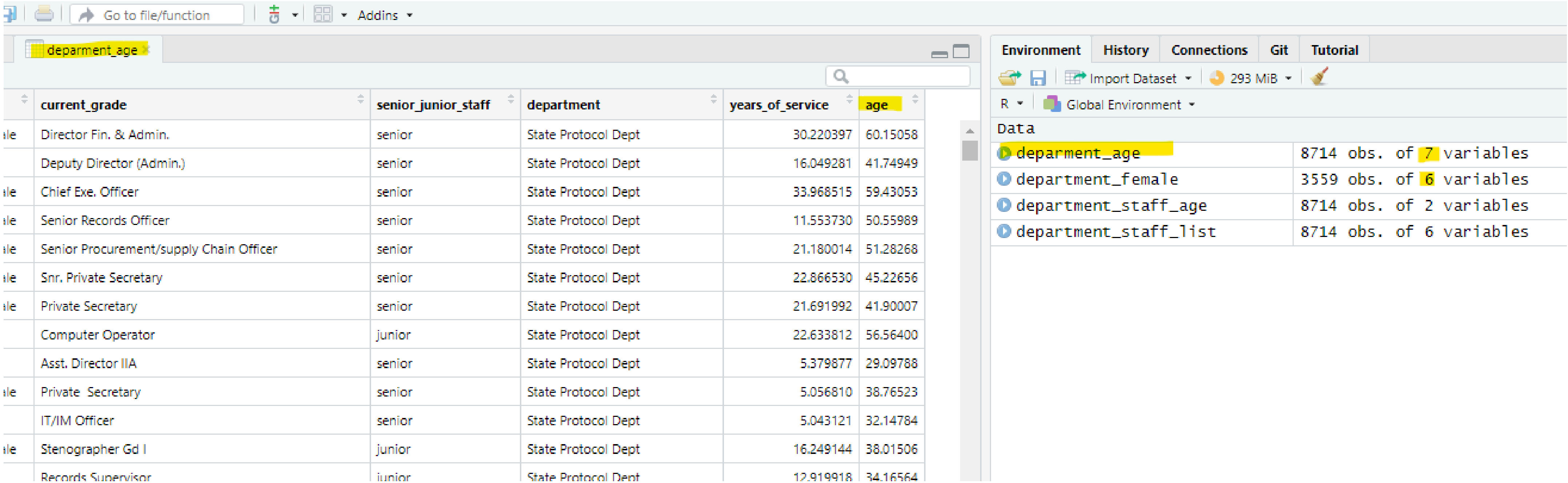
```
department_age <- left_join(department_staff_list, department_staff_age) # Our original data frame
```



Exercise 5: Now let's do it.

- You can copy and paste the code to your exercise script.

```
deparment_age <- left_join(deparment_staff_list, department_staff_age) # Our original data frame
```

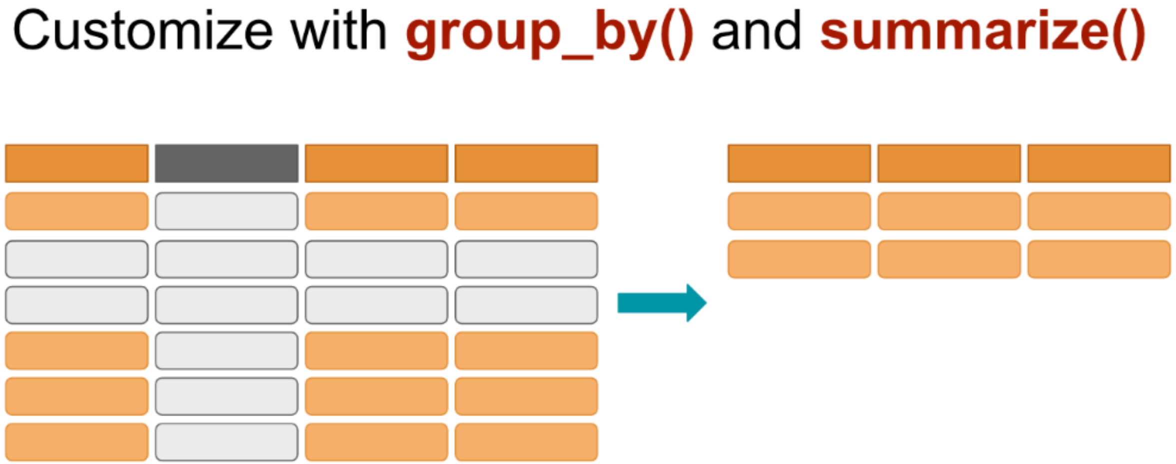


	current_grade	senior_junior_staff	department	years_of_service	age
ale	Director Fin. & Admin.	senior	State Protocol Dept	30.220397	60.15058
	Deputy Director (Admin.)	senior	State Protocol Dept	16.049281	41.74949
ale	Chief Exe. Officer	senior	State Protocol Dept	33.968515	59.43053
ale	Senior Records Officer	senior	State Protocol Dept	11.553730	50.55989
ale	Senior Procurement/supply Chain Officer	senior	State Protocol Dept	21.180014	51.28268
ale	Snr. Private Secretary	senior	State Protocol Dept	22.866530	45.22656
ale	Private Secretary	senior	State Protocol Dept	21.691992	41.90007
	Computer Operator	junior	State Protocol Dept	22.633812	56.56400
	Asst. Director IIA	senior	State Protocol Dept	5.379877	29.09788
ale	Private Secretary	senior	State Protocol Dept	5.056810	38.76523
	IT/IM Officer	senior	State Protocol Dept	5.043121	32.14784
ale	Stenographer Gd I	junior	State Protocol Dept	16.249144	38.01506
	Records Supervisor	junior	State Protocol Dept	12.919918	34.16564

group_by() and summarize()

We use this when we want to aggregate your data (by groups).

This is one of the most commons operations. Sometimes we want to calculate statistics by groups



Government analytics request

Let's say that we are interested again in the number of employees by department

We will use our department_staff_list dataset to do this.

1. The first step is to define the group, in this case `department`

```
employees_by_deparment <- group_by(department_staff_list, department)
```

If we **only** do this, this won't do anything, to complete the function we need to use this with `summarise()`

Exercise 6: Government analytics request

summarize works in a similar way to mutate:

```
variable_name = some_calculation
```

In this case the `some_calculation` will be to count the number of employees

```
temp1 <- group_by(department_staff_list, department)
employees_by_department <- summarise(temp1, number = n())
```

Government analytics request

This will create the following dataframe/table:

```
## # A tibble: 23 × 2
##    department                     number
##    <chr>                           <int>
##  1 Births & Deaths                   172
##  2 Bureau of Languages                58
##  3 Co-operatives                     293
##  4 Controller & Accountant General  3776
##  5 Department of Comm Devt           109
##  6 Dept of Children                  101
##  7 Dept of Gender                    100
##  8 Dept of Labour                    290
##  9 Factories Inspectorate            121
## 10 Feeder Roads                      102
## # ℹ 13 more rows
```

More wrangling operations

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 (**tomorrow**))

Exporting outputs

Exporting outputs

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

Input

department_staff_list.csv

Data process

Instructions

1. Read data
2. Select columns
3. Sort
4. Filter rows
5. Create averages
6. Produce dataframe

employees_by_department.csv

Output

Exporting dataframes

- We can export it as a csv file 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

Exporting outputs

Exercise 7 (if time allows): Export `employees_by_deparment` and `department_staff_final`

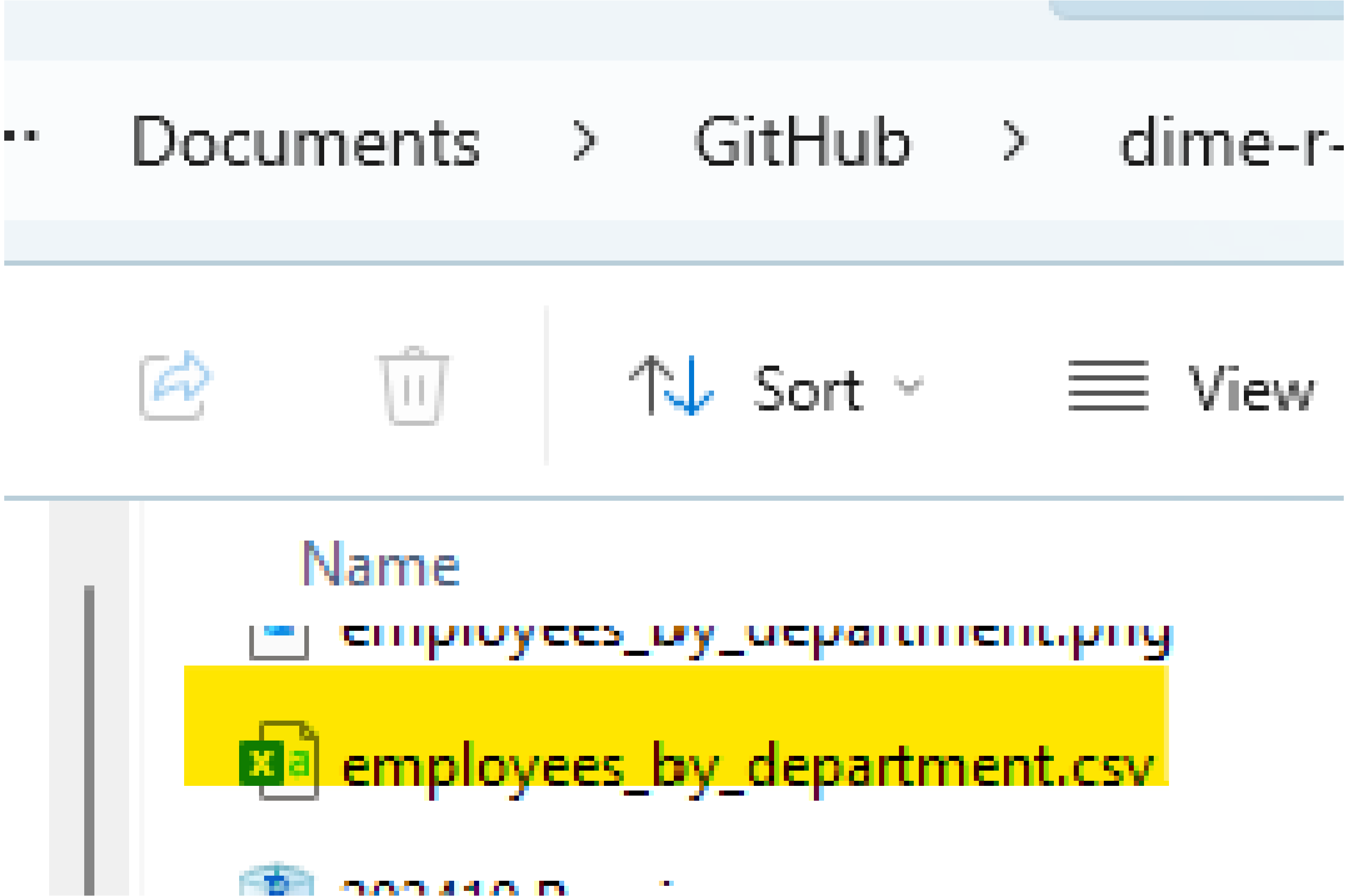
1. Use this code to export the results of the last two exercises:

```
write.csv(employees_by_deparment, "employees_by_deparment", row.names = FALSE)
write.csv(department_age, "department_staff_final", row.names = FALSE)
```

Note: This files are already in our data folder online, so if you don't have time to do it don't worry.

Exporting outputs

Now `employees_by_deparment.csv` (probably in your `Documents` folder).



Exporting outputs

Some notes on file paths

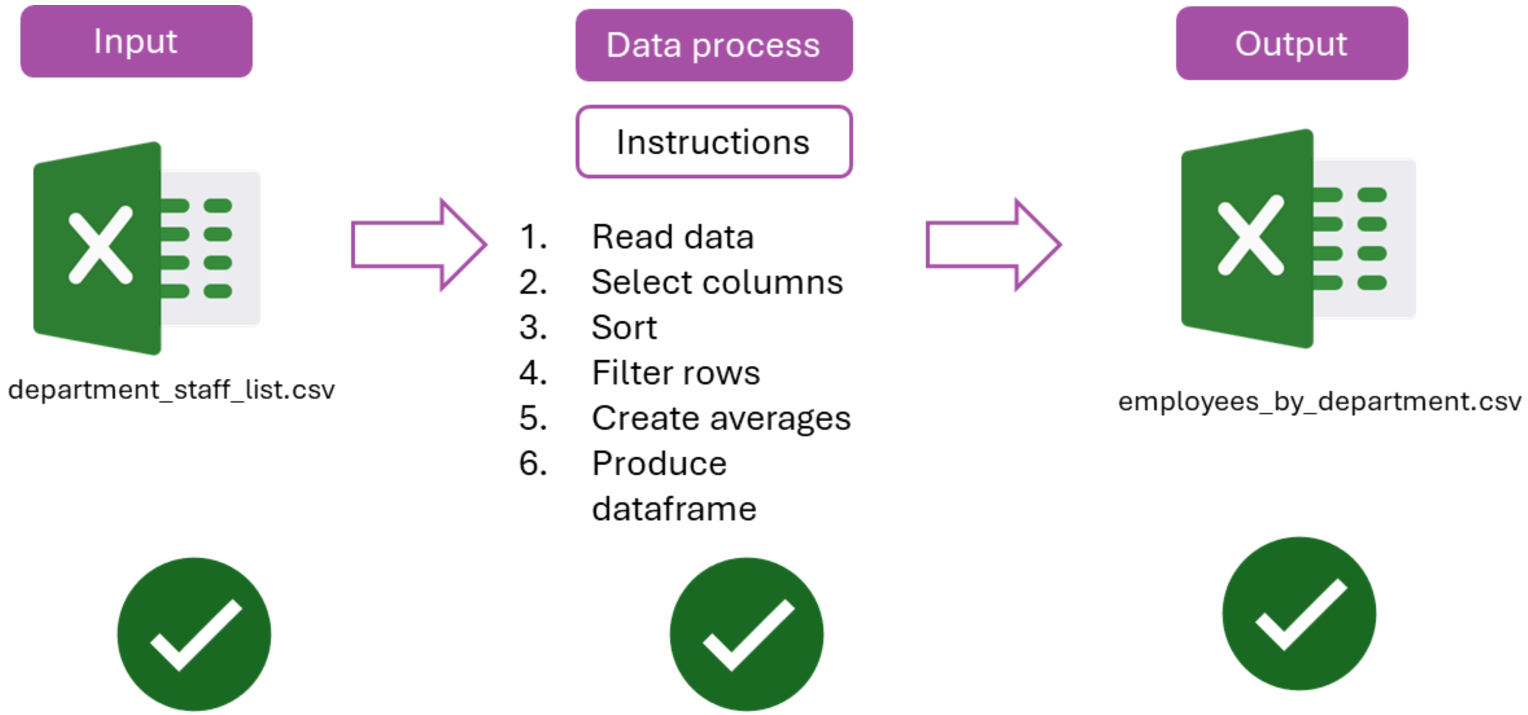
- The second argument of `write.csv()` specifies the file path we export the dataframe to

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

- You can include any path in your computer and R will write the file in that location
    - For example: `"C:/Users/wb614536/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

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

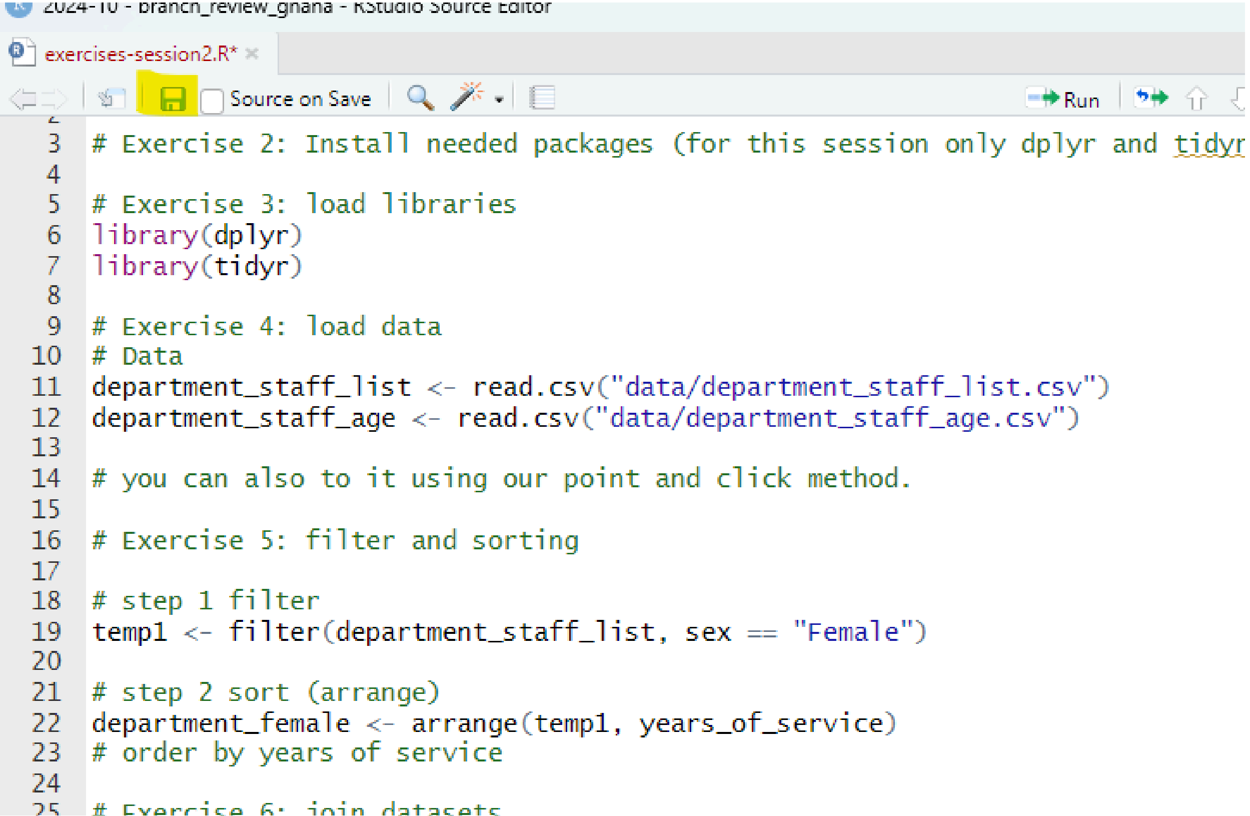


Wrapping up

Wrapping up

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

Government analytics pipeline

Input

department_staff_list.csv

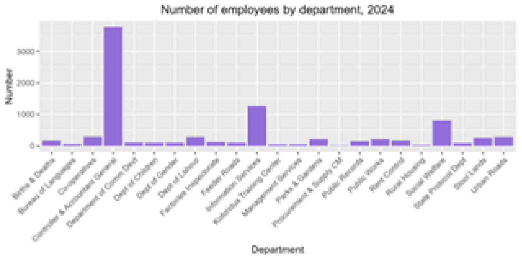
Data process

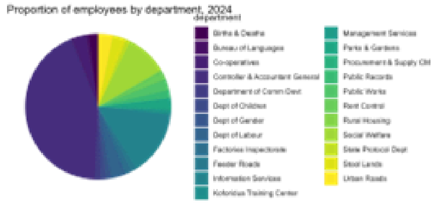
Instructions

1. Read data
2. Select columns
3. Sort
4. Filter rows
5. Create averages
6. Produce dataframe

employees_by_department.csv

Output





Wrapping up

Government analytics pipeline

Input

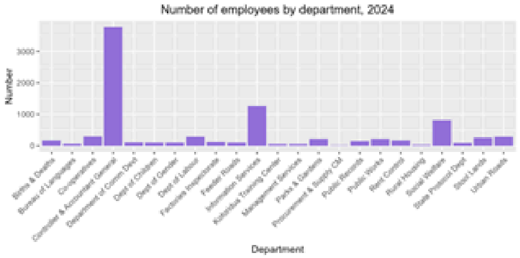
department_staff_list.csv

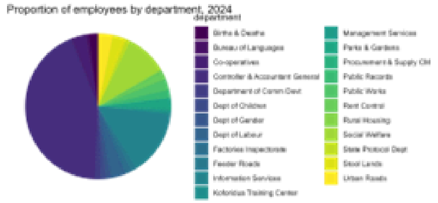
Data process

Instructions

1. Read data
2. Select columns
3. Sort
4. Filter rows
5. Create averages
6. Produce dataframe

Output





employees_by_department.csv

Tomorrow: more refined outputs

Thanks! // ¡Gracias! // Obrigado!

Appendix

Keeping only relevant columns

Imaging I only want a list with ID and department.

Use `select()` for this:

```
temp1 <-  select(department_staff_list, ID, department)
```

Calculating aggregated columns: total and average income

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

For our current data this does not apply, but this will be really useful for budget dataframes.

This would look something like this

```
budget_2024 <-  group_by(budget_data, year, product)

budget_2024 <- summarise(budget_2024,
                         total = sum(income),
                         average = mean(income))
```

Useful links

- **R for Data Science** by Hadley Wickham and Garrett Grolemund
  - Comprehensive introduction to data wrangling and visualization.
  - Read it online for free.
- **Tidyverse Cookbook**
  - Practical solutions for common data wrangling tasks.
  - Tidyverse Cookbook GitHub.
- **Tidyverse Cheat Sheets**
  - Official cheat sheets for `dplyr`, `tidyr`, and other Tidyverse packages.
  - Tidyverse Resources.