UNIVERSITY OF AMSTERDAM

CS Staff Course:

Using and Teaching Python

Day 3

Dr. Joanna Strycharz & Dr. A. Marthe Möller

Day 3

- Checking in
- Recap: Data aggregation
- Recap: Data visualization
- Warm-up exercise 5
- In-class exercise 1
- Recap: Where do data come from?
- Warm-up exercise 6
- In-class exercise 2
- Teaching exercise

Checking in: How is it going?

University of Amsterdam

Data aggregation

From rows to groups of rows

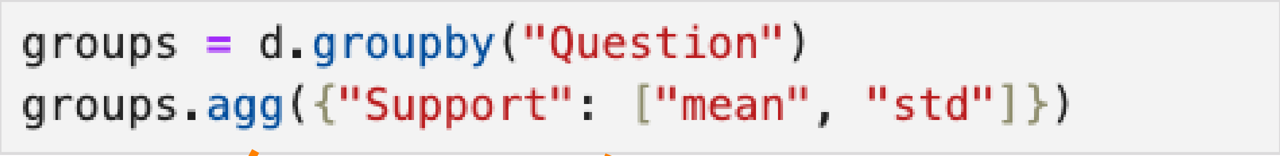
- So far, we’ve used functions on individual rows
- Applying them on groups of row can be useful: higher level of abstraction
- Two steps in data aggregation:

1. Define which rows are grouped together based on columns
2. Specify one or multiple summary (aggregation) functions (e.g., mean, sum)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6.3). and the materials of the course CCS-1.

Data aggregation: Two steps, two lines

Name that was given to the dataset



```
groups = d.groupby("Question")
groups.agg({"Support": ["mean", "std"]})
```

Step 1: Use groupby to indicate on which columns you want the rows to be grouped. In this case, it will be grouped by a column named Question

Step 2: Use the aggregate function

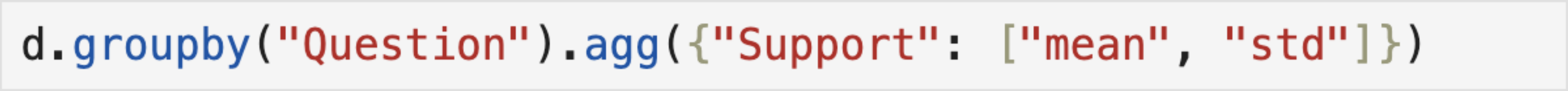
Step 2: Specifically, give the *M* and *SD* doe for the variable Support for the groups as categorized based on the Question variable (Step 1)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6.3). and the materials of the course CCS-1.

Data aggregation: Two steps, one line

Name that was given to the dataset

Step 1: Use groupby to indicate on which columns you want the rows to be grouped. In this case, it will be grouped by a column named Question



```
d.groupby("Question").agg({"Support": ["mean", "std"]})
```

Step 2: Use the aggregate function

Step 2: Specifically, give the *M* and *SD* doe for the variable Support for the groups as categorized based on the Question variable (Step 1)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6.3). and the materials of the course CCS-1.

Data aggregation: Two steps

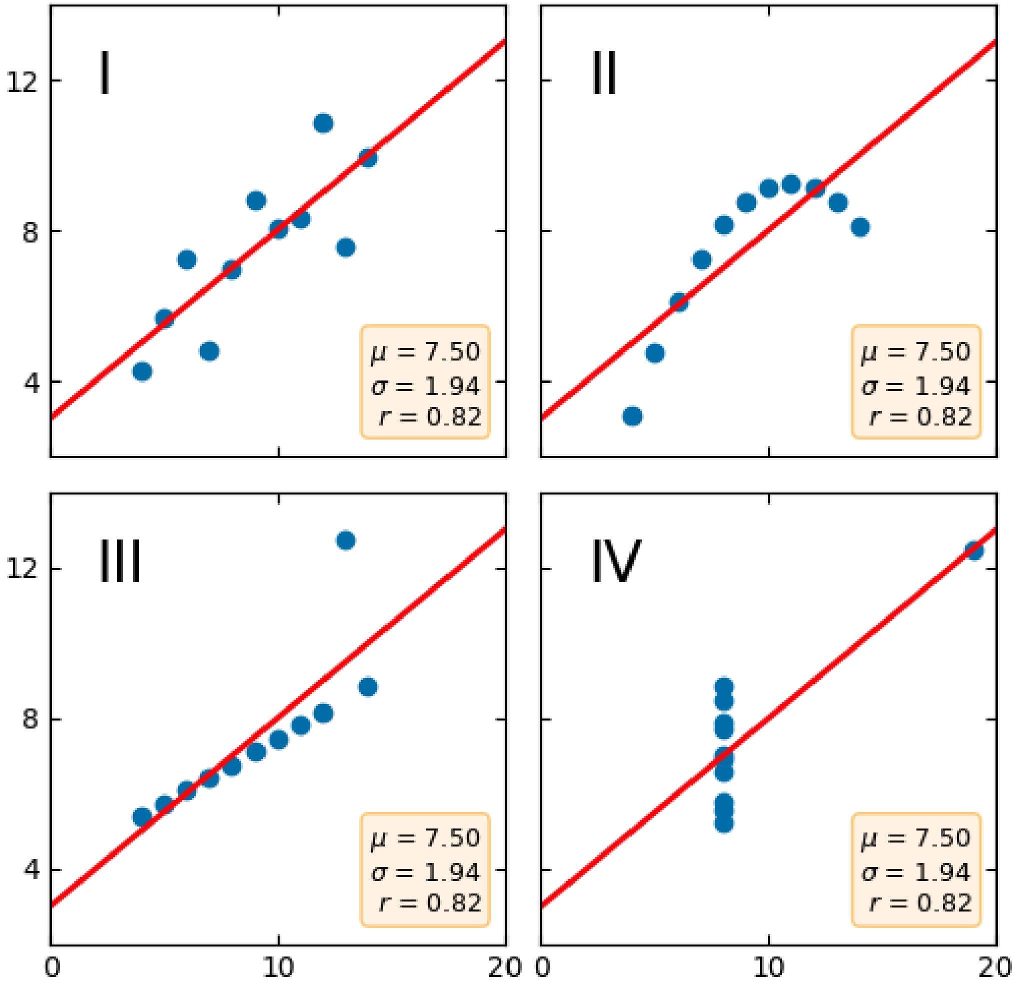
- I could already do that with the describe function used in week 2!
- True, but this way is more flexible
- Note: You can add your new variable to the dataset (e.g., if you want to compare individuals’ scores to their group average)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 6.3). and the materials of the course CCS-1.

Data visualization

Data visualization: Why would you?

- Relatively little attention in bachelor stats-courses
- Why bother? Anscombe's quartet!
- Often, visualization of *Exploratory Data Analysis (EDA)*



This slide is based on the materials of the course CCS-1. Image retrieved from: https://matplotlib.org/stable/gallery/specialty_plots/anscombe.html

Stop – let’s recap!

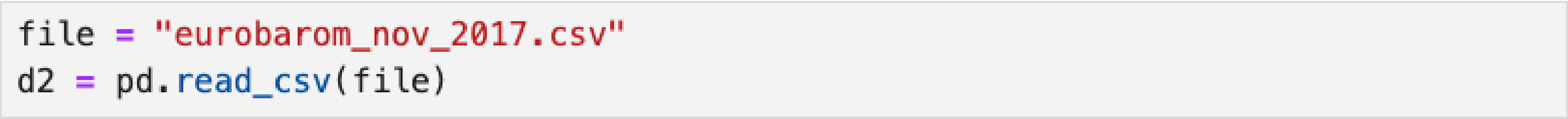
Let’s recap: What happens when?

- Data wrangling: Transforming raw data into a shape that is suitable for analyses
  - Exploratory Data Analyses: Get to know your data a bit
  - Detect and deal with missing values
  - Any other steps you need (e.g., create a new variable based on data aggregation and add it to your dataset)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

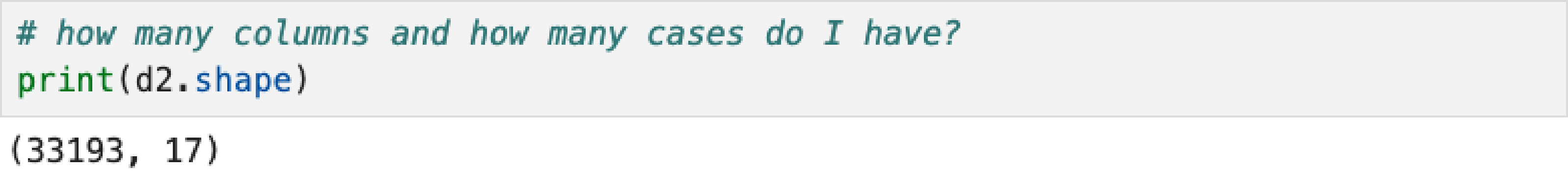
Process: Basic data exploration

Read in a dataset (called ‘d2’):



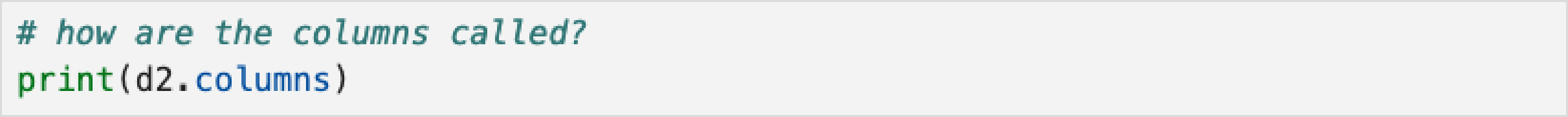
```
file = "eurobarom_nov_2017.csv"
d2 = pd.read_csv(file)
```

Explore the very basics:



```
# how many columns and how many cases do I have?
print(d2.shape)
```

```
(33193, 17)
```

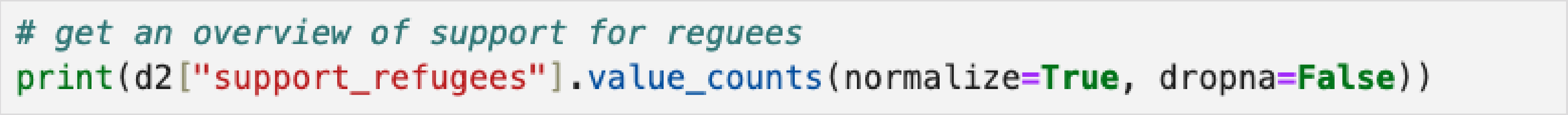


```
# how are the columns called?
print(d2.columns)
```

This slide is based on Van Atteveldt et al. (2022). [Computational Analysis of Communication]() (chapter 7). and the materials of the [course CCS-1]().

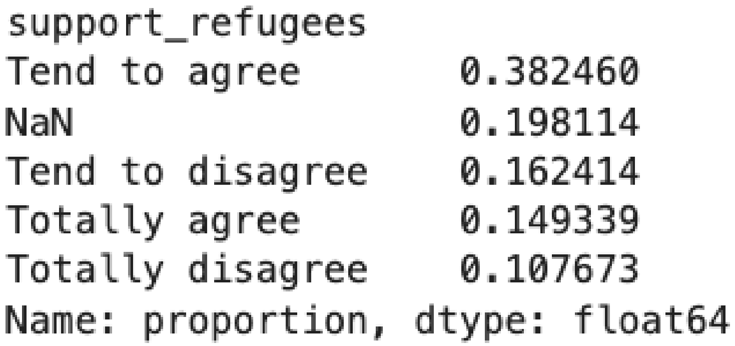
Process: Detecting and dealing with missing data

Explore (one of) the variable(s) we are interested in:



```
# get an overview of support for reguees
print(d2["support_refugees"].value_counts(normalize=True, dropna=False))
```

Aaah, we have missing data!

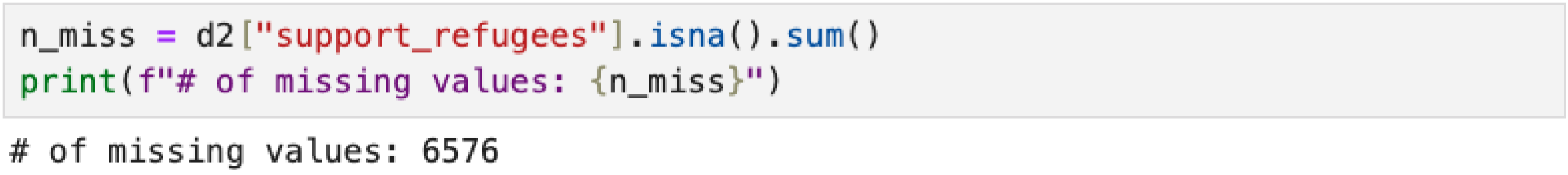


```
support_refugees
Tend to agree        0.382460
NaN                  0.198114
Tend to disagree     0.162414
Totally agree        0.149339
Totally disagree     0.107673
Name: proportion, dtype: float64
```

This slide is based on Van Atteveldt et al. (2022). [Computational Analysis of Communication]() (chapter 7). and the materials of the [course CCS-1]().

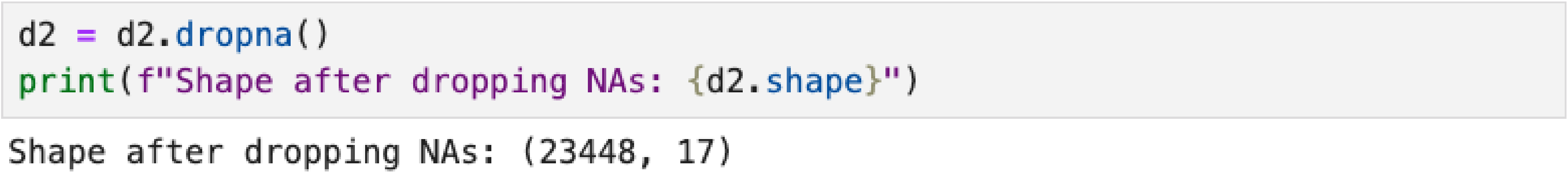
Process: Detecting and dealing with missing data

Let’s drop all records that contains NaN:



```
n_miss = d2["support_refugees"].isna().sum()
print(f"# of missing values: {n_miss}")
```

```
# of missing values: 6576
```



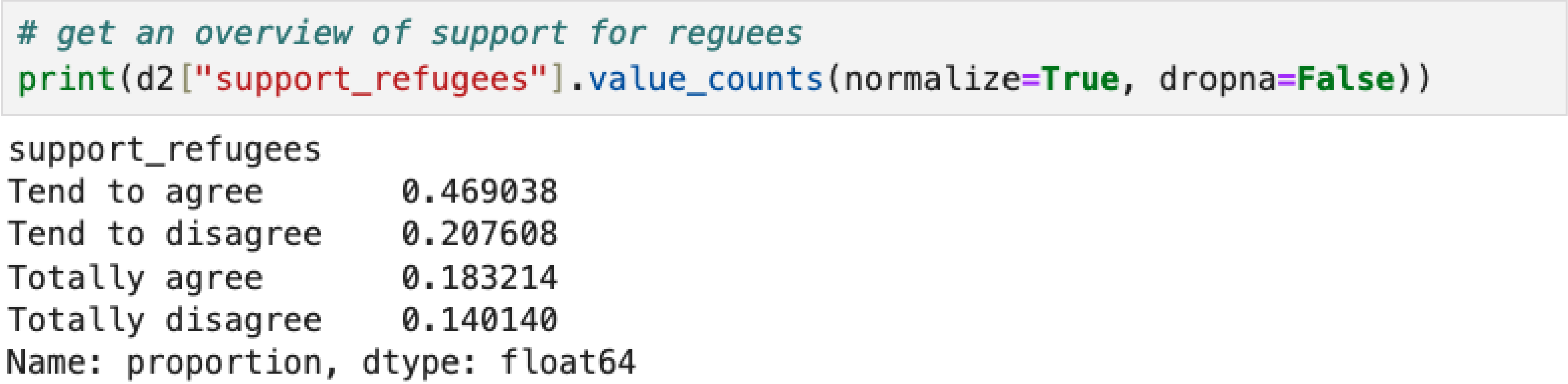
```
d2 = d2.dropna()
print(f"Shape after dropping NAs: {d2.shape}")
```

```
Shape after dropping NAs: (23448, 17)
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Process: Visualizing

Pfew, looks better:



```
# get an overview of support for reguees
print(d2["support_refugees"].value_counts(normalize=True, dropna=False))
```

```
support_refugees
Tend to agree        0.469038
Tend to disagree     0.207608
Totally agree        0.183214
Totally disagree     0.140140
Name: proportion, dtype: float64
```

Now, we can go on to visualizing!

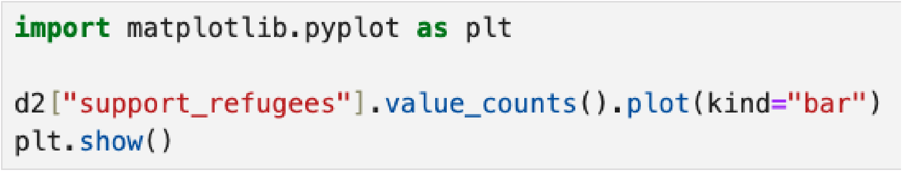
This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Visualizing

- Basic approach: matplotlib & seaborn
- Different types of plots, amongst others:
  - Plotting frequencies/frequency distribution
  - Plotting relationships
  - Plotting geospatial data
  - And more

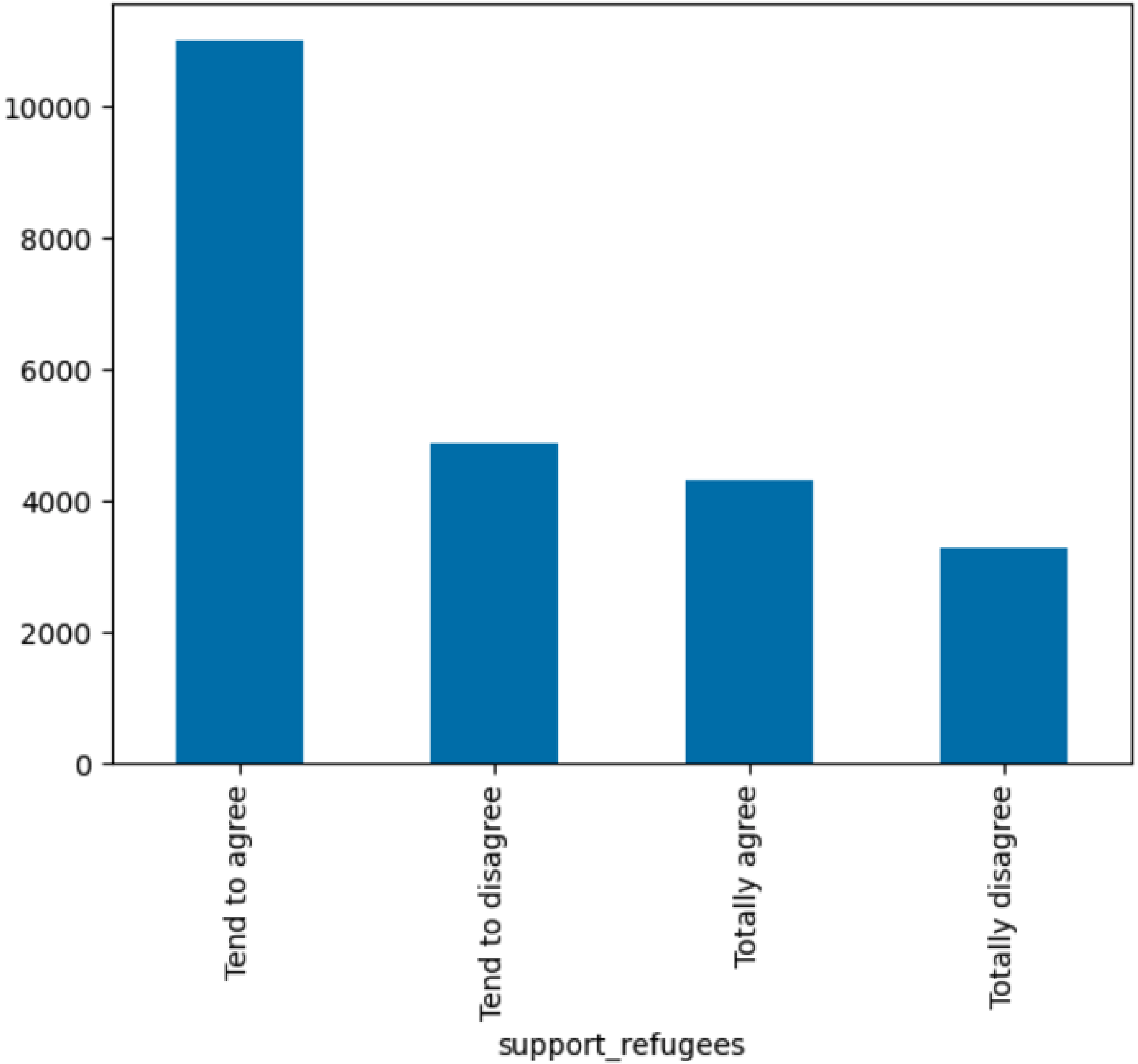
This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Plotting frequencies



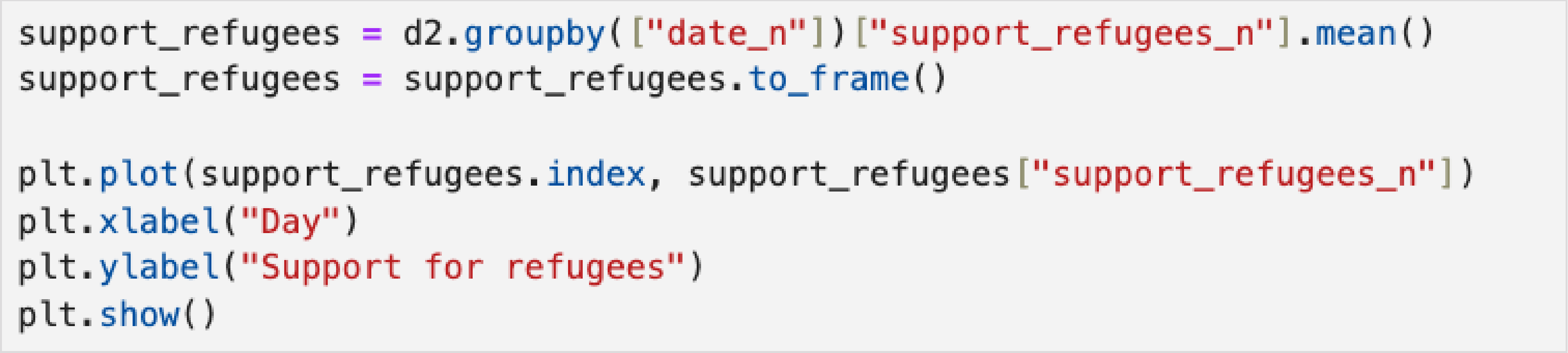
```
import matplotlib.pyplot as plt

d2["support_refugees"].value_counts().plot(kind="bar")
plt.show()
```



This slide is based on Van Atteveldt et al. (2022). [Computational Analysis of Communication] (chapter 7). and the materials of the [course CCS-1].

Plotting one relationship

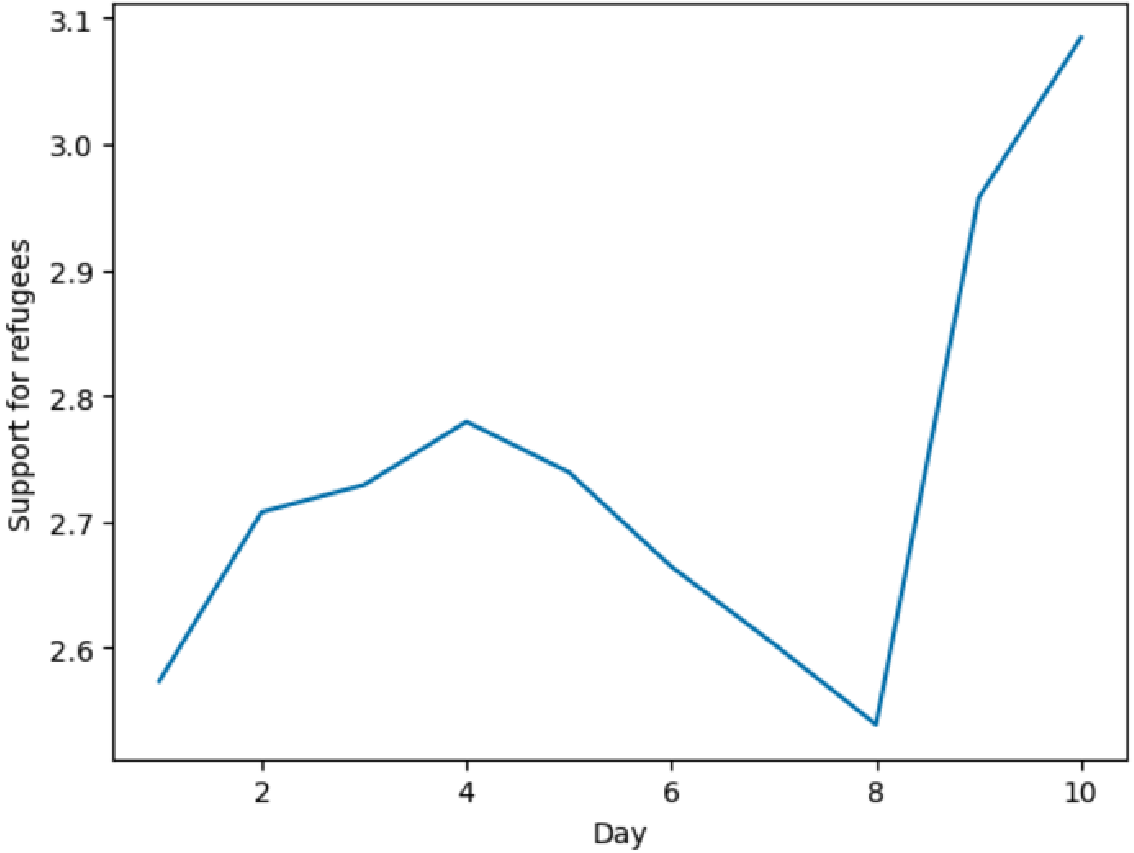


```
support_refugees = d2.groupby(["date_n"])["support_refugees_n"].mean()
support_refugees = support_refugees.to_frame()

plt.plot(support_refugees.index, support_refugees["support_refugees_n"])
plt.xlabel("Day")
plt.ylabel("Support for refugees")
plt.show()
```

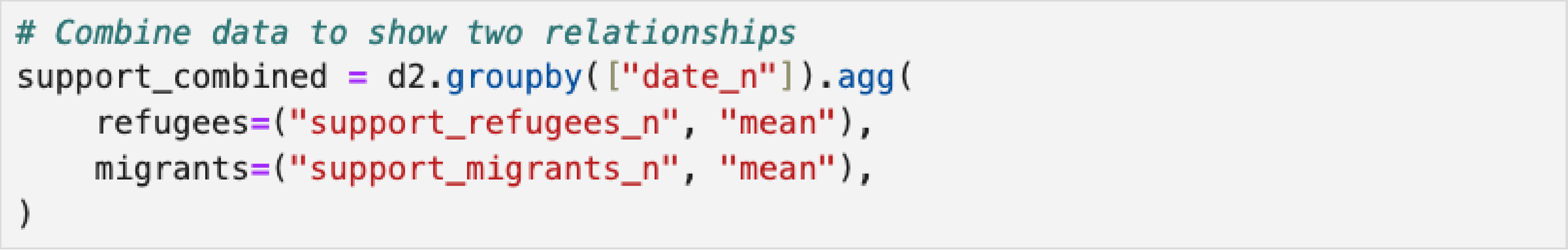
This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Plotting one relationship

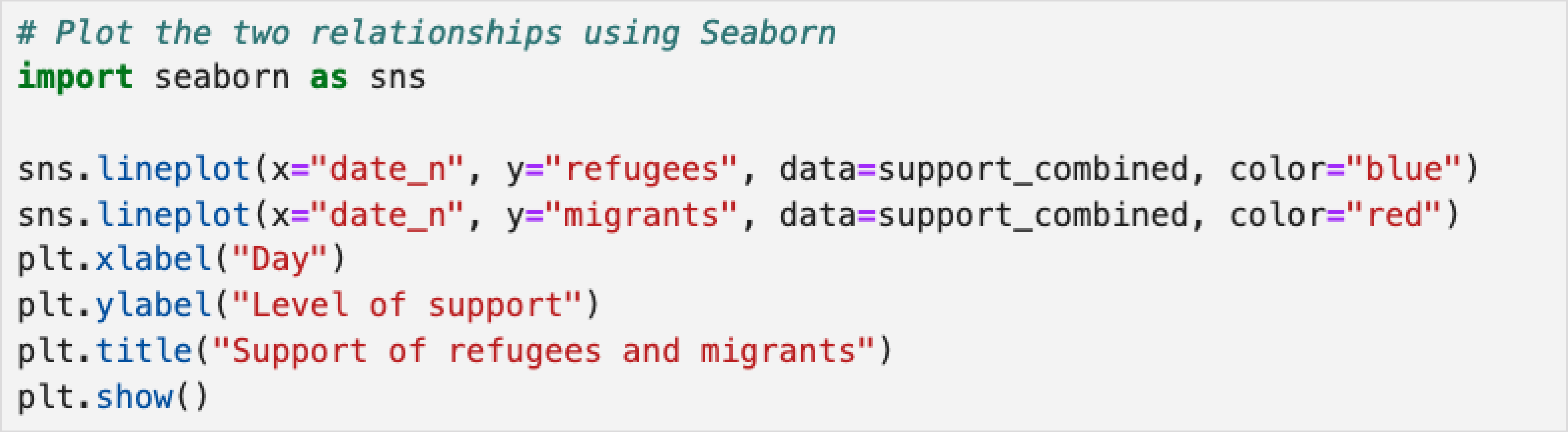


This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Plotting multiple relationships



```python
# Combine data to show two relationships
support_combined = d2.groupby(["date_n"]).agg(
    refugees=("support_refugees_n", "mean"),
    migrants=("support_migrants_n", "mean"),
)
```

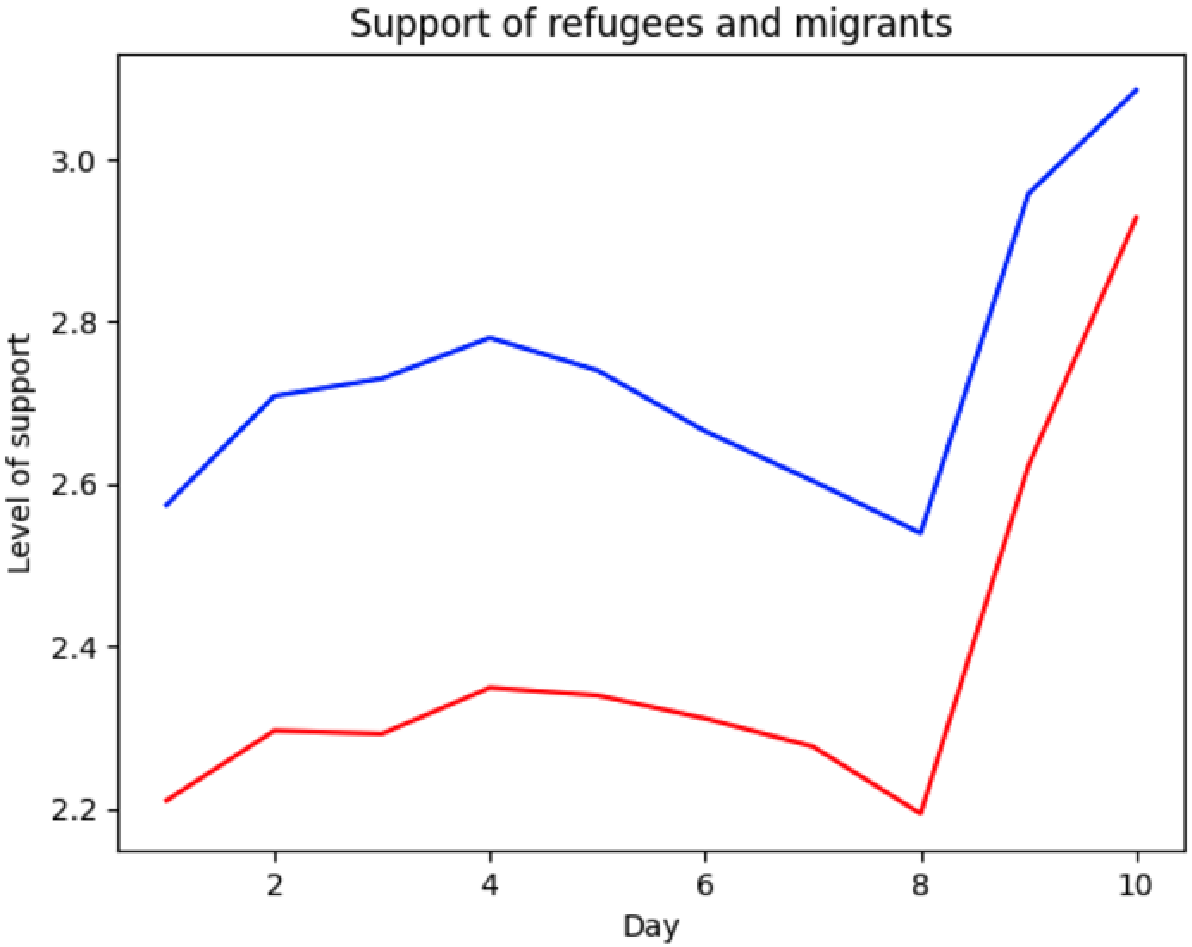


```python
# Plot the two relationships using Seaborn
import seaborn as sns

sns.lineplot(x="date_n", y="refugees", data=support_combined, color="blue")
sns.lineplot(x="date_n", y="migrants", data=support_combined, color="red")
plt.xlabel("Day")
plt.ylabel("Level of support")
plt.title("Support of refugees and migrants")
plt.show()
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Plotting multiple relationships



This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

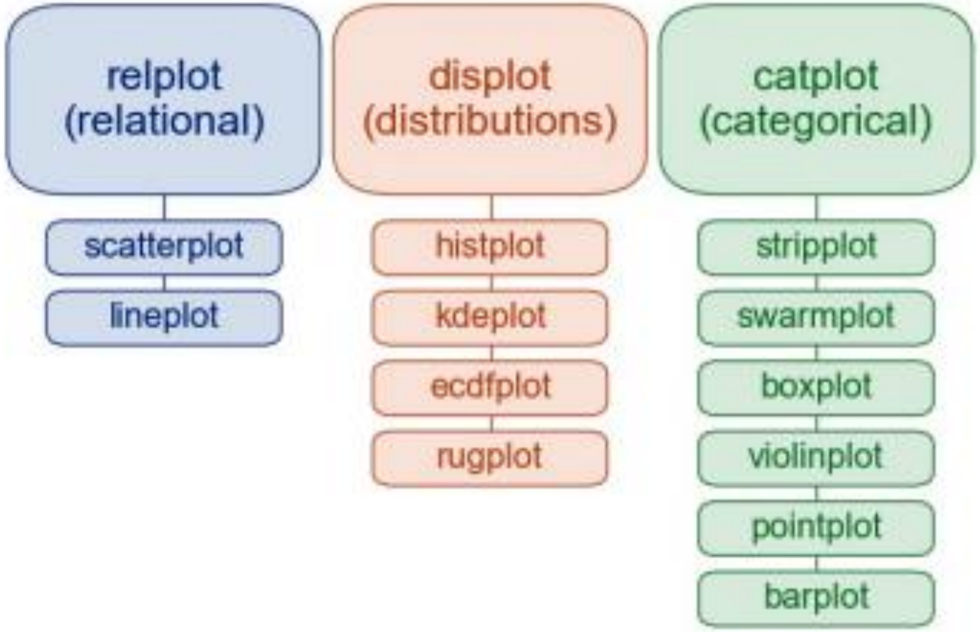
Working with Seaborn

- You always need:
  - data: indicates which dataframe you want to plot
  - x, y: what will be plotted on which axis
- Additional options:
  - hue: which columns to use for grouping of the data by color
  - col: which column to use to group the data into subplots
  - style: which columns to use to group the data into different styles

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 7). and the materials of the course CCS-1.

Types of visualizations

- Many more types of visualizations are possible
  - Univariate vs. bivariate plots
  - Relational vs. distributional vs. categorical



This slide is based on Van Atteveldt et al. (2022). [Computational Analysis of Communication] (chapter 7). and the materials of the [course CCS-1].

University of Amsterdam

Warm-up Exercise 5

University of Amsterdam

After the break:

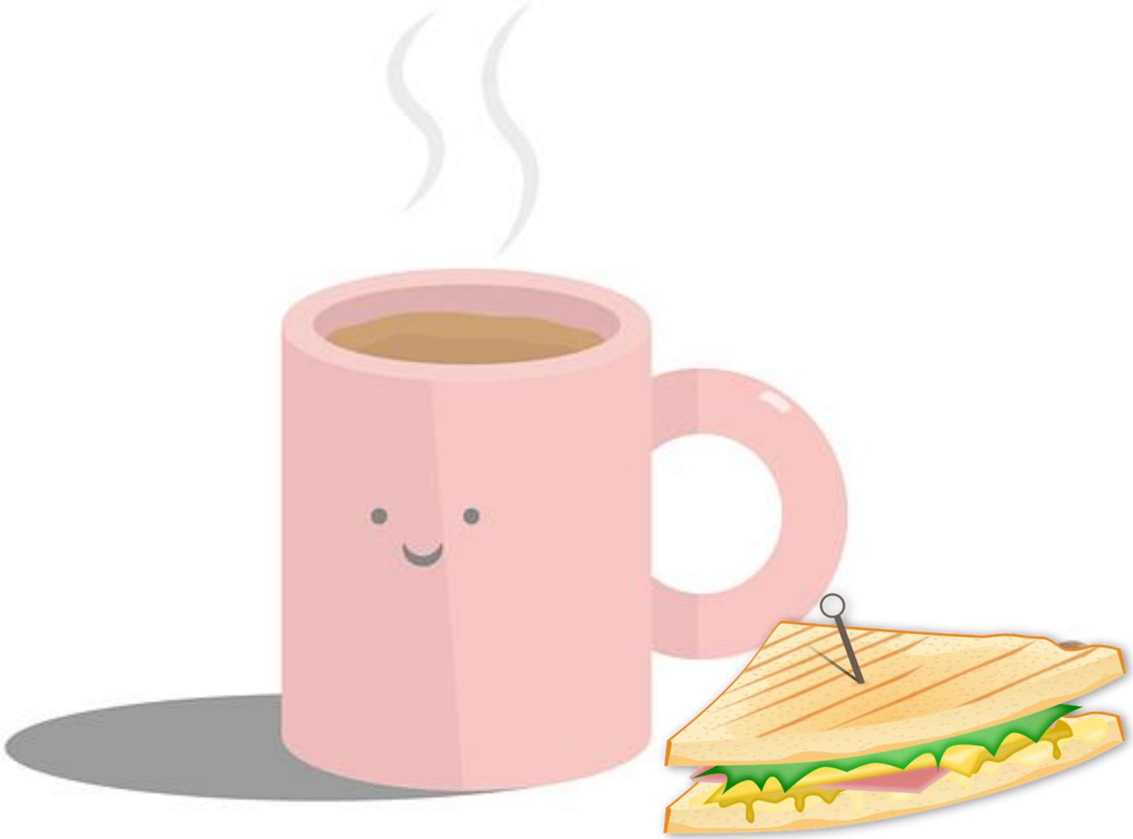
- ~~Checking in~~
- ~~Recap: Data aggregation~~
- ~~Recap: Data visualization~~
- ~~Warm-up exercise 5~~
- In-class exercise 1
- Recap: Where do data come from?
- Warm-up exercise 6
- In-class exercise 2
- Teaching exercise

UNIVERSITY OF AMSTERDAM

Peer coding:
In-class exercise I

After lunch:

- ~~Checking in~~
- ~~Recap: Data aggregation~~
- ~~Recap: Data visualization~~
- ~~Warm-up exercise 5~~
- ~~In-class exercise 1~~
- Recap: Where do data come from?
- Warm-up exercise 6
- In-class exercise 2
- Teaching exercise



Where do data come from?

Where do data come from?

- What we are used to: We create it (surveys, experiments)
- But increasingly often: From the web e.g., digital trace data
  - Data donations
  - Scraping: Powerful, but tricky
  - Application Programming Interfaces (APIs)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

APIs: Usage

- You send a *request* to some URL, you get back a JSON object and an HTTP response
- APIs *should* come with clear documentation

For example, Google Books:

https://developers.google.com/books/docs/v1/using

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

APIs: What can you do with them?

• GET data	• Retrieve all tweets from account X
• PUT (edit) data	• Edit the description of a YouTube video
• POST data	• Send a Telegram message to a million people
• DELETE data	• Remove my latest Instagram post

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

APIs: Why are they here?

- To facilitate interaction with platforms
- To facilitate research and data access
- To generate profit (selling data)
- To enable easy access to internal databases

- APIs differ in terms of their friendliness (catfacts vs. Google)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

APIs: Easy?

- No, often not.
- Documentation is often messy, unclear, and/or not updated
- Access is often restricted
- Authentication is a pain
- But luckily: Many *wrappers* exist
    - E.g., https://pypi.org/project/spotify/

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

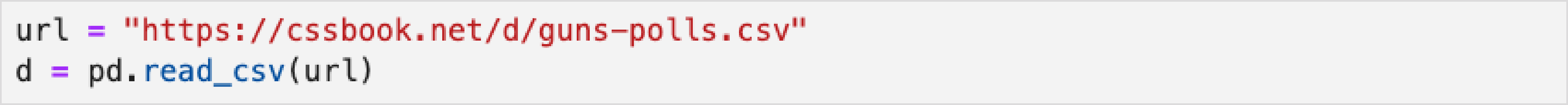
APIs and HTTPs

- APIs largely work through HTTP “endpoints”
- Important HTTP response codes:
  1. 200 – OK
  2. 201 – Created
  3. 400 – Bad request
  4. 401 – Unauthorized
  5. 403 – Forbidden
  6. 429 – Too many requests
  7. 500 – Internal server error

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

HTTP?

- In past code examples, you already took data directly from an URL – but you can only do this with datafiles that are available via a specific URL



```
url = "https://cssbook.net/d/guns-polls.csv"
d = pd.read_csv(url)
```

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 12). and the materials of the course CCS-1.

APIs: What do they give me?

- JSON file, often with some text as data
- Chapter 9.1: Some examples of how to handle text-data
- In CCS-2: Focus is mostly on text as data, and students learn to do advanced things with text (e.g., SML)

This slide is based on Van Atteveldt et al. (2022). Computational Analysis of Communication (chapter 9). and the materials of the course CCS-1.

UNIVERSITY OF AMSTERDAM

Warm-up Exercise 6

UNIVERSITY OF AMSTERDAM

Peer coding:
In-class exercise II

University of Amsterdam

After the break:

- ~~Checking in~~
- ~~Recap: Data aggregation~~
- ~~Recap: Data visualization~~
- ~~Warm-up exercise 5~~
- ~~In-class exercise 1~~
- ~~Recap: Where do data come from?~~
- ~~Warm-up exercise 6~~
- ~~In-class exercise 2~~
- Teaching exercise

University of Amsterdam

Teaching exercise

How to teach Python?

- **Live coding:** demonstrate concepts by writing code in real-time. This shows students how to approach problem-solving and debugging naturally.
- **Encourage pseudocode:** before diving into code, teach students to outline their logic in plain language. This helps to think logically without being overwhelmed by syntax.
- **Peer programming:** Let students work in pairs/small groups. This encourages collaboration, helps them to learn from each other and mimic real-world coding practices.
- **Code reviews:** Ask students to review each others' code to learn alternative approaches, catch mistakes, and teach the ability to critique code constructively.

Try teaching it yourself!

1. Make groups of 3
2. Pick a concept
3. Prepare:
   - Explanation
   - Small exercise
   - Max. 10 minutes
4. Present to your group + feedback

Concepts

1. What is data wrangling and what are its basics steps?
2. How can different dataframes be merged in Python?
3. What is the basic approach to data visualization and how to select the type of visualization that is best to use?