Data journalism
Week 4

Today

- Working with textual data
    - Top down vs bottom up
    - Enriching (textual) datasets
- Ingesting textual data

Beyond numbers

Working with textual data

Bottom-up vs top-down

- What are the texts about?
- What language is used?
- **We have no ‘hypothesis’, no a priori list of words we are interested in**

 Counting the most frequently used words

 [clustering, topic modeling]

- How often is party X mentioned?
- How many texts are about the economy?
- **We know what we are looking for, we have an a priori list of words we are interested in**

 Search terms, regular expressions

 [supervised machine learning]

A simple bottom-up approach

- We count the most frequently occuring words
  - (probably after cleaning up a bit: stopword and punctuation removal, lowercasing)
- Variations:
  - Word clouds
  - bigrams

A simple bottom-up approach

```
>>> from collections import Counter
>>> text = 'This is some text with more text and text. Yes it is.'
>>> text_clean = text.replace('.','').lower()
>>> Counter(text_clean.split()).most_common(5)
[('text', 3), ('is', 2), ('and', 1), ('this', 1), ('some', 1)]
```

A simple bottom-up approach

Discuss:

When do we do this? Think of *journalistic* use cases!

Advantages/disadvantages?

A simple top-down approach

- We make a list of words we are interested in and count how often (or whether) they occur
- Or we select only documents in which these words occur
- Variations
    - Regular expressions, e.g. “ABN.?Amro” (google for more info),

Another top-down approach: Sentiment analysis

- Essentially, we have lists of words with associated valence scores

- Using some math, we calculate an overall measure per text.

We can get it for an individual string:

```
from nltk.sentiment import vader

classifier = vader.SentimentIntensityAnalyzer()

classifier.polarity_scores('This is crappy shit')
{'neg': 0.783, 'neu': 0.217, 'pos': 0.0, 'compound': -0.802}
```

Or add columns to a dataset (and choose what you care about):

```
df['sentiment'] = df['text'].map(classifier.polarity_scores)

df['neg'] = df['sentiment'].map(lambda x: x.get('neg'))
```


Hutto, C. J. C. J., & Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media* (pp. 216–225). Retrieved from http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109%5Cnhttp://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

A simple top-down approach

Discuss:

When do we do this? Think of *journalistic* use cases!

Advantages/disadvantages?

Towards a complete workflow

Enriching data

But first: Our workflow

(1) retrieval --> (2) preprocessing --> (3) enrichment --> (4) analysis

- We talked about retrieval (file formats, APIs)
- We talked about (simple) analyses (descriptive statistics, [first] visualizations)
- We now learn how to preprocess and enrich our data

Recap: (1) Data retrieval

- APIs and files (CSV, JSON)
- Files:
    - direct via, e.g., pd.Dataframe.read_csv()
    - to extracts part from a deeply nested JSON-file: via json-module and for-loops

Recap: (4) Analysis

- .describe() for numerical data
- .value_counts() for frequency tables (especially nominal data)
- First .groupby() if you want to distinguish between groups
- Possibly .plot(), .hist() etc.

(2) Preprocessing

- Cleaning up, e.g. renaming columns
- Applying a function to a column (and create a new one):
  - df[‘numbers’] = df[‘comlumnwithnumbersasstrings’].map(int)
  - df[‘mytextlower’] = df[‘sometext’].map(lambda x: x.lower())
  - Stopword removal (‘the’, ‘and’, ‘a’, …)
  - Punctuation removal
  - … .

Let's try something…

- In a dataset of all speeches in the EU parliament, let's check out who talks most about terror and then read that speech.

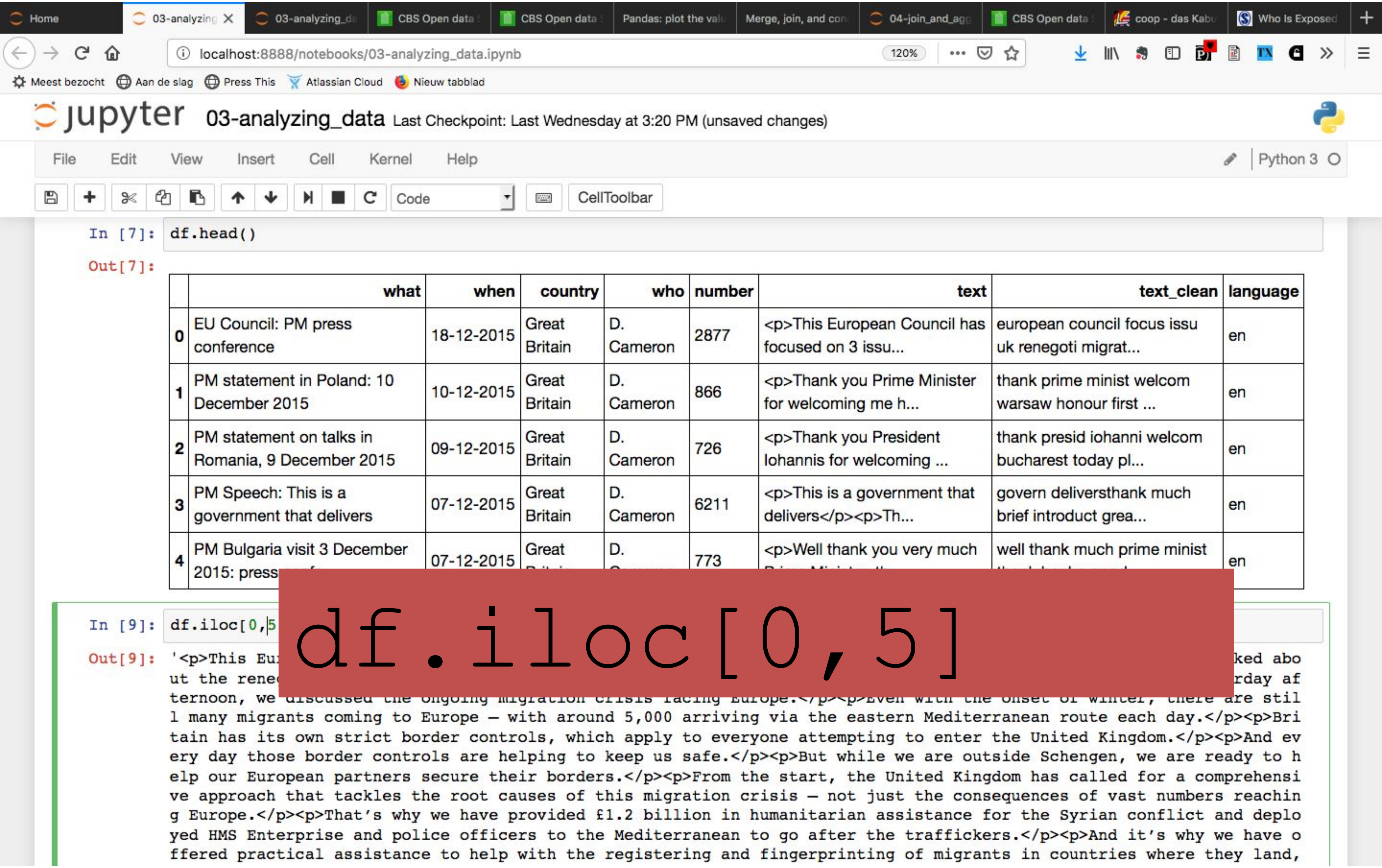
[This example will come back]

(3) Enrichment

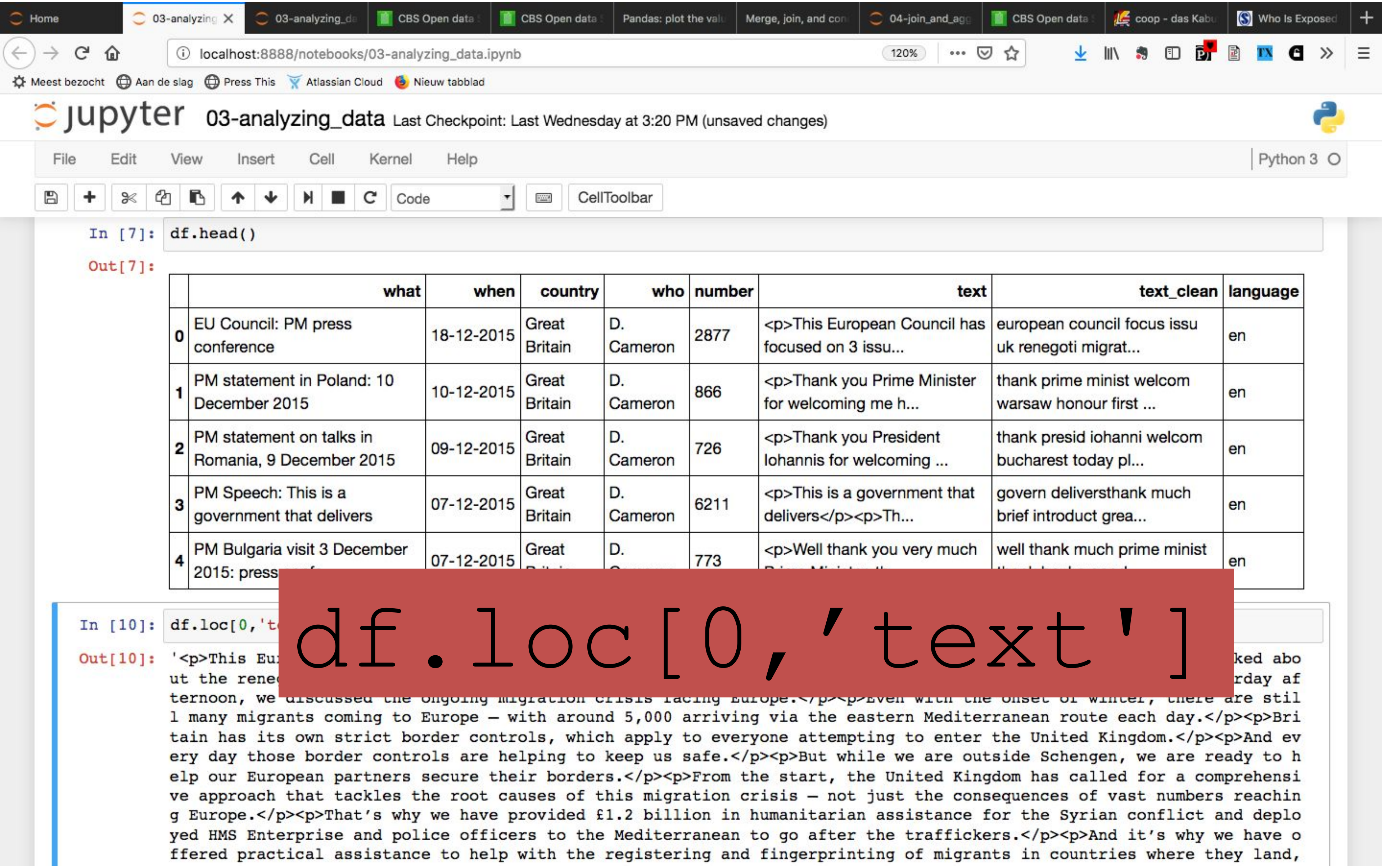
- Create new columns by, for example, counting mentions of a term/regular expression

  - df[‘terror’] = df[‘speech’].str.count(‘[tT]error’)

→ and here comes back our example from last week!


03-analyzing_data Last Checkpoint: Last Wednesday at 3:20 PM (unsaved changes)
localhost:8888/notebooks/03-analyzing_data.ipynb
File Edit View Insert Cell Kernel Help
Python 3
Code
CellToolbar
In [7]: df.head()
Out[7]:
what | when | country | who | number | text | text_clean | language
0 | EU Council: PM press conference | 18-12-2015 | Great Britain | D. Cameron | 2877 | <p>This European Council has focused on 3 issu... | european council focus issu uk renegoti migrat... | en
1 | PM statement in Poland: 10 December 2015 | 10-12-2015 | Great Britain | D. Cameron | 866 | <p>Thank you Prime Minister for welcoming me h... | thank prime minist welcom warsaw honour first ... | en
2 | PM statement on talks in Romania, 9 December 2015 | 09-12-2015 | Great Britain | D. Cameron | 726 | <p>Thank you President Iohannis for welcoming ... | thank presid iohanni welcom bucharest today pl... | en
3 | PM Speech: This is a government that delivers | 07-12-2015 | Great Britain | D. Cameron | 6211 | <p>This is a government that delivers</p><p>Th... | govern deliversthank much brief introduct grea... | en
4 | PM Bulgaria visit 3 December 2015: press | 07-12-2015 | Great | D. | 773 | <p>Well thank you very much | well thank much prime minist | en
In [9]: df.iloc[0,5
Out[9]:
l many migrants coming to Europe – with around 5,000 arriving via the eastern Mediterranean route each day.</p><p>Britain has its own strict border controls, which apply to everyone attempting to enter the United Kingdom.</p><p>And every day those border controls are helping to keep us safe.</p><p>But while we are outside Schengen, we are ready to help our European partners secure their borders.</p><p>From the start, the United Kingdom has called for a comprehensive approach that tackles the root causes of this migration crisis – not just the consequences of vast numbers reaching Europe.</p><p>That's why we have provided £1.2 billion in humanitarian assistance for the Syrian conflict and deployed HMS Enterprise and police officers to the Mediterranean to go after the traffickers.</p><p>And it's why we have offered practical assistance to help with the registering and fingerprinting of migrants in countries where they land,

df.iloc[0,5]



jupyter 03-analyzing_data Last Checkpoint: Last Wednesday at 3:20 PM (unsaved changes)

```
In [7]: df.head()
```

Out[7]:

	what	when	country	who	number	text	text_clean	language
0	EU Council: PM press conference	18-12-2015	Great Britain	D. Cameron	2877	<p>This European Council has focused on 3 issu...	european council focus issu uk renegoti migrat...	en
1	PM statement in Poland: 10 December 2015	10-12-2015	Great Britain	D. Cameron	866	<p>Thank you Prime Minister for welcoming me h...	thank prime minist welcom warsaw honour first ...	en
2	PM statement on talks in Romania, 9 December 2015	09-12-2015	Great Britain	D. Cameron	726	<p>Thank you President Iohannis for welcoming ...	thank presid iohanni welcom bucharest today pl...	en
3	PM Speech: This is a government that delivers	07-12-2015	Great Britain	D. Cameron	6211	<p>This is a government that delivers</p><p>Th...	govern deliversthank much brief introduct grea...	en
4	PM Bulgaria visit 3 December 2015: press	07-12-2015	Great	D.	773	<p>Well thank you very much	well thank much prime minist	en

```
In [10]: df.loc[0,'t
```

```
Out[10]: '<p>This Eu                                                  ked abo
ut the rene                                                          rday af
ternoon, we
l many migrants coming to Europe – with around 5,000 arriving via the eastern Mediterranean route each day.</p><p>Bri
tain has its own strict border controls, which apply to everyone attempting to enter the United Kingdom.</p><p>And ev
ery day those border controls are helping to keep us safe.</p><p>But while we are outside Schengen, we are ready to h
elp our European partners secure their borders.</p><p>From the start, the United Kingdom has called for a comprehensi
ve approach that tackles the root causes of this migration crisis – not just the consequences of vast numbers reachin
g Europe.</p><p>That's why we have provided £1.2 billion in humanitarian assistance for the Syrian conflict and deplo
yed HMS Enterprise and police officers to the Mediterranean to go after the traffickers.</p><p>And it's why we have o
ffered practical assistance to help with the registering and fingerprinting of migrants in countries where they land,
```

df.loc[0,'text']

More subsetting and slicing

- Advanced example: Get the whole row where the column 'terrorrefs' has the highest value in the whole dataset:
    - df.iloc[df['terrorrefs'].idxmax()]
- That works because df.iloc[] expects an integer to identify the row number, and df['terrorrefs'].idxmax() returns an integer (687 in our case)
- We could also do it in two steps:

df.iloc[df['terrorrefs'].idxmax()] is the same as:

```
df['terrorrefs'].idxmax()
```

```
687
```

```
df.iloc[687]
```

```
what          Permanent Link to Press conference in Islamabad
when                                               14-12-2008
country                                         Great Britain
who                                                  G. Brown
number                                                   2954
text         <p>Transcript of a press conference given by t...
text_clean   transcript press confer given prime minist mr ...
language                                                   en
terrorrefs                                                 44
Name: 687, dtype: object
```

From files to Pandas (or not)

Ingesting textual data

tabular file vs text files*

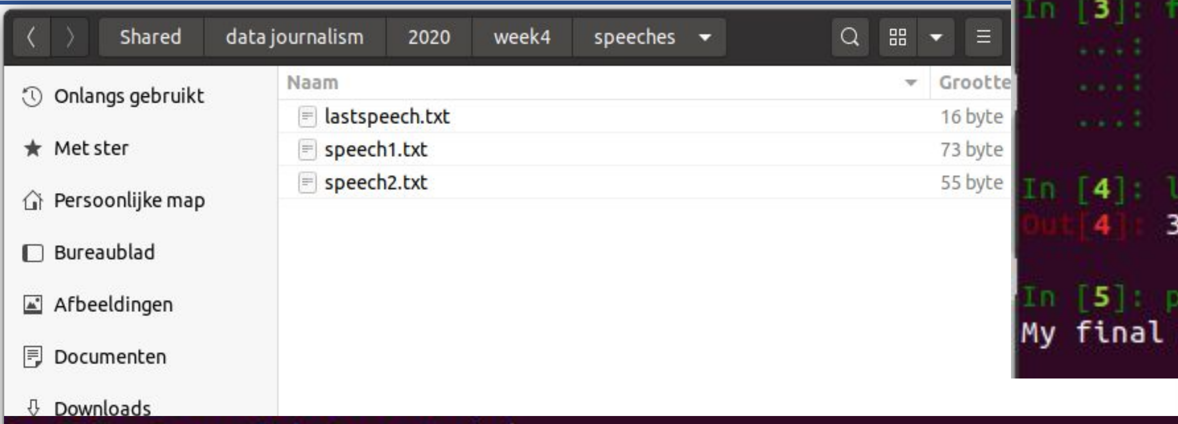
- You have a table where one column contains text (e.g., speeches), and the other columns metadata (e.g., who gave it, when, ...)
  - → pd.read_csv(), as always
  - → Pandas string functions *on that column*

    e.g., df['terror'] = df['speech'].str.count('[tT]error'

- You have one long file
  - → you need to *parse* it (e.g., using regular expressions)
  - → the less clearly structured, the harder (discuss individually)
- You have one file per text (speech, article,…)
  - → for loop to read files and add to a list or dict
  - → potentially transform to dataframe (partly matter of preference)

* we assume plain text files (often called *.txt). There are Python modules for extracting plain text from other file formats (such as PDF).

Looping over files




```
In [1]: from glob import glob

In [2]: myspeeches = []

In [3]: for filename in glob("speeches/*.txt"):
   ...:     with open(filename) as f:
   ...:         myspeeches.append(f.read())
   ...:

In [4]: len(myspeeches)
Out[4]: 3

In [5]: print(myspeeches[0])
My final words.
```

to a list

```
In [1]: from glob import glob

In [2]: import pandas as pd

In [3]: myspeeches = {}

In [4]: for filename in glob("speeches/*.txt"):
   ...:     with open(filename) as f:
   ...:         myspeeches[filename] = f.read()
   ...:

In [5]: df = pd.DataFrame.from_dict(myspeeches, orient='index')

In [6]: df
Out[6]:
                                                                    0
speeches/lastspeech.txt                                 My final words.\n
speeches/speech1.txt     This is some speech. A really interesting one....
speeches/speech2.txt     This is also a speech. the second one. a short...
```

to a dict and then a dataframe

Parsing a single file

This is difficult and for advanced use...

```
Transcript of debate XXX.

10.05 Chairman opens the meeting.

...
...

Smith: I really oppose what the chairman said.

Miller: Smith is wrong. He should be ashamed.
He also has no clue what he is talking about

Smith: How dare you!
```

Platte tekst ▾ Tabbreedte: 8 ▾

```
In [1]: import re

In [2]: import pandas as pd

In [3]: transcript = open('singletextfile.txt').read()

In [4]: parsed = re.findall(r".*:[ a-zA-Z\S]*",transcript, flags=re.M)
   ...:

In [5]: parsed
Out[5]:
['Smith: I really oppose what the chairman said.',
 'Miller: Smith is wrong. He should be ashamed.',
 'Smith: How dare you!']

In [6]: pd.DataFrame([x.split(":") for x in parsed])
Out[6]:
        0                                          1
0   Smith   I really oppose what the chairman said.
1  Miller     Smith is wrong. He should be ashamed.
2   Smith                              How dare you!
```

this is a regular expression that says:
“some characters followed by a colon followed by some characters including possible spaces or line breaks”

Here we say that we want to split on the colon to create two columns

Think about your data

- **1. How do my data look like?**
  - Ready-to-use table vs semi-structured vs unstructured
  - Plain text files versus Word, PDF, HTML
- **2. What do I want to do with it?**
  - Bottom-up: most likely you just need a (list of) string(s)
  - Top-Down: maybe pandas (but not always)

A range of possibilities

- Working with text can range from almost **trivial** (you have a dataframe already, you only need to apply .str.count(), .map() or similar and you are done) to very **complicated** (parsing badly structured text)

- Device a workflow and discuss the bottlenecks (with each other or us): e.g. "I can do A and C to achieve D, but don't know how to do B to get from A to C"

Summing up

Important modules, methods, functions

For reading files beyond using pandas

- glob to get a list of all files (with certain name pattern) in a folder
- `with open(filename) as f: f.read()` to read plain text files

For matching patterns in strings

- Google “regular expression" and/or read the recommended reading in the syllabus for this week!

- re, in particular re.findall()

For working with text in pandas

- .map() to map some input value onto some output value using a function
  - e.g., df[‘lengthofspeech’] = df[‘speech’].map(len)
  - Or, more complex, with a user defined function (to get length in words):
    df[‘lengthofspeech’] = df[‘speech’].map(lambda x: len(x.split()))
  - Or, by using a function supplied by a tool such as Vader
- .str.count() to count how often a pattern (in fact, a regular expression) occurs in a string

For bottom-up text analysis

- Counter , in particular Counter.most_common() for getting the most common elements.
  - e.g. Counter.most_common(mystring.split()) to get the most common words by splitting a string into a list of words

How further?

Have a look at notebook 06-analyzing_text.ipynb and today's slides to try out some text analysis techniques.

Thursday: OPEN LAB (= time to work on your own projects)

Good
luck!