Analysing data

- Currently 33 zettabytes
  - 2025: 173 zettabytes [IDC](https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf (https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf)
  - 173'000'000'000'000'000'000'000 (https://en.wikipedia.org/wiki/Zettabyte)

- Your future company will bathe in data about everything
- You need to learn how to use data and extract valuable information from it

Analysing data with models

> All models are wrong, but some are useful

Source: Several (https://en.wikipedia.org/wiki/All_models_are_wrong)

Today: Three relevant questions + geographical show-off

1. Are young people really getting poorer?
2. How green is Denmark really?
3. How many from the Ivory Coast lives in Denmark?

- Plotting with Python

Pandas

Pandas (https://pandas.pydata.org/) is a library for Python that helps you do this.

Install it now by typing:

```
$ pip install pandas
```

Pandas dataframes

Pandas works a bit like the `openpyxl` module: you get a 'sheet' of data and read it in columns.

Let's do that together now.

In [ ]:

```python
import pandas as pd
import matplotlib.pyplot as plt
```

But we need a data source!

https://www.dr.dk/nyheder/indland/aeldre-bliver-rigere-unge-fattigere (https://www.dr.dk/nyheder/indland/aeldre-bliver-rigere-unge-fattigere)

https://www.dst.dk/da/Statistik/emner/arbejde-indkomst-og-formue/indkomster (https://www.dst.dk/da/Statistik/emner/arbejde-indkomst-og-formue/indkomster)

https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=29483 (https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=29483)

In [ ]:

```python

```

In [ ]:

```python
df = pd.read_csv('data/20197211835442536700 48AINDK267296307071.csv', header=N
```

In [ ]:

```python
print(df)
```

In [ ]:

```python
print(df.head())
```

In [ ]:

```python
df.head()
```

In [ ]:

```python
df.columns = ['','', 'region', 'sex', 'age', '2018', '2017', '2016', '2015',
```

In [ ]:

```python
df.head()
```

In [ ]:

```python
df['age']
```

In [ ]:

```python
both = pd.DataFrame()
```

In [ ]:

```
both['age'] = df['age'][1:14]
```

In [ ]:

```
both
```

In [ ]:

```
# Crop columns to only contain 1 to 13
```

In [ ]:

```
for i in range(2012, 2019):
    name = str(i)
    both[name] = df[name][1:14]
```

In [ ]:

```
both
```

In [ ]:

```
both.plot()
```

In [ ]:

```
both.T[1:].plot()
```

Plotting with pandas

Pandas `DataFrame` s have a simple `.plot()` method, which plots **columns as x values** and **rows as y values.**

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html#pandas.DataFrame.plot.line (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html#pandas.DataFrame.plot.line)

You can also plot barcharts, histograms etc.

In [ ]:

```
both_index = pd.DataFrame()
```

In [ ]:

```
both_index['age'] = both['age']
```

In [ ]:

```
for i in range(2012, 2019):
    name = str(i)
    both_index[name] = df[name][1:14]
```

In [ ]:

```
for i in range(2012, 2019):
    name = str(i)
    both_index[name] = df[name][1:14] - df['2012'][1:14]
```

In [ ]:

```
both_index
```

In [ ]:

```
both_index.T[1:].plot()
```

Exercise!

Use the dataset available from GitHub: `session-10/data`

- Can you do the same thing for men and women exclusively?
    - Try to turn the problem into a function. The code will almost be the same, but what will change?
- Bonus question: Normalise to percentages
    - The numbers in the graph are pretty big, can you divide by the maximum number in all the columns?
        - This should give you values that are at maximum 1

How green is Denmark?!

http://databank.worldbank.org/data/reports.aspx?source=&series=EG.ELC.RNEW.ZS (http://databank.worldbank.org/data/reports.aspx?source=&series=EG.ELC.RNEW.ZS)

https://databank.worldbank.org/source/world-development-indicators (https://databank.worldbank.org/source/world-development-indicators)

In [ ]:

```
pd.read_csv('data/abcc94ae-70a2-4ca3-bfc8-5d6ddc6803d8_Data.csv')
```

In [ ]:

```
wb = pd.read_csv('data/abcc94ae-70a2-4ca3-bfc8-5d6ddc6803d8_Data.csv')
```

wb

Pandas `.loc()`

Slices the dataset using both row and column indices:

```
df.loc[rows, columns]
```

For example:

- `df.loc[:, :]`
- `df.loc[0, 1960:2018]`

Great resource: https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c (https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c)

In [ ]:

```
wb.loc[:, '1960 [YR1960]':'2018 [YR2018]']
```

In [ ]:

```
wb_data = wb.loc[:10, '1960 [YR1960]':'2018 [YR2018]'].T
```

In [ ]:

```
wb_data
```

In [ ]:

```
wb_data = wb.loc[:10, '1990 [YR1990]':'2015 [YR2015]'].T
```

In [ ]:

```
wb_data.columns = wb['Country Name'][:11]
```

In [ ]:

```
wb_data.plot()
```

In [ ]:

```
wb_data.plot(figsize=(20, 20))
```

Exercise

We only care about our northern bretheren of course. Can you filter our all the other countries? Perhaps using `.loc` ?

Use the dataset available from GitHub: `session-10/data`

```
import pandas as pd
import matplotlib.pyplot as plt

wb = pd.read_csv('abcc94ae-70a2-4ca3-bfc8-5d6ddc6803d8_Data.csv')
wb_data = wb.loc[:10, '1990 [YR1990]':'2015 [YR2015]'].T
wb_data.columns = wb['Country Name'][:11]
wb_data.plot()
plt.show()
```

How many from the Ivory Coast lives in Copenhagen?

In [ ]:

```
cph = pd.read_csv('data/befkbhalderstatkode_small.csv')
```

In [ ]:

```
KBH_NEIGHBORHOODS = {
    '1': 'Indre By',
    '2': 'Østerbro',
    '3': 'Nørrebro',
    '4': 'Vesterbro/Kgs. Enghave',
    '5': 'Valby',
    '6': 'Vanløse',
    '7': 'Brønshøj-Husum',
    '8': 'Bispebjerg',
    '9': 'Amager Øst',
    '10': 'Amager Vest',
    '99': 'Udenfor inddeling'
}
```

In [ ]:

```
COUNTRY_CODES = {
    '0': 'Uoplyst (1)',
    '5001': 'Uoplyst (2)',
    '5100': 'Danmark',
    '5101': 'Grønland',
    '5102': 'Udlandet uoplyst',
    '5103': 'Statsløs',
    '5104': 'Finland',
    '5105': 'Island, ligeret dansk',
    '5106': 'Island',
    '5107': 'Liechtenstein',
    '5108': 'Luxembourg',
    '5109': 'Monaco',
    '5110': 'Norge',
    '5114': 'Europa uoplyst',
    '5115': 'Kongelig',
    '5120': 'Sverige',
    '5122': 'Albanien',
    '5124': 'Andorra',
    '5126': 'Belgien',
    '5128': 'Bulgarien',
    '5129': 'Tjekkoslovakiet',
    '5130': 'Frankrig',
    '5134': 'Grækenland',
    '5140': 'Nederlandene',
    '5142': 'Irland',
    '5150': 'Italien',
    '5151': 'Serbien og Montenegro',
    '5152': 'Jugoslavien',
    '5153': 'Malta',
    '5154': 'Polen',
    '5156': 'Portugal',
    '5158': 'Rumænien',
    '5159': 'San Marino',
    '5160': 'Schweiz',
    '5162': 'Sovjetunionen',
    '5164': 'Spanien',
    '5170': 'Storbritannien',
    '5172': 'Tyrkiet',
    '5174': 'Ungarn',
    '5176': 'Vatikanstaten',
    '5180': 'Tyskland',
    '5182': 'Østrig',
    '5199': 'Europa uoplyst',
    '5202': 'Algeriet',
    '5204': 'Angola',
    '5207': 'Botswana',
    '5213': 'Burundi',
    '5214': 'Etiopien',
    '5215': 'Comorerne',
    '5216': 'Eritrea',
    '5222': 'Gambia',
    '5228': 'Ghana',
    '5230': 'Ækvatorialguinea',
    '5231': 'Guinea-Bissau',
    '5232': 'Guinea',
    '5233': 'Kap Verde',
    '5234': 'Kenya',
    '5235': 'Lesotho',
    '5236': 'Liberia',
    '5238': 'Libyen',
    '5240': 'Mozambique',
    '5242': 'Madagaskar',
    '5243': 'Mali',
    '5244': 'Marokko',
    '5245': 'Mauritius',
    '5246': 'Nigeria',
    '5247': 'Namibia',
        '5248': 'Marshalløerne',
        '5255': 'Sierra Leone',
        '5258': 'Sudan',
        '5259': 'Swaziland',
        '5260': 'Sydsudan',
        '5262': 'Sydafrika',
        '5266': 'Tanzania',
        '5268': 'Tunesien',
        '5269': 'Uganda',
        '5272': 'Egypten',
        '5273': 'Tuvalu',
        '5274': 'Kiribati',
        '5275': 'Vanuatu',
        '5276': 'Centralafrikanske Republik',
        '5277': 'Cameroun',
        '5278': 'Congo, Demokratiske Republik',
        '5279': 'Congo, Republikken',
        '5281': 'Benin',
        '5282': 'Elfenbenskysten',
        '5283': 'Gabon',
        '5284': 'Mauretanien',
        '5285': 'Niger',
        '5287': 'Rwanda',
        '5288': 'Senegal',
        '5289': 'Somalia',
        '5292': 'Tchad',
        '5293': 'Togo',
        '5294': 'Burkina Faso',
        '5295': 'Zimbabwe',
        '5296': 'Zambia',
        '5297': 'Malawi',
        '5298': 'Seychellerne',
        '5299': 'Afrika uoplyst',
        '5302': 'Argentina',
        '5303': 'Bahamas',
        '5304': 'Bolivia',
        '5305': 'Barbados',
        '5306': 'Brasilien',
        '5308': 'Guyana',
        '5309': 'Antigua og Barbuda',
        '5310': 'Nauru',
        '5311': 'Skt. Vincent og Grenadinerne',
        '5314': 'Canada',
        '5316': 'Chile',
        '5318': 'Colombia',
        '5319': 'Syd- og Mellemamerika uoplyst',
        '5322': 'Costa Rica',
        '5324': 'Cuba',
        '5326': 'Dominikanske Republik',
        '5328': 'Ecuador',
        '5338': 'Guatemala',
        '5339': 'Grenada',
        '5342': 'Haiti',
        '5344': 'Surinam',
        '5345': 'Dominica',
        '5347': 'Skt. Lucia',
        '5348': 'Honduras',
        '5352': 'Jamaica',
    '5354': 'Mexico',
    '5356': 'Nicaragua',
    '5358': 'Panama',
    '5364': 'Paraguay',
    '5366': 'Peru',
    '5372': 'El Salvador',
    '5374': 'Trinidad og Tobago',
    '5376': 'Uruguay',
    '5390': 'USA',
    '5392': 'Venezuela',
    '5395': 'Vestindiske Øer',
    '5397': 'Nordamerika uoplyst',
    '5398': 'Syd- og Mellemamerika uoplyst',
    '5402': 'Yemen',
    '5403': 'Forenede Arabiske Emirater',
    '5404': 'Afghanistan',
    '5406': 'Bahrain',
    '5408': 'Bhutan',
    '5410': 'Bangladesh',
    '5412': 'Brunei',
    '5414': 'Myanmar',
    '5416': 'Cambodja',
    '5418': 'Sri Lanka',
    '5422': 'Cypern',
    '5424': 'Taiwan',
    '5432': 'Indien',
    '5434': 'Indonesien',
    '5435': 'Østtimor',
    '5436': 'Irak',
    '5438': 'Iran',
    '5442': 'Israel',
    '5444': 'Japan',
    '5446': 'Jordan',
    '5448': 'Kina',
    '5452': 'Kuwait',
    '5454': 'Laos',
    '5456': 'Libanon',
    '5457': 'Maldiverne',
    '5458': 'Malaysia',
    '5459': 'Mongoliet',
    '5462': 'Oman',
    '5464': 'Nepal',
    '5466': 'Nordkorea',
    '5468': 'Vietnam (1)',
    '5471': 'Asien uoplyst',
    '5472': 'Pakistan',
    '5474': 'Filippinerne',
    '5478': 'Saudi-Arabien',
    '5482': 'Singapore',
    '5484': 'Sydkorea',
    '5486': 'Syrien',
    '5487': 'Mellemøsten uoplyst',
    '5488': 'Vietnam (2)',
    '5492': 'Thailand',
    '5496': 'Qatar',
    '5499': 'Asien uoplyst',
    '5502': 'Australien',
    '5505': 'Tonga',
    '5508': 'Fiji',
    '5514': 'New Zealand',
    '5522': 'Samoa',
    '5525': 'Djibouti',
    '5526': 'Belize',
    '5534': 'Papua Ny Guinea',
    '5599': 'Øer i Stillehavet',
    '5607': 'Estland',
    '5609': 'Letland',
    '5611': 'Litauen',
    '5621': 'Sao Tome og Principe',
    '5623': 'Salomonøerne',
    '5625': 'Skt. Kitts og Nevis',
    '5700': 'Rusland',
    '5704': 'Ukraine',
    '5706': 'Hviderusland',
    '5708': 'Armenien',
    '5710': 'Aserbajdsjan',
    '5712': 'Moldova',
    '5714': 'Usbekistan',
    '5716': 'Kasakhstan',
    '5718': 'Turkmenistan',
    '5720': 'Kirgisistan',
    '5722': 'Tadsjikistan',
    '5724': 'Georgien',
    '5750': 'Kroatien',
    '5752': 'Slovenien',
    '5754': 'Bosnien-Hercegovina',
    '5756': 'Makedonien',
    '5757': 'Serbien',
    '5758': 'Jugoslavien, Forbundsrepublikken',
    '5759': 'Montenegro',
    '5761': 'Kosovo',
    '5776': 'Tjekkiet',
    '5778': 'Slovakiet',
    '5779': 'Cookøerne',
    '5800': 'Land ukendt (2)',
    '5901': 'Færøerne uoplyst',
    '5902': 'Færøerne',
    '5999': 'Land ukendt (1)'
}
```

In [ ]:

```
cph.groupby('STATKODE').sum().loc[5100]
```

Exercise

How many from the Ivory Coast lives in Copenhagen?

1. Open the `data/befkbhalderstatkode_small.csv` dataset using `pandas`
2. Group by the `'STATKODE'` column
3. Sum all the values
4. Find the row corresponding to the Ivory Coast (`5282`)

Plotting maps with folium (showcase)

https://python-visualization.github.io/folium/ (https://python-visualization.github.io/folium/)

In [ ]:

```
cph_lat, cph_lon = 55.6867243, 12.5700724
```

In [ ]:

```
import requests

# Copenhagen map data: http://wfs-kbhkort.kk.dk/web/
url = 'http://wfs-kbhkort.kk.dk/k101/ows?service=WFS&version=1.0.0&request=Ge
geo_json = requests.get(url).json()
```

In [ ]:

```
import folium

# Create a map on a specific location (tuple)
map_osm = folium.Map(location=(cph_lat, cph_lon), zoom_start=10)
# Using geospatial data formatted in JSON, add the points from the dataset to
folium.GeoJson(geo_json, name='geojson').add_to(map_osm)
# Show the map
map_osm
```

Data sources

All ripe for the taking!

- World Bank (https://www.worldbank.org/)
- WTO (https://data.wto.org/)
- WHO (https://www.who.int/hiv/data/en/)
- Twitter (http://www.tweepy.org/)
- Kaggle (https://www.kaggle.com/datasets)
- Københavns datasæt (https://data.kk.dk/dataset)

- Your future company will bathe in data about everything
- You need to learn how to use data and extract valuable information from it

Analysing data with models

> All models are wrong, but some are useful

Source: [Several (https://en.wikipedia.org/wiki/All_models_are_wrong)](https://en.wikipedia.org/wiki/All_models_are_wrong)

Today: Three relevant questions + geographical show-off

1. Are young people really getting poorer?
2. How green is Denmark really?
3. How many from the Ivory Coast lives in Denmark?

- Plotting with Python

Pandas

[Pandas (https://pandas.pydata.org/)](https://pandas.pydata.org/) is a library for Python that helps you do this.

Install it now by typing:

```
$ pip install pandas
```

Pandas dataframes

Pandas works a bit like the `openpyxl` module: you get a 'sheet' of data and read it in columns.

Let's do that together now.

```
In [ ]:
```

But we need a data source!

[https://www.dr.dk/nyheder/indland/aeldre-bliver-rigere-unge-fattigere (https://www.dr.dk/nyheder/indland/aeldre-bliver-rigere-unge-fattigere)](https://www.dr.dk/nyheder/indland/aeldre-bliver-rigere-unge-fattigere) https://www.dst.dk/da/Statistik/emner/arbejde-indkomst-og-formue/indkomster (https://www.dst.dk/da/Statistik/emner/arbejde-indkomst-og-formue/indkomster)

https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=29483 (https://www.dst.dk/da/Statistik/nyt/NytHtml?cid=29483)

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

Plotting with pandas

Pandas `DataFrame` s have a simple `.plot()` method, which plots **columns as x values** and **rows as y values**.

[https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html#pandas.DataFrame.plot.line (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html#pandas.DataFrame.plot.line)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.line.html#pandas.DataFrame.plot.line)

You can also plot barcharts, histograms etc.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Exercise!

Use the dataset available from GitHub: `session-10/data`

- Can you do the same thing for men and women exclusively?
  - Try to turn the problem into a function. The code will almost be the same, but what will change?
- Bonus question: Normalise to percentages
  - The numbers in the graph are pretty big, can you divide by the maximum number in all the columns?
    - This should give you values that are at maximum 1

How green is Denmark?!

http://databank.worldbank.org/data/reports.aspx?source=&series=EG.ELC.RNEW.ZS (http://databank.worldbank.org/data/reports.aspx?source=&series=EG.ELC.RNEW.ZS)

https://databank.worldbank.org/source/world-development-indicators (https://databank.worldbank.org/source/world-development-indicators)

```
In [ ]:
```

```
In [ ]:
```

wb

Pandas `.loc()`

Slices the dataset using both row and column indices:

```
df.loc[rows, columns]
```

For example:

- `df.loc[:, :]`
- `df.loc[0, 1960:2018]`

Great resource: https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c (https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c)

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

In [ ]:

```

```

Exercise

We only care about our northern bretheren of course. Can you filter our all the other countries? Perhaps using `.loc` ?

Use the dataset available from GitHub: `session-10/data`

```
import pandas as pd
import matplotlib.pyplot as plt

wb = pd.read_csv('abcc94ae-70a2-4ca3-bfc8-5d6ddc6803d8_Data.csv')
wb_data = wb.loc[:10, '1990 [YR1990]':'2015 [YR2015]'].T
wb_data.columns = wb['Country Name'][:11]
wb_data.plot()
plt.show()
```

How many from the Ivory Coast lives in Copenhagen?

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

Exercise

How many from the Ivory Coast lives in Copenhagen?

1. Open the `data/befkbhalderstatkode_small.csv` dataset using `pandas`
2. Group by the `'STATKODE'` column
3. Sum all the values
4. Find the row corresponding to the Ivory Coast (`5282`)

Plotting maps with folium (showcase)

[https://python-visualization.github.io/folium/ (https://python-visualization.github.io/folium/)](https://python-visualization.github.io/folium/)

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Data sources

All ripe for the taking!

- World Bank (https://www.worldbank.org/)
- WTO (https://data.wto.org/)
- WHO (https://www.who.int/hiv/data/en/)
- Twitter (http://www.tweepy.org/)
- Kaggle (https://www.kaggle.com/datasets)
- Københavns datasæt (https://data.kk.dk/dataset)