Python for biologists

Tutorial 5 - Reading, writing and transforming data with pandas

by Tobias Andermann

After covering the numpy library, we will now proceed to working with real data and get to know the pandas package a bit better, which we already briefly used in one of the previous tutorials. Pandas is a very handy library that allows working with dataframes of any kind. You can load a range of different types of input data, form csv files, over actual excel files (which is pretty handy), to even loading information from the computer clipboard (e.g. if you copied a large table from e.g. a webpage, you don't even need to paste it anywhere or save to file, but pandas can directly read it).

Installation

If you haven't installed pandas yet, you can do it like this. We will also need matplotlib in this tutorial:

```
pip install pandas
```

In [3]:
```
import pandas as pd
```

```
You can read a table into python using pandas as follows (adjust the path to
where you downloaded the data folder from this github repo (you will have to re-
download the GitHub folder to todays files, since I just added the new file
after yesterdays tutorial). Here we are reading a tab-delimited text file, using
the `read_csv()` function.:
```

In [4]:
```
data = pd.read_csv('../data/co2_temp.txt',sep='\t')
```

You can view the first lines of the loaded data by using the `.head()` function (instead of printing the complete data on screen, which can be disorienting sometimes):

In [5]:
```
data.head()
```

Out[5]:

	year	co2	temperature
0	1880	289.469999	13.81
1	1881	289.736999	13.89
2	1882	290.018999	13.89
3	1883	290.262999	13.80
4	1884	290.511999	13.71

Another fun way of reading data into python via pandas is to directly load data from the clipboard (i.e. whatever you last copied). For example you can load the data from the `petal_sepal_length.txt` file we used yesterday directly from GitHub by copying the file content in your web-browser and then loading the clipboard here.

- Open the file with this link (https://raw.githubusercontent.com/tobiashofmann88/python_for_biologists/master/data/petal_sepal_length.txt)
- Copy the complete file content (but don't paste it anywhere)
- Now run the following command (without copying anything else in between, including the command! Pandas will read whatever you last copied)

In [8]:
```
pd.read_clipboard()
```

Out[8]:

	sample_name	petal_length	sepal_length
0	sample1	2.037214	5.172704
1	sample2	4.915837	6.149615
2	sample3	1.614314	5.939969
3	sample4	1.525709	6.603432
4	sample5	3.729975	6.065105
5	sample6	0.778760	5.173370
6	sample7	5.291489	7.381162
7	sample8	5.486099	6.529424
8	sample9	1.588885	5.447183
9	sample10	3.506057	5.821899
10	sample11	1.034731	5.719114
11	sample12	3.918415	5.324338
12	sample13	4.704223	6.241321
13	sample14	1.274312	5.460195
14	sample15	4.027879	6.018649
15	sample16	1.052670	6.149158
16	sample17	5.124328	5.799524
17	sample18	0.727854	5.111055
18	sample19	1.889160	5.093237
19	sample20	5.857770	4.775423
20	sample21	0.982377	4.570141
21	sample22	1.642252	5.496335
22	sample23	1.813812	4.620661
23	sample24	1.675574	6.098174
24	sample25	1.402000	4.044937
25	sample26	5.010073	6.460642
26	sample27	1.547179	3.810783
27	sample28	5.583385	6.768361
28	sample29	5.602358	5.734823
29	sample30	0.999512	5.644809
30	sample31	6.357546	6.109699
31	sample32	1.598450	5.633242
32	sample33	6.383405	5.473322
33	sample34	3.502309	6.466854
34	sample35	1.489134	5.992310
35	sample36	1.414486	4.211974
36	sample37	4.695659	5.889835
37	sample38	4.314651	6.078578
38	sample39	4.332901	6.525604
39	sample40	5.355377	6.257550
40	sample41	4.215726	5.083585

Pandas can even read directly from an excel file (which is super useful, as often biological data is stored in horribly large excel files). You can also deal with quite messy excel files as in the example below. View the file in excel to get an idea of what it looks like. In the command below we read the data by specifying the name of the excel sheet (`sheet_name='ESTIMATES'` ,the file contains multiple sheets) and by skipping the first 16 lines of unnecessary information (`header=16`):

In [6]:
```
data_from_excel = pd.read_excel('../data/world_pop_1950_2015.xlsx',
                                header=16,
                                sheet_name='ESTIMATES')
data_from_excel.head()
```

Out[6]:

	Index	Variant	Region, subregion, country or area *	Notes	Country code	1950	1951	1952	1953	
0	1	Estimates	WORLD	NaN	900	2536274.721	2583816.786	2630584.384	2677230.358	272430
1	2	Estimates	More developed regions	a	901	814865.069	824212.665	834074.389	844263.515	85463
2	3	Estimates	Less developed regions	b	902	1721409.652	1759604.121	1796509.995	1832966.843	186967
3	4	Estimates	Least developed countries	c	941	195259.056	199052.136	202904.860	206885.020	21104
4	5	Estimates	Less developed regions, excluding least develo...	d	934	1526150.596	1560551.985	1593605.135	1626081.823	165862

5 rows × 71 columns

You can select specific subsets of the data by using row and column indeces or by using the actual column names. Let's for example extract the column with the names of all regions:

In [27]:
```
data_from_excel['Region, subregion, country or area *']
```

Out[27]:
```
0                                                  WORLD
1                                More developed regions
2                                Less developed regions
3                              Least developed countries
4      Less developed regions, excluding least develo...
5                  Less developed regions, excluding China
6                                  High-income countries
7                               Middle-income countries
8                         Upper-middle-income countries
9                         Lower-middle-income countries
10                                 Low-income countries
11                                   Sub-Saharan Africa
12                                               AFRICA
13                                       Eastern Africa
14                                              Burundi
15                                              Comoros
16                                             Djibouti
17                                              Eritrea
18                                             Ethiopia
19                                                Kenya
20                                           Madagascar
21                                               Malawi
22                                            Mauritius
23                                              Mayotte
24                                           Mozambique
25                                              Réunion
26                                               Rwanda
27                                           Seychelles
28                                              Somalia
29                                          South Sudan
                              ...
243                           Saint Pierre and Miquelon
244                            United States of America
245                                              OCEANIA
246                                 Australia/New Zealand
247                                            Australia
248                                          New Zealand
249                                            Melanesia
250                                                 Fiji
251                                        New Caledonia
252                                      Papua New Guinea
253                                       Solomon Islands
254                                              Vanuatu
255                                           Micronesia
256                                                 Guam
257                                             Kiribati
258                                      Marshall Islands
259                           Micronesia (Fed. States of)
260                                                Nauru
261                              Northern Mariana Islands
262                                                Palau
263                                           Polynesia
264                                      American Samoa
265                                         Cook Islands
266                                      French Polynesia
267                                                 Niue
268                                                Samoa
269                                              Tokelau
270                                                Tonga
271                                               Tuvalu
272                           Wallis and Futuna Islands
Name: Region, subregion, country or area *, Length: 273, dtype: object
```

The selected column in printed as a Pandas Series object. You can easily transform the column values into an array by adding `.values`. This makes it sometimes easier to work with the data, particularly if it's numerical data:

In [28]:
```
data_from_excel['Region, subregion, country or area *'].values
```

Out[28]:
```
array(['WORLD', 'More developed regions', 'Less developed regions',
       'Least developed countries',
       'Less developed regions, excluding least developed countries',
       'Less developed regions, excluding China', 'High-income countries',
       'Middle-income countries', 'Upper-middle-income countries',
       'Lower-middle-income countries', 'Low-income countries',
       'Sub-Saharan Africa', 'AFRICA', 'Eastern Africa', 'Burundi',
       'Comoros', 'Djibouti', 'Eritrea', 'Ethiopia', 'Kenya',
       'Madagascar', 'Malawi', 'Mauritius', 'Mayotte', 'Mozambique',
       'Réunion', 'Rwanda', 'Seychelles', 'Somalia', 'South Sudan',
       'Uganda', 'United Republic of Tanzania', 'Zambia', 'Zimbabwe',
       'Middle Africa', 'Angola', 'Cameroon', 'Central African Republic',
       'Chad', 'Congo', 'Democratic Republic of the Congo',
       'Equatorial Guinea', 'Gabon', 'Sao Tome and Principe',
       'Northern Africa', 'Algeria', 'Egypt', 'Libya', 'Morocco', 'Sudan',
       'Tunisia', 'Western Sahara', 'Southern Africa', 'Botswana',
       'Lesotho', 'Namibia', 'South Africa', 'Swaziland',
       'Western Africa', 'Benin', 'Burkina Faso', 'Cabo Verde',
       "Côte d'Ivoire", 'Gambia', 'Ghana', 'Guinea', 'Guinea-Bissau',
       'Liberia', 'Mali', 'Mauritania', 'Niger', 'Nigeria',
       'Saint Helena', 'Senegal', 'Sierra Leone', 'Togo', 'ASIA',
       'Eastern Asia', 'China', 'China, Hong Kong SAR',
       'China, Macao SAR', 'China, Taiwan Province of China',
       "Dem. People's Republic of Korea", 'Japan', 'Mongolia',
       'Republic of Korea', 'South-Central Asia', 'Central Asia',
       'Kazakhstan', 'Kyrgyzstan', 'Tajikistan', 'Turkmenistan',
       'Uzbekistan', 'Southern Asia', 'Afghanistan', 'Bangladesh',
       'Bhutan', 'India', 'Iran (Islamic Republic of)', 'Maldives',
       'Nepal', 'Pakistan', 'Sri Lanka', 'South-Eastern Asia',
       'Brunei Darussalam', 'Cambodia', 'Indonesia',
       "Lao People's Democratic Republic", 'Malaysia', 'Myanmar',
       'Philippines', 'Singapore', 'Thailand', 'Timor-Leste', 'Viet Nam',
       'Western Asia', 'Armenia', 'Azerbaijan', 'Bahrain', 'Cyprus',
       'Georgia', 'Iraq', 'Israel', 'Jordan', 'Kuwait', 'Lebanon', 'Oman',
       'Qatar', 'Saudi Arabia', 'State of Palestine',
       'Syrian Arab Republic', 'Turkey', 'United Arab Emirates', 'Yemen',
       'EUROPE', 'Eastern Europe', 'Belarus', 'Bulgaria', 'Czechia',
       'Hungary', 'Poland', 'Republic of Moldova', 'Romania',
       'Russian Federation', 'Slovakia', 'Ukraine', 'Northern Europe',
       'Channel Islands', 'Denmark', 'Estonia', 'Faeroe Islands',
       'Finland', 'Iceland', 'Ireland', 'Isle of Man', 'Latvia',
       'Lithuania', 'Norway', 'Sweden', 'United Kingdom',
       'Southern Europe', 'Albania', 'Andorra', 'Bosnia and Herzegovina',
       'Croatia', 'Gibraltar', 'Greece', 'Holy See', 'Italy', 'Malta',
       'Montenegro', 'Portugal', 'San Marino', 'Serbia', 'Slovenia',
       'Spain', 'TFYR Macedonia', 'Western Europe', 'Austria', 'Belgium',
       'France', 'Germany', 'Liechtenstein', 'Luxembourg', 'Monaco',
       'Netherlands', 'Switzerland', 'LATIN AMERICA AND THE CARIBBEAN',
       'Caribbean', 'Anguilla', 'Antigua and Barbuda', 'Aruba', 'Bahamas',
       'Barbados', 'British Virgin Islands', 'Caribbean Netherlands',
       'Cayman Islands', 'Cuba', 'Curaçao', 'Dominica',
       'Dominican Republic', 'Grenada', 'Guadeloupe', 'Haiti', 'Jamaica',
       'Martinique', 'Montserrat', 'Puerto Rico', 'Saint Kitts and Nevis',
       'Saint Lucia', 'Saint Vincent and the Grenadines',
       'Sint Maarten (Dutch part)', 'Trinidad and Tobago',
       'Turks and Caicos Islands', 'United States Virgin Islands',
       'Central America', 'Belize', 'Costa Rica', 'El Salvador',
       'Guatemala', 'Honduras', 'Mexico', 'Nicaragua', 'Panama',
       'South America', 'Argentina', 'Bolivia (Plurinational State of)',
       'Brazil', 'Chile', 'Colombia', 'Ecuador',
       'Falkland Islands (Malvinas)', 'French Guiana', 'Guyana',
       'Paraguay', 'Peru', 'Suriname', 'Uruguay',
       'Venezuela (Bolivarian Republic of)', 'NORTHERN AMERICA',
       'Bermuda', 'Canada', 'Greenland', 'Saint Pierre and Miquelon',
       'United States of America', 'OCEANIA', 'Australia/New Zealand',
       'Australia', 'New Zealand', 'Melanesia', 'Fiji', 'New Caledonia',
       'Papua New Guinea', 'Solomon Islands', 'Vanuatu', 'Micronesia',
       'Guam', 'Kiribati', 'Marshall Islands',
```

Let's say we want to specifically extract data for a given country or region from the dataframe, for example `AFRICA`. We can do that by telling pandas to select everything where the `'Region, subregion, country or area *'` column has the value `AFRICA`:

In [41]:
```
africa_data = data_from_excel[data_from_excel['Region, subregion, country or area
africa_data
```

Out[41]:

	Index	Variant	Region, subregion, country or area *	Notes	Country code	1950	1951	1952	1953	1954
12	13	Estimates	AFRICA	NaN	903	228670.019	233277.049	238113.121	243177.791	248471.497

1 rows × 71 columns

We can accomplish the same by finding the index of the `AFRICA` data row by hand and selecting it that way, using `.iloc[]`. The only difference is that the resulting object in this case is a pandas series, whereas above it is a pandas dataframe (does not make a real difference for most operations).

In [42]:
```
africa_data = data_from_excel.iloc[12,]
```

If you want to only select the actual population data for Africa, you can do that by providing the column indeces you want ot extract, using the same logic as we did in the numpy tutorial in the array slicing section:

In [47]:
```
africa_data = data_from_excel.iloc[12,5:]
africa_data.head()
```

Out[47]:
```
1950    228670
1951    233277
1952    238113
1953    243178
1954    248471
Name: 12, dtype: object
```

Similarly you can also index the target columns using the other approach of data selection from above:

In [52]:
```
data_from_excel[data_from_excel['Region, subregion, country or area *']=='AFRICA'
```

Out[52]:

	1950	1951	1952	1953	1954	1955	1956	1957	1
12	228670.019	233277.049	238113.121	243177.791	248471.497	253995.025	259750.195	265739.281	271965.

1 rows × 66 columns

To have best control over the data, let's store the values into numpy arrays, using the `.values` function. Since we are extracting the values of a dataframe, it will return one array row per dataframe row. In this case we only have selected one single row to start with, so in order to just store it as a regular 1D array, we select the first row (the only one) by using the index `[0]`:

In [69]:
```
africa_pop_values = data_from_excel[data_from_excel['Region, subregion, country o
```

To remember which value corresponds to which year, let's also store the column names in an array. You can get the column names using the `.columns` function. Then I store the column names in an array, using the `.values` command and transform them into integers using the `.astype()` function:

In [70]:
```
years_array = data_from_excel.iloc[:,5:].columns.values.astype(int)
years_array
```

Out[70]:
```
array([1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960,
       1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971,
       1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982,
       1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
       1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004,
       2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015])
```

Plotting the data

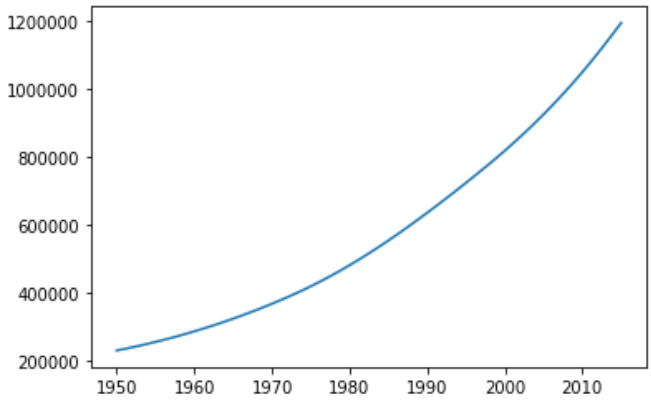
We can plot the data using matplotlib.

In [46]:
```
import matplotlib.pyplot as plt
```

Now we can plot the data we extracted. Let's plot years on the x-axis and the human population on the y-axis:

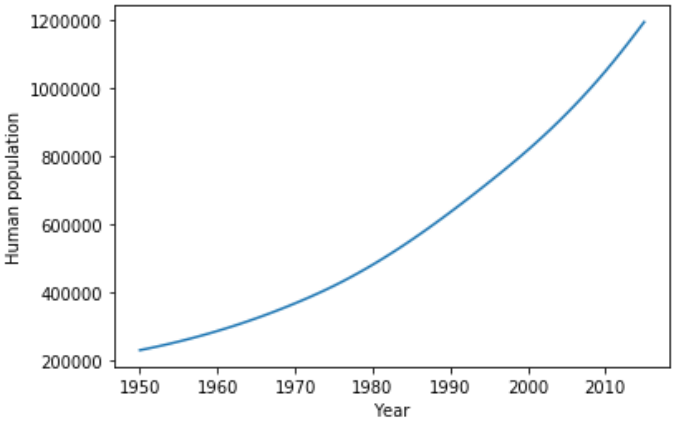
In [73]:
```
plt.plot(years_array,africa_pop_values)
```

Out[73]:
```
[<matplotlib.lines.Line2D at 0x11eabf6a0>]
```



We can improve the looks of the plot a bit by labeling the axes, using the `plt.xlabel()` and `plt.ylabel()` commands:

In [78]:
```
plt.plot(years_array,africa_pop_values);
plt.xlabel('Year');
plt.ylabel('Human population');
```

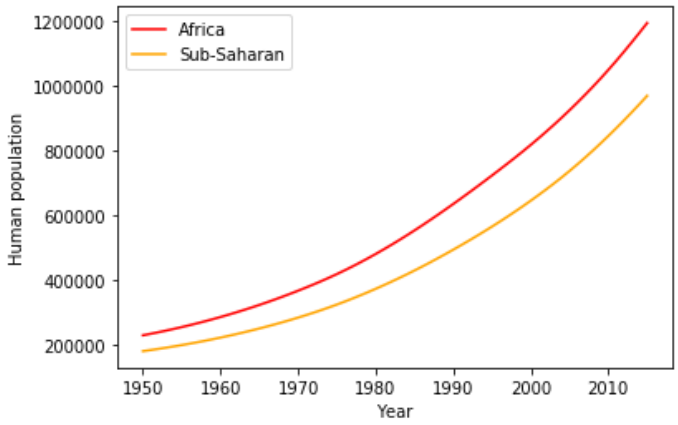


Let's also define the color of the plotted line, as well as adding a legend. Color can eb defined using the `color=` argument in the plotting command. By giving the plot a label (`label=`) we can plot a legend witht he `plt.legend()` command that automatically pulls the colors and titles of all data in the plot (especially useful when plotting multiple lines):

In [84]:
```
sub_saharan_africa_pop_values = data_from_excel[data_from_excel['Region, subregio

plt.plot(years_array,africa_pop_values,color='red', label='Africa');
plt.plot(years_array,sub_saharan_africa_pop_values,color='orange', label='Sub-Saha
plt.xlabel('Year');
plt.ylabel('Human population');
plt.legend()
```

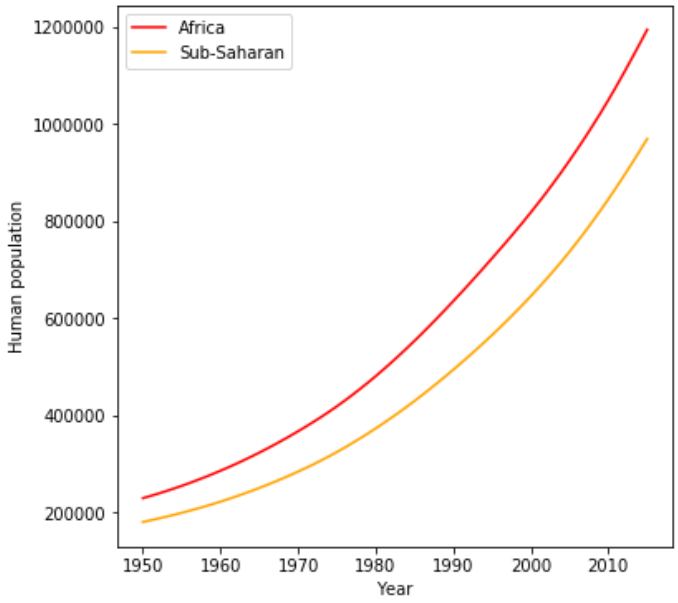
Out[84]:
```
<matplotlib.legend.Legend at 0x11f26f898>
```



You can find more plotting options to tweek your plots here: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html (https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html)

If you want to store your plot to a pdf file you first have to define a figure instance (`fig = plt.figure()`) and then save it (`fig.savefig()`) after the plotting commands are executed. You can also adjust the size of the figure by providing the `figsize =` setting in the `plt.figure()` command.

In [87]:
```
fig = plt.figure(figsize = (6,6))
plt.plot(years_array,africa_pop_values,color='red', label='Africa');
plt.plot(years_array,sub_saharan_africa_pop_values,color='orange', label='Sub-Saha
plt.xlabel('Year');
plt.ylabel('Human population');
plt.legend()
fig.savefig('/Users/tobias/GitHub/workshops/advanced_python/output_files/human_po
```



TASK: Plot the human population trends for all continents (or any set of countries/regions you prefer), using the pandas data-selection tools you just learned. You can plot multiple lines into the same plot by just executing multiple plot commands at the same time. You may need to log-transform the population size values in order to properly visualize the trends for all continents in comparison, mainly because Asia has such a large population.

Explore the multitude of pandas options on the official pandas page (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html).

Let's move on to tutorial 5 (https://github.com/tobiashofmann88/python_for_biologists/blob/master/tutorials/tutorial_5.ipynb)