Data Analysis – Advanced Statistics with Python

Dr. Julia Jerke

jerke@soziologie.uzh.ch

Thursday, 12.15pm – 13.45pm, AND 2.46

Session 2 – Getting and exploring data with *Pandas*

Agenda

1. The basics of programming with Python
2. How to *Pandas*
3. Hands on
4. (Optional: Jupyter Notebook)

1. The basics of programming with Python

Programming essentials

- Learning Python programming basics is not mandatory but might help you throughout the course
- The most important parts:
  - Strings
  - Lists
  - Dictionaries
  - Control flow (e.g., `if`, `else`, `elif`)
  - Loops (e.g., `for`, `while`)
  - Functions
  - (Classes)

Where/how to learn Python

- Online platforms, e.g.:
  - codecademy
  - coursera
  - udemy
  - DataCamp
- Video tutorials, e.g.:
  - freeCodeCamp.org: https://www.youtube.com/watch?v=rfscVS0vtbw
  - Bro Code: https://www.youtube.com/watch?v=XKHEtdqhLK8
- Books, e.g.:
  - *Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming* (author: Eric Matthes)
  - *Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners* (author: Al Sweigart)
  - *Learning Python, 5th Edition* (author: Mark Lutz)
- You will find several cheat sheets on OLAT covering those programming essentials in a condensed manner

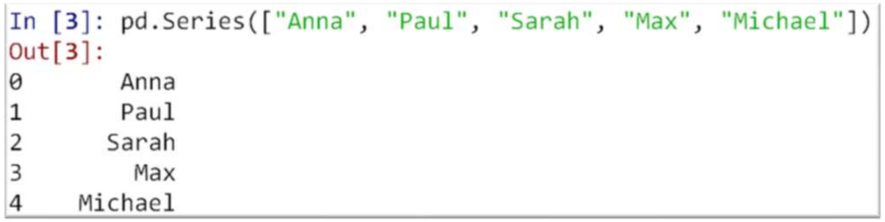
2. How to «*Pandas*»

Pandas overview

- One of the most important and basic library for data analysis in Python
- It is used for importing and cleaning data
- Key funcionalities:
  - Importing/creating and saving data sets
  - Data cleaning
  - Data management and manipulation
  - Handling missing data
  - Grouping data
  - Describing data
  - Summary statistics
  - Merging of data sets

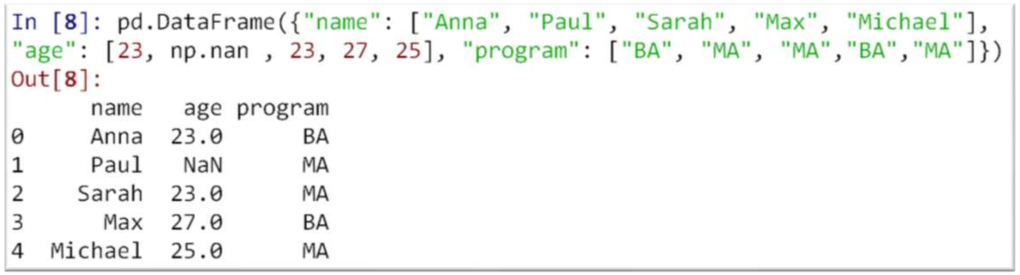
Data objects in *Pandas*

1) Series
   - One-dimensional data structure
   - can store data of different types (including characters, integers, floating point values, categorical data and more)
   - Rows are labelled



```
In [3]: pd.Series(["Anna", "Paul", "Sarah", "Max", "Michael"])
Out[3]:
0       Anna
1       Paul
2      Sarah
3        Max
4    Michael
```

2) Dataframe
   - Two-dimensional data structure
   - Basically a sequence of series that are organized in columns
   - similar to a spreadsheet, a SQL table or the data.frame in R
   - Rows and columns are labelled



```
In [8]: pd.DataFrame({"name": ["Anna", "Paul", "Sarah", "Max", "Michael"],
"age": [23, np.nan , 23, 27, 25], "program": ["BA", "MA", "MA","BA","MA"]})
Out[8]:
      name   age program
0     Anna  23.0      BA
1     Paul   NaN      MA
2    Sarah  23.0      MA
3      Max  27.0      BA
4  Michael  25.0      MA
```

Pandas documention

- Documentation and tutorials:
  https://pandas.pydata.org/

- *10 minutes to pandas* quick guide:
  https://pandas.pydata.org/docs/user_guide/10min.html

- Brief tutorials:
  https://pandas.pydata.org/docs/getting_started/intro_tutorials/index.html

- Comparison with other statistical languages:
  https://pandas.pydata.org/docs/getting_started/comparison/index.html

Importing libraries

- Make sure that the library you want to import is already installed
- Use the `import` command:

  `import` *`library_name`*

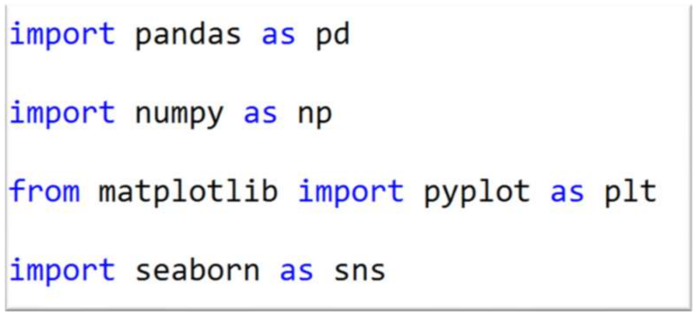
- If you are only using some parts of the library use the `from` prefix:

  `from` *`library_name`* `import` *`part_name`*

- You can (and should!) define abbreviations for the libraries you are importing using the `as` suffix:

  `import` *`library_name`* `as` *`abbreviation`*

- Stick to the community conventions for defining abbreviations, this will increase the readability of your code



```
import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns
```

How to use functions from libraries

Using functions

- Each library has its own specific functions and methods that you can use if you import the library
- You can call these functions by combing the library name as a prefix with the function name as a suffix:

  `library_name.function_name()`
- It now becomes clear why you should abbreviate the library names:

  Instead of typing `pandas.Series()` you can type `pd.Series()` to create a data frame

Return value of functions

- Most functions return a result, that can be saved into a variable, e.g.:

  `df = pd.Series([23, np.nan , 23, 27, 25])` or `df_nomiss = df.dropna(“any”)`
- Some functions don’t have a return value but directly execute an action: `df.describe()`
- Many *Pandas* functions have an inplace parameter:
  - `inplace=False` (default): the object is not changed directly; the result has to be saved to a variable
  - `inplace=True`: the object is changed directly

Method chaining

- Chaining different methods in a row as a sequence: the result from one method is used in the next method
- Improves the readabilty of your code

```
covid_survey.dropna(how="any")
```

Drops all cases that have at least one missing value

➕

```
covid_survey.info()
```

Provides basic information about the data frame such as number of columns and rows

⬇

```
covid_survey.dropna(how="any“).info()
```

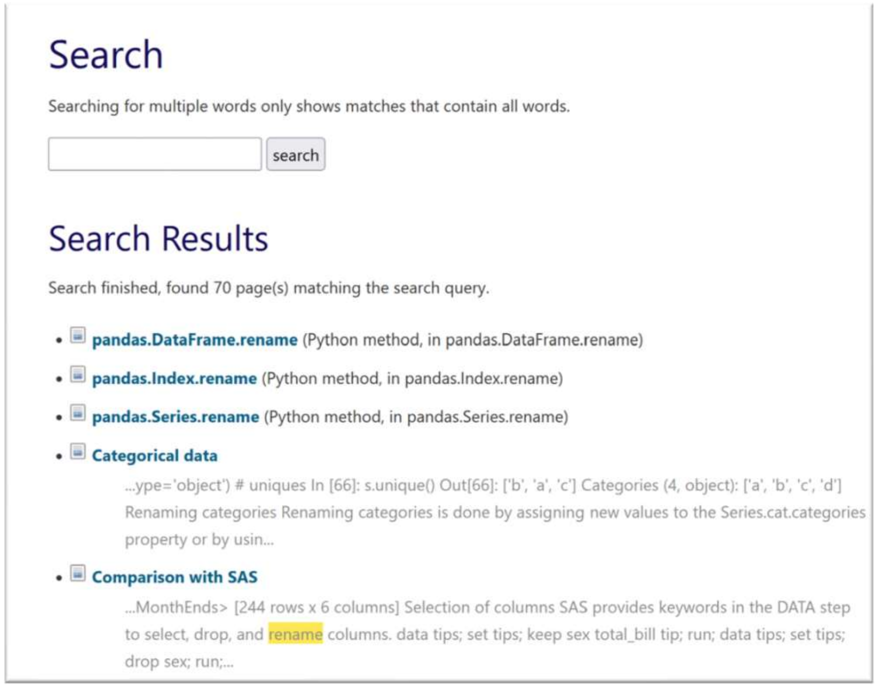
Drops all cases that have at least one missing value and directly displays the data frame informations

Getting help

- You can’t possibly know all functions and their parameters
- But a good programmer/statistician knows where and how to find help!

Pandas documentation:

https://pandas.pydata.org/docs/search.html?q=



Search

Searching for multiple words only shows matches that contain all words.

search

Search Results

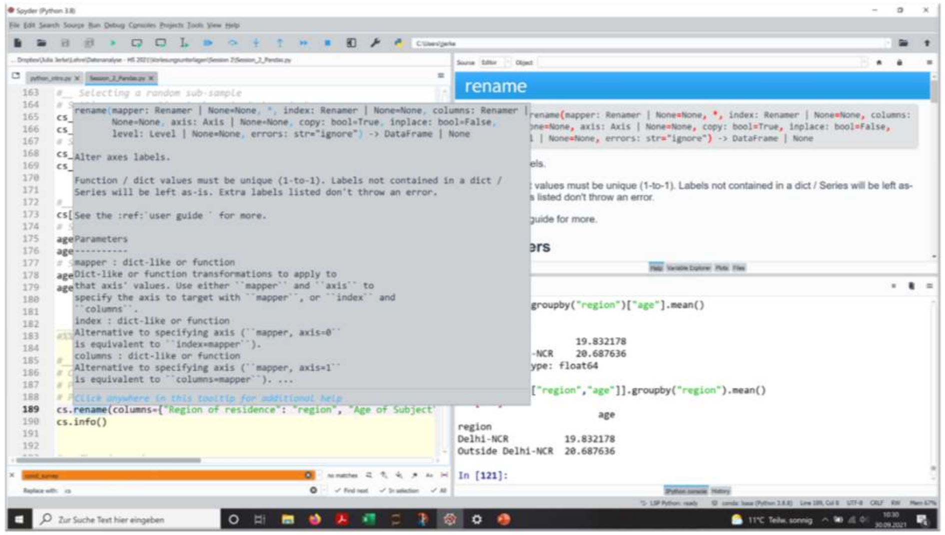
Search finished, found 70 page(s) matching the search query.

- **pandas.DataFrame.rename** (Python method, in pandas.DataFrame.rename)
- **pandas.Index.rename** (Python method, in pandas.Index.rename)
- **pandas.Series.rename** (Python method, in pandas.Series.rename)
- **Categorical data**
  ...ype='object') # uniques In [66]: s.unique() Out[66]: ['b', 'a', 'c'] Categories (4, object): ['a', 'b', 'c', 'd'] Renaming categories Renaming categories is done by assigning new values to the Series.cat.categories property or by usin...
- **Comparison with SAS**
  ...MonthEnds> [244 rows x 6 columns] Selection of columns SAS provides keywords in the DATA step to select, drop, and rename columns. data tips; set tips; keep sex total_bill tip; run; data tips; set tips; drop sex; run;...

In Spyder

– Set to cursor inside the function name and press Ctrl+I (in Windows), *or*

– Hover across the function name and click inside the pop-up window



```
rename(mapper: Renamer | None=None, *, index: Renamer | None=None, columns: Renamer |
        None=None, axis: Axis | None=None, copy: bool=True, inplace: bool=False,
        level: Level | None=None, errors: str="ignore") -> DataFrame | None

Alter axes labels.

Function / dict values must be unique (1-to-1). Labels not contained in a dict /
Series will be left as-is. Extra labels listed don't throw an error.

See the :ref:`user guide ` for more.

Parameters
----------
mapper : dict-like or function
Dict-like or function transformations to apply to
that axis' values. Use either ``mapper`` and ``axis`` to
specify the axis to target with ``mapper``, or ``index`` and
``columns``.
index : dict-like or function
Alternative to specifying axis (``mapper, axis=0``
is equivalent to ``index=mapper``).
columns : dict-like or function
Alternative to specifying axis (``mapper, axis=1``
is equivalent to ``columns=mapper``). ...
Click anywhere in this tooltip for additional help
```

```
cs.rename(columns={"Region of residence": "region", "Age of Subject"
cs.info()
```

```
groupby("region")["age"].mean()

            19.832178
-NCR        20.687636
ype: float64

["region","age"]].groupby("region").mean()

                          age
region
Delhi-NCR           19.832178
Outside Delhi-NCR   20.687636

In [121]:
```

3. Hands on

… Open *Session_2_Pandas.py*