DIS08 – Data Modeling

08.2 – Introduction to Pandas

Technology
Arts Sciences
TH Köln

Disclaimer

Slides are mainly based on https://pandas.pydata.org/docs/index.html and https://www.w3schools.com/python/pandas/pandas_intro.asp
→ Find everything you need to know there!

Some “Pandas Cheat Sheets”:

- Official cheat sheet: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf
- More beginner-friendly: https://blog.finxter.com/pandas-cheat-sheets/
- Extensive, illustrated, more like a booklet: https://www.enthought.com/wp-content/uploads/Enthought-Python-Pandas-Cheat-Sheets-1-8-v1.0.2.pdf

What is Pandas? Wy should you use it?

- Pandas is a **Python library** used for working with data sets
- It has functions for analyzing, cleaning, exploring, and manipulating data
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008
- Pandas allows us to analyze big data and make conclusions based on statistical theories
- Pandas can clean messy data sets, and make them readable and relevant

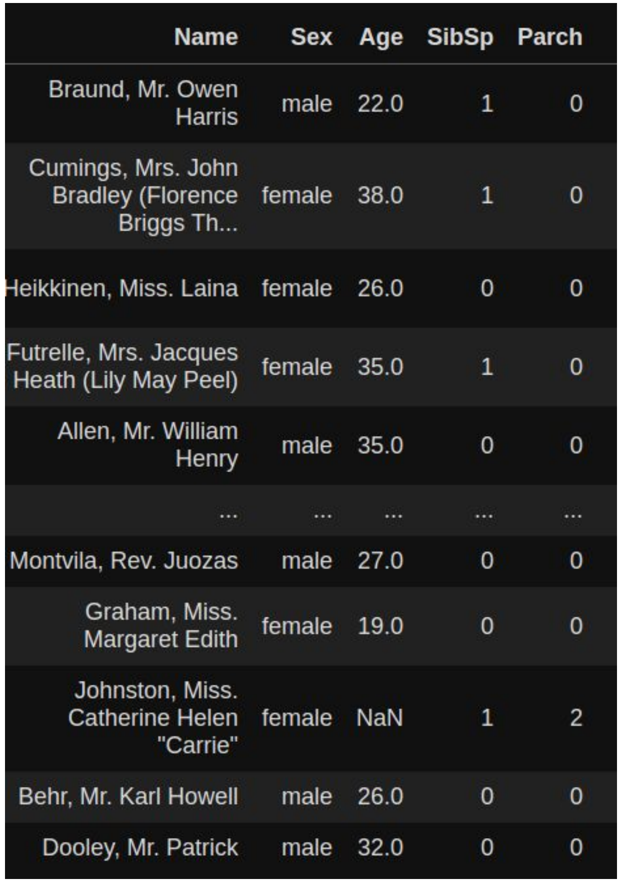
image: "panda" by Mathias Appel is marked with CC0 1.0

What can I do with Pandas?

You can, for example:

- View your data
- get a quick idea of what you are dealing with
- What is Mean, Modus, Median?
- Max value?
- Min value?
- Data types?
- Grouping, selecting, applying functions
- much more!

→ Excel, SPSS and the power of Python combined!



Name	Sex	Age	SibSp	Parch
Braund, Mr. Owen Harris	male	22.0	1	0
Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0
Heikkinen, Miss. Laina	female	26.0	0	0
Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0
Allen, Mr. William Henry	male	35.0	0	0
...	...	...	...	...
Montvila, Rev. Juozas	male	27.0	0	0
Graham, Miss. Margaret Edith	female	19.0	0	0
Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2
Behr, Mr. Karl Howell	male	26.0	0	0
Dooley, Mr. Patrick	male	32.0	0	0

Examining your Data – Find the Tasty Parts

- Pandas’ core data structure: **Series Objects**
  - one-dimensional labeled array capable of holding any **single data type** (integers, strings, floating point numbers, Python objects, etc.)
  - can be created f.e. from Dicts, Lists or Arrays
  - Can have indices (for example when created from dictionaries, if none are passed, an index will be created having the values [0, ..., len(data) - 1]
- Combine Series Objects and you get **DataFrames**
  - 2-dimensional labeled data structure with columns of different types.
  - You can think of it like a spreadsheet or SQL table, or a dict of Series objects.
  - A DataFrame has columns and indices (row keys)

image: "panda" by Mathias Appel is marked with CC0 1.0

Getting Started: Load your Data

From dictionary

```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

df = pd.DataFrame(mydataset)
```

From CSV

```
data = pd.read_csv("titanic.csv")
```

From XLSX

```
data2 = pd.read_excel(“tmp.xlsx”)
```

From JSON

```
data3 = pd.read_json(“data.json”)
```

Export your data

```
data.to_csv(“data_output.csv”)
```

For saving DataFrames, use binary Pickles

```
data.to_pickle(“data_output.pkl”)
```

And for reading the pickled file

```
df = pd.read_pickle("data_output.pkl")
```

There is Readers/Writers for many more formats. Check out the list of IO Tools: https://pandas.pydata.org/docs/user_guide/io.html

Handling Series Objects and DataFrames: Some Examples

Get indices or column names:

data.columns

and

data.index

→no function, no ()!

→you can do the same with index

Get as list, with and without Pandas function:

data.columns.to_list()

list_names = [x for x in data.Name]

→Columns, Indices and Series Objects are iterable

Get columns of a DataFrame (as SeriesObject or list):

data.VarName

and

data.VarName.to_list()

Get descriptive information:

data.info()
data.describe()
data.VarName.describe()

data.mean()
data.Income.sum()

also:

std(), min(), max(), mean(), count(), value_counts(),...

Plot your data:

data.plot()
data.plot(kind=”scatter”, x=”ColName1”, y=”ColName2”)

data.VarName.plot.hist()

→ works with a dataframe as well, rarely useful

Indexing: Select and Transform Data

Select cell content at position:

Object Type	Indexers
Series	s.loc[indexer]
DataFrame	df.iloc[row_indexer,column_indexer] and df.loc[row_indexer,column_name]

loc is label based, iloc integer based

Object Type	Selection	Return Value Type
Series	series[label]	Scalar value
DataFrame	frame[colname]	Series corresponding to colname

Filter DataFrame by criteria:

```
data.loc[data["Age"] > 35]
```

→returns copy, does not change your original DataFrame

Multiple criteria in boolean statement:

```
data.loc[(data["Age"] > 35) & (data["Survived"] == 1)]
```

read:

data.loc	→get me a copy of data
[(data["Age"] > 35)	→where* “Age” > 35
& (data["Survived"] == 1)]	→and where* “Survived” is 1

* everything in [ ] just returns a list of True and False booleans

Match Text by Regular Expression:

```
data.loc[(data["Name"].str.contains(r"^J.*"))]
```

→RegEx in quotation marks, r enables use of \ instead of \\

Editing and Adding Columns

Edit a column with simple functions:

```
data[“Name”] = data[“Name”].str.lower()
```

Add a new column to a Dataframe:

```
data[“One”] = 1
```

Add new column from other Column:

```
data[“Name2”] = data[“Name”]
```

add new Column based on condition:

```
data["Survived2"] = data["Survived"] == 1
```

add new column using the apply function:

```
data["Surname"] = data.apply(lambda row: row["Name"].split(",")[0], axis=1)
```

you can apply any function here!

image: "Panda cat" by fox_kiyo is licensed under CC BY-SA 2.0