Python
for Data Analysis

Lecture 2

Why Python?

+ Easy syntax
+ For quick scripting, but also for production software
+ Ecosystem of libraries and open-source projects

- On the minus side:
  GIL (global interpreter locker), not the best parallelization and multithreading available

Useful links & tools:
Python
Learn Python
Jupyter
Anaconda
Aalto Jupyter Hub
Google CoLab

Numpy

- n-dimensional arrays (from 1d to matrices and tensors)
- Efficient vectorized operations, great support for linear algebra

```
import numpy as np  # Following naming conventions is good

my_first_array = np.array([1,2,3])
print(array.shape)

identity_matrix_2by2 = np.eye(N=2)
```

- Array indexing and slicing (`array[row, column, channel]`), boolean indexing (`array[array>0]`), fancy indexing (`array[list_of_indexes]`)
- Data I/O

```
np.save(file, array)
array = np.load(file)
```

- Not good for mixed-type arrays and for handling missing data (but it has a good placeholder: `np.nan`)

Useful links & tools:
Numpy

Dataframes: pandas

- For tabular data and heterogeneous data
  - Both databases-like and spreadsheets-like operations
  - Fast conversion from and to numpy

```
import pandas as pd # Good to follow naming conventions

df = pd.DataFrame(...)
print(df.describe())

matrix_of_values = df.values       # this is np.array!

series = pd.Series(...)
array_of_values = series.values    # this is np.array!
```

- Access data by column name (`df["column"]`), by numerical index (`df.iloc(...)`), by index label (`df.loc(...)`), boolean selection (`df[df["column"]>0]`)

- Data I/O

```
df.to_csv(..) # or to_pickle(),to_json(),to_hdf(), ...
df = pd.read_csv(...) # or read_pickle(), read_json(), etc.
```

Useful links & tools:
Pandas
GeoPandas
Dask
Vaex
pySpark and koalas

Data Visualization

- x, y, area, and colors below can be lists, np.arrays or pd.Series

```
import matplotlib.pyplot as plt

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

- Export figures to file

```
plt.savefig("fig.png")
```

- Equivalently with other libraries

```
import seaborn as sns

sns.scatterplot(...)
```

Useful links & tools:
Matplotlib
Seaborn
Bokeh
Plotly
Datashader

Machine Learning: SciKit-Learn

- Based on numpy
- Includes classes for preprocessing (scaling, imputing, encoding), regression, classification, clustering, dimensionality reduction, model selection and evaluation
- All objects have similar API using `.fit()` and `.predict()`

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X, y)
y_pred = reg.predict(X_pred)
```

- For big learning tasks, setting the parameter `n_jobs=-1` utilizes all available processors
- Export and import trained models with pickle or joblib

```
from joblib import dump, load
dump(clf, 'filename.joblib')
clf = load('filename.joblib')
```

Useful links & tools:
Statsmodels
SciPy
Scikit-learn
Tensorflow & Keras
PyTorch

Graphs and networks

- Create and modify directed or undirected graphs

```
import networkx as nx

G = nx.Graph()

G.add_node(1)
G.add_node(2)
G.add_edge(1, 2, weight=4.7)
```

- Includes algorithms and methods for network analysis

- Easy network visualization

```
nx.draw(G)
```

Useful links & tools:
networkx

Extracting data from HTML, XML, etc.

- Navigate the tree structure of an HTML file and find all element with given tag

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html_doc, 'html.parser')

# Find all the <a> tags
soup.find_all('a')
```

- Crawl a web-page

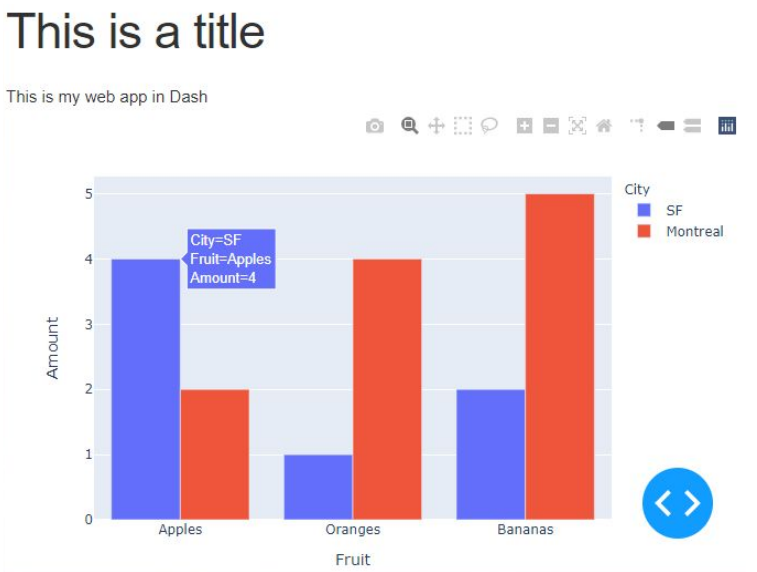
Useful links & tools:
requests
Beautiful Soup (bs4)
Selenium

Natural Language Processing

- Tokenize and tag text
- Extract entities (nouns, verbs, etc.)
- Topic modelling, document indexing and similarity retrieval
- Word2Vec (and other) pre-trained models

Useful links & tools:
Natural Language ToolKit (nltk)
Gensim

Interactive presentations



Useful links & tools:
[Flask]
[Plotly Dash]
[Streamlit]
[Django]

Database connectors

```
import pandas as pd
import duckdb


# Connect to the DuckDB database Stocks.DB
# If the database does not exist, it will be created
conn = duckdb.connect('Stocks.DB')



# Make an in-memory view of a pandas DataFrame
conn.register('stocks_prices_view', df)

# Save view to table in 'Stocks.DB'
conn.execute('CREATE TABLE stocks_prices AS
                SELECT * FROM stocks_prices_view;')



# Read a database table in a pandas dataframe
_query = "SELECT * FROM stocks_prices;"
df = conn.execute(_query).fetchdf()
```

Useful links & tools:
Sqlite3
DuckDB

Demo

- Code available from github.com/letiziaia/DSP_2021
- … and you can also copy-paste code from the book to see how it works

Have fun with your projects!