Using Python for Scientific Research

Dr. Uwe Ziegenhagen

August 20, 2016

Python and SciPy

Data Handling with pandas

Data Analysis “Swiss Banknote Data”

About Me

- Dipl.-Kfm, M. Sc. & Ph.D. in Statistics
- Seven years in Private Equity division of Deutsche Bank
- since October '15: Analyst Credit & Treasury Operations
- LaTeX enthusiast ⇒ see the Dante e.V. booth
- Treasurer for "Dingfabrik Köln e.V.", fablab & makerspace

Python

- Implementation started in the late 1980s by Guido van Rossum in the Netherlands
- emphasizes readable, understandable code
- “batteries included” ⇒ rich standard library
- my introduction to Python: download-script for SaveTV

Python “Hello World”

```python
print('Hello Python')
a = 123.4
print(a+2)

def myFunction(a):
    b = a + a
    return b

print(myFunction(2)) # 4
print(myFunction('a')) # 'aa'
```

Listing 1: Hello World in Python 3.x

Pandas

- my introduction to scientific Python: data consistency and completeness checks with `pandas`
- "pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language."[1]
- Initiated 2008 by Wes McKinney while at AQR Capital Management for high performance quantitative analysis
- Important parts implemented in C/Cython, quite fast
- Current version is 0.18.1

[1] Source: `pandas.pydata.org`

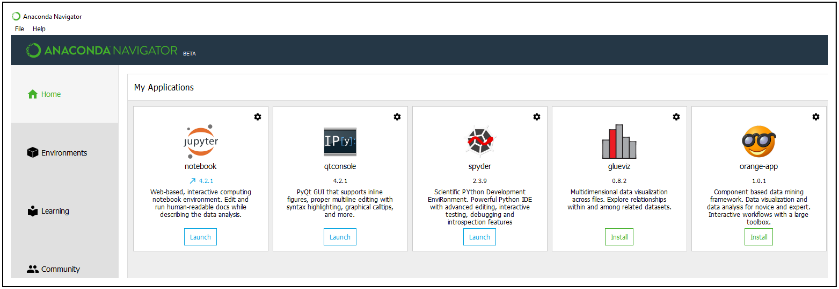
The SciPy Framework

Besides `pandas` there are

NumPy	matrices, vectors, algorithms
IPython	Matlab/Mathematica-like environment
Matplotlib	scientific plotting, basis for `seaborn` library
SymPy	symbolic mathematics
...	etc, etc

Scientific Python Distributions

- Linux/MacOS X bring Python, but not SciPy
- Install manually or use dedicated distribution
  - WinPython (`https://winpython.github.io`)
  - Anaconda (`https://www.continuum.io/downloads`)



notebook
4.2.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.2.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
2.3.9
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.8.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

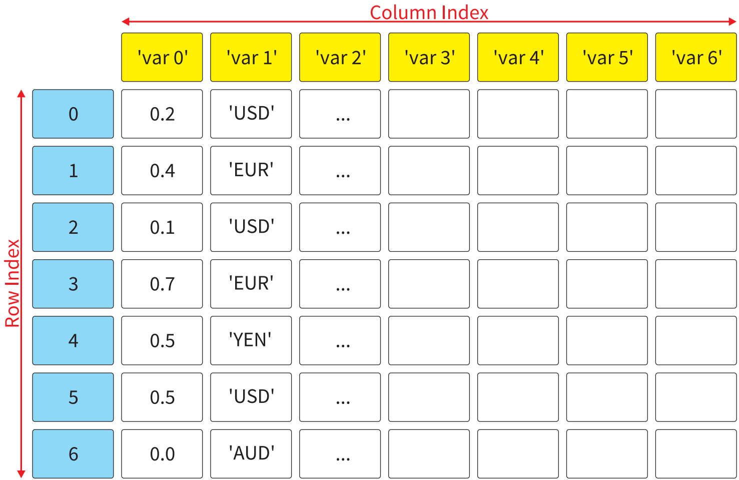
orange-app
1.0.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Structure of this Presentation

- Introduction ✓
- Data handling with pandas
    - Loading data
    - Transforming and Filtering
    -
- Analysing a real data set

Series and DataFrames

- central data structures in pandas



	'var 0'	'var 1'	'var 2'	'var 3'	'var 4'	'var 5'	'var 6'
0	0.2	'USD'	...				
1	0.4	'EUR'	...				
2	0.1	'USD'	...				
3	0.7	'EUR'	...				
4	0.5	'YEN'	...				
5	0.5	'USD'	...				
6	0.0	'AUD'	...				

Creating Series and Dataframes

- Pandas objects can also be created manually

```python
import pandas as pd

a = pd.Series([1,2,3,4,5,6,7,8,9,10])
b = pd.Series(['A','C','D','B','F','G','I','K','L','P'])
df = pd.concat([a,b], axis=1)
# alternativ
df = pd.DataFrame({'a': a,'b':b})
df = a.to_frame().join(b.to_frame())
df = pd.DataFrame(data=dict(a=a, b=b))
```

Reading Data

Command	Description
read_pickle	read Pickle objects
read_table	for general table-like formats
read_csv	Comma-Separated Values
read_fwf	for weird fixed-width formats
read_clipboard	read from clipboard
read_excel	read Excel files

other commands for HTML, JSON, HDF5, . . .

Pandas Example I: Date conversion

- Proprietary software uses **”14 Mar 1983”** as date format in CSV, Excel understands it just sometimes. . .
- Task:
  - Take the CSV data
  - Transform the “evil” dates and
  - Save the data in Excel format

```python
import pandas as pd
data = pd.read_csv(somefile.csv)
data['datecol'] = pd.to_datetime(data['datecol'])
data.to_excel('somefile.xlsx')
```

Learning from the Example. . .

Reading Data

- `import pandas as pd`
  Load the pandas library
- `read_csv`
  Load data in CSV format
- `pd.to_datetime(data['datecol'])`
  convert to Python `datetime` object, see next slides
- `to_excel`
  save data in Excel format

Pandas Dataframe Operations

Selection and Filtering

- Select only certain columns

  ```
  df = df[['colA', 'colB']]
  ```

- Select only first two rows

  ```
  df.iloc[:1]
  ```

- Select only rows where column value is greater

  ```
  df[df['colA'] > 50]
  ```

- Select only rows where column value is greater than and small than

  ```
  df[(df['colA'] > 500)| (df['colA'] < 50)]
  ```

- Select only rows where column value is not

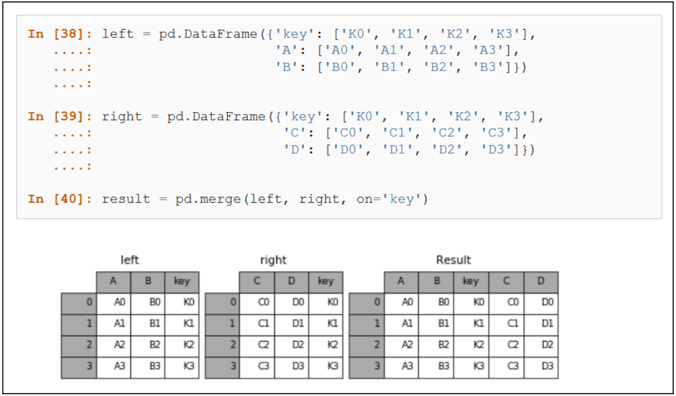
  ```
  df[˜(df['colA'] == 'HelloWorld')]
  ```

- See more here: `http://chrisalbon.com/python/pandas_indexing_selecting.html`

Pandas Dataframe Operations

Merging

- pandas supports SQL-like merging: left, right, inner, outer
- very handy to combine different datasets



```
In [38]: left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
   ....:                      'A': ['A0', 'A1', 'A2', 'A3'],
   ....:                      'B': ['B0', 'B1', 'B2', 'B3']})
   ....:

In [39]: right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
   ....:                       'C': ['C0', 'C1', 'C2', 'C3'],
   ....:                       'D': ['D0', 'D1', 'D2', 'D3']})
   ....:

In [40]: result = pd.merge(left, right, on='key')
```

left

	A	B	key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	A3	B3	K3

right

	C	D	key
0	C0	D0	K0
1	C1	D1	K1
2	C2	D2	K2
3	C3	D3	K3

Result

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K2	C2	D2
3	A3	B3	K3	C3	D3

Figure: merge, source: pandas documentation

Example: Merging Rows into Columns

Spalte	Wert
ColA	Andi
ColB	Berni
ColC	Cesar
ColA	Dorian
ColB	Ernst
ColC	Frank

```python
import pandas as pd
daten = pd.read_excel('combine.xlsx')
result = pd.DataFrame(columns=['ColA', 'ColB', 'ColC'])
for i, row in daten.iterrows():
    result.loc[i // 3, row['Spalte']] = row['Wert']

print(result)
```

Example: Creating Tax Donation Receipts

- Donations to Dingfabrik are tax-deductible
- Manual creation error-prone and labor-intensive
- Last year: complicated mix (Python, MySQL, LaTeX)
- This year: pandas, much easier
- Interested? `http://uweziegenhagen.de/?p=3359`

Example: Checking the Payment Status

- Treasurer task: check payments from Dingfabrik members
- Annoying job, lots of Excel “Mouse Schubsing”
- Idea: Analyze payment data with pandas, merge with master data
- Interested? `http://uweziegenhagen.de/?p=3350`

The Swiss Banknote Data

- Well-analyzed dataset for multivariate statistics
- see Flury/Riedwyl (1988) or Härdle/Simar (2012) for details
- consists of 100 counterfeit and 100 genuine Swiss banknotes

X_1	Length of bill in mm
X_2	Width of left edge in mm
X_3	Width of right edge in mm
X_4	Bottom margin width in mm
X_5	Top margin width in mm
X_6	Length of diagonal in mm
X_7	Status: `genuine` or `counterfeit`

The Swiss Banknote Data

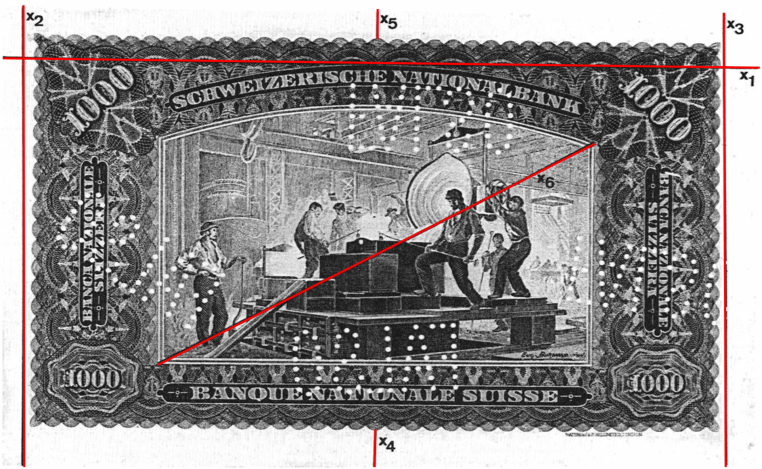


Figure: Old Swiss banknote (Source: Flury & Riedwyl)

Loading the Data

```python
import pandas as pd
import numpy as np
import seaborn as sns

data = pd.read_csv('banknote.csv', sep=';', decimal=',')
```

Generating a Summary

```python
summary = data.describe()
summary = summary[['Length', 'Left', 'Right', 'Bottom']]

print(summary)
```

```
          Length       Left      Right     Bottom
count 200.000000 200.000000 200.000000 200.000000
mean  214.896000 130.121500 129.956500   9.417500
std     0.376554   0.361026   0.404072   1.444603
min   213.800000 129.000000 129.000000   7.200000
25%   214.600000 129.900000 129.700000   8.200000
50%   214.900000 130.200000 130.000000   9.100000
75%   215.100000 130.400000 130.225000  10.600000
max   216.300000 131.000000 131.100000  12.700000
```

Generating a Boxplot

```
box = sns.boxplot(x="Status", y="Diagonal", data=data);
# save image as PDF
box.figure.savefig("../img/box.pdf")
```

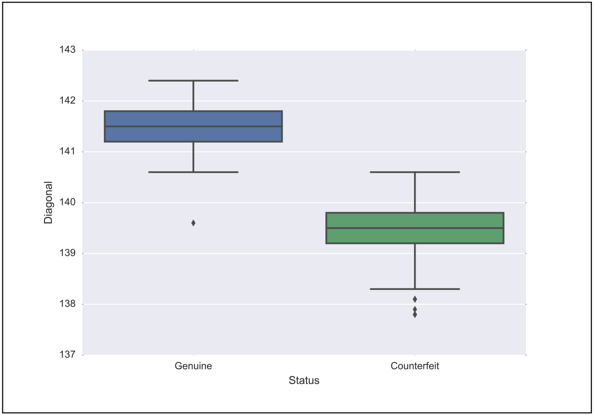


Figure: Boxplot, grouped by status

Generating a Scatterplot

```python
scatterdata = data[['Status','Length','Left','Diagonal']]
scatter = sns.pairplot(scatterdata, hue="Status")
scatter.savefig('../img/scatter.pdf')
```

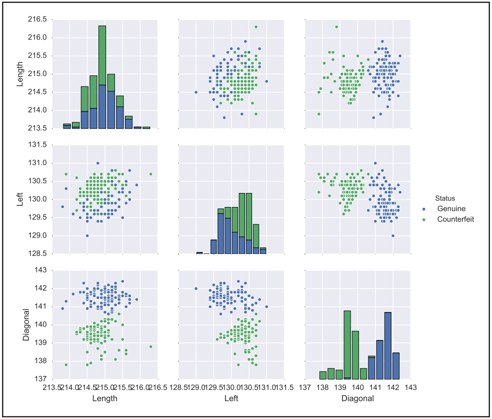


Figure: Scatterplot matrix

Cluster Analysis

- Cluster analysis tries to find groups of similar items
- hundreds of algorithms
- here *k*-means clustering, as it is rather simple to explain
- *k* is parameter for group count, here set to 2 corresponding to genuine & counterfeit

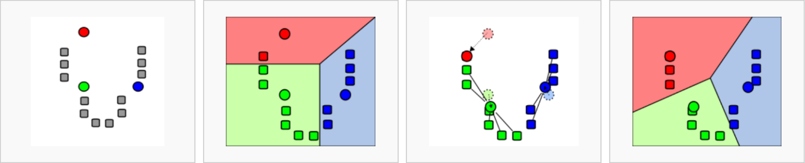


Figure: *k*-means algorithm, source: Wikipedia

Cluster Analysis

```python
import pandas as pd
from scipy.cluster.vq import kmeans,vq
import seaborn as sns

data = pd.read_csv('banknote.csv', sep=';', decimal=',')
data = data[['Length', 'Diagonal']]
clusterData = data.as_matrix()

# Compute K-Means with K = 2 clusters
centroids,_ = kmeans(data,2)
# Assign each observation to a cluster
assignment,_ = vq(data,centroids)
data['Assignment'] = assignment

scatter = sns.pairplot(data, hue='Assignment')
scatter.savefig("../img/cluster.pdf")
```

Cluster Analysis

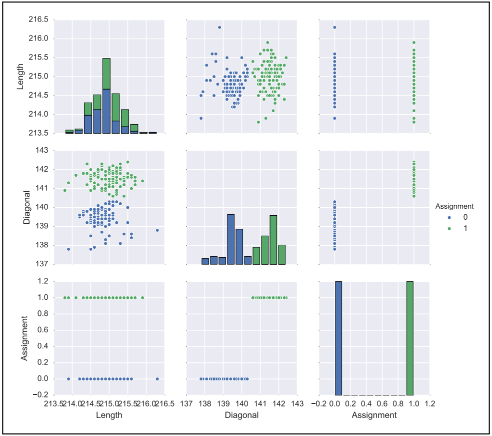


Figure: Scatterplot matrix, color not based on Status but on the computed cluster assignment

Conclusion

- Python with `pandas` proved to be a valuable tool
- Greatly simplifies my life in everyday analyses
- Check it out!
- If you have any questions, visit the Dante e.V. booth!

Literature

Besides Stackexchange. . .

- “Learning pandas”, Michael Heydt, 2015
- “Mastering pandas for Finance”, Michael Heydt, 2015
- “Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython”, Wes McKinney, 2012
- “Python Data Analytics: Data Analysis and Science using pandas, matplotlib and the Python Programming Language”, Fabio Nelli, 2015