Big Data and Automated Content Analysis

Week 5 – Thursday
»Statistics with Python«

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

30 April 2020

Today

Statistics in Python

General considerations

General considerations

After having done all your nice text processing (and got numbers instead of text!), you probably want to analyse this further.
You can always export to .csv and use R or Stata or SPSS or whatever. . .

General considerations

After having done all your nice text processing (and got numbers instead of text!), you probably want to analyse this further. You can always export to .csv and use R or Stata or SPSS or whatever. . .

BUT:

Reasons for not exporting and analyzing somewhere else

- the dataset might be too big
- it's cumbersome and wastes your time
- it may introduce errors and makes it harder to reproduce

What statistics capabilities does Python have?

- Basically all standard stuff (bivariate and multivariate statistics) you know from SPSS
- Some advanced stuff (e.g., time series analysis)
- However, for some fancy statistical modelling (e.g., structural equation modelling), you can better look somewhere else (R)

Statistics in Python

Useful packages

Useful packages

numpy (numerical python) Provides a lot of frequently used functions, like mean, standard deviation, correlation, . . .

scipy (scientific python) More of that ;-)

statsmodels Statistical models (e.g., regression or time series)

matplotlib Plotting

seaborn Even nicer plotting

Example 1: basic numpy

```
import numpy as np
x = [1,2,3,4,3,2]
y = [2,2,4,3,4,2]
z = [9.7, 10.2, 1.2, 3.3, 2.2, 55.6]
np.mean(x)
```

```
2.5
```

```
np.std(x)
```

```
0.9574271077563381
```

```
np.corrcoef([x,y,z])
```

```
array([[ 1.        ,  0.67883359, -0.37256219],
       [ 0.67883359,  1.        , -0.56886529],
       [-0.37256219, -0.56886529,  1.        ]])
```

Characteristics

- Operates (also) on simple lists
- Returns output in standard datatypes (you can print it, store it, calculate with it, . . .)
- it's fast! `np.mean(x)` is faster than `sum(x)/len(x)`
- it is more accurate (less rounding errors)

Example 2: basic plotting

```python
import matplotlib.pyplot as plt
x = [1,2,3,4,3,2]
y = [2,2,4,3,4,2]
plt.hist(x)
plt.plot(x,y)
plt.scatter(x,y)
```

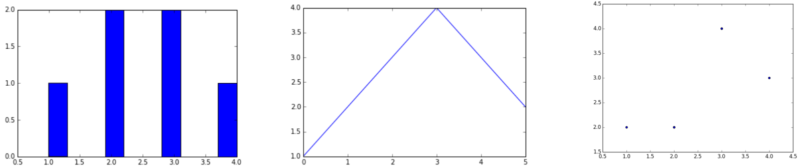

Figure: Examples of plots generated with `matplotlib`

Pandas

Working with dataframes

When to use dataframes

Native Python data structures (lists, dicts, generators)

pro:

- flexible (especially dicts!)
- fast
- straightforward and easy to understand

con:

- if your data is a table, modeling this as, e.g., lists of lists feels unintuitive
- very low-level: you need to do much stuff 'by hand'

When to use dataframes

Native Python data structures (lists, dicts, generators)

pro:

- flexible (especially dicts!)
- fast
- straightforward and easy to understand

con:

- if your data is a table, modeling this as, e.g., lists of lists feels unintuitive
- very low-level: you need to do much stuff 'by hand'

Pandas dataframes

pro:

- like an R dataframe or a STATA or SPSS dataset
- many convenience functions (descriptive statistics, plotting over time, grouping and subsetting, . . .)

con:

- not always necessary ('overkill')
- if you deal with really large datasets, you don't want to load them fully into memory (which pandas does)

Pandas

Plotting and calculating with Pandas

More examples here: https://github.com/damian0604/bdaca/blob/master/ipynb/basic_statistics.ipynb

OLS regression in pandas

```python
import pandas as pd
import statsmodels.formula.api as smf

df = pd.DataFrame({’income’: [10,20,30,40,50], ’age’: [20, 30, 10, 40, 50], ’facebooklikes’: [32, 234, 23, 23, 42523]})

# alternative: read from CSV file (or stata...):
# df = pd.read_csv(’mydata.csv’)

myfittedregression = smf.ols(formula=’income ~ age + facebooklikes’, data=df).fit()
print(myfittedregression.summary())
```

```
OLS Regression Results
==============================================================================
Dep. Variable:              income  R-squared:                          0.579
Model:                         OLS  Adj. R-squared:                     0.158
Method:               Least Squares F-statistic:                        1.375
Date:              Mon, 05 Mar 2018 Prob (F-statistic):                 0.421
Time:                     18:07:29  Log-Likelihood:                   -18.178
No. Observations:                5  AIC:                                42.36
Df Residuals:                    2  BIC:                                41.19
Df Model:                        2
Covariance Type:         nonrobust
=================================================================================
coef   std err         t     P>|t|     [95.0% Conf. Int.]
---------------------------------------------------------------------------------
Intercept      14.9525    17.764     0.842     0.489      -61.481   91.386
age             0.4012     0.650     0.617     0.600       -2.394    3.197
facebooklikes   0.0004     0.001     0.650     0.583       -0.002    0.003
==============================================================================
Omnibus:                       nan  Durbin-Watson:                      1.061
Prob(Omnibus):                 nan  Jarque-Bera (JB):                   0.498
Skew:                       -0.123  Prob(JB):                           0.780
Kurtosis:                    1.474  Cond. No.                        5.21e+04
==============================================================================
```

Other cool df operations

df['age'].plot() to plot a column

df['age'].describe() to get descriptive statistics

df['age'].value_counts() to get a frequency table

and MUCH more...

Recoding and transforming

To transform your data, you can use `.apply()`, `.applymap()`, and `.map()` or the .str.XXX() methods:

```
df['is_center'] = df['hood'].str.contains('[cC]enter')
```

or define your own function:

```
def is_center(x):
    return int(x.lower().find('center') > -1)

df['is_center'] = df['hood'].map(is_center)
```

or use a throwaway-function:

```
df['is_center'] = df['hood'].map(lambda x: int(x.lower().find('center') > -1))
```

A notebook

https://github.com/damian0604/bdaca/blob/master/ipynb/basic_statistics.ipynb