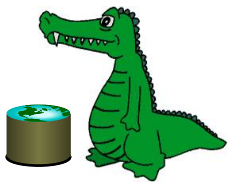
Data Processing with Pandas

Lab 0 – Part 2

Miguel Rodríguez

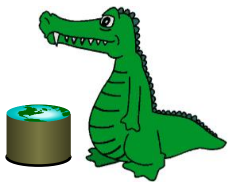
Goal

- Learn how to use more advanced data processing tools
  - Python
  - Pandas – data processing
  - Matplotlib – Visualization



Pandas

- Main data structures
  - Series: one-dimensional collections of any data type.
  - DataFrames: two-dimensional data structures similar to a database table.



The Basics

- Import libraries

```
from pylab import *
import pandas as pd
```

- Create DataFrame

```
df = pd.DataFrame( {    'a' : [1, 2, 3, 4],
                        'b': [ 'w', 'x', 'y', 'z'] })
```

	a	b
0	1	W
1	2	X
2	3	Y
3	4	Z

The Basics - explore

- Detailed information about schema
  `df.info()`
- Check first / last few rows
  - `df.head(n)`
  - `df.tail(n)`
- Any range
  - `df[1:3]`

Basics – describe

- `df.describe()`

	a
count	4
mean	2.5
std	1.290994
min	1
Max	4

Import Dataset

- Dataset can be downloaded from canvas

```
log_df = pd.read_csv(
#Path
"/home/datascience/wc_day6_1_sample.csv",
#Column Headers
names=[“ClientID”,”Date”,”Time”,”URL”,
       ”ResponseCode”,“Size”],
#Non-Value
na_values=['-'])
```

SQL Operators - basic

- Row filters (selection from RA)

```
is_may1st = log_df['Date'] == '01/May/1998’
may1_df = log_df[is_may1st]
OR
may1_df = log_df[log_df['Date'] == '01/May/1998']
```

- Column filters (project from RA)

```
url_codes = log_df[['URL', 'ResponseCode']]
```

SQL operators - grouping

- Form groups (SQL group by)

```
grouped = log_df.groupby('ResponseCode')
grouped.groups.keys()
grouped.get_group(200)
```

- Resturns a DataFrameGroupBy object
  - Much like a dictionary: Keys are grouping values that maps to a DataFrame with all objects in that group
- Operations for each group

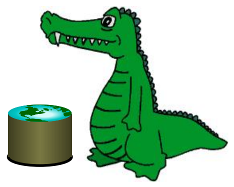
```
grouped.describe()
grouped.size()
grouped.sum(), grouped.mean(), grouped.median()
```

SQL operations – Aggregate Functions

- Use python **lambda** functions for anonymous definition and apply to use the function

```
log_df['DateTime'] =
    pd.to_datetime( #Transform str to datetime
        log_df.apply( #Use the function
            lambda row: row['Date'] + ' ' + row['Time'],
axis=1))
```

  - The **axis = 1** parameter indicates for each row, 0 indicates column



SQL Operators – Aggregate functions

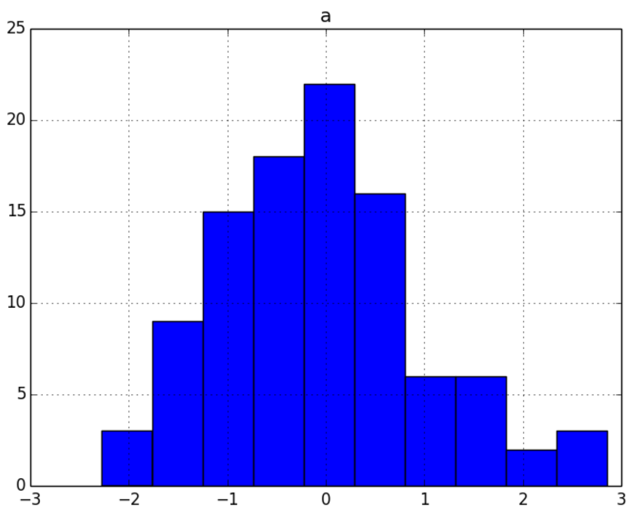
- Aggregate functions can be used to filter

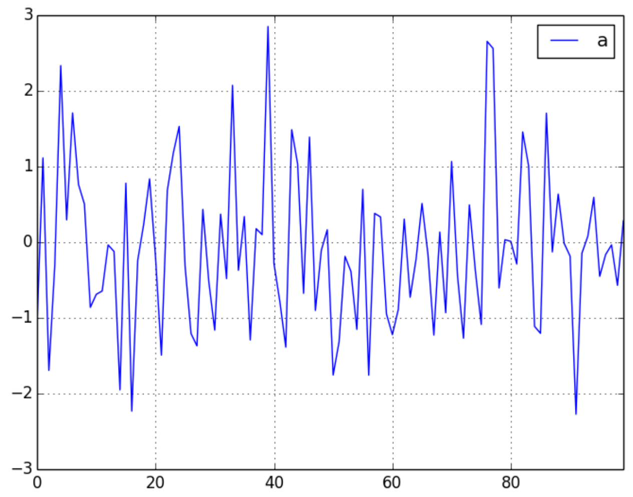
```
hour_grouped = log_df.groupby(lambda
row: log_df['DateTime'][row].hour)
```

- Note that aggregate functions (lambda) can be applied to every group in a `DataFrameGroupBy` using `apply()`

Visualize a DataFrame

```
rand_df = pd.DataFrame({'a' :
randn(100)})
rand_df.plot()
rand_df.hist()
show()
```


a
25
20
15
10
5
0
−3
−2
−1
0
1
2
3


a
3
2
1
0
−1
−2
−3
0
20
40
60
80