Python: Data Handling

Popular libraries

iPython

Pandas

Matplotlib

Outline

- iPython
  - Greater flexibility than a command line
- Pandas
  - Quick data handling
- Matplotlib
  - Visualization
- Use iPython for:
  - Prototyping
  - Data Exploration

Scenario (Data 1)

- Listing of museums:

NOMREG	Region name
NOMDEP	**Department name**
DATEAPPELLATION	Name date
FERME	Closed (NON,OUI)
ANNREOUV	
ANNEXE	Annexed name
NOM DU MUSEE	Name
ADR	address
VILLE	site
SITWEB	url
FERMETURE ANNUELLE	Closed period
PERIODE OUVERTURE	Open period
JOURS NOCTURNES	“After dark” days

https://www.data.gouv.fr/en/datasets/liste-et-localisation-des-musees-de-france/

Scenario (Data 2)

- Population per department

Correlation between Number of museums and population in a Department?

Quickly :scatter plot the two values

2010_Rank	Rank of department
Department	**Name**
Pop_31	1931 population
Pop_99	1999 population
Pop_08	2008 population
Pop_10	**2010 population**
Area	Area km2
Area_pop/km2	pop/km2
INSEE Dept_No	INSEE nunmber

https://en.wikipedia.org/wiki/List_of_French_departments_by_population

iPython

- Enchanted command line : *ipython*
- Editor in browser : *ipython notebook*
  - *http://localhost:8888/tree*
  - *Start a new notebook*
    - *It will be saved in the path where you started the server*
  - *Separates code in sequences of "cells"*
    - *Edit/Run code only in a cell*
    - *Run all code*

Pandas

- A framework to handle data by columns
  - Dataframes
- Useful to load CSV files
- Some RDBMS concepts included
    - e.g. JOIN, group by, selecting, indexes

```
import pandas as pd
s = pd.Series([1,2,3,np.nan,4,5])
print s
```

```
df = pd.DataFrame({ 'A' : np.arange(1,5),
                    'B' :
[pd.Timestamp('2015090%d'%x) for x in np.arange(1,5)],
                    'C' : 'foo' })
print df
```

Lets load the data

```
mus_list = pd.read_csv(
        ‘path_to_file\Liste_musees_de_France_utf8.tsv',
        sep='\t')

dep_pop_area = pd.read_csv(‘path_to_file\dep_pop_area.csv')

#view a little bit of the data
print mus_list.head()
```

Count museums per Department

```
#group by column(s)
#get the number of elements per group
#alternatives : .agg(['count']) we can add others like sum
#seset index : new dataframe is indexed on the grouping column
#get the dataframe of the grouping
count_bydep = pd.DataFrame(
               mus_list.groupby(['NOMDEP']).size().reset_index())
#rename columns
count_bydep.columns=['NOMDEP','COUNT']
#Get max value of count
maxc=count_bydep['COUNT'].max()

print maxc
```

Join with department population

```
#in order for the join to work the values must match
#in the two datasets
dep_pop_area['Department']=dep_pop_area['Department'].str.upper()
#merge on the common field(s)
data = pd.merge(
               count_bydep,
               dep_pop_area,
               left_on='NOMDEP',
               right_on='Department')
#get the row with the highest count of museums
print data.ix[data['COUNT'].idxmax()]

print "-------"

print data.head()
```

Plotting : simple scatter plot

- Matplotlib : Matlab like ploting
  - A great variety of functions and parameters to plot data

```
%matplotlib inline#put this at the very begining

import matplotlib.pyplot as plt

#get the values of the two columns we want to investigate
vals=data[['COUNT','Pop_10']].values

fig = plt.figure() # initialize a figure
ax = fig.add_subplot(111) #we need the axes to put names
plt.scatter(vals[:,0],vals[:,1]) # this does the scatter plot
ax.set_title('Number of museums vs population')
ax.set_xlabel('Number of museums')
ax.set_ylabel('Population')
ax.set_yscale('log') # we can change the scale of axes
plt.show() #show the plot … we could also save it : savefig
```

Scenario 2 : Titanic data

- Listing of passengers in the RMS Titanic

PassengerId	
Survived	0=no , 1=yes
Pclass	Passenger class 1=1st , 2=2nd, 3=3rd
Name	
Sex	Male, Female
Age	(could be missing)
SibSp	Number of Siblings/ Spouses Aboard

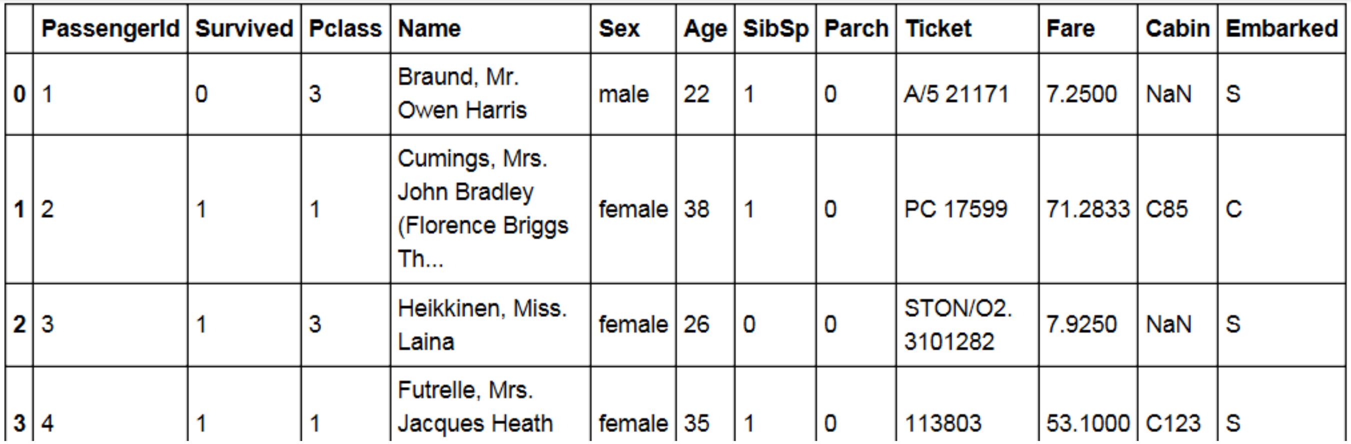
Parch	Number of Parents/Children Aboard
Ticket	Ticket number
Fare	
Cabin	Cabin No

Embarked	Port of Embarkation
	(C = Cherbourg;
	Q = Queenstown;
	S = Southampton)

https://www.kaggle.com/c/titanic

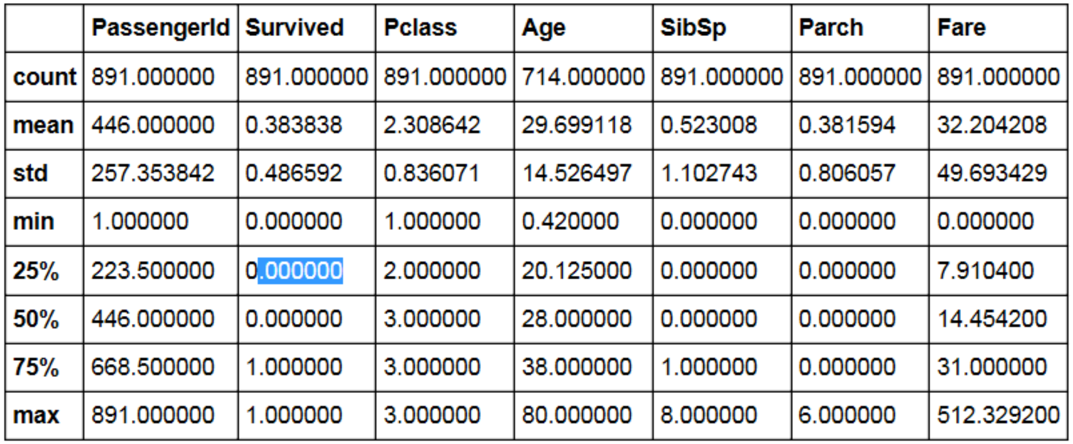
Quick view of the data

```
titanic.head()
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath	female	35	1	0	113803	53.1000	C123	S

```
titanic.describe()
```



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Quick Statistics

```
titanic['Age'].median()
titanic['Age'].var()
titanic['Sex'].unique()
#how many below the age of 10?
titanic[titanic['Age']<10].shape[0]
#how many are missing the field Age
titanic[pd.isnull(titanic['Age'])].shape[0]
```

- How about a histogram?

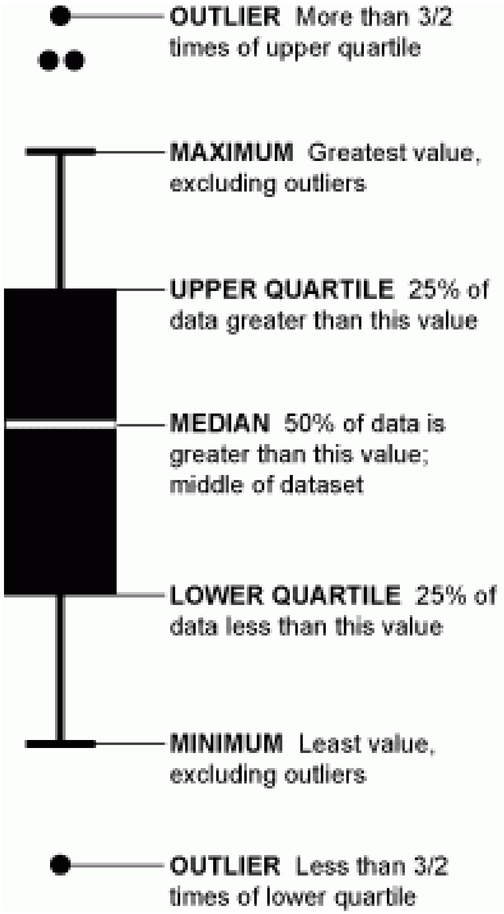
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(
        titanic['Age'],
        bins = 20,
        range = (titanic['Age'].min(),titanic['Age'].max()))
plt.title('Age distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

Box plots to evaluate outliers

```
titanic.boxplot(column='Fare')

#OR

titanic.boxplot(column='Fare', by='Pclass')
```



OUTLIER More than 3/2 times of upper quartile

MAXIMUM Greatest value, excluding outliers

UPPER QUARTILE 25% of data greater than this value

MEDIAN 50% of data is greater than this value; middle of dataset

LOWER QUARTILE 25% of data less than this value

MINIMUM Least value, excluding outliers

OUTLIER Less than 3/2 times of lower quartile

Analysis on Categorical Data

```
#count passengers per class
groupbyclass = titanic.groupby('Pclass')['Survived'].count()

fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.set_xlabel('Pclass')
ax1.set_ylabel('Count')
ax1.set_title("Passengers by Pclass")

groupbyclass.plot(kind='bar')
```

Survived vs Not Survived

```
#count passengers per class
surv_class = pd.crosstab(
               [titanic['Pclass']],
               titanic['Survived'].astype(bool))

surv_class.plot(kind='bar’)
```

- What do you notice?
- What if we want to combine categories:
  - e.g. class and sex

Filling missing values

- Many learning models don’t handle missing values

```
#get mean value of age
meanAge = np.mean(titanic['Age'])
#fill in missing with mean value
titanic['Age'] = titanic['Age'].fillna(meanAge)
```

- Wouldn’t be better if we had a mean per sex?

```
titanic['Name'].head()
```

Apply function to DataFrame column

```
#function get the title from the first name (Mr. Miss etc)
def title(w):
 return w.split(',')[1].split('.')[0].strip()

#Apply function to column and get a new column as result
titanic['title']=titanic['Name'].apply(title)
```

- How many people per title?
  - Are all titles usable?
  - Try to group the infrequent titles into a new title
  - Lets al

Mean by group

```
mean_byg=titanic[['Pclass','Sex','title','Age']].
               groupby(['Pclass','Sex','title']).
               agg('mean').
               reset_index().
               values

#we can set ranges with multiple Boolean indexes

titanic.loc[
        (pd.isnull(titanic['Age'])) & (titanic['Sex']=='female'),
        ['Age']]
        ]==value
```

- Set all the missing ages per group
- What would you do for the outliers?