Data Visualization

Andrea Giachetti
Department of Computer Science, University of Verona, Italy

andrea.giachetti@univr.it

Vis tools

- Many tools are available for data visualization
- See
  - https://mschermann.github.io/data_viz_reader/fundamentals.html#data-visualization-tools
  - Note: typically Scivis tool not included

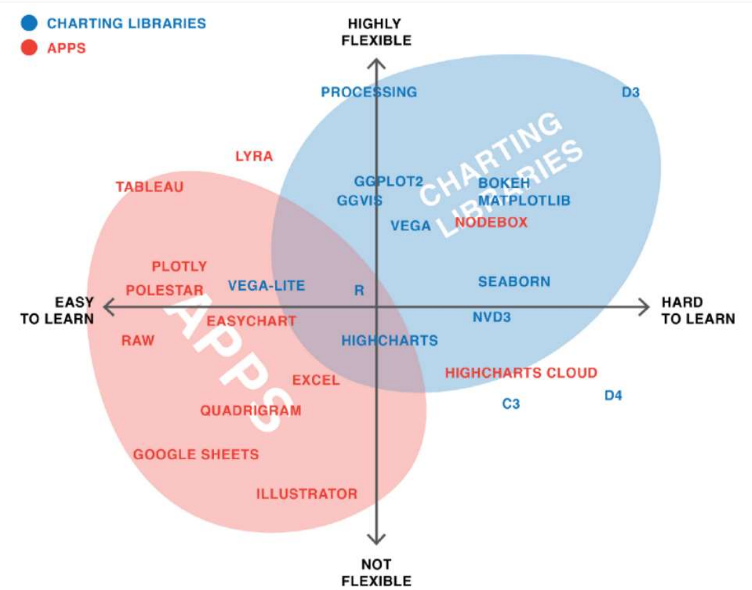
Vis tools

- Many tools are available for data visualization
- See
  - https://mschermann.github.io/data_viz_reader/fundamentals.html#data-visualization-tools
  - Note: typically Scivis tool not included

	STATIC	WEB - INTERACTIVE
APPS	ILLUSTRATOR, NODEBOX, EXCEL, POLESTAR, RAW	HIGHCHARTS CLOUD, QUADRIGRAM, EASYCHRT, DATAWRAPPER, TABLEAU, PLOTLY, GOOGLE SHEETS
CHARTING LIBRARIES	GGPLOT2, MATPLOTLIB, R, SEABORN, BOKEH, PROCESSING	D3, D4, C3, NVD3, GGVIS, HIGHCHARTS, SHINY, VEGA, VEGA-LITE

Vis tools

- Many tools are available for data visualization
- See
  - https://mschermann.github.io/data_viz_reader/fundamentals.html#data-visualization-tools
  - Note: typically Scivis tool not included



Vis libraries/tools

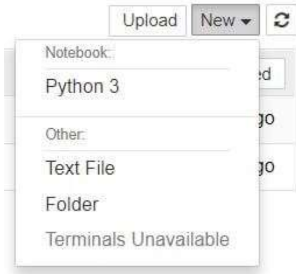
- Python has many options for data visualization
- Each visualisation library has a particular audience
- Javascript backend is mostly used to extend power of the visualisation
- Python's extensive data processing tools integrates well with visualisation requirements
- https://mode.com/blog/python-data-visualization-libraries/

Lab

- We will use Jupyter-notebook
  - Template and exercises on github
    - https://github.com/giach68/DataVis2122
  - Local way
    - install anaconda https://www.anaconda.com
  - Alternative: use (for example) binder
    - Use link on the web page

Local setup

- Download anaconda
  - https://www.anaconda.com/
  - Follow instructions
  - Other method for python users: pip install jupyter
- Start jupyter-notebook from anaconda menu or from command line. Create new notebook

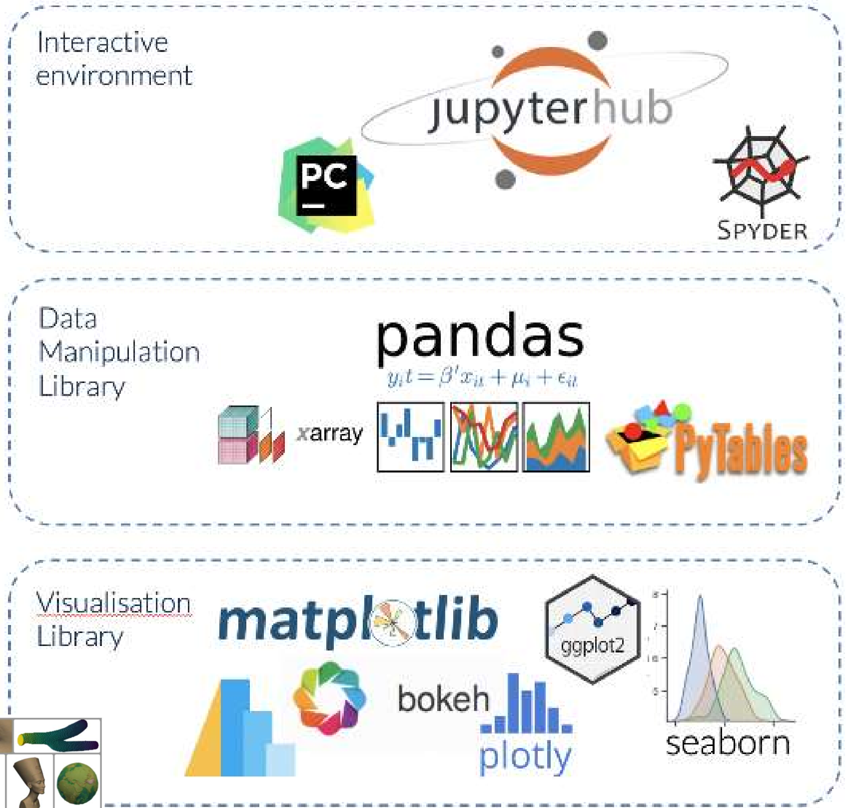


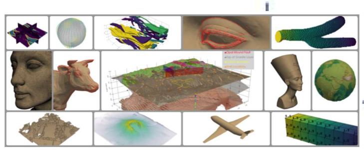
- Browser opened on
  - http://localhost:8888/tree. Local web server
- Quick tutorial here
  - https://www.dataquest.io/blog/jupyter-notebook-tutorial/

Python visualization setup

- We can add tools for geometry/volume visualization, e.g. pyvista

From Shane Lynn





pyvista

We will use

- Pandas: tabular data management
- Matplotlib: Matlab-like plotting
- Seaborn: Based on Matplotlib
  - simplified drawing of complex chart with attractive visual impact
  - Similar (in style) to the popular ggplot2 library in R
- Plotly: interactive plotting, maps
- Folium: maps
- Pyvista: scientific visualization (on top of VTK)

Pandas dataframes (Table data)

- Python | Pandas DataFrame
- Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Reading data using pandas

```
In [ ]: #Read csv file
        df = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
```

Note: The above command has many optional arguments to fine-tune the data import process.

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Exploring data frames

```
In [3]: #List first 5 records
        df.head()
```

Out[3]:

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Data Frame data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame data types

In [4]:
```python
#Check a particular column type
df['salary'].dtype
```

Out[4]: dtype('int64')

In [5]:
```python
#Check types for all the columns
df.dtypes
```

Out[4]:
```
rank          object
discipline    object
phd           int64
service       int64
sex           object
salary        int64
dtype: object
```

Data Frames attributes

Python objects have *attributes* and *methods*.

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data Frames methods

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Selecting a column in a Data Frame

- Method 1: Subset the data frame using column name:
  df['sex']

- Method 2: Use the column name as an attribute:
  df.sex

Note: there is an attribute *rank* for pandas data frames, so to select a column with a name "rank" we should use method 1.

Data Frames groupby method

- Using "group by" method we can:
  - Split the data into groups based on some criteria
  - Calculate statistics (or apply a function) to each group

```
In [ ]: #Group data using rank
        df_rank = df.groupby(['rank'])
```

```
In [ ]: #Calculate mean value for each numeric column per each group
        df_rank.mean()
```

rank	phd	service	salary
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Data Frames groupby method

- Once groupby object is create we can calculate various statistics for each group:

In [ ]:
```
#Calculate mean salary for each professor rank:
df.groupby('rank')[['salary']].mean()
```

	salary
rank	
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Note: If single brackets are used to specify the column (e.g. salary), then the output is Pandas Series object. When double brackets are used the output is a Data Frame

Data Frames groupby method

- groupby performance notes:
  - no grouping/splitting occurs until it's needed. Creating the groupby object only verifies that you have passed a valid mapping
  - by default the group keys are sorted during the groupby operation. You may want to pass sort=False for potential speedup:

```
In [ ]: #Calculate mean salary for each professor rank:
        df.groupby(['rank'], sort=False)[['salary']].mean()
```

Data Frame: filtering

- To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the salary value is greater than $120K:

```
In [ ]: #Calculate mean salary for each professor rank:
        df_sub = df[ df['salary'] > 120000 ]
```

Any Boolean operator can be used to subset the data:

\> greater; >= greater or equal;

< less; <= less or equal;

== equal; != not equal;

```
In [ ]: #Select only those rows that contain female professors:
        df_f = df[ df['sex'] == 'Female' ]
```

Data Frames: Slicing

- There are a number of ways to subset the Data Frame:
  - one or more columns
  - one or more rows
  - a subset of rows and columns

  - Rows and columns can be selected by their position or label

Data Frames: Slicing

When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

In [ ]:
```
#Select column salary:
df['salary']
```

When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

In [ ]:
```
#Select column salary:
df[['rank','salary']]
```

Data Frames: Selecting rows

If we need to select a range of rows, we can specify the range using ":"

```
In [ ]: #Select rows by their position:
        df[10:20]
```

Notice that the first row has a position 0, and the last value in the range is omitted:
So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

Data Frames: method loc

If we need to select a range of rows, using their labels we can use method loc:

In [ ]:
```
#Select rows by their labels:
df_sub.loc[10:20,['rank','sex','salary']]
```

Out[ ]:

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Data Frames: method iloc

If we need to select a range of rows and/or columns, using their positions we can use method iloc:

```
In [ ]: #Select rows by their labels:
        df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

Data Frames: method iloc (summary)

```
df.iloc[0]   # First row of a data frame
df.iloc[i]   #(i+1)th row
df.iloc[-1]  # Last row
```

```
df.iloc[:, 0]   # First column
df.iloc[:, -1]  # Last column
```

```
df.iloc[0:7]        #First 7 rows
df.iloc[:, 0:2]     #First 2 columns
df.iloc[1:3, 0:2]   #Second through third rows and first 2 columns
df.iloc[[0,5], [1,3]]  #1st and 6th rows and 2nd and 4th columns
```

Data Frames: Sorting

We can sort the data by a value in the column. By default the sorting will occur in ascending order and a new data frame is return.

In [ ]:
```
# Create a new data frame from the original sorted by the column Salary
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Data Frames: Sorting

We can sort the data using 2 or more columns:

In [ ]:
```
df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False])
df_sorted.head(10)
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

Missing Values

Missing values are marked as NaN

In [ ]:
```
# Read a dataset with missing values
flights = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/flights.csv")
```

In [ ]:
```
# Select the rows that have at least one missing value
flights[flights.isnull().any(axis=1)].head()
```

Out[ ]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N31412	1228	EWR	SAN	NaN	2425	18.0	7.0
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EHAA	791	LGA	DFW	NaN	1389	NaN	NaN
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EVAA	1925	LGA	MIA	NaN	1096	NaN	NaN
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N12221	1299	EWR	RSW	NaN	1068	21.0	45.0
858	2013	1	2	NaN	NaN	NaN	NaN	AA	NaN	133	JFK	LAX	NaN	2475	NaN	NaN

Missing Values

There are a number of methods to deal with missing values in the data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Values

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded
- Many descriptive statistics methods have skipnan option to control if missing data should be excluded. This value is set to True by default

Aggregation Functions in Pandas

- Aggregation - computing a summary statistic about each group, i.e.
  - compute group sums or means
  - compute group sizes/counts
- Common aggregation functions:
  - min, max
  - count, sum, prod
  - mean, median, mode, mad
  - std, var

Aggregation Functions in Pandas

- agg() method are useful when multiple statistics are computed per column:

```
In [ ]: flights[['dep_delay','arr_delay']].agg(['min','mean','max'])
```

Out[ ]:

	dep_delay	arr_delay
min	-16.000000	-62.000000
mean	9.384302	2.298675
max	351.000000	389.000000

Basic Descriptive Statistics

df.method()	description
describe	Basic statistics (count, mean, std, min, quantiles, max)
min, max	Minimum and maximum values
mean, median, mode	Arithmetic average, median and mode
var, std	Variance and standard deviation
sem	Standard error of mean
skew	Sample skewness
kurt	kurtosis

Matplotlib

- Set of methods that make python work like matlab
- Note Pandas has integrated visualization API based on Matplotlib
- Flexible, complete.
  - 2 interfaces:
    - Matlab style plotting (Stateful)
      - Plotting methods are called from the **pyplot** package
      - They all work on the current Figure and Axes
    - Object oriented (Stateless)
      - Plot functions are called as methods of a specific Figure and Axes
      - This allows modifying many objects at a time(the system does not keep a "current object" state)
- Complete tutorials https://matplotlib.org/stable/tutorials/index.html

Matplotlib

- Local
  - Downloadthe Lab1 folder
  - Launch from there Jupyter-notebook
    - Load notebooks or create new ones
- Binder
  - Click on the link

Let’s see

  - Line plots, scatter plots, bar charts, pie charts
  - styles, oo interface, load data save figs, and more

Examples (01_plot_1)

- Line plots - lineplots.ipynb
- Scatter plots - scatterplots.ipynb
- Bar charts - bars.ipynb
- Pie charts - PieChart.ipynb
- Styling - styles.ipynb
- Color.ipynb