COMP47250

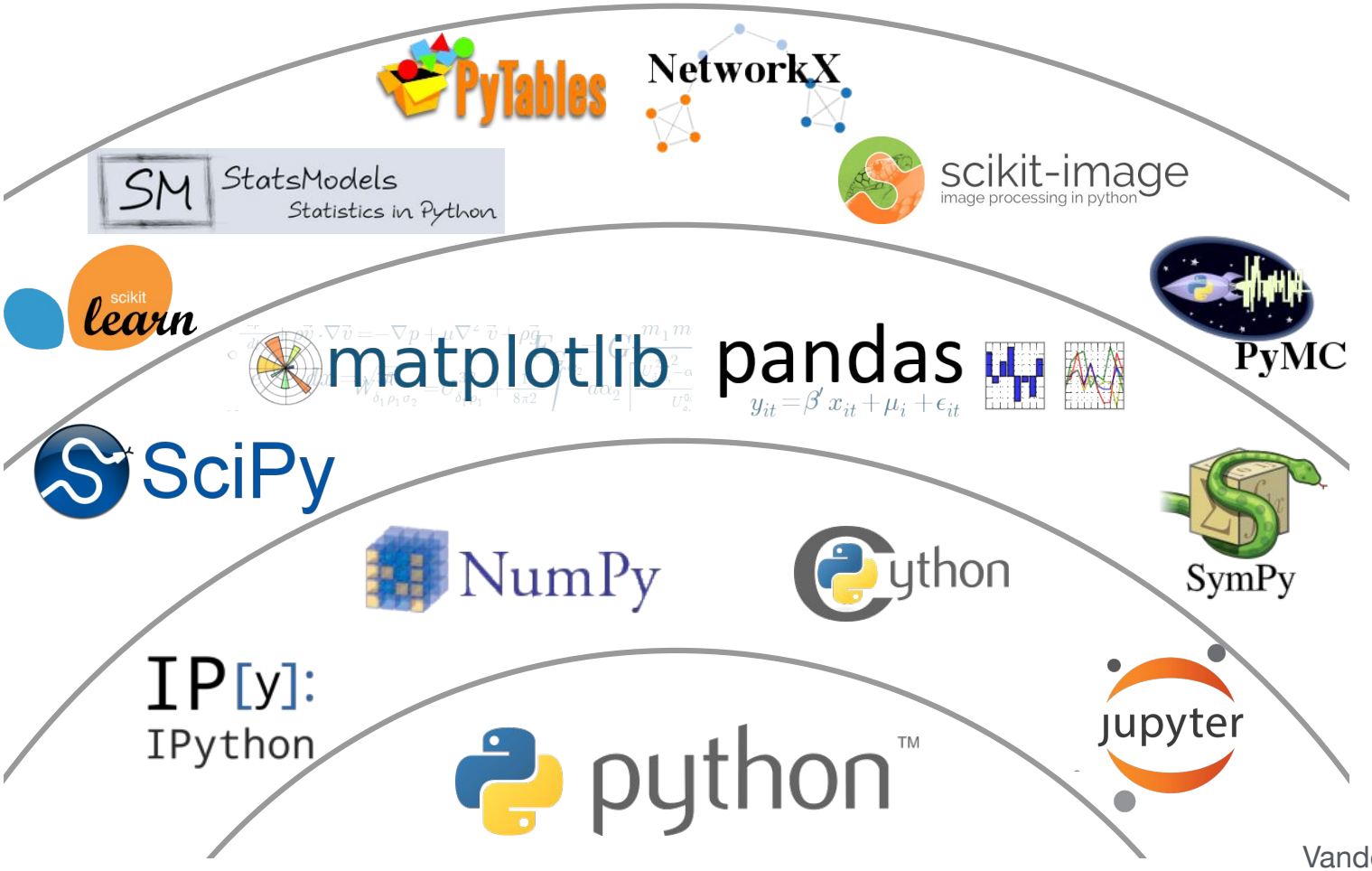
Overview of the Python Data Science Stack

Simon Caton
Based on slides by Derek Green

UCD School of Computer Science
Summer 2019

Python Data Science Stack

- Python provides a wealth of third-party open source packages and tools for data science and analysis tasks.



VanderPlas, 2016

Anaconda Python Distribution

- Easiest way to access the Python Data Science Stack is to install the **Anaconda** Python Distribution, which includes most of the relevant packages pre-installed.

https://www.continuum.io/downloads

- Anaconda provides distributions for both Python 2 and 3.
- Python 3.x is recommended for new code and fixes many of the issues and inconsistencies from Python 2.
- Once Anaconda is installed, to add extra third party packages, run the **conda** tool on the terminal/command line.

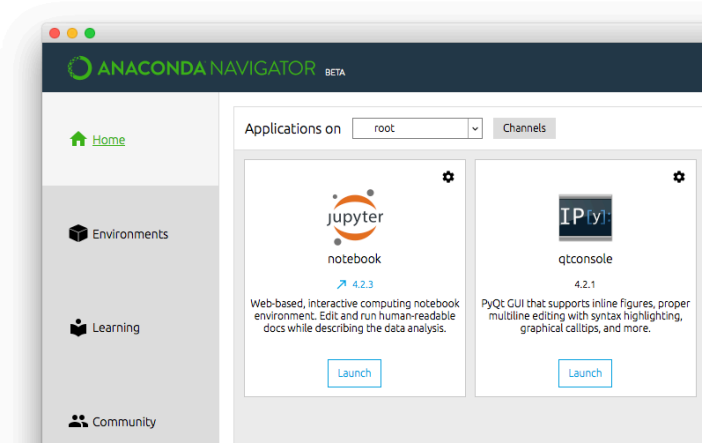
```
conda install <package_name>
```

IPython / Jupyter Notebooks

- **Jupyter project:** Web application for interactive data science and scientific computing.
- **IPython Notebooks:** The engine for running Python code under the Jupyter system.
- To start the Notebook server, either:
  1. In the terminal, type

     ```
     jupyter notebook
     ```

  2. Or click the Anaconda Navigator icon, then choose **jupyter-notebook** from the list of apps.



- This should load the IPython Notebook dashboard in your browser. Later you can also manually go to http://localhost:8888

Datacamp

- An online e-learning platform for data science
- Has many online tutorials for R, Python, SQL, etc. from beginner to intermediate
  - Great to pick up the basics or refresh what you forgot

- We've arranged for premium access for you to datacamp.com you will receive an email this week to your ucd email.
  - If you DO NOT want to be signed up, let me know by 4pm 29/5/2019 by email to simon.caton@ucd.ie
  - You will have access for at least the duration of the module to tutorials, but not sample projects

NumPy Package

- Standard Python lists are convenient but not designed for efficient large scale data analysis.
- **NumPy** is the standard Python package for scientific computing:
  - Provides support for multidimensional arrays (i.e. matrices).
  - Implemented closer to hardware for efficiency.
  - Designed for scientific computation, useful for linear algebra and data analysis.
- NumPy can "turn Python into the equivalent of a free and more powerful version of Matlab".

http://www.numpy.org

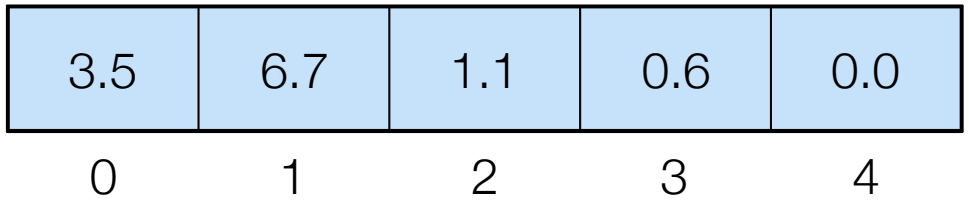
- NumPy included in the Anaconda distribution. General convention to import numpy is using:

```
import numpy as np
```

NumPy Arrays

- The fundamental NumPy data structure is an array: a memory-efficient container that provides fast numerical operations.
- Unlike standard Python lists, a NumPy array can only contain a single type of value (e.g. only floats; only integers etc).
- The simplest type of array is 1-dimensional - i.e. a vector.
- We can access and modify a 1-dimensional array just like a Python list.

```
import numpy as np
d = np.array([3.5, 6.7, 1.1, 0.6, 0.0])
```




```
d[1]
```
```
6.7
```

```
d[:2]
```
```
array([ 3.5,  6.7])
```

```
d + 1
```
```
array([ 4.5, 7.7, 2.1, 1.6, 1. ])
```

```
d[4]
```
```
0.0
```

```
d[0:2]
```
```
array([ 3.5,  6.7])
```

Can easily apply arithmetic operations to all entries in an array.

NumPy Arrays

- An array can have > 1 dimension. A 2-dimensional array can be viewed as a matrix, with rows and columns. It has these properties:
  - Rank of array: Number of dimensions it has.
  - Shape of array: A tuple of integers giving the length of the array in each dimension.
  - Size of array: Total number of entries it contains.

0.4	2.3	4.5
1.5	0.1	1.3
3.2	0.4	3.2
2.7	2.3	6.3
0.1	0.1	0.9

Example:

Rank = 2
i.e. 2 dimensions: rows, columns

Shape = 5x3
i.e. 5 rows x 3 columns

Size = 15
i.e. 5 x 3 = 15 total elements

NumPy Arrays

- A variety of functions are available in NumPy for conveniently creating arrays, or loading them from disk.

```
a = np.zeros((2,3))
```

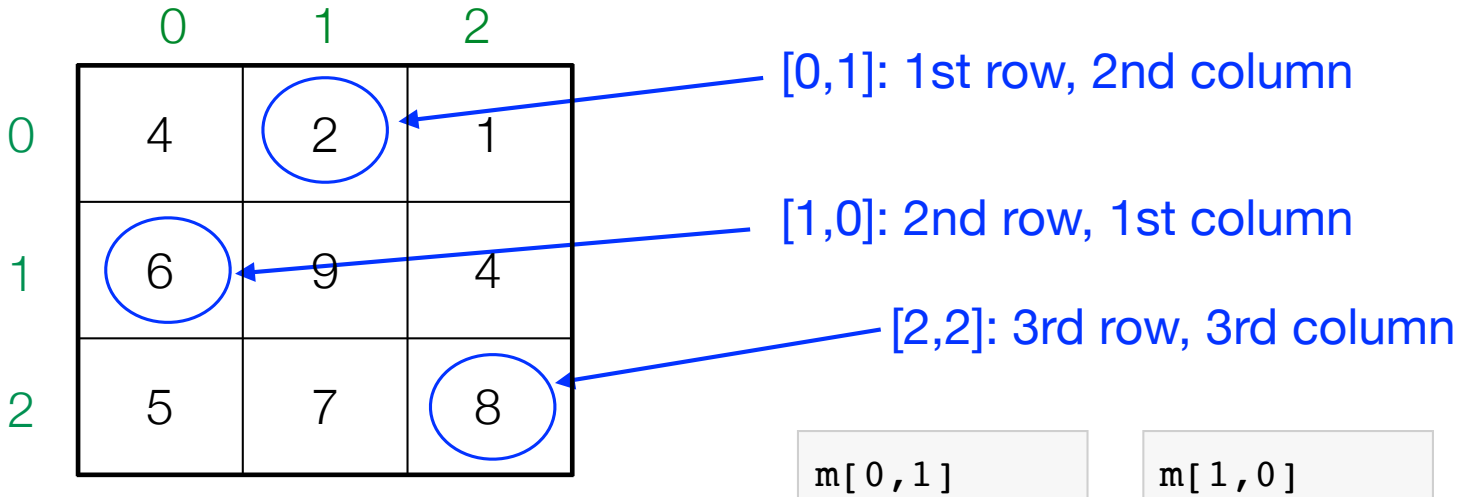
```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

Create an array of zeros. For 2D arrays, specify shape as (rows, columns).

- When working with arrays with more than 1 dimension, use the notation `[i,j]`- i.e. [row, column]

```
np.array([[4,2,1],[6,9,4],[5,7,8]])
```

Create 3x3 array




```
m[0,1]
```

```
2
```

```
m[1,0]
```

```
6
```

```
m[2,2]
```

```
8
```

Pandas Package

- Manipulating and cleaning data can be slow and tedious. Python can (partially) simplify and automate this process.
- **Pandas** is an open source package providing high-performance data structures and analysis tools for Python.
- It provides a Python equivalent of the data analysis and manipulation functionality available in the R programming language. Full details at:

http://pandas.pydata.org

- After installing Anaconda, you will have access to Pandas without needing to install anything else.
- Once the package is installed, we can import it as pd for shorthand.

```
import pandas as pd
```

Pandas Package

- Pandas offers two new data structures that are optimised for data analysis and manipulation.
  1. A Data Frame is a flexible two-dimensional, potentially heterogeneous tabular data structure.
  2. A Series is a data structure for a single column of a Data Frame.

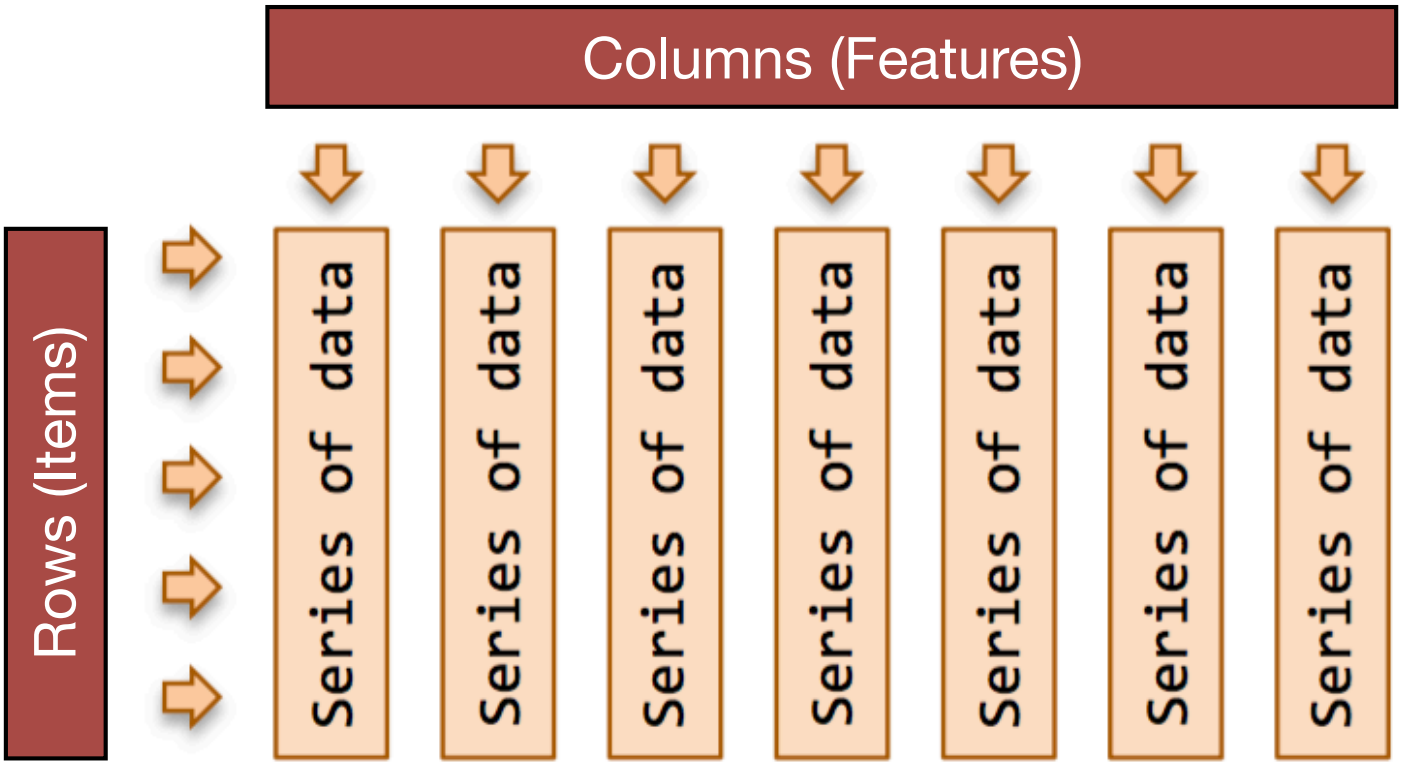
Lastname
Ryan
Lynch
Ward
Grealish

DOB
02/11/1965
03/02/1981
18/12/1972
NaN

- Key distinction of these data structures over basic Python data structures is that they make it easy to associate an index with data - i.e. row and column names.

Pandas Data Frames

- Rich datasets can be represented using a Pandas Data Frame: a 2-dimensional labelled data structure with columns of data that can be of different types.
- Every column in a Data Frame is itself a Pandas Series.



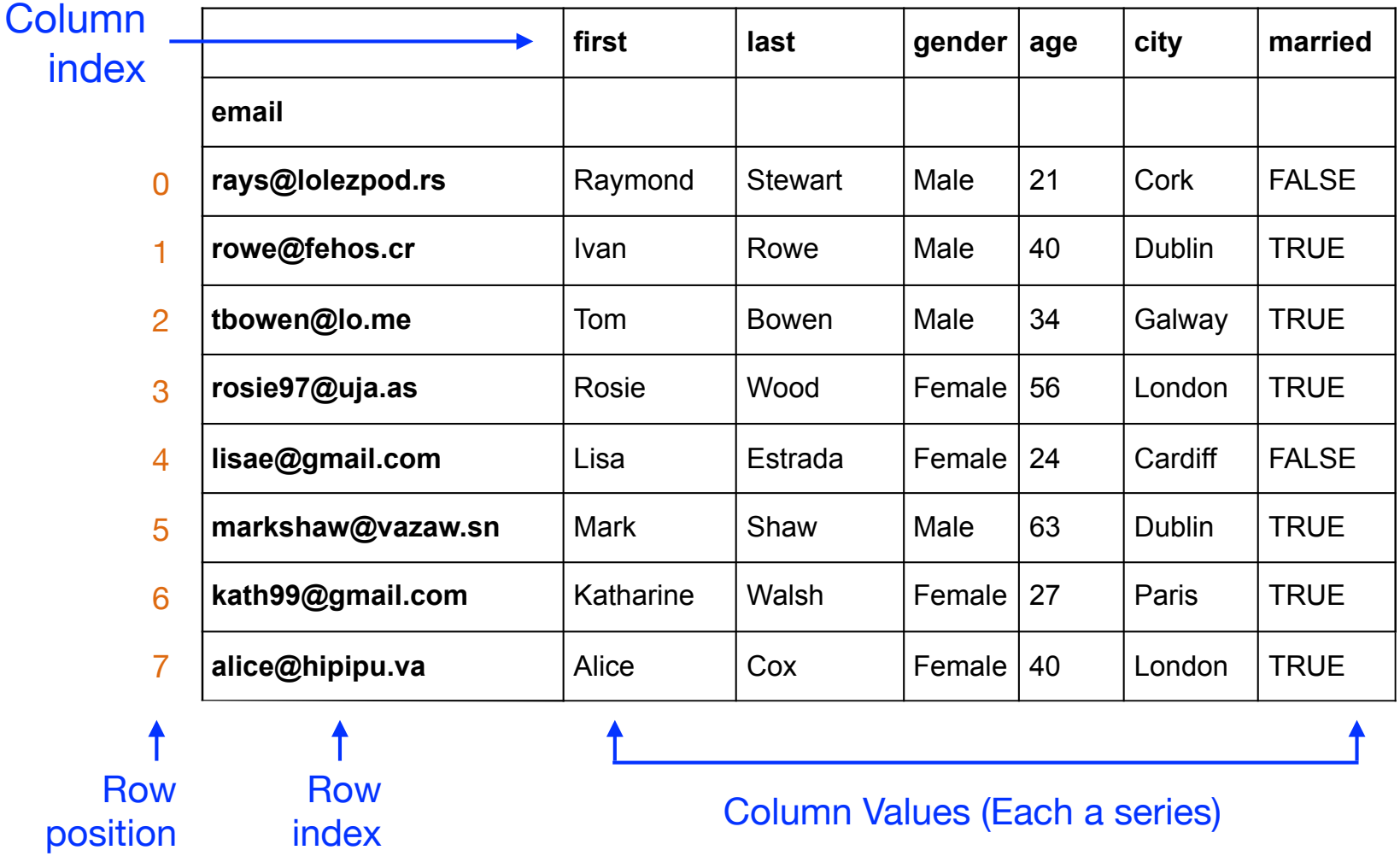
Pandas Data Frames

- The number, type, and meaning of the values stored in each column of a Data Frame depends on the data being analysed.
- **Example:** Data Frame of size 4 rows x 3 columns, with both a row and column index. The column index indicates the feature name, the row index indicates the country name. Both are unique.

Column position →		0	1	2
Column index →		Capital	Population	GDP-BN
0	Ireland	Dublin	4613000	250.8
1	Belgium	Brussels	11190845	524.4
2	France	Paris	66627000	2833.0
3	Spain	Madrid	46439000	1619.0
↑ Row position	↑ Row index			

Pandas Data Frames

- **Example:** Data Frame of size 8 rows x 6 columns. Each row is identified by a unique index (an email address).



	first	last	gender	age	city	married
email						
0 **rays@lolezpod.rs**	Raymond	Stewart	Male	21	Cork	FALSE
1 **rowe@fehos.cr**	Ivan	Rowe	Male	40	Dublin	TRUE
2 **tbowen@lo.me**	Tom	Bowen	Male	34	Galway	TRUE
3 **rosie97@uja.as**	Rosie	Wood	Female	56	London	TRUE
4 **lisae@gmail.com**	Lisa	Estrada	Female	24	Cardiff	FALSE
5 **markshaw@vazaw.sn**	Mark	Shaw	Male	63	Dublin	TRUE
6 **kath99@gmail.com**	Katharine	Walsh	Female	27	Paris	TRUE
7 **alice@hipipu.va**	Alice	Cox	Female	40	London	TRUE

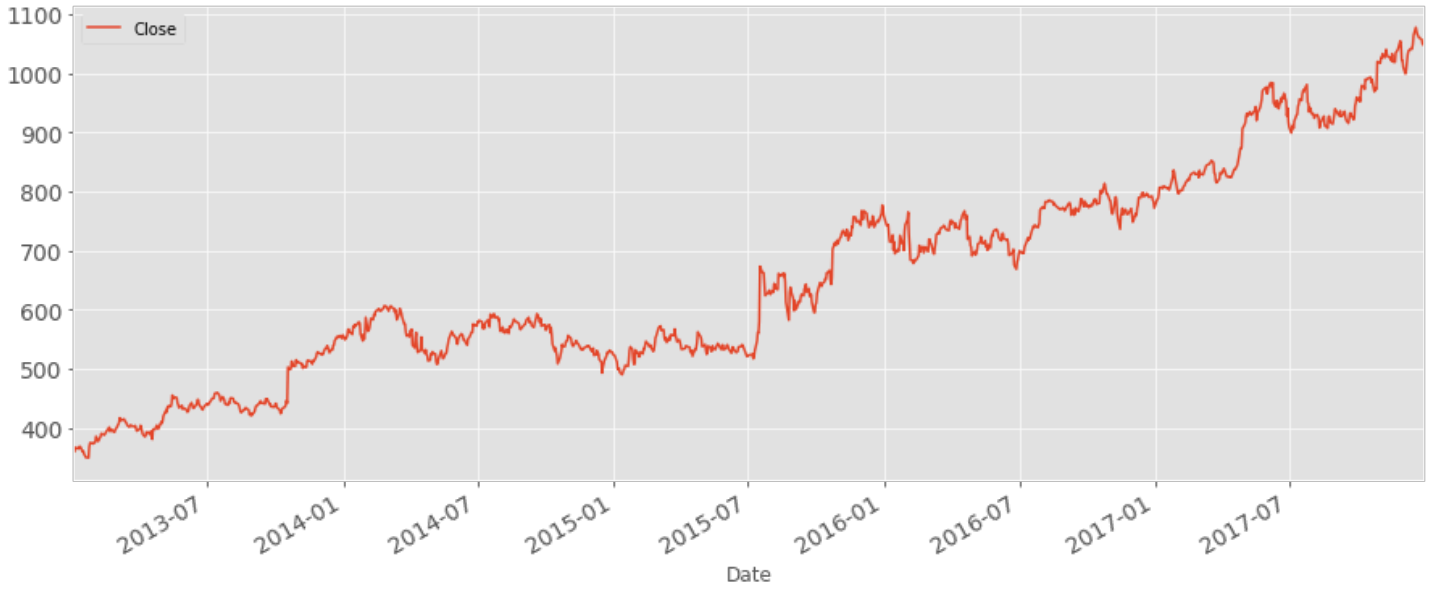
Pandas Time Series

- Time series: A dataset consisting of the values of a function sampled across different points in time.
- Pandas Series and Data Frames can be useful for storing and analysing time series data.

```
Date,Close
2013-01-02,359.288
2013-01-03,359.497
2013-01-04,366.601
2013-01-07,365.001
2013-01-08,364.281
...
```

```
import pandas as pd
ts = pd.read_csv("stock-google-close.csv", index_col="Date", parse_dates=True)
ts.plot(figsize=(15,6),fontsize=14)
```

Date	Close
2013-01-02	359.288
2013-01-03	359.497
2013-01-04	366.601
2013-01-07	365.001
2013-01-08	364.281
...	



Pandas vs NumPy

- NumPy is primarily useful for working with arrays. Highly optimised for efficient operations on numeric arrays.
- Pandas provides higher level data manipulation tools built on top of NumPy arrays, along with more semantics (e.g. indexes).
- Some operations are not as efficient, but Pandas provides additional functionality - e.g dictionary-style access via row or column index to tabular data.
- Since Pandas is built on top of NumPy, we can easily convert values between a NumPy array and a Pandas Series or Data Frame.

Create a 1D array, then construct a Series from it.

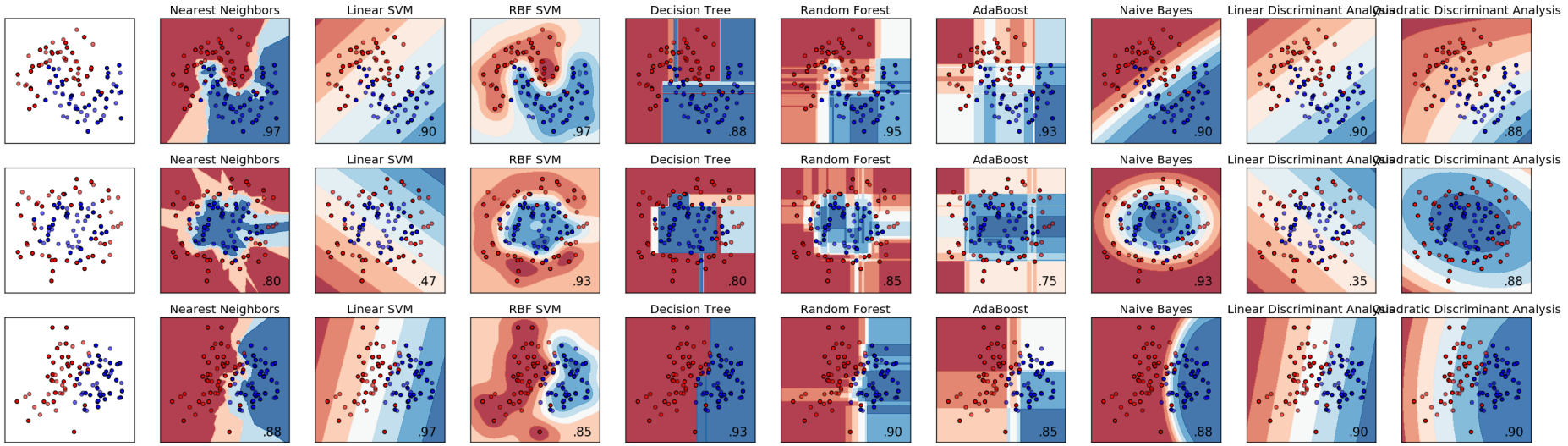
```
import numpy as np
import pandas as pd
a = np.array([0.1,0,1.4,0.04])
s = pd.Series(a)
print(s)
```

```
0    0.10
1    0.00
2    1.40
3    0.04
dtype: float64
```

Scikit-Learn Package

- Scikit-learn is a comprehensive open source Python package for machine learning and data analysis: http://scikit-learn.org
- Anaconda includes Scikit-learn as part of its distribution.
- Scikit-learn algorithm inputs and outputs are usually represented as NumPy arrays, although we can also work with input data as Pandas Data Frames.

```
import sklearn
```



Scikit-Learn - Linear Regression

- Scikit-Learn provides functions to apply **linear regression** to NumPy arrays for simple predictive modelling.
- To build a model for input variable *x* and response variable *y*:

```
from sklearn.linear_model import LinearRegression
```

Create and fit the model based on the training data

```
model = LinearRegression()
model.fit(x, y)
```

Get the intercept coefficient

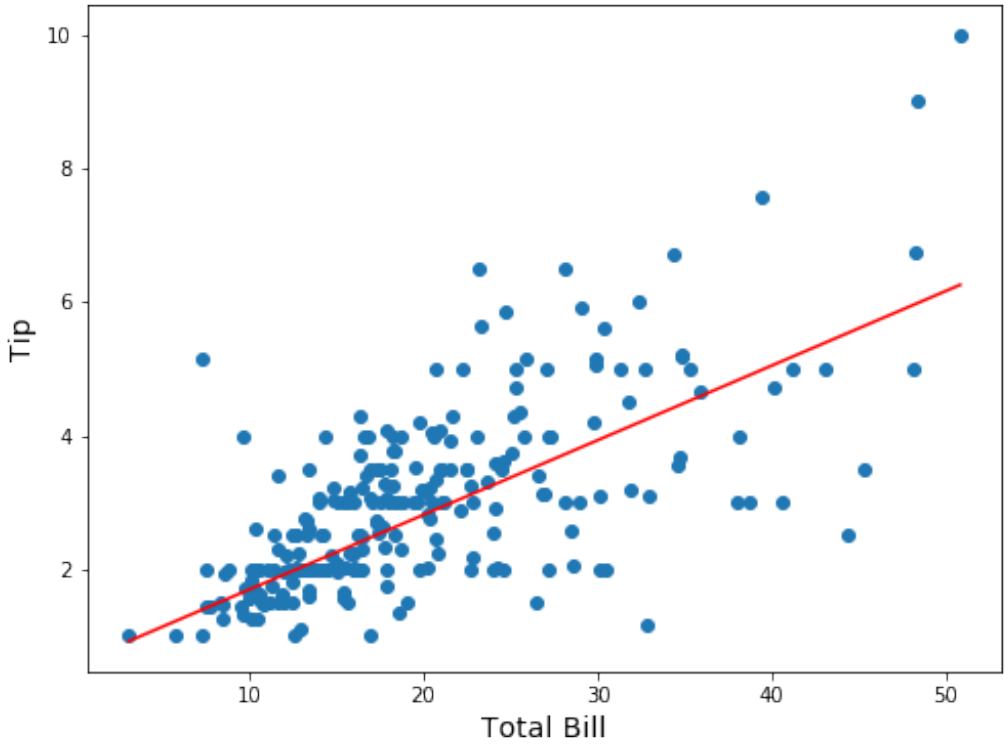
```
model.intercept_
```

```
0.92026961
```

Get the slope coefficient

```
model.coef_[0]
```

```
0.10502452
```



Scikit-Learn - Classification

- Scikit-Learn includes implementations for a range of **classification** algorithms, which use supervision from training data to make predictions of the class label for new input examples.

```
from sklearn.neighbors import KNeighborsClassifier
```

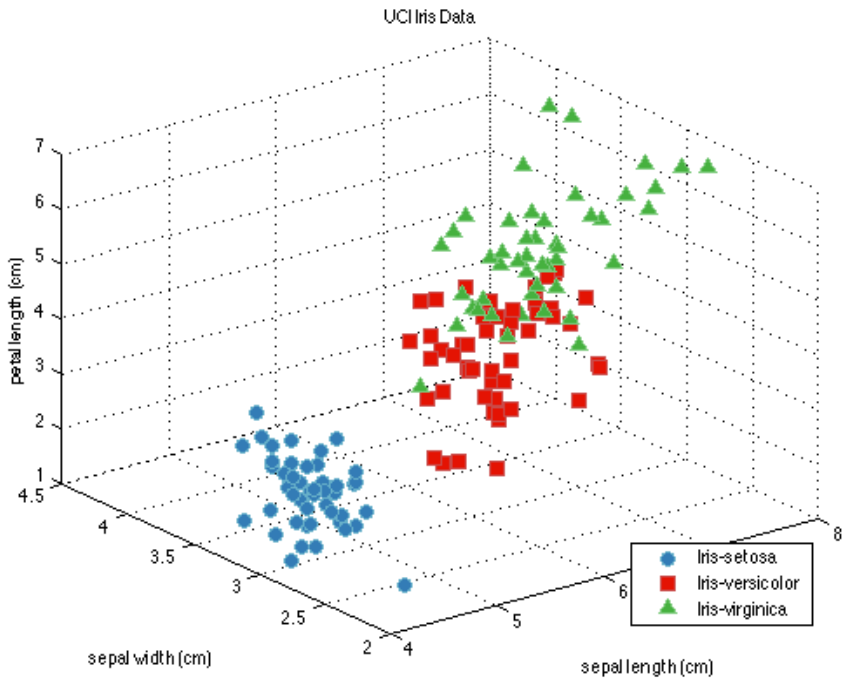
Create and fit the model based on training data

```
model = KNeighborsClassifier(n_neighbors=3)
model.fit(iris.data, iris.target)
```

Make predictions on new unseen data

```
xinput = np.array([[3.0, 5.0, 4.1, 2.0]])
pred_class_number = model.predict(xinput)
print( iris.target_names[pred_class_number] )
```

```
['virginica']
```



Scikit-Learn - Clustering

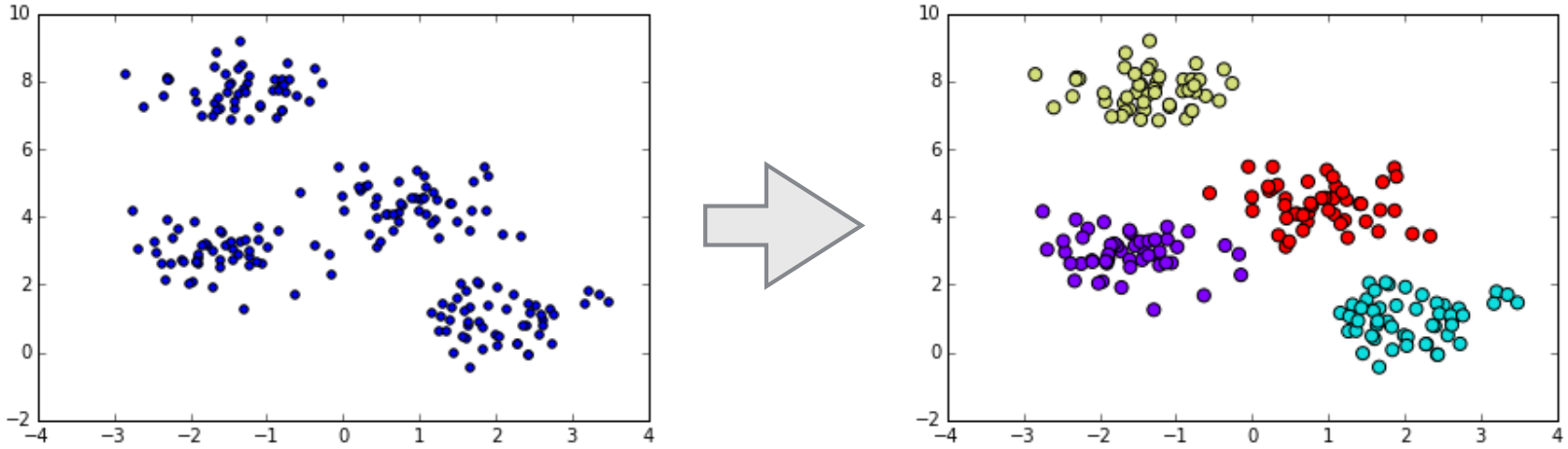
- When no manually labelled inputs are available, we can attempts to identify meaningful patterns by applying **clustering** algorithms to split the data into distinct groups or clusters.
- As with classifiers, we build a model with the `fit()` function. We get the actual assignments from the `labels_` attribute afterwards.

```
from sklearn.cluster import KMeans
model = KMeans(4)
model.fit(X)
```

```
clustering = model.labels_
```

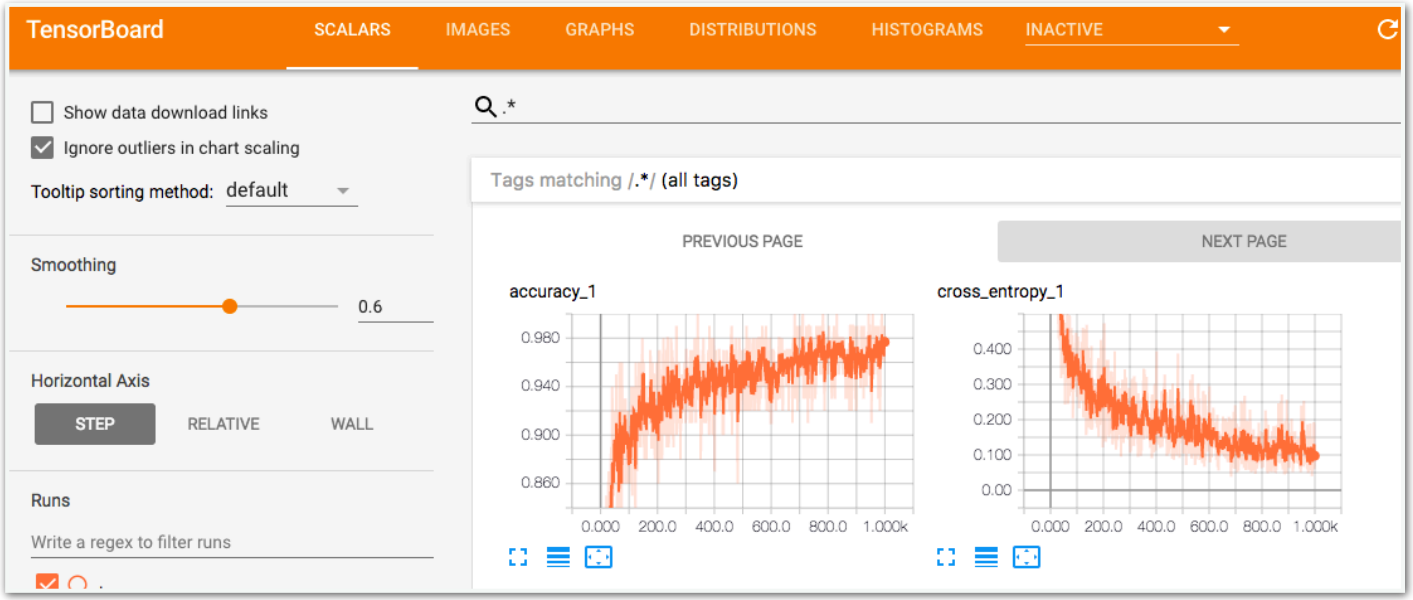
Apply k-means algorithm to find k=4 clusters in the data X

Get the assignments for the items in X afterwards



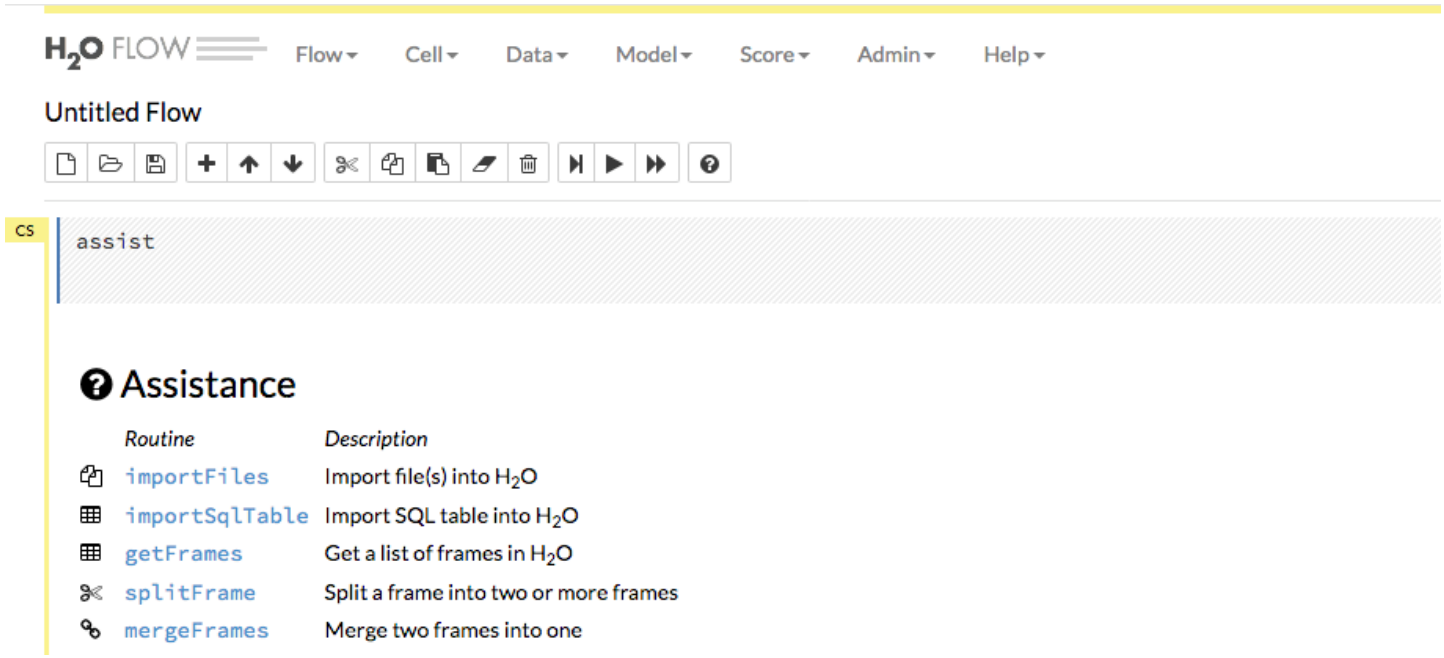
TensorFlow Package

- Open source machine learning library, originally developed by Google: https://www.tensorflow.org
- Fast but flexible. Low-level core (C++/GPU) with a simple Python API to define analysis workflows.
- Strong focus on Deep Learning. Many pre-built functions to ease the task of building complex neural networks.
- Can handle a range of data input types: images, videos, text etc. Also supports visualisation via TensorBoard.



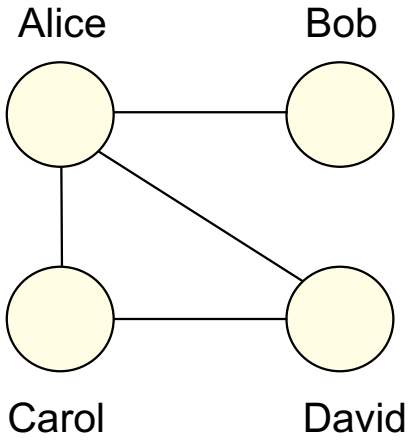
h2o.ai

- Open source machine learning library: https://www.h2o.ai/
- A parallel backend useful for preprocessing, transformations, and machine learning on Hadoop/Yarn, Spark
- Strong focus on AutoML committees.
- Can handle a range of data input types: images, videos, text etc.
- Can be used programmatically (e.g. Jupyter), via other applications (e.g. Tableau) or via h2o's notebook system Flow.



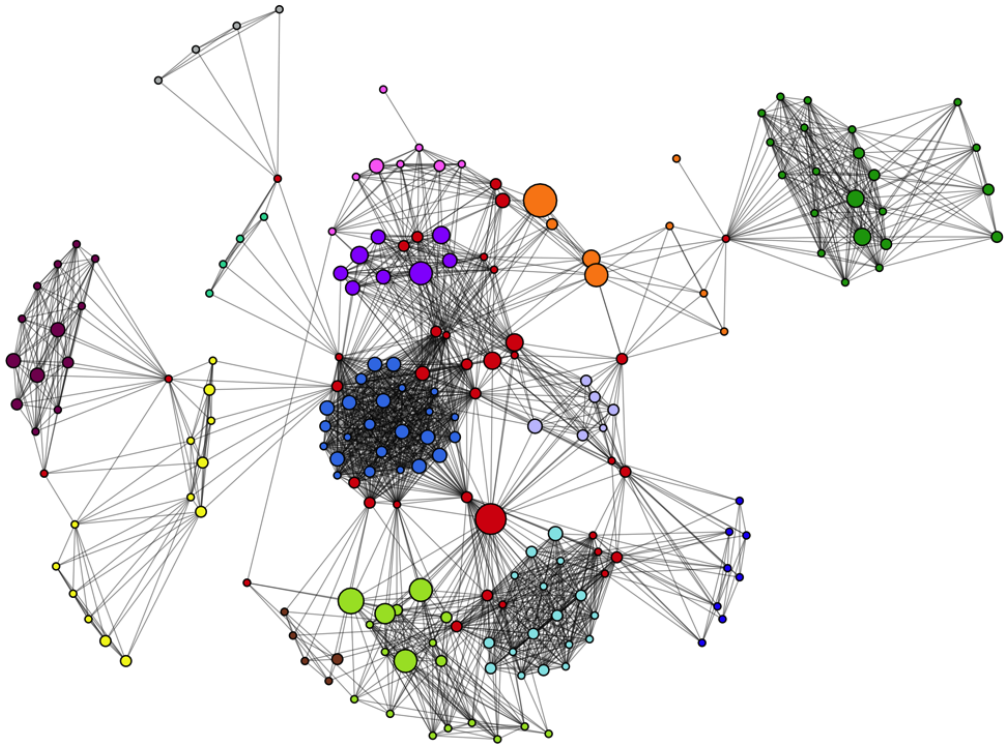
NetworkX Package

- Some data is naturally represented based on relations between pairs of items - e.g. a friendship link between 2 Facebook users.
- The Python **NetworkX** is designed for the creation, manipulation, and study of network data structures.



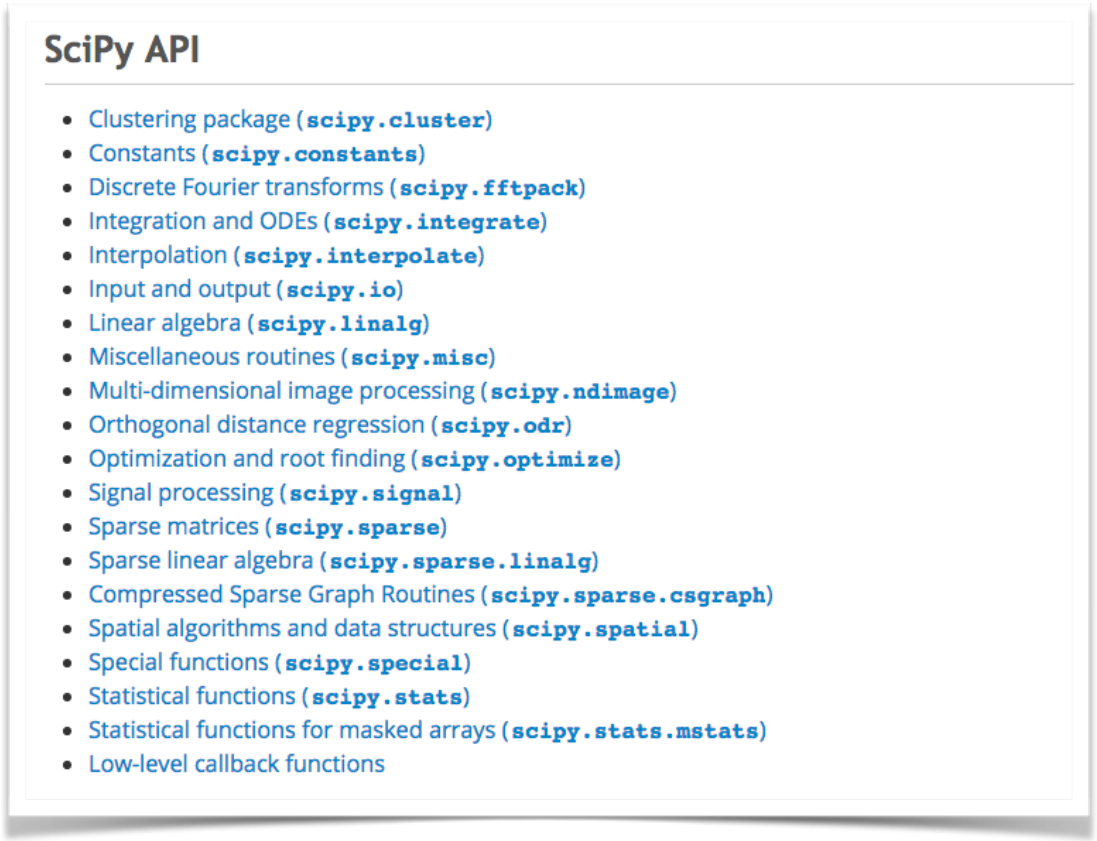
https://networkx.github.io

```
conda install networkx
```



SciPy Package

- Provides an interface to common scientific computing Tasks, including wrappers of various scientific tools.



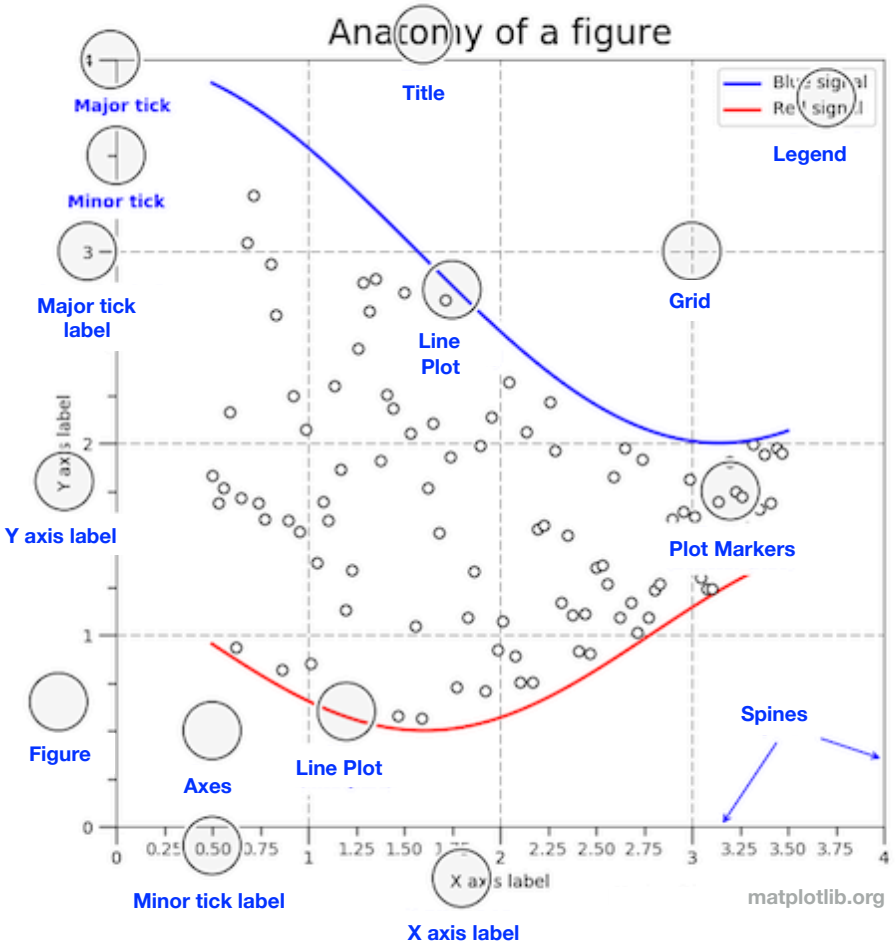
SciPy API

- Clustering package (`scipy.cluster`)
- Constants (`scipy.constants`)
- Discrete Fourier transforms (`scipy.fftpack`)
- Integration and ODEs (`scipy.integrate`)
- Interpolation (`scipy.interpolate`)
- Input and output (`scipy.io`)
- Linear algebra (`scipy.linalg`)
- Miscellaneous routines (`scipy.misc`)
- Multi-dimensional image processing (`scipy.ndimage`)
- Orthogonal distance regression (`scipy.odr`)
- Optimization and root finding (`scipy.optimize`)
- Signal processing (`scipy.signal`)
- Sparse matrices (`scipy.sparse`)
- Sparse linear algebra (`scipy.sparse.linalg`)
- Compressed Sparse Graph Routines (`scipy.sparse.csgraph`)
- Spatial algorithms and data structures (`scipy.spatial`)
- Special functions (`scipy.special`)
- Statistical functions (`scipy.stats`)
- Statistical functions for masked arrays (`scipy.stats.mstats`)
- Low-level callback functions

https://docs.scipy.org/doc/scipy/reference

Visualisation - Matplotlib

- Matplotlib: Standard Python plotting library. Provides a variety of plot types. Included by default in the Anaconda distribution.
- Matplotlib supports the customisation of plot appearance.
- You can control almost every property: figure size, line width, colour and style, axes, axis and grid properties, text and font properties etc.
- See: https://matplotlib.org/api



Visualisation - Matplotlib

- Matplotlib can be used to dynamically generate plot images or to embed plots in IPython notebooks.

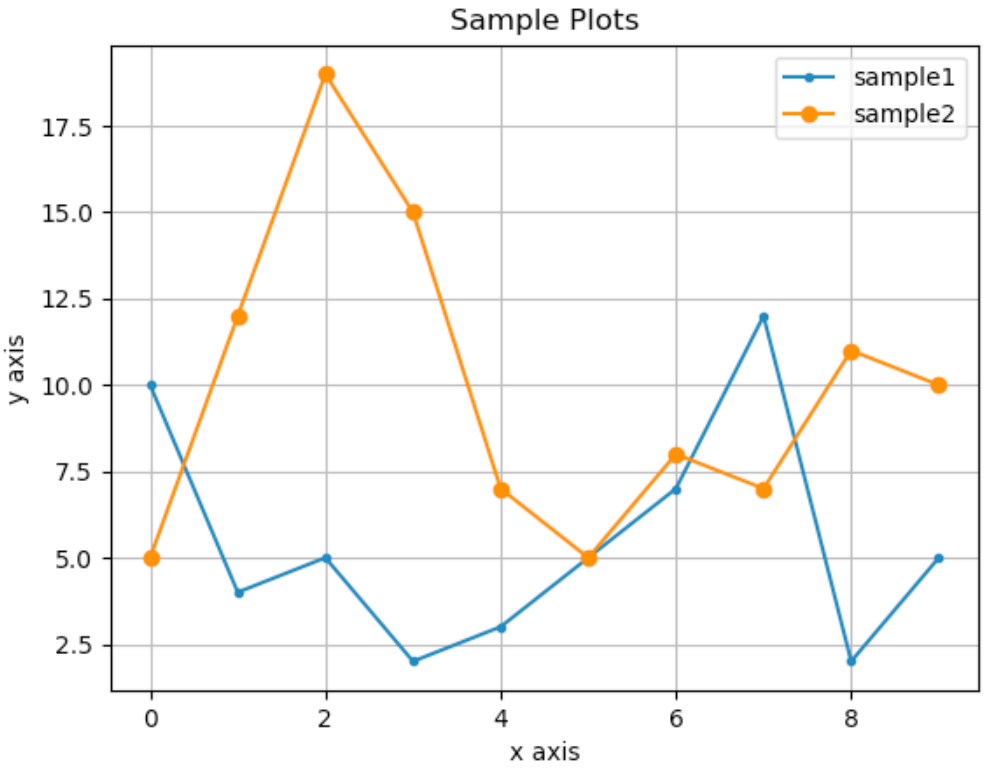
```
import matplotlib.pyplot as plt

x = list(range(0,10))
y1=[10,4,5,2,3,5,7,12,2,5]
y2=[5,12,19,15,7,5,8,7,11,10]

plt.plot(x, y1, '.-', label='sample1')
plt.plot(x, y2, 'o-', label='sample2')

plt.xlabel('x axis')
plt.ylabel('y axis')
plt.title('Sample Plots')
plt.legend(('sample1','sample2'))
plt.grid(True)

plt.savefig("sample.png")
plt.show()
```



Visualisation - Matplotlib

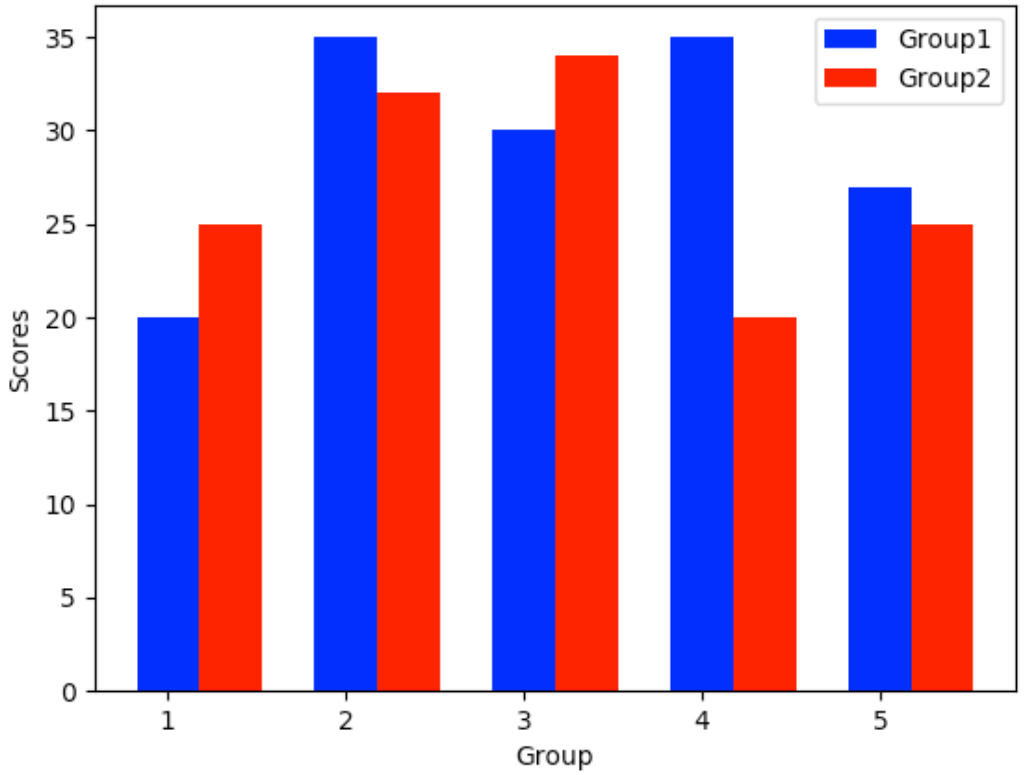
- Matplotlib can be used to dynamically generate plot images or to embed plots in IPython notebooks.

```
import matplotlib.pyplot as plt
import numpy as np

index = np.arange(1,6)
group1 = [20, 35, 30, 35, 27]
group2 = [25, 32, 34, 20, 25]

bar_width = 0.35
error_config = {'ecolor': '0.3'}
plt.bar(index, group1, bar_width,
        color='b', label='Group1')
plt.bar(index + bar_width, group2,
bar_width,
        color='r', label='Group2')

plt.xlabel('Group')
plt.ylabel('Scores')
plt.xticks(index)
plt.legend()
plt.show()
```



Visualisation - Pandas

- Pandas also provides basic visualisation functionality that uses Matplotlib under the hood, but with a simpler interface based around Series and Data Frames using the `plot()` function.

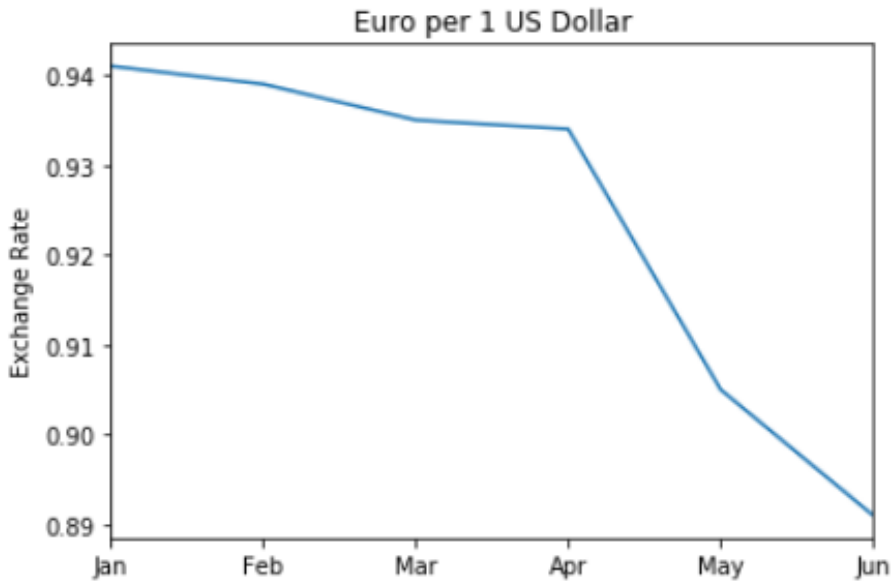
Import Pandas, create a Series

```
import pandas as pd
data = [0.941,0.939,0.935, 0.934,0.905,0.891]
labels = ["Jan","Feb","Mar","Apr","May","Jun"]
exchange = pd.Series( data, labels )
```

Create line chart from the Series using `plot()`, and customise it

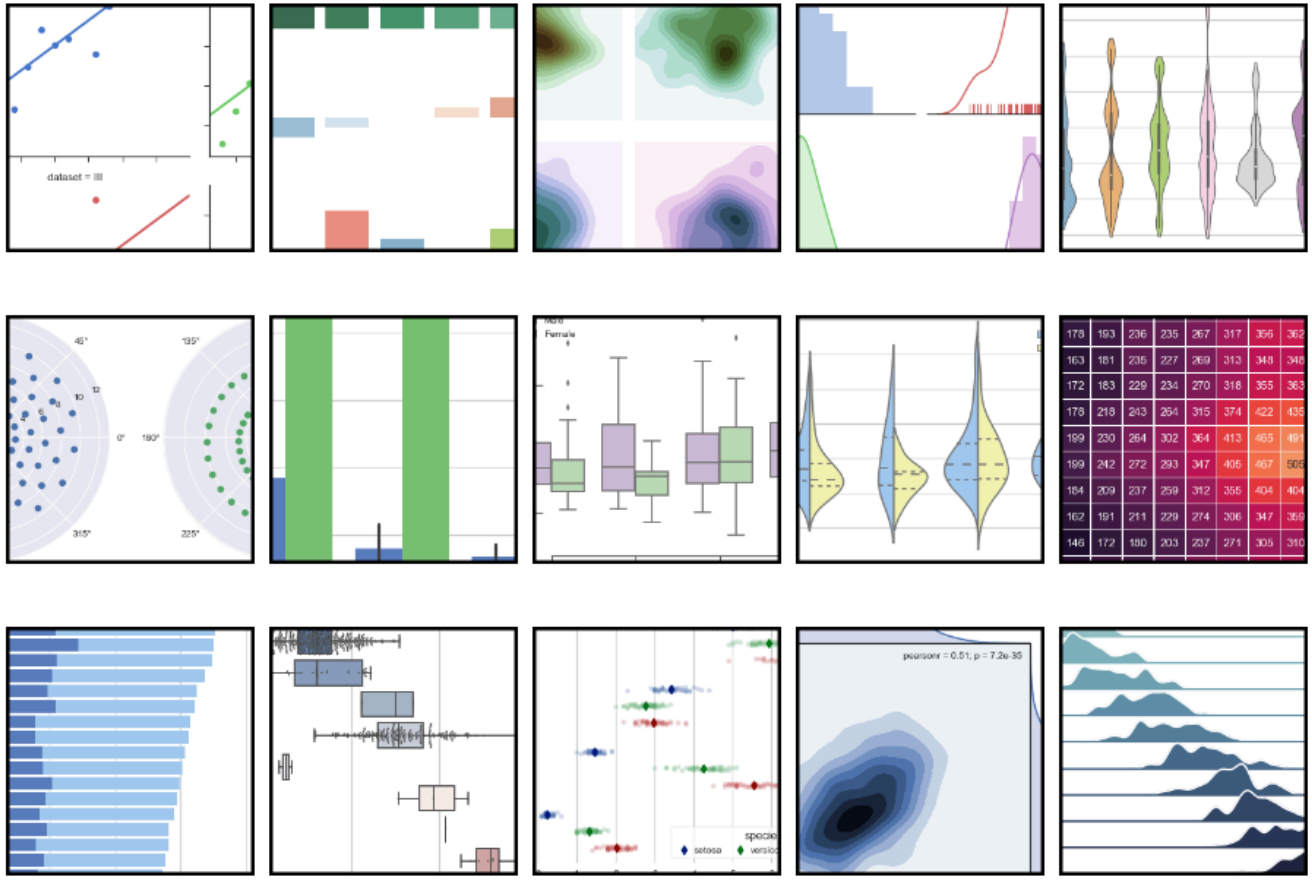
```
p = exchange.plot(title="Euro per 1 US Dollar")
p.set_ylabel("Exchange Rate")
```

The figure will get created and displayed automatically.



Visualisation - Seaborn

- **Seaborn** is a Python statistical visualisation library based on Matplotlib which provides a higher level interface for drawing attractive statistical graphics.

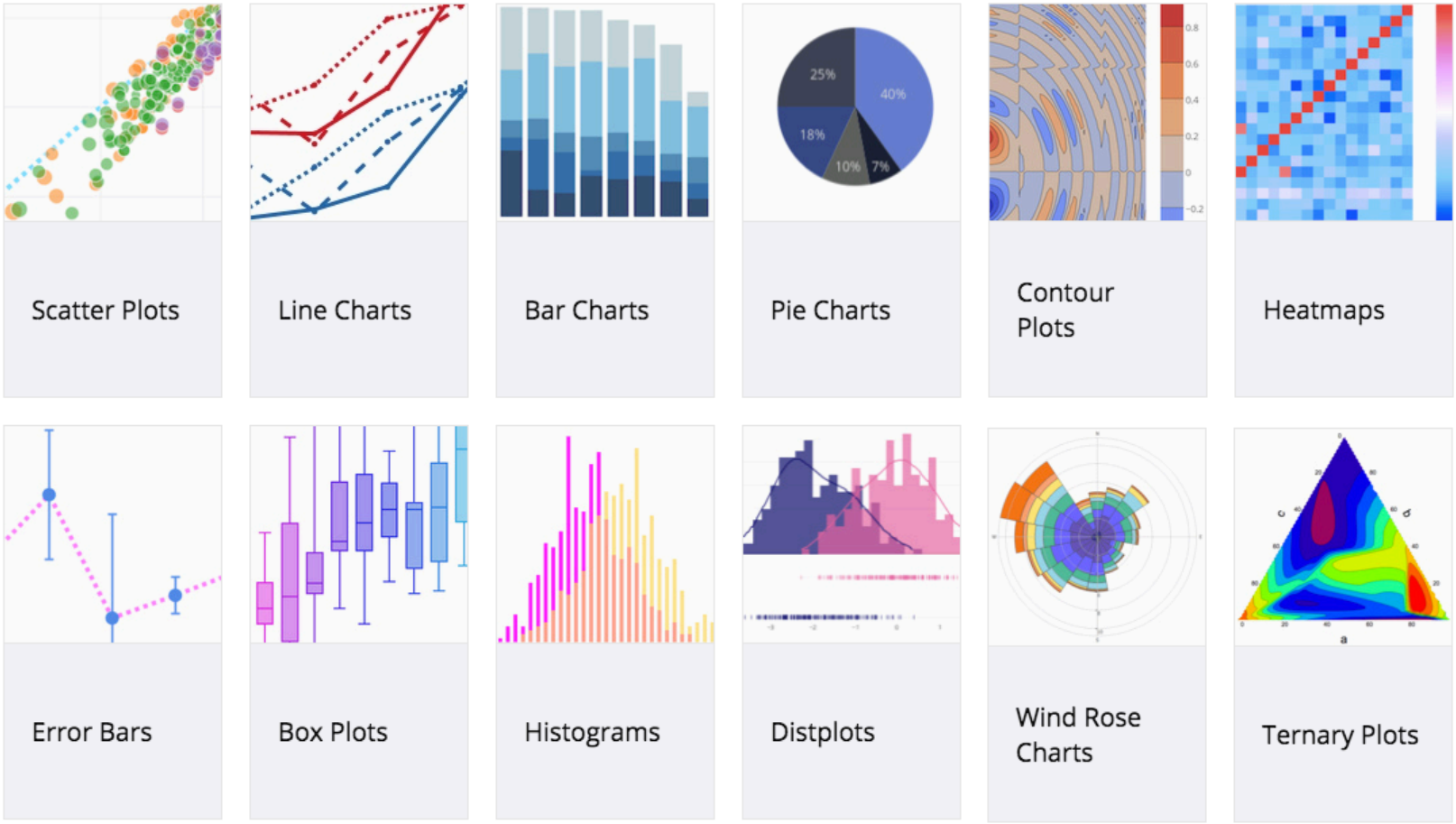


```
conda install seaborn
```

https://seaborn.pydata.org

Visualisation - Plotly

- **Plotly** is an online data visualisation tool, which can be used to make interactive graphs directly from Pandas Data Frames. Plots can also be embedded in IPython notebooks and Flask pages.

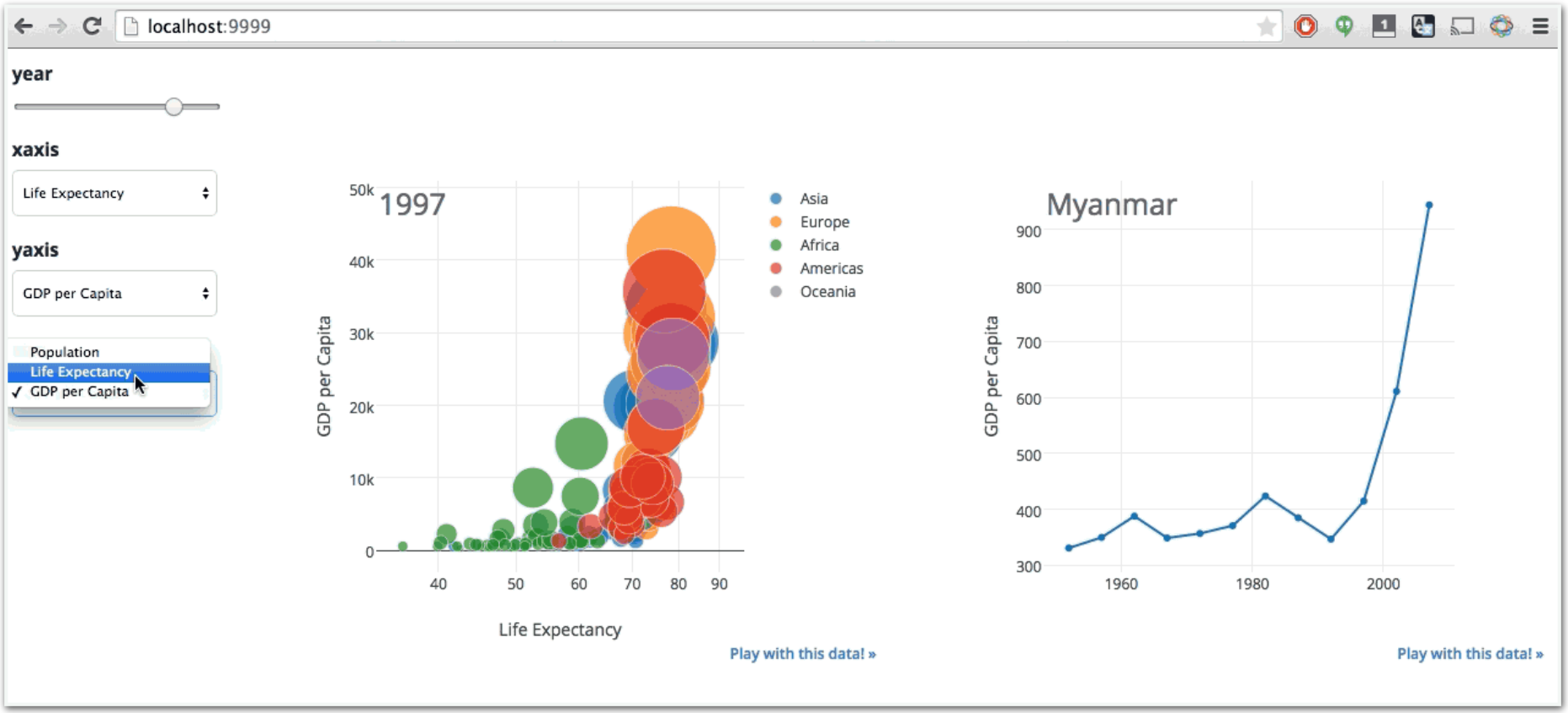



```
conda install plotly
```

https://plot.ly/pandas

Visualisation - Dash Framework

- **Dash** is a Python framework for building analytical web applications. No JavaScript programming is required. It is built on Flask and the Plotly visualisation package.

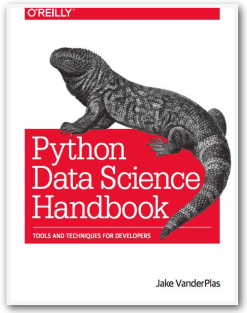


https://plot.ly/products/dash

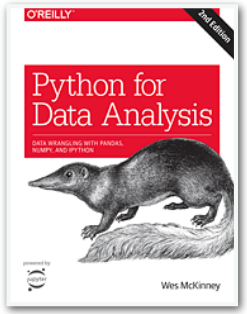
Example Datacamp Courses

- Fundamentals of python: https://www.datacamp.com/tracks/python-programming
- Stats refresher: https://www.datacamp.com/tracks/statistics-fundamentals-with-python
- Intro to Scikit: https://www.datacamp.com/tracks/machine-learning-with-python
- Time Series 101: https://www.datacamp.com/courses/manipulating-time-series-data-in-python
- Data Visualisation: https://www.datacamp.com/tracks/data-visualization-with-python
- Tensorflow 101: https://www.datacamp.com/courses/introduction-to-tensorflow-in-python
- Keras 101: https://www.datacamp.com/courses/deep-learning-in-python
- Recommender Systems with PySpark: https://www.datacamp.com/courses/recommendation-engines-in-pyspark
- Twitter 101: https://www.datacamp.com/courses/analyzing-social-media-data-in-python
- Network Analysis: https://www.datacamp.com/courses/network-analysis-in-python-part-1

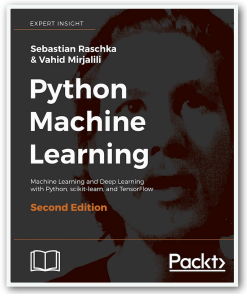
Further Reading



Python Data Science Handbook
Jake VanderPlas
http://shop.oreilly.com/product/0636920034919.do

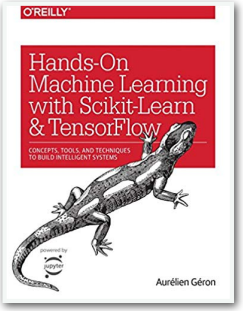


Python for Data Analysis, 2nd Edition
William McKinney
http://shop.oreilly.com/product/0636920050896.do

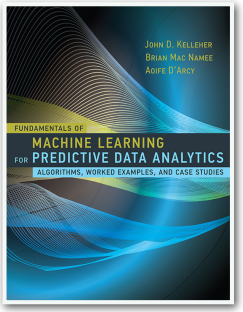


Python Machine Learning, 2nd Edition
Sebastian Raschka
https://github.com/rasbt/python-machine-learning-book-2nd-edition

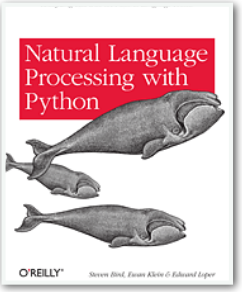
Further Reading



Hands-On Machine Learning with Scikit-Learn and TensorFlow
Aurelien Geron
http://shop.oreilly.com/product/0636920052289.do



Fundamentals of Machine Learning for Predictive Data Analytics
John D. Kelleher, Brian Mac Namee and Aoife D'Arcy
https://mitpress.mit.edu/books/fundamentals-machine-learning-predictive-data-analytics



Natural Language Processing with Python
Steven Bird, Ewan Klein, and Edward Loper
http://www.nltk.org/book/