UNIVERSITY OF PORTSMOUTH

Python for Data Analysis

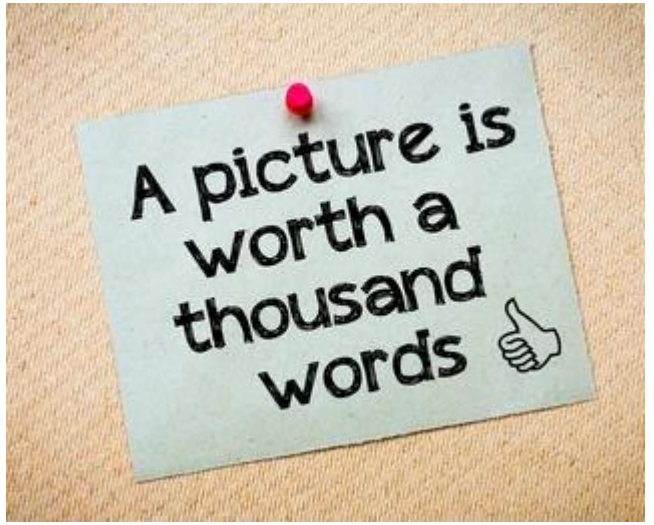
Plotting and Visualisation (Week 8)

Atefeh Khazaei

atefeh.khazaei@port.ac.uk

What we will learn this week?

- A Brief matplotlib API Primer
- Plotting with pandas and seaborn



Plotting and Visualisation

- Making informative visualizations (sometimes called plots) is one of the most important tasks in data analysis.
- It may be a part of the exploratory process (Data understanding phase in CRISP-DM)
  - For example, to help identify outliers or needed data transformations, or as a way of generating ideas for models.
- Python has many add-on libraries for making static or dynamic visualizations.

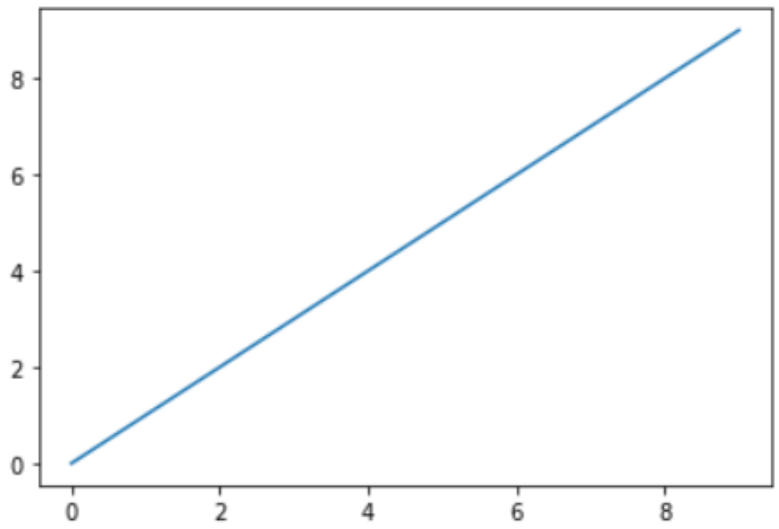
A Brief matplotlib API Primer

- **matplotlib** is a desktop plotting package designed for creating (mostly twodimensional) publication-quality plots.
- The project was started by John Hunter in 2002 to enable a **MATLAB-like plotting** interface in Python.
- matplotlib supports various GUI backends on all operating systems and additionally can export visualizations to all of the common vector and raster graphics formats (PDF, SVG, JPG, PNG, BMP, GIF, etc.).
- Over time, matplotlib has spawned a number of add-on toolkits for data visualization that use matplotlib for their underlying plotting.
  - One of these is seaborn.

A Brief matplotlib API Primer (cont.)
A Brief matplotlib API Primer

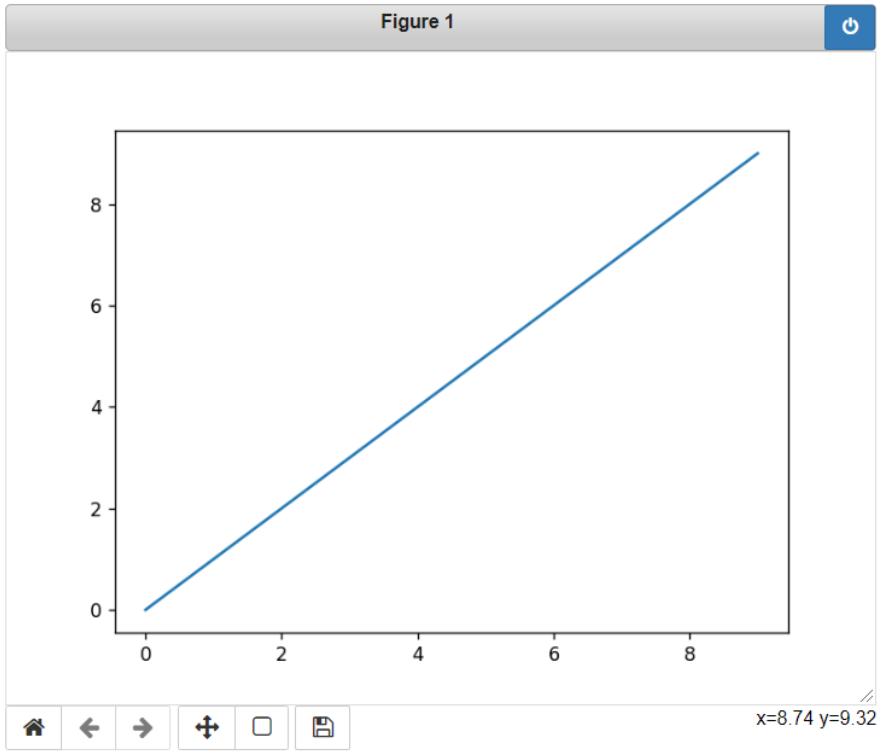
```
import matplotlib.pyplot as plt
import numpy as np
data = np.arange(10)
print(data)
plt.plot(data)
```

```
[0 1 2 3 4 5 6 7 8 9]

[<matplotlib.lines.Line2D at 0x1c5a386baf0>]
```



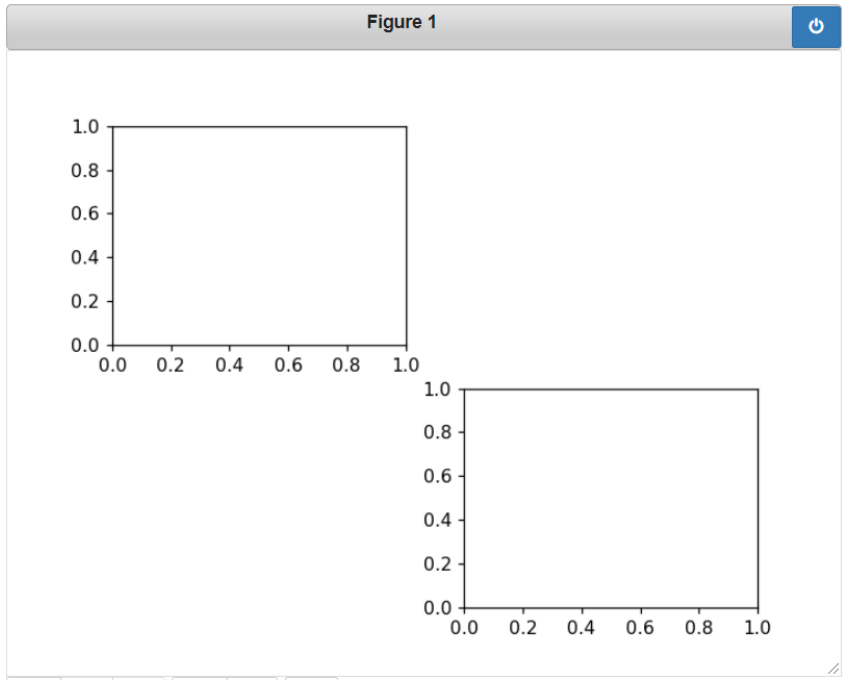
- To use interactive plotting in the Jupyter notebook:

```
%matplotlib notebook
plt.plot(data)
```



A Brief matplotlib API Primer (cont.)
Figures and Subplots

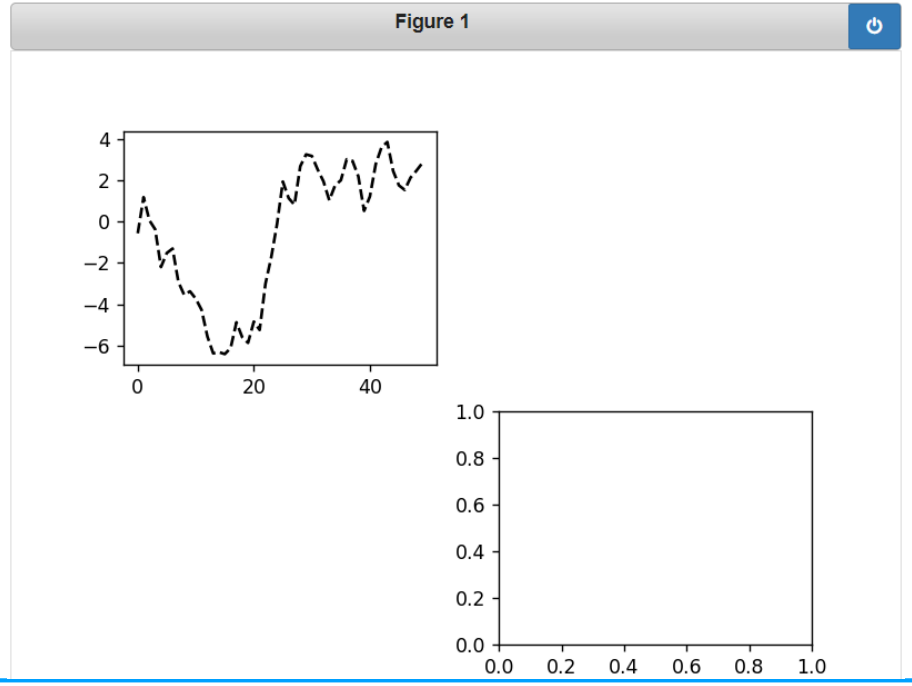
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax4 = fig.add_subplot(2, 2, 4)
```



- Second line means that the figure should be 2 × 2 (so up to four plots in total), and we're selecting the first of four subplots (numbered from 1).

A Brief matplotlib API Primer (cont.)
Figures and Subplots

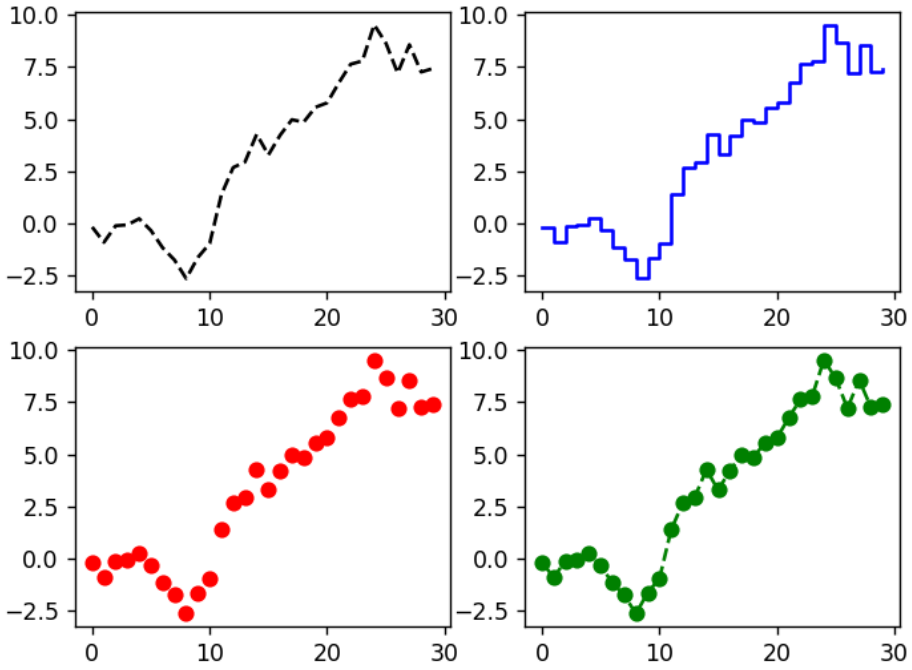
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
plt.plot(np.random.randn(50).cumsum(), 'k--')
ax4 = fig.add_subplot(2, 2, 4)
```



- The 'k--' is a style option instructing matplotlib to plot a black dashed line.

A Brief matplotlib API Primer (cont.)
Colors, Markers, and Line Styles

```
from numpy.random import randn
import matplotlib.pyplot as plt

data = randn(30).cumsum()
fig = plt.figure()

ax1 = fig.add_subplot(2, 2, 1)
plt.plot(data, 'k--')

ax2 = fig.add_subplot(2, 2, 2)
plt.plot(data, 'b-', drawstyle='steps-post', label='steps-post')

ax3 = fig.add_subplot(2, 2, 3)
plt.plot(data, 'ro')

ax4 = fig.add_subplot(2, 2, 4)
plt.plot(data, color='g', linestyle='dashed', marker='o')
```



Plotting with pandas and seaborn

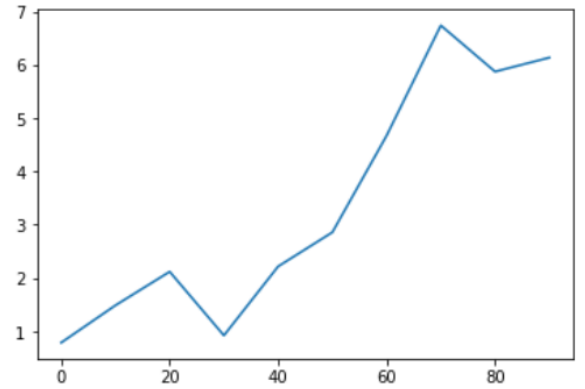
- In **matplotlib**, you assemble a plot from its base components: the data display (i.e., the type of plot: line, bar, box, scatter, contour, etc.), legend, title, tick labels, and other annotations.
- **Pandas** has built-in methods that simplify creating visualizations from DataFrame and Series objects.
- Another library is **seaborn**, a statistical graphics library that simplifies creating many common visualization types.

Plotting with pandas and seaborn (cont.)
Line Plots

- Series and DataFrame each have a plot attribute for making some basic plot types.
- By default, plot() makes line plots.
- Series:

```
import pandas as pd
import numpy as np
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
s.plot()
```

```
<AxesSubplot:>
```



Plotting with pandas and seaborn (cont.)
Line Plots

Series.plot method arguments

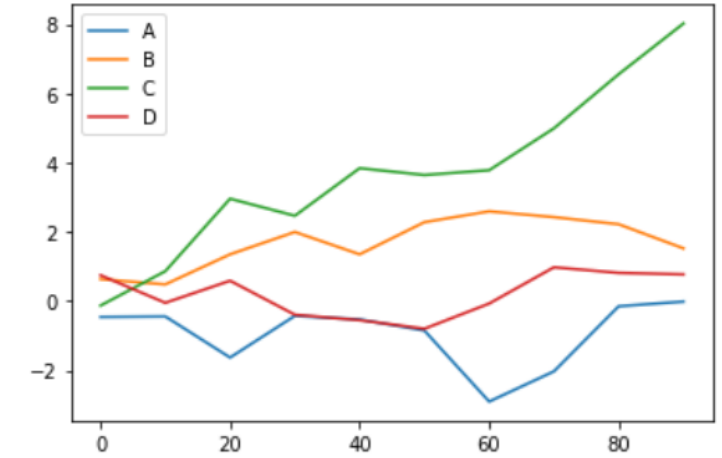
Argument	Description
`label`	Label for plot legend
`ax`	matplotlib subplot object to plot on; if nothing passed, uses active matplotlib subplot
`style`	Style string, like `'ko--'`, to be passed to matplotlib
`alpha`	The plot fill opacity (from 0 to 1)
`kind`	Can be `'area'`, `'bar'`, `'barh'`, `'density'`, `'hist'`, `'kde'`, `'line'`, `'pie'`
`logy`	Use logarithmic scaling on the y-axis
`use_index`	Use the object index for tick labels
`rot`	Rotation of tick labels (0 through 360)
`xticks`	Values to use for x-axis ticks
`yticks`	Values to use for y-axis ticks
`xlim`	x-axis limits (e.g., `[0, 10]`)
`ylim`	y-axis limits
`grid`	Display axis grid (on by default)

Plotting with pandas and seaborn (cont.)
Line Plots

- DataFrame's plot method plots each of its columns as a different line on the same subplot, creating a legend automatically.
- The plot attribute contains a "family" of methods for different plot types.
  - E.g. df.plot() is equivalent to df.plot.line().

```
df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                  columns=['A', 'B', 'C', 'D'],
                  index=np.arange(0, 100, 10))
df.plot()
```

```
<AxesSubplot:>
```



Plotting with pandas and seaborn (cont.)
Line Plots

- DataFrame has a number of options allowing some flexibility with how the columns are handled;
  - For example, whether to plot them all on the same subplot or to create separate subplots.

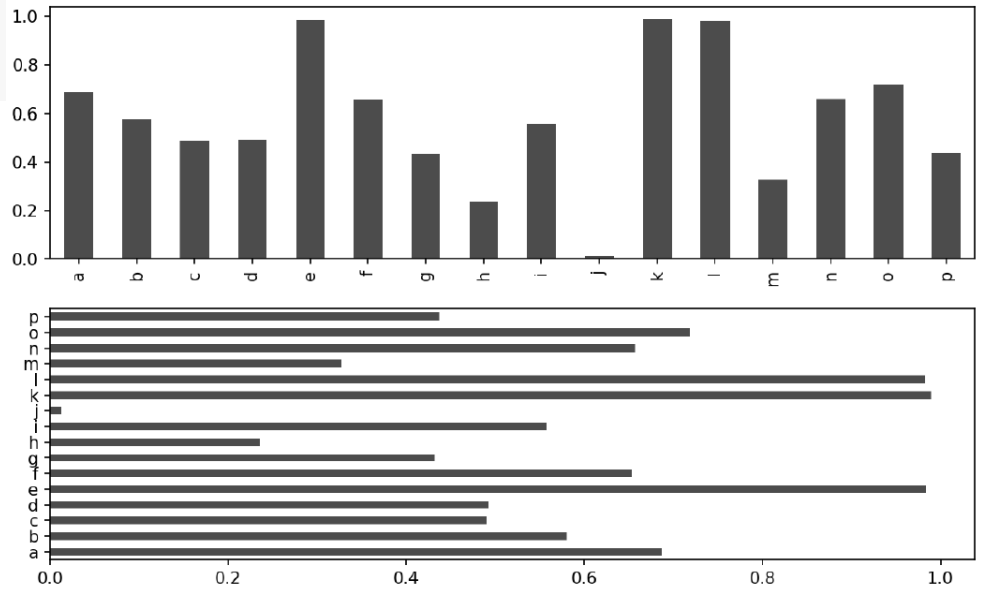
DataFrame-specific plot arguments

Argument	Description
`subplots`	Plot each DataFrame column in a separate subplot
`sharex`	If `subplots=True`, share the same x-axis, linking ticks and limits
`sharey`	If `subplots=True`, share the same y-axis
`figsize`	Size of figure to create as tuple
`title`	Plot title as string
`legend`	Add a subplot legend (`True` by default)
`sort_columns`	Plot columns in alphabetical order; by default uses existing column order

Plotting with pandas and seaborn (cont.)
Bar Plots

- The plot.bar() and plot.barh() make vertical and horizontal bar plots, respectively.

```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.7)
data.plot.barh(ax=axes[1], color='k', alpha=0.7)
```



- color='k' and alpha=0.7 set the color of the plots to black and use partial transparency

Plotting with pandas and seaborn (cont.)
Bar Plots

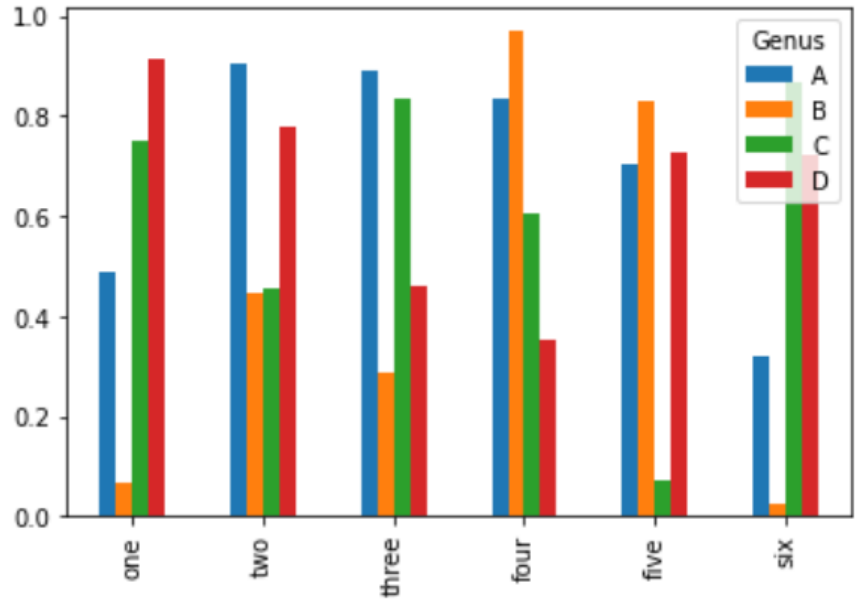
- With a DataFrame, bar plots group the values in each row together in a group in bars, side by side, for each value.

```
df
```

Genus	A	B	C	D
one	0.268421	0.508130	0.359521	0.233919
two	0.574972	0.403399	0.620198	0.364512
three	0.420616	0.099504	0.792812	0.003439
four	0.051271	0.967978	0.297355	0.404143
five	0.266953	0.974317	0.204322	0.044546
six	0.742583	0.930324	0.381838	0.678193

```
df.plot.bar()
```

```
<AxesSubplot:>
```

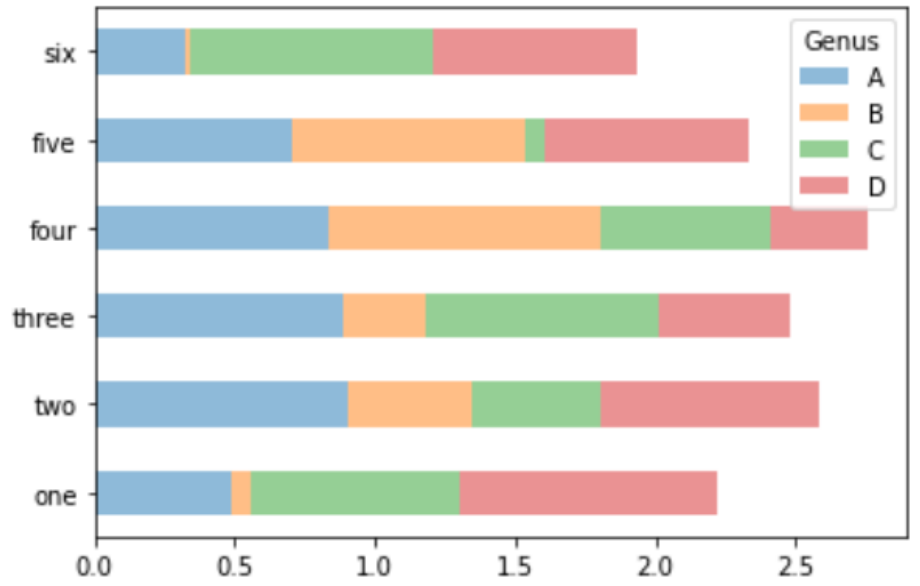


Plotting with pandas and seaborn (cont.)
Bar Plots

- We create stacked bar plots from a DataFrame by passing stacked=True, resulting in the value in each row being stacked together.

```
df.plot.barh(stacked=True, alpha=0.5)
```

```
<AxesSubplot:>
```



Plotting with pandas and seaborn (cont.)
Histograms and Density Plots

- A histogram is a kind of bar plot that gives a discretized display of value frequency.
- The data points are split into discrete, evenly spaced bins, and the number of data points in each bin is plotted.

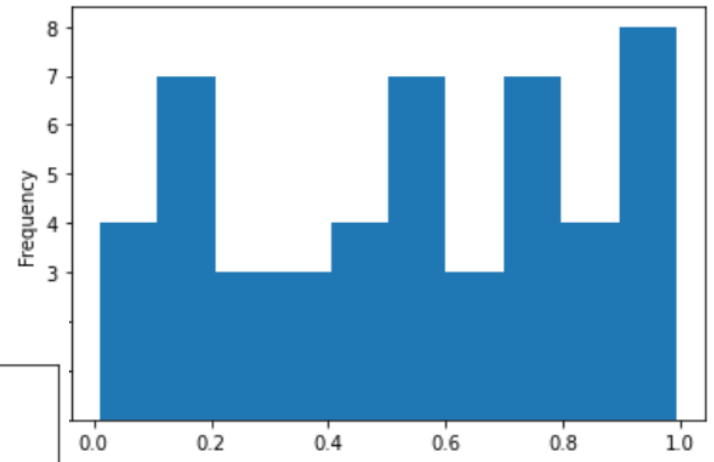
Plotting with pandas and seaborn (cont.)
Histograms and Density Plots

```
df = pd.DataFrame(np.random.rand(50, 4),
     columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df.head()
```

Genus	A	B	C	D
0	0.437273	0.824941	0.420519	0.074335
1	0.051603	0.277836	0.372194	0.710426
2	0.660555	0.215963	0.198941	0.526249
3	0.523742	0.713590	0.552567	0.838830
4	0.582686	0.997710	0.893496	0.590996

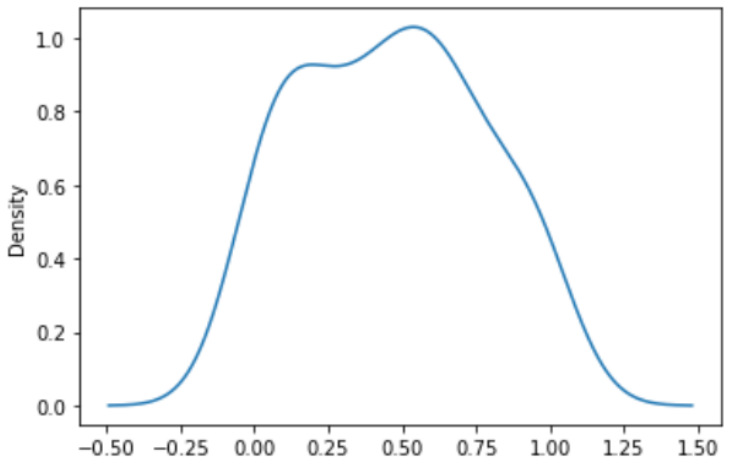
```
df['A'].plot.hist(bins=10)
```

```
<AxesSubplot:ylabel='Frequency'>
```




```
df['A'].plot.density()
```

```
<AxesSubplot:ylabel='Density'>
```

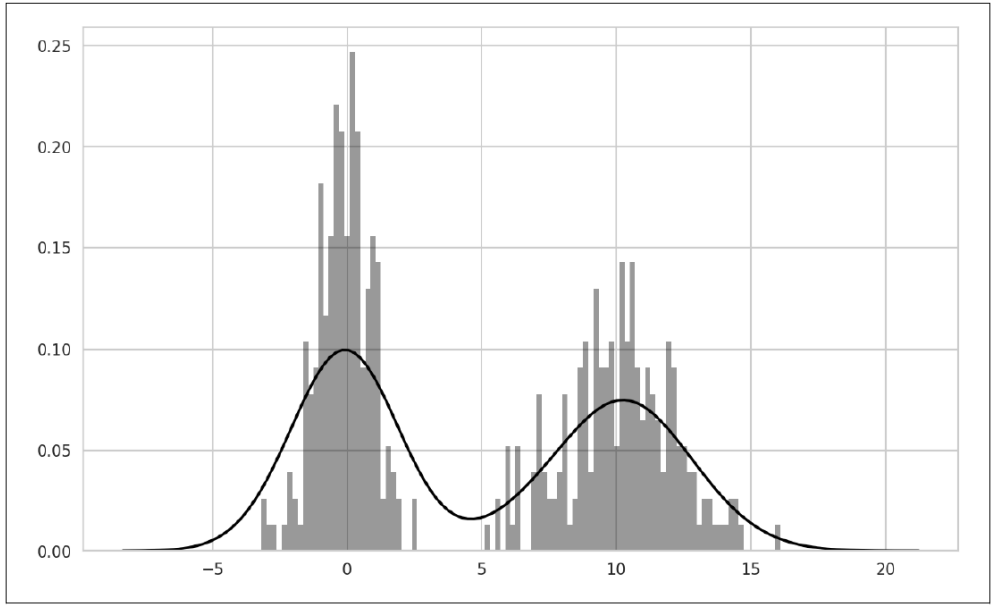


- **Density plot** is a representation of the distribution of a numeric variable.

Plotting with pandas and seaborn (cont.)
Histograms and Density Plots

- **Seaborn** makes histograms and density plots even easier through its **distplot** method, which can plot both a histogram and a continuous density estimate simultaneously.
  - Example, consider a bimodal distribution consisting of draws from two different standard normal distributions.

```
import seaborn as sns
comp1 = np.random.normal(0, 1, size=200)
comp2 = np.random.normal(10, 2, size=200)
values = pd.Series(np.concatenate([comp1, comp2]))
sns.distplot(values, bins=100, color='k')
```



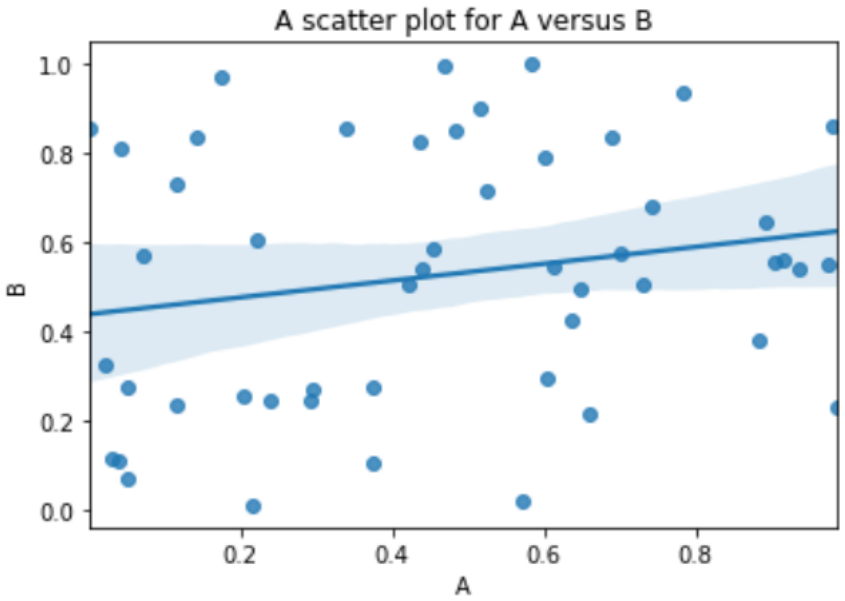
Plotting with pandas and seaborn (cont.)
Scatter or Point Plots

- **Seaborn's regplot** method, which makes a scatter plot and fits a linear regression line.

```
sns.regplot('A', 'B', data=df)
plt.title('A scatter plot for %s versus %s' % ('A', 'B'))
```

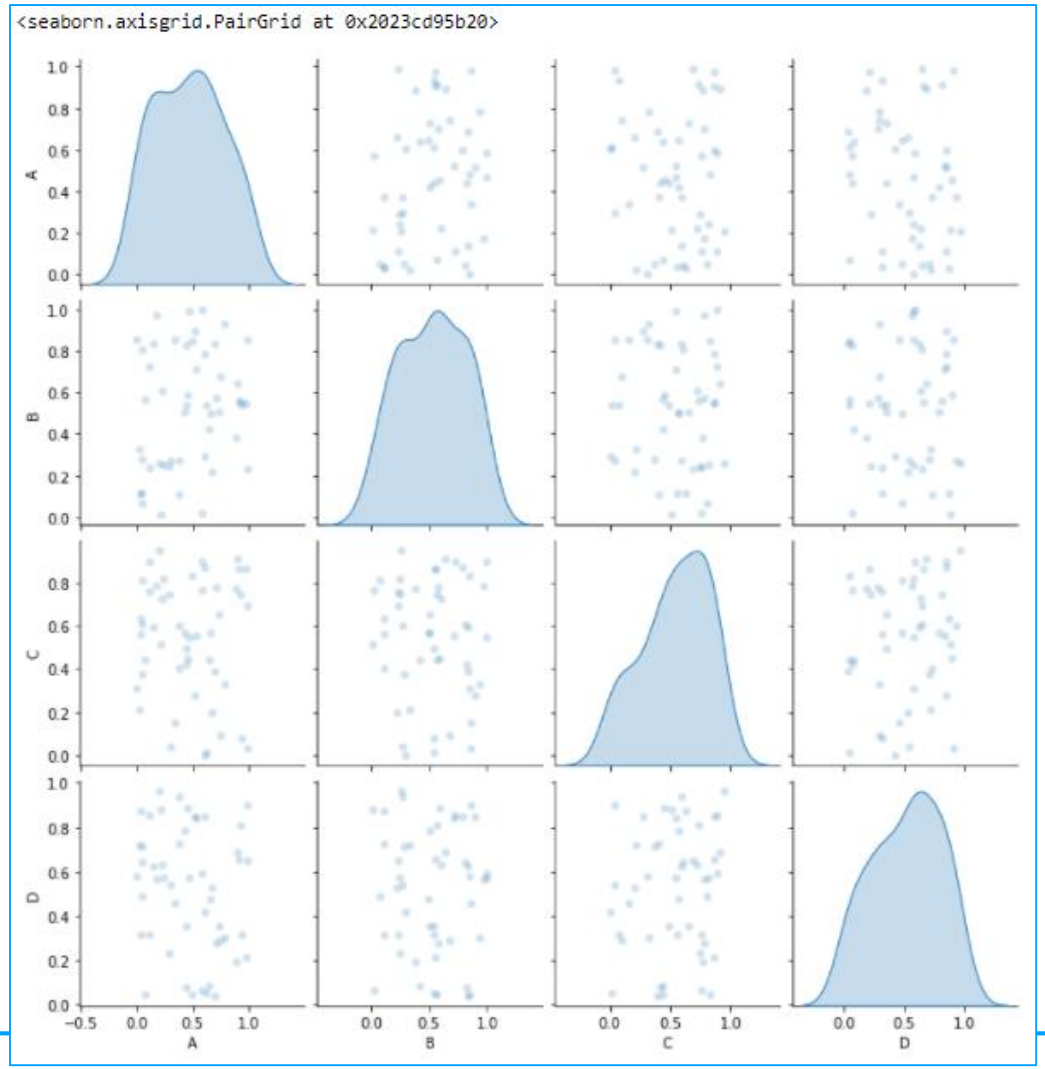
```
df = pd.DataFrame(np.random.rand(50, 4),
     columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df.head()
```

Genus	A	B	C	D
0	0.437273	0.824941	0.420519	0.074335
1	0.051603	0.277836	0.372194	0.710426
2	0.660555	0.215963	0.198941	0.526249
3	0.523742	0.713590	0.552567	0.838830
4	0.582686	0.997710	0.893496	0.590996



Plotting with pandas and seaborn (cont.)
Scatter or Point Plots

```
sns.pairplot(df, diag_kind='kde', plot_kws={'alpha': 0.2})
```

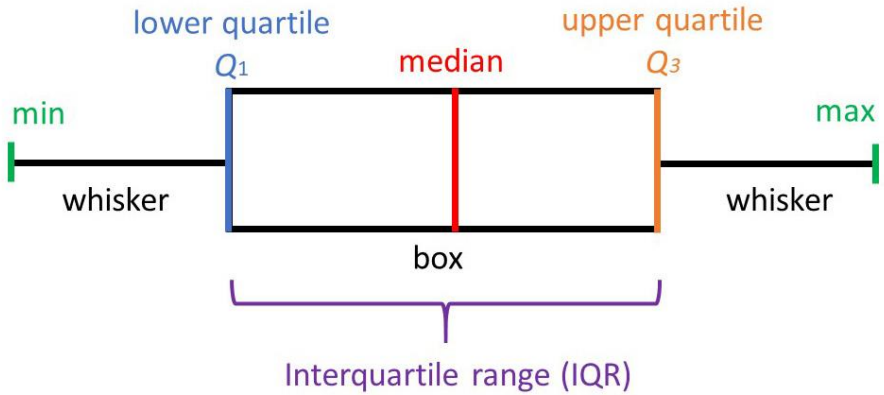



```
df = pd.DataFrame(np.random.rand(50, 4),
     columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df.head()
```

Genus	A	B	C	D
0	0.437273	0.824941	0.420519	0.074335
1	0.051603	0.277836	0.372194	0.710426
2	0.660555	0.215963	0.198941	0.526249
3	0.523742	0.713590	0.552567	0.838830
4	0.582686	0.997710	0.893496	0.590996

Plotting with pandas and seaborn (cont.)
Box Plot

- Box plot is a method for depicting groups of numerical data through their quartiles.
- Boxplot summarizes a column of DataFrame using **25th, 50th, and 75th percentiles**.
- These percentiles are known as the **lower quartile, median and upper quartile**.
- Box plots may also have lines extending from the boxes (**whiskers**) indicating variability outside the upper and lower quartiles.
- Box plots can be used to detect **outliers**.
- Outliers may be plotted as individual points.



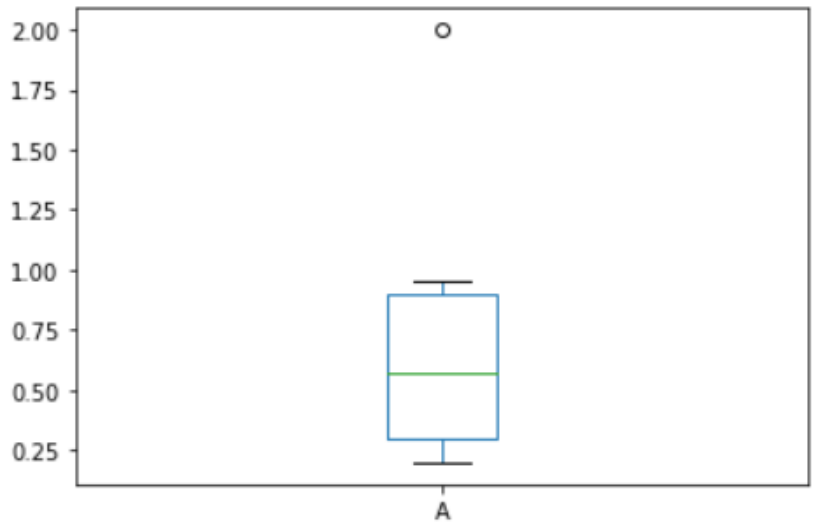
Plotting with pandas and seaborn (cont.)
Box Plot

```
df
```

	A	B
0	0.069013	0.648209
1	2.000000	0.323115
2	0.383175	0.090842
3	0.972882	1.500000
4	0.444677	0.855002
5	0.424524	-1.000000

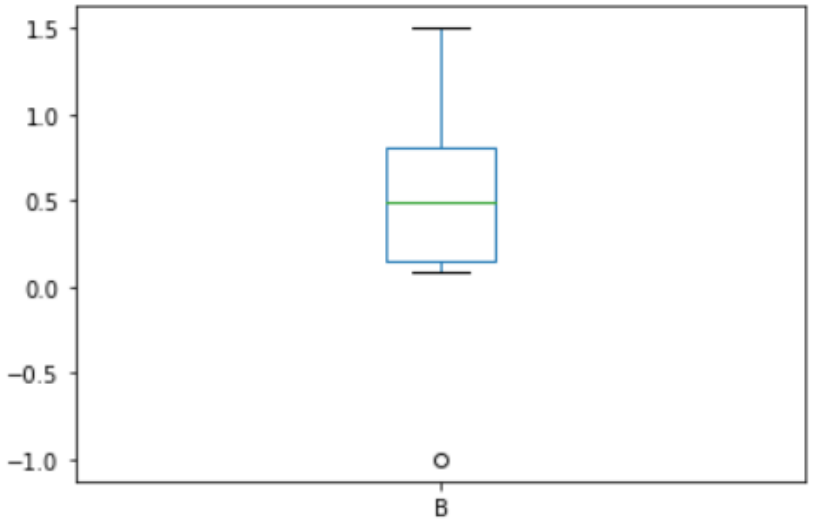
```
df.boxplot(column =['A'], grid = False)
```

```
<AxesSubplot:>
```

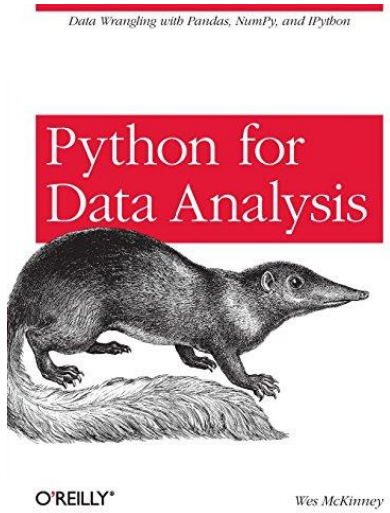



```
df.boxplot(column =['B'], grid = False)
```

```
<AxesSubplot:>
```



References & More Resources



- References:
  - McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.

- More Resources:
  - Machine Learning with Python: Foundations: https://www.linkedin.com/learning/machine-learning-with-python-foundations
  - Python Data Analysis on Linkedin Learning: https://www.linkedin.com/learning/python-data-analysis-2
    - To use Linkedin Learning, you can log in with your university account: https://myport.port.ac.uk/study-skills/linkedin-learning

Practical Session

- Please download Week08_Visualisation.ipynb file, and run it to learn new points.
- Please read the practical sheet (Week08_Practicals.pdf) and do the exercise.