INTRODUCTION TO DATA ANALYTICS

Data Exploration

การสำรวจข้อมูล

Dr. Rathachai Chawuthai

Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Agenda

- Data Format
- Charts
- Tools

Data Exploration อย่างมุมมองของ

เชื่อมต่อไป → ใช้แผนภาพ

A preliminary exploration of the data to better understand its characteristics.

ทำให้เราสามารถเลือกเครื่องมือที่เหมาะสมกับข้อมูลได้

- Key motivations of data exploration include
  - Helping to select the right tool for preprocessing or analysis
  - Making use of humans' abilities to recognize patterns
    - People can recognize patterns not captured by data analysis tools

Data Format

Data Table

- Iris

iris

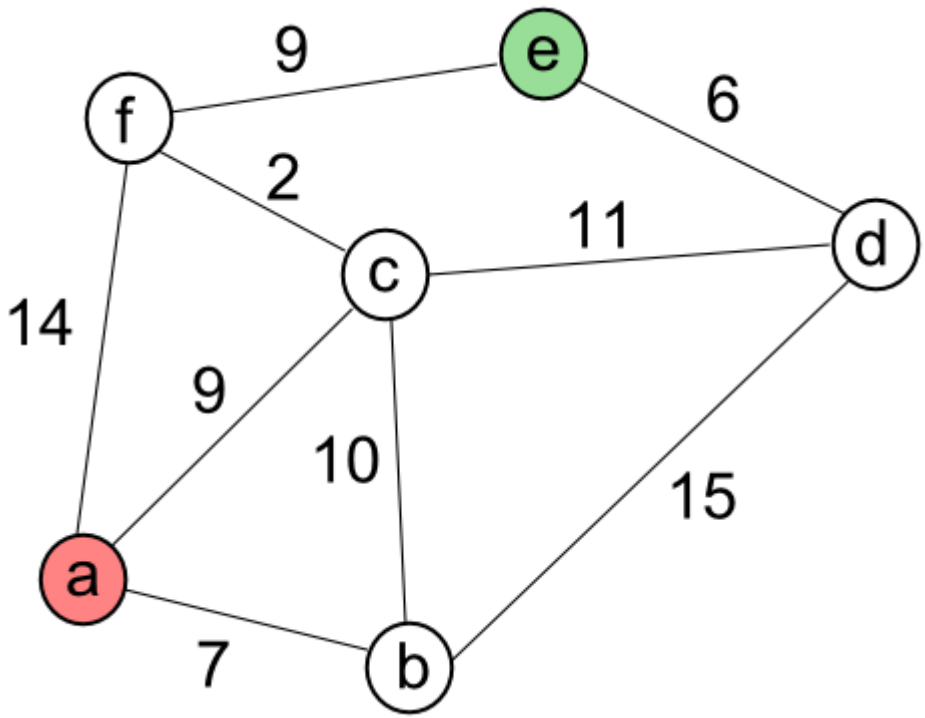
Filter

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
14	4.3	3.0	1.1	0.1	setosa
9	4.4	2.9	1.4	0.2	setosa
39	4.4	3.0	1.3	0.2	setosa
43	4.4	3.2	1.3	0.2	setosa
42	4.5	2.3	1.3	0.3	setosa
4	4.6	3.1	1.5	0.2	setosa
7	4.6	3.4	1.4	0.3	setosa
23	4.6	3.6	1.0	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa

Showing 1 to 10 of 150 entries

Ref: • (image) https://support.rstudio.com/hc/en-us/articles/205175388-Using-the-Data-Viewer

Graph



Ref: • (image) http://www.bogotobogo.com/python/python_graph_data_structures.php

Data Table

- Iris

คนใหม่ต้องเข้าใจแยกกรณี

iris

Filter

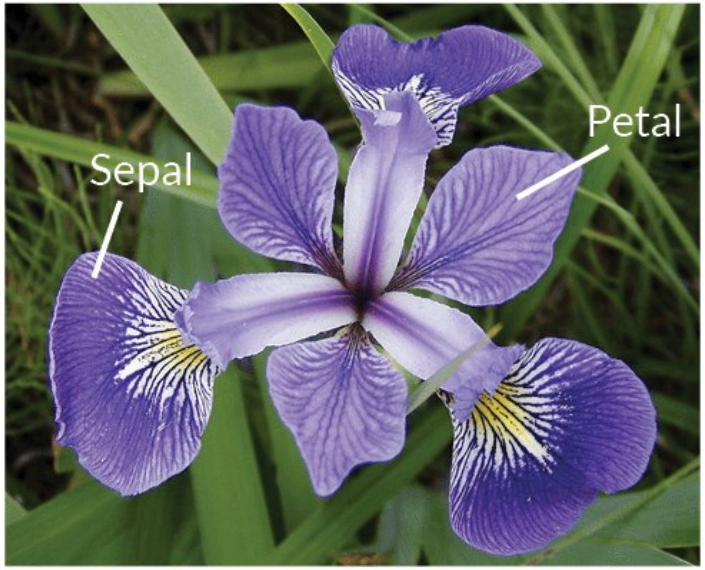
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
14	4.3	3.0	1.1	0.1	setosa
9	4.4	2.9	1.4	0.2	setosa
39	4.4	3.0	1.3	0.2	setosa
43	4.4	3.2	1.3	0.2	setosa
42	4.5	2.3	1.3	0.3	setosa
4	4.6	3.1	1.5	0.2	setosa
7	4.6	3.4	1.4	0.3	setosa
23	4.6	3.6	1.0	0.2	setosa
48	4.6	3.2	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa

Showing 1 to 10 of 150 entries

Ref: • (image) https://support.rstudio.com/hc/en-us/articles/205175388-Using-the-Data-Viewer

Data Table

- Iris ดอกไม้ 3 ชนิดที่คุณชอบใช้งาน



Iris Versicolor

Iris Setosa

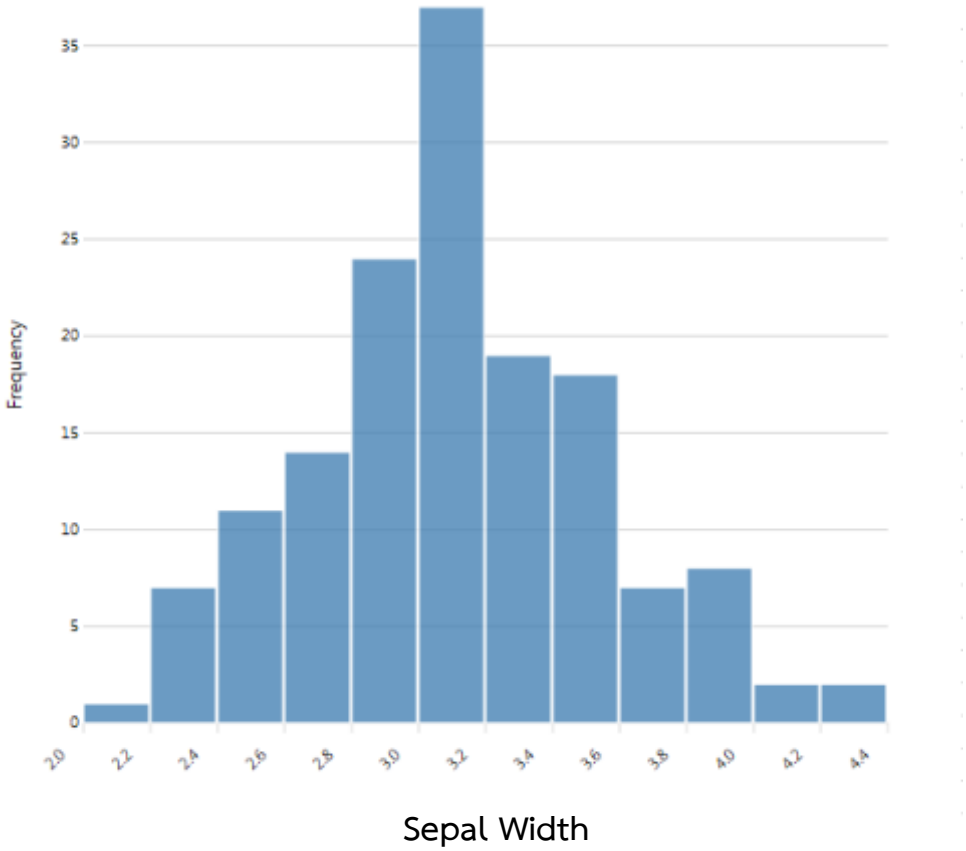
Iris Virginica

Ref: • (image) https://support.rstudio.com/hc/en-us/articles/205175388-Using-the-Data-Viewer

Charts

Histogram

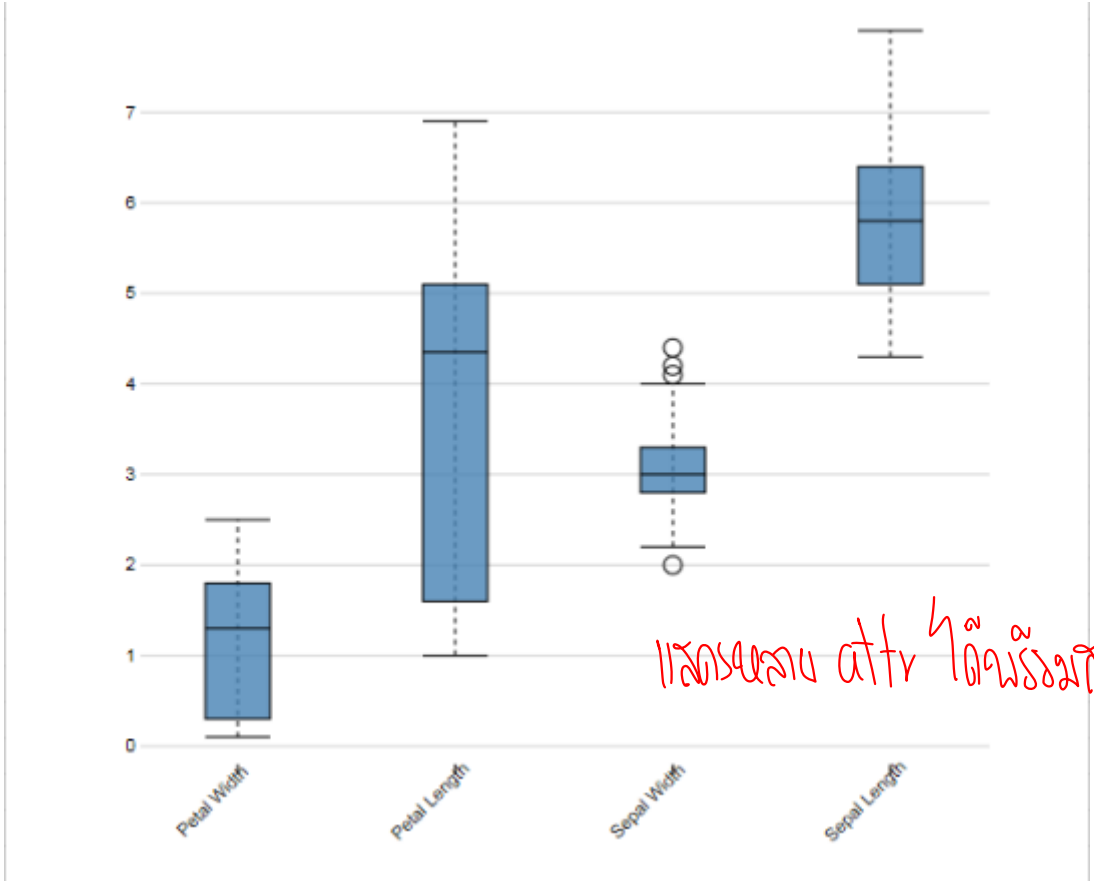
- Iris



Ref: • (image) https://www.solver.com/xlminer-dataviz-app-help-and-support

Boxplot

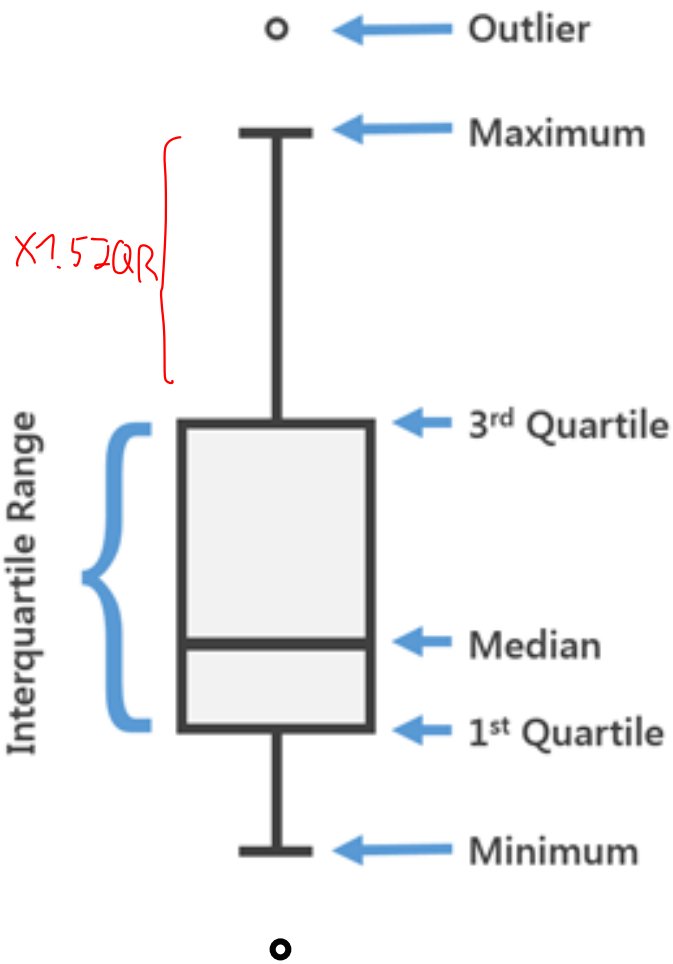
- Iris



แสดงผลของ attr ได้พร้อมกัน

Boxplot

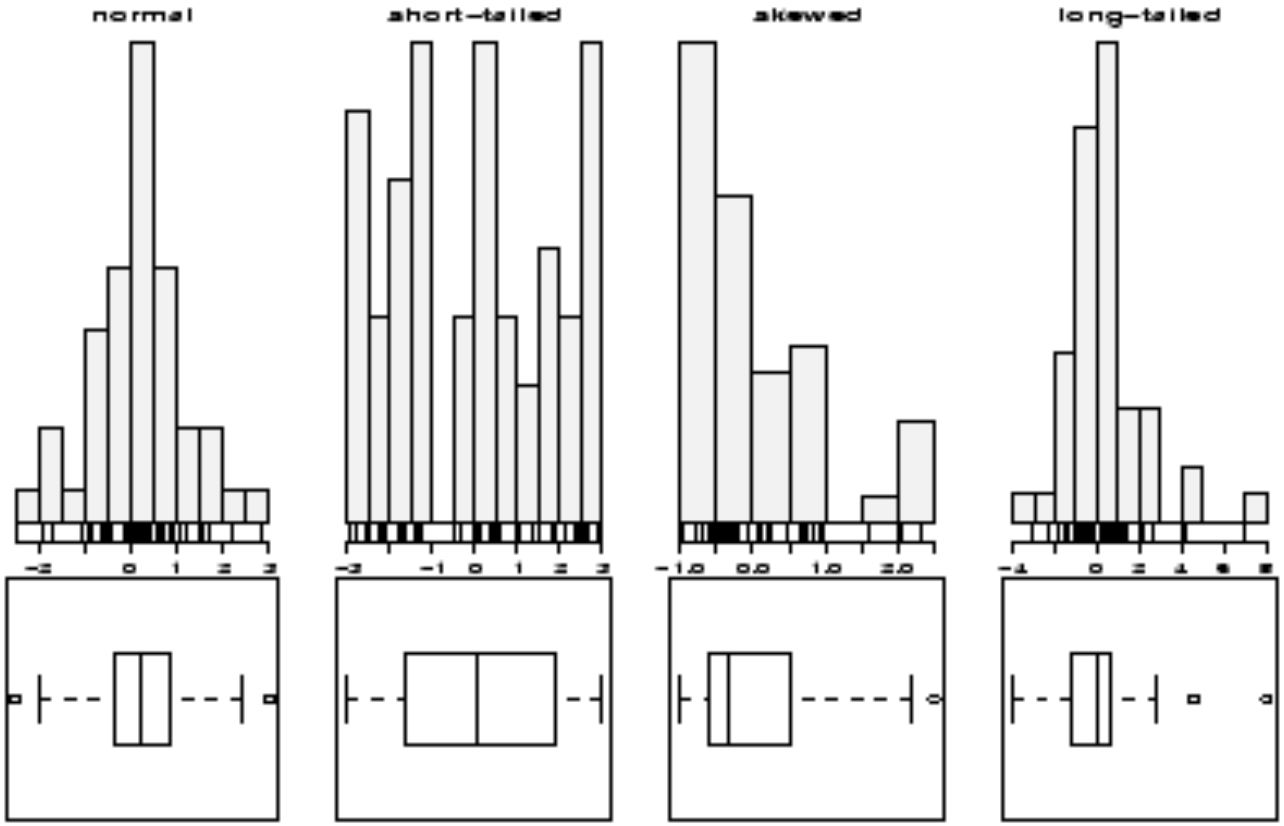
- Interpretation



Ref: • (image) https://www.solver.com/xlminer-dataviz-app-help-and-support

Histogram & Boxplot

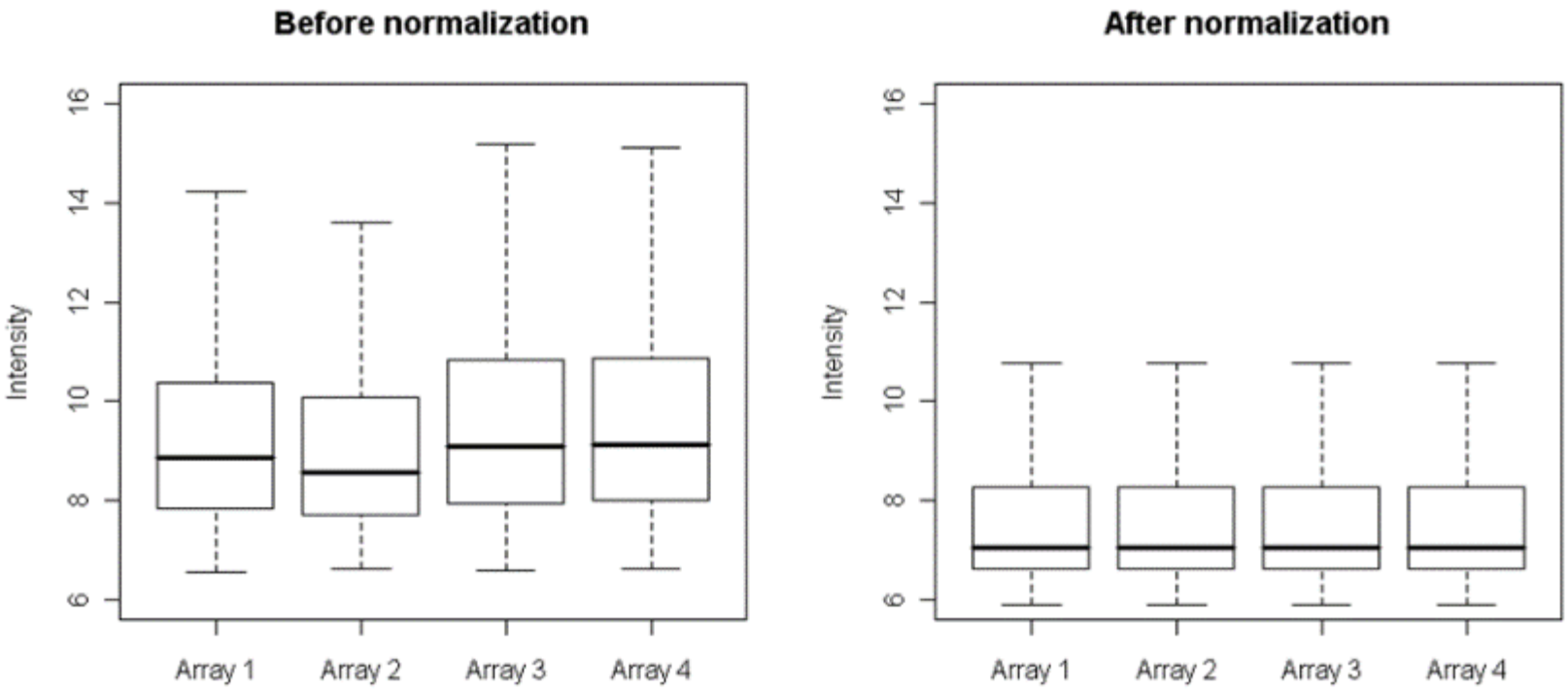
- Interpretation



Ref: • (image) https://www.solver.com/xlminer-dataviz-app-help-and-support

Boxplot : Normalization

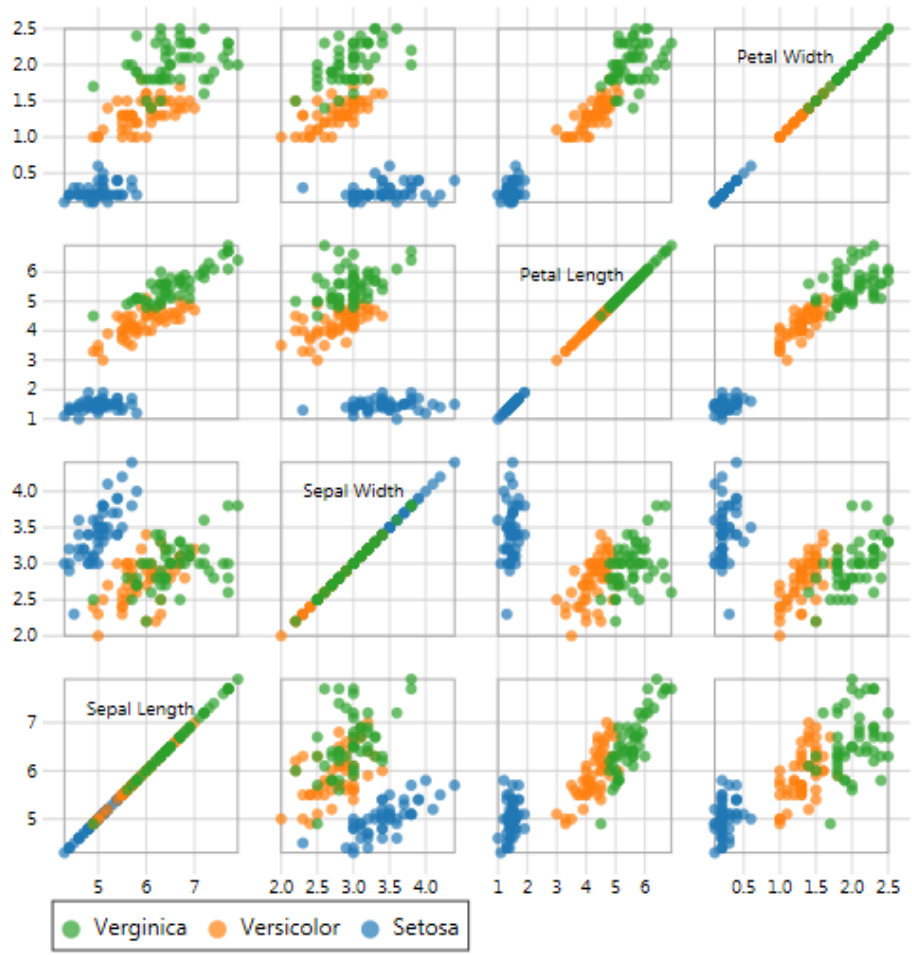
เมื่อเอามา normalize ค่าจะอยู่ในช่วงเดียวกัน



Ref: • (image) http://telemedicina.med.muni.cz/genomic-proteomic-analysis/res/image/boxplotNormalization.png

Scatter Plot

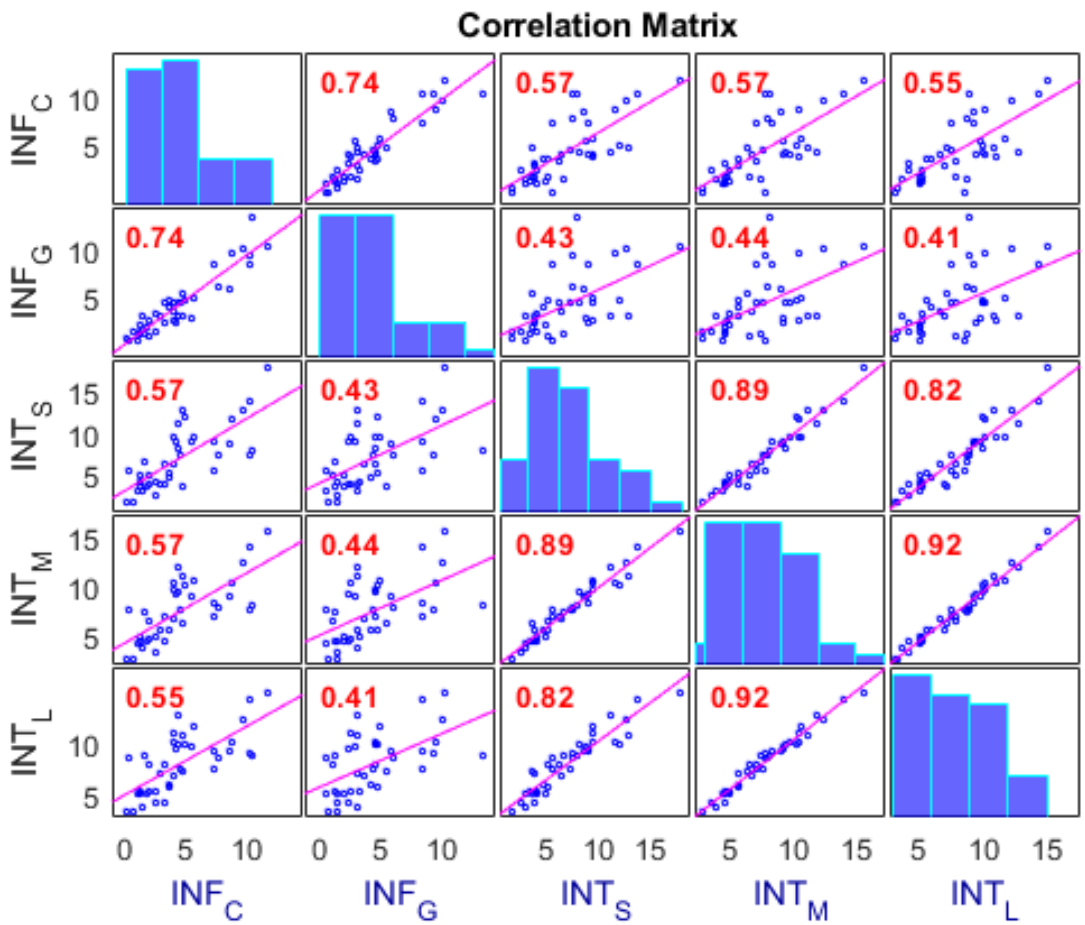
- Iris



Ref: • (image) https://www.solver.com/xlminer-dataviz-app-help-and-support

Correlation Matrix

มีทั้ง histogram, scatter ในตัว



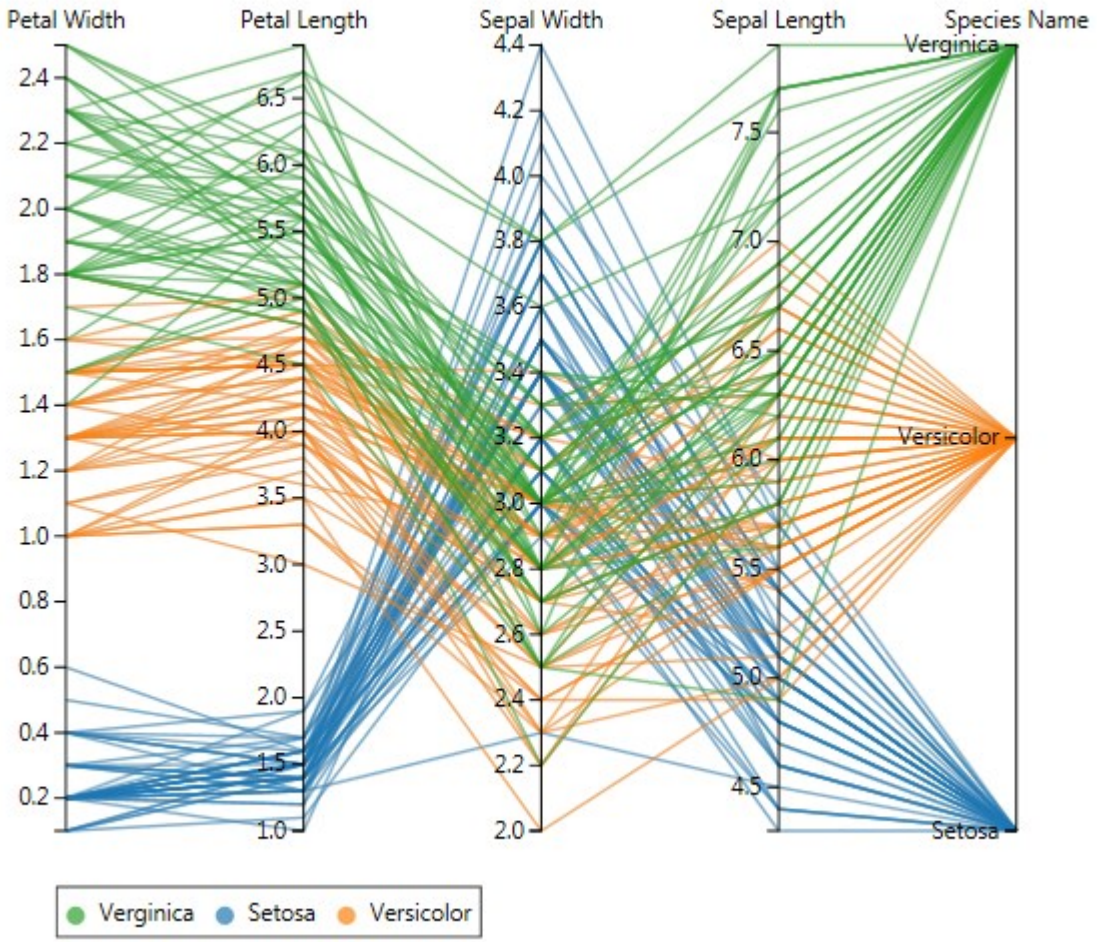
Ref: • (image) https://freshbiostats.wordpress.com/2013/09/04/an-example-of-principal-components-analysis/

Heatmap

Ref: • (image) https://tryolabs.com/blog/2017/03/16/pandas-seaborn-a-guide-to-handle-visualize-data-elegantly/

Heatmap

- Iris



Ref: • (image) https://www.solver.com/xlminer-dataviz-app-help-and-support

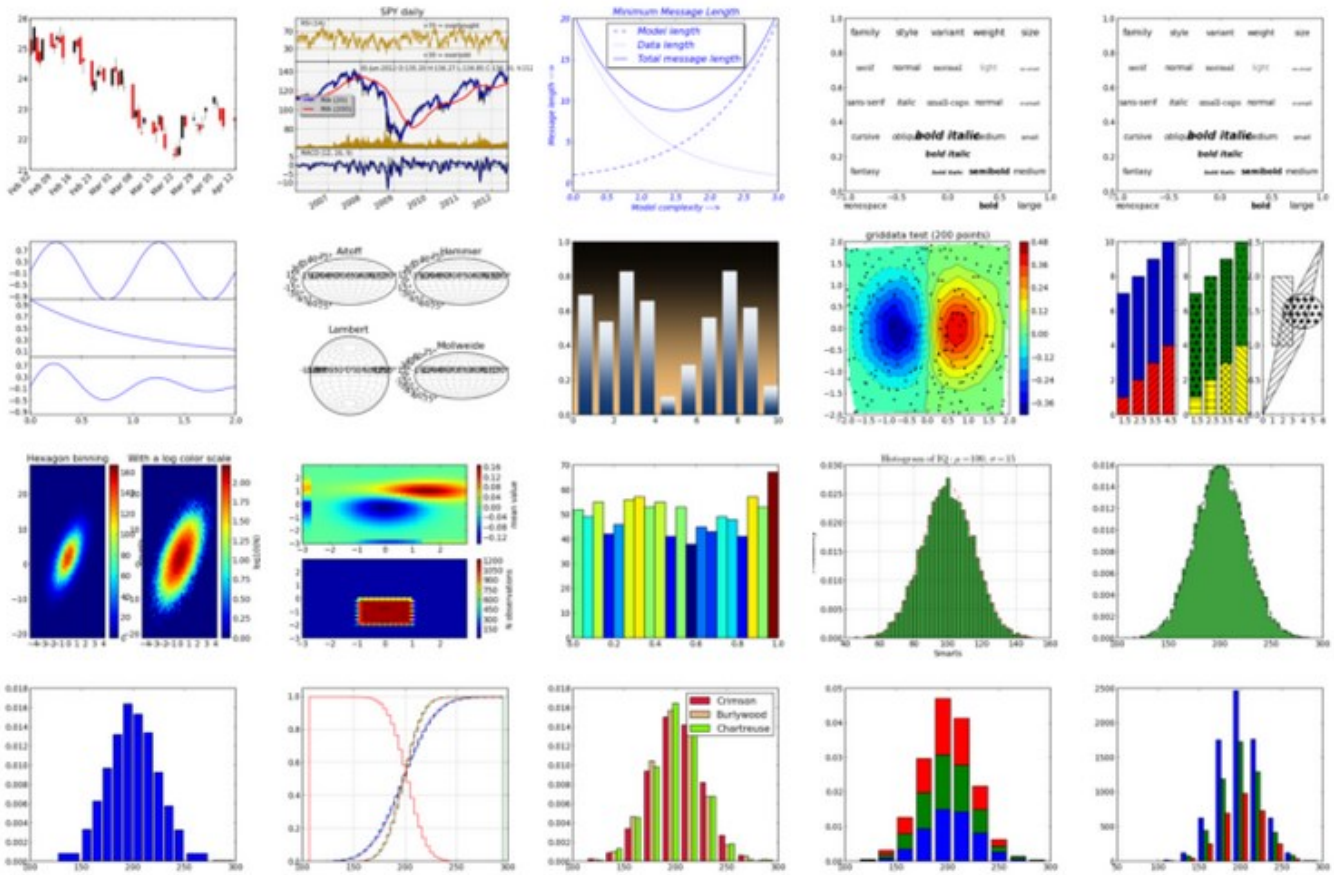
Tools ใช้เครื่องมือในการแสดงแผนภูมิ กราฟ

Tools

- mathplotlib → ใช้กับ pd ช้าๆ เหมือนแกะมาอู้กัน
  - https://matplotlib.org/
  - `import matplotlib.pyplot as plt`
- pandas plot
  - https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.plot.html
  - `import pandas as pd`
- seaborn → สวย
  - https://seaborn.pydata.org/
  - `import seaborn as sns`

https://matplotlib.org

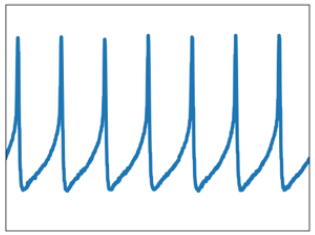
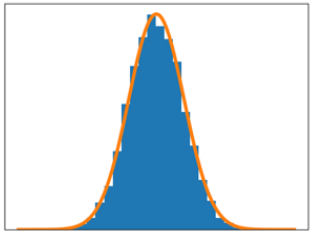
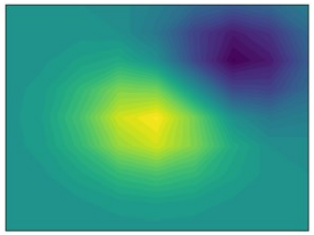
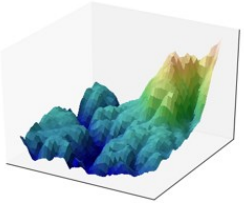
```
import matplotlib.pyplot as plt
```



Ref: • (image) https://wiki.openhatch.org/wiki/Matplotlib

mathplotlib

- Matplotlib is a Python **2D plotting** library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Matplotlib can be used in **Python scripts**, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

matplotlib Advantages

- A multi-platform data visualization tool built on the numpy and scipy framework. Therefore, it's fast and efficient.
- It possesses the ability to work well with many operating systems and graphic backends.
- It possesses high-quality graphics and plots to print and view for a range of graphs such as histograms, bar charts, pie charts, scatter plots and heat maps.
- With Jupyter notebook integration, the developers have been free to spend their time implementing features rather than struggling with compatibility.
- It has large community support and cross-platform support as it is an open source tool.
- It has full control over graph or plot styles such as line properties, thoughts, and access properties.

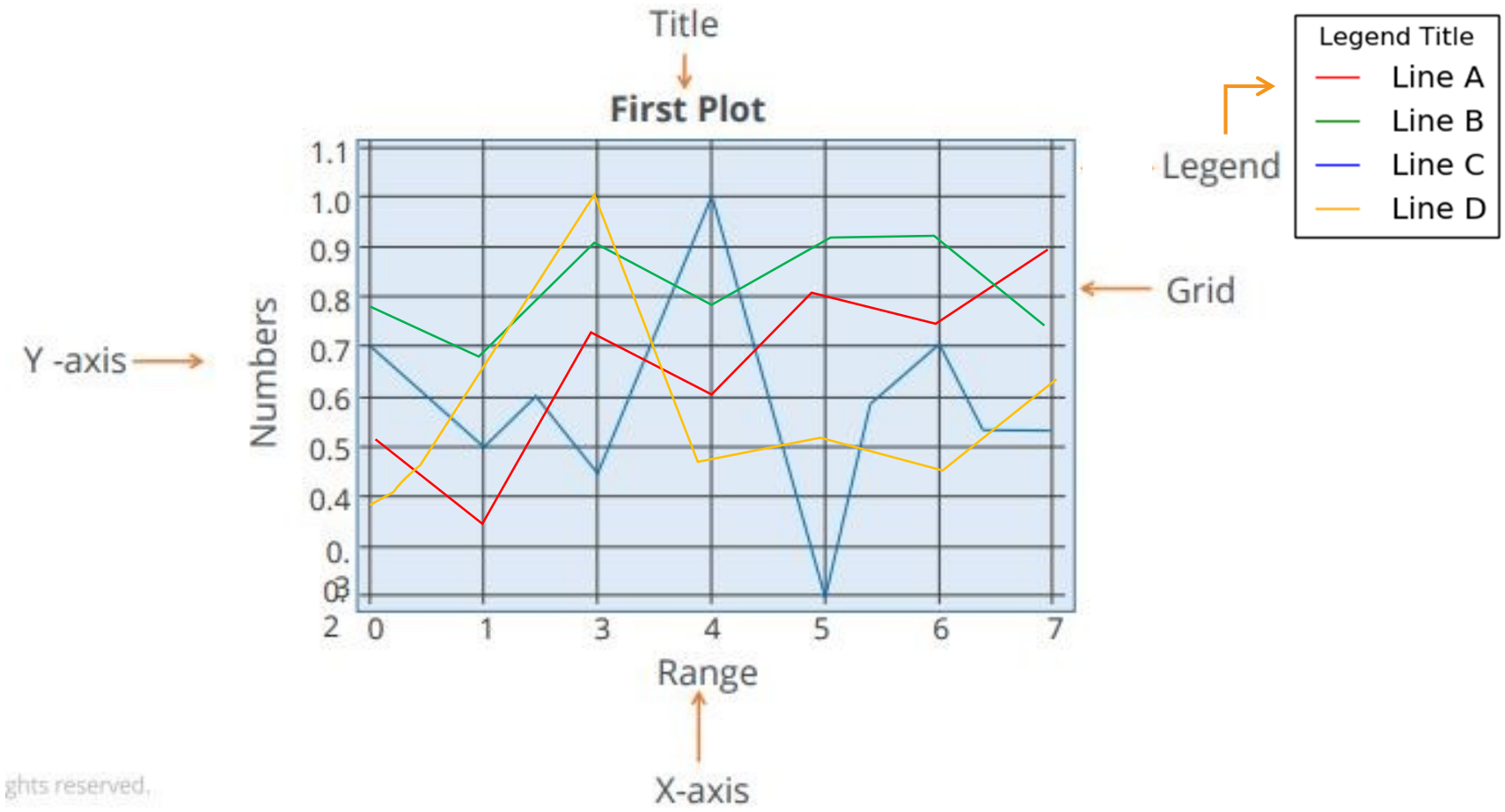
Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Understanding the Plot

- A plot is a graphical representation of data, which shows the relationship between two variables or the distribution of data.
- This is a line plot of the random numbers on the y-axis and the range on the x-axis. The background of the plot is called a grid. The text first plot denotes the title of the plot and text line one denotes the legend.

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Understanding the Plot



ghts reserved.

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Steps

1. Import the required libraries
2. Define or import the required dataset
3. Set the plot parameters
4. Display the created plot

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Steps

```
In [1]: #import numpy for generating random numbers
        import numpy as np
        #import matplotib library
        import matplotlib.pyplot as plt
        from matplotlib import style
        %matplotlib inline
```

Generate random numbers — numpy

Plot the numbers — pyplot

set the grid style — style

```
In [21]: #generate random numbers (total 10)
         randomNumber = np.random.rand(10)
```

used numpy random method — Defined the dataset

```
In [22]: #view them
         print randomNumber

[ 0.71892609  0.49065612  0.61092193  0.43397501  0.94771363  0.31505178
  0.58568599  0.6929941   0.4288734   0.43774794]
```

view the created random numbers — Print method

```
In [23]: #select the style of the plot
         style.use('ggplot')
         #plot the random number
         plt.plot(randomNumber,'g',label='line one',linewidth=2)
         #x axis is number of random numbers (index)
         plt.xlabel('Range')
         #y axis is actual random number
         plt.ylabel('Numbers')
         #Title of the plot
         plt.title('First Plot')
```

ggplot — Set the style

Set the legend

Set line width

Set coordinates labels

Set the title

```
plt.legend()
plt.show()
```

Plot the graph

Display the created plot

Step 01 — Import the required libraries

Step 02 — Define or import the required dataset

Step 03 — Set the plot parameters

Step 04 — Display the created plot

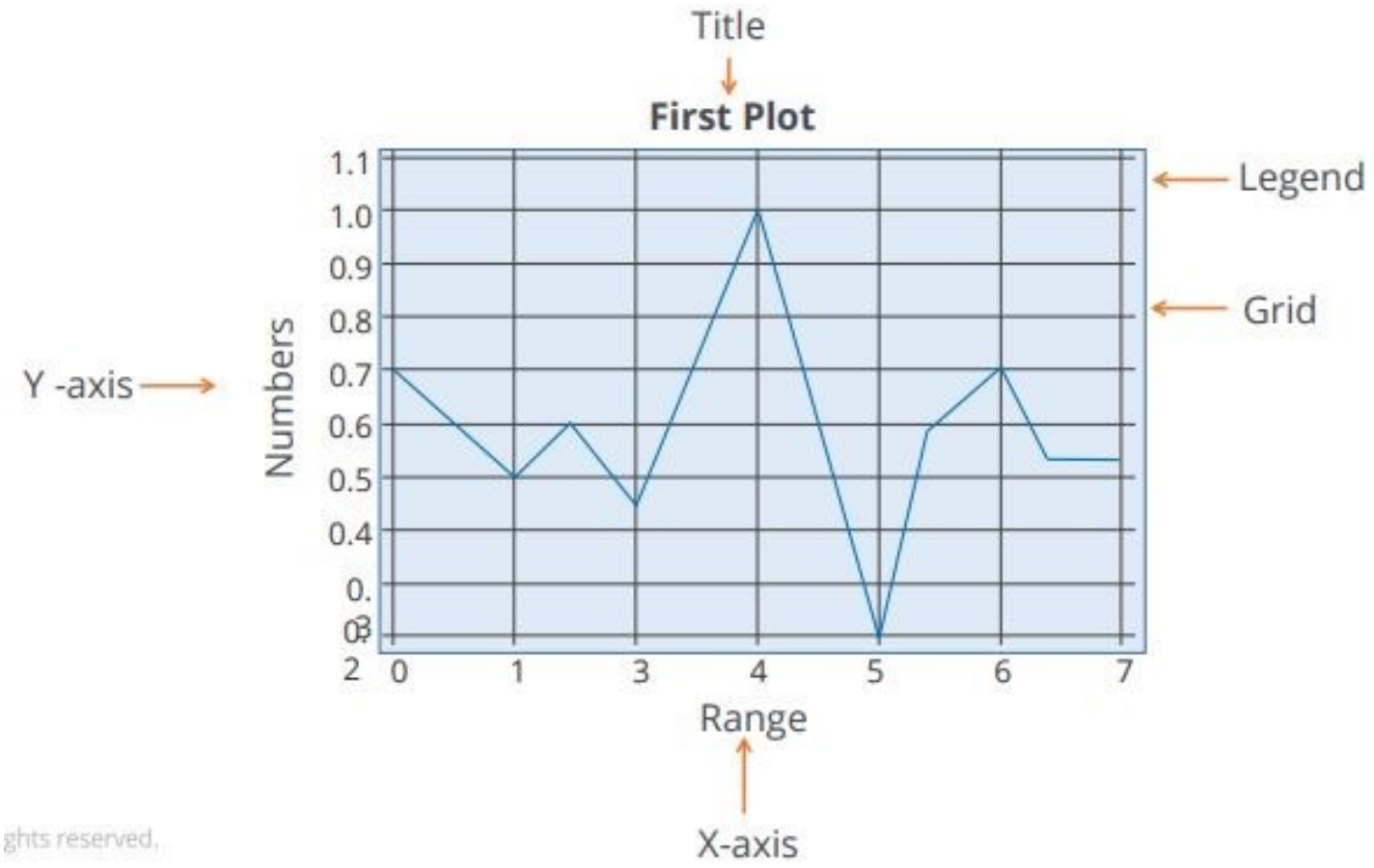
Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

In the 3rd Step

- The third step is to set the plot parameters.
- In this step, we set the style of the plot, labels of the coordinates, titles of the plot, the legend and the linewidth.
- In this example,
  - we have used ggplot as the plot style.
  - The plot method is used to plot the graph against the random numbers.
  - In the plot method the word 'g' denotes the plotline color as green, the label denotes the legend label and is named as line one.
  - Also the linewidth=2.
  - Note that we have labeled the x-axis as range and the as labels and set the title as First Plot.

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Result



ghts reserved.

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Creating a 2-D Plot

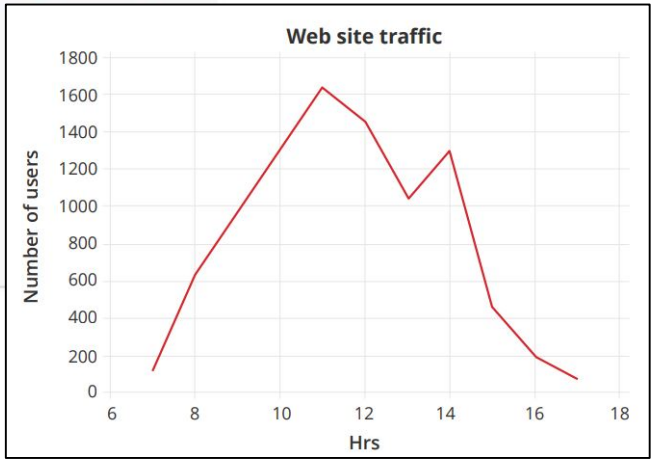
```
In [1]: #import matplotib library
        import matplotlib.pyplot as plt
        from matplotlib import style
        %matplotlib inline
```

```
In [2]: #website traffic data
        #number of users/ visitors on the web site
        web_customers = [123,645,950,1290,1630,1450,1034,1295,465,205,80 ]
        #Time distribution (hourly)
        time_hrs = [7,8,9,10,11,12,13,14,15,16,17]
```

List of users

Time

```
In [3]: #select the style of the plot
        style.use('ggplot')
        #plot the web site traffif data (X-axis hrs and Y axis as number of users)
        plt.plot(time_hrs,web_customers)
        #set the title of the plot
        plt.title('Web site traffic')
        #set label for x axis
        plt.xlabel('Hrs')
        #set label for y axis
        plt.ylabel('Number of users')
        plt.show()
```



Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

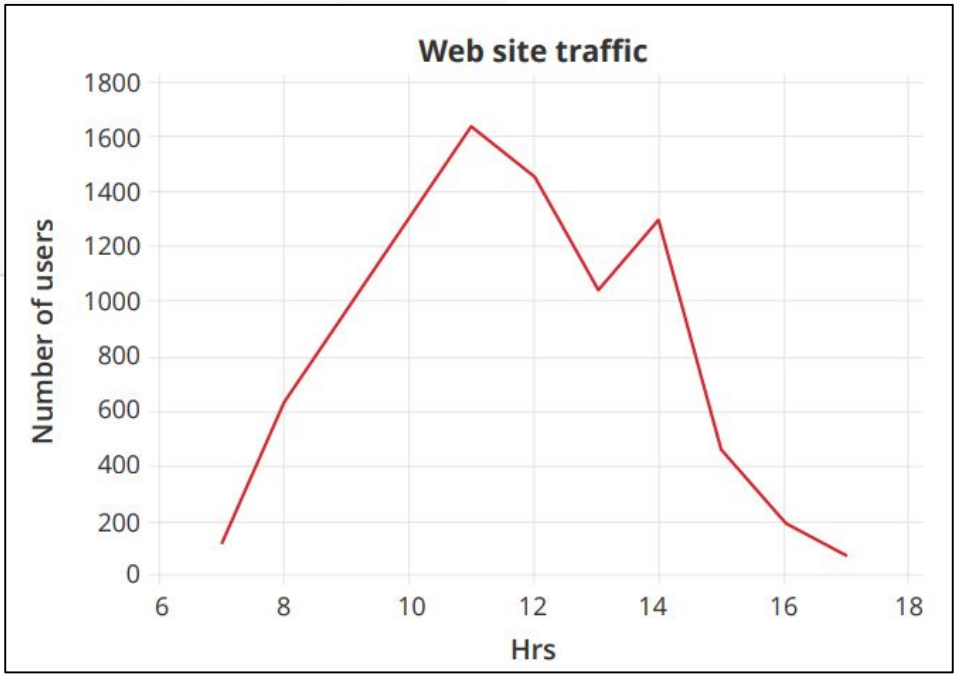
Creating a 2-D Plot

```
In [2]: #website traffic data
        #number of users/ visitors on the web site
        web_customers = [123,645,950,1290,1630,1450,1034,1295,465,205,80 ]
        #Time distribution (hourly)
        time_hrs = [7,8,9,10,11,12,13,14,15,16,17]
```

List of users

Time

```
In [3]: #select the style of the plot
        style.use('ggplot')
        #plot the web site traffif data (X-axis hrs and Y axis as number of users)
        plt.plot(time_hrs,web_customers)
        #set the title of the plot
        plt.title('Web site traffic')
        #set label for x axis
        plt.xlabel('Hrs')
        #set label for y axis
        plt.ylabel('Number of users')
        plt.show()
```



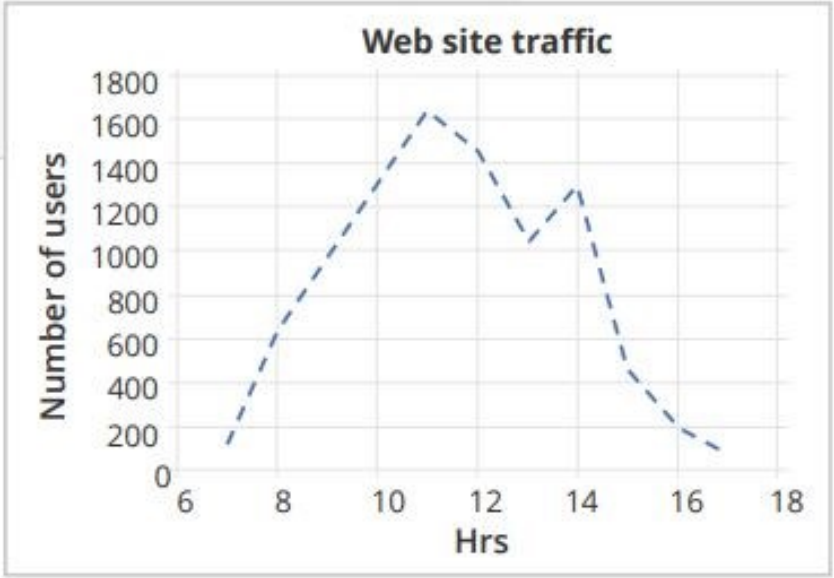
Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Creating a 2-D Plot

```
#select the style of the plot
style.use('ggplot')
#plot the web stite traffic data (x axis hrs and y asis as number of users)
plt.plot(time_hrs,web_customers,color = 'b',linestyle = '--',linewidth=2.5)
#set the title of the plot
plt.title('Web site traffic')
#set the label for x axis
plt.xlabel('hrs')
#set the label for y axis
plt.ylabel('number of users')
plt.show()
```

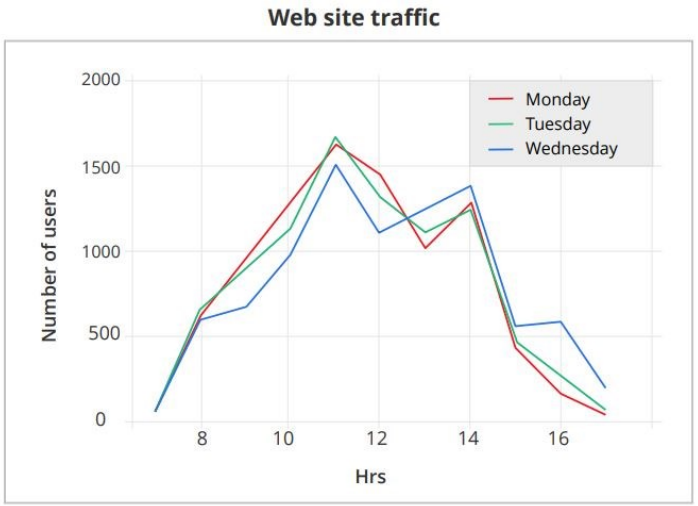
Line Color (blue)

Dashed (--)



Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Multiple Plots




```
In [4]: #website traffic data
        #number of users/ visitors on the web site
        #monday web traffic
        web_monday = [123,645,950,1290,1630,1450,1034,1295,465,205,80]
        #tuesday web traffic
        web_tuesday= [95,680,889,1145,1670,1323,1119,1265,510,310,110]
        #wednesday web traffic
        web_wednesday= [105,630,700,1006,1520,1124,1239,1380,580,610,230]
        #Time distribution (hourly)
        time_hrs = [7,8,9,10,11,12,13,14,15,16,17]
```

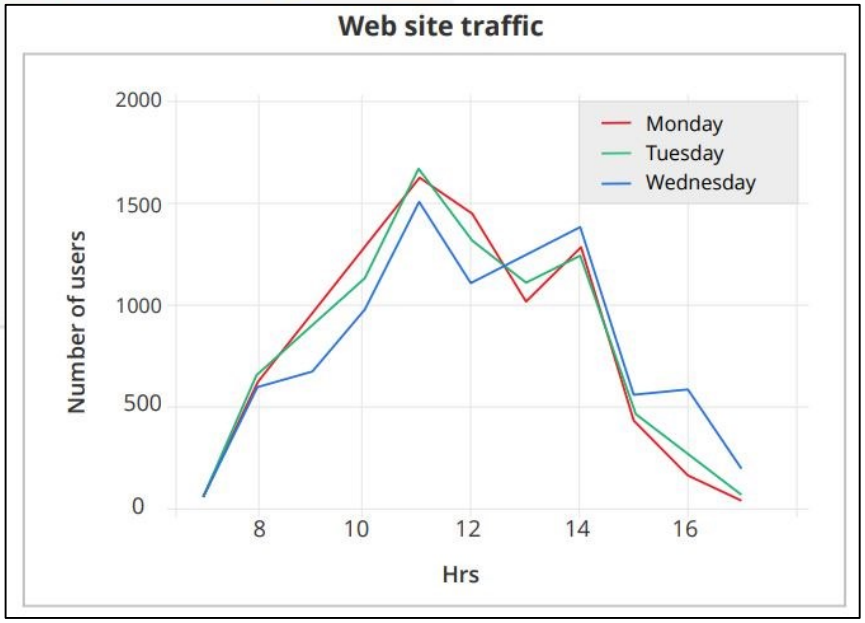
Web traffic data

Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

Multiple Plots

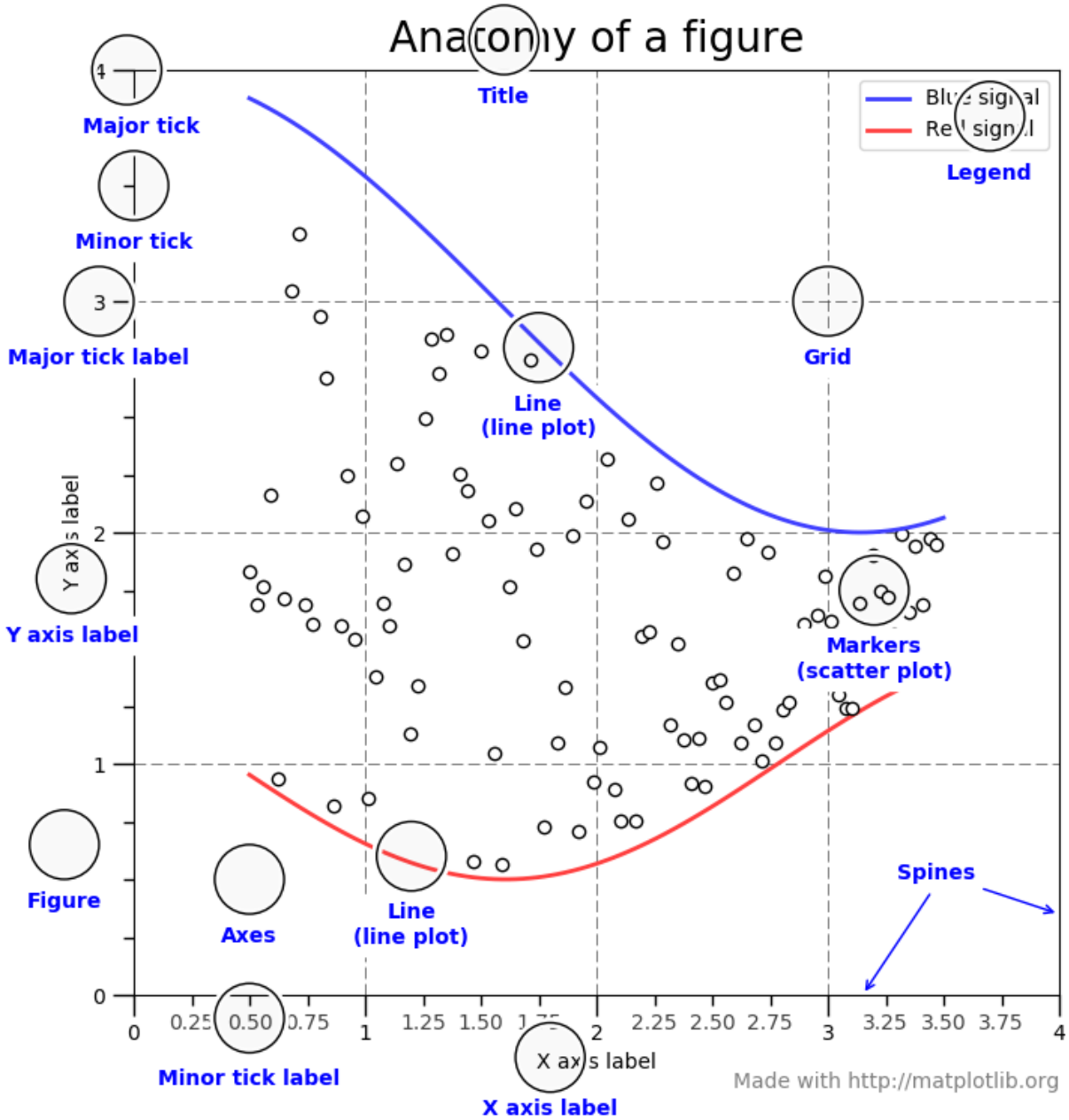
```
In [5]: #select the style of the plot
        style.use('ggplot')
        #plot the web site traffic data (X-axis hrs and Y axis as number of users)
        #plot the monday web traffic with red color
        plt.plot(time_hrs,web_monday,'r',label='monday',linewidth=1)
        #plot the monday web traffic with green color
        plt.plot(time_hrs,web_tuesday,'g',label='tuesday',linewidth=1.5)
        #plot the monday web traffic with blue color
        plt.plot(time_hrs,web_wednesday,'b',label='wednesday',linewidth=2)
        plt.axis([6.5,17.5,50,2000])
        #set the title of the plot
        plt.title('Web site traffic')
        #set label for x axis
        plt.xlabel('Hrs')
        #set label for y axis
        plt.ylabel('Number of users')
        plt.legend()
        plt.show()
```

Set different colors and line widths for different days



Ref:
- https://matplotlib.org
- https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

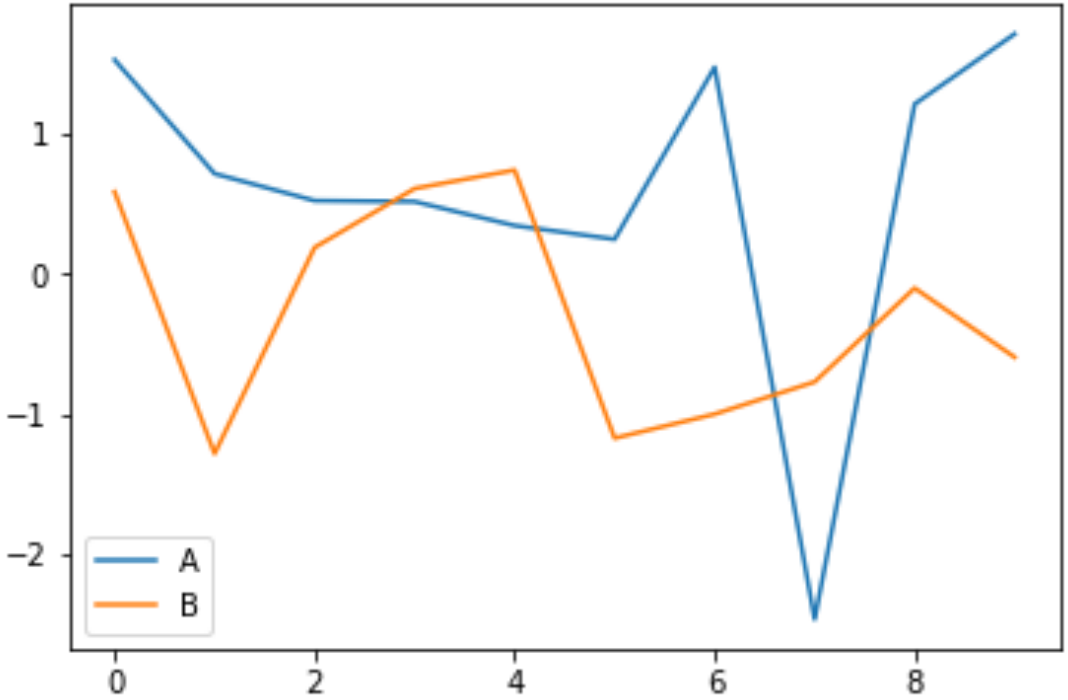
mathplotlib



Ref: • https://www.simplilearn.com/data-visualization-in-python-using-matplotlib-tutorial

pandas plot : Lines

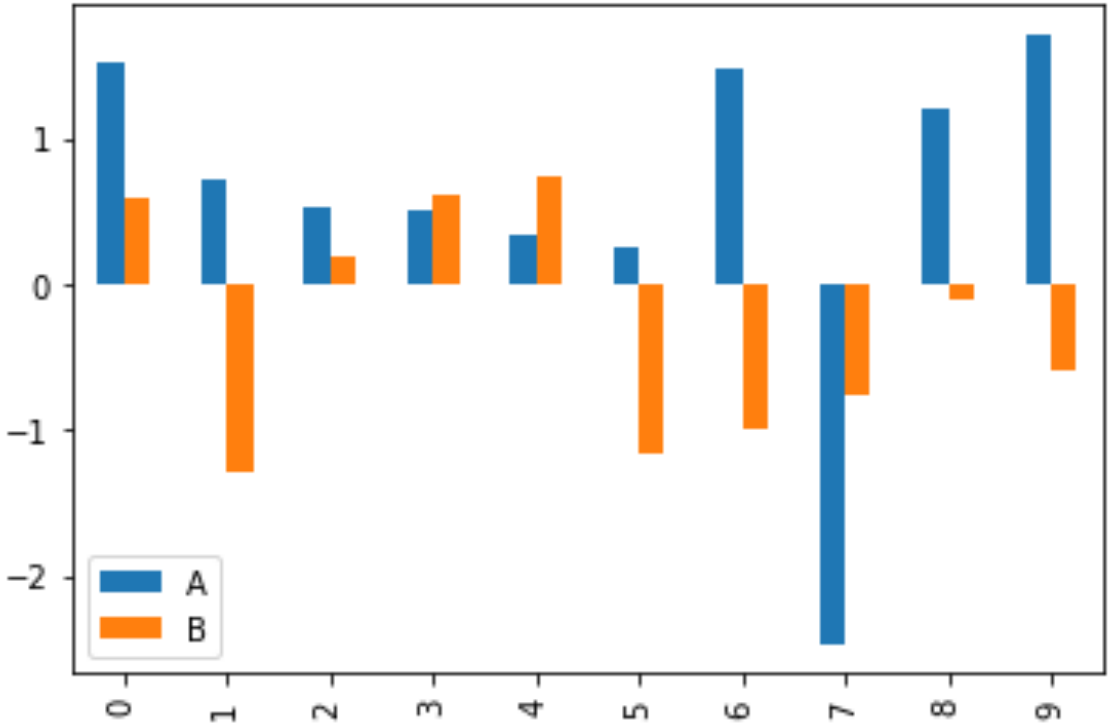
```
df = pd.DataFrame({"A": np.random.randn(10),
                   "B": np.random.randn(10) })
df.plot()
```



Ref:
- https://matplotlib.org
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

pandas plot : Bar

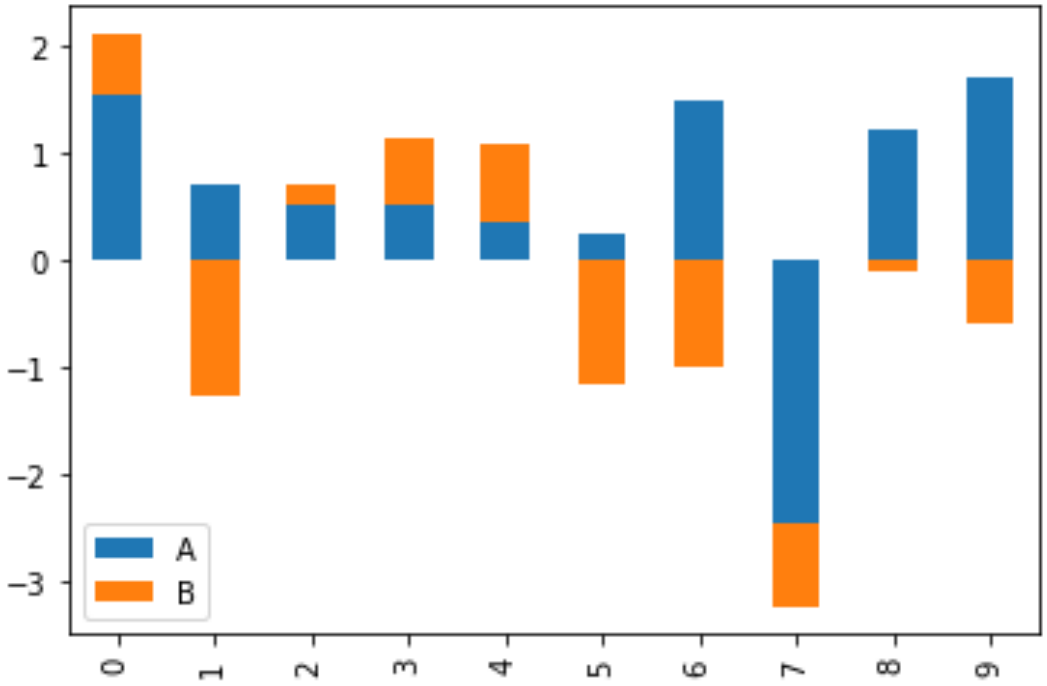
```
df = pd.DataFrame({"A": np.random.randn(10),
                   "B": np.random.randn(10) })
df.plot(kind="bar")
```



Ref:
- https://matplotlib.org
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

pandas plot : Stacked Bar

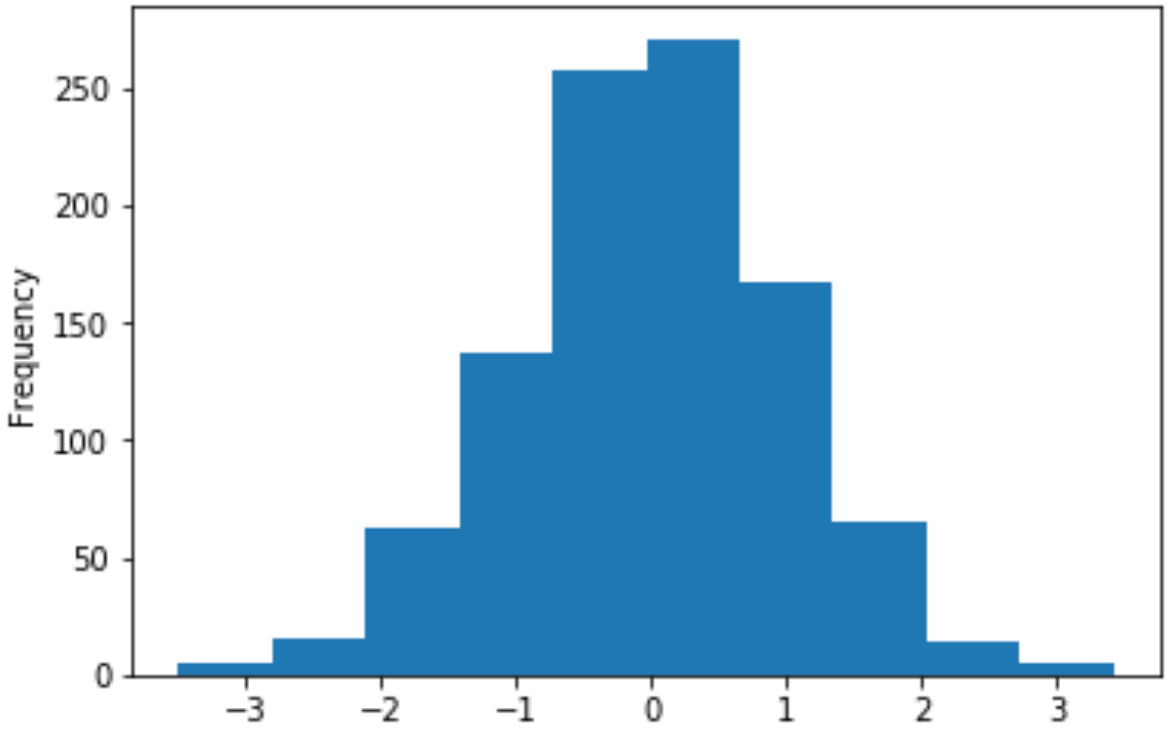
```
df = pd.DataFrame({"A": np.random.randn(10),
                   "B": np.random.randn(10) })
df.plot.bar(stacked=True)
```



Ref:
- https://matplotlib.org
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

pandas plot : Histogram

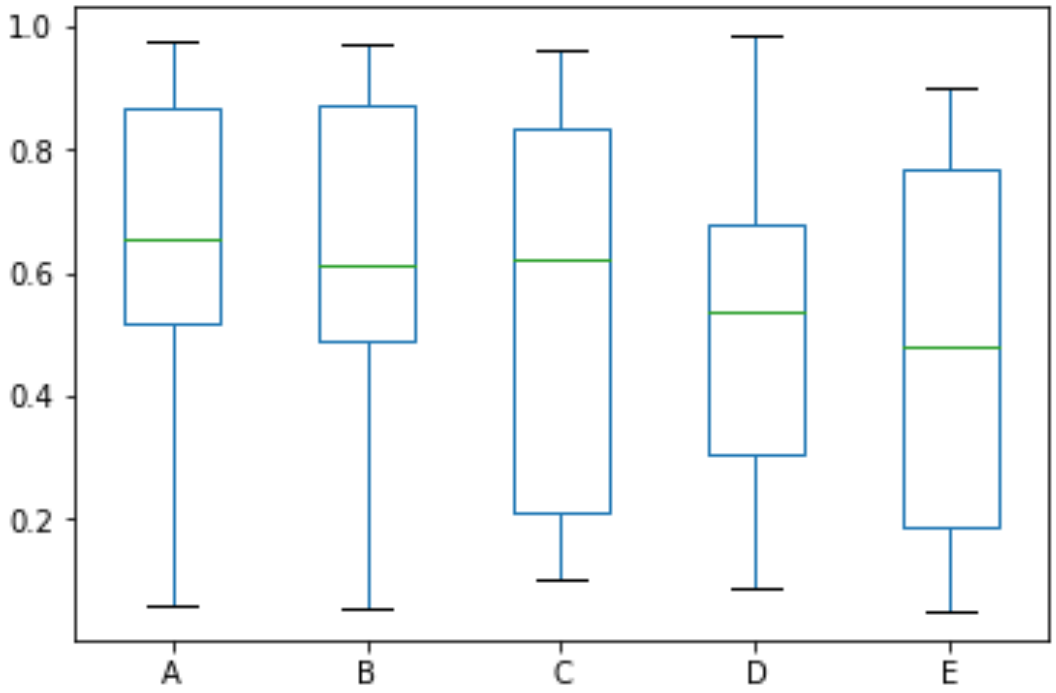
```
df = pd.DataFrame({"A": np.random.randn(1000)})
df["A"].plot.hist()
```



Ref:
- https://matplotlib.org
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

pandas plot : Box Plot

```
df = pd.DataFrame(np.random.rand(10, 5),
                  columns=['A', 'B', 'C', 'D', 'E’])
df.plot.box()
```

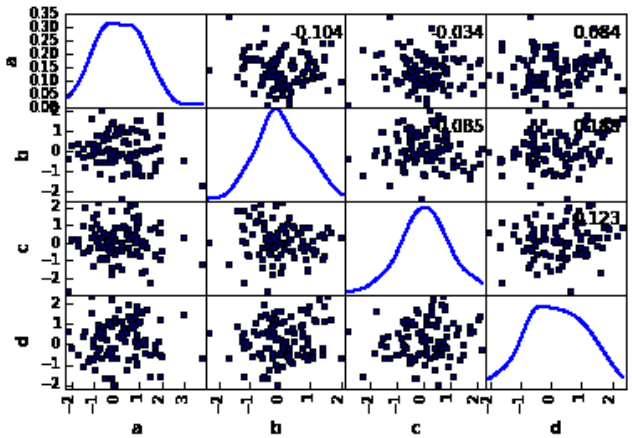


Ref:
- https://matplotlib.org
- https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

pandas plot : Scattergram

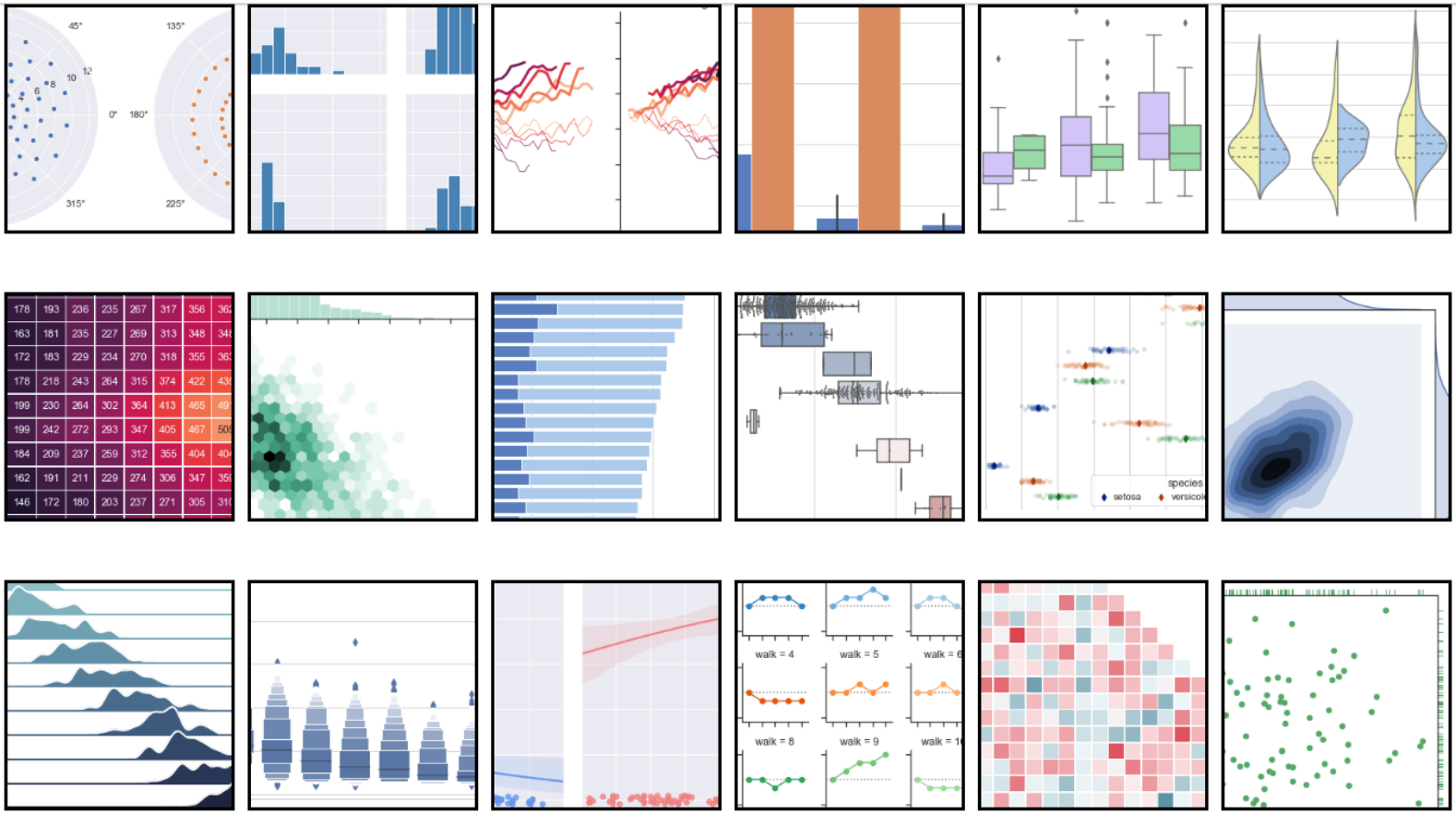
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pandas.plotting import scatter_matrix

df = pd.DataFrame(np.random.randn(100, 4),
                  columns=['a', 'b', 'c', 'd'])
axes = scatter_matrix(df, alpha=0.5, diagonal='kde')
corr = df.corr().as_matrix()
for i, j in zip(*plt.np.triu_indices_from(axes, k=1)):
    axes[i, j].annotate("%.3f" %corr[i,j], (0.8, 0.8),
xycoords='axes fraction', ha='center', va='center')
plt.show()
```



https://seaborn.pydata.org

```
import seaborn as sns
```



Ref: • (image) https://seaborn.pydata.org

seaborn

Seaborn is a library for making **statistical graphics in Python**. It is **built on top of matplotlib** and closely **integrated with pandas** data structures. Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- **Concise control over matplotlib figure styling with several built-in themes**
- Tools for choosing color palettes that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Ref: • https://seaborn.pydata.org

seaborn : Heatmap

```
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline

cnames = ['A', 'B', 'C', 'D']
df = pd.DataFrame(
        abs(np.random.randn(5, 4)),
        columns=cnames)

sns.heatmap(df, annot=True)
```

Ref:
- https://seaborn.pydata.org
- https://stackoverflow.com/questions/27768677/pandas-scatter-matrix-display-correlation-coefficient

“

Most of the world will make decisions by either guessing or using their gut. They will be either lucky or wrong.

”

Suhail Doshi