CS3750 – Data Visualisation

Assignment 1

Data Visualization using Python: Matplotlib and Seaborn

Kajanan Selvanesan

190287R

❖ Objectives of Visualization

- Record information.
- Analyse data to support reasoning.
- Confirm hypotheses.
- Communicate ideas to others.
- Point out interesting things.

❖ Useful Python Libraries for Data Visualization

We can use the following Python libraries to manage and store the data before using them for visualisation.

- NumPy: Used for data manipulation.
- pandas
  - Used for storing, handling and analysing input data.
  - It is particularly suited for tabular data.
  - We can use it to do powerful data operations like description, mean, median, etc.
  - Main data structures which are included here: DataFrame and Series.

We can use the following Python libraries to visualise the data by giving those data to them in the form of NumPy ndarray or pandas DataFrame.

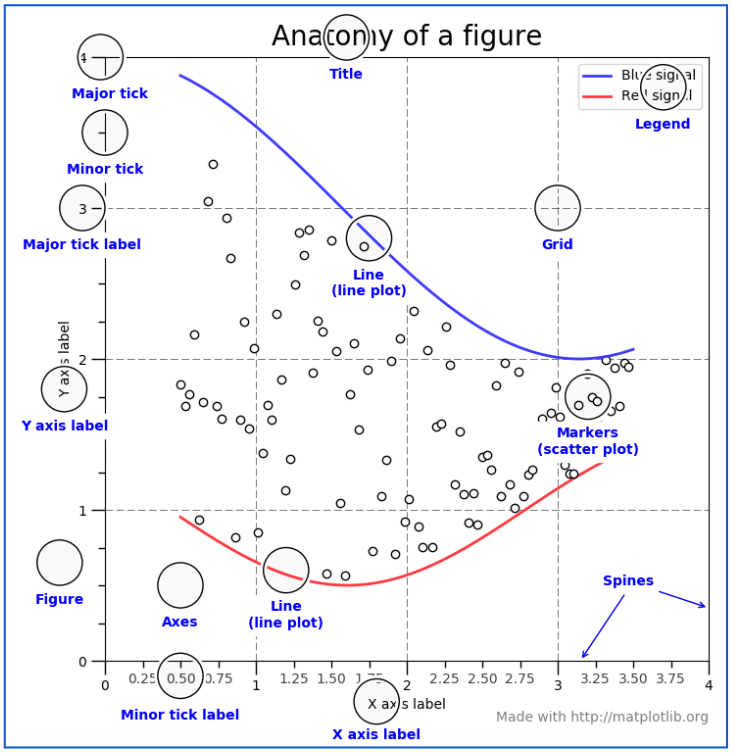
- matplotlib
  - Used for basic plotting.
  - **Advantages**:
    - Highly customizable.
    - Works well with NumPy and pandas.
  - **Disadvantage**: Requires more lines of code than that of seaborn.
- seaborn
  - Used for statistical data visualisation.
  - **Advantages**:
    - Can get visualisations with good default themes using a few lines of code.
    - It is integrated to work great with pandas's data-frame.
  - **Disadvantage**: It's not highly customizable as matplotlib since it uses matplotlib under the hood.
- bokeh
  - Used for interactive data visualisation.
  - **Advantage**: Can get interactive visualisations using it.
  - **Disadvantage**: Requires modern web browsers to run since it integrates with JavaScript.

❖ Some Basics of matplotlib

```
from matplotlib import pyplot as plt # importing the library
plt.style.use('seaborn-whitegrid') # setting theme for styling

# define a single container that contains all the objects
representing axes, graphics, text, and labels.
fig = plt.figure()

# define a bounding box with ticks and labels, which will
eventually contain the plot elements that make up our
```

```
visualization. This box is associated with the above fig container.
ax = plt.axes()
```

Object ax has several customisable attributes. The following image shows those attributes and the following codes show how to customise some of those attributes in Python using matplotlib.

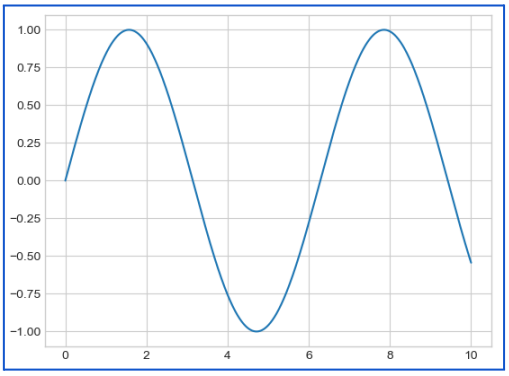



```
# adding a title for ax
ax.set_title('Title')
# adding X axis label for ax
ax.set_xlabel('x label')
# adding Y axis label for ax
ax.set_ylabel('y label')
# adding legend for ax
ax.legend()
# defining X axis limit for ax
ax.set_xlim(-5, 15)
# defining Y axis limit for ax
ax.set_ylim(-3, 3)

plt.show() # showing the plot

fig, axs = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes

# NOTE: above codes show how to customise only a few of the ax's attributes. But we can customise even more attributes like colour of the plot, the pattern of the markers, xtick's labels, etc.
```

❖ Some Basics of seaborn

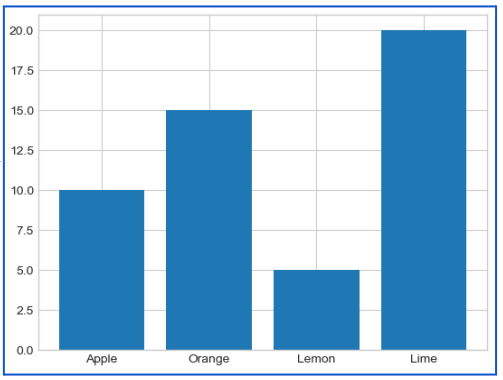
```
# Here, we can use some of the matplotlib codes as well for the plot customizations. However, here, most of the attributes of ax are defined automatically.
import seaborn as sns # importing the library
sns.set_style('darkgrid') # setting theme for styling
```

❖ Types of Plots

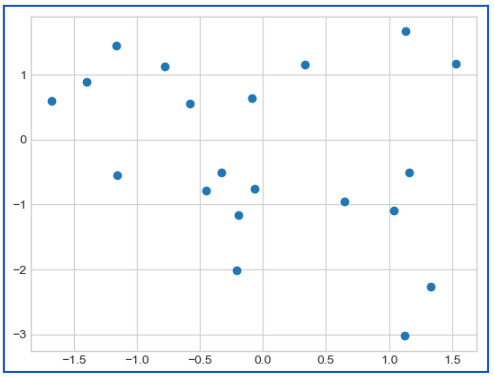


- Line plots
  - Used for numeric data.
  - Used to show trends.
  - Compares two or more different variables over time.
  - Could be used to make predictions.
  - **matplotlib**: `ax.plot(x, y)`
  - **seaborn**: `sns.lineplot(data_frame, x, y)`

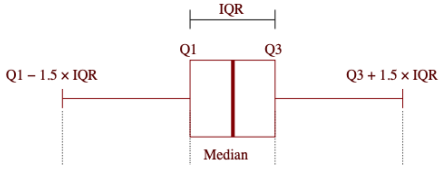
- Bar plots
  - Used for nominal or ordinal categories.
  - Compares data amongst different categories.
  - Horizontal bar charts should be preferred over vertical bar charts when we have many categories.
  - Types of bar charts: Simple, Grouped, and Stacked.
  - **matplotlib**:
    - `ax.bar(x, y)`
    - `ax.barh(x, y) # horizontal bar chart`
  - **seaborn**:
    - `sns.countplot(data_frame, x)`
    - `sns.countplot(data_frame, y) # horizontal bar chart`
    - `sns.barplot(data_frame, x, y)`



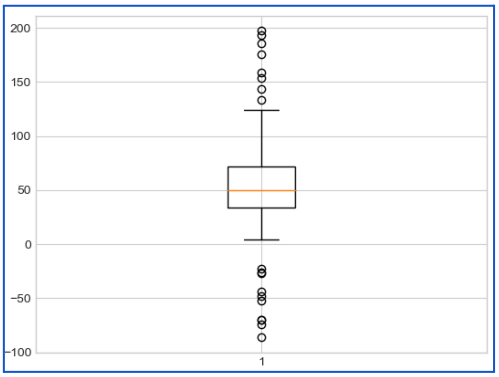
- Scatter plots
  - Used to visualise relation between two numeric variables.
  - Used to visualise correlation in a large data set.
  - Predicts behaviour of dependent variable based on the measure of the independent variable.
  - **matplotlib**:
    - `ax.scatter(x, y)`
    - `plt.plot(x, y, 'o') # plt.plot should be preferred over plt.scatter for large datasets.`
  - **seaborn**: `sns.scatterplot(data_frame, x, y)`



- Box plots/ Whisker plot
  - Statistical graph used on sets of numerical data.
  - Shows the minimum, first quartile, median, third quartile, maximum, and outliers.



  - Used to compare data from different categories.
  - **matplotlib**: `ax.boxplot(data)`
  - **seaborn**: `sns.boxplot(data_frame, x, y)`



- Histograms
  - Used for continuous data.
  - Displays the frequency distribution (shape).
  - Summarises large data sets graphically.
  - Compares multiple distributions.
  - **matplotlib**: `plt.hist(data)`
  - **seaborn**:
    - `sns.histplot(data_frame, x)`
    - `sns.histplot(data_frame, y) # horizontal histogram`