Cheatsheet - Matplotlib Charts

This kernel is all about matplotlib charts -- the basic python data visualization tool.

Please Upvote my kernel and keep it in your favourite section if you think it is helpful.

In the last few months, I was working extensively on plots and to be honest it is hard to remember each and everything and I am tired of looking back and forth in my notebooks for the reference. This hectic process motivates me to prepare this notebook not only for me but also others. Now I dont have to look at different notebooks for the reference only this one enough. I am making this notebook specially copy paste purpose and for beginners to learn matplotlib.

I used different datasets such as titanic, heart-disease, iris, pima-indians-diabetes, heart-failure-clinical data and pokemon datasets available on kaggle. All this datasets are used for visulization purpose only, I not trying to say or interpret any conclusion here.

Enjoy! 😄

Table of Content

Introduction

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. With its extensive functionality, Matplotlib allows data scientists and analysts to represent data in a wide variety of formats, making it a powerful tool for exploratory data analysis, presentation, and communication of findings.

In this Kaggle notebook, we'll delve into the diverse range of charts and plots that Matplotlib offers. From basic line plots to intricate 3D visualizations, Matplotlib provides the flexibility to create stunning graphics tailored to your data and analytical needs. We'll categorize these visualizations into distinct groups to facilitate understanding and organization.

Libraries

In [ ]:
```
# Import dependencies

import numpy as np
```

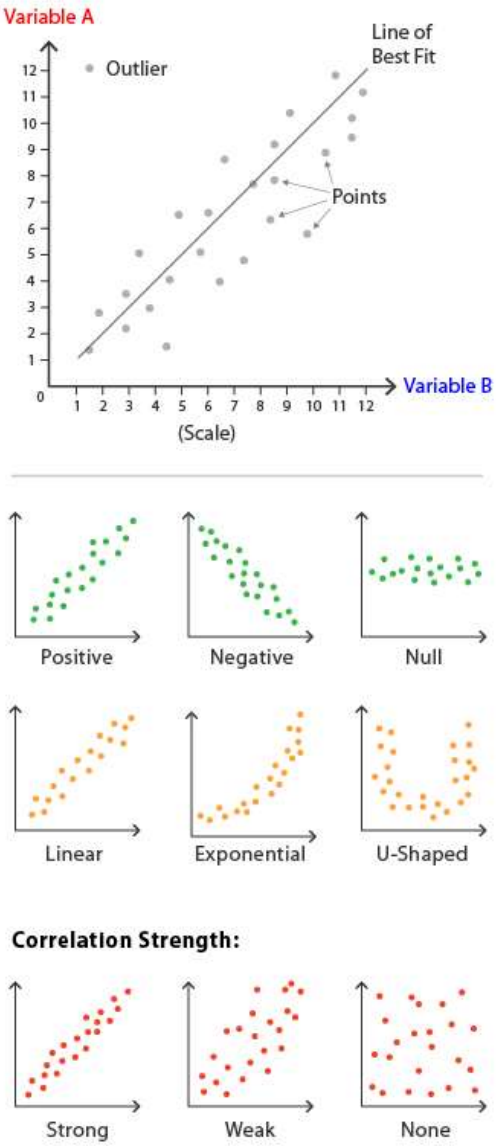
```
import pandas as pd

import matplotlib.pyplot as plt
```

Load Data

In [ ]:
```
titanic=pd.read_csv('../input/titanic/train.csv')
heart_diseases=pd.read_csv('../input/heart-disease-uci/heart.csv')
iris=pd.read_csv('../input/iris/Iris.csv')
diabetes=pd.read_csv('../input/pima-indians-diabetes-database/diabetes.csv')
pokemon=pd.read_csv('../input/pokemon/Pokemon.csv')
heart_failure=pd.read_csv('../input/heart-failure-clinical-data/heart_failure_clinical_records_dataset.csv')
corona_virus = pd.read_csv('../input/novel-corona-virus-2019-dataset/covid_19_data.csv')
latest_covid_cases = pd.read_csv('../input/novel-covid19-dataset/cases_country.csv')
```

Scatter Plot

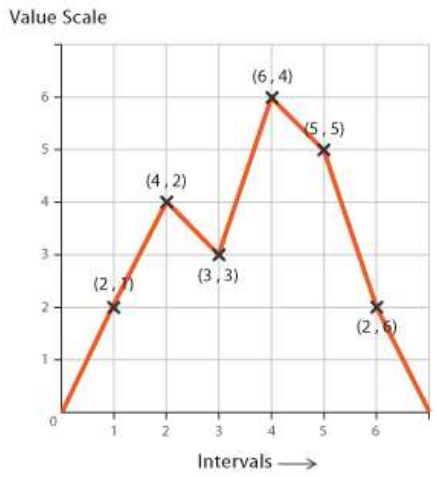
A Scatterplot displays the value of 2 sets of data on 2 dimensions. Each dot represents an observation. The position on the X (horizontal) and Y (vertical) axis represents the values of the 2 variables. It is really useful to study the relationship between both variables. It is common to provide even more information using colors or shapes (to show groups, or a third variable). It is also possible to map another variable to the size of each dot, what makes a bubble plot. If you have many dots and struggle with overplotting, consider using 2D density plot.



Source: DataVizCatalogue

In [ ]:
```
plt.plot(iris['PetalLengthCm'], iris['PetalWidthCm'], linestyle='none', marker='o', color='b')
plt.show()
```

Line Plot

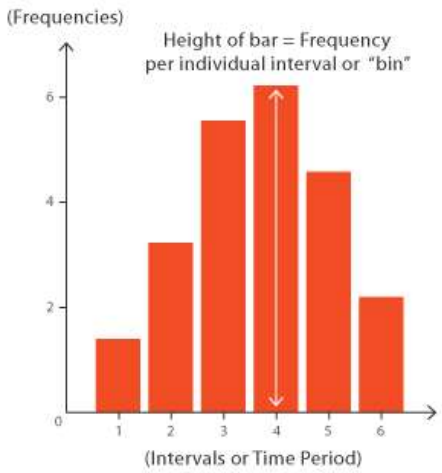


Source: DataVizCatalogue

In [ ]:
```
x = np.linspace(0, 20, 1000)
y = np.cos(x)

plt.plot(x,y, color='b', linestyle='--')
plt.show()
```

Histogram

An histogram is an accurate graphical representation of the distribution of numerical data. It takes as input one numerical variable only. The variable is cut into several bins, and the number of observation per bin is represented by the height of the bar.

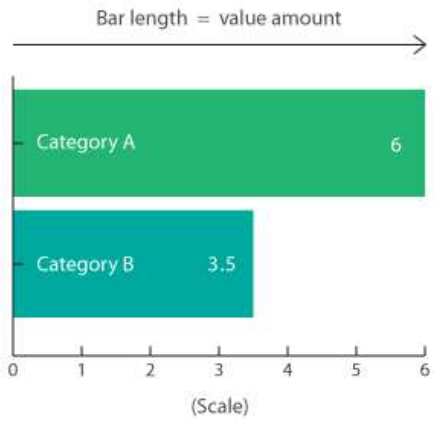


Source: DataVizCatalogue

In [ ]:
```
plt.hist(heart_diseases['age'])
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age distribution.')
```

Bar Plot

A barplot (or barchart) is one of the most common type of plot. It shows the relationship between a numerical variable and a categorical variable. For example, you can display the height of several individuals using bar chart.



Source: DataVizCatalogue

In [ ]:
```
bar_data = pokemon['Type 1'].value_counts().reset_index()
bar_data['err'] = pokemon['Type 1'].value_counts().std()
```

In [ ]:
```
plt.bar(bar_data['index'], bar_data['Type 1'])
plt.xticks(rotation=90)
plt.xlabel('Ability')
plt.ylabel('Count')
plt.title("Abilities of Pokemons")
```

Horizontal Bar Plot

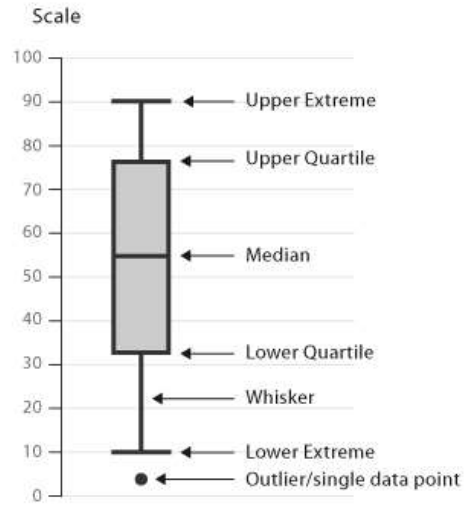
In [ ]:
```
plt.figure(figsize=(10,6))
plt.barh(bar_data['index'], bar_data['Type 1'])
plt.xlabel('Ability')
plt.ylabel('Count')
plt.title("Abilities of Pokemons")
```

Error Bar Plot

In [ ]:
```
plt.bar(bar_data['index'], bar_data['Type 1'], yerr=bar_data['err'])
plt.xticks(rotation=90)
plt.xlabel('Ability')
plt.ylabel('Count')
plt.title("Abilities of Pokemons")
```

Box Plot

A Box and Whisker Plot (or Box Plot) is a convenient way of visually displaying the data distribution through their quartiles.



Source: DataVizCatalogue

Simple Box Plot

In [ ]:
```
plt.figure(figsize=(10,6))
plt.boxplot(heart_diseases['chol'])
plt.ylabel('Value')
plt.title("Simple Box Plot")
```

Multiple Box Plot

In [ ]:
```
plt.figure(figsize=(10,6))
plt.boxplot([diabetes['Age'], diabetes['BMI']])
plt.ylabel('Value')
plt.title("Box Plot")
```

Adding Xticks to Box Plot

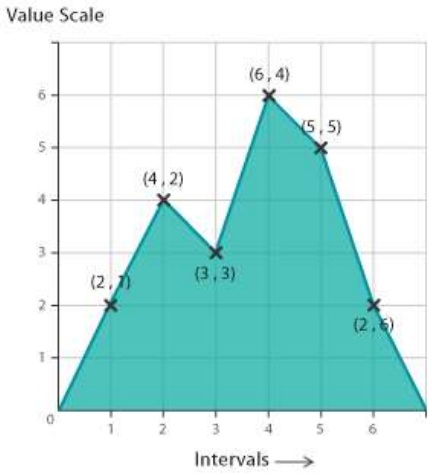
In [ ]:
```
plt.figure(figsize=(10,6))
plt.boxplot([diabetes['Age'], diabetes['BMI'], diabetes['BloodPressure'], diabetes['Glucose']])
plt.xticks([1, 2, 3, 4], ['Age', 'BMI', 'BloodPressure', 'Glucose'])
plt.ylabel('Value')
plt.title("Box Plot")
```

Horizontal Box Plot

In [ ]:
```
plt.figure(figsize=(10,6))
plt.boxplot([diabetes['Age'], diabetes['BMI'], diabetes['BloodPressure'], diabetes['Glucose']], vert=False)
plt.yticks([1, 2, 3, 4], ['Age', 'BMI', 'BloodPressure', 'Glucose'])
plt.xlabel('Value')
plt.title("Box Plot")
```

Area Plot

An area chart is really similar to a line chart, except that the area between the x axis and the line is filled in with color or shading. It represents the evolution of a numerical variable following another numerical variable.



Source: DataVizCatalogue

```
In [ ]: temp = corona_virus.groupby('ObservationDate')['Confirmed'].sum().reset_index()
```

```
In [ ]: fig = plt.figure(figsize=(20,12))
plt.fill_between(temp['ObservationDate'][:30], temp['Confirmed'][:30], color='lightblue')
plt.xticks(rotation=90)
plt.show()
```

Stacked Area Chart

```
In [ ]: india = corona_virus[corona_virus['Country/Region']=='India'].groupby('ObservationDate')['Confirmed'].sum().reset_index()
us = corona_virus[corona_virus['Country/Region']=='US'].groupby('ObservationDate')['Confirmed'].sum().reset_index()
brazil = corona_virus[corona_virus['Country/Region']=='Brazil'].groupby('ObservationDate')['Confirmed'].sum().reset_index()
```

```
In [ ]: fig = plt.figure(figsize=(20,12))
plt.stackplot(temp['ObservationDate'][-50:], us['Confirmed'][-50:],india['Confirmed'][-50:],
              brazil['Confirmed'][-50:], labels=['US', 'India', 'Brazil'])
plt.xticks(rotation=90)
plt.legend(loc='upper left')

plt.show()
```

Lollipop Plot

A lollipop plot is an hybrid between a scatter plot and a barplot. It shows the relationship between a numerical variable and another variable, numerical OR categorical.

```
In [ ]: temp = latest_covid_cases.sort_values('Confirmed', ascending= False)[['Country_Region','Confirmed']]
```

```
In [ ]: plt.stem(temp['Country_Region'][:10], temp['Confirmed'][:10])
plt.xticks(rotation=90)
plt.show()
```

Horizontal Lollipop

```
In [ ]: plt.hlines(y=temp['Country_Region'][:10][::-1], xmin=0, xmax=temp['Confirmed'][:10][::-1], color='skyblue')
plt.plot(temp['Confirmed'][:10][::-1], temp['Country_Region'][:10][::-1], 'D')
plt.xticks(rotation=90)
plt.show()
```

Don't forget to upvote if you like it!.

If you have any doubt reagrding any part of the notebook, feel free to comment your doubt in the comment box.

Thank you!!

Work in Progress... ⏳