Introduction to Data Visualization

Overview of Data Visualization

What will Learn

- Explain what data visualization is and why it's important.
- Describe the uses for data visualization. Explore the best practices for data visualization.

What is Data Visualization?

Data visualization is the graphical representation of data and information. It involves the process of creating visual representations of data.

Why build visuals?

- **Easily Understand Data**

Data visualization helps us to easily understand complex datasets that might be difficult to comprehend in their raw form.

It can highlight patterns, trends, and relationships that might not be immediately apparent from looking at the data.

- **Communicate Insights**

Visualization enables us to communicate insights and findings to stakeholders in a more compelling and understandable way.

It allows us to tell a story with data, making it easier for stakeholders to understand and act on the information presented.

- **Identify trends and data pattern**

Visualization helps us identify trends and data patterns that might otherwise be difficult to see.

We can gain new insights and make informed decisions by examining data over time or across different dimensions.

- **Present data for better understanding and interpretation**

Presenting data in a more easily understood and interpreted way enables us to identify opportunities, spot potential problems and make more informed choices.

EXAMPLES:

Companies can gain insights into market trends, financial performance, and customer behavior.

Healthcare professionals can identify patterns in patient data and develop targeted treatment plants.

Visualization helps analyze student performance and inform instructional decisions leading to better academic outcomes.

Visualization helps governments make informed decisions and communicate data to the public.

Visualization helps scientists and researchers analyze complex data, develop new insights, and share findings effectively.

Visualization helps workers and the entertainment industry determined ratings and ultimately signals to the creators what the audience wants.

Best Practices

To create effective visualizations;

It's important to follow best practices that can help ensure that the data is accurately represented and the message is clearly communicated.

- **Choice Right Visualization Type**

One key best practice is choosing the appropriate visualization type for the presented data.

Different types of data require different visualization techniques. Choose the right one to communicate that data effectively

- **Keep it simple**

Keep the visualization simple and easy to read.

A line chart shows trends over time. Well, a bar chart is ideal for comparing values.

- **Use clear labelling and formatting**

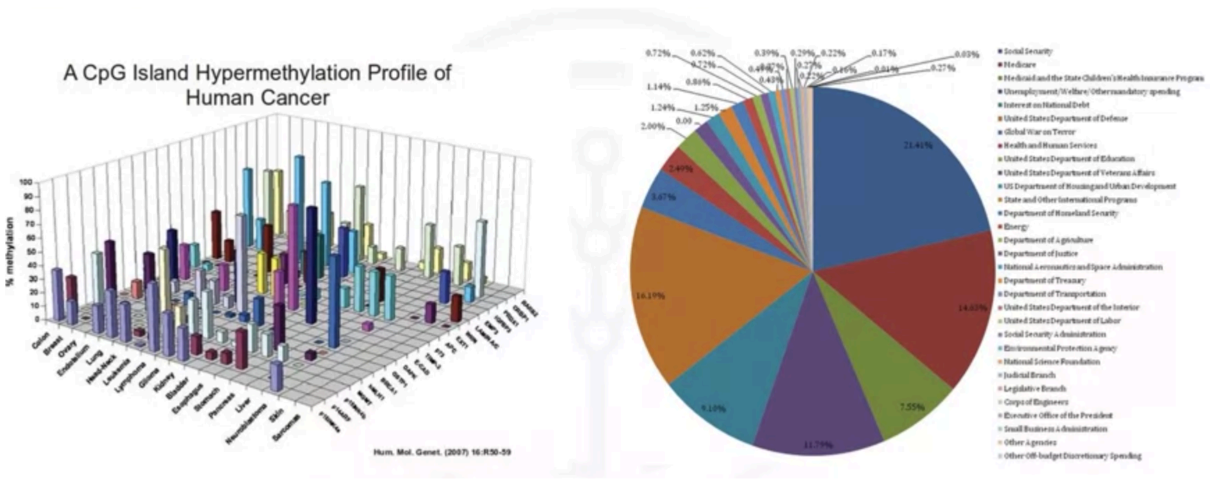
To ensure that the data is accurately represented, it's also important to label the axis clearly and provide context for the data being presented. Include a clear title, summarizing the main message and a legend explaining the meaning of any symbols or colors used.

- **Keep visualization focused**

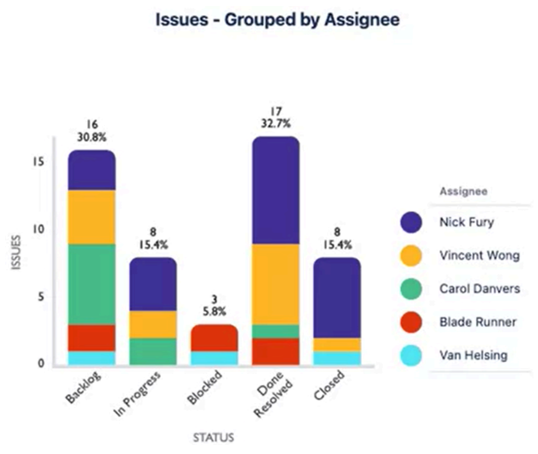
    Including unnecessary graphics or information, can confuse the audience and distract them from the main message.
    Keep visualization focused on the main point and use only the necessary data and labels.

- **Consider the audience**

    It's important to consider the audience when creating the visualization. Who will view the visualization? What do they need to know?
    The visualization should be tailored to their needs and understanding.



In these charts, there are a lot of variables and the information is more difficult to understand. This is an example of bad data visualization.



This chart represents a team's board. The stacked bars depict the percentage of the work assigned to the individual users and their work progress. Colors are distinguishable and consistent. The legend shows the individual color's meanings. The chart provides more clarity with the values and percentages on top of each bar.

When creating visual , always remember;

Less is more effective
Less is more attractive
Less is more impactful

Check out the Darkhorse Analytics website for examples of good and bad plots.

Types of Plots

What will Learn

- Explore different types of plots available for visualizing data.
- Identify the characteristics and appropriate use cases for each type of plot.

The different types of plots,
line plot, bar plot, scatter plot, box plot, histogram.

DATA TYPES

Before we get into some common visualisation techniques, we need to understand the different types of statistical data. In Data Science, we tend to work with two main categories of variables: categorical and continuous.

Categorical Variables
Categorical variables are also known as discrete or qualitative variables. Examples of categorical variables are race, sex, age group, and educational level. According to Laerd (n.d.), these variables can be further divided into the following categories: nominal, ordinal or dichotomous.

- **Nominal variables** have two or more categories, which do not have a specific or predefined order. For example, properties could be classified as houses, condos or bungalows. Therefore, the variable that holds the property type is a nominal variable. For example, the state or province a person lives in would also be a nominal variable.

- **Dichotomous variables** are nominal variables which have only two categories or levels. For example, we could use a dichotomous variable to describe whether a person is a pensioner or not. In this case, the categories would be “True” or “False”.

- **Ordinal variables** are nominal variables, but the categories can be ordered or ranked. For example, if you were asked to rate your satisfaction with this course, your responses could be “Completely satisfied”, “Mostly satisfied”, “A little dissatisfied”, or “Very dissatisfied”.

Continuous variables
A continuous variable can take on infinitely many uncountable numerical values, including integers and floating points (decimals). They are also known as quantitative variables. Continuous variables can be further categorised as either interval or ratio variables:

- **Interval variables** are "variables for which their central characteristic is that they can be measured along a continuum and they have a numerical value (for example, temperature measured in degrees Celsius or Fahrenheit)" (Laerd, n.d.).

- **Ratio variables** are interval variables, but a 0 value means there is none of that particular variable. For example, "weight" is a ratio variable because if a variable measuring the sugar stock at a bakery was equal to 0 kgs it would mean there is no sugar in stock. Temperature (measured in degrees Celsius) would not be a ratio variable because 0°C does not mean that there is no temperature.

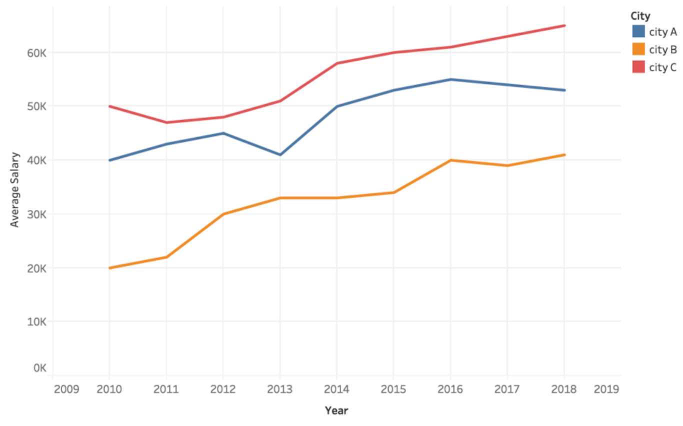
Line Plot

Use Cases:

- Line plots display trends, such as stock market fluctuations or temperature changes over time.
- They compare datasets with a continuous independent variable like age or time.
- Line plots illustrate cause and effect relationships, such as sales revenue changes based on marketing budget.
- They also visualize continuous data, like height measurements over time.

Line plots can be misleading, if the scales on the axes are not carefully chosen to reflect the data accurately.

Best used for data that is:
○ Linear or continuous
○ Time series



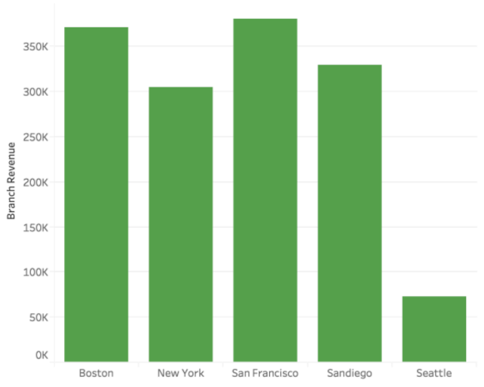
Bar Plot

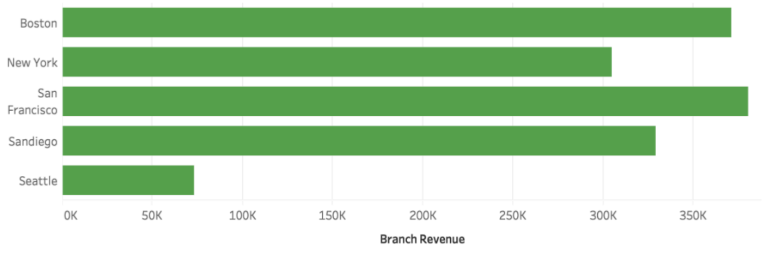
Displays data using rectangular bars, where the height or length of the bars represents the magnitude of the data.The bars can be oriented either vertically or horizontally. A vertically oriented bar chart is often referred to as a vertical bar chart or a column chart.

Use Cases:

- Bar plots are ideal for comparing different categories or groups.
- They excel with discrete data, like comparing sales revenue by product.
- They show how different categories contribute to the whole and rankings, such as sales percentage or budget allocation.
- Bar plots can visualize data that you can easily rank, like displaying the bestselling books in the market.
- Their simplicity and interpretability, make them favoured for data visualization.

Line plots can be misleading, if Inaccurate bar choices or axis scales





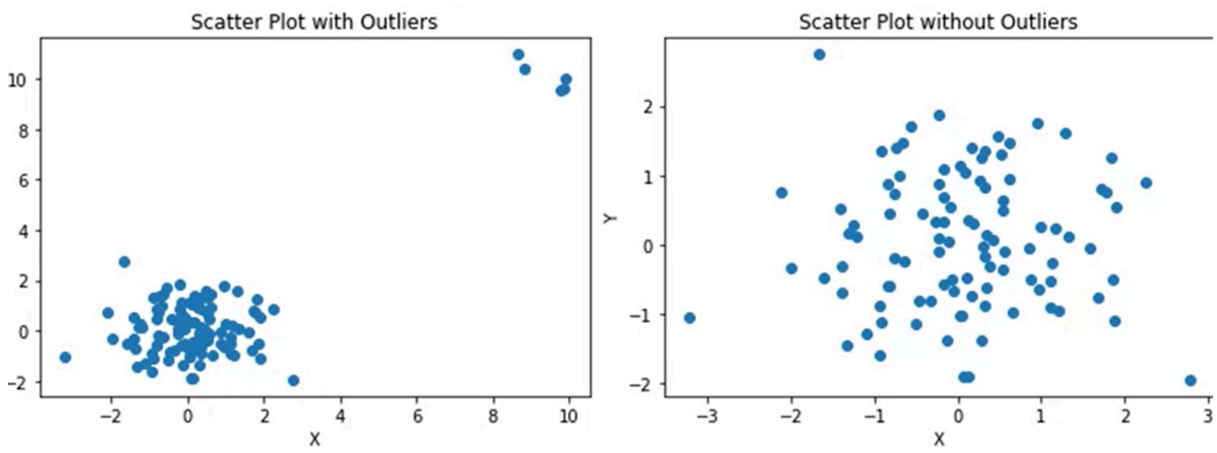
Best used for data that is:
○ Discrete
○ Categorical
Things to keep in mind:
○ Doesn’t work well with too many categories of data (too many bars are hard to look at). For instance, if you’re showing rainfall in mm for each day of a year, you would end up with 365 bars!
○ You can colour-code the categories for patterns to stand out.

Scatter Plot

A scatter plot, is a type of plot that presents values for two variables for a set of data using Cartesian coordinates. The data points are displayed as a collection of points, where one variable's value determines the position on the horizontal axis and the other variable's value determines the position on the vertical axis.

Use Cases:

- Examining the relationship between two continuous variables like temperature and energy consumption.
- Investigating patterns or trends in data, such as house prices versus size.
- Detecting outliers or unusual observations such as outliers in test scores or abnormal stock behavior.
- Visualizing data with many observations to identify clusters or groups.
- Exploring complex data

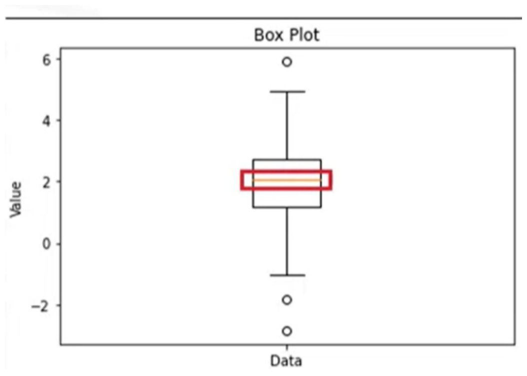


Outliers significantly impact interpretation, requiring consideration of their inclusion or exclusion. On the screen, the data is plotted with and without outliers. With outliers, it shows two clusters, but removing outliers makes the remaining data more visible. Proper outlier handling, enhances accuracy and meaningful insights in scatter plots.

Best used for data that is characterised by:
○ Numerical values
○ Two variables (X, Y)
Not useful for:
○ Categorical data
○ Time series

Box Plot

A box plot, also known as a box and whisker plot, is a type of plot that displays the distribution of a data set, along with key statistical measures. It consists of a box, representing the interquartile range, IQR, a line inside the box representing the median and lines whiskers extending from the box to indicate the range of the data, excluding outliers. Outliers may be represented as individual data points beyond the whiskers.



Use Cases:

- While comparing the distribution of a continuous variable across different categories or groups. For example, employees' salaries across departments.
- Examining spread and skewness of data sets, visualizing quartiles and outliers.
- Identifying and analyzing potential outliers within a data set.
- Visualizing summary statistics, median, quartiles, and range in a concise and informative manner.
- Comparing distributions of multiple variables in datasets side by side.

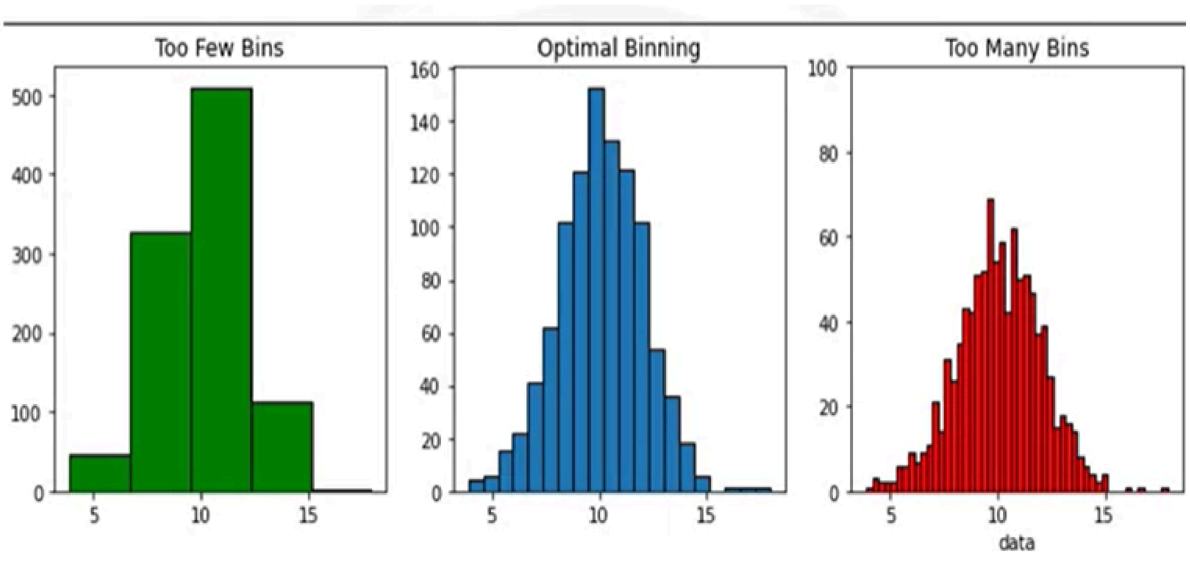
Remember, box plots provide valuable information about outliers, such as their presence and extent. Ignoring or mishandling outliers, can distort the interpretation of the data and mask important insights, same as with scatter plots.

Histograms

A histogram, is a graphical representation of the distribution of a data set, showing the frequency or relative frequency of values within specific intervals. It consists of bars, where the height represents the data count in each interval.

Use Cases:

- Histograms offer valuable insights into data distribution, outliers, skewness and variability.
- They visually depict the shape of the data, whether it's symmetric skewed or bimodal.
- Skewness can be assessed by examining the histogram's shape.
- Histograms also showcase data variability, allowing you to observe concentrations, gaps and clusters that reveal patterns or subgroups.



Apart from issues with scale and inadequate labeling, be careful while choosing the bins to create a histogram. We have generated a random data set and created three histograms, with three different binning options. Too few bins, five represented in green color, an optimal number of bins, 20 represented in blue and too many bins 50, the yellow one. By comparing these histograms, you can observe how the choice of binning affects the representation of
the data distribution and results in a misleading representation of the data. Selecting too few or too many bins, can oversimplify or overcomplicate the distribution, respectively.

RECAP

Line plots, capture trends and changes over time, allowing us to see patterns and fluctuations.

Bar plots, compare categories or groups, providing a visual comparison of their values.

Scatter plots, explore relationships between variables, helping us identify correlations or Trends.

Box plots, display the distribution of data, showcasing the median, quartiles and outliers.

Histograms, illustrate the distribution of data within specific intervals, allowing us to understand its shape and concentration.

Plot Libraries

Data visualization is a vital tool for gaining insights, and effectively communicating complex information. In the world of data visualization, there are several plot libraries in Python that provide unique features and capabilities.

Some of the popular libraries

- Matplotlib
- Seaborn
- Folium
- Plotly
- PyWaffle

By harnessing the power of these plot libraries, you can unlock insights from your data and effectively communicate your findings.

What is Data Visualization?

Data visualization is a vital tool for gaining insights, and effectively communicating complex information. In the world of data visualization, there are several plot libraries in Python that provide unique features and Capabilities.

Matplotlib, is a plotting library that offers a wide range of plotting capabilities.

Pandas, is a plotting library that provides integrated plotting functionalities for data analysis.

Seaborne, is a specialized library for statistical visualizations, offering attractive default aesthetics and color palettes.

Folium, It allows you to create interactive and customizable maps. Whether it's choroplasmaps point maps or heat maps, folium provides the tools to visually represent your geospatial data. It's a popular choice for geospatial data visualization and analysis in Python.

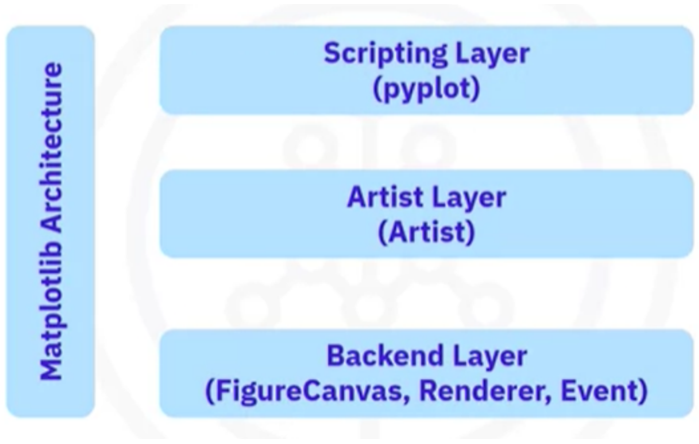
Plotly, It offers highly interactive plots and dashboards. With Plotly, you can create line plots, scatter plots, bar charts, pie charts, 3D plots, and Choropleth maps, to name a few. Here are some of the key features of Plotly. Its Plotly dash framework allows you to build interactive dashboards with rich visualizations and controls. Since Plotly is web based, it enables the rendering and viewing of plots in web browsers. This makes it convenient for sharing visualizations online, embedding them in web applications or dashboards, and collaborating with others.

PyWaffle, enables you to visualize proportional representation using squares or rectangles.If you want to visualize categorical data using Waffle charts, PyWaffle is a simple yet effective library. With PyWaffle, you can create waffle charts, square pie charts, donut charts, and many more types of plots by providing a unique way to represent proportions.

Introduction to Matplotlib

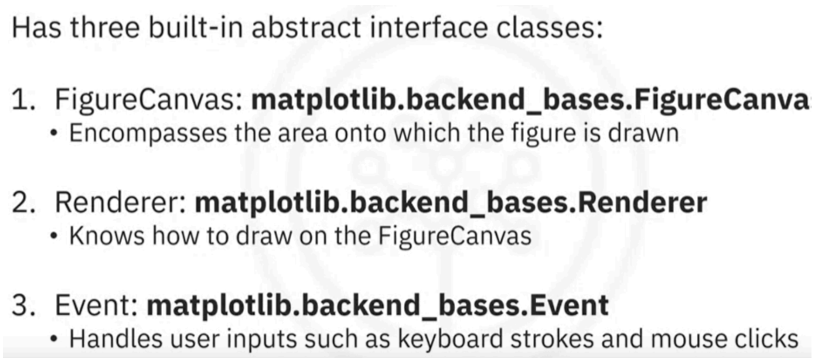
Matplotlib Architecture

Matplotlib architecture is composed of three main layers, the backend layer, the artist layer, and the scripting layer.



The backend layer has three built-in abstract interface classes.

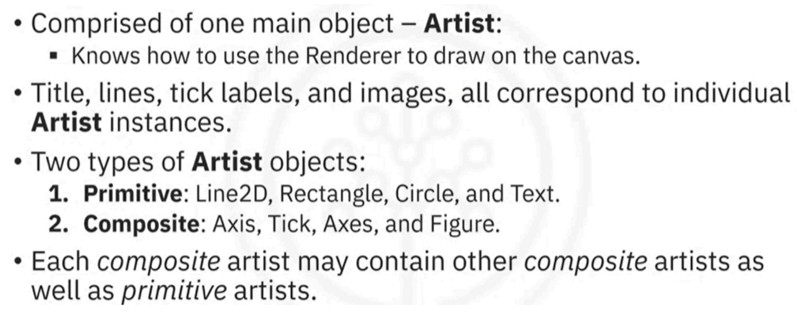
- **FigureCanvas** defines end encompasses the area on which the figure is drawn.
- **Renderer**, an instance of the renderer class knows how to draw on figure canvas.
- **Event,** handles user inputs such as keyboard strokes and mouse clicks.



Has three built-in abstract interface classes:

1. FigureCanvas: **matplotlib.backend_bases.FigureCanva**
   - Encompasses the area onto which the figure is drawn
2. Renderer: **matplotlib.backend_bases.Renderer**
   - Knows how to draw on the FigureCanvas
3. Event: **matplotlib.backend_bases.Event**
   - Handles user inputs such as keyboard strokes and mouse clicks

The artist layer is the object that knows how to take the renderer and put ink on the Canvas. Everything you see in a Matplotlib figure is an artist instance. The title, the lines, the tick labels, the images, and so on, all correspond to an individual artist. There are two types of artist objects.

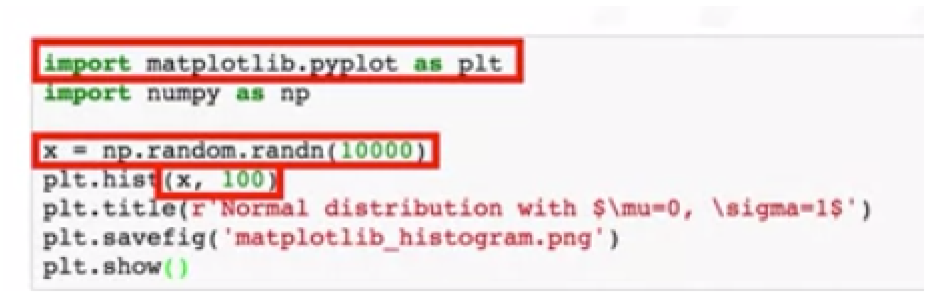


- Comprised of one main object – **Artist**:
  - Knows how to use the Renderer to draw on the canvas.
- Title, lines, tick labels, and images, all correspond to individual **Artist** instances.
- Two types of **Artist** objects:
  1. **Primitive**: Line2D, Rectangle, Circle, and Text.
  2. **Composite**: Axis, Tick, Axes, and Figure.
- Each *composite* artist may contain other *composite* artists as well as *primitive* artists.

- **Primitive:** Line2D,Rectangle, Circle, and Text.
- **Composite:** Axis, Tick, Axes, and Figure. The top-level Matplotlib object that contains and manages all the elements in each graphic is the figure artist.

The most important composite artist is the axis because it's where most of the Matplotlib API plotting methods are defined, including methods to create and manipulate the ticks, the axis lines, the grid, and the plot background. Each composite artists may contain other composite artists as well as primitive artists. A figure artists can have an axis artist, a rectangle, and a text artist. The artist layer is syntactically heavy. Programmers work directly with the backend and artists layers as they offer greater convenience while integrating Matplotlib with application servers.

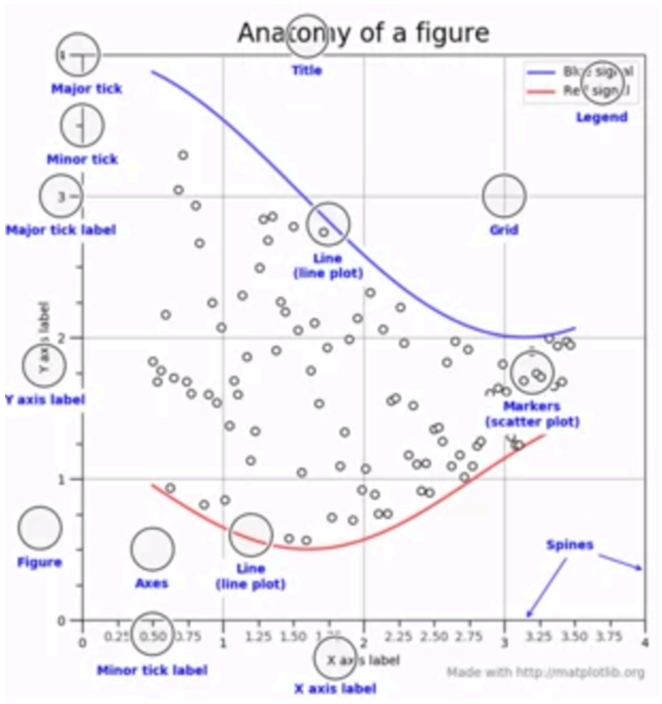
Matplotlib scripting layer is the Matplotlib.When it comes to the daily tasks of scientists involving data visualization or exploratory interactions, the scripting layer known as Pyplot works better. Pyplot interface, which automatically defines a Canvas and a figure artist instance and connects them.



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

x = np.random.randn(10000)
plt.hist(x, 100)
plt.title(r'Normal distribution with $\mu=0, \sigma=1$')
plt.savefig('matplotlib_histogram.png')
plt.show()
```

Let's generate a histogram of 10,000 random numbers with the scripting layer. First we import the Pyplot interface and you can see all the methods associated with creating the histogram and other art objects and manipulating them. Whether the hist method or showing the figure is part of the Pyplot interface. Notice the use of Numpy's random module with random.randn for creating random floats from the Pyplot hist method is called, hist creates a sequence of rectangle artists for each histogram bar and adds them to the axes container. To the hist function, we have passed the variable X containing an array of 10,000 random numbers and 100, which means creating 100 bins. The result is a histogram. It's simple to work in the scripting layer.

Components of a Plot

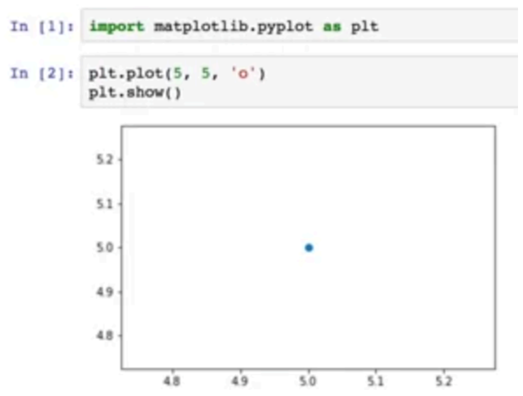


The main component is **the window or Canvas** containing the plot or subplots. Matplotlib figure as a Canvas and axis represents an individual plot within a figure, you must first create two-axis parts to make two plots on one figure, the axis provides scales and tick marks for plotting the data. The actual data being plotted is represented as points or markers on the plot like the axis.

A good plot must have **a title** to provide a summary or explanation of the plot. Likewise, **labels** describe data being plotted on each axis. You may like to include a **legend** to explain the meaning of different elements or data series and applaud a grid to help visually aligned data points and aid and reading values from the plot and annotations to provide supplemental information or explanations about specific data points are regions in the plot. You can choose symbols for individual data points, colors, and styles.

Basic Plotting with Matplotlib

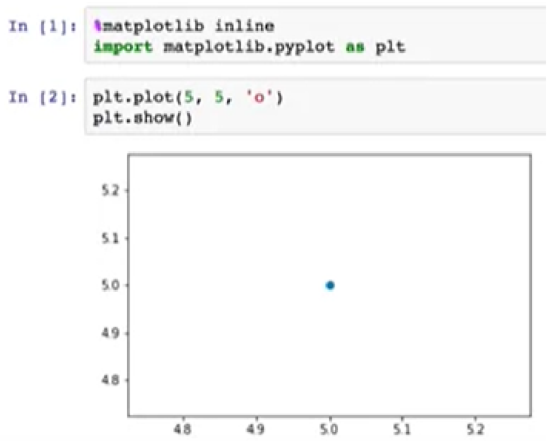
Plot Function



```
In [1]: import matplotlib.pyplot as plt

In [2]: plt.plot(5, 5, 'o')
        plt.show()
```

First, import the Scripting interface as plt and let's plot a circular mark at the position 5-5. Notice how the plot was generated within the browser and not in a separate window.

Matplotlib Backends- Inline

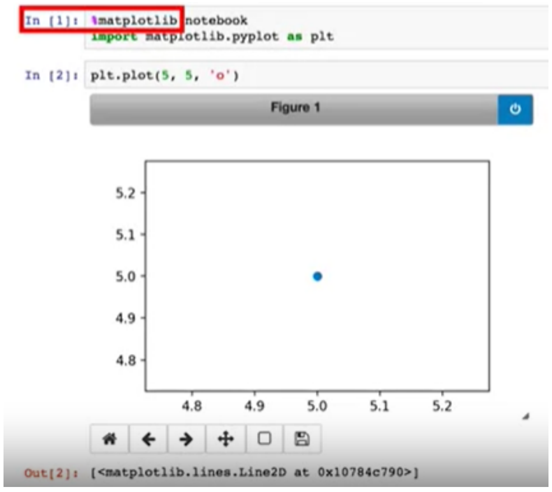


```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt

In [2]: plt.plot(5, 5, 'o')
        plt.show()
```

If the plot gets generated in the new popup window, then you can enforce generating plots into the browser using what's called the magic function, quote, %matplotlib, end quote, and you pass in inline as the back end.

Matplotlib has a number of different backends available. One limitation of this backend is that you cannot modify a figure once it's rendered. So after rendering the figure, there is no way for us to add, for example, a figure title or labels to its axis. We will need to generate a new plot and add a title and the axis labels before calling the show function. A backend that overcomes this limitation is the notebook backend.

Matplotlib Backends- Notebook



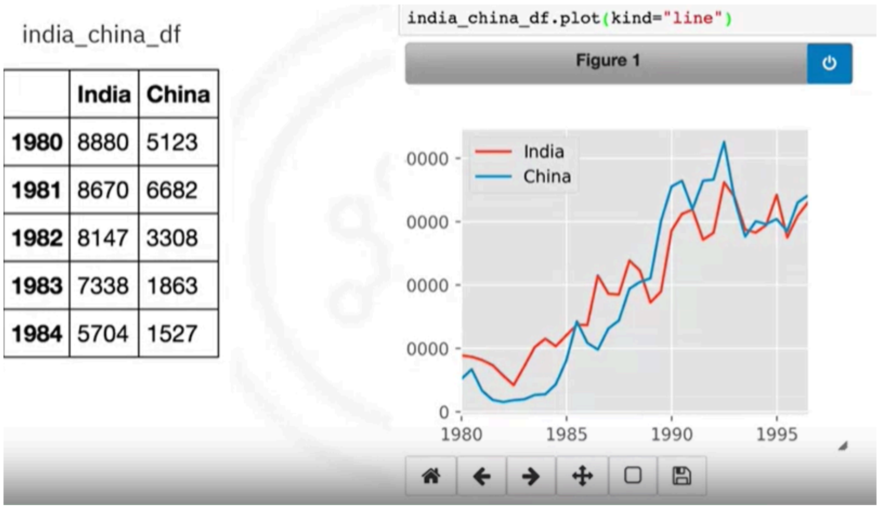
```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt

In [2]: plt.plot(5, 5, 'o')
```

```
Out[2]: [<matplotlib.lines.Line2D at 0x10784c790>]
```

With the notebook backend in place, if a plt function is called, it applies them to the active figure if it exists. If a figure does not exist, it renders a new figure. So when we call the plt.plot function to plot a circular mark at position 5-5, the backend checks if an active figure exists.

Since no active figure exists, it generates a figure and adds a circular mark to position 5-5. And what is beautiful about this backend is that now we can easily add a title or labels to the axis after the plot was rendered without the need of regenerating the figure.

Matplotlib Backends- Pandas

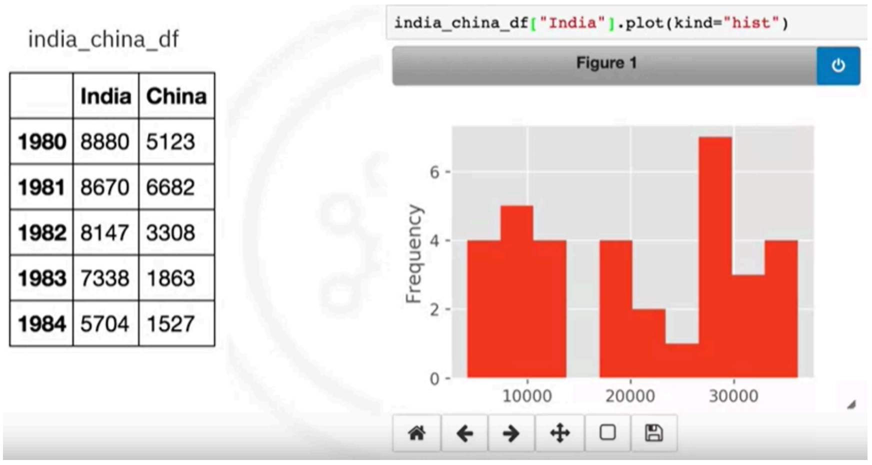


india_china_df

	India	China
1980	8880	5123
1981	8670	6682
1982	8147	3308
1983	7338	1863
1984	5704	1527

```
india_china_df.plot(kind="line")
```

Great about Matplotlib is that Pandas also has a built in implementation of it. Therefore, plotting in Pandas is as simple as calling the Plot function on a given Pandas series or Pandas data frame. So say we have a data frame of the number of immigrants to Canada from India and China for the years 1980 to 1996. And we're interested in generating a line plot of this data. All we have to do is call the plot, the function on this data frame, which we have called India_China underscore_df and pass in kind = line. And there you have it, a line plot of the data in the data frame.



india_china_df

	India	China
1980	8880	5123
1981	8670	6682
1982	8147	3308
1983	7338	1863
1984	5704	1527

```
india_china_df["India"].plot(kind="hist")
```

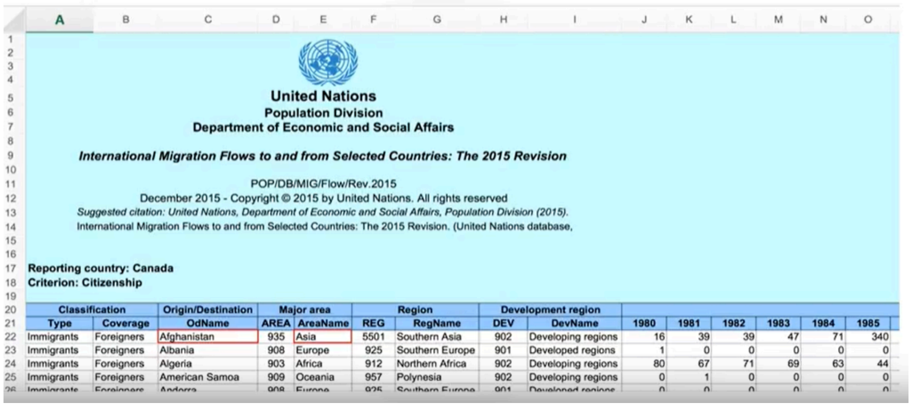
Plotting a histogram of the data is not any different. So say we would like to plot a histogram of the India column in our data frame. All we have to do is call the plot function on that column and pass in kind as hist for histogram. And there you have it a histogram of the number of Indian immigrants to Canada from 1980 to 1996.

Dataset on Immigration to Canada

What will Learn

- Understand the data set to be used in this course for data visualization.
- Import data with pandas as a DataFrame in your program.
- Process data to make it suitable for plotting.

We will need to import the pandas library, as well as the **openpyxl library**, which is required to extract data from Excel spreadsheets files.



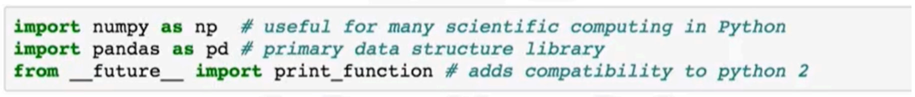
United Nations
Population Division
Department of Economic and Social Affairs

International Migration Flows to and from Selected Countries: The 2015 Revision

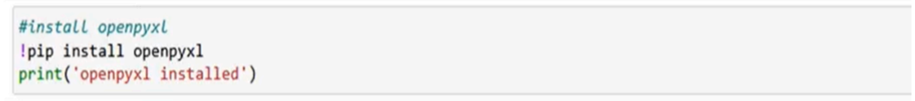
POP/DB/MIG/Flow/Rev.2015
December 2015 - Copyright © 2015 by United Nations. All rights reserved
Suggested citation: United Nations, Department of Economic and Social Affairs, Population Division (2015).
International Migration Flows to and from Selected Countries: The 2015 Revision. (United Nations database,

Reporting country: Canada
Criterion: Citizenship

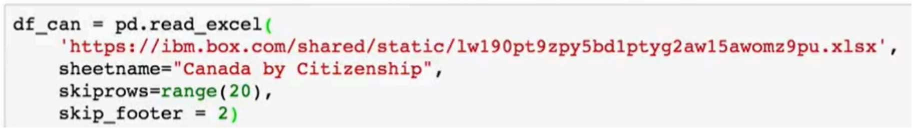
Classification		Origin/Destination	Major area		Region		Development region							
Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	1982	1983	1984	1985
Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	39	39	47	71	340
Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	0	0	0	0	0
Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	67	71	69	63	44
Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	1	0	0	0	0
Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	0	0	0	0	0



```
import numpy as np  # useful for many scientific computing in Python
import pandas as pd # primary data structure library
from __future__ import print_function # adds compatibility to python 2
```

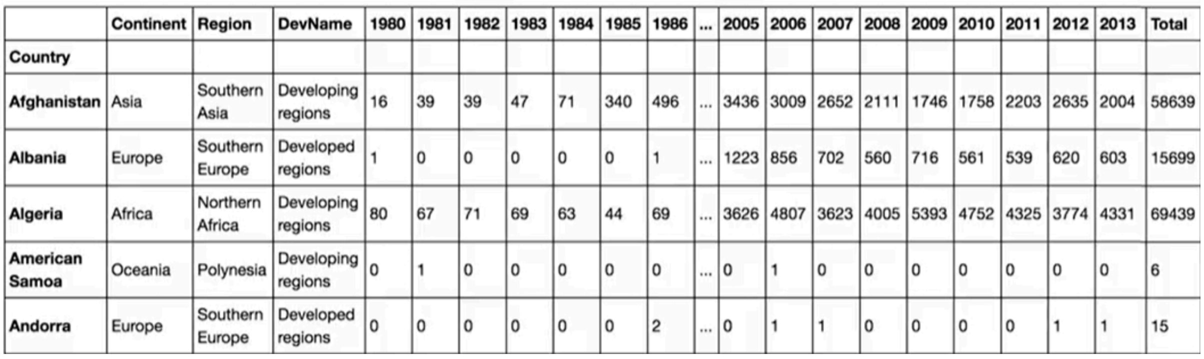


```
#install openpyxl
!pip install openpyxl
print('openpyxl installed')
```



```
df_can = pd.read_excel(
    'https://ibm.box.com/shared/static/lw190pt9zpy5bd1ptyg2aw15awomz9pu.xlsx',
    sheetname="Canada by Citizenship",
    skiprows=range(20),
    skip_footer = 2)
```

Notice how we're skipping the first 20 rows to read only the data corresponding to each country. If you want to confirm that you have imported your data correctly, you can always use the head function to display the first five rows in the DataFrame.



	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Country																					
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	561	539	620	603	15699
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	6
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	0	0	1	1	15

Now, let's process the data frame so that the country name becomes the index of each row. This should make querying specific countries easier. Also, let's add an extra column which represents the total immigration for each country from 1980 to 2013. So, for Afghanistan it's 58,639, and for Albania it's 15,699, and so on. Now let's name our DataFrame **df_canada**. In this video, you learned that, the population division of the United Nations compiled mmigration data pertaining to 45 countries. The UN data on immigration to Canada shows data related to the number of people who migrated. In order to start creating different types of plots of the data, you will need to import the data into a Pandas data frame.

Line Plots

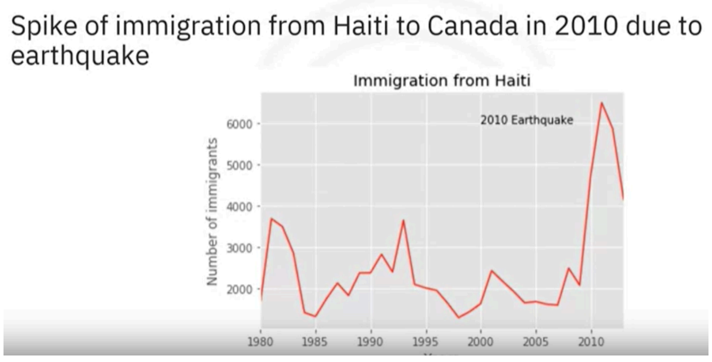
What will Learn

- Describe line plot and its function.
- Determine when to use a line plot
- Create a line plot from data in the dataset.

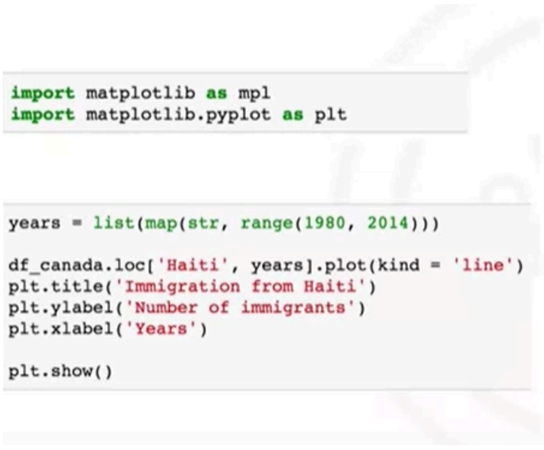
Line plot, it's a plot that displays information as a series of data points connected by straight lines.

When to use a line plot

- Line plots are useful for visualizing trends and changes over time, making them a popular choice for time-series data, such as changes in stock prices, website traffic, or temperature fluctuations.
- You can also use a line plot to show relationships between two variables.
- Also be used to compare multiple data series on one chart.
- Line plots can also effectively highlight sudden changes or anomalies in data.

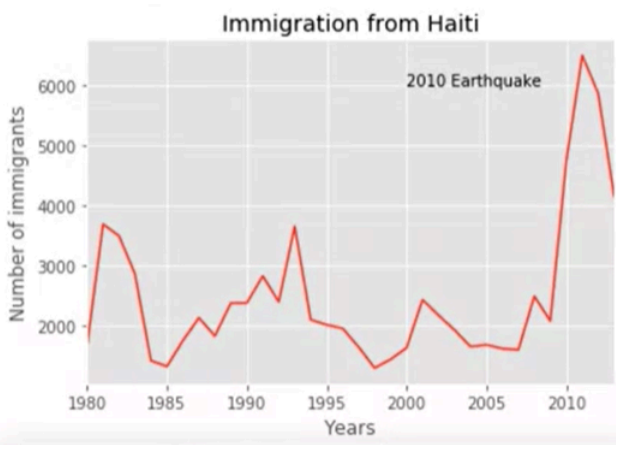


As an example, from our dataset, we can generate a line plot to see the trend of immigrants from Haiti to Canada. Based on this line plot, we see that there is a spike of immigration from Haiti to Canada in 2010. We can then research for justifications of obvious anomalies or changes from a quick Google search for major events in Haiti in 2010 one would easily learn about the tragic earthquake that took place in 2010. Therefore, this influx of immigration to Canada was mainly due ton that tragic earthquake.



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

```
years = list(map(str, range(1980, 2014)))

df_canada.loc['Haiti', years].plot(kind = 'line')
plt.title('Immigration from Haiti')
plt.ylabel('Number of immigrants')
plt.xlabel('Years')

plt.show()
```



Line plot, all of which can be accessed by passing kind keyword to plot(). The full list of available plots are as follows:

bar for vertical bar plots
barh for horizontal bar plots
hist for histogram
box for boxplot
hexbin for hexbin plot
kde or density for density plots
area for area plots
pie for pie plots
scatter for scatter plots

Summary: Introduction to Data Visualization Tools

Congratulations! You have completed this module. At this point in the course, you know:

- Data visualization is the process of presenting data in a visual format, such as charts, graphs, and maps, to help people understand and analyze data easily.
- Data visualization has diverse use cases, such as in business, science, healthcare, and finance.
- It is important to follow best practices, such as selecting appropriate visualizations for the data being presented, choosing colors and fonts that are easy to read and interpret, and minimizing clutter.
- There are various types of plots commonly used in data visualization.
- Line plots capture trends and changes over time, allowing us to see patterns and fluctuations.
- Bar plots compare categories or groups, providing a visual comparison of their values.
- Scatter plots explore relationships between variables, helping us identify correlations or trends.
- Box plots display the distribution of data, showcasing the median, quartiles, and outliers.
- Histograms illustrate the distribution of data within specific intervals, allowing us to understand its shape and concentration.
- Matplotlib is a plotting library that offers a wide range of plotting capabilities.
- Pandas is a plotting library that provides Integrated plotting functionalities for data analysis.
- Seaborn is a specialized library for statistical visualizations, offering attractive default aesthetics and color palettes.
- Folium is a Python library that allows you to create interactive and customizable maps.
- Plotly is an interactive and dynamic library for data visualization that supports a wide range of plot types and interactive features.
- PyWaffle enables you to visualize proportional representation using squares or rectangles.
- Matplotlib is one of the most widely used data visualization libraries in Python.
- Matplotlib was initially developed as an EEG/ECoG visualization tool.
- Matplotlib's architecture is composed of three main layers: Backend layer, Artist layer, and the Scripting layer.
- The anatomy of a plot refers to the different components and elements that make up a visual representation of data.
- Matplotlib is a well-established data visualization library that can be integrated in different environments.
- Jupyter Notebook is an open-source web application that allows you to create and share documents.
- Matplotlib has a number of different backends available.
- You can easily include the label and title to your plot with plt.
- In order to start creating different types of plots of the data, you will need to import the data into a Pandas DataFrame.
- A line plot is a plot in the form of a series of data points connected by straight line segments.
- Line plot is one of the most basic type of chart and is common in many fields.
- You can generate a line plot by assigning "line" to 'Kind' parameter in the plot() function.