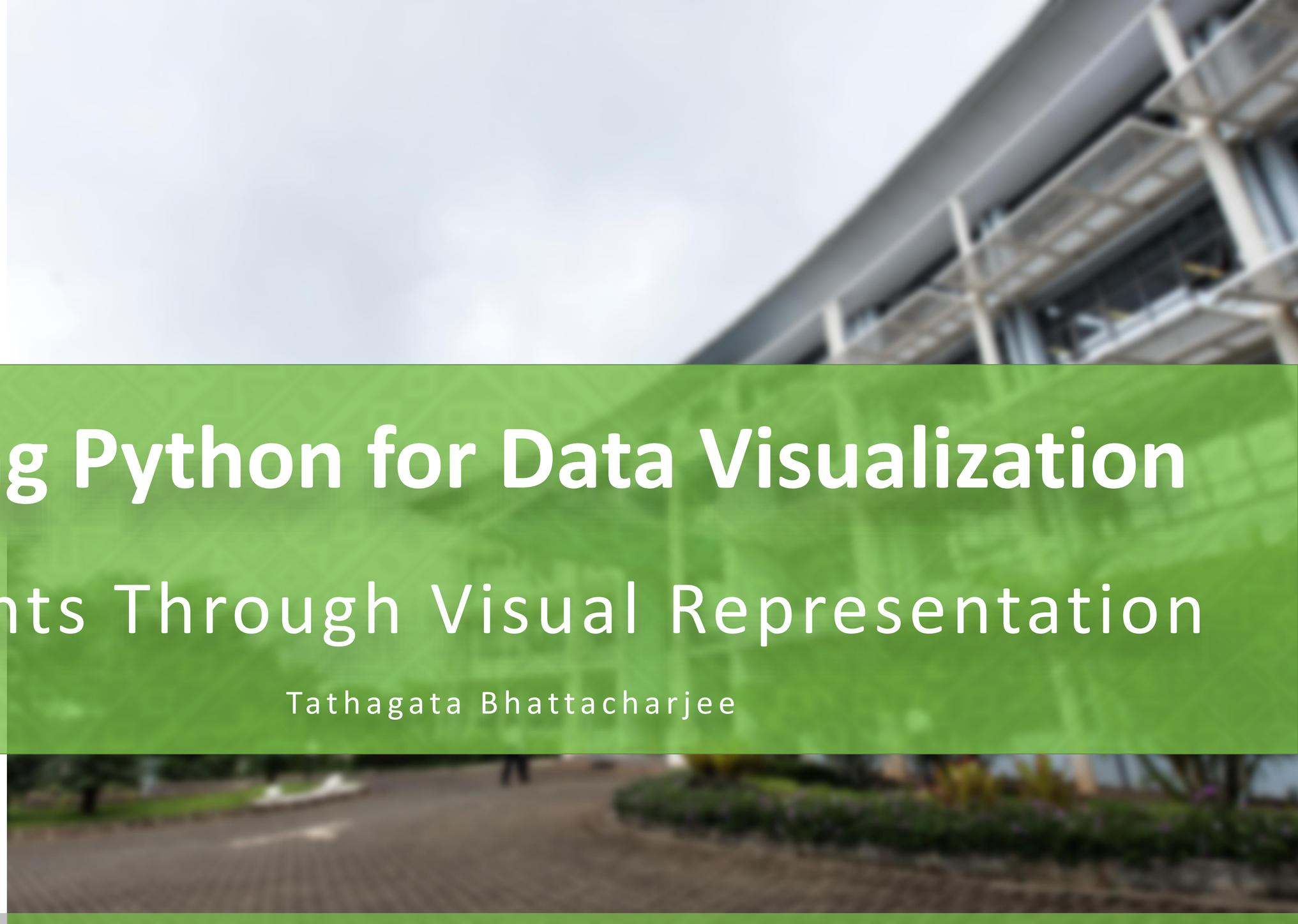


African Population and Health Research Center

Transforming lives in Africa through research.

Using Python for Data Visualization

Insights Through Visual Representation

Tathagata Bhattacharjee

Training Agenda

1

Introduction

•Brief overview of the importance of data visualization
•Introduction to Python as a tool for data visualization

2

Why Python?

- Advantages of using Python for data visualization
  - Versatility
  - Rich ecosystem of libraries (Matplotlib, Seaborn, Plotly, etc.)

3

Python Data Visualization Libraries

Overview of popular Python libraries for data visualization: Matplotlib, Seaborn, Plotly, Bokeh, Altair, Pandas Visualization

4

Getting Started with Matplotlib

- Brief introduction to Matplotlib
- Syntax overview
- Simple example plot (e.g., line plot or scatter plot)

5

Enhancing Visualizations with Seaborn

- Introduction to Seaborn
- Features and advantages compared to Matplotlib
- Example plots (e.g., maps)

6

Interactive Visualizations with Dash and Plotly

- Introduction to Plotly
- Creating interactive plots
- Example interactive visualization

Introduction

Data visualization is a field in data analysis that deals with visual representation of data

It is a critical component of data analysis, enabling an effective way to communicate inferences from data

Python is a leading language for data visualization due to its flexibility, robust libraries, and ease of use

With pictures, maps, and graphs, the human mind has an easier time processing and understanding any given data

It plays a significant role in the representation of both small and large data sets but is useful with large data sets where it is difficult to see all of the data and understand it manually

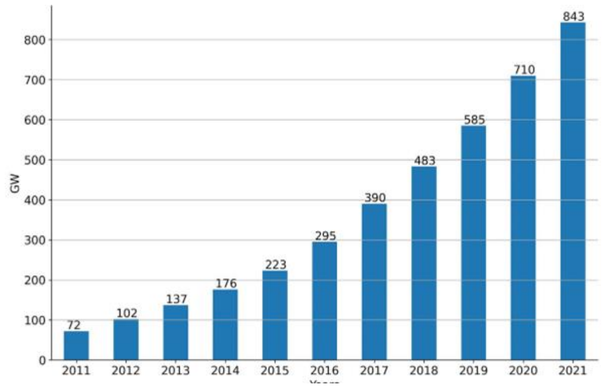
In this session, we'll explore how Python can empower us to create impactful visualizations to extract actionable insights from data.

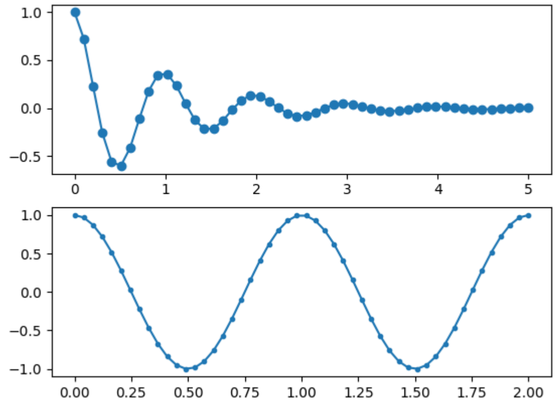
Importance of Data Visualization

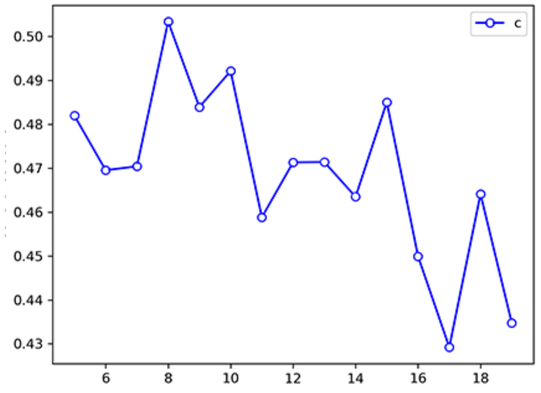
Enhanced Understanding: Visual representations help us comprehend complex data patterns and relationships more intuitively than raw numbers or text

Effective Communication: Visualizations facilitate clear and concise communication of findings to stakeholders, enabling informed decision-making

Exploratory Analysis: Visualization serves as a powerful tool for exploratory data analysis, allowing us to uncover hidden trends, outliers, and correlations.





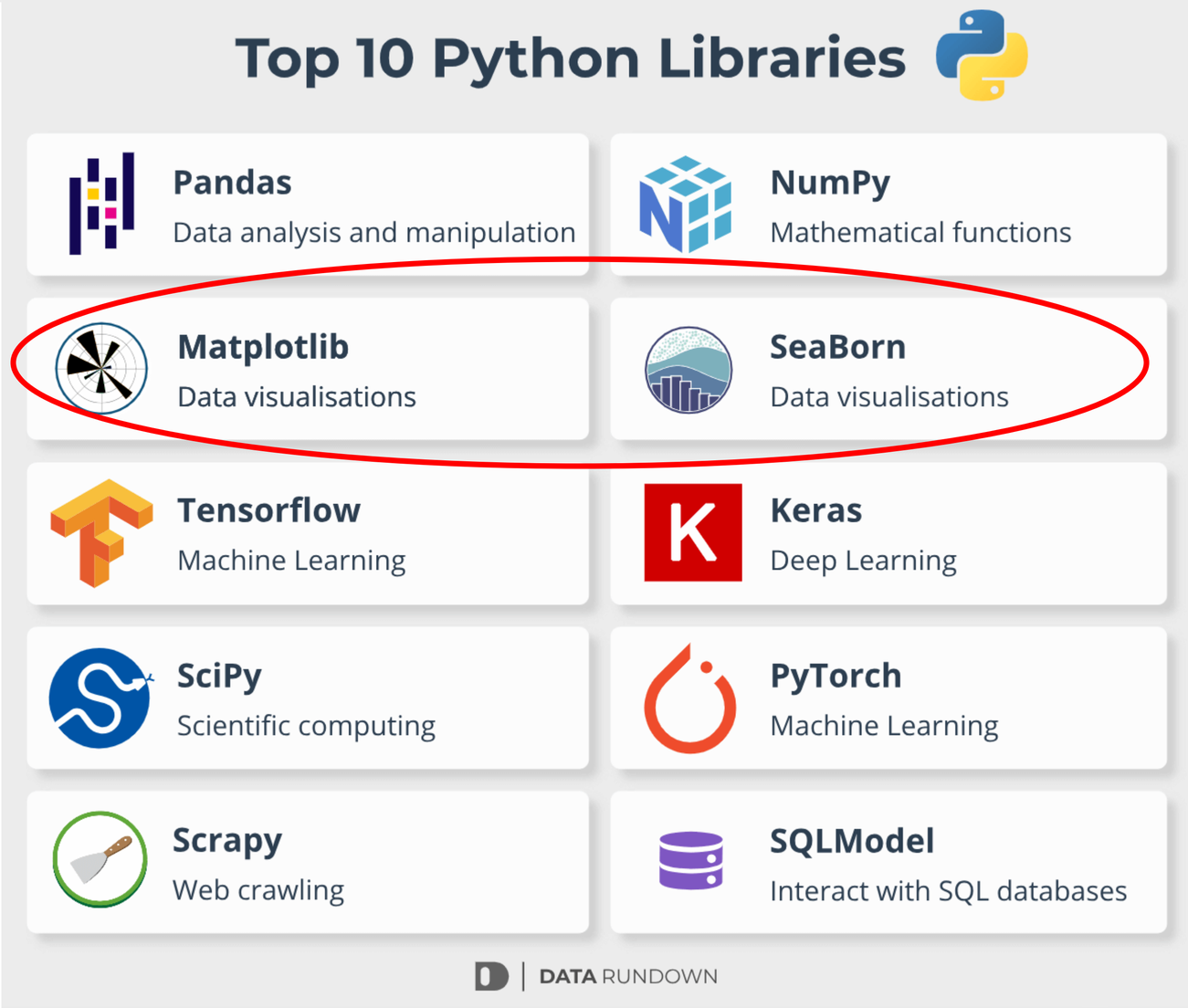


Why Python for Data Visualization?

Rich Ecosystem: Python boasts a vast ecosystem of libraries designed for data visualization, including Matplotlib, Seaborn, Plotly, and Bokeh

Flexibility: Python's flexibility allows for customization and integration of visualizations into various data analysis workflows and applications

Ease of Use: Python's intuitive syntax and extensive documentation make it accessible to both beginners and experienced data scientists, facilitating rapid development of visualizations.



Data Visualization in Python

Some of the key Python libraries and techniques for data visualization

1. Matplotlib: Foundation of Python Visualization

2. Seaborn: Statistical Data Visualization and Maps

3. Plotly: Interactive Data Visualization

Matplotlib: Foundation of Python Visualization

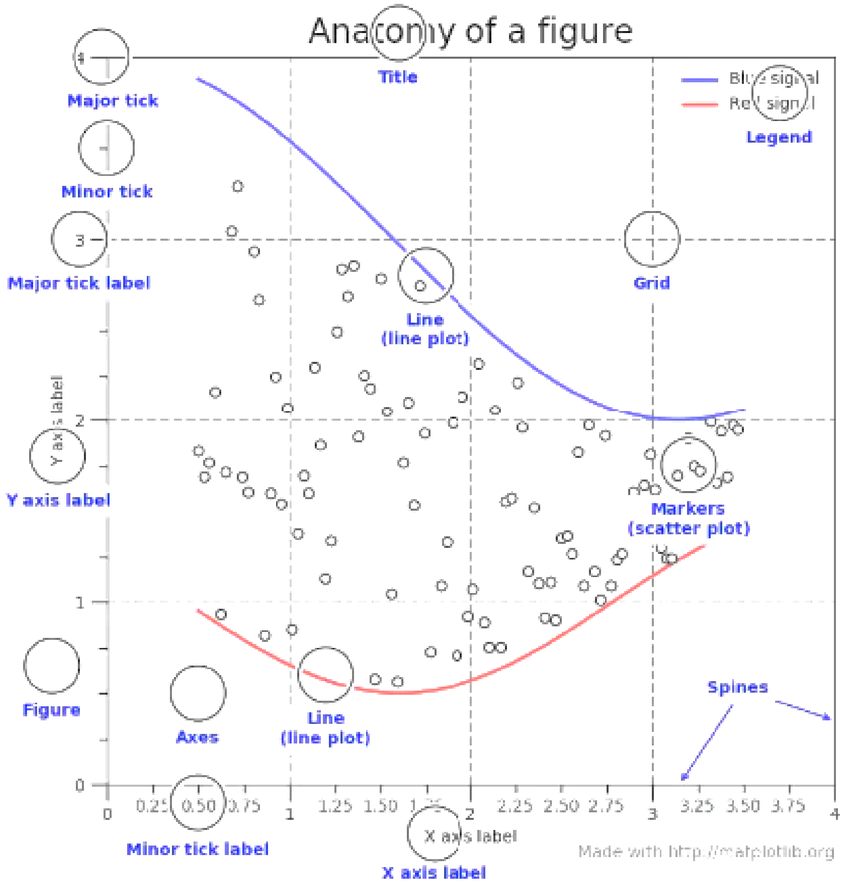
It is the foundation of Python visualization.

Overview: Matplotlib is a widely used Python library for creating static, interactive, and publication-quality visualizations

Features: Matplotlib provides a plethora of customizable 2D plotting functions for creating various types of plots, including line plots, scatter plots, bar plots, histograms, and more

Customization: Matplotlib offers extensive customization options to control every aspect of the plot, including colors, styles, labels, annotations, and axes formatting

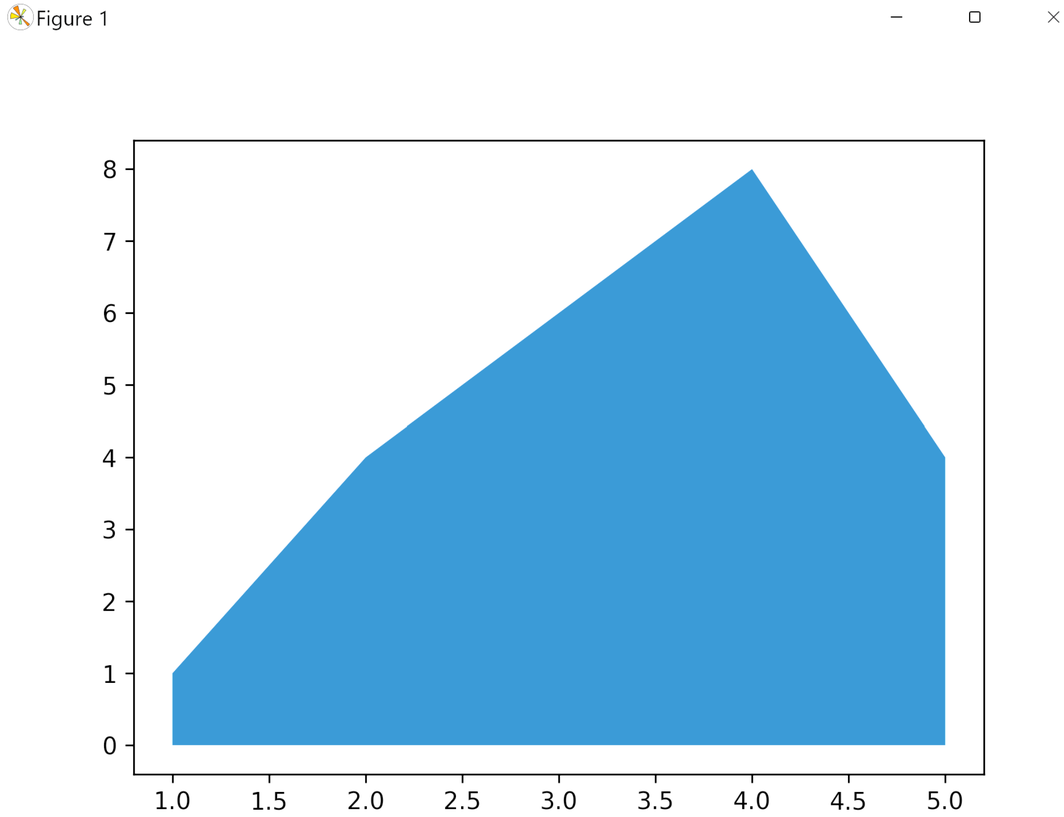
The anatomy of a **matplotlib** charts



Matplotlib: Area Plot

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

# Create data
x=range(1,6)
y=[1,4,6,8,4]

# Area plot
plt.fill_between(x, y)
plt.show()
```

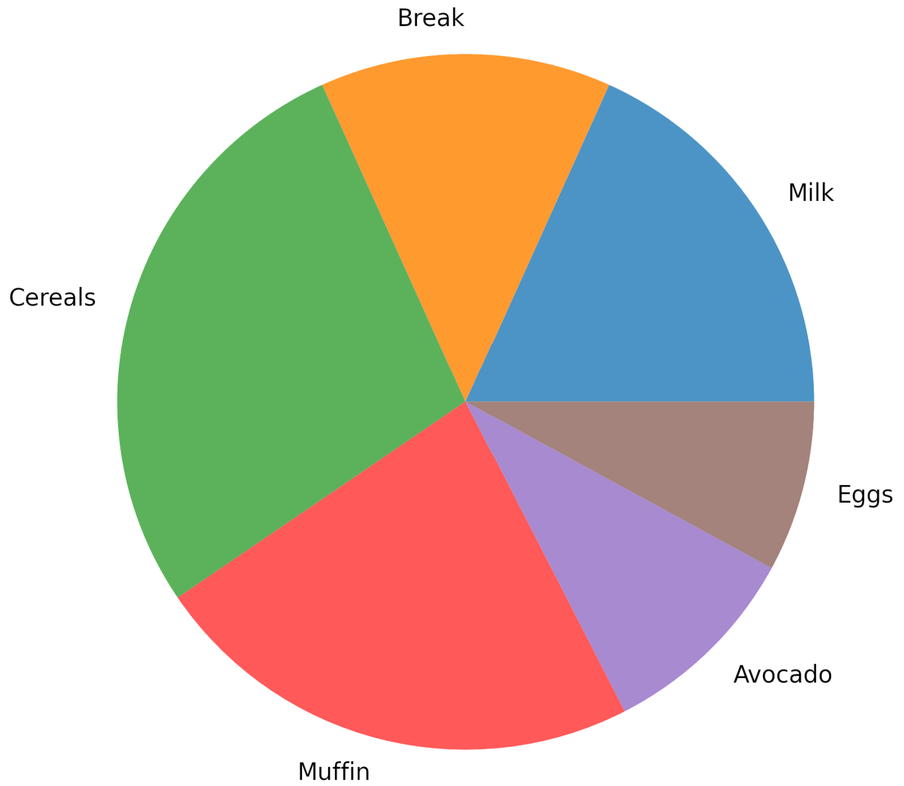
An **area chart** is a visual representation of data that utilizes both lines and filled areas to convey information. This type of chart is particularly effective in showcasing data trends and variations over a specified period or across different categories



Matplotlib: Pie Chart

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

# Creating dataset
food = ['Milk', 'Break', 'Cereals',
        'Muffin', 'Avocado', 'Eggs']

data = [23, 17, 35, 29, 12, 10]

# Creating plot
fig = plt.figure(figsize=(10, 7))
plt.pie(data, labels=food)

# show plot
plt.show()
```

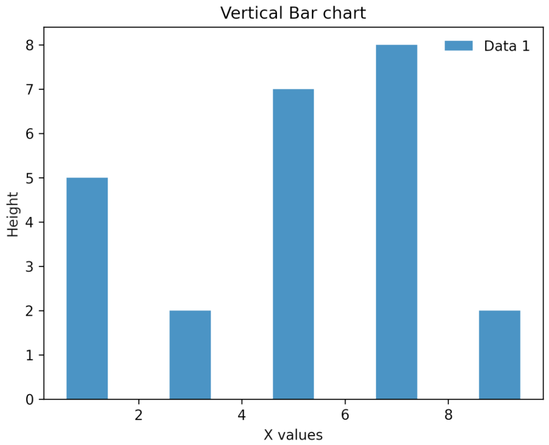
A pie chart is a pictorial representation of data in the form of a pie where the slices of the pie show the size of the data. A list of numerical variables along with categorical variables is needed to represent data in the form of a pie chart.



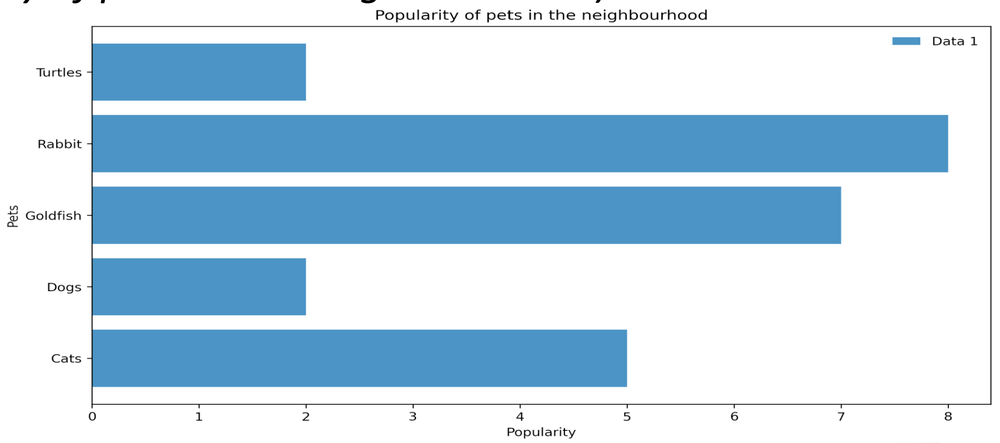
Matplotlib: Bar Chart

Bar graphs are the pictorial representation of data (generally grouped), in the form of vertical or horizontal rectangular bars, where the length of bars is proportional to the measure of data.

```
import matplotlib.pyplot as plt

# Pass the x and y cordinates of the bars to the

# function. The label argument gives a label to the data.

plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Data 1")

plt.legend()

# The following commands add labels to our figure.

plt.xlabel('X values')

plt.ylabel('Height')

plt.title('Vertical Bar chart')

plt.show()
```




```
import matplotlib.pyplot as plt
# Pass the x and y cordinates of the bars to the
# function. The label argument gives a label to the data.
plt.barh(["Cats","Dogs","Goldfish","Rabbit","Turtles"],
[5,2,7,8,2], align='center', label="Data 1")
plt.legend()

plt.ylabel('Pets')
plt.xlabel('Popularity')
plt.title('Popularity of pets in the neighbourhood')

plt.show()
```

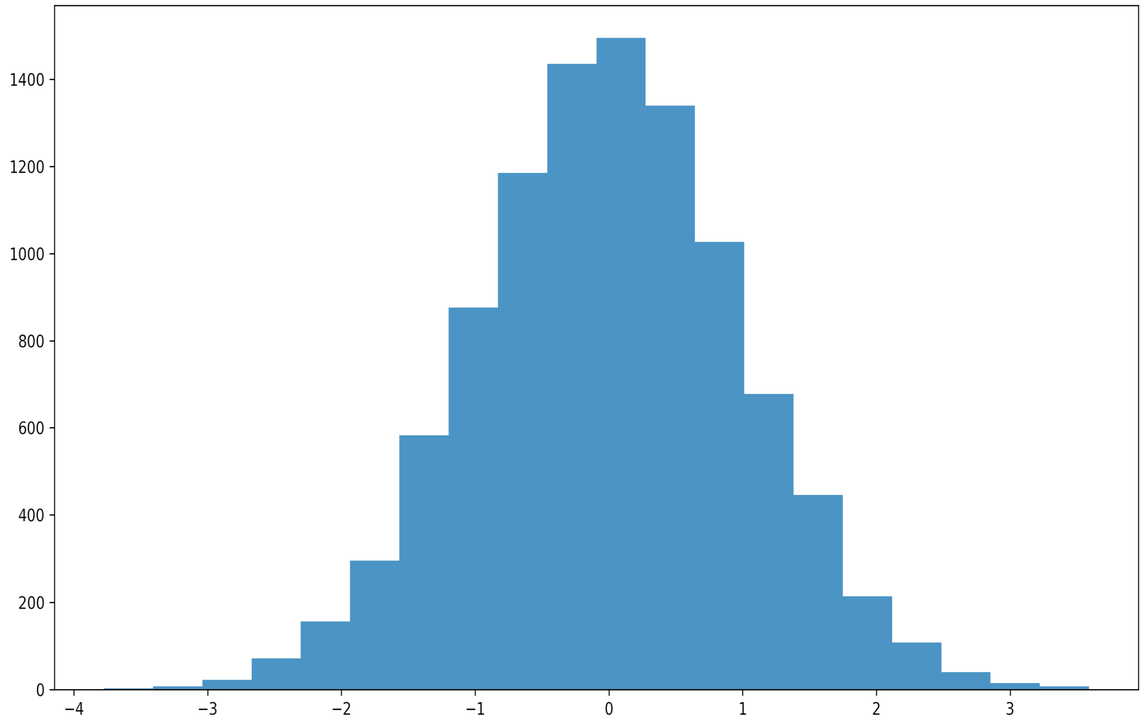


Matplotlib: Histogram

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
# Creating dataset
np.random.seed(23685752)
N_points = 10000
n_bins = 20
# Creating distribution
x = np.random.randn(N_points)
y = .8 ** x + np.random.randn(10000) + 25
# Creating histogram
fig, axs = plt.subplots(1, 1,
                        figsize=(10, 7),
                        tight_layout=True)
axs.hist(x, bins=n_bins)
# Show plot
plt.show()
```

A histogram is the most used graph to show frequency distributions. A frequency distribution shows how often each different value in a set of data occurs.

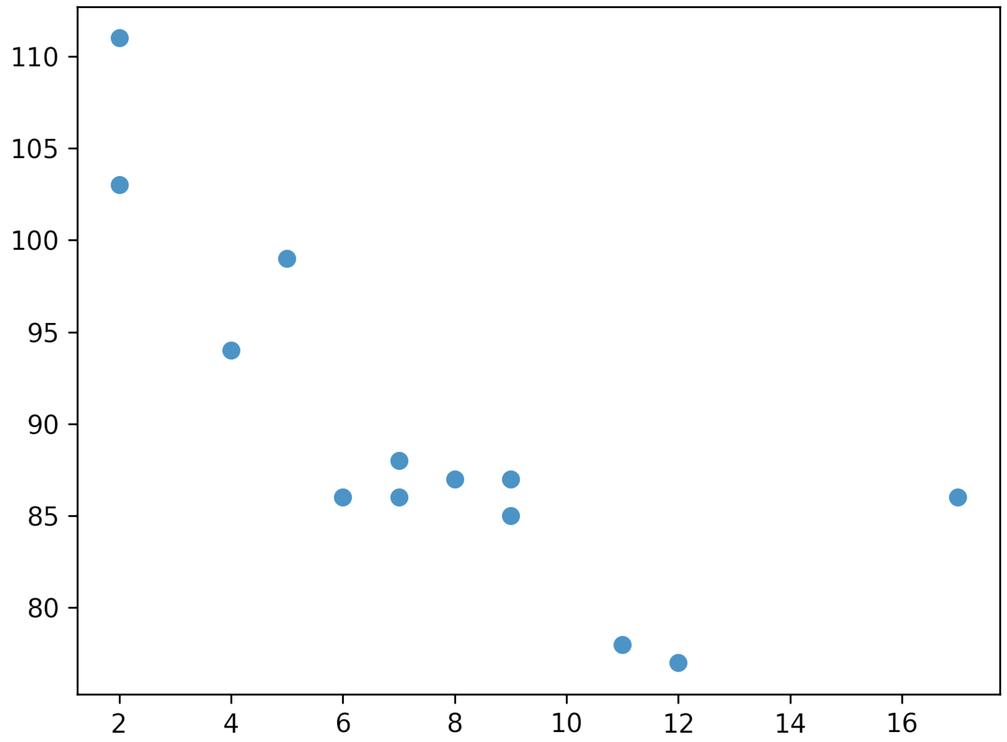
A bar graph is used to compare discrete or categorical variables in a graphical format whereas a histogram depicts the frequency distribution of variables in a dataset



Matplotlib: Scatter Plot

```
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y)
plt.show()
```

A scatter plot is a diagram where each value in the data set is represented by a dot. It is a numerical data pair, with one variable on each axis, showing their relationship



Seaborn

Built on top of Matplotlib, Seaborn is a well-known Python library for data visualization that offers a user-friendly interface for producing visually appealing and informative statistical graphics. Seaborn offers a variety of powerful tools for visualizing data, including scatter plots, line plots, bar plots, heat maps, and many more

Key benefits:

to generate attractive plots with minimal coding efforts

a range of default themes and color palettes, that can easily be customized

a range of built-in statistical functions to perform complex statistical analysis and visualizations

to create complex multi-plot visualizations

Matplotlib	Seaborn
low-level plotting library that provides a wide range of tools for creating highly customizable visualizations.	high-level interface for creating statistical graphics
highly flexible library, allowing users to create almost any type of plot they can imagine	a simpler, more intuitive interface for creating common statistical plots
provides a limited set of default styles and color palettes, requiring users to customize their plots manually to achieve a desired look	offers a range of default styles and color palettes that are optimized for different types of data and visualizations making it easy for users to create visually appealing plots with minimal customization.

Seaborn Plot types

- **Univariate** – x only (contains only one axis of information)
- **Bivariate** – x and y (contains two axis of information)
- **Trivariate** – x, y, z (contains three axis of information)

Seaborn: Commonly Used Plot Types

Scatter Plot: A scatter plot is used to visualize the relationship between two variables

Line Plot: A line plot is used to visualize the trend of a variable over time

Histogram: A histogram is used to visualize the distribution of a variable

Box Plot: A box plot is used to visualize the distribution of a variable

Violin Plot: A violin plot is similar to a box plot but provides a more detailed view of the distribution of the data

Heatmap: A heatmap is used to visualize the correlation between different variables

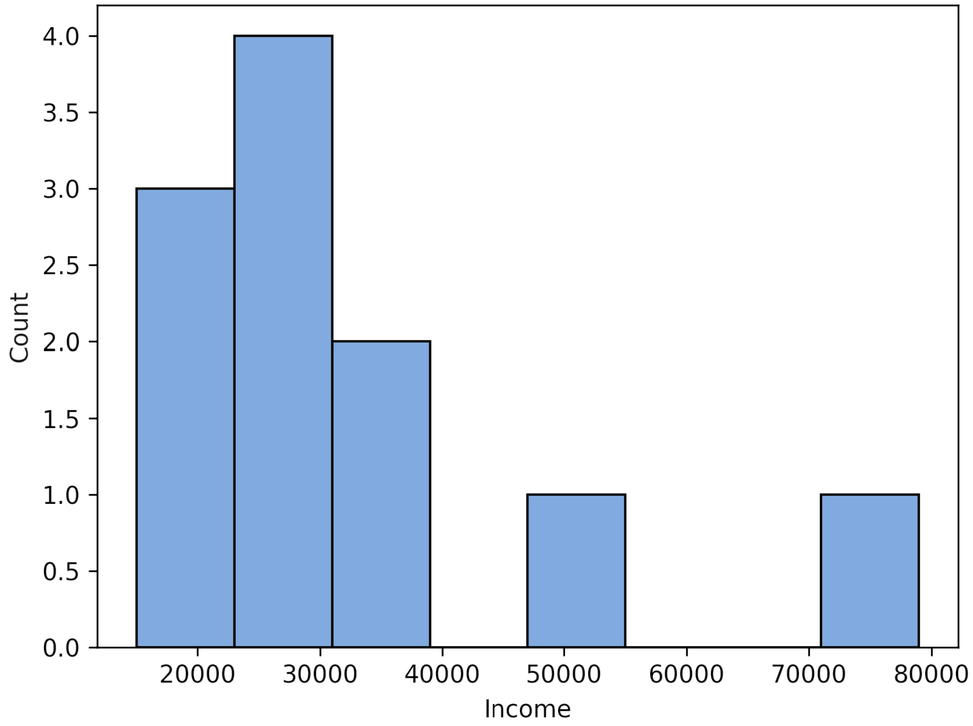
Pairplot: A pairplot is used to visualize the relationship between multiple variables

Sample Dataset

Gender	Age	Education	Income
male	42	Under-Graduate	15000
female	41	Graduate	25000
female	40	Graduate	22000
male	48	Graduate	26000
female	34	Post-Gradute	79000
female	50	Under-Graduate	16000
male	30	Graduate	23000
male	42	Graduate	33000
male	50	Post-Gradute	50000
male	47	Graduate	28000

Seaborn: Bar Chart

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

#Create a histogram of the income amounts
sns.histplot(data=data, x="Income")

plt.show()
```

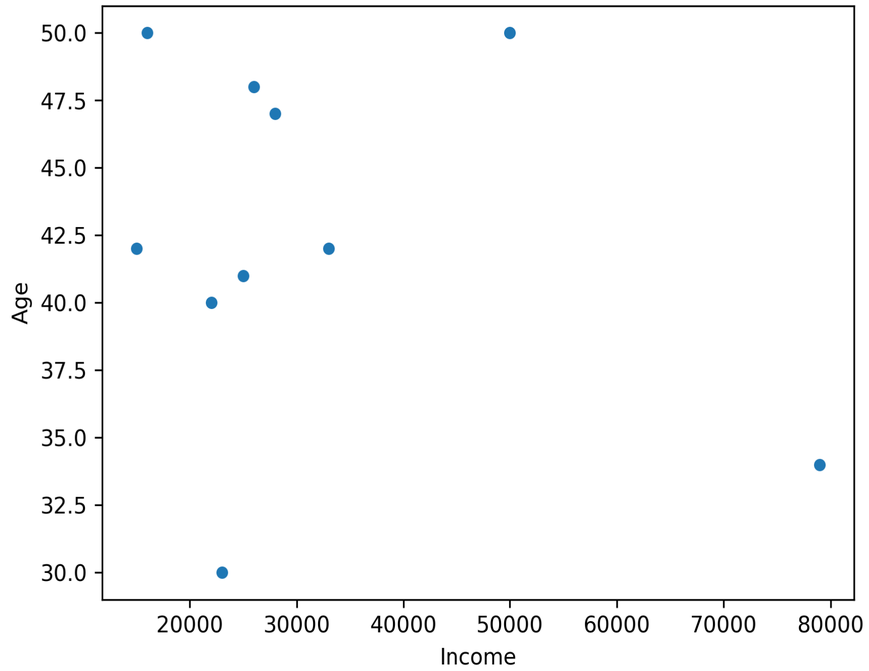


Seaborn: Scatter Chart

To visualize the relationship between two continuous variables.

Each point on the plot represents a single data point, and the position of the point on the x and y-axis represents the values of the two variables.

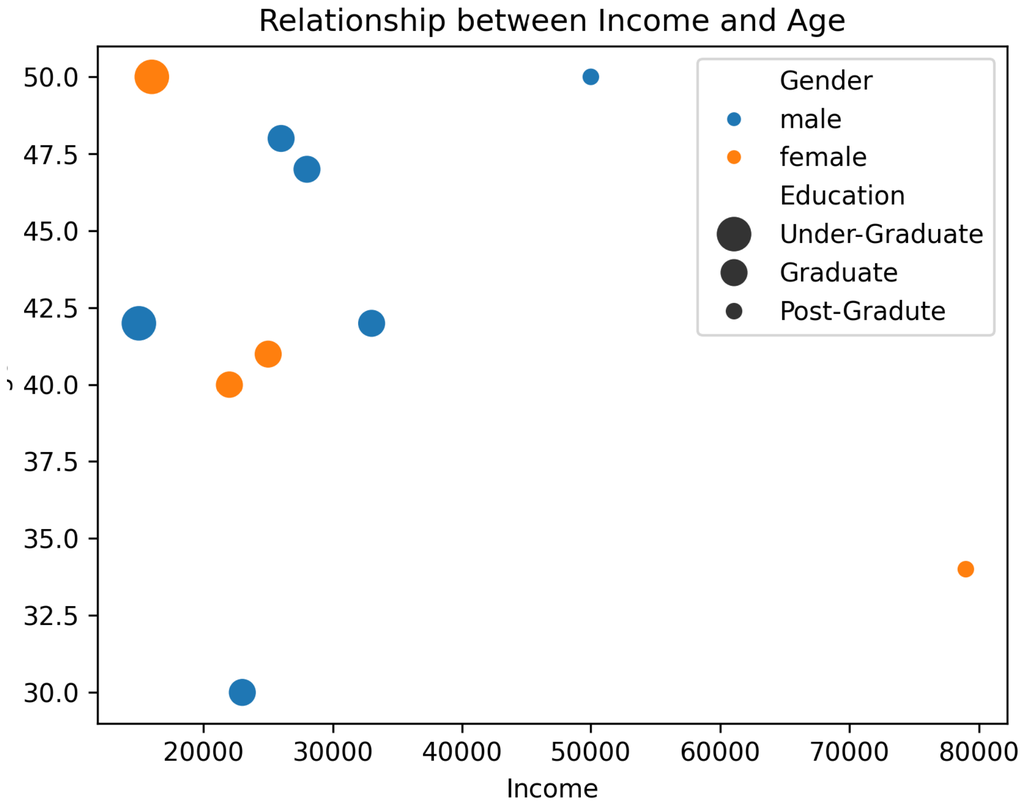
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

#Create a Scatter Plot of the income and age
sns.scatterplot(x="Income", y="Age", data=data)

plt.show()
```



Seaborn: Enhanced Scatter Plot

Improved by customizing the `hue` and `size` parameters of the plot.

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# customize the scatter plot
sns.scatterplot(x="Income", y="Age", hue="Gender", size="Education", sizes=(50, 200), data=data)

# add labels and title
plt.xlabel("Income")
plt.ylabel("Age")
plt.title("Relationship between Income and Age")

# display the plot
plt.show()
```

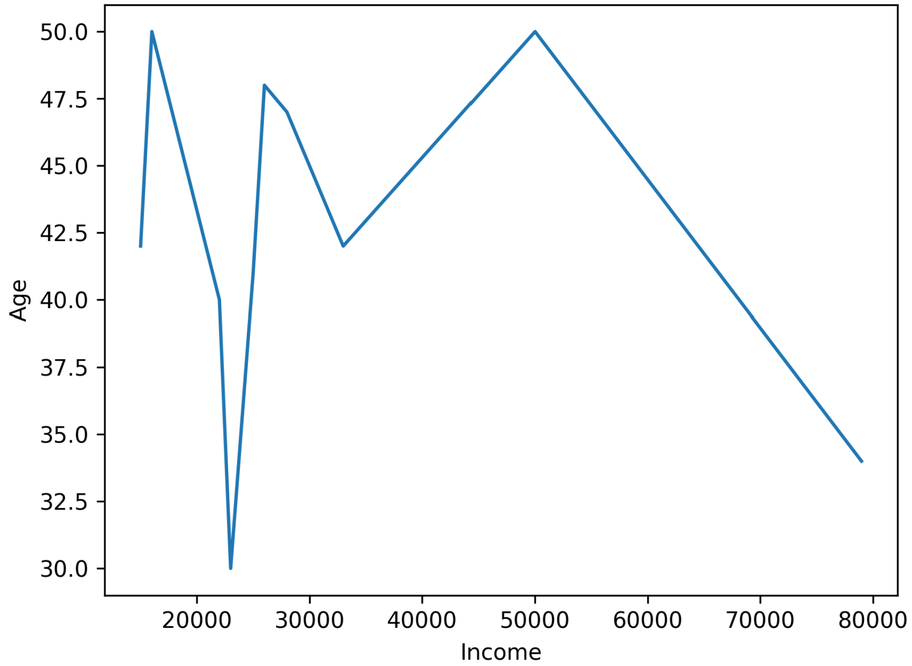


Seaborn: Line Plots

Used to visualize trends in data over time or other continuous variables.

Each data point is connected by a line, creating a smooth curve

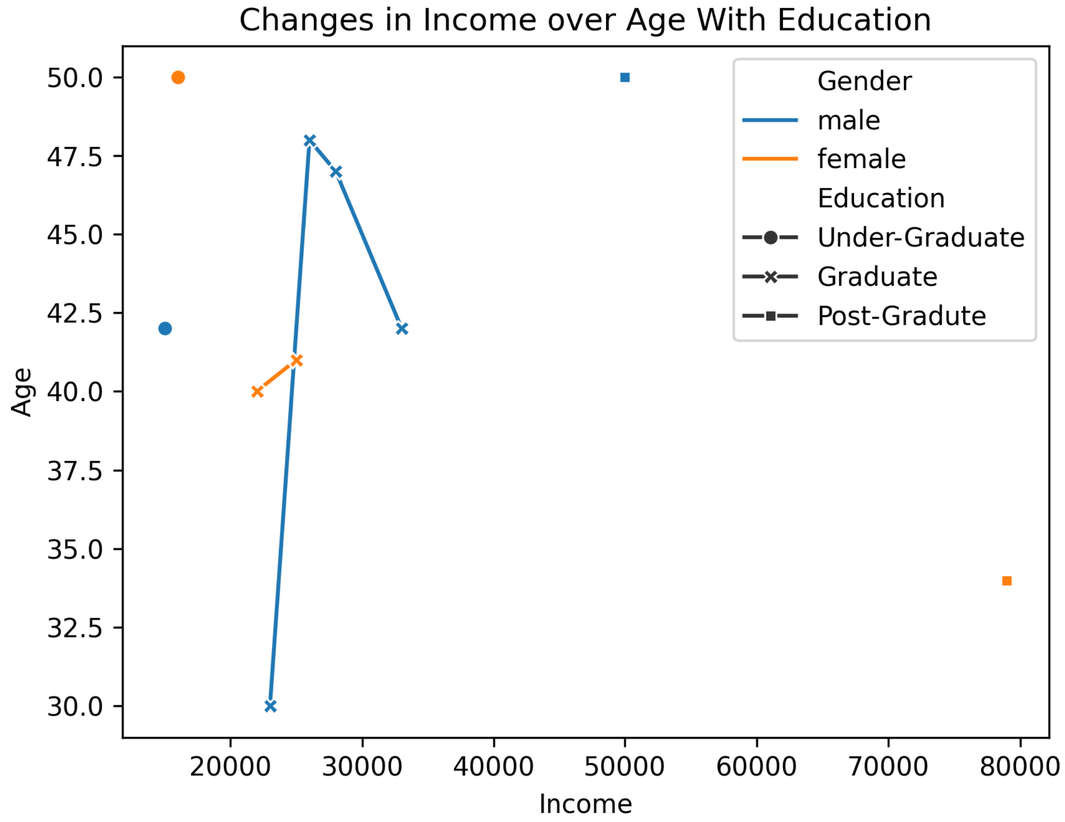
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

#Create a Line Plot of the income and age
sns.lineplot(x="Income", y="Age", data=data)

# display the plot
plt.show()
```



Seaborn: Enhanced Line Plots

Customized by using the `Gender` and `Education` columns from the dataset.

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# customizd the line plot
sns.lineplot(x="Income", y="Age", hue="Gender", style="Education", markers=True, dashes=False, data=data)

# add labels and title
plt.xlabel("Income")
plt.ylabel("Age")
plt.title("Changes in Income over Age With Education")

# display the plot
plt.show()
```

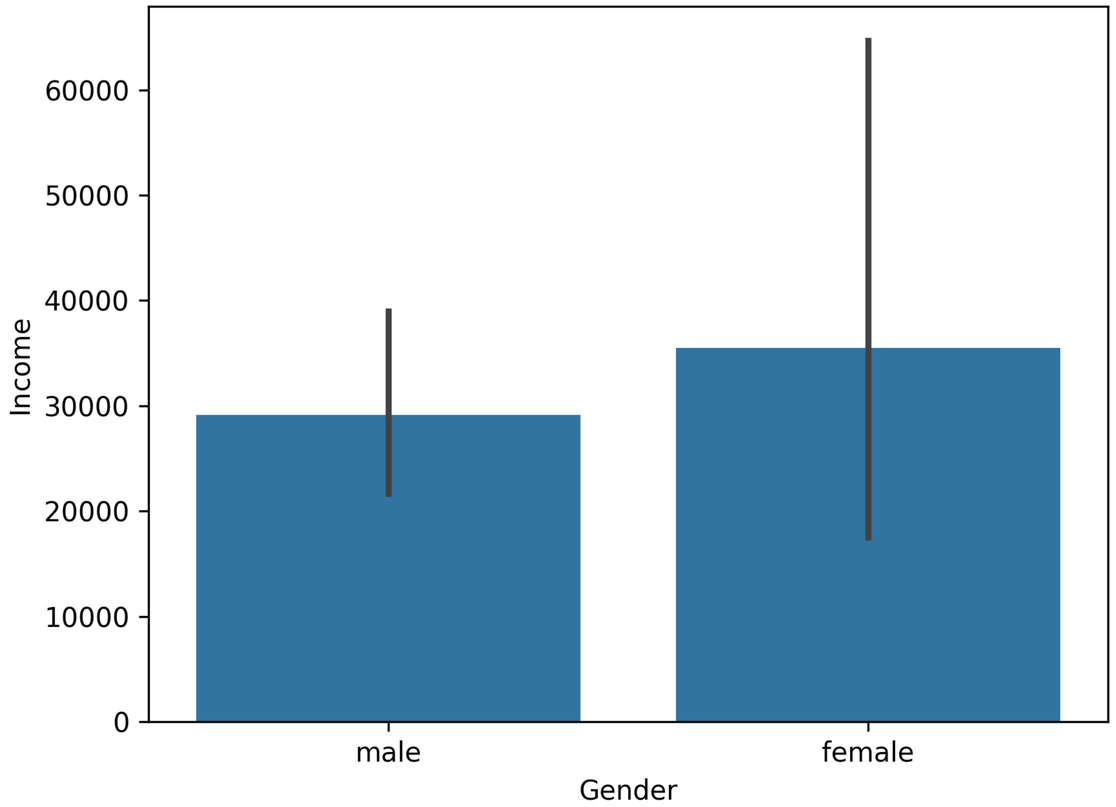


Seaborn: Bar Plot

Used to visualize the relationship between a categorical variable and a continuous variable.

Each bar represents the mean or median (or any aggregation) of the continuous variable for each category

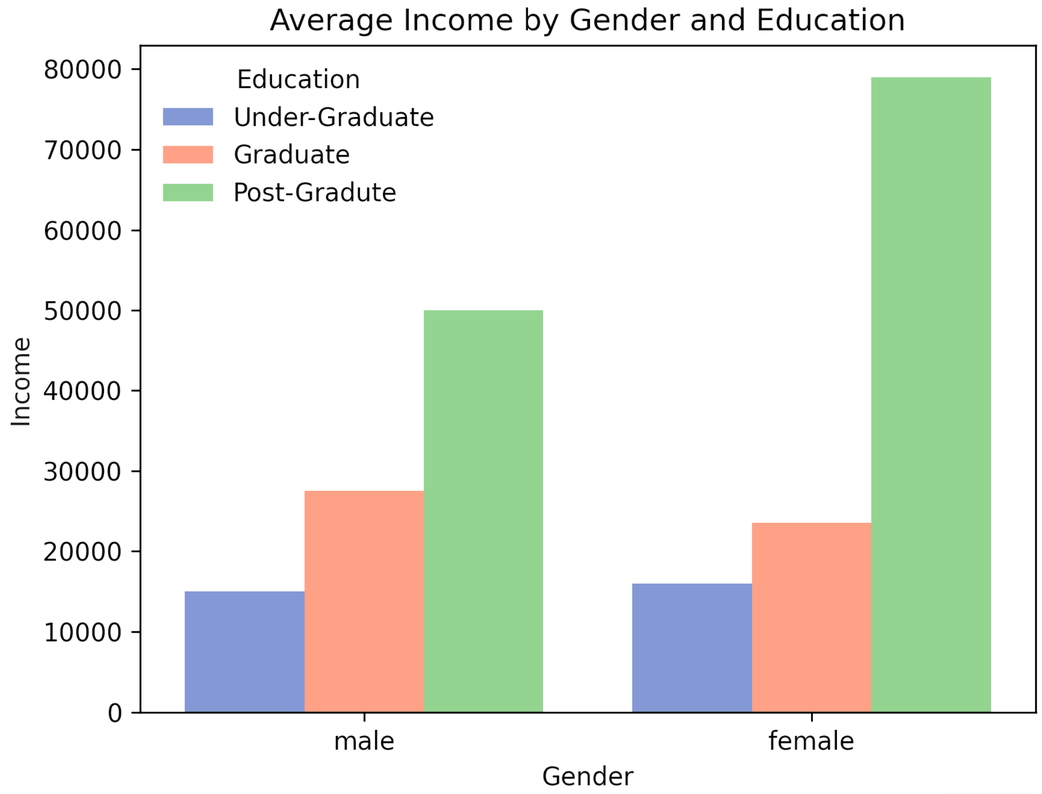
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

sns.barplot(x="Gender", y="Income", data=data)

# display the plot
plt.show()
```



Seaborn: Enhanced Bar Plot

Customized plot by including `Education` column from the dataset

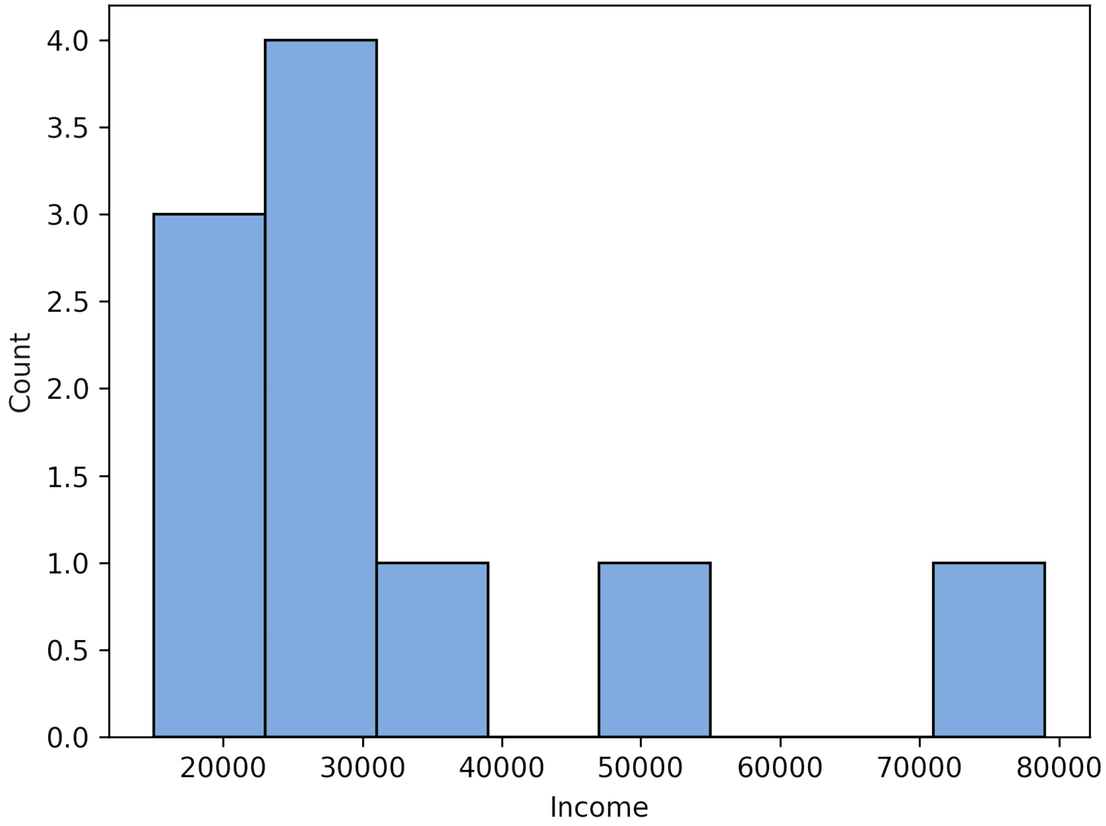
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# customized bar plot
sns.barplot(x="Gender", y="Income", hue="Education", ci=None, palette="muted", data=data)

# add labels and title
plt.xlabel("Gender")
plt.ylabel("Income")
plt.title("Average Income by Gender and Education")

# display the plot
plt.show()
```



Seaborn: Histogram

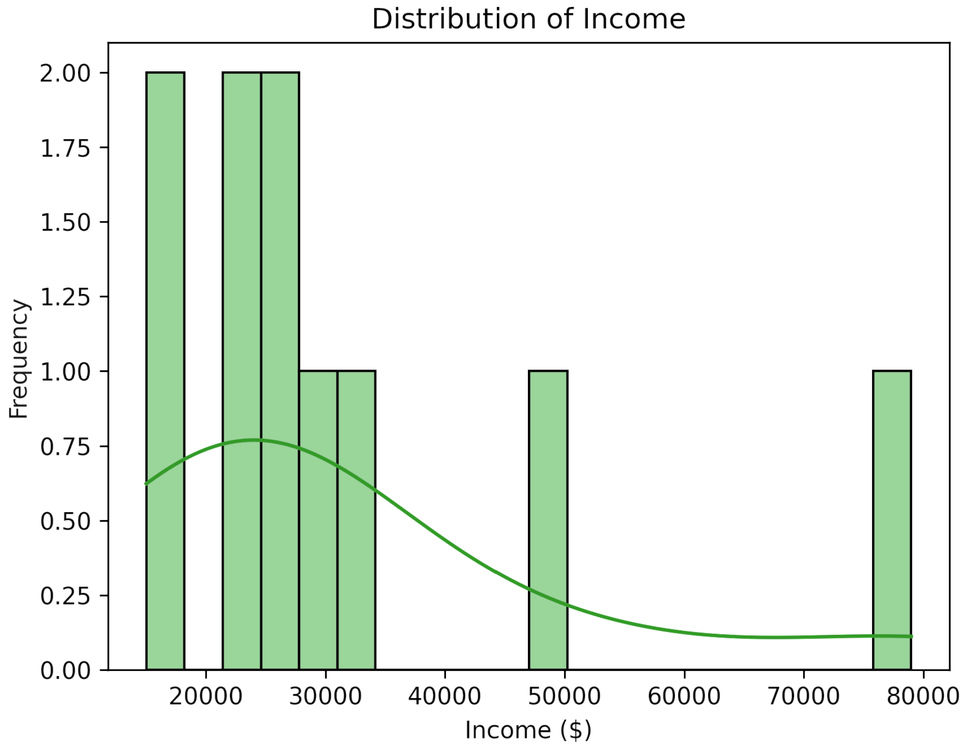
Visualize the distribution of a continuous variable

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

sns.histplot(x="Income", data=data)

# display the plot
plt.show()
```



Seaborn: Enhanced Histogram

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# customize the histogram
sns.histplot(data=data, x="Income", bins=20, kde=True, color="green")

# add labels and title
plt.xlabel("Income ($)")
plt.ylabel("Frequency")
plt.title("Distribution of Income")

# display the plot
plt.show()
```

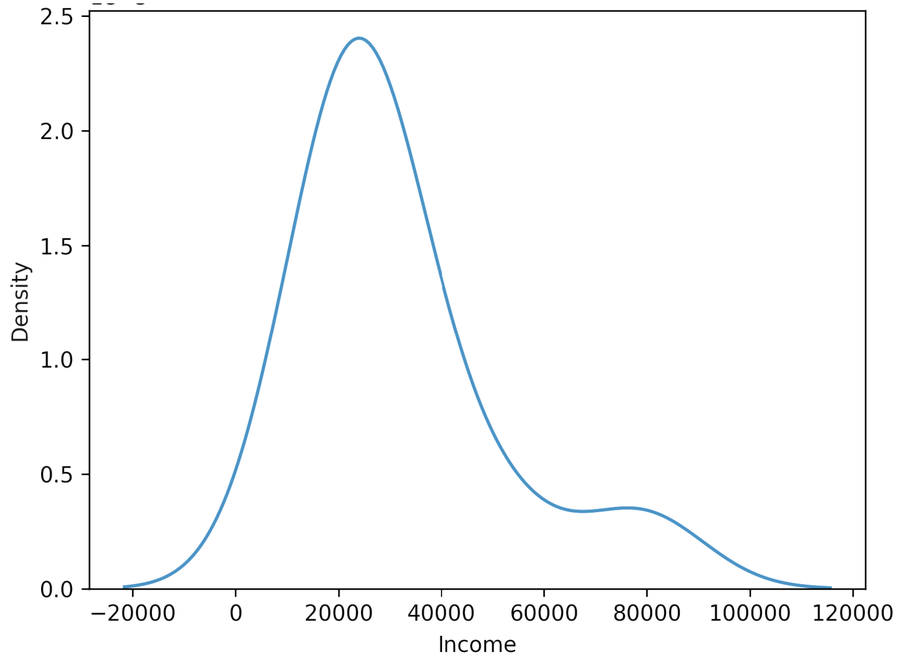


Seaborn: Density Plot

A type of data visualization that displays the distribution of a continuous variable

Like histograms, but instead of representing the data as bars, density plots use a smooth curve to estimate the density of the data

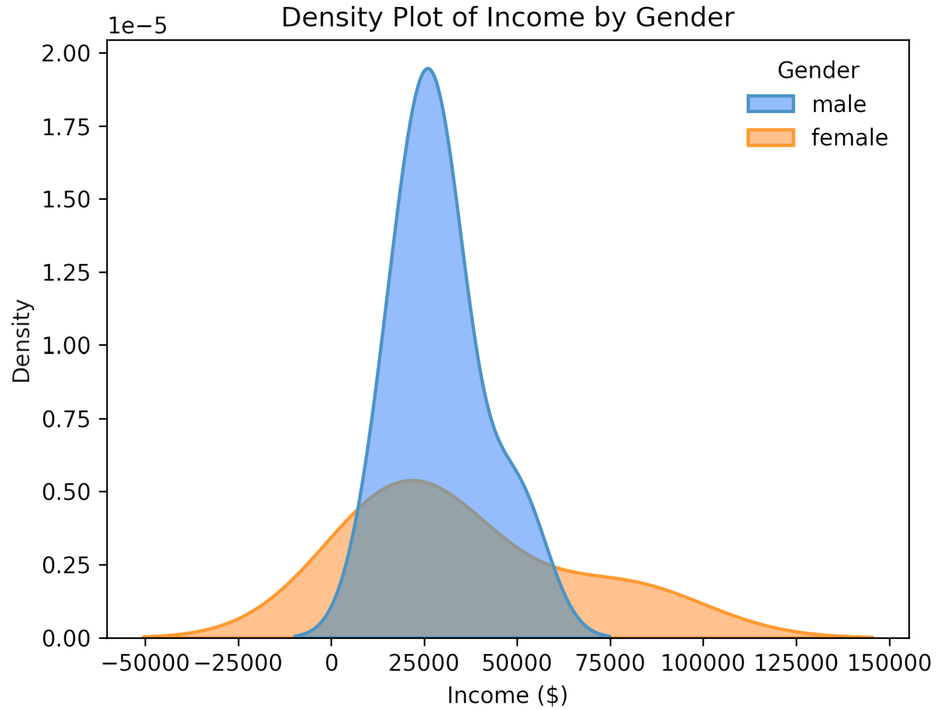
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

sns.kdeplot(data=data, x="Income")

# display the plot
plt.show()
```



Seaborn: Enhanced Density Plot

A density plot of the "Income" column and use the "hue" parameter to differentiate between “Gender" and “Education“. Use the "fill" parameter to fill the area under the curve. Adjust the "alpha" and "linewidth" parameters to make the plot more visually appealing

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

sns.kdeplot(data=data, x="Income", hue="Gender", fill=True, alpha=0.6, linewidth=1.5)

# Add a title and labels to the plot using Matplotlib
plt.title("Density Plot of Income by Gender")
plt.xlabel("Income ($)")
plt.ylabel("Density")

# Show the plot
plt.show()
```

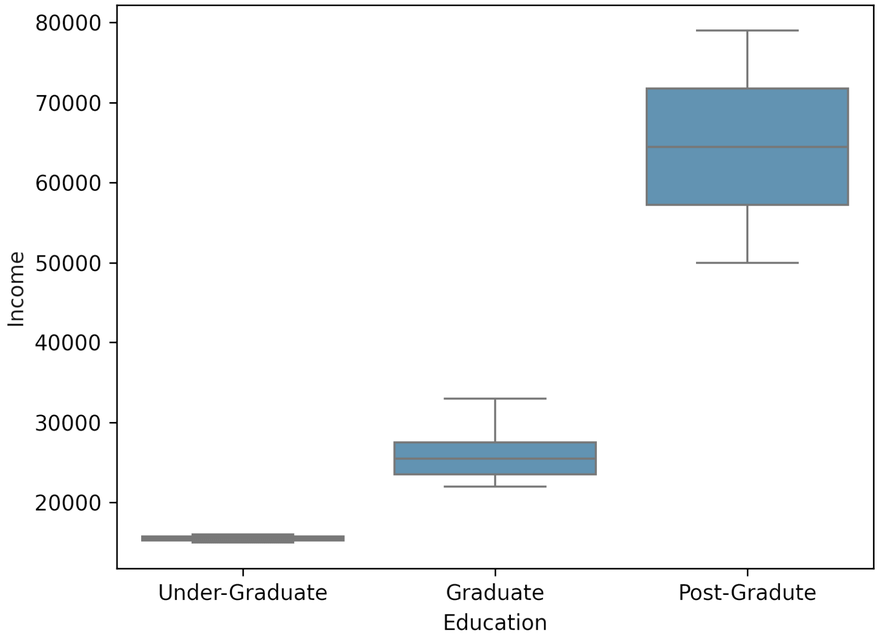


Seaborn: Boxplot

A type of visualization that shows the distribution of a dataset

Commonly used to compare the distribution of one or more variables across different categories

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

sns.boxplot(x="Education", y="Income", data=data)

# Show the plot
plt.show()
```



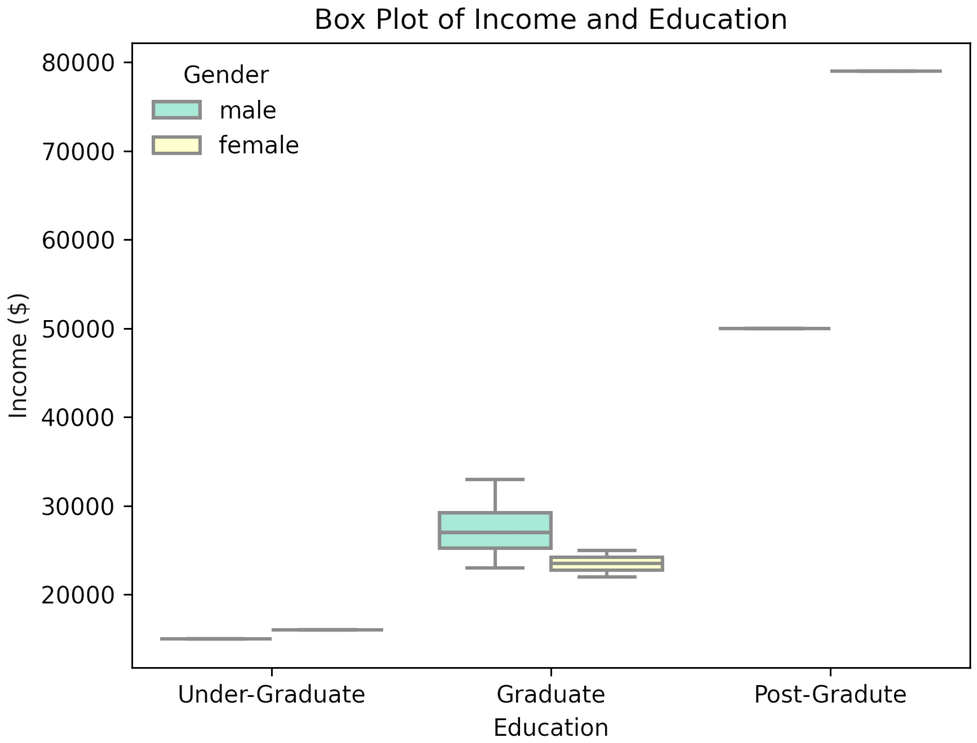
Seaborn: Enhanced Boxplot

Customized box plot by including `Age` column from the data

Customize the color scheme using the "palette" parameter

Adjust the linewidth and fliersize parameters to make the plot more visually appealing

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# create a box plot of total bill by day and meal time, using the "hue" parameter to differentiate between lunch and dinner
sns.boxplot(x="Education", y="Income", hue="Gender", data=data, palette="Set3", linewidth=1.5, fliersize=4)

# add a title, xlabel, and ylabel to the plot using Matplotlib functions
plt.title("Box Plot of Income and Education")
plt.xlabel("Education")
plt.ylabel("Income ($)")

# display the plot
plt.show()
```

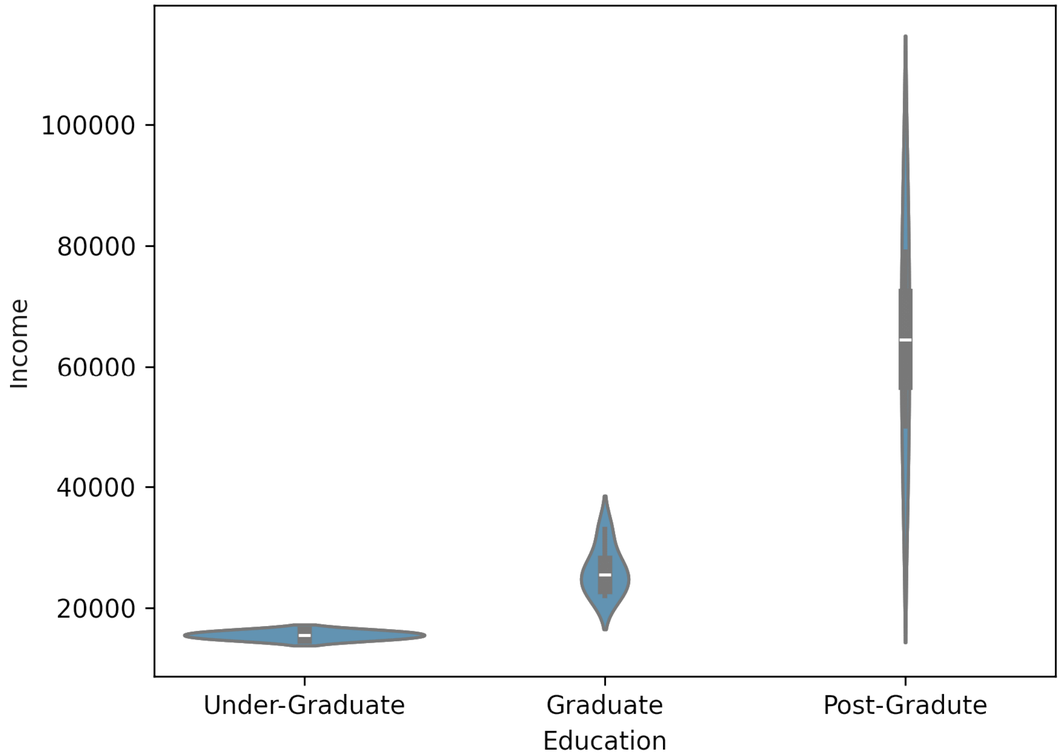


Seaborn: Violin Plot

It is a type of data visualization that combines aspects of both box plots and density plots

It displays a density estimate of the data, usually smoothed by a kernel density estimator, along with the interquartile range (IQR) and median in a box plot-like form

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("demonstration.csv")

# create a violin plot of Income by Age
sns.violinplot(x="Education", y="Income", data=data)

# display the plot
plt.show()
```



Making a Dashboard

Creating a dashboard involves using a web framework like Flask or Django along with visualization libraries like Plotly or Matplotlib for generating interactive plots

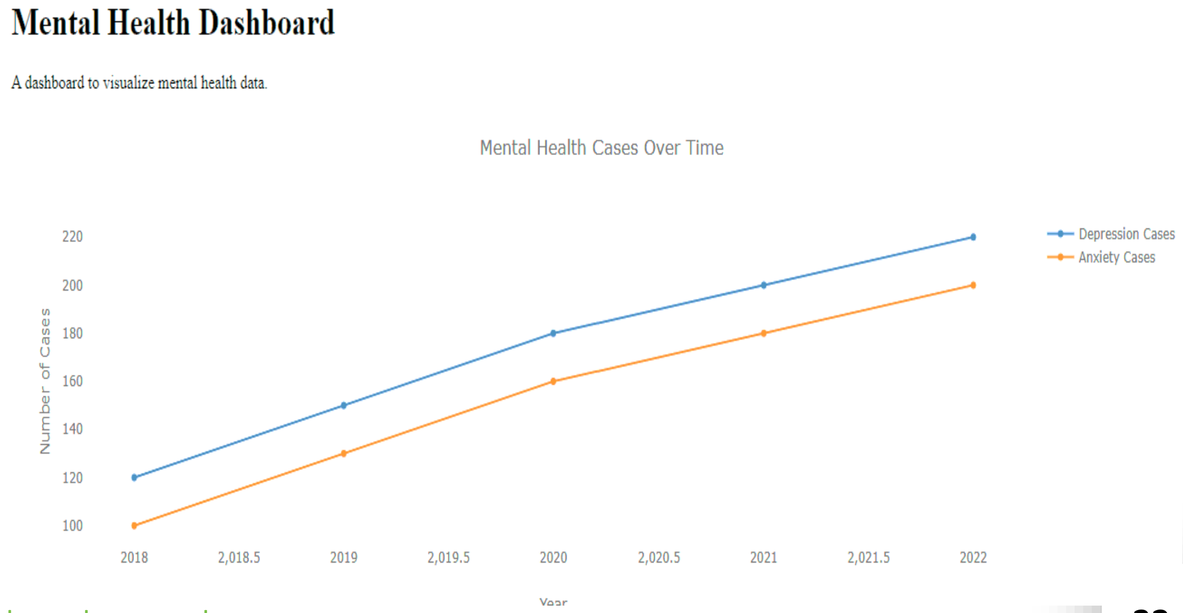
Dash is an open-source framework for building data visualization interfaces

The next example demonstrated how to create a simple dashboard using Dash, which is built on top of Plotly and designed specifically for creating interactive web-based dashboards in Python

Making a Dashboard: Example 1

```
import dash
from dash import dcc, html
import pandas as pd
# Sample mental health data
data = {
    'Year': [2018, 2019, 2020, 2021, 2022],
    'Depression Cases': [120, 150, 180, 200, 220],
    'Anxiety Cases': [100, 130, 160, 180, 200]
}
df = pd.DataFrame(data)
# Initialize the Dash app
app = dash.Dash(__name__)

# Define the layout of the dashboard
app.layout = html.Div(children=[
    html.H1(children='Mental Health Dashboard'),
    html.Div(children='''
        A dashboard to visualize mental health data.
    '''),
```

```
    dcc.Graph(
        id='mental-health-graph',
        figure={
            'data': [
                {'x': df['Year'], 'y': df['Depression Cases'], 'type': 'line', 'name': 'Depression Cases'},
                {'x': df['Year'], 'y': df['Anxiety Cases'], 'type': 'line', 'name': 'Anxiety Cases'}
            ],
            'layout': {
                'title': 'Mental Health Cases Over Time',
                'xaxis': {'title': 'Year'},
                'yaxis': {'title': 'Number of Cases'}
            }
        }
    )
])

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```



Making a Dashboard: Example 2 (Interactive Dashboard)

```
from dash import Dash, dcc, html, Input, Output, callback
import plotly.express as px
import  pandas as pd
# reading the database
df = pd.read_csv("demonstration-2.csv")
#df = px.data.tips()
app = Dash(__name__)
dropdown = dcc.Dropdown(["T1", "T2","T3"], "T1", clearable=False)
graph = dcc.Graph()
app.layout = html.Div([html.H4("Employment Vs stress"), dropdown, graph])
@callback(Output(graph, "figure"), Input(dropdown, "value"))
def update_bar_chart(day):
    mask = df["timepoint"] == day
    fig = px.histogram(df[mask], x="sex", y="stress count", color="emptype",pattern_
barmode="group")
    return fig
if __name__ == "__main__":
    app.run_server(debug=True)
```

sex	emptype	type	stress count	timepoint
female	employe	not_stressed	3842	T1
female	employe	stressed	912	T1
male	employe	not_stressed	3843	T1
male	employe	stressed	832	T1
female	unemploye	not_stressed	162	T1
female	unemploye	stressed	152	T1
male	unemploye	not_stressed	142	T1
male	unemploye	stressed	115	T1
female	employe	not_stressed	3870	T2
female	employe	stressed	885	T2
male	employe	not_stressed	3860	T2
male	employe	stressed	808	T2
female	unemploye	not_stressed	160	T2
female	unemploye	stressed	153	T2

African Population and
Health Research Center

Transforming lives in Africa through research.

THANK YOU

@aphrc

www.aphrc.org