II. Data Visualization with Python

MEXEE 402

DATA SCIENCE, MACHINE LEARNING AND AI

1. Introduction to Data Visualization with Python

The Pivotal Role of Data Visualization

- Transforms complex datasets into clear visual stories
- Acts like a detective’s tool: organizing scattered clues into solutions
- Real-world impact:
  - **Healthcare** → heat maps reveal disease spread
  - **Business** → line graphs show sales trends

Benefits:

- Quickly absorb information
- Spot patterns, correlations, and trends
- Enable data-driven decisions
- Communicate findings effectively

1. Introduction to Data Visualization with Python

Python's Data Visualization Libraries

- **Matplotlib** → Flexible, powerful 2D/3D plots; steep learning curve
- **Seaborn** → Simplifies plotting; excellent for statistical visualization
- **Plotly** → Interactive plots; higher complexity, but engaging visuals
- **Pandas Visualization** → Simple plots; integrates with data workflows
- **Plotnine** → Inspired by R's ggplot2; effective for layered graphics
- **Altair** → Declarative, clean, and user-friendly statistical graphics

Beyond Python

- **Tableau, Power BI, D3.js, R's ggplot2** → Other effective visualization tools

1. Introduction to Data Visualization with Python

Getting Started with Google Colab

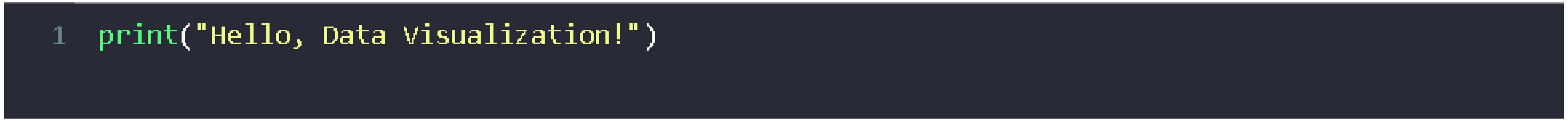
Google Colab – A free, browser-based coding environment requiring no setup and offering free computing power. To get started:

- Visit the Google Colab website.
- Click on 'File' > 'New notebook' to create a new notebook.
- You are now in a Python environment. You can write code in the cells and run them by clicking the play button on the left or by pressing Shift+Enter.

1. Introduction to Data Visualization with Python

Getting Started with Google Colab

As a test run, paste the following simple Python code into a cell:



```python
print("Hello, Data Visualization!")
```

- Running the cell prints: **“Hello, Data Visualization!”**
- **Google Colab** will be the primary tool for this course
  - Used to interact with Python
  - Employ data visualization libraries

2. Data Visualization with Google Colab, Pandas, and Matplotlib

Getting Acquainted with Google Colab

- Cloud-based Python environment
- Runs in the browser, no setup needed
- Works like a Python notebook powered by Google

Setting Up Your First Notebook

- Go to **Google Colab website**
- Click **File** → **New notebook**

Writing & Executing Code

- Use a **code cell** (click + Code)
- Type Python code
- Run with **Shift+Enter** → output shown below cell

2. Data Visualization with Google Colab, Pandas, and Matplotlib

Saving & Sharing Notebooks

- Saved directly to **Google Drive**
- **File → Save** to store work
- **Share** button → invite via email or link
- Option to **download** for offline sharing

Harnessing Pandas for Data Manipulation

- Core Python library for data analysis
- Provides DataFrames (spreadsheet-like structures)

Uses in Workflow:

- **Load datasets** (CSV, Excel, SQL, etc.)
- **Filter data** with simple conditions (e.g., Age > 30)

2. Data Visualization with Google Colab, Pandas, and Matplotlib

Utilizing Pandas in Your Data Science Workflow

Performing basic data analysis: Pandas allows statistical analysis, e.g., df.describe() gives descriptive statistics of a DataFrame.

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Pandas_for_Data_Manipulation.ipynb

In this code:

- **Import Pandas** library
- **Load dataset** from URL into a DataFrame
- **Print column names** of the DataFrame
- **Filter data** based on conditions
- **Use describe()** for descriptive statistics
- Prepares and structures data for **effective visualization**

2. Data Visualization with Google Colab, Pandas, and Matplotlib

Matplotlib—Your Tool for Effective Data Visualization

Matplotlib Overview

- Versatile Python plotting library
- Supports **static, animated, and interactive plots**
- Works seamlessly with **Pandas DataFrames**

Basic Plots with Matplotlib

- **Line Chart (plot)** → Track changes over time
- **Bar Chart (bar)** → Compare categorical data
- **Histogram (hist)** → Show data distribution

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Creating_Basic_Plots_with_Matplotlib.ipynb

Each plot reveals unique insights into beer servings, highlighting the importance of data visualization for understanding data.

3. Unraveling Data Distribution Using Histograms

Understanding Histograms

- Graphical representation of data grouped into **bins**
- **Bins** = value ranges; **height** = frequency of data points
- Provides a clear view of data distribution

Key Insights from Histograms:

- Frequency of values in specific ranges
- Detection of outliers
- Identification of skewness (how the data distribution leans or tilts on a histogram)
  - **Symmetrical (no skew):** Data is evenly distributed around the center (bell-shaped).
  - **Positive skew (right-skewed):** Long tail extends to the right → more values concentrated on the lower end.
  - **Negative skew (left-skewed):** Long tail extends to the left → more values concentrated on the higher end.

3. Unraveling Data Distribution Using Histograms

Real-World Example

Teacher analyzing exam scores

- Create bins (0–10, 11–20, etc.)
- Visualize class performance distribution

Creating Histograms in Google Colab

- Use **Matplotlib** to generate histograms
- Steps:
  - Open **Google Colab**
  - Start a **new notebook**

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/tree/main/Python_Visualization/Scores

Importing Necessary Libraries

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

3. Unraveling Data Distribution Using Histograms

Real-World Example

Creating and Loading Your Data

For our illustration, we'll use a CSV file containing the final exam scores of a class. You'll need to create this file. Follow these steps:

1. Open a text editor on your computer, such as Notepad on Windows or TextEdit on Mac. Or in Microsoft excel.

2. Copy and paste the following lines into your text editor:

Save the CSV file "scores.csv"

```
scores

85

90

78

92

88

76

95

89
```

3. Unraveling Data Distribution Using Histograms

Real-World Example

Upload the CSV file to Google Colab:

- In **Google Colab**, click the **folder icon** (left sidebar)
- Click **Upload to session storage** (upward arrow icon)
- Select **scores.csv** from your local files
- File is uploaded to your Colab session

Load the data into a pandas DataFrame:

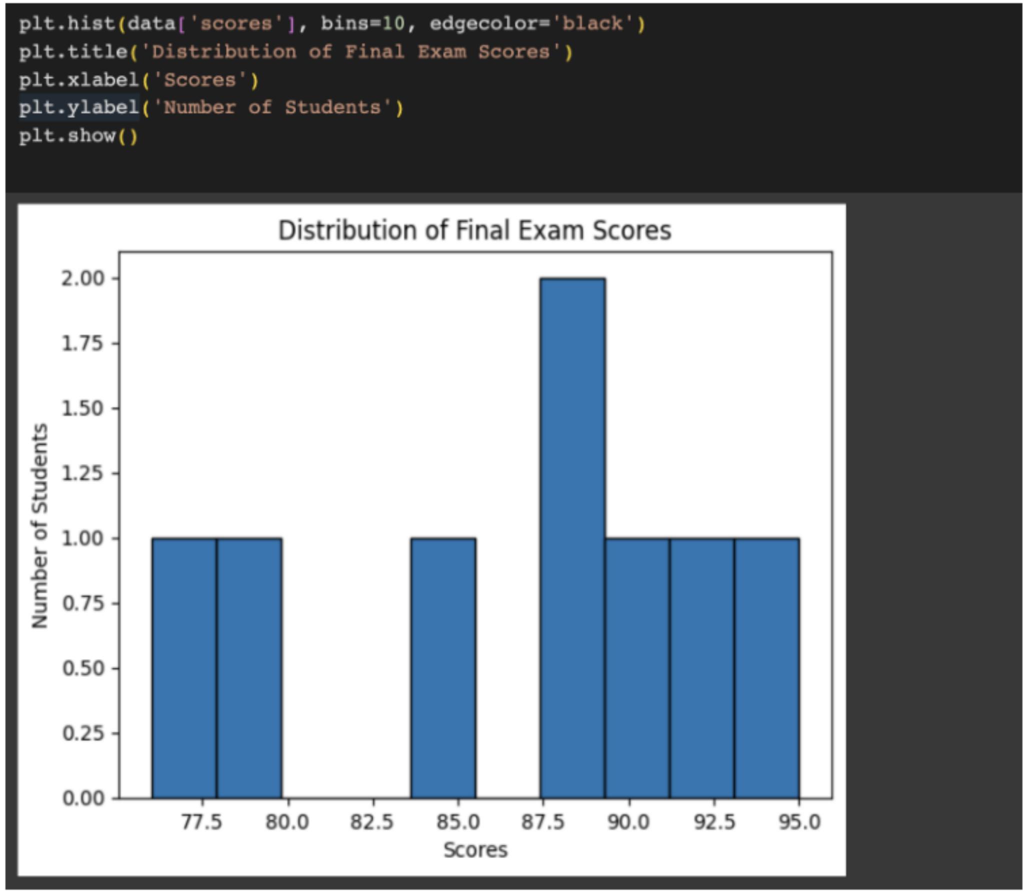
Finally, load the data into a pandas DataFrame using the following code:

- Generates a **histogram of final exam scores**
- **Bins** = score ranges
- **Height of bins** = number of students in each range

3. Unraveling Data Distribution Using Histograms

Real-World Example

- Generates a **histogram of final exam scores**
- **Bins** = score ranges
- **Height of bins** = number of students in each range



```
plt.hist(data['scores'], bins=10, edgecolor='black')
plt.title('Distribution of Final Exam Scores')
plt.xlabel('Scores')
plt.ylabel('Number of Students')
plt.show()
```

4. Mastering Time Series Visualization with Line Charts in Google Colab

Understanding Time Series Data

- Collection of **data points over time** (chronological order matters)
- Example: **Hourly temperature readings** form a time series
- Applications:
  - **Finance** → stock trends
  - **Weather** → forecasting patterns
  - **Healthcare** → tracking patient vitals
  - **E-commerce** → analyzing website traffic

Key Value:

- Explains the **past** and provides insights into the **future**

Visualizing Time Series Data

- **Line charts** → simple & effective for time series
- Reveal **patterns, trends, and outliers** (data points that deviate significantly from the overall pattern)
  - Appear as **unusually high or low values** in a time series
  - May indicate **errors, rare events, or important anomalies**

4. Mastering Time Series Visualization with Line Charts in Google Colab

Importing Libraries

We will need to import two Python libraries, pandas and matplotlib.pyplot, to get started with data manipulation and visualization respectively. You can import these libraries by running the following code in a new cell:

```
import pandas as pd

import matplotlib.pyplot as plt
```

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/tree/main/Python_Visualization/Line_Charts

4. Mastering Time Series Visualization with Line Charts in Google Colab

Creating and Loading Data

For the purpose of this tutorial, we'll consider a dataset representing the daily temperature of a city for a month. Let's walk through the steps to create this data in a CSV file and subsequently load it into our notebook.

Begin by creating a CSV file with the following content, which includes the date and corresponding temperature:

Save the CSV file “temperature.csv"

```
Date,Temperature

2023-01-01,15

2023-01-02,18

2023-01-03,20

2023-01-04,17

2023-01-05,16

2023-01-06,19

2023-01-07,21

2023-01-08,16

2023-01-09,17

2023-01-10,18
```

4. Mastering Time Series Visualization with Line Charts in Google Colab

Saving the CSV File

- Save as **temperature_data.csv**
- Ensure extension is **.csv** (not .txt)

Uploading to Google Colab

- Click **folder icon → Upload (up arrow)**
- Select **temperature_data.csv** to upload

Using Pandas

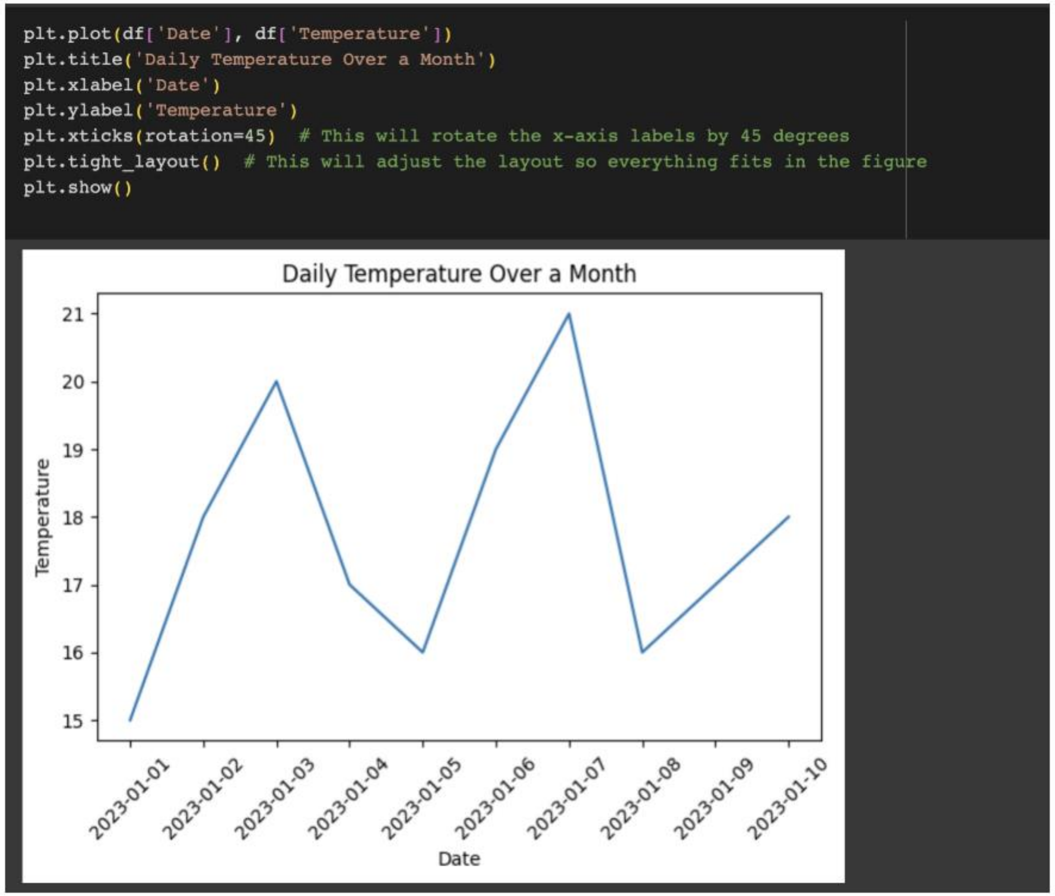
```
data =
pd.read_csv("temperature_data.csv")
```

Plotting the Data

- ❖ Use **plt.plot()** to create a line chart
- ❖ Customize with:
  - ❖ **Title** → “Daily Temperature Over a Month”
  - ❖ **X-axis** → Date (rotated 45° for clarity)
  - ❖ **Y-axis** → Temperature
  - ❖ **plt.tight_layout()** → ensures labels fit properly
- ❖ **Output:** Line chart showing daily temperature trends
- ❖ Helps identify **patterns** and **anomalies** for deeper analysis/forecasting

4. Mastering Time Series Visualization with Line Charts in Google Colab

```
plt.plot(df['Date'], df['Temperature'])

plt.title('Daily Temperature Over a Month')

plt.xlabel('Date')

plt.ylabel('Temperature')

plt.xticks(rotation=45) # Rotates the x-axis labels by 45 degrees

plt.tight_layout() # Adjusts the layout so everything fits in the figure

plt.show()
```



```
plt.plot(df['Date'], df['Temperature'])
plt.title('Daily Temperature Over a Month')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.xticks(rotation=45)  # This will rotate the x-axis labels by 45 degrees
plt.tight_layout()  # This will adjust the layout so everything fits in the figure
plt.show()
```

5. Using Scatter Plots in Google Colab

Understanding Scatter Plots

- Display relationship between **two variables** using Cartesian coordinates
- **X-axis** = one variable, **Y-axis** = another
- Each point = data observation
- Example:
  - City planner analyzing **population density vs. number of parks**
  - Scatter plot reveals possible **positive correlation**

Why Scatter Plots Matter

- **Identify correlations** (positive, negative, or none)
- **Spot trends** (increases, decreases, fluctuations)
- **Detect outliers** or anomalies in data

5. Using Scatter Plots in Google Colab

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/tree/main/Python_Visualization/Scatter_Plot

6. Comparing Data with Bar Graphs in Google Colab

Understanding Bar Graphs

❖ **Bar Graphs (Bar Charts):** Use rectangular bars to represent data categories.

❖ **Bar Length/Height:** Proportional to value or frequency → higher value = taller bar.

❖ **Purpose:** Simplifies complex data into **clear, visual comparisons** for better decisions.

❖ Example (School Performance):

- ❖ Categories = Subjects (Math, English, Science, History).
- ❖ Values = Average student scores.
- ❖ Bars show quick comparison of strengths & weaknesses.

6. Comparing Data with Bar Graphs in Google Colab

Creating Bar Graphs with Matplotlib in Google Colab
Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Advance_DataVisualization/BarGraphs_Matplotlib_in_Google_Colab.ipynb

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Advance_DataVisualization/average_scores.csv

Creating Bar Graphs with Matplotlib using Seaborn's Titanic Dataset
Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Advance_DataVisualization/Seaborn_Titanic_Dataset.ipynb

7. Advanced Data Visualization Techniques in Google Colab

Plotnine, Plotly, and Altair

Plotnine

- Python clone of R's **ggplot2**
- Built on **Matplotlib** + works with **Pandas**
- Great for **complex, layered statistical graphics** with simple syntax
- Sample import of Plotnine:
  - from plotnine import ggplot, aes, stat_summary, ggtitle

7. Advanced Data Visualization Techniques in Google Colab

Plotnine, Plotly, and Altair

❖Plotly

- ❖ Best for **interactive visuals**
- ❖ Supports **3D charts, maps, and network graphs**
- ❖ Highly engaging for presentations & dashboards
- ❖ Sample import of Plotly:
  - ❖ import plotly.express as px

7. Advanced Data Visualization Techniques in Google Colab

Plotnine, Plotly, and Altair

- Altair
  - Declarative library based on **Vega-Lite**
  - Produces **clear, concise, statistical graphics**
  - Ideal for **data exploration & interpretation**
  - Sample import of Plotly:
    - import altair as alt

7. Advanced Data Visualization Techniques in Google Colab

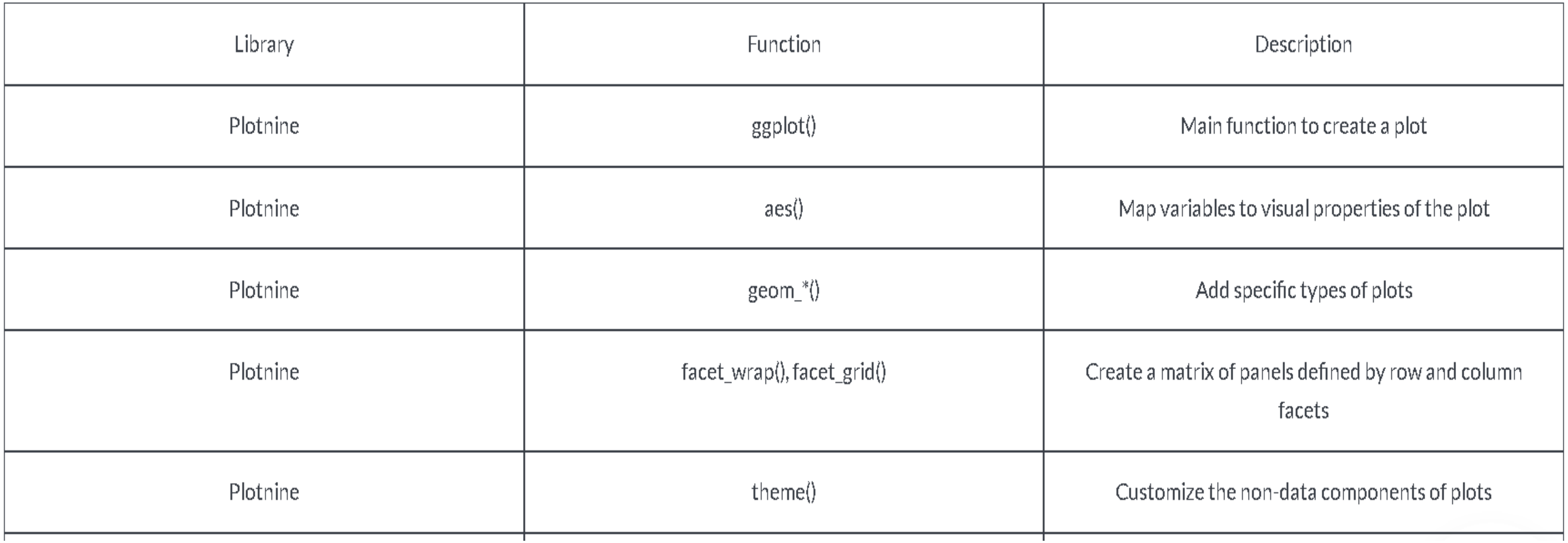
Plotnine_Plotly_Altair_Titanic_Dataset

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Advance_DataVisualization/Plotnine_Plotly_Altair_Titanic_Dataset.ipynb

7. Advanced Data Visualization Techniques in Google Colab

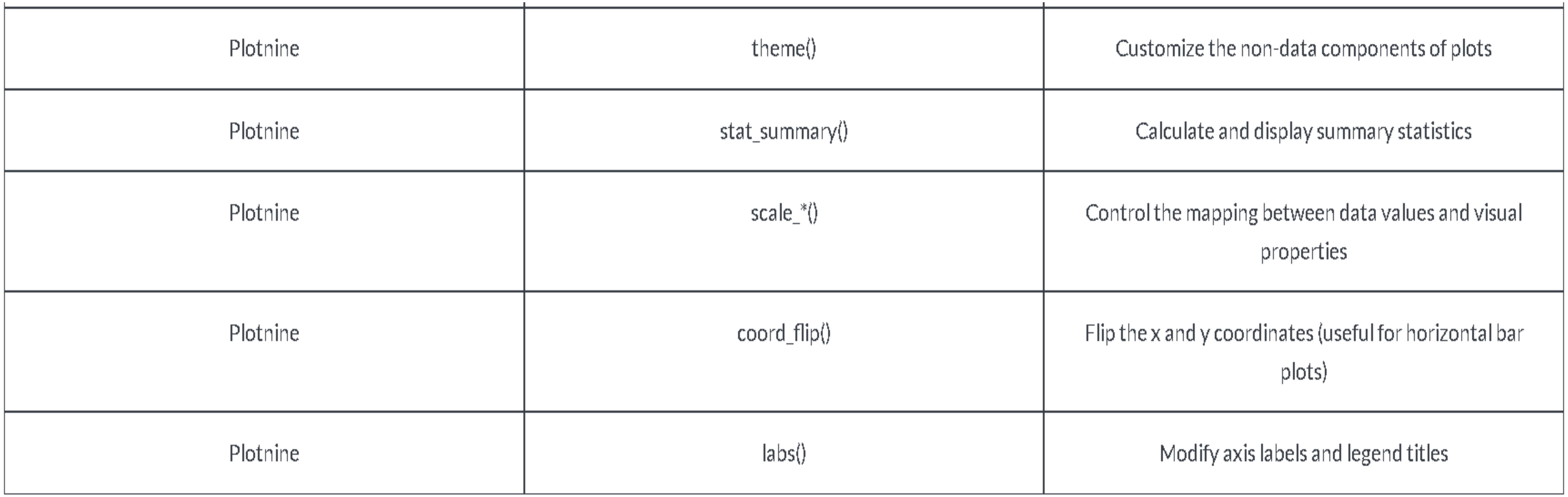
Plotnine, Plotly, and Altair: Table of Common Functions



Library	Function	Description
Plotnine	ggplot()	Main function to create a plot
Plotnine	aes()	Map variables to visual properties of the plot
Plotnine	geom_*()	Add specific types of plots
Plotnine	facet_wrap(), facet_grid()	Create a matrix of panels defined by row and column facets
Plotnine	theme()	Customize the non-data components of plots

7. Advanced Data Visualization Techniques in Google Colab

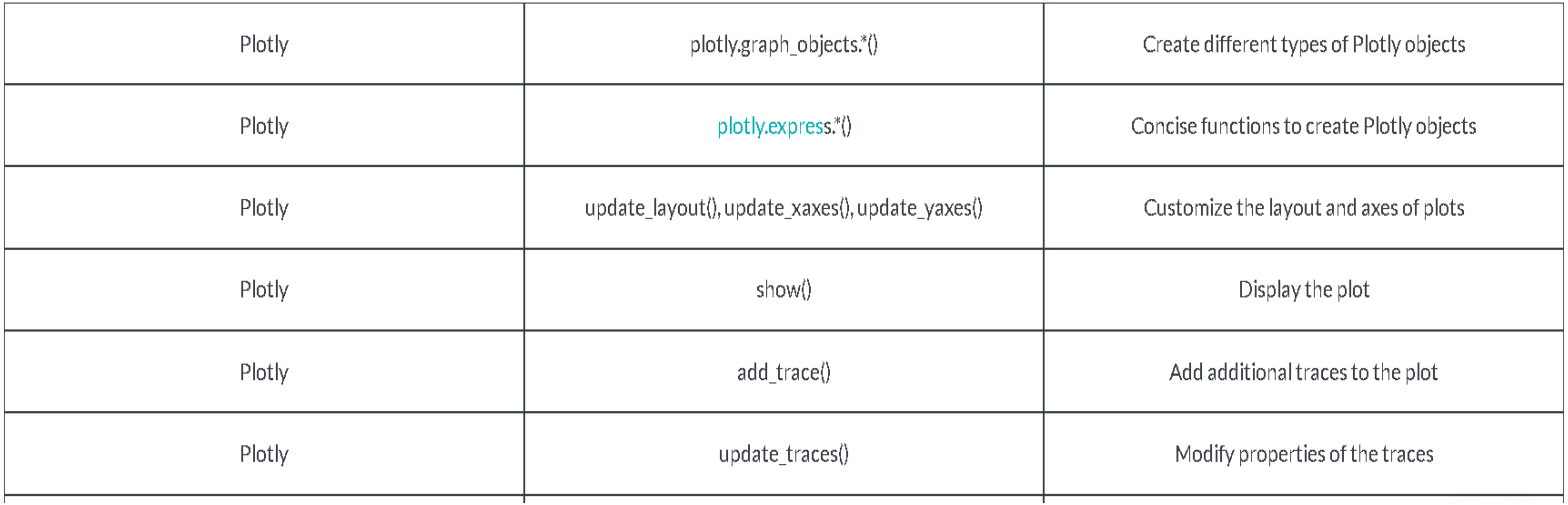
Plotnine, Plotly, and Altair: Table of Common Functions



Plotnine	theme()	Customize the non-data components of plots
Plotnine	stat_summary()	Calculate and display summary statistics
Plotnine	scale_*()	Control the mapping between data values and visual properties
Plotnine	coord_flip()	Flip the x and y coordinates (useful for horizontal bar plots)
Plotnine	labs()	Modify axis labels and legend titles

7. Advanced Data Visualization Techniques in Google Colab

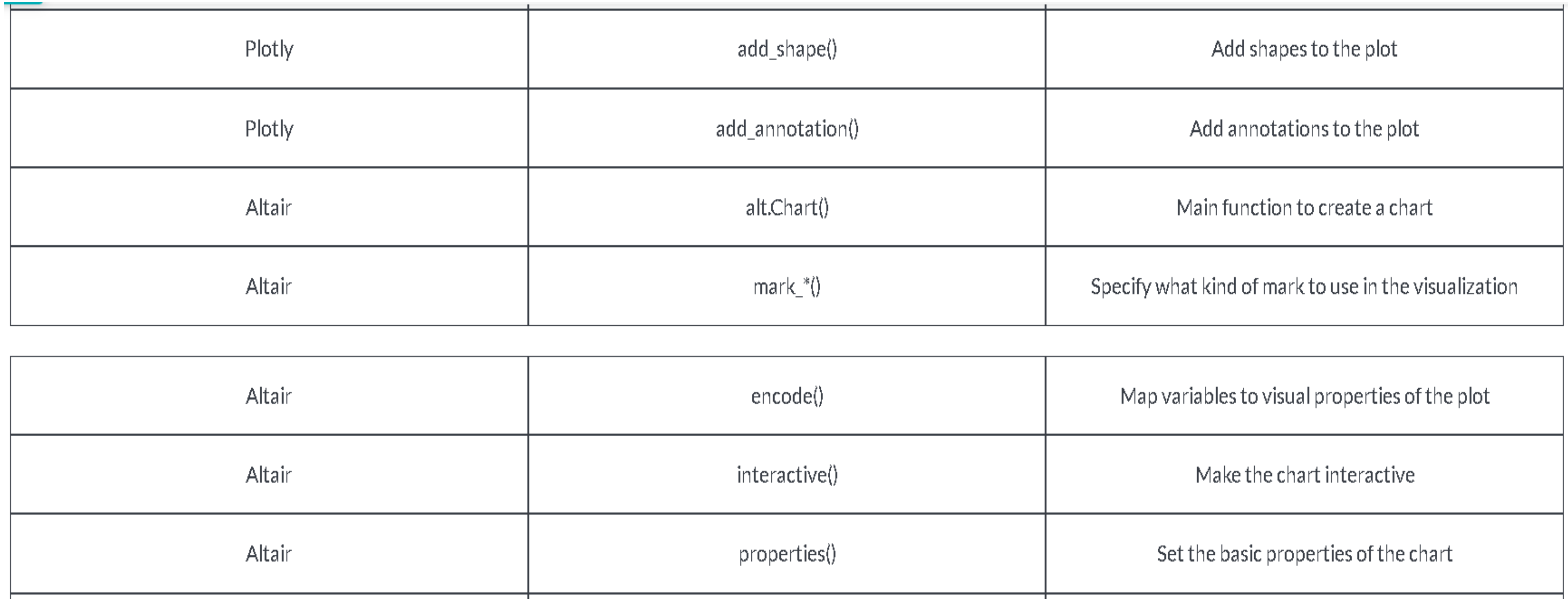
Plotnine, Plotly, and Altair: Table of Common Functions



Plotly	plotly.graph_objects.*()	Create different types of Plotly objects
Plotly	plotly.express.*()	Concise functions to create Plotly objects
Plotly	update_layout(), update_xaxes(), update_yaxes()	Customize the layout and axes of plots
Plotly	show()	Display the plot
Plotly	add_trace()	Add additional traces to the plot
Plotly	update_traces()	Modify properties of the traces

7. Advanced Data Visualization Techniques in Google Colab

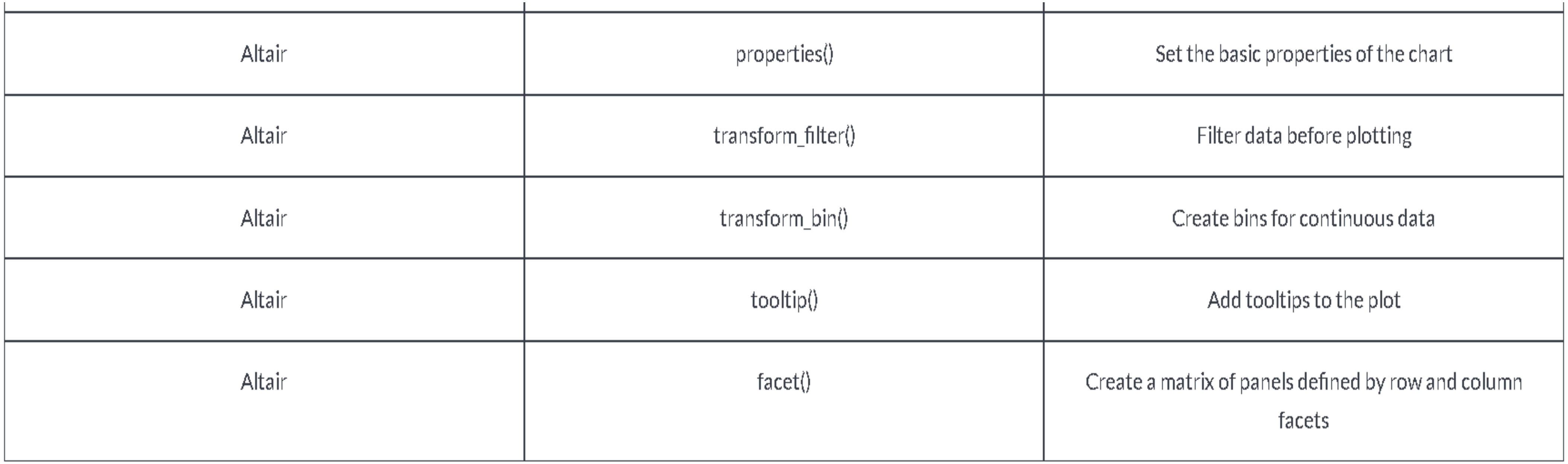
Plotnine, Plotly, and Altair: Table of Common Functions



Plotly	add_shape()	Add shapes to the plot
Plotly	add_annotation()	Add annotations to the plot
Altair	alt.Chart()	Main function to create a chart
Altair	mark_*()	Specify what kind of mark to use in the visualization
Altair	encode()	Map variables to visual properties of the plot
Altair	interactive()	Make the chart interactive
Altair	properties()	Set the basic properties of the chart

7. Advanced Data Visualization Techniques in Google Colab

Plotnine, Plotly, and Altair: Table of Common Functions



Altair	properties()	Set the basic properties of the chart
Altair	transform_filter()	Filter data before plotting
Altair	transform_bin()	Create bins for continuous data
Altair	tooltip()	Add tooltips to the plot
Altair	facet()	Create a matrix of panels defined by row and column facets

8. Visual Data Analysis Project in Google Colab

Iris_Data_Visualization

Go to this link:

https://github.com/MikkoDT/MexEE402_AI/blob/main/Python_Visualization/Advance_DataVisualization/Iris_Data_Visualization.ipynb