

Plotting Magic: Visualizing Weather Data with Matplotlib

Create Eye-Catching Visuals with Python

Michael Borck

Table of contents

Today	2
Why Visualise Data?	3
What is Data Visualisation?**	3
Popular Libraries in Python**	3
Workflow	3
Common Visualisation Roles	3
Python Graph Gallery	4
Dashboard	4
Introduction to Matplotlib	5
Reading Data	5
Creating Basic Plots	5
Line plot for temperature	5
Bar plot for humidity	5
Pie chart for weather description	6
Customising Plots	7

Adding annotations	7
Customising colors and styles	8
Effective Data Visualisation	8
Conclusion	8
Homework	8

***ELECTRONIC WARNING NOTICE FOR COPYRIGHT
STATUTORY LICENCES***

WARNING

This material has been reproduced and communicated to you by or on behalf of **Curtin University** in accordance with section 113P of the *Copyright Act 1968 (the Act)*

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Today

- Introduce the importance of data visualisation in Python.
- Learn how to visualise weather data using Matplotlib
- Understand how to create basic plots to represent weather data
- Customise plots to enhance readability and presentation

Why Visualise Data?

- Simplifies complex data
- Identifies patterns and trends
- Aids in decision-making
- Enhances data storytelling
- Facilitates better communication

What is Data Visualisation?**

- Visual representation of data
- Uses charts, graphs, and plots
- Helps in understanding data insights

Popular Libraries in Python**

- **Matplotlib:** Highly customisable, widely used
- **Seaborn:** Built on Matplotlib, easier syntax
- **Plotly:** Interactive visualisations, web-based
- **Bokeh:** Interactive plots, for web apps
- **Altair:** Declarative statistical visualisation

Workflow

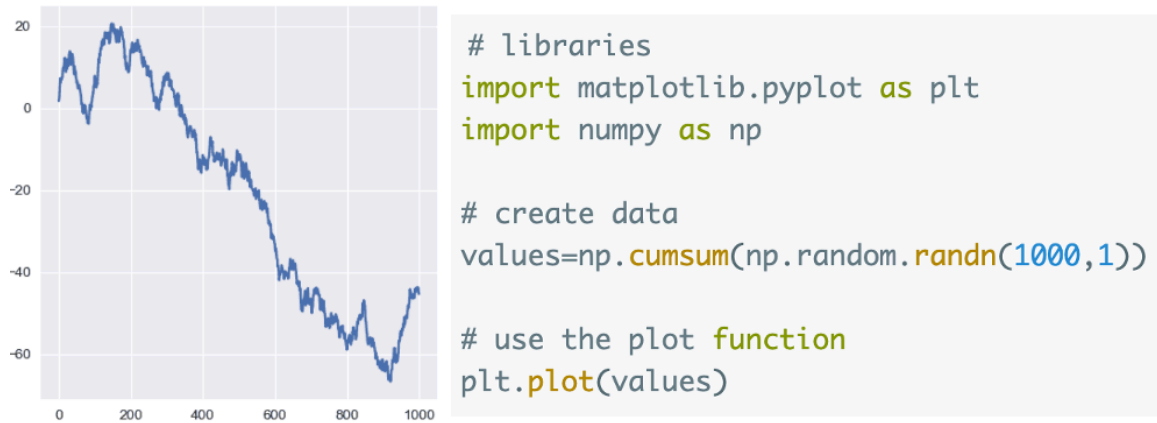
- Find the data you want
- What story do you want to tell
- Pick from the Python Graph Galley
- Get data (upload, wget, etc)
- Implement in notebook (or script)

Common Visualisation Roles

- Showing change over time
- Showing a part-to-whole composition
- Looking at how data is distributed
- Comparing values between groups
- Observing relationships between variables

- Looking at geographical data

Python Graph Gallery



Dashboard



Introduction to Matplotlib

- Matplotlib is a widely used Python library for creating static, animated, and interactive visualisations
- Import required libraries: `import pandas as pd` and `import matplotlib.pyplot as plt`

Reading Data

- Read processed weather data from the CSV file created in the previous session

```
import pandas as pd
df = pd.read_csv('data/processed/processed_weather_data.csv')
```

Creating Basic Plots

- line plot for temperature
- bar plot for humidity
- pie chart ofr weather description
- scatter plots temperature vs windspeed

Line plot for temperature

```
plt.plot(df['Location'], df['Temperature (C)'], marker='o')
plt.title('Temperature by Location')
plt.xlabel('Location')
plt.ylabel('Temperature (°C)')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

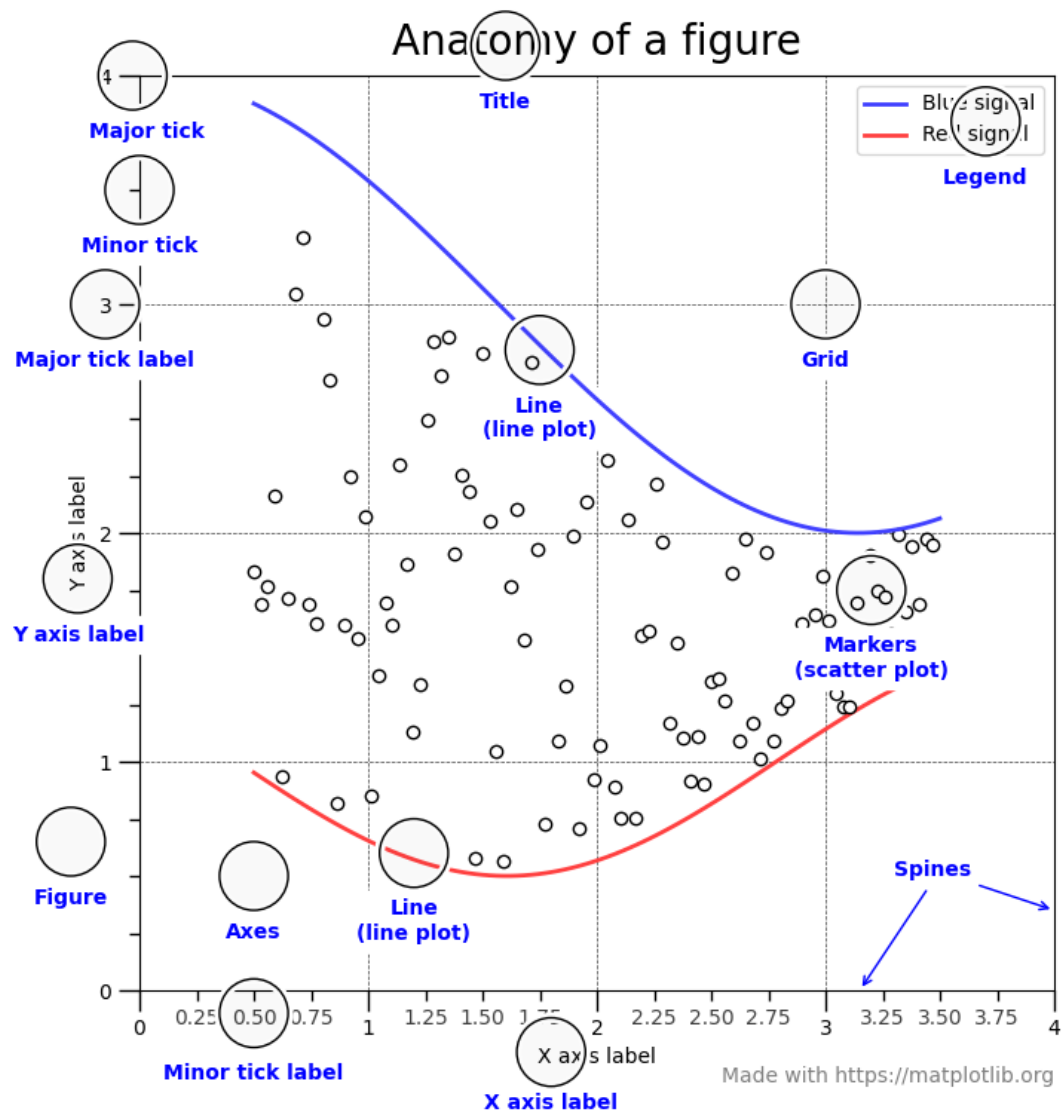
Bar plot for humidity

```
plt.bar(df['Location'], df['Humidity (%)'], color='skyblue')
plt.title('Humidity by Location')
plt.xlabel('Location')
plt.ylabel('Humidity (%)')
plt.xticks(rotation=45)
plt.show()
```

Pie chart for weather description

```
weather_counts = df['Weather'].value_counts()
plt.pie(weather_counts, labels=weather_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Weather Description Distribution')
plt.axis('equal')
plt.show()
```

Customising Plots



Adding annotations

```
plt.annotate(f'Max Temp: {max_temp:.2f}°C', xy=(max_temp_location, max_temp),
            xytext=(max_temp_location, max_temp+2),
            arrowprops=dict(facecolor='black', shrink=0.05))
```

Customising colors and styles

```
plt.plot(df['Location'], df['Temperature (C)'], marker='o', linestyle='--', color='b')
plt.title('Temperature by Location', fontsize=14, fontweight='bold')
plt.xlabel('Location', fontsize=12)
plt.ylabel('Temperature (°C)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.show()
```

Effective Data Visualisation

- Choose the right type of chart
- Maintain clarity and simplicity
- Use colour wisely
- Ensure accuracy and precision
- Tailor visualisations to the audience

Conclusion

- In this session, we learned how to visualise weather data using Matplotlib
- We created basic plots such as line plots, bar plots, and pie charts, and we customised these plots to improve readability and presentation
- Next session, we will focus on advanced data visualisation techniques using subplots and grids.

Homework

- Create additional plots to visualise other aspects of the weather data (e.g., wind speed, pressure)
- Experiment with different types of plots and customisations to enhance the visualisations