Plotting Perfection: Mastering Matplotlib Dashboards

Create Engaging and Insightful Visual Stories

Michael Borck

Table of contents

Today

- Create daily forecast cards with high/low temperatures.
- Design a layout for the dashboard using subplots and grids.
- Use icons to represent weather conditions.
- **Output:**
  - `04_advanced_vix.ipynb`
  - `visualise_data.py`

Recap and Introduction (10 minutes)

- **Recap of Session 4:**
  - Key takeaways
  - Any remaining questions
- **Introduction:**
  - Importance of advanced data visualisation
  - Advanced visualisations can provide more insights by combining multiple plots into a single figure, allowing for easier comparison and analysis.
  - Overview of dashboard layout techniques

Principles of Effective Data Visualisation

- **Clarity:**
  - Ensure the visual is easy to understand.
  - Avoid clutter and unnecessary elements.
- **Accuracy:**
  - Represent data truthfully.
  - Avoid misleading scales or distortions.
- **Efficiency:**
  - Convey information quickly.
  - Use visual cues like color and sise effectively.

Dashboard Design Best Practices

- **Consistency:**
  - Maintain a uniform style and layout.
  - Use consistent colors, fonts, and iconography.
- **Relevance:**
  - Include only necessary information.
  - Prioritise key metrics and insights.
- **Interactivity:**
  - Enhance user engagement.
  - Allow for exploration and filtering of data.

What Makes Plots Advanced?

- Combining Multiple Plots:
  - Use subplots to create complex visualisations within a single figure.
  - Examples: 2x2 grids, multi-panel plots.
- Customisations:
  - Annotations, colour schemes, and styles to enhance readability.
  - Interactive elements for user engagement.

- Data Transformation:
  - Using statistical transformations (e.g., regression lines, smoothing).
  - Aggregating and summarising data for better insights.

Import Required Libraries

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

Reading Data

- We'll start by reading the processed weather data from the CSV file.

```
df = pd.read_csv('processed_weather_data.csv')
df['Datetime'] = pd.to_datetime(df['Datetime'])
df.head()
```

Creating a 2x2 Grid of Plots

- Subplots allow us to create multiple plots within a single figure.
- We'll create a 2x2 grid of plots to visualise different aspects of the weather data.

```
fig, axs = plt.subplots(2, 2, figsize=(14, 10))
```

fig & axs

- fig: Entire figure container object
- axs: Array of subplot axes
- You can use fig to adjust figure-wide settings (like the overall size) and
- axs to customise each subplot individually.

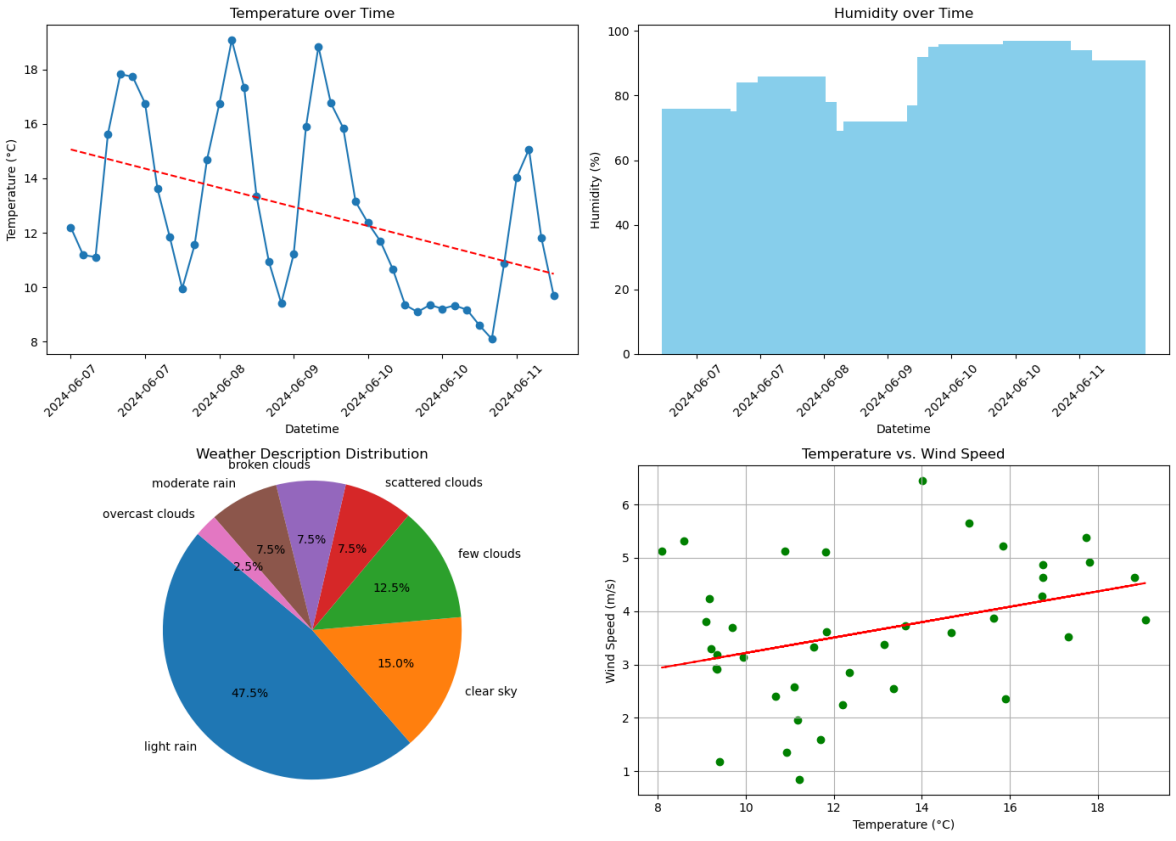
Access each subplot by indexing into axs

```
axs[0, 0].plot([1, 2, 3], [1, 4, 9])  # Top-left subplot
axs[0, 1].plot([1, 2, 3], [1, 2, 3])  # Top-right subplot
axs[1, 0].plot([1, 2, 3], [9, 4, 1])  # Bottom-left subplot
axs[1, 1].plot([1, 2, 3], [3, 2, 1])  # Bottom-right subplot

plt.tight_layout()
plt.show()

# Save the figure
fig.savefig('my_plot.png')
```

Four plots



Plot 1: Line plot for Temperature over Time

```
# Select every 6th datetime value for x-ticks
x_ticks = df['Datetime'][::6]

axs[0, 0].plot(df['Datetime'], df['Temperature (C)'], marker='o')
axs[0, 0].set_title('Temperature over Time')
axs[0, 0].set_xlabel('Datetime')
axs[0, 0].set_ylabel('Temperature (°C)')
axs[0, 0].set_xticks(x_ticks)
axs[0, 0].tick_params(axis='x', rotation=45)
```

Add line of best fit for Temperature over Time

```
s = np.polyfit(df['Datetime'].astype(np.int64) // 10**9, df['Temperature (C)'], 1)
p = np.poly1d(s)
axs[0, 0].plot(df['Datetime'], p(df['Datetime'].astype(np.int64) // 10**9), "r--")
```

Plot 2: Bar plot for Humidity over Time

```
axs[0, 1].bar(df['Datetime'], df['Humidity (%)'], color='skyblue')
axs[0, 1].set_title('Humidity over Time')
axs[0, 1].set_xlabel('Datetime')
axs[0, 1].set_ylabel('Humidity (%)')
axs[0, 1].set_xticks(x_ticks)
axs[0, 1].tick_params(axis='x', rotation=45)
```

Plot 3: Pie chart for Weather Description

```
weather_counts = df['Weather'].value_counts()
axs[1, 0].pie(weather_counts, labels=weather_counts.index, autopct='%1.1f%%', startangle=140)
axs[1, 0].set_title('Weather Description Distribution')
axs[1, 0].axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
```

Plot 4: Scatter plot for Temperature vs. Wind Speed

```
axs[1, 1].scatter(df['Temperature (C)'], df['Wind Speed (m/s)'], c='g', marker='o')
axs[1, 1].set_title('Temperature vs. Wind Speed')
axs[1, 1].set_xlabel('Temperature (°C)')
axs[1, 1].set_ylabel('Wind Speed (m/s)')
axs[1, 1].grid(True)
```

Add line of best fit for Temperature vs. Wind Speed

```
s = np.polyfit(df['Temperature (C)'], df['Wind Speed (m/s)'], 1)
p = np.poly1d(s)
axs[1, 1].plot(df['Temperature (C)'], p(df['Temperature (C)']), "r--")

# Adjust layout
plt.tight_layout()
plt.show()
```

Customising Subplots

- Customising subplots can enhance their readability and presentation.
- Annotations
- Colours and Styles

Setup a plot

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(df['Datetime'], df['Temperature (C)'], marker='o')
ax.set_title('Temperature over Time')
ax.set_xlabel('Datetime')
ax.set_ylabel('Temperature (°C)')
ax.set_xticks(x_ticks)
ax.tick_params(axis='x', rotation=45)
ax.grid(True)
```

Annotate the highest temperature

```
max_temp = df['Temperature (C)'].max()
max_temp_time = df.loc[df['Temperature (C)'] == max_temp, 'Datetime'].iloc[0]
ax.annotate(f'Max Temp: {max_temp:.2f}°C', xy=(max_temp_time, max_temp),
             xytext=(max_temp_time, max_temp+2),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.show()
```

Customising Colours and Styles

```
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(df['Datetime'], df['Temperature (C)'], marker='o', linestyle='--', color='b')
ax.set_title('Temperature over Time', fontsize=14, fontweight='bold')
ax.set_xlabel('Datetime', fontsize=12)
ax.set_ylabel('Temperature (°C)', fontsize=12)
ax.set_xticks(x_ticks)
ax.tick_params(axis='x', rotation=45)
ax.grid(True, linestyle='--', linewidth=0.5)

plt.show()
```

Summary

- In this session, we learned how to visualise weather data using advanced techniques in Matplotlib. We created subplots and grids, combined multiple plots into a single figure, and customised these plots for better presentation.

Homework - Design Dashboard

- **Task:**
  - Modify, Refine and enhance existing visualisations
    - For example: remove line of best fit, change colours, titles

  - Create additional subplots to visualise other aspects of the weather data (e.g., wind speed, pressure).
    - For example: compare min/max over 5 days, Word Cloud for description
  - Experiment with different types of subplots and customisations to enhance the visualisations.
  - Inspiration form internet or Search
    - Python Graph Library
    - Matplotlib Gallery

- **Submission:**
  - Upload the refined notebook and script to the shared folder