COMP2200/COMP6200 – Week 9A Matplotlib foundations

Greg Baker

7th October 2025

Public holiday

- Welcome back from break
- The "live" lecture is a collection of short videos. This is the first one.
- Practicals are back on (except for the Monday rebound session)

Starting a Matplotlib workflow

Step-by-step mental model

1. Import the libraries.
2. Prepare your data in a DataFrame or Series.
3. Request a figure/axes pair with `plt.subplots`.
   - You can skip this step and use a default-created figure. Don't. You'll regret it.
4. Call plotting methods *on the axes*.
   - Or, pass `ax=...` as an argument to the `pandas` graphing functions
   - Or (if you went with default-created), use `plt.*` functions
5. Label everything
6. Show or save.

Figures and axes

```
3  import matplotlib.pyplot as plt
4  import pandas as pd
5
6  fig, ax = plt.subplots(figsize=(6, 4))
7  print(type(fig))  # matplotlib.figure.Figure
8  print(type(ax))  #
       matplotlib.axes._axes.Axes
```

- A figure is the whole canvas. An axes object is the actual plot area.
- Use `plt.subplots()` to ask Matplotlib for both at once.
- `figsize=(width, height)` uses inches. Bump it up if the default feels cramped.
  - But inside a Jupyter notebook, it will get scaled anyway.

Imports and sample data

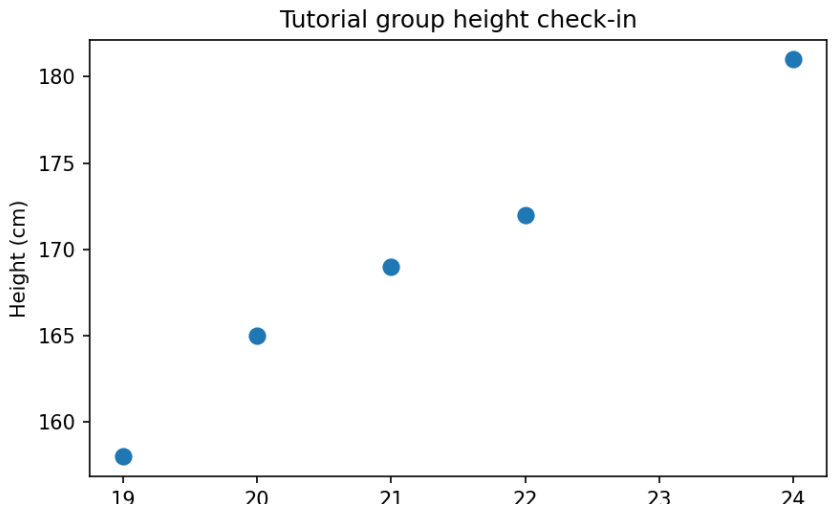
```
10  heights = pd.DataFrame(
11      {
12          "student": ["Ana", "Bailey",
                "Casey", "Dylan", "Em"],
13          "height_cm": [158, 172, 165, 181,
                169],
14          "age": [19, 22, 20, 24, 21],
15      }
16  )
```

- We have used `pandas.read_csv` to create dataframes in the past, but we can code them in-line as well.
- That is just a function call with a dictionary as an argument, where the keys are strings and the values are lists

A first scatter plot

```
18  ax.scatter(x=heights["age"],
        y=heights["height_cm"], s=60)
19  ax.set_xlabel("Age (years)")
20  ax.set_ylabel("Height (cm)")
21  ax.set_title("Tutorial group height
        check-in")
22  fig.tight_layout()
```

- `s` controls marker area in points squared – adjust until the dots read clearly.
- `tight_layout()` asks Matplotlib to tidy labels automatically.



Numbers or labels look squashed?

- Increase `figsize`
  - In jupyter the figure won't appear bigger, but the fonts will appear smaller
  - Because the image got scaled
- or rotate tick labels: `ax.tick_params(axis="x", labelrotation=45)`.

Saving figures like a pro

Consistent exports

```
from pathlib import Path

output_path = Path.cwd() /
    "height_scatter.png"
fig.savefig(output_path, dpi=150,
    bbox_inches="tight")
```

- Use `Path` objects so your code works on Windows, macOS, and Linux.
- `bbox_inches="tight"` trims whitespace without chopping labels.
- Saving both PNG and PDF covers reports and slides in one go.
- In Google Colab, open the folder pane on the left, right-click your saved file, and choose **Download** to copy it to your laptop.

Using pandas plot methods

Same plot, pandas style

```
13  fig, ax = plt.subplots(figsize=(6, 4))
14  heights.plot.scatter(x="age",
        y="height_cm", s=60, ax=ax)
15  ax.set_xlabel("Age (years)")
16  ax.set_ylabel("Height (cm)")
17  ax.set_title("Tutorial group height
        check-in")
```

- Instead of calling `ax.scatter()`, use `DataFrame.plot.scatter()`
- Pass the axes object via `ax=...` to control which plot area to use
- Still need to request `fig, ax` first – no shortcuts there

Why pandas plot methods?

- Slightly more compact when working directly with DataFrame columns
- Handles some labelling automatically (column names become labels)
- Still gives you full control via the `ax=` parameter
- Under the hood, it's just calling matplotlib for you

When to use which

Use `ax.scatter()` when you need fine control. Use `df.plot.scatter()` when convenience matters more.

The result: identical

Matplotlib approach:

```
18  ax.scatter(x=heights["age
        y=heights["height_cm"]
        s=60)
```

Pandas approach:

```
15  heights.plot.scatter(x="a
        y="height_cm",
        s=60, ax=ax)
```

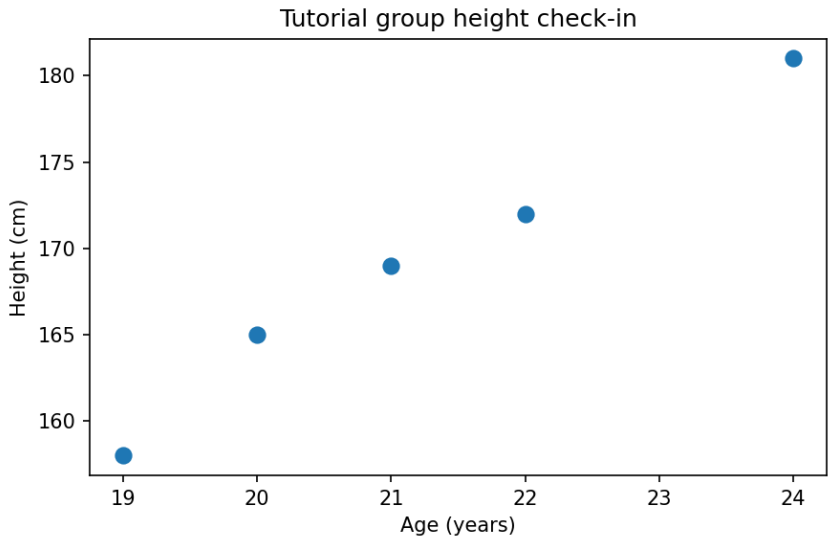


Figure: *

Same data, same result

Wrap-up for part A

- You can spin up Matplotlib from scratch and control the canvas size.
- You have a reliable five-step recipe that mirrors Orange widgets.