COMP2200/COMP6200 – Week 9C Trend lines and tidy layouts

Greg Baker

7th October 2025

Setting the stage

When to use what

When to use what: matching viz to questions

- **Scatter**: exploring relationships between two numeric variables (e.g. age vs height).
- **Line**: tracking a metric across an ordered dimension (e.g. weeks in session).
- **Bar**: comparing categories when order does not matter (e.g. majors in the cohort).
- Ask: "What question am I answering? What pattern am I trying to highlight?" (ULO1)

Decision prompt

If the question involves how one value changes alongside another, default to a scatter first so the raw points stay visible.

Visual choices shape interpretation

- Squeezed `figsize` can exaggerate slopes; generous space keeps trends honest.
- Colours, marker size, and axes limits nudge the reader toward specific stories.
- Always explain why this view answers the question and what caveats remain (ULO4).

Example

Comparing two tutorial groups? Scatter with group-coded markers shows overlap. A line plot would fabricate continuity that does not exist.

Quick recap

- Part A gave us the Matplotlib scaffold and confidence with `fig, ax = plt.subplots`.
- Part B coloured categories and sprinkled in guidance lines.
- Now we learn how to fit simple lines, manage multiple axes, and export publication-ready figures.

Learning goals

1. Plot a fitted line through a data set with scikit-learn and pandas.
2. Combine scatter plots and line plots on the same axes.
3. Lay out multiple subplots and label them clearly.
4. Save figures with consistent sizing for assignments.

Fitting a simple line

Prepare the data

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import
    LinearRegression

temps = pd.DataFrame(
    {
        "day": list(range(1, 11)),
        "max_temp": [18, 19, 21, 22, 24,
            23, 25, 26, 27, 28],
        "attendance": [42, 45, 47, 51, 54,
            53, 58, 59, 63, 66],
    }
)
```

Prepare the data

- A tiny synthetic set keeps the maths gentle but shows a trend.
- Always start with the scatter so people see the raw observations.

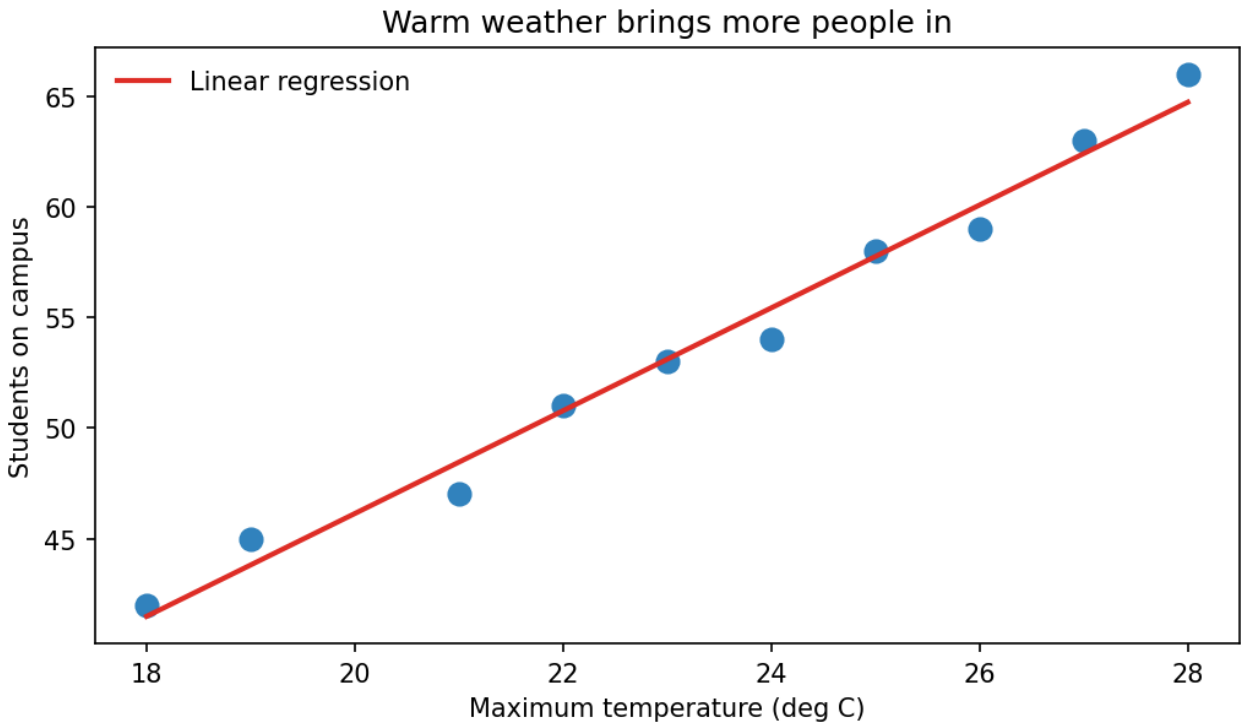
Add a least-squares line

```
model = LinearRegression()
model.fit(temps[["max_temp"]],
    temps["attendance"])

min_temp = temps["max_temp"].min()
max_temp = temps["max_temp"].max()
points = 50
step = (max_temp - min_temp) / (points - 1)
temp_range = [min_temp + i * step for i in
    range(points)]
prediction_frame =
    pd.DataFrame({"max_temp": temp_range})
predicted_attendance =
    model.predict(prediction_frame).tolist()
```

Add a least-squares line

- `LinearRegression` fits the least-squares line; we feed it a column of temperatures and attendance targets.
- Build a smooth x-range manually, wrap it in a DataFrame, call `predict`, and plot the red line alongside the scatter.
- Legends now combine scatter and line handles automatically.



What story does the line tell?

- Linear fits answer: "Does attendance rise steadily with temperature?" (ULO1).
- Consider alternatives: rolling averages smooth noise, polynomial fits show curves.
- Always state the assumptions and warn about extrapolating beyond observed data (ULO4).

Discussion prompt

Would a curved line change how campus planners react? Why or why not?

Small multiples for comparisons

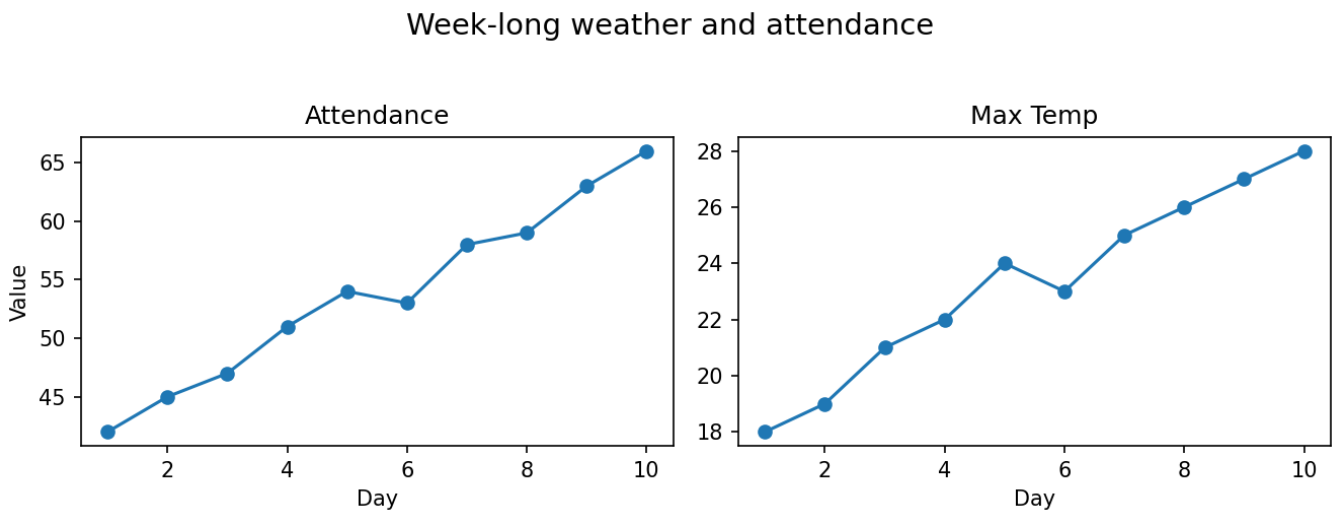
Create subplots the friendly way

```
metrics = temps.melt(
    id_vars="day",
    value_vars=["max_temp", "attendance"],
    var_name="metric",
    value_name="value",
)

fig2, axes = plt.subplots(1, 2, figsize=(9,
    3.5), sharex=True)
for (metric, group), axis in
    zip(metrics.groupby("metric"), axes):
    axis.plot(group["day"], group["value"],
        marker="o")
    axis.set(title=metric.replace("_", "
        ").title(), xlabel="Day")
```

Create subplots the friendly way

- `sharex=True` synchronises the x-axis, so days line up across plots.
- Iterating with `zip` keeps the code approachable for students new to Python.



Bar charts with pandas

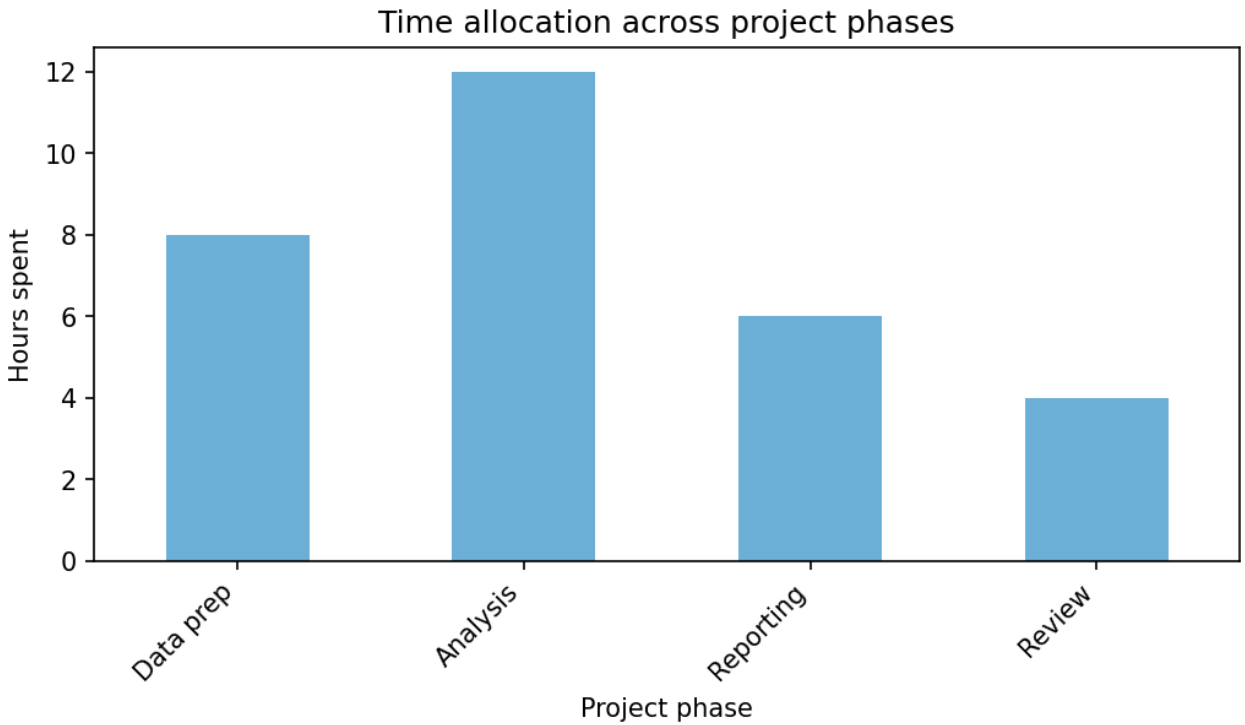
Creating a bar chart

```
categories = pd.DataFrame(
    {
        "category": ["Data prep",
           "Analysis", "Reporting",
           "Review"],
        "hours": [8, 12, 6, 4],
    }
)

fig3, ax3 = plt.subplots(figsize=(7, 4.2))
categories.plot.bar(x="category",
   y="hours", ax=ax3, legend=False,
   color="#6baed6")
```

Creating a bar chart

- `DataFrame.plot.bar()` is the pandas shortcut for bar charts.
- Pass the axes object with `ax=` so you retain full control over labels and styling.
- Rotate x-tick labels with `rotation=45, ha="right"` for readability.



Part C wrap-up

- You can fit and plot a line through data with scikit-learn and Matplotlib.
- Subplots let you compare metrics without throwing students into seaborn yet.
- You know how to export figures at the right size for assignments and reports.

Looking ahead

- Practical this week reinforces scatter plots, legends, and figure exports.
- Bring any stubborn plots to drop-ins – tutors love a good Matplotlib bug hunt.
- Next week we pivot to seaborn for quicker statistical plots.