COMP2200/COMP6200 Practical Exercise (Week 9): Story-first Matplotlib

School of Computing

Preparation

Choose your environment: either run locally with `uv` or use Google Colab.

1. **Local with `uv`**:
   (a) Install `uv`.
   MacOS/Linux: `curl -LsSf https://astral.sh/uv/install.sh | sh`
   Windows: `powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"`
   (b) Create a folder for this practical (for example, `week9-prac`) and open a terminal in it.
   (c) Set up your environment:

   ```
   uv init
   uv add pandas numpy scikit-learn matplotlib
   ```

   (d) Launch Jupyter with `uv run --with jupyter jupyter lab` (or `uv run --with jupyter jupyter notebook`).
2. **Google Colab**:
   (a) Open colab.research.google.com and create a new notebook.
   (b) Upload `student_engagement.csv` from iLearn or this week's zip file.
   (c) Install packages with `!pip install pandas numpy scikit-learn matplotlib`.

Part A — Plot type face-off (10 min)

Form groups of four. Lay out the provided "question cards" (e.g. "Which prac groups are drifting behind on project progress?", "Did the drop-in blitz help quiz scores?"), plus the chart-type cards from Weeks 2–3.

1. Pair each question with the *two* chart types you think could answer it.
2. Justify your picks using language from the Week 9 lecture (what structure the data has, and what story each chart highlights).
3. Swap cards with a neighbouring group and see if you agree with their matches.

This is a quick refresher on reading the question first and reaching for the chart second.

Part B — Orange to Python remix (35 min)

We will reuse Orange skills from Weeks 5–7 and turn them into Matplotlib storyboards. The engagement dataset includes a few missing entries and the features live on different scales, so every workflow needs an explicit preprocessing step.

B1. Cluster before you colour

1. Open Orange, create a new workflow and load `student_engagement.csv` with the **File** widget.
2. Add **Select Columns** to keep `prep_minutes`, `dropin_visits`, `practice_quiz_attempts`, and `quiz3_score`.
3. Slot in **Impute** (median strategy) and **Normalize** (z-score) widgets before **KMeans**. The clusters will not compute without these steps.
4. Attach a **KMeans** widget ($k = 3$) and inspect the scatter plot output. Export the labelled data (`File` → `Save Data`) so you can cross-check later.

B2. Translate the pipeline

Launch your notebook and mirror the workflow.

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

engagement = pd.read_csv("student_engagement.csv")
features = ["prep_minutes", "dropin_visits", "practice_quiz_attempts", "quiz3_score"]

cluster_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler()),
    ("cluster", KMeans(n_clusters=3, random_state=2025))
])

engagement["cluster"] = cluster_pipeline.fit_predict(engagement[features])
centroids = cluster_pipeline.named_steps["scaler"].inverse_transform(
    cluster_pipeline.named_steps["cluster"].cluster_centers_
)
```

Create a scatter plot that uses the lecture's figure/axes pattern.

1. Plot `prep_minutes` on the x-axis and `quiz3_score` on y.
2. Colour points by `cluster`; use a dict mapping (`{0: #1b9e77, ...}`) so you can quote the palette in your legend.
3. Give each cluster a proxy artist and call `ax.legend(title="Cluster story", frameon=False)`.
4. Annotate the inverse-transformed centroids (the `centroids` array above) with `ax.text` so someone skimming the plot understands the takeaway.

B3. Logistic sanity check

Revision from Week 8: can we predict who passes Quiz 3?

1. Assemble a `Pipeline`: inside the `ColumnTransformer`, wrap the same numeric features with `SimpleImputer(strategy="median")` and `StandardScaler`. Append `LogisticRegression(max_iter=500)` as the final step.
   A skeleton to start from:

   ```
   from sklearn.compose import ColumnTransformer
   from sklearn.impute import SimpleImputer
   from sklearn.linear_model import LogisticRegression
   ```

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

numeric_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler()),
])

preprocess = ColumnTransformer([
    ("num", numeric_pipeline, features)
])

clf = Pipeline([
    ("preprocess", preprocess),
    ("model", LogisticRegression(max_iter=500))
])
```

2. Evaluate the model with stratified cross-validation, reporting both accuracy and macro-F1 (`cross_validate(. scoring=["accuracy", "f1_macro"])`).
3. Fit the pipeline on the full dataset, then produce a tidy DataFrame with columns `actual` (from `passed_quiz`), `predicted` (0/1) and `probability` (use `predict_proba`).
4. Plot the probabilities against `prep_minutes`, colour by `cohort`, and draw a horizontal line at the 0.5 decision threshold.
5. Use `ax.fill_between` to shade the "at risk" region and label it directly on the chart.

Part C — Trend lines and small multiples (30 min)

Now we practise this week's Matplotlib layout tools.

1. Build a weekly summary table:

```
weekly = (engagement
          .groupby("week")
          .agg({"prep_minutes": "mean",
                "quiz3_score": "mean",
                "dropin_visits": "sum"})
          .reset_index())
```

2. Create a two-panel figure with `fig, axes = plt.subplots(1, 2, figsize=(10, 4), sharex=True)`.
3. Left subplot: line plot of average `prep_minutes` per week and a least-squares trend line (`numpy.polyfit`). Annotate the slope in plain English.
4. Right subplot: bar plot of total `dropin_visits` per week with a secondary axis for mean `quiz3_score`. Use contrasting but accessible colours and call out Week 9's spike with `ax.annotate`.
5. Add a figure-level title, use `fig.tight_layout()` and finish with `fig.savefig("week9-engagement.png", dpi=150, bbox_inches="tight")`.

Part D — Gallery walk and feedback (15 min)

Stick your final Matplotlib figure on the shared drive or Discord channel.

1. Swap laptops with a neighbour. Use the "two stars and a wish" format: two specific things that work, one suggestion.

2. Check that legends, colour choices and annotations make sense without additional commentary.
3. Capture any peer feedback as TODO comments in your notebook so you can iterate before the assignment.

Stretch goals

- Re-run the clustering with DBSCAN. Which epsilon/min samples produce the cleanest separation, and how does that change your colour story?
- Export a second figure sized for a report (`figsize=(6, 4)`). Compare the typography and readability between the slide and report versions.
- Build a "small multiples" layout (`fig, axes = plt.subplots(2, 2)`) splitting by `cohort`. Does the story change for postgrads versus undergrads?