COM6018 Assignment 1 - UK Energy Generation Analysis

Document version 1.0. – 24th October 2025

Due: 15:00, Friday 7th November 2025

Contents

[TOC]

---

1. Introduction

Welcome to the first assignment for COM6018! In this exercise, you'll get hands-on experience with data analysis and visualization by exploring how the **UK's energy generation mix** changes over time and across seasons.

The "energy generation mix" refers to the different sources used to generate electricity — such as solar, wind, gas, nuclear, and imported power. Understanding how these sources contribute at different times helps us see patterns in renewable energy usage, seasonal variations, and the role of fossil fuels in meeting demand.

Your task is to create **four insightful plots** using real UK energy data, and to answer **one quantitative question for each plot**. The questions are designed so that you can estimate the answers

directly from your visualizations — this means reading values from axes, comparing visual patterns, and interpreting trends from the plots you create.

To help you get started, we've provided Python template files that include the basic structure for each task. You'll be completing these templates with your own code to load the data, generate the plots, write captions, and provide answers. The entire project is managed through **GitHub Classroom**, which will handle distribution, submission, and automatic report generation.

This assignment will give you practical experience with:

- Loading and preprocessing real-world datasets
- Creating effective data visualizations using Python
- Presenting your analysis in a professional report format
- Using version control (Git) to manage your work

Don't worry if some aspects seem challenging at first — we've designed the assignment with clear tasks and helpful tools to support you along the way. Take it one step at a time, and remember to commit your work regularly! Use the Blackboard discussion boards and the lecture and lab sessions to ask questions if you need help.

---

2. Repository & Environment

After accepting the GitHub Classroom invitation, you will receive a private repository with the following contents:

```
data
    energy_demand_2024.csv
    energy_mix_2024.json
pyproject.toml
README.md
report
    figures
        plot_01.png
        plot_02.png
        plot_03.png
        plot_04.png
    template.tex
src
    __init__.py
    generate_report.py
    task_01.py
    task_02.py
    task_03.py
    task_04.py
    utils.py
tests
    test_load_data.py
tools
```

```
        __init__.py
        push_with_report.py
```

Use `git clone <your-repo-url>` to clone your repository locally.

Setting up the Environment

You **must** use `uv` for dependency management and execution.

Install dependencies:

```
uv sync
```

Activate the environment:

```
# macOS/Linux
source .venv/bin/activate
# Windows (PowerShell)
.venv\Scripts\Activate.ps1
```

The assignment involves generating a `LaTeX` report. We recommend that you install `tectonic` so that you can compile the LaTeX report and view the final PDF.

---

3. Workflow & Report Generation

Your repository is a self-contained Python project that automatically builds a short LaTeX report summarising your analyses. You will complete and test your code locally, and GitHub Classroom will compile the LaTeX code into a PDF document on submission.

What You Are Provided With

The repository includes:

File/Folder	Purpose
`data/energy_demand_2024.csv`	The dataset containing half-hourly UK energy demand.
`data/energy_mix_2024.json`	The dataset containing half-hourly UK generation data by source.
`src/utils.py`	Helper functions for loading and processing the data. You will complete `load_data()`.
`src/task_01.py` – `src/task_04.py`	Template files for the four tasks. Each file defines functions to produce a plot, caption, and answer.
`src/generate_report.py`	Script that assembles your outputs into a single LaTeX report.
`tests/`	Simple tests that help check basic functionality (e.g. data loading).

File/Folder	Purpose
`report/`	Where your generated `.tex` and `.pdf` report files will appear.
`tools/push_with_report.py`	A tool to help you build and push your report.

How the System Fits Together

1. **You implement the load_data() function**
   - In `src/utils.py`, complete the `load_data()` function to read and preprocess the CSV and JSON data files.
   - You can run `pytest tests/test_load_data.py` to check your implementation.
2. **You implement the task functions**
   - In each `src/task_01.py` – `src/task_04.py` file, complete:
     - `make_plot()` – creates and returns a Matplotlib figure.
     - `get_caption()` – returns a one- or two-sentence caption for the figure.
     - `get_answer()` – returns a short textual answer to the quantitative question.
   - All plots should be generated programmatically (not by manually saving figures).
3. **Local testing and previewing**
   - You can test each task independently, e.g.:

     ```
     python src/task_01.py
     ```

     This will display the plot and verify that your functions run correctly.
4. **Building the LaTeX report**
   - Once all four tasks run correctly, build the combined report:

     ```
     python src/generate_report.py
     ```

   - This script:
     - Imports your task modules (`task_01` – `task_04`).
     - Calls each of their functions to generate the plots and text.
     - Saves all figures to `report/figures/`.
     - Inserts your captions and answers into the LaTeX template (`report/assignment1.tex`).
5. **(Optional) Compile the LaTeX report**
   - To view the final PDF, compile the LaTeX file:

     ```
     tectonic report/assignment1.tex
     ```

   - The resulting `report/assignment1.pdf` is what will be marked.

4. Committing Changes

Make **small, frequent commits** as you work — ideally whenever you complete a meaningful change (e.g., you've fixed a bug, improved a plot, or finished a paragraph). This helps you:

- Keep a clear record of how your work developed,
- Revert easily if something breaks, and
- Show your workflow and understanding of the task.

Example:

```
git add .
git commit -m "task_01: labelled axes and added grid"
```

> We will look at the sequence of commits as part of understanding your approach and progress. Clear, incremental commits make it easier for us to appreciate your working process and will lead to a better mark.

Push to GitHub regularly, but not after every single commit. Push at sensible milestones, such as:

- When you finish a task file,
- Before you stop working for the day,
- When you want the GitHub Action to build the report for you (if you don't have Tectonic locally),
- And a **final push** to submit.

```
git push
```

Note: Each push runs the GitHub Action and uses classroom quota. Please avoid pushing on every tiny change.

Recommended: one-step push (sync-safe)

This helper regenerates the report locally, commits report files if needed, and pushes — but only if your src/ code is already committed, so the report matches your code. Commit your code changes as usual before running this.

```
python tools/push_with_report.py
```

What it does:

1. Refuses to run if `src/` has uncommitted changes (commit your work first).
2. Runs `src/generate_report.py` locally (uses uv run if available).
3. Stages `report/assignment1.tex` and `report/figures/`.
4. Commits only if those changed (regenerating the report [commit made by script]).
5. Pushes; if remote is ahead (new `submission/assignment1.pdf`), pulls with rebase and retries.

Note: Make your own commits for code changes as usual. This script just ensure that the report files are up to date on GitHub when you push.

Automatic Report Generation (GitHub Actions)

When you push to GitHub:

1. The GitHub Action will compile your existing `report/assignment1.tex` file into a PDF. It does not run your Python code or regenerate the LaTeX file.
2. The compiled PDF will be available as a **workflow artifact**. To download it:
   - Go to the **Actions** tab in your repository.
   - Click on the most recent workflow run.
   - Scroll down to the **Artifacts** section and download the PDF.

Allowed libraries

You may use:

- **Python's standard library**, and
- Any modules already installed in the provided environment.

You **must not** install or use additional third-party packages.

Summary of What You Should and Should Not Do

Do	Do Not
Edit only `src/task_01.py` to `src/task_04.py` and `src/utils.py`.	Edit `generate_report.py` or any other file.
Use only standard libraries and the provided environment.	Install extra Python packages.
Commit and push your work regularly.	Wait until the deadline to push your code.
Check your generated `assignment1.pdf`.	Assume the GitHub PDF will "just work" without checking.

---

5. Tasks

You have been provided with template code, `src/utils.py` and `src/task*.py`, which you will complete to produce the required plots and answers.

In `src/utils.py`, you will need to complete:

- the `load_data()` function to read the data files

In each of the `src/task*.py` files, you will need to complete:

- the `make_plot()` function to generate the appropriate plot
- the `get_caption()` function which simply returns a string caption for the plot
- the `get_answer()` function which returns a string answering the question posed for that plot.

The plots and corresponding questions are described below.

Plot 1 – Monthly Average Energy Mix

Plot requirements:

- Create a **3×4 grid of pie charts**, one for each month **January–December**.
- Each pie chart shows the **percentage contribution** of the following energy sources: solar, wind, gas, nuclear, imports and other (where 'other' is *all* remaining sources combined).
- Use **consistent colours** for each energy source across all months.

Questions (answer from your plot): In which month is the biggest percentage contribution from **gas generation**, and what is that **percentage**? In which month is the smallest percentage contribution from **solar generation**, and what is that **percentage**?

Plot 2 – Average Generation by Time of Day for Each Season

Plot requirements:

- Create a **2×2 grid of line plots** showing **average energy generation (MW) by time of day (0–24 h)**.
- Each panel corresponds to a **3-month season** (*Winter, Spring, Summer, Autumn*). Consider Dec-Jan-Feb to be Winter; Mar-Apr-May to be Spring; Jun-Jul-Aug to be Summer; and Sep-Oct-Nov to be Autumn.
- Include separate lines for **solar, wind, gas, and nuclear**.
- Keep the **y-axis comparable across panels** so seasonal differences are visible.

Questions (answer from your plot):

- At what **time of day** does **gas generation peak in winter**, and what is the total gas generation (in MW) at that peak?
- At what **time of day** does **gas generation peak in summer**, and what is the total gas generation (in MW) at that peak?

Plot 3 – Distribution of Midday Solar and Wind Generation by Month

Plot requirements:

- Produce **seaborn distribution plots** stacked vertically:
  - **Top:** solar generation (MW) during midday (**11:00–13:00**).
  - **Bottom:** wind generation (MW) during the same hours.
- The **x-axis** shows **month**, i.e. a separate distribution for each month.
- Choose the most suitable way to visualise the distributions (e.g., strip plot, swarm plot, violin plot, box plot, etc.).

Questions (answer from your plot): Using the difference between the monthly minimum and maximum values shown in your plot, which energy source has the **largest monthly range** on average? For that source, which month has the **largest range**, and what is that range (in MW)?

Plot 4 – Relationship Between Solar and Wind Generation

Plot requirements:

- Create a **2×2 grid of scatter plots**, one for each month (**May–August**).
- For each month, plot **solar generation (MW)** on the **x-axis** and **wind generation (MW)** on the **y-axis**.
- Each point represents a **midday (11:00–13:00)** observation.
- Include a **fitted line or smooth trend** if helpful.

Questions (answer from your plot): By visual inspection of your plots, is there a **positive or negative correlation** between solar and wind generation during the 11:00-13:00 time period for these months? What is the highest observed wind generation (in MW) when solar generation is **above 6000 MW**?

---

6. Tips and Recommended Approach

- Read this document fully before starting and ask questions early if anything is unclear.
- Make sure that you can run the provided template code and generate the default report before making changes.
- Start by writing the code that loads the data. All tasks will require this. You can use 'pytest tests/test_load_data.py' to check your data loading function.
- Work on one task at a time and commit your changes frequently with clear messages.
- Use functions to avoid code duplication. Functions can be shared between tasks, e.g., by adding them to `src/utils.py`.
- *Plots*: Think carefully about the best way to visualise the data for each task. Be prepared to work iteratively, i.e. making plots and then studying them to see if they are clear. Adapt as necessary. The task description state the broad requirements, but there is a lot of flexibility in the details.
- *Captions*: Make sure your captions describe what is shown in the plot clearly and concisely.
- *Answers*: If you are finding it hard to answer the question using the plot, consider changing the plot to make it easier to read the required values.

---

7. Marking

Each task is worth **10 marks** (**40 total**, i.e. **40 %** of the module). Marks per task:

Component	Description	Marks
Plot	Clarity, labelling, caption and relevance of the visualisation	**6**
Answer	Correct and accurate quantitative estimate(s) derived from the plot	**1**

Component	Description	Marks
Code	Structure, clarity, and efficiency; appropriate use of libraries; appropriate use of git	**3**

You will receive qualitative feedback on each of the four tasks, and aggregate marks for the plot, answer, and code components.

Detailed Rubric

Criterion	Excellent (Full Marks)	Good	Adequate	Poor
Plots (24 marks total)	Correct data shown; clear labels and legends; consistent styling; visually accurate and easy to interpret.	Minor issues (e.g. inconsistent labels or colour use).	Basic structure present but limited clarity or accuracy.	Data incorrect, unreadable, or misleading plot.
Answers (4 marks total)	Estimated values correct and plausibly estimated from plot			Incorrect or implausibly accurate value
Code (12 marks total)	Clean, modular, efficient, well-commented, and fully reproducible.	Mostly clear, with minor redundancy or style issues.	Runs but disorganised or repetitive.	Fails to run, unclear, or incomplete.

---

8. Submission

- Work **individually** in your **GitHub Classroom** repository.
- Make frequent commits as you work to document your progress.
- Review the pdf report generated as `report/assignment1.pdf` to ensure your plots and answers appear correctly. This is the document that will be marked.
- Your **latest commit before the deadline** (**15:00, Fri 7 Nov 2025**) will be marked.

---

9. Academic Integrity

Discuss ideas, but all **code, plots, and text must be your own**. Do **not** share your repository or solutions. Misconduct (plagiarism, collusion, unacknowledged AI use) will be handled under University regulations.

See: https://www.sheffield.ac.uk/new-students/unfair-means

End of Document