Mathematical Modeling for Computational Science
AM115/AM215 Fall 2025

Harvard John A. Paulson School of Engineering and Applied Sciences

Michael P. Brenner
Chris Gumb

Homework 3

Extreme Value Distributions, Dimensional analysis and Time Series

Issued:	October 28, 2025
Due:	November 13, 2025

Problem 1: Extreme Value Distributions

Deliverables: For this problem, you are required to submit your analysis, plots, and the code you develop.

1. `/P1/answers_P1.pdf` containing the text answers, analysis, and plots for this problem.
2. `/P1/modules.py` containing the Python class for the analysis.
3. `/P1/main.py` script that uses the class to generate your results.
4. `/P1/data/` folder containing any new dataset you find for the final subquestion.

In class, we discussed extreme value distributions and their universality. This problem asks you to build a reusable software tool to explore these concepts computationally. Your goal is to implement a single, well-structured Python class that can perform extreme value analysis on both simulated and real-world data. This will streamline your work and maximize code reuse across all parts of the problem.

a) Create a Python class named `ExtremeValueAnalyzer` in a file called `modules.py`.

- The class should be initializable to handle two cases: generating artificial data from a given distribution (e.g., `scipy.stats.pareto`) or analyzing a provided data array.
- Implement a method to simulate trials, extract the extreme values, and return them.
- Implement a method to fit a Gumbel distribution to a set of extreme values.
- Implement a method that plots a histogram of the extreme values against the fitted Gumbel distribution.

Using your class, computationally verify the universality property for a power-law distribution (`scipy.stats.pareto` with $b > 4$) and at least one other distribution of your choice from `numpy.random` or `scipy.stats`. Your `main.py` script should import and use your class to generate these results.

b) Use your `ExtremeValueAnalyzer` class to analyze the provided dataset of 100-meter dash running times. Compare the distribution of the fastest times in this dataset to the theoretical extreme value distribution. Does the data agree with the theory? If not, can you hypothesize what might cause the discrepancy?

c) Instantiate a new object of your class to analyze the racing times of Usain Bolt from the provided dataset. Using your class methods, determine whether his times are consistent with the extreme value distribution. Can you draw any conclusions or make any hypotheses based on your analysis?

d) Find another example of a real-world dataset where extreme values naturally occur. Place the data in the `/P1/data/` directory. Use your `ExtremeValueAnalyzer` class to analyze this data and compare the results to theory. A prize will be awarded for the most creative and insightful solution!

Problem 2: McMahon's Rowers, revisited

Deliverables: For this problem, you are required to submit your answers as text, along with the code used to generate those answers.
Feel free to include plots or any other supporting material.

1. `/P2/answers_P2.pdf` containing the text answers for this problem.
2. `/P2/coverage.html` containing the test coverage report for your code.
3. `/P2/docs/` containing the `HTML` documentation for your code.
4. `/P2/library/` containing **ONLY** your library code.

In class we discussed McMahon's argument for rowers, which is an example of intelligent estimation. Use the provided dataset from the 2022 and 2023 World Championship to analyze whether it obeys McMahon's $N^{1/9}$ law.

a) Your task is to build a Python library that exposes a predefined interface to the user. The library must include a test suite and comprehensive documentation.

   i) The library should expose the following methods:
   - `fit`: Fits the model to the provided data.
   - `evaluate`: Outputs at least two metrics to assess the model's performance. The output **must** be a Pandas DataFrame.
   - `__call__`: The predict method, which should output the estimated speed given an input.

   You may implement any additional methods you require, but the core interface should only expose these three methods.
   Every component must contain a docstring. You are free to choose the format of the docstrings.

   ii) Build a test suite to thoroughly test the exposed interface. Ensure your tests cover expected behavior, edge cases, and data types.
   The line coverage must be at least 95%. Use `pytest` to run your tests.

   iii) Generate HTML documentation using Sphinx and upload it to the folder `/P2/docs/`.

b) List factors in a rowing race that could disrupt the basic argument proposed by McMahon.

c) Suppose you want to implement an improvement to McMahon's model. What function would need to be changed? How would the function's signature change? How would your test suite need to adapt to this new functionality?

   Implement this change as a new method and identify it with the `_new` suffix. ***Hint:*** *Avoid breaking the existing code as much as possible.*

Problem 3: Estimation

Deliverables: for this problem you are required to write your answers as text. Feel free to include plots or any other supporting material.

1. `/P3/answers_P3.pdf` containing the text answers for this problem.

Let's try two estimation questions for yourself. Your goal here is to invent an intelligent estimate to come up with an answer based on facts that you might know.

a) How many total hours do Americans spend commuting to and from work in one year?

b) How many cumulative miles of human hair grow in the United States every 24 hours?

Problem 4: Covid Epidemiology

Deliverables: For this problem, you are required to submit your analysis, plots, and the code you develop.

1. `/P4/answers_P4.pdf` containing the text answers, analysis, and plots for this problem, including a brief description of your Kaggle strategy/implementation/approach.
2. `/P4/library/` containing your Python library code for data querying and SIR modeling.
3. `/P4/tests/` containing your `pytest` test suite.

Our discussion of Covid epidemiology covered both the exploration of large datasets related to Covid, as well as an exploration of Covid models. Your task is to build a Python library to interact with Covid-19 data and apply modeling techniques.

a) Develop a Python library to query the Covid19 database on Google Cloud Platform. Your library should provide functions to retrieve basic information needed for an SIR model (e.g., confirmed cases, deaths, recoveries) for a specified region and time period. It should also be designed with flexibility for users to query additional fields as needed.

b) Using your developed library, retrieve and plot the confirmed Covid cases over time for the county or region where you grew up.

c) Using your library, retrieve and plot mobility information for your chosen region alongside the confirmed case data. Comment on how people responded to the pandemic and whether you think case numbers had anything to do with people's behaviors.

d) The dataset includes search trend columns, for example, 'search trends: common cold'. Investigate whether this data exists for your chosen region and comment on its correlation with Covid cases. Include relevant plots or statistical analysis in your answer.

e) Extend your Python library to include functionality for fitting the confirmed case data from your region to an SIR model, using the code from the Lecture 14 notebook as a reference. How good is the fit? Your answer must include a quantitative assessment of the model's accuracy (e.g., using metrics like R-squared, RMSE, or similar).

f) Create a comprehensive test suite for your Python library, including functions for database interactions and SIR model fitting. Use `pytest` and `monkeypatch` to effectively simulate database queries and ensure your tests are robust and independent of live database access.

g) Participate in a Kaggle competition for time series prediction, focusing on predicting COVID-19 cases on a synthetic dataset. The link to the competition will be posted on the Canvas assignment page soon. You do not need to submit your prediction

CSV with your pset, as this will be submitted directly to Kaggle. However, you must include a brief description of your strategy, implementation, and approach in your `/P4/answers_P4.pdf` submission.

Problem 5: Optional: Repository Contribution

This problem is optional and will reward you with extra points.

This semester, we are launching three new community repositories for collaborative development in computational modeling:

- `covid-modeling-2025`
- `financial-modeling-2025`
- `tournament-modeling-2025`

These repositories are designed to be central hubs for core functionalities relevant to their respective modeling tasks, with contributions from both course staff and students. You are encouraged to familiarize yourself with their structure and existing content by reviewing their `README.md` files.

Your task for this optional problem is to contribute to one or more of these repositories by submitting a Pull Request (PR) that adds useful functionality.

a) Choose one or more of the provided repositories and identify a feature or module that would enhance its capabilities. This could be:

   - A new prediction model or algorithm.
   - An improved data acquisition method.
   - A novel analysis or visualization tool.
   - An optimization for existing code.
   - Any other module that you deem a valuable addition to the library.

   Your contribution should be implemented as a new library module within the repository's structure. **You are highly encouraged to adapt and polish code from your current or previous projects and problem sets for this contribution, rather than starting from scratch.**

   We expect high-quality software engineering practices for your contribution, including:

   - **Good Code Organization:** Adhere to the repository's existing structure and Python best practices.
   - **Comprehensive Tests:** Include a robust `pytest` test suite for your new functionality, aiming for at least 95% line coverage. All tests must pass for your PR to be considered.
   - **Clear Docstrings:** Every function, class, and method should have clear and informative docstrings.
   - **Modern Project Standards:** Ensure your code follows general modern Python project standards.

   You are also encouraged to use the GitHub Issues feature to propose new features, report bugs, or discuss potential contributions with the teaching fellows acting as project managers.

Deliverables: For this optional problem, you are required to submit:

1. `/P5/pr_links.txt` containing the URL(s) to your submitted Pull Request(s) on GitHub.