# Assignment: Mushroom cultivation

Marco Forgione

### Use AI responsibly

If you generate some code using AI:

- Point this out explicitly
- Explain with your own words what the code is doing
- Discuss how you test that the code works as intended
- Is there anything you can do to improve the AI code?
- DO NOT copy answers or code from AI and turn them in as your own; copying is cheating!

### Anti plagiarism check

- Share code and text among groups is forbidden.
- The submitted assignment will be checked for anti-plagiarism.

## Problem and data

A mushroom farm is testing how different storage temperatures affect the spoilage rate of harvested oyster mushrooms. To this end, they perform a trial with $N = 4$ storage temperature levels $x_i, i = 1, 2, 3, 4$. For each temperature level, a batch of $n_i$ mushrooms is stored for 5 days, after which the number of spoiled mushrooms $y_i$ is recorded. The experimental data is summarized in the table below.

Table 1: Effect of storage temperature on oyster mushroom spoilage after 5 days. Each row represents a treatment level.

| Level ID | Storage Temperature $x$ (°C) | Total Mushrooms $n$ | Spoiled Mushrooms $y$ |
|---|---|---|---|
| 1 | 2 | 30 | 2 |
| 2 | 8 | 25 | 4 |
| 3 | 15 | 20 | 5 |
| 4 | 25 | 30 | 20 |

For instance, the storage temperature $x_1 = 2$ °C is tested on a batch of $n_1 = 30$ mushrooms. Out of the $n_1 = 30$ mushrooms, $y_1 = 2$ are observed to be spoiled after 5 days.

## Modeling assumptions

For the probabilistic model, we make the following assumptions:

1. The outcome of the $n_i$ mushrooms within each group $i$ are *independent*. Each animal in the group has probability $p_i$ of death.
2. The probability $p_i$ that a mushroom spoils depends on the temperature level $x_i$ as follows:

   $$p_i = \text{sigm}(\alpha + \beta x_i),$$

   where

   $$\text{sigm}(z) = \frac{1}{1 + e^{-z}}.$$

3. The prior probability of the parameters $\theta = [\alpha \;\; \beta]^\top$ is Gaussian:

$$\alpha \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha^2), \qquad \mu_\alpha = 0, \sigma_\alpha = 2 \tag{1}$$
$$\beta \sim \mathcal{N}(\mu_\beta, \sigma_\beta^2), \qquad \mu_\beta = 0, \sigma_\beta = 1. \tag{2}$$

4. The outcomes in the four groups are independent of each other, given $\theta$.

In answering the following questions, clearly state all the steps of your reasoning with text, formulas, and comments in your code.

### 1.1: Probabilistic model

- Derive and comment the full probabilistic model.

### 1.2: Maximum Likelihood estimation

- Derive an analytical expression of the likelihood function $\mathcal{L}(\theta) = P(y \mid \theta)$.
- Derive an analytical expression of the log-likelihood function $\ell(\theta)$.
- Write a Python function corresponding to the log-likelihood function $\ell(\theta)$ (possibly up to an additive factor).
- Visualize the likelihood $\mathcal{L}(\theta)$ and the log-likelihood function $\ell(\theta)$ in 2D and comment the obtained figures.

  Hints:

  - you may use the `pcolormesh` function of `matplotlib`
  - you need to find appropriate range and step size for both $\alpha$ and $\beta$

- Compute the maximum likelihood (ML) estimates $\alpha^{\mathrm{ML}}, \beta^{\mathrm{ML}}$ of the parameters $\alpha, \beta$ through numerical optimizations.

  Hints:

  - You may use the Python function `scipy.optimize.minimize`.
  - You may look at the figures above to define a good starting point for optimization
  - You may either minimize the likelihood or the log-likelihood. What is your choice?

- Visualize the likelihood function in 2D together with the ML estimate. Comment the obtained figure.

## 1.2: Maximum A Posteriori Estimation

- Derive an analytical expression of the posterior $f(\theta \mid y)$, up to a multiplicative factor not depending on $\theta$.

Hint: exploit the already-obtained likelihood and the known functional form of the Gaussian pdf.

- Derive an analytical expression of the log-posterior $g(\theta) = \log f(\theta \mid y)$, up to an additive factor not depending on $\theta$.
- Write the unnormalized log-posterior (up to a multiplicative/additive factor, respectively) as Python functions.
- Compute the maximum a posteriore estimates $\alpha^{\text{MAP}}, \beta^{\text{MAP}}$.
- Visualize the MAP and the ML estimate together with the unnormalized posterior in 2D. Comment the results.

## 1.3 Brute-force posterior estimation

- Compute a gridding approximation of the *normalized* posterior.
  - Use logarithm transformations for better numerical stability.
- Compute the *marginal* posterior distributions of $\alpha$ and $\beta$ from the gridding approximation.
- Using the grid-based approximation of the posterior, compute the posterior mean of $\alpha$ and $\beta$.

## 1.4 Metropolis

- Obtain a sample-based approximation of the posterior $f(\theta \mid y)$ by implementing the Metropolis algorithm from scratch.
  - Use logarithm transformation for better numerical robustness.
  - Remove the first 1000 samples (they depend on the initial conditions; this is the burn-in period)
  - Run multiple chains and compare visually the posterior distribution provided by the different chains. Do they overlap well?
  - Inspect the trace. Does it show a strong correlation? Which is the rate of acceptance (aim at 50%)?
  - Ajust the algorithm settings if needed, and show how the sampling performs before and after the adjustment.

- Compare the Metropolis samples with the gridding-based approximation of the posterior distribution $f(\theta \mid y)$ and comment the result.