Project 1: 3D printer materials estimation

Use the template material in the zip file `project01.zip` in Learn to write your report. Add all your function definitions on the `code.R` file and write your report using `report.Rmd`. You must upload the following three files as part of this assignment: `code.R`, `report.html`, `report.Rmd`. Specific instructions for these files are in the `README.md` file.

The main text in your report should be a coherent presentation of theory and discussion of methods and results, showing code for code chunks that perform computations and analysis but not code for code chunks that generate functions, figures, or tables.

Use the `echo=TRUE` and `echo=FALSE` to control what code is visible.

The `styler` package addin is useful for restyling code for better and consistent readability. It works for both `.R` and `.Rmd` files.

The `Project01Hints` file contains some useful tips, and the `CWmarking` file contains guidelines. Both are attached in Learn as PDF files.

Submission should be done through Gradescope.

1 The data

A 3D printer uses rolls of *filament* that get heated and squeezed through a moving nozzle, gradually building objects. The objects are first designed in a CAD program (Computer Aided Design) that also estimates how much material will be required to print the object.

The data file `"filament1.rda"` contains information about one 3D-printed object per row. The columns are

- `Index`: an observation index
- `Date`: printing dates
- `Material`: the printing material, identified by its colour
- `CAD_Weight`: the object weight (in grams) that the CAD software calculated
- `Actual_Weight`: the actual weight of the object (in grams) after printing

Start by loading the data and plotting it. Comment on the variability of the data for different `CAD_Weight` and `Material`.

2 Classical estimation

Consider two linear models, named A and B, for capturing the relationship between `CAD_Weight` and `Actual_Weight`. We denote the `CAD_weight` for observation i by x_i, and the corresponding `Actual_Weight` by y_i. The two models are defined by

- Model A: $y_i \sim \mathsf{Normal}[\beta_1 + \beta_2 x_i, \exp(\beta_3 + \beta_4 x_i)]$
- Model B: $y_i \sim \mathsf{Normal}[\beta_1 + \beta_2 x_i, \exp(\beta_3) + \exp(\beta_4) x_i^2)]$

The printer operator reasons that random fluctuations in the material properties (such as the density) and room temperature should lead to a *relative* error instead of an additive error, leading them to model B as an approximation of that. The basic physics assumption is that the error in the CAD software calculation of the weight is proportional to the weight itself. Model A on the other hand is slightly more mathematically convenient, but has no such motivation in physics.

Create a function `neg_log_like()` that takes arguments `beta` (model parameters), `data` (a data.frame containing the required variables), and `model` (either A or B) and returns the negated log-likelihood for the specified model.

Create a function `filament1_estimate()` that uses the R built in function `optim()` and `neg_log_like()` to estimate the two models A and B using the `filament1` data. As initial values for $(\beta_1, \beta_2, \beta_3, \beta_4)$ in the optimization use (-0.1, 1.07, -2, 0.05) for model A and (-0.15, 1.07, -13.5, -6.5) for model B. The inputs of the function should be: a `data.frame` with the same variables as the `filament1` data set (columns `CAD_Weight` and `Actual_Weight`) and the model choice (either A or B). As the output, your function should return the best set of parameters found and the estimate of the Hessian at the solution found.

First, use `filament1_estimate()` to estimate models A and B using the `filament1` data:

- fit_A = filament1_estimate(filament1, "A")
- fit_B = filament1_estimate(filament1, "B")

Use the approximation method for large n and the outputs from `filament1_estimate()` to construct an approximate 90% confidence intervals for $\beta_1, \beta_2, \beta_3$, and β_4 in Models A and B. Print the result as a table using the knitr::kable function. Compare the confidence intervals for the different parameters and their width. Comment on the differences to interpret the model estimation results.

3 Bayesian estimation

Now consider a Bayesian model for describing the actual weight (y_i) based on the CAD weight (x_i) for observation i:

$$y_i \sim \mathsf{Normal}[\beta_1 + \beta_2 x_i, \beta_3 + \beta_4 x_i^2)].$$

To ensure positivity of the variance, the parameterisation $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4] = [\beta_1, \beta_2, \log(\beta_3), \log(\beta_4)]$ is introduced, and the printer operator assigns independent prior distributions as follows:

$$\begin{aligned}
\theta_1 &\sim \mathsf{Normal}(0, \gamma_1), \\
\theta_2 &\sim \mathsf{Normal}(1, \gamma_2), \\
\theta_3 &\sim \mathsf{LogExp}(\gamma_3), \\
\theta_4 &\sim \mathsf{LogExp}(\gamma_4),
\end{aligned}$$

where $\mathsf{LogExp}(a)$ denotes the logarithm of an exponentially distributed random variable with rate parameter a, as seen in Tutorial 4. The $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$ values are positive parameters.

3.1 Prior density

With the help of `dnorm` and the `dlogexp` function (see the `code.R` file for documentation), define and document (in `code.R`) a function `log_prior_density` with arguments `theta` and `params`, where theta is the $\boldsymbol{\theta}$ parameter vector, and `params` is the vector of $\boldsymbol{\gamma}$ parameters. Your function should evaluate the logarithm of the joint prior density $p(\boldsymbol{\theta})$ for the four θ_i parameters.

3.2 Observation likelihood

With the help of `dnorm`, define and document a function `log_like`, taking arguments `theta`, `x`, and `y`, that evaluates the observation log-likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ for the model defined above.

3.3 Posterior density

Define and document a function `log_posterior_density` with arguments `theta`, `x`, `y`, and `params`, which evaluates the logarithm of the posterior density $p(\boldsymbol{\theta}|\boldsymbol{y})$, apart from some unevaluated normalisation constant.

3.4 Posterior mode

Define a function `posterior_mode` with arguments `theta_start`, `x`, `y`, and `params`, that uses `optim` together with the `log_posterior_density` and filament data to find the mode $\boldsymbol{\mu}$ of the log-posterior-density and evaluates the Hessian at the mode as well as the inverse of the negated Hessian, $\boldsymbol{S}$. This function should return a list with elements `mode` (the posterior mode location), `hessian` (the Hessian of the log-density at the mode), and `S` (the inverse of the negated Hessian at the mode). See the documentation for `optim` for how to do maximisation instead of minimisation.

3.5 Gaussian approximation

Let all $\gamma_i = 1$, $i = 1, 2, 3, 4$, and use `posterior_mode` to evaluate the inverse of the negated Hessian at the mode, in order to obtain a multivariate Normal approximation $\mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{S})$ to the posterior distribution for $\boldsymbol{\theta}$. Use start values $\boldsymbol{\theta} = \mathbf{0}$.

3.6 Importance sampling function

The aim is to construct a 90% Bayesian credible interval for each β_j using importance sampling, similarly to the method used in lab 4. There, a one dimensional Gaussian approximation of the posterior of a parameter was used. Here, we will instead use a multivariate Normal approximation as the importance sampling distribution. The functions `rmvnorm` and `dmvnorm` in the `mvtnorm` package can be used to sample and evaluate densities.

Define and document a function `do_importance` taking arguments `N` (the number of samples to generate), `mu` (the mean vector for the importance distribution), and `S` (the covariance matrix), and other additional parameters that are needed by the function code.

The function should output a `data.frame` with five columns, `beta1`, `beta2`, `beta3`, `beta4`, `log_weights`, containing the β_i samples and normalised log-importance-weights, so that `sum(exp(log_weights))` is 1. Use the `log_sum_exp` function (see the `code.R` file for documentation) to compute the needed normalisation information.

3.7 Importance sampling

Use your defined functions to compute an importance sample of size $N = 10000$. With the help of the `stat_ewcdf` function defined in `code.R`, plot the empirical weighted CDFs together with the unweighted CDFs for each parameter and discuss the results. To achieve a simpler `ggplot` code, you may find `pivot_longer(???, starts_with("beta"))` and `facet_wrap(vars(name))` useful.

Construct 90% credible intervals for each of the four model parameters based on the importance sample. In addition to `wquantile` and `pivot_longer`, the methods `group_by` and `summarise` are helpful. You may wish to define a function `make_CI` taking arguments `x`, `weights`, and `prob` (to control the intended coverage probability), generating a 1-row, 2-column `data.frame` to help structure the code.

Discuss the results both from the sampling method point of view and the 3D printer application point of view (this may also involve, e.g., plotting prediction intervals based on point estimates of the parameters, and plotting the importance log-weights to explain how they depend on the sampled β-values).