Exercise Sheet 03

Exercise 1: A location-scale regression model in Liesel

Continue using the `rent99` dataset from Sheet 2. Now we create our first full-fledged location-scale regression model. That means:

- We now predict *both* the location and the scale of the response by covariates.
- To take into account that the scale must be positive, we place the covariate model for the scale on the logarithm of the scale.

To make the model complete, we place priors on all regression coefficients in the model. The intercept terms receive constant priors.

a) Set up the following statistical model as a Liesel model:

$$
\begin{aligned}
\text{area}_i &\sim \mathcal{N}(\mu_i, \sigma_i^2) && \text{(observation model)} \\
\\
\mu_i &= \beta_0 + \beta_1 \text{area}_i && \text{(location model)} \\
p(\beta_0) &\propto \text{const.} && \text{(prior for } \beta_0\text{)} \\
\beta_1 &\sim \mathcal{N}(0, 10^2) && \text{(prior for } \beta_1\text{)} \\
\\
\log \sigma_i &= \gamma_0 + \gamma_1 \text{area}_i && \text{(log-scale model)} \\
p(\gamma_0) &\propto \text{const.} && \text{(prior for } \gamma_0\text{)} \\
\gamma_1 &\sim \mathcal{N}(0, 10^2) && \text{(prior for } \gamma_1\text{)}
\end{aligned}
$$

b) Plot the model graph.
c) Use goose to sample from the posterior.

Exercise 2: A semiparametric model with `{rliesel}`

Now, we use the `mcycle` dataset as an example.[1] You can download the data from `https://s.gwdg.de/50F2v6`.[2]

The dataset contains a series of measurements of head acceleration in a simulated motorcycle accident, used to test crash helmets. The contained variables are

times in milliseconds after impact.
accel acceleration in g.

[1] See the `{MASS}` R package.
[2] Points to `https://raw.githubusercontent.com/liesel-devs/cmstats-tutorial-public/main/data/mcycle.csv`

We start by considering a model that assumes the acceleration to be normally distributed, with the mean being some nonlinear smooth function of the time:

$$\begin{aligned}\text{accel}_i &\sim \mathcal{N}(\mu_i, \sigma^2)\\ \mu_i &= \beta_0 + s(\text{times}_i).\end{aligned}$$

a) Create a scatter plot of `times` and `accel` to familiarize yourself with the dataset.
b) Use `rliesel` to set up your model as a Liesel model in R. You can refer to the `rliesel` documentation. A solution for Python-only participants to obtain a model object to work with is included at the end of the sheet.
c) Plot your model in Python using `lsl.plot_vars`.
d) Describe the default sampling scheme for a semi-parametric distributional regression model: how is each parameter sampled?
e) Use `lsl.dist_reg_mcmc` to set up an `gs.EngineBuilder`, define the number of warmup and posterior samples, and run the sampling process using `gs.Engine.sample_all_epochs`.
f) Inspect your sampling results using `gs.Summary` and create trace plots.
g) Plot your posterior estimate of $s(\text{times}_i)$.

Exercise 3: A semiparametric distributional regression model with `{rliesel}`

We now expand our model from Exercise 1 to a full semiparametric distributional regression model. That is, we now assume the following model:

$$\begin{aligned}\text{accel}_i &\sim \mathcal{N}(\mu_i, \sigma_i^2)\\ \mu_i &= \beta_0 + s_\mu(\text{times}_i)\\ \sigma_i^2 &= \exp\big(\gamma_0 + s_\sigma(\text{times}_i)\big)^2.\end{aligned}$$

a) Use `rliesel` to set up your model as a Liesel model in R. A solution for Python-only participants to obtain a model object to work with is included at the end of the sheet.
b) Plot your model in Python using `lsl.plot_vars`.
c) Use `lsl.dist_reg_mcmc` to set up an `gs.EngineBuilder`, define the number of warmup and posterior samples, and run the sampling process using `gs.Engine.sample_all_epochs`.
d) Inspect your sampling results using `gs.Summary` and create trace plots.
e) Plot your posterior estimate of $s_\mu(\text{times}_i)$.
f) Plot your posterior estimate of $s_\sigma(\text{times}_i)$.

Exercise 4: Recover the statistical model from the graph (Bonus exercise)

Use the model from Exercise 2 for this exercise.

Write down the statistical model for the scale σ that `rliesel` automatically set up for us. You can access the model's variables through the `lsl.Model.vars` attribute. Use the model graph combined with inspection of the model variables in your search for information.

Model objects for Python-only participants

We provide saved model objects for both exercises for you to download. To load the model objects, you can use the following code:

````
```{python}
from urllib.request import urlopen
import dill

url = "paste url here"
model = dill.load(urlopen(url))
```
````

The links are:

- Exercise 2: `"https://s.gwdg.de/un4W29"`
- Exercise 3: `"https://s.gwdg.de/exn3LQ"`