Exercise Sheet 02

For this sheet, we use the `rent99` dataset as an example.[1] We use a standardized subset of the variables for our examples. You can download the data from `https://s.gwdg.de/mzAkHV`.[2] The dataset contains the following variables:

rent the monthly net rent in Euro per month (z-standardized).
area living area in square meters (z-standardized).

Exercise 1: Statistical models as directed acyclic graphs

Draw the following statistical models as directed acyclic graphs by hand:

a) The response variable rent_i is normally distributed with mean μ and variance σ^2. The model for an observation with index $i = 1, \dots, n$ is

$$\text{rent}_i \sim \mathcal{N}(\mu, \sigma^2).$$

b) Extend the model from a) with a linear regression model for the mean, using the living area as a predictor i.e.

$$\begin{aligned}\text{rent}_i &\sim \mathcal{N}(\mu_i, \sigma^2)\\ \mu_i &= \beta_0 + \beta_1 \text{area}_i,\end{aligned}$$

where β_0 is an intercept and β_1 is the linear regression coefficient for the living area.

c) Extend the model from b) by adding an inverse gamma prior for σ^2 with shape (=concentration) $a = 0.01$ and scale $b = 0.01$, i.e.

$$\begin{aligned}\text{rent}_i &\sim \mathcal{N}(\mu_i, \sigma^2)\\ \mu_i &= \beta_0 + \beta_1 \text{area}_i\\ \sigma^2 &\sim \mathcal{IG}(0.01, 0.01).\end{aligned}$$

Exercise 2: Your first Liesel model

Implement the statistical models from 1) as Liesel-models.

a) Implement the model from Exercise 1a) using `lsl.obs`, `lsl.param`, `lsl.Calc` `lsl.Dist` and `tfd.Normal`.

b) Add your response node to a `lsl.GraphBuilder`, build your model and plot your graph using the `lsl.plot_vars()` method.

c) Implement the model from Exercise 1b). Create and fill a `lsl.GraphBuilder` and plot this model, too.

d) Implement the model from Exercise 1c). You can use `lsl.Var` for the hyperparameters shape (=concentration) $a = 0.01$ and scale $b = 0.01$. Look up the distribution class for the prior of σ^2 from the TensorFlow-Probability documentation.[3] Create

[1] See the `{gamlss.data}` R package and Fahrmeir, Ludwig and Kneib, Thomas and Lang, Stefan and Marx, Brian (2013) *Regression: models, methods and applications*, Springer.
[2] Points to `https://raw.githubusercontent.com/liesel-devs/cmstats-tutorial-public/main/data/rent99.csv`
[3] https://www.tensorflow.org/probability/api_docs/python/tfp

and fill a `lsl.GraphBuilder` and plot this model, too.

Exercise 3: Manipulate a model graph

a) Take your existing model from Exercise 2d). Without creating a new model, change the inverse-gamma prior $\mathcal{IG}(0.01, 0.01)$ for the variance σ^2 to $\mathcal{IG}(2, 0.5)$. Print the model's log posterior before and after the change to see its effect.
b) Now replace the inverse-gamma prior for the **variance** σ^2 in your model from Exercise 2d) with a half-Cauchy prior for the **scale** σ. The half-Cauchy prior should have a location of zero and a scale of 25. Plot the resulting graph.

$$\underbrace{\sigma^2 \sim \mathcal{IG}(a, b)}_{\text{old}} \quad \to \quad \underbrace{\sigma \sim \text{HalfCauchy}(0, 25)}_{\text{new}}$$

c) Include the natural logarithm of the scale, $\log(\sigma)$, in your Liesel model. To do this, take your `lsl.GraphBuilder` object and use the `lsl.GraphBuilder.transform()`[4] method. Plot the resulting graph.

Exercise 4: Sample from the posterior using Goose

a) Use the `gs.LieselInterface` class to connect your Liesel model from Exercise 3 to an `gs.EngineBuilder` and set its initial values. Define a seed and use four chains.
b) Add a `gs.IWLSKernel` for sampling β_0 and β_1 to your `EngineBuilder`.
c) Add a `gs.NUTSKernel` for sampling $\log(\sigma)$ `EngineBuilder`[5]
d) Set the warmup and posterior duration on the `EngineBuilder`.
e) Build your engine and sample from your model.
f) Use `gs.Summary` to print a summary table of your results.
g) Use `gs.plot_trace` to investigate your chains.

[4] https://docs.liesel-project.org/en/latest/generated/liesel.model.model.GraphBuilder.transform.html#liesel.model.model.GraphBuilder.transform
[5] We use different kernels here for illustration.

Exercise 5: A semiparametric model in Liesel

Now consider the following model, which assumes that μ_i is a smooth function of area$_i$:

$$\begin{aligned} \text{rent}_i &\sim \mathcal{N}(\mu_i, \sigma^2) \\ \mu_i &= s(\text{area}_i) \\ \sigma^2 &\sim \mathcal{IG}(a, b). \end{aligned}$$

a) Formulate a Bayesian P-spline model for the smooth function $s(\text{area}_i)$.
b) Draw a directed acyclic graph to represent the model.
c) Implement your model in Liesel, using 20 basis functions. Note the tips below for setting up your basis matrix and penalty matrix. Also note that you can use the `MultivariateNormalDegenerate` distribution class offered by Liesel (link).
d) Set up an `gs.EngineBuilder` and sample from the posterior. You can assume the inverse smoothing parameter τ^2 and the variance σ^2 to be fixed or sample them with a logarithmic transformation.
e) Use `gs.plot_trace` to investigate your chains.
f) Plot the posterior mean estimate of your smooth function.

To set up the matrix of basis function evaluations and the penalty matrix for your model, you can use functionality offered by the R package `mgcv`:

~~~
```{r}
library(mgcv)
smooth <- smooth.construct(s(area, k=20, bs="ps"), data = rent99, knots = NULL)
basis_matrix <- smooth$X
penalty_matrix <- smooth$S[[1]]
```
~~~

If you are following along in pure Python, you can find a basis matrix in the public repository: https://s.gwdg.de/LZnQMC. You can import this basis matrix with numpy:

~~~
```{python}
import numpy as np
basis_matrix = np.loadtxt("https://s.gwdg.de/LZnQMC")
```
~~~

You can generate the penalty matrix in Python with the following code:

~~~
```{python}
import jax.numpy as jnp

D = jnp.diff(jnp.eye(20), n=2, axis=0)
K = D.T @ D
```
~~~