PRACTICAL 1 - LINEAR (MIXED) MODELS

In this practical we are going to fit linear (mixed) models in `inlabru`. We are going to to:

- Fit a simple linear regression
- Fit a linear regression with discrete covariates and interactions
- Fit a linear mixed model.

In all cases we are first going to simulate our data. We then fit the data and explore the results.

> **Note**
>
> You can download the R-script of this practical by clicking the button below:

Start by loading useful libraries:

```
library(dplyr)
library(INLA)
library(ggplot2)
library(patchwork)
library(inlabru)
# load some libraries to generate nice plots
library(scico)
```

1 Simple linear regression

We consider a simple linear regression model with Gaussian observations

$$y_i \sim \mathcal{N}(\mu_i, \sigma^2), \qquad i = 1, \dots, N$$

where σ^2 is the observation error, and the mean parameter μ_i is linked to the **linear predictor** (η_i) through an identity function:

$$\eta_i = \mu_i = \beta_0 + \beta_1 x_i.$$

Here $\mathbf{x} = (x_1, \dots, x_N)$ is a continuous covariate and β_0, β_1 are parameters to be estimated.

To finalize the Bayesian model we assign prior distribution as $\tau = 1/\sigma^2 \sim$ Gamma(a, b) and $\beta_0, \beta_1 \sim \mathcal{N}(0, 1/\tau_\beta)$ (we will use the default prior settings in INLA for now).

> **Question**
>
> What is the dimension of the hyperparameter vector and latent Gaussian field?
> Answer
> The hyperparameter vector has dimension 1, $\boldsymbol{\theta} = (\tau)$ while the latent Gaussian field $\boldsymbol{u} = (\beta_0, \beta_1)$ has dimension 2, 0 mean, and sparse precision matrix:

$$\boldsymbol{Q} = \begin{bmatrix} \tau_{\beta_0} & 0 \\ 0 & \tau_{\beta_1} \end{bmatrix}$$

Note that, since β_0 and β_1 are fixed effects, the precision parameters τ_{β_0} and τ_{β_1} are fixed.

Note

We can write the linear predictor vector $\boldsymbol{\eta} = (\eta_1, \dots, \eta_N)$ as

$$\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}_1\boldsymbol{u}_1 + \boldsymbol{A}_2\boldsymbol{u}_2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \beta_1$$

Our linear predictor consists then of two components: an intercept and a slope.

1 Simulate example data

We fix the model parameters β_0, β_1 and the hyperparameter τ_y to a given value and simulate the data accordingly using the code below. The simulated response and covariate data are then saved in a `data.frame` object.

```
# set seed for reproducibility
set.seed(1234)

# Fix the model parameters
beta = c(2,0.5)
sd_error = 0.1

# simulate the data
n = 100
x = rnorm(n)
y = beta[1] + beta[2] * x + rnorm(n, sd = sd_error)

# create the data frame object
df = data.frame(y = y, x = x)
```

1 Fitting a linear regression model with `inlabru`

Step1: Defining model components

The first step is to define the two model components: The intercept and the linear covariate effect.

Task

Define an object called `cmp` that includes and (i) intercept `beta_0` and (ii) a covariate `x` linear effect `beta_1`.
Take hint

The `cmp` object is here used to define model components. We can give them any useful names we like, in this case, `beta_0` and `beta_1`. You can remove the automatic intercept construction by adding a `-1` in the components
Click here to see the solution

```
cmp =  ~ -1 + beta_0(1) + beta_1(x, model = "linear")
```

Note

Note that we have excluded the default Intercept term in the model by typing `-1` in the model components. However, `inlabru` has automatic intercept that can be called by typing `Intercept()` , which is one of `inlabru` special names and it is used to define a global intercept, e.g.

```
cmp =  ~  Intercept(1) + beta_1(x, model = "linear")
```

Step 2: Build the observation model

The next step is to construct the observation model by defining the model likelihood. The most important inputs here are the `formula`, the `family` and the `data`.

Task

Define a linear predictor `eta` using the component labels you have defined on the previous task.
Take hint
The `eta` object defines how the components should be combined in order to define the model predictor.
Click here to see the solution

```
eta = y ~ beta_0 + beta_1
```

The likelihood for the observational model is defined using the `bru_obs()` function.

Task

Define the observational model likelihood in an object called `lik` using the `bru_obs()` function.
Take hint
The `bru_obs` is expecting three arguments:

- The linear predictor `eta` we defined in the previous task
- The data likelihood (this can be specified by setting `family = "gaussian"`)
- The data set `df`

Click here to see the solution

```
lik =  bru_obs(formula = eta,
               family = "gaussian",
               data = df)
```

Step 3: Fit the model

We fit the model using the `bru()` functions which takes as input the components and the observation model:

```
fit.lm = bru(cmp, lik)
```

Step 5: Extract results

There are several ways to extract and examine the results of a fitted `inlabru` object.

The most natural place to start is to use the `summary()` which gives access to some basic information about model fit and estimates

```
summary(fit.lm)
## inlabru version: 2.13.0.9016
## INLA version: 25.11.22
## Components:
## Latent components:
## beta_0: main = linear(1)
## beta_1: main = linear(x)
## Observation models:
##   Family: 'gaussian'
##     Tag: <No tag>
##     Data class: 'data.frame'
##     Response class: 'numeric'
##     Predictor: y ~ beta_0 + beta_1
##     Additive/Linear: TRUE/TRUE
##     Used components: effects[beta_0, beta_1], latent[]
## Time used:
##     Pre = 0.951, Running = 0.181, Post = 0.0162, Total = 1.15
## Fixed effects:
##         mean   sd 0.025quant 0.5quant 0.975quant  mode kld
## beta_0 2.004 0.01      1.983    2.004      2.024 2.004   0
## beta_1 0.497 0.01      0.477    0.497      0.518 0.497   0
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 94.85 13.41      70.43    94.22
##                                         0.975quant  mode
## Precision for the Gaussian observations     122.92 92.96
##
## Marginal log-Likelihood:  63.27
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

We can see that both the intercept and slope and the error precision are correctly estimated.

Another way, which gives access to more complicated (and useful) output is to use the `predict()` function.

Below we take the fitted `bru` object and use the `predict()` function to produce predictions for μ given a new set of values for the model covariates or the original values used for the model fit

```
new_data = data.frame(x = c(df$x, runif(10)),
                      y = c(df$y, rep(NA,10)))
pred = predict(fit.lm, new_data, ~ beta_0 + beta_1,
               n.samples = 1000)
```

The `predict()` function generate samples from the fitted model. In this case we set the number of samples to 1000.

2 Plot

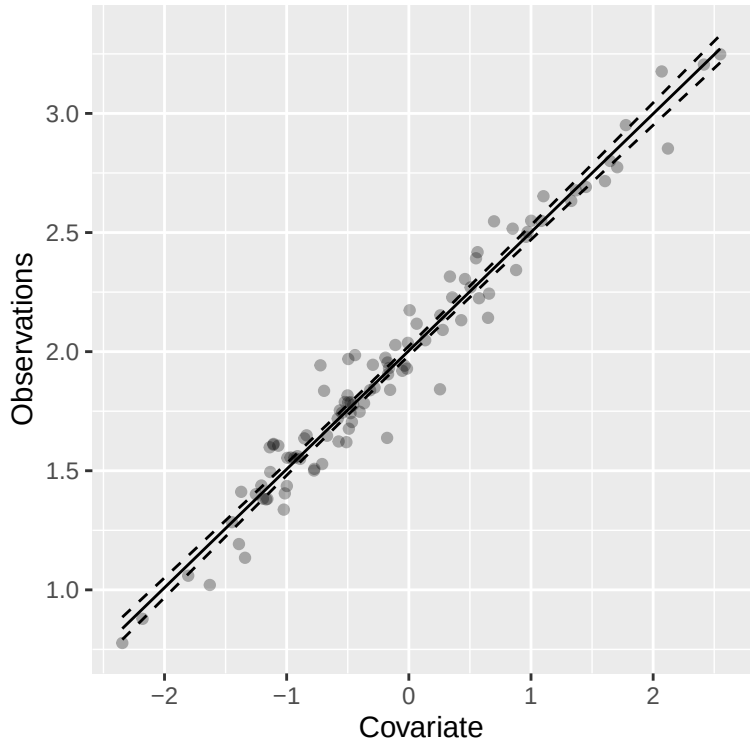

Figure 1: Data and 95% credible intervals

3 R Code

```
pred %>% ggplot() +
  geom_point(aes(x,y), alpha = 0.3) +
  geom_line(aes(x,mean)) +
  geom_line(aes(x, q0.025), linetype = "dashed")+
  geom_line(aes(x, q0.975), linetype = "dashed")+
  xlab("Covariate") + ylab("Observations")
```

Task

Generate predictions for the linear predictor μ when the covariate has value $x_0 = 0.45$.
What is the predicted value for μ? And what is the uncertainty?
Take hint
You can create a new data frame containing the new observation x_0 and then use the

`predict` function.
Click here to see the solution

```
new_data = data.frame(x = 0.45)
pred = predict(fit.lm, new_data, ~ beta_0 + beta_1,
               n.samples = 1000)

pred
```

```
     x    mean         sd   q0.025     q0.5   q0.975   median sd.mc_std_err
1 0.45 2.22759 0.01158763 2.206017 2.227908 2.249129 2.227908  0.0002438256
  mean.mc_std_err
1    0.0003818539
```

You can see the predicted mean and sd by examining the produced `pred` object. In this case the mean is ca 2.22 with sd ca 0.01. This gives a 95% CI ca [2.20, 2.25].

NOTE Now we have produced a credible interval for the expected mean μ if we want to produce a *prediction* interval for a new observation y we need to add the uncertainty that comes from the likelihood with precision τ_y. To do this we can again use the `predict()` function to compute a 95% prediction interval for y.

```
pred2 = predict(fit.lm, new_data,
                formula = ~ {
                  mu = beta_0 + beta_1
                  sigma = sqrt(1/Precision_for_the_Gaussian_observations)
                  list(q1 = qnorm(0.025, mean = mu, sd = sigma),
                       q2 =  qnorm(0.975, mean = mu, sd = sigma))},
                n.samples = 1000)
round(c(pred2$q1$mean, pred2$q2$mean),2)
```

```
[1] 2.03 2.43
```

Notice that now the interval we obtain is larger.

4 Linear model with interactions