PRACTICAL

Start by loading usefull libraries:

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

Aim of this practical: In this first practical we are going to look at some simple models

1. A Gaussian model with simulated data
2. A GLM model with random effects

we are going to learn:

- How to fit a simple model with `inlabru`
- How to explore the results
- How to change the prior distributions
- How to get predictions for missing data points

0 Linear Model

As our first example we consider a simple linear regression model with Gaussian observations $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $i = 1, \dots, N$ where σ^2 is the observation error, and the mean parameter μ_i is linked to the linear predictor through an identity function:

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

where x_i is a covariate and β_0, β_1 are parameters to be estimated.

To finalize the Bayesian model we need to assign a Gamma(a, b) prior to the precision parameter $\tau = 1/\sigma^2$ and two independent Gaussian priors with mean 0 and precision τ_β to the regression parameters β_0 and β_1.

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:

$$\boldsymbol{Q} = \tau_\beta \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Note

We can write the linear predictor vector $\boldsymbol{\eta} = (\eta_i, \ldots, \eta_N)$ 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.

0.1.1 Simulate example data

In this practical we will use simulated Gaussian data to get familiar with the `inlabru` workflow. Moreover, we will see how to change the prior distributions both for the fixed effects β_0 and β_1 and for the hyperparameter $\tau = 1/\sigma^2$. First, we simulate data from the model

$$y_i \sim \mathcal{N}(\eta_i, 0.1^2),\ i = 1, \ldots, 100$$

with

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

where $\beta_0 = 2$,$\beta_1 = 0.5$ and the values of the covariate x are generated from an Uniform(0,1) distribution. The simulated response and covariate data are then saved in a `data.frame` object.

```
beta = c(1,1)
sd_error = 1

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

df = data.frame(y = y, x = x)
```

0.1.2 Fitting a linear regression model with `inlabru`

Defining model components

The model has two parameters to be estimated β_1 and β_2. We need to define the two corresponding model components:

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

The cmp object is here used to define model components. We can give them any useful names we like

Note

Note that Intercept() is one of inlabru special names and it is used to define a global intercept. You should explicitly exclude automatic intercept when not using the special Intercept name, e.g.

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

Observation model construction

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.

The formula defines how the components should be combined in order to define the model predictor.

```
formula = y ~ Intercept + beta_1
```

Note

In this case we can also use the shortcut formula = y ~ .. This will tell inlarbu that the model is linear and that it is not necessary to linearize the model and assess convergence.

The likelihood is defined using the bru_obs() function as follows:

```
lik =  bru_obs(formula = y ~.,
            family = "gaussian",
            data = df)
```

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)
```

The summary() function will give access to some basic information about model fit and estimates

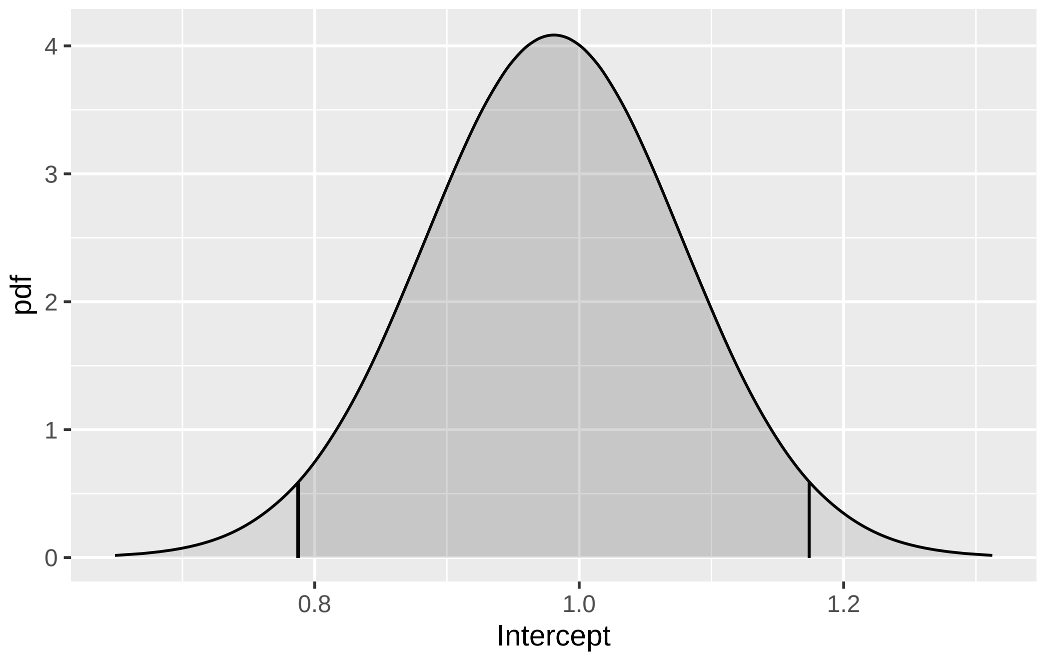
```
summary(fit.lm)
```

```
inlabru version: 2.12.0
INLA version: 25.02.10
Components:
Intercept: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
beta_1: main = linear(x), group = exchangeable(1L), replicate = iid(1L), NULL
Likelihoods:
  Family: 'gaussian'
    Tag: ''
    Data class: 'data.frame'
```

```
    Response class: 'numeric'
    Predictor: y ~ .
    Used components: effects[Intercept, beta_1], latent[]
Time used:
    Pre = 0.383, Running = 0.182, Post = 0.0761, Total = 0.642
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant  mode kld
Intercept 0.981 0.099      0.787    0.981      1.175 0.981   0
beta_1    1.120 0.103      0.917    1.120      1.322 1.120   0

Model hyperparameters:
                                        mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 1.04 0.148      0.776     1.04
                                        0.975quant mode
Precision for the Gaussian observations       1.35 1.02

Deviance Information Criterion (DIC) ...............: 285.35
Deviance Information Criterion (DIC, saturated) ....: 105.43
Effective number of parameters .....................: 3.00

Watanabe-Akaike information criterion (WAIC) ...: 285.24
Effective number of parameters .................: 2.79

Marginal log-Likelihood:  -162.09
 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. We can then plot the marginal posterior for β_0 as follows:

```
plot(fit.lm, "Intercept")
```



Task

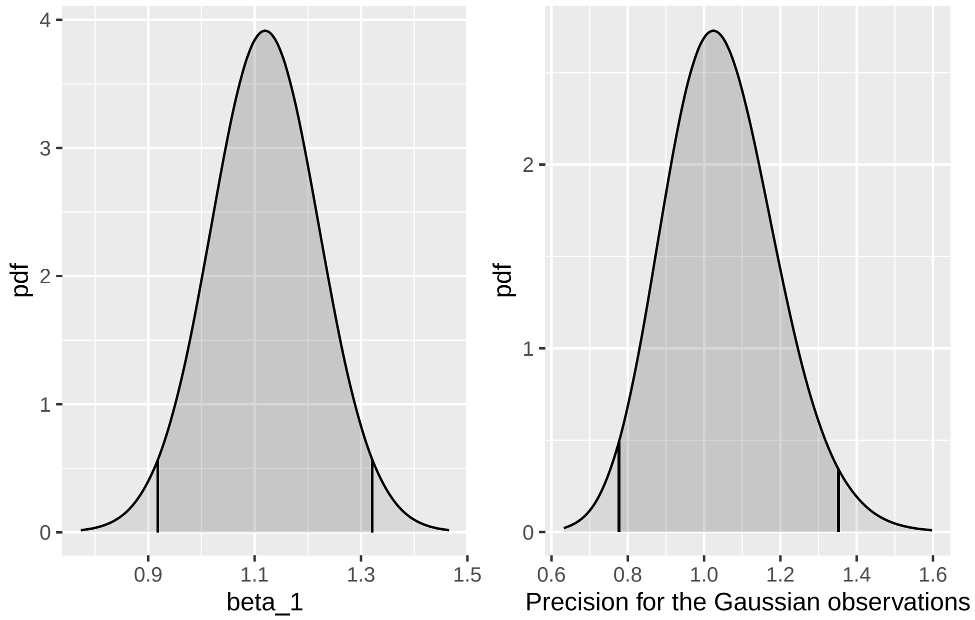
Plot the posterior marginals for β_1 and for the precision of the observation error $\pi(\tau|y)$

Take hint

See the `summary()` output to check the names for the different model components.

Click here to see the solution

```
plot(fit.lm, "beta_1") +
plot(fit.lm, "Precision for the Gaussian observations")
```



Task

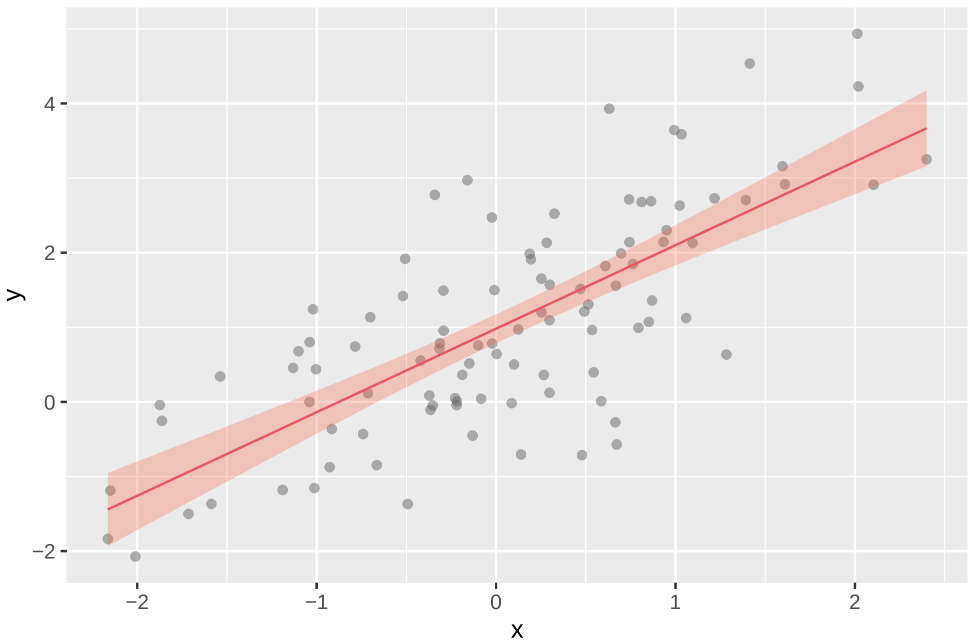
Plot the fitted values with 95% Credible intervals.

Take hint

`bru` objects information about the linear predictor can be accessed through `fit.lm$summary.fitted.values`.

Click here to see the solution

```
df %>% mutate(post_mean = fit.lm$summary.fitted.values[1:100,"mean"],
              q25 = fit.lm$summary.fitted.values[1:100,"0.025quant"],
              q975 = fit.lm$summary.fitted.values[1:100,"0.975quant"])%>%
  ggplot()+geom_point(aes(x=x,y=y),alpha=0.5,color="grey40")+
  geom_line(aes(x=x,y=post_mean),col=2)+
  geom_ribbon(aes(x = x, ymax = q975, ymin = q25),fill="tomato", alpha = 0.3)
```



0.1.3 Generate model predictions

Now we can take the fitted `bru` object and use the `predict` function to produce predictions 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, ~ Intercept + beta_1)
```

0 Plot

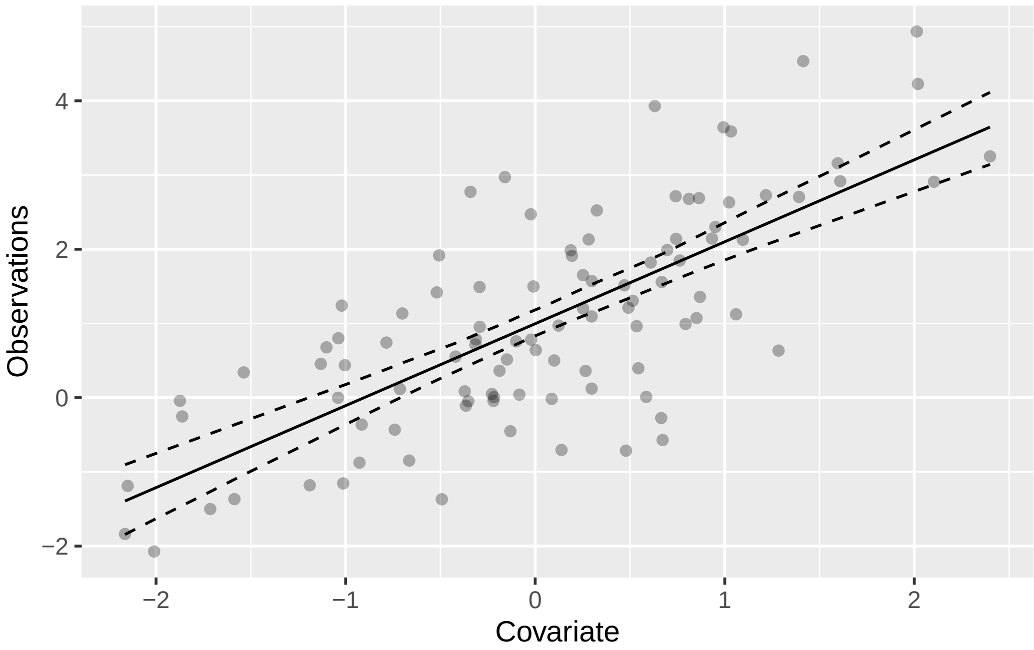


Figure 1: Data and 95% credible intervals

0 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")
```