PRACTICAL 1

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

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)
```

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 $\boldsymbol{\eta} = (\eta_i, \dots, \eta_N)$ 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.

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

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

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

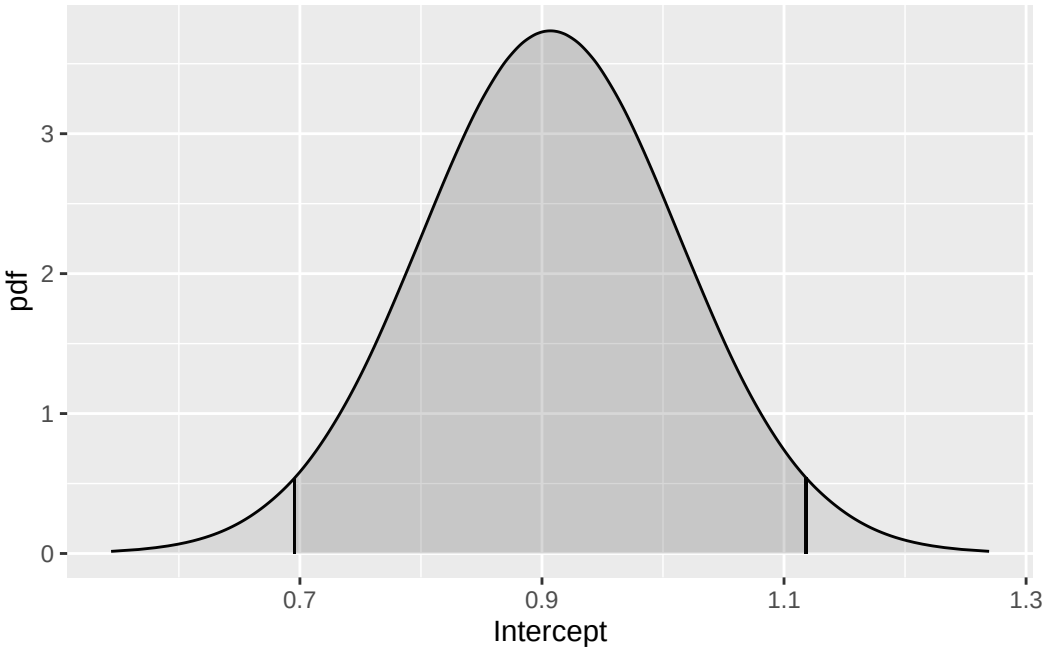
```
summary(fit.lm)
```

```
inlabru version: 2.12.0
INLA version: 24.06.27
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 = 0.569, Running = 0.323, Post = 0.119, Total = 1.01
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant  mode kld
Intercept 0.907 0.108      0.695    0.907      1.119 0.907   0
beta_1    0.944 0.129      0.691    0.944      1.197 0.944   0

Model hyperparameters:
                                          mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 0.888 0.126      0.659    0.882
                                        0.975quant mode
Precision for the Gaussian observations       1.15 0.87

Deviance Information Criterion (DIC) ...............: 301.69
Deviance Information Criterion (DIC, saturated) ....: 105.39
Effective number of parameters .....................: 3.00

Watanabe-Akaike information criterion (WAIC) ...: 301.94
Effective number of parameters .................: 3.11

Marginal log-Likelihood:  -170.12
 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

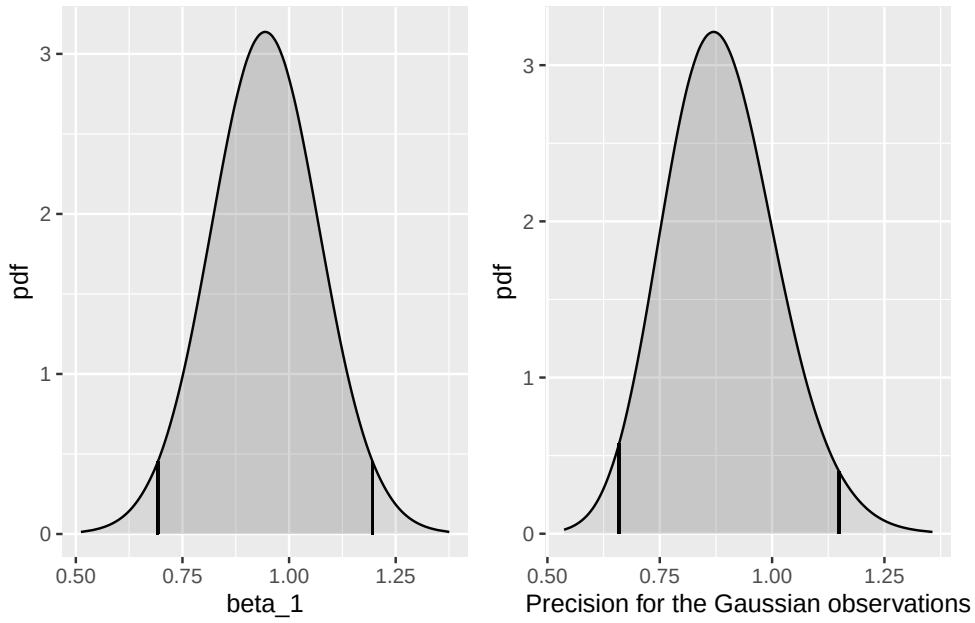
Plot the posterior marginals for β_1 and for the precision of the observation error $\pi(\tau|y)$

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

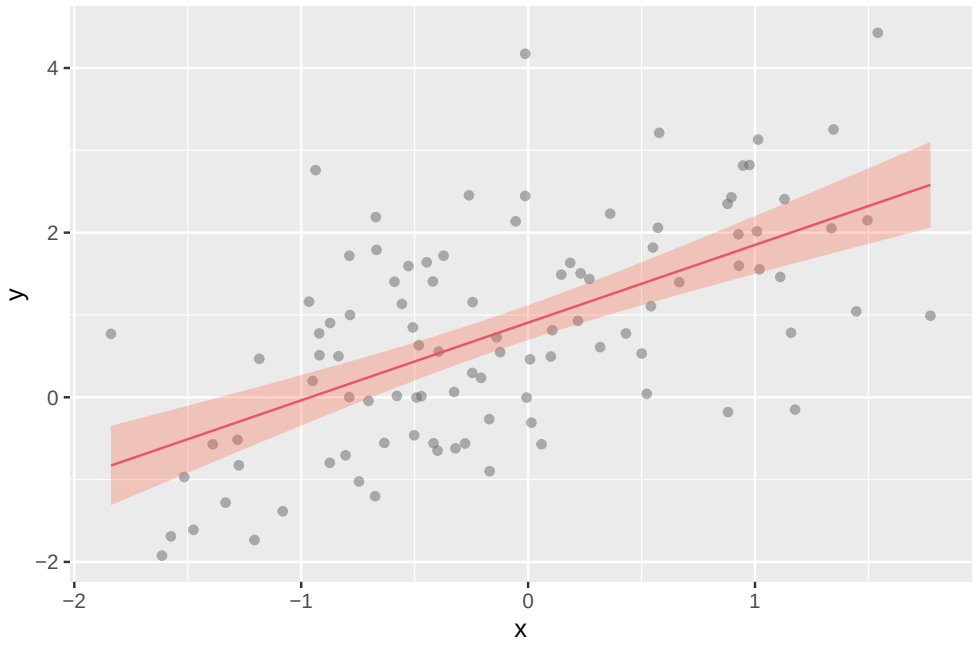
Plot the fitted values with 95% Credible intervals.

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

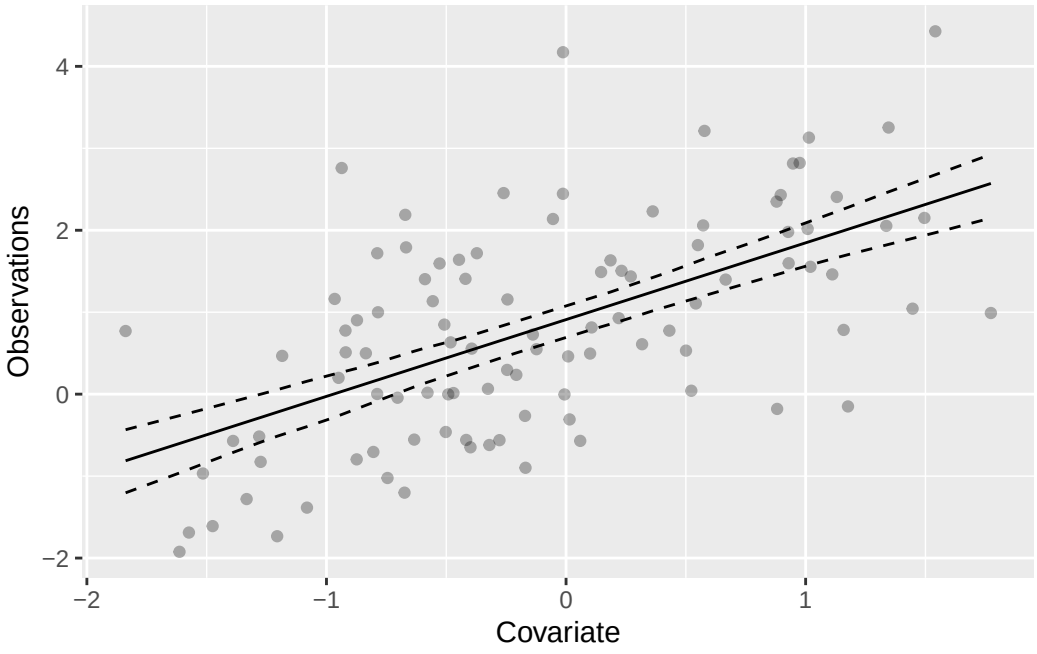


Figure 1: Data and 95% credible intervals

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")
```

1 Linear Mixed Model

Consider the same linear model as in Section 0.1 except with the addition that the data that comes in groups. For each group j, we have an associated variable $v_j \sim \mathcal{N}(0, \tau_v^{-1})$.

The predictor for this model is

$$\mathbb{E}\left[y_{ij}\right] = \beta_0 + \beta_1 x_i + v_j$$

The model design matrix for the random effect has one row for each observation. The row of the design matrix associated with the ij-th observation consists of zeroes except for the element associated with v_j, which has a one.

1.0.0.0.1 Simulate example data

```
library(inlabru)
library(INLA)
library(ggplot2)
```

```
library(dplyr)
```

```
beta = c(1.5,1)
sd_error = 1
tau_group = 1

n = 100
n.groups = 5
x = rnorm(n)
v = rnorm(n.groups, sd = tau_group^{-1/2})
y = beta[1] + beta[2] * x + rnorm(n, sd = sd_error) +
  rep(v, each = 20)

df = data.frame(y = y, x = x, j = rep(1:5, each = 20))
```

Note that `inlabru` expects an integer indexing variable to label the groups.

```
df$jfac = as.factor(df$j)
ggplot(df) +
  geom_point(aes(x = x, colour = jfac, y = y)) +
  theme_classic() +
  scale_colour_discrete("Group")
```

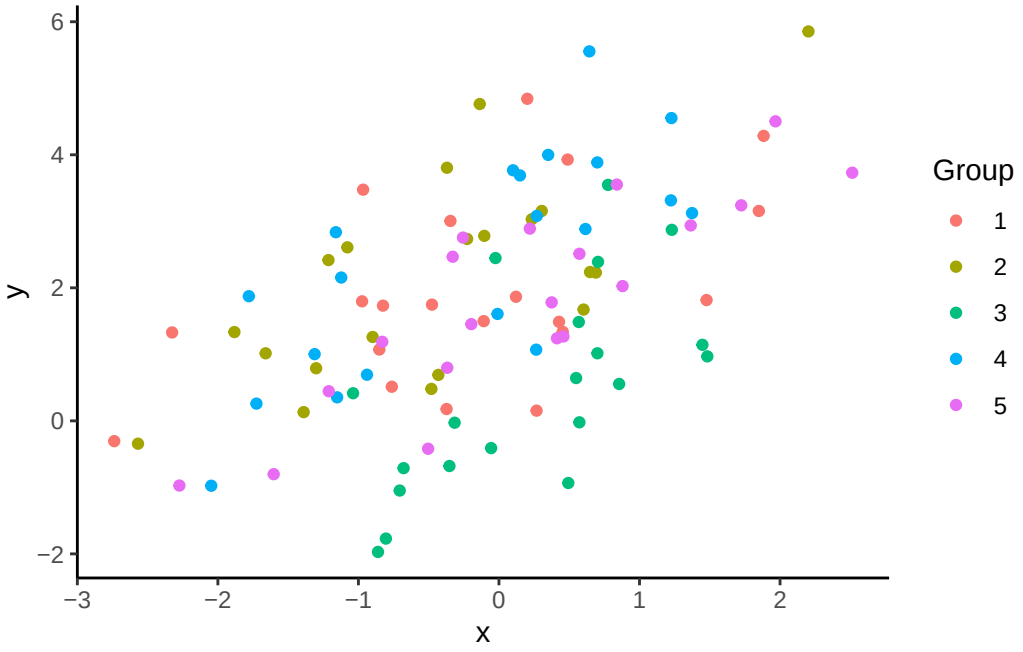


Figure 2: Data for the linear mixed model example with 5 groups

1.0.0.1 Define model components and likelihood In order to specify this model we must use the `group` argument to tell `inlabru` which variable indexes the groups. The `model = "iid"` tells INLA that the groups are independent from one another.

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

```
    v(j, model = "iid")
```

The group variable is indexed by column `j` in the dataset. We have chosen to name this component `v()` to connect with the mathematical notation that we used above.

```
# Construct likelihood
lik =  like(formula = y ~.,
            family = "gaussian",
            data = df)
```

1.0.0.1.1 Fit the model The model can be fitted exactly as in the previous examples by using the `bru` function with the components and likelihood objects.

```
fit = bru(cmp, lik)
summary(fit)
```

```
inlabru version: 2.12.0
INLA version: 24.06.27
Components:
Intercept: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
beta_1: main = linear(x), group = exchangeable(1L), replicate = iid(1L), NULL
v: main = iid(j), 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, v], latent[]
Time used:
    Pre = 0.487, Running = 0.408, Post = 0.282, Total = 1.18
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant  mode kld
Intercept 1.852 0.397      1.055    1.851      2.649 1.851   0
beta_1    0.968 0.106      0.758    0.968      1.176 0.968   0

Random effects:
  Name    Model
    v IID model

Model hyperparameters:
                                          mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 0.818 0.119      0.607    0.811
Precision for v                         2.088 1.447      0.452    1.724
                                        0.975quant  mode
Precision for the Gaussian observations       1.07 0.798
Precision for v                               5.86 1.129

Deviance Information Criterion (DIC) ...............: 314.06
Deviance Information Criterion (DIC, saturated) ....: 108.97
Effective number of parameters .....................: 6.69
```

```
Watanabe-Akaike information criterion (WAIC) ...: 313.91
Effective number of parameters .................: 6.17

Marginal log-Likelihood:  -188.93
 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)')
```