PRACTICAL 2

Aim of this practical:

1. Set priors for different linear models
2. Compute and visualize posterior densities and summaries for marginal effects
3. Fit hierarchical flexible models

we are going to learn:

- How to change some of the R default priors in `inlabru`
- How to explore and visualize model parameters
- Fit different flexible models

1 Setting priors and model checking for Linear Models

In this exercise we will:

- Learn how to set priors for linear effects β_0 and β_1
- Learn how to set the priors for the hyperparameter $\tau = 1/\sigma^2$.
- Visualize marginal posterior distributions

Start by loading useful libraries:

```
library(dplyr)
library(INLA)
library(ggplot2)
library(patchwork)
library(inlabru)
```

Recall a simple linear regression model with Gaussian observations

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

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. In INLA, we assume that the model is a latent Gaussian model, i.e., we have to assign β_0 and β_1 a Gaussian prior. For the precision hyperparameter $\tau = 1/\sigma^2$ a typical prior choice is a Gamma(a, b) prior.

In `R-INLA`, the default choice of priors for each β is

$$\beta \sim \mathcal{N}(0, 10^3).$$

and the prior for the variance parameter in terms of the log precision is

$$\log(\tau) \sim \text{logGamma}(1, 5 \times 10^{-5})$$

> **Note**
>
> If your model uses the default intercept construction (i.e., `Intercept(1)` in the linear predictor) `inlabru` will assign a default $\mathcal{N}(0, 0)$ prior to it.

Lets see how can we change the default priors using some simulated data

1.0.1 Simulate example data

We simulate data from a simple linear regression model

```
beta = c(2,0.5)
sd_error = 0.1

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

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

1.0.2 Fitting the linear regression model with `inlabru`

Now we fit a simple linear regression model in `inlabru` by defining (1) the model components, (2) the linear predictor and (3) the likelihood.

```
# Model components
cmp =  ~ -1 + beta_0(1) + beta_1(x, model = "linear")
# Linear predictor
formula = y ~ Intercept + beta_1
# Observational model likelihood
lik =  bru_obs(formula = y ~.,
            family = "gaussian",
            data = df)
# Fit the Model
fit.lm = bru(cmp, lik)
```

1 Change the prior distributions

Until now, we have used the default priors for both the precision τ and the fixed effects β_0 and β_1. Let's see how to customize these.

To check which priors are used in a fitted model one can use the function `inla.prior.used()`

```
inla.priors.used(fit.lm)
```

```
section=[family]
    tag=[INLA.Data1] component=[gaussian]
        theta1:
            parameter=[log precision]
            prior=[loggamma]
```

```
            param=[1e+00, 5e-05]
section=[linear]
    tag=[beta_0] component=[beta_0]
        beta:
            parameter=[beta_0]
            prior=[normal]
            param=[0.000, 0.001]
    tag=[beta_1] component=[beta_1]
        beta:
            parameter=[beta_1]
            prior=[normal]
            param=[0.000, 0.001]
```

From the output we see that the precision for the observation $\tau \sim$ Gamma$(1e+00, 5e-05)$ while β_0 and β_1 have precision 0.001, that is variance $1/0.001$.

Change the precision for the linear effects

The precision for linear effects is set in the component definition. For example, if we want to increase the precision to 0.01 for β_0 we define the relative components as:

```
cmp1 =  ~-1 +  beta_0(1, prec.linear = 0.01) + beta_1(x, model = "linear")
```

Task

Run the model again using 0.1 as default precision for both the intercept and the slope parameter.
Click here to see the solution

```
cmp2 =  ~ -1 +
          beta_0(1, prec.linear = 0.1) +
          beta_1(x, model = "linear", prec.linear = 0.1)

lm.fit2 = bru(cmp2, lik)
```

Note that we can use the same observation model as before since both the formula and the dataset are unchanged.

Change the prior for the precision of the observation error τ

Priors on the hyperparameters of the observation model must be passed by defining argument `hyper` within `control.family` in the call to the `bru_obs()` function.

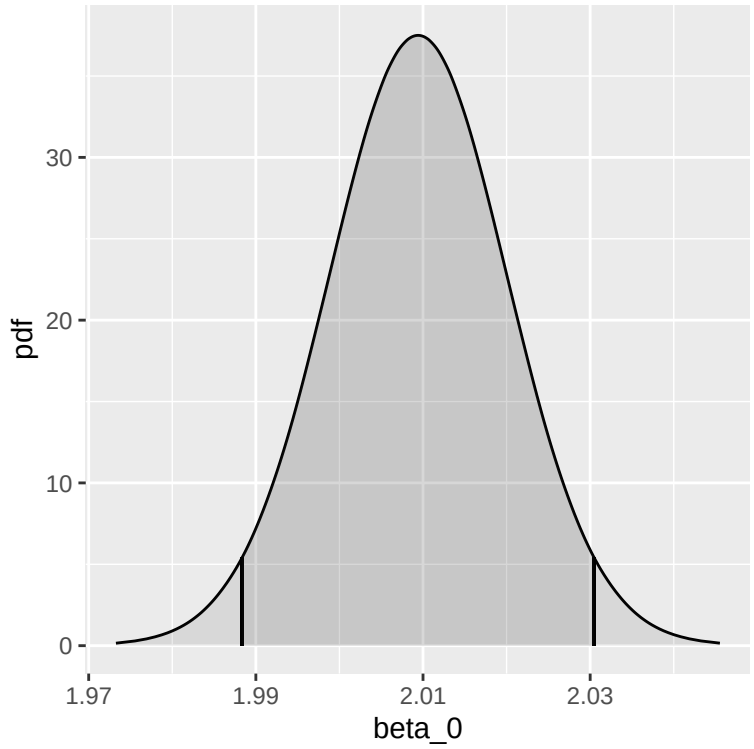
```
# First we define the logGamma (0.01,0.01) prior

prec.tau <- list(prec = list(prior = "loggamma",   # prior name
                             param = c(0.01, 0.01))) # prior values

lik2 =  bru_obs(formula = y ~.,
                family = "gaussian",
                data = df,
                control.family = list(hyper = prec.tau))

fit.lm2 = bru(cmp2, lik2)
```

The names of the priors available in **R-INLA** can be seen with `names(inla.models()$prior)`

1 Visualizing the posterior marginals

Posterior marginal distributions of the fixed effects parameters and the hyperparameters can be visualized using the `plot()` function by calling the name of the component. For example, if want to visualize the posterior density of the intercept β_0 we can type:

```
plot(fit.lm, "beta_0")
```



Task

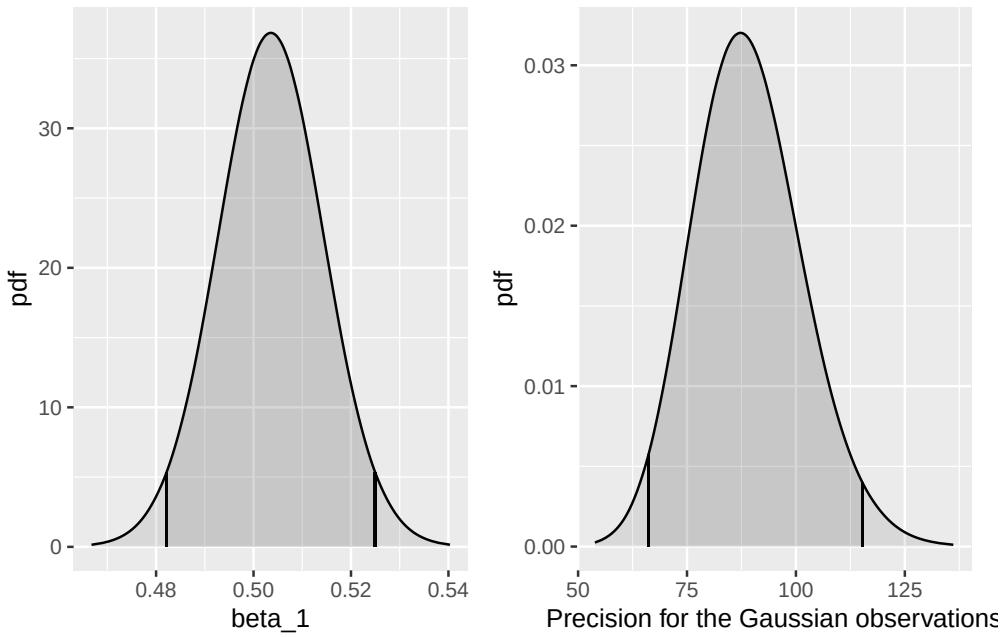
Plot the posterior marginals for β_1 and for the precision of the observation error $\pi(\tau|y)$

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")
```



2 Linear Mixed Model for fish weight-length relationship

In this exercise we will:

- Plot random effects of a LMM
- Compute posterior densities and summaries for the variance components

Libraries to load:

```
library(dplyr)
library(INLA)
library(ggplot2)
library(patchwork)
library(inlabru)
```

In this exercise, we will use a subset of the Pygmy Whitefish (*Prosopium coulterii*) dataset from the `FSAdata` R package, containing biological data collected in 2001 from Dina Lake, British Columbia.

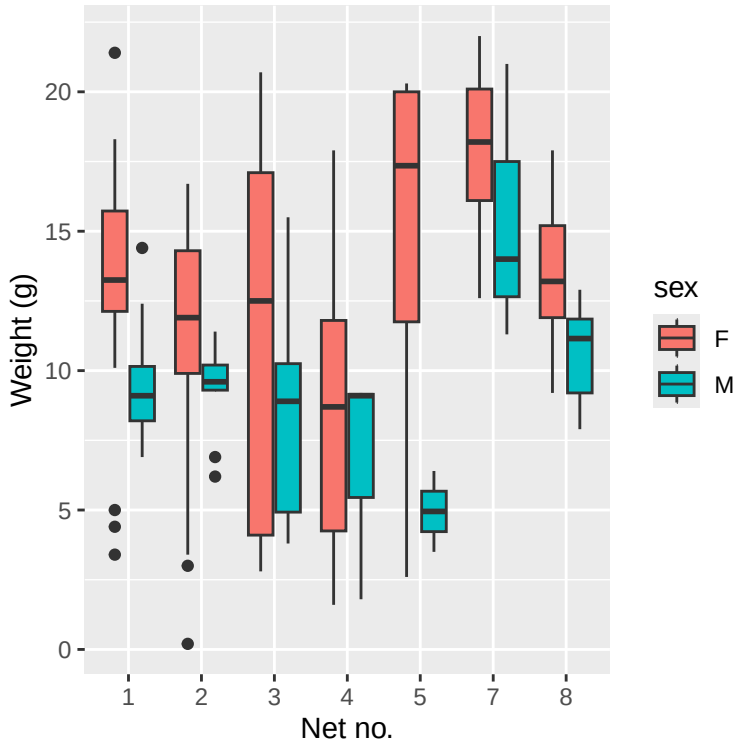
The data set contains the following information:

- `net_no`Unique net identification number
- `wt` Fish weight (g)
- `tl` Total fish length (cm)
- `sex` Sex code (`F`=Female, `M` = Male)

We can visualize the distribution of the response (weight) across the nets split by sex as follows:

```
PygmyWFBC <- read.csv("datasets/PygmyWFBC.csv")

ggplot(PygmyWFBC, aes(x = factor(net_no), y = wt,fill = sex)) +
```

```
  geom_boxplot() +
  labs(y="Weight (g)",x = "Net no.")
```



Suppose we are interested in modelling the weight-length relationship for captured fish. The exploratory plot suggest some important variability in this relationship, potentially attributable to differences among sampling nets deployed across various sites in the Dina Lake.

To account for this between-net variability, we model net as a random effect using the following linear mixed model:

$$
\begin{aligned}
y_{ij} &\sim \mathcal{N}(\mu_{ij}, \sigma_e^2), \qquad i = 1, \dots, a \qquad j = 1, \dots, n \\
\eta_{ij} &= \mu_{ij} = \beta_0 + \beta_1 \times \text{length}_j + \beta_2 \times \mathbb{I}(\text{Sex}_{ij} = \text{M}) + u_i \\
u_i &\sim \mathcal{N}(0, \sigma_u^2)
\end{aligned}
$$

where:

- y_{ij} is the weight of the j-th fish from net i
- length_{ij} is the corresponding fish length
- $\mathbb{I}(\text{Sex}_{ij} = \text{M})$ is an indicator/dummy such that for the *i*th net

$$
\mathbb{I}(\text{Sex}_{ij}) \begin{cases} 1 & \text{if the } j\text{th fish is Male} \\ 0 & \text{otherwise} \end{cases}
$$

- u_i represents the random intercept for net i
- σ_u^2 and σ_ϵ^2 are the between-net and residual variances, respectively

To run this model in`inlabru` we first need to create our sex dummy variable :

```
PygmyWFBC$sex_M <- ifelse(PygmyWFBC$sex=="F",0,1)
```

inlabru will treat 0 as the reference category (i.e., the intercept β_0 will represent the baseline weight for females). Now we can define the model component, the likelihood and fit the model.

```
cmp =  ~ -1 + sex_M +
  beta_0(1)  +
  beta_1(tl, model = "linear") +
  net_eff(net_no, model = "iid")

lik =  bru_obs(formula = wt ~ .,
               family = "gaussian",
               data = PygmyWFBC)

fit = bru(cmp, lik)

summary(fit)
```

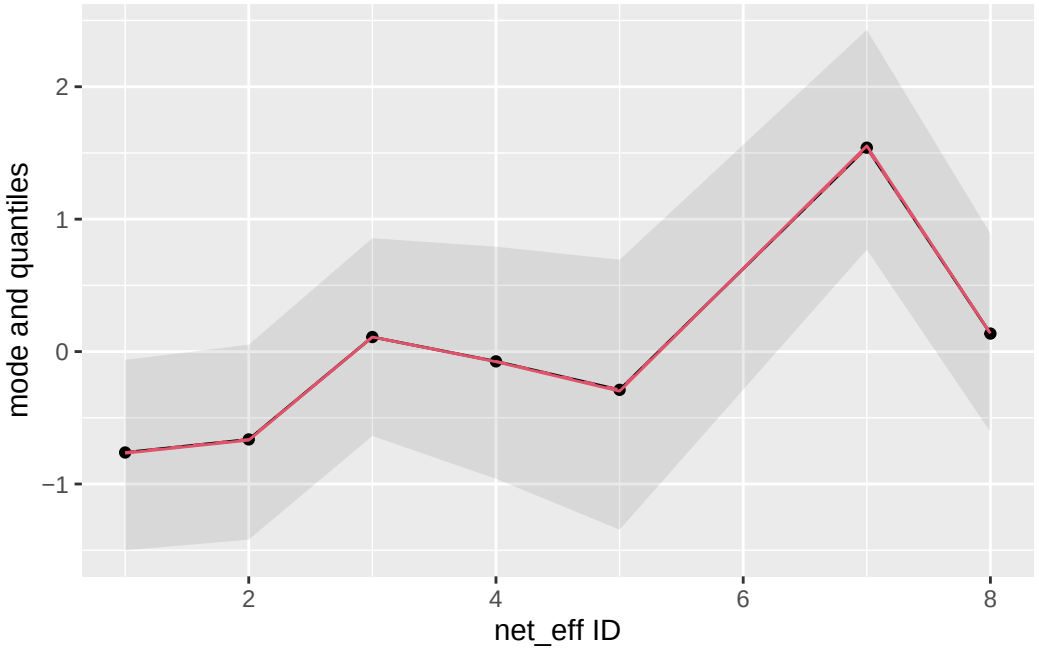
```
inlabru version: 2.13.0.9011
INLA version: 25.09.19
Components:
Latent components:
sex_M: main = linear(sex_M)
beta_0: main = linear(1)
beta_1: main = linear(tl)
net_eff: main = iid(net_no)
Observation models:
  Family: 'gaussian'
    Tag: <No tag>
    Data class: 'data.frame'
    Response class: 'numeric'
    Predictor: wt ~ .
    Additive/Linear: TRUE/TRUE
    Used components: effects[sex_M, beta_0, beta_1, net_eff], latent[]
Time used:
    Pre = 0.909, Running = 0.198, Post = 0.034, Total = 1.14
Fixed effects:
          mean    sd 0.025quant 0.5quant 0.975quant    mode kld
sex_M   -1.106 0.218     -1.534   -1.106     -0.678  -1.106   0
beta_0 -15.815 0.870    -17.516  -15.818    -14.098 -15.818   0
beta_1   2.555 0.072      2.414    2.555      2.696   2.555   0

Random effects:
  Name    Model
    net_eff IID model

Model hyperparameters:
                                          mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations 0.475 0.044      0.393    0.473
Precision for net_eff                   2.150 1.324      0.565    1.837
                                        0.975quant mode
Precision for the Gaussian observations      0.567 0.47
```

```
Precision for net_eff                          5.563 1.32

Marginal log-Likelihood:  -467.54
 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)')
```

For interpretability, we could have centered the predictors, but our primary focus here is on estimating the variance components of the mixed model.

We can plot the posterior density of the nets random intercept as follows:

```
plot(fit,"net_eff")
```



For theoretical and computational purposes, INLA works with the precision which is the inverse of the variance. To obtain the posterior summaries on the SDs scale we can sample from the posterior distribution for the precision while back-transforming the samples and then computing the summary statistics. Transforming the samples is necessary because some quantities such as the mean and mode are not invariant to monotone transformation; alternatively we can use some of the in-built `inlabru` functions to achieve this (see supplementary note).

We use the `predict` function to draw samples from the approximated joint posterior for the hyperparameters, then invert them to get variances and lastly compute the mean, std. dev., quantiles, etc.

> **Note**
>
> To get the right name for the hyperparameters to use in the `predict()` function, you can use the function `bru_names()`.

```
sampvars <-  predict(fit,PygmyWFBC, ~ {
   tau_e <- Precision_for_the_Gaussian_observations
   tau_u <- Precision_for_net_eff
   list(sigma_u = 1/tau_u,
```

```
        sigma_e = 1/tau_e)
    },
    n.samples = 1000
   )

  names(sampvars) = c("Error variance","Between-net Variance")

  sampvars
```

```
$`Error variance`
       mean        sd    q0.025      q0.5    q0.975    median sd.mc_std_err
1 0.6485094 0.4603619 0.179613 0.5215865 1.716175 0.5215865    0.03294335
  mean.mc_std_err
1      0.01664144

$`Between-net Variance`
      mean        sd    q0.025     q0.5    q0.975   median sd.mc_std_err
1 2.130805 0.1884648 1.761079 2.108925 2.543643 2.108925   0.005137609
  mean.mc_std_err
1     0.006284711

attr(,"class")
[1] "bru_prediction" "list"
```

Task

Another useful quantity we can compute is the intraclass correlation coefficient (ICC) which help us determine how much the response varies within groups compared to between groups. The intraclass correlation coefficient is defined as:

$$\text{ICC} = \frac{\sigma_u^2}{\sigma_u^2 + \sigma_e^2}$$

Compute the mean, median, and quantiles for the ICC by drawing posterior samples for σ_e^2 and σ_u^2 using the `predict` function.
Click here to see the solution

```
ICC <-  predict(fit,PygmyWFBC, ~ {
    tau_e <- Precision_for_the_Gaussian_observations
    tau_u <- Precision_for_net_eff
    sigma_u = 1/tau_u
    sigma_e = 1/tau_e
    list(ICC = sigma_u/ (sigma_u+sigma_e))
    },
    n.samples = 1000
   )
```

Supplementary Material

The marginal densities for the hyper parameters can be also found by calling`inlabru_model$marginals.hyperpar`. We can then apply a transformation using the `inla.tmarginal` function to transform the precision posterior distribu-

tions.

```
var_e <- fit$marginals.hyperpar$`Precision for the Gaussian observations` %>%
  inla.tmarginal(function(x) 1/x,.)

var_u <- fit$marginals.hyperpar$`Precision for net_eff` %>%
  inla.tmarginal(function(x) 1/x,.)
```

The marginal densities for the hyper parameters can be found with `inlabru_model$marginals.hyperpar`, then we can apply a transformation using the `inla.tmarginal` function to transform the precision posterior distributions. Then, we can compute posterior summaries using `inla.zmarginal` function as follows:

```
post_var_summaries <- cbind( inla.zmarginal(var_e,silent = T),
                             inla.zmarginal(var_u,silent = T))
colnames(post_var_summaries) <- c("sigma_e","sigma_u")
post_var_summaries
```

```
           sigma_e   sigma_u
mean       2.124465  0.6499602
sd         0.1981121 0.4144963
quant0.025 1.764751  0.1802423
quant0.25  1.985333  0.3685947
quant0.5   2.11362   0.5420264
quant0.75  2.251957  0.8074379
quant0.975 2.542278  1.747777
```