PRACTICAL 2 - GLMM

In this practical we are going to fit a Generalized (Mixed) Linear Model in `inlabru`.

We are going to:

- Fit a Poisson regression
- Change the prior distributions for the model hyperparameters (both for fixed and random effects)
- Compute and visualize posterior densities and summaries for marginal effects

We start by loading some useful libraries

```
library(dplyr)
library(INLA)
library(ggplot2)
library(patchwork)
library(inlabru)
library(lme4)
```

In this practical we are going to analyse the dataset `grouseticks`. The data, contained in the library `lme4`, Number of ticks on the heads of red grouse chicks sampled in the field. See `?grouseticks` for details.

The data are analysed in the paper

Elston et al. "Analysis of aggregation, a worked example: numbers of ticks on red grouse chicks." Parasitology

and in this practical we will follow their analysis.

```
data(grouseticks)
?grouseticks
head(grouseticks)
```

```
  INDEX TICKS BROOD HEIGHT YEAR LOCATION   cHEIGHT
1     1     0   501    465   95       32  2.759305
2     2     0   501    465   95       32  2.759305
3     3     0   502    472   95       36  9.759305
4     4     0   503    475   95       37 12.759305
5     5     0   503    475   95       37 12.759305
6     6     3   503    475   95       37 12.759305
```

0 Fitting Poisson regression

We assume that the number of ticks y_{ijk} counted on chick i of brood j in year k follows a Poisson distribution with mean λ_{ijk}

$$y_{ijk}|\lambda_{ijk} \sim \text{Poisson}(\lambda_{ijk})$$

We then model log mean counts $\eta_{ijk} = \log(\lambda_{ijk})$ as a linear function of year, altitude, brood, and individual chick within brood. We assume a fixed effect α_k of year k, a linear

effect of altitude x_{ij}, two random effects e_{jk} and ϵ_{ijk} brood and individual within brood respectively. Thus:

$$\eta_{ijk} = \log(\lambda_{ijk}) = \alpha_k + \beta x_{ij} + e_{jk} + \epsilon_{ijk} \quad (1)$$

where $e_{jk} \sim \mathcal{N}(0, \tau_e^{-1})$ and $\epsilon_{jk} \sim \mathcal{N}(0, \tau_\epsilon^{-1})$.

We first fit the model using the default priors for fixed and random effects.

To fit the model we first have to create two more variable, one that indexes the combination jk of brood and year and another that indexes the combination ijk of individuals per brood per year.

```
grouseticks  = grouseticks %>%
  group_by(BROOD, YEAR) %>%
  mutate(ij = cur_group_id()) %>%
  ungroup() %>%
  mutate(ijk = seq_along(INDEX))
```

now we can fit the model.

Task

Complete the code to fit the model in Equation 1. You need to define the components, the likelihood and finaly use the `bru()` function to get the results

```
cmp= ~ -1 + year(...) +
  height(...) +
  brood_year(...) +
  brood_year_chicken(...)

lik = bru_obs(forumula = ...,
              data = ...,
              family = ...)

fit = bru(cmp, lik)
```

Click here to see the solution

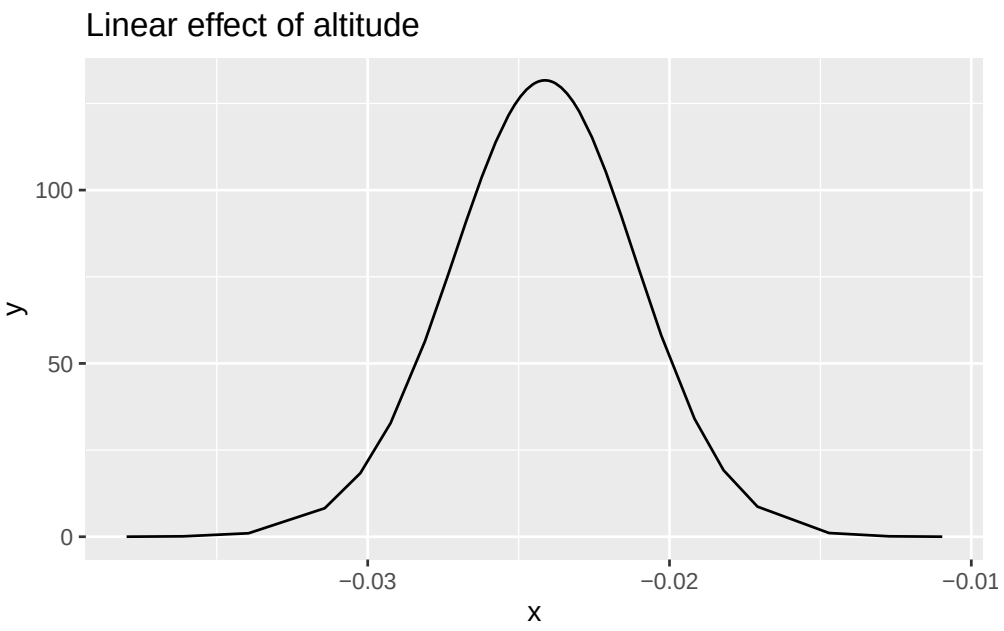
```
cmp= ~ -1 + year(YEAR, model = "iid", initial = log(0.01), fixed = T) +
  height(cHEIGHT, model =  "linear") +
  brood_year(ij , model = "iid") +
  brood_year_chicken(ijk, model= "iid")

lik = bru_obs(TICKS ~ .,
              data = grouseticks,
              family = "Poisson")


fit = bru(cmp, lik)
```

Now we can check the results.

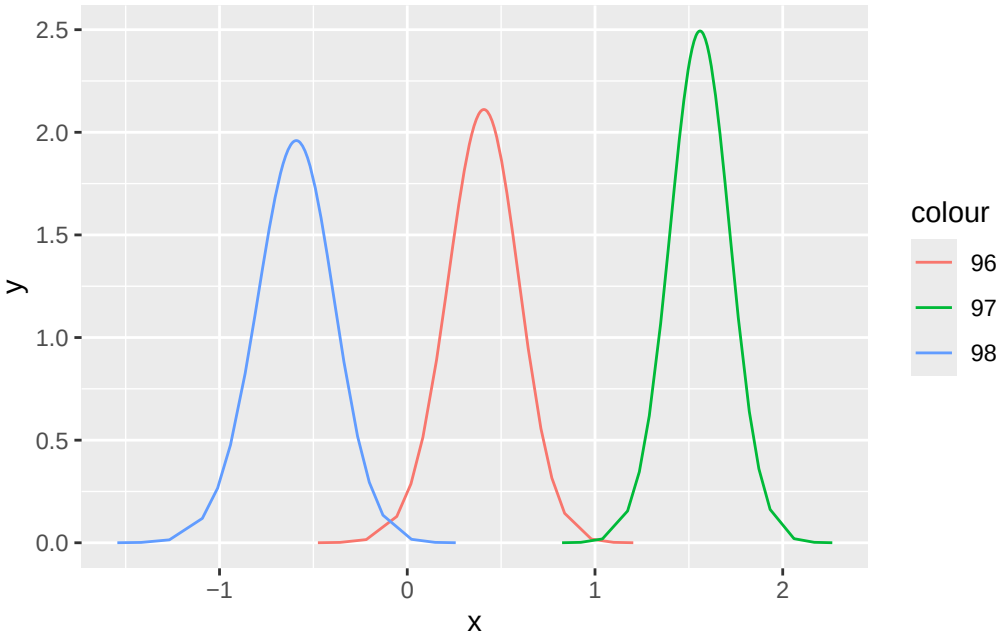
Task

Plot the posterior distribution of the fixed effects: β and α_k.
Click here to see the solution

```
fit$marginals.fixed$height %>% ggplot() + geom_line(aes(x,y)) + ggtitle("Linear effect of
```




```
ggplot() + geom_line(data = fit$marginals.random$year[[1]], aes(x,y, color = "96") )+
  geom_line(data = fit$marginals.random$year[[2]], aes(x,y, color = "97")) +
  geom_line(data = fit$marginals.random$year[[3]], aes(x,y, color = "98"))
```



We now want to look at the hyperparameters of the model

Task

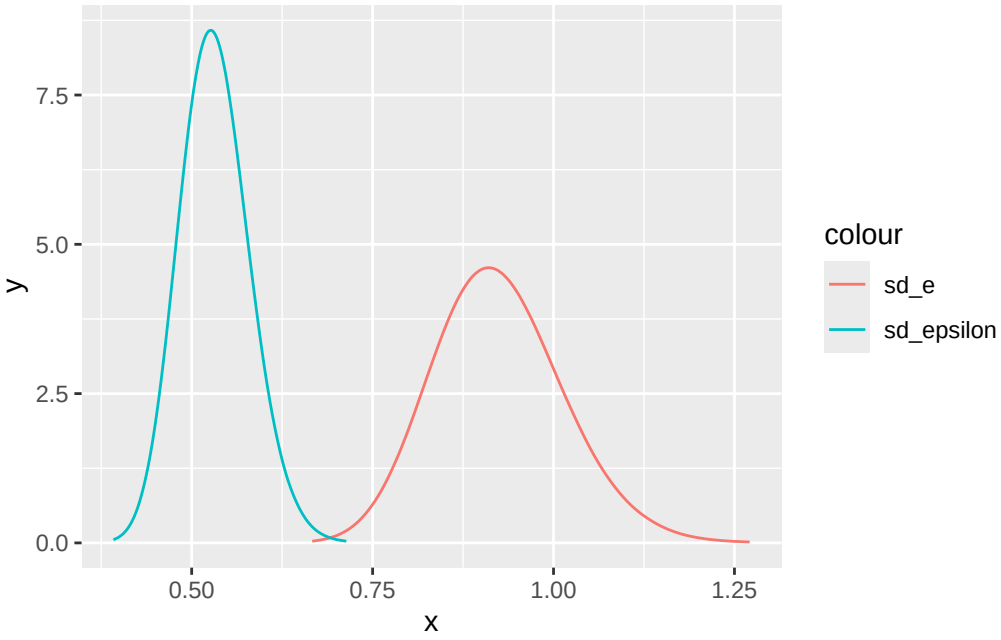
Plot the posterior estimated of the standard deviation $\sigma_e = \sqrt{1/\tau_e}$ and $\sigma_\epsilon = \sqrt{1/\tau_\epsilon}$ of the two random effects.

Take hint

You can use the function `inla.tmarginal()` or the `generate()` function

Click here to see the solution

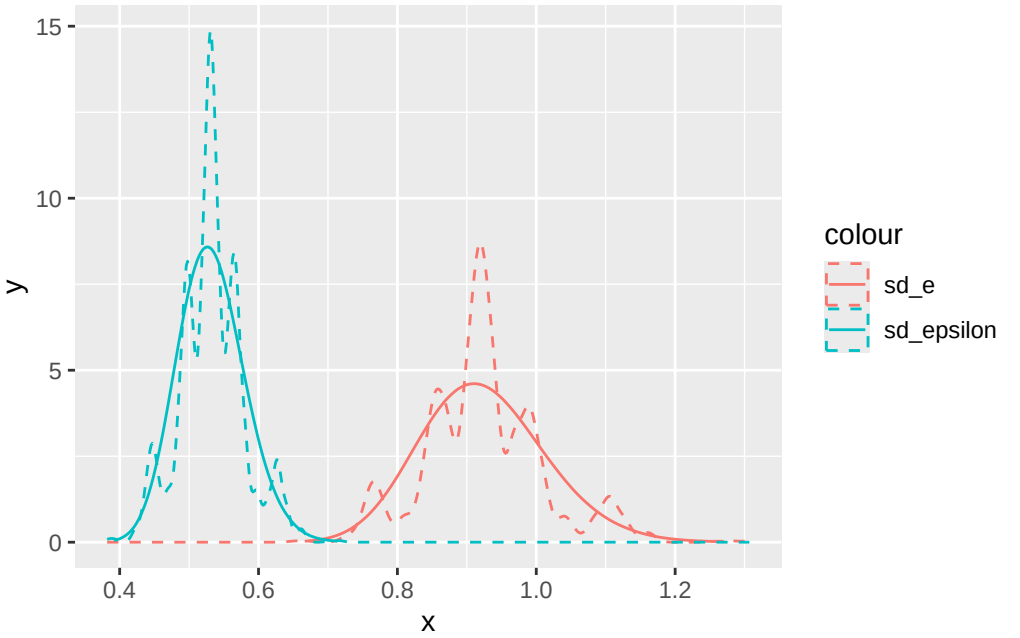
```
# Method 1 : use the inla.tmarginal() function

sd_e = inla.tmarginal(fun = function(x)sqrt(1/x), fit$marginals.hyperpar$`Precision for br

sd_eps = inla.tmarginal(fun = function(x)sqrt(1/x), fit$marginals.hyperpar$`Precision for

ggplot() + geom_line(data = sd_e, aes(x,y, color = "sd_e")) + geom_line(data = sd_eps, aes
```




```
# Method 1 : use the generate() function


# we use this to get the correct name of the parameters
bru_names(fit)
```

```
                    height                     year
                  "height"                   "year"
                 brood_year        brood_year_chicken
               "brood_year"      "brood_year_chicken"
  Precision for brood_year Precision for brood_year_chicken
"Precision_for_brood_year" "Precision_for_brood_year_chicken"
```

```
out = generate(fit, formula = ~ data.frame(sd_e = sqrt(1/Precision_for_brood_year),
                                           sd_eps = sqrt(1/Precision_for_brood_year_chicken)),
               n.samples = 10000)

sd_sample = data.frame(sd_e = unlist(sapply(out, function(x) x[1])),
                       sd_eps = unlist(sapply(out, function(x) x[2])))


ggplot() + geom_line(data = sd_e, aes(x,y, color = "sd_e")) +
  geom_density(data = sd_sample, aes(x = sd_e, color = "sd_e"), linetype = "dashed") +
  geom_line(data = sd_eps, aes(x,y, color = "sd_epsilon"))+
  geom_density(data = sd_sample, aes(x = sd_eps, color = "sd_epsilon"), linetype = "dashed
```




```
# Note that, if one is interested in hyperparameters, sampling is not always the best choi
```

1 Change prior distributions

Before trying to change the priors for the hyperparameters we can check which priors are actually used in the model

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

```
section=[family]
    tag=[INLA.Data1] component=[poisson]
section=[random]
    tag=[year] component=[year]
        group.theta1:
            parameter=[logit correlation]
            prior=[normal]
            param=[0.0, 0.2]
```

```
    tag=[] component=[]
    tag=[brood_year] component=[brood_year]
        theta1:
            parameter=[log precision]
            prior=[loggamma]
            param=[1e+00, 5e-05]
        group.theta1:
            parameter=[logit correlation]
            prior=[normal]
            param=[0.0, 0.2]
    tag=[brood_year_chicken] component=[brood_year_chicken]
        theta1:
            parameter=[log precision]
            prior=[loggamma]
            param=[1e+00, 5e-05]
        group.theta1:
            parameter=[logit correlation]
            prior=[normal]
            param=[0.0, 0.2]
section=[linear]
    tag=[height] component=[height]
        beta:
            parameter=[height]
            prior=[normal]
            param=[0.000, 0.001]
```

From the output we see that the precision for the linear effect of altitude is 0.001 (which means the sd is $1/\sqrt{0.001} = 31.62$)

The precisions for the random effects have a Gamma prior with parameters 1 and 5e-05.

1 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 0.1 for we define the relative components as:

```
cmp= ~ -1 + year(YEAR, ... ) +
  height(cHEIGHT, model =  "linear", prec.linear = 0.1) + ...
```

Task

Run the model again using 0.1 as default precision for the altitude slope parameter and for the `year` fixed effect
Click here to see the solution

```
cmp2 =  ~ -1 + year(YEAR, model = "iid", initial = log(0.1), fixed = T) +
  height(cHEIGHT, model =  "linear", prec.linear = 0.1) +
  brood_year(ij , model = "iid") +
  brood_year_chicken(ijk, model= "iid")

fit2 = bru(cmp2, lik)
```

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

Changing the precision for the random effects

Priors on the hyperparameters of the random effects model must be passed by defining argument `hyper` within component of interest

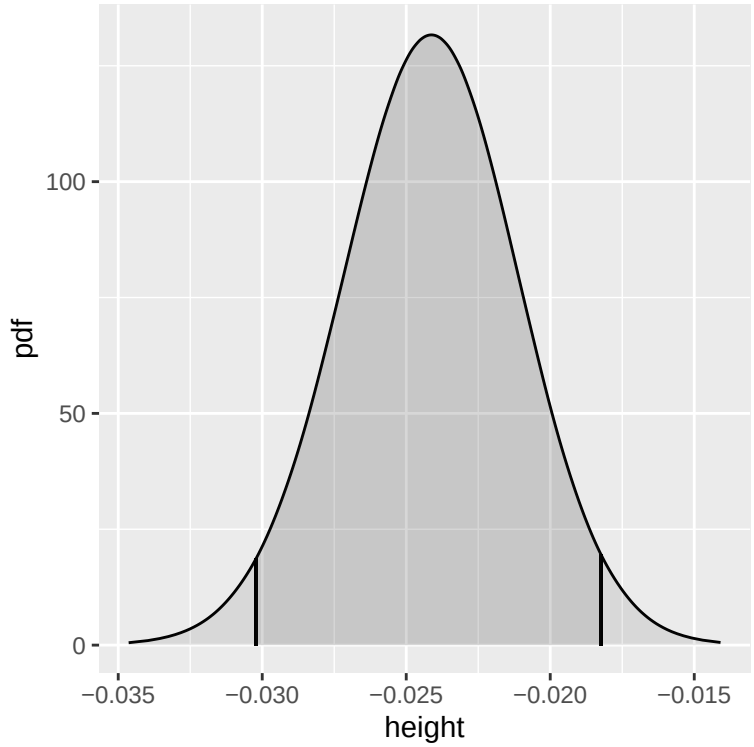
```
# First we define the logGamma (0.01,0.01) prior

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


cmp3 =  ~ -1 + year(YEAR, model = "iid", initial = log(0.1), fixed = T) +
  height(cHEIGHT, model =  "linear", prec.linear = 0.1) +
  brood_year(ij , model = "iid", hyper = prec.prior) +
  brood_year_chicken(ijk, model= "iid", hyper = prec.prior)


fit3 = bru(cmp3, lik)
```

Note that for `model = "linear"` the value for the precision is expressed in natural scale while for `model = "iid'` (and all other random effects) the value is expressed in log scale.

1.1.1 Visualizing the posterior marginals

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

```
plot(fit, "height")
```



Task

Plot the posterior marginals for the precision of brood and individual random effect $\pi(\tau_e|y)$

Take hint

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

Click here to see the solution

```
plot(fit, "Precision for brood_year")
```