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.
- Compare models using DIC and WAIC

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

Start by loading useful libraries:

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

At the end of this exercise we are going to compare the different models we fit using DIC and WAIC. Therefore we need to tell the `bru()` function to compute those scores, we can do that by the `bru_options_set()` functions:

```
bru_options_set(control.compute = list(dic = T, waic = T))
```

This is a global option that will make `bru()` compute scores everytime. It is also possible to set the option locally as

```
fit = bru(cmp, lik,
          options = list(control.compute = list(dic = TRUE)))
```

1 Simple linear regression

We consider a simple linear regression model with Gaussian observations

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

where $\sigma^2 y$ 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 $\tau_y = 1/\sigma_y^2 \sim$ Gamma(a, b) and $\beta_0, \beta_1 \sim \mathcal{N}(0, 1/\tau_\beta)$ (we will use the default prior settings in `R-INLA` for now).

Question

What is the dimension of the hyperparameter vector and latent Gaussian field?
Answer
The hyperparameter vector has dimension 1, $\boldsymbol{\theta} = (\tau_y)$ 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

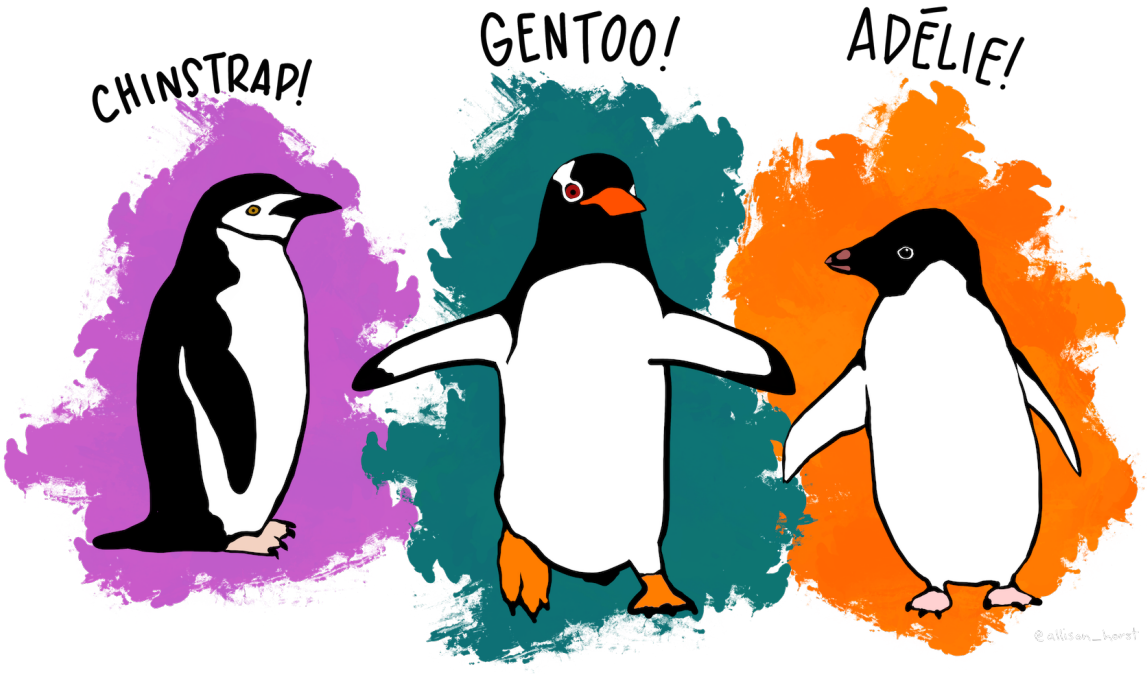
$$\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}_1\boldsymbol{u}_1 + \boldsymbol{A}_2\boldsymbol{u}_2 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \beta_0 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \beta_1$$

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

2 Modelling penguins bodymass

Let's look at the dataset `penguins` in R.

These are data on adult penguins covering three species found on three islands in the Palmer Archipelago, Antarctica, including their size (flipper length, body mass, bill dimensions), and sex.



```
data("penguins")
glimpse(penguins)
```

```
Rows: 344
Columns: 8
$ species     <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Ad~
$ island      <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen, Tor~
$ bill_len    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, 42.0, ~
$ bill_dep    <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, 20.2, ~
$ flipper_len <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186, 180,~
$ body_mass   <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, 4250, ~
$ sex         <fct> male, female, female, NA, female, male, female, male, NA, ~
$ year        <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

The dataset contains the following variables:

- `species` a factor denoting penguin species (Adélie, Chinstrap and Gentoo)
- `island` a factor denoting island in Palmer Archipelago, Antarctica (Biscoe, Dream or Torgersen)
- `bill_length` a number denoting bill length (millimeters)
- `bill_depth_mm` a number denoting bill depth (millimeters)
- `flipper_length` an integer denoting flipper length (millimeters)
- `body_mass` an integer denoting body mass (grams)
- `sex` a factor denoting penguin sex (female, male)
- `year` and integer indicating the study year: (2007, 2008, 2009)

We are going to use the bodymass as our response variable. To make the interpretation easier we are going to express the bodymass in Kg instead of grams.

```
penguins$body_mass = penguins$body_mass/1000
```

In general, while INLA can fit models with missing observations in the response by computing their predictive distribution, it cannot directly handle missing observations in the covariates, since these enter the latent field. For this reason, we will remove rows with missing values before fitting.

```
penguins = penguins %>%  drop_na()
```

> **!** Important
>
> Note that you should be careful when removing NA's from your data. Dropping incomplete rows is generally safe when the values are missing at random. However, if missingness is related to the outcome or to the covariates themselves, discarding those rows can bias the results, so the pattern of missing data should always be inspected before removing anything.

2 Fitting a linear regression model with `inlabru`

In a first step we want to fit a simple linear regression model to link the body mass (`body_mass`) to flipper length (`flipper_len`).

$$y_i \sim \mathcal{N}(\mu_i, \sigma_y^2)$$
$$\mu_i = \beta_0 + \beta_i x_i$$

where y_i indicates the body mass and x_i the flipper length of penguin i.

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) and a linear effect `beta_1` of the `flipper_len`.
> 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(flipper_len, 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(flipper_len, model = "linear")
```

Another way to code this model would be to use the `model = "fixed"`. In this case we would define the components as:

```
cmp = ~ -1 + effects(~ flipper_len, model = "fixed")
```

Specifying components this way, generate a model matrix automatically via `MatrixModels::model.Matrix(formula, data = .data.)`. This lets users define individual fixed effects and their interactions concisely, drawing on functionality already provided by the `MatrixModels` package. Also notice that we have explicitly removed the global intercept from the model components. This is because specifying `model = "fixed"` calls the `model.Matrix` function which automatically creates an `(Intercept)` column in the design matrix.

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

```
formula = body_mass ~ effects
```

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 `penguins`

Click here to see the solution

```
lik = bru_obs(formula = formula,
              data = penguins,
              family = "Gaussian")
```

Step 3: Fit the model

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

```
fit1 = bru(cmp, lik)
```

Step 4: 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()` function which gives access to some basic information about model fit and estimates

```
summary(fit1)
## inlabru version: 2.15.0.9002
## INLA version: 26.08.22
## Latent components:
## effects: main = fixed(~flipper_len)
## Observation models:
##   Model tag: <No tag>
##     Family: 'Gaussian'
##     Data class: 'data.frame'
##     Response class: 'numeric'
```

```
##     Predictor: body_mass ~ effects
##     Additive/Linear/Rowwise: TRUE/TRUE/TRUE
##     Used components: effect[effects], latent[]
## Time used:
##     Pre = 1.91, Running = 0.482, Post = 0.0336, Total = 2.42
## Random effects:
##   Name     Model
##     effects IID model
##
## Model hyperparameters:
##                                          mean    sd 0.025quant 0.5quant
## Precision for the Gaussian observations 6.50 0.504       5.55     6.49
##                                          0.975quant mode
## Precision for the Gaussian observations       7.53 6.46
##
## Deviance Information Criterion (DIC) ...............: 327.57
## Deviance Information Criterion (DIC, saturated) ....: 338.38
## Effective number of parameters .....................: 2.99
##
## Watanabe-Akaike information criterion (WAIC) ...: 327.53
## Effective number of parameters .................: 2.92
##
## Marginal log-Likelihood:  -187.71
##  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 further inspect posterior summaries of hyperparameters, fixed and random effects by calling `model$summary.hyperpar`, `model$summary.fixed` and `model$summary.random` respectively.

Note that for implementation technical reasons, `model = "fixed"` will make the estimated parameters to appear in `summary.random` instead of the normal `summary.fixed` part of the `inlabru` output object. E.g.,

```
fit1$summary.random
```

```
$effects
           ID        mean          sd  0.025quant    0.5quant  0.975quant
1 (Intercept) -5.87152801 0.310089701 -6.47995937 -5.87152970 -
5.26308705
2 flipper_len  0.05015047 0.001539261  0.04713022  0.05015048  0.05317068
         mode          kld
1 -5.87152969 5.749127e-10
2  0.05015048 5.748692e-10
```

2 Predictions with `inlabru`

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 η given a new set of values for the penguins flipper length

```
new_data = data.frame(flipper_len  = 170:240)

pred = predict(fit1, new_data, ~ effects,
               n.samples = 1000)
```

The `predict()` function generates samples from the fitted model and then summarizes those samples by computing some statistics like posterior mean, standard deviation, quantiles etc. In this case we set the number of samples to 1000.

> **Note**
>
> Note that we are predicting within the range of our covariate, as extrapolating beyond the observed data is risky because the model has no information there, so predictions rely entirely on the assumed functional form, and their uncertainty can be badly understated

We can plot the predictions for η together with the observed data:

```
pred %>% ggplot() +
  geom_line(aes(flipper_len,mean)) +
  geom_ribbon(aes(flipper_len, ymin = q0.025, ymax = q0.975), alpha = 0.5) +
  xlab("Flipper length") + ylab("body_mass") +
  geom_point(data = penguins, aes(flipper_len, body_mass))
```

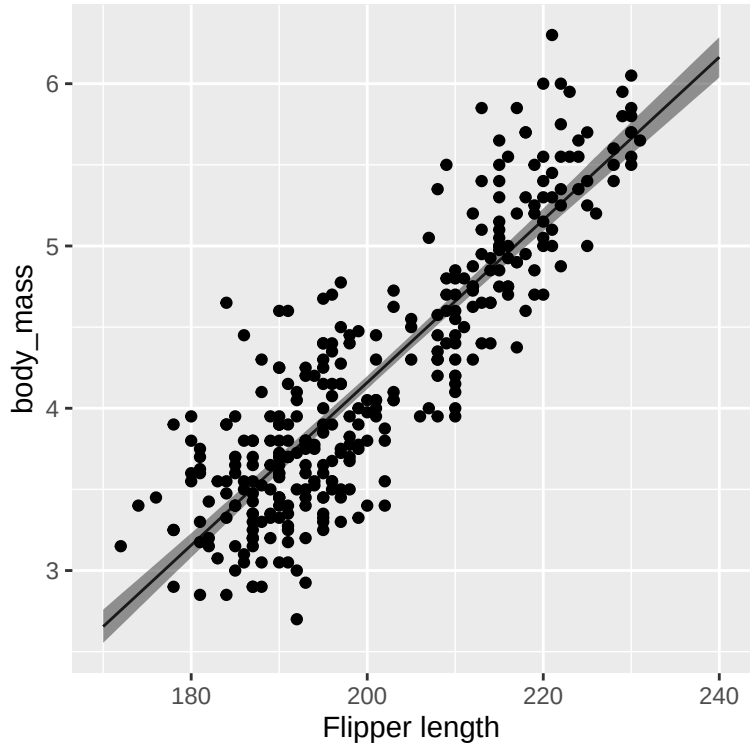


Figure 1: Data and 95% credible intervals

NOTE: The uncertainty we have computed now is relative to the prediction of η. If we want to predict new body mass, we need to add the observation (likelihood) uncertainty σ_y^2. We can do this using `predict()` again:

```
new_data = data.frame(flipper_len  = 170:240)

pred1 = predict(fit1, new_data,
                formula = ~ { eta = effects
                              sigma = sqrt(1/Precision_for_the_Gaussian_observations)
                              list(mean = eta,
                                   q1 = qnorm(0.025, mean = eta, sd = sigma),
                                   q2 =  qnorm(0.975, mean = eta, sd = sigma))
                            },
                n.samples = 1000)
```

Let's compare the two predictions:

```
ggplot() +
  geom_line(data = pred, aes(flipper_len,mean)) +
  geom_ribbon(data = pred,aes(flipper_len, ymin = q0.025, ymax = q0.975), alpha = 0.5) +
  geom_line(data = pred1$mean, aes(flipper_len,mean), color = "red") +
  geom_line(data = pred1$q1,aes(flipper_len, mean),
                color = "red") +
   geom_line(data = pred1$q2,aes(flipper_len, mean),
                color = "red") +
  xlab("flipper length") + ylab("body_mass") +
  geom_point(data = penguins, aes(flipper_len, body_mass))
```

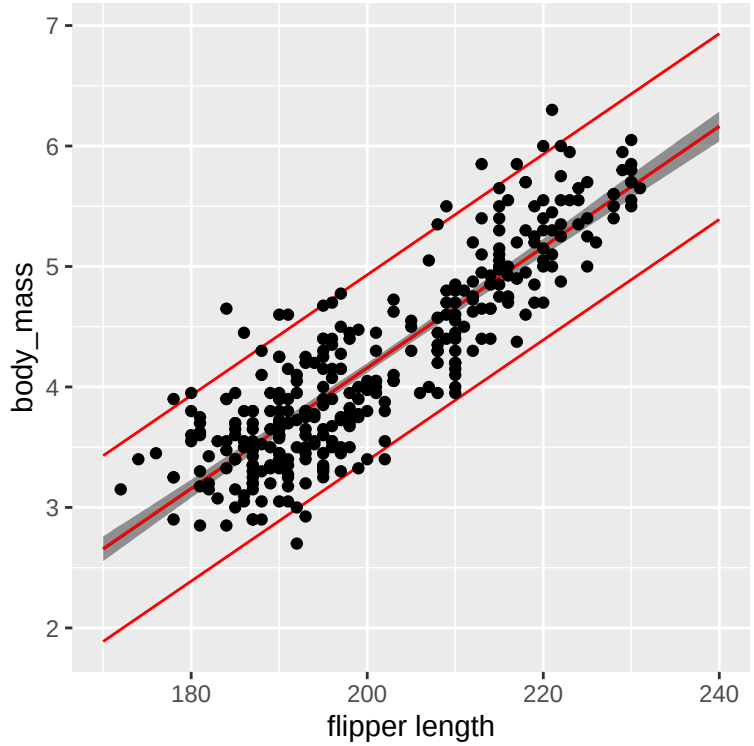


Figure 2: Data and 95% credible intervals

Task

Compute predictions for the mean body mass η when the flipper length is $x_0 = 193$ mm

What is the predicted value for η? 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(flipper_len = 193)
pred = predict(fit1, new_data, ~ effects,
               n.samples = 1000)

pred
```

```
  flipper_len     mean         sd  q0.025     q0.5   q0.975   median
1         193 3.807196 0.02545676 3.75609 3.807185 3.855756 3.807185
  mean.mc_std_err sd.mc_std_err
1    0.0008416666  0.0005795359
```

You can see the predicted mean and sd by examining the produced `pred` object. In this case the mean is ca 4 kg with sd ca 0.03. This gives a 95% CI ca [3.76, 3.86].

Task

On the previous task you computed a credible interval for the expected mean η when the flipper length is $x_0 = 193$ mm. Now produce a 95% *prediction* interval for a new observation y_0 (i.e., prediction interval for the body mass of a new penguin's whose flipper length is $x_0 = 193$ mm) by adding the uncertainty that comes from the likelihood with precision $\tau_y = 1/\sigma_y^2$
Take hint
You can use the `bru_names(fit1)` function to check the names for the different model components.
Click here to see the solution

```
pred2 = predict(fit1, new_data,
                formula = ~ {
                  mu = effects
                  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)
# Notice that now the interval we obtain is much bigger!
round(c(pred2$q1$mean, pred2$q2$mean),2)
```

```
[1] 3.04 4.58
```

3 Linear model with discrete variables and interactions

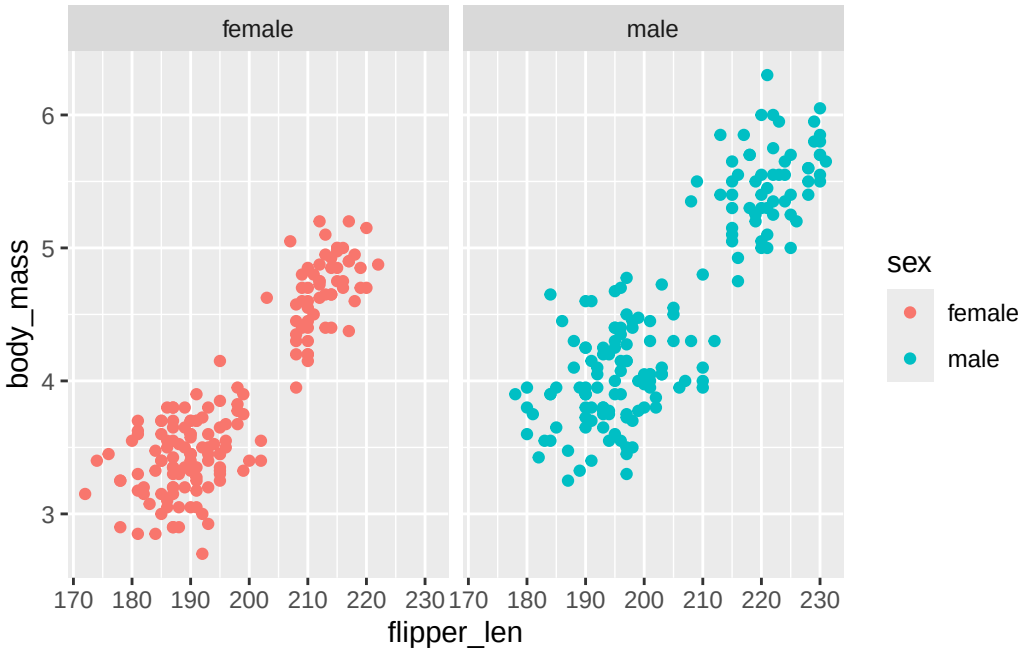
Now we want to check if there is any difference, both in mean body_mass and in mean increase of the body_mass with flipper length, between males and females

To do this we have to fit a model with interactions:

$$y_i \sim \mathcal{N}(\eta_i, \sigma_y^2)$$
$$\eta_i = \beta_0 + \beta_{0,\text{Male}} + \beta_1 x_i + \beta_{1,\text{Male}} x_i$$

Let's first look at an exploratory plot

```
penguins %>%
  ggplot() + geom_point(aes(flipper_len, body_mass, color= sex)) +
  facet_wrap(.~sex)
```



We fit the model using the `model = "fixed"` option.

```
cmp = ~ -1 + effects( ~ sex*flipper_len, model = "fixed")
formula = body_mass ~ .
lik = bru_obs(formula = formula,
              data = penguins
              )
fit2 = bru(cmp, lik)
```

Note that specifying a full stop in the linear predictor `~ .` is equivalent to summing all the components defined in the model specification. Instead of listing each component by name, the dot acts as a shorthand for "*include every component*" so `~ .` expands to the sum of all the effects you defined in `cmp`.

We can get the results looking at the `summary.random` object as follows:

```
fit2$summary.random
```

```
$effects
                   ID          mean          sd   0.025quant      0.5quant
1         (Intercept) -5.4428283012 0.439938837 -6.306038543 -5.4428316937
2             sexmale  0.4056078850 0.586824367 -0.745831558  0.4056114601
3         flipper_len  0.0471470110 0.002224648  0.042781899  0.0471470281
4 sexmale:flipper_len -0.0002882055 0.002921826 -0.006021174 -0.0002882235
```

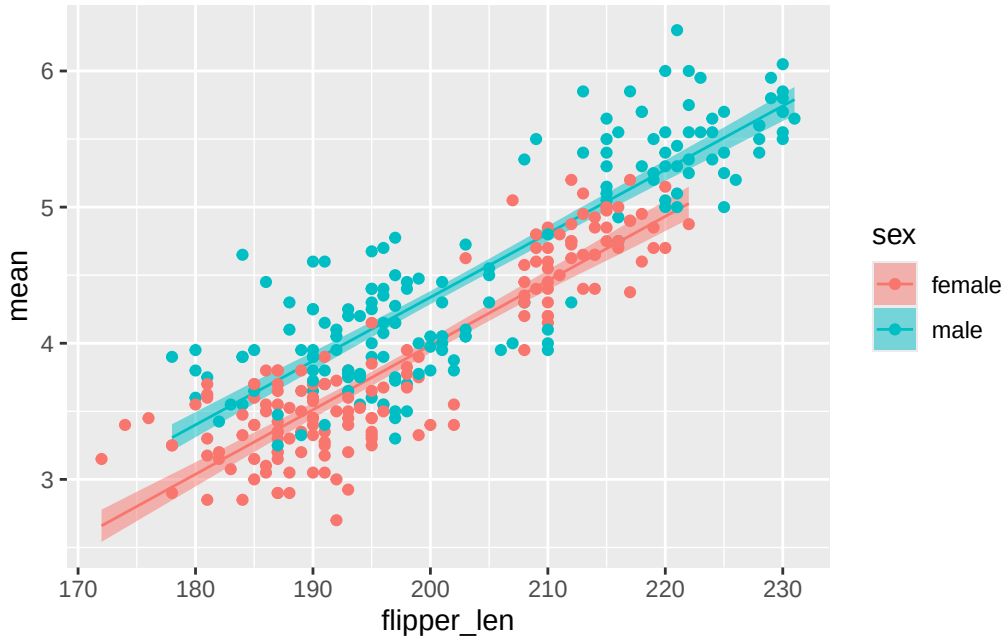
```
    0.975quant           mode          kld
1 -4.579598702 -5.4428316889 5.830033e-10
2  1.557026929  0.4056114551 5.828025e-10
3  0.051512025  0.0471470281 5.830160e-10
4  0.005444865 -0.0002882235 5.827987e-10
```

Task

Use the `predict()` function to obtain the two regression lines for each sex. Would you conclude that there are differences between males and females?
Click here to see the solution

```
pred = predict(fit2, penguins, ~ effects)

pred %>% ggplot() + geom_line(aes(flipper_len, mean, group = sex, color = sex)) +
  geom_ribbon(aes(flipper_len, ymin = q0.025, ymax = q0.975,
                  group = sex, fill = sex), alpha = 0.5) +
  geom_point(aes(flipper_len, body_mass,color = sex))
```




```
# Differences on baseline body mass only, not in the slope.
```

We can also use the `predict()` function to get an estimate of the mean increase of body_mass per mm of flipper length for males and females, i.e.,

- β_0: mean body mass for females
- $\beta_0 + \beta_{0,\text{Male}}$: mean body mass for males
- β_1: effect of flipper length on body mass for females
- $\beta_1 + \beta_{1,\text{Male}}$: effect of flipper length on body mass for males

Since we are not interested in prediction but just on the parameters values, we can use the `_latent` suffix to get only the estimated parameters

```
# Here we use the _latent "trick" to recover the parameters
# here we do not need any new data to predict for, we can use an
# empty data frame

params = predict(fit2, data.frame() ,
   ~ data.frame( intercept_female = effects_latent[1],
                 intercept_male = effects_latent[1] + effects_latent[2],
                 slope_female = effects_latent[3] ,
                 slope_male = effects_latent[3] + effects_latent[4]))
```

Two things to notice here:

1. We have used a empty object as `newdata` in the `predict()` function.
2. The suffix `_latent` indicates that we are interested in the latent model effect.

Another way to fit this model is to realize that a fixed effect can be seen as an iid effect with fixed precision, one of the `inlabru` ways to fit the model is:

```
cmp = ~ -1 + sex_intercept(sex, model = "iid", initial = log(0.001), fixed = T) +
  sex_slope(sex, flipper_len,  model = "iid", fixed = T, initial = log(0.001))

lik = bru_obs(formula = body_mass ~ .,
              data = penguins)

fit2b = bru(cmp, lik)
```

Notice that we fix the precision of the `iid` effect to the same value as the precision of the linear effects which is 0.001.

The fitted values can be inspected as

```
fit2b$summary.random$sex_intercept
```

```
     ID      mean        sd 0.025quant  0.5quant 0.975quant      mode
1 female -5.442907 0.4399824  -6.306203 -5.442910  -4.579592 -5.442910
2   male -5.036399 0.3884275  -5.798541 -5.036401  -4.274245 -5.036401
          kld
1 5.835813e-10
2 5.837117e-10
```

```
fit2b$summary.random$sex_slope
```

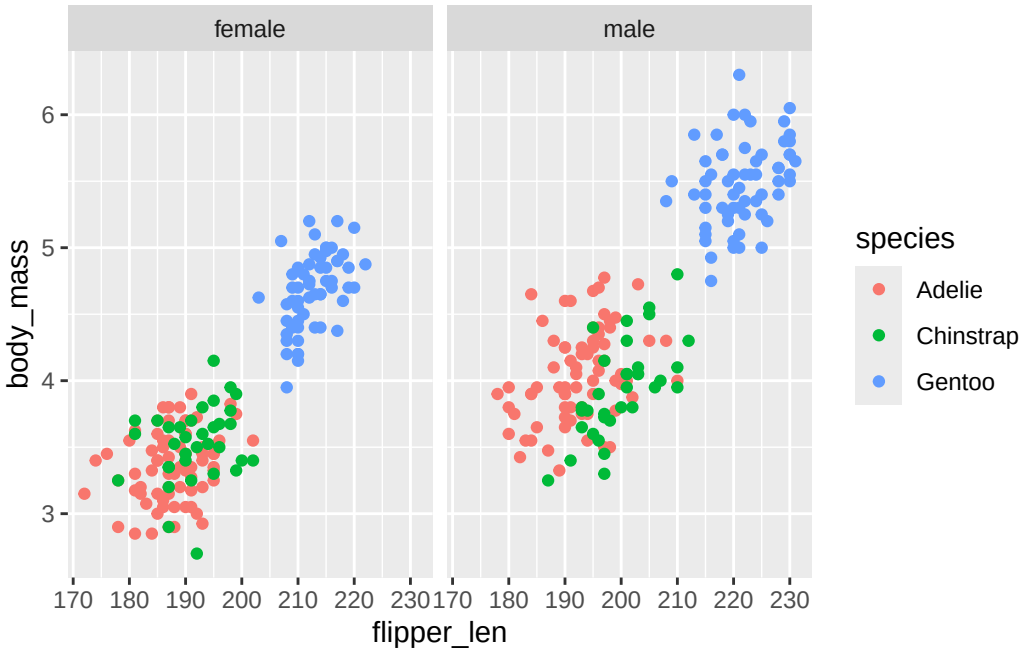
```
     ID       mean          sd 0.025quant   0.5quant 0.975quant       mode
1 female 0.04714741 0.002224868 0.04278187 0.04714742 0.05151286 0.04714742
2   male 0.04685481 0.001894587 0.04313734 0.04685482 0.05057222 0.04685482
          kld
1 5.835820e-10
2 5.837021e-10
```

The results from the `fit2b` model are not the same we got from the `fit2` model. What is happening? It is just a matter of parametrization. You can go from one parametrization to the other by using the `predict()` function as we saw earlier.

4 Linear Mixed Model

When looking at the data we see that there are differences in body mass depending on the species.

```
penguins %>%
  ggplot() +
  geom_point(aes(flipper_len, body_mass, color = species)) +
  facet_wrap(.~sex)
```



To account for this we introduce `species` as a random effect in the model.

Task

Modify the code below to include an iid random effect for species:

```
cmp = ~ -1 + effects( ~ sex*flipper_len, model = "fixed") +
  species(species, ... )
formula = ...
lik = bru_obs(formula = formula,
              data = penguins
              )
fit3 = bru(cmp, lik)
```

Click here to see the solution

```
  cmp = ~ -1 + effects( ~ sex*flipper_len, model = "fixed") +
    species(species, model = "iid")

  formula = body_mass ~ .
  lik = bru_obs(formula = formula,
                data = penguins
                )

  fit3 = bru(cmp, lik)
```

We now look at the results for the fixed effects and compare with those from the previous model

```
  fit3$summary.random$effects[,c(1,3,5)]
```

```
                  ID          sd     0.5quant
1         (Intercept) 0.727666803  0.351108936
2             sexmale 0.489339109 -0.351997930
3         flipper_len 0.003410093  0.017629028
4 sexmale:flipper_len 0.002449756  0.004398591
```

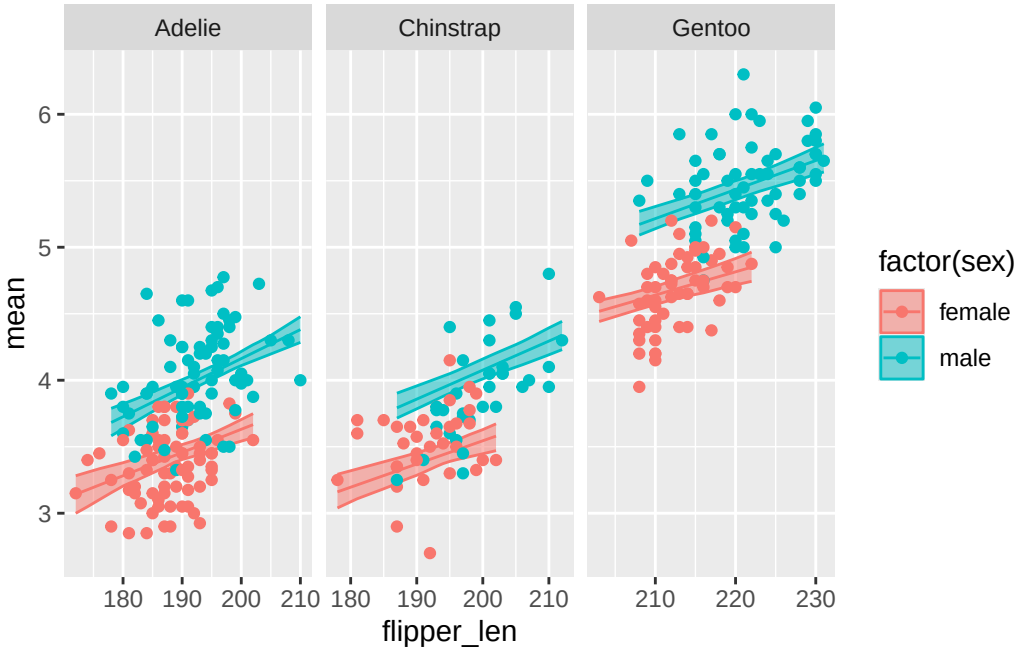
```
  fit2$summary.random$effects[,c(1,3,5)]
```

```
                  ID          sd      0.5quant
1         (Intercept) 0.439938837 -5.4428316937
2             sexmale 0.586824367  0.4056114601
3         flipper_len 0.002224648  0.0471470281
4 sexmale:flipper_len 0.002921826 -0.0002882235
```

We can also visualize the fitted values using the `predict` function as we did before:

```
  # Predict
  pred = predict(fit3, penguins, formula = ~ effects + species)

  pred %>%
    ggplot(aes(x=flipper_len,y=mean,color=factor(sex)))+
    geom_line()+
    geom_ribbon(aes(flipper_len,ymin = q0.025, ymax= q0.975,fill=factor(sex)), alpha = 0.5) +
    geom_point(data=penguins,aes(x=flipper_len,y=body_mass,colour=factor(sex)))+
    facet_wrap(~species,scales="free_x")
```



Task

We want to include a random effect, dependent on species, also on the slope. Modify the code below to include that random effect.

```
cmp = ~ -1 + effects( ~ sex*flipper_len, model = "fixed") +
  species1(species, ... ) +
  species2(species, ... )

formula = ...

lik = bru_obs(formula = formula,
              data = penguins
              )

fit4 = bru(cmp, lik)
```

Click here to see the solution

```
cmp = ~ -1 + effects( ~ sex*flipper_len, model = "fixed") +
  species1(species, model = "iid") +
  species2(species, flipper_len, model = "iid")

formula = body_mass ~ .

lik = bru_obs(formula = formula,
              data = penguins
              )

fit4 = bru(cmp, lik)
```

5 Model comparison

As a by-product of the main computations `inlabru` can compute some scores that can be useful to compare models (we will talk more about this later in the course).

The easiest scores are the Deviance Information Criteria (DIC) and the Wakaike Information Criteria (WAIC).

These scores take into account goodness-of-fit and a penalty term that is based on the complexity of the model via the estimated effective number of parameters.

The lower the value of the score, the better the model is rated.

```
deltaIC(fit1, fit2, fit3, fit4, criterion = c("DIC","WAIC"))
```

```
Warning in deltaIC(fit1, fit2, fit3, fit4, criterion = c("DIC", "WAIC")): WAIC
values from INLA are not well-defined for point process models. Use with
caution, or better, not at all.TRUE
```

```
  Model      DIC  Delta.DIC     WAIC Delta.WAIC
1  fit4 137.3337   0.000000 137.0981   0.000000
2  fit3 138.6548   1.321112 138.4345   1.336314
3  fit2 263.9099 126.576163 263.6026 126.504458
4  fit1 327.5693 190.235562 327.5294 190.431260
```