PRACTICAL

Aim of this practical:

1. Set priors for different linear models
2. Compute and visualize posterior densities and summaries for marginal effects
3. Calculate different model validation metrics
4. 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
- How to compare different models

0 Setting priors and model checking for Linear Models

In this excercise 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 postrerior distributions
- Perform model checks for linear models

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) INLA will assign a default $\mathcal{N}(0, 0)$ prior to it.

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

0.1.0.1 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)
```

0.1.0.2 Fitting the linear regression model with `inlabru` Now we fit a simple linear regression model in `inalbru` 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)
```

0.1.1 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.

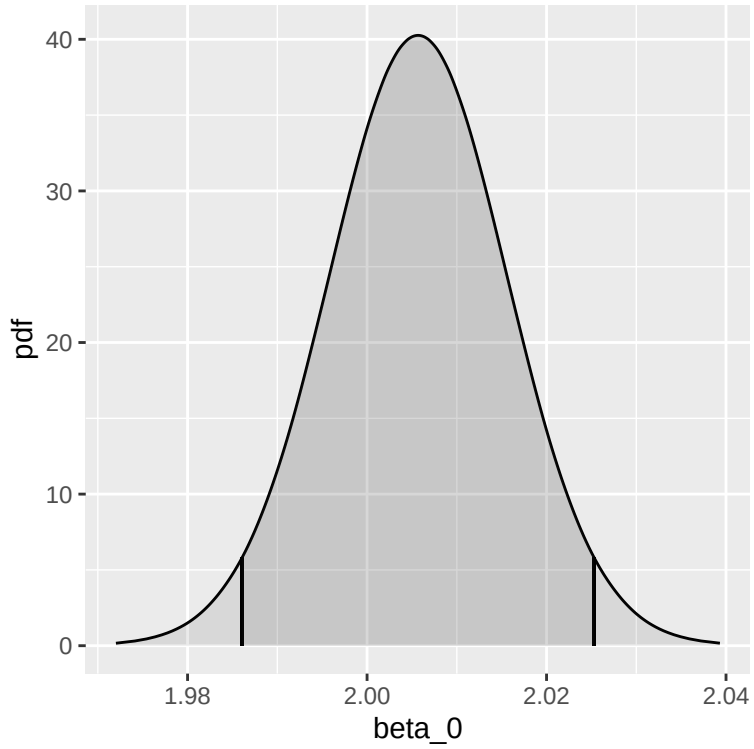
```
# First we define the logGamma (0.01,0.01) prior

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)`

0.1.2 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

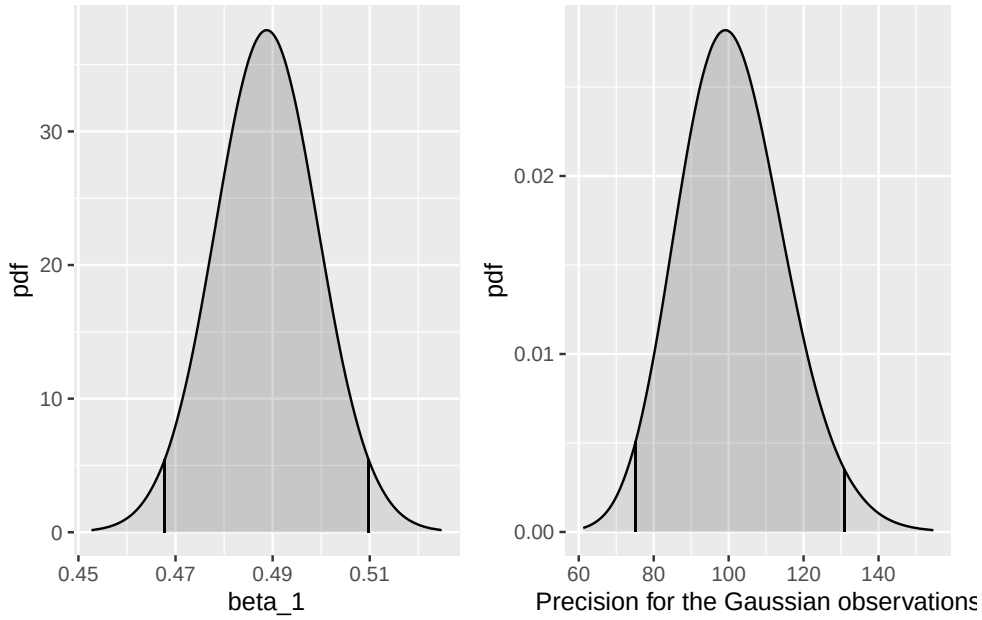
Plot the posterior marginals for β_1 and for the precision of the observation error $\pi(\tau|y)$

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")
```



0.1.3 Model Checking

A common way for model diagnostics in regression analysis is by checking residual plots. In a Bayesian setting residuals can be defined in multipleways depending on how you account for posterior uncertainty. Here, we will adopt a Bayesian approach by generating samples from the posterior distribution of the model parameters and then draw samples from the residuals defined as:

$$r_i = y_i - x_i^T \beta$$

We can use the `predict` function to achieve this:

```
res_samples <- predict(
  fit.lm,          # the fitted model
  df,              # the original data set
  ~ data.frame(
    res = y-(beta_0 + beta_1)  # compute the residuals
  ),
  n.samples = 1000   # draw 1000 samples
)
```

The resulting data frame contains the posterior draw of the residuals mean for which we can produce some diagnostics plots , e.g.

```
ggplot(res_samples,aes(y=mean,x=1:100))+geom_point() +
ggplot(res_samples,aes(y=mean,x=x))+geom_point()
```

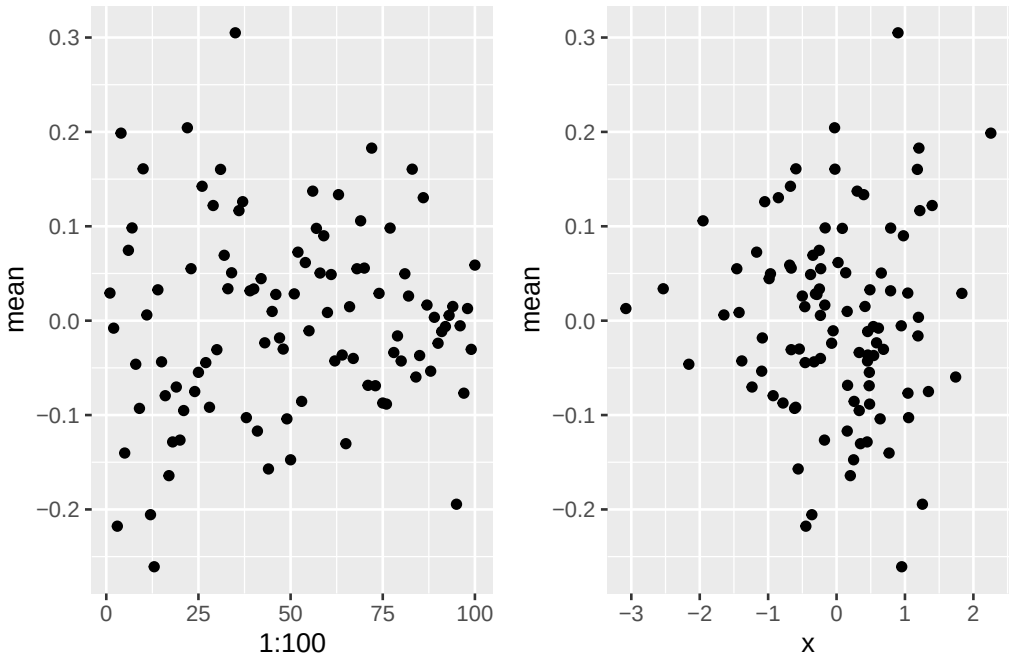
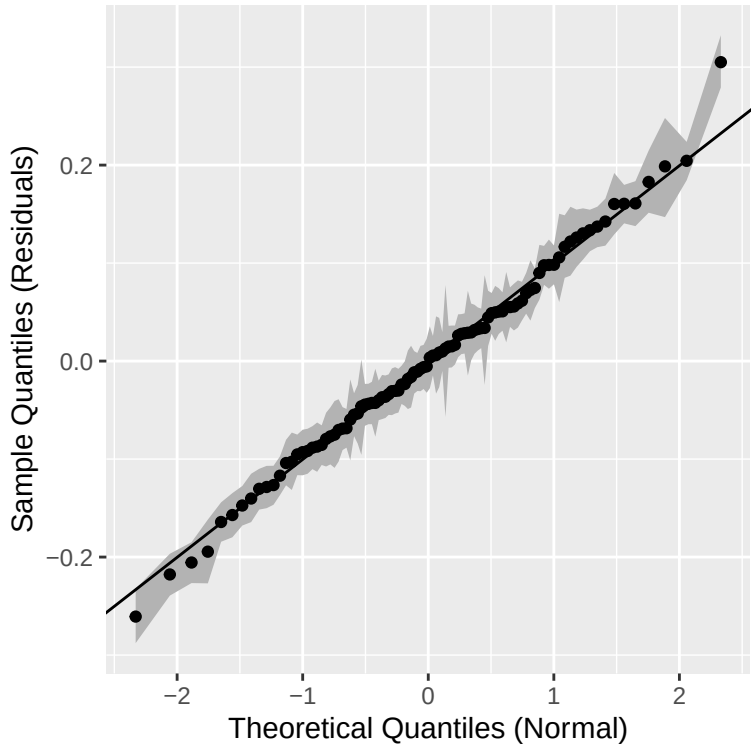


Figure 1: Bayesian residual plots: the left panel is the residual index plot; the right panel is the plot of the residual versus the covariate x

We can also compare these against the theoretical quantiles of the Normal distribution as follows:

```
arrange(res_samples, mean) %>%
  mutate(theortical_quantiles = qnorm(1:100 / (1+100))) %>%
  ggplot(aes(x=theortical_quantiles,y= mean)) +
  geom_ribbon(aes(ymin = q0.025, ymax = q0.975), fill = "grey70")+
  geom_abline(intercept = mean(res_samples$mean),
              slope = sd(res_samples$mean)) +
  geom_point() +
  labs(x = "Theoretical Quantiles (Normal)",
       y= "Sample Quantiles (Residuals)")
```



0 Linear Mixed Model for fish weight-length relationship

In this excercise 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.

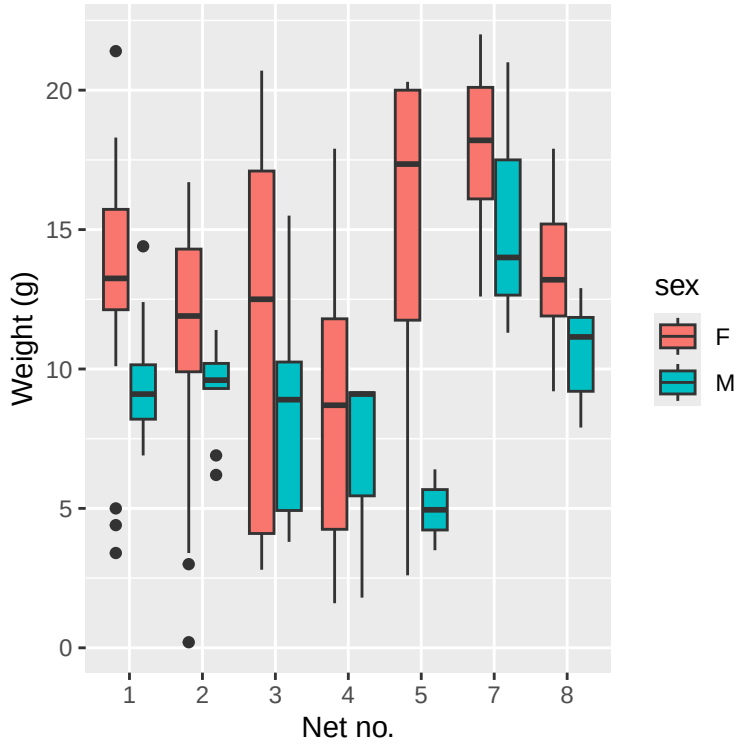
The data set contains the following information:

- `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:

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

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 =

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

fit = bru(cmp, lik)

summary(fit)
```

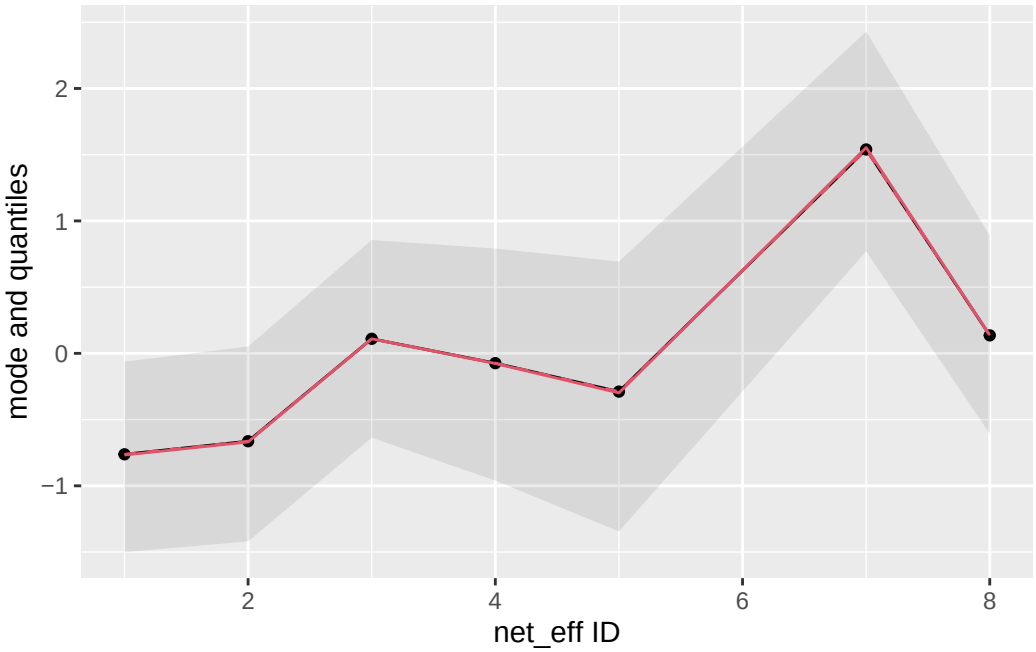
```
inlabru version: 2.12.0
INLA version: 25.06.22-1
Components:
sex_M: main = linear(sex_M), group = exchangeable(1L), replicate = iid(1L), NULL
beta_0: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
beta_1: main = linear(tl), group = exchangeable(1L), replicate = iid(1L), NULL
net_eff: main = iid(net_no), group = exchangeable(1L), replicate = iid(1L), NULL
Likelihoods:
  Family: 'gaussian'
    Tag: ''
    Data class: 'data.frame'
    Response class: 'numeric'
    Predictor: wt ~ .
    Used components: effects[sex_M, beta_0, beta_1, net_eff], latent[]
Time used:
    Pre = 0.368, Running = 0.322, Post = 0.192, Total = 0.881
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  11.501
beta_0 -15.816 0.870    -17.515  -15.819    -14.099 -15.818 145.165
beta_1   2.555 0.072      2.414    2.555      2.696   2.555 366.144

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      0.392    0.473
Precision for net_eff                   2.146 1.311      0.569    1.839
                                        0.975quant mode
Precision for the Gaussian observations      0.567 0.47
Precision for net_eff                        5.523 1.32

Deviance Information Criterion (DIC) ...............: 858.10
```

```
Deviance Information Criterion (DIC, saturated) ....: 247.57
Effective number of parameters .....................: 9.10

Watanabe-Akaike information criterion (WAIC) ...: 863.53
Effective number of parameters .................: 13.69

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 `R-INLA` functions to achieve this (see supplementary note).

We use the `inla.hyperpar.sample` 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.

```
sampvars <- 1/inla.hyperpar.sample(1000,fit,improve.marginals = T)

colnames(sampvars) <- c("Error variance","Between-net Variance")
```

```
apply(sampvars,2,
      function(x) c("mean"=mean(x),
                    "std.dev" = sd(x),
                    quantile(x,c(0.025,0.5,0.975))))
```

```
        Error variance Between-net Variance
mean         2.1251709             0.6491103
std.dev      0.1982571             0.4107745
2.5%         1.7681539             0.1836901
50%          2.1141260             0.5432609
97.5%        2.5427251             1.7305738
```

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 median, and quantiles for the ICC using the posterior samples we draw for σ_e^2 and σ_u^2.

Take hint

The `rowSums` function can be used to compute $\sigma_{u,s}^2 + \sigma_{e,s}^2$ for the sth posterior draw.

Click here to see the solution

```
sampicc <- sampvars[,2]/(rowSums(sampvars))
quantile(sampicc, c(0.025,0.5,0.975))
```

```
      2.5%        50%      97.5%
0.07756813 0.20382040 0.44778103
```

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 distributions.

```
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        2.124752  0.6484444
sd          0.1981695 0.4110607
quant0.025 1.766159   0.1815586
quant0.25   1.985458  0.3692471
quant0.5    2.113383  0.541605
quant0.75   2.251993  0.8051191
quant0.975 2.543882   1.736735
```

0 GLM model checking

In this exercise we will:

- Learn about some model assessments techniques available in INLA
- Conduct posterior predictive model checking

Libraries to load:

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

In this exercise, we will use data on horseshoe crabs (*Limulus polyphemus*) where the number of satellites males surrounding a breeding female are counted along with the female's color and carapace width.

A possible model to study the factors that affect the number of satellites the for female crabs is

$$y_i \sim \text{Poisson}(\mu_i), \qquad i = 1, \dots, N$$
$$\eta_i = \mu_i = \beta_0 + \beta_1 x_i + \dots$$

We can explore the conditional means and variances given the female's color:

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

# conditional means and variances
crabs %>%
  summarise( Mean = mean(satell ),
             Variance = var(satell),
                     .by = color)
```

```
   color     Mean  Variance
1 medium 3.294737 10.273908
2   dark 2.227273  6.737844
3  light 4.083333  9.719697
4 darker 2.045455 13.093074
```

The mean of the number of satellites vary by color which gives a good indication that color might be useful for predicting satellites numbers. However, notice that the mean is lower than its variance suggesting that overdispersion might be present and that a negative binomial model would be more appropriate for the data (we will cover this later).

Fitting the model

First, lets begin fitting the Poisson model above using the carapace's color and width as predictors. Since, color is a categorical variable in our model we need to create a dummy variable for it. We can use the `model.matrix` function to help us constructing the design matrix and then append this to our data:

```
crabs_df = model.matrix( ~  color , crabs) %>%
  as.data.frame() %>%
  select(-1) %>%         # drop intercept
  bind_cols(crabs) %>%  # append to original data
  select(-color)         # remove original color categorical variable
```

The new data set `crabs_df` contains a dummy variable for the different color categories (`dark` being the reference category). Then we can fit the model in `inlabru` as follows:

```
cmp =  ~ -1 + beta0(1) +  colordarker +
       colorlight + colormedium +
       w(weight, model = "linear")

lik =  bru_obs(formula = satell ~.,
            family = "poisson",
            data = crabs_df)

fit_pois = bru(cmp, lik)

summary(fit_pois)
```

```
inlabru version: 2.12.0
INLA version: 25.06.22-1
Components:
beta0: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
colordarker: main = linear(colordarker), group = exchangeable(1L), replicate = iid(1L), NULL
colorlight: main = linear(colorlight), group = exchangeable(1L), replicate = iid(1L), NULL
colormedium: main = linear(colormedium), group = exchangeable(1L), replicate = iid(1L), NULL
w: main = linear(weight), group = exchangeable(1L), replicate = iid(1L), NULL
Likelihoods:
  Family: 'poisson'
    Tag: ''
    Data class: 'data.frame'
    Response class: 'integer'
    Predictor: satell ~ .
```

```
    Used components: effects[beta0, colordarker, colorlight, colormedium, w], latent[]
Time used:
    Pre = 0.334, Running = 0.226, Post = 0.0675, Total = 0.627
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant   mode          kld
beta0       -0.501 0.196     -0.885   -0.501     -0.117 -0.501        7.726
colordarker -0.008 0.180     -0.362   -0.008      0.345 -0.008        7.184
colorlight   0.445 0.176      0.101    0.445      0.790  0.445        9.259
colormedium  0.248 0.118      0.017    0.248      0.479  0.248       18.674
w            0.001 0.000      0.000    0.001      0.001  0.001 53888561.050

Deviance Information Criterion (DIC) ...............: 917.12
Deviance Information Criterion (DIC, saturated) ....: 561.74
Effective number of parameters .....................: 5.01

Watanabe-Akaike information criterion (WAIC) ...: 929.70
Effective number of parameters .................: 16.51

Marginal log-Likelihood:  -489.43
 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)')
```

0.3.1 Model assessment and model choice

Now that we have fitted the model we would like to carry some model assessments. In a Bayesian setting, this is often based on posterior predictive checks. To do so, we will use the CPO and PIT - two commonly used Bayesian model assessment criteria based on the **posterior predictive distribution**.

> **Posterior predictive model checking**
>
> The posterior predictive distribution for predicted value $\hat{y}$ is
>
> $$\pi(\hat{y}|\mathbf{y}) = \int_{\theta} \pi(\hat{y}|\theta)\pi(\theta|\mathbf{y})d\theta.$$
>
> The probability integral transform (PIT) introduced by Dawid (1984) is defined for each observation as:
>
> $$\text{PIT}_i = \pi(\hat{y}_i \leq y_i|\mathbf{y}{-}i)$$
>
> The PIT evaluates how well a model's predicted values match the observed data distribution. It is computed as the cumulative distribution function (CDF) of the observed data evaluated at each predicted value. If the model is well-calibrated, the PIT values should be *approximately uniformly distributed*. Deviations from this uniform distribution may indicate issues with model calibration or overfitting.
> Another metric we could used to asses the model fit is the conditional predictive ordinate (CPO) introduced by Pettit (1990), and defined as:
>
> $$\text{CPO}_i = \pi(y_i|\mathbf{y}{-}i)$$
>
> The CPO measures the probability of the observed value of y_i when model is fit using all data but y_i. CPO provides a measure of how well the model predicts each individual observation while taking into account the rest of the data and the model. *Large*

> *values indicate a better fit* of the model to the data, while small values indicate a bad fitting of the model

To compute PIT and CPO we can either:

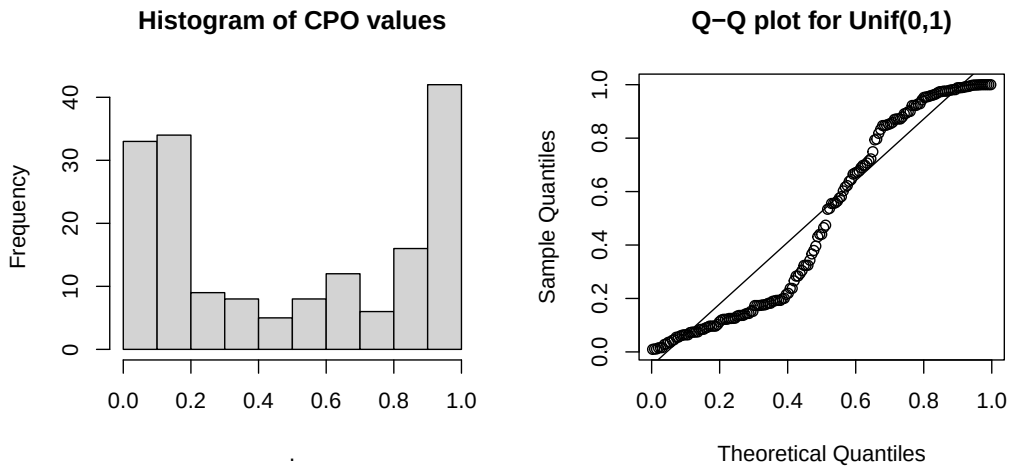
1. ask `inlabru` to compute them by set `options = list(control.compute = list(cpo = TRUE))` in the `bru()` function arguments.
2. set this as default in `inlabru` global option using the `bru_options_set` function.

Here we will do the later and re run the model

```
bru_options_set(control.compute = list(cpo = TRUE))

fit_pois = bru(cmp, lik)
```

Now we can produce histograms and QQ plots to assess for uniformity in the CPO values which can be accessed through `inlabru_model$cpo$pit` :

1 Plot



.

2 R Code

```
fit_pois$cpo$pit %>%
  hist(main = "Histogram of PIT values")

qqplot(qunif(ppoints(length(fit_pois$cpo$pit))),
       fit_pois$cpo$pit,
       main = "Q-Q plot for Unif(0,1)",
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles")

qqline(fit_pois$cpo$pit,
       distribution = function(p) qunif(p),
```

```
        prob = c(0.1, 0.9))
```

For the CPO values, usually the following summary of the CPO is often used:

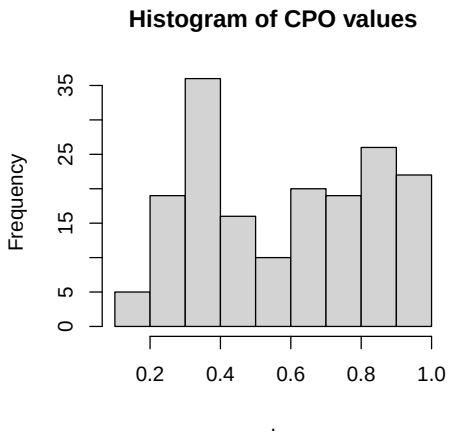
$$-\sum_{i=1}^{n} \log(\text{CPO\_}i)$$

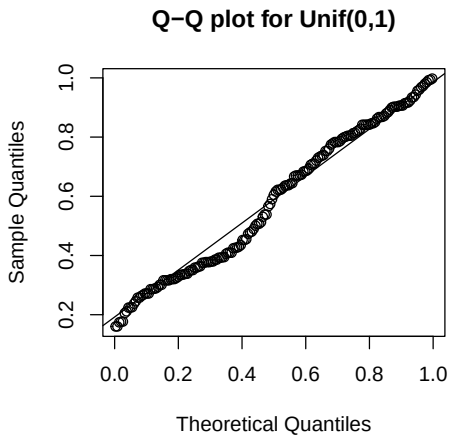
when comparing different models a smaller values indicate a better model fit. CPO values can be accessed by typing `inlabru_model$cpo$cpo`.

Task

The model assessment above suggests that a Poisson model might not be the most appropriate model, likely due to the overdispersion we detected previously. Fit a Negative binomial to relax the Poisson model assumption that the conditional mean and variance are equal. Then, compute the CPO summary statistic and PIT QQ plot to decide which model gives the better fit.
Take hint
To specify a negative binomial model you only need to change the family distribution to `family = "nbinomial"`.
Click here to see the solution

```
par(mfrow=c(1,2))

# Fit the negative binomial model

lik_nbinom =  bru_obs(formula = satell ~.,
            family = "nbinomial",
            data = crabs_df)

fit_nbinom = bru(cmp, lik_nbinom)

# PIT checks

fit_nbinom$cpo$pit %>%
  hist(main = "Histogram of CPO values")

qqplot(qunif(ppoints(length(fit_nbinom$cpo$pit))),
       fit_nbinom$cpo$pit,
       main = "Q-Q plot for Unif(0,1)",
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles")

qqline(fit_nbinom$cpo$pit,
       distribution = function(p) qunif(p),
       prob = c(0.1, 0.9))
```






```
# CPO comparison

data.frame( CPO = c(-sum(log(fit_pois$cpo$cpo)),
                    -sum(log(fit_nbinom$cpo$cpo))),
            Model = c("Poisson","Negative Binomial"))
```

```
       CPO              Model
1 465.4061            Poisson
2 379.3340 Negative Binomial
```

Overall, we can see that the negative binomial model provides a better fit to the data.

2 Hierarchical generalised additive mixed models with `inlabru`

In this excercise we will:

- Fit an hierarchical generalised additive mixed models
- Fit a model with a global smooth term
- Fit a model with global and group-level smooth terms

Libraries to load:

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

The oceans represent Earth's largest habitat, with life distributed unevenly across depths primarily due to variations in light, temperature, and pressure. Biomass generally decreases with depth, though complex factors like water density layers create non-linear patterns. A significant portion of deep-sea organisms exhibit bioluminescence, which scientists measure using specialized equipment like free-fall camera systems to profile vertical distribution.

In this exercise, we analyze the `ISIT` dataset, which contains bioluminescence measurements from the northeast Atlantic Ocean. This dataset was previously examined in Zuur et

al. (2009) and Gillibrand et al. (2007) and consists of observations collected across a depth gradient (0–4,800 m) during spring and summer cruises in 2001–2002 using an ISIT free-fall profiler.

The focus of this excersice will be on characterizing seasonal variation in the relationship between bioluminescent source density (sources m^2) and depth. We begin by exploring distribution patterns of pelagic bioluminescence through source-depth profiles, with each profile representing measurements from an individual sampling station. These profiles will be grouped by month to examine temporal patterns in the water column's bioluminescent structure.

3 Plot

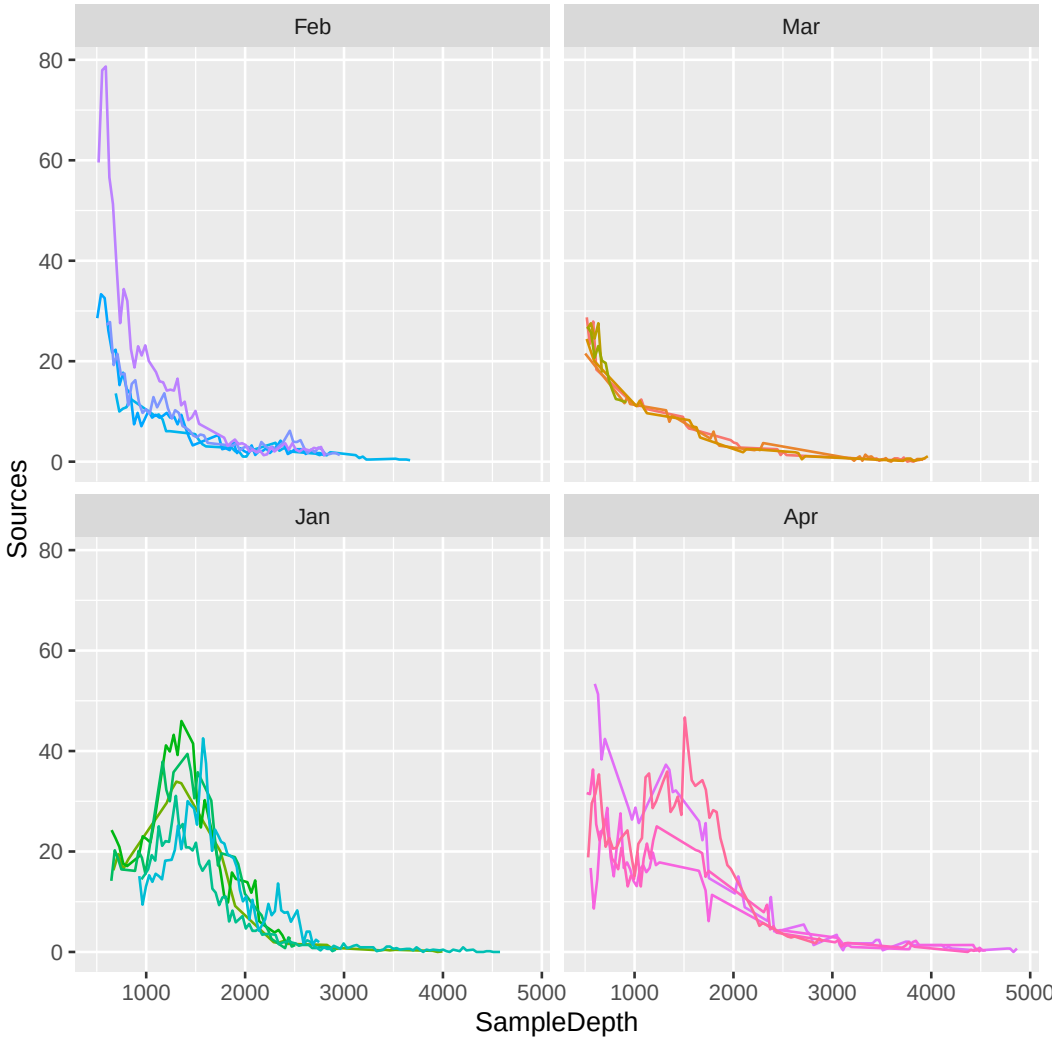


Figure 2: Source–depth profiles per month. Each line represents a station.

4 R-Code

```
icit <- read.csv("datasets/ISIT.csv")

icit$Month <- as.factor(icit$Month)
levels(icit$Month) <- month.abb[unique(icit$Month)]

ggplot(icit,aes(x=SampleDepth,y= Sources,
                group=as.factor(Station),
                colour=as.factor(Station)))+
  geom_line()+
  facet_wrap(~Month)+
  theme(legend.position = "none")
```

As expected, there seems to be a non-linear depth effect with some important variability across months.

4.0.1 Fitting a global smoother

We could begin analysing these data with a global smoother and a random intercept for each month. Thus, a possible model is of the form:

$$S_{is} = \beta_0 + f(\text{Depth})_s + \text{Month}_i + \epsilon_{is} \text{ such that } \epsilon \sim \mathcal{N}(0, \sigma_e^2);\ \text{Month} \sim \text{N}(0, \sigma_m^2).$$

where the source during month i at depth s, S_{is}, are modelled as smoothing function of depth and a month effect. The model has one smoothing curve for all months and can be fitted in `inlabru` as follows:

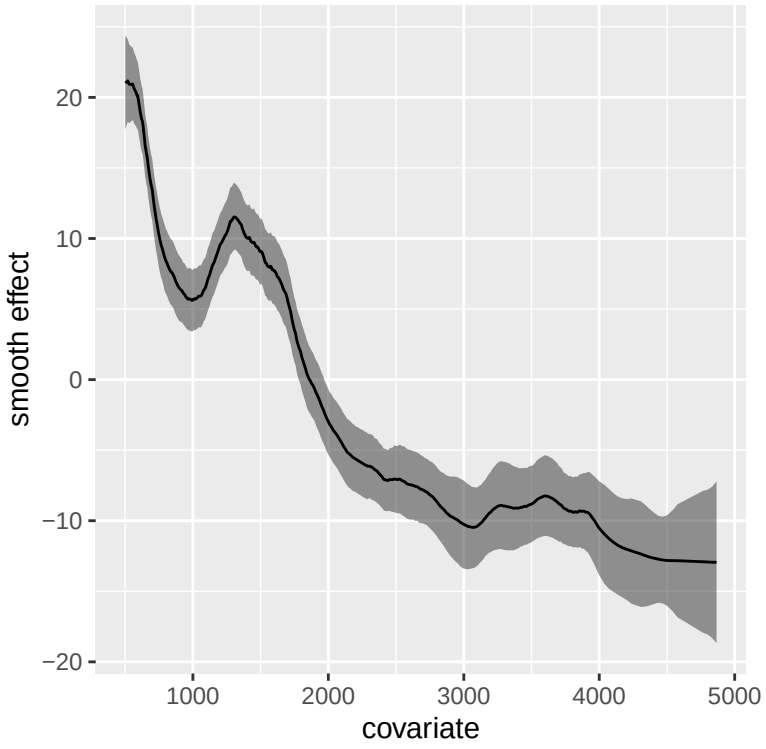
```
icit$Month_id <- as.numeric(icit$Month) # numeric index for the i-th month

cmp_g =  ~ -1+ beta_0(1) +
  smooth_g(SampleDepth, model = "rw1") +
  month_reff(Month_id, model = "iid")

lik =  bru_obs(formula = Sources ~.,
               family = "gaussian",
               data = icit)

fit_g = bru(cmp_g, lik)

summary(fit_g)
```

```
inlabru version: 2.12.0
INLA version: 25.06.22-1
Components:
beta_0: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
smooth_g: main = rw1(SampleDepth), group = exchangeable(1L), replicate = iid(1L), NULL
month_reff: main = iid(Month_id), group = exchangeable(1L), replicate = iid(1L), NULL
Likelihoods:
```

```
  Family: 'gaussian'
    Tag: ''
    Data class: 'data.frame'
    Response class: 'numeric'
    Predictor: Sources ~ .
    Used components: effects[beta_0, smooth_g, month_reff], latent[]
Time used:
    Pre = 0.33, Running = 0.643, Post = 0.173, Total = 1.15
Fixed effects:
         mean    sd 0.025quant 0.5quant 0.975quant   mode    kld
beta_0 10.017 1.615       6.68   10.026     13.299 10.024 34.949

Random effects:
  Name    Model
    smooth_g RW1 model
   month_reff IID model

Model hyperparameters:
                                          mean    sd 0.025quant 0.5quant
Precision for the Gaussian observations  0.024 0.001      0.021    0.024
Precision for smooth_g                  21.214 5.474     12.485   20.534
Precision for month_reff                 0.138 0.094      0.026    0.115
                                         0.975quant   mode
Precision for the Gaussian observations       0.026  0.024
Precision for smooth_g                       33.884 19.232
Precision for month_reff                      0.377  0.072

Deviance Information Criterion (DIC) ...............: 5233.13
Deviance Information Criterion (DIC, saturated) ....: 826.14
Effective number of parameters .....................: 34.43

Watanabe-Akaike information criterion (WAIC) ...: 5237.81
Effective number of parameters .................: 36.91

Marginal log-Likelihood:  -2217.23
CPO, PIT 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 plot the smoother marginal effect as follows:

```
data.frame(fit_g$summary.random$smooth_g) %>%
  ggplot() +
  geom_ribbon(aes(ID,ymin = X0.025quant, ymax= X0.975quant), alpha = 0.5) +
  geom_line(aes(ID,mean)) +
  xlab("covariate") + ylab("smooth effect")
```



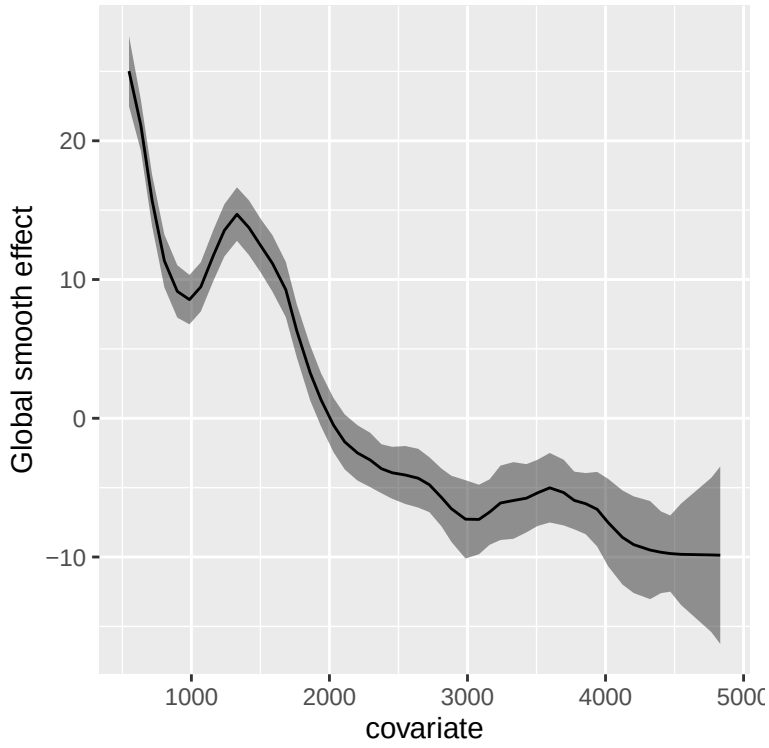
You might want to have a smoother function by placing a RW2 prior. Unfortunately, this assumes that all the knots are regularly spaced and some depth values are too close to be used for building the RW2 priors. For the case, it is possible to use function `inla.group()` to bin data into groups according to the values of the covariate:

```
icit$depth_grouped <- inla.group(icit$SampleDepth,n=50)
```

Task

Re-run the global smoother model using a RW2 prior for the depth smoother and compare your results with the RW1 model.
Take hint
Use the `depth_grouped` covariate to define the smoother.
Click here to see the solution

```
cmp_rw2 =  ~ -1+ beta_0(1) +
  smooth_g(depth_grouped, model = "rw2") +
  month_reff(Month_id, model = "iid")

lik =  bru_obs(formula = Sources ~.,
               family = "gaussian",
               data = icit)

fit_rw2 = bru(cmp_rw2, lik)

data.frame(fit_rw2$summary.random$smooth_g) %>%
  ggplot() +
  geom_ribbon(aes(ID,ymin = X0.025quant, ymax= X0.975quant), alpha = 0.5) +
  geom_line(aes(ID,mean)) +
  xlab("covariate") + ylab("Global smooth effect")
```



4.0.2 Fitting group-level smoothers

Here we fit a model where each month is allowed to have its own smoother for depth, i.e., $f_i(\text{Depth})_s$. The model structure is given by:

$$S_{is} = \beta_0 + f_i(\text{Depth})_s + \text{Month}_i + \epsilon_{is}.$$

Notice the only different between the global smoother model (Model G) and the group level model (Model GS) is the indexing of the smooth function for depth. We can fit a group-level smoother using the `group` argument within the model component as follows:

```
cmp_gs =  ~ -1+ beta_0(1) +
  smooth_g(SampleDepth, model = "rw1") +
  month_reff(Month_id, model = "iid")+
  smooth_loc(SampleDepth, model = "rw1", group = Month_id)
```

Then, we simply run the model (since the observational model has not changed -only the model components have):

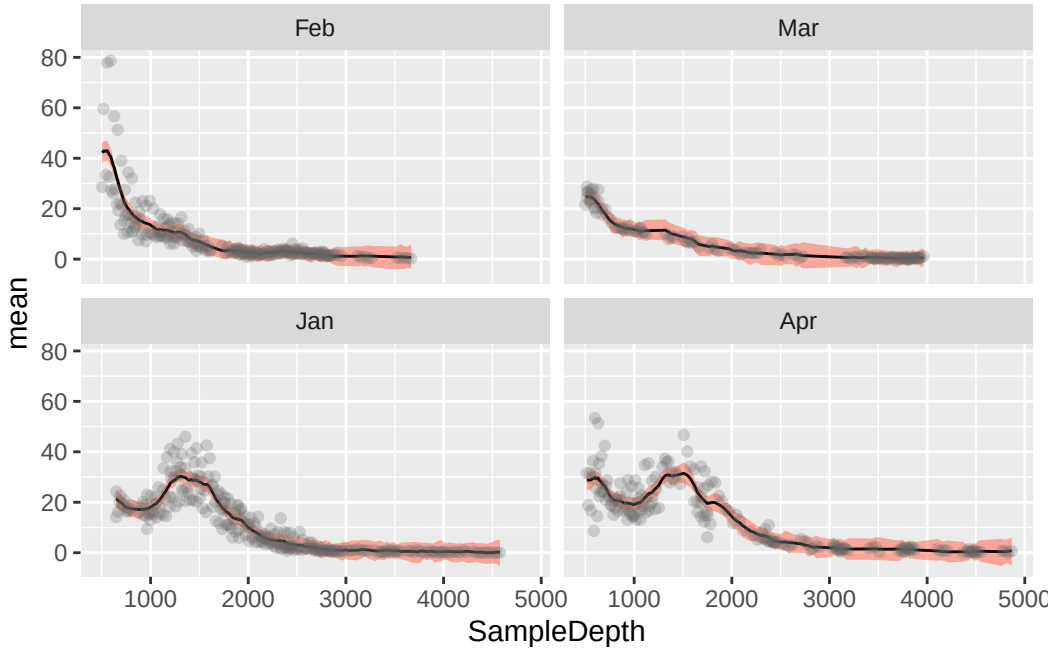
```
fit_gs = bru(cmp_gs, lik)
```

Lastly, we can generate model predictions using the `predict` function.

```
pred_gs = predict(fit_gs, icit, ~ (beta_0 + smooth_g+month_reff+smooth_loc))
```

Lastly, we plot the predicted mean values with their corresponding 95% CrIs.

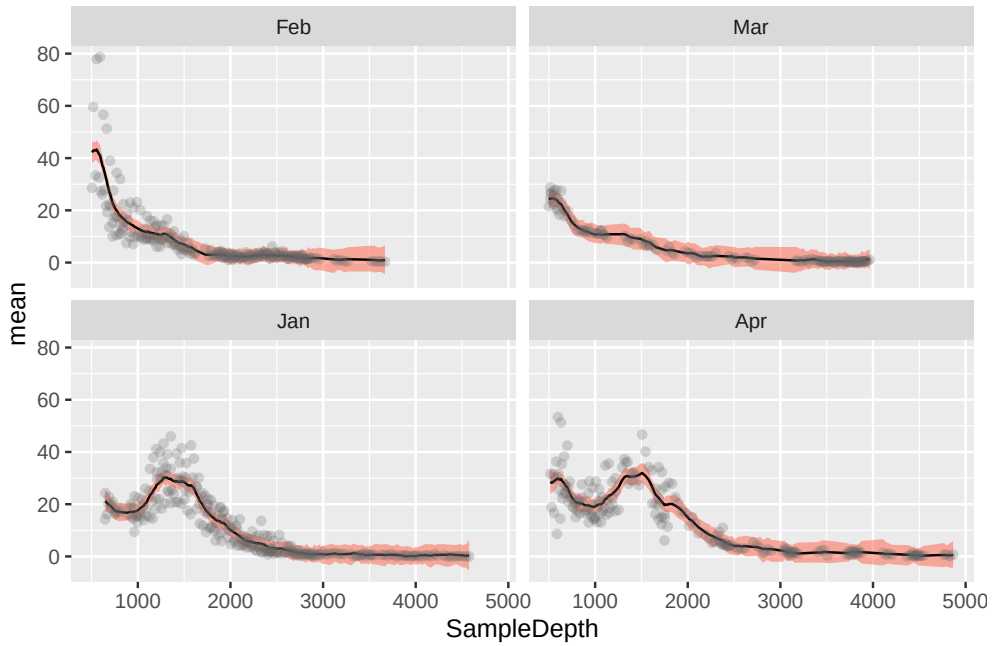
```
ggplot(pred_gs,aes(y=mean,x=SampleDepth))+
  geom_ribbon(aes(SampleDepth,ymin = q0.025, ymax= q0.975), alpha = 0.5,fill="tomato") +
  geom_line()+
  geom_point(aes(x=SampleDepth,y=Sources ),alpha=0.25,col="grey40")+
  facet_wrap(~Month)
```



Task

Re-fit the model GS without the global smoother. By omitting the global smoother, we do not longer force group-level smooths to follow a shared pattern, which is useful when groups may differ substantially from a common trend.
Take hint
You only need to modify the model components `cmp_gs`
Add hint details here…
Click here to see the solution

```
cmp_s =  ~ -1+ beta_0(1) +
  month_reff(Month_id, model = "iid")+
  smooth_loc(SampleDepth, model = "rw1", group = Month_id)

fit_s = bru(cmp_s, lik)

pred_s = predict(fit_s, icit, ~ (beta_0 +month_reff+smooth_loc))

ggplot(pred_s,aes(y=mean,x=SampleDepth))+
  geom_ribbon(aes(SampleDepth,ymin = q0.025, ymax= q0.975), alpha = 0.5,fill="tomato") +
  geom_line()+
  geom_point(aes(x=SampleDepth,y=Sources ),alpha=0.25,col="grey40")+
  facet_wrap(~Month)
```


Feb
Mar
Jan
Apr
mean
SampleDepth
0
20
40
60
80
1000
2000
3000
4000
5000