PRACTICAL 7 - MULTIPLE LIKELIHOODS MODELS

Aim of this practical:

In this first practical we are going to look multiple likelihood models

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

1 Multiple likelihood Models

Libraries to load:

```
library(dplyr)
library(INLA)
library(inlabru)
library(sf)
library(terra)
library(tidyverse)
library(fmesher)
library(tidyterra)

# load some libraries to generate nice map plots
library(scico)
library(ggplot2)
library(patchwork)
```

1 Simple simulated example

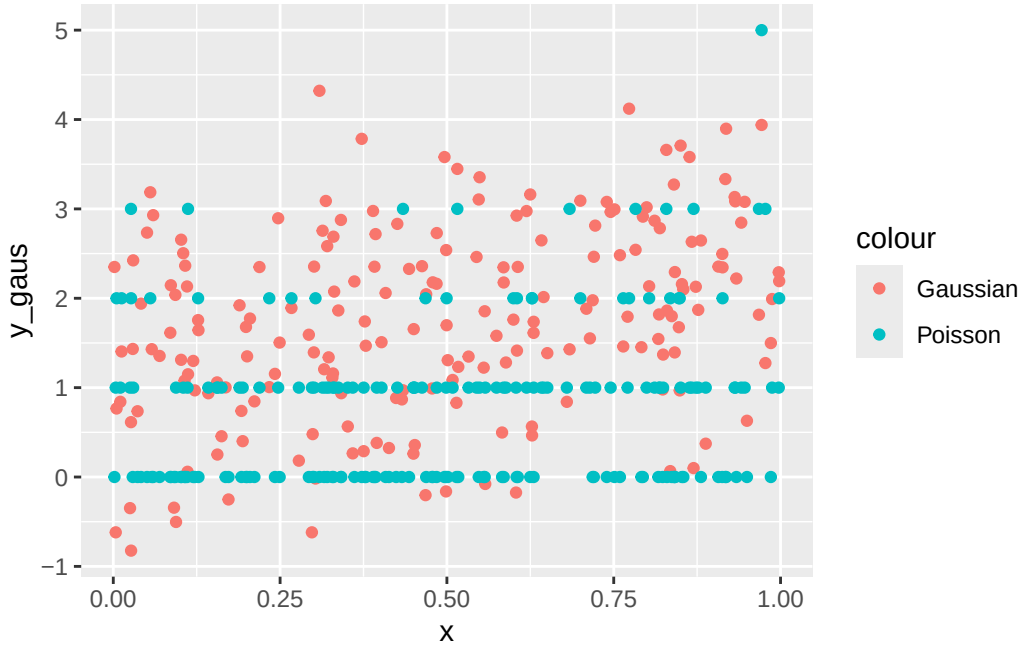
In this example we are going to fit a model where one covariate acts in the same way for two different responses

- Gaussian observations
- Poisson observations

We first define the unobserved latent field and then simulate both Gaussian and Poisson observations.

```
N = 200
x =  runif(N)
df = data.frame(idx = 1:N,
                x = x)

# simulate data
df = df %>%
  mutate(y_gaus = rnorm(N, mean = 1 + 1.5 * x), sd = 0.5) %>%
  mutate(y_pois = rpois(N, lambda  = exp( -1 + 1.5 * x)))

# plot the data
```

```
df %>% ggplot() +
  geom_point(aes(x, y_gaus, color = "Gaussian")) +
  geom_point(aes(x, y_pois, color = "Poisson"))
```



The model we aim to fit is

$$
\begin{aligned}
\eta_t &= \beta_0 + \beta_1 x_t,\ t = 1, \dots, T \\
y^{\text{Gaus}} &\sim \mathcal{N}(\mu_t, \sigma_y^2) \\
\eta_t^{\text{Gaus}} &= \mu_t \\
y_t^{\text{Pois}} &\sim \text{Poisson}(\lambda_t) \\
\eta_t^{\text{Pois}} &= \log(\lambda_t)
\end{aligned}
$$

We first define the components

```
cmp = ~ -1 +
  Intercept_gaus(1) +
  Intercept_pois(1) +
  covariate(x, model = "linear")
```

and the two likelihoods:

```
lik_gaus = bru_obs(formula = y_gaus ~ Intercept_gaus + covariate,
                   data = df)

lik_pois = bru_obs(formula = y_pois ~ Intercept_pois + covariate,
                   data = df,
                  family = "poisson")
```

Task

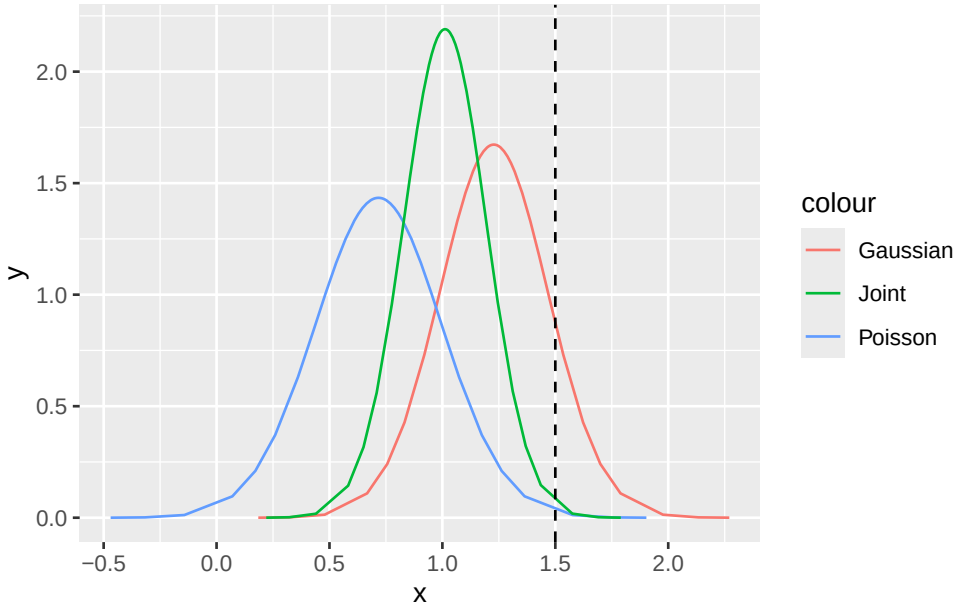
Fit three models using the components and likelihoods above:

- A model that only considers the Gaussian data

- A model that only considers the Poisson data
- A model that considers both data sets

Click here to see the solution

```
fit_gaus = bru(cmp, lik_gaus)

fit_pois = bru(cmp, lik_pois)

fit_join = bru(cmp, lik_gaus, lik_pois)


ggplot() +
  geom_line(data = fit_gaus$marginals.fixed$covariate, aes(x,y, color = "Gaussian"))+
  geom_line(data = fit_pois$marginals.fixed$covariate, aes(x,y, color = "Poisson"))+
  geom_line(data = fit_join$marginals.fixed$covariate, aes(x,y, color = "Joint"))+
  geom_vline(xintercept = 1.5, linetype = "dashed")
```



1 Modelling Pacific Cod Biomass Density

We will revisit the data on the Pacific Cod (*Gadus macrocephalus*) from a trawl survey in Queen Charlotte Sound. The `pcod` dataset is available from the `sdmTMB` package or can be downloaded by clicking the button below:

The dataset contains the presence/absence records of the Pacific Cod during each surveys along with the biomass density of Pacific cod in the area swept (kg/Km^2). The `qcs_grid` data contain the depth values stored as 2×2 km grid for Queen Charlotte Sound. The data contains presence/absence data from 2003 to 2017. In this practical we only consider year 2003.

We first load the dataset and select the year of interest

```r
load(here::here("datasets/pcod.RData"))

pcod_df = pcod_df %>% filter(year==2003)
pcod_sf =   st_as_sf(pcod_df, coords = c("lon","lat"), crs = 4326)
pcod_sf = st_transform(pcod_sf,
                       crs = "+proj=utm +zone=9 +datum=WGS84 +no_defs +type=crs +units=km" )

depth_r <- rast(qcs_grid, type = "xyz")
crs(depth_r) <- crs(pcod_sf)
```

We are no interested in modelling the biomass density (kg/km^2) of Pacific cod in the area swept for a given survey in 2003. However, We can see that there are an important amount of locations where fish were not caught , so we then have a dilemma:

- If we omit the zeros, we'll get a good, accurate model fit for non-zero data, but we'll be throwing away all the data with zeros
- If we include the zeros, we won't be throwing any data away, but we'll get a strange-fitting model that both under- and over-predicts values.
- So what do we do?

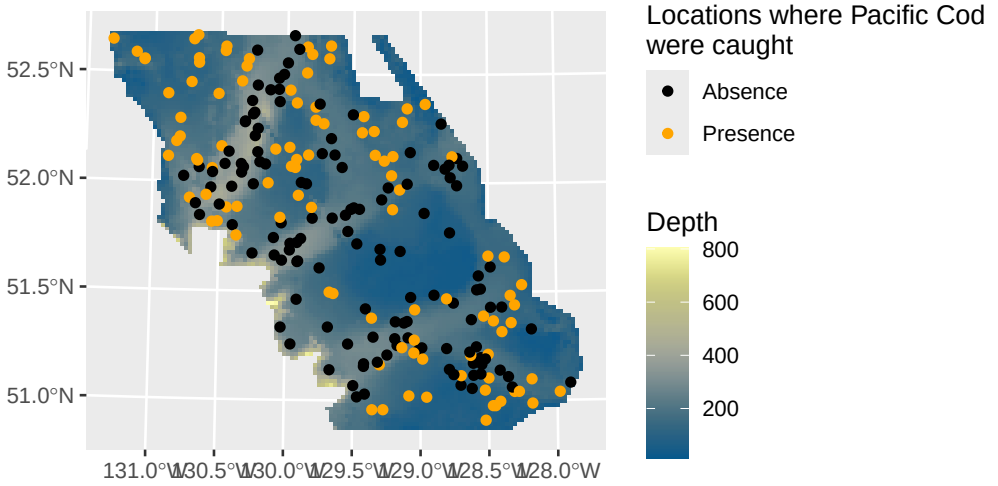


Figure 1: Map of the locations where Pacfic Cod were caught and the depth if the study area

1.2.1 A multilikelihood Hurdle Geostatistical Model

Here we present a multilikelihood approach to jointly model the the log-biomass density while accounting for the presence of zeros in our data. A two-part model can be constructed to accommodate zero-inflated continuous data by combining separate likelihoods: one for the occurrence (e.g., Bernoulli) and one for the conditional positive amount (e.g., log-normal). The primary advantage of this framework is the ability to model the probability of an event and its magnitude independently.

- **Stage 1** Model for the response(s)

$$y_i|\eta_i^{(1)} \sim \text{Binomial}(1, \pi_i)$$
$$\log(z_i)|\eta_i^{(2)} \sim \text{Normal}(\mu_i, \tau_e^{-1})$$

  - We then define a likelihood for each outcome.
    - $y_i = \begin{cases} 1 & \text{if fishes have been caught at location } \mathbf{s}_i \\ 0 & \text{otherwise} \end{cases}$
    - $z_i = \begin{cases} NA & \text{if no fish were caught at location } \mathbf{s}_i \\ \text{biomass density at location } \mathbf{s}_i & \text{otherwise} \end{cases}$

This structure is equivalent to a **Hurdle-log-Normal model**, where the overall expected value of log biomass is given by the product $\pi_i * \mu_i$, with μ_i representing the conditional expectation from the log-normal component.

Next we define the components of our linear predictor. Notice how we are defining this model in a LGM framework:

- **Stage 2** Latent field model

$$\eta_i^{(1)} = \text{logit}(\pi_i) = X'\beta + \xi_i$$
$$\eta_i^{(2)} = \mu_i = X'\alpha + \omega_i$$

  - $\{\alpha, \beta\}$ = Intercepts + covariate effects.
  - $\{\xi, \omega\}$ = are the Gaussian fields with Matérn covariance (separate for each outcome).

For the occurrence of fish, the linear predictor gets mapped to the logit of the probability of the Bernoulli model while the linear predictor for the biomass density is mapped to the mean of a log normal distribution.

- **Stage 3** Hyperparameters

The hyperparameter for the model are:

- observational error (nugget) τ_e
- Matérn field(s) parameters $\{\rho^{(1)}, \rho^{(2)}, \tau_d^{(1)}, \tau_d^{(2)}\}$

1.2.2 Define the mesh and SPDE

```
mesh = fm_mesh_2d(loc = pcod_sf,            # Build the mesh
                  cutoff = 2,
                  max.edge = c(10,20),      # The largest allowed triangle edge length.
                  offset = c(5,50))         # The automatic extension distance


spde_model =  inla.spde2.pcmatern(mesh,
                                  prior.sigma = c(1, 0.5),
                                  prior.range = c(100, 0.5))
```

Now we define the model components:

```
cmp_hurdle <- ~
  Intercept_biomass(1) +
    depth_biomass(depth_scaled, model = "linear") +
    depth2_biomass(depth_scaled2, model = "linear") +
    space_biomass(geometry, model = spde_model) +
    Intercept_caught(1) +
    depth_caught(depth_scaled, model = "linear") +
    depth2_caught(depth_scaled2, model = "linear") +
    space_caught(geometry, model = spde_model)
```

Lastly we define the two likelihoods and run the model:

```
biomass_obs <- bru_obs(formula = density ~  Intercept_biomass + depth_biomass + depth2_bioma
      family = "lognormal",
      data = pcod_sf  %>% filter(density>0))

presence_obs <- bru_obs(formula = present ~ Intercept_caught + depth_caught + depth2_caught
                        space_caught,
  family = "binomial",
  data = pcod_sf,
)

fit_hurdle <- bru(
  cmp_hurdle,
  biomass_obs,
  presence_obs
)
```

We can print a summary of the fixed effects and hyperparameters using the `tidy` function

```
tidy(fit_hurdle)
```

```
# A tibble: 6 x 5
  term              estimate std.error conf.low conf.high
  <chr>                <dbl>     <dbl>    <dbl>     <dbl>
1 Intercept_biomass    3.50      0.332    2.89      4.23
2 depth_biomass       -0.509     0.294   -1.10      0.0583
3 depth2_biomass      -0.339     0.228   -0.789     0.106
```

```
4 Intercept_caught     1.74      2.03    -2.36     6.04
5 depth_caught        -2.61      0.551   -3.82    -1.68
6 depth2_caught       -1.52      0.340   -2.26    -0.935
```

```
tidy(fit_hurdle,"hyperpar")
```

```
# A tibble: 5 x 5
  term                                        estimate std.error conf.low conf.high
  <chr>                                          <dbl>     <dbl>    <dbl>     <dbl>
1 Precision for the lognormal observations       1.16      0.558    0.464      2.59
2 Range for space_biomass                       36.6      25.9      6.84     104.
3 Stdev for space_biomass                        1.00      0.280    0.571      1.66
4 Range for space_caught                       158.       90.8     57.2      398.
5 Stdev for space_caught                         2.20      0.656    1.19       3.75
```

1.2.3 A model with shared spatial component

Note that in the hurdle model we fitted there is no direct link between the parameters of the two likelihoods. However, the two likelihoods could share some of the components; for example the Matérn field could be used for both predictors. Thus, we will fit a model that estimates this field jointly and compare it with our previous model

```
cmp_joint <- ~
  Intercept_biomass(1) +
    depth_biomass(depth_scaled, model = "linear") +
    depth2_biomass(depth_scaled2, model = "linear") +
    Intercept_caught(1) +
    depth_caught(depth_scaled, model = "linear") +
    depth2_caught(depth_scaled2, model = "linear") +
    space(geometry, model = spde_model) +
    space_copy(geometry, copy = "space", fixed = FALSE)
```

Notice that we have replaced the component `space_caught` with `space_copy`, which is effectively telling `inlabru` to reuse the same spatial Matérn field estimated in `space`, rather than fitting a second, independent field for the "*caught*" likelihood. The `copy = "space"` argument points to the shared field, so both linear predictors draw on one common spatial structure, i.e. the *biomass predictor* uses `space` directly, and the *caught predictor* uses a scaled version of it through the copy feature.

We set `fixed = FALSE` because we want to estimate a scaling parameter λ that links the two fields, i.e., the copied field enters the second predictor as $\lambda \times$ `space`, rather than as an exact duplicate. With `fixed = FALSE`, `inlabru` treats λ as an unknown to be inferred from the data, allowing the shared spatial pattern to contribute with a different magnitude in the two likelihoods. Had we set `fixed = TRUE`, λ would be held at 1 and the field would be copied identically into both predictors, forcing the spatial effect to have exactly the same strength in each data source.

Now we define each likelihood as follows:

```
biomass_obs <- bru_obs(formula = density ~  Intercept_biomass + depth_biomass + depth2_bioma
      family = "lognormal",
      data = pcod_sf  %>% filter(density>0))
```

```
presence_obs <- bru_obs(formula = present ~ Intercept_caught + depth_caught + depth2_caught
  family = "binomial",
  data = pcod_sf,
)
```

Notice that the *biomass likelihood* only uses the locations where fish were caught (en hence `pcod_sf %>% filter(density>0)`). No we can run the model by combining the model components and the two likelihoods:

```
fit_hurdle_shared <- bru(
  cmp_joint,
  biomass_obs,
  presence_obs
)
```

Task

Use the `glance` function to compare the two models we just fitted
Click here to see the solution

```
  bind_rows(
    glance(fit_hurdle_shared) |> mutate(model = "Shared Field"),
    glance(fit_hurdle) |> mutate(model = "Separate Fields")
  ) |>
    select(model, dic, waic, marginal_loglik, nobs)
```

```
# A tibble: 2 x 5
  model             dic  waic marginal_loglik  nobs
  <chr>           <dbl> <dbl>           <dbl> <int>
1 Shared Field       NA    NA           -666.   333
2 Separate Fields    NA    NA           -661.   333
```

Task

Based on the model you chose form the previous task, predict Posterior mean for the catch probability $\pi(s)$ using the `predict` function and also compute the following conditional means to estimate the biomass density in the region:

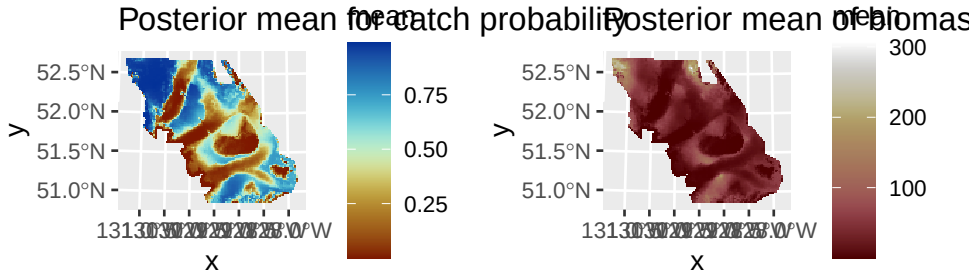
- $\mathbb{E}[Z(s)|Y(s)] = \exp\left(\mu(s) + \frac{1}{2\tau_e}\right)$
- $\mathbb{E}(Z(s)) = \pi(s) \times \mathbb{E}[Z(s)|Y(s)]$

Take hint
You can create a prediction grid using the `depth_r` raster and pass this on to the `predict()` function:

```
  pxl1 = data.frame(crds(depth_r),
                    as.data.frame(depth_r)) %>%
         filter(!is.na(depth)) %>%
  st_as_sf(coords = c("x","y"),crs=st_crs(pcod_sf))
```

Click here to see the solution

```
pred <- predict( fit_hurdle , pxl1,
  ~ {
    pi <- plogis(Intercept_caught + depth_caught + depth2_caught + space_caught)  # catchi
    mu_log  <-  Intercept_biomass + depth_biomass + depth2_biomass + space_biomass
    sd <- sqrt(1/Precision_for_the_lognormal_observations)
    conditional_mean <- exp(mu_log + 0.5 * sd^2)
    dens <- pi * conditional_mean # biomass density
    list(
      pi = pi,
      conditional_mean = conditional_mean,
      dens = dens)
  },n.samples = 2500)


ggplot() +
  gg(pred$pi, geom = "tile",aes(fill = mean)) +
  scale_fill_scico(palette = "roma") +
  ggtitle("Posterior mean for catch probability")+
ggplot() +
  gg(pred$dens, geom = "tile",aes(fill = mean))+
  scale_fill_scico()+
  ggtitle("Posterior mean of biomass density ")
```



1 Coregionalization model

In this exercise we present a way to fit the Bayesian coregionalization model similar to the one presented in Chapter 8 in Blangiardo and Cameletti (2015).

These models are often used when measurement stations record several variables; for example, a station measuring pollution may register values of CO2 and NO2. Instead of modelling these as several univariate datasets, the models we present in this section deal with the joint dependency structure. Dependencies among the different outcomes are modelled through shared components at the predictor level.

Usually, in coregionalization models, the different responses are assumed to be observed at the same locations. With the INLA-SPDE approach, we do not require the different outcome variables to be measured at the same locations. Hence, in the code example below we show how to model responses observed at different locations.

1.3.1 The model

We consider the following model

$$y_1(s) = \beta_1 + \omega_1(s) + e_1(s)$$
$$y_2(s) = \beta_2 + \lambda\,\omega_1(s) + \omega_2(s) + e_1(s),$$

where the

- β_k are intercepts, $k = 1, 2$.
- $\omega_1(s)$ is a Gaussian spatial effect that is common for both observations
- λ is a weight for the spatial effect $\omega_1(s)$.
- $\omega_2(s)$ is a Gaussian spatial effect specific to the second observations
- $e_k(s)$ are uncorrelated error terms, $k = 1, 2$.

1.3.2 Data simulation

We start by simulating the data. We first define a function to sample from a Matern RF with given range and sd.

```
rMatern <- function(n, coords, sigma=1, range,
                    kappa = sqrt(8*nu)/range,
                    variance = sigma^2,
                    nu=1) {
  m <- as.matrix(dist(coords))
  m <- exp((1-nu)*log(2) + nu*log(kappa*m)-
             lgamma(nu))*besselK(m*kappa, nu)
  diag(m) <- 1
  return(drop(crossprod(chol(variance*m),
                        matrix(rnorm(nrow(coords)*n), ncol=n))))
}
```

We then define the true values of all parameters

```
# Intercept on reparametrized model
beta <- c(-5, 3)
# Random field marginal variances for omega1 and omega2:
m.var <- c(0.5, 0.4)
# GRF range parameters for omega1 and omega2:
range <- c(4, 6)
# Copy parameters: reparameterization of coregionalization
# parameters
lambda <- c(0.7)
# Standard deviations of error terms
e.sd <- c(0.3, 0.2)
```

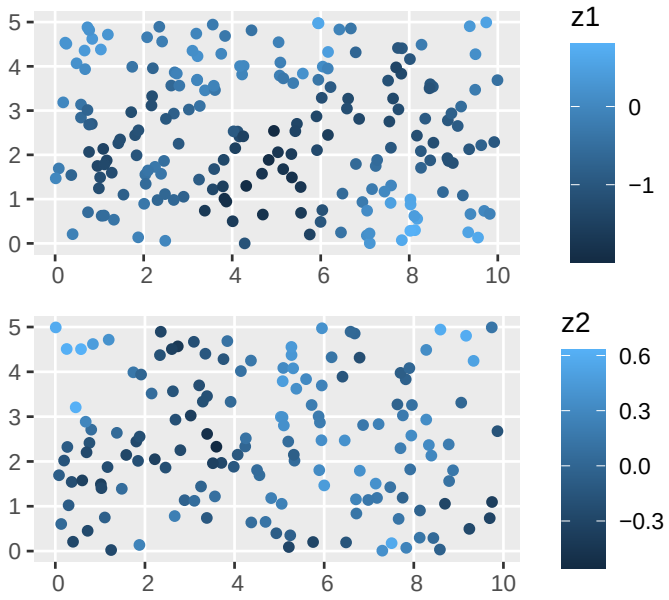
and simulate our data. We assume that we observe data in a window $(0:10) \times (0:5)$. We assume that in some locations we observe both y_1 and y_2 while in others we only observe one of them

```
# define the area of interest
poly_geom = st_polygon(list(cbind(c(0,10,10,0,0), c(0,0,5,5,0)) ))
# Wrap it in an sfc (simple feature collection)
poly_sfc <- st_sfc(poly_geom)
# Now create the sf object
border <- st_sf(id = 1, geometry = poly_sfc)



# how many observation we have
n1 <- 200
n2 <- 150
n_common = 50

# simulate observation locations

loc_common = st_sf(geometry = st_sample(border, n_common))
loc_only1 = st_sf(geometry = st_sample(border, n1-n_common))
loc_only2 = st_sf(geometry = st_sample(border, n2-n_common))



# simulate the two gaussian field at the locations
z1 <- rMatern(1, st_coordinates(rbind( loc_common,loc_only1, loc_only2)), range = range[1],
                  sigma = sqrt(m.var[1]))

z2 <- rMatern(1, st_coordinates(rbind(loc_common, loc_only2)), range = range[2],
                  sigma = sqrt(m.var[2]))


## Create data.frame
loc1 = rbind( loc_common, loc_only1)
loc2 = rbind( loc_common, loc_only2)

df1 =  loc1 %>% mutate(z1 = z1[1:n1])
df2 =  loc2 %>% mutate(z1 = z1[-c(1:(n1-n_common))], z2 =z2)


## create the linear predictors

df1  = df1 %>%
  mutate(eta1 = beta[1] + z1)

df2  = df2 %>%
  mutate(eta2 = beta[2] + lambda * z1 + z2)


# simulate data by addint the obervation noise
```

```
df1  = df1 %>%
  mutate(y = rnorm(n1, mean = eta1, sd = e.sd[1]))

df2  = df2 %>%
  mutate(y = rnorm(n2, mean = eta2, sd = e.sd[1]))
```

We can visualize the observations

```
p1 = ggplot(data = df1) + geom_sf(aes(color = z1))
p2 = ggplot(data = df2) + geom_sf(aes(color = z2))
p1+p2+plot_layout(ncol = 1)
```

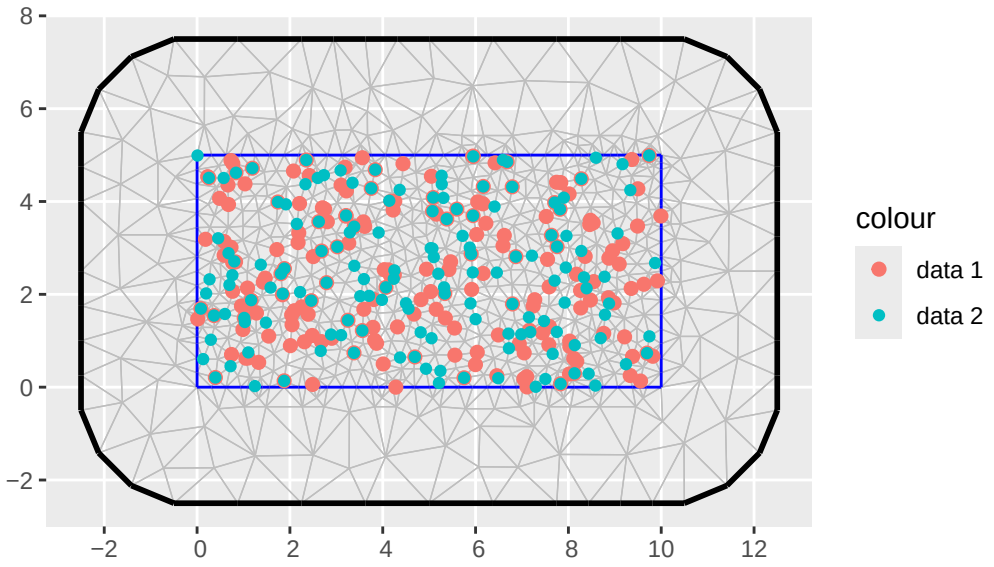


1.3.3 Mesh definition

We need to define a mesh. We use the location observations and the area of interest as a starting point.

```
mesh <-  fm_mesh_2d(loc = rbind(loc1, loc2),
                    boundary = border,
                      max.edge = c(0.5, 1.5),
                      offset = c(0.1, 2.5),
                      cutoff = 0.1)
```

Here is the mesh, together with the observations y_1 (red) and y_2 (black).



1.3.4 SPDE definition

Task

Define one `spde` object that contains information about priors for the range and the standard deviation.
Use the same priors for both z_1 and z_2 so we create only one `spde` object.

$$\text{Prob(range} < 0.5) = 0.01$$
$$\text{Prob(sd} > 1) = 0.01$$

Click here to see the solution

```
spde <- inla.spde2.pcmatern(
  mesh = mesh,
  prior.range = c(0.5, 0.01), # P(range < 0.5) = 0.01
  prior.sigma = c(1, 0.01)) # P(sigma > 1) = 0.01
```

1.3.5 Run the model

We now need to define the components of the model:

```
cmp = ~ -1 +  Intercept1(1) + Intercept2(1) +
  omega1(geometry, model = spde) +
  omega1_copy(geometry, copy = "omega1", fixed = FALSE) +
  omega2(geometry, model = spde)
```

Task

Run the joint model. To do this you need to

- Define the two likelihoods using the `bru_obs` function

- Fit the model using the `bru()` function

$$\text{Prob(range} < 0.5) = 0.01$$
$$\text{Prob(sd} > 1) = 0.01$$

Click here to see the solution

```
lik1 = bru_obs(formula = y ~ Intercept1 + omega1,
               family  = "gaussian",
               data = df1)

lik2 = bru_obs(formula = y ~ Intercept2 + omega1_copy + omega2,
               family = "gaussian",
               data = df2)


res = bru(cmp, lik1, lik2)
```

1.3.6 Model Results

Task

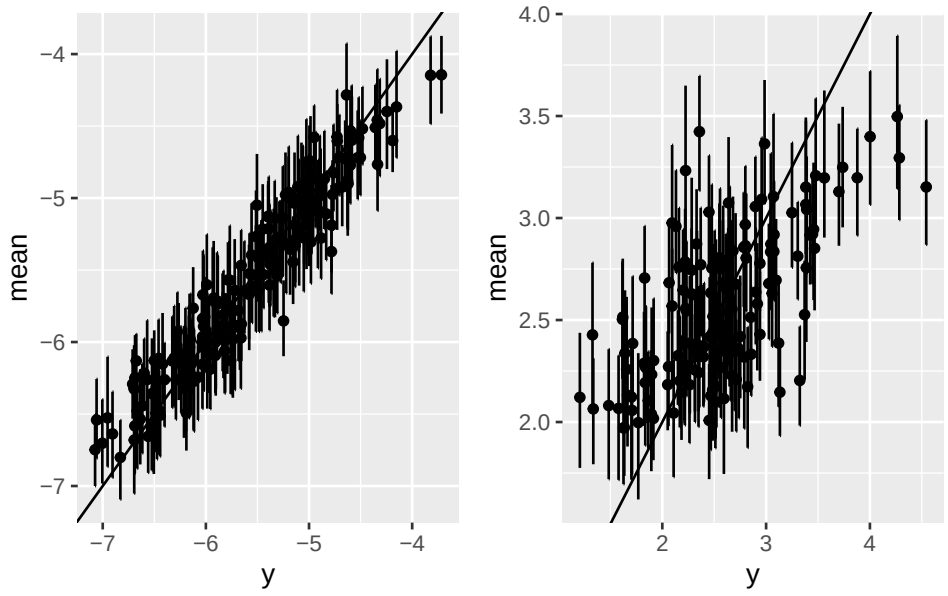
Check the model results and see that the model manages to recover the true value of the parameters.
Click here to see the solution

```
# fixed effects

#fixed effects
fixed = data.frame(true = beta, res$summary.fixed[,c(1,3,5)])
#hyperparameters
hyper = data.frame(true = c(1/e.sd^2, range[1], sqrt(m.var[1]),
                            range[2], sqrt(m.var[2]),
                      lambda),
             res$summary.hyperpar[,c(1,3,5)])
```

Task

Compute predictions from the model at the observation points and compare them with the observed values.
Click here to see the solution

```
pred1 = predict(res, df1, ~Intercept1 + omega1)
pred2 = predict(res, df2, ~Intercept2 + omega1_copy + omega2)

p1 = ggplot() + geom_point(data = pred1 , aes(y, mean)) +
  geom_errorbar(data = pred1 ,aes(y, ymin = q0.025, ymax = q0.975)) +
  geom_abline(intercept = 0, slope = 1)

p2 = ggplot() +   geom_point(data = pred2 , aes(y, mean)) +
  geom_errorbar(data = pred2 ,aes(y, ymin = q0.025, ymax = q0.975)) +
  geom_abline(intercept = 0, slope = 1)
p1+p2
```



Task

Compute predictions from the model over the area of interest. Plot the posterior mean and the posterior sd.
Click here to see the solution

```
pxl = fm_pixels(mesh, mask = border)
pred1 = predict(res, pxl, ~Intercept1 + omega1)
pred2 = predict(res, pxl, ~Intercept2 + omega1_copy + omega2 )


p1 = ggplot() + gg(pred1, aes(color = mean)) +
  ggtitle("Posterior mean for eta_1") +  xlab("") + ylab("")
p2 = ggplot() + gg(pred2, aes(color = mean)) +
  ggtitle("Posterior mean for eta_2")+  xlab("") + ylab("")

p3 = ggplot() + gg(pred1, aes(color = sd)) +
  ggtitle("Posterior sd for eta_1")+  xlab("") + ylab("")
p4 = ggplot() + gg(pred2, aes(color = sd)) +
  ggtitle("Posterior sd for eta_2")+ xlab("") + ylab("")
# p1 + p2 + p3 + p3
```

Task

Use the function `generate()` to create 4 simulations from $\tilde{\pi}(\omega_1(s)|y_1, y_2)$ and $\tilde{\pi}(\omega_2(s)|y_1, y_2)$

Click here to see the solution

```
samples = generate(res, pxl,
                   ~ data.frame(omega1 = omega1,
                                omega2 = omega2),
                   n.samples = 5)


omega1 = sapply(samples, function(x) x$omega1)
p1 = cbind(pxl,omega1) %>%
  pivot_longer(-geometry) %>% ggplot() +
  geom_sf(aes(color =value)) + facet_wrap(.~name) + ggtitle("Omega 1")

omega2 = sapply(samples, function(x) x$omega2)
p2 = cbind(pxl,omega2) %>%
  pivot_longer(-geometry) %>% ggplot() +
  geom_sf(aes(color =value)) + facet_wrap(.~name) + ggtitle("Omega 2")
```