PRACTICAL 7

Aim of this practical:

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

---

Libraries to load:

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

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

1 Multiple likelihood models

1 Simple simulated example

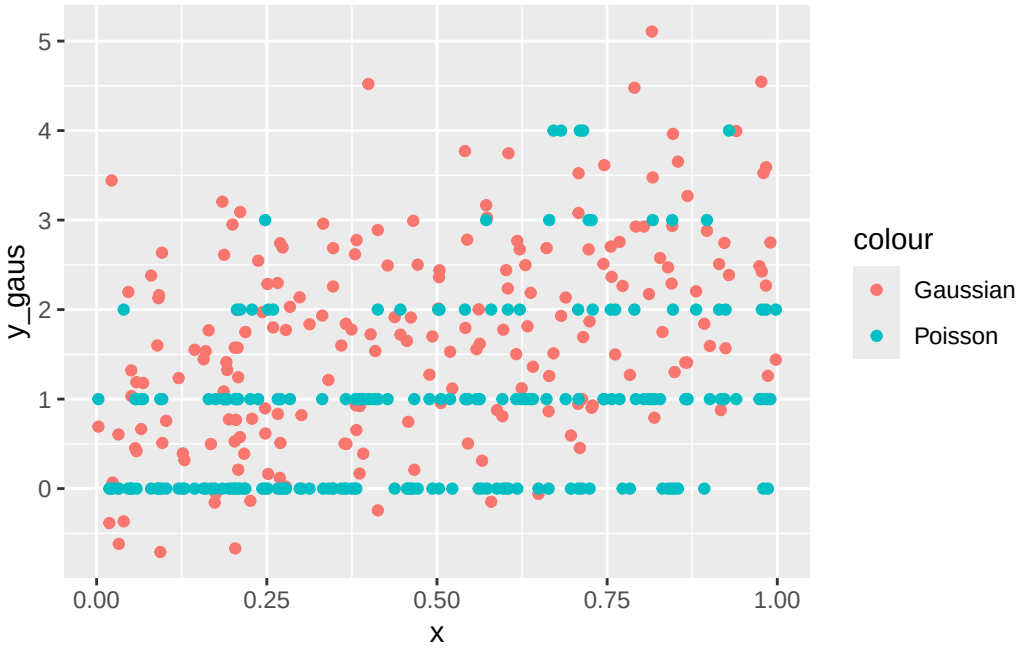
In this example we are going to fit a model where one covariate acts in the same way for two different responses

- 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")
```

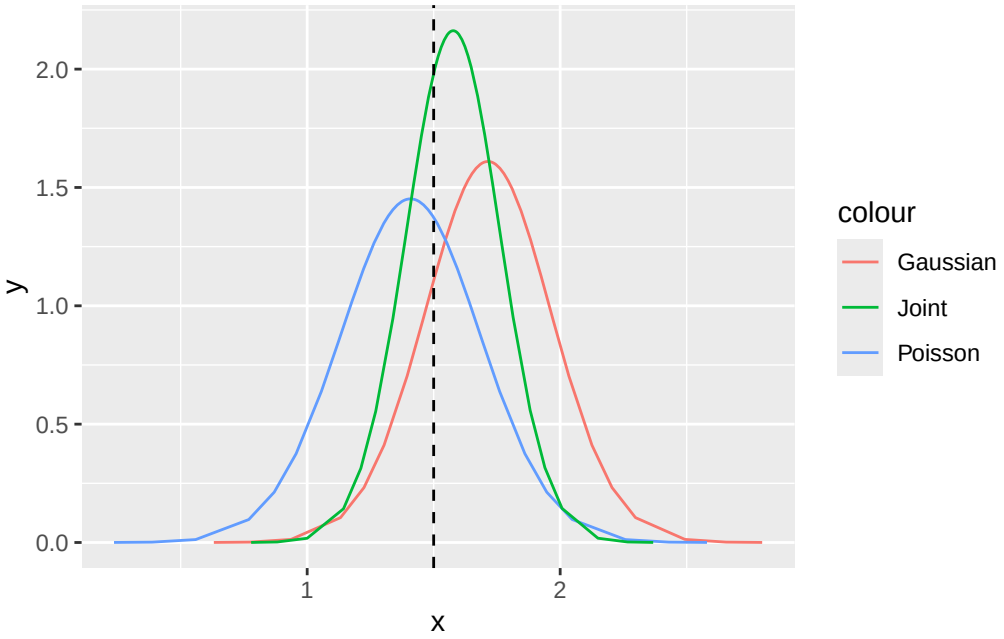
Task

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")
```



2 Coregionalization model

In this practical 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.

2 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, $k = 1, 2$.

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)
```

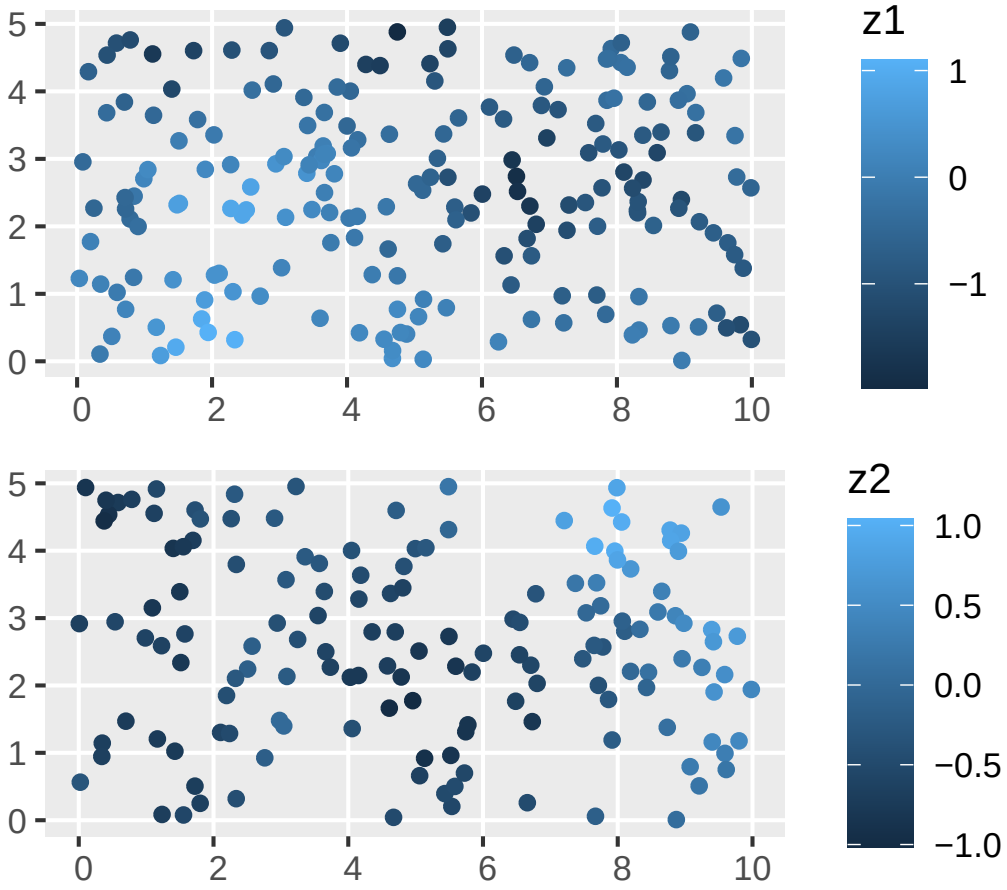
and simulate our data. We assume that we observe data in a window $(0:10) \times (0:5)$. We assume that in some locations we observe both y_1 and y_2 while in others we only observe one of them

```r
# 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)
```

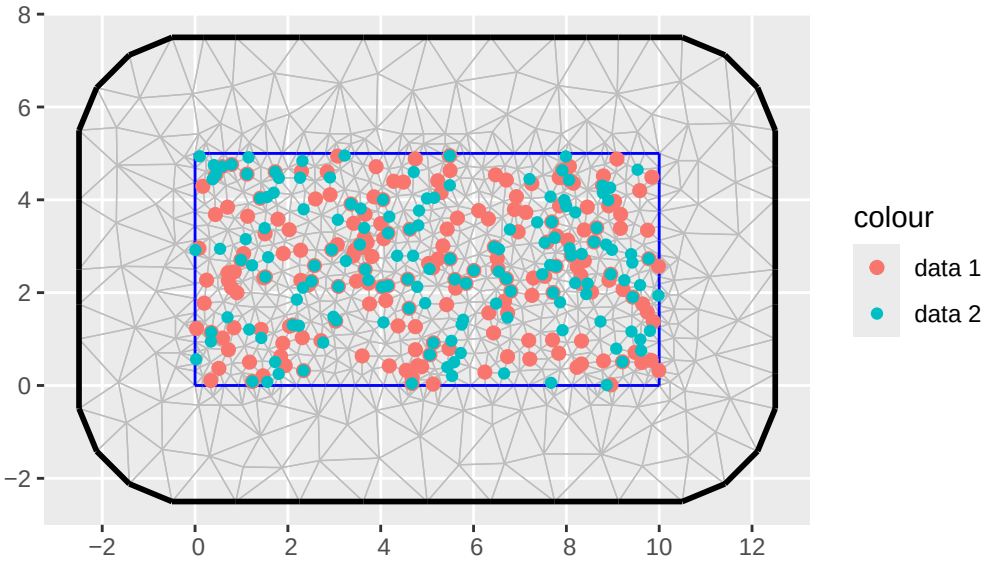


2 Mesh definition

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).



2 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
```

2 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)
```

2 Model Results

Task

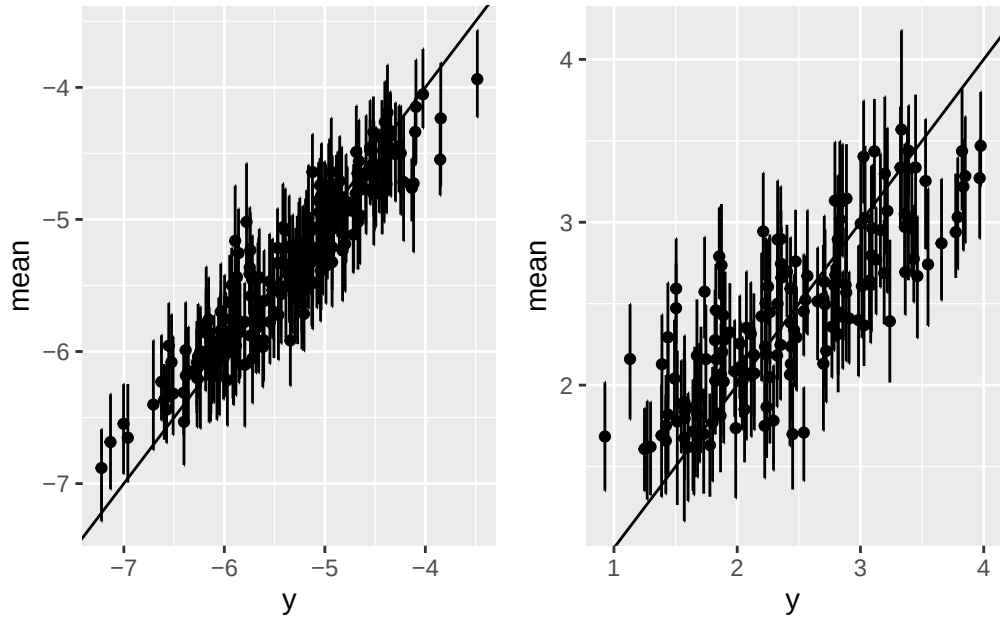
Check the model results and see that the model manages to recover the true value of the parameters.
Click here to see the solution

```
# 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")
```