PRACTICAL 6

Aim of this practical:

In this practical we are going to look at some model comparison and validation techniques.

0 Model Checking for Linear Models

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

Recall a simple linear regression model with Gaussian observations

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

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.

0.1.1 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)
```

0.1.2 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)
```

0.1.3 Residuals analysis

A common way for model diagnostics in regression analysis is by checking residual plots. In a Bayesian setting residuals can be defined in multiple ways 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()
```

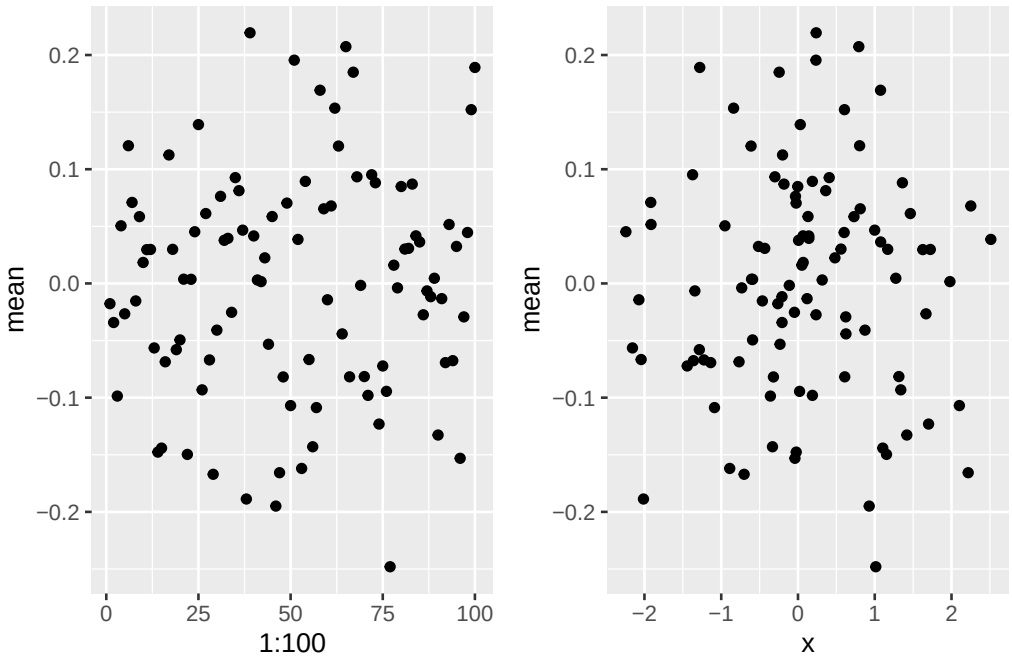
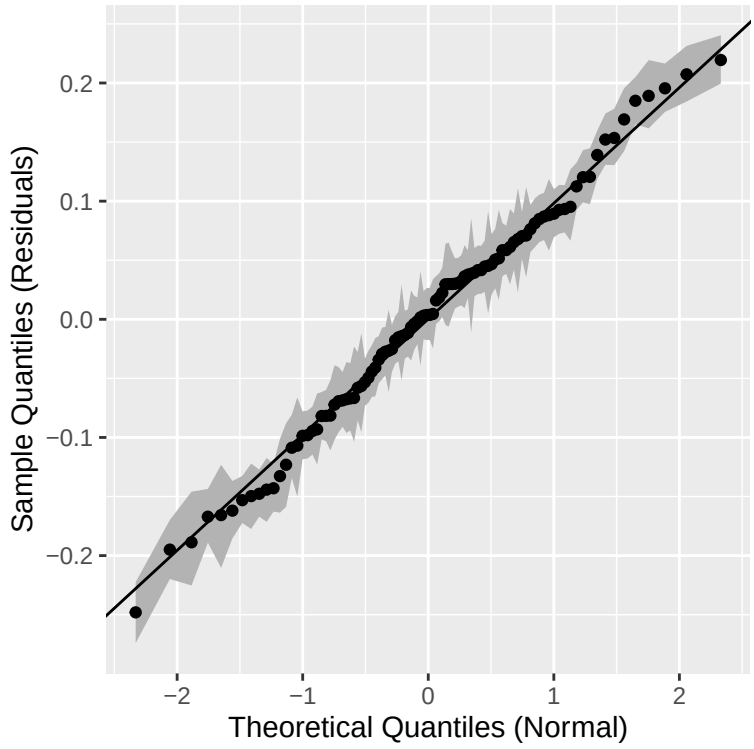


Figure 1: Bayesian residual plots: the left panel is the residual index plot; the right panel is the plot of the residual versus the covariate x

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)")
```



0.1.4 Posterior Predictive Checks

Now, instead of generating samples from the mean, we will account for the observational process uncertainty by:

1. Sampling $y_i^{1k} \sim \pi(y_i|\mathbf{y})\ k = 1, \dots, M;\ i = 1, \dots, 100$ using `generate()` (here we will draw $M = 500$ samples)

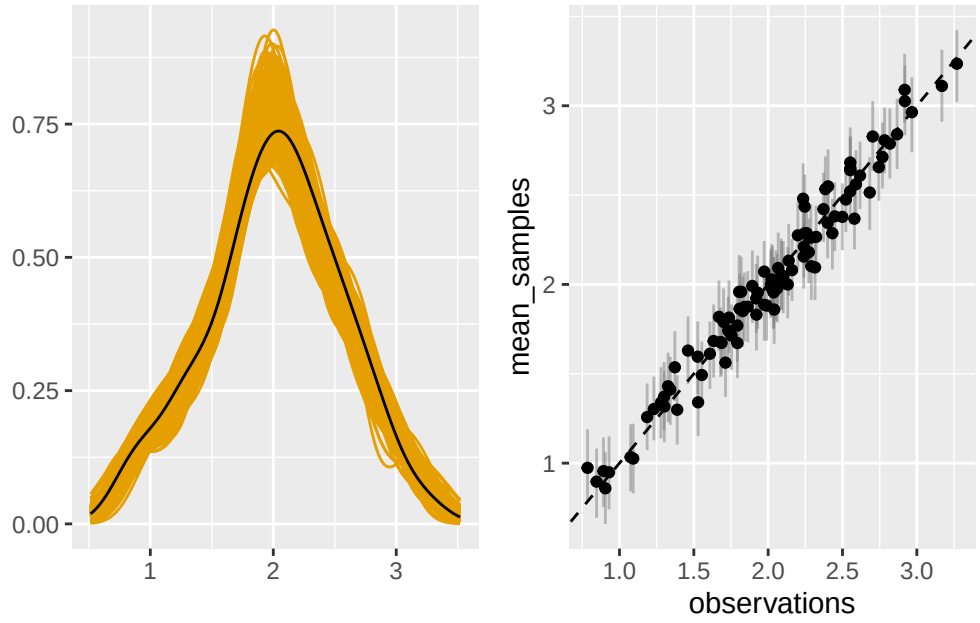
```
samples =  generate(fit.lm, df,
  formula = ~ {
    mu <- (beta_0 + beta_1)
    sd <- sqrt(1 / Precision_for_the_Gaussian_observations)
    rnorm(100, mean = mu, sd = sd)
  },
  n.samples = 500
)
```

2. Comparing some summaries of the simulated data with the one of the observed one

Here we compare (i) the estimated posterior densities $\widehat{\pi}^k(y|\mathbf{y})$ with the estimated data density and (ii) the samples means and 95% credible intervals against the observations.

```
# Tidy format for plotting
samples_long = data.frame(samples) %>%
  mutate(id = 1:100) %>% # i-th observation
  pivot_longer(-id)

# compute the mean and quantiles for the samples
draws_summaries = data.frame(mean_samples = apply(samples,1,mean),
q25 = apply(samples,1,function(x)quantile(x,0.025)),
```

```
q975 = apply(samples,1,function(x)quantile(x,0.975)),
observations = df$y)

p1 = ggplot() + geom_density(data = samples_long,
                         aes(value, group = name),  color = "#E69F00") +
  geom_density(data = df, aes(y))  +
  xlab("") + ylab("")

p2 = ggplot(draws_summaries,aes(y=mean_samples,x=observations))+
  geom_errorbar(aes(ymin = q25,
                    ymax = q975),
                alpha = 0.5, color = "grey50")+
geom_point()+geom_abline(slope = 1,intercept = 0,lty=2)+labs()

p1 +p2
```



0 GLM model checking

In this exercise we will:

- Learn about some model assessments techniques available in INLA
- Conduct posterior predictive model checking using CPO and PIT

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 for female crabs is

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

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: 2.13.0
INLA version: 25.08.21-1
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
Observation models:
  Family: 'poisson'
    Tag: <No tag>
    Data class: 'data.frame'
    Response class: 'integer'
    Predictor: satell ~ .
    Additive/Linear: TRUE/TRUE
    Used components: effects[beta0, colordarker, colorlight, colormedium, w], latent[]
Time used:
    Pre = 0.339, Running = 0.263, Post = 0.0708, Total = 0.673
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   0
colordarker -0.008 0.180     -0.362   -0.008      0.345 -0.008   0
colorlight   0.445 0.176      0.101    0.445      0.790  0.445   0
colormedium  0.248 0.118      0.017    0.248      0.479  0.248   0
w            0.001 0.000      0.000    0.001      0.001  0.001   0

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)')
```

0.2.1 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 a 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 density 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:

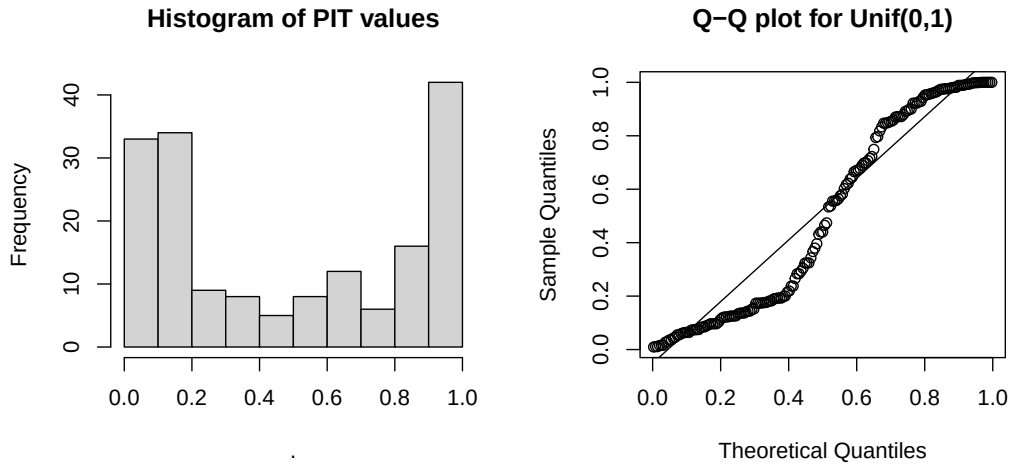
1. ask `inlabru` to compute them by set `options = list(control.compute = list(cpo = TRUE))` in the `bru()` function arguments.
2. set this as default in `inlabru` global option using the `bru_options_set` function.

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 PIT 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))
```

Both Q-Q plots and histogram of the PIT values suggest a not so great model fit. For the CPO values, usually the following summary of the CPO is often used:

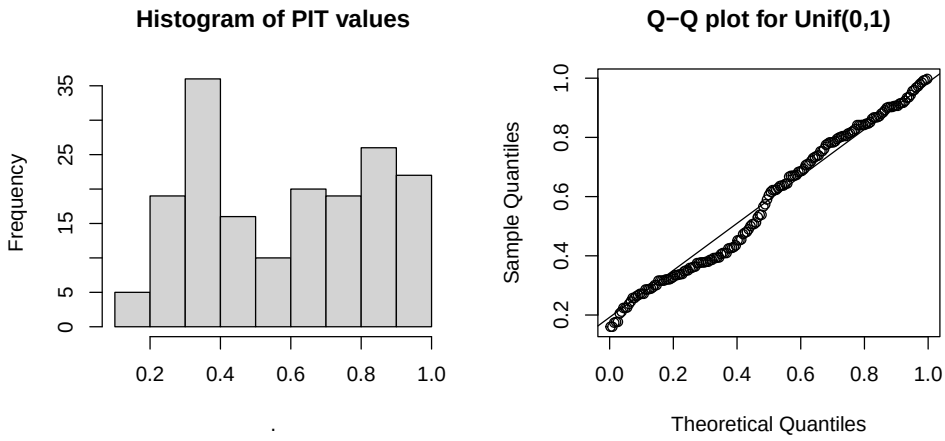
$$-\sum_{i=1}^{n} \log(\text{CPO\_}i)$$

This quantities is useful 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

```r
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 PIT 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))
```




```r
# 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
```

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

2 Leave-Group-Out Cross validation

In this practical we are going to fit a geostatistical model. We will:

- Learn how to to compute model comparison in `inlabru` using LGOCV

Libraries to load:

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


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

2.1.1 The data

In this practical, 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 and 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 dataset contains presence/absence data from 2003 to 2017. Lets consider year 2003 for now.

We first load the dataset and select the year of interest

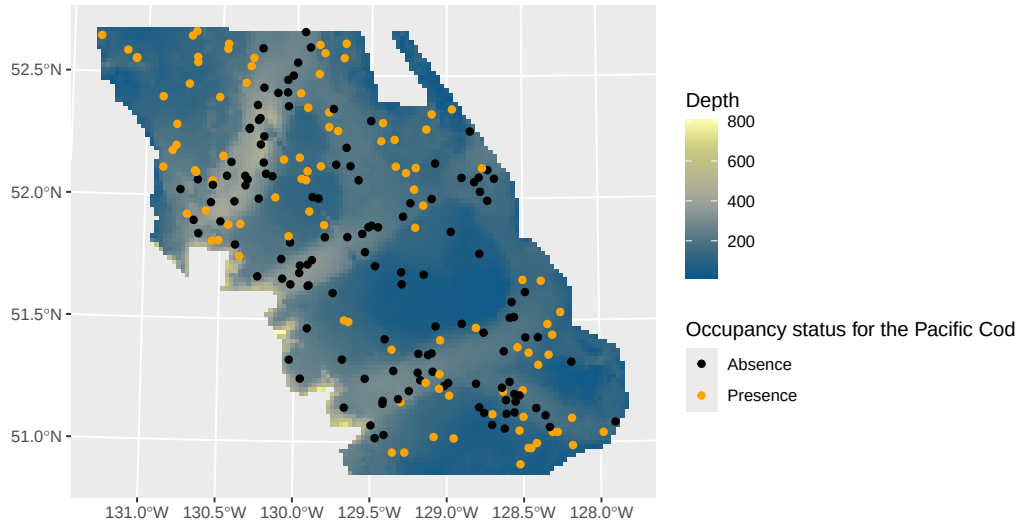
```
library(sdmTMB)

pcod_df = sdmTMB::pcod %>% filter(year==2003)
qcs_grid = sdmTMB::qcs_grid
```

Then, we create ab `sf` object and assign the rough coordinate reference to it:

```
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" )
```

We convert the covariate into a raster and assign the same coordinate reference:

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



2.1.2 The models

Here, we will model the binary presence-absence data as a Binomial model of the form:

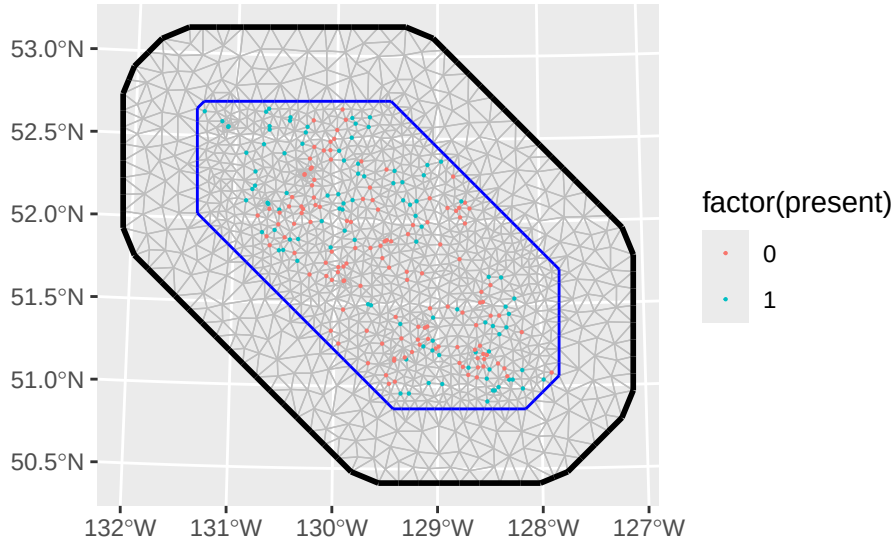
$$
\begin{aligned}
y(s)|\eta(s) &\sim \text{Binom}(1, p(s)) \\
\eta(s) &= \text{logit}(p(s)) \\
\omega(s) &\sim \text{ GF with range } \rho \text{ and maginal variance } \sigma^2
\end{aligned}
$$

We will fit three models. One where we consider the observation as Bernoulli and where the linear predictor contains only one intercept and the GR field $\omega(s)$ defined through the SPDE approach. The other two models will also include depth as a linear and non-linear smoothed covariate in the linear predictor.

Construct the mesh and the SPDE model

```
mesh = fm_mesh_2d(loc = pcod_sf,
                  max.edge = c(10,20),
                  offset = c(5,50))

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

```
                                    prior.range = c(100, 0.5))
```



Model 1 - spatial only

1. **Model 1 - spatial only** This is a model with that includes a structured spatial effect only. Here the linear predictor is defined as:

$$\eta(s) = \text{logit}(p(s)) = \beta_0 + \omega(s)$$

```
# Model 1 components
cmp_1 = ~ Intercept(1) + space(geometry, model = spde_model)

# Model 1 linear predictor
formula_1 = present ~ Intercept  + space

# Model 1 observational model
lik_1 = bru_obs(formula = formula_1,
                data = pcod_sf,
                family = "binomial")
# fit Model 1
fit_spatial = bru(cmp_1,lik_1)
```

2. **Model 2 - Depth covariate** The depth enters the model in a linear way. The linear predictor is then defined as:

$$\eta(s) = \text{logit}(p(s)) = \beta_0 + \omega(s) + \beta_1 \, \text{depth}(s)$$

```
# Model 2 components
cmp_2 = ~ Intercept(1) +
  space(geometry, model = spde_model)+
  covariate(depth_r$depth_scaled, model = "linear")

# Model 2 linear predictor
formula_2 = present ~ Intercept  + covariate + space

# Model 2 observational model
lik_2 = bru_obs(formula = formula_2,
              data = pcod_sf,
              family = "binomial")

# Fit Model 2
fit_depth_linear = bru(cmp_2,lik_2)
```

3. **Model 2 - Smooth depth covariate** The depth enters the model in a non linear way. The linear predictor is then defined as:

$$\eta(s) = \text{logit}(p(s)) = \beta_0 + \omega(s) + f(\text{depth}(s))$$

```
# create the grouped variable
depth_r$depth_group = inla.group(values(depth_r$depth_scaled))

# Model 3 components
cmp_3 = ~ Intercept(1) +
  space(geometry, model = spde_model)+
  covariate(depth_r$depth_group, model = "rw2")

# Model 3 linear predictor
formula_3 = present ~ Intercept  + covariate + space

# Model 2 observational model
lik_3 = bru_obs(formula = formula_2,
              data = pcod_sf,
              family = "binomial")

# Fit Model 2
fit_depth_smooth = bru(cmp_3,lik_3)
```

2.1.3 Model Comparison through LGOCV

Next we illustrate how to implement modelling comparison using leave-out group cross validation (LGOCV). The underlying idea is that of a Bayesian prediction setting where we approximate the posterior predictive density $\pi(\tilde{\mathbf{y}}|\mathbf{y})$ defined as the integral over the posterior distribution of the parameters, i.e.

$$\pi(\mathbf{\Psi}|\mathbf{y}) = \int_{\theta} \pi(\mathbf{\Psi}|\theta, \mathbf{y})\pi(\theta|\mathbf{y})d\theta$$

the LGOCV selects a fixed test point i and remove a certain group of data $\mathbb{I}_i$ according to a specific prediction task. Thus, we are interested in the posterior predictive density

$$\pi(Y_i|\mathbf{y}{-}\mathcal{J}i) = \int \theta\pi(Y_i|\theta, \mathbf{y}{-}\mathbb{I}_i)\pi(\theta|\mathbf{y})d\theta$$

With this, a point estimate $\tilde{Y}_i$ can be computed based on $\pi(Y_i|\mathbf{y}_{-\mathbb{I}_i})$ and the predictive performance be assessed using an appropriate scoring function $U(\tilde{Y}_i, Y_i)$, for example, the log-score function

$$\frac{1}{n}\sum_{i=1}^{n} \log \pi(\mathbf{\Psi}|\mathbf{y}).$$

In this example, the LGOCV strategy will be used to compare the previous fitted spatially explicit models. Here, the leave-out group $\mathbb{I}_i$ is manually defined for the ith row of the data based on a buffer of size $b = 25$ centered at each data point:

```
# create buffer of size 25 centred at each site
buffer <- st_buffer(pcod_sf, dist = 25)

# Lists of the indexes of the leave-out-group for each observation i
Ii <- st_intersects(pcod_sf,buffer)
```

Figure 2 illustrate the manual construction of the leave-out-group for the 2nd observation in our data.

We then can use the `inla.group.cv` function to compute the log-scores for Model 3 as follows:

```
lgocv_depth_smooth = inla.group.cv(result = fit_depth_smooth,
                                   groups = Ii)
log_depth_smooth = mean(log(lgocv_depth_smooth$cv),
                        na.rm=T)
```

When we do model comparison, we want to have the same groups for different models. We can easily do this by passing an `inla.group.cv` class object to `inla.group.cv`function. If we want to use the groups constructed by model 3 to compute LGOCV, we have the following code:

```
lgocv_spatial = inla.group.cv(result = fit_spatial,
                              group.cv = lgocv_depth_smooth)

lgocv_depth_linear = inla.group.cv(result = fit_depth_linear,
                                   group.cv = lgocv_depth_smooth)

log_score_spatial<- mean(log(lgocv_spatial$cv),na.rm=T)
log_depth_linear <-mean(log(lgocv_depth_linear$cv),na.rm=T)
log_depth_smooth <-mean(log(lgocv_depth_smooth$cv),na.rm=T)
```

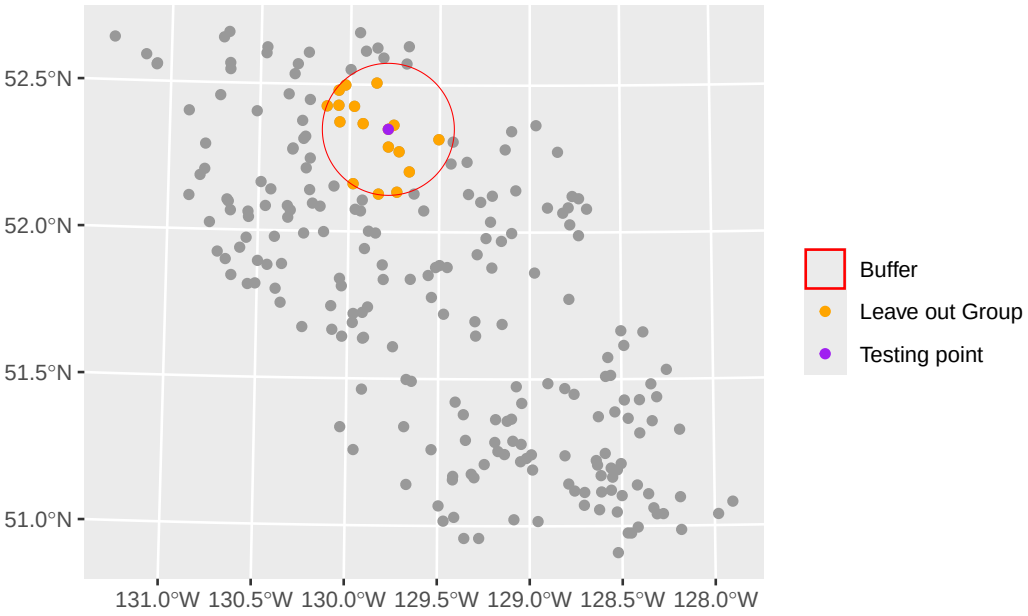


Figure 2: Example of the CV strategy for the 2nd testing point.

Log-Score LGOCV

Continuous spatial Gaussian field	Linear depth covariate effect	Smooth depth</
-0.71	-0.64	-

In this example, the model that includes the smoothed covariate has the highest log-score and thus it is preferred over the other two "simpler" models.

```
data.frame(logspat= log_score_spatial,
           logdepthl = log_depth_linear,
           logdepths = log_depth_smooth )
```

2.1.4 Spatio-temporal LGOCV comparison

2.1.4.1 Model fitting Now lets compare two different space-time models using LGOCV and some information criteria metrics. The general model structure is given by:

$$y(s,t)|\eta(s,t) \sim \text{Binom}(1, p(s,t))$$
$$\eta(s,t) = \text{logit}(p(s,t))$$

We begin by loading the data between 2003 and 2011 and convert it to an `sf` object (we will also create a time index for modelling):

```
pcod_spat = sdmTMB::pcod %>%
  filter(year %in% 2003:2011) %>%
  st_as_sf( coords = c("lon","lat"), crs = 4326) %>%
  st_transform(., crs = "+proj=utm +zone=9 +datum=WGS84 +no_defs +type=crs +units=km" ) %>%
   mutate(time_idx = match(year, c(2003, 2004, 2005, 2007, 2009, 2011, 2013, 2015, 2017)),
          id = 1:nrow(.)) # Observation id for CV
```

Now we tell `inlabru` that we want to compute WAIC, DIC and model likelihood as follows:

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

1. **Model 1 - time iid effect** We consider a separable space-time model with a linear predictors given by:

$$\eta(s,t) = \beta_0 + f_1(\text{depth}(s)) + f_2(t) + \omega(s)$$

- $f_1(\text{depth}(s))$ is a smooth covariate effect of depth
- $f_2(t)$ is an IID effect of time
- $\omega(s)$ is Matérn random field.

```
# Model components
cmp_spat = ~ Intercept(1) +
  covariate(depth_r$depth_group, model = "rw2")+
  trend(time_idx, model = "iid")+
  space(geometry, model = spde_model)
```

```
# Linear predictor
formula_spat = present ~ Intercept  + covariate  + trend + space

# Observational model
lik_spat = bru_obs(formula = formula_spat,
              data = pcod_spat,
              family = "binomial")

# Fit Model
fit_spat = bru(cmp_spat,lik_spat)
```

> **Note**
>
> Note that there are some survey locations in certain years that fall outside the depth raster region. `inlabru` will input these missing covariate values using the nearest available value. This can be computationally expensive, but you can avoid it by supplying a raster layer that encompasses all of your data points (e.g., by pre-imputing these missing values with your preferred method of choice).

1. **Model 2 - spatiotemporal field** We consider a separable space time model with a linear predictor given by:

$$\eta(s,t) = \beta_0 + f_1(\text{depth}(s)) + \omega(s,t)$$

- $f_1(\text{depth}(s))$ is a smooth covariate effect of depth
- $\omega(s,t)$ is a space-time Matérn spatial field with AR1 time component

Now we fit the model as follows (this might take a couple of minutes to run):

```
# PC prior for AR(1) correlation parameter
h.spec <- list(rho = list(prior = 'pc.cor0', param = c(0.5, 0.1)))

# Model components
cmp_spat_ar1 = ~ Intercept(1) +
  covariate(depth_r$depth_group, model = "rw2")+
  space_time(geometry,
        group = time_idx ,
        model = spde_model,
        control.group = list(model = 'ar1',hyper = h.spec))

# Linear predictor
formula_spat_ar1 = present ~ Intercept  + covariate  + space_time

# Observational model
lik_spat_ar1 = bru_obs(formula = formula_spat_ar1,
              data = pcod_spat,
              family = "binomial")

# Fit Model
fit_spat_ar1 = bru(cmp_spat_ar1,lik_spat_ar1)
```

2.1.4.2 Model comparison To manually define the leave-out groups we can apply the buffer approach as before while considering a time window of $t \pm q$. Here we set $q = 2$ as we believe time dependency is weaker after a 2 year period (Figure 3 illustrate this strategy for the 750th observation). To compute this we can create a buffer surrounding each location and then loop through every observation to identify the leave-out groups based on those location that are inside the buffer within a 2 year span:

```
# create buffer of size 25  centred at each site

buffer_25 <- st_buffer(pcod_spat, dist = 25)


# empty lists to include the indexes of the leave-out-group for each observation i
I_i <- list()

# loop though each observation and store the leave-out-group based on the buffer
for( i in 1:nrow(pcod_spat)){

  # Temporal filtering of data within a 2 years of span of  observation i
  df_sf_subset <- pcod_spat %>%
    filter( between(time_idx,left = pcod_spat$time_idx[i]-2,
                    right = pcod_spat$time_idx[i]+2))
  # Spatial filtering of the observations that are within the buffer of the ith observation
  Buffer_i <-df_sf_subset %>% st_intersects(buffer_25[i,],sparse = FALSE) %>% # identify
    unlist()

  # obtain the indexes of the leave out group
  I_i[[i]] <-  df_sf_subset[Buffer_i,] %>%  pull(id)

}
```

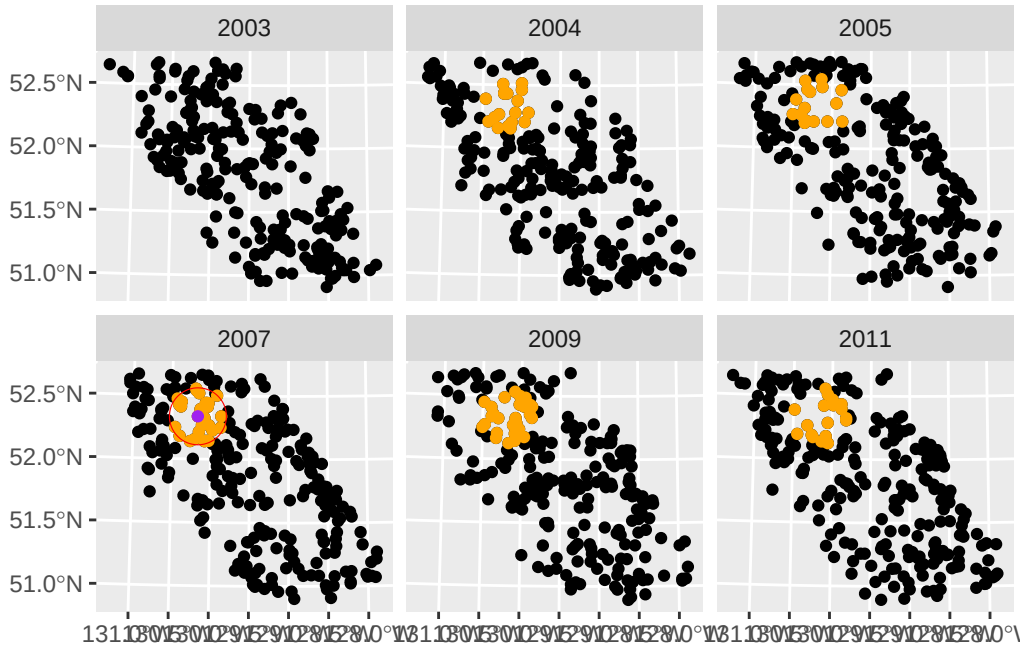


Figure 3: Example of the spatiotemporal CV strategy for the 100nd testing point.

Now that we have the leave-out group indexes for each observation, we can compute the

log-score using the `inla.group.cv` function as before:

```
lgocv_spat_ar1 <- inla.group.cv(result = fit_spat_ar1, groups  = I_i)
logscore_spat_ar1 = mean(log(lgocv_spat_ar1$cv),na.rm=T)


lgocv_spat <- inla.group.cv(result = fit_spat, group.cv  = lgocv_spat_ar1)
logscore_spat = mean(log(lgocv_spat$cv),na.rm=T)
```

The following table summarises different model comparison metrics, which favors the model with a spatio-temporal field over the simples *iid* model.

```
table = data.frame( DIC = c(fit_spat_ar1$dic$dic, fit_spat$dic$dic),
                    WAIC = c(fit_spat_ar1$waic$waic, fit_spat$waic$waic),
                    mlik = c(fit_spat_ar1$mlik[1,1],fit_spat$mlik[1,1]),
                    LGOCV = c(logscore_spat_ar1,logscore_spat))

 rownames(table) = c("Model 2 - spatiotemporal field",
                     "Model 1 - time iid effect")
```

	DIC	WAIC	mlik	LGOCV
Model 1 - time iid effect	1356.179	1351.832	-763.2828	-0.5001042
Model 2 - spatiotemporal field	1338.464	1330.124	-758.5321	-0.4985709

Task

Modify **Model 1** so that instead of an iid yearly trend, the $f_2(t)$ trend follow an AR(1) process. Compare this again the the other two models using the LGOCV scores.
Take hint
You only need to change the model component: `trend(time_idx, model = ???)`.
You may also use the PC prior we set for the spatiotemporal field in Model 2.
Click here to see the solution

```r
# Model components
cmp_spat_2 = ~ Intercept(1) +
  covariate(depth_r$depth_group, model = "rw2")+
  trend(time_idx, model = "ar1",
        control.group = list(model = 'ar1',hyper = h.spec))+
  space(geometry, model = spde_model)

# Linear predictor
formula_spat_2 = present ~ Intercept  + covariate  + trend + space

# Observational model
lik_spat_2 = bru_obs(formula = formula_spat,
              data = pcod_spat,
              family = "binomial")

# Fit Model
fit_spat_2 = bru(cmp_spat_2,lik_spat_2)

# Compute log-score

lgocv_spat_2 <- inla.group.cv(result = fit_spat_2, group.cv  = lgocv_spat_ar1)
mean(log(lgocv_spat_2$cv),na.rm=T)
```