Lab session 3

Aim of this practical session:

In this practical session we are going

- Manipulate and visualize different types of spatial objects.
- Familiarize with the syntax for fitting LGMs in the `inlabru` software
- Learn how to fit a Bayesian areal model for disease risk mapping (Section 1)
- Learn how to fit a Geostatistical spatial model (Section 2).
- Fit point process models in `inlabru` (Section 3).

The data for this session is available throughout different packages, if you cannot access some of these libraries you can the download the data using the link below and then load it into R using the `load()` function:

```
load("lab_3_DataFiles.RData")
```

First lets load the R libraries we will use during this session:

```
# Libraries for Data manipulation
library(tidyr)
library(dplyr)
library(sf)
library(terra)

# Libraries for producing maps
library(ggplot2)
library(scico)
library(tidyterra)
library(mapview)
library(patchwork)

# Libraries for the Analysis
library(spdep)
library(gstat)
library(variosig)
library(INLA)
library(inlabru)
```

1 Disease risk modelling

The Data: Lip cancer rates in Scotland

In this example we model the number of lip cancer rates in Scotland in the years 1975–1980 at the county level in order to evaluate the presence of an association between sun exposure and lip cancer.

The data is available on the `SpatialEpi` R package

```
library(SpatialEpi)
data(scotland_sf)
```

The `scotland_sf` data is a Simple Features (`sf`) object containing the spatial polygon information for the set of 56 counties. The `sf` package allows us to work with vector data which is used to represent points, lines, and polygons. It can also be used to read vector data stored as a shapefiles.

The dataset contains the following variables of interest:

Variable	Meaning
`county.names`	Scotland County name
`cases`	Number of Lip Cancer cases per county
`expected`	Expected number of lip cancer cases
`AFF`	Proportion of the population who work in agricultural fishing and farming
`geometry`	Geometric representation of counties in Scotland

1.1 Standardized Mortality Ratios and spatial correlation

In epidemiology, disease risk is usually estimated using Standardized Mortality Ratios (SMR). The SMR for a given spatial areal unit i is defined as the ratio between the observed (Y_i) and expected (E_i) number of cases:

$$SMR_i = \frac{Y_i}{E_i}$$

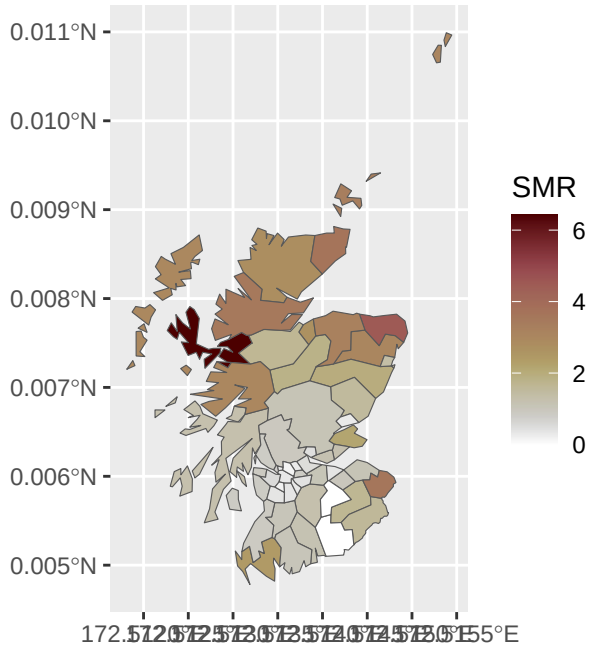
A value $SMR > 1$ indicates that there are more observed cases than expected which corresponds to a high risk area. On the other hand, if $SMR < 1$ then there are fewer observed cases than expected, suggesting a low risk area.

We can manipulate `sf` objects the same way we manipulate standard data frame objects via the `dplyr` package. Lets use the pipeline command %>% and the `mutate` function to calculate the yearly SMR values for each county:

```
# Compute the SMR and add a region index (for later modelling)
scotland_sf <- scotland_sf %>% mutate(
  SMR = cases/expected,
  region_id = 1:nrow(scotland_sf))
```

Now we use `ggplot` to visualize our data by adding a `geom_sf` layer and coloring it according to our variable of interest (i.e., SMR) choosing an appropriate color palette using the `scale_fill_scico` from the `scico` package:

```
# Visualize the regions colored by the SMR
ggplot()+geom_sf(data=scotland_sf,aes(fill=SMR))+scale_fill_scico(direction = -1)
```

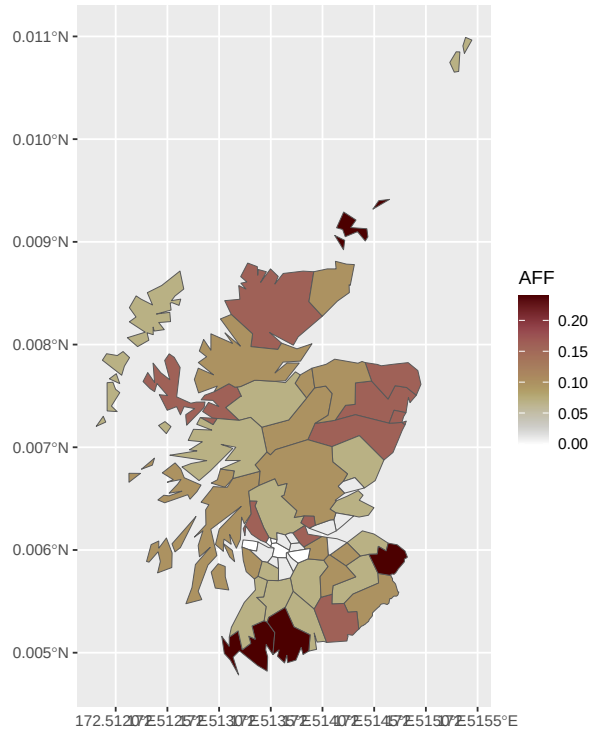


We can see that higher lip cancer risks occurs in the north of Scotland.

Task

Produce a map that shows the spatial distribution of the proportion of the population who work in agricultural fishing and farming (`AFF`) for each county.
Click here to see the solution

```
ggplot()+geom_sf(data=scotland_sf,aes(fill=AFF))+scale_fill_scico(direction = -1)
```



1.1.1 Neighbourhood structure

A key aspect of any spatial analysis is that observations closer together in space are likely to have more in common than those further apart. This can lead us towards approaches similar to those used in time series, where we consider the spatial *closeness* of our regions in terms of a *neighbourhood structure*.

The function `poly2nb()` of the spdep package can be used to construct a list of neighbors based on areas with contiguous boundaries (e.g., using Queen contiguity).

```
W.nb <- poly2nb(scotland_sf,queen = TRUE)
```

```
Warning in poly2nb(scotland_sf, queen = TRUE): some observations have no neighbours;
if this seems unexpected, try increasing the snap argument.
```

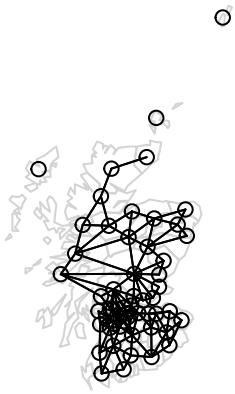
```
Warning in poly2nb(scotland_sf, queen = TRUE): neighbour object has 4 sub-graphs;
if this sub-graph count seems unexpected, try increasing the snap argument.
```

```
W.nb
```

```
Neighbour list object:
Number of regions: 56
Number of nonzero links: 234
Percentage nonzero weights: 7.461735
Average number of links: 4.178571
3 regions with no links:
3, 53, 55
4 disjoint connected subgraphs
```

The warning tell us that the neighbourhood is comprised of 4 interconnected regions. By looking at the neighbourhood graph below, we can see that these are the three separate island in the north.

```
plot(st_geometry(scotland_sf), border = "lightgray")
plot.nb(W.nb, st_geometry(scotland_sf), add = TRUE)
```



1.1.2 Moran's I

Once we have identified a set of neighbours using our chosen method, we can use this to account for correlation.

We can use Moran's I as a measure of global spatial autocorrelation based on a neighbourhood matrix:

$$w_{ij} = \begin{cases} 1 & \text{if areas } (B_i, B_j) \text{ are neighbours.} \\ 0 & \text{otherwise.} \end{cases}$$

In R we can use the `nb2listw` function to create w while specifying `zero.policy = TRUE` as we have some regions with no neighbours.

```
# neighbors list
nbw <- nb2listw(W.nb, style = "W",zero.policy = TRUE)
```

Moran's I ranges between -1 and 1, and can be interpreted in a similar way to a standard correlation coefficient.

- $I = 1$ implies that we have **perfect spatial correlation**.
- $I = 0$ implies that we have **complete spatial randomness**.

- $I = -1$ implies that we have **perfect dispersion** (negative correlation).

Our observed I is a point estimate, and we may also wish to assess whether it is significantly different from zero. We can test for a statistically significant spatial correlation using a permutation test, with hypotheses:

$$H_0 : \text{ negative or no spatial association } (I \leq 0)$$
$$H_1 : \text{ positive spatial association } (I > 0)$$

We can use `moran.test()` to test this hypothesis by setting `alternative = "greater"`. To do so, we need to supply list containing the neighbors via the `nb2listw()` function from the `spdep` package. Lets assess now the spatial autocorrelation of the SMR in 2011:

```
# Global Moran's I
gmoran <- moran.test(scotland_sf$SMR, nbw,
                     alternative = "greater")
gmoran
```

```
    Moran I test under randomisation

data:  scotland_sf$SMR
weights: nbw
n reduced by no-neighbour observations

Moran I statistic standard deviate = 5.6068, p-value = 1.03e-08
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.497024704      -0.019230769       0.008478093
```

> Question
>
> What do we conclude from the Moran's I test?
> Answer
> Since have set the alternative hypothesis to be $ I > 0$ and have a p-value < 0.05, we then reject the null hypothesis and conclude there is evidence for positive spatial autocorrelation.

1.2 Fitting a BYM model

As with the other types of spatial modelling, our goal is to observe and explain spatial variation in our data. Generally, we are aiming to produce a smoothed map which summarises the spatial patterns we observe in our data.

1.2.1 The Model structure

Now will fit a BYM model to the data where we consider a Poisson model for the observed cases. Remember that we can formulate this as a Latent Gaussian Model (LGM) on threes stages:

- **Stage 1:** We assume the responses are Poisson distributed:

$$y_i|\eta_i \sim \text{Poisson}(E_i\lambda_i)$$
$$\log(\lambda_i) = \eta_i = \beta_0 + \beta_1 \text{AFF} + u_i + z_i$$

- **Stage 2:** η_i is linked to the disease relative risks (RR) via the log link function and is a linear function of **four components**: β_0 the overall intercept, β_1 the AFF effect,**u** $= (u_1, \dots, u_n)$ a spatially structured model (or ICAR) with precision matrix τ_u**Q** and an unstructured *iid* random effect **z** $= (z_1, \dots, z_n)$ with precision τ_z
- **Stage 3:** $\{\tau_z, \tau_u\}$: Precision parameters for the random effects

The latent field is then given by **x** $= (\beta_0, \beta_1, u_1, u_2, \dots, u_n, z_1, ..., z_n)$, the hyperparameters are $\theta = (\tau_u, \tau_z)$.

We can fit this model with `inlabru` by following the next workflow and defining:

1. The model components (`cmp`)
2. The `formula` for the linear predictor
3. The observational model (via the `bru_obs` function)
4. fit the model using the `bru` function

Model components

Our model has 4 compontents, (i) a global intercept, (ii) the covariate AFF effect (which is a column in our data set), (iii) a unstructured random effect **z** $\sim N(0, \tau_z^{-1})$) and (iv) a structured random effect **u** that follows an ICAR model structure of the form:

$$u_i|\mathbf{u}_{-i}, \tau_u, \sim N\left(\frac{1}{d_i}\sum_{j\sim i} u_j, \frac{1}{d_i\tau_u}\right)$$

Where

- $\mathbf{u}_{-i} = (u_i, \dots, u_{i-1}, u_{i+1}, \dots, u_n)^T$
- τ_u is the precision parameter (inverse variance).
- d_i is the number of neighbours
- Q denotes the precision matrix defined as

$$Q_{i,j} = \begin{cases} d_i, & i = j \\ -1, & i \sim j \\ 0, & \text{otherwise} \end{cases}$$

Thus, we can define the precision matrix Q according the proximity matrix using the `nb2mat` function (this is similar to `nb2listw` we used before but it will give us a matrix rather than a list):

```
R <- nb2mat(W.nb, style = "B", zero.policy = TRUE)
diag = apply(R,1,sum)
Q = -R
```

```
diag(Q) = diag
```

Recall The precision matrix Q depends on the neighboring structure and τ_u (which will be estimated). Now, to make the precision parameters of models with different intrinsic Gaussian random field comparable we add a sum-to-zero constrain $\sum_i^n u_i = 0$ (see `scale.model = TRUE` in the code below).

```
cmp = ~ Intercept(1) + beta_1(AFF, model = "linear") +
   z_i(region_id , model = "iid") +
  u_i(region_id, model = "besag", graph = Q,scale.model = TRUE)
```

Here, `Intercept`, `beta_1`, `u_i` and `z_i` are the names of the model components (note these are just labels we could have chosen any other names for them)

The linear predictor and the observational model

Now we need to specify:

1. The linear predictor formula for the model components (here we declare the labels that we used in the previous step)

```
formula = cases ~ Intercept + beta_1 + u_i + z_i
```

2. The observational model/likelihood for our observations:

```
lik = bru_obs(formula = formula,

              family = "poisson",

              E = expected,

              data = scotland_sf)
```

Notice how we add the **expected number of cases** as our offset through the `E` argument of `bru_obs` and specify the distribution of our data through the `family` argument.

Fit the Model and Extract the Results

We finally fit the model using the `bru` function and extract some posterior summaries using the `summary` function:

```
fit = bru(cmp, lik)
summary(fit)
```

```
inlabru version: 2.13.0.9016
INLA version: 25.08.21-1
Components:
Latent components:
Intercept: main = linear(1)
beta_1: main = linear(AFF)
z_i: main = iid(region_id)
```

```
u_i: main = besag(region_id)
Observation models:
  Family: 'poisson'
    Tag: <No tag>
    Data class: 'sf', 'data.frame'
    Response class: 'numeric'
    Predictor: cases ~ Intercept + beta_1 + u_i + z_i
    Additive/Linear: TRUE/TRUE
    Used components: effects[Intercept, beta_1, z_i, u_i], latent[]
Time used:
    Pre = 0.695, Running = 0.448, Post = 0.269, Total = 1.41
Fixed effects:
            mean    sd 0.025quant 0.5quant 0.975quant   mode kld
Intercept -0.306 0.119     -0.538   -0.307     -0.069 -0.307   0
beta_1     4.317 1.272      1.758    4.337      6.761  4.337   0

Random effects:
  Name    Model
    z_i IID model
   u_i Besags ICAR model

Model hyperparameters:
                      mean       sd 0.025quant 0.5quant 0.975quant    mode
Precision for z_i 22037.69 24119.08    1474.64 14479.14   86078.65 4023.44
Precision for u_i     4.14     1.43       2.02     3.91       7.60    3.48

Marginal log-Likelihood:  -193.98
 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)')
```

Looking at $\exp(\hat{\beta}_1 \times 0.10)$ we can see that a 10 percentage-point increase in the proportion working in agriculture/fishing/farming is associated with about a 2 increase in disease risk, after accounting for the spatial structures. In general, areas with a higher proportion of people working in agriculture, fishing, and farming tend to have substantially higher disease risk, even after accounting for spatial correlation using the ICAR effect.

Question

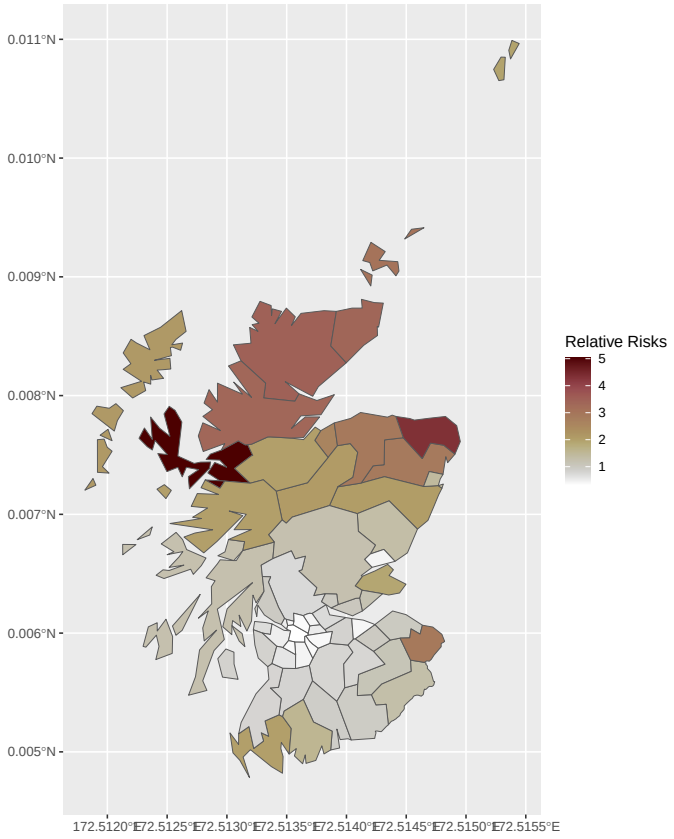
1. What is the estimated value for β_0? _____
2. Look at the estimated values of the hyperparameters , which of the two spatial components (structured or unstructured) explains more of the variability in the counts?

- (A) unstructured
- (B) structured

1.3 Plot predictions

We can then create a map of the RR using the `predict` function and supply the formula of our relative risks:

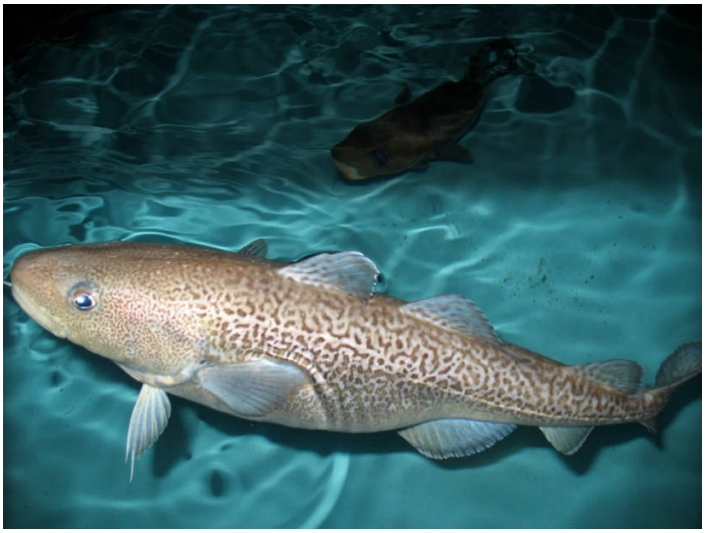
$$\lambda_i = \exp\{\beta_0 + \beta_1 \text{AFF} + u_i + z_i\}$$



2 Geostatistical Modelling

2.1 The Data

In this example, we will explore 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) from 2003 to 2017. In this example we **only** consider year 2003.

Let's create an initial sf spatial object using the standard geographic coordinate system (EPSG:4326). This correctly defines the point locations based on latitude and longitude.

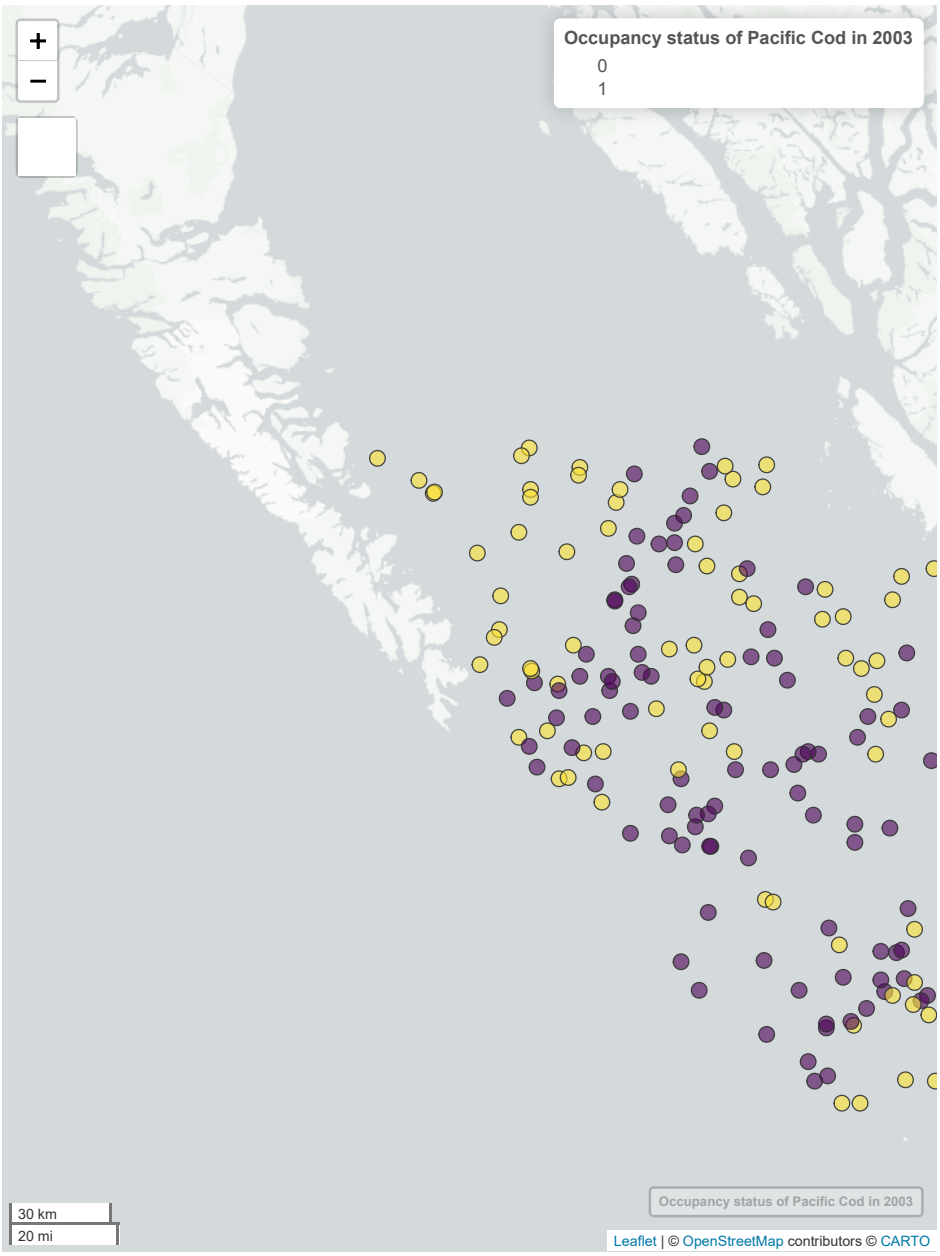
```
library(sdmTMB)
pcod = sdmTMB::pcod %>% filter(year==2003)
pcod_sf =   st_as_sf(pcod, coords = c("lon","lat"), crs = 4326)
```

Now we can transform to the standard UTM Zone 9N projection (EPSG:32609) and change the spatial units to *km* to better reflect the scale of our ecological study and to make resulting distance/area values more intuitive to interpret:

```
pcod_sf = st_transform(pcod_sf,
                       crs = "+proj=utm +zone=9 +datum=WGS84 +no_defs +type=crs +units=km" )
```

Let's map the presence/absence of the Pacific Cod in 2003 using the mapview function:

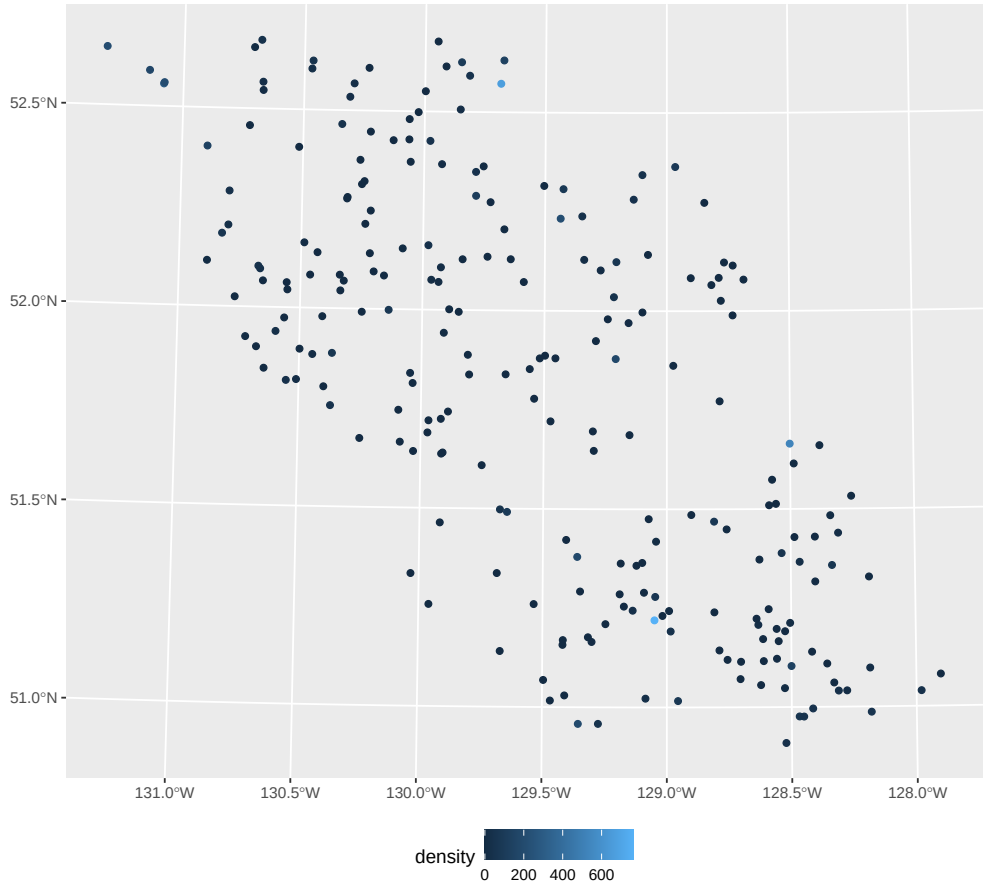
```
pcod_sf %>%
  mutate(present = as.factor(present)) %>%
mapview(zcol = "present",
        layer.name = "Occupancy status of Pacific Cod in 2003")
```



Task

Use `ggplot` and the `sf` library to map the biomass density of the pacific cod
hint
You can plot an`sf` object by adding a `geom_sf` layer to a ggplot object.
Click here to see the solution

```
ggplot()+
  geom_sf(data=pcod_sf,aes(color=density))+
  theme(legend.position = "bottom")
```



2.1.1 Raster Data

The `qcs_grid` data contain the depth values stored as 2×2 km grid for Queen Charlotte Sound. Environmental data are typically stored in raster format, which represents spatially continuous phenomena by dividing a region into a grid of equally-sized cells, each storing a value for the variable of interest. In R, the `terra` package is a modern and powerful tool for efficiently working with raster data. The function `rast()`, can be used both to read raster files from standard formats (e.g., `.tif` or `.tiff`) and to create a new raster object from a data frame. For instance, the following code creates a raster from the `qcs_grid` grid data for Queen Charlotte Sound.

```
qcs_grid = sdmTMB::qcs_grid
depth_r <- rast(qcs_grid, type = "xyz")
```

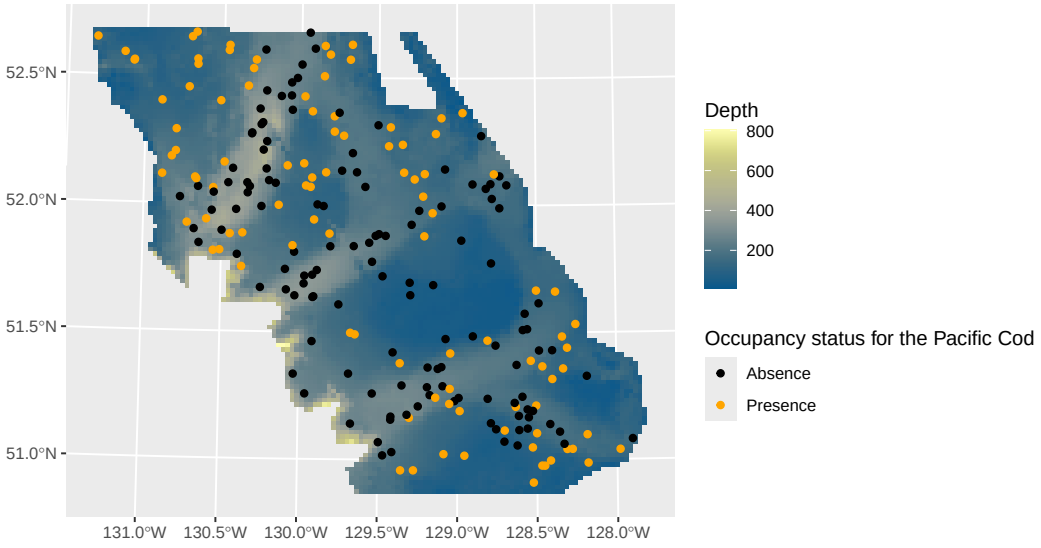
The raster object contains three layers corresponding to the (i) depth values, (ii) the scaled depth values and (iii) the squared depth values.

Notice that there are no CRS associated with the raster. Thus, we can assign appropriate CRS using the `crs` function. Additionally, we also want the raster CRS to match the CRS in the survey data (recall that we have previously reprojected our data to utm coordinates). We can assign an appropriate CRS that matches the CRS of the `sf` object as follows:

```
crs(depth_r) <- crs(pcod_sf)
```

We can use the `tidyterra` package to plot raster data using `ggplot` by adding a `geom_spatraster` function and then select an appropriate `fill` and `color` palettes:

```
ggplot()+
  geom_spatraster(data=depth_r$depth)+
  geom_sf(data=pcod_sf,aes(color=factor(present))) +
    scale_color_manual(name="Occupancy status for the Pacific Cod",
                       values = c("black","orange"),
                       labels= c("Absence","Presence"))+
  scale_fill_scico(name = "Depth",
                   palette = "nuuk",
                   na.value = "transparent" )
```



We can check the spatial extension of our study area using the `ext` function:

```
ext(depth_r)[1:2] %>% diff() # difference in x coord
```

```
xmax
 242
```

```
ext(depth_r)[3:4] %>% diff()  # difference in y coord
```

```
ymax
 204
```

Note that this is in Km since we have transformed our original data.

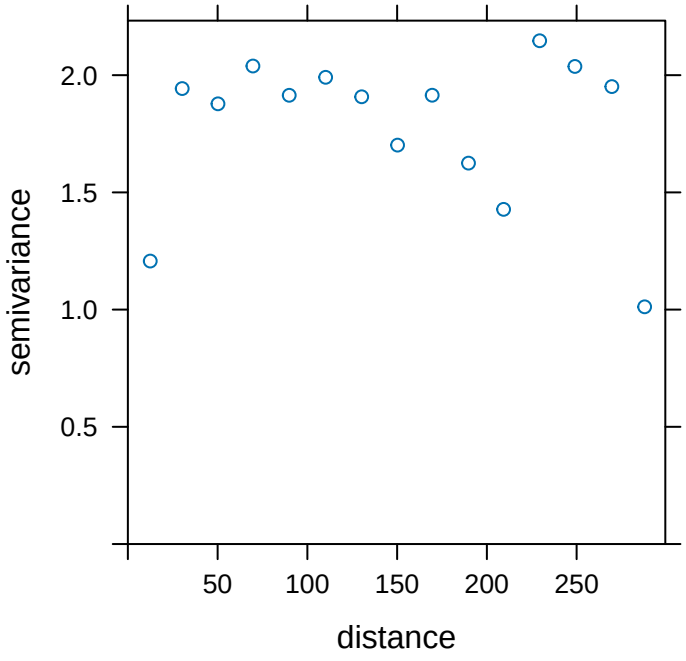
2.1.2 Autocorrelation and Variograms

Spatial statistics quantifies the fundamental principle that nearby things are closely related. This spatial dependence means that observation are not independent, as assumed by most classical statistical models, but are instead correlated relative to their proximity. While this correlation can be a valuable source of information, it must be explicitly accounted for to avoid wrong inference and incorrect conclusions.

The first step is to assess whether there is any evidence of spatial dependency in our data. Spatial dependence in georeferenced data can be explored by a function known as a variogram $2\gamma(\cdot)$ (or semivariogram $\gamma(\cdot)$). The variogram is similar in many ways to the autocorrelation function used in time series modelling. In simple terms, it is a function which measures the difference in the spatial process between a pair of locations a fixed distance apart.

We can calculate the binned- empirical variogram for the data using `variogram` function from the `gstat` library. This plot shows the semi-variances for each pair of points. Lets compute a variogram for the biomass log-density of the Pacific Cod in 2003:

```
vario_binned <- gstat::variogram(log(density) ~ depth_scaled  + depth_scaled2,
                                 data = pcod_sf %>% filter(density>0),
                                 cloud = FALSE,
                                 cutoff = 300)
plot(vario_binned)
```

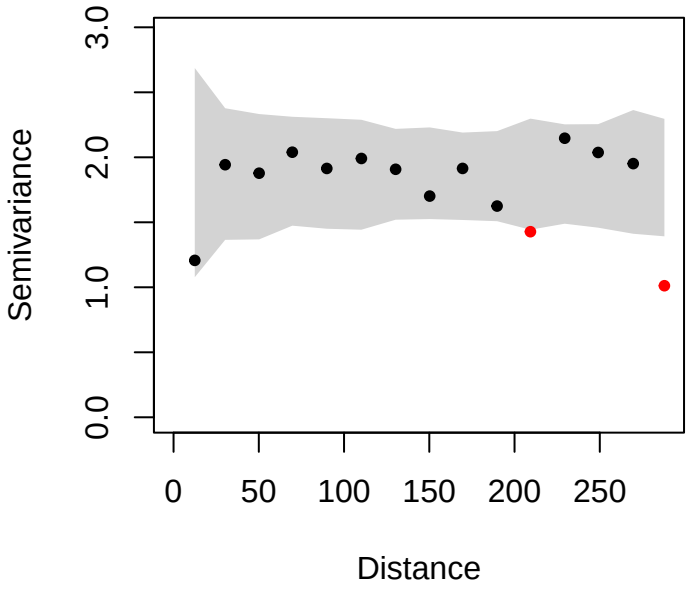


Assessing spatial dependence

We can construct null envelope based on permutations of the data values across the locations, i.e. envelopes built under the assumption of no spatial correlation. By overlapping these envelopes with the empirical variograms we can determine whether there is some spatial dependence in our data ,e.g. if our observed variograms falls outside of the envelopes constructed under spatial randomness.

We can construct permutation envelopes on the gstat empirical variogram using the `envelope` function from the `variosig` R package. Then we can visualize the results using the `envplot` function:

```
varioEnv <- envelope(vario_binned,
                     data = pcod_sf %>% filter(density>0),
                     locations = st_coordinates(pcod_sf %>% filter(density>0)),
                     formula = log(density) ~ depth_scaled  + depth_scaled2,
                     nsim = 499)
envplot(varioEnv)
```




```
[1] "There are 2 out of 15 variogram estimates outside the 95% envelope."
```

It seems that the depth and squared depth covariates account for large part of the spatial variability in the log density as only 2 point fall outside the null envelopes. However, we have discarded all locations were no fish were caught so we might prefer to fit a model that explicitly accounts for this.

2.2 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.

2.2.1 The Model structure

- **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 $\{\rho^{(1)}, \rho^{(2)}, \tau_d^{(1)}, \tau_d^{(2)}\}$

2.2.2 The workflow

When fitting a geostatistical model we need to fulfill the following tasks:

1. Build the mesh
2. Define the SPDE representation of the spatial GF. This includes defining the priors for the range and sd of the spatial GF
3. Define the *components* of the linear predictor. This includes the spatial GF and all eventual covariates
4. Define the observation model using the `bru_obs()` function
5. Run the model using the `bru()` function

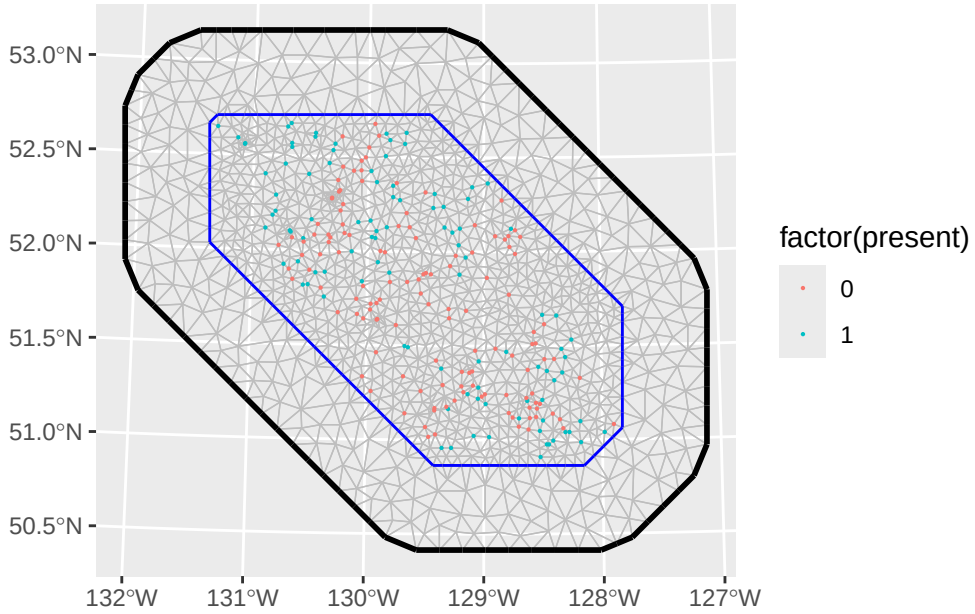
1. Building the mesh

The first task, when dealing with geostatistical models is to build the mesh that covers the area of interest. For this purpose we use the function `fm_mesh_2d`.

One way to build the mesh is to start from the locations where we have observations, these are contained in the dataset `pcod_sf`.

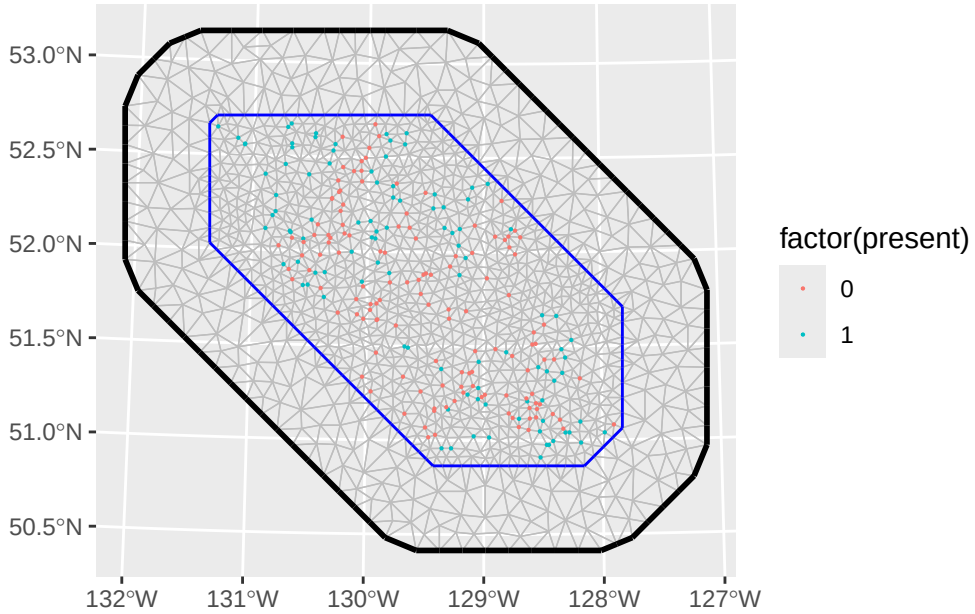
```
mesh = fm_mesh_2d(loc = pcod_sf,            # Build the mesh
                  max.edge = c(10,20),      # The largest allowed triangle edge length.
```

```
                  offset = c(5,50))       # The automatic extension distance
ggplot() + gg(mesh) + geom_sf(data= pcod_sf, aes(color = factor(present)), size = 0.1) + xla
```



As you can see from the plot above, some of the locations are very close to each other, this causes some very small triangles. This can be avoided using the option `cutoff =` which collapses the locations that are closer than a cutoff (those points are collapsed in the mesh construction but, of course, not when it come to estimaation.)

```
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
ggplot() + gg(mesh) + geom_sf(data= pcod_sf, aes(color = factor(present)), size = 0.1) + xla
```



Task

Look at the documentation for the `fm_mesh_2d` function typing

```
?fm_mesh_2d
```

play around with the different options and create different meshes.
The *rule of thumb* is that your mesh should be:

- fine enough to well represent the spatial variability of your process, but not too fine in order to avoid computation burden
- the triangles should be regular, avoid long and thin triangles.
- The mesh should contain a buffer around your area of interest (this is what is defined in the `offset` option) in order to avoid boundary artefact in the estimated variance.

2. Define the SPDE representation of the spatial GF

To define the SPDE representation of the spatial GF we use the function `inla.spde2.pcmatern`. This takes as input the mesh we have defined and the PC-priors definition for ρ and σ (the range and the marginal standard deviation of the field).

PC priors Gaussian Random field are defined in (Fuglstad et al. 2018). From a practical perspective for the range ρ you need to define two paramters ρ_0 and p_ρ such that you believe it is reasonable that

$$P(\rho < \rho_0) = p_\rho$$

while for the marginal variance σ you need to define two parameters σ_0 and p_σ such that you believe it is reasonable that

$$P(\sigma < \sigma_0) = p_\sigma$$

Here are some alternatives for defining priors for our model

```
spde_model1 =  inla.spde2.pcmatern(mesh,
                                   prior.sigma = c(.1, 0.5),
                                   prior.range = c(30, 0.5))
spde_model2 =  inla.spde2.pcmatern(mesh,
                                   prior.sigma = c(10, 0.5),
                                   prior.range = c(1000, 0.5))
spde_model3 =  inla.spde2.pcmatern(mesh,
                                   prior.sigma = c(1, 0.5),
                                   prior.range = c(100, 0.5))
```

Question

Consider the `pcod_sf`, the spatial extension and type of the data...is some of the previous choices more reasonable than other?

- (A) spde_model1
- (B) spde_model2

- (C) spde_model3

NOTE Remember that a prior should be reasonable..but the model should not totally depend on it.

3. Define the components of the linear predictor

We have now defined a mesh and a SPDE representation of the spatial GF. We now need to define the model components:

```
cmp <- ~
   Intercept_biomass(1) +
    depth_biomass(depth_scaled, model = "linear") +
    depth2_biomass(depth_scaled2, model = "linear") +
    space_biomass(geometry, model = spde_model3) +
    Intercept_caught(1) +
    depth_caught(depth_scaled, model = "linear") +
    depth2_caught(depth_scaled2, model = "linear") +
    space_caught(geometry, model = spde_model3)
```

NOTE since the dataframe we use (`pcod_sf`) is an `sf` object the input in the `space()` component is the geometry of the dataset.

4. Define the observation model

Since we have two likelihoods we need to define two observational models

```
fml_1 = density ~  Intercept_biomass + depth_biomass + depth2_biomass + space_biomass
fml_2 = present ~ Intercept_caught + depth_caught + depth2_caught + space_caught

biomass_obs <- bru_obs(formula = fml_1,
      family = "lognormal",
      data = pcod_sf  %>% filter(density>0)) # restrict to those locations where fish were c

presence_obs <- bru_obs(formula = fml_2 ,
  family = "binomial",
  data = pcod_sf,
)
```

5. Run the model & Extract results

Now we fit the model and produce some summaries:

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

```
summary(fit_hurdle)
```

```
inlabru version: 2.13.0.9016
INLA version: 25.08.21-1
Components:
Latent components:
Intercept_biomass: main = linear(1)
depth_biomass: main = linear(depth_scaled)
depth2_biomass: main = linear(depth_scaled2)
space_biomass: main = spde(geometry)
Intercept_caught: main = linear(1)
depth_caught: main = linear(depth_scaled)
depth2_caught: main = linear(depth_scaled2)
space_caught: main = spde(geometry)
Observation models:
  Family: 'lognormal'
    Tag: <No tag>
    Data class: 'sf', 'tbl_df', 'tbl', 'data.frame'
    Response class: 'numeric'
    Predictor:
        density ~ Intercept_biomass + depth_biomass + depth2_biomass +
            space_biomass
    Additive/Linear: TRUE/TRUE
    Used components: effects[Intercept_biomass, depth_biomass, depth2_biomass, space_biomass],
  Family: 'binomial'
    Tag: <No tag>
    Data class: 'sf', 'tbl_df', 'tbl', 'data.frame'
    Response class: 'numeric'
    Predictor: present ~ Intercept_caught + depth_caught + depth2_caught + space_caught
    Additive/Linear: TRUE/TRUE
    Used components: effects[Intercept_caught, depth_caught, depth2_caught, space_caught], lat
Time used:
    Pre = 1.59, Running = 9.22, Post = 0.842, Total = 11.7
Fixed effects:
                    mean    sd 0.025quant 0.5quant 0.975quant   mode   kld
Intercept_biomass  3.498 0.332      2.885    3.482      4.227  3.435 0.000
depth_biomass     -0.509 0.294     -1.098   -0.505      0.058 -0.505 0.000
depth2_biomass    -0.339 0.228     -0.789   -0.339      0.106 -0.339 0.000
Intercept_caught   1.761 2.370     -3.217    1.607      7.270  1.265 0.003
depth_caught      -2.564 0.492     -3.625   -2.533     -1.693 -2.460 0.000
depth2_caught     -1.494 0.305     -2.143   -1.478     -0.944 -1.433 0.000

Random effects:
  Name    Model
    space_biomass SPDE2 model
   space_caught SPDE2 model

Model hyperparameters:
                                          mean     sd 0.025quant 0.5quant
Precision for the lognormal observations  1.15  0.550      0.465    1.028
Range for space_biomass                  36.65 25.890      6.893   30.135
Stdev for space_biomass                   1.00  0.278      0.571    0.962
Range for space_caught                  158.39 91.057     56.964  135.571
```

```
Stdev for space_caught                      2.20  0.661      1.192    2.105
                                          0.975quant    mode
Precision for the lognormal observations        2.57   0.827
Range for space_biomass                       103.69  18.520
Stdev for space_biomass                         1.66   0.888
Range for space_caught                        398.96 101.343
Stdev for space_caught                          3.77   1.924

Marginal log-Likelihood:  -657.88
 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)')
```

2.2.3 Model Predictions:

We now want to extract the estimated posterior mean and sd of spatial GF. To do this we first need to define a grid of points where we want to predict:

```
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))
```

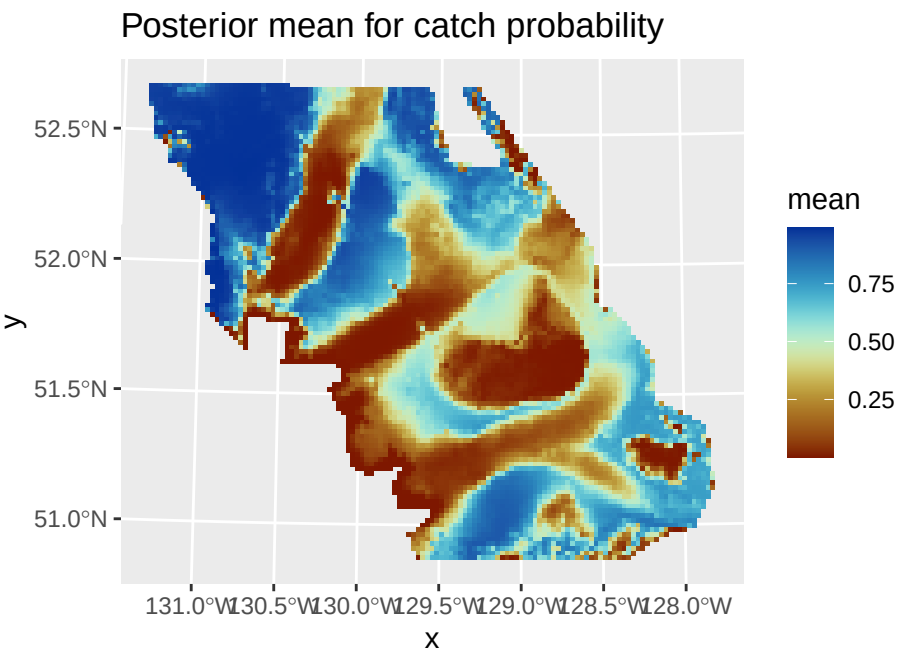
then compute the prediction for both the spatial GFs and both linear predictors (note that we use the `plogis` function to map $\eta_i^{(1)}$ to $\pi_i^{(i)}$). To compute the expected log-biomass density we need:

- $\pi_i^{(i)}$ = Catching probability
- $\mathbb{E}[Z(s)|Y(s)] = \exp\left(\mu(s) + \frac{1}{2\tau_e}\right)$ = conditional mean
- $\mathbb{E}(Z(s)) = \pi(s) \times \mathbb{E}[Z(s)|Y(s)]$ = unconditional mean

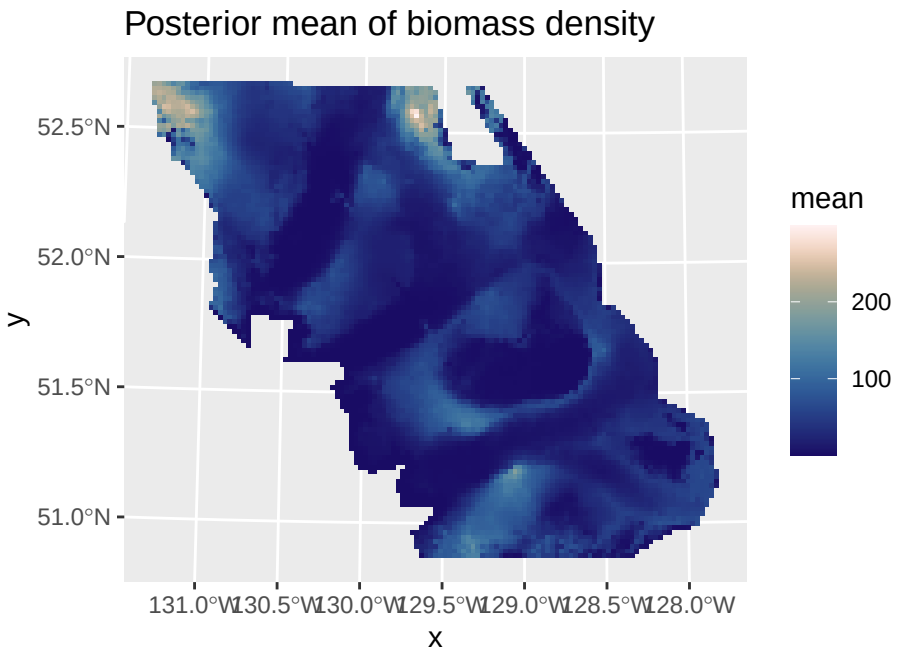
```
pred <- predict( fit_hurdle , pxl1,
  ~ {
    pi <- plogis(Intercept_caught + depth_caught + depth2_caught + space_caught)  # catching
    mu_log  <-  Intercept_biomass + depth_biomass + depth2_biomass + space_biomass
    sd <- sqrt(1/Precision_for_the_lognormal_observations)
    conditional_mean <- 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)
```

Finally, we can plot the maps

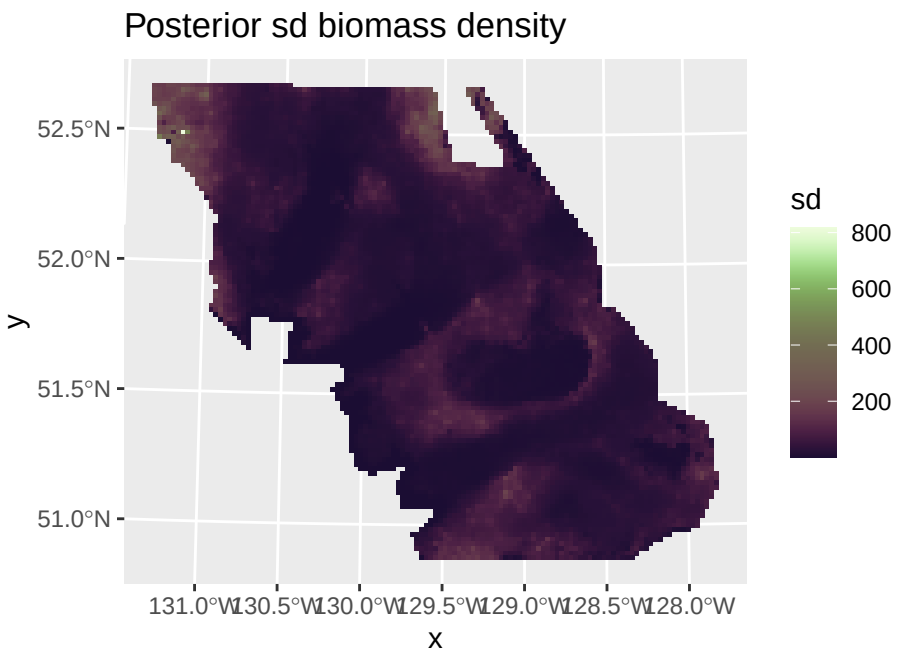
```
ggplot() + gg(pred$pi, geom = "tile",aes(fill = mean)) + scale_fill_scico(palette = "roma")
```




```
ggplot() + gg(pred$dens, geom = "tile",aes(fill = mean))+ scale_fill_scico(palette="lapaz")
```




```
ggplot() + gg(pred$dens, geom = "tile",aes(fill = sd)) + scale_fill_scico(palette = "tokyo")
```



Note The posterior sd is lowest at the observation points. Note how the posterior sd is inflated around the border, this is the "border effect" due to the SPDE representation.

3 Point Process Modelling

3.1 Gorilla Nesting Sites

In this example we will fit a point process model to analyse the locations of nesting sites of gorillas, and associated covariates, in a National Park in Cameroon.

The data set is available from the `inlabru` R package. The data come from a study of gorillas in the Kagwene Gorilla Sanctuary, Cameroon, by the Wildlife Conservation Society Takamanda-Mone Landscape Project (WCS-TMLP). The dataset contains the spatial locations of 647 nesting sites of gorilla groups observed in the sanctuary over time. Locations

are given as UTM (Zone 32N) coordinates in metres. The observation window is the boundary of the sanctuary, represented as a polygon.

First lets load the data:

```
data(gorillas_sf, package = "inlabru")
```

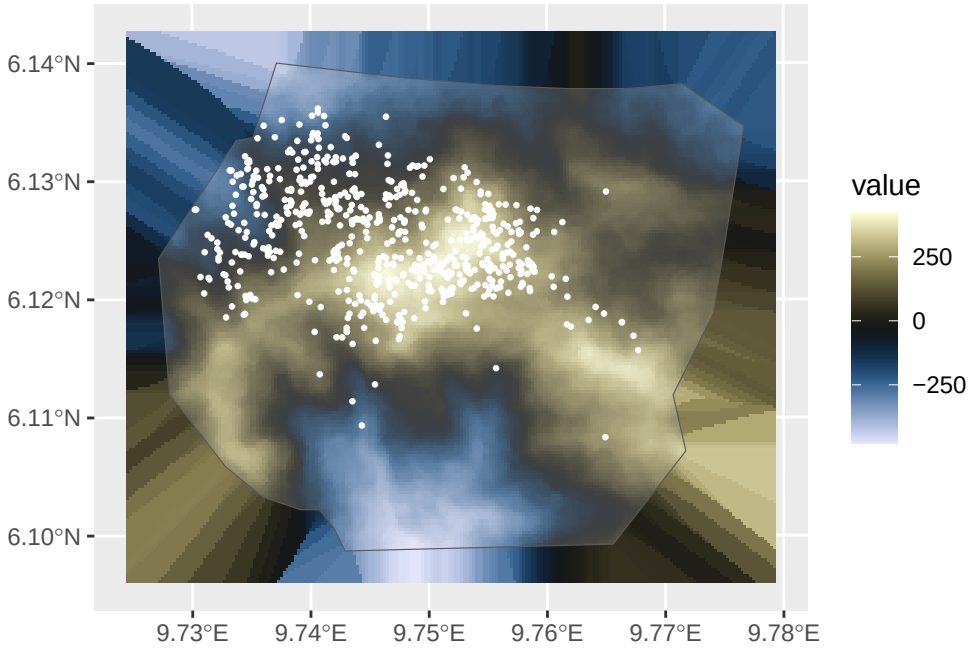
Here we will extract the information about:

- The spatial location of the nests (`nests`)
- The mesh for building our spatial model (`mesh`)
- The observational window (`boundary`)
- The Digital elevation of terrain, in metres (`elev`).

```
nests <- gorillas_sf$nests
mesh <- gorillas_sf$mesh
boundary <- gorillas_sf$boundary
gcov <- gorillas_sf_gcov()
elev <- gcov$elevation
elev <- elev - mean(terra::values(elev), na.rm = TRUE) #scale the  covariate
```

We can visualize the Data as follows:

```
ggplot() +
  gg(elev) +
  gg(boundary, alpha = 0.2) +
  gg(nests, color = "white", cex = 0.5)+
  scale_fill_scico(palette = "lisbon")
```



We can see that the elevation covariate has been extended beyond the region of interest. Recall that this is need for computational purposes as we need the covariate to be available at the quadrature locations for approximating our point process model.

3.2 The Model

We are now going to use the scaled-elevation as a covariate in addition to a GF to explain the variability of the intensity $\lambda(s)$ over the domain of interest.

Our model is

$$\log \lambda(s) = \beta_0 + \beta_1 x(s) + \xi(s)$$

where $x(s)$ is the altitude at location s and $\xi(s)$ is a GF

The likelihood becomes:

$$\begin{aligned} p(\mathbf{y}|\lambda) &\propto \exp\left(-\int_\Omega \lambda(\mathbf{s})d\mathbf{s}\right)\prod_{i=1}^{n}\lambda(\mathbf{s}_i) \\ &= \exp\left(-\int_\Omega \exp(\beta_0+\beta_1 x(s))d\mathbf{s}\right)\prod_{i=1}^{n}\lambda(\mathbf{s}_i) \end{aligned}$$

where $|\Omega|$ is the area of the domain of interest.

We need to approximate the integral using a numerical integration scheme as:

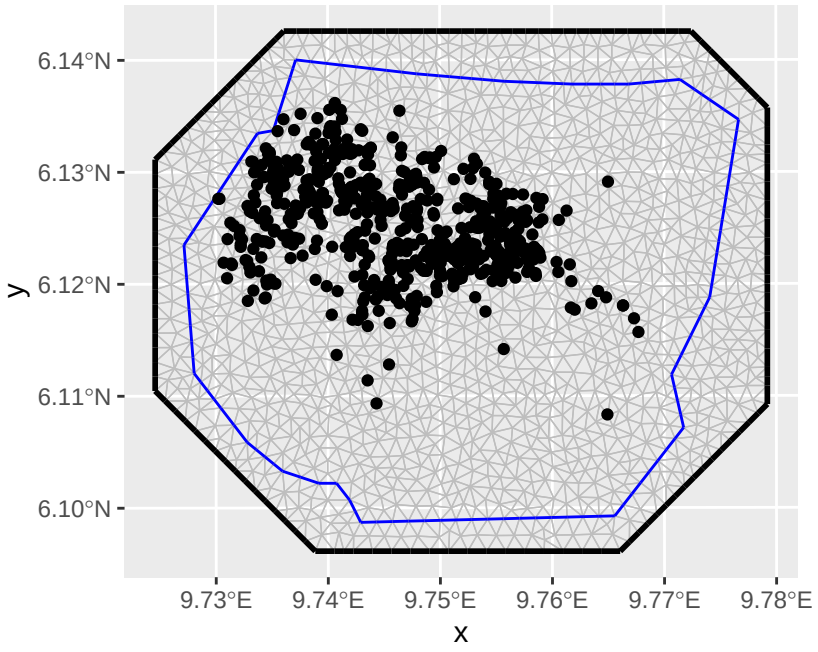
$$\approx \exp\left(-\sum_{k=1}^{N_k} w_k \lambda(s_k)\right)\prod_{i=1}^{n}\lambda(\mathbf{s}_i)$$

Where N_k is the number of integration points $s_1, \ldots, s_{N_k}$ and $w_1, \ldots, w_{N_k}$ are the integration weights. In addition we also need to approximate the GF - we can use the SPDE approach as before to achieve this.

The first step, as any time we use the SPDE approach is to define the mesh and the priors for the marginal variance and range:

```
mesh <- gorillas_sf$mesh

ggplot() + gg(mesh) + geom_sf(data = nests)
```

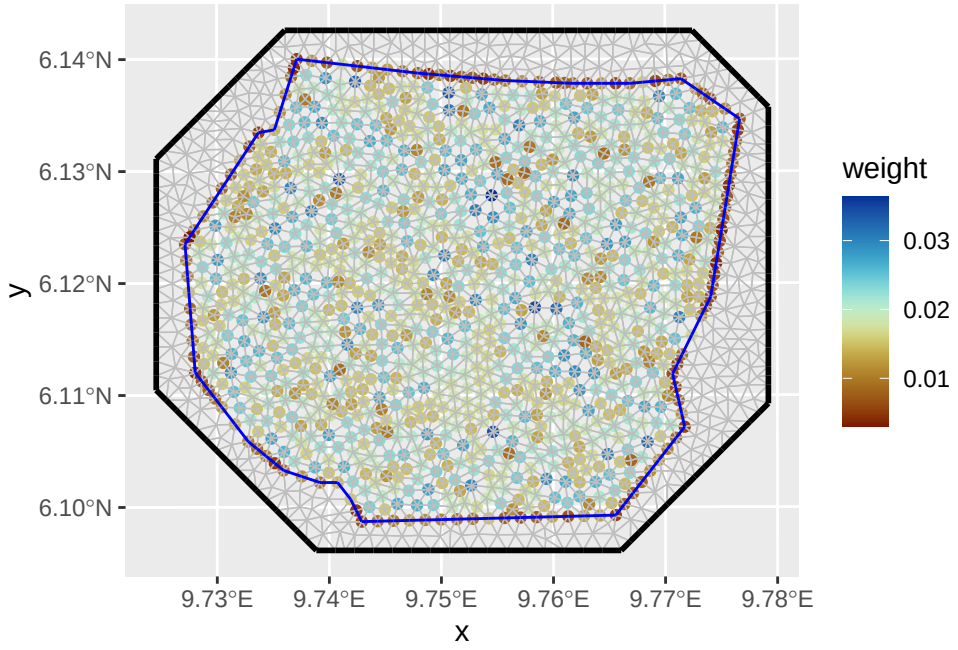


```
spde_model =  inla.spde2.pcmatern(mesh,
  prior.sigma = c(0.1, 0.01),
  prior.range = c(0.1, 0.01)
)
```

We can then define the integration weights. Here we use the same points to `define the SPDE approximation` and to `approximate the integral` in the likelihood (note that it does not have to be like this, BUT integration weight and SPDE weights have to be consistent with each other)

```
ips = fm_int(mesh, samplers = boundary)

ggplot() + geom_sf(data = ips, aes(color = weight)) +
  gg(mesh) +
   scale_color_scico(palette = "roma")
```



3.3 Fitting the Model

The model has three components: intercept, linear effect of altitude and the spatial GRF

```
cmp = ~ Intercept(1) + space(geometry, model = spde_model) + elev(elev, model = "linear")

formula = geometry ~ Intercept + space + elev


lik = bru_obs("cp",
              formula = formula,
              data = nests,
              ips = ips)

fit_gorillas = bru(cmp, lik)
```

```
summary(fit_gorillas)
```

```
inlabru version: 2.13.0.9016
INLA version: 25.08.21-1
Components:
Latent components:
Intercept: main = linear(1)
space: main = spde(geometry)
elev: main = linear(elev)
Observation models:
  Family: 'cp'
    Tag: <No tag>
    Data class: 'sf', 'tbl_df', 'tbl', 'data.frame'
    Response class: 'numeric'
    Predictor: geometry ~ Intercept + space + elev
    Additive/Linear: TRUE/TRUE
    Used components: effects[Intercept, space, elev], latent[]
Time used:
    Pre = 1.1, Running = 9.34, Post = 0.526, Total = 11
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant  mode kld
Intercept 1.127 0.478      0.154    1.137      2.038 1.137   0
elev      0.004 0.001      0.002    0.004      0.006 0.004   0

Random effects:
  Name    Model
    space SPDE2 model

Model hyperparameters:
                mean    sd 0.025quant 0.5quant 0.975quant  mode
Range for space 1.76 0.217      1.376     1.75       2.23 1.713
Stdev for space 1.00 0.085      0.848     1.00       1.18 0.995

Marginal log-Likelihood:  -1254.95
 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)')
```

3.4 Model Predictions

The predict function of `inlabru` can take as argument an sf objects. So, we can use the `inlabru` function `fm_pixels` to generate an sf object with points only within the boundary, using its `mask` argument, as shown below:

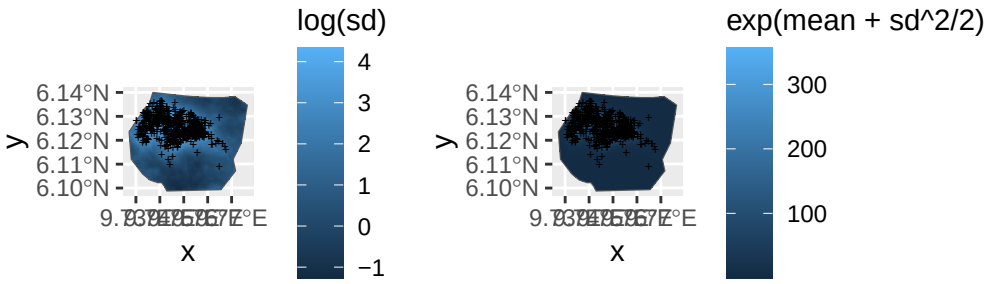
```
pred.df <- fm_pixels(mesh, mask = boundary)
```

Then, we simply supply the intensity and log intensity :

```
e.pred <- predict(
  fit_gorillas,
  pred.df,
```

```
  ~ list(
    int = exp(space + elev + Intercept),
    int.log = space + elev + Intercept
  )
)

p1 <- ggplot() +
  gg(e.pred$int, aes(fill = log(sd)), geom = "tile") +
  gg(boundary, alpha = 0) +
  gg(nests, shape = "+")
p2 <- ggplot() +
  gg(e.pred$int.log, aes(fill = exp(mean + sd^2 / 2)), geom = "tile") +
  gg(boundary, alpha = 0) +
  gg(nests, shape = "+")
(p1 | p2)
```



3.4.1 Estimate abundance

Finally, we want to use the fitted model to estimate the total number of nesting sites in the whole region. To do this, we first have to define the expected number of sites as:

$$E(N_\Omega) = \int_\Omega \exp(\lambda(s))\, ds$$

Then simulate a realization of N_Ω to include also the likelihood variability in our estimate. To do so, we pass both the fitted model and the integration weights to the predict function. Then, we specify a reasonable range for where posterior poisson density is defined:

```
Nest.e <- predict(
  fit_gorillas,
  fm_int(mesh, boundary),
  ~ data.frame(
```

```
    N = 400:900,
    density = dpois(400:900,
      lambda = sum(weight * exp(space + elev + Intercept))
    )
  ),
  n.samples = 2000
)
```

We can plot the posterior density of the abundance as follows:

```
ggplot(data = Nest.e) +
  geom_line(aes(x = N, y = mean, colour = "Posterior")) +
    geom_vline(xintercept = nrow(nests),
               colour = "red") +
  xlab(expression(Lambda))
```