Session 7: Spatio-temporal models with INLA-SPDE

Learning Objectives

At the end of this session you should be able to:

- know the definition of spatio-temporal process in the geostatistics framework;
- develop spatio-temporal models for geostatistics data;
- use `R-INLA` for implementing a (separable) space-time geostatistical models.

The topics covered in this lecture can be found in:

- Chapter 10 of the book **Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny** https://www.paulamoraga.com/book-geospatial/index.html
- Section 7.2 of the book **Spatio-Temporal Bayesian Models with** `R-INLA` https://github.com/michelacameletti/Spatial-and-Spatio-Temporal-Bayesian-Models-with-R-INLA
- Section 7.1 of the book **Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA** https://becarioprecario.bitbucket.io/spde-gitbook/index.html

Outline

1. Spatio-temporal processes + a space-time hierarchical model for air pollution

2. Implementation of a spatio-temporal model using `R-INLA`

Spatio-temporal processes + a space-time hierarchical model for air pollution

Spatio-temporal processes

- The concept of spatial process can be extended to the spatio-temporal case including a time dimension. The data are then defined by a process $\{y(s,t),(s,t)\in\mathcal{D}\subset\mathbb{R}^2\times\mathbb{R}\}$ and are observed at n spatial locations and at T time points.

- When spatio-temporal geostatistical data are considered, we need to define a valid **spatio-temporal covariance function** given by

$$\text{Cov}\left(y(\boldsymbol{s}_i,t),y(\boldsymbol{s}_j,u)\right)=\mathcal{C}(y_{it},y_{ju})$$

- If we assume **stationarity in space and time**, the space-time covariance function can be written as a function of the spatial Euclidean distance $\Delta_{ij}=||\boldsymbol{s}_i-\boldsymbol{s}_j||$ and of the temporal lag $\Lambda_{tu}=|t-u|$ so that $\text{Cov}\left(y_{it},y_{ju}\right)=\mathcal{C}(\Delta_{ij},\Lambda_{tu})$.

- If we assume **separability** the stationary space-time covariance function is decomposed into the product of a purely spatial and a purely temporal term:

$$\text{Cov}\left(y_{it},y_{ju}\right)=\mathcal{C}_1(\Delta_{ij})\mathcal{C}_2(\Lambda_{tu})$$

Hierarchical spatio-temporal model for PM concentrations [1]

- We present a spatio-temporal model for fine air particulate matter (PM2.5; particles less than 2.5 micrometers in diameter) measured yearly in Spain.
- The spatio-temporal model was firstly proposed by Cameletti, Lindgren, Simpson, and Rue (2013), while the data are from Moraga (2019).
- These data have been collected over the years 2015 to 2017 from a set of sparse ground monitors.
- The objective of our analysis is to predict expected air pollution concentrations at arbitrary space-time locations from the observed data. This will allow us to construct high-resolution maps of air pollution for each year.
- The hierarchical spatio-temporal model that we use to reach this objective is a separable model, where the space-time covariance structure can be decomposed into a spatial and a temporal term, i.e., the spatio-temporal covariance can be expressed as a Kronecker product of a spatial and a temporal covariance.

Hierarchical spatio-temporal model for PM concentrations [2]

- We denote by y_{it} the PM2.5 concentrations measured at site $\boldsymbol{s}_i$, with $i = 1, \dots, n$, and year $t = 1, \dots, T$.
- The following distribution is assumed for the observations:

$$y_{it} \sim \text{Normal}(\eta_{it}, \sigma_e^2)$$

  where σ_e^2 is the variance of the measurement error defined by a Gaussian white-noise process, both serially and spatially uncorrelated.

- The linear predictor is given by

$$\eta_{it} = b_0 + \omega_{it}$$

  where b_0 is the intercept.

Hierarchical spatio-temporal model for PM concentrations [3]

- The term ω_{it} refers to the **latent spatio-temporal process** (i.e. the true unobserved level of pollution), which changes in time with first order autoregressive dynamics and spatially correlated innovations:

$$\omega_{it} = \rho\omega_{i(t-1)} + \xi_{it}$$

with $t = 2, \ldots, T$, $|\rho| < 1$, $\omega_{i1} \sim \text{Normal}\left(0, \sigma^2/(1-\rho^2)\right)$.

- The term ξ_{it} is a zero-mean **Gaussian field**, assumed to be **temporally independent** and characterized by the following spatio-temporal covariance function:

$$\text{Cov}\left(\xi_{it}, \xi_{ju}\right) = \begin{cases} 0 & \text{if} \quad t \neq u \\ \text{Cov}(\xi_i, \xi_j) & \text{if} \quad t = u \end{cases}$$

for $i \neq j$, where $\text{Cov}(\xi_i, \xi_j)$ is given by Matérn spatial covariance function.

- This model is characterized by a **separable spatio-temporal covariance** as it can be rewritten as the product of a purely spatial and a purely temporal covariance function (see Cameletti, Ignaccolo, and Bande (2011)).

Hierarchical spatio-temporal model for PM concentrations [4]

- For each time point $\boldsymbol{\xi}_t \sim \mathrm{Normal}(\mathbf{0}, \boldsymbol{\Sigma})$ and through the SPDE approach

$$\boldsymbol{\xi}_t \rightarrow \tilde{\boldsymbol{\xi}}_t \sim \mathrm{Normal}(\mathbf{0}, \boldsymbol{Q}_S^{-1})$$

  where the precision matrix $\boldsymbol{Q}_S$ comes from the SPDE representation. The matrix $\boldsymbol{Q}_S$ does not change in time - due to the serial independence hypothesis - and its dimension is given by the number of vertices of the domain triangulation.

- The joint distribution of the Tn-dimensional GMRF $\boldsymbol{\omega} = (\boldsymbol{\omega}_1', \ldots, \boldsymbol{\omega}_T')'$ is

$$\boldsymbol{\omega} \sim \mathrm{Normal}(\mathbf{0}, \boldsymbol{Q}^{-1})$$

  with $\boldsymbol{Q} = \boldsymbol{Q}_T \otimes \boldsymbol{Q}_S$, where $\otimes$ denotes the Kronecker product and $\boldsymbol{Q}_T$ is the T-dimensional precision matrix of the AR(1) process.

- For the considered model the latent process is given by $\boldsymbol{\theta} = \{\boldsymbol{\omega}, b_0\}$ while the hyperparameter vector is $\psi = (\sigma_e^2, \rho, \sigma^2, r)$.

Implementation of a spatio-temporal process using `R-INLA`

Particle matter data

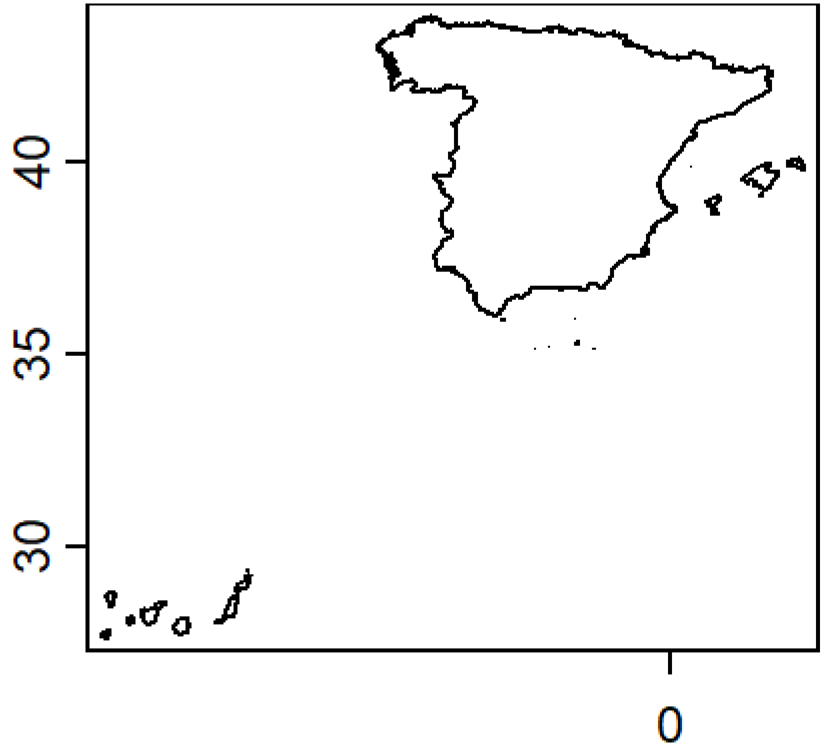
- The data used to demonstrate the modelling framework are **PM2.5 concentrations** measured at several monitoring stations in Spain over the years 2015 to 2017 from the European Environment Agency. This example is taken from Moraga (2019), chapter 10.

```
> library(tidyverse)
> library(INLA)
>
> df = read.csv("data/dataPM25.csv")
> df = df[, c(
+   "ReportingYear", "StationLocalId",
+   "SamplingPoint_Longitude",
+   "SamplingPoint_Latitude",
+   "AQValue"
+ )]
> names(df) = c("year", "id", "long", "lat", "value")
```

Boundary

```
> library(lwgeom)
> library(geodata)
> library(sf)
> library(ggplot2)
> library(dplyr)
> spain = geodata::gadm(country="Spain",
+                        level=0, path=tempdir())
> class(spain)
```

```
[1] "SpatVector"
attr(,"package")
[1] "terra"
```



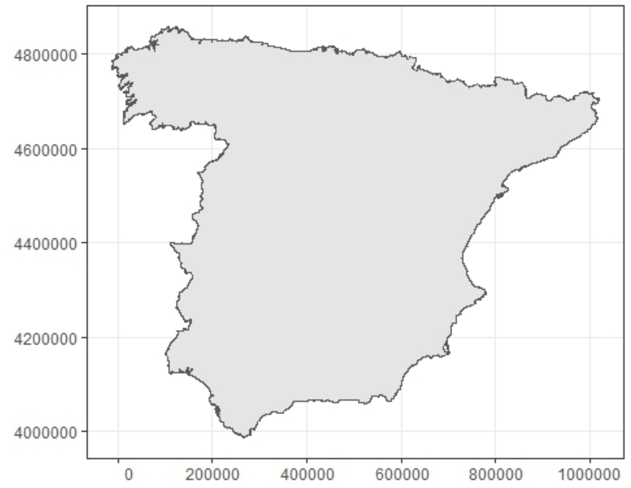
Domain of the analysis [1]

- We are interested in predicting fine air particle in the main territory of Spain, and therefore we need to remove the islands from the map.
- To do so, we keep the polygon of the map that has the largest area using the packages `sf` and `dplyr`.
- In detail, we convert the object `spain`, which is a `terra` object, to an `sf` object, we cast geometry to polygon, and we calculate the areas of the polygons of the object, and keep the polygon with the largest area.
- Additionally, we transform our map to an object with Universal Transverse Mercator (UTM) projection to work with meters or kilometers. To do this, we use the `st_transform()` function specifying the ESPG code of Spain (code 25830) which corresponds to UTM zone 30 North.

```
> spain = spain %>%
+   st_as_sf() %>% # terra vector to sf
+   st_cast("POLYGON") %>% # convert to polygon
+   mutate(area = st_area(.)) %>% # Extract geometric info (area)
+   arrange(desc(area)) %>%
+   slice(1) # slices the data by row index
>
> spain = spain %>% st_transform(25830)
```

Domain of the analysis [2]

```
> ggplot(spain) + geom_sf() +
+   theme_bw() +
+   coord_sf(datum = st_crs(spain)) +
+   labs(x = "", y = "")
```



Data preparation

- We now project the data which uses geographical coordinates (longitude and latitude) to UTM projection.
- We create an `sf` object with the longitude and latitude values of the monitoring stations, and set the CRS to EPSG 4326 which corresponds to geographical coordinates. Then we use `st_transform()` to project the data to EPSG code 25830 and we include such coordinates in our PM2.5 data set

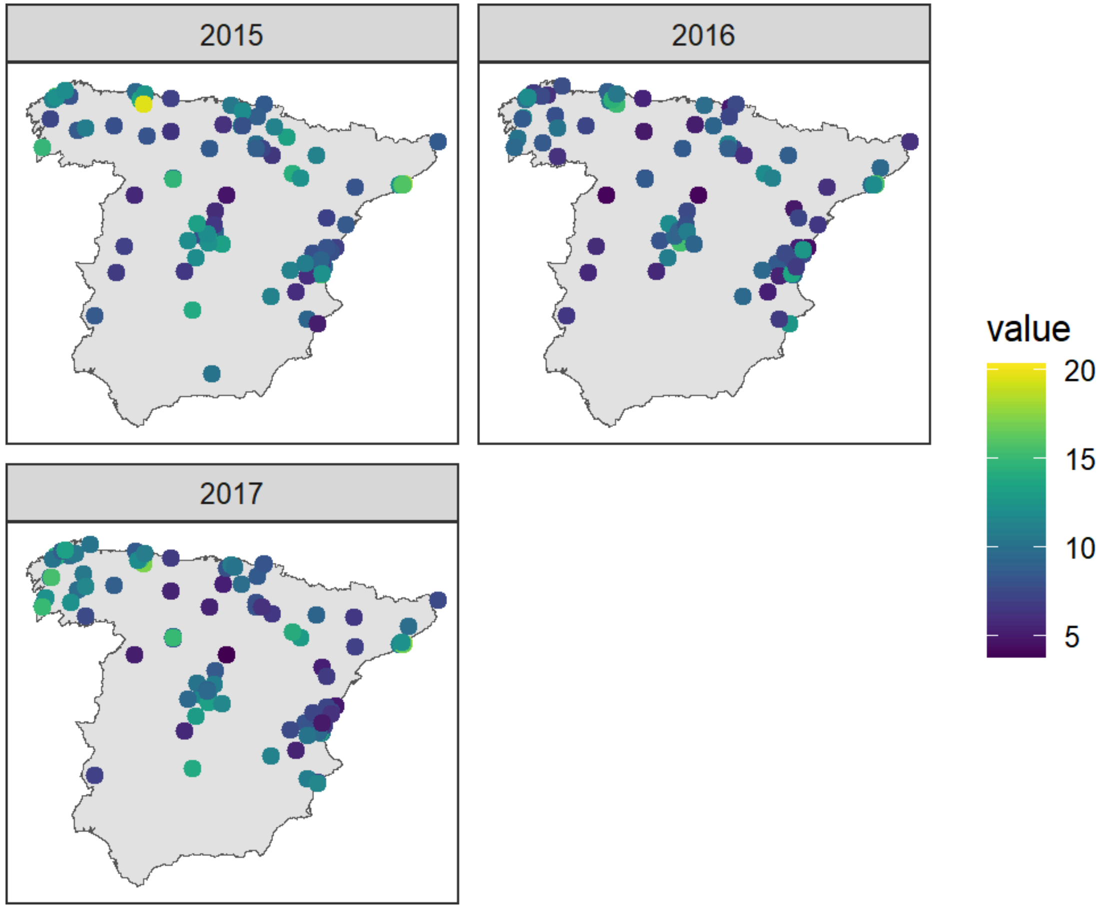
```
> # from lon lat to UTM
> geo.sf = st_as_sf(data.frame(long = df$long, lat = df$lat),
+                   coords = c("long", "lat"))
> st_crs(geo.sf) = st_crs(4326)
> geo.sf = geo.sf %>% st_transform(25830)
> df[, c("x", "y")] = st_coordinates(geo.sf)
>
> # keep station in the main territory of Spain
> ind = st_intersects(spain, geo.sf)
> df = df[ind[[1]], ]
> head(df)[1:5,]
```

```
  year        id      long      lat     value        x       y
1 2015 STA_ES1938A -3.690278 40.43972 11.022667 441457.6 4476793
3 2015 STA_ES0691A  2.204523 41.40388 17.963018 935101.1 4596682
4 2015 STA_ES1417A -0.403890 42.13611 11.332180 714552.3 4668151
5 2015 STA_ES1649A -1.744000 42.17600  6.366621 603732.6 4670081
6 2015 STA_ES1997A -6.147220 40.07778  7.034714 231636.0 4441138
```

Map of particle data

We plot PM2.5 concentrations measured at the monitoring stations by year

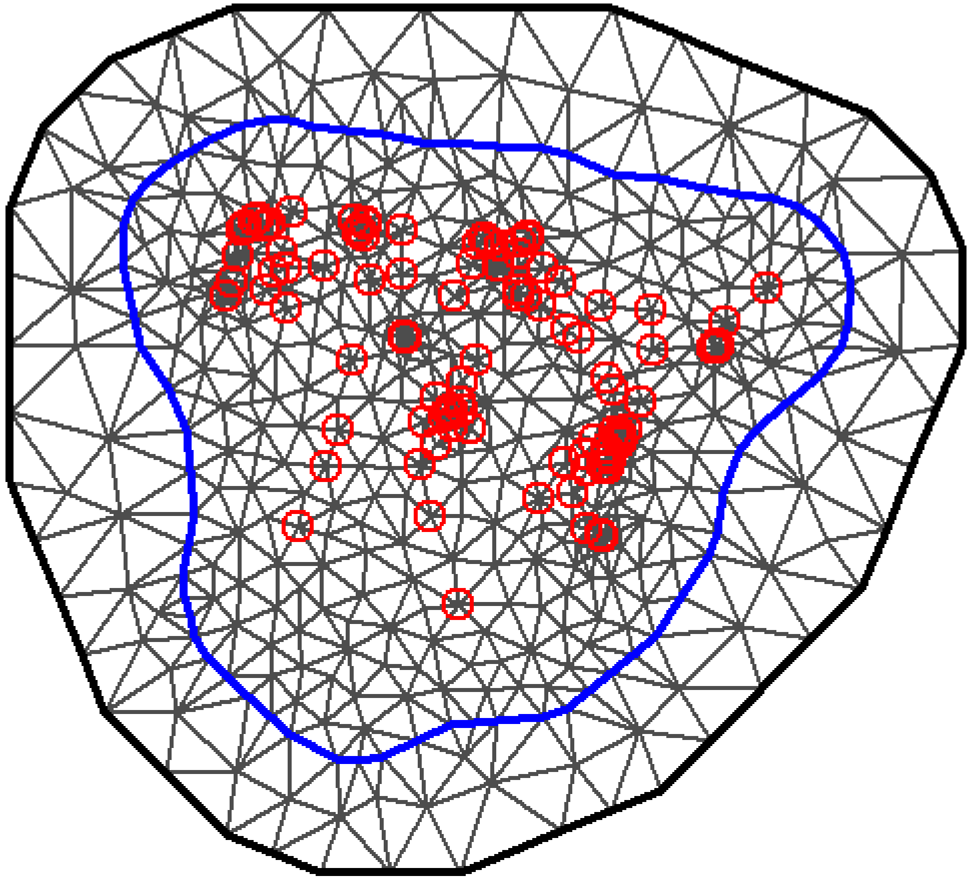
```
> library(viridis)
>
> ggplot(spain) + geom_sf() +
+   coord_sf(datum = NA) +
+   geom_point(
+   data = df, aes(x = x, y = y, color = value),
+   size = 2) +
+   labs(x = "", y = "") +
+   scale_color_viridis() +
+   facet_wrap(~year, ncol = 2, nrow = 2) +
+   theme_bw()
```



Create the mesh and the SPDE model

We obtain a triangulation of the domain using the function `inla.mesh.2d`

```
> sc = 1/1000 ## scaling (so we go from m to Km)
> coo = cbind(df$x, df$y)*sc
> bnd = inla.nonconvex.hull(
+   st_coordinates(spain)[, 1:2]*sc)
>
> mesh = inla.mesh.2d(
+   loc = coo, boundary = bnd,
+   max.edge = c(100000, 200000)*sc,
+   cutoff = 1000*sc)
```



Then we create the SPDE model using the `inla.spde2.matern` or `inla.spde2.pcmatern` functions. Here we use `inla.spde2.pcmatern` that adopts PC-priors for the model **parameters range and marginal standard deviation.**

```
> spde = inla.spde2.pcmatern(mesh = mesh,
+  prior.range = c(7, 0.01), # P(range<7)=0.01
+  prior.sigma = c(1, 0.01)) # P(sigma>1)=0.01
> spde$n.spde #n. of mesh vertices
```

Index set

- The function `inla.spde.make.index`, generates vectors of indices for the spatial and temporal components of the model. We specify the name of the effect and the number of vertices in the SPDE model (`spde$n.spde`)

```
> n_years = length(unique(df$year))
>
> indexs = inla.spde.make.index("spatial.field",
+                               n.spde = spde$n.spde,
+                               n.group = n_years
+ )
> names(indexs)
```

```
[1] "spatial.field"       "spatial.field.group" "spatial.field.repl"
```

where:

– `spatial.field`: indices of the SPDE vertices repeated the number of times,

– `spatial.field.group`: indices of the times repeated the number of mesh vertices,

– `spatial.field.repl`: vector of 1s with length given by the number of mesh vertices times the number of times (`spde$n.spde*n_years`; because we have a single replication of the spatial process at each time point).

A matrix for the estimation part

- We construct an observation matrix that extracts the values of the spatio-temporal field at the measurement locations and time points used for the parameter estimation. This is the projection matrix A, that is used to project the Gaussian random field from the observations to the triangulation vertices
- The projection matrix is defined using the coordinates of the observed data. In order to construct a Kronecker product model in `R-INLA`, we use the `group` feature in the `inla.spde.make.A`

```
> group = df$year - min(df$year) + 1
> A_est = inla.spde.make.A(mesh = mesh, loc = coo, group = group)
> dim(A_est)
```

```
[1]  299 2082
```

This is a matrix equal in dimension to (number of observations) $\times$ (number of indices) of our basis functions in space and time:

```
> nrow(df)
```

```
[1] 299
```

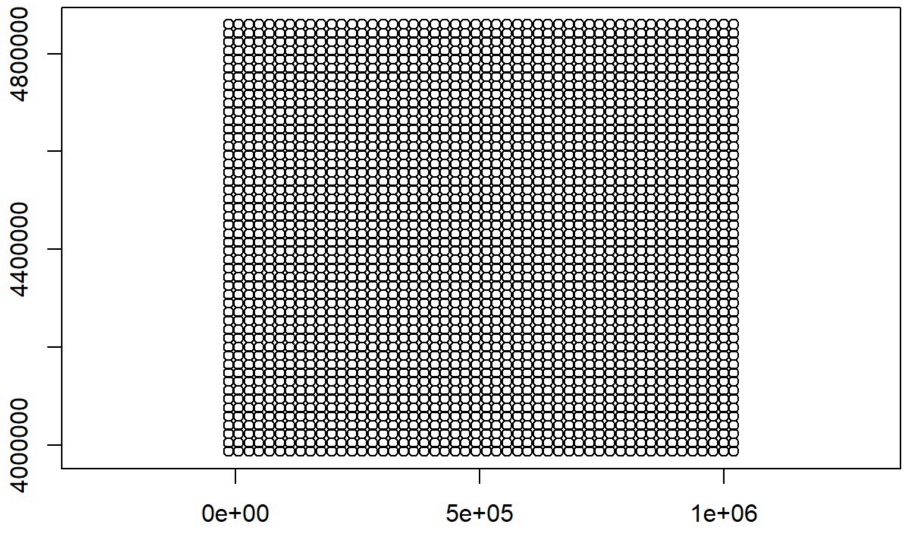
```
> length(indexs$spatial.field)
```

```
[1] 2082
```

Grid locations for predictions [1]

- We want to calculate predictions of the expected particle concentrations for the entire Spain.
- To do so, we need to create a **spatial grid** upon which we map the predictions. Here, we create an 50 $\times$ 50 grid.

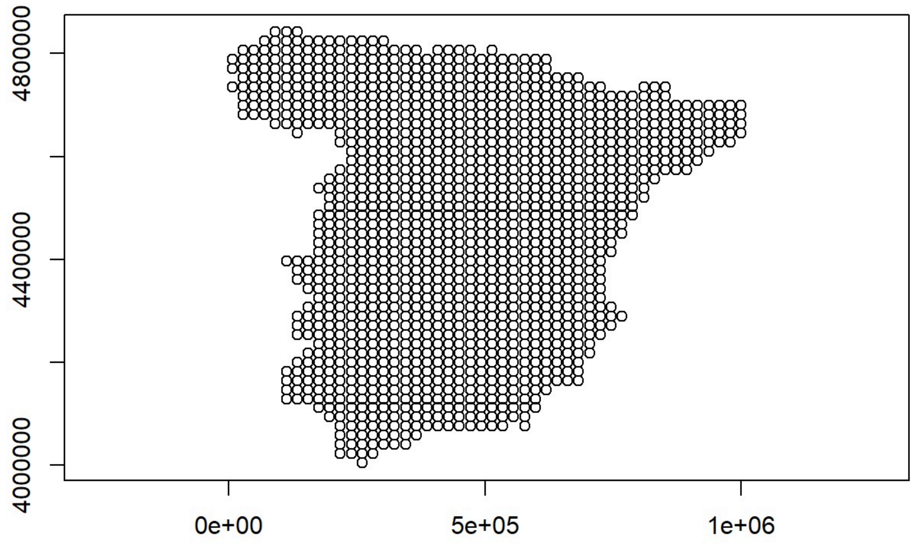
```
> # Grid construction
> bb = st_bbox(spain)
> x = seq(bb$xmin - 1, bb$xmax + 1, length.out = 50)
> y = seq(bb$ymin - 1, bb$ymax + 1, length.out = 50)
> dp = as.matrix(expand.grid(x, y))
> #plot(dp, asp = 1, xlab="", ylab="")
```



Grid locations for predictions [2]

- We intersect the created grid with the map of Spain, to keep only the locations that lie within that map

```
> # keep only locations within borders of Spain
> p = st_as_sf(data.frame(x = dp[, 1], y = dp[, 2]),
+              coords = c("x", "y"))
> st_crs(p) = st_crs(25830)
> ind = st_intersects(spain, p)
> dp = dp[ind[[1]], ]
> #plot(dp, asp = 1, xlab="", ylab="")
```



Grid for predictions & A matrix for the predicton part

- Finally, as we want to predict over three years, we bind three times the coordinates at the prediction points, specifying the times as: time 1 for 2015, time 2 for 2016, and time 3 for 2017.

```
> dp_final = rbind(cbind(dp, 1), cbind(dp, 2), cbind(dp, 3))
> head(dp_final)
```

```
         Var1    Var2
[1,] 260558.8 4004853 1
[2,] 218306.0 4022657 1
[3,] 239432.4 4022657 1
[4,] 260558.8 4022657 1
[5,] 281685.2 4022657 1
[6,] 218306.0 4040460 1
```

```
> # A matrix for predictions
> coo_pred = dp_final[, 1:2]*sc
> groupp = dp_final[, 3]
> A_pred = inla.spde.make.A(mesh = mesh,
+                           loc = coo_pred, # prediction locations
+                           group = groupp) # time indices
```

Data stack preparation

To make predictions, we need to construct the stacks for the estimation and prediction, then to join them.

- First, we build the estimation stack:

```
> stack_est = inla.stack(
+   tag = "est",
+   data = list(y = df$value),
+   A = list(1, A_est),
+   effects = list(data.frame(b0 = rep(1, nrow(df))), indexs))
```

- Then, to predict within the `inla` fitting function, we need to create a stack also for prediction. Note that for the response in the prediction stack we will set it to $y =$NA:

```
> stack_pred = inla.stack(
+   tag = "pred",
+   data = list(y = NA),
+   A = list(1, A_pred),
+   effects = list(data.frame(b0 = rep(1, nrow(dp_final))), indexs))
```

- Lastly, we join the prediction and observed data stack together

```
> stack = inla.stack(stack_est, stack_pred)
```

Define the formula

- We define the formula, specifying a PC prior for the temporal correlation parameter ρ linked to the AR(1) model, such that $p(\rho > 0 = 0.9)$

```
> rho_hyper = list(theta = list(prior = "pccor1", param = c(0, 0.9)))
>
> formula = y ~ -1 + b0 + f(spatial.field,
+                           model = spde, group = spatial.field.group,
+                           control.group = list(model = "ar1", hyper = rho_hyper))
```

Note that using the options `group` and `control.group` we specify that at each time point the spatial locations are linked by the `spde` model object, while across time the process evolves according to an AR(1) dynamics.

Run the space-time model!

```
> fit = inla(formula,
+            data = inla.stack.data(stack, spde=spde),
+            family = "gaussian",
+            control.predictor = list(A = inla.stack.A(stack), compute = TRUE),
+            control.compute = list(return.marginals.predictor = TRUE))
```

```
> round(fit$summary.fixed[,c("mean","0.025quant","0.975quant")],3)
```

```
   mean 0.025quant 0.975quant
b0 8.59      8.025      9.161
```

```
> round(fit$summary.hyperpar, 3)
```

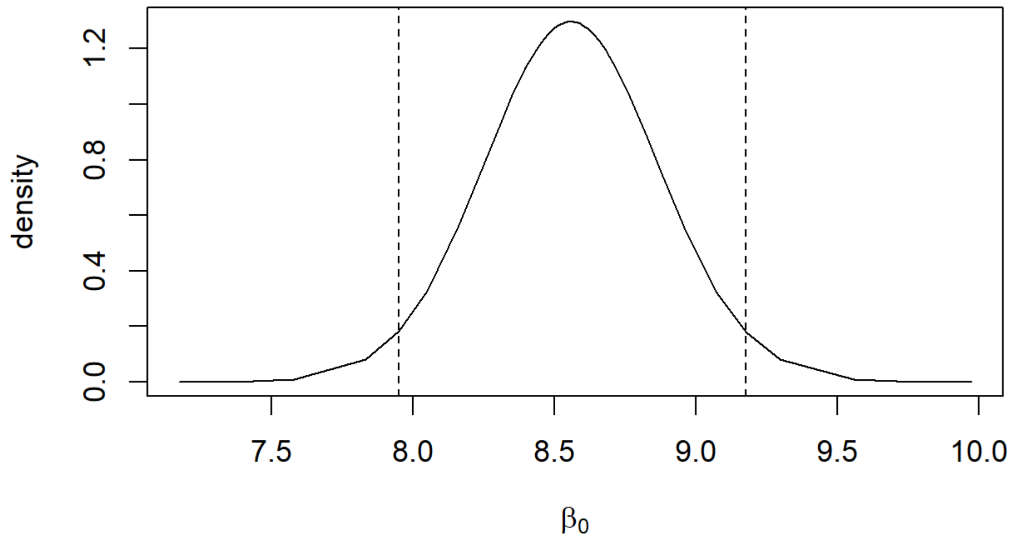
```
                                          mean    sd 0.025quant 0.5quant 0.975quant   mode
Precision for the Gaussian observations  0.852 0.172      0.554    0.838      1.229  0.816
Range for spatial.field                 18.876 1.864     15.457   18.790     22.788 18.630
Stdev for spatial.field                  4.768 0.367      4.085    4.755      5.530  4.729
GroupRho for spatial.field               0.963 0.014      0.929    0.965      0.985  0.969
```

Marginal posterior distribution of the intercept $b0$

```
> modfix = fit$summary.fixed
> modfix
```

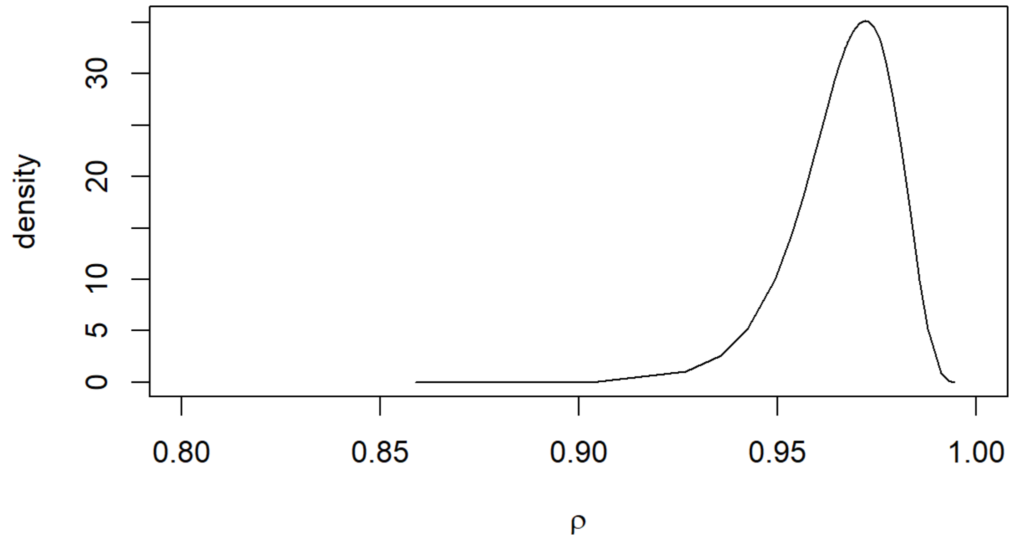
```
       mean        sd 0.025quant 0.5quant 0.975quant     mode          kld
b0 8.590264 0.2890149   8.024509 8.589366   9.161146 8.589374 1.633059e-08
```

```
> plot(fit$marginals.fix$b0,type ='l',xlab=expression(beta[0]),ylab="density")
> abline(v = modfix[1, c(3, 5)], lty=2)
```



Plot of the posterior distribution of the temporal correlation ρ

```
> # AR1 parameter
> plot(fit$marginals.hyperpar$`GroupRho for spatial.field`,
+       type = 'l',xlab = expression(rho),ylab = "density", xlim=c(0.8,1))
```



We can see that the AR(1) coefficient of the latent field, ρ, is large and most of the mass of the posterior distribution is close to 1.

Plot of the variance and range parameters [1]

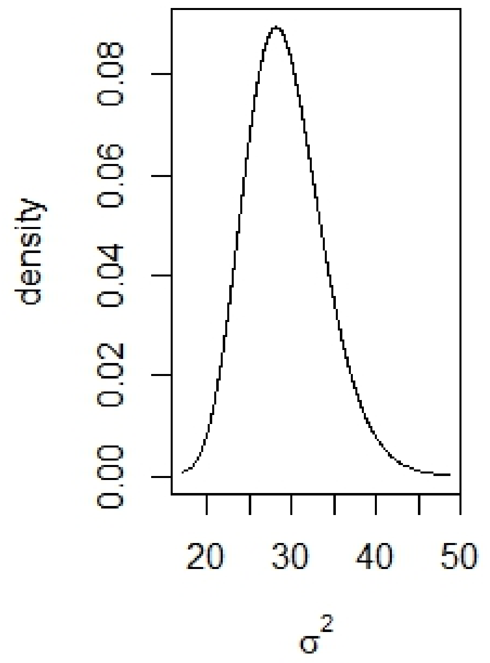
- We use the function `inla.spde2.result` to extract field and parameter values and distributions for the `inla.spde` SPDE effect from the INLA result object

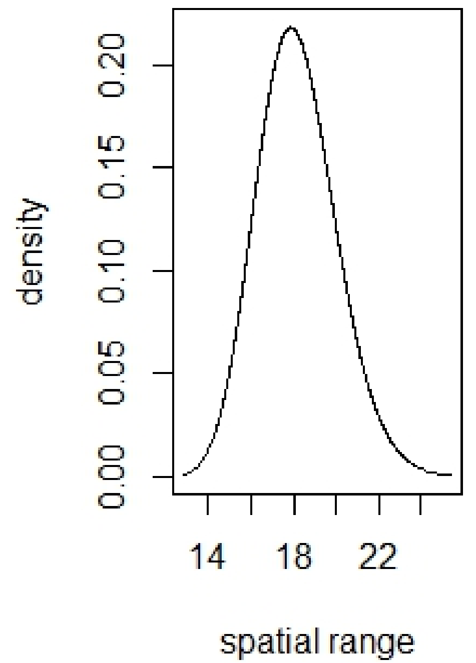
```
> output.field = inla.spde2.result(inla = fit,
+                                  name = "spatial.field",
+                                  spde = spde,
+                                  do.transf = TRUE)
```

- Then we plot the parameters of the spatial field

```
> par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
>
> plot(output.field$marginals.variance.nominal[[1]],type = 'l',
+      xlab = expression(sigma^2),ylab = "density")
>
> plot(output.field$marginals.range.nominal[[1]],type = 'l',
+      xlab = "spatial range",ylab = "density")
```

Plot of the variance and range parameters [2]





Map of the predicted particle concentrations (and 95% CI) by year

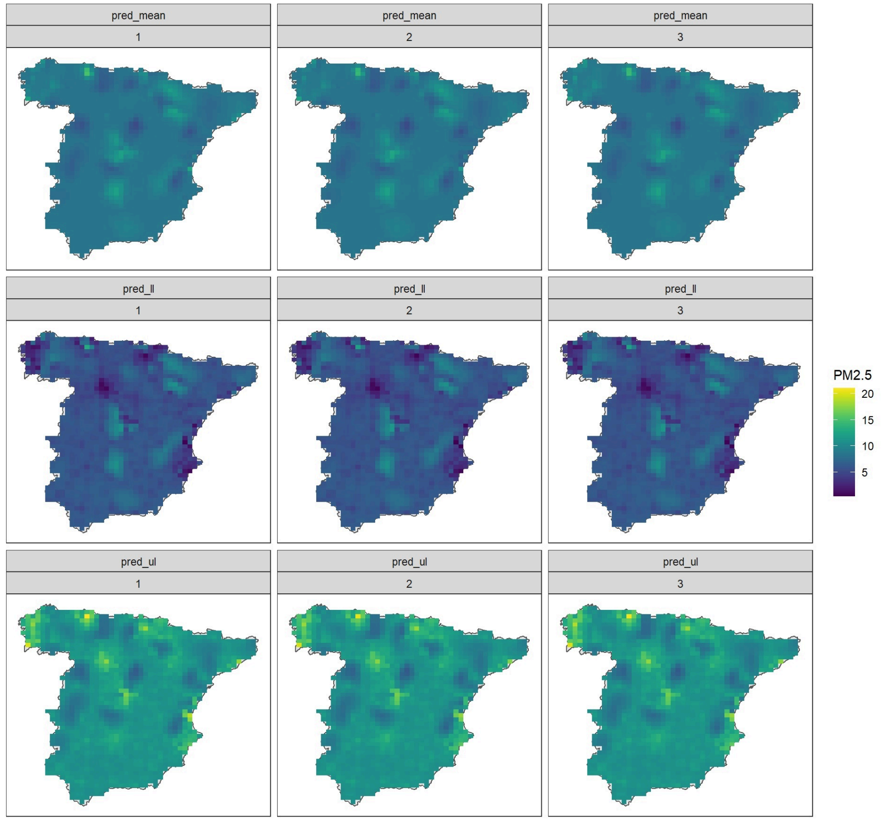
- To map the predicted concentrations of PM2.5, we need (i) to extract the indices to the prediction nodes, and then (ii) to extract the posterior mean (and related 95%CI) of the response. We use `inla.stack.index()` to obtain the indices of the `stack` that correspond to `tag = "pred"`

```
> index = inla.stack.index(stack = stack, tag = "pred")$data
> dp_final = data.frame(dp_final)
> names(dp_final) = c("x", "y", "time")
> # posterior mean
> dp_final$pred_mean = fit$summary.fitted.values[index, "mean"]
> # lower limits of the 95% credible intervals
> dp_final$pred_ll = fit$summary.fitted.values[index, "0.025quant"]
> # upper limits of the 95% credible intervals
> dp_final$pred_ul = fit$summary.fitted.values[index, "0.975quant"]
>
> library(reshape2)
> dpm = melt(dp_final, # melt dp_final into a long data frame
+             id.vars = c("x", "y", "time"),
+             measure.vars = c("pred_mean", "pred_ll", "pred_ul"))
> head(dpm)[1:3,]
```

```
         x       y time  variable    value
1 260558.8 4004853    1 pred_mean 8.590796
2 218306.0 4022657    1 pred_mean 8.590795
3 239432.4 4022657    1 pred_mean 8.590795
```

And finally the yearly maps for PM2.5 concentrations!

```
> # map of posterior means and associated uncertainty
> ggplot(spain) + geom_sf() + coord_sf(datum = NA) +
+   geom_tile(data = dpm, aes(x = x, y = y, fill = value)) + labs(x = "", y = "") +
+   facet_wrap(variable ~ time) + scale_fill_viridis("PM2.5") + theme_bw()
```



Model Validation

- An important aspect of spatial and spatio-temporal models for point-referenced data is related to validation through the evaluation on their predictive performance.
- A very powerful methodology to check the predictive capability is through **cross-validation**.
- Cross-validation seeks to evaluate model predictions by splitting up the data into a **training sample** and a **validation sample**, then fitting the model with the training sample and evaluating it with the validation sample.
- In Session 8 of this workshop we will introduce this methodology.

References

Cameletti, M., R. Ignaccolo, and S. Bande (2011). "Comparing spatio-temporal models for particulate matter in Piemonte". In: *Environmetrics* 22.8, pp. 985-996.

Cameletti, M., F. Lindgren, D. Simpson, et al. (2013). "Spatio-temporal modeling of particulate matter concentration through the SPDE approach". In: *AStA Advances in Statistical Analysis* 97, pp. 109-131.

Moraga, P. (2019). *Geospatial health data: Modeling and visualization with R-INLA and shiny*. CRC Press.