Lab 4.5: Non-Gaussian Spatio-Temporal Models with INLA

Integrated Nested Laplace Approximation (INLA) is a Bayesian method that provides approximate marginal (posterior) distributions over all states and parameters. The package **INLA** allows for a variety of modeling approaches, and the reader is referred to the book by Blangiardo and Cameletti (2015) for an extensive treatment. Other useful resources are Lindgren and Rue (2015) and Krainski et al. (2019).

In this Lab we shall predict expected counts at arbitrary space-time locations from the vector of observed counts $\mathbf{Z}$. The data we use are the Carolina wren counts in the BBS data set described in Lab 4.3. For this Lab, we require the package **INLA** as well as **dplyr, tidyr, ggplot2** and **STRbook**.

```r
library("INLA")
library("dplyr")
library("tidyr")
library("ggplot2")
library("STRbook")
data("MOcarolinawren_long", package = "STRbook")
```

Consider the data model,

$$\mathbf{Z}_t | \mathbf{Y}_t \;\sim\; indep.\; NB(\mathbf{Y}_t, r), \tag{1}$$

and the process model,

$$\log(\mathbf{Y}_t) = \mathbf{X}_t \boldsymbol{\beta} + \boldsymbol{\Phi}_t \boldsymbol{\alpha}_t. \tag{2}$$

In (1) and (2), $\mathbf{Z}_t$ is an m_t-dimensional data vector of counts at m_t spatial locations, $E(\mathbf{Z}_t|\mathbf{Y}_t) = \mathbf{Y}_t$, $\mathbf{Y}_t$ represents the latent spatio-temporal mean process at m_t locations, $\boldsymbol{\Phi}_t$ is an $m_t \times n_\alpha$ matrix of spatial basis functions, r is the size parameter, and the associated random coefficients are modeled as $\boldsymbol{\alpha}_t \;\sim\; Gau(\mathbf{0}, \mathbf{C}_\alpha)$.

In order to fit this hierarchical model, we need to generate the basis functions with which to construct the matrices $\{\boldsymbol{\Phi}_t : t = 1, \ldots, T\}$. In **INLA**, the basis functions used are typically "tent" (finite element) functions constructed over a triangulation of the domain. To establish a "boundary" for the domain, we can use the function `inla.nonconvex.hull`, as follows.

```r
coords <- unique(MOcarolinawren_long[c("loc.ID", "lon", "lat")])
boundary <- inla.nonconvex.hull(as.matrix(coords[, 2:3]))
```

The triangulation of the domain is then carried out using the function `inla.mesh.2d`. This function takes several arguments (see its help file for details). Two of the most important arguments are `max.edge` and `cutoff`. When the former is supplied with a vector of length 2, the first element is the maximum edge length in the interior of the domain, and

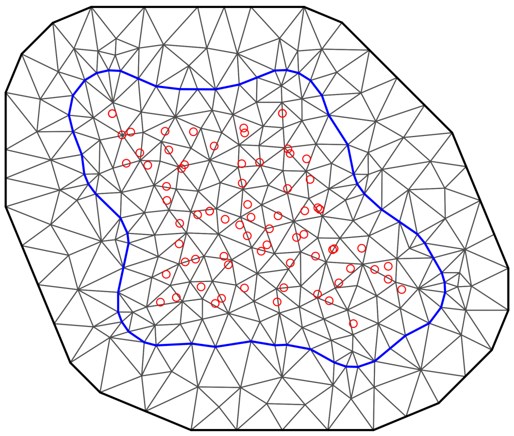

Figure 1: Triangulation for the Carolina wren data locations over which the "tent" functions are constructed (black), and the observation locations (red circles) are superimposed. The blue line denotes the interior non-convex domain of interest that includes all the data points.

the second element is the maximum edge length in the exterior of the domain (obtained from a small buffer that is automatically created to reduce boundary effects). The second argument, `cutoff`, establishes the minimum edge length. Below we choose a maximum edge length of `0.8` in the domain interior. This is probably too large for the problem at hand, but reducing this considerably increases the computational burden when fitting the model.

```
MOmesh <- inla.mesh.2d(boundary = boundary,
                       max.edge = c(0.8, 1.2), # max. edge length
                       cutoff = 0.1)           # min. edge length
```

The mesh and the data locations are plotted using the following commands.

```
plot(MOmesh, asp = 1, main = "")
lines(coords[c("lon", "lat")], col = "red", type = "p")
```

These are shown in Figure 1. Note that the triangulation is irregular and contains an extension with triangles that are larger than those in the interior of the domain.

As in the standard Gaussian case, the modeling effort lies in establishing the covariance matrix of $\boldsymbol{\alpha} \equiv (\boldsymbol{\alpha}_1', \dots, \boldsymbol{\alpha}_T')'$. When using **INLA**, typically the covariance matrix of $\boldsymbol{\alpha}$ is chosen to be separable and of the form $\boldsymbol{\Sigma}_t(\rho) \otimes \boldsymbol{\Sigma}_s(\tau, \kappa, \nu)$ in such a way that its inverse (i.e., the precision matrix) is sparse. The matrix $\boldsymbol{\Sigma}_t$ is constructed assuming an AR(1) process, and thus it is parameterized using a single AR parameter, ρ. This parameter dictates the correlation of $\boldsymbol{\alpha}$ across time; the closer ρ is to 1, the higher the temporal correlation. The matrix $\boldsymbol{\Sigma}_s$ is parameterized using three parameters, and it reflects the spatial covariance required such that the reconstructed field is, approximately, a solution to the stochastic partial differential equation (SPDE)

$$(\kappa^2 - \Delta)^{\alpha/2}(\tau Y(\cdot)) = \epsilon(\cdot),$$

where Δ is the Laplacian, $\epsilon(\cdot)$ is a white-noise process, and τ controls the variance. The resulting field has a Matérn covariance function. The parameter κ is a scaling parameter that translates to a "practical" spatial correlation length (i.e., the spatial separation at which the correlation is 0.1) $l = (\sqrt{8\nu})/\kappa$, while $\alpha = \nu + d/2$ is a smoothness parameter and d is the spatial dimension. Here we fix $\nu = 1$ $(\alpha = 2)$; this parameter is notoriously difficult to estimate and frequently set using cross-validation. Note that there are other "practical" length scales used to characterize the range of a correlation function (e.g., "effective range" when the correlation is 0.05); our choice here is motivated by the **INLA** package that readily provides a marginal distribution over the parameter l as defined here.

The SPDE can be constructed on the mesh using the function `inla.spde2.pcmatern`. The `pc` in `pcmatern` is short for "penalized complexity," and it is used to refer to prior distributions over the hyperparameters that are both interpretable and that have interesting theoretical properties (Simpson et al., 2017). We define prior distributions below over the range parameter l such that $P(l < 1) = 0.01$, and over the marginal standard deviation such that $P(\sigma > 4) = 0.01$. We elicited these distributions by looking at the count data – it is highly unlikely that the spatial correlation length is less than 1 degree and that the expected counts are of the order of 50 or more (we will use a log link, and $e^4 \approx 55$).

```r
spde <- inla.spde2.pcmatern(mesh = MOmesh,
                            alpha = 2,
                            prior.range = c(1, 0.01),
                            prior.sigma = c(4, 0.01))
```

With the discretization shown in Figure 1, $\alpha_{t,i}$ can be viewed as the weight of the ith basis function at time t. The observation matrix $\boldsymbol{\Phi}_t$ then maps the observations to the finite-element space at time t; if the observation lies exactly on a vertex, then the associated row in $\boldsymbol{\Phi}_t$ will be 0 everywhere except for a 1 in the column corresponding to the vertex. Otherwise, the row has three non-zero elements, with each representing the proportion being assigned to each vertex. For point predictions or areal averages, all rows in $\boldsymbol{\Phi}_t$ sum to 1. Finally, for this example, we choose each element in $\mathbf{X}_t$ to be equal to 1. The coefficient β_0 is then the intercept.

The package **INLA** requires space and time to be "blocked up" with an ordering of the variables in which space runs faster than time (i.e., the first few variables are spatial nodes at the first time point, the next few are at the second time point, and so on). Hence we have the block-matrix structure

$$\log\left(\begin{bmatrix}\mathbf{Y}_1\\ \vdots \\ \mathbf{Y}_T\end{bmatrix}\right) = \begin{bmatrix}\mathbf{X}_1\\ \vdots \\ \mathbf{X}_T\end{bmatrix}\boldsymbol{\beta} + \begin{bmatrix}\boldsymbol{\Phi}_1 & \mathbf{0} & \dots \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\Phi}_T\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \vdots \\ \boldsymbol{\alpha}_T\end{bmatrix}, \tag{3}$$

where $\log(\cdot)$ corresponds to a vector of elementwise logarithms. This can be further simplified to

$$\log(\mathbf{Y}) = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\Phi}\boldsymbol{\alpha}, \tag{4}$$

where $\mathbf{Y} = (\mathbf{Y}_1', \dots, \mathbf{Y}_T')'$, $\mathbf{X} = (\mathbf{X}_1', \dots, \mathbf{X}_T')'$, $\boldsymbol{\Phi} \equiv \mathrm{bdiag}(\{\boldsymbol{\Phi}_t : t = 1, \dots, T\})$, $\mathrm{bdiag}(\cdot)$ constructs a block-diagonal matrix from its arguments, and $\boldsymbol{\alpha} \equiv (\boldsymbol{\alpha}_1', \dots, \boldsymbol{\alpha}_T')'$.

A space-time index needs to be constructed for this representation. This index is a double index that identifies both the spatial location and the associated time point. In Lab 2.2 we saw how the function **expand.grid** can be used to generate such indices from a set of spatial locations and time points. In **INLA**, we instead use the function **inla.spde.make.index**. It takes as arguments the index name, the number of spatial points in the mesh, and the number of time points.

```r
n_years <- length(unique(MOcarolinawren_long$t))
n_spatial <- MOmesh$n
s_index <- inla.spde.make.index(name = "spatial.field",
                                n.spde = n_spatial,
                                n.group = n_years)
```

The list `s_index` contains two important items, the `spatial.field` index, which runs from 1 to `n_spatial` for `n_years` times, and `spatial.field.group`, which runs from 1 to `n_years`, with each element replicated `n_spatial` times. Note how this is similar to what one would obtain from **expand.grid**.

The matrix $\boldsymbol{\Phi}$ in (4) is found using the **inla.spde.make.A** function. This function takes as arguments the mesh, the measurement locations `loc`, the measurement group (in our case the year of observation) and the number of groups.

```r
coords.allyear <- MOcarolinawren_long[c("lon", "lat")] %>%
                  as.matrix()
PHI <- inla.spde.make.A(mesh = MOmesh,
                        loc = coords.allyear,
                        group = MOcarolinawren_long$t,
                        n.group = n_years)
```

Note that

```
dim(PHI)

## [1] 1575 5439
```

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

```
nrow(MOcarolinawren_long)

## [1] 1575

length(s_index$spatial.field)

## [1] 5439
```

The latent Gaussian model is constructed in **INLA** through a *stack*. Stacks are handy as they allow one to define data, effects, and observation matrices in groups (e.g., one accounting for the measurement locations and another accounting for the prediction locations), which can then be stacked together into one bigger stack. In order to build a stack we need to further block up (3) into a representation amenable to the `inla` function (called later on) as follows:

$$\log(\mathbf{Y}) = \mathbf{\Pi}\boldsymbol{\gamma},$$

where $\mathbf{\Pi} = (\mathbf{\Phi}, \mathbf{1})$ and $\boldsymbol{\gamma} = (\boldsymbol{\alpha}', \beta_0)'$.

A stack containing the data and covariates at the measurement locations is constructed by supplying the data (argument `data`), the matrix $\mathbf{\Pi}$ (argument `A`), and information on the vector $\boldsymbol{\gamma}$. The stack is then tagged with the label `"est"`.

```
## First stack: Estimation
n_data <- nrow(MOcarolinawren_long)
stack_est <- inla.stack(
                 data = list(cnt = MOcarolinawren_long$cnt),
                 A = list(PHI, 1),
                 effects = list(s_index,
                                list(Intercept = rep(1, n_data))),
                 tag = "est")
```

We next construct a stack containing the matrices and vectors defining the model at the prediction locations. In this case, we choose the triangulation vertices as the prediction locations; then $\mathbf{\Phi}$ is simply the identity matrix, and $\mathbf{X}$ is a vector of ones. We store the information on the prediction locations in `df_pred` and that for $\mathbf{\Phi}$ in `PHI_pred`.

```r
df_pred <- data.frame(lon = rep(MOmesh$loc[,1], n_years),
                      lat = rep(MOmesh$loc[,2], n_years),
                      t = rep(1:n_years, each = MOmesh$n))
n_pred <- nrow(df_pred)
PHI_pred <- Diagonal(n = n_pred)
```

The prediction stack is constructed in a very similar way to the estimation stack, but this time we set the data values to `NA` to indicate that prediction should be carried out at these locations.

```r
## Second stack: Prediction
stack_pred <- inla.stack(
                data = list(cnt = NA),
                A = list(PHI_pred, 1),
                effects = list(s_index,
                               list(Intercept = rep(1, n_pred))),
                tag = "pred")
```

The estimation stack and prediction stack are combined using the **`inla.stack`** function.

```r
stack <- inla.stack(stack_est, stack_pred)
```

All **`inla.stack`** does is block-concatenate the matrices and vectors in the individual stacks. Denote the log-expected counts at the prediction locations as $\mathbf{Y}^*$, the covariates as $\mathbf{X}^*$, and the basis functions evaluated at the prediction locations as $\boldsymbol{\Phi}^*$. Then

$$\begin{bmatrix}\log(\mathbf{Y})\\ \log(\mathbf{Y}^*)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Pi}\\ \boldsymbol{\Pi}^*\end{bmatrix}\gamma,$$

recalling that $\boldsymbol{\gamma} = (\boldsymbol{\alpha}', \beta_0)'$. Note that, internally, some columns of $\boldsymbol{\Pi}$ and $\boldsymbol{\Pi}^*$ corresponding to unobserved states are not stored. For example $\boldsymbol{\Phi}$, internally, has dimension

```r
dim(stack_est$A)

## [1] 1575 1702
```

The number of rows corresponds to the number of data points, while the number of columns corresponds to the number of observed states (`sum(colSums(PHI) > 0)`) plus one for the intercept term.

All that remains before fitting the model is for us to define the formula, which is a combination of a standard `R` formula for the fixed effects and an **INLA** formula for the spatio-temporal residual component. For the latter, we need to specify the name of the index we created as the first argument (in this case `spatial.field`), the model (in this case spde), the name of the grouping/time index (in this case spatial.field.group) and, finally, the model to be constructed across groups (in this case an AR(1) model). The latter modeling choice implies that $E(\boldsymbol{\alpha}_{t+1} \mid \boldsymbol{\alpha}_t) = \rho\boldsymbol{\alpha}_t$, $t = 1, \ldots, T-1$. Our choice for the prior on the AR(1) coefficient, ρ, is a penalized complexity prior, such that $P(\rho > 0) = 0.9$ to reflect the prior belief that we highly doubt a negative temporal correlation.

```
## PC prior on rho
rho_hyper <- list(theta=list(prior = 'pccor1',
                             param = c(0, 0.9)))

## Formula
formula <- cnt ~ -1 + Intercept +
                 f(spatial.field,
                   model = spde,
                   group = spatial.field.group,
                   control.group = list(model = "ar1",
                                        hyper = rho_hyper))
```

Now we have everything in place to run the main function for fitting the model, **inla**. This needs the data from the stack (extracted through **inla.stack.data**) and the exponential family (in this case negative-binomial). The remaining options indicate the desired outputs. In the command given below, we instruct **inla** to fit the model and also to compute the predictions at the required locations.

```
output <- inla(formula,
               data = inla.stack.data(stack, spde = spde),
               family = "nbinomial",
               control.predictor = list(A = inla.stack.A(stack),
                                        compute = TRUE))
```

This operation takes a long time. In **STRbook** we provide the important components of this object, which can be loaded through

```
data("INLA_output", package = "STRbook")
```

INLA provides approximate marginal posterior distributions for each $\boldsymbol{\alpha}_t$ in $\boldsymbol{\alpha}$ and $\{\boldsymbol{\beta}, \rho, \tau\, \kappa\}$. The returned object, output, contains all the results as well as summaries of these results for quick analysis. From the posterior distributions over the precision parameter τ and scale parameter κ, we can readily obtain marginal posterior distributions over the more interpretable variance parameter σ^2 and practical range parameter l. Posterior distributions of some of the parameters are shown in Figure 2, where we can see that the AR(1) coefficient of the latent field, ρ, is large (most of the mass of the posterior distribution is close to 1), and the practical range parameter, l, is of the order of 2 degrees ($\approx$ 200 km).

The posterior distribution of the marginal variance of the latent field is largest between 2 and 4, These values suggest that there are strong spatial and temporal dependencies in the data. We give code below for plotting the posterior marginal distributions shown in Figure 2.

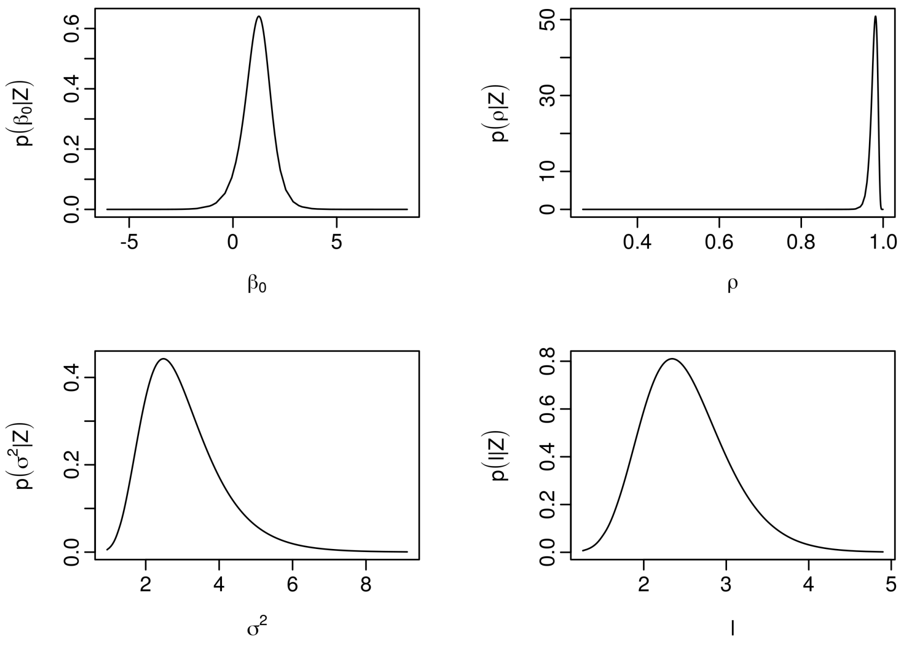

Figure 2: Marginal posterior distributions of β_0, the temporal correlation ρ, the variance σ^2, and the range parameter l.

```
output.field <- inla.spde2.result(inla = output,
                                  name = "spatial.field",
                                  spde = spde,
                                  do.transf = TRUE)
## plot p(beta0 | Z)
plot(output$marginals.fix$Intercept,
     type = 'l',
     xlab = expression(beta[0]),
     ylab = expression(p(beta[0]*"|"*Z)))

## plot p(rho | Z)
plot(output$marginals.hyperpar$`GroupRho for spatial.field`,
     type = 'l',
     xlab = expression(rho),
     ylab = expression(p(rho*"|"*Z)))

## plot p(sigma^2 | Z)
plot(output.field$marginals.variance.nominal[[1]],
     type = 'l',
     xlab = expression(sigma^2),
     ylab = expression(p(sigma^2*"|"*Z)))

## plot p(range | Z)
plot(output.field$marginals.range.nominal[[1]],
     type = 'l',
     xlab = expression(l),
     ylab = expression(p(l*"|"*Z)))
```

We provide the prediction (posterior mean) and prediction standard error (posterior standard deviation) for $\log(Y(\cdot))$ in Figures 3 and 4, respectively. These figures were generated by linearly interpolating the posterior mean and posterior standard deviation of $\log(\mathbf{Y}^*)$ on a fine grid. Note how a high observed count at a certain location in one year affects the predictions at the same location in neighboring years, even if unobserved.

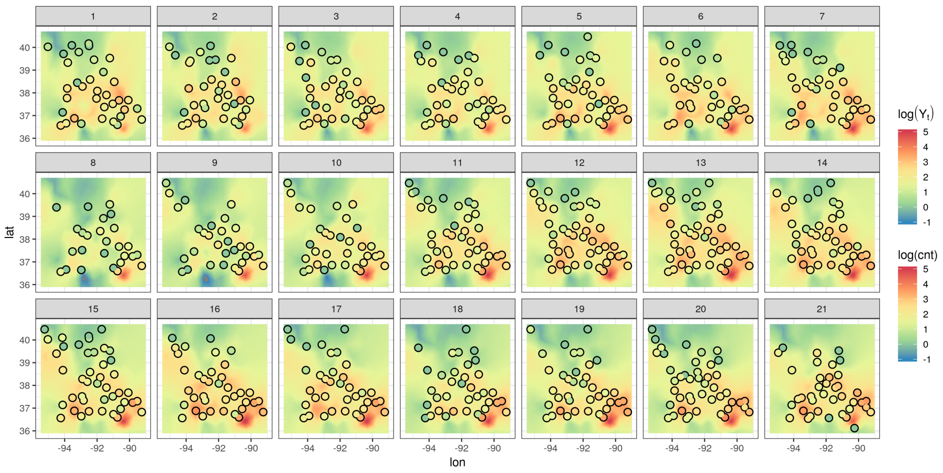

Figure 3: Posterior mean of $\log(Y(\cdot))$ on a grid for $t = 1$ (the year 1994) to $t = 21$ (the year 2014), based on a negative-binomial data model using the package **INLA**. The log of the observed count is shown in circles using the same color scale.

Plotting spatial fields, such as those shown in Figures 3 and 4, from the **INLA** output can be a bit involved since each prediction and prediction standard error of $\boldsymbol{\alpha}_t$ for each t

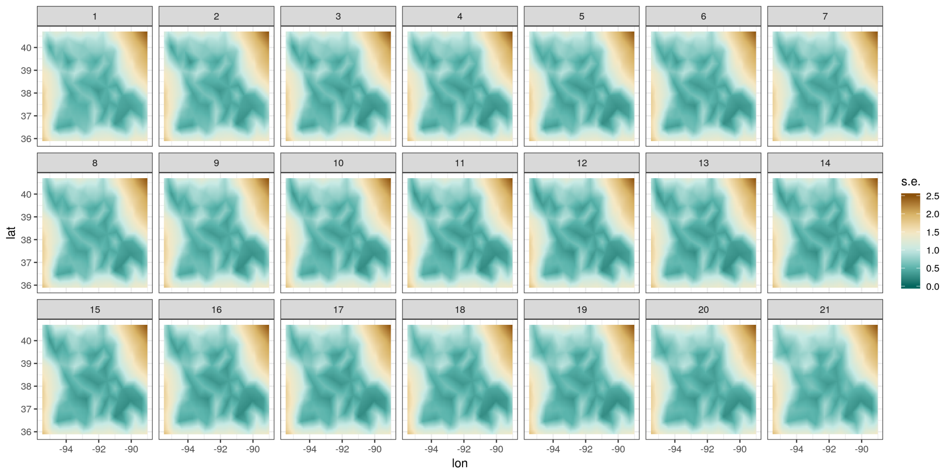

Figure 4: Posterior standard deviation (i.e., prediction standard error) of $\log(Y(\cdot))$ on a grid for $t = 1$ (the year 1994) to $t = 21$ (the year 2014), based on a negative-binomial data model using the package **INLA**.

needs to be projected spatially. First, we extract the predictions and prediction standard errors of $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1', \dots, \boldsymbol{\alpha}_T')'$ as follows.

```
index_pred <- inla.stack.index(stack, "pred")$data
lp_mean <- output$summary.fitted.values$mean[index_pred]
lp_sd <- output$summary.fitted.values$sd[index_pred]
```

Next, we need to create a spatial grid upon which we map the predictions and their associated prediction standard errors. This can be constructed using the function `expand.grid`. We construct an 80×80 grid below.

```
grid_locs <- expand.grid(
                lon = seq(min(MOcarolinawren_long$lon) - 0.2,
                          max(MOcarolinawren_long$lon) + 0.2,
                          length.out = 80),
                lat = seq(min(MOcarolinawren_long$lat) - 0.2,
                          max(MOcarolinawren_long$lat) + 0.2,
                          length.out = 80))
```

The function `inla.mesh.projector` provides all the information required, based on the created spatial grid, to carry out the mapping.

```
proj.grid <- inla.mesh.projector(MOmesh,
                    xlim = c(min(MOcarolinawren_long$lon) - 0.2,
                             max(MOcarolinawren_long$lon) + 0.2),
                    ylim = c(min(MOcarolinawren_long$lat) - 0.2,
                             max(MOcarolinawren_long$lat) + 0.2),
                    dims = c(80, 80))
```

Now we have everything in place to map each $\boldsymbol{\alpha}_t$ on our spatial grid. We iterate through t, and for each $t = 1, \ldots, T$ we map both the prediction and prediction standard errors of $\boldsymbol{\alpha}_t$ on the spatial grid as follows.

```
pred <- sd <-  NULL
for(i in 1:n_years) {
    ii <- (i-1)*MOmesh$n + 1
    jj <- i*MOmesh$n
    pred[[i]] <- cbind(grid_locs,
                       z = c(inla.mesh.project(proj.grid,
                                               lp_mean[ii:jj])),
                       t = i)
    sd[[i]] <- cbind(grid_locs,
                     z = c(inla.mesh.project(proj.grid,
                                             lp_sd[ii:jj])),
                     t = i)
}
```

The last thing we need to do is compile all the data (which are in lists) into one data frame for plotting with **ggplot2**. We concatenate all the list elements rowwise and remove those elements that are `NA` because they fall outside of the support of any basis function.

```
pred <- do.call("rbind", pred) %>% filter(!is.na(z))
sd <- do.call("rbind", sd) %>% filter(!is.na(z))
```

The data frames `pred` and `sd` now contain the spatio-temporal predictions and spatio-temporal prediction standard errors. Plotting of these fields using **ggplot2** is left as an exercise for the reader.

Bibliography

Blangiardo, M. and Cameletti, M. (2015), *Spatial and Spatio-Temporal Bayesian Models with R-INLA*, Hoboken, NJ: John Wiley & Sons.

Krainski, E., Gómez-Rubio, V., Bakka, H., Lenzi, A., Castro-Camilo, D., Simpson, D., Lindgren, F., and Rue, H. (2019), *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*, Boca Raton, FL: Chapman and Hall/CRC.

Lindgren, F. and Rue, H. (2015), "Bayesian spatial modelling with R-INLA," *Journal of Statistical Software*, 63, 1–25.

Simpson, D., Rue, H., Riebler, A., Martins, T. G., and Sørbye, S. H. (2017), "Penalising model component complexity: A principled, practical approach to constructing priors," *Statistical Science*, 32, 1–28.