**Simulation study documentation**

**Simulation part**:

**Set up the simulation study region**:

Set up the simulation study region, 100\*100 grids for both x-axis and y-axis in a [0, 1] square region. The coordinates for the 100\*100 grids are the simulation data scale and are also the target location for the grid’s prediction.

R>

grid.x <- 100

grid.y <- 100

pred.grid <- expand.grid(x = seq(0, 1, length.out = grid.x), y = seq(0, 1, length.out = grid.y))

**Simulation data model structure:**

, is assumed to follow the Matérn class of covariance function such that for generic grid points and ,

is the marginal variance of the process, is the scaling parameter, is the modified Bessel function of the second kind, is the smoothness parameter which is usually set at 1.

is the gaussian observation residual that is not explained by the fixed effects and the spatial random effects.

**Set up the model’s parameters:**

Now, for this spatial regression model, the parameters that need to be set up are , , , .

R>

beta0 <- 10 # intercept

nu <- 1 # smooth parameter is set at 1.

var <- 30 # marginal variances for the Matérn covariance function

vare <- 12 # variances for the gaussian observations

kappa <- 6 # kappa parameter for the Matérn covariance function

scale <- sqrt(2)/kappa

**Simulate the spatial random effect:**

So, the simulation data consists of three parts, the fixed effect part that is a constant for all the grids (), the spatial random effect part **GP(s)** that is controlled by the distance of the grids, and the gaussian observation noises that determines the variances of the gaussian observations for the grids.

1. Spatial random effect simulation

The RMmatern() function is used to set up the Matérn covariance function and the RFsimulate() function is used to simulate a realization for the spatial random effect.

R>

# spatial random effects (Matérn function)

model <- RMmatern(nu = nu, var = var, scale = scale)

area.point <- SpatialPoints(as.matrix(pred.grid))

# spatial random effects simulation results

simu <- RFsimulate(model, x = area.point, n=3) # n indicates how many realizations wanted

gp1 <- simu$variable1.n1 # retrieve the first simulated realization

1. Gaussian observations variances

The rnorm() function is used to simulate a realization for the gaussian observations noise.

R> the error might come from here

error <- rnorm(nrow(pred.grid), mean = 0, sd = sqrt(vare))

1. Synthesize the simulation data

R>

Pred.grid$simu <- gp1 + error + beta0

Here, I have one realization of the simulation data from the model defined above:



Thus, the goal of this simulation study is to reveal the underlying simulation data model parameters as well as predict the grids value from a coarser areal observational level.

**Regression and prediction part**:

The stochastic partial differential equation (SPDE) consists in representing a continuous spatial process using a discretely indexed spatial random process (i.e. a GMRF). The starting point is the linear fractional stochastic partial differential equation (SPDE)

where , is Laplacian, controls the smoothness, is the scale parameter, controls the marginal variances of the Matérn covariance function, and is a Gaussian spatial white noise process. The exact and stationary solution to this SPDE is the stationary GF with Matérn covariance function given by

The solution to the SPDE – represented by the stationary and isotropic Matérn GF – can be approximated using the finite element method through a basis function representation defined on a triangulation of the domain D:

Here G is the total number of vertices of the triangulation, { is the set of basis functions, and {} are zero mean Gaussian distributed weights.

Thus, for each linear predictor,

where is the value of the gth basis function evaluated in location. More generally, it is possible to express the linear predictor as

where is the generic element of the sparse matrix A which maps the GMRF from the G triangulation vertices to the n observation locations.

Based on the theory, we can fit and predict the grids level value based on the polygon/areal observational data.

Create a raster that is coarser than the grids resolution to serve as observation areal data

R>

Raster <- raster(ncol = n, nrow = n, xmn = 0, xmx = 1, ymn = 0, ymx = 1) # n = 10, 25, 50, 80 to indicate coarser observation areal data with different level of resolution.

# convert areal raster to sf object

area <- raster %>%

as('SpatialPolygonsDataFrame') %>%

st\_as\_sf()

# extract grids that are within each coarser area and calculate mean value for each areal observation from true underlying simulation grids data raster

area$layer <- exact\_extract(simulation.true.raster, area, 'mean')

Set up the areal data and assign polygon indexes to the prediction grids points location

R>

# set up the spatialPolygonsDataFrame object containing the areas/polygons

area.poly.1 <- as(area, 'Spatial') %>%

spTransform(crs(raster))

# set up the spatialPolygons object containing the areas/polygons

area.poly <- as(st\_geometry(area), "Spatial") %>%

spTransform(crs(raster))

# assign the polygon indexes to the prediction grids points location based on overlapping relationship

grid.poly.no <- rep(NA, nrow(pred.grid))

for(i in 1:length(area.poly)){

grid.poly.no[as.vector(which(!is.na(sp::over(SpatialPoints(pred.grid, proj4string = crs(area.poly)), area.poly[i]))))] <- i

}

After that, create a mesh on the prediction grids points location

R>

# mesh is created based on prediction grids points location coordinates

coords <- as.matrix(pred.grid[, 1:2])

# create the mesh, the mesh.edge.1 and mesh.edge.2 can be selected to indicate the maximum allowed triangle edge length.

mesh <- inla.mesh.2d(loc = coords, max.edge = c(mesh.edge.1, mesh.edge.2))

Once the mesh is created on the study domain, the spatial structure defined by the spde object is set up. The SPDE object is approximated at the mesh node.

R>

# set up the spde object

spde <- inla.spde2.matern(mesh=mesh, alpha=2)

Following, the mesh coordinates that are within each areal polygon are located and indexed to create the projector A matrix for the areal polygon observations.

R>

# locate the mesh coordinates that are within each areal polygon

mesh.coord.in <- mesh$loc[as.vector(which(!is.na(over(SpatialPoints(SpatialPoints(mesh$loc), proj4string = crs(area.poly)), area.poly)))),] %>%

as.data.frame() %>%

rename(x = V1,y = V2) %>%

dplyr::select(-V3) %>%

as.matrix()

# Assign polygon indexes for the mesh coordinates based on the spatial overlapping relationship

mesh.coord.poly.no <- numeric(nrow(mesh.coord.in))

for(i in 1:length(area.poly)){

mesh.coord.poly.no[as.vector(which(!is.na(over(SpatialPoints(mesh.coord.in, proj4string = crs(area.poly)), area.poly[i]))))] <- i

}

# create the projector A matrix for the areal polygon observations

# block argument specifies block groupings: entries with the same block value are joined into a single row in the resulting matrix, and the block values are the row indices.

# block.rescale specifies what scaling method should be used when joining entries as grouped by a block specification.

1. ‘none’: straight sum, no rescaling
2. ‘count’: divide by the number of entries in the block
3. ‘weights’: divide by the sum of the weight values within each block
4. ‘sum’: divide by the resulting row sums

Aarea <- inla.spde.make.A(mesh=mesh, loc=mesh.coord.in,

block=mesh.coord.poly.no, block.rescale="sum")

Then, the required indexes for the SPDE model including naming the spatial field as ‘s’. Besides, apply the projector A matrix to the spatial effect by using the inla.stack() function at both areal observational level and grid prediction level.

R>

# create the indexes for the SPDE model

s.index <- inla.spde.make.index(name = 's', n.spde = spde$n.spde)

# apply the projector A matrix to the spatial effect at the areal observational level.

stack.area <- inla.stack(tag='areal',

data=list(y=area$layer),

A=list(Aarea),

effects=list(c(s.index, data.frame(intercept=1))))

# create the projector A matrix for the grids prediction locations

Apred <- inla.spde.make.A(mesh=mesh, loc=as.matrix(pred.grid %>% dplyr::select(-simu)))

# apply the projector A matrix to the spatial effect at the grids prediction level.

stack.pred <- inla.stack(tag='pred',

data=list(y=NA),

A=list(Apred),

effects=list(c(s.index, data.frame(intercept=1))))

Finally, both the observational and prediction level objects are stacked together and the formula is set up to fit the model as well as making the prediction simultaneously.

R>

#Stack all the objects

stack.all <- inla.stack(stack.area, stack.pred)

# set up the model formula

formula <- y ~ -1 + intercept + f(s, model = spde)

# fit the model

modfit <- inla(formula, data = inla.stack.data(stack.all), control.predictor = list(A = inla.stack.A(stack.all), compute = TRUE), verbose = TRUE)

After the model is fitted, the marginal posterior distribution of the fitted parameters for all the scenarios can be obtained. Taking the scenario of 0.1m resolution areal observation to predict the underlying grids as an example, the marginal posterior distribution of model parameters , , , can be plotted as follows.

The intercept



The scaling parameter



The marginal variance of the covariance function



The nugget effect



As the graphs shown here that the polygon/areal observational SPDE model can identify the underlying model parameters , , with a reasonable error. However, the nugget effect parameter is completely lost in this model fitting process. To further illustrate this problem, two simple compositing nugget effects experiments are conducted in the following.

**Compositing for the Nugget Effect Experiments**

For the compositing problem that from the coarser areal data to predict the simulated sample data, a relationship exists between the nugget effect of the sample data and composited data. If the original simulated sample grids data are at the scale of *v*, consider the averaging to a larger scale *V* and without loss of generality that *V = nv*. Thus, if the sample data are , so the composited areal/polygon observational data in the *V* area are calculated as below:

Also, it is assumed that the data are random as pure nugget effect and stationary so the variance is equal to nugget effect. With this assumption that the simulated grids sample is identical and individually distributed (iid), covariances between the simulated samples within the composites area can be written as:

The variance of the simulated sample composite that is the polygon/areal observation can be calculated by:

This shows that the nugget effect drops very quickly when the data are composited. Following, a simple example is used to illustrate this idea.

**Nugget Effect composition experiments I** (grids samples are iid and not correlated with each other)

In this nugget effect composition experiments example, 100\*100 resolution grids level within a [0,1] square sample data like before is simulated. Differently, the grids level sample data is simulated based on solely nugget effect without any spatial correlations. That is, the grids level sample data is identically and individually distributed (iid) like non-spatial data. The multivariate normal distribution function with simple diagonal covariance function is used to simulate a realization of the grids level data based on this configuration.

Then, a non-correlated grids level nugget effects simulation is generated, and the 0.1m area observation can also be obtained.



Based on the averaging configuration, n is 100 in this case since we only can observe averaged 100 grids from the 0.1m resolution area observation. Thus, we expect the INLA() function can reveal the nugget effect value = 12 for the grids level observation and for the 0.1m area observation.





**Nugget Effect composition experiments II** (grids samples are not iid and are spatially correlated with each other)

The nugget effect composition experiments I in the above shows that the intra-unit variation can be directly calculated if the grid level sample units are iid and are not spatially correlated. However, in the case of spatial variations of aggregated data from spatially correlated grid level sample units, the intra-unit variation is the information loss when aggregating grids data into areal observations and the grids level nugget effect cannot be calculated directly like before. To illustrate this phenomenon, a multivariate normal distribution function with covariance function is used to simulate a realization of the grids level data based on this configuration. To test another covariance function different than Matérn covariance function before, the exponential covariance is used here as a test example whose corresponding covariance function depends on the distance r between two points.

Similarly, an exponential correlated grids level simulation is generated, and the 0.1m area observation can also be obtained.



Based on the averaging configuration, n is 100 in this case since we only can observe averaged 100 grids from the 0.1m resolution area observation. Thus, we expect the INLA() function can reveal the nugget effect value = 12 for the grids level observation and for the 0.1m area observation when covariance function is 0. However, since the grids level is exponentially correlated with each other in this experiment, the variances of gaussian observations for the 0.1m area observation is expected to be larger than .



