# Continuous disease mapping of the Yellow fever in Nigeria

Kamila Kazimierska & Paula Moraga

December 2020

## References


- (Moraga et al. 2017), code of the areal data estimation based on the SPDE.
- (Moraga 2019), code from Chapter 9, Spatial Modelling of geostatistical data
- (Krainski et al. 2018), chapter 2, spde example.
- `vignette("SPDEhowto")` E. Krainski, H. Rue, prediction using sampling.
- (Wilson and Wakefield 2020), code for creating matrix D based on the population intensity (https://github.com/wilsonka/pointless-spatial-modeling/blob/master/Code/setup_simulation.R).
- https://rdrr.io/github/dickoa/rgeoboundaries/, Nigeria administrative boundaries.
- https://apps.worldpop.org/woprVision/, raster population of Nigeria.
- https://ncdc.gov.ng/data, Yellow Fever diesase counts were obtained from publicly available resource, extracted from dashboard as of 23 Nov 2020.


## Data analysis setup

Specify the data analysis setup. The data analysis code based on different setup sepcification is almost the same.

- The `adm.lvl`$\in \{1, 2\}$. The data analysis code is different in two places: loading the data set depends on the administrative level partition, specifying the spatial field sigma prior depends on the data variability.
- The `weights`$\in \{$"*equal*","*pop.int*"$\}$. The data analysis code is different in one place: specifying the projector matrix, either based on the equal weights or the weights that depend on the population intensity.
- If `pred.stk = T`, then the prediction of the risk surface on the fine grid is made using a prediction stack. Note that fitting the risk surface using the prediction stack takes 6279 s (104 min). The computationally efficient alternative is by using sampling, which is a default setup.

```
adm.lvl = 2 # 1,2
weights = "pop.int" # c("equal", "pop.int")
pred.stk = T # c(T,F)
```

## Load data and auxiliary functions

The data consists of

- `YF1` data frame with disease counts, ID, and name for each administrative boundary at level 1.
- `YF2` data frame with disease counts, ID, and name for each administrative boundary at level 2.
- `pop.dat.km` raster object of the Nigeria population aggregated to 43485 cells.

```
load(file.path(lap_data, "Data/ng.RData", fsep = ""))
source.all(file.path(lap_data, "Code/auxiliary_functions/", fsep = ""))
```

```
## *** source areal_mesh_param.R
```

```
## *** source create_matrix.R
## *** source create_mesh.R
## *** source create_stack_full.R
## *** source create_stack.R
## *** source extract_results.R
## *** source extractCoords.R
## *** source fit_spde_full.R
## *** source fit_spde.R
## *** source MC_samples.R
```

# Dataset

We study the incidence proportion of yellow fever in 37 Nigeria states and 773 Local Government Areas (LGA) of Nigeria in 2017–2019 for males and females combined.

## Nigeria administrative boundaries

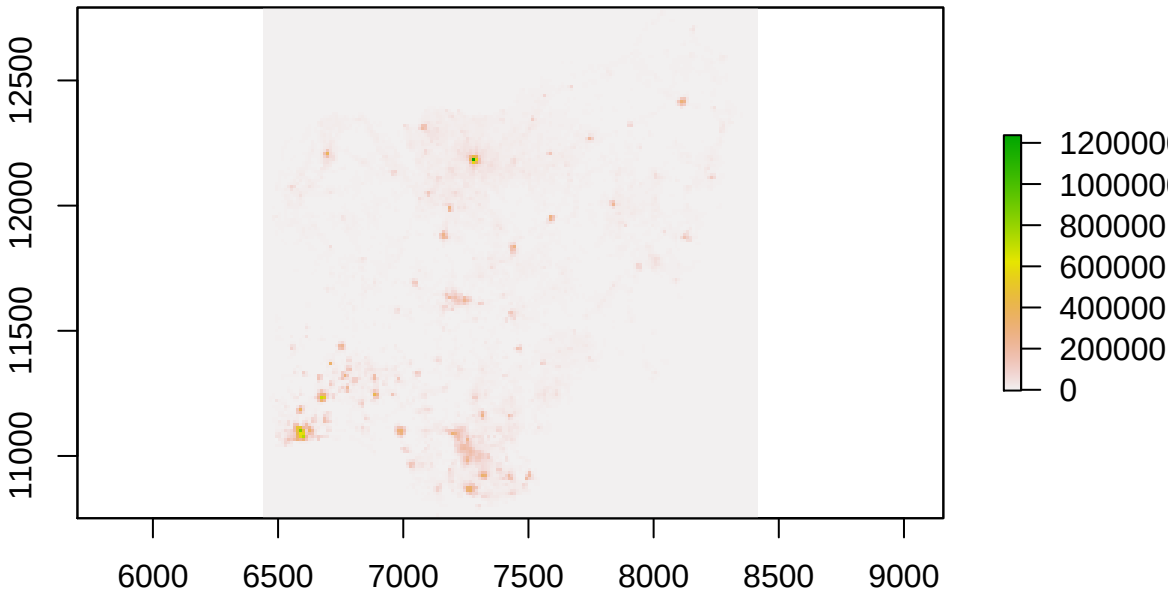
The Nigeria administrative boundaries were obtained from package `rgeoboundaries`, which can be found on the website of the GitHub package.

```
if(adm.lvl == 1){
 map <- st_geometry(gb_adm1("nigeria"))
} else {
 map <- st_geometry(gb_adm2("nigeria"))
}

n = length(map)
map <- as_Spatial(map, IDs =as.character(1:n))
# Transform the CRS to km
map.km <- spTransform(
  map, CRS('+proj=utm +zone=23 +south +ellps=GRS80 +units=km'))
plot(map.km)
```

## Population of Nigeria

The population of Nigeria was obtained from the website of World population.

```r
# Raster is loaded
pop.dat.ng <- crop(pop.dat.km, map.km)
plot(pop.dat.ng)
```




## Combining map with disease counts data frame

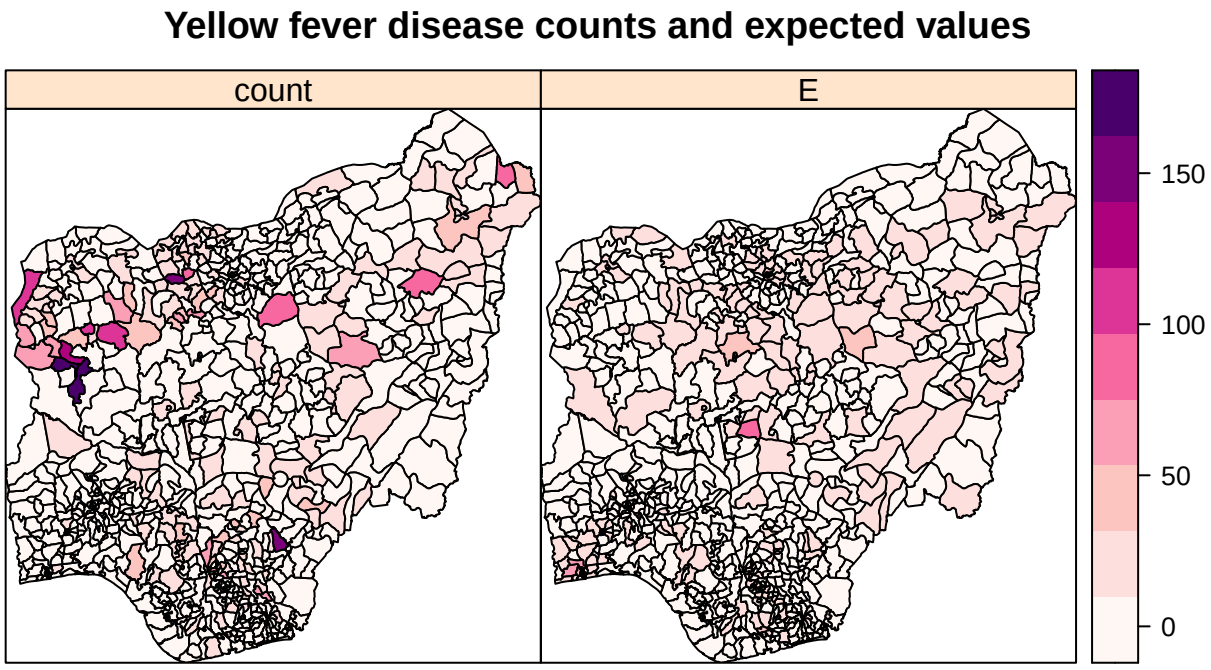
The dataset was created by merging the disease counts data frame with the spatial polygon by the `ID`. * The `df` data frame contains `coordinates` of the Nigeria map, that are labelled as `ID`.
* The disease count data frame `YF1` or `YF2` is merged with the `df` data frame by `ID`. * The `row.names` of the data frame `df` and the spatial Polygons `IDs` have to match.

```r
# Create a SpatialPolygonsDataFrame
df = data.frame(lat = coordinates(map.km)[,1],
                lon = coordinates(map.km)[,2],
                 ID =  seq(1:length(map.km)),
                 row.names = 1:length(map.km))

# Merge disease counts
if(adm.lvl==1){
df2 = left_join(df, YF1, by = "ID")
} else {
df2 = left_join(df, YF2, by = "ID")
}
rownames(df2) = rownames(df)

# Polygons ids are ID i
# map.km is a spatial polygon
mapdf.km= SpatialPolygonsDataFrame(map.km, df2)
```

## Extract the population and compute the expected counts

The expected values takes into account the population. The counts for the North-East state Kebbi are the highest. The expected counts are highest for the state Kano.

```r
# Extract the population for the map polygons from the raster
mapdf.km@data$pop <- extract(pop.dat.km, map.km, fun = sum, na.rm = TRUE)
# compute the expected counts
E = mapdf.km@data$pop*sum(mapdf.km@data$count)/sum(mapdf.km@data$pop)
mapdf.km@data$E <- E

spplot(mapdf.km, c("count", "E"), col.regions = brewer.pal(n = 9, name = "RdPu"),
       main = "Yellow fever disease counts and expected values", cuts = 8)
```

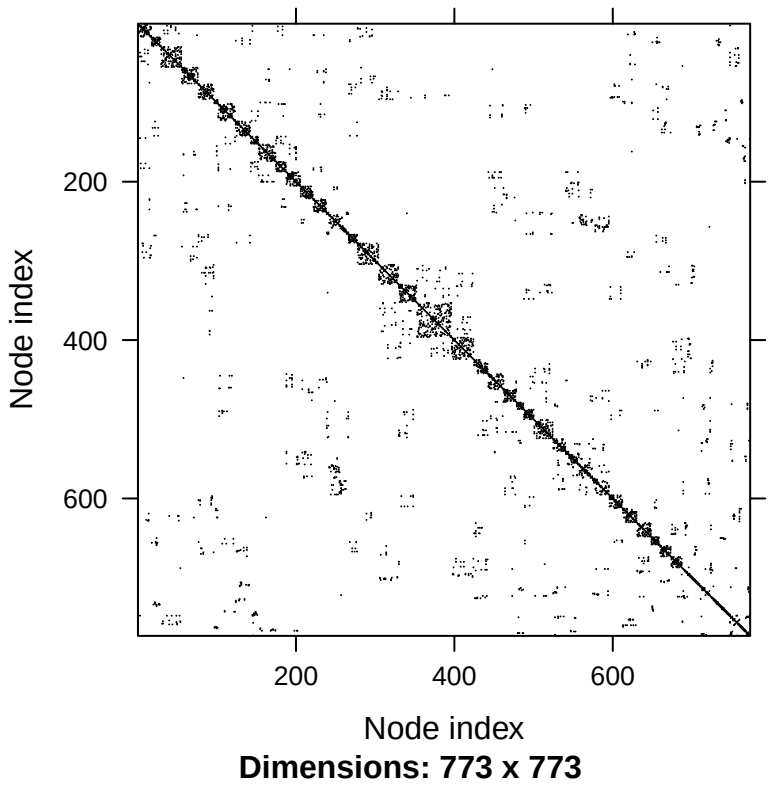



# Inference with INLA BYM2

## Adjecency matrix

Create rook adjecency matrix, save it and plot the image.

```r
# We create rook adjacency matrix, the simplest, considering neighbours
adj.r <- poly2nb(map.km, queen = FALSE) # not a queen means rook
W.bin <- nb2listw(adj.r, style = "B")  # binary, not weighted
# convert nb object to graph object compatible with INLA, save it at selceted path
nb2INLA(file.path(lap_data, "Data/graph", fsep = ""), adj.r)
LD.graph = inla.read.graph(file.path(lap_data, "Data/graph", fsep = ""))
image(inla.graph2matrix(LD.graph), xlab = "Node index", ylab = "Node index")
```



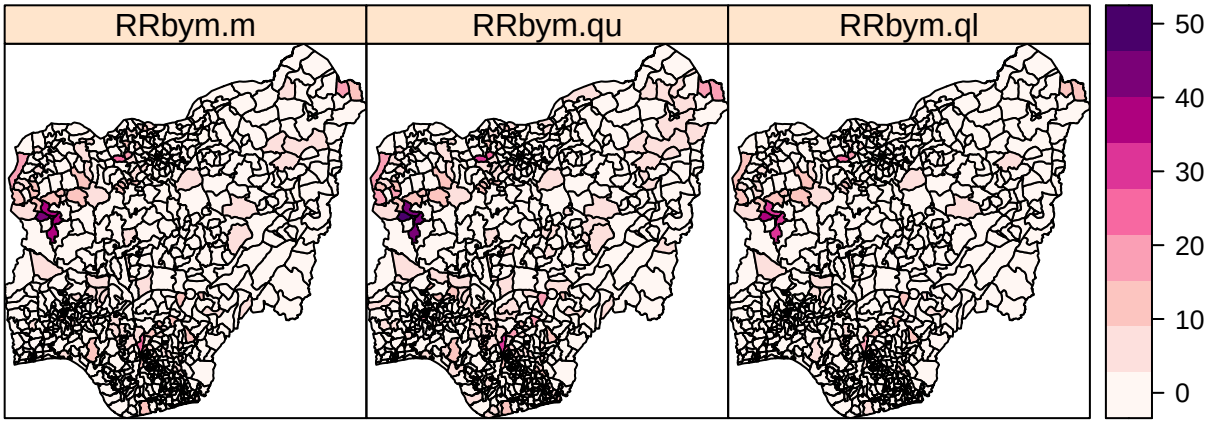

### Fitting

```
formula.model <- count ~ 1 + f(ID, model = "bym2", graph = LD.graph,
                       constr = T, scale.model = T)
res = inla(formula = formula.model,
           family = "poisson",
           data = mapdf.km@data,
           E =  E,
           control.compute = list(dic = T, waic = T, cpo = T, config = TRUE),
           control.predictor = list(compute = TRUE),
           num.threads="4:1",
           quantiles = c(0.05, 0.95))
```

```
## Warning in inla.model.properties.generic(inla.trim.family(model), mm[names(mm) == : Model 'bym2' in s
##   Use this model with extra care!!! Further warnings are disabled.
```

### Relative risk mapping

```
# summary statistics of the relative risk
mapdf.km@data$RRbym.m <- res$summary.fitted.values[, "mean"]
# proportion of times when rr is bgger than 1, based on the joint posterior sample
mapdf.km@data$RRbym.qu <- res$summary.fitted.values[, "0.95quant"]# 95% CI
mapdf.km@data$RRbym.ql <- res$summary.fitted.values[, "0.05quant"]

### plot fitted values
spplot(mapdf.km, c("RRbym.m", "RRbym.qu", "RRbym.ql"), col.regions = brewer.pal(n = 9, name = "RdPu"), n
```




# Inference with INLA - SPDE model

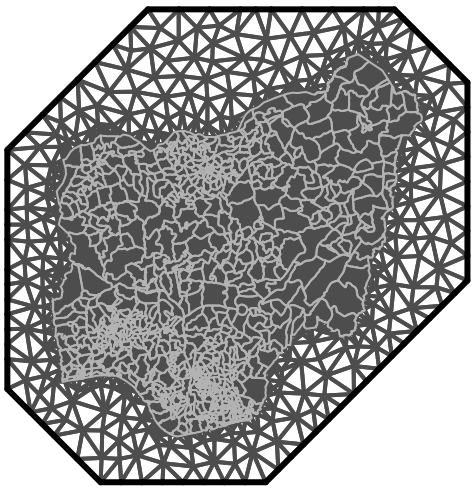
## Create a mesh for the areal data

- We can compute the `max.edge` for the Nigeria domain as a quarter of the range. We assume the range to be not less than 10% of the mean of the domain's extent.

- We avoid inner extension because it creates many triangles in the areas outside the polygon of interest.
- We build an initial mesh first. Using `inla.sp2segment`, we create a boundary of the spatial polygon (SPDF) `mapdf.km` of Nigeria compatible with the option `boundary` in the `inla.mesh.2d` function. From the initial mesh, we build a mesh with points from the initial mesh nodes' locations.
- Function `areal_mesh_param` specify required arguments to build a mesh: `max.edge`, `cutoff` and `offset` for the initial mesh based on the boundary and the mesh based on the initial mesh nodes locations.
- The parameters are computed based on the domain extent. We have computational power to fit the model with maximum 10 thousands nodes.
- Specifying the `cutoff` value considers the desired values for the `max.edge`. The `cutoff` value is related to the `max.edge`, such that max.edge >2 cutoff.
-

```
param = areal_mesh_param(spdf = mapdf.km)
mesh = create_mesh(spdf = mapdf.km, max.edge = param$max.edge, cutoff = param$cutoff,
                   offset = param$offset, plot = T)
```

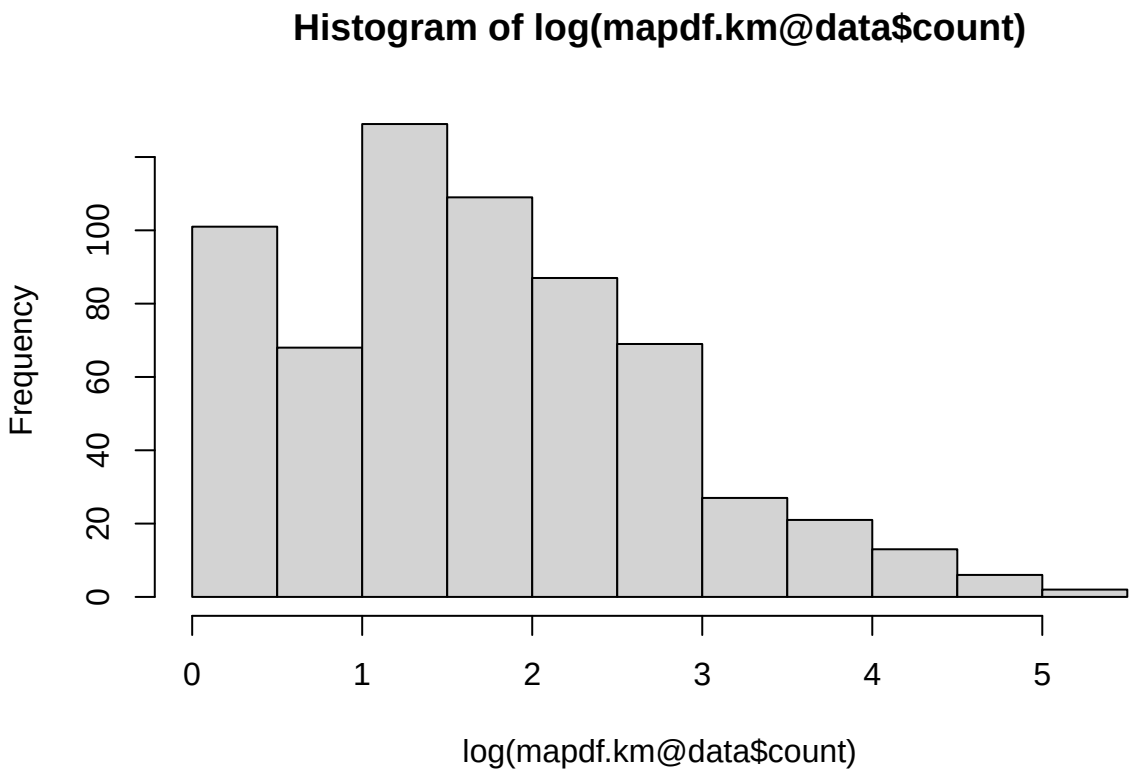
## Constrained refined Delaunay triangulation



### SPDE model

- We assume the log predictor to follow a normal distribution. The normal distribution characteristic is that 95% of the values lie within two standard deviations. The value of the log disease counts lies within $[3, 7]$ ( $[0, 6]$ for administrative level 2). Based on the data variability, the spatial field's standard error should not be bigger than one (one point half).
- We assume the prior for the standard error of the spatial field $P(\sigma > 1) = 0.01$
- We assume prior infromation about the range $P(\text{range} < 15) = 0.01$, where 15 km is only 1% of the domain extent.

```
hist(log(mapdf.km@data$count), breaks = 10)
```




```
sd(log(mapdf.km@data$count), na.rm=TRUE)
```

```
## [1] NaN
```

```
if(adm.lvl==1){
  spde <- inla.spde2.pcmatern(
    mesh, alpha=2,
    prior.sigma=c(1, 0.01), # P(sigma > 1) = 0.01,
    prior.range=c(15, 0.01)) # P(r < 15) = 0.01,

} else {
  spde <- inla.spde2.pcmatern(
    mesh, alpha=2,
    prior.sigma=c(2.5, 0.01), # P(sigma > 2.5) = 0.01,
    prior.range=c(15, 0.01)) # P(r < 15) = 0.01,
}
```

### Create projector matrix

- The projector matrix is created for the areal data in a way that each row of the projection matrix contains non-zero elements at the nodes (columns) that falls in the area.
- The non-zero elements take the weights that depend on the population density or are equal. They sum to one.
- The projector matrix is created based on the mesh nodes that fall in some areas only.

```
if(weights == "equal" ){
  Aa <- create_matrix(sp = map.km, mesh = mesh, weights = "equal")
  str(Aa[1,Aa[1,]!=0])
} else {
  Aa <- create_matrix(sp = map.km, mesh = mesh, weights = "pop.int", pop.raster = pop.dat.ng)
  str(Aa[1,Aa[1,]!=0])
```

```
}
```

```
## Joining by: mlong, mlat
```

```
##  num [1:3989] NA NA NA NA NA NA NA NA NA NA ...
```

### Create stack

```
stack.a <- create_stack(tag = "area", matrix = Aa, spdf = mapdf.km, spde = spde)
```

### Fit the model

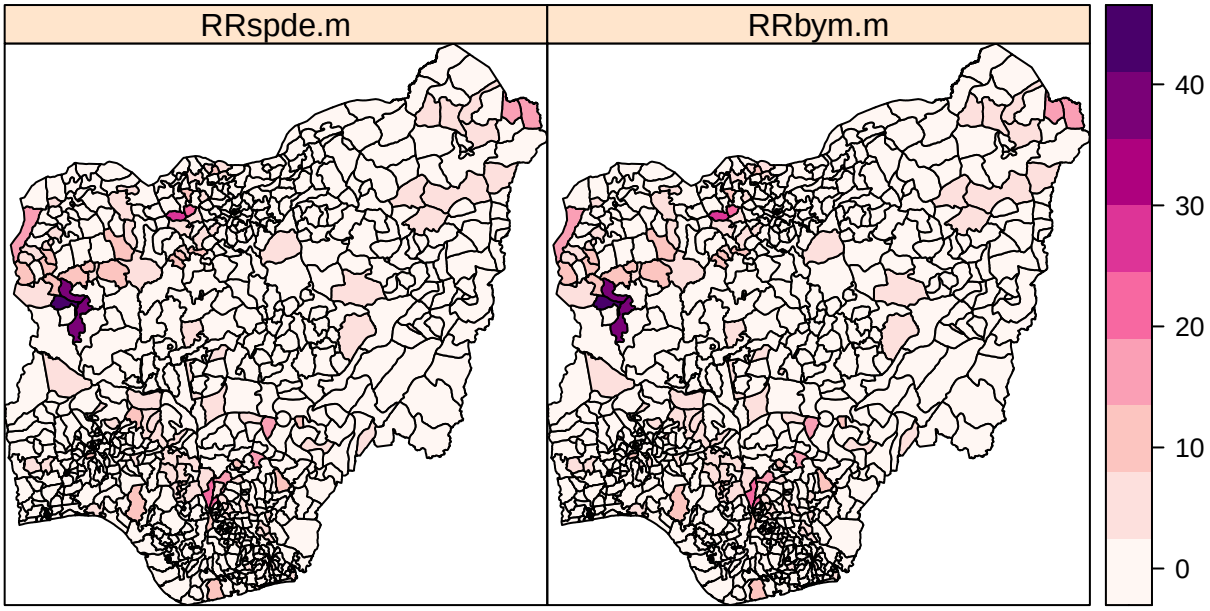
```
res.spde <- fit_spde(spde = spde, stack = stack.a)
res.spde$cpu
```

```
##       Pre   Running      Post     Total
##  7.372664 29.822647  0.652611 37.847922
```

### Results

- Compare SPDE model with BYM2 fitted values

```
mapdf.km@data$RRspde.m <- res.spde$summary.fitted.values[inla.stack.index(stack.a,
                                                                          tag='area')$data, "mean"]
mapdf.km@data$RRbym.m <- res$summary.fitted.values[, "mean"] # using full stack is similar

spplot(mapdf.km, c("RRspde.m","RRbym.m"), col.regions = brewer.pal(n = 9, name = "RdPu"), cuts = 8)
```

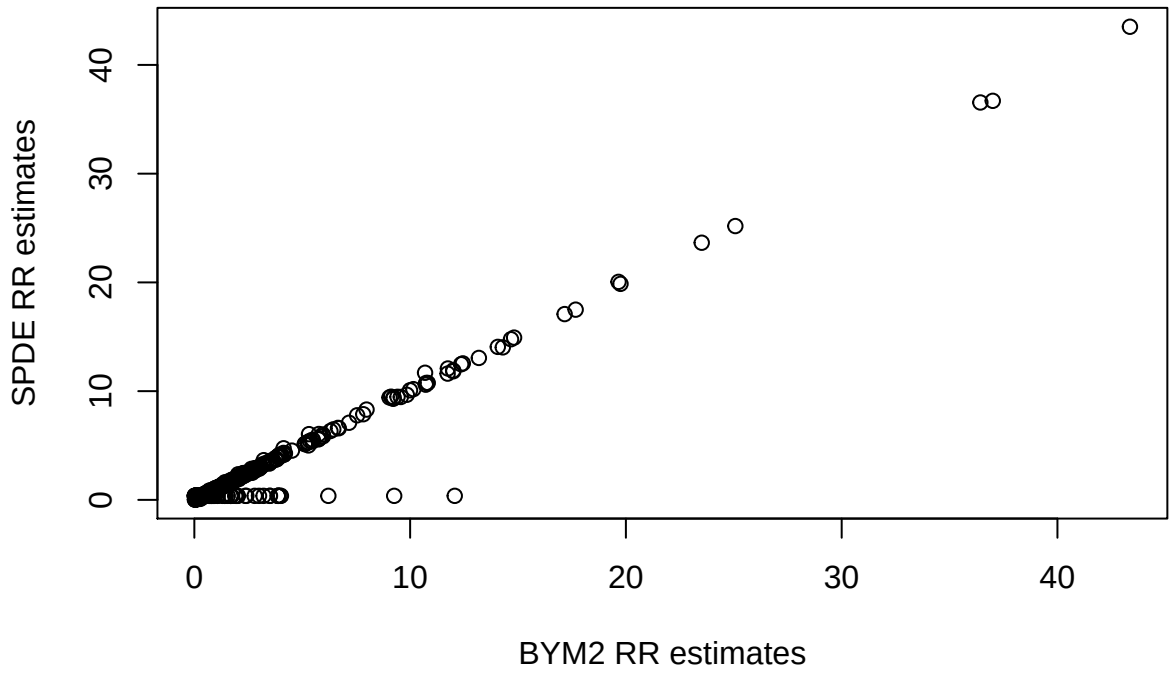



```
dt = data.frame(
DIC = c(res$dic$dic, res.spde$dic$dic),
WAIC = c(res$waic$waic,res.spde$waic$waic),
CPO = c(-sum(log(res$cpo$cpo)), -sum(log(res.spde$cpo$cpo))))
rownames(dt) <- c("BYM2", "SPDE")
kable(dt, caption = "Summary comparison")
```

Table 1: Summary comparison

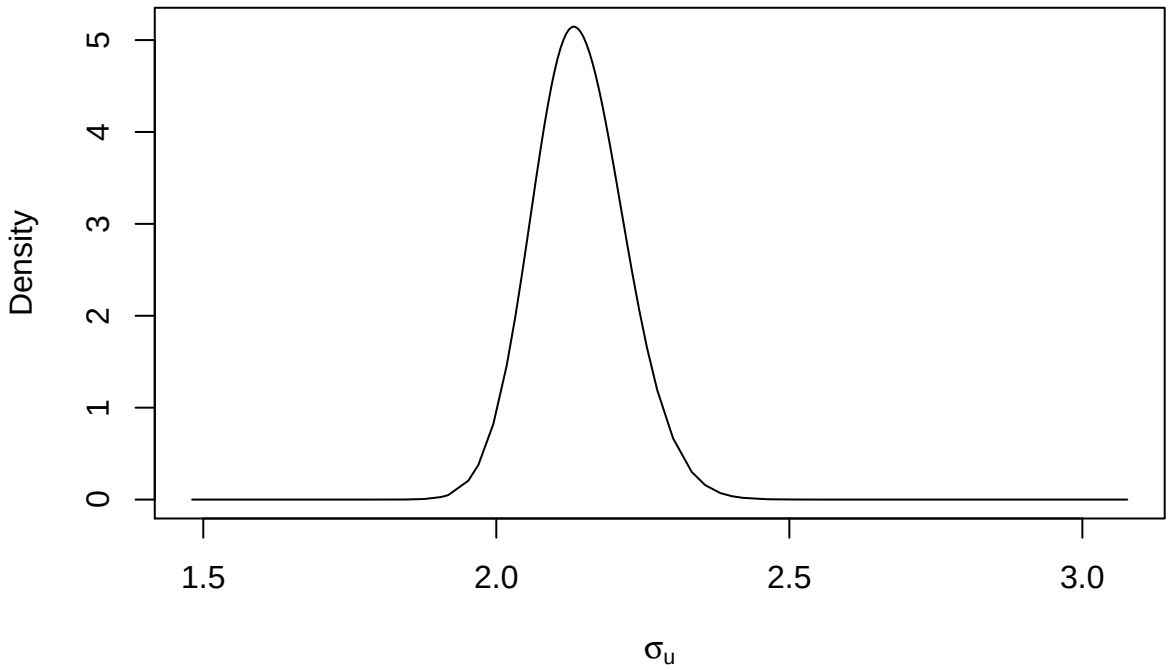
| | DIC | WAIC | CPO |
|---|---|---|---|
| BYM2 | 3736.236 | 3644.330 | 4551.334 |
| SPDE | 4876.084 | 4821.524 | 4693.268 |

```
plot(mapdf.km@data$RRbym.m , mapdf.km@data$RRspde.m,
     xlab = "BYM2 RR estimates",
     ylab = "SPDE RR estimates")
```

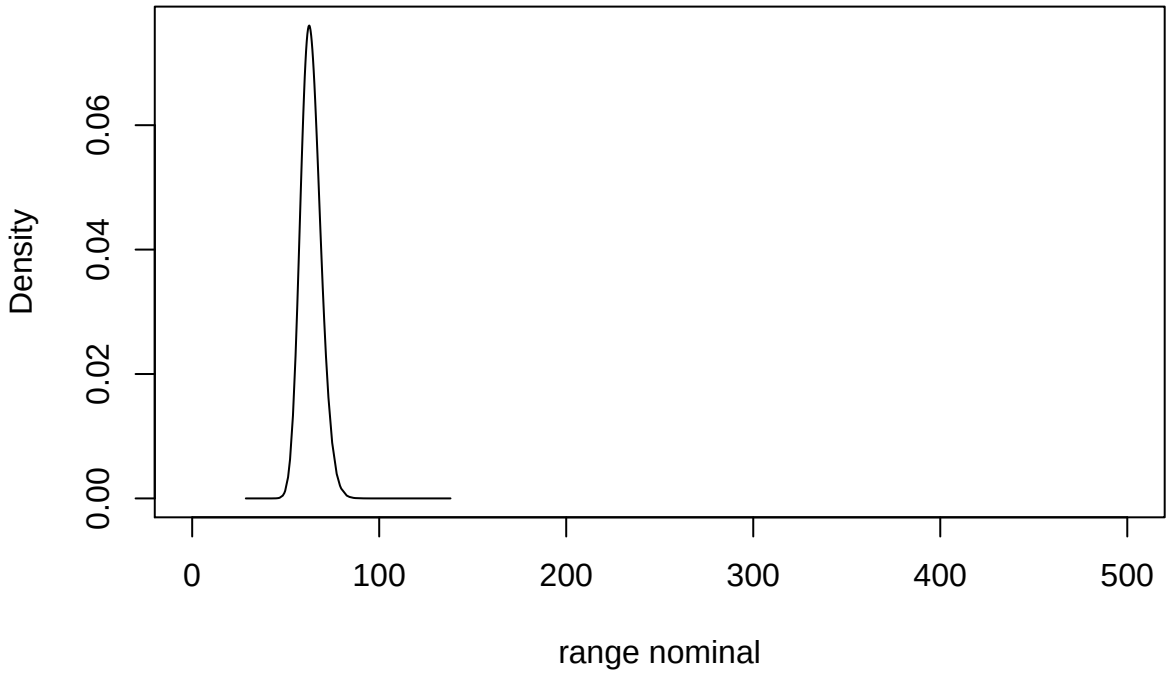



### Posterior marginal distrbution - PMD

```
plot(res.spde$marginals.hy[[2]], type='l', xlab=expression(sigma[u]), yla='Density')
```




```
plot(res.spde$marginals.hy[[1]], type='l', xlab='range nominal', ylab='Density',
     xlim = c(0,500))
```



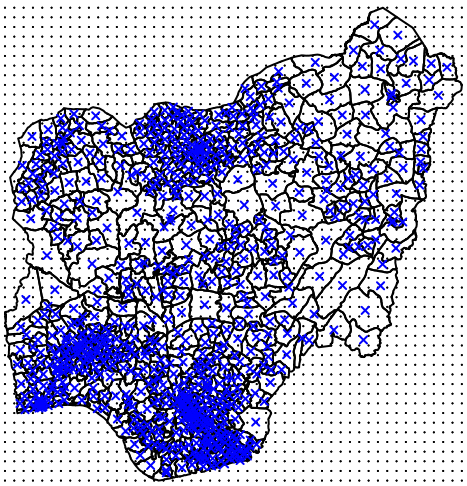

```
# xlim to avoid the log tail
```

- Estimated standard error is higher than assumed by the prior.
- Based on the rule, if the estimated range is smaller than the max.edge, we would have to refine the mesh. Specify the max.edge smaller than the estimated range, but here it is not the case.

# Prediction

- We observe disease counts at the centroids of each spatial area.
- We are interested in obtaining the continous risk surface.
- We obtain the predictions of the relative risk on a regular lattice, 50 cells in each direction.
- Function `inla.mesh.projector` builds a projector matrix and lattice.
- By specifying options `xlim` and `ylim`, we are limiting the lattice to the inner extension of the mesh.

```
coo = coordinates(mapdf.km)  # centroids
bcoo = extractCoords(unionSpatialPolygons(mapdf.km,
                                          IDs = rep(1, nrow(mapdf.km)))) # border coordinates

nxy <- c(50, 50) # lattice dimension
gproj <- inla.mesh.projector(mesh, xlim = range(bcoo[, 1]),
ylim = range(bcoo[, 2]), dims = nxy)

tcoo = gproj$lattice$loc # target points for prediction

plot(mapdf.km) # region of interest
points(tcoo, pch = 3, cex = .01) # prediction points
points(coo, cex = 0.5, pch = 4, col = "blue") # observed points
```



## Prediction using stack

Prediction using stack is a computationally expensive way because INLA computes posterior marginal in each grid cell. * We have defined mesh, spde model, and projector matrix for the prediction on the lattice. * We need to define a full stack based on the estimation stack and prediction stack and fit the model.

## Full stack

```
stk.full <- create_stack_full(tag = "pred", gproj = gproj, stk.est = stack.a, spde = spde)
```

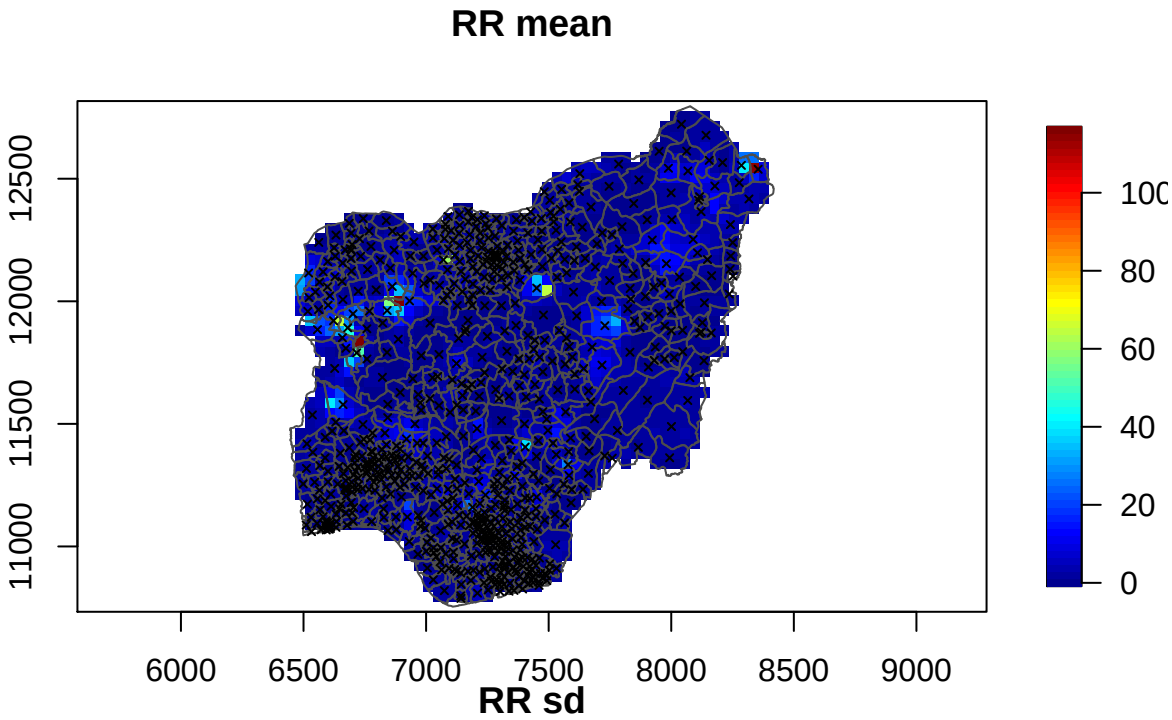
## Fit the prediction (joint) model

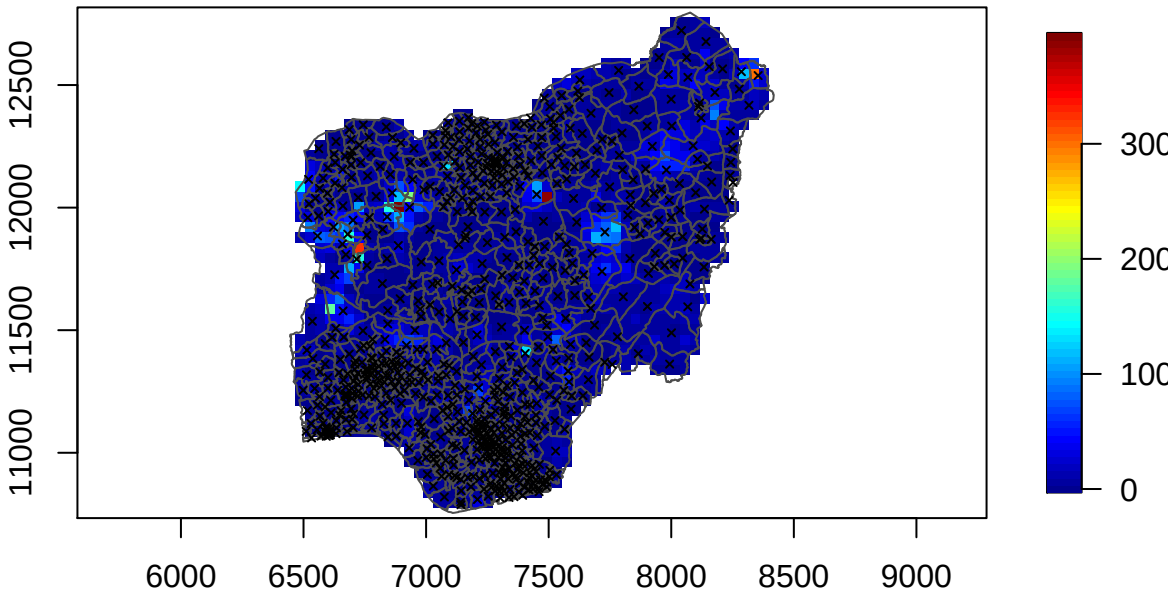
Fitting using joint stack takes 105 min.

```
if(pred.stk == T){
  p.res.spde <- fit_spde_full(stk.full = stk.full, res.spde = res.spde, spde = spde)
}
```

### Results

```
if(pred.stk == T){
rr = extract_results(res = p.res.spde, tag = "pred",  grpoj = grpoj, plot = T, sp = map.km)
}
```




### Prediction using Monte Carlo samples

- Prediction using Monte Carlo samples is computationally cheaper, especially if the number of locations to be predicted is not small.
- Computational effort in the prediction using MC samples boils down to the sampling and matrix

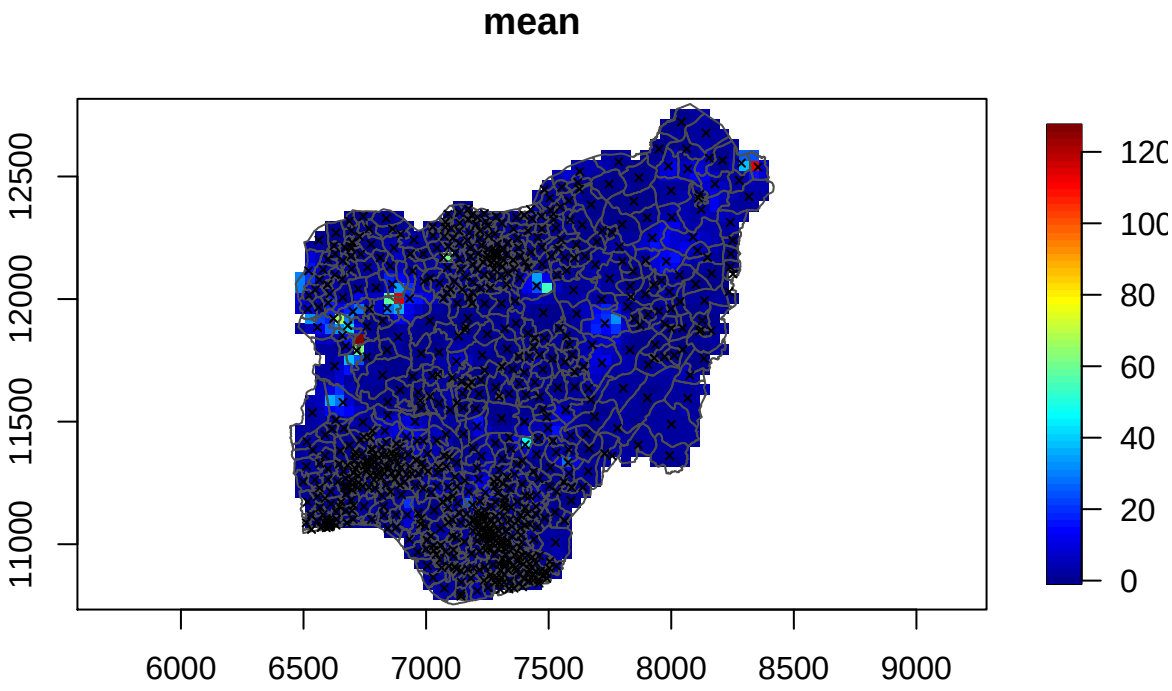
multiplication.

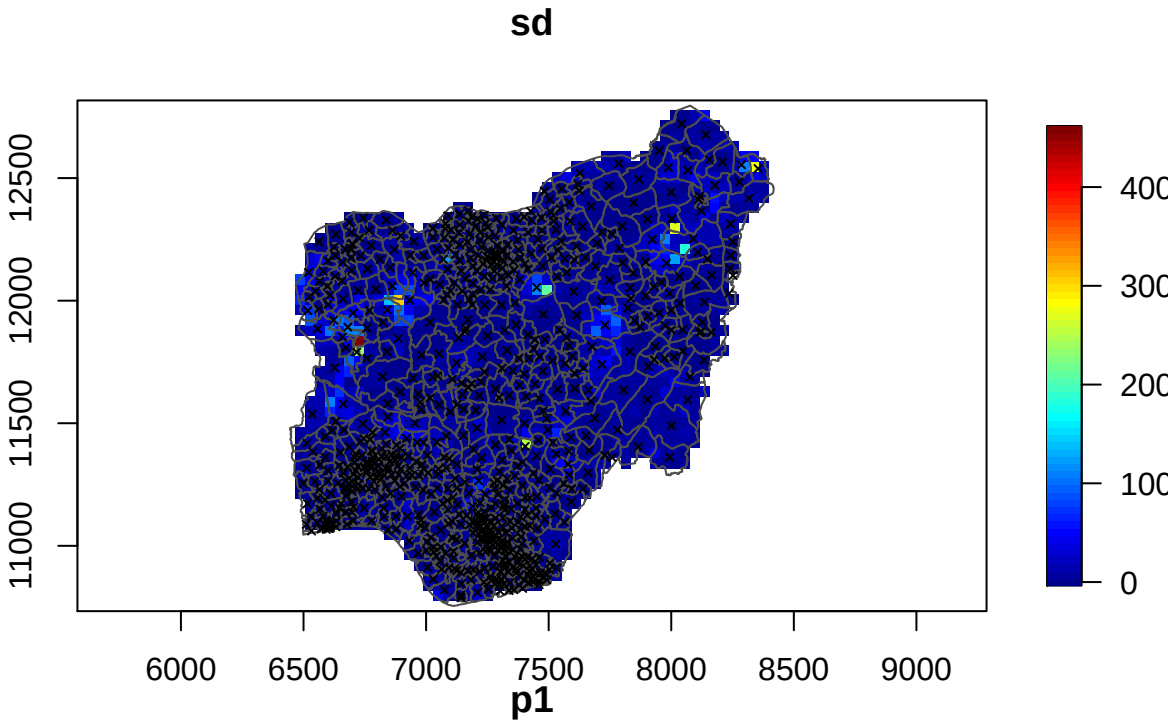
- We draw samples from the latent field: intercept, and spatial field. We collect the samples in the columns. Each column is pre-multiplied by the vector $(1, A_i)$. Vector $A_i$ is the ith row of the projector matrix for the prediction. *To obtain the relative risk values, we exponentiate the linear predictor.* Based on samples at every cell, we compute summary statistics, the mean, standard deviation, $P(E(Y) > 1)$ (proportion of times the samples that are bigger than 1).

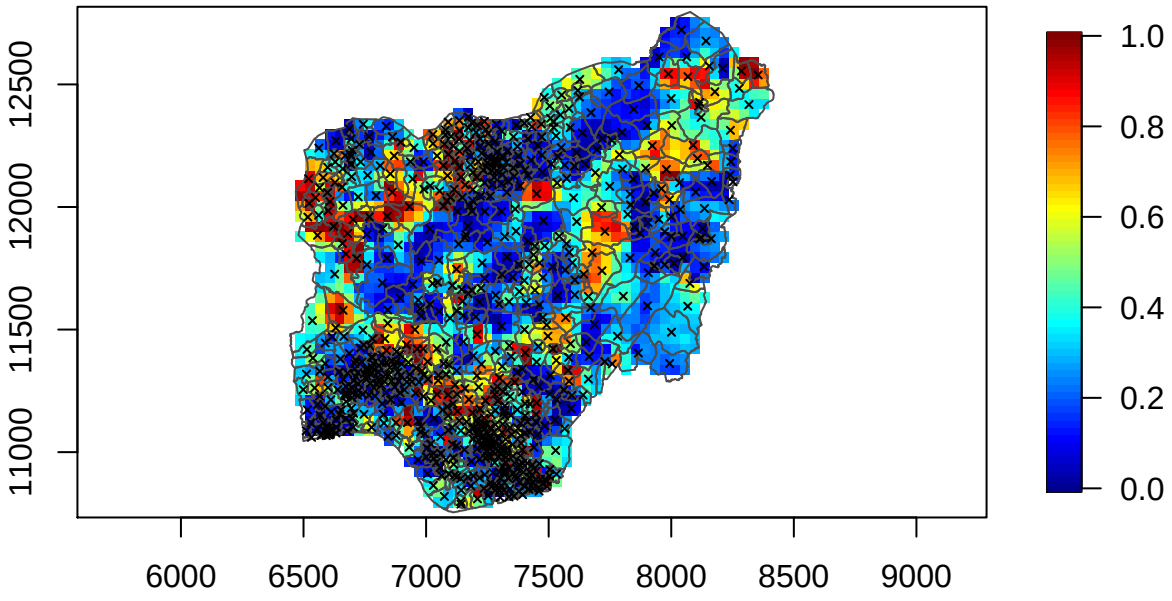
$$\eta^{(1)} = \log(\theta^{(1)}) = \mathbf{1}\beta_0^{(1)} + A\boldsymbol{u}^{(1)} = [\mathbf{1}; A] \times \begin{bmatrix} \beta_0^{(1)} \\ \boldsymbol{u}^{(1)} \end{bmatrix}$$

$$\theta^{(1)} = e^{\eta^{(1)}}$$

```
rrMC = MC_samples(result = res.spde, N = 2e3, plot = T, sp = map.km, gproj = gproj, coo = coo)
```






## References

Krainski, Elias T, Virgilio Gómez-Rubio, Haakon Bakka, Amanda Lenzi, Daniela Castro-Camilo, Daniel Simpson, Finn Lindgren, and Håvard Rue. 2018. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and Inla.* CRC Press.

Moraga, Paula. 2019. *Geospatial Health Data: Modeling and Visualization with R-Inla and Shiny.* CRC Press.

Moraga, Paula, Susanna M Cramb, Kerrie L Mengersen, and Marcello Pagano. 2017. "A Geostatistical Model for Combined Analysis of Point-Level and Area-Level Data Using Inla and Spde." *Spatial Statistics* 21.

Elsevier: 27–41.

Wilson, Katie, and Jon Wakefield. 2020. "Pointless Spatial Modeling." *Biostatistics* 21 (2). Oxford University Press: e17–e32.