Simulation study of the continious versus disceret disease mapping model

Kamila Kazimierska, Paula Moraga

December 2020

Simulation study of the continious versus discrete disease mapping models

Objective

The point-referenced dataset is measured at known locations. In spatial epidemiology, disease counts are the point-referenced data, but they are aggregated to the health districts or municipalities due to confidentiality reasons. In other words, the continuous process of the disease counts is discretized. In the simulation study, we simulate the continuous disease risk surface and counts. Continuous counts means the counts for the fine grid. We compare the performance of the BYM2 model and the model based on the SPDE for the simulated data aggregated into two different administrative levels.

References

Additional references to the one specified in the data analysis are:

- (F. Wang et al. 2019), simulation of the disease counts using coherent generative model.
- (Abrahamsen 1997), Gaussian random fields and correlation functions.
- (Guttorp and Gneiting 2006), history of the Matern correlation family.

Load data and auxiliary functions

The data consists of

- `pop.dat.km` raster object of the Nigeria population aggregated to 43485 cells, `NA` values were replaced by 0.

```
load(file.path(lap_data, "Data/pop.RData", fsep = ""))
pop.dat.km = setMinMax(pop.dat.km)
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
## *** source scenario_param.R
```

Scenarios

In each scenario for one of the specified models (`iid`, `bym2`, `spde`):

- Number of the replicated data sets is 100.
- The smoothness parameter of the spatial field is fixed, because we vary the standard error σ of the field.
- Using function `scenario_param`, we create 16 scenarios based on the four parameters.
- The possible values of the parameters are in the brackets. The parameters are: spatial fields' range (7.8% or 38% of the domain extent); spatial fields' variance (0.2 or 0.4); prevalence ($0.20 - 0.30\%$ or $1.45 - 1.60\%$ of the 180 mln population) encoded as an intercept β_0 (-6 or -4.3) and administrative level (level 1 with 37 states or level 2 with 773 Local Government Areas (LGA)).
- The boundaries of the spatial polygon `map.km` are used to determine the values for the range parameters.

```
map <- st_geometry(gb_adm1("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'))

scenarios = scenario_param(sp = map.km)

sum.scen = scenarios[[1]]
for(j in 2:16){
 sum.scen = rbind(sum.scen, scenarios[[j]])
}
kable(sum.scen, caption = "Summary of all scenarios")
```

Table 1: Summary of all scenarios

	range	sigma2u	beta0	adm.lev
sum.scen	121	0.2	-6	1
	121	0.2	-4.3	1
	121	0.4	-6	1
	121	0.4	-4.3	1
	600	0.2	-6	1
	600	0.2	-4.3	1
	600	0.4	-6	1
	600	0.4	-4.3	1
	121	0.2	-6	2
	121	0.2	-4.3	2
	121	0.4	-6	2
	121	0.4	-4.3	2
	600	0.2	-6	2
	600	0.2	-4.3	2
	600	0.4	-6	2
	600	0.4	-4.3	2

```
scenario.idx = 1
model = "bym2" # iid, spde
```

```
weights = "pop.int"
n.rep = 1
range = scenarios[[scenario.idx]]$range
sigma2 = scenarios[[scenario.idx]]$sigma2u

beta0 = scenarios[[scenario.idx]]$beta0
adm.lev = scenarios[[scenario.idx]]$adm.lev
```

Nigeria administrative boundaries

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

- `map.km`

```
if(adm.lev == 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)
```

Algorythm of the data simulation

Algorithm:

1. The raster `pop.dat.km` contains the population values n_k of Nigeria at the fine-scale level, $k = 1, \ldots, m$.
2. Let `coo` denote the spatial coordinates of the population's raster cells. Simulate the underlying Gaussian Field at m locations `coo`. We assume the spatially structured random effect follows a zero-mean Gaussian process with the Matérn correlation function. We use the pre-specified parameters for range and spatial field variance.

3. Simulate the linear predictor $\eta_k = \text{logit}(p_k)$, where p_k denote the incidence proportion (or 'reference' specific risk) at the kth raster cell. Linear predictor is additive with respect to other effects

$$\eta_k = \beta_0 + b_k, \tag{1}$$

where β_0 is an intercept, quantifying the average outcome rate in the study region, b_k a spatially structured random effect. We use the pre-specified parameter for the intercept. For example $\beta_0 = -6$ gives prevalence $0.20 - 0.30\%$ of the 180 mln population.

4. Create the risk surface. The relative risk value is computed as

$$r_k = \frac{p_k}{\bar{p}} \tag{2}$$

$$\bar{p} = \frac{\sum_k p_k n_k}{\sum_k n_k}. \tag{3}$$

5. Simulate the disease counts. Let Y_k denote the random variable at the kth cell, such that

$$Y_k|\eta_k \sim \text{Poisson}(n_k p_k = n_k \frac{e^{\eta_k}}{1 + e^{\eta_k}}). \tag{4}$$

$$\tag{5}$$

6. Aggregate the population and disease count values to the administrative levels. We consider two administrative levels with 37 or 773 areas.

7. Compute the expected counts for each area i, $i = 1, \ldots, n$

$$E_i = n_i \frac{\sum_i y_i}{\sum_i n_i},$$

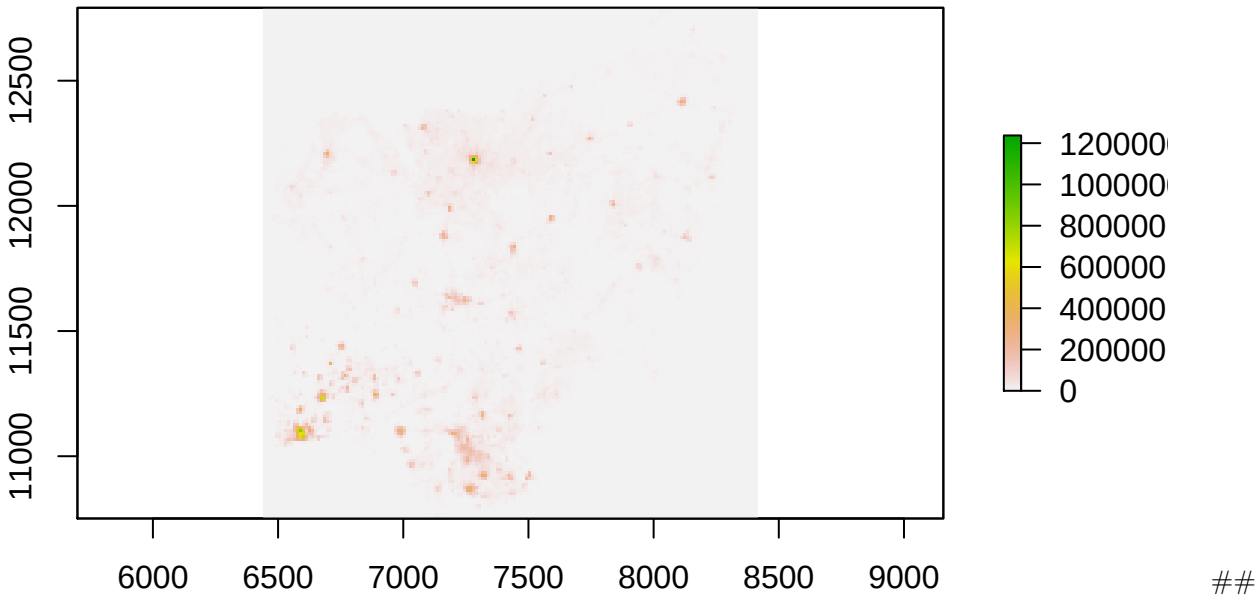
which implies

$$\sum_i E_i = \sum_i y_i.$$

1. Population of Nigeria

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

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



##

Combining map with data frame

- The `df` data frame contains `coordinates` of the Nigeria map, that are labelled as `ID`.

```
# 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))
# Polygons ids are ID i
mapdf.km = SpatialPolygonsDataFrame(map.km, df)
```

2. The Gaussian random field

(Abrahamsen 1997) review Gaussian random fields and correlation functions. A Gaussian random field (GF) is a stochastic process, $\{B(\boldsymbol{s}) : \boldsymbol{s} \in D \subset \mathbb{R}^2\}$.

We assume the Gaussian field $B(\boldsymbol{s})$ is

- continuous over space D. Continiouity implies, that it is possible to collect its realization at any finite set of locations.
- stationary and isotropic, i.e., its covariance function depends only on the distance h and it does not depend on the direction.

The realization of the GF, at finite set of locations $(b(\boldsymbol{s}_1), \ldots, b(\boldsymbol{s}_m))$ has a stationary multivariate Gaussian distribution, i.e

$$(b(\boldsymbol{s}_1), \ldots, b(\boldsymbol{s}_m)) \sim \text{MVN}(0, \boldsymbol{\Sigma}).$$

Matern cavariance function

(Guttorp and Gneiting 2006) present a history of the Matern family of spatial correlations.

Matern covariance function is of the form

$$\Sigma_{ij} = C(h) = \sigma^2 2^{1-\nu} \Gamma(\nu)^{-1} (\kappa h)^{\nu} K_{\nu}(\kappa h)$$

where

- κ is a scale parameter,

- ν is a smoothness parameter,
- σ^2 is the marginal variance of the process,
- $h = ||\boldsymbol{s}_i - \boldsymbol{s}_j||$ denote the Euclidian distance betweeen two spatial points,
- K_ν is the modified Bessel function of the second kind.

Simulation of the GF

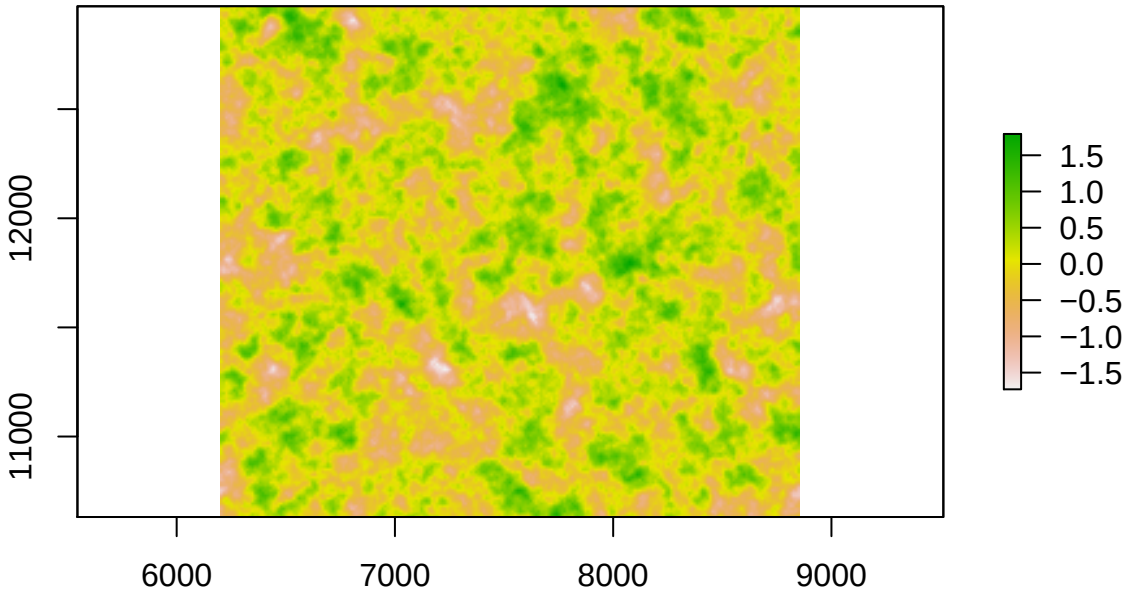
Using the `RandomFields` package, the simulation of the GF is computationally efficient. See `RMwhittlematern`, for the parametrization of the Matern correlation function. We use the scale parameter equal to half of the range, where range is distance that gives the correlation near 0.14 and has the expression

$$\text{range} = \frac{\sqrt{8\nu}}{\kappa}$$

```
coo = xyFromCell(pop.dat.km, 1:ncell(pop.dat.km)) # coordinates from all cells of raster r
colnames(coo) = c("coords.x1", "coords.x2")

m.model <- RMmatern(nu = 1, var = sigma2, scale = range/2)
u <-  RFsimulate(m.model, x= coo[,1], y= coo[,2])
```

```
## New output format of RFsimulate: S4 object of class 'RFsp';
## for a bare, but faster array format use 'RFoptions(spConform=FALSE)'.
```

```
r.GF = pop.dat.km # copy the raster but change the values
values(r.GF) <- u@data$variable1 #  assign the data values to the raster object
plot(r.GF)
```



3 - 7 Simulation of the coherent generative model

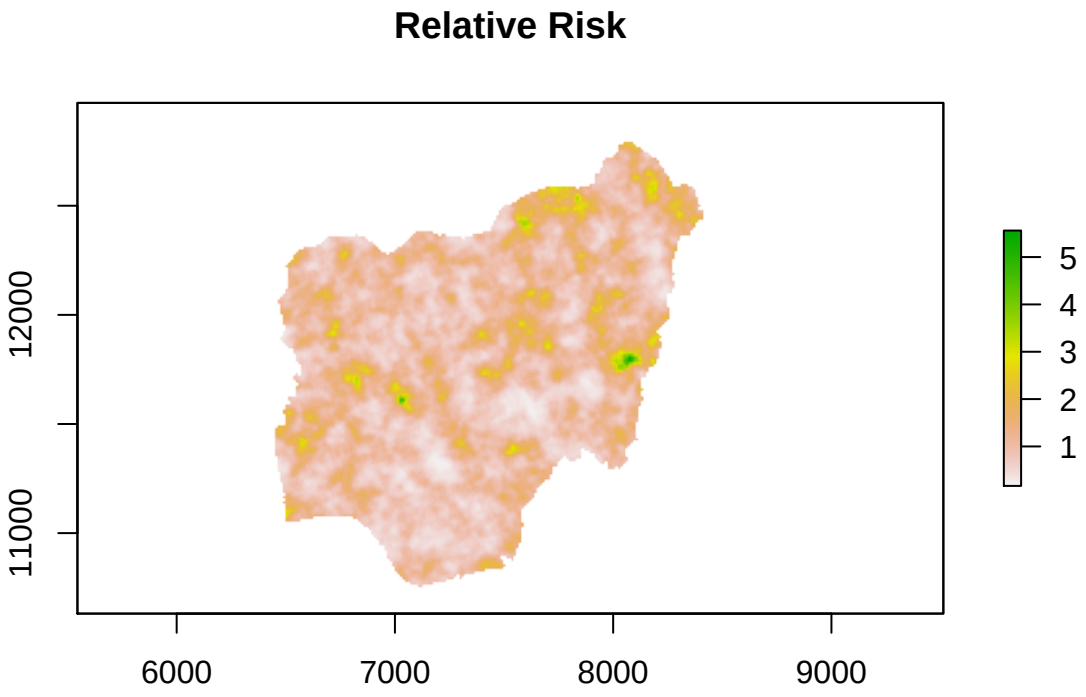
Simulate the risk surface at the fine-scale level, observed and the expected disease counts at the areal level.

```
  # 3. linear predictor
  eta = beta0 + values(r.GF)

  # 4. relative risk surface
  p.true = exp(eta)/(1+exp(eta))
```

```r
pop = values(pop.dat.km)
p.av.true = sum(pop*p.true)/sum(pop) # average probability
rr.true = p.true/p.av.true # relative risk

# 5. disease counts based on true p and n
ysim = rpois(dim(coo)[1], lambda = pop*p.true)
r.y <- pop.dat.km
values(r.y) <- ysim

# 6. pop and counts aggregation
mapdf.km@data$pop <- extract(pop.dat.km, map.km, fun = sum, na.rm = TRUE)
mapdf.km@data$count <- extract(r.y, map.km, fun = sum, na.rm = TRUE)

# 7. expected counts
E = mapdf.km@data$pop*sum(mapdf.km@data$count)/sum(mapdf.km@data$pop)
mapdf.km@data$E <- E
```

Visualization

```r
r.RRt = pop.dat.km
values(r.RRt) = rr.true
r.RRt <- mask(r.RRt, map.km)
plot(r.RRt, main = c("Relative Risk"))
```



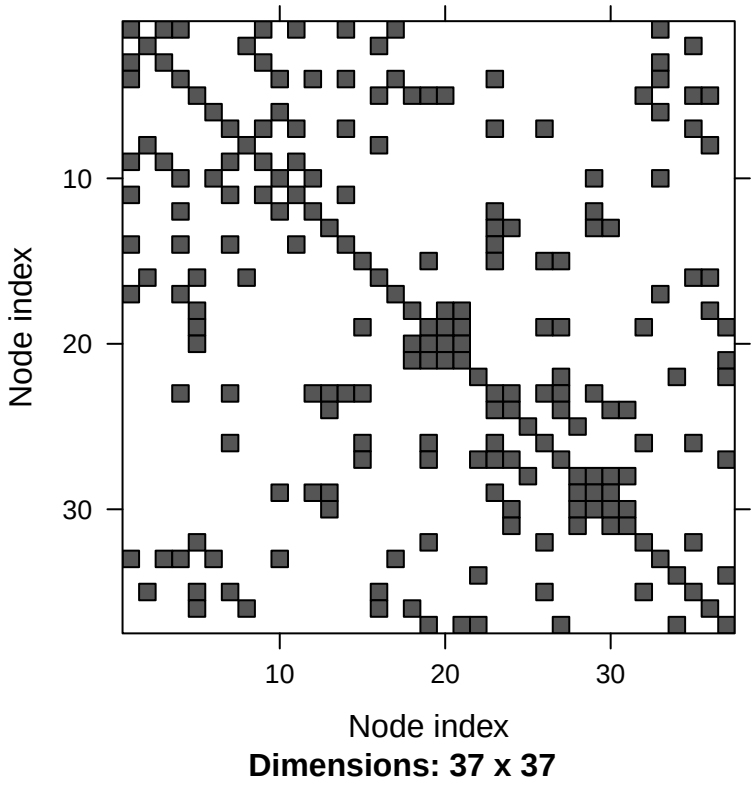
Inference with INLA - iid, BYM2, SPDE

For a given scenario, we perform the inference for one of the specified models: `iid`, `bym2`, or `spde`.

Precision matrix for BYM2

A spatial adjacency matrix is required for `bym2` model fitting. We create the graph and adjacency matrix based on the neighboring structure encoded in the spatial polygon `map.km`. The graph `ng.graph` is defined once for all simulations.

```
# create rook (right) adjacency matrix, the simplest, condiering neighbours
adj.r <- poly2nb(map.km, queen = FALSE) # not a queen
# convert nb object to graph object compatible with INLA, save it and read it
nb2INLA(file.path(lap_data, "graph", eval(scenario.idx), eval(model), fsep = ""), adj.r)
ng.graph = inla.read.graph(file.path(lap_data, "graph", eval(scenario.idx), eval(model), fsep = ""))

image(inla.graph2matrix(ng.graph), xlab = "Node index", ylab = "Node index")
```

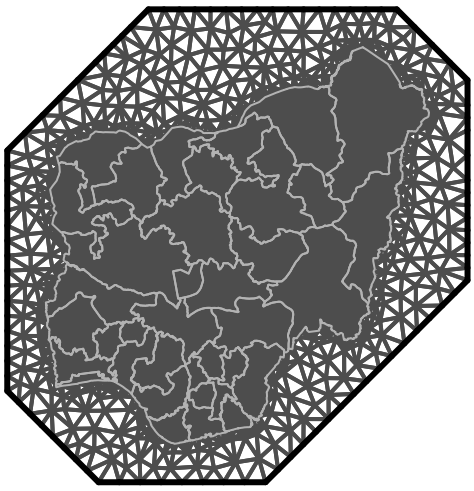


Mesh, SPDE model and projector matrix

- If we use the `spde` model, we need to define triangulation of the domain `mesh`, `spde` model with the prior specification, projector matrix for areal data, `Aa` and prediction lattice and matrix. These are defined once for all simulations.
- The `mesh` is defined based on the spatial polygon `map.km`
- The `spde` model priors are defined based on true value $\sigma \in \{0.2, 0.4\}$ and the true value of range$\in \{121, 600\}$.
- Projector matrix `Aa` is created based on the spatial polygons `map.km` and the mesh nodes.

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

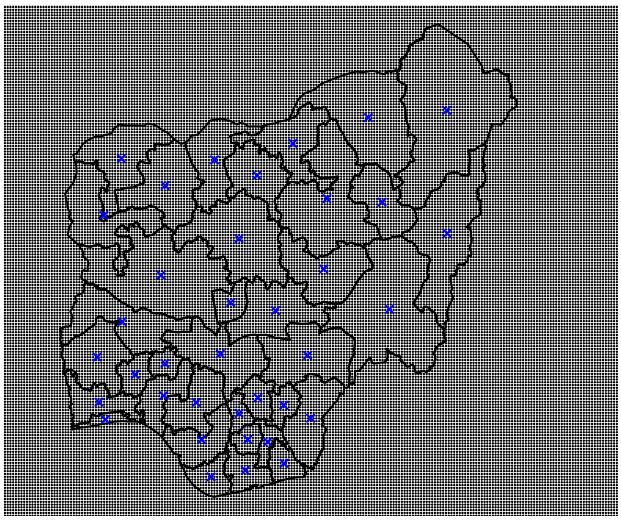
Constrained refined Delaunay triangulation



```
# spde model
spde <- inla.spde2.pcmatern(
    mesh, alpha=2,
    prior.sigma=c(1, 0.01),  # P(sigma > 1) = 0.01,
    prior.range=c(range, 0.5)) # P(r < range) = 0.5

# projector matrix
  if(weights == "equal" ){
  Aa <- create_matrix(sp = map.km, mesh = mesh, weights = "equal")
  } else {
  Aa <- create_matrix(sp = map.km, mesh = mesh, weights = "pop.int", pop.raster = pop.dat.ng)
  }
```

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

```
# prediction lattice and matrix
# coo are the coordinates of the population raster
Ap = inla.spde.make.A(mesh = mesh, loc = coo)

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



Fitting - SPDE, BYM2

- Stack and spde model are required for the spde model fitting.
- Due to the computational efficiency, the Monte Carlo samples are used to predict the continuous risk surface at the population's raster coordinates.
- Neighbouring structure graph is required for the BYM2 model fitting. The predictions of the risk surface are obtained at the administrative boundaries level.

```
# SPDE
stack.a <- create_stack(tag = "area", matrix = Aa, spdf = mapdf.km, spde = spde)
res <- fit_spde(spde = spde, stack = stack.a)
rrMC = MC_samples(result = res, N = 2e3, Ap = Ap)

# BYM2
formula.model <- count ~ 1 + f(ID, model = "bym2", graph = ng.graph,
                        constr = T, scale.model = T,
                        hyper=list(prec=list(prior="pc.prec", param=c(1, 0.01)), # P(sd >1) = 0.01
                                   phi =list(prior="pc", param=c(0.5, 0.5)))) # P(phi <0.5) = 0.5
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(0.1, 0.9))
```

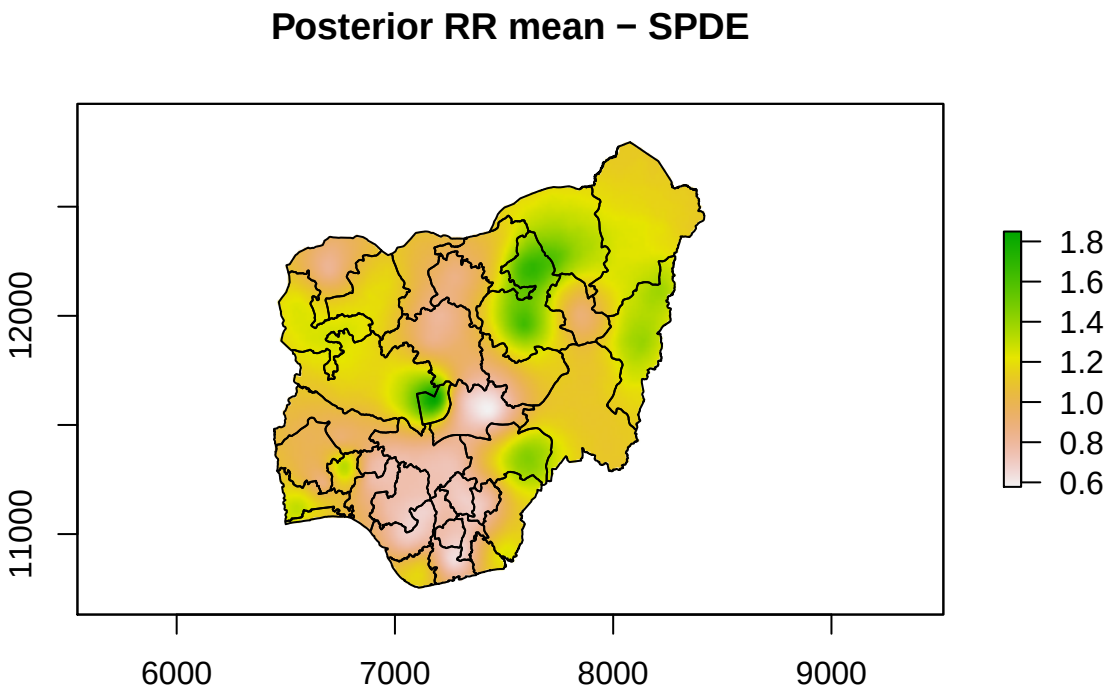
```
## Warning in inla.model.properties.generic(inla.trim.family(model), mm[names(mm) == : Model 'bym2' in s
##   Use this model with extra care!!! Further warnings are disabled.
```

```
mapdf.km@data$RR.m <- res$summary.fitted.values[, "mean"]
mapdf.km@data$RR.qu <- res$summary.fitted.values[, "0.9quant"]
mapdf.km@data$RR.ql <- res$summary.fitted.values[, "0.1quant"]
```

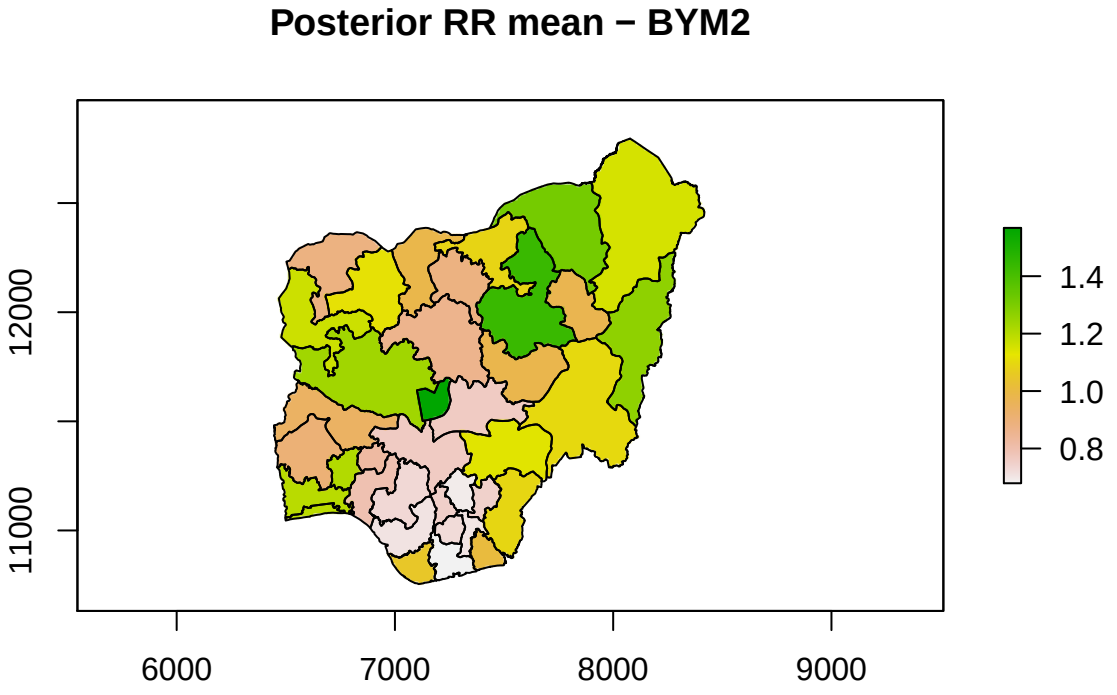
Relative risk mapping - SPDE, BYM2

```
# visualize the spde prediction
r.RRm <- r.RRt
values(r.RRm) <- rrMC$rr$mean
r.RRm = mask(r.RRm, map.km)
```

```r
plot(r.RRm, main = "Posterior RR mean - SPDE")
lines(map.km)
```




```r
# visualize the bym2 prediction
r.RRm <- rasterize(mapdf.km, pop.dat.km, field = "RR.m", fun = "sum")
# fun "sum"" or default "last" gives the same value beacuse we rasterize the polygons
r.RRm <- mask(r.RRm, map.km) # bc of NA
plot(r.RRm, main = "Posterior RR mean - BYM2")
lines(map.km)
```



Simulation study

In the simulation study, we replicate the dataset `n.rep` number of times and compare the models based on the model performance measures.

Model performance measures

Let

$$\boldsymbol{r} = (r_1, \ldots, r_m)$$

denote the vector of the relative risk at the high resolution grid inside the spatial polygon of interest, where m is the number of cells. Let $\hat{r}_k^j$ denote the posterior mean of the fitted relative risk at kth cell and jth simulation, where \$k = 1, \ldots, m \$ and $j = 1, \ldots, 100$. For the iid or bym2 model, this estimate has the same value for all cells that fall in the area. The absolute error, which depends on jth simulation is

$$\text{AE}(r_k^j) = |\hat{r}_k^j - r_k|$$

The overall mean absolute error for the relative risk is computed as

$$\text{MAE}(\boldsymbol{r}) = \frac{1}{100}\sum_{j}^{100}\frac{1}{m}\sum_{k}^{m}\text{AE}(r_k^j)$$

In a given simulation j

$$\text{coverage}(r_k) = \begin{cases} 1 & r_k \in (q_l(\hat{r}_k^j), q_u(\hat{r}_k^j)) \\ 0 & \text{ow.} \end{cases}$$

The overall coverage is computed as

$$\text{coverage}(\boldsymbol{r}) = \frac{1}{100}\sum_{j=1}^{100}\frac{1}{m}\sum_{k=1}^{m}\text{coverage}(r_k).$$

Additionally, we compute the credible interval lengths and their mean across cells and simulations.

```
# globally defined
# m.model, coo, r.GF, pop, r.RRt, r.y, mapdf.km@data$pop pop.dat.km, map.km, r.RRm

sim.coverage = function(id) {

  ## simulate the data
  u <-  RFsimulate(m.model, x= coo[,1], y= coo[,2]) #  update the GF values
  values(r.GF) <- u@data$variable1
  # 3. linear predictor
  eta = beta0 + values(r.GF)
  # 4. relative risk surface
  p.true = exp(eta)/(1+exp(eta))
  p.av.true = sum(pop*p.true)/sum(pop) # average probability
  rr.true = p.true/p.av.true # relative risk
  values(r.RRt) = rr.true # update the simulated relative risk values
  r.RRt <- mask(r.RRt, map.km) # contains NA
  # 5. disease counts based on true p and n
```

```
 ysim = rpois(dim(coo)[1], lambda = pop*p.true)
 values(r.y) <- ysim
 # 6. pop and counts aggregation
 mapdf.km@data$count <- extract(r.y, map.km, fun = sum, na.rm = TRUE)
 # 7. expected counts
 E = mapdf.km@data$pop*sum(mapdf.km@data$count)/sum(mapdf.km@data$pop)
 mapdf.km@data$E <- E

 ## INFERENCE and prediction
 if(model == "spde"){
 stack.a <- create_stack(tag = "area", matrix = Aa, spdf = mapdf.km, spde = spde)
 res <- fit_spde(spde = spde, stack = stack.a)
 # prediction and CI
 rrMC = MC_samples(result = res, N = 2e3, Ap = Ap)
 r.RRm <- r.RRt
 values(r.RRm) <- rrMC$rr$mean
 r.RRm = mask(r.RRm, map.km) # contains NA

 r.RRql = r.RRqu = r.RRm
 values(r.RRql) = rrMC$rr$q1
 r.RRql = mask(r.RRql, map.km)
 values(r.RRqu) = rrMC$rr$q9
 r.RRqu = mask(r.RRqu, map.km)
 CI <- cbind(values(r.RRql), values(r.RRqu))
 } else {
 if(model == "bym2"){
 formula.model <- count ~ 1 + f(ID, model = "bym2", graph = ng.graph,
                       constr = T, scale.model = T,
                       hyper=list(prec=list(prior="pc.prec", param=c(1, 0.01)), # P(sd >1) = 0.01
                                  phi =list(prior="pc", param=c(0.5, 0.5)))) # P(phi <0.5) = 0.5)
} else {
 formula.model <- count ~ 1 + f(ID, model = "iid")
}
 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(0.1, 0.9))
 # prediction and CI
 mapdf.km@data$RR.m <- res$summary.fitted.values[, "mean"]
 mapdf.km@data$RR.qu <- res$summary.fitted.values[, "0.9quant"]
 mapdf.km@data$RR.ql <- res$summary.fitted.values[, "0.1quant"]
 r.RRm <- rasterize(mapdf.km, pop.dat.km, field = "RR.m", fun = "sum") # has NA
 r.RRql <- rasterize(mapdf.km, pop.dat.km, field = "RR.ql", fun = "sum") # CI
 r.RRqu <- rasterize(mapdf.km, pop.dat.km, field = "RR.qu", fun = "sum") # CI
 CI <- cbind(values(r.RRql), values(r.RRqu))
 }

 ## model performance measures
 # absolute error
```

```
  ae = abs(values(r.RRt) - values(r.RRm)) # true - posterior mean
  # credible interval lenght
  cov.len = CI[,2] - CI[,1]
  # binary coverage vector
  cov.bin <- c()
  for(idx in 1:length(r.RRt)){
   cov.bin[idx] = between(values(r.RRt)[idx], CI[idx, 1], CI[idx, 2])
  }

  # save the perf measures for cells inside spatial polygons
  return(list(cov.bin = cov.bin[!is.na(cov.bin)],
              cov.len = cov.len[!is.na(cov.len)],
              ae = ae[!is.na(ae)]))

}
```

Results for one simulation in a given scenario.

```
perf = sim.coverage(id = 1)
mean(perf$cov.bin)
```

```
## [1] 0.02370137
```

```
mean(perf$ae)
```

```
## [1] 0.3263252
```

```
mean(perf$cov.len)
```

```
## [1] 0.02297105
```

Computations on the server

The simulations were done on the server, running 64 tasks in parallel and using 125 GB memory. Fitting 46 scenarios took approximately X hours. The code below is presented, but not executed. The obtained list of lists `res.sim.raw` was processed. Each list (for a given simulation) consited of `cov.bin`, `cov.len` and `ae` vectors of size m.

For each scenario, we saved the matrices `cov.bin`, `cov.len` and `ae` in `mp.Rdata` files, which are processed in the next subsection.

```
## Run the model for n.rep number of datasets
res.sim.raw = mclapply(as.list(1:n.rep), mc.cores=3, FUN = sim.coverage)
m = length(res.sim.raw[[1]]$cov.bin)

## Extract model performance matrices of dimension n.rep x m
cov.bin <- matrix(NA, nrow = n.rep, ncol = m)
cov.len <- matrix(NA, nrow = n.rep, ncol = m)
ae <- matrix(NA, nrow = n.rep, ncol = m)

for(id in 1:n.rep){
  # index of the list corresponds to the sim index
  cov.bin[id, ] = res.sim.raw[[id]]$cov.bin
  cov.len[id, ] = res.sim.raw[[id]]$cov.len
  ae[id, ] = res.sim.raw[[id]]$ae
}
```

```
## Save the results
save(cov.bin, cov.len, ae, file = file.path( "Results/mp", eval(scenario.idx),
                                             eval(model), ".RData", fsep = ""))
```

Server computaion details

The script without plots and text was run using the terminal:

- Log in to the server using the `ssh server` command.
- Execute the *shell* scripts using `./run.jobs` command where the scripts `run.jobs` is as follows

```
#!/bin/bash -x
for sc in $(seq 1 16); do
    for lik in "spde" "bym2" "iid"; do
        nice Rscript simraw.R $sc $lik  > LOG-$sc-$lik 2>&1 &
    done
done
```

- To be able to execute the `simraw.R` script over the scenario and model arguments, we need to define the `input` at the beginning of the `simraw.R` script:

```
input = commandArgs(trailingOnly=TRUE)
scenario.idx = as.integer(input[1])
model = input[2]
```

- The `input[1]` will take the value of one of the 16 scenarios, the `input[2]` will take the value of one of the three models.
- Both `inputs` are passed to the script. Hence, we can run the scripts in parallel for different input combinations.
- The results are saved depending on the scenario and the model.

Processing the results - read the results

- We read the results, that were saved from the runs on the server.
- For each of the 48 scenarios, we obtained the model performance measures: credible interval length, binary vector if the true value falls in CI, and absolute error.
- For a given scenario and for all 100 replicate datasets, we have three matrices of size $100 \times m$, where m is a number of raster cells inside the spatial polygon of interests.
- We create `cov.bin.s`, `cov.len.s` and `ae.s` lists to store the matrices from all scenarios.

```
cov.bin.s = list()
cov.len.s = list()
ae.s = list()
combination = list()
idx = 1
set = seq(1:16)

for(scenario.idx in set){
  for( model in c("iid", "bym2", "spde")){
   ## Save the results
   load(file.path(lap_data, "Results/mp", eval(scenario.idx),eval(model), ".RData", fsep = ""))
   cov.bin.s[[idx]]  = cov.bin
   cov.len.s[[idx]]  = cov.len
   ae.s[[idx]]  = ae
   combination[[idx]] = paste(scenario.idx, model)
```

```
   idx = idx +1
  }
}

for(i in 1:48){
  print(mean(as.matrix(ae.s[[i]])))
}
```

```
## [1] 0.3328423
## [1] 0.321012
## [1] 0.3108334
## [1] 0.3395906
## [1] 0.3034095
## [1] 0.3404393
## [1] 0.4735788
## [1] 0.4830954
## [1] 0.5262924
## [1] 0.4871375
## [1] 0.4389613
## [1] 0.4807998
## [1] 0.194399
## [1] 0.1791598
## [1] 0.1595507
## [1] 0.2253738
## [1] 0.1735789
## [1] 0.1208049
## [1] 0.335453
## [1] 0.2224728
## [1] 0.2494831
## [1] 0.2053218
## [1] 0.2718047
## [1] 0.2215681
## [1] 0.2072739
## [1] 0.2454797
## [1] 0.1867787
## [1] 0.2385918
## [1] 0.2181303
## [1] 0.1834985
## [1] 0.2927382
## [1] 0.3360134
## [1] 0.2371041
## [1] 0.3160528
## [1] 0.3342717
## [1] 0.2441966
## [1] 0.0956367
## [1] 0.08792073
## [1] 0.0819891
## [1] 0.08278846
## [1] 0.07849945
## [1] 0.05598528
## [1] 0.1180204
## [1] 0.1371319
## [1] 0.09889359
## [1] 0.1131934
```

```
## [1] 0.1433651
## [1] 0.07978843
```

Boxplot of the absolute error

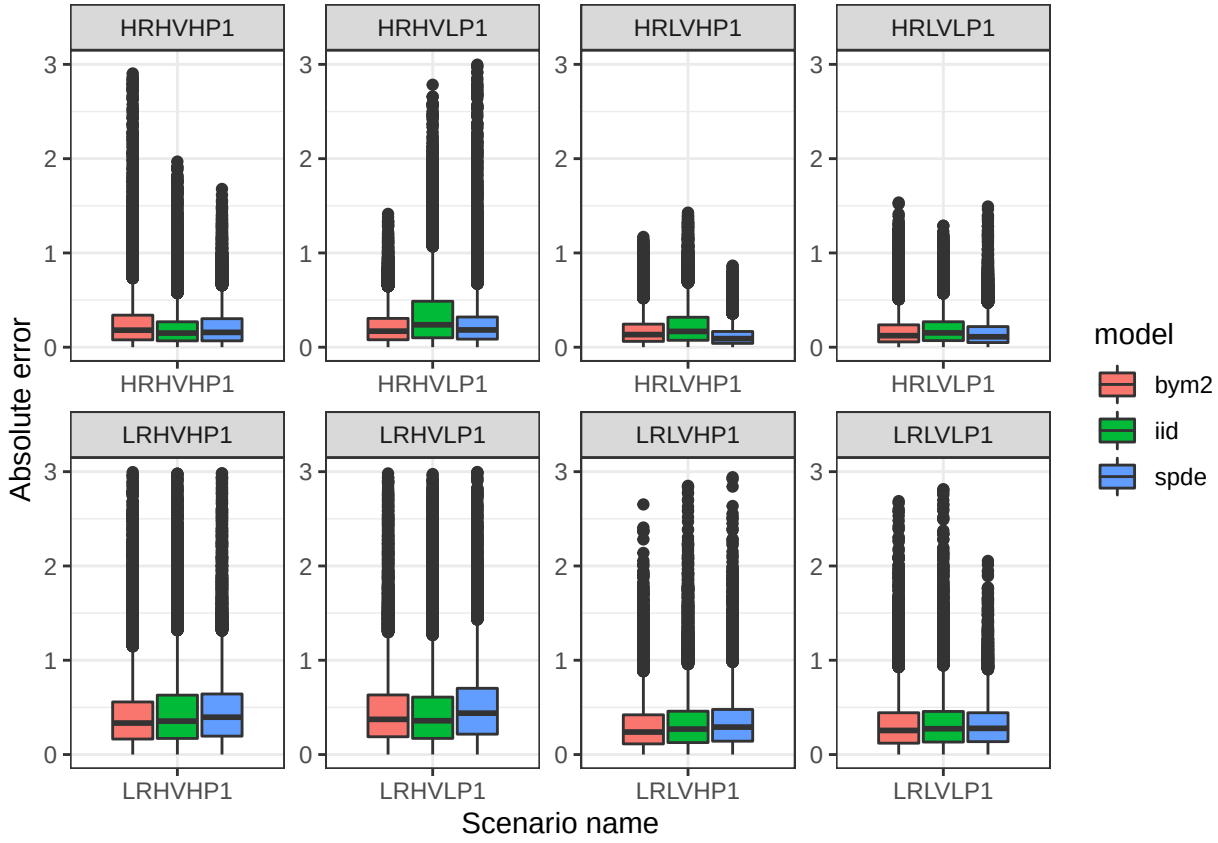
Let `n.mod` denotes the number of the models considered, `m` number of raster cells inside the spatial polygon of interests, and `n.rep` number of replicated data sets. We create a boxplot of the absolute error obtained for m cells across $n.rep$ simulations. We present boxplots for three analyzed models together. We create two different plots for all scenarios.

To do so, we create a data frame consisting of 3 columns:

- The first column indicate `scenario` (from 1 to `n.sc`). Each `scenario` index is repeated `n.mod` $\times$ `m` $\times$ `n.rep` number of times.
- The second column is a vector denoting the model name. Each model name is repeated `m` $\times$ `n.rep` number of times per each scenario.
- For a given scenario and a given model, we obtain a vector (vectorized matrix) of the absolute errors of size `m` $\times$ `n.rep`.

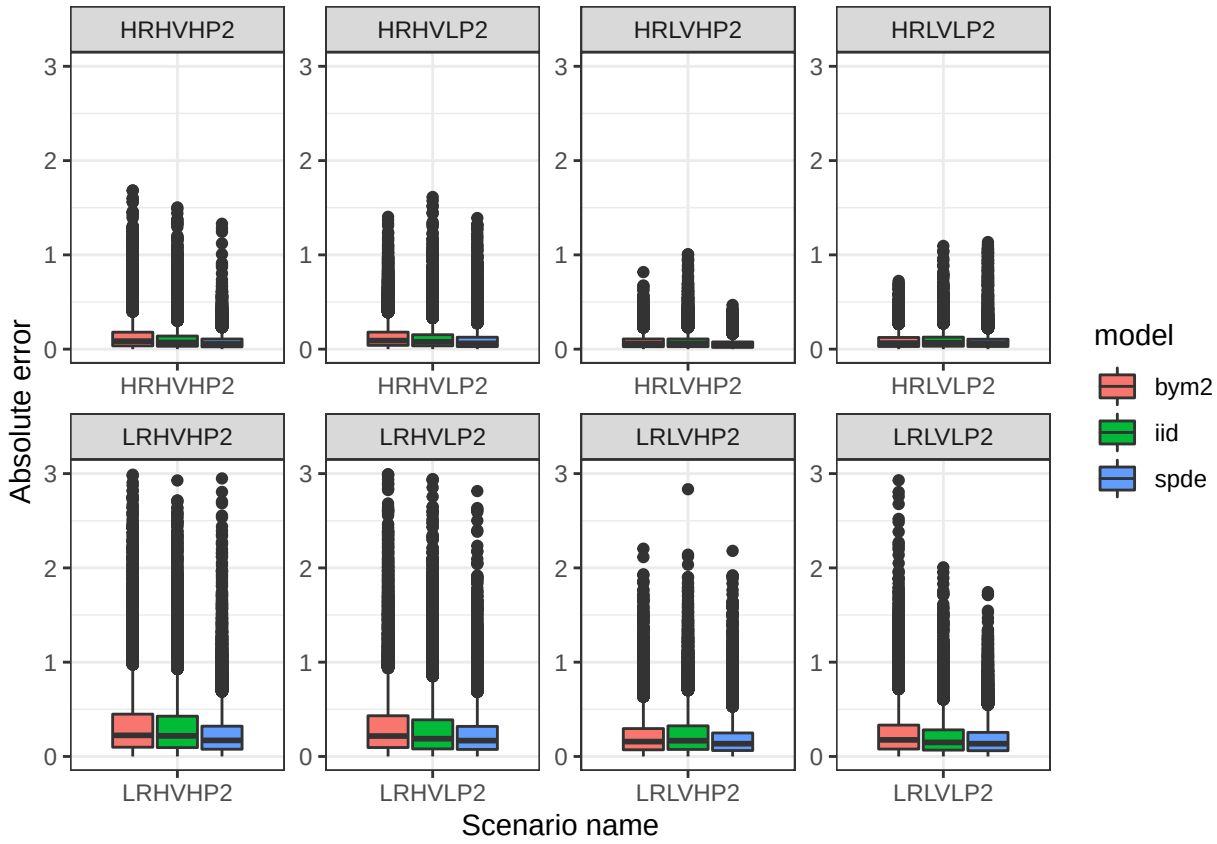
```
n.sc = 16
n.mod = 3
n.rep = dim(ae.s[[1]])[1]
m = dim(ae.s[[1]])[2]

## create a data frame
large.ae = c()
# define the scenario names
sc.names = c("LRLVLP1", "LRLVHP1","LRHVLP1", "LRHVHP1",
             "HRLVLP1", "HRLVHP1","HRHVLP1", "HRHVHP1",
             "LRLVLP2", "LRLVHP2","LRHVLP2", "LRHVHP2",
             "HRLVLP2", "HRLVHP2","HRHVLP2", "HRHVHP2")
dt = data.frame(scenario = as.character(rep(sc.names, each = n.mod*m*n.rep)),
                model = rep(c("iid", "bym2", "spde"), each = m*n.rep))
for(i in 1:length(combination)){
  large.ae = c(large.ae,as.vector(as.matrix(ae.s[[i]])))
}
dt$ae = large.ae


# split scenarios in two groups by name
set = list(g1 = sc.names[1:8], g2 = sc.names[9:16])

s = 1
p <- list()

# to save the plots at the selected path do not save them under list p
for(v in set){
  p[[s]] <- ggplot(dt[dt[,"scenario"] %in% v,],
          aes(x = scenario, y = ae, fill = model)) + geom_boxplot() +
          facet_wrap(~scenario, scale = "free", ncol = 4) + theme_bw() + ylab("Absolute error") +
          xlab("Scenario name") + ylim(c(0, 3))
  # ggsave(file.path(lap_data, "ae", names(set)[s], ".jpeg", fsep = ""))
  s = s+1
}
p[[1]]
```

```
## Warning: Removed 465 rows containing non-finite values (stat_boxplot).
```




```
p[[2]]
```

```
## Warning: Removed 59 rows containing non-finite values (stat_boxplot).
```



Model performace summary for all scenarios

The comparison is done overall row and column - wise average of `n.rep` × `m`.

```
ss = matrix(NA, nrow = length(combination), ncol = 10)

ss[,1]<- unlist(combination)
ss[,2] <- rep(unlist(sum.scen[, "range"]), each = 3) # we have three models
ss[,3] <- rep(unlist(sum.scen[, "sigma2u"]), each = 3)
ss[,4] <- rep(unlist(sum.scen[, "beta0"]), each = 3)
ss[,5] <- rep(unlist(sum.scen[, "adm.lev"]), each = 3)

for(i in 1:length(combination)){
  ss[i,6] = round(mean(as.matrix(cov.bin.s[[i]])),3) # overal
  ss[i,7] = round(mean(as.matrix(cov.len.s[[i]])),3) # overal
  ss[i,8] = round(mean(as.matrix(ae.s[[i]]), na.rm = TRUE),3)
  ss[i,9] = round(quantile(as.matrix(ae.s[[i]]),0.025, na.rm = TRUE),3)
  ss[i,10] = round(quantile(as.matrix(ae.s[[i]]),0.975, na.rm = TRUE),3)

}

colnames(ss) <- c("Combination","range","sigma2u", "beta0",
                                "adm.lvl", "Coverage", "CIs lenght", "MAE", "AEq0.05", "AEq0.95")
```

Averages of the $100 \times m$ performance measures across all scenarios are:

```
kable(ss, caption = "Summary results across all models and scenarios")
```

Table 2: Summary results across all models and scenarios

Combination	range	sigma2u	beta0	adm.lvl	Coverage	CIs lenght	MAE	AEq0.05	AEq0.95
1 iid	121	0.2	-6	1	0.022	0.022	0.333	0.013	1.057
1 bym2	121	0.2	-6	1	0.024	0.023	0.321	0.012	1.056
1 spde	121	0.2	-6	1	0.799	1.034	0.311	0.013	0.824
2 iid	121	0.2	-4.3	1	0.011	0.01	0.34	0.012	1.155
2 bym2	121	0.2	-4.3	1	0.012	0.01	0.303	0.01	0.971
2 spde	121	0.2	-4.3	1	0.774	1.053	0.34	0.016	0.915
3 iid	121	0.4	-6	1	0.016	0.022	0.474	0.017	1.803
3 bym2	121	0.4	-6	1	0.014	0.023	0.483	0.021	1.669
3 spde	121	0.4	-6	1	0.736	1.497	0.526	0.023	1.565
4 iid	121	0.4	-4.3	1	0.006	0.01	0.487	0.016	1.777
4 bym2	121	0.4	-4.3	1	0.006	0.01	0.439	0.017	1.544
4 spde	121	0.4	-4.3	1	0.683	1.261	0.481	0.019	1.486
5 iid	600	0.2	-6	1	0.037	0.021	0.194	0.007	0.636
5 bym2	600	0.2	-6	1	0.044	0.022	0.179	0.006	0.672
5 spde	600	0.2	-6	1	0.755	0.462	0.16	0.005	0.586
6 iid	600	0.2	-4.3	1	0.02	0.01	0.225	0.006	0.776
6 bym2	600	0.2	-4.3	1	0.017	0.009	0.174	0.007	0.554
6 spde	600	0.2	-4.3	1	0.917	0.528	0.121	0.004	0.414
7 iid	600	0.4	-6	1	0.026	0.023	0.335	0.009	1.148
7 bym2	600	0.4	-6	1	0.037	0.022	0.222	0.007	0.735
7 spde	600	0.4	-6	1	0.798	0.78	0.249	0.008	0.894
8 iid	600	0.4	-4.3	1	0.017	0.01	0.205	0.007	0.752
8 bym2	600	0.4	-4.3	1	0.014	0.008	0.272	0.007	1.166
8 spde	600	0.4	-4.3	1	0.72	0.64	0.222	0.007	0.823
9 iid	121	0.2	-6	2	0.183	0.101	0.207	0.007	0.725
9 bym2	121	0.2	-6	2	0.166	0.11	0.245	0.008	0.891
9 spde	121	0.2	-6	2	0.609	0.414	0.187	0.006	0.669
10 iid	121	0.2	-4.3	2	0.082	0.049	0.239	0.006	0.879
10 bym2	121	0.2	-4.3	2	0.084	0.045	0.218	0.007	0.79
10 spde	121	0.2	-4.3	2	0.611	0.423	0.183	0.006	0.635
11 iid	121	0.4	-6	2	0.133	0.092	0.293	0.008	1.156
11 bym2	121	0.4	-6	2	0.13	0.11	0.336	0.01	1.403
11 spde	121	0.4	-6	2	0.612	0.556	0.237	0.007	0.863
12 iid	121	0.4	-4.3	2	0.062	0.047	0.316	0.008	1.188
12 bym2	121	0.4	-4.3	2	0.057	0.049	0.334	0.01	1.269
12 spde	121	0.4	-4.3	2	0.633	0.593	0.244	0.007	0.937
13 iid	600	0.2	-6	2	0.406	0.114	0.096	0.003	0.362
13 bym2	600	0.2	-6	2	0.348	0.089	0.088	0.003	0.301
13 spde	600	0.2	-6	2	0.665	0.205	0.082	0.003	0.287
14 iid	600	0.2	-4.3	2	0.173	0.042	0.083	0.003	0.313
14 bym2	600	0.2	-4.3	2	0.19	0.04	0.078	0.003	0.272
14 spde	600	0.2	-4.3	2	0.678	0.144	0.056	0.002	0.198
15 iid	600	0.4	-6	2	0.379	0.129	0.118	0.003	0.465
15 bym2	600	0.4	-6	2	0.264	0.098	0.137	0.004	0.543
15 spde	600	0.4	-6	2	0.672	0.246	0.099	0.003	0.404
16 iid	600	0.4	-4.3	2	0.168	0.048	0.113	0.003	0.501
16 bym2	600	0.4	-4.3	2	0.18	0.055	0.143	0.003	0.661
16 spde	600	0.4	-4.3	2	0.696	0.225	0.08	0.002	0.291

```r
write.csv(ss,file = file.path(lap_data, "Results/summaryscenario.csv", fsep = ""))
```

References

Abrahamsen, Petter. 1997. "A Review of Gaussian Random Fields and Correlation Functions." Norsk Regnesentral/Norwegian Computing Center Oslo.

Guttorp, Peter, and Tilmann Gneiting. 2006. "Studies in the History of Probability and Statistics Xlix on the Matern Correlation Family." *Biometrika* 93 (4). Oxford University Press: 989–95.

Wang, Feifei, Jian Wang, Alan E Gelfand, and Fan Li. 2019. "Disease Mapping with Generative Models." *The American Statistician* 73 (3). Taylor & Francis: 213–23.