TMA4250-Project-3

Ole Riddervold, Ole Kristian Skogly

2023-04-13

Geography data

The data is loaded into the R environment as **nigeriaAdm1** and **nigeriaAdm2** respectively.

```
load("data/Admin1Geography.RData")
load("data/Admin2Geography.RData")
```

a)

```
# Load admin1 and admin2 neighborhood matrices

Admin1Graph <- read.table("data/Admin1Graph.txt", sep="")
Admin2Graph <- read.table("data/Admin2Graph.txt", sep="")

#Precision matrix
precisionMatrix <- function(M){

  # Compute number of neighbors for each node
  n_Neighbour = colSums(M)

  # Construct the precision matrix up to scaling
  R = -M
  diag(R) = n_Neighbour
  return(R)
}

#Define structure matrix R1
R1 = precisionMatrix(Admin1Graph)

#Define structure matrix R2
R2 = precisionMatrix(Admin2Graph)

#Calculate dimension and rank for R1
dim(R1)
```

```
## [1] 37 37
```

```r
qr(R1)$rank
```

```
## [1] 36
```

```r
#Calculate dimension and rank for R2
dim(R2)
```

```
## [1] 775 775
```

```r
qr(R2)$rank
```

```
## [1] 774
```

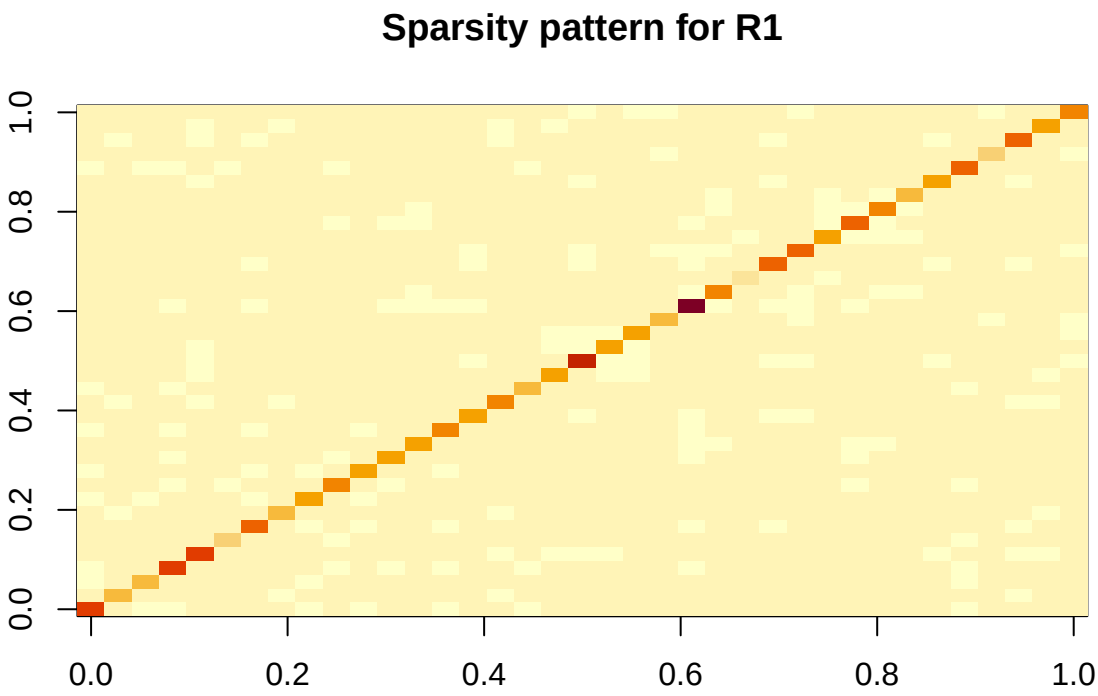
```r
# Compute proportion of non-zero elements
prop_nonzero_R1 <- sum(R1 != 0)/length(R1)^2
cat("The proportion of non-zero elements for R1:", prop_nonzero_R1)
```

```
## The proportion of non-zero elements for R1: 0.1526662
```

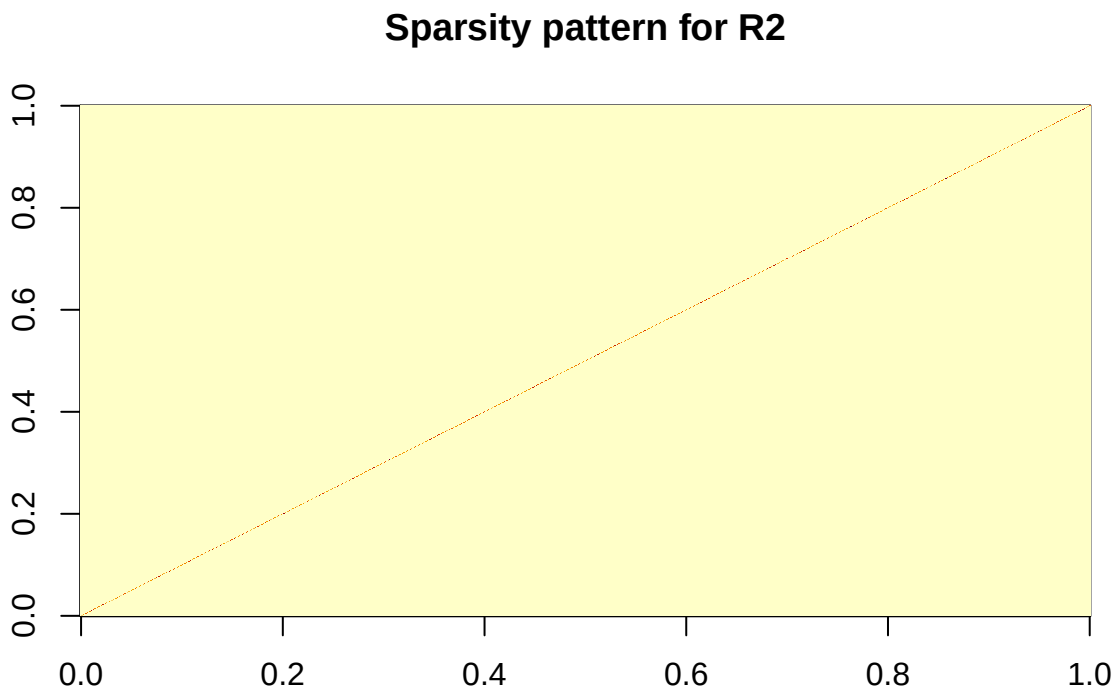
```r
prop_nonzero_R2 <- sum(R2 != 0)/length(R2)^2
cat("The proportion of non-zero elements for R2:", prop_nonzero_R2)
```

```
## The proportion of non-zero elements for R2: 0.008792508
```

```r
# Display sparsity pattern for R1
R1_dense <- as.matrix(R1)
image(R1_dense, main = "Sparsity pattern for R1")
```

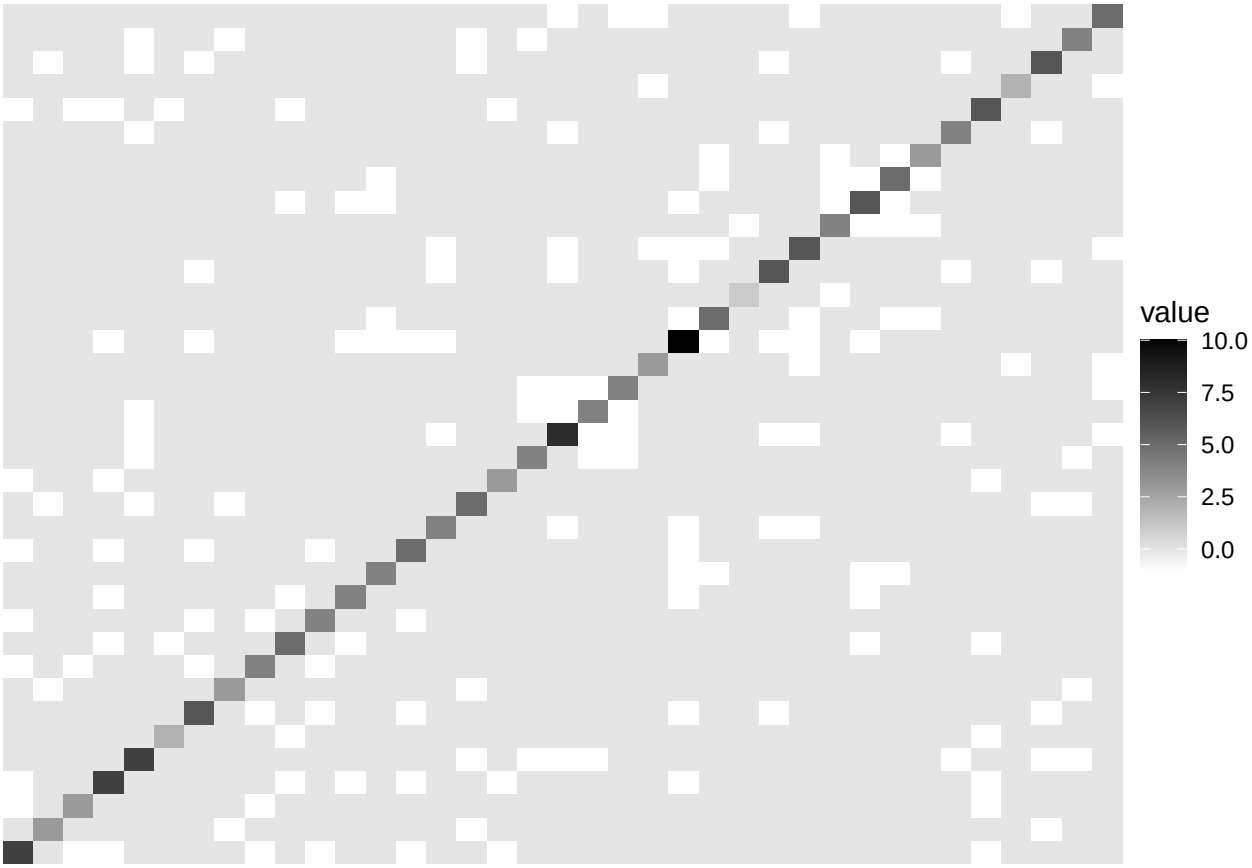



```
# Display sparsity pattern for R2
R2_dense <- as.matrix(R2)
image(R2_dense, main = "Sparsity pattern for R2")
```



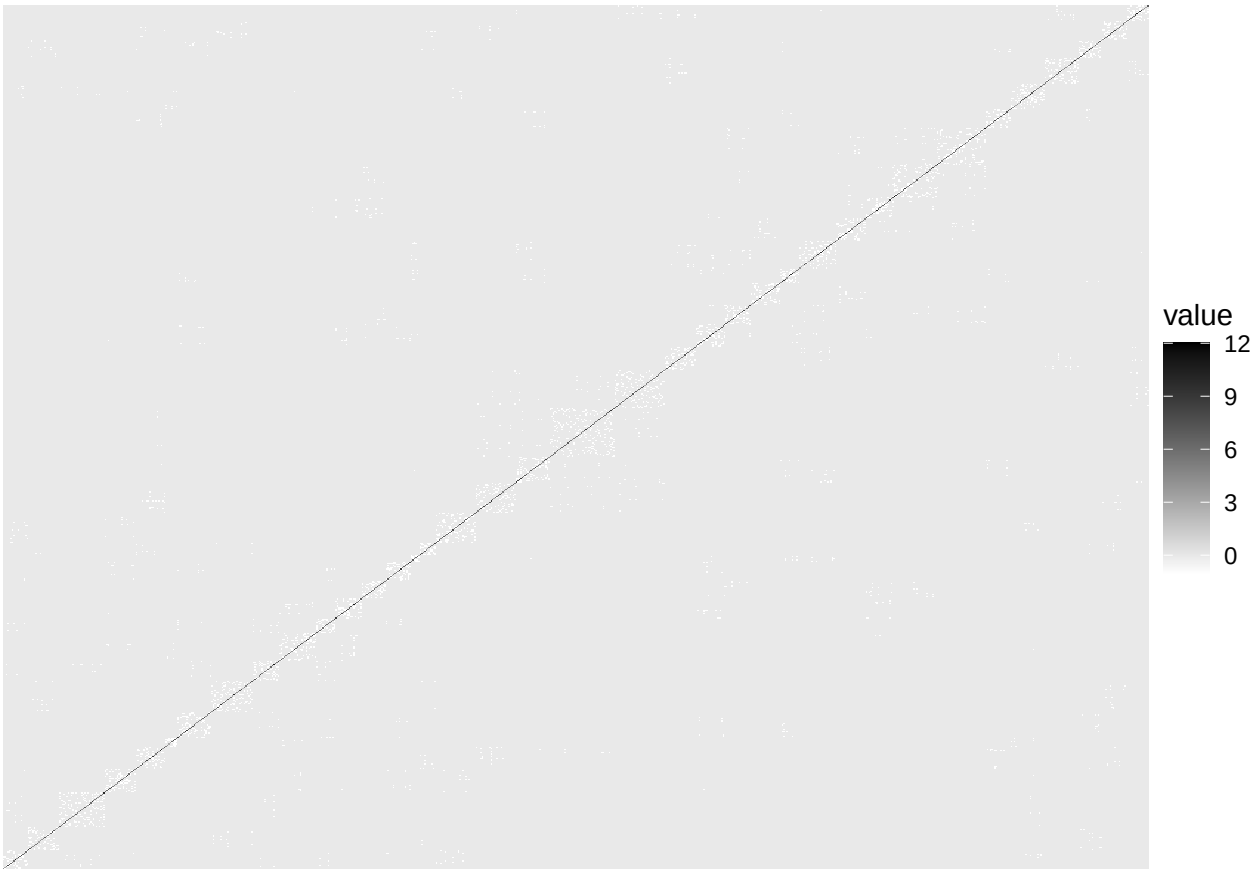

```
# Convert R1 to long format
R1_df <- melt(as.matrix(R1))

# Plot the sparsity pattern using ggplot2
ggplot(R1_df, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black") +
  theme_void()
```




```
# Convert R2 to long format
R2_df <- melt(as.matrix(R2))

# Plot the sparsity pattern using ggplot2
ggplot(R2_df, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black") +
  theme_void()
```



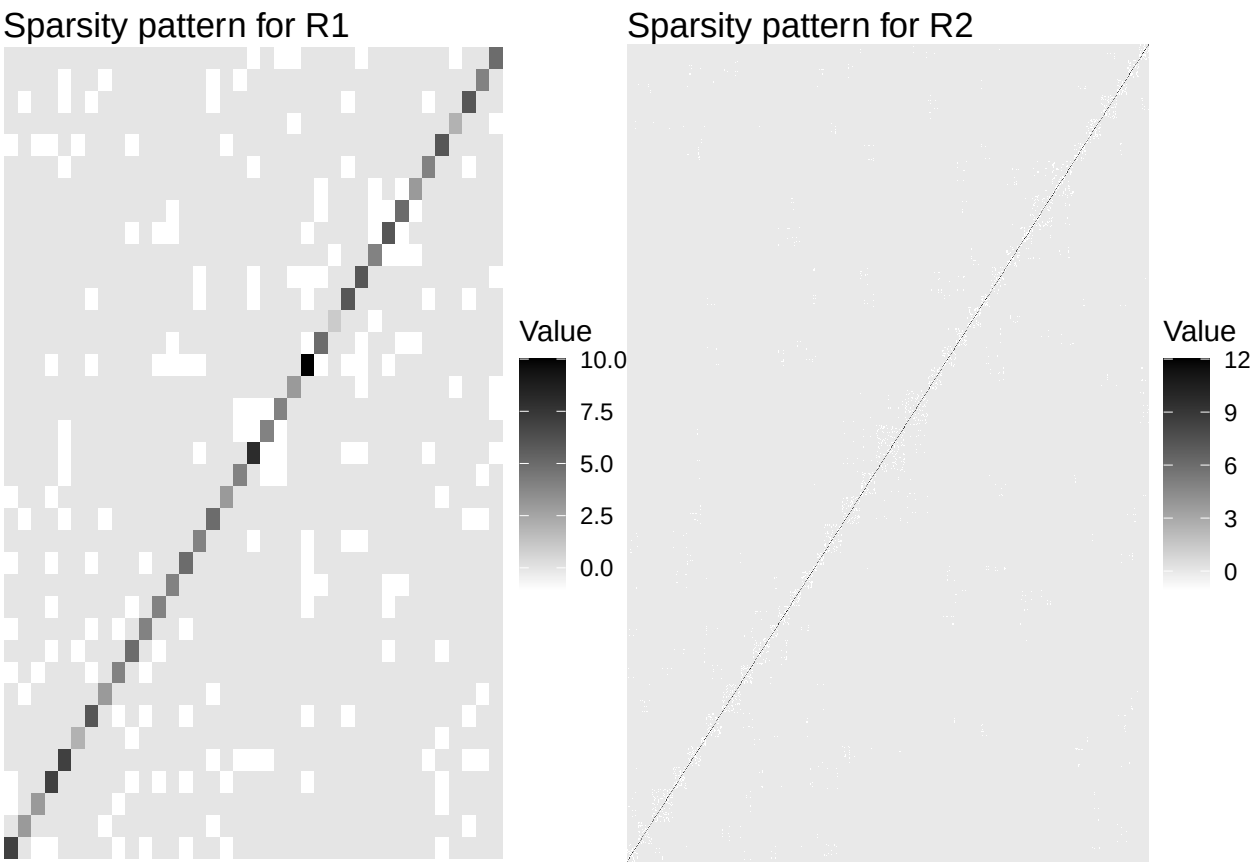

```
##CODE FOR GETTING THE SPARSE MATRICES IN ONE PICTURE
library(ggplot2)
library(reshape2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Convert R1 to long format
R1_df <- melt(as.matrix(R1))
# Plot the sparsity pattern for R1 using ggplot2
p1 <- ggplot(R1_df, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black", name = "Value") + # added name argument
  theme_void() +
  labs(title = "Sparsity pattern for R1")
# Convert R2 to long format
R2_df <- melt(as.matrix(R2))
# Plot the sparsity pattern for R2 using ggplot2
p2 <- ggplot(R2_df, aes(x = Var2, y = Var1, fill = value)) +
  geom_tile() +
```

```
  scale_fill_gradient(low = "white", high = "black", name = "Value") + # added name argument
  theme_void() +
  labs(title = "Sparsity pattern for R2")
# Arrange the two ggplots in one picture
grid.arrange(p1, p2, ncol = 2)
```



b)

```
# Define a function for calculating first-order GMRF
# Q is the precision matrix
# epsilon is a small value added to the diagonal of Q for numerical stability
firstOrder_GMRF <- function(Q, epsilon = 1e-10) {
  n <- length(Q)

  # Add epsilon to the diagonal of Q for numerical stability
  Q_thilde <- Q + epsilon * diag(n)

  # Compute the Cholesky decomposition of Q_thilde
  L <- chol(Q_thilde)

  # Generate a vector of n random normal variables
  z <- rnorm(n)
```

```
  # Solve the linear system L %*% v = z to obtain v
  v <- solve(L) %*% z

  # Compute x by centering v around its mean
  x <- v - mean(v) * rep(1, n)

  return(x)
}

set.seed(4250)
# simulate data from the first-order GMRF using Admin1
Admin1_Besagrealization1 <- firstOrder_GMRF(R1)
Admin1_rnormrealization1 <- rnorm(nrow(R1))
Admin1_Besagrealization2 <- firstOrder_GMRF(R1)
Admin1_rnormrealization2 <- rnorm(nrow(R1))

#The realizations on the Nigeria map of the Besag model on the admin1 and the normalization distributio
plotAreaCol(fName="figures/1b_Besag1.jpg",width = 20, height = 15, estVal= Admin1_Besagrealization1,geol
```

```
## Regions defined for each Polygons
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
```

```
plotAreaCol(fName="figures/1b_rnormreal1.jpg",width = 20, height = 15, estVal= Admin1_rnormrealization1
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(fName="figures/1b_Besag2.jpg",width = 20, height = 15, estVal= Admin1_Besagrealization2,geol
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(fName="figures/1b_normreal2.jpg",width = 20, height = 15, estVal= Admin1_rnormrealization2,g
```

```
## Regions defined for each Polygons
```

c)

```
set.seed(4250)

# simulate data from the first-order GMRF using Admin2
Admin2_Besagrealization1 <- firstOrder_GMRF(R2)
Admin2_rnormrealization1 <- rnorm(nrow(R2))
Admin2_Besagrealization2 <- firstOrder_GMRF(R2)
Admin2_rnormrealization2 <- rnorm(nrow(R2))


#The realizations on the Nigeria map of the Besag model on the admin1 and the normalization distributio
plotAreaCol(fName="figures/1c_Besag1.jpg",width = 20, height = 15, estVal= Admin2_Besagrealization1,geol
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(fName="figures/1c_rnormreal1.jpg",width = 20, height = 15, estVal= Admin2_rnormrealization1
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(fName="figures/1c_Besag2.jpg",width = 20, height = 15, estVal= Admin2_Besagrealization2,geo
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(fName="figures/1c_rnormreal2.jpg",width = 20, height = 15, estVal= Admin2_rnormrealization2
```

```
## Regions defined for each Polygons
```

d)

```
set.seed(4250)
# Set the number of realizations to generate
n_realizations <- 100

# Create an empty matrix to store the simulated data
simulate100_Admin2 <- matrix(NA_real_, nrow = nrow(R2), ncol = n_realizations)

# Loop over the number of realizations and simulate data
for (i in 1:n_realizations) {
  simulate100_Admin2[, i] <- firstOrder_GMRF(R2)
}

# Compute the empirical marginal variance for each admin2 area
margVAR_100 <- apply(simulate100_Admin2, 1, var)
#margVAR_100

Varmax <- ceiling(max(margVAR_100 ))
#Varmax


#Display the var on the map
plotAreaCol(fName="figures/1d_margvar.jpg",width = 20, height = 15, estVal= margVAR_100,geoMap=nigeriaA
```

```
## Regions defined for each Polygons
```

```
correlations <- rep(NA_real_, nrow(simulate100_Admin2))
for (i in 1:775){
  correlations[i] <- cor(simulate100_Admin2[150,], simulate100_Admin2[i,])

}

#Display the cov on the map
plotAreaCol(fName="figures/1d_cov.jpg",width = 20, height = 15, estVal= correlations,geoMap=nigeriaAdm2
```

```
## Regions defined for each Polygons
```

```
#Display the cor on the map
plotAreaCol(fName="figures/1d_cor.jpg",width = 20, height = 15, estVal= correlations,geoMap=nigeriaAdm2
```

```
## Regions defined for each Polygons
```

Problem 2

a)

```
direct_estimates <- read.table(
  "data/DirectEstimates.txt",
  skip=1,
  col.names=c("area_name", "obs", "stdev")
)


plotAreaCol(
  "figures/2a.jpg",
  width=20,
  height=15,
  estVal=invlogit(direct_estimates$obs),
  geoMap=nigeriaAdm1,
  leg=TeX("$\\hat{p}$")
)
```

```
## Regions defined for each Polygons
```

b)

Calculating the parameters for posterior distribution

```
# Treating data:
y <- direct_estimates$obs
D <- diag(direct_estimates$stdev)
D.inv <- solve(D)
n <- length(y)
sigma_squared <- 100^2

# Posterior distribution parameters:
simple.Sigma <- solve( D.inv + (1/sigma_squared)*identity(n) )
simple.mu    <- simple.Sigma %*% D.inv %*% y
```

Empirical estimation of median and standard deviation

```
realization_summary <- function(realizations) {
  # Given a (kxn) array of realizations for many observations,
  # calculates the meadian and standard deviation for each realization.
  # Args:
  #   realizations ((kxn) array): Array of k realizations of n variables
  # Returns:
  #   Dataframe with n rows containing median and std.
  return(
    realizations %>%
      melt(varnames=c("sample", "a")) %>%
      group_by(a) %>%
      summarise(median.est=median(value), std.est=std(value)/mean(value))
  )
}
```

```
simple.samples <- sigmoid(mvrnorm(simple.mu, simple.Sigma, n=100))
simple.sum <- realization_summary(simple.samples)
```

Plotting:

```
plotAreaCol(
  "figures/2b_median.jpg",
  width=20,
  height=15,
  estVal=simple.sum$median.est,
  geoMap=nigeriaAdm1,
  leg="median est."
)
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(
  "figures/2b_std.jpg",
  width=20,
  height=15,
  estVal=simple.sum$std.est,
  geoMap=nigeriaAdm1,
  leg=TeX("$\\hat{CV}$")
)
```

```
## Regions defined for each Polygons
```

c)

Helper function to generate structure matrix:

```
generate_structure_matrix <- function(G) {
  # Generates the structure matrix of a Besag model based
  # on the graph G.
```

```
  # Args:
  #   G ((nxn) matrix): Matrix representation of G
  # Returns:
  #   (nxn) structure matrix
  R <- -G
  diag(R) <- colSums(t(G))
  return(R)
}
```

Reading the graph and generating the structure matrix:

```
G <- read.table("data/Admin1Graph.txt") %>% as.matrix()
R <- generate_structure_matrix(G)
```

Execution:

```
evaluate_besag_and_plot <- function(tau, filenames) {
  # Evaluates the besag posterior, samples and summarises, and
  # finally plots the data. Relevant for task c and e.
  # Args:
  #   tau               (float): Besag hyperparameter
  #   filenames (length-2 list): Filenames of plots
  # Returns:
  #   Model summary
  # Posterior distribution parameters:
  besag.Sigma <- solve(tau*R + D.inv)
  besag.mu <- besag.Sigma %*% D.inv %*% y

  # Sampling from the distribution
  besag.samples <- sigmoid(mvrnorm(besag.mu, besag.Sigma, n=100))
  besag.sum <- realization_summary(besag.samples)

  plotAreaCol(
    filenames[1],
    width=20,
    height=15,
    estVal=besag.sum$median.est,
    geoMap=nigeriaAdm1,
    leg="median est."
  )

  plotAreaCol(
    filenames[2],
    width=20,
    height=15,
    estVal=besag.sum$std.est,
    geoMap=nigeriaAdm1,
    leg=TeX("$\\hat{CV}$")
  )

  return(besag.sum)
}
```

```r
besag_orig.sum <- evaluate_besag_and_plot(
  tau=1.0,
  filenames=c("figures/2c_median.jpg", "figures/2c_std.jpg")
)
```

```
## Regions defined for each Polygons
## Regions defined for each Polygons
```

d)

```r
evaluate_updated_besag_and_plot <- function(tau, update, filenames) {
  # Evaluates the updated besag posterior with Kaduna as an additional sample,
  #and summarises, and finally plots the data. Relevant for task c and e.
  # Args:
  #   tau               (float): Besag hyperparameter
  #   update             (list): Observation and variance of update
  #   filenames (length-2 list): Filenames of plots
  # Returns:
  #   Model summary
  # Posterior distribution parameters:
  A <- diag(1, nrow=38, ncol=37)
  A[38, 19] <- 1
  D.tilde.inv <- diag(c(direct_estimates$stdev, update$var)) %>% solve()
  besag.Sigma <- solve(tau*R + t(A)%*%D.tilde.inv%*%A)
  besag.mu <- besag.Sigma %*% t(A) %*% D.tilde.inv %*% c(y, update$obs)

  # Sampling from the distribution
  besag.samples <- sigmoid(mvrnorm(besag.mu, besag.Sigma, n=100))
  besag.sum <- realization_summary(besag.samples)

  plotAreaCol(
    filenames[1],
    width=20,
    height=15,
    estVal=besag.sum$median.est,
    geoMap=nigeriaAdm1,
    leg="median est."
  )

  plotAreaCol(
    filenames[2],
    width=20,
    height=15,
    estVal=besag.sum$std.est,
    geoMap=nigeriaAdm1,
    leg=TeX("$\\hat{CV}$")
  )

  return(besag.sum)
}
```

```
besag_updated.sum <- evaluate_updated_besag_and_plot(
  tau=1.0,
  update=list(obs=0.5, var=0.1^2),
  filenames=c("figures/2d_median.jpg", "figures/2d_std.jpg")
)
```

```
## Regions defined for each Polygons
## Regions defined for each Polygons
```

```
plotAreaCol(
  "figures/2d_median_diff.jpg",
  width=20,
  height=15,
  estVal=besag_updated.sum$median.est - besag_orig.sum$median.est,
  geoMap=nigeriaAdm1,
  leg=TeX("median.diff")
)
```

```
## Regions defined for each Polygons
```

```
plotAreaCol(
  "figures/2d_std_diff.jpg",
  width=20,
  height=15,
  estVal=besag_updated.sum$std.est - besag_orig.sum$std.est,
  geoMap=nigeriaAdm1,
  leg=TeX("CV.diff")
)
```

```
## Regions defined for each Polygons
```

e)

$\tau = 0.1$

```
evaluate_besag_and_plot(
  tau=0.1,
  filenames=c("figures/2e_0,1_median.jpg", "figures/2e_0,1_std.jpg")
)
```

```
## Regions defined for each Polygons
## Regions defined for each Polygons
```

```
## # A tibble: 37 x 3
##    a           median.est std.est
##    <fct>            <dbl>   <dbl>
##  1 Abia             0.791  0.108
##  2 Adamawa          0.647  0.147
##  3 Akwa.Ibom        0.641  0.158
##  4 Anambra          0.803  0.0977
```

```
##  5 Bauchi           0.381  0.242
##  6 Bayelsa          0.722  0.125
##  7 Benue            0.653  0.152
##  8 Borno            0.461  0.229
##  9 Cross.River      0.649  0.200
## 10 Delta            0.741  0.108
## # ... with 27 more rows
```

$\tau = 10$

```
evaluate_besag_and_plot(
  tau=10.0,
  filenames=c("figures/2e_10_median.jpg", "figures/2e_10_std.jpg")
)
```

```
## Regions defined for each Polygons
## Regions defined for each Polygons
```

```
## # A tibble: 37 x 3
##    a           median.est std.est
##    <fct>            <dbl>   <dbl>
##  1 Abia             0.665  0.0540
##  2 Adamawa          0.524  0.0880
##  3 Akwa.Ibom        0.657  0.0768
##  4 Anambra          0.670  0.0553
##  5 Bauchi           0.489  0.0806
##  6 Bayelsa          0.674  0.0806
##  7 Benue            0.603  0.0648
##  8 Borno            0.485  0.108
##  9 Cross.River      0.634  0.0699
## 10 Delta            0.669  0.0557
## # ... with 27 more rows
```

f)

```
Q_C <- function(tau) {
  # Precision matrix of a Besag posterior model.
  # Args:
  #   tau (float)
  # Returns:
  #   Q_C ((nxn) matrix)
  return(
    tau*R + D.inv
  )
}

mu_C <- function(Q_C) {
  # Mean of a Besag posterior model.
  # Args:
  #   Q_C ((nxn) matrix): Precision matrix of a besag posterior model
  # Returns:
```

```
  #   mu_C (length-n list)
  return(
    solve(Q_C) %*% D.inv %*% y
  )
}

log_likelihood <- function(tau) {
  # Log likelihood function of tau given the data y
  # The constant is not included, as this is not important to perform an MLE.
  # Args:
  #   tau (float)
  # Returns:
  #   l(tau; y)
  Q_C <- Q_C(tau)
  mu_C <- mu_C(Q_C)
  return(
    (36/2)*log(tau) - (tau/2)*t(mu_C) %*% R %*% mu_C - (1/2)*t(y-mu_C) %*% D.inv %*% (y-mu_C) -
      (1/2)*log(det(Q_C))
  )
}
```

```
tau_hat <- optimize(log_likelihood, interval=c(0, 10000), maximum=TRUE)$maximum
```

```
tau_hat
```

```
## [1] 1.472776
```

```
evaluate_besag_and_plot(
  tau=tau_hat,
  filenames=c("figures/2f_median.jpg", "figures/2f_std.jpg")
)
```

```
## Regions defined for each Polygons
## Regions defined for each Polygons
```

```
## # A tibble: 37 x 3
##    a           median.est std.est
##    <fct>            <dbl>   <dbl>
##  1 Abia             0.713  0.0737
##  2 Adamawa          0.562  0.130
##  3 Akwa.Ibom        0.684  0.0998
##  4 Anambra          0.724  0.0871
##  5 Bauchi           0.445  0.144
##  6 Bayelsa          0.709  0.119
##  7 Benue            0.631  0.102
##  8 Borno            0.460  0.158
##  9 Cross.River      0.664  0.110
## 10 Delta            0.710  0.0894
## # ... with 27 more rows
```