Homework 3

Peter Tran

1. Obtain data on 2021 STD cases in Georgia using the following steps.

   a-h. **ALREADY DONE**

      i. Download the Georgia shape file using the tigris package in R.

```
library(rgdal)
library(tigris)
```

```
ga <- counties("Georgia")
```

```
## Retrieving data for the year 2020
```

```
##   |                                                                              |
```

j. Load all three data frames in R (the shape file from tigris, the STD file, and the population file).

```
library(readxl)
library(tidyverse)
```

```
std <- read_excel("../data/ga_std.xlsx", skip = 2, n_max = 159) %>%
  mutate(`...1` = toupper(`...1`)) %>%
  mutate(`STD Cases` = replace(`STD Cases`, `...1` == "GLASCOCK", 20)) %>%
  mutate(`STD Cases` = as.numeric(gsub(",", "", `STD Cases`)))
```

```
## New names:
## * `` -> `...1`
```

```
pop <- read_csv("../data/georgia_population_22.csv")
```

```
## Rows: 159 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): County
## num (1): Pop
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data <- ga %>%
  mutate(NAME = toupper(NAME)) %>%
  inner_join(std, by = c("NAME" = "...1")) %>%
  inner_join(pop, by = c("NAME" = "County"))
```

k. Use the head function in R to print out the first few rows of your combined file.

```
head(data)
```

```
## Simple feature collection with 6 features and 19 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
```

```
## Bounding box:  xmin: -85.46221 ymin: 31.01044 xmax: -81.73169 ymax: 34.58748
## Geodetic CRS:  NAD83
##   STATEFP COUNTYFP COUNTYNS GEOID     NAME        NAMELSAD LSAD CLASSFP MTFCC
## 1      13      189 00348794 13189 MCDUFFIE McDuffie County   06      H1 G4020
## 2      13      025 00351605 13025 BRANTLEY Brantley County   06      H1 G4020
## 3      13      171 00326713 13171    LAMAR    Lamar County   06      H1 G4020
## 4      13      115 00353665 13115    FLOYD    Floyd County   06      H1 G4020
## 5      13      273 00352238 13273  TERRELL  Terrell County   06      H1 G4020
## 6      13      063 01672399 13063  CLAYTON  Clayton County   06      H1 G4020
##   CSAFP CBSAFP METDIVFP FUNCSTAT      ALAND   AWATER    INTPTLAT     INTPTLON
## 1  <NA>  12260     <NA>        A  666590014 23114032 +33.4806126 -082.4795333
## 2  <NA>  15260     <NA>        A 1147972258 10291563 +31.1973339 -081.9829779
## 3   122  12060     <NA>        A  475264404  6044329 +33.0744405 -084.1466893
## 4   122  40660     <NA>        A 1320404595 22414013 +34.2636918 -085.2136851
## 5  <NA>  10500     <NA>        A  869695791  4951325 +31.7771909 -084.4394464
## 6   122  12060     <NA>        A  366879097  6962586 +33.5426863 -084.3555727
##   STD Cases    Pop                       geometry
## 1       211  21162 MULTIPOLYGON (((-82.44998 3...
## 2        64  19202 MULTIPOLYGON (((-81.91012 3...
## 3       103  19261 MULTIPOLYGON (((-84.24837 3...
## 4       608  98604 MULTIPOLYGON (((-85.24134 3...
## 5       164   8523 MULTIPOLYGON (((-84.56317 3...
## 6      3927 292646 MULTIPOLYGON (((-84.45856 3...
```

2. Compute the raw STD rate. Compute Moran's I for this variable. Test the hypothesis that there is no spatial autocorrelation. Report and discuss your results.

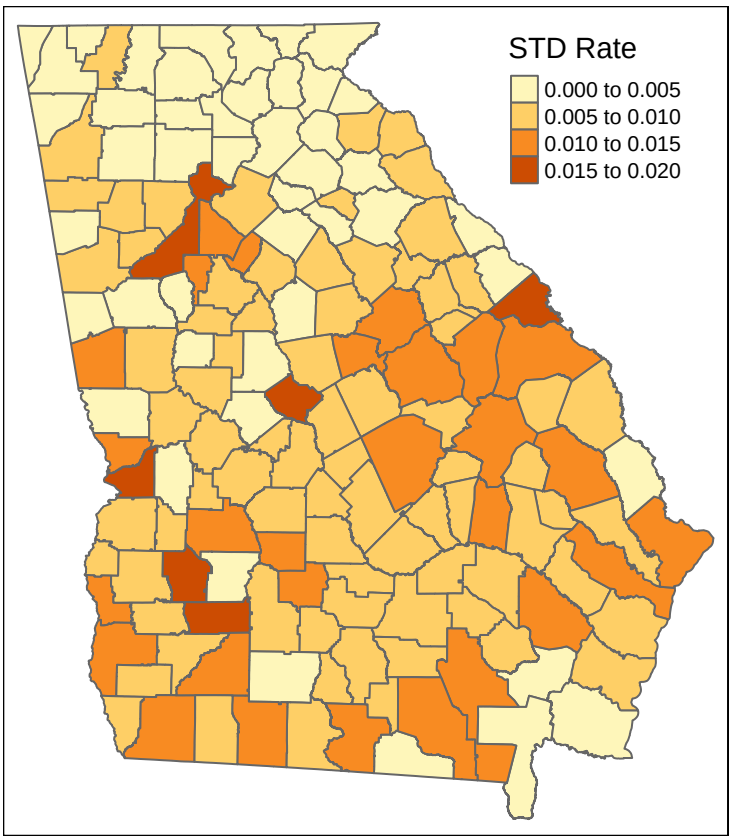
```
library(ape)
library(spdep)
library(tmap)
```

```
data$stdrate <- data$`STD Cases`/data$Pop

Moran.I(data$stdrate, listw2mat(nb2listw(poly2nb(data, queen = F))))
```

```
## $observed
## [1] 0.2773134
##
## $expected
## [1] -0.006329114
##
## $sd
## [1] 0.04938405
##
## $p.value
## [1] 9.268197e-09
```

It's difficult to interpret Moran's I by itself, let's plot a map to accompany it.

```
tm_shape(data) +
  tm_polygons(col = "stdrate", title = "STD Rate")
```



The p-value for Moran's I is significant. This tells us that there is spatial autocorrelation present in our data. Looking at the map above, we can see that STD rates tend to be a bit higher in the southern part of the state compared to the north, with some noticeable clustering of the 3 darkest color shown.

3. Run the following models:

    a. Non-spatial model with just the v_i terms.

```
library(nimble)
```

```
v_i_code <- nimbleCode({
  for (i in 1:N) {
    y[i] ~ dpois(pop[i]*theta[i])
    log(theta[i]) <- beta_0+v[i]
    v[i] ~ dnorm(0, tau_v)
  }
  beta_0 ~ dnorm(-3, 1/10)
  tau_v ~ dgamma(1, 1/10000)
})

v_i_data <- list(y = data$`STD Cases`)

v_i_consts <- list(N = nrow(data), pop = data$Pop)

v_i_inits <- list(beta_0 = 0, v = rep(0, nrow(data)), tau_v = 0.1)

v_i_model <- nimbleModel(v_i_code, data = v_i_data, constants = v_i_consts,
                         inits = v_i_inits)
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.

## Checking model sizes and dimensions
```

```
v_i_compile <- compileNimble(v_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
v_i_conf <- configureMCMC(v_i_compile, print = T, thin = 100, enableWAIC = T)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_v
## ===== Samplers =====
## RW sampler (160)
##   - beta_0
##   - v[]  (159 elements)
## conjugate sampler (1)
##   - tau_v
```

```
v_i_conf$addMonitors(c("beta_0", "tau_v", "theta"))
```

```
## thin = 100: beta_0, tau_v, theta
```

```
v_i_mcmc <- buildMCMC(v_i_conf)
v_i_mcmc_compile <- compileNimble(v_i_mcmc, project = v_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

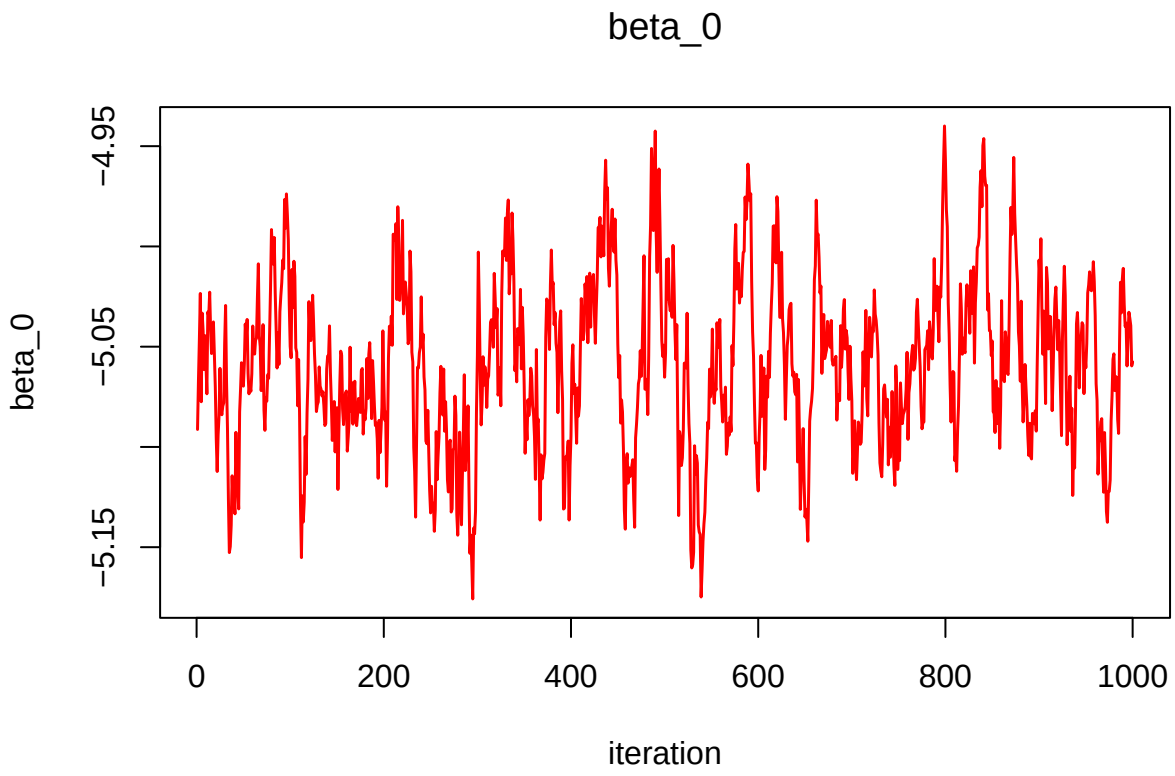
```
v_i_samples <- runMCMC(v_i_mcmc_compile, niter = 120000, nburnin = 20000,
                       inits = v_i_inits, nchains = 1, samplesAsCodaMCMC = T, WAIC = T)
```

```
## Running chain 1 ...
```

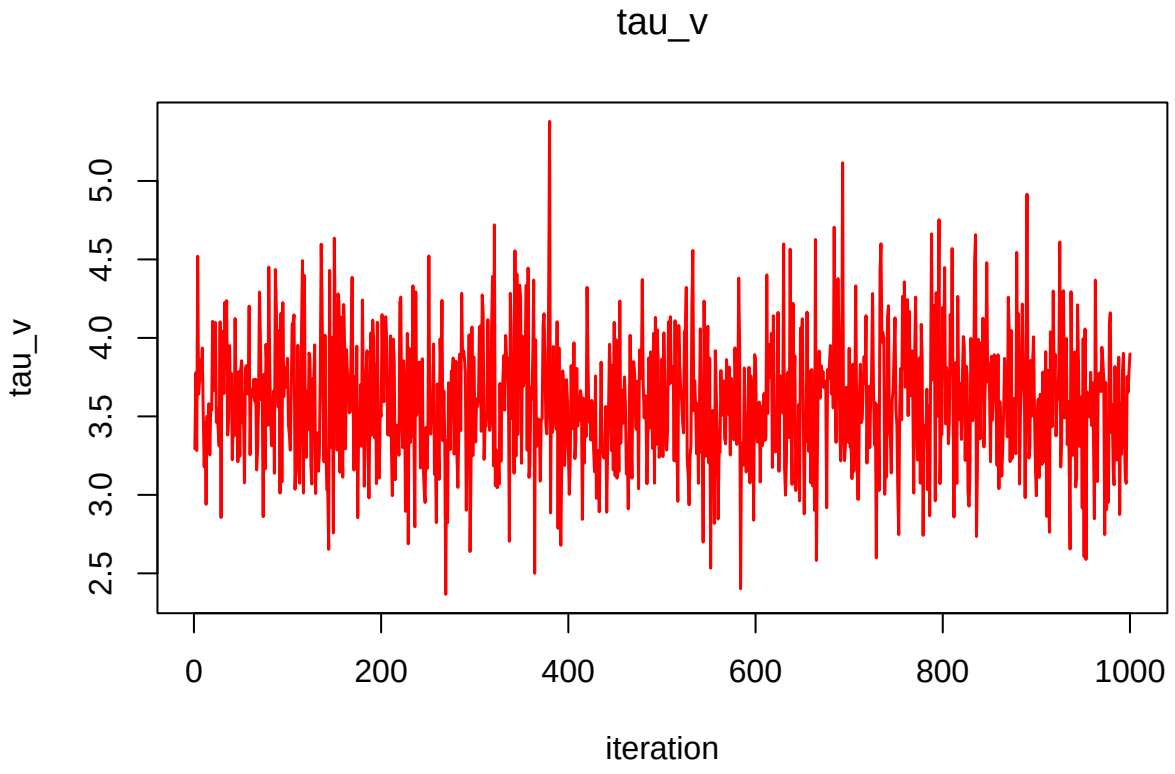
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
##   [Warning] There are individual pWAIC values that are greater than 0.4. This may indicate that the W
```

```
data$v_i_rates <- apply(v_i_samples$samples[,3:ncol(v_i_samples$samples)], 2, mean)
```

```
ts.plot(v_i_samples$samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```




```
ts.plot(v_i_samples$samples[,"tau_v"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_v), main = expression(tau_v))
```



b. Spatial model with just the s_i terms.

```
data_sp <- as(data, "Spatial")
data_nb <- poly2nb(data_sp, queen = F)

num <- map_int(data_nb, length)

adj <- c()
for (i in 1:nrow(data_sp))
  adj <- c(adj, data_nb[[i]])
```

```
s_i_code <- nimbleCode({
  for (i in 1:N) {
    y[i] ~ dpois(pop[i]*theta[i])
    log(theta[i]) <- beta_0+s[i]
  }
  s[1:N] ~ dcar_normal(adj[1:L], weights[1:L], num[1:N], tau_s, zero_mean = 1)
  beta_0 ~ dnorm(-3, 1/10)
  tau_s ~ dgamma(1, 1/10000)
})

s_i_data <- list(y = data$`STD Cases`)

s_i_consts <- list(N = nrow(data), pop = data$Pop, adj = adj, num = num, L = length(adj),
                   weights = rep(1, length(adj)))

s_i_inits <- list(beta_0 = 0, s = rep(0, nrow(data)), tau_s = 0.1)

s_i_model <- nimbleModel(s_i_code, data = s_i_data, constants = s_i_consts,
                         inits = s_i_inits)
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.

## Checking model sizes and dimensions
```

```
s_i_compile <- compileNimble(s_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
s_i_conf <- configureMCMC(s_i_compile, print = T, thin = 100, enableWAIC = T)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_s
## ===== Samplers =====
## RW sampler (2)
##   - beta_0
##   - tau_s
## CAR_normal sampler (1)
##   - s[1:159]
```

```
s_i_conf$addMonitors(c("beta_0", "tau_s", "theta"))
```

```
## thin = 100: beta_0, tau_s, theta
```

```
s_i_mcmc <- buildMCMC(s_i_conf)
s_i_mcmc_compile <- compileNimble(s_i_mcmc, project = s_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

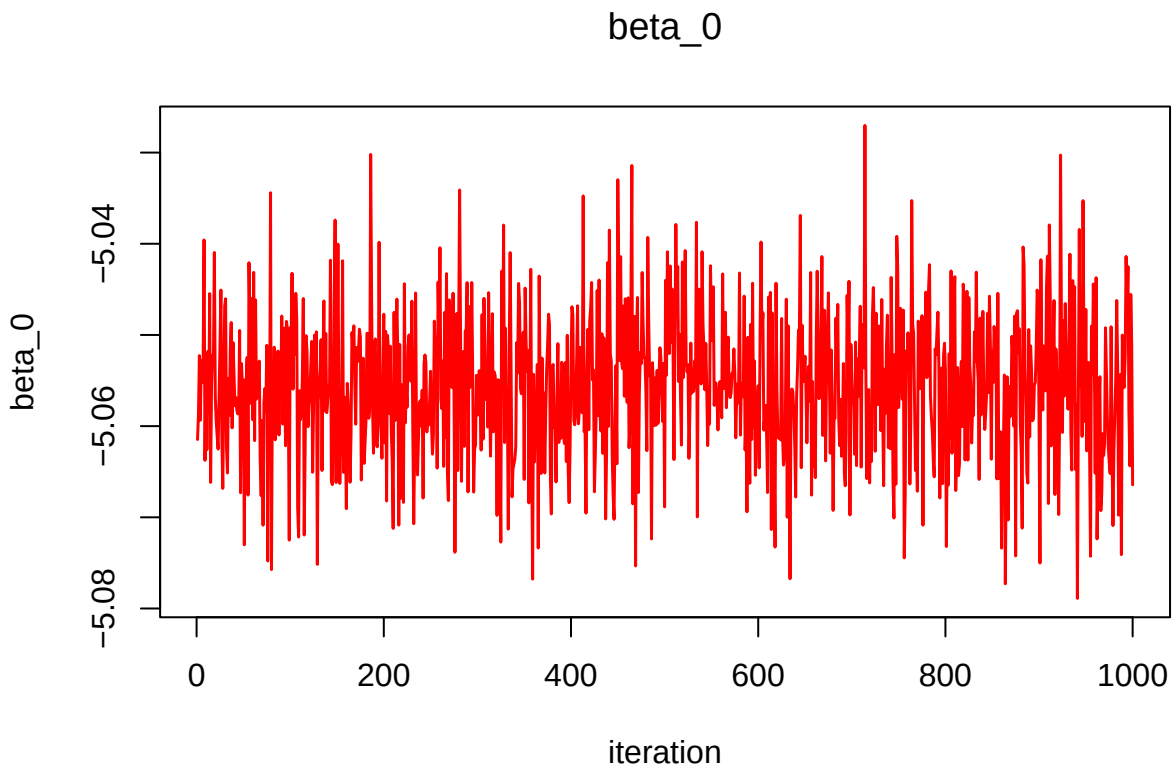
```
s_i_samples <- runMCMC(s_i_mcmc_compile, niter = 120000, nburnin = 20000,
                       inits = s_i_inits, nchains = 1, samplesAsCodaMCMC = T, WAIC = T)
```

```
## Running chain 1 ...
```

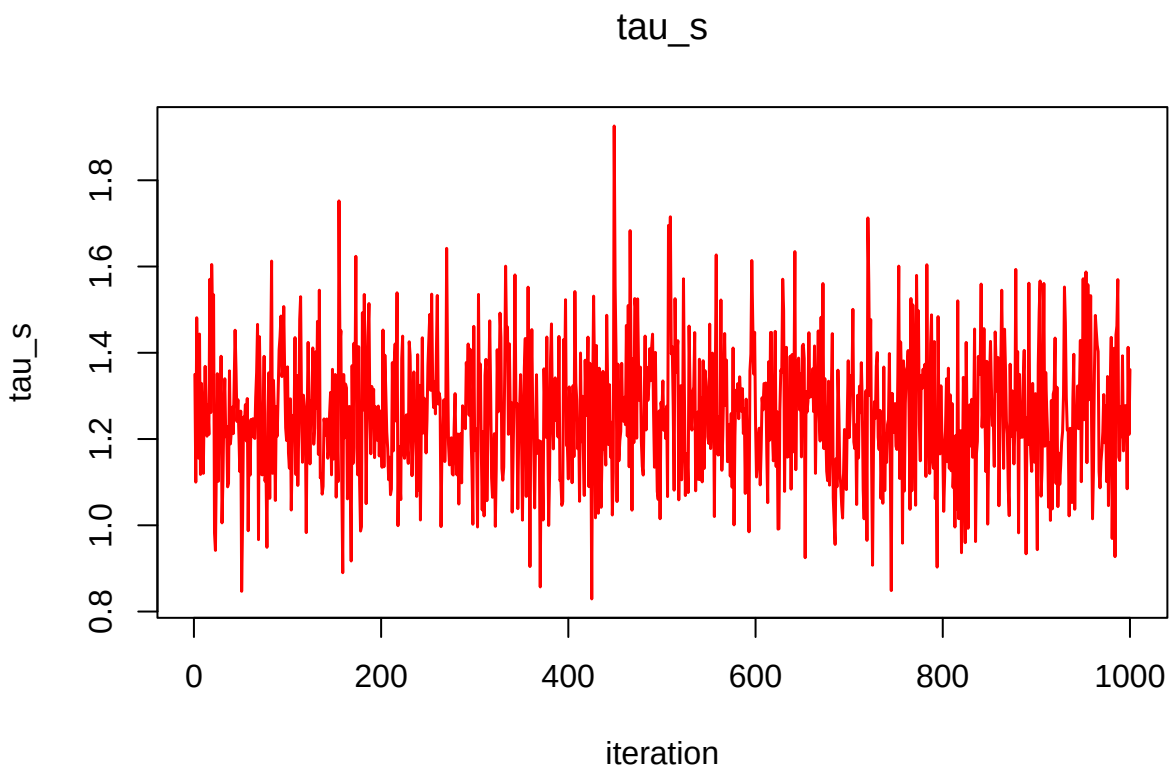
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
##   [Warning] There are individual pWAIC values that are greater than 0.4. This may indicate that the W
```

```
data$s_i_rates <- apply(s_i_samples$samples[,3:ncol(s_i_samples$samples)], 2, mean)
```

```
ts.plot(s_i_samples$samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```




```
ts.plot(s_i_samples$samples[,"tau_s"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_s), main = expression(tau_s))
```



c. Spatial model with both v_i and s_i terms.

```
v_i_s_i_code <- nimbleCode({
  for (i in 1:N) {
    y[i] ~ dpois(pop[i]*theta[i])
    log(theta[i]) <- beta_0+s[i]
    v[i] ~ dnorm(0, tau_v)
  }
  s[1:N] ~ dcar_normal(adj[1:L], weights[1:L], num[1:N], tau_s, zero_mean = 1)
  beta_0 ~ dnorm(-3, 1/10)
  tau_v ~ dgamma(1, 1/10000)
  tau_s ~ dgamma(1, 1/10000)
})

v_i_s_i_data <- list(y = data$`STD Cases`)

v_i_s_i_consts <- list(N = nrow(data), pop = data$Pop, adj = adj, num = num,
                       L = length(adj), weights = rep(1, length(adj)))

v_i_s_i_inits <- list(beta_0 = 0, v = rep(0, nrow(data)), tau_v = 0.1,
                      s = rep(0, nrow(data)), tau_s = 0.1)

v_i_s_i_model <- nimbleModel(v_i_s_i_code, data = v_i_s_i_data,
                             constants = v_i_s_i_consts, inits = v_i_s_i_inits)
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
```

```
##   [Note] Any error reports that follow may simply reflect missing values in model variables.
```

```
## Checking model sizes and dimensions
```

```
v_i_s_i_compile <- compileNimble(v_i_s_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
v_i_s_i_conf <- configureMCMC(v_i_s_i_compile, print = T, thin = 100, enableWAIC = T)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_s, tau_v
## ===== Samplers =====
## RW sampler (2)
##   - beta_0
##   - tau_s
## posterior_predictive_branch sampler (1)
##   - tau_v
## CAR_normal sampler (1)
##   - s[1:159]
```

```
v_i_s_i_conf$addMonitors(c("beta_0", "tau_s", "theta"))
```

```
## thin = 100: beta_0, tau_s, tau_v, theta
```

```
v_i_s_i_mcmc <- buildMCMC(v_i_s_i_conf)
v_i_s_i_mcmc_compile <- compileNimble(v_i_s_i_mcmc, project = v_i_s_i_model)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

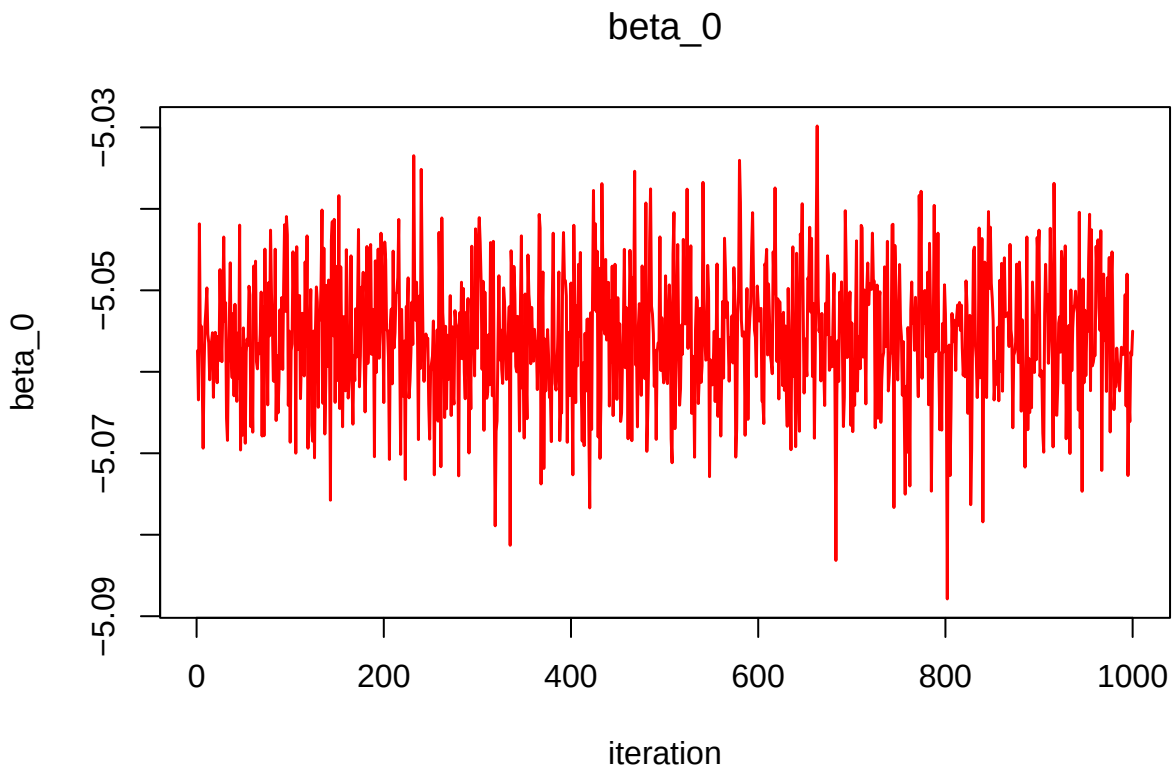
```
v_i_s_i_samples <- runMCMC(v_i_s_i_mcmc_compile, niter = 120000, nburnin = 20000,
                           inits = v_i_s_i_inits, nchains = 1, samplesAsCodaMCMC = T,
                           WAIC = T)
```

```
## Running chain 1 ...
```

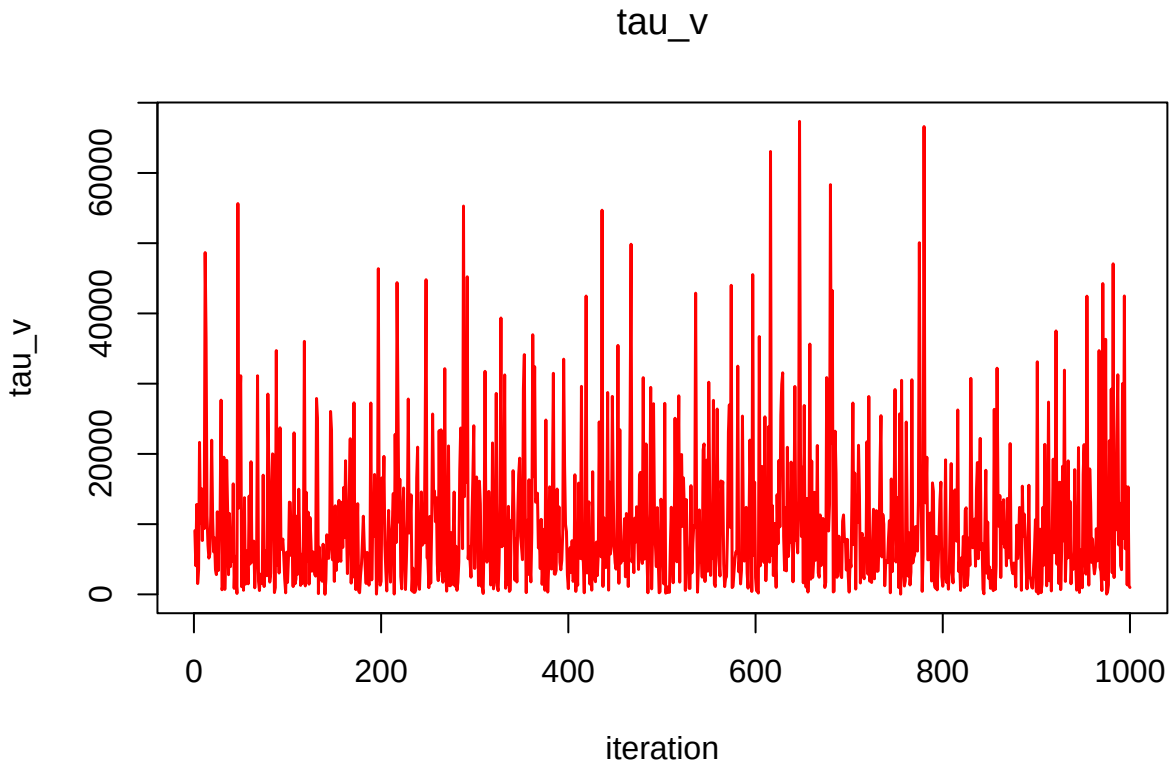
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
##   [Warning] There are individual pWAIC values that are greater than 0.4. This may indicate that the W
```

```
data$v_i_s_i_rates <- apply(v_i_s_i_samples$samples[,4:ncol(v_i_s_i_samples$samples)],
                            2, mean)
```

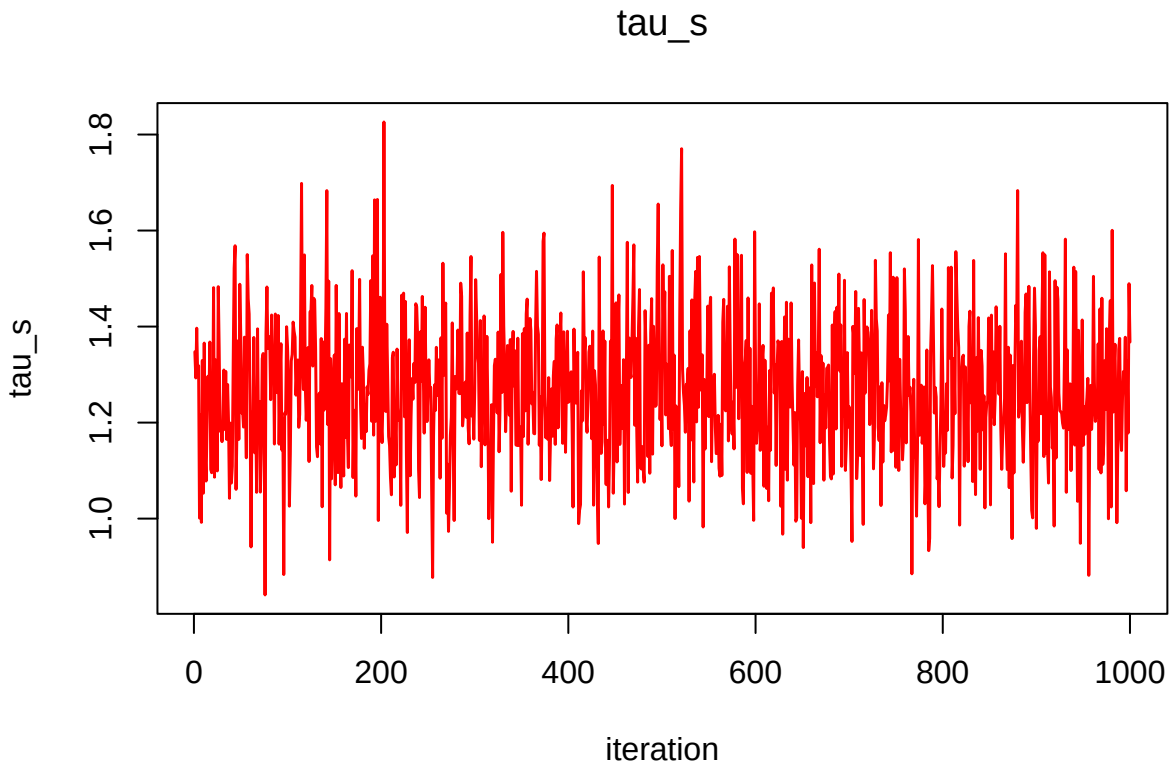
```
ts.plot(v_i_s_i_samples$samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```




```
ts.plot(v_i_s_i_samples$samples[,"tau_v"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_v), main = expression(tau_v))
```



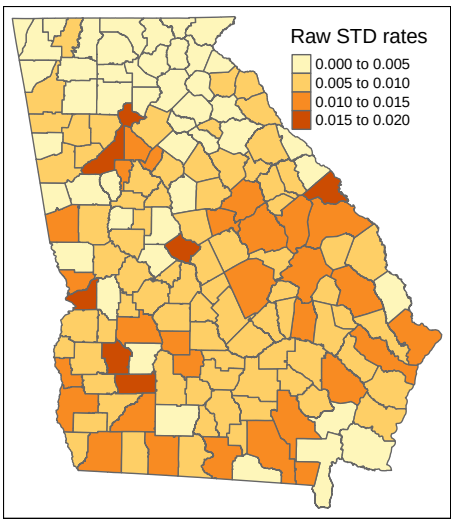

```
ts.plot(v_i_s_i_samples$samples[,"tau_s"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_s), main = expression(tau_s))
```

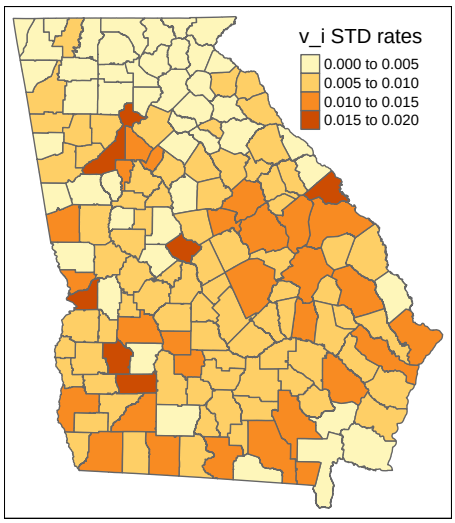


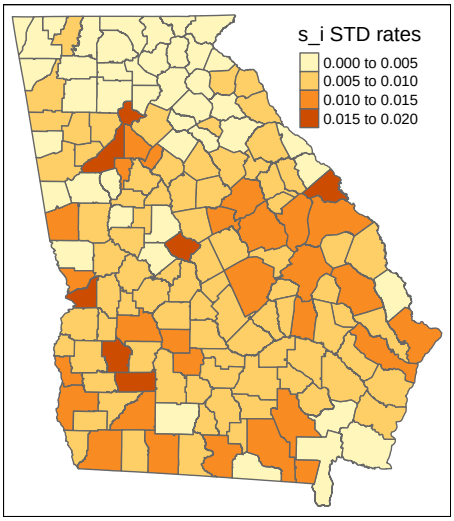
4. Create four choropleth maps, one for the raw rates and one for each of the above models.

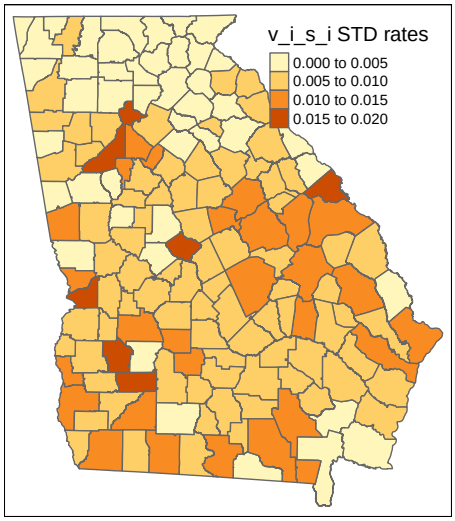
```
library(grid)
```

```
raw_plot <- tm_shape(data) +
  tm_polygons(col = "stdrate", title = "Raw STD rates")

v_i_plot <- tm_shape(data) +
  tm_polygons(col = "v_i_rates", breaks = seq(0, 0.020, 0.005), title = "v_i STD rates")

s_i_plot <- tm_shape(data) +
  tm_polygons(col = "s_i_rates", breaks = seq(0, 0.020, 0.005), title = "s_i STD rates")

v_i_s_i_plot <- tm_shape(data) +
  tm_polygons(col = "v_i_s_i_rates", breaks = seq(0, 0.020, 0.005),
              title = "v_i_s_i STD rates")

grid.newpage()
pushViewport(viewport(layout = grid.layout(2, 2)))
print(raw_plot, vp = viewport(layout.pos.row = 1, layout.pos.col = 1, height = 5))
print(v_i_plot, vp = viewport(layout.pos.row = 1, layout.pos.col = 2, height = 5))
print(s_i_plot, vp = viewport(layout.pos.row = 2, layout.pos.col = 1, height = 5))
print(v_i_s_i_plot, vp = viewport(layout.pos.row = 2, layout.pos.col = 2, height = 5))
```









5. Compute the AIC for each of the models from problem 3. Report and discuss your results.

```
v_i_samples$WAIC
```

```
## nimbleList object of type waicList
## Field "WAIC":
## [1] 1381.124
## Field "lppd":
## [1] -610.8833
## Field "pWAIC":
## [1] 79.67872
```

```
s_i_samples$WAIC
```

```
## nimbleList object of type waicList
## Field "WAIC":
## [1] 1414.026
## Field "lppd":
## [1] -612.3092
## Field "pWAIC":
## [1] 94.70369
```

```
v_i_s_i_samples$WAIC
```

```
## nimbleList object of type waicList
## Field "WAIC":
## [1] 1411.338
## Field "lppd":
## [1] -612.2927
## Field "pWAIC":
## [1] 93.37638
```

We can see that the model with the lowest WAIC is the non-spatial model with just the v_i terms. Additionally, it's not that different from the other models, which we can see reflected in the choropleth maps above. In fact, all 3 models aren't really that different from the raw rates either!