Homework 2

Peter Tran

BST 5610

Spatial Epidemiology and Disease Mapping

Homework 2

Due 11:59 PM, November 4, 2022

1. (2 Points) Consider the state of Connecticut, which has only 8 counties. Find the num and adj vectors as defined on Slide 21 of DiseaseMapping.pdf. You can do this by hand because Connecticut has only 8 counties

We will assign numbers to counties by their alphabetical order:

1: Fairfield

2: Hartford

3: Litchfield

4: Middlesex

5: New Haven

6: New London

7: Tolland

8: Windham

```
num = [2, 5, 3, 3, 4, 4, 3, 2]
adj = [3, 5, 3, 4, 5, 6, 7, 1, 2, 5, 2, 5, 6, 1, 2, 3, 4, 2, 5, 7, 8, 2, 6, 8, 6, 7]
```

2. (2 Points) Use the network of counties in eastern Missouri to illustrate the global Markov property of a GMRF. Color the nodes as necessary and give a brief explanation of which colored nodes are independent of which others conditioned on a particular set of nodes. Your explanation should be brief, like that shown in slides 24 and 25 of DiseaseMapping.pdf.

See Figure 1. Conditioned on the red nodes, the nodes in green are independent of the nodes in yellow and vice versa.

3. (6 Points) For the Pennsylvania lung cancer data, run models

Model 2b: Hierarchical model with Poisson (slide 10)

```r
set.seed(11042022)
```

```r
library(SpatialEpi)
library(nimble)
library(sp)
library(spdep)
library(tidyverse)
library(tmap)
```

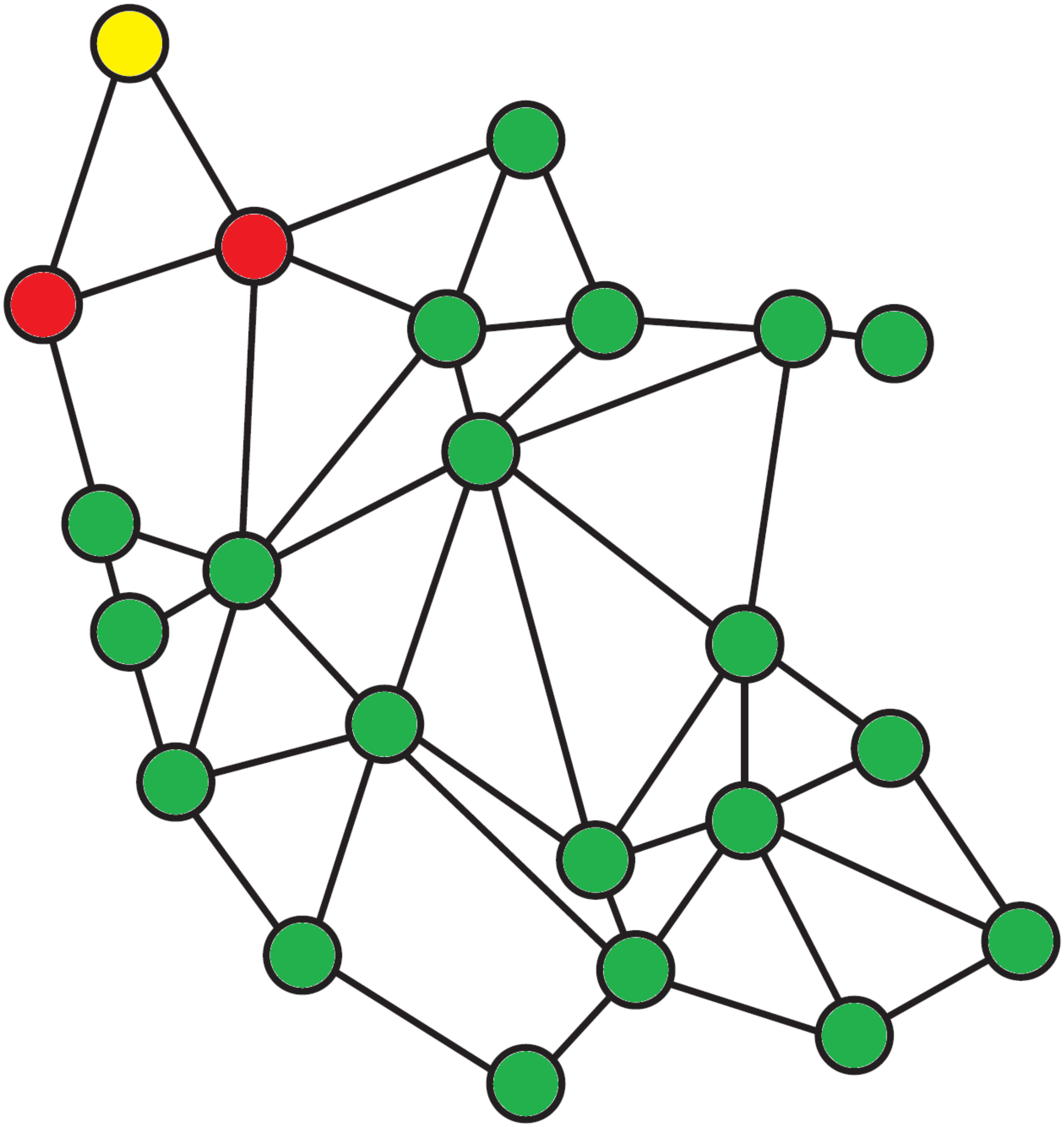


Figure 1: Global Markov Property Graph

First, we load in and process the data.

```
data(pennLC_sf)
pennLC_sf <- pennLC_sf %>%
  group_by(county) %>%
  summarize(pop = sum(population), cases = sum(cases)) %>%
  mutate(LCrate = cases/pop)
```

Now let's define our model and set up nimble. Relatively uninformative priors were selected where needed, and iterations and burn in time selected for reaching a steady state on parameters of interest.

```
model_2b_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(0, 1/10)
  tau_v ~ dgamma(1, 1/10000)
})

model_2b_data <- list(y = pennLC_sf$cases)

model_2b_consts <- list(N = nrow(pennLC_sf), pop = pennLC_sf$pop)

model_2b_inits <- list(beta_0 = 0, v = rep(0, nrow(pennLC_sf)), tau_v = 0.1)

model_2b_model <- nimbleModel(model_2b_code, data = model_2b_data,
                              constants = model_2b_consts, inits = model_2b_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
```

```
model_2b_compile <- compileNimble(model_2b_model)
```

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

```
model_2b_conf <- configureMCMC(model_2b_compile, print = T, thin = 100)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_v
## ===== Samplers =====
## RW sampler (68)
##   - beta_0
##   - v[]  (67 elements)
## conjugate sampler (1)
##   - tau_v
```

```
model_2b_conf$addMonitors(c("beta_0", "tau_v", "theta"))
```

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

```
model_2b_mcmc <- buildMCMC(model_2b_conf)
model_2b_mcmc_compile <- compileNimble(model_2b_mcmc, project = model_2b_model)
```

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

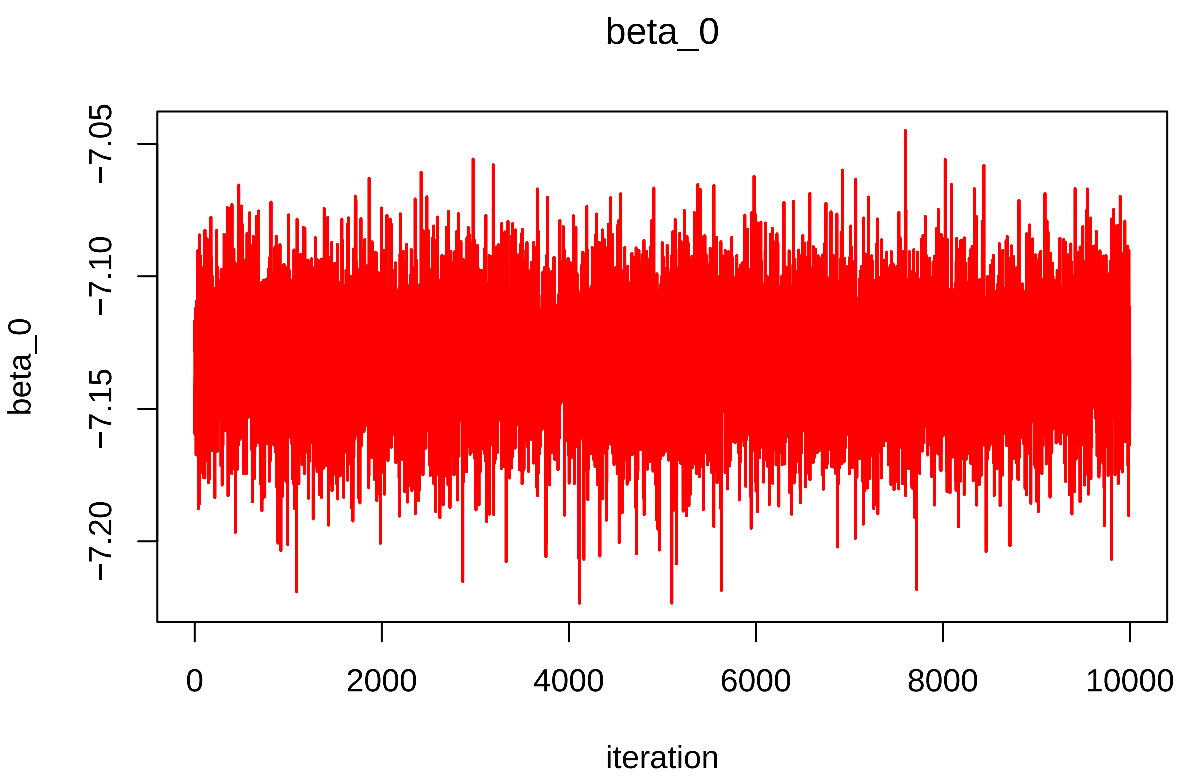
```
model_2b_samples <- runMCMC(model_2b_mcmc_compile, niter = 1100000, nburnin = 100000,
                            inits = model_2b_inits, nchains = 1, samplesAsCodaMCMC = T)
```

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

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

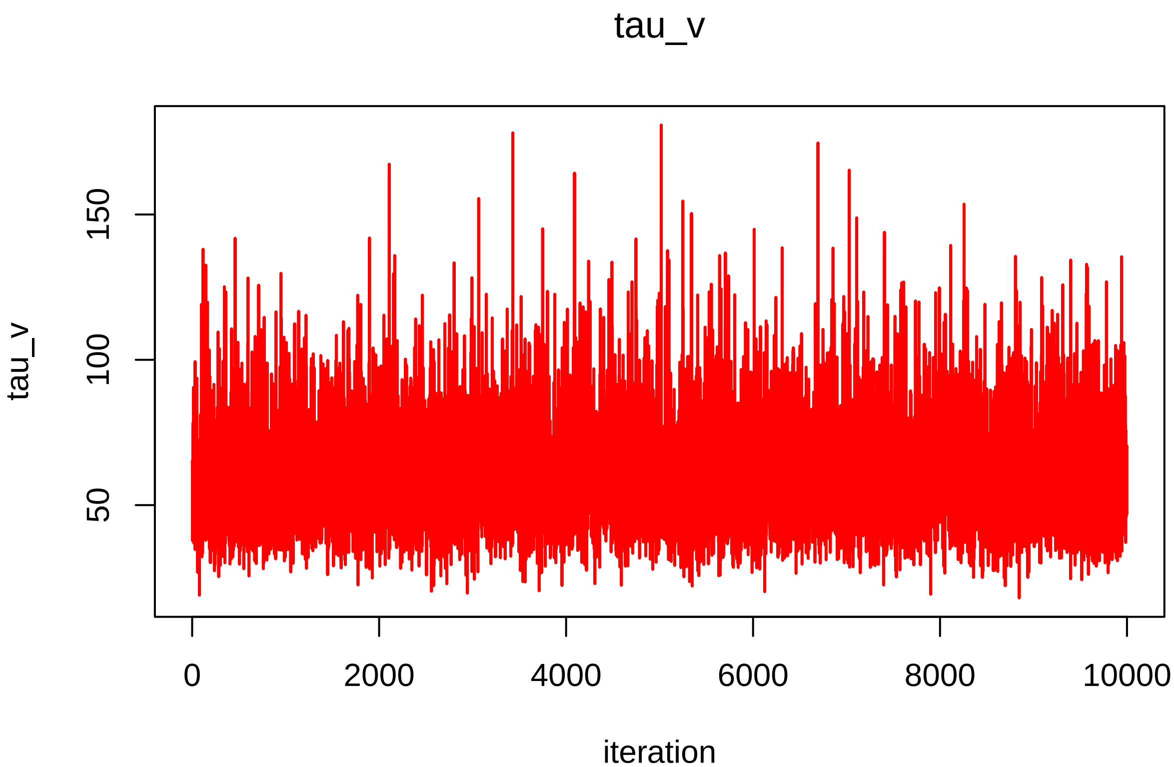
Examining posterior sampling distribution for β_0. It appears we have reached a steady state.

```
ts.plot(model_2b_samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```



Examining posterior sampling distribution for τ_v. It appears we have reached a steady state.

```
ts.plot(model_2b_samples[,"tau_v"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_v), main = expression(tau_v))
```



We take the posterior mean for each county as our point estimate.

```
model_2b_post_mean <- apply(model_2b_samples[,colnames(model_2b_samples)
                                            [startsWith(colnames(model_2b_samples),
                                                        "theta")]], 2, mean)
model_2b_post_mean
```

```
##     theta[1]     theta[2]     theta[3]     theta[4]     theta[5]     theta[6]
## 0.0006939008 0.0009851917 0.0007406995 0.0009082417 0.0007783504 0.0008209039
##     theta[7]     theta[8]     theta[9]    theta[10]    theta[11]    theta[12]
## 0.0009217707 0.0008691730 0.0007644487 0.0008773355 0.0007957820 0.0008517513
##    theta[13]    theta[14]    theta[15]    theta[16]    theta[17]    theta[18]
## 0.0007642995 0.0005844011 0.0006912374 0.0007916022 0.0008411131 0.0007750532
##    theta[19]    theta[20]    theta[21]    theta[22]    theta[23]    theta[24]
## 0.0008520543 0.0008120036 0.0006672428 0.0006998370 0.0008708092 0.0008303347
##    theta[25]    theta[26]    theta[27]    theta[28]    theta[29]    theta[30]
## 0.0008191651 0.0009450805 0.0008077061 0.0008097704 0.0008013710 0.0008518996
##    theta[31]    theta[32]    theta[33]    theta[34]    theta[35]    theta[36]
## 0.0007150281 0.0007905849 0.0007792608 0.0006803397 0.0008369994 0.0006637607
##    theta[37]    theta[38]    theta[39]    theta[40]    theta[41]    theta[42]
## 0.0008027047 0.0007036143 0.0008069480 0.0008767708 0.0008116624 0.0007891939
##    theta[43]    theta[44]    theta[45]    theta[46]    theta[47]    theta[48]
## 0.0008115317 0.0008713328 0.0007958065 0.0007986467 0.0007319898 0.0008666836
##    theta[49]    theta[50]    theta[51]    theta[52]    theta[53]    theta[54]
## 0.0008468239 0.0007705720 0.0009264359 0.0007847689 0.0008949329 0.0008717689
##    theta[55]    theta[56]    theta[57]    theta[58]    theta[59]    theta[60]
## 0.0007840519 0.0007374398 0.0007786965 0.0008060401 0.0007587600 0.0007576926
##    theta[61]    theta[62]    theta[63]    theta[64]    theta[65]    theta[66]
## 0.0009990638 0.0008027161 0.0009176778 0.0008607397 0.0009738957 0.0007214273
##    theta[67]
```

```
## 0.0007431210
```

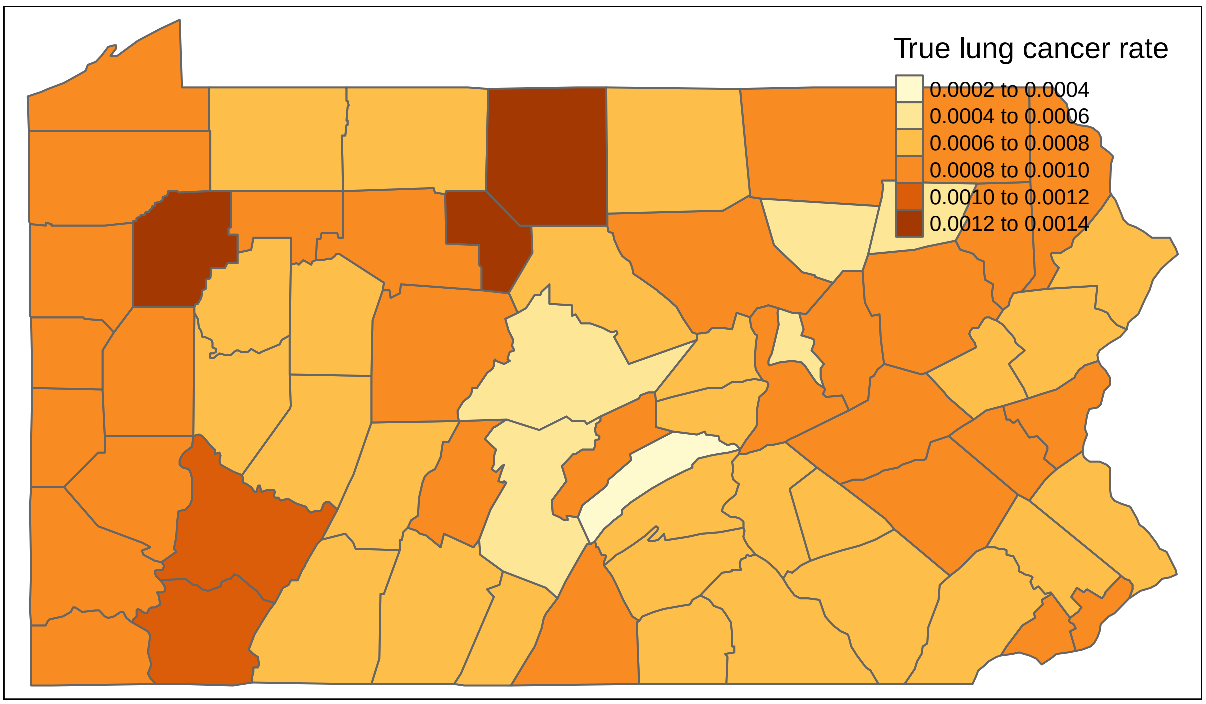
```r
summary(model_2b_post_mean)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0005844 0.0007644 0.0008027 0.0008073 0.0008564 0.0009991
```

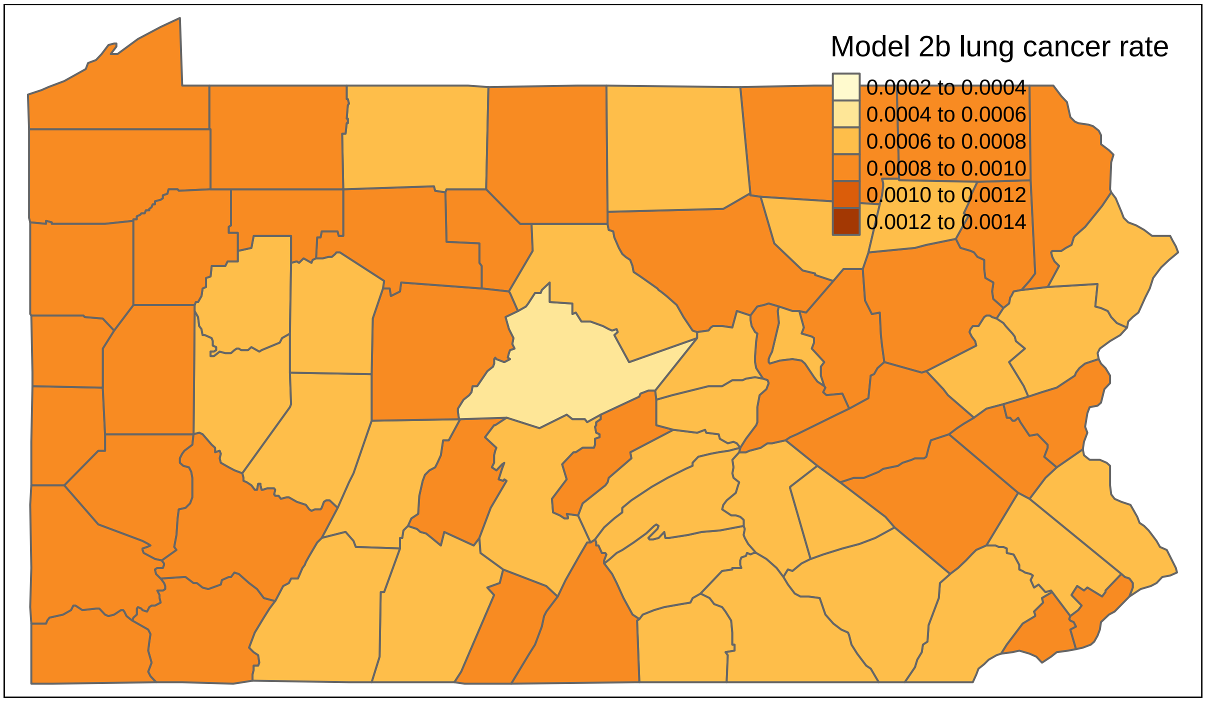
Let's plot a map of the rate estimates from our model and compare it to the true rates. We can definitely see the Bayesian "shrinkage" happening here, where far data points are pulled back a bit.

```r
pennLC_map_breaks <- seq(0.0002, 0.0014, 0.0002)

tm_shape(pennLC_sf) +
  tm_polygons(col = "LCrate", title = "True lung cancer rate",
              breaks = pennLC_map_breaks)
```




```r
mutate(pennLC_sf, model_2b_LCrate = model_2b_post_mean) %>%
  tm_shape() +
  tm_polygons(col = "model_2b_LCrate", title = "Model 2b lung cancer rate",
              breaks = pennLC_map_breaks)
```



Model 3a: CAR model with only correlated heterogeneity (slide 28)

First we need to define `num` and `adj` for the CAR model.

```
pennLC_sp <- as(pennLC_sf, "Spatial")
pennLC_nb <- poly2nb(pennLC_sp, queen = F)

num <- map_int(pennLC_nb, length)

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

Now we can define the model.

```
model_3a_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(0, 1/10)
  tau_s ~ dgamma(1, 1/10000)
})

model_3a_data <- list(y = pennLC_sf$cases)

model_3a_consts <- list(N = nrow(pennLC_sf), pop = pennLC_sf$pop, adj = adj, num = num,
                        L = length(adj), weights = rep(1, length(adj)))

model_3a_inits <- list(beta_0 = 0, s = rep(0, nrow(pennLC_sf)), tau_s = 0.1)

model_3a_model <- nimbleModel(model_3a_code, data = model_3a_data,
```

```r
                              constants = model_3a_consts, inits = model_3a_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
```

```r
model_3a_compile <- compileNimble(model_3a_model)
```

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

```r
model_3a_conf <- configureMCMC(model_3a_compile, print = T, thin = 100)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_s
## ===== Samplers =====
## RW sampler (2)
##   - beta_0
##   - tau_s
## CAR_normal sampler (1)
##   - s[1:67]
```

```r
model_3a_conf$addMonitors(c("beta_0", "tau_s", "theta"))
```

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

```r
model_3a_mcmc <- buildMCMC(model_3a_conf)
model_3a_mcmc_compile <- compileNimble(model_3a_mcmc, project = model_3a_model)
```

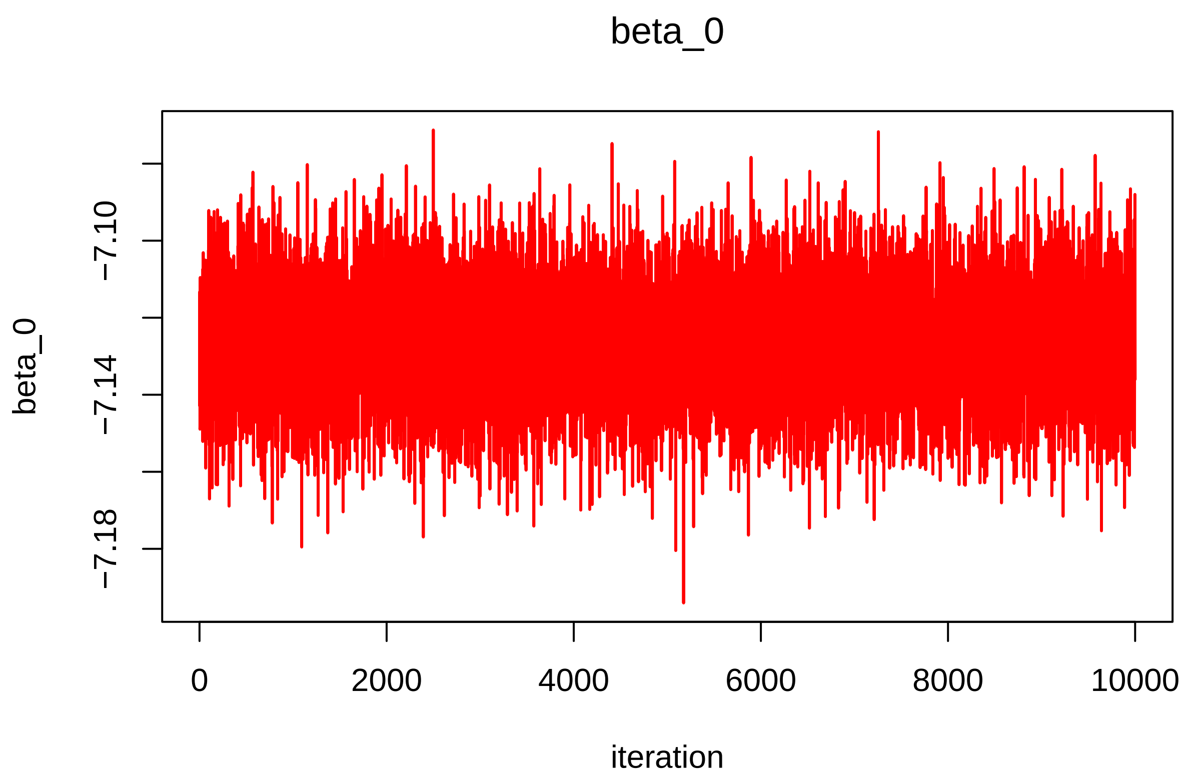
```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```r
model_3a_samples <- runMCMC(model_3a_mcmc_compile, niter = 1100000, nburnin = 100000,
                            inits = model_3a_inits, nchains = 1, samplesAsCodaMCMC = T)
```

```
## Running chain 1 ...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

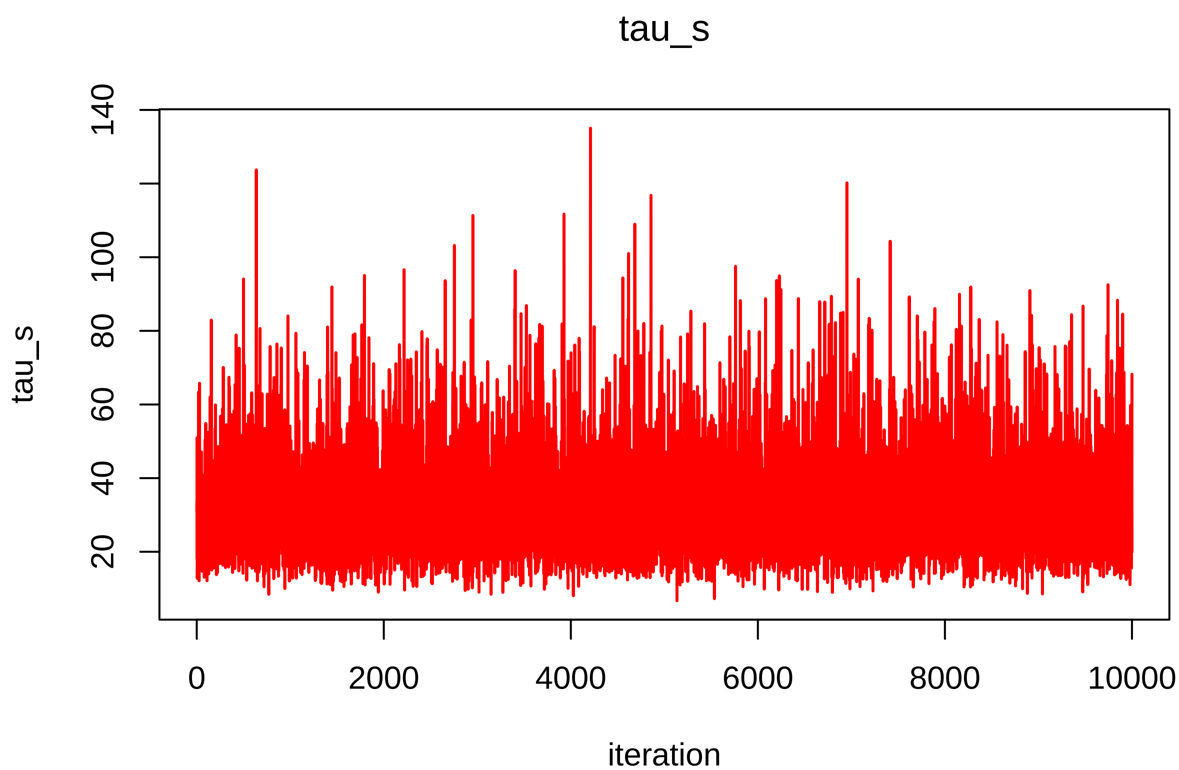
Examining posterior sampling distribution for β_0. It appears we have reached a steady state.

```r
ts.plot(model_3a_samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```



Examining posterior sampling distribution for τ_s. It appears we have reached a steady state.

```
ts.plot(model_3a_samples[,"tau_s"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_s), main = expression(tau_s))
```



We take the posterior mean for each county as our point estimate.

```r
model_3a_post_mean <- apply(model_3a_samples[,colnames(model_3a_samples)
                                            [startsWith(colnames(model_3a_samples),
                                                        "theta")]], 2, mean)
model_3a_post_mean
```

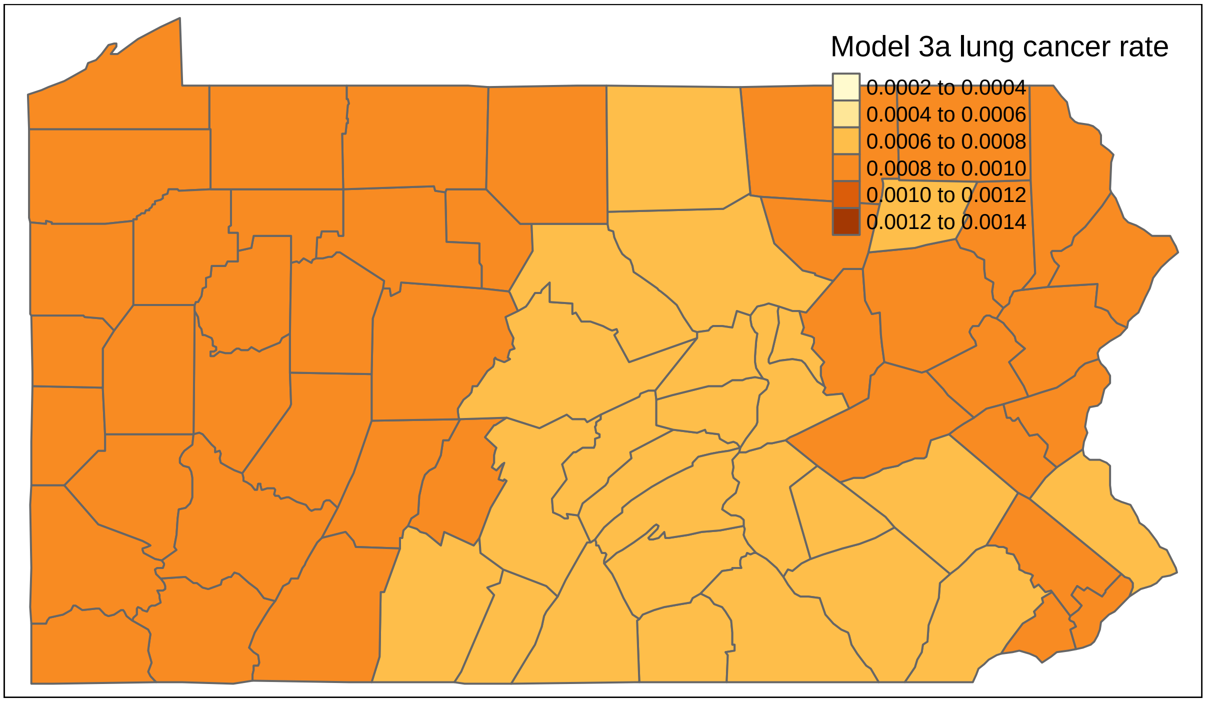
```
##     theta[1]     theta[2]     theta[3]     theta[4]     theta[5]     theta[6]
## 0.0006626010 0.0009891861 0.0008084601 0.0009372587 0.0007764835 0.0007942404
##     theta[7]     theta[8]     theta[9]    theta[10]    theta[11]    theta[12]
## 0.0008536386 0.0008384225 0.0007769162 0.0008954683 0.0008176065 0.0008315601
##    theta[13]    theta[14]    theta[15]    theta[16]    theta[17]    theta[18]
## 0.0008032668 0.0006581121 0.0007008036 0.0008422870 0.0008105631 0.0007674594
##    theta[19]    theta[20]    theta[21]    theta[22]    theta[23]    theta[24]
## 0.0008279298 0.0008452509 0.0006611964 0.0006958159 0.0008672454 0.0008357892
##    theta[25]    theta[26]    theta[27]    theta[28]    theta[29]    theta[30]
## 0.0008259553 0.0009647210 0.0008526328 0.0007383252 0.0007497225 0.0009519293
##    theta[31]    theta[32]    theta[33]    theta[34]    theta[35]    theta[36]
## 0.0007182663 0.0008211036 0.0008120335 0.0006874359 0.0008389934 0.0006658002
##    theta[37]    theta[38]    theta[39]    theta[40]    theta[41]    theta[42]
## 0.0008509856 0.0006991851 0.0008079434 0.0008582297 0.0007991708 0.0008178392
##    theta[43]    theta[44]    theta[45]    theta[46]    theta[47]    theta[48]
## 0.0008498764 0.0007589867 0.0008178562 0.0008006176 0.0007544280 0.0008543512
##    theta[49]    theta[50]    theta[51]    theta[52]    theta[53]    theta[54]
## 0.0007896110 0.0007135251 0.0009239941 0.0008043144 0.0008399804 0.0008238957
##    theta[55]    theta[56]    theta[57]    theta[58]    theta[59]    theta[60]
## 0.0007444301 0.0008030386 0.0008114739 0.0008235257 0.0007845644 0.0007393823
##    theta[61]    theta[62]    theta[63]    theta[64]    theta[65]    theta[66]
## 0.0009385788 0.0008430611 0.0009564166 0.0008521635 0.0009561342 0.0007874317
##    theta[67]
## 0.0007122022
```

```r
summary(model_3a_post_mean)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0006581 0.0007632 0.0008120 0.0008096 0.0008476 0.0009892
```

Let's plot a map of the rate estimates from our model and compare it to the true rates. The strength of the Bayesian shrinkage is similar to the previous model, but it clearly has picked up on the spatial context in the data.

```r
mutate(pennLC_sf, model_3a_LCrate = model_3a_post_mean) %>%
  tm_shape() +
  tm_polygons(col = "model_3a_LCrate", title = "Model 3a lung cancer rate",
              breaks = pennLC_map_breaks)
```



Model 3b: CAR model with both correlated and uncorrelated heterogeneity (slide 36)

Let's define the model.

```
model_3b_code <- nimbleCode({
  for (i in 1:N) {
    y[i] ~ dpois(pop[i]*theta[i])
    log(theta[i]) <- beta_0+s[i]+v[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(0, 1/10)
  tau_s ~ dgamma(1, 1/10000)
  tau_v ~ dgamma(1, 1/10000)
})

model_3b_data <- list(y = pennLC_sf$cases)

model_3b_consts <- list(N = nrow(pennLC_sf), pop = pennLC_sf$pop, adj = adj, num = num,
                        L = length(adj), weights = rep(1, length(adj)))

model_3b_inits <- list(beta_0 = 0, s = rep(0, nrow(pennLC_sf)),
                       v = rep(0, nrow(pennLC_sf)), tau_s = 0.1, tau_v = 0.1)

model_3b_model <- nimbleModel(model_3b_code, data = model_3b_data,
                              constants = model_3b_consts, inits = model_3b_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
```

```
model_3b_compile <- compileNimble(model_3b_model)
```

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

```
model_3b_conf <- configureMCMC(model_3b_compile, print = T, thin = 100)
```

```
## ===== Monitors =====
## thin = 100: beta_0, tau_s, tau_v
## ===== Samplers =====
## RW sampler (69)
##   - beta_0
##   - tau_s
##   - v[]  (67 elements)
## conjugate sampler (1)
##   - tau_v
## CAR_normal sampler (1)
##   - s[1:67]
```

```
model_3b_conf$addMonitors(c("beta_0", "tau_s", "tau_v", "theta"))
```

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

```
model_3b_mcmc <- buildMCMC(model_3b_conf)
model_3b_mcmc_compile <- compileNimble(model_3b_mcmc, project = model_3b_model)
```

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

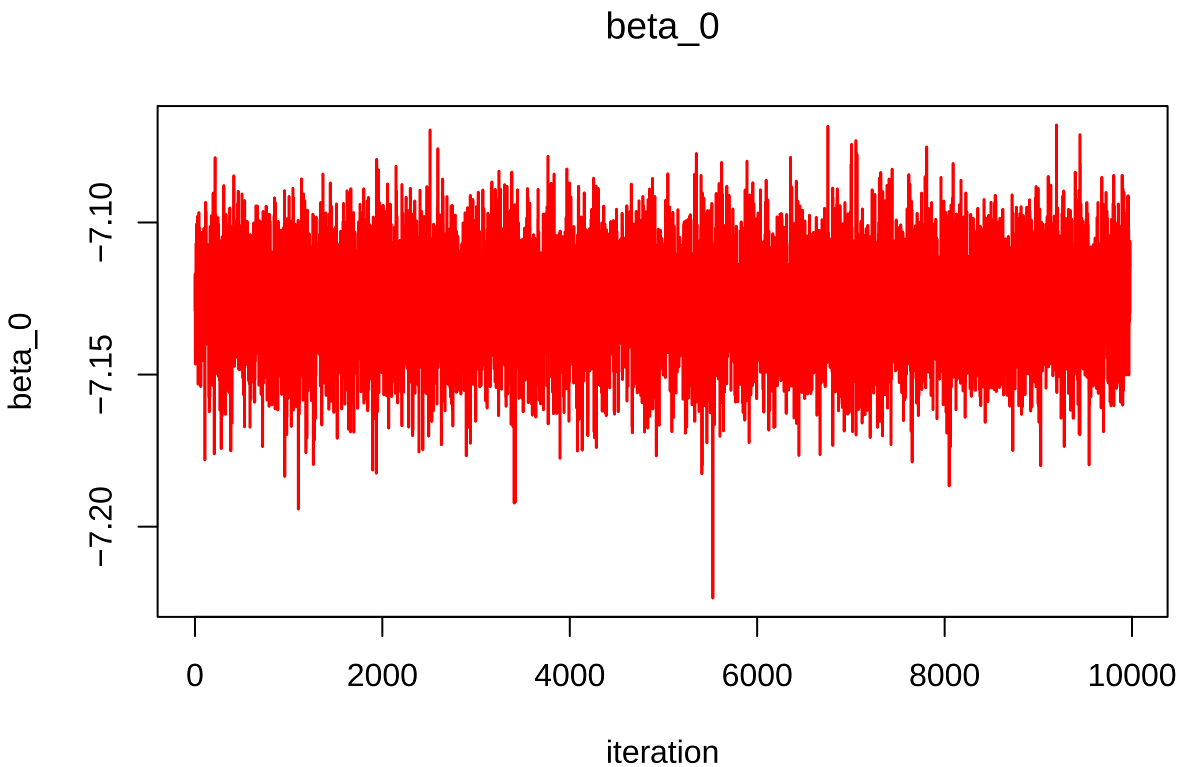
```
model_3b_samples <- runMCMC(model_3b_mcmc_compile, niter = 1100000, nburnin = 102000,
                            inits = model_3b_inits, nchains = 1, samplesAsCodaMCMC = T)
```

```
## Running chain 1 ...

## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

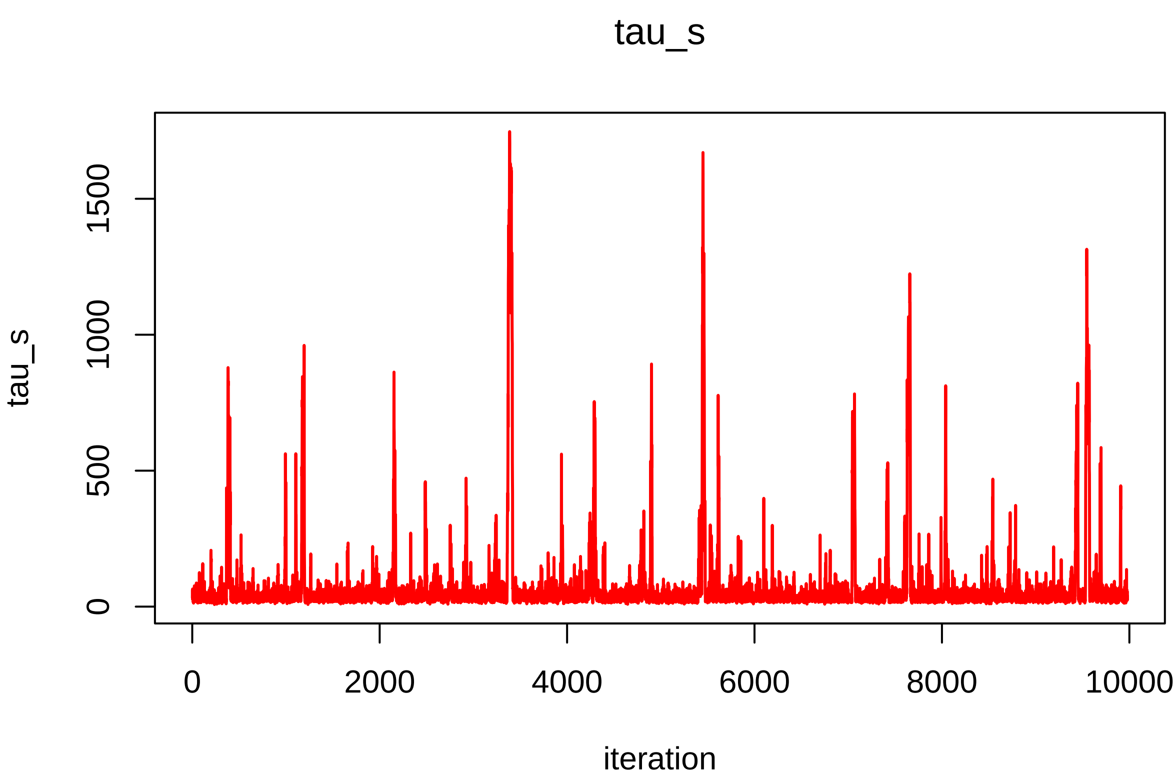
Examining posterior sampling distribution for β_0. It appears we have reached a steady state.

```
ts.plot(model_3b_samples[,"beta_0"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(beta_0), main = expression(beta_0))
```



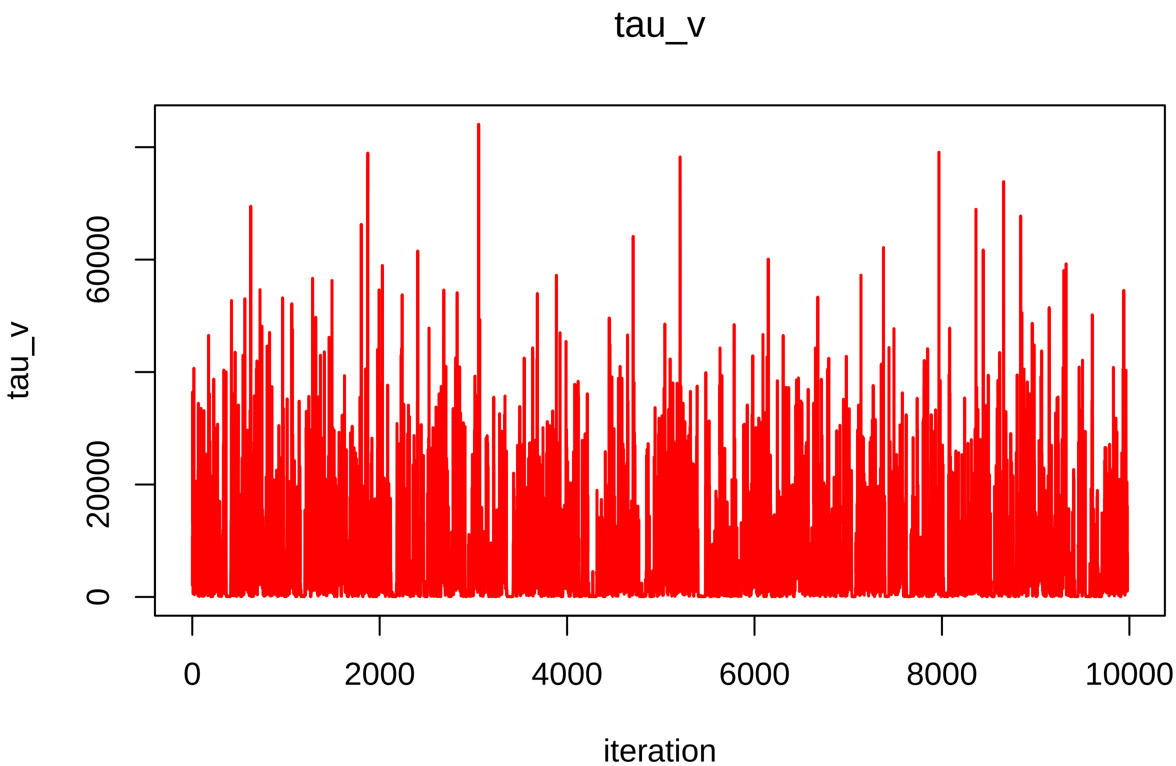
Examining posterior sampling distribution for τ_s. It appears we have reached a steady state; though there is a noticeable spike where it accepted a really far out there proposal.

```
ts.plot(model_3b_samples[,"tau_s"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_s), main = expression(tau_s))
```



Examining posterior sampling distribution for τ_v. It appears we have reached a steady state.

```
ts.plot(model_3b_samples[,"tau_v"], xlab = "iteration", col = "red", lwd = 1.5,
        ylab = expression(tau_v), main = expression(tau_v))
```



We take the posterior mean for each county as our point estimate.

```
model_3b_post_mean <- apply(model_3b_samples
                            [,colnames(model_3b_samples)
                              [startsWith(colnames(model_3b_samples),
                                          "theta")]], 2, mean)
model_3b_post_mean
```

```
##     theta[1]     theta[2]     theta[3]     theta[4]     theta[5]     theta[6]
## 0.0006653053 0.0009888289 0.0008021121 0.0009345678 0.0007780129 0.0007989340
##     theta[7]     theta[8]     theta[9]    theta[10]    theta[11]    theta[12]
## 0.0008620165 0.0008400328 0.0007752183 0.0008941459 0.0008147046 0.0008334922
##    theta[13]    theta[14]    theta[15]    theta[16]    theta[17]    theta[18]
## 0.0007962698 0.0006491398 0.0007004955 0.0008371113 0.0008159454 0.0007697506
##    theta[19]    theta[20]    theta[21]    theta[22]    theta[23]    theta[24]
## 0.0008279733 0.0008429217 0.0006618310 0.0006955892 0.0008671244 0.0008354715
##    theta[25]    theta[26]    theta[27]    theta[28]    theta[29]    theta[30]
## 0.0008268043 0.0009621336 0.0008488230 0.0007478196 0.0007550102 0.0009420805
##    theta[31]    theta[32]    theta[33]    theta[34]    theta[35]    theta[36]
## 0.0007188631 0.0008181033 0.0008096945 0.0006854381 0.0008371347 0.0006651778
##    theta[37]    theta[38]    theta[39]    theta[40]    theta[41]    theta[42]
## 0.0008489931 0.0006999365 0.0008068173 0.0008602890 0.0007995036 0.0008146132
##    theta[43]    theta[44]    theta[45]    theta[46]    theta[47]    theta[48]
## 0.0008471205 0.0007737671 0.0008141288 0.0008006688 0.0007519649 0.0008562827
##    theta[49]    theta[50]    theta[51]    theta[52]    theta[53]    theta[54]
## 0.0007952545 0.0007179828 0.0009242354 0.0008026664 0.0008457706 0.0008294270
```

```
##    theta[55]    theta[56]    theta[57]    theta[58]    theta[59]    theta[60]
## 0.0007480890 0.0007959049 0.0008033128 0.0008203209 0.0007801064 0.0007434390
##    theta[61]    theta[62]    theta[63]    theta[64]    theta[65]    theta[66]
## 0.0009476823 0.0008390119 0.0009522649 0.0008526659 0.0009589800 0.0007785365
##    theta[67]
## 0.0007166560
```

```
summary(model_3b_post_mean)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0006491 0.0007718 0.0008141 0.0008094 0.0008464 0.0009888
```

Let's plot a map of the rate estimates from our model and compare it to the true rates. This model seems to be very similar to the previous. It doesn't look like allowing for some uncorrelated heterogeneity on top of the correlated heterogeneity really adds or changes much.

```
mutate(pennLC_sf, model_3b_LCrate = model_3b_post_mean) %>%
  tm_shape() +
  tm_polygons(col = "model_3b_LCrate", title = "Model 3b lung cancer rate",
              breaks = pennLC_map_breaks)
```