Bayesian model in JAGS

Tutorial 3 for transition

Zhe Zheng (Gigi)

2023-01-16

```r
knitr::opts_chunk$set(echo = TRUE)
require(rjags)
#"Note that the rjags package does not include a copy of
# the JAGS you must install this separately.
# For instructions on downloading JAGS,
# see the home page at https://mcmc-jags.sourceforge.io."
require(mcmcplots) # to plot the trace plots
require(coda) # for gelman diagnostic
```

Outline

In this tutorial, we will learn how to use JAGS (Just Another Gibbs Sampler) in R to estimate unknown parameters with Markov Chain Monte Carlo (MCMC) methods in Bayesian framework. If you are interested in learning more about the theoretical knowledge of MCMC and Bayesian modeling, please check out courses "Probability and Statistics" (taught by Prof. Joseph Chang) and "Bayesian Statistics" (taught by Prof. Josh Warren).

In the first section, we will introduce a simple jags model to help understand the syntax for rjags package.

In the second section, we will learn how to build up complicated models step by step (censor data). Before reading this section, you will need to familiar yourself with the RSV_timing_explain tutorials and a research article on the relative RSV timing

In the third section, we will introduce the concept of spatial autocorrelation and how to incorporate this in jags model.

Example 1: a simple jags model to estimate the unknown parameters

First: Simulate a dataset for analysis

In this example, I simulate my own data in R in order to compare the MCMC results with our input parameters. I create a continuous outcome variable y as a function of one predictor x and an error term ϵ . I simulate 100 observations. The linear regression looks like

$$y = \beta_0 + \beta_1 x + \epsilon$$

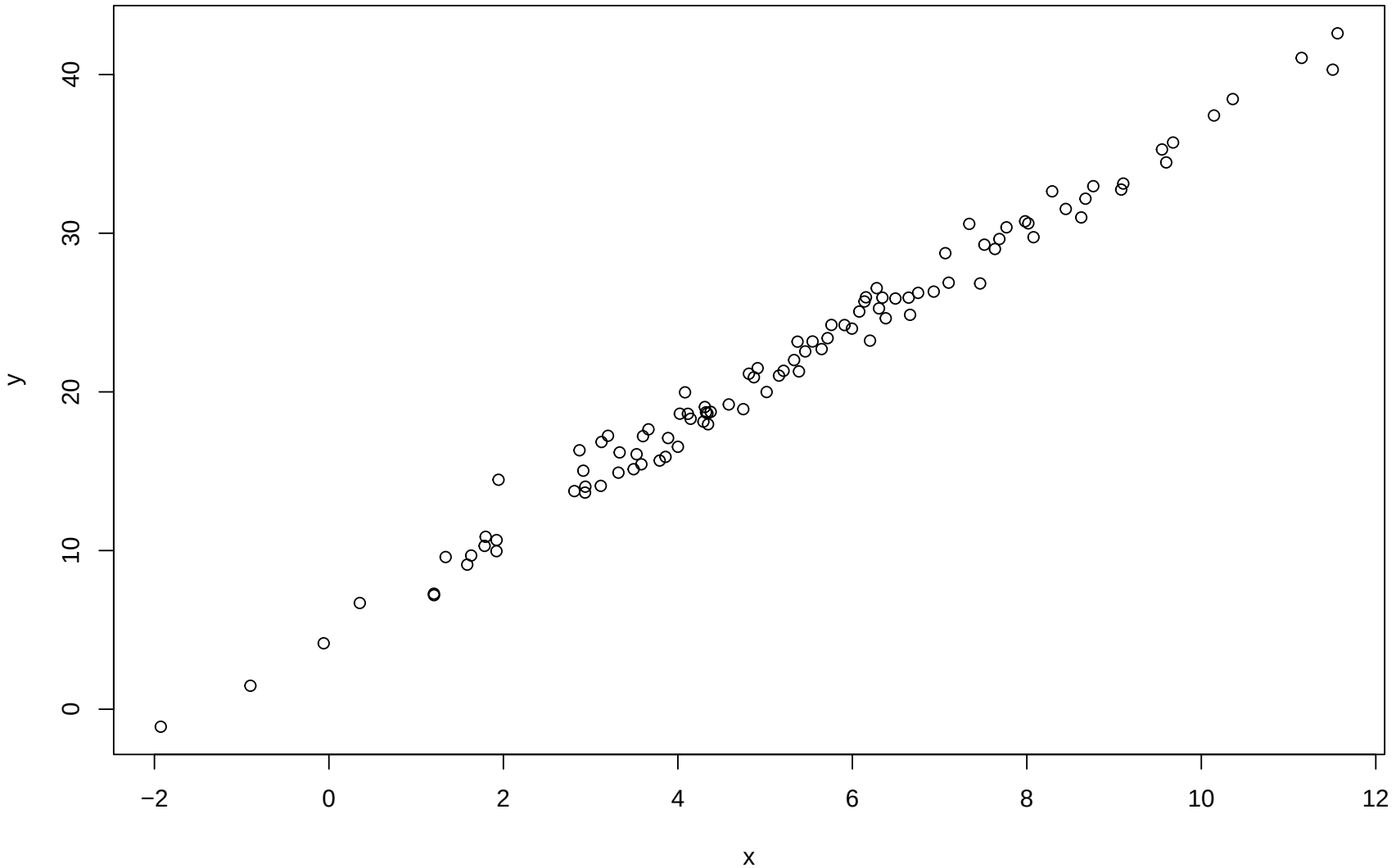
Our goal is to estimate the unknown parameters β_0 and β_1 given the observations and the linear relationship

```
n.sim=100; set.seed(123)
x=rnorm(n.sim, mean = 5, sd = 3)
epsilon=rnorm(n.sim, mean = 0, sd = 1)
beta0=5
beta1=3.2
y=beta0 + beta1 * x + epsilon
```

Visualize the observations

Question: what is the intercept and slope for this linear regression?

```
plot(x,y,type="p")
```



rjags mamual

The following is quoted from rjags document, authored by Martyn Plummer:

"JAGS is a clone of BUGS (Bayesian analysis Using Gibbs Sampling). See Lunn et al (2009) for a history of the BUGS project. Note that the rjags package does not include a copy of the JAGS library: you must install this separately. For instructions on downloading JAGS, see the home page at https://mcmc-jags.sourceforge.io.

To fully understand how JAGS works, you need to read the JAGS User Manual. The manual explains the basics of modelling with JAGS and shows the functions and distributions available in the dialect of the BUGS language used by JAGS. It also describes the command line interface. The rjags package does not use the command line interface but provides equivalent functionality using R functions."

rjags mamual (continued)

"Analysis using the rjags package proceeds in steps: 1. Define the model using the BUGS language in a separate file.

2. Read in the model file using the jags.model function. This creates an object of class "jags".

3. Update the model using the update method for "jags" objects. This constitutes a 'burn-in' period.

4. Extract samples from the model object using the coda.samples function. This creates an object of class "mcmc.list" which can be used to summarize the posterior distribution. The coda package also provides convergence diagnostics to check that the output is valid for analysis (see Plummer et al 2006)."

Jags model structure (Step 1)

```
basic_mod <-  "model{
 #model
  for(i in 1:n.sim){
   y[i] ~ dnorm(mu[i], tau)
# tau is the error term

   mu[i] = beta0 + beta1 * x[i]
# beta0 and beta1 are the unknown parameters
  }

 #priors; these are weakly informative priors
 beta0 ~ dnorm(0, 0.01)
# dnorm(mean, precision)
# precision = 0.01 --> var = 100
 beta1 ~ dnorm(0, 0.01)
 tau ~ dgamma(0.01,0.01)
# dgamma(shape,rate)
 }
 "
```

Initialize model with known data and initial guess of unknown parameters (Step 2)

```
######## Define known data ############
datalist=list("y"=y,"x"=x,"n.sim"=n.sim)

######## Set initial values for parameter estimate ########
inits=function(){list("beta0"=rnorm(1), "beta1"=rnorm(1))}
```

Read in the model file using the jags.model function (Step 3)

```
### Read in the model file using the jags.model function ##
jags.basic.mod = jags.model(textConnection(basic_mod),
                            data = datalist,
                            inits = inits,
                            n.chains = 4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 100
##    Unobserved stochastic nodes: 3
##    Total graph size: 406
##
## Initializing model
```

Discard burn-in period and sample for posterior distribution (Step 4)

```
############ burn-in period ############
update(jags.basic.mod,5000) # n.burnin = 1000

##### sample for posterior distribution #######
model_sample <- coda.samples(jags.basic.mod,
                             c("beta0","beta1"),
                             100000,thin=5)
#You can also use jags.samples(). The results will be the same
```

Summarize the posterior distribution

```
summary(model_sample)
```

```
##
## Iterations = 5005:105000
## Thinning interval = 5
## Number of chains = 4
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##         Mean      SD  Naive SE Time-series SE
## beta0 4.983 0.21338 0.0007544      0.0010266
## beta1 3.183 0.03596 0.0001271      0.0001746
##
## 2. Quantiles for each variable:
##
##         2.5%   25%   50%   75% 97.5%
## beta0 4.563 4.840 4.983 5.126 5.400
## beta1 3.112 3.159 3.183 3.207 3.253
```

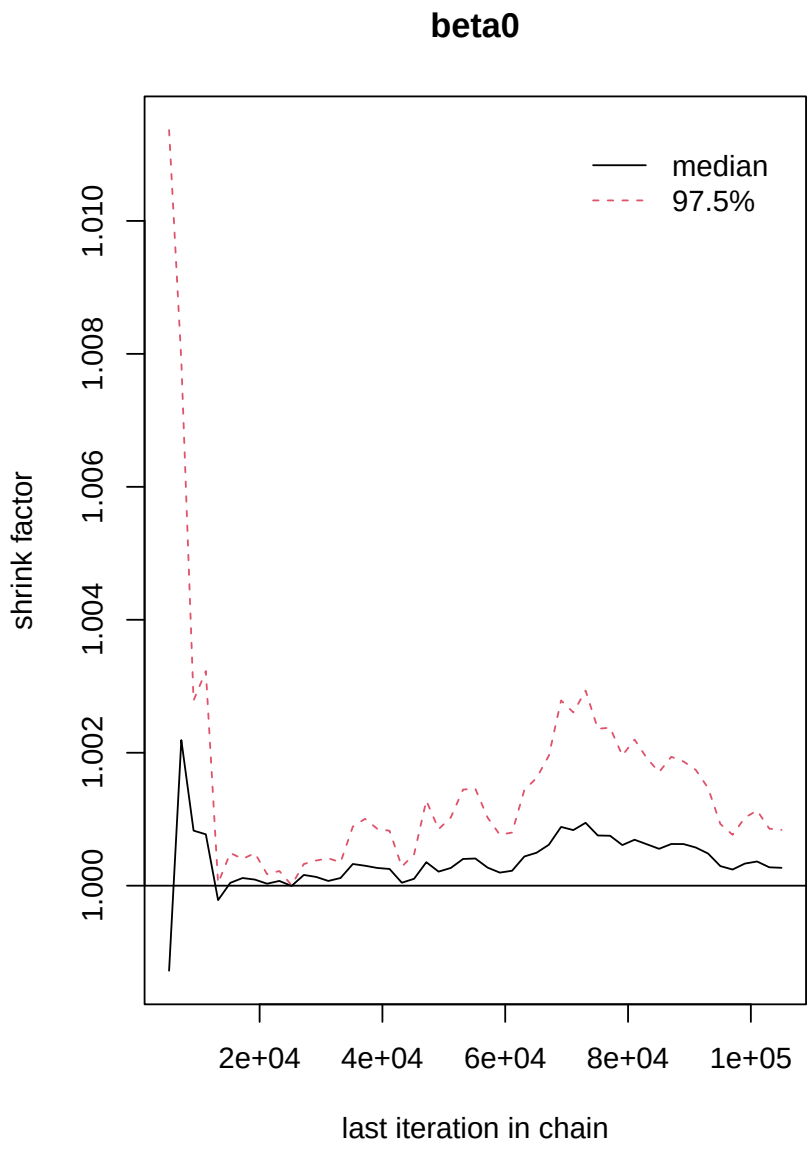
Check for model convergence (gelman diagnotics)

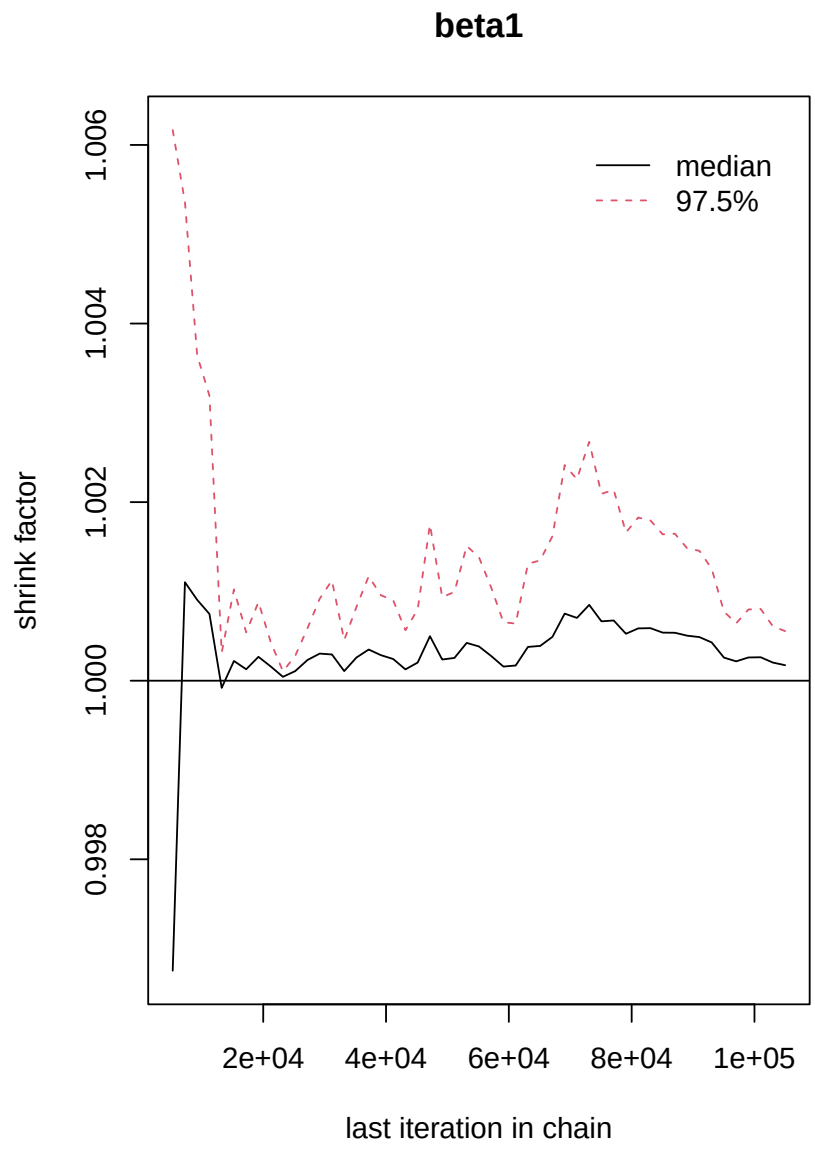
```
gelman.diag(model_sample) ## <=1.1 means well converge
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
##
## Multivariate psrf
##
## 1
```

Check for model convergence (gelman plot)

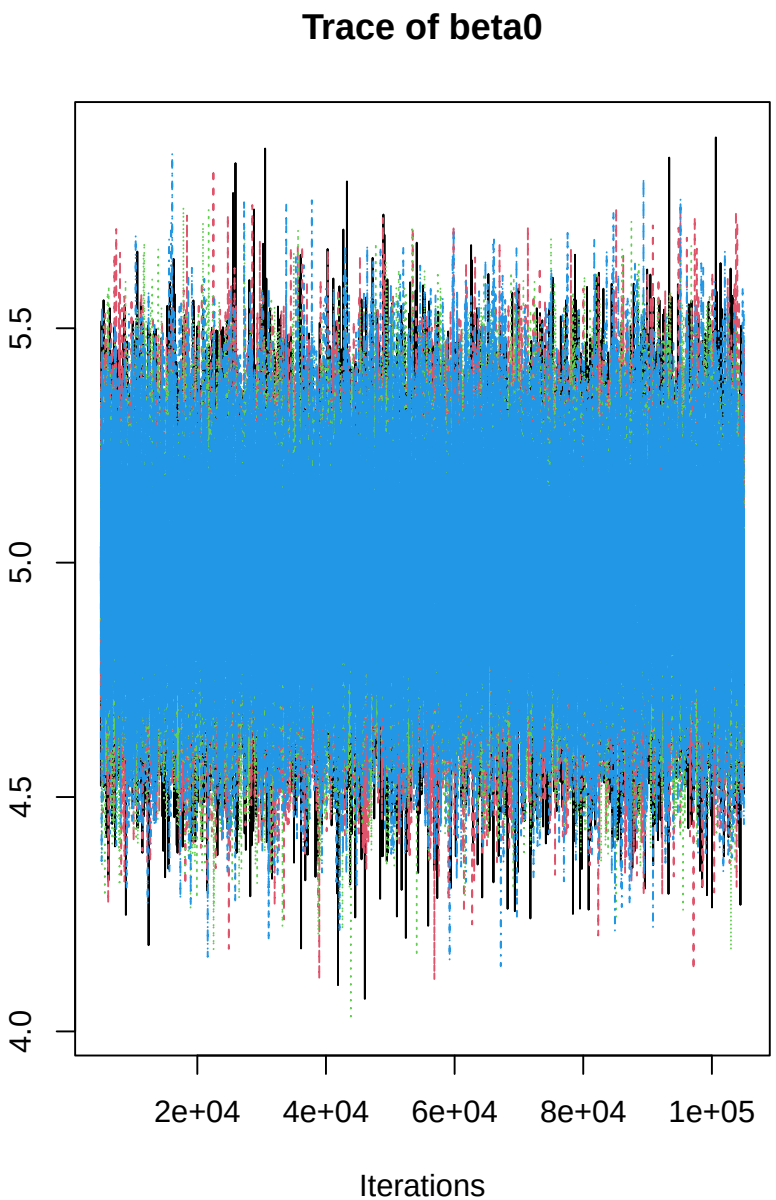
```
gelman.plot(model_sample)
```

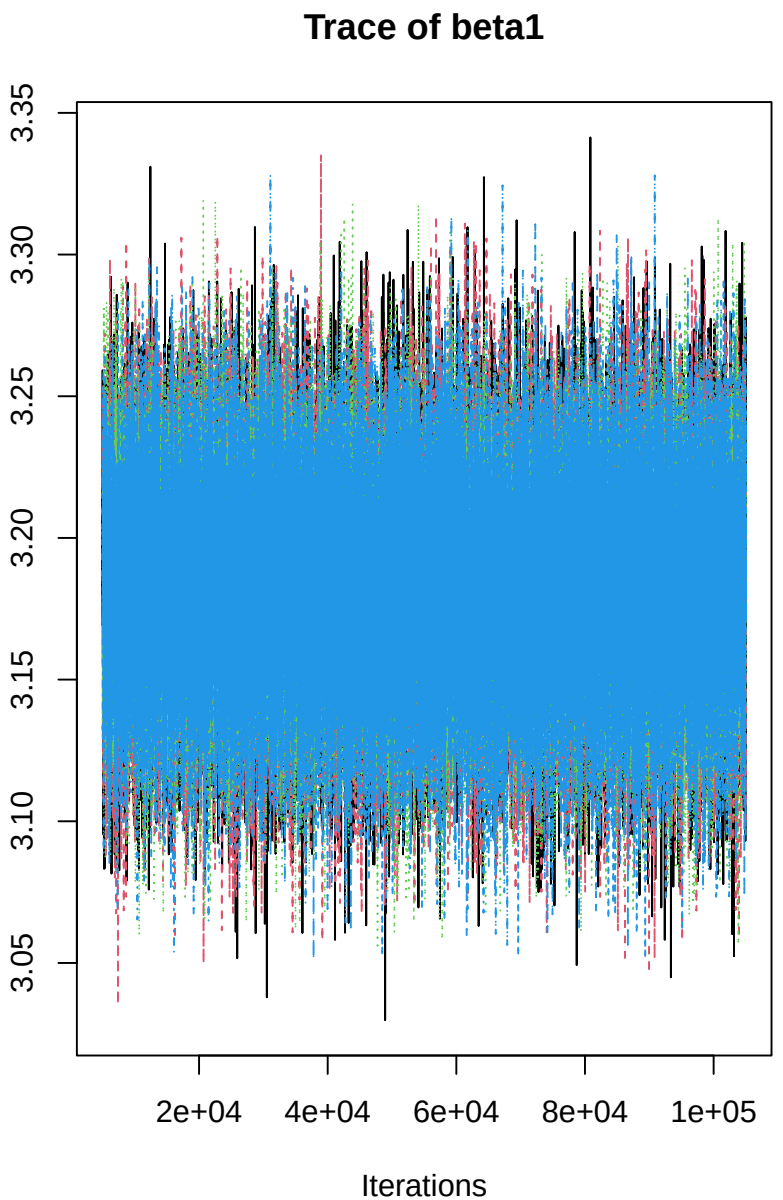




Plot the trace plot

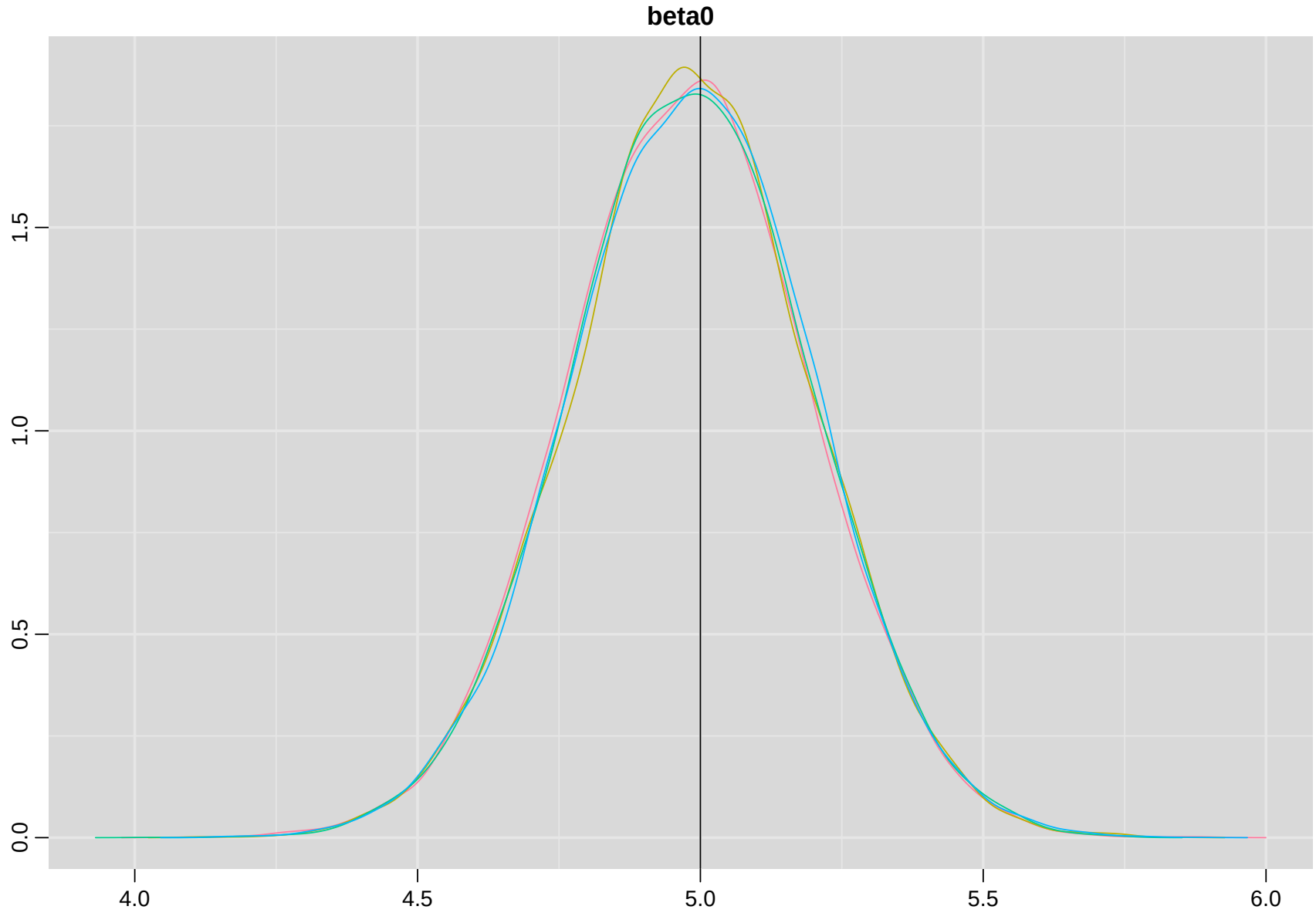
```
par(mfrow=c(1,2))
traceplot(model_sample)
```





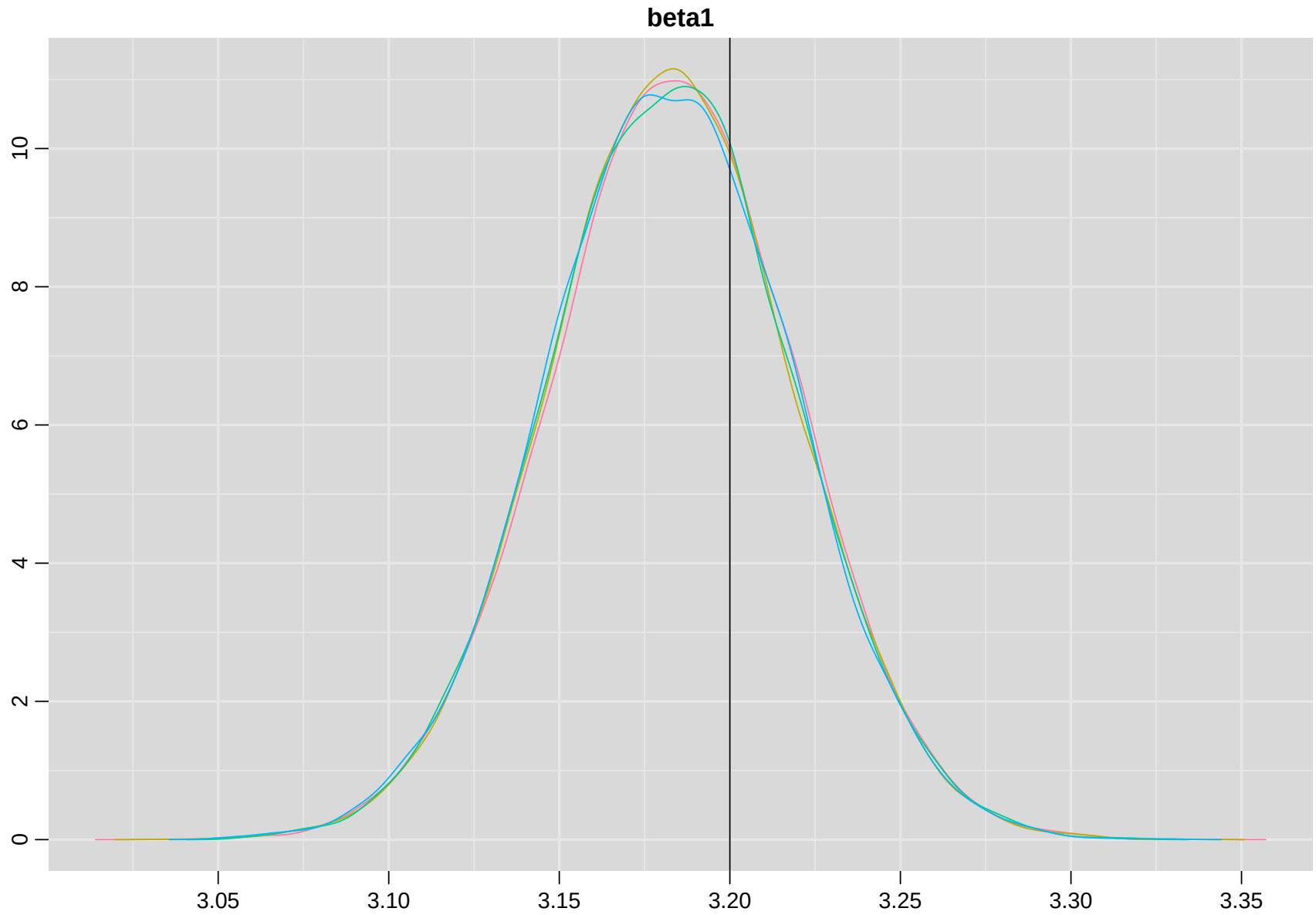
Compare the posterior density and real value for beta0

```
denplot(model_sample, parms = c("beta0"))
abline(v=5)  # True value
```



Compare the posterior density and real value for beta1

```
denplot(model_sample, parms = c("beta1"))
abline(v=3.2) # True value
```



Section 2: estimate the parameters for the relative RSV timing

After we estimated the RSV timing using second derivative method (please check: RSV_timing_explain), we would like to know if some potential drivers contribute to the relative timing difference in RSV epidemic onsets before and during COVID-19 pandemic. (In reality, we manually coded the spatial autocorrelated model. As this requires taking Josh's classes, I will not go over it in the introduction tutorials.)

The model is given as:

$$Y_i = \alpha_{0i} + \beta_1 \overline{W_i} + \epsilon_i$$

$$\alpha_{0i} = \beta_0 + \boldsymbol{x_i^T \lambda} + \theta_i$$

Meaning of the variables and parameters

where Y_i is the relative timing of RSV epidemic onset with respect to Florida in 2021 in state i (measured in weeks), and $\overline{w_i}$ is the average relative timing of RSV seasons in state i compared to Florida from 2016-2019. The global intercept parameter β_0 and slope parameter β_1 describe the overall similarity between a typical year's relative onset timing across all states and the relative timings observed in 2021. The observation-level random effect ϵ_i $N(0, \sigma^2)$ accounts for residual variability in the data.

The state-specific intercepts, $\alpha_{0i} = \beta_0 + \boldsymbol{x_i^T \lambda} + \theta_i$, represent the state-level similarity in relative timing for 2021 compared to previous seasons, where x_i is the vector of covariates that could potentially explain differences in the relative timing of RSV between 2021 and previous years. The covariates include population density, average household size, and stringency index of non-pharmaceutical interventions against COVID-19.

θ_i represents a spatially correlated random effect, which we will learn more in detail later.

Jags model code without considering spatial autocorrelation

```
Nospatial_model <- "model{
################## model ###################################
for(i in 1:n){
Y[i] ~ dnorm(mu[i], inv_var)
# for the observed timing (we log transform Y already)
 mu[i] <- alpha_0[i] + beta1*log_W[i]
 # log_W is the log transform average relative timing of RSV

alpha_0[i]= beta0+x1[i]*lambda1+x2[i]*lambda2+lambda3*x3[i]
# this is alpha_0i = beta0 + X_i*lambda (coviariates)
}

################## priors ##################################
beta0~dnorm(0,0.0001)
beta1~dnorm(0,0.0001)
lambda1~dnorm(0,0.0001)
lambda2~dnorm(0,0.0001)
lambda3~dnorm(0,0.0001)
```

Load the data and run the jags model to sample posteriors

```
## loading data from the working directory
data_nospatial <- readRDS("./Nospatial_data.rds")

### Read in the model file using the jags.model function ##
non_spatial <- jags.model(textConnection(Nospatial_model),
                          data=data_nospatial,n.chains = 4)
```

```
## Warning in jags.model(textConnection(Nospatial_model), data =
## Unused variable "Z" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 0
##    Unobserved stochastic nodes: 53
##    Total graph size: 491
##
## Initializing model
```

```
############ burn-in period ############
update(non_spatial,10000)
```

Summarize the posterior distribution

```
summary(non_spatial_sample)
```

```
##
## Iterations = 10020:110000
## Thinning interval = 20
## Number of chains = 4
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD Naive SE Time-series SE
## beta0     0.13434  99.600  0.70428        0.70429
## beta1     0.54632 100.173  0.70833        0.71039
## inv_var   0.97189   8.898  0.06292        0.06359
## lambda1 -1.03338  99.502  0.70359        0.70355
## lambda2 -0.76331  99.989  0.70703        0.70702
## lambda3  0.02948 100.590  0.71128        0.71637
##
## 2. Quantiles for each variable:
##
```

Check for model convergence (gelman diagnostic)

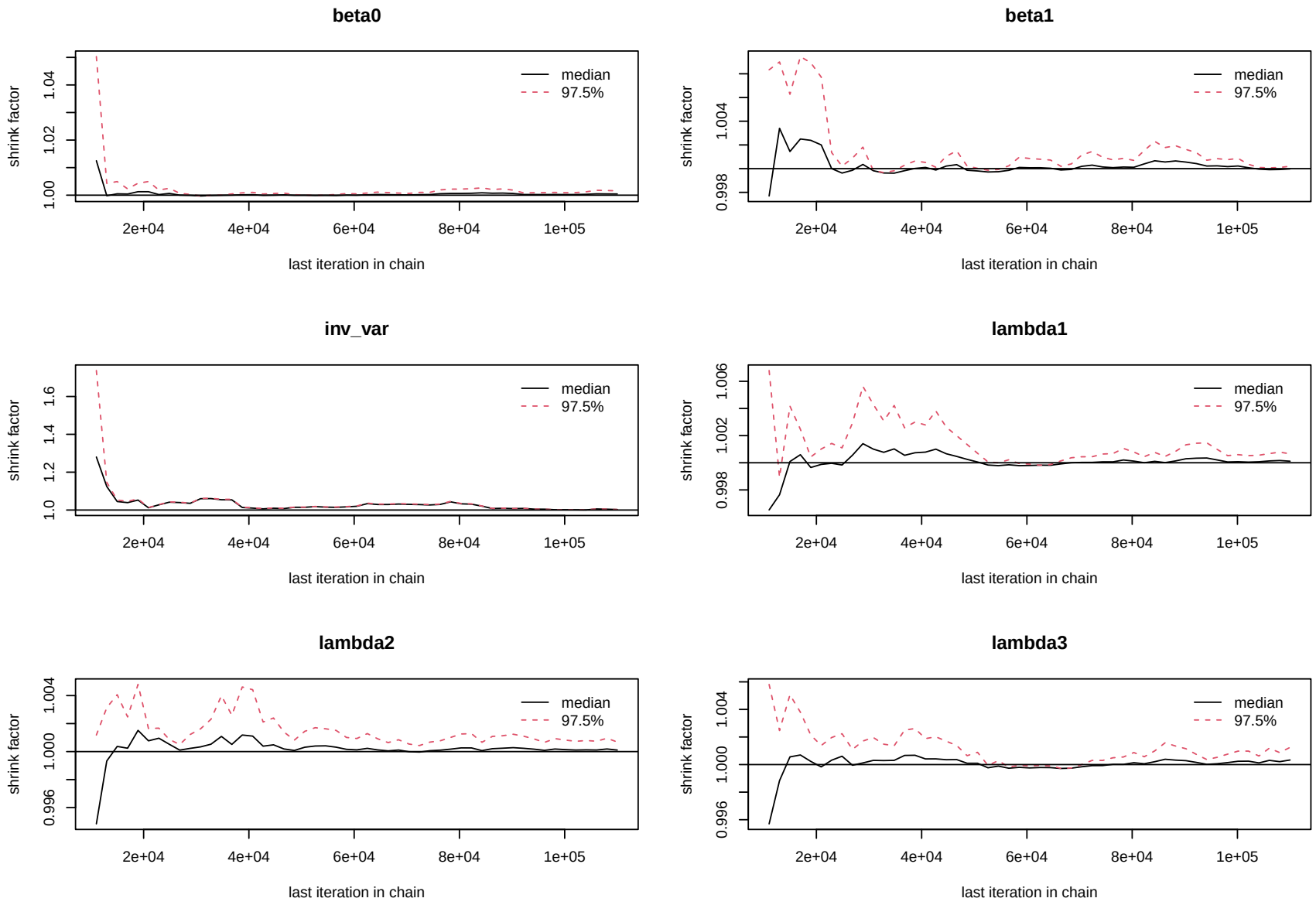
inv_var did not converge well. This indicates that we should either prolong the iterations or re-parameterize the model. Using gelman.plot (next slides), we could tell that prolong iterations may help convergence.

```
gelman.diag(non_spatial_sample)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## beta0            1          1
## beta1            1          1
## inv_var          1          1
## lambda1          1          1
## lambda2          1          1
## lambda3          1          1
##
## Multivariate psrf
##
## 1
```

Gelman plot

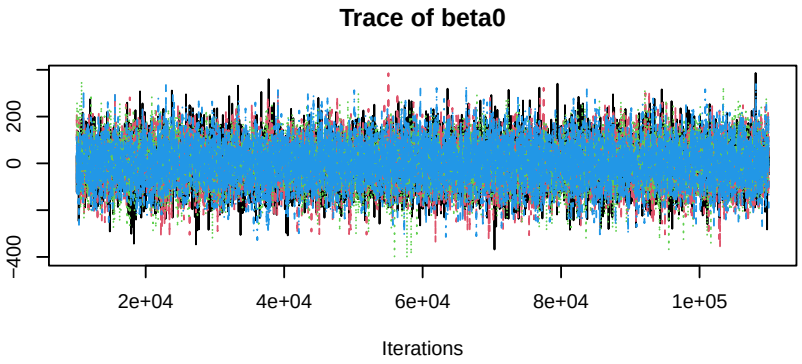
```
par(mfrow=c(3,2))
gelman.plot(non_spatial_sample)
```

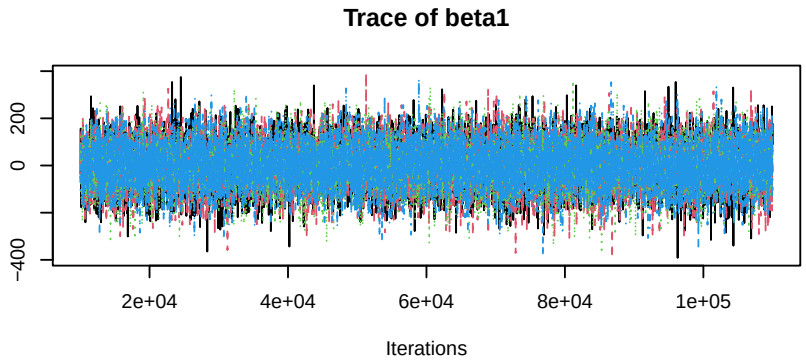


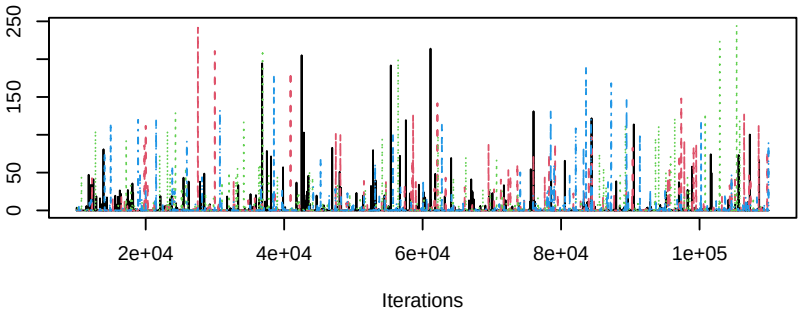
Trace plots

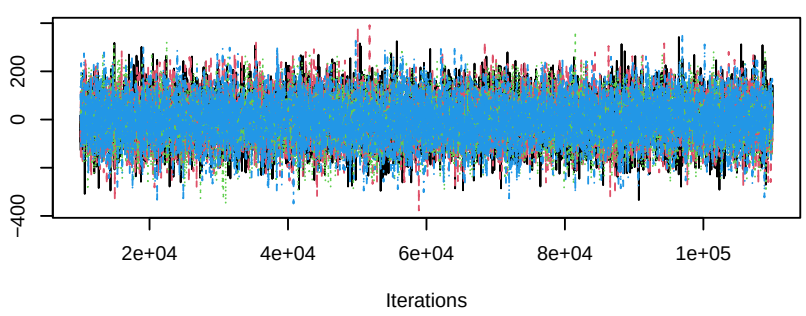
note: inv_var is precision = the inverse of variance

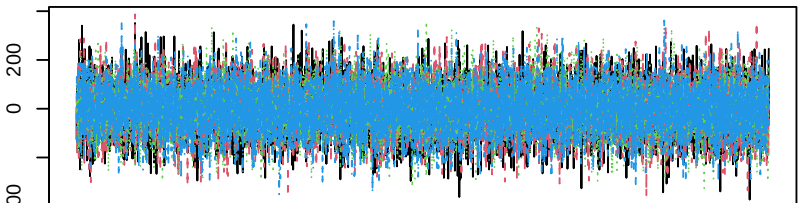
```
par(mfrow=c(3,2))
traceplot(non_spatial_sample)
```

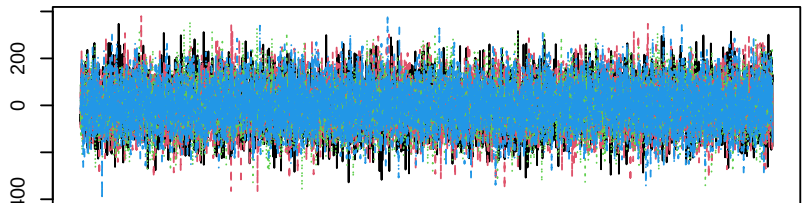






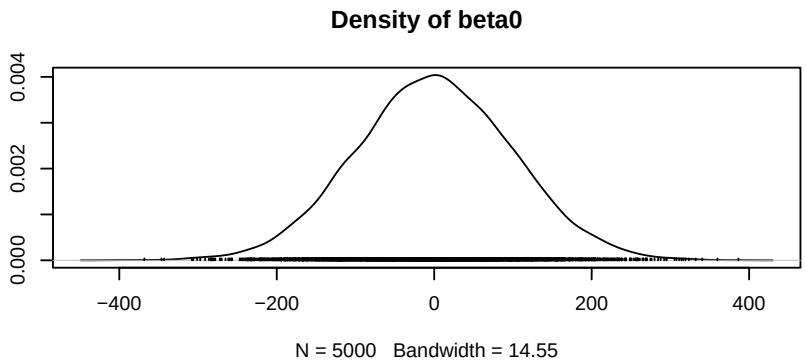


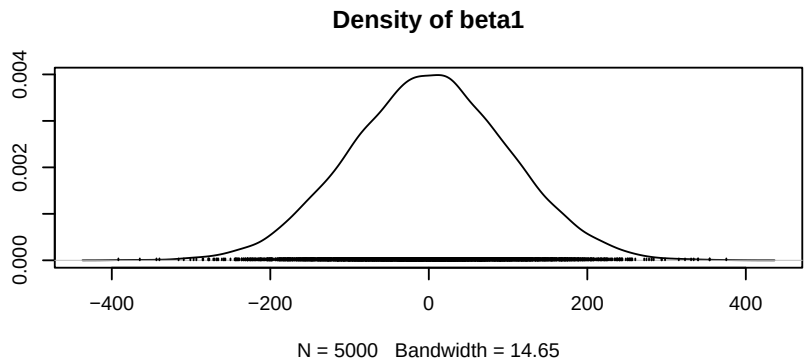


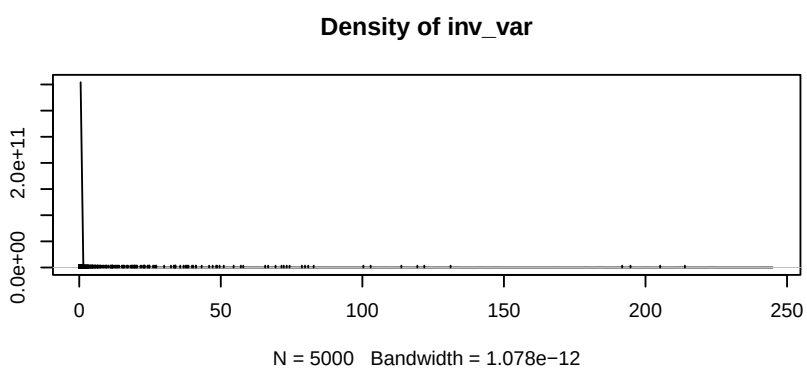


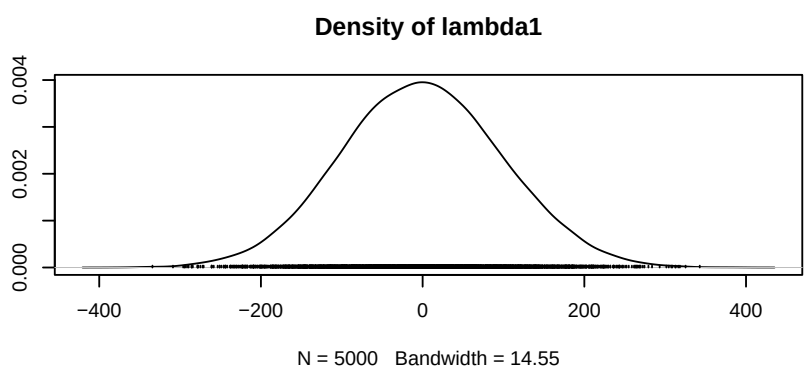
Density plot of the posterior distributions

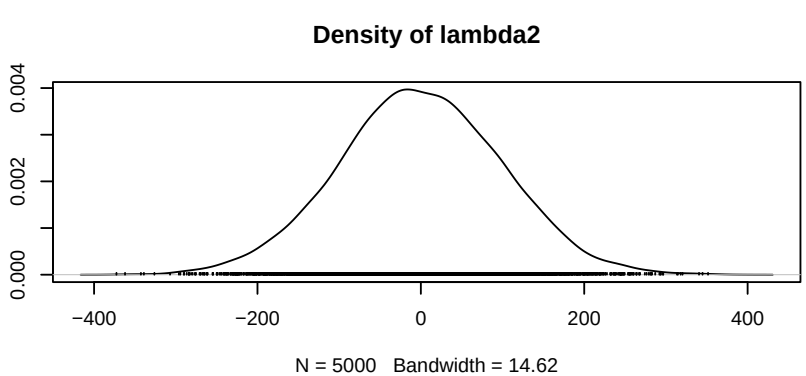
```
par(mfrow=c(3,2))
densplot(non_spatial_sample)
```

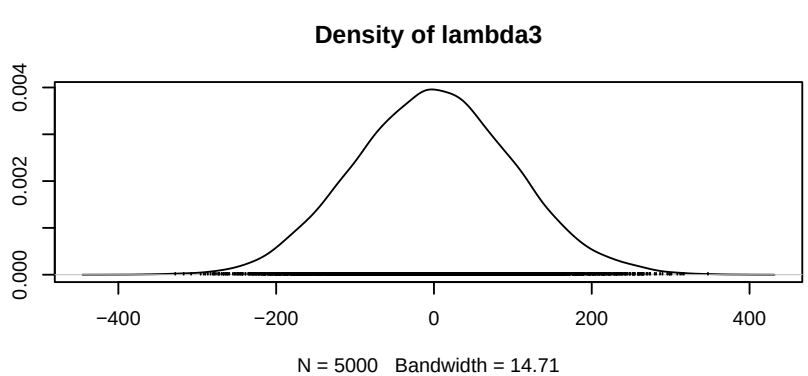












Section 2.1: right censored survival analysis

In the early phase of an epidemic, we may not yet observe the onset point. For example, if you consider an onset threshold and the cases are still below the threshold, the epidemic has not taken off. In this case, we need to use right censored survival analysis.

Click to know more about right censored survival analysis

Model structure for right censored survival analysis

```
censormodel <- "model{
for(i in 1:n){
is.censor[i] ~ dinterval(Z[i],t.cen)
# whether the data is censor and the censoring time

Z[i] ~ dlnorm(mu[i], inv_var)
# for the observed timing (we did not log transform Z)
# if we already log transform Z then we use dnorm

 mu[i] <- alpha_0[i] + beta1*log_W[i]
# log_W is the log transform average relative timing of RSV
alpha_0[i]= beta0+x1[i]*lambda1+x2[i]*lambda2
# this is alpha_0i = beta0 + X_i*lambda (coviariates)
}
```

Model structure for right censored survival analysis (continued)

```
################## priors ###################################
beta0~dnorm(0,0.0001)
beta1~dnorm(0,0.0001)
lambda1~dnorm(0,0.0001)
lambda2~dnorm(0,0.0001)
inv_var ~ dgamma(0.01, 0.01)}"
```

Since the model running process is the same, I will skip the model fitting process in this tutorial (it takes time to knit the PDF). To run the model, you need to combine the two chunks of model structures and change the eval from False to True.

Same process to sample posteriors of right censored model

```
## loading data from the working directory
data_censor <- readRDS("./censor_data.rds")

### Read in the model file using the jags.model function ##
censor_model_jags <- jags.model(textConnection(censormodel),
                        data=data_censor,n.chains = 4)

############ burn-in period ############
update(censor_model_jags,10000)

######## sample for posterior distribution ##########
censor_model_sample <- coda.samples(censor_model_jags,
                                    c("beta0","beta1",
                                    "lambda1","lambda2",
                                    "lambda3","inv_var"),
                                     100000,thin=20)
```

Section 3: spatial autocorrelation