STAT 578 - Fall 2019 - Assignment 3

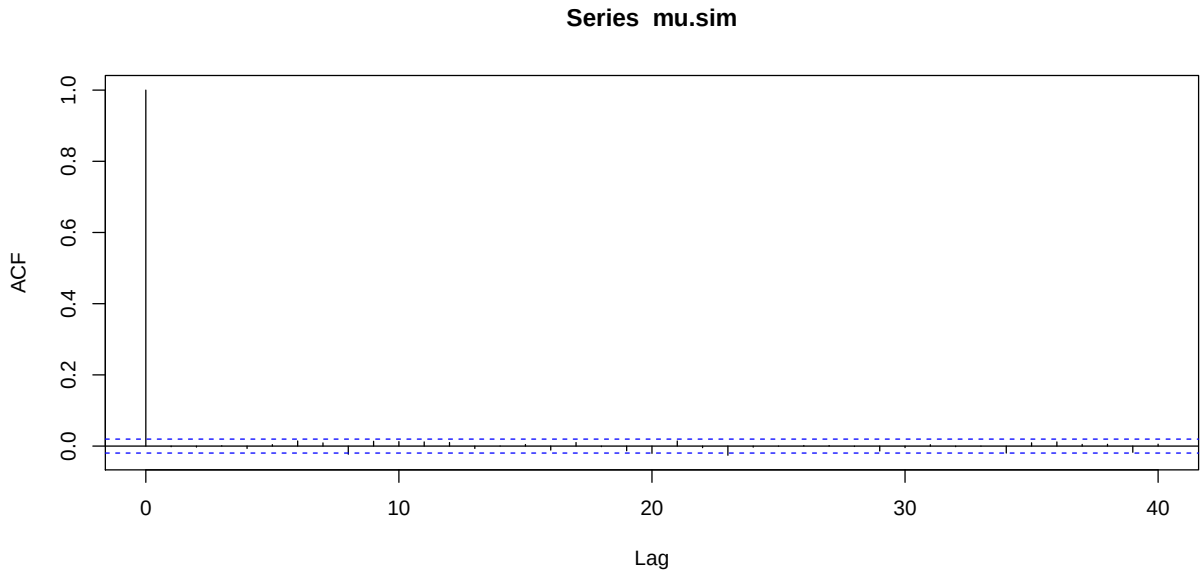
Frederick (Eric) Ellwanger - fre2

October 18, 2019

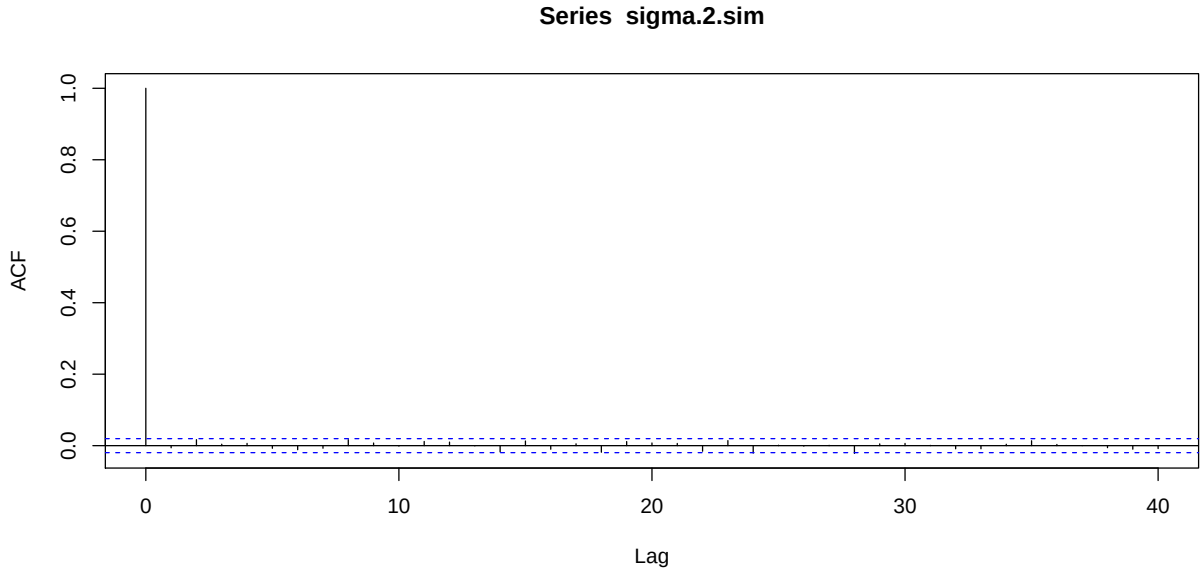
Exercise 1

(1)(a) Use the `FlintGibbs.R` script to simulate from the posterior for μ and σ^2. Then use R function acf to produce an autocorrelation plot for the successive μ variates and an autocorrelation plot for the successive σ^2 variates.

```
### Gibbs sampler for (partially conjugate) analysis of Flint data

n <- 271
ybar <- 1.40
s.2 <- 1.684


mu0 <- 1.10
tau.2.0 <- sigma.2.0 <- 1.17
nu0 <- 1


mun <- function(sigma.2){
  (mu0/tau.2.0 + n*ybar/sigma.2) / (1/tau.2.0 + n/sigma.2)
}

tau.2.n <- function(sigma.2){
  1 / (1/tau.2.0 + n/sigma.2)
}

sigma.2.n <- function(mu){
  (nu0*sigma.2.0 + (n-1)*s.2 + n*(mu-ybar)^2) / (nu0 + n)
}


n.sim <- 10000
mu.sim <- numeric(n.sim)
sigma.2.sim <- numeric(n.sim)

mu.sim[1] <- 1.4          # starting value
sigma.2.sim[1] <- 1.7     # starting value

for(t in 2:n.sim){
  mu.sim[t] <- rnorm(1, mun(sigma.2.sim[t-1]),
                     sqrt(tau.2.n(sigma.2.sim[t-1])))

  sigma.2.sim[t] <- 1 / rgamma(1, (nu0+n)/2,
                               (nu0+n)*sigma.2.n(mu.sim[t])/2)
}

#Plot the autocorrelation of succesive mu.sim variates
acf(mu.sim, type = c("correlation"))
```




```
#Plot the autocorrelation of succesive sigma.2.sim variates
acf(sigma.2.sim, type = c("correlation"))
```



(1)(b)(i) Experiment with different settings for the proposal variance rho (by uncommenting its line and changing the value). Find a value of rho that gives an overall (average) acceptance rate of about 0.35. What value of rho did you find?

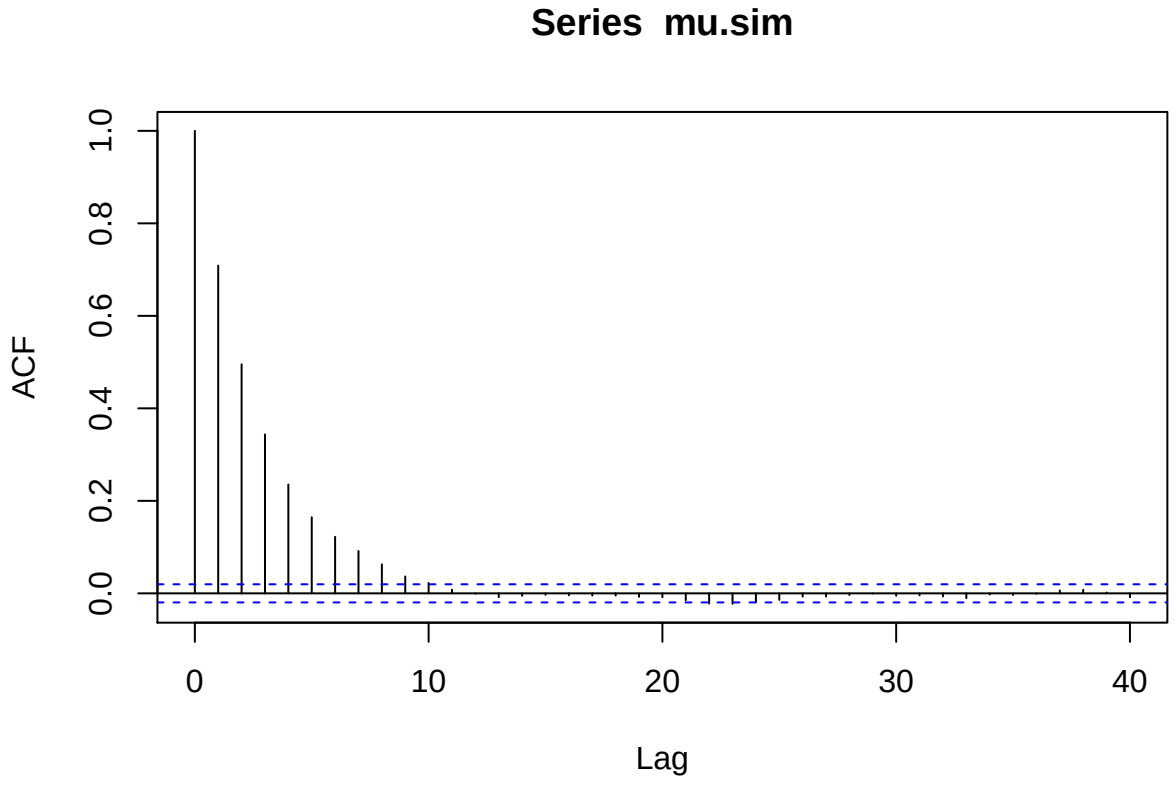
```
### Metropolis sampler for analysis of Flint data

n <- 271
ybar <- 1.40
s.2 <- 1.684
```

```r
mu0 <- 1.10
tau.2.0 <- sigma.2.0 <- 1.17
nu0 <- 1


dinvchisq <- function(x, df, scalesq){  # inverse chi-square density
  if(x>0)
    ((df/2)^(df/2) / gamma(df/2)) * sqrt(scalesq)^df * x^(-(df/2+1)) *
      exp(-df*scalesq/(2*x))
  else 0
}

likelihood <- function(mu, sigma.2){
  sigma.2^(-n/2) * exp(-(n-1)*s.2/(2*sigma.2)) *
    exp(-n*(mu-ybar)^2/(2*sigma.2))
}

ratio <- function(mu.prop, sigma.2.prop, mu.old, sigma.2.old){
  dnorm(mu.prop,mu0,sqrt(tau.2.0)) * dinvchisq(sigma.2.prop,nu0,sigma.2.0) *
    likelihood(mu.prop,sigma.2.prop)  /
  (dnorm(mu.old,mu0,sqrt(tau.2.0)) * dinvchisq(sigma.2.old,nu0,sigma.2.0) *
    likelihood(mu.old,sigma.2.old))
}


n.sim <- 10000
mu.sim <- numeric(n.sim)
sigma.2.sim <- numeric(n.sim)
accept.prob <- numeric(n.sim-1)

rho <- 0.0306

mu.sim[1] <- 1.4         # starting value
sigma.2.sim[1] <- 1.7    # starting value

for(t in 2:n.sim){
  mu.prop <- rnorm(1, mu.sim[t-1], sqrt(rho))
  sigma.2.prop <- rnorm(1, sigma.2.sim[t-1], sqrt(rho))

  accept.prob[t-1] <-
    min(ratio(mu.prop,sigma.2.prop,mu.sim[t-1],sigma.2.sim[t-1]), 1)
  if(runif(1) < accept.prob[t-1]){
    mu.sim[t] <- mu.prop
    sigma.2.sim[t] <- sigma.2.prop
  }else{
    mu.sim[t] <- mu.sim[t-1]
    sigma.2.sim[t] <- sigma.2.sim[t-1]
  }
}

#mean(accept.prob)
```

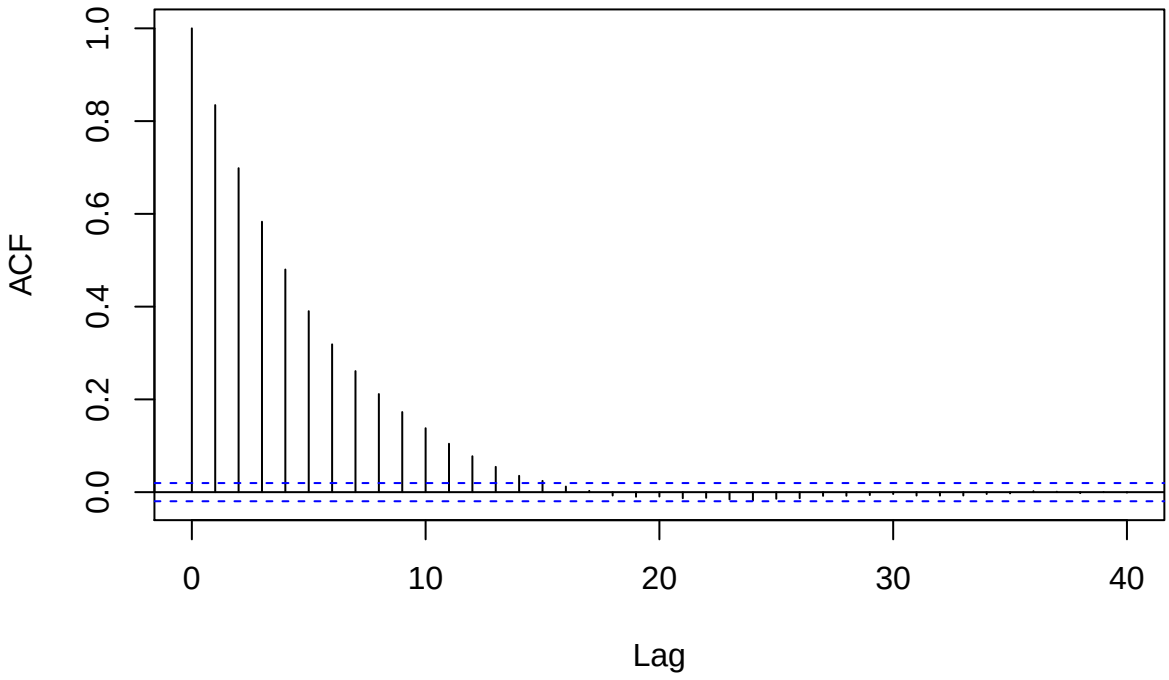
A value of rho $\approx$ 0.0306 gives an average acceptance rate of about 0.35

(1)(b)(ii) With the value of rho that you found, use the script to simulate from the posterior for μ and σ^2. Then use R function acf to produce an autocorrelation plot for the successive μ variates and an autocorrelation plot for the successive σ^2 variates.

```
#Plot the autocorrelation of succesive mu.sim variates
acf(mu.sim, type = c("correlation"))
```




```
#Plot the autocorrelation of succesive sigma.2.sim variates
acf(sigma.2.sim, type = c("correlation"))
```



(1)(c) Compare the autocorrelation plots from the previous two parts. Which method exhibited faster mixing: the Gibbs sampler or the Metropolis sampler?

Comparing the autocorrelation plots from the previous 2 parts, the Gibss Sampler seems to exhibit faster mixing. This is indicated by the faster decay of the autocorrelation to approximately 0 in the Gibbs sampler.

Exercise 2

(2)(a)(i) Use the model in polls20161.bug to create an initialization list (in R) supporting 4 chains, with a different initialization for each chain. Set initial values for mu to +/- 100 and values for tau to 100 or 0.01. Then use jags.model to create the JAGS model R object with these initializations. List all of the R code you used.

```
library(rjags)
d = read.table("polls2016.txt", header=TRUE)
d$sigma = d$ME/2
initial.vals <- list(list(mu = -100, tau = 0.01),
                     list(mu = -100, tau = 100),
                     list(mu = 100, tau = 0.01),
                     list(mu = 100, tau = 100))
m1 <- jags.model("polls20161.bug", d, initial.vals, n.chains=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
```

```
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 9
##    Total graph size: 42
##
## Initializing model
```

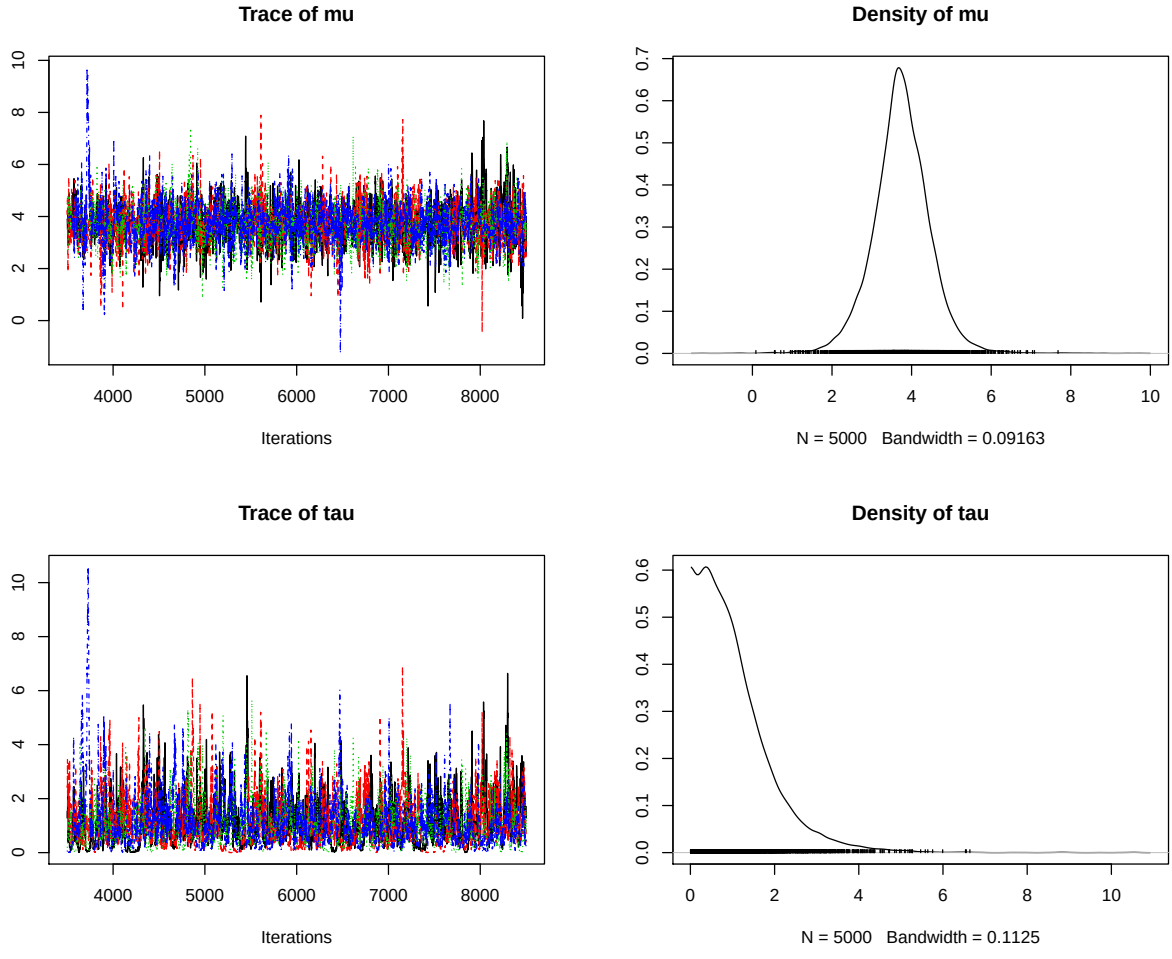
(2)(a)(ii) Perform a burn-in of 2500 iterations, then monitor the mu and tau nodes for 5000 iterations (for each chain). List all of the R code you used.

```
#Burn In
update(m1, 2500)

#Run 5000 iterations
x1 <- coda.samples(m1, c("mu","tau"), n.iter=5000)
```

(2)(a)(iii) For the iterations you monitored, produce trace plots of mu and tau. Do there appear to be any convergence problems? Display the plots and R code that produced them.

The below graphs do not indicate any problems with convergance. Both mu and tau show that all 4 chains are sampling from the same region over all of the samples. There doesn't seem to be any issues with the estimated posterior densities of mu and tau either.

```
plot(x1, smooth=FALSE)
```



(2)(a)(iv) For the iterations you monitored, compute Gelman-Rubin statistics (potential scale reduction factors) for mu and tau. Do there appear to be any convergence problems? Show your R code and its output.

As can be seen below, there does not appear to be any problems with convergence. The Gelman-Rubin statistics show that both mu and tau are near 1 and below the threshold of 1.1 suggested by our textbook and 1.05 suggested in lectures.

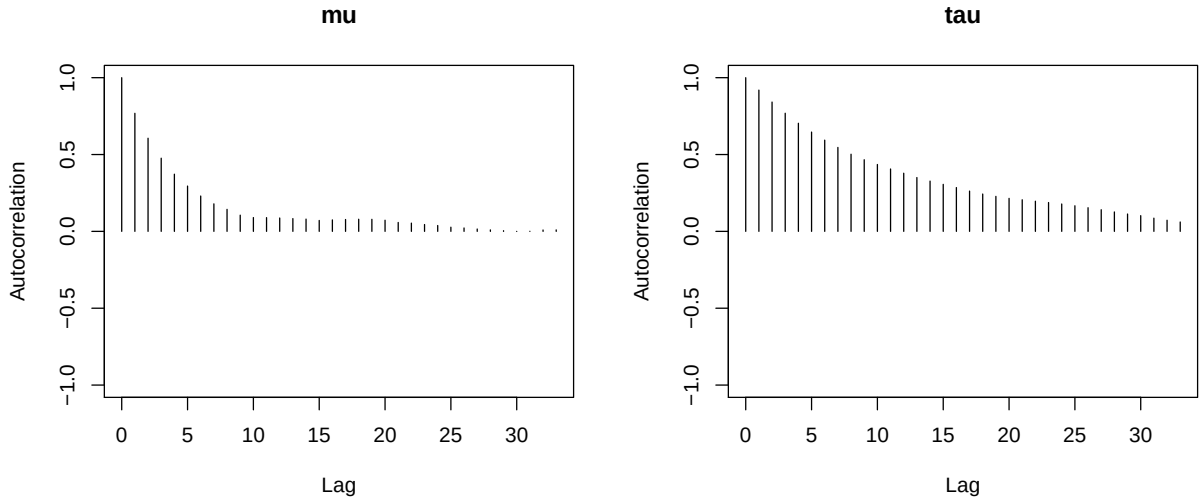
```
gelman.diag(x1, autoburnin=FALSE)
```

```
## Potential scale reduction factors:
##
##     Point est. Upper C.I.
## mu        1.00       1.00
## tau       1.01       1.02
##
## Multivariate psrf
##
## 1
```

(2)(a)(v) For the iterations you monitored, display autocorrelation plots for mu and tau for one of the chains. (Hint: For example, to reference the first chain of an mcmc.list object named x, use x[[1]].) Comment on the apparent speed of mixing.

As can be seen in the below graphs mu appears to have quicker mixing than tau, but tau appears to have adequate mixing as the lag approaches 30

```
autocorr.plot(x1[[1]])
```



(2)(a)(vi) For the iterations you monitored, compute effective sample sizes (over all chains) for mu and tau. Would they be considered adequate? Show your R code and its output.

As seen below the effective sample sizes for both mu and tau are well above an effective sample size of 400 which is cosidered the minimum necessary to be considered adequate.

```
effectiveSize(x1)
```

```
##        mu       tau
## 2136.0827  891.9829
```

(2)(b)(i) Display all of the code for your new JAGS model.

```
model {

  for (j in 1:length(y)) {
    y[j] ~ dnorm(theta[j], 1/sigma[j]^2)
    theta[j] ~ dnorm(mu, 1/tau^2)
  }

  mu ~ dunif(-1000,1000)
  logtau ~ dunif(-100,100)

  tau <- exp(logtau)
}
```

(2)(b)(ii) Create an initialization list (in R) supporting 4 chains, with a different initialization for each chain. Set initial values for mu to +/-100 and values for logtau to log 100 or log 0.01. Then use jags.model to create the JAGS model R object with these initializations. List all of the R code you used.

```
initial.vals <- list(list(mu = -100, logtau =log(0.01)),
                     list(mu = -100, logtau = log(100)),
                     list(mu = 100, logtau = log(0.01)),
                     list(mu = 100, logtau = log(100)))
m2 <- jags.model("polls20161chg.bug", d, initial.vals, n.chains=4)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7
##    Unobserved stochastic nodes: 9
##    Total graph size: 44
##
## Initializing model
```

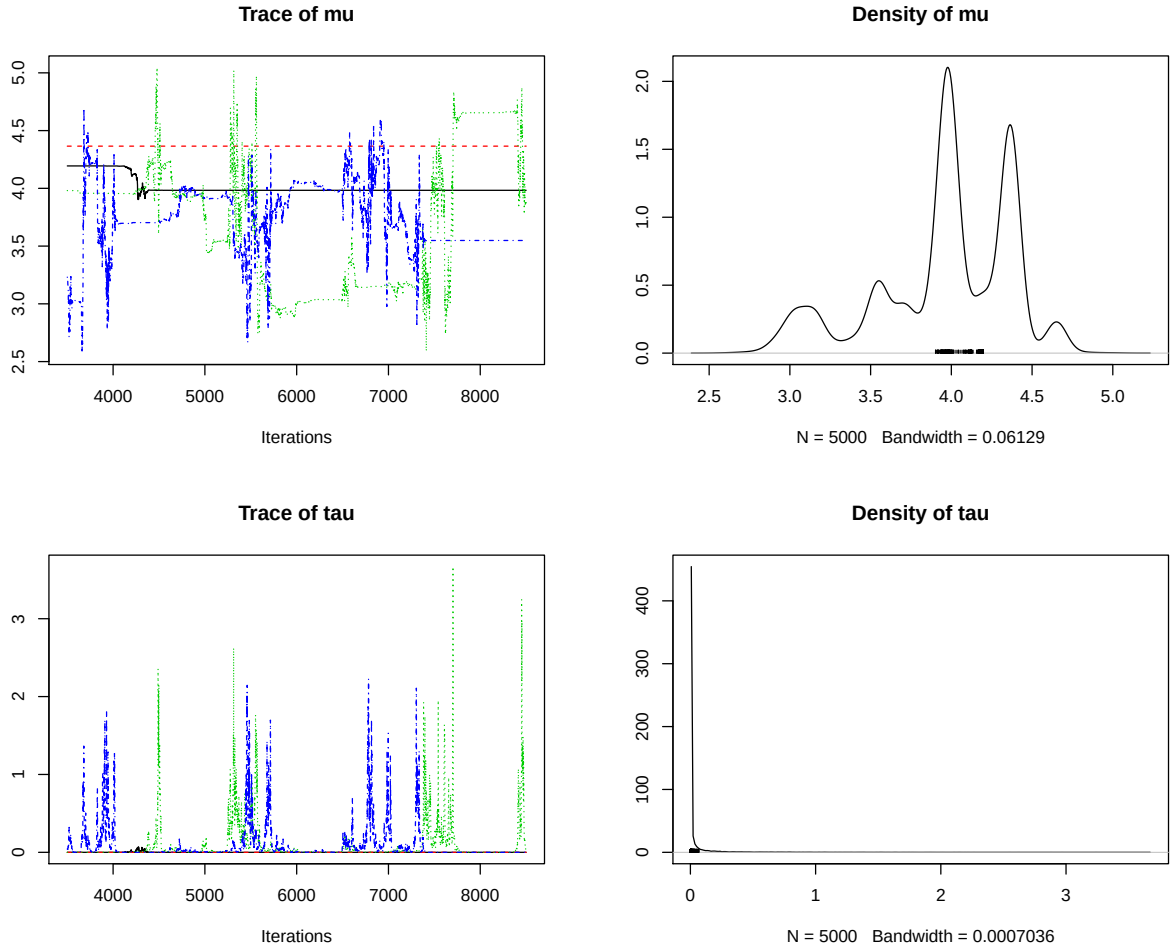
(2)(b)(iii) Perform a burn-in of 2500 iterations, then monitor the mu and tau nodes for 5000 iterations (for each chain). List all of the R code you used.

```
#Burn In
update(m2, 2500)

#Run 5000 iterations
x2 <- coda.samples(m2, c("mu","tau"), n.iter=5000)
```

(2)(b)(iv) For the iterations you monitored, produce trace plots of mu and tau. Do there appear to be any convergence problems? Display the plots and R code that produced them.

As can be seen in the graphs below there seem to be several issues with convergance. The trace plots seem to show that the different chains are sampling from different areas in the parameter space. The estimated posterior density of mu also seems to show a multi-modal distribution which can sometimes suggest issues with convergence.The estimated posterior density of tau is concentrated at values very close to 0, which could also indicate a problem.

```
plot(x2, smooth=FALSE)
```



(2)(b)(v) For the iterations you monitored, compute Gelman-Rubin statistics (potential scale reduction factors) for mu and tau. Do there appear to be any convergence problems? Show your R code and its output.

The Gelman-Rubin statistics indicate there may be a problem with convergence. Both mu and tau have large values, well above 1.1

```
gelman.diag(x2, autoburnin=FALSE)
```

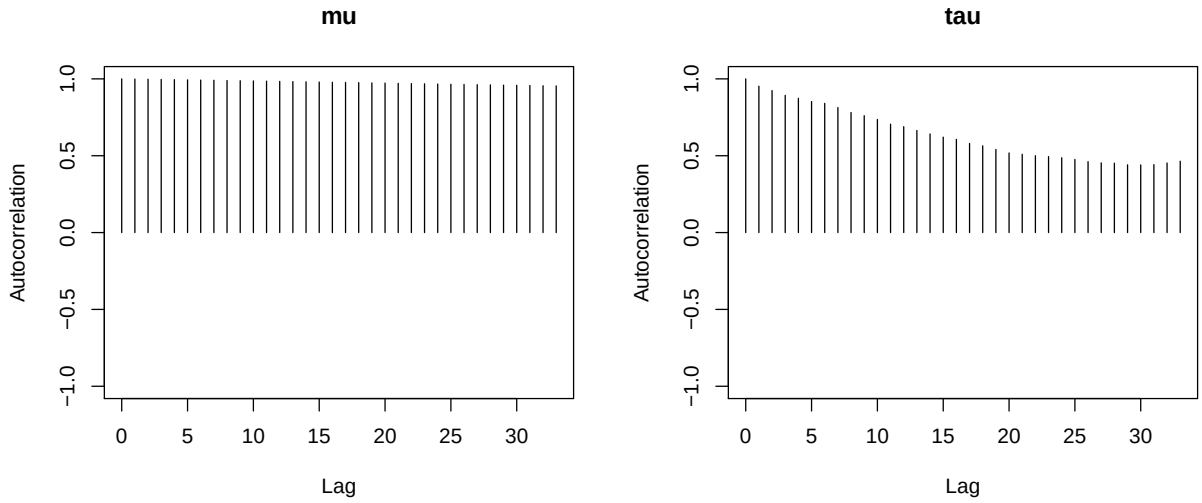
```
## Potential scale reduction factors:
##
##     Point est. Upper C.I.
## mu        1.60       4.03
## tau       1.18       1.48
##
## Multivariate psrf
##
## 1.56
```

(2)(b)(vi) For the iterations you monitored, display autocorrelation plots for mu and tau for one of the chains. Comment on the apparent speed of mixing.

The below plots show that both mu and tau have what appears to be very slow mixing

compared to the other model.

```
autocorr.plot(x2[[1]])
```



(2)(b)(vii) Suggest an explanation for the presence of any problems you noted. (Hint: What would happen if you used an improper prior on log τ ?)

As we have learned, using an improper prior can lead to an improper posterior. The uniform distribution on log(tau) is just such an improper prior. It would seem from the posterior data that this has led to an improper posterior. We see the variance, tau, essentially zero which has made the mu data essentially the same. This can be seen in the autocorrelation plot of mu and the estimated posterior plot of tau, where mu has extremely high autocorrelation and tau has an estimated posterior density that has most (if not all) of its density extremely close to 0.

Apeendix - Data

```
# 2016 U.S. presidential election race between H. Clinton and D. Trump
# National poll results for two-way race, conducted November 3 and later
# y = percentage of Clinton lead, with margin of error
# From: https://en.wikipedia.org/wiki/Nationwide_opinion_polling_for_the_United_States_presidential_ele

poll        y   ME
YouGov      4   1.7
Bloomberg   3   3.5
ABCWaPo     3   2.5
Fox         4   2.5
IBD         1   3.1
Monmouth    6   3.6
NBCWSJ      5   2.73
```