Final Project - EXST 7151

Dina Dinh

2023-11-21

Libraries Needed

```
library(ggplot2)
library(dplyr)
library(MCMCpack)
library(msm)
library(psych)
```

Loading Data and Data Cleaning

```
df1 = read.csv("C:/Users/ddinh4/Documents/EXST 7151/Final Project/Data/data_date.csv")
unique_data <- df1 %>% distinct(Country, .keep_all = TRUE)
aqi_2 <- unique_data[,c("Status","AQI_Value")]
aqi <- aqi_2[,2]
aqi <-  aqi[aqi <= 500]
aqi_df <- data.frame(aqi)
```

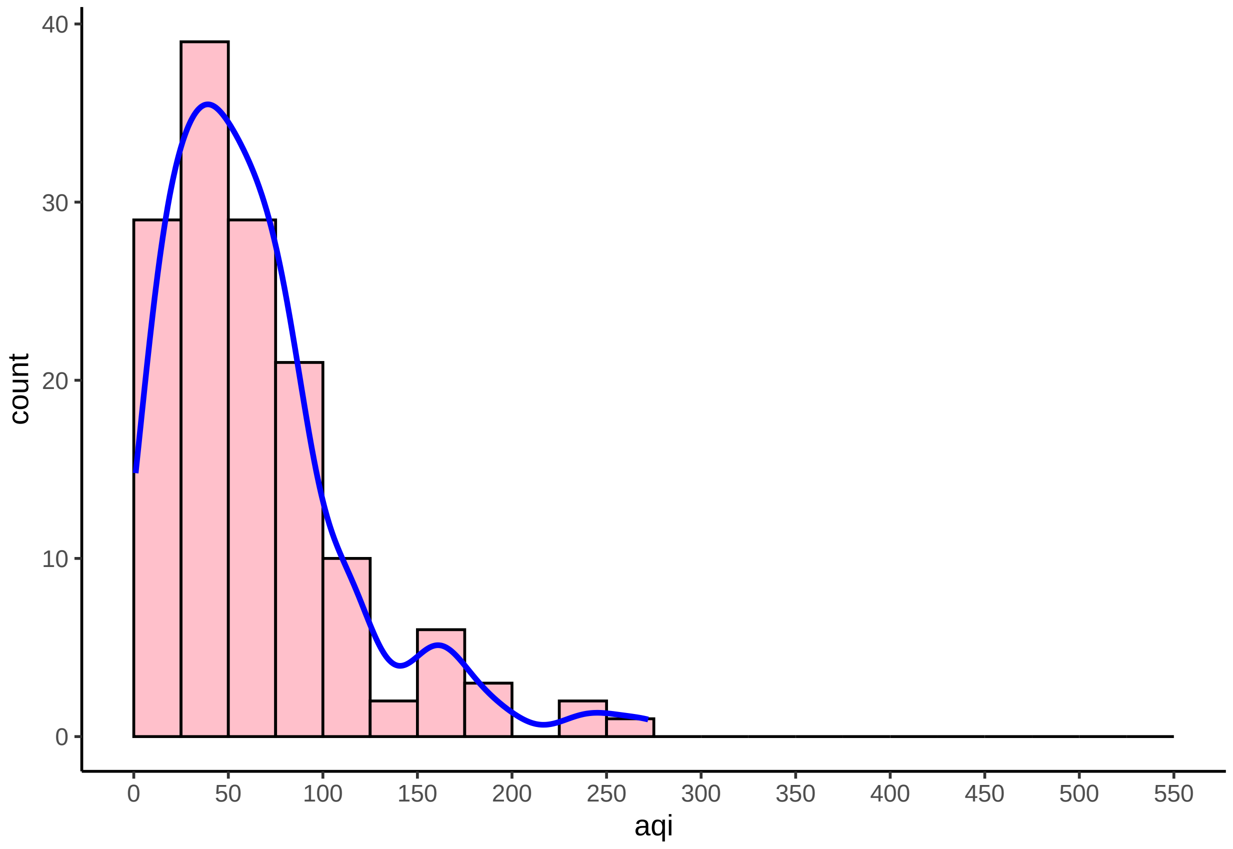
Summary Statistics of the Data

```
sum.aqi = describe(aqi)
knitr::kable(sum.aqi)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
X1	1	142	64.33803	50.13611	55	56.75439	38.5476	1	272	271	1.583802	3.046024	4.207329

Histogram of AQI

```r
hist_aqi <- ggplot(aqi_df, aes(x = aqi)) + geom_histogram(aes(y = ..count..),
breaks = seq(0, 550, by = 25), colour = "black",
fill = "pink") +
geom_density(aes(y = ..count.. * 25), color = "blue", linewidth = 1) +
scale_x_continuous(breaks = seq(0,
550, by = 50)) + theme_classic()

hist_aqi
```



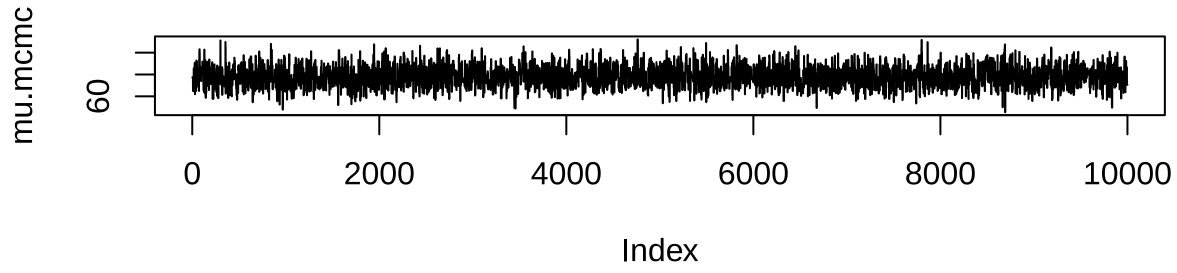
Metropolis Within Gibbs Sampler

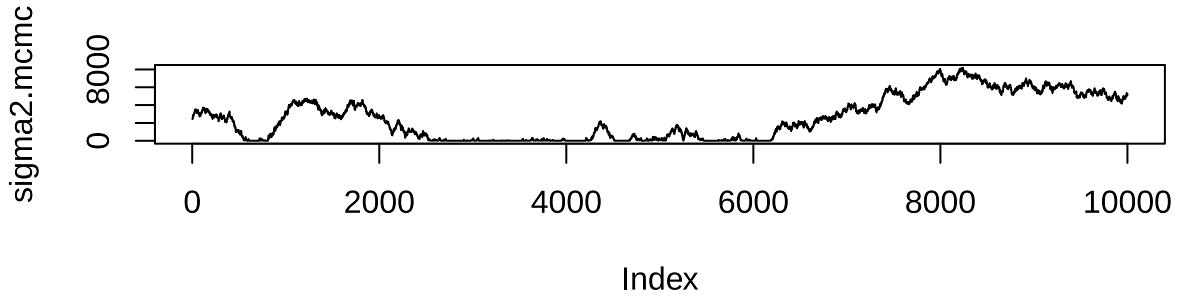
Using Gamma for Distribution of Data

Using the Gamma distribution for the data reparametrized in terms of μ and σ^2 with unknown μ and σ^2. Assuming μ and σ^2 independent, the joint prior distribution is $p(\mu, \sigma^2) = p(\mu)p(\sigma^2)$. Informative prior for $\mu \sim N(70, 70^2/5)$ and vague prior for $\sigma^2 \sim IG(0.1, 0.1)$.

```r
a = 5
log.posterior.mu <- function(mu,sigma2,mu0,tau2,y)
  sum(dgamma(y, shape = (mu^2)/((mu^2)/a), rate = mu/((mu^2)/a), log=TRUE)) + dnorm(mu,mu0, sqrt(tau2),

log.posterior.sigma2 <- function(sigma2,mu,alpha,beta,y) {
  if (sigma2<0) return(-Inf) else
```

```
    return(sum(dgamma(y, shape = (mu^2)/((mu^2)/a), rate = mu/((mu^2)/a), log=TRUE)) + log(dinvgamma(si
}

y <- aqi #data

## prior hyperparameters
mu0 <- 70
tau2 <- (70^2)/5
alpha <- 0.1
beta <- 0.1


sigma2 <- var(aqi) # initial value for sigma2
mu<- mean(aqi) # initial value for mu
n.sim<-10000 ; mu.mcmc<-NULL ; sigma2.mcmc <- NULL; accept.mu <- 0;accept.sigma2 <- 0
delta1 <- 40 ; ## normal proposal var for mu
delta2 <- 6500 ## normal proposal var for sigma2

for ( ite in 1 : n.sim) {

  mu.star<-rnorm(1, mu, sqrt(delta1))

  log.r <- log.posterior.mu(mu.star,sigma2,mu0,tau2,y) - log.posterior.mu(mu,sigma2,mu0,tau2,y)

  if (log(runif(1))< log.r) {
    mu <- mu.star
    accept.mu <- accept.mu+1
  }
  mu.mcmc<-c(mu.mcmc, mu)

  sigma2.star<-rnorm(1, sigma2, sqrt(delta2) )

  log.r <- log.posterior.sigma2(sigma2.star,mu,alpha,beta,y) - log.posterior.sigma2(sigma2,mu,alpha,beta

  if (log(runif(1))< log.r) {
    sigma2 <- sigma2.star
    accept.sigma2 <- accept.sigma2+1
  }
  sigma2.mcmc<-c(sigma2.mcmc, sigma2)


}

par(mfrow=c(2,1))
plot(mu.mcmc,type='l')
plot(sigma2.mcmc,type='l')
```





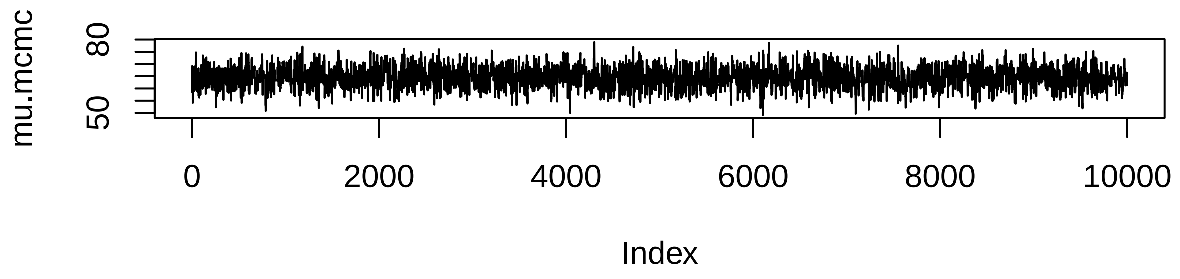
```
## [1] "Acceptance rate of mu = 0.4133"

## [1] "Acceptance rate of sigma2 = 0.7498"

## [1] "Mean of mu = 64.5344540497742"

## [1] "Mean of sigma2 = 2641.89449940602"

## [1] "Variance of mu = 5.54714888048173"

## [1] "Variance of sigma2 = 6364516.13328974"
```

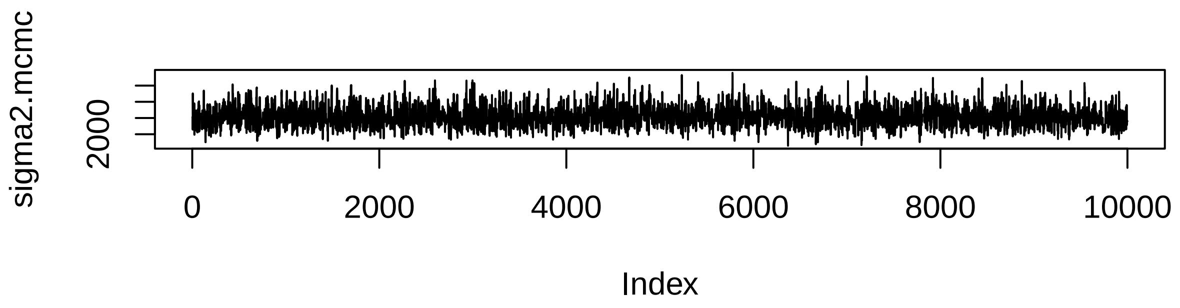
We can see that the Markov chain doesn't converge for sigma at all and is very unstable. Notice the tuning parameter of the normal proposal for σ^2 is very large.

Using Normal for Distribution of Data

Keeping the priors for μ and σ^2 the same as before.

```
log.posterior.mu <- function(mu,sigma2,mu0,tau2,y)
  sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + dnorm(mu,mu0, sqrt(tau2),log=TRUE)

log.posterior.sigma2 <- function(sigma2,mu,alpha,beta,y) {
  if (sigma2<0) return(-Inf) else
    return(sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + log(dinvgamma(sigma2,alpha,beta)))
}

y <- aqi #data

## prior hyperparameters
mu0 <- 70
tau2 <- (70^2)/5
alpha <- 0.1
beta <- 0.1


sigma2 <- var(aqi) # initial value for sigma2
mu<- mean(aqi) # initial value for mu
n.sim<-10000 ; mu.mcmc<-NULL ; sigma2.mcmc <- NULL; accept.mu <- 0;accept.sigma2 <- 0
delta1 <- 150 ; ## normal proposal var for mu
delta2 <- 505000 ## normal proposal var for sigma2

for ( ite in 1 : n.sim) {

  mu.star<-rnorm(1, mu, sqrt(delta1))

  log.r <- log.posterior.mu(mu.star,sigma2,mu0,tau2,y) - log.posterior.mu(mu,sigma2,mu0,tau2,y)

  if (log(runif(1))< log.r) {
    mu <- mu.star
    accept.mu <- accept.mu+1
  }
  mu.mcmc<-c(mu.mcmc, mu)

  sigma2.star<-rnorm(1, sigma2, sqrt(delta2) )

  log.r <- log.posterior.sigma2(sigma2.star,mu,alpha,beta,y) - log.posterior.sigma2(sigma2,mu,alpha,bet

  if (log(runif(1))< log.r) {
    sigma2 <- sigma2.star
    accept.sigma2 <- accept.sigma2+1
  }
  sigma2.mcmc<-c(sigma2.mcmc, sigma2)


}

par(mfrow=c(2,1))
plot(mu.mcmc,type='l')
plot(sigma2.mcmc,type='l')
```





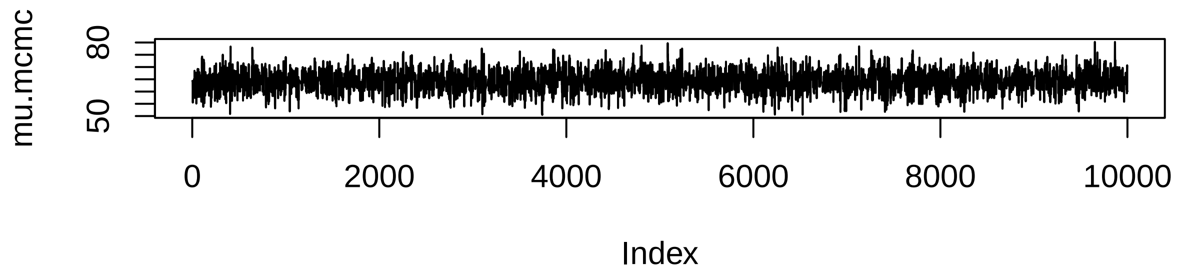
```
## [1] "Acceptance rate of mu = 0.3759"

## [1] "Acceptance rate of sigma2 = 0.4406"

## [1] "Mean of mu = 64.5012887160956"

## [1] "Mean of sigma2 = 2540.43693822367"

## [1] "Variance of mu = 18.2149399399099"

## [1] "Variance of sigma2 = 96296.6981692037"
```

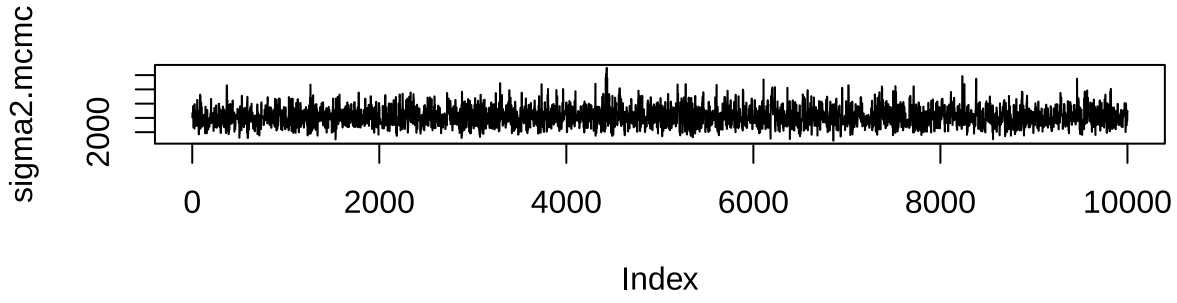
Using the normal distribution for the data, both of the parameters converged; however, the tuning parameter for both proposals are quite large especially for σ^2.

Using the Normal for Distribution of the Data and Gamma for Prior of μ

Keeping prior of σ^2 the same as before.

```
a=5
log.posterior.mu <- function(mu,sigma2,mu0,tau2,y)
  sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + dgamma(mu,a, a/mu0,log=TRUE)

log.posterior.sigma2 <- function(sigma2,mu,alpha,beta,y) {
  if (sigma2<0) return(-Inf) else
    return(sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + log(dinvgamma(sigma2,alpha,beta)))
}

y <- aqi #data

## prior hyperparameters
mu0 <- 70
tau2 <- (70^2)/5
alpha <- 0.1
beta <- 0.1


sigma2 <- var(aqi) # initial value for sigma2
mu<- mean(aqi) # initial value for mu
n.sim<-10000 ; mu.mcmc<-NULL ; sigma2.mcmc <- NULL; accept.mu <- 0;accept.sigma2 <- 0
delta1 <- 130 ; ## normal proposal var for mu
delta2 <- 600000 ## normal proposal var for sigma2

for ( ite in 1 : n.sim) {

  mu.star<-rnorm(1, mu, sqrt(delta1))

  log.r <- log.posterior.mu(mu.star,sigma2,mu0,tau2,y) - log.posterior.mu(mu,sigma2,mu0,tau2,y)

  if (log(runif(1))< log.r) {
    mu <- mu.star
    accept.mu <- accept.mu+1
  }
  mu.mcmc<-c(mu.mcmc, mu)

  sigma2.star<-rnorm(1, sigma2, sqrt(delta2) )

  log.r <- log.posterior.sigma2(sigma2.star,mu,alpha,beta,y) - log.posterior.sigma2(sigma2,mu,alpha,beta

  if (log(runif(1))< log.r) {
    sigma2 <- sigma2.star
    accept.sigma2 <- accept.sigma2+1
  }
  sigma2.mcmc<-c(sigma2.mcmc, sigma2)


}

par(mfrow=c(2,1))
plot(mu.mcmc,type='l')
plot(sigma2.mcmc,type='l')
```





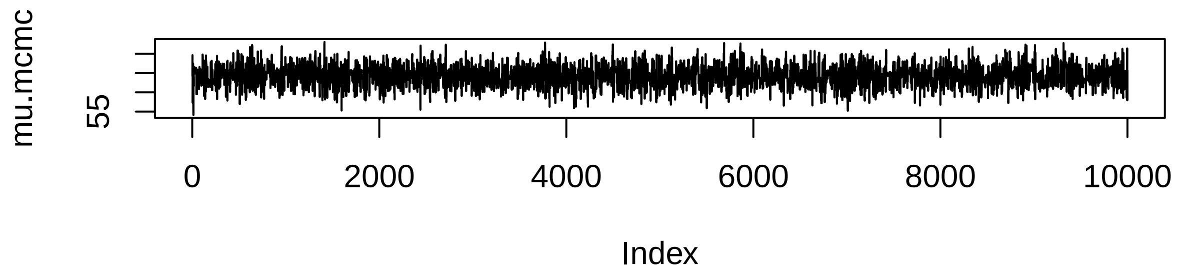
```
## [1] "Acceptance rate of mu = 0.4098"

## [1] "Acceptance rate of sigma2 = 0.415"

## [1] "Mean of mu = 64.186177703055"

## [1] "Mean of sigma2 = 2551.34139939622"

## [1] "Variance of mu = 18.5611228651648"

## [1] "Variance of sigma2 = 101552.408763733"
```

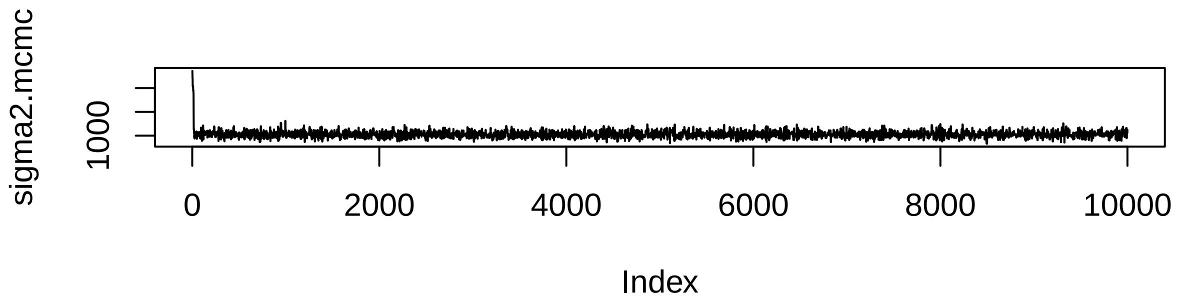
Both chains for the parameters converges. The mean of μ and the mean of σ^2 is very similar to when using the normal prior distribution for μ.Both tuning parameters are still very large.

Using Normal for Distrubtion of Data with Gamma Prior for μ and Gamma Prior for σ^2

```
a=5
log.posterior.mu <- function(mu,sigma2,mu0,tau2,y)
  sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + dgamma(mu,a, a/mu0,log=TRUE)
```

```r
log.posterior.sigma2 <- function(sigma2,mu,alpha,beta,y) {
  if (sigma2<0) return(-Inf) else
    return(sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + log(dgamma(sigma2,alpha,beta)))
}

y <- aqi #data

## prior hyperparameters
mu0 <- 70
tau2 <- (70^2)/5
alpha <- 0.1
beta <- 0.1


sigma2 <- var(aqi) # initial value for sigma2
mu<- mean(aqi) # initial value for mu
n.sim<-10000 ; mu.mcmc<-NULL ; sigma2.mcmc <- NULL; accept.mu <- 0;accept.sigma2 <- 0
delta1 <- 60 ; ## normal proposal var for mu
delta2 <- 30000 ## normal proposal var for sigma2

for ( ite in 1 : n.sim) {

  mu.star<-rnorm(1, mu, sqrt(delta1))

  log.r <- log.posterior.mu(mu.star,sigma2,mu0,tau2,y) - log.posterior.mu(mu,sigma2,mu0,tau2,y)

  if (log(runif(1))< log.r) {
    mu <- mu.star
    accept.mu <- accept.mu+1
  }
  mu.mcmc<-c(mu.mcmc, mu)

  sigma2.star<-rnorm(1, sigma2, sqrt(delta2) )

  log.r <- log.posterior.sigma2(sigma2.star,mu,alpha,beta,y) - log.posterior.sigma2(sigma2,mu,alpha,beta

  if (log(runif(1))< log.r) {
    sigma2 <- sigma2.star
    accept.sigma2 <- accept.sigma2+1
  }
  sigma2.mcmc<-c(sigma2.mcmc, sigma2)


}

par(mfrow=c(2,1))
plot(mu.mcmc,type='l')
plot(sigma2.mcmc,type='l')
```





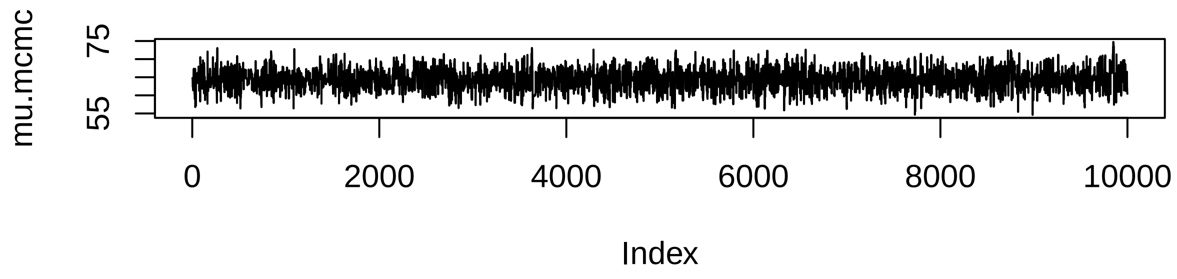
```
## [1] "Acceptance rate of mu = 0.3877"

## [1] "Acceptance rate of sigma2 = 0.3842"

## [1] "Mean of mu = 64.1583196421503"

## [1] "Mean of sigma2 = 1026.83145910419"

## [1] "Variance of mu = 7.47089276483445"

## [1] "Variance of sigma2 = 3843.59043189507"
```

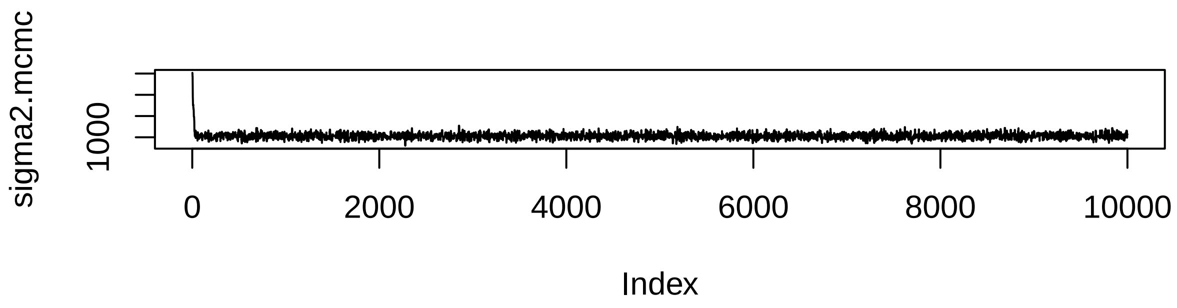
Both chains for the parameters converges and the tuning parameters are as large as the previous models. The posterior estimates are also a bit more reasonable and aren't as large.

Metropolis-Hastings Within Gibbs Sampler

Using the Normal distribution for the data and the Gamma for both the priors of μ and σ^2, the truncated normal from 0 to 500 was used as the proposal distribution for μ and truncated normal from 0 to ∞ for σ^2.

```r
a=5
log.posterior.mu <- function(mu,sigma2,mu0,tau2,y)
  sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + dgamma(mu,a, a/mu0,log=TRUE)

log.posterior.sigma2 <- function(sigma2,mu,alpha,beta,y) {
  if (sigma2<0) return(-Inf) else
    return(sum(dnorm(y, mu, sqrt(sigma2), log=TRUE)) + log(dgamma(sigma2,alpha,beta)))
}

y <- aqi #data

## prior hyperparameters
mu0 <- 70
tau2 <- (70^2)/5
alpha <- 0.1
beta <- 0.1


sigma2 <- var(aqi) # initial value for sigma2
mu<- mean(aqi) # initial value for mu
n.sim<-10000 ; mu.mcmc<-NULL ; sigma2.mcmc <- NULL; accept.mu <- 0;accept.sigma2 <- 0
delta1 <- 60 ; ## normal proposal var for mu
delta2 <- 30000 ## normal proposal var for sigma2

for ( ite in 1 : n.sim) {

  mu.star<-rtnorm(1, mu, sqrt(delta1), 0, 500)

  log.r <- log.posterior.mu(mu.star,sigma2,mu0,tau2,y) + dtnorm(mu, mu.star, sqrt(delta1), 0,500, log =

  if (log(runif(1))< log.r) {
    mu <- mu.star
    accept.mu <- accept.mu+1
  }
  mu.mcmc<-c(mu.mcmc, mu)

  sigma2.star<-rtnorm(1, sigma2, sqrt(delta2),0, Inf)

  log.r <- log.posterior.sigma2(sigma2.star,mu,alpha,beta,y) + dtnorm(sigma2,sigma2.star,sqrt(delta2),0

  if (log(runif(1))< log.r) {
    sigma2 <- sigma2.star
    accept.sigma2 <- accept.sigma2+1
  }
  sigma2.mcmc<-c(sigma2.mcmc, sigma2)


}

par(mfrow=c(2,1))
plot(mu.mcmc,type='l')
plot(sigma2.mcmc,type='l')
```





```
## [1] "Acceptance rate of mu = 0.3836"

## [1] "Acceptance rate of sigma2 = 0.3924"

## [1] "Mean of mu = 64.268776739047"

## [1] "Mean of sigma2 = 1027.91945049566"

## [1] "Variance of mu = 7.91361836718265"

## [1] "Variance of sigma2 = 3735.55966288623"
```

The results are similar to the Metropolis within Gibbs using the normal proposal.

Bayesian Hypothesis Testing

Assuming Y_i follows a normal distribution with unknown μ and σ^2,

H_0: $\mu < 100$

H_1: $\mu \geq 100$

Using Normal for Distribution of Data with Gamma Prior for μ and Gamma Prior for σ^2 and the normal proposal distribution.

Computing Prior Odds

```
prior.prob.h0 = pgamma(100, a, a/mu0)
prior.prob.h1 = 1 - prior.prob.h0
prior.odds = prior.prob.h0/prior.prob.h1
```

Computing Posterior Odds

Using the example in notes to check if I'm integrating and calculating posterior odds correctly.

```
p = function(y){
  # Replace this with your actual pdf function
  # For example, a normal distribution with mean 0 and standard deviation 1:
  return(dnorm(y, mean = 175.79, sd = 0.93))
}

# Define the range over which you want to calculate the cdf
lower_limit <- -Inf  # replace with your lower limit
upper_limit <- 175     # replace with your upper limit

# Use integrate to calculate the cdf
cdf_result <- integrate(p, lower_limit, upper_limit)$value
cdf_result/(1-cdf_result)
```

```
## [1] 0.2465906
```

```
cdf = pnorm(175, mean = 175.79, sd = 0.93)

print(paste("Comparing the CDF using pnorm and integrating:", cdf,cdf_result))
```

```
## [1] "Comparing the CDF using pnorm and integrating: 0.197812031510041 0.197812031510039"
```

Trying to integrate and calculate posterior odds for this dataset.

```
mu0 = 70
tau2 = (70^2)/5
mu = mean(mu.mcmc[2000:10000])
sigma2 = var(sigma2.mcmc[2000:10000])

post.density = function(x){
   return(sum(dnorm(x, mu, sqrt(sigma2), log=TRUE)) + dgamma(x,a, a/mu0,log=TRUE))
}

post.prob.h0 = integrate(post.density, 0, 100)$value
post.prob.h1 = 1-post.prob.h0
```

I noticed that the density of the posterior is not normalized so the integral does not equal 1.

Here I normalized the density.

```r
# Define your unnormalized post.density function
post.density_unnormalized <- function(x) {
  return(sum(dnorm(x, mu, sqrt(sigma2), log = TRUE)) + dgamma(x, a, a/mu0, log = TRUE))
}

# Integrate the unnormalized density over the specified range
integral_unnormalized <- integrate(post.density_unnormalized, 0, 100)$value

# Define the normalized post.density function
post.density_normalized <- function(x) {
  return(post.density_unnormalized(x) / integral_unnormalized)
}

# Integrate the normalized density over the entire range to confirm it integrates to 1
post.prob.h0_normalized <- integrate(post.density_normalized, 0, 100)$value
post.prob.h1_normalized = 1-post.prob.h0_normalized

post.odds = post.prob.h0_normalized/post.prob.h1_normalized
```

Computing Bayes Factor

```r
BF = post.odds/prior.odds
print(paste("BF =", BF))
```

```
## [1] "BF = -172017329243323"
```

Since the BF is very large, the data strongly supports the null hypothesis that μ is less than 100 AQI with P(μ>100|y) = 0.999.