Michael Leibert

Math 640

Exam 2, Computing

1. The REIGN dataset calculates the monthly risk of a coup occurring for each country in the world. We are interested in modeling the log transformed risk of coup from the month of December in the year 1980. The data is roughly symmetric and unimodal, however it is unclear that it is normal. One model we can use to help determine if it is normal is the generalized normal which has as its pdf the following:

$$p(z_i) = \frac{\beta}{2\Gamma\left(\frac{1}{\beta}\right)} \exp\left(-|z_i|^{\beta}\right)$$

If $\beta \approx 2$, the model suggests a Gaussian likelihood and if $\beta \approx 1$, it suggests the Laplacian. Thus we can use the generalized normal model as one a way to determine which of these two likelihoods is a better fit for the data.

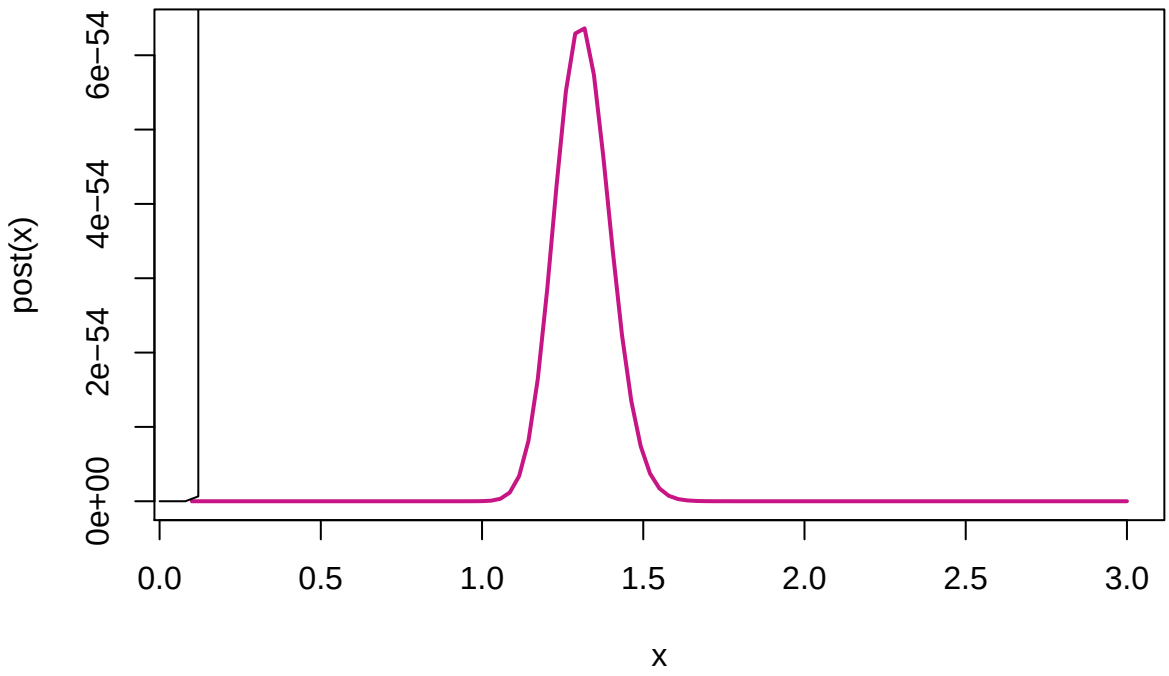
Derive the likelihood and posterior assuming a flat prior for β, $\pi(\beta) \propto 1$.

$$\begin{aligned}\mathcal{L}(z_i|\beta) &\propto \prod_{i=1}^{n} \frac{\beta}{2\Gamma\left(\frac{1}{\beta}\right)} \exp\left(-|z_i|^{\beta}\right) \\ &\propto \left[\frac{\beta}{\Gamma\left(\frac{1}{\beta}\right)}\right]^n \exp\left(-\sum_{i=1}^{n}|z_i|^{\beta}\right)\end{aligned}$$

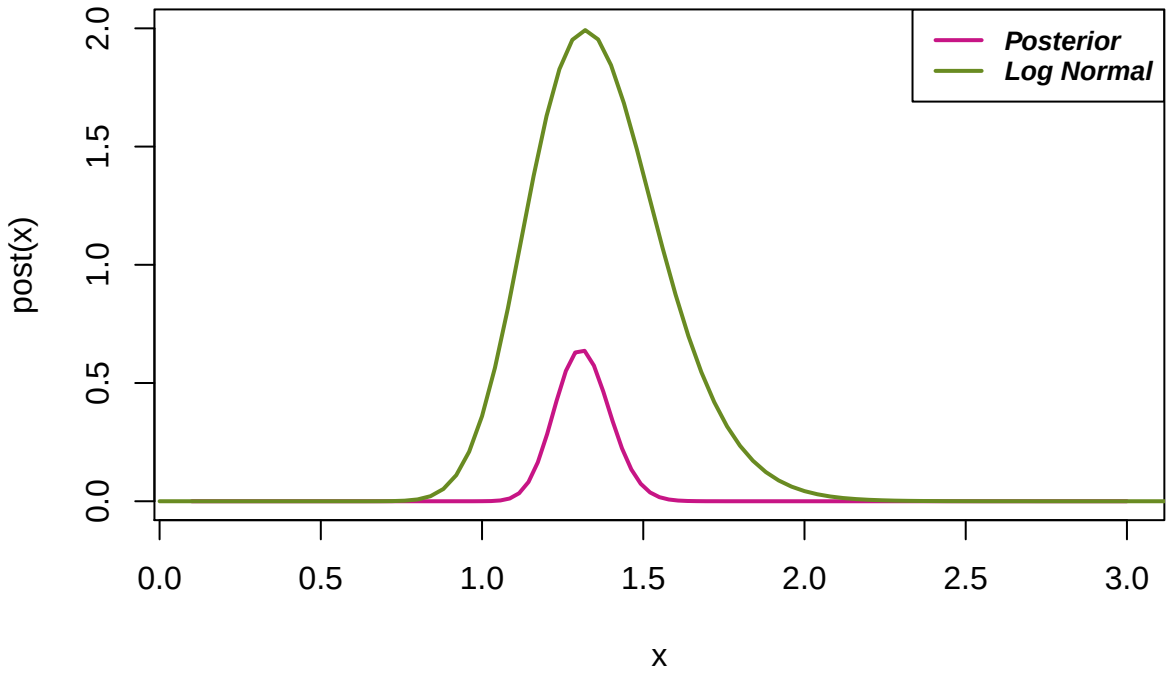
$$p(\beta|z_i) \propto \left[\frac{\beta}{\Gamma\left(\frac{1}{\beta}\right)}\right]^n \exp\left(-\sum_{i=1}^{n}|z_i|^{\beta}\right)$$

Then implement a sampler of your choosing to generate posterior samples for β using the data making sure to standardize it first.

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")

reign <- read.table("coup1280.txt", stringsAsFactors = F , header =T)
reign$z <- scale(reign[,2])

POST <- function( B , w ){  (B/gamma(1/B))^(length(w)) * exp(- sum(abs( w )^B) )  }
Post <- function(B){ POST(B,reign$z)}
post <- function(B){ return(sapply(B, Post )) }

a =.3; b =  .15

curve(post  , from = 0.1, to = 3  , col = "mediumvioletred",lwd = 2 )
curve(   dlnorm(x, .3, .15 ), from = 0, to = 4 , add = T)
```

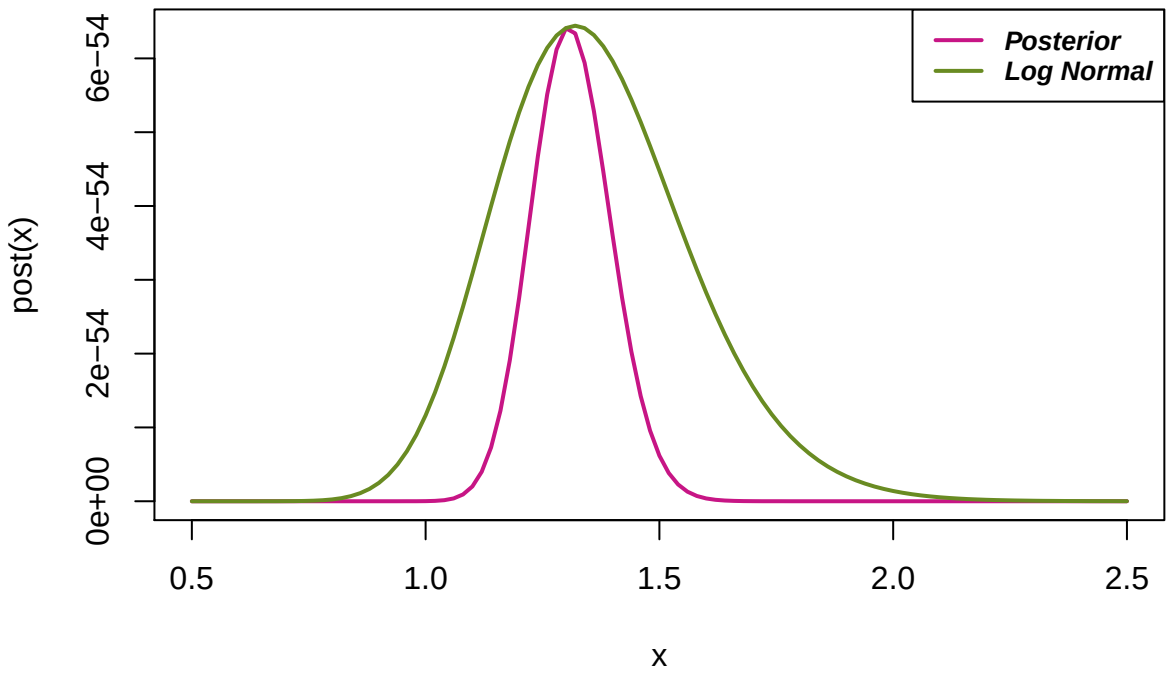



```
# Multiply Posterior by a constant to get a better look
POST <- function( B , w ){ 10e52* (B/gamma(1/B))^(length(w)) * exp(- sum(abs( w )^B) )  }
Post <- function(B){ POST(B,reign$z)}
post <- function(B){ return(sapply(B, Post )) }
curve(post   , from = 0.1, to = 3, ylim = c(0,2), col = "mediumvioletred",lwd = 2  )
curve(   dlnorm(x, .3, .15 ), from = 0, to = 4 , add = T, col = "olivedrab4" ,lwd = 2)
legend("topright", legend=c("Posterior", "Log Normal"),
       col=c("mediumvioletred", "olivedrab4"), lty=1 , lwd = 2, cex=0.8, text.font=4 )
```

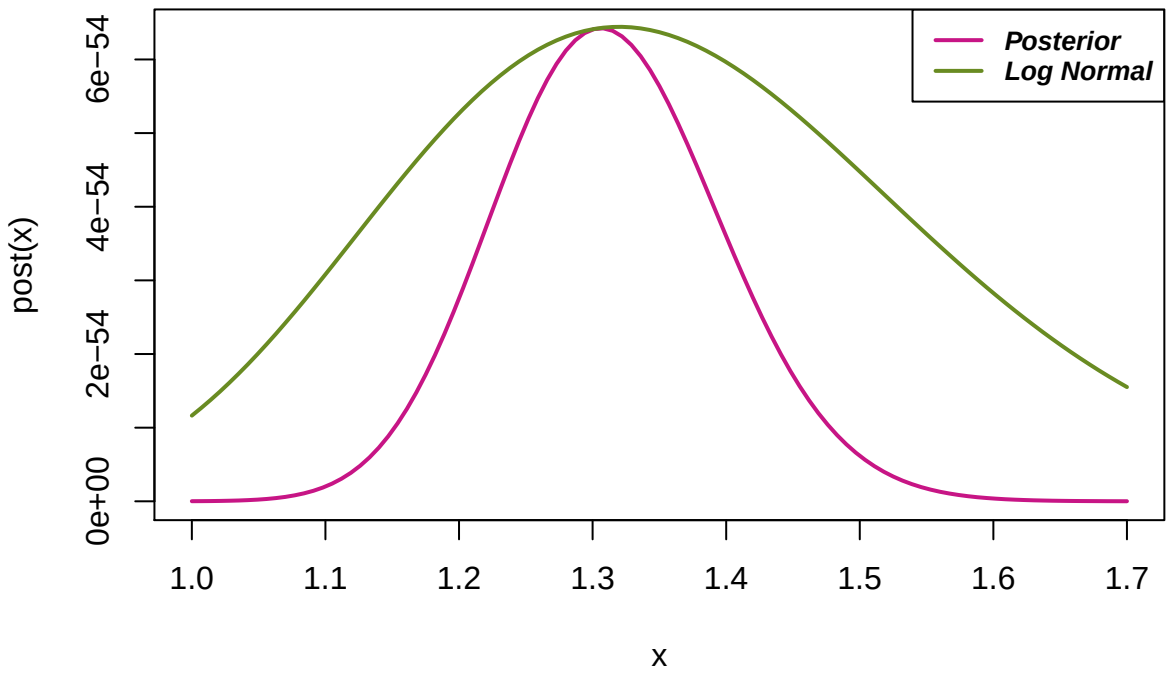


Generate 8,000 retained samples (i.e. post-burnin, thinned samples) with your sampler, setting the seed to 23.

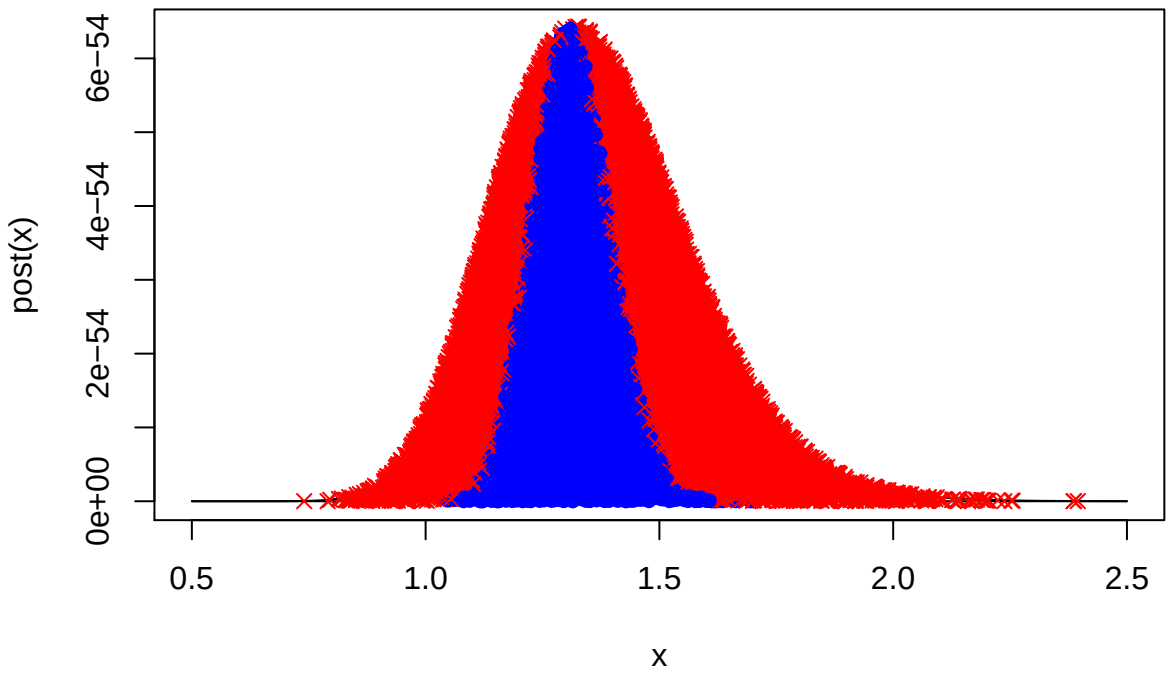
```
# Remove constant
POST <- function( B , w ){  (B/gamma(1/B))^(length(w)) * exp(- sum(abs( w )^B) )  }
Post <- function(B){ POST(B,reign$z)}
post <- function(B){ return(sapply(B, Post )) }

# Find M
q <- function(v){post(v) /   dlnorm(v, .3, .15  ) }
M <- optimize( q , lower=.2, upper = 3, maximum = T )$objective

curve(post, from = .5, to = 2.5 , col = "mediumvioletred",lwd = 2  )
curve( M* dlnorm(x, a, b ), add = T, col = "olivedrab4" ,lwd = 2)
legend("topright", legend=c("Posterior", "Log Normal"),
       col=c("mediumvioletred", "olivedrab4"), lty=1 , lwd = 2, cex=0.8, text.font=4 )
```




```
#zoom
curve(post, from = 1, to = 1.7 , col = "mediumvioletred",lwd = 2 )
curve( M*dlnorm(x, a, b ), add = T, col = "olivedrab4" ,lwd = 2)
legend("topright", legend=c("Posterior", "Log Normal"),
       col=c("mediumvioletred", "olivedrab4"), lty=1 , lwd = 2, cex=0.8, text.font=4 )
```




```
curve(post, from = .5, to = 2.5  )
curve( M* dlnorm(x, a, b ), add = T)

Beta <- rep(NA, 8000)
i <- 1
count <- 1

set.seed(23)
while(i < 8001){
  tb <- rlnorm(1,a,  b)
  U <- runif(1)
  r <- post(tb) / (M*dlnorm(tb,a,  b))
  if(U < r){
    Beta[i] <- tb
    i <- i + 1
    points( tb , M*U*dlnorm(tb,a,  b), col = "blue" , pch = 16) } else {
      points( (tb) , M*U*dlnorm(tb,a,  b), col = "red" , pch = 4) }
  count <- count + 1 }
```




```
8000/count
```

```
## [1] 0.426826
```

```
median(Beta)
```

```
## [1] 1.309744
```

Based on your samples of β, determine which of the two likelihoods matches the data best (it may not be exact).

It is clear that β is closer to 1 and the Laplacian likelihood would be the preferred choice.

Next, return to the original data, i.e. unstandardized data x_i, and model it with your chosen likelihood. Derive the posterior and full conditionals.

Because the preferred choice is the double exponential, we let $x_i \sim L(\mu, \sigma)$. We can represent this model as a mixture of a normal likelihood and inverse-gamma priors. Thus, let $x_i \sim N\left(\mu, \ \frac{4\sigma^2}{\alpha_i}\right)$, where $\sigma^2 \sim IG(a, b)$, and $\alpha_i \overset{iid}{\sim} IG\left(1, \frac{1}{2}\right)$. Additionally, we will use a flat prior on μ, $\pi(\mu) \propto 1$.

$$\mathcal{L}(x_1, ..., x_n|\mu, \sigma^2, \alpha_1, ..., \alpha_n) \propto \prod_{i=1}^{n} \left(\frac{\sigma^2}{\alpha_i}\right)^{-\frac{1}{2}} \exp\left[-\frac{\alpha_i}{2 \cdot 4\sigma^2}\,(x_i - \mu)^2\right]$$

$$P(\mu, \sigma^2, \alpha_1, ..., \alpha_n|x_1, ..., x_n) \propto \left(\sigma^2\right)^{-(a+1)}\ \exp\left(-\frac{b}{\sigma^2}\right)\ \prod_{i=1}^{n}\ \left(\frac{\alpha_i}{\sigma^2}\right)^{\frac{1}{2}}\ \exp\left[-\frac{\alpha_i}{2 \cdot 4\sigma^2}\,(x_i - \mu)^2\right]\ \alpha_i^{-2}\ \exp\left(-\frac{1}{2\alpha_i}\right)$$

$$\begin{aligned}
P(\mu|\text{rest}) &\propto \prod_{i=1}^{n}\ \exp\left[-\frac{\alpha_i}{2 \cdot 4\sigma^2}\,(x_i - \mu)^2\right] \\
&= \exp\left[-\frac{1}{2 \cdot 4\sigma^2}\ \sum_{i=1}^{n} \alpha_i\ (x_i - \mu)^2\right] \\
&= \exp\left[-\frac{1}{2 \cdot 4\sigma^2}\ \sum_{i=1}^{n} \left(\alpha_i x_i^2 - 2\mu\alpha_i x_i + \alpha_i \mu^2\right)\right] \\
&\propto \exp\left[-\frac{1}{2 \cdot 4\sigma^2}\ \left(\mu^2 \sum_{i=1}^{n} \alpha_i - 2\mu \sum_{i=1}^{n} \alpha_i x_i\right)\right] \\
&= \exp\left[-\frac{\sum_{i=1}^{n} \alpha_i}{2 \cdot 4\sigma^2}\ \left(\mu^2 - 2\mu\ \frac{\sum_{i=1}^{n} \alpha_i x_i}{\sum_{i=1}^{n} \alpha_i}\right)\right]
\end{aligned}$$

$$\mu|\text{rest} \sim N\left(\frac{\sum_{i=1}^{n} \alpha_i x_i}{\sum_{i=1}^{n} \alpha_i},\ \frac{4\sigma^2}{\sum_{i=1}^{n} \alpha_i}\right)$$

$$\begin{aligned}
P(\sigma^2|\text{rest}) &\propto \left(\sigma^2\right)^{-(a+1)}\ \exp\left(-\frac{b}{\sigma^2}\right)\ \prod_{i=1}^{n}\ \left(\frac{\alpha_i}{\sigma^2}\right)^{\frac{1}{2}}\ \exp\left[-\frac{\alpha_i}{2 \cdot 4\sigma^2}\,(x_i - \mu)^2\right] \\
&\propto \left(\sigma^2\right)^{-\left(\frac{n}{2}+a+1\right)}\ \exp\left[-\frac{1}{\sigma^2}\left(b + \frac{1}{8}\ \sum_{i=1}^{n} \alpha_i (x_i - \mu)^2\ \right)\right]
\end{aligned}$$

$$\sigma^2|\text{rest} \sim IG\left(\frac{n}{2} + a,\ b + \frac{1}{8}\ \sum_{i=1}^{n} \alpha_i (x_i - \mu)^2\right)$$

$$
\begin{aligned}
P(\alpha_k|\text{rest}) &\propto \alpha_k^{\frac{1}{2}} \ \exp\left[-\frac{\alpha_k}{2\cdot 4\sigma^2}\,(x_k-\mu)^2\right] \ \alpha_k^{-2} \ \exp\left(-\frac{1}{2\alpha_k}\right) \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{1}{2}\left(\frac{\alpha_k(x_k-\mu)^2}{4\sigma^2}+\frac{1}{\alpha_k}\right)\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{1}{2}\left(\frac{\alpha_k^2(x_k-\mu)^2+4\sigma^2}{4\sigma^2\alpha_k}\right)\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{1}{2}\left(\frac{\alpha_k^2+\frac{4\sigma^2}{(x_k-\mu)^2}}{\frac{4\sigma^2}{(x_k-\mu)^2}\alpha_k}\right)\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{1}{2}\left(\frac{\alpha_k^2+\frac{4\sigma^2}{(x_k-\mu)^2}}{\frac{4\sigma^2}{(x_k-\mu)^2}\alpha_k}\right)\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{1}{2}\left(\frac{\alpha_k^2-2\alpha_k\frac{2\sigma}{|y_k-\mu|}+\frac{4\sigma^2}{(x_k-\mu)^2}+2\alpha_k\frac{2\sigma}{|y_k-\mu|}}{\frac{4\sigma^2}{(x_k-\mu)^2}\alpha_k}\right)\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{\left(\alpha_k-\frac{2\sigma}{|y_k-\mu|}\right)^2+2\alpha_k\frac{2\sigma}{|x_k-\mu|}}{2\left(\frac{2\sigma}{|x_k-\mu|}\right)^2\alpha_k}\right] \\
&\propto \alpha_k^{-\frac{3}{2}} \ \exp\left[-\frac{\left(\alpha_k-\frac{2\sigma}{|y_k-\mu|}\right)^2}{2\left(\frac{2\sigma}{|x_k-\mu|}\right)^2\alpha_k}\right]
\end{aligned}
$$

$$
\alpha_k|\text{rest} \sim Inverse-Gaussian\left(\frac{2\sigma}{|y_k-\mu|},1\right)
$$

Use a Gibbs Sampler to draw posterior estimates. Summarize the results for all model parameters in the usual fashion.

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")
suppressMessages(library(rmutil, quietly = T))
suppressMessages(library(MCMCpack, quietly = T))
library(beepr)

reign <- read.table("coup1280.txt", stringsAsFactors = F , header =T)
B = 10000
mus <- sigmas <- rep(NA,B)
n <- nrow(reign)
alphas <- matrix(NA, B, n )
x <- reign$logCoup

mup <- function( Alpha, y ){ sum(y*Alpha)/sum(Alpha) }
```

```
vp <- function( Alpha, ss ){ (4*ss) / sum(Alpha)  }
thetap <- function( Mu, ss, y){ (2*sqrt(ss))/ abs(y-Mu) }
bp <- function( Mu, Alpha, y ){(1/8) * sum(Alpha * (y-Mu)^2 )}

a = b = 1

alphas[1,] <- rinvgamma(n, 1, .5 )
sigmas[1] <- rinvgamma(1, (n/2) + a, b + (1/8) )
mus[1] <- rnorm(1 , mup(alphas[1,],x) , sqrt( vp(alphas[1,],sigmas[1] ) )  )
```

```
set.seed(23)
for( i in 2:B){
  mus[i] <- rnorm( 1, mup(alphas[i-1,],x) , sqrt( vp(alphas[i-1,],sigmas[i-1] ) ) )
  sigmas[i] <- rinvgamma(1, (n/2) + a , b + bp(mus[i-1],alphas[i-1] , x)  )
  for(k in 1:n){ alphas[i,k] <- rinvgauss(1, thetap(mus[i],sigmas[i],x[k] ),  1) }
}
```

```
mus <- tail(mus,B/2)
sigmas <- tail(sigmas,B/2)

require(mcmcplots)
```

```
## Loading required package: mcmcplots
```

```
quantile(mus , probs = c(0.5, 0.025, 0.975))
```

```
##       50%      2.5%     97.5%
## -6.061660 -6.172859 -5.947990
```

```
quantile(sigmas , probs = c(0.5, 0.025, 0.975))
```

```
##        50%       2.5%      97.5%
## 0.06494110 0.03675039 0.12836293
```

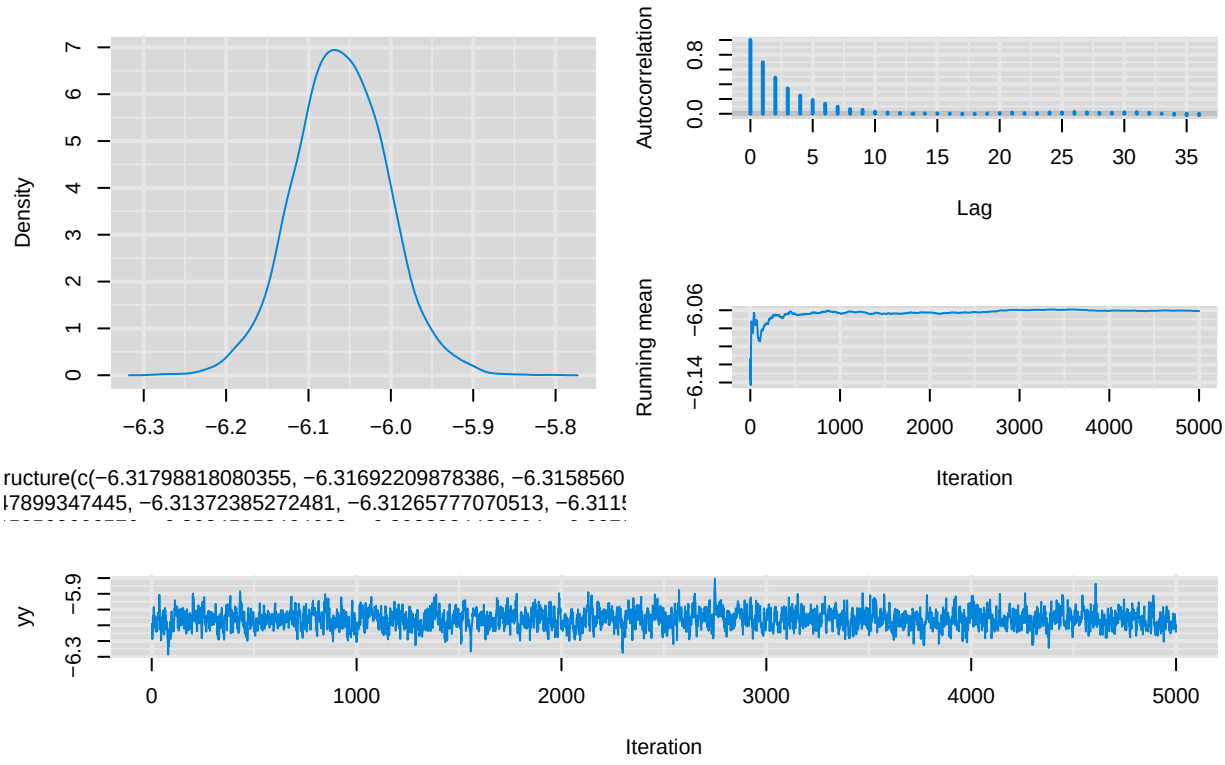
```
geweke.diag(mcmc( mus ) )
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##    var1
## -0.5867
```

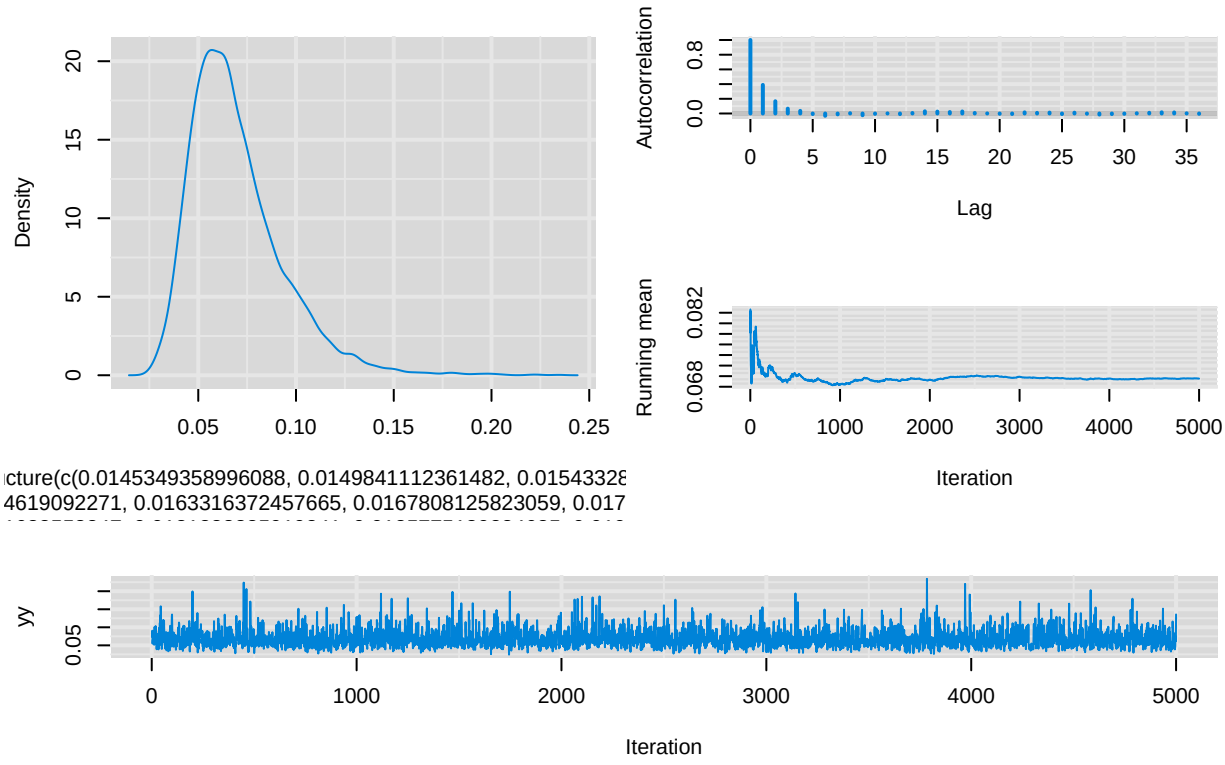
```
geweke.diag(mcmc( sigmas ) )
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## 0.7492
```

```
mcmcplot1( matrix(mus , ncol = 1)  )
```




```
mcmcplot1( matrix(sigmas , ncol = 1)  )
```



For both parameters, the Geweke diagnoistic was sufficiently small enough to suggest that both converged. This, along with the trace and ruuning mean plots, signals convergence of the parameters.

2. The Veteran's Administration conducted a study of time to death in veterans with various types of lung cancer. In total, 137 veterans were part of the study with lung cancer types including small cell, adenocarcinoma, squamous cell, and large cell. We are interested in building a parametric model for the survivor function of the 27 vets with large cell lung cancer. The Weibull distribution is commonly used as a parametric model for survivor functions. The form of the Weibull distribution is

$$p(t_i) = \frac{\theta}{\lambda^\theta}\, t_i^{\theta-1}\ \exp\left[-\left(\frac{t_i}{\lambda}\right)^\theta\right] \text{ for } t_i > 0 \text{ and } \lambda,\ \theta > 0.$$

Determine the likelihood and, using the non-informative joint prior of $\pi(\lambda, \theta) \propto \left(\lambda^\theta\right)^{-1}$, find the posterior and full conditionals.

$$\begin{aligned}\mathcal{L}(t_i|\theta,\lambda) &\propto \prod_{i=1}^{n}\ \frac{\theta}{\lambda^\theta}\, t_i^\theta\ \exp\left[-\left(\frac{t_i}{\lambda}\right)^\theta\right] \\ &\propto \left(\frac{\theta}{\lambda^\theta}\right)^n\ \left[\prod_{i=1}^{n} t_i\right]^\theta\ \exp\left[-\frac{1}{\lambda^\theta}\sum_{i=1}^{n} t_i^\theta\right]\end{aligned}$$

$$\begin{aligned}p(\theta,\lambda|t_i) &\propto \left(\frac{\theta}{\lambda^\theta}\right)^n\ \left[\prod_{i=1}^{n} t_i\right]^\theta\ \exp\left[-\frac{1}{\lambda^\theta}\sum_{i=1}^{n} t_i^\theta\right]\ \left(\lambda^\theta\right)^{-1} \\ &\propto \left(\lambda^\theta\right)^{-(n+1)}\ \theta^n \left[\prod_{i=1}^{n} t_i\right]^\theta\ \exp\left[-\frac{1}{\lambda^\theta}\sum_{i=1}^{n} t_i^\theta\right]\end{aligned}$$

$$p(\theta|\lambda,t_i) \propto \left(\lambda^\theta\right)^{-(n+1)}\ \theta^n \left[\prod_{i=1}^{n} t_i\right]^\theta\ \exp\left[-\frac{1}{\lambda^\theta}\sum_{i=1}^{n} t_i^\theta\right]$$

$$\begin{aligned}p(\lambda|\theta,t_i) &\propto \left(\lambda^\theta\right)^{-(n+1)}\ \exp\left[-\frac{1}{\lambda^\theta}\sum_{i=1}^{n} t_i^\theta\right] \qquad \text{Let } \mu = \lambda^\theta \\ &\propto (\mu)^{-(n+1)}\ \exp\left[-\frac{1}{\mu}\sum_{i=1}^{n} t_i^\theta\right] \\ \mu &\sim IG\left(n, \sum_{i=1}^{n} t_i^\theta\right)\end{aligned}$$

Write a Gibbs-MH sampler to implement your model taking $B = 50000$ total samples and, post-burnin, determine an appropriate level of thinning. For the first run, use the starting values of $\theta^{(1)} = 0.1$ and $\lambda^{(1)} = 1$ and set the seed to 121 - use this run to tune your acceptance rate

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")
library(mcmcplots, quietly = T)
suppressMessages(library(MCMCpack, quietly = T))

VA <- read.table('valc.txt', header = T)
x <- VA$t

Thetas <- Lambdas <- Ars <- list()
thets <- c(0.1,0.2,0.15,0.05)
lams <- c( 1,3,0.5, 1.5)
seeds <- c(121,75,340,19)
a <- 4.2
b <- 3.9

thetaden <- function(y, theta, lambda ){
  lamthet <- lambda^theta
  ( theta / lamthet  )^n *
    (prod(y))^theta *
    exp( - (1/ lamthet) * sum(y^theta)  ) *
    (lamthet)^(-1)
}

B <- 50000*2
```

```
for( j in 1){

thetas <- lambdas <- rep(NA,B)
Ar <- rep(0,B)
thetas[1] <- th <- thets[j]
lambdas[1] <- lams[j]
n <- length(x)

set.seed(seeds[j])
for( i in 2:B){

  mu <- rinvgamma(1,  n ,  sum( x^thetas[i-1] ) )
  lambdas[i] <- mu^(1/thetas[i-1])

  thetastar <- rgamma(1, a, b)

  rho <- ( thetaden(x, thetastar, lambdas[i-1] ) / thetaden(x, thetas[i-1], lambdas[i-1] )
       ) / (
      dgamma(thetastar, a, b )/dgamma(thetas[i-1], a, b )
       )
  rho <- min(rho,1)

  U  <- runif(1)
  if( U < rho ){
```

```
    th <- thetastar
    Ar[i] <- 1
  }
  thetas[i]  <- th
}

Ars[[j]] <- mean(Ar)
Thetas[[j]] <- tail(thetas,B/2)
Lambdas[[j]] <- tail(lambdas,B/2)
}
```

```
Ars
```

```
## [[1]]
## [1] 0.37189
```

Repeat this step three more times using the starting values of $\theta^{(1)} = 0.2$ and $\lambda^{(1)} = 3$ with seed 75, $\theta^{(1)} = 0.15$ and $\lambda^{(1)} = 0.5$ with seed 340, and $\theta^{(1)} = 0.05$ and $\lambda^{(1)} = 1.5$ with seed 19.

```
for( j in 2:4){

thetas <- lambdas <- rep(NA,B)
Ar <- rep(0,B)
thetas[1] <- th <- thets[j]
lambdas[1] <- lams[j]
n <- length(x)

set.seed(seeds[j])
for( i in 2:B){

  mu <- rinvgamma(1,  n ,  sum( x^thetas[i-1] ) )
  lambdas[i] <- mu^(1/thetas[i-1])

  thetastar <- rgamma(1, a, b)

  rho <- ( thetaden(x, thetastar, lambdas[i-1] ) / thetaden(x, thetas[i-1], lambdas[i-1] )
       ) / (
      dgamma(thetastar, a, b )/dgamma(thetas[i-1], a, b )
       )
  rho <- min(rho,1)

  U  <- runif(1)
  if( U < rho ){
    th <- thetastar
    Ar[i] <- 1
  }
  thetas[i]  <- th
}

Ars[[j]] <- mean(Ar)
Thetas[[j]] <- tail(thetas,B/2)
Lambdas[[j]] <- tail(lambdas,B/2)
```

```
}
```

Using the full post-burnin chains from each of the four runs, asses convergence for all model parameters with the Gelman-Rubin diagnostic.

```
mcmcT <- mcmcL <- list()
for(w in 1:4){
  mcmcT[[w]] <- mcmc(Thetas[[w]])
  mcmcL[[w]] <- mcmc(Lambdas[[w]])
  }

GRtheta <- mcmc.list(list(mcmcT[[1]], mcmcT[[2]], mcmcT[[3]], mcmcT[[4]]))
GRlambda <- mcmc.list(list(mcmcL[[1]], mcmcL[[2]], mcmcL[[3]], mcmcL[[4]]))

gelman.diag(GRtheta)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

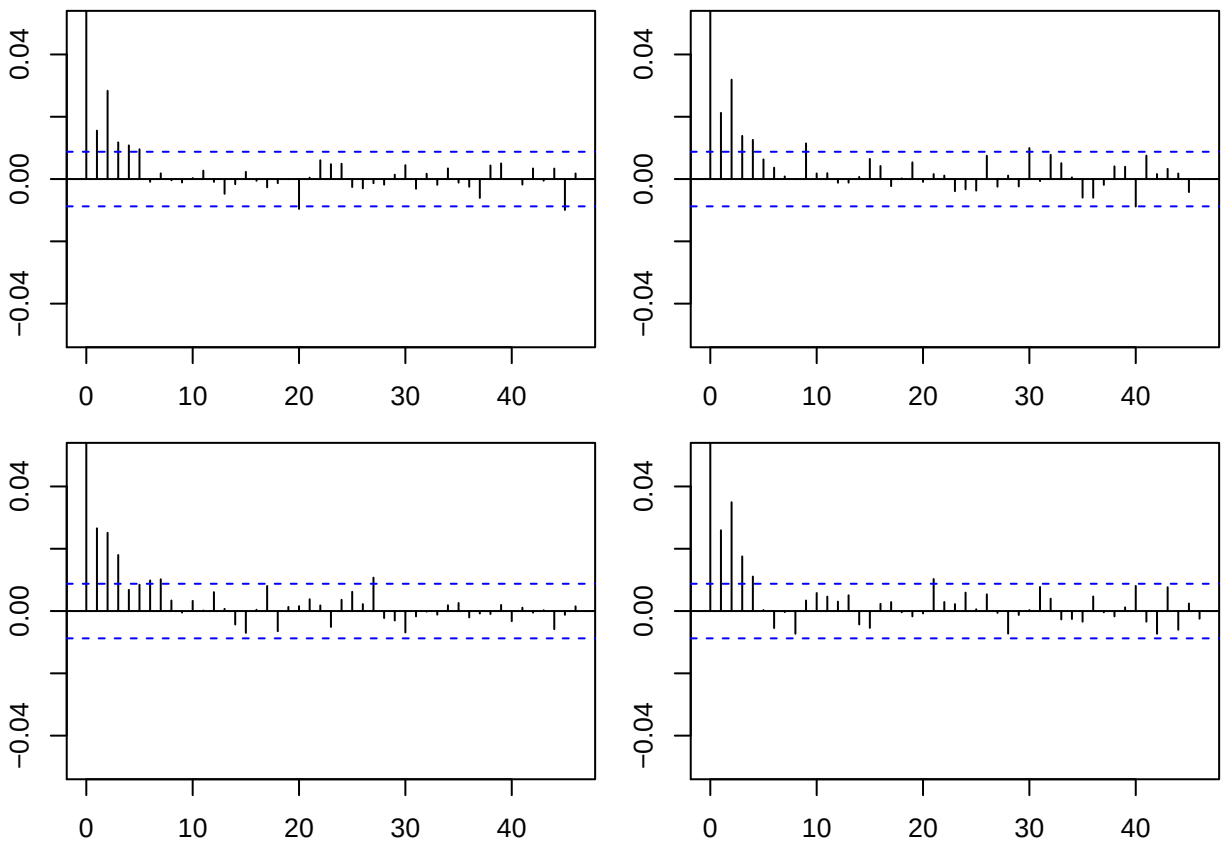
```
gelman.diag(GRlambda)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

$\hat{R}$ is near 1 for all parameters so we can be confident that convergence has been achieved.

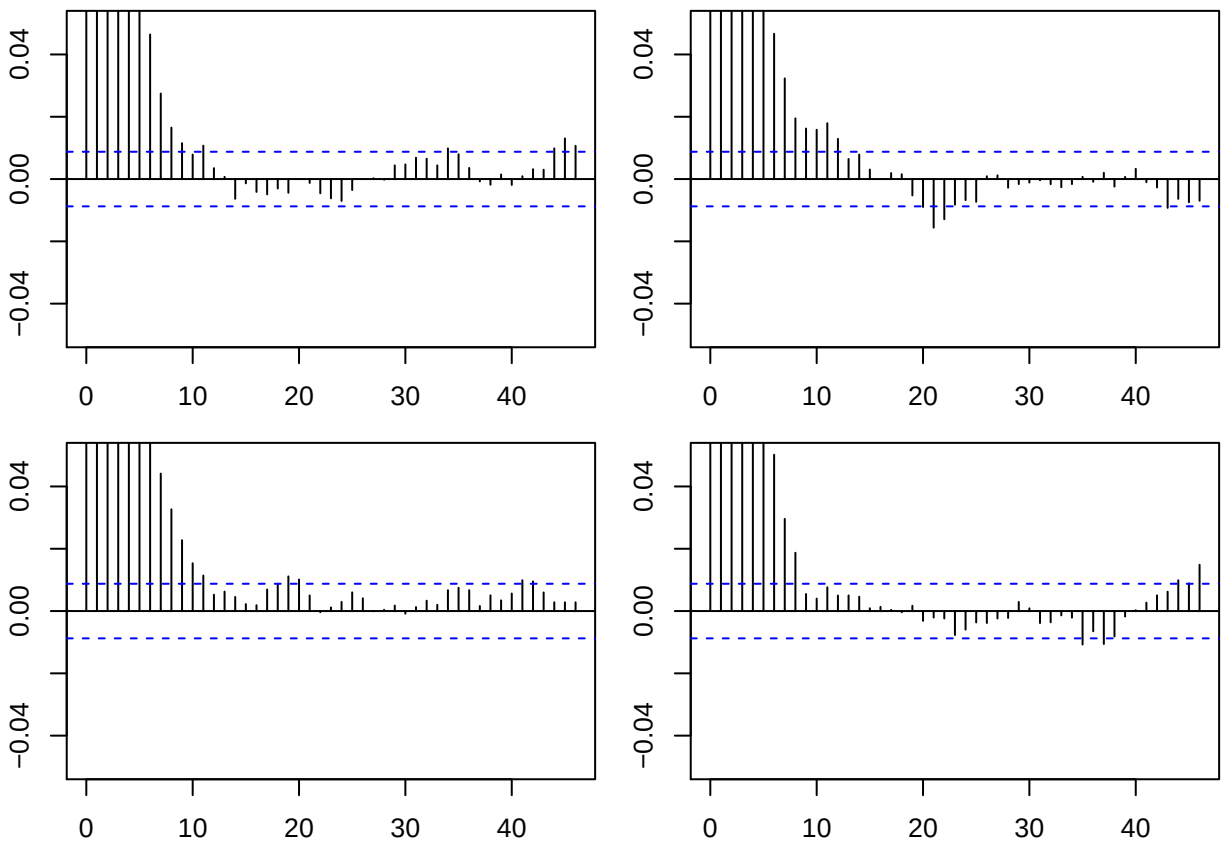
Then thin the chains based on your selected level of thinning and combine to determine posterior summaries of λ and θ.

```
par(mfrow=c(2,2) , mar= c(2, 2 , 1, 1 ) )
  for( w in 1:4){ acf(  Lambdas[[w]] , ylim=c(-.05,.05) )  }
```



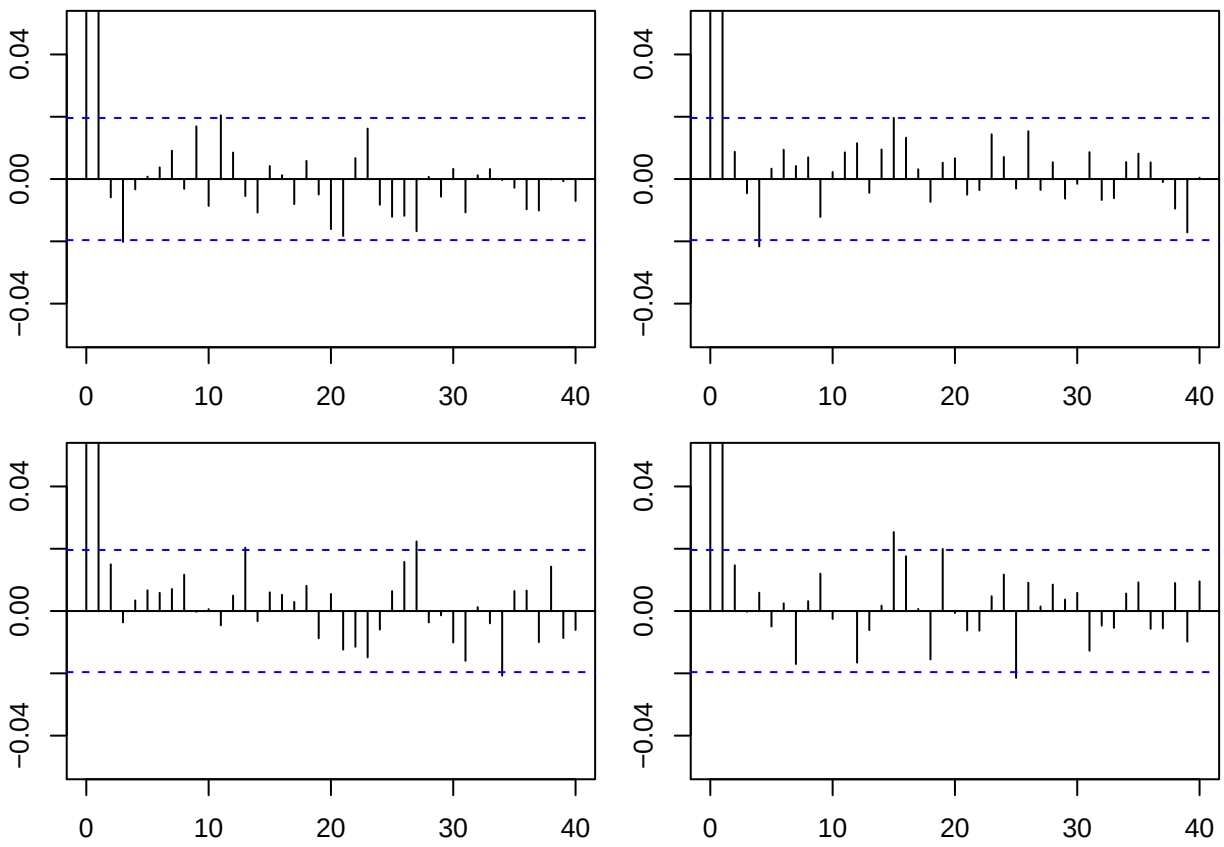
We see some autocorrelation among the λ's.

```
par(mfrow=c(2,2) , mar= c(2, 2 , 1, 1 ) )
for( w in 1:4){ acf(  Thetas[[w]]   , ylim=c(-.05,.05) )  }
```



Had to zoom in to see the differences, but there is surely autocorrelation detected among the θ's. We will need to thin the θ's. First, we try a thinning by 5.

```
par(mfrow=c(2,2) , mar= c(2, 2 , 1, 1 )  )
for( w in 1:4){ acf(  Thetas[[w]][ seq(1,B/2,5) ], ylim=c(-.05,.05) ) }
```



Thinning by 5 has reduced the autocorrelation sufficiently. So we need to rerun our original sampler to get updated posterior estimates.

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")

VA <- read.table('valc.txt', header = T)
x <- VA$t

Thetas <- Lambdas <- Ars <- list()
thets <- c(0.1,0.2,0.15,0.05)
lams <- c( 1,3,0.5, 1.5)
seeds <- c(121,75,340,19)
a <- 4.2
b <- 3.9

thetaden <- function(y, theta, lambda ){
  lamthet <- lambda^theta
  ( theta / lamthet  )^n *
    (prod(y))^theta *
    exp( - (1/ lamthet) * sum(y^theta)  ) *
    (lamthet)^(-1)
}
```

```
B <- 50000*2*5

for( j in 1:4){

thetas <- lambdas <- rep(NA,B)
Ar <- rep(0,B)
thetas[1] <- th <- thets[j]
lambdas[1] <- lams[j]
n <- length(x)

set.seed(seeds[j])
for( i in 2:B){

  mu <- rinvgamma(1,  n ,  sum( x^thetas[i-1] ) )
  lambdas[i] <- mu^(1/thetas[i-1])

  thetastar <- rgamma(1, a, b)

  rho <- ( thetaden(x, thetastar, lambdas[i-1] ) / thetaden(x, thetas[i-1], lambdas[i-1] )
       ) / (
      dgamma(thetastar, a, b )/dgamma(thetas[i-1], a, b )
       )
  rho <- min(rho,1)

  U  <- runif(1)
  if( U < rho ){
    th <- thetastar
    Ar[i] <- 1
  }
  thetas[i]  <- th
}

Ars[[j]] <- mean(Ar)
Thetas[[j]] <- tail(thetas,B/2)[ seq(1,B/2,5) ]
Lambdas[[j]] <- tail(lambdas,B/2)[ seq(1,B/2,5) ]
}
```

Posterior Summaries:

```
library(ggplot2, quietly = T)
require(mcmcplots)

#acceptance rates
unlist( Ars )
```

```
## [1] 0.370788 0.372962 0.372088 0.371798
```

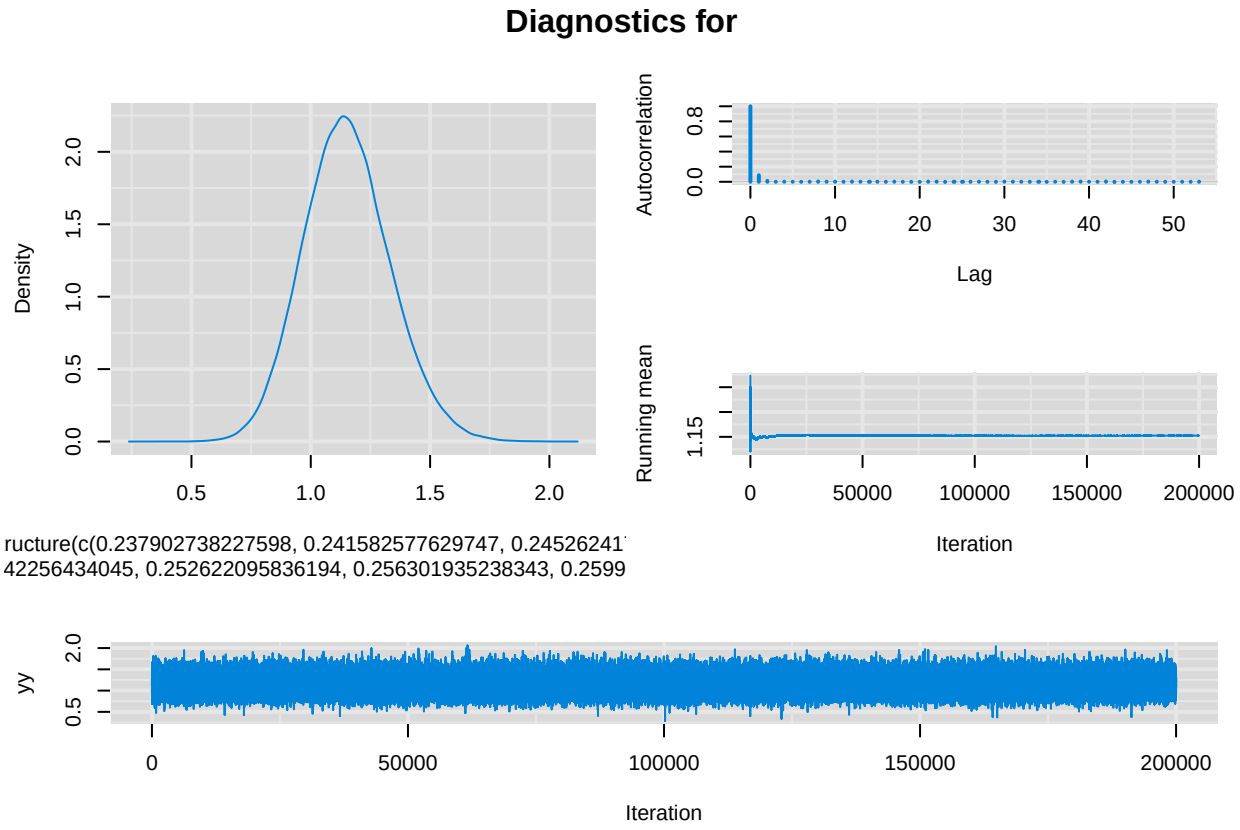
```
# Theta
thets <- Thetas; lambs <- Lambdas
Thetas <- unlist(Thetas)
quantile(Thetas, probs = c(0.025, 0.5, 0.975))
```

```
##      2.5%       50%     97.5%
## 0.8150899 1.1460680 1.5287173
```

```
# Lambda
Lambdas <- unlist(Lambdas)
quantile(Lambdas, probs = c(0.025, 0.5, 0.975))
```

```
##     2.5%      50%    97.5%
## 122.1319 175.3741 249.6503
```

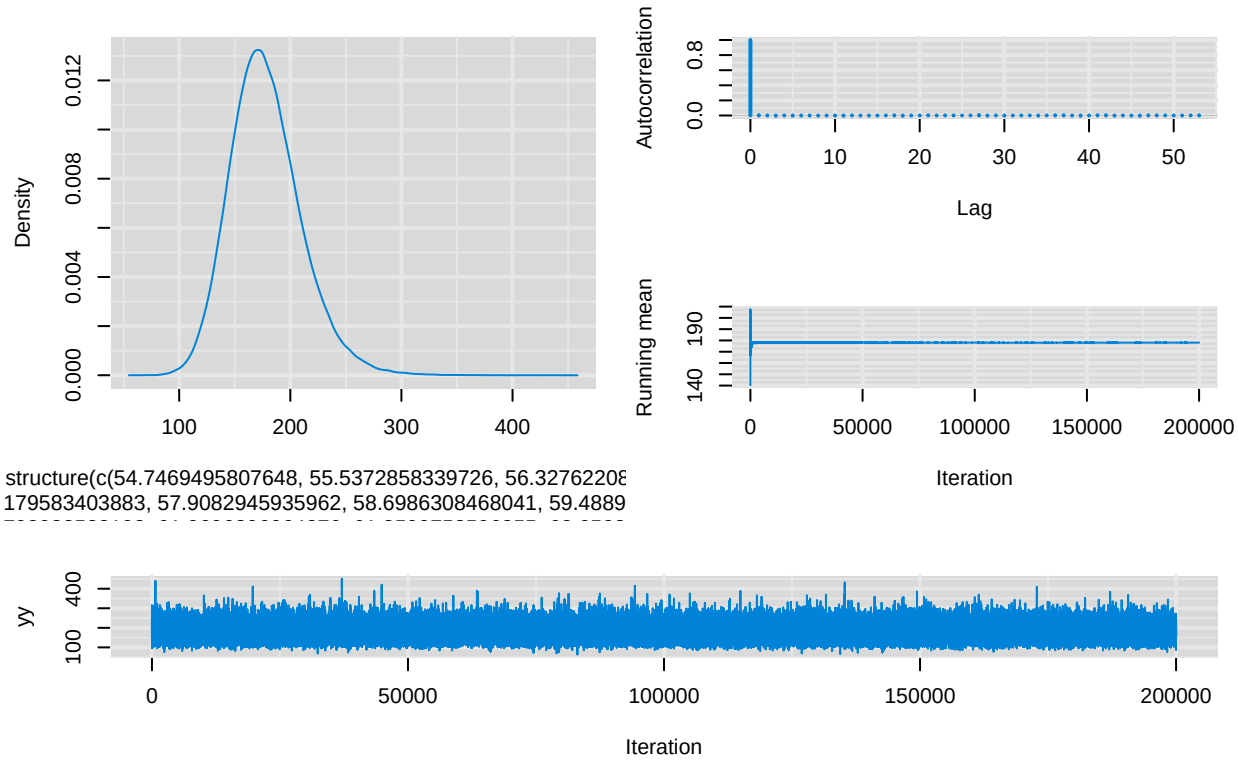
```
mcmcplot1( matrix(Thetas , ncol = 1)  )
```




```
geweke.diag(mcmc( Thetas ) )
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## -0.115
```

```
mcmcplot1( matrix(Lambdas , ncol = 1)  )
```

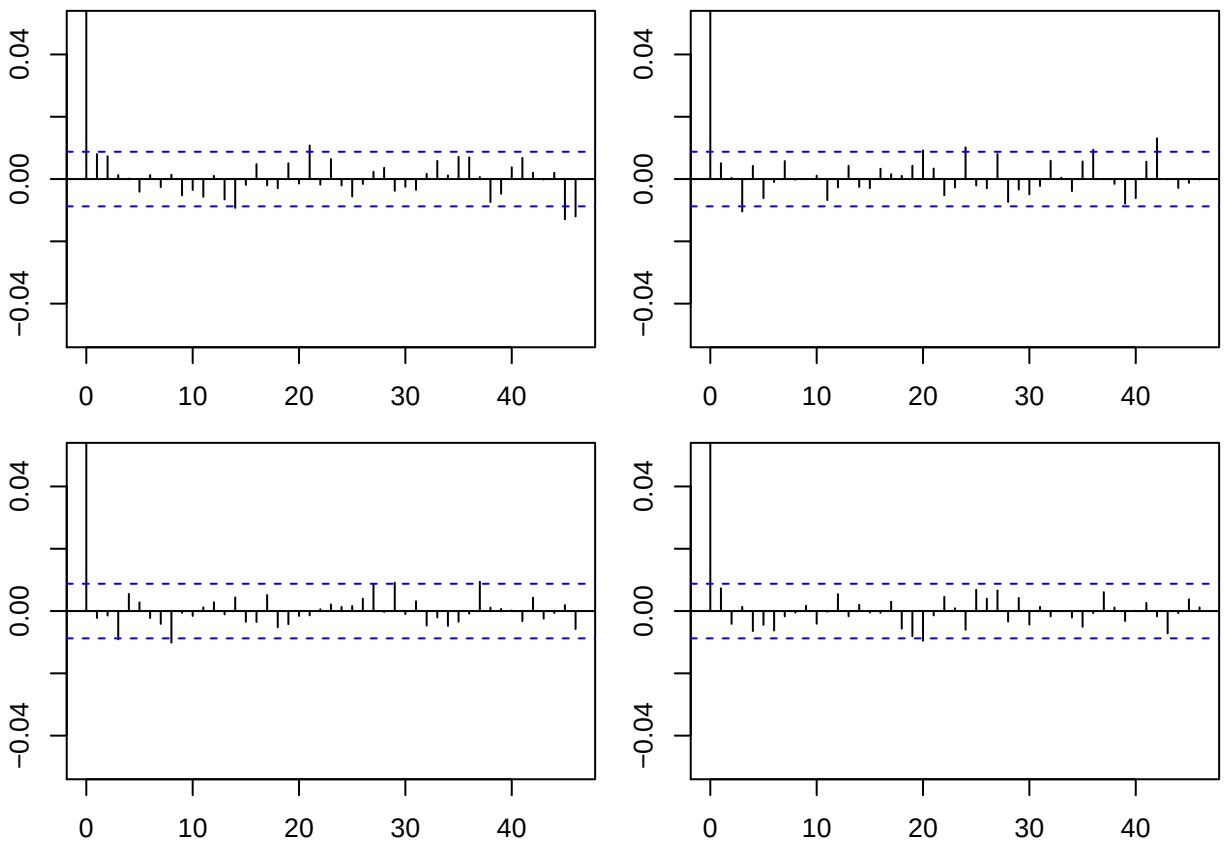



```
geweke.diag(mcmc( Lambdas ) )
```

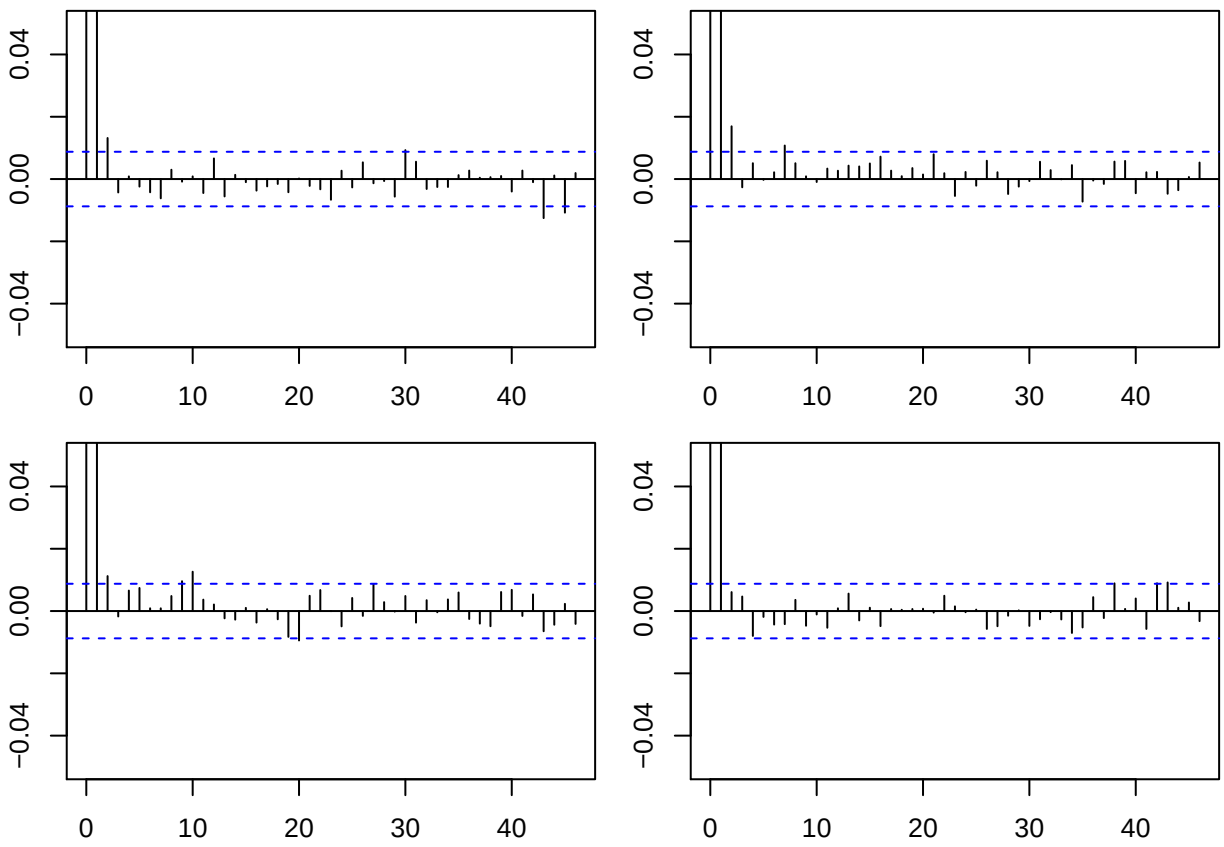
```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## 0.6957
```

Although we combined the chains, we see that seperately the thinning reduced the autocorrelation for both θ and λ.

```
par(mfrow=c(2,2) , mar= c(2, 2 , 1, 1 ) )
  for( w in 1:4){ acf(  lambs[[w]] , ylim=c(-.05,.05) )  }
```




```
par(mfrow=c(2,2) , mar= c(2, 2 , 1, 1 ) )
  for( w in 1:4){ acf(  thets[[w]] , ylim=c(-.05,.05) )  }
```



Once you have the posterior samples, estimate the survivor function using the following relationship:

$$S(t) = 1 - F(t) = 1 - \int_0^t p(a)\ \mathrm{d}a.$$

First, find the CDF of the Weibull.

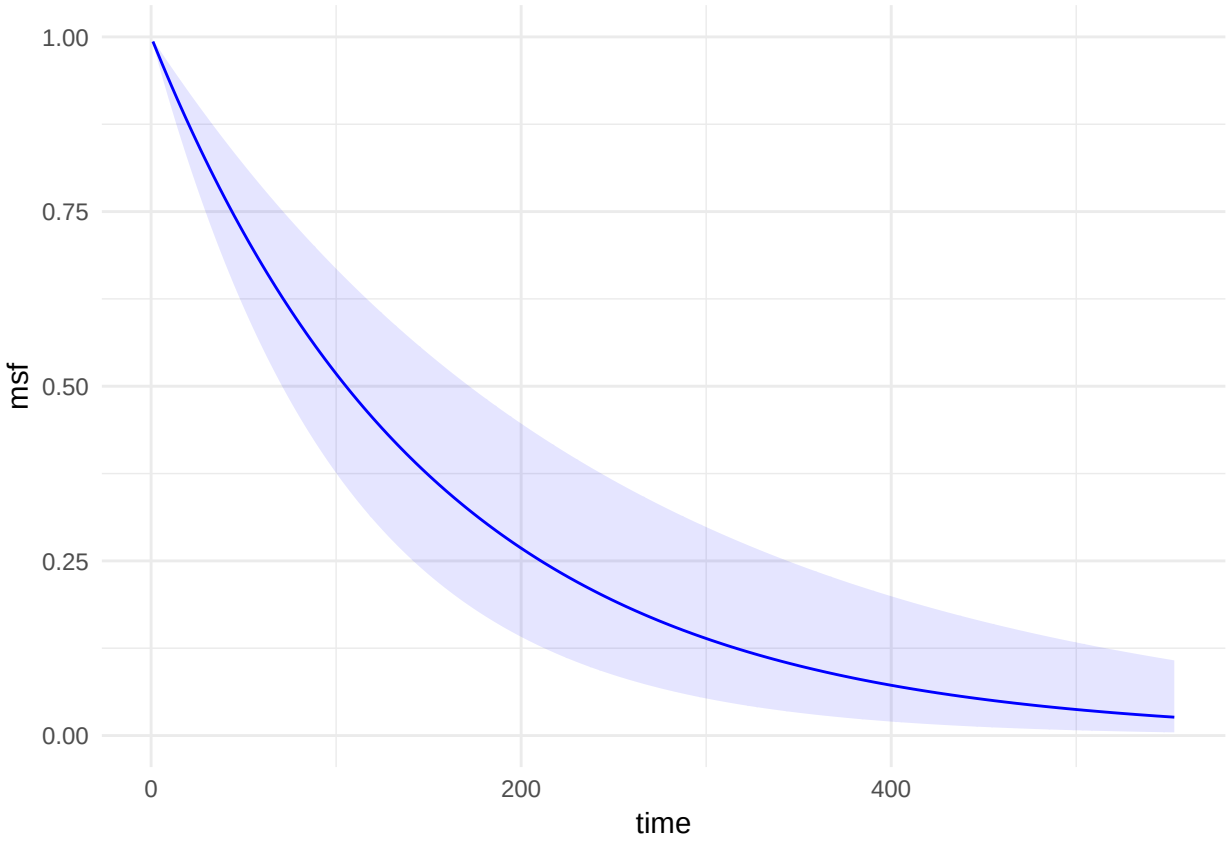
$$F(t) = \begin{cases} 1 - \exp\left(-\dfrac{t}{\lambda}\right)^{\theta} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

Next, using your posterior samples of θ and λ, generate a posterior distribution of survivor functions. To do this, you will need to assume a grid of possible survival times starting. Set your grid to begin at 1 and go to the largest observed survival time in the data by 1. Note $S(t)$ will be a function of your posterior samples and your grid t, no new model needs to be fit.

```
St <- function( y, theta, lambda  ){ exp( -y / lambda )^theta }

smat <- matrix(NA, max(x),length(Thetas )   )
for( k in 1:length(Thetas ) ){
```

```
  smat[ ,k ] <- St( 1:max(x), Thetas[k], Lambdas[k] )
}

SCI <- matrix(NA, nrow(smat) , 2)

for( i in 1:nrow(smat)) {
  Q <- quantile(smat[i,], probs = c(0.025, 0.5, 0.975))
  SCI[i,] <- Q[c(1,3)]; rm(Q)
}

msf <- apply(smat, 1,median )
msf <- data.frame( time = 1:max(x) , msf )
msf$lower <- SCI[,1]
msf$upper <- SCI[,2]
rm(smat)
```

Summarize your results graphically with a plot of the median survivor function and credible interval for the survivor function.



3. Data on the physiochemical properties of Vinho Verde, a Portuguese wine that is typically white, is in the file `vinhoverde.txt`. It contains data on a wide variety of properties recorded on nearly 5,000 different bottles. The properties include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. We are interested in building a logistic regression model with a flat prior using these properties to predict whether or not the wine was rated as a quality wine (variable `quality`).

Using an M-H Sampler, build a logistic regression model to predict quality.

The likelihood for this model is

$$\mathcal{L}(y_i|\beta, x_i) \propto \exp\left[\sum_{i=1}^{n}\left(y_i\left(x_i^T\beta\right) - \log\left[1+\exp\left(x_i^T\beta\right)\right]\right)\right]$$

for $\beta = \begin{bmatrix}\beta_0 & \beta_1 & \dots & \beta_{11}\end{bmatrix}^T$ and $x_i^T = \begin{bmatrix}1 & x_{1i} & \dots & x_{11i}\end{bmatrix}^T$

We take the joint prior on β to be flat, $\pi(\beta) \propto 1$, the posterior has the same form.

$$P(\beta|y_i, x_i) \propto \exp\left[\sum_{i=1}^{n}\left(y_i\left(x_i^T\beta\right) - \log\left[1+\exp\left(x_i^T\beta\right)\right]\right)\right]$$

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")
library(mcmcplots)
library(MCMCpack)
library(mvtnorm)

B = 8000
wine <- read.table('vinhoverde.txt', header = TRUE)

fit  <- glm(quality ~ . ,   family = binomial, data = wine)
vbeta <- vcov(fit)
X <- model.matrix(fit)
Y <- wine$quality

betas  <- matrix(0, nrow = B, ncol = ncol(X))
betas[1,] <- coef(fit)
Ar <- rep(NA,B-1)

tdens <- function(b, W, z){
  sum( z*(W%*%b) ) - sum(log(  1 + exp(W%*%b)))
}

tau = .3

set.seed(870)
for(i in 2:B){

  bstar <- rmvnorm(1, betas[i-1,], tau*vbeta )
  r  <- exp(tdens(t(bstar), X, Y)-tdens(betas[i-1,], X, Y))
```

```
  U  <- runif(1)
  if(U < min(1,r)){
    betas[i,] <-bstar
    Ar[i-1]  <- 1
  } else{
    betas[i,] <- betas[i-1,]
    Ar[i-1]  <- 0
  }

}
mean(Ar)
```

```
## [1] 0.3624203
```

Justify your choice of final model using statistical support.

```
CI <- t(apply(betas, 2, quantile, probs = c(0.5, 0.025, 0.975)))
rownames(CI) <- colnames(X)
CI <- round(CI,6)
CI
```

```
##                              50%        2.5%       97.5%
## (Intercept)           261.159528  123.700023  401.092587
## fixed.acidity           0.034616   -0.100051    0.173641
## volatile.acidity       -6.444501   -7.271839   -5.690868
## citric.acid             0.143200   -0.506947    0.779630
## residual.sugar          0.171911    0.119625    0.225785
## chlorides               0.870129   -2.587673    4.025413
## free.sulfur.dioxide     0.009723    0.004528    0.015363
## total.sulfur.dioxide   -0.001454   -0.003827    0.001114
## density              -273.875864 -415.604431 -135.121019
## pH                      1.110369    0.384959    1.817080
## sulphates               1.845109    1.112047    2.584246
## alcohol                 0.739581    0.560141    0.923949
```

We remove the variables with 0 in the credible interval to build our final model.

```
CI0 <- (CI[ ,2 ] < 0 & CI[ ,3] > 0 )
CI <-  CI[     !CI0, ]
CI <- as.data.frame(CI)
#CI$dif <- CI[,3]-CI[,2]
CI
```

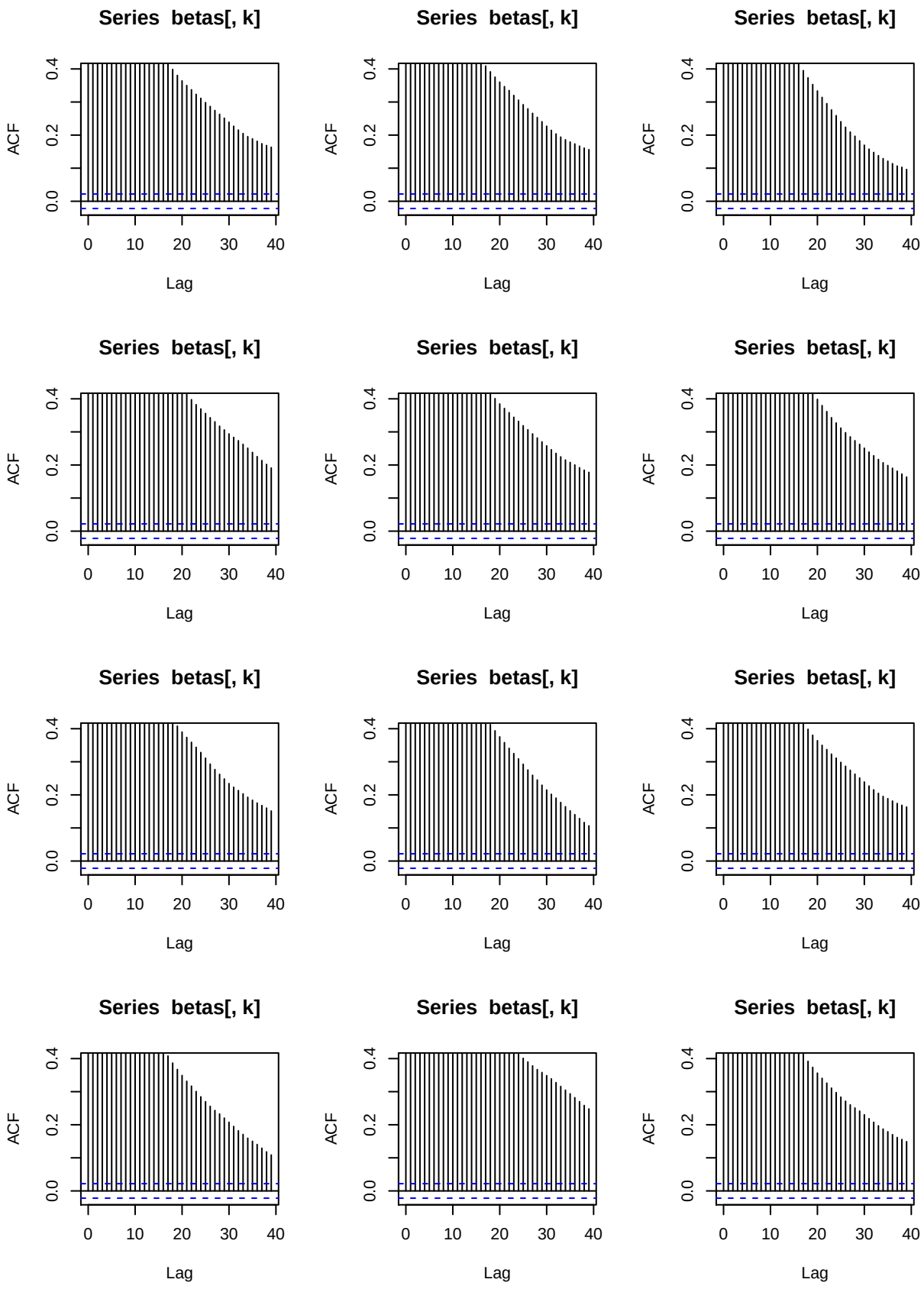
```
##                              50%        2.5%       97.5%
## (Intercept)           261.159528  123.700023  401.092587
## volatile.acidity       -6.444501   -7.271839   -5.690868
## residual.sugar          0.171911    0.119625    0.225785
## free.sulfur.dioxide     0.009723    0.004528    0.015363
## density              -273.875864 -415.604431 -135.121019
## pH                      1.110369    0.384959    1.817080
## sulphates               1.845109    1.112047    2.584246
```

```
## alcohol                0.739581    0.560141    0.923949
```

Design your sampler so to retain a total of 4,000 total samples (after burn-in and, if necessary, thinning). What variables best predict the quality of the wine?

Looking for autocorrelation.

```
par(mfrow=c(4,3))
for( k in 1:12){acf(betas[,k], ylim=c(-.05/2,.4 ))}
```


Series betas[, k]
ACF
Lag
0.0
0.2
0.4
0
10
20
30
40

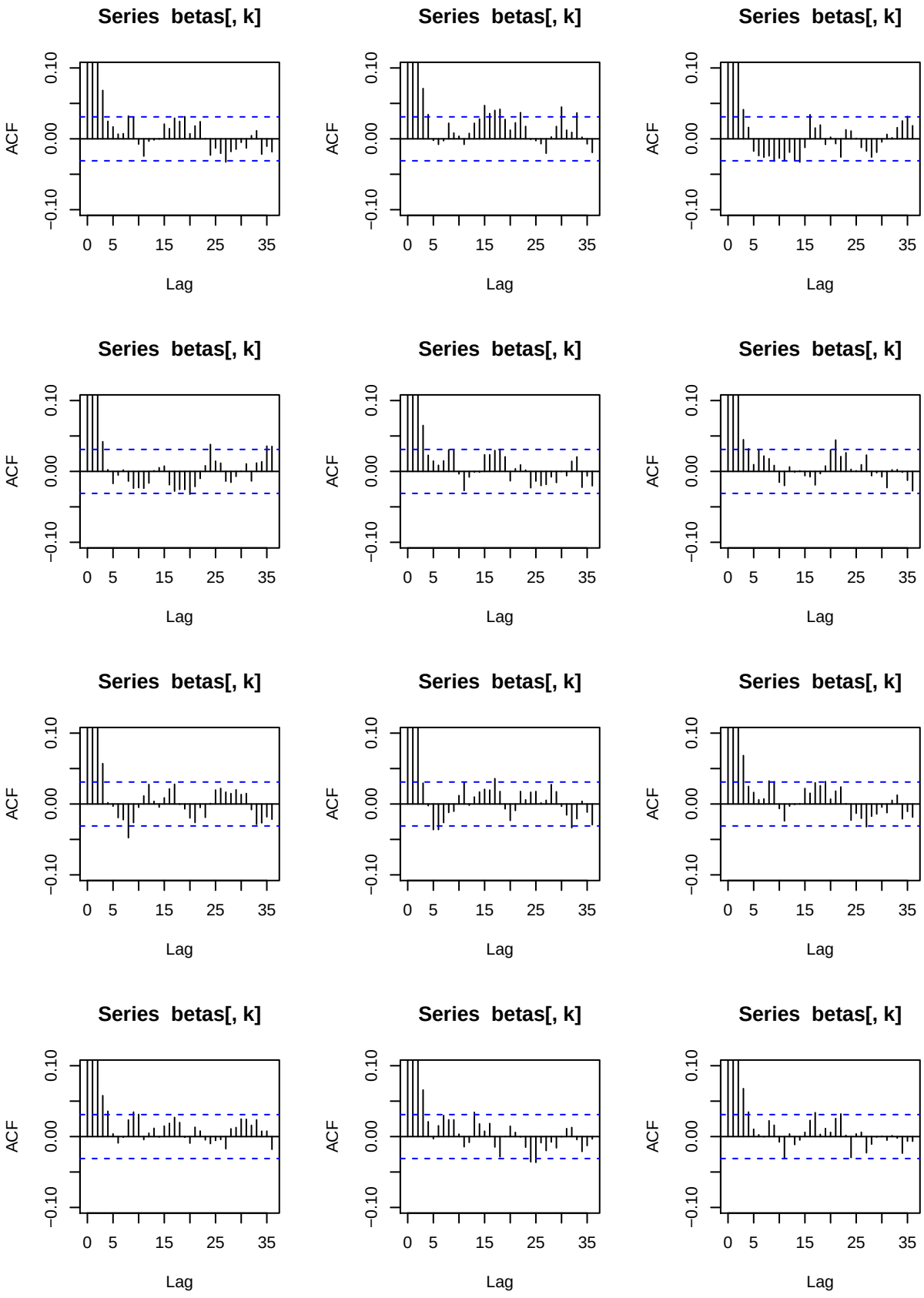
The graphs are not exactly identical, but each variable is showing the same type of autocorrelation. We will rerun the sampler to address autocorrelation by thinning.

```
rm(list=ls());invisible(gc())
setwd("G:\\math\\640")
library(mcmcplots)
library(MCMCpack)
library(mvtnorm)

B = 8000 * 20
wine <- read.table('vinhoverde.txt', header = TRUE)

fit  <- glm(quality ~ . ,   family = binomial, data = wine)

vbeta <- vcov(fit)
X <- model.matrix(fit)
Y <- wine$quality

betas  <- matrix(0, nrow = B, ncol = ncol(X))
betas[1,] <- coef(fit)
Ar <- rep(NA,B-1)

tdens <- function(b, W, z){
  sum( z*(W%*%b) ) - sum(log(  1 + exp(W%*%b)))
}

tau = .3

set.seed(870)
for(i in 2:B){

  bstar <- rmvnorm(1, betas[i-1,], tau*vbeta )
  r  <- exp(tdens(t(bstar), X, Y)-tdens(betas[i-1,], X, Y))
  U  <- runif(1)
  if(U < min(1,r)){
    betas[i,] <-bstar
    Ar[i-1]  <- 1
  } else{
    betas[i,] <- betas[i-1,]
    Ar[i-1]  <- 0
  }
}
mean(Ar)
```

```
## [1] 0.3621898
```

```
betas <- tail(betas,B/2)
betas <- betas[ seq(1,B/2,20) , ]
```

```
par(mfrow=c(4,3))
for( k in 1:12){acf(betas[,k], ylim=c(-.1,.1))}
```


Series betas[, k]
ACF
0.10
0.00
−0.10
0 5 15 25 35
Lag

The thinning has done an adequate job of reducing the autocorrelation.

What variables best predict the quality of the wine?

```
CI <- t(apply(betas , 2, quantile, probs = c(0.5, 0.025, 0.975)))
rownames(CI) <- colnames(X)
CI <- round(CI,6)
CI0 <- (CI[ ,2 ] < 0 & CI[ ,3] > 0 )
CI <-  CI[     !CI0 , ]
CI <- as.data.frame(CI)
CI$dif <- CI[,3]-CI[,2]
CI
```

```
##                             50%        2.5%       97.5%        dif
## (Intercept)          259.932722  129.270643  400.017468 270.746825
## volatile.acidity      -6.487580   -7.276855   -5.678922   1.597933
## residual.sugar         0.171417    0.120666    0.223245   0.102579
## free.sulfur.dioxide    0.009528    0.004210    0.014920   0.010710
## density             -272.979128 -413.636872 -139.756332 273.880540
## pH                     1.087487    0.416702    1.803380   1.386678
## sulphates              1.827809    1.145935    2.513641   1.367706
## alcohol                0.743510    0.564177    0.917048   0.352871
```

The variables that best predict the quality of wine are: `volatile.acidity`, `residual.sugar`, `free.sulfur.dioxide`, `density`, `pH`, `sulphates`, and `alcohol`.

Once you have selected the variables for your final model and run your M-H sampler, run the Normal Approximation to logistic regression setting the seed to 1908, generating 4,000 total samples.

The likelihood for this model is

$$\mathcal{L}(y_i|\beta, x_i) \propto \exp\left[\sum_{i=1}^{n}\left(y_i\left(x_i^T\beta\right) - \log\left[1+\exp\left(x_i^T\beta\right)\right]\right)\right]$$

for $\beta = \begin{bmatrix}\beta_0 & \beta_1 & \dots & \beta_7\end{bmatrix}^T$ and $x_i^T = \begin{bmatrix}1 & x_{1i} & \dots & x_{7i}\end{bmatrix}^T$

We take the joint prior on β to be flat, $\pi(\beta) \propto 1$, the posterior has the same form.

$$P(\beta|y_i, x_i) \propto \exp\left[\sum_{i=1}^{n}\left(y_i\left(x_i^T\beta\right) - \log\left[1+\exp\left(x_i^T\beta\right)\right]\right)\right]$$

There is an approximate Normal Posterior, so the posterior of β can then be sampled from

$$\beta|y \sim N\left(\hat{\beta}, V_\beta\right)$$

Where V_β is the last working variance from the iterative solution to the posterior mode.

```r
library(mvtnorm)
bhat <- coef(fit)[ which( names( coef(fit) ) %in% rownames(CI) ) ];bhat
```

```
##         (Intercept)    volatile.acidity      residual.sugar
##        2.582369e+02       -6.458963e+00        1.700658e-01
## free.sulfur.dioxide             density                  pH
##        9.600953e-03       -2.708743e+02        1.089958e+00
##           sulphates             alcohol
##        1.797398e+00        7.429412e-01
```

```r
# names( which( (summary(fit)$coeff[-1,4] < 0.05) == T) ) == names(bhat)[-1]

vbetas <- vbeta[,which( names( coef(fit) ) %in% rownames(CI) ) ]
vbetas <- vbetas[ which( names( coef(fit) ) %in% rownames(CI) ), ]

set.seed(1908)
Betas   <- rmvnorm(4000, mean = bhat, sigma = vbetas); dim(Betas)
```

```
## [1] 4000    8
```

```r
lna <- round( t(apply(Betas, 2, quantile, probs = c(0.5, 0.025, 0.975))) , 6)
lna <- as.data.frame(lna)
lna$dif <- lna[,3]-lna[,2]
lna;CI
```

```
##                             50%        2.5%       97.5%        dif
## (Intercept)          255.957982  122.668004  399.791277 277.123273
## volatile.acidity      -6.460959   -7.260500   -5.646162   1.614338
## residual.sugar         0.169361    0.118628    0.224587   0.105959
## free.sulfur.dioxide    0.009613    0.004366    0.014974   0.010608
## density             -268.602069 -414.566627 -133.420567 281.146060
## pH                     1.092846    0.389139    1.818357   1.429218
## sulphates              1.797095    1.108265    2.496783   1.388518
## alcohol                0.745869    0.562911    0.922651   0.359740
```

```
##                             50%        2.5%       97.5%        dif
## (Intercept)          259.932722  129.270643  400.017468 270.746825
## volatile.acidity      -6.487580   -7.276855   -5.678922   1.597933
## residual.sugar         0.171417    0.120666    0.223245   0.102579
## free.sulfur.dioxide    0.009528    0.004210    0.014920   0.010710
## density             -272.979128 -413.636872 -139.756332 273.880540
## pH                     1.087487    0.416702    1.803380   1.386678
## sulphates              1.827809    1.145935    2.513641   1.367706
## alcohol                0.743510    0.564177    0.917048   0.352871
```

```r
lna[,1] - CI[,1]
```

```
## [1] -3.974740  0.026621 -0.002056  0.000085  4.377059  0.005359 -0.030714
## [8]  0.002359
```

```r
lna$dif- CI$dif
```

```
## [1]  6.376448  0.016405  0.003380 -0.000102  7.265520  0.042540  0.020812
## [8]  0.006869
```

Compare the results from the two models focusing on the coeffcients and intervals. In particular, does one model tend to have smaller credible intervals? Or do the models give the same/similar results? Comment on any differences you see.

The models are relatively similar in their estimates, with very slight differences in the coefficient estimates. The first dataframe above is the normal approximation and the second one is the M-H sampler. The last column in each dataframe is the width of the credible interval. Subtracting each of the respective widths, we see that the normal approximation produces larger credible intervals for each of the coefficients except for one.