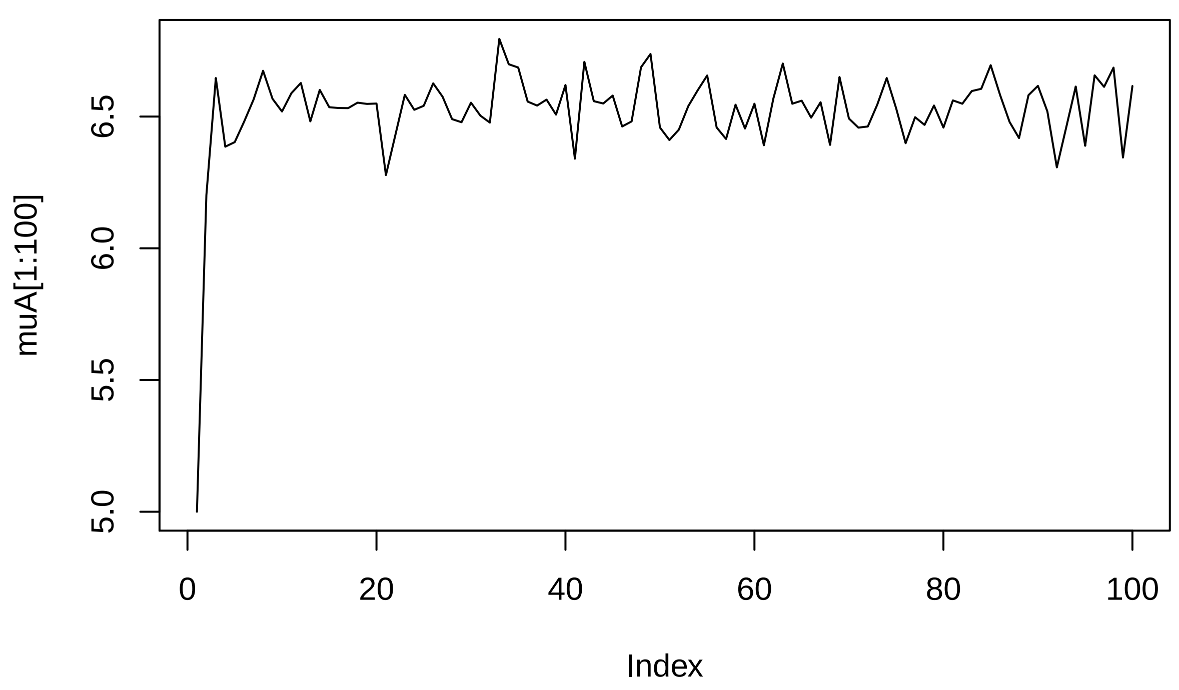
Final Exam S251

Neil Thompson

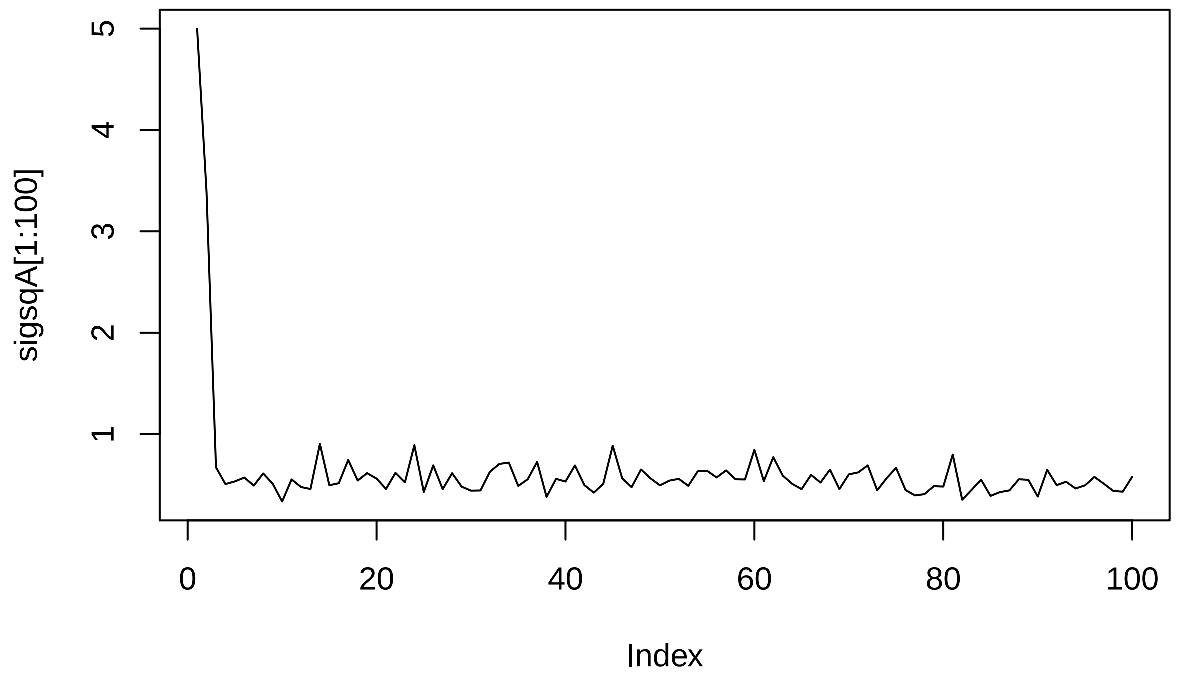
4/16/2021

#1a

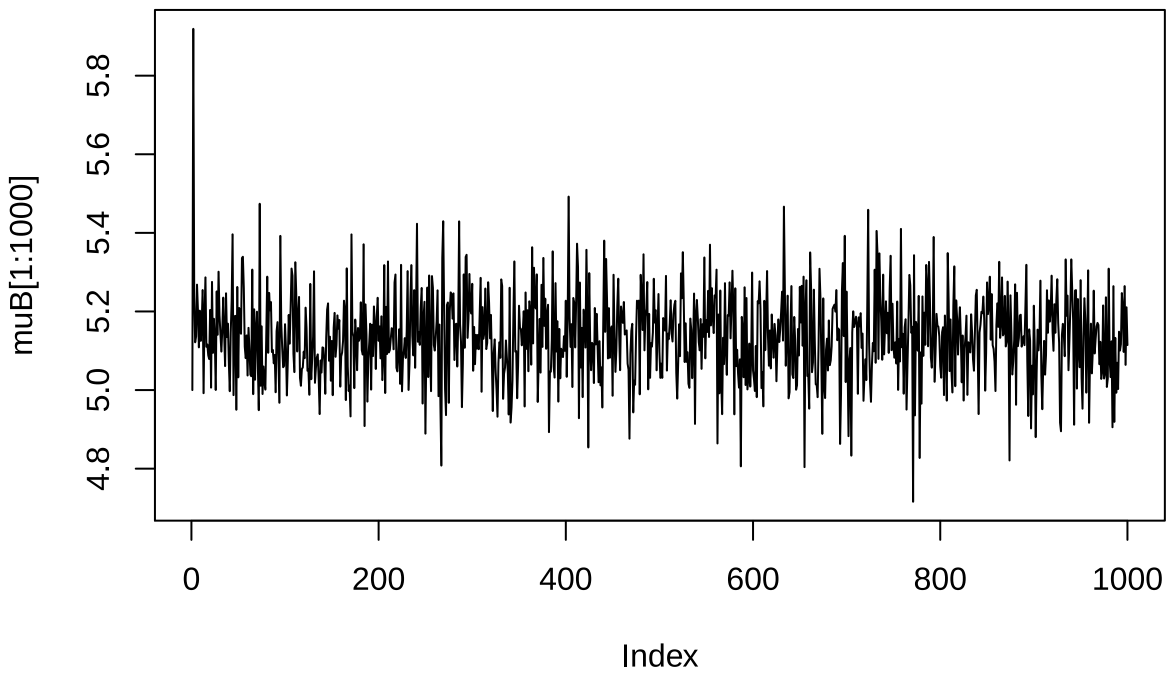
```r
load("Stat251FinalExamDataW21.Rdata")
#Setting prior parameters
lambda <- 5
tau <- 5
gamma <- 1.5
phi <- 2.5
n <- 1e5

#Above
muA <- numeric(n+1)
sigsqA <- numeric(n+1)

muA[1] <- 5
sigsqA[1] <- 5

#Gibbs Sampling
for (i in 1:n){
  lambda1 <- (tau^2*sum(ladder.aboveSS) + sigsqA[i]*lambda)/(tau^2*length(ladder.aboveSS) + sigsqA[i])
  tau1 <- sqrt((sigsqA[i]*tau^2)/(tau^2*length(ladder.aboveSS) + sigsqA[i]))
  muA[i+1] <- rnorm(1, lambda1, tau1)

  gamma1 <- gamma + length(ladder.aboveSS)/2
  phi1 <- phi + sum((ladder.aboveSS-muA[i])^2)/2
  sigsqA[i+1] <- 1/rgamma(1,gamma1,phi1)
}
#Check for burn-in
plot(muA[1:100], type="l")
```



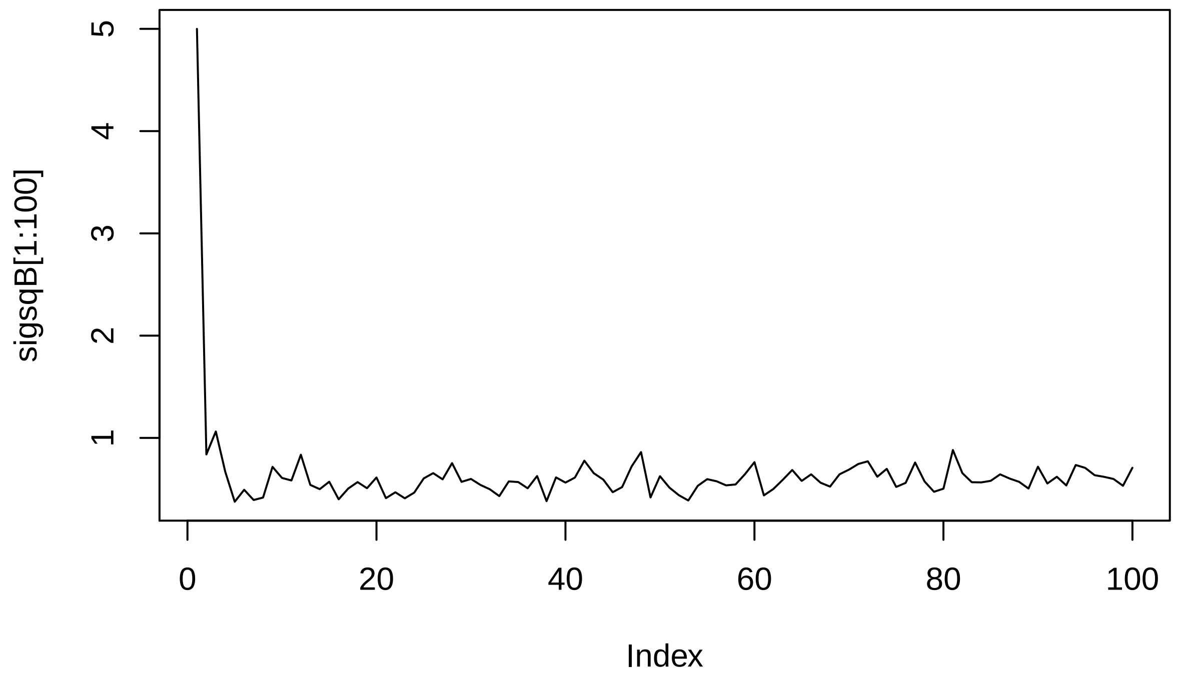
```
plot(sigsqA[1:100],type="l")
```



```r
muA <- muA[-(1:5)]
sigsqA <- sigsqA[-(1:5)]

# Below
muB <- numeric(n+1)
sigsqB <- numeric(n+1)

muB[1] <- 5
sigsqB[1] <- 5

#Gibbs Sampling
for (i in 1:n){
  lambda1 <- (tau^2*sum(ladder.belowSS) + sigsqB[i]*lambda)/(tau^2*length(ladder.belowSS) + sigsqB[i])
  tau1 <- sqrt((sigsqB[i]*tau^2)/(tau^2*length(ladder.belowSS) + sigsqB[i]))
  muB[i+1] <- rnorm(1, lambda1, tau1)

  gamma1 <- gamma + length(ladder.belowSS)/2
  phi1 <- phi + sum((ladder.belowSS-muB[i])^2)/2
  sigsqB[i+1] <- 1/rgamma(1,gamma1,phi1)
}
#Check for burn-in
plot(muB[1:1000], type="l")
```



```
plot(sigsqB[1:100],type="l")
```



```
muB <- muB[-(1:5)]
sigsqB <- sigsqB[-(1:5)]
```

#1b

```
quantile(muA, c(0.025,0.975))
```

```
##     2.5%    97.5%
## 6.314279 6.735354
```

Given our data and prior knowledge, there is a 95% chance that the mean happiness score for the countries whose social support score was at or above the median is between approximately 6.316 and 6.738.

#1c

```
muD <- muA - muB
mean(muD > 0)
```

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

Given our data and prior knowledge, the probability that $\mu_A > \mu_B$ is 1. Hence, the mean of happiness score of counties whose social support score is at or above the median is 100% higher than that of countries whose social support score is below the median.

#1d

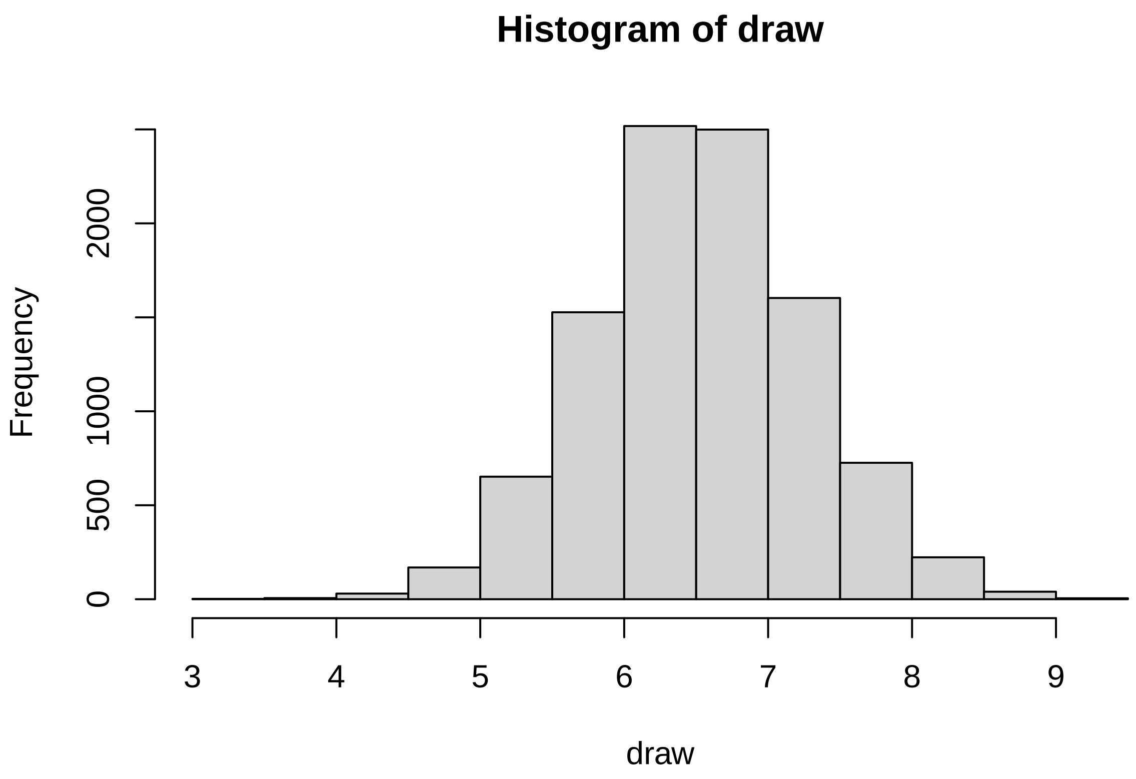
```r
quantile(sigsqA - sigsqB, c(0.025, 0.975))
```

```
##       2.5%      97.5%
## -0.3819571  0.2946230
```

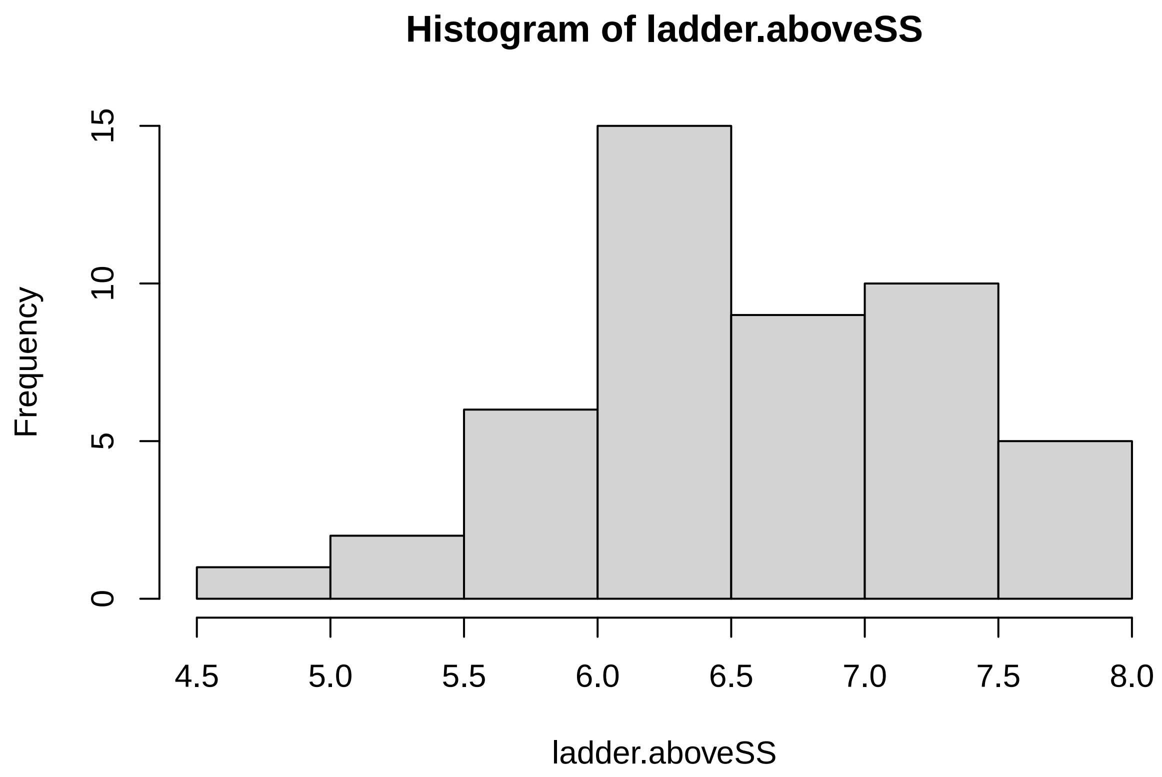
Given data and prior knowledge, there is a 95% chance that the difference between σ_A^2 and σ_B^2 is between -0.383 and 0.298. Since this interval straddles 0, the populations of countries above and below median social support score have similar variances (no significant difference) in their happiness score.

#1e

```r
draw <- numeric(1e4)
for(i in 1:1e4) {
  mu <- sample(muA, 1)
  sd <- sample(sqrt(sigsqA),1)
  draw[i] <- rnorm(1,mu,sd)
}
hist(draw)
```



```r
hist(ladder.aboveSS)
```



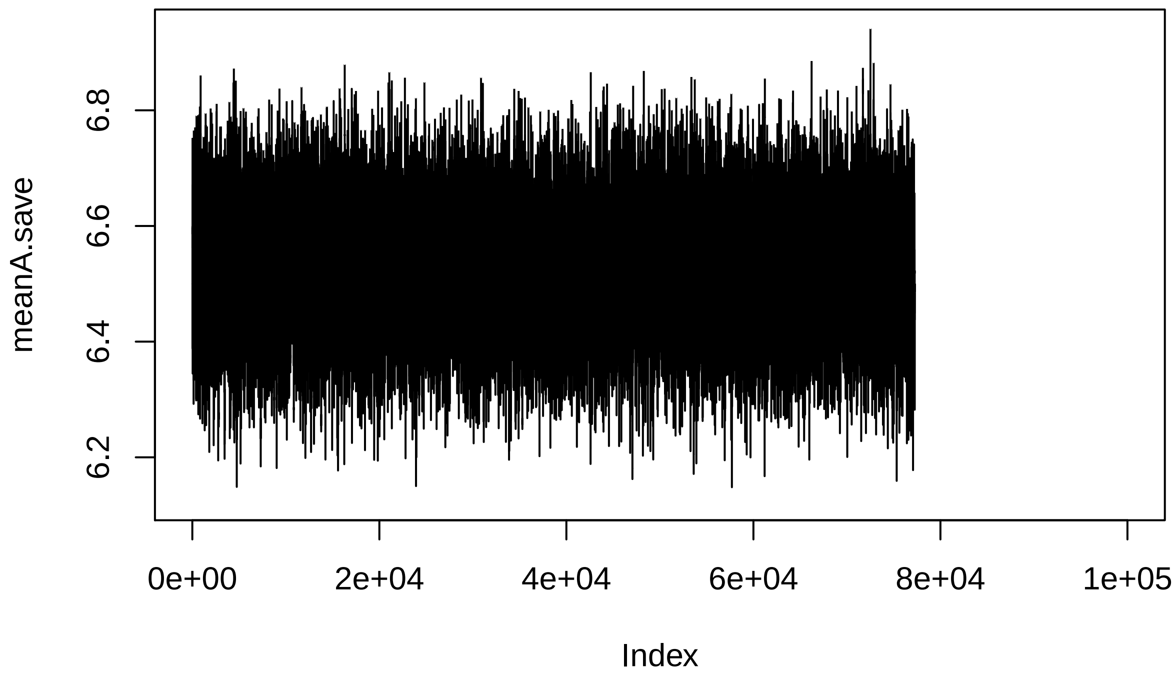
I would feel comfortable using this posterior predictive distribution to predict the the happiness score of another country whose social support score is at or above the median, since the distributions fall in the same range and have a peak at a similar place. The posterior predictive will likely predict something between 5 and 8 and almost surely between 4 and 9, which will look adequately like our data.

#2a

```
varA <- 0.7

iters <- 1e5 #number of iterations
meanA.save <- rep(0, iters) #initialize a vector to save the accepted values of the parameter
meanA <- 6.5 #starting value of parameter
n.accept <- 0 #how many times do we accept the proposed value?
c <- 0.25 #standard deviation of the proposal distribution

#MCMC algorithm (metropolis random walk)
for(i in 1:iters){
    meanA.dot <- rnorm(1, meanA, c)
    if(meanA.dot >=0){
        r <- prod(dnorm(ladder.aboveSS,meanA.dot,varA))*dnorm(meanA.dot, lambda, tau)/(prod(dnorm(ladde
        if(r > 1){
            r <- 1
        }
        accept <- sample(c(T, F), 1, prob=c(r, 1-r))
        if(accept==T){
            meanA <- meanA.dot
            n.accept <- n.accept+1
        }
    }
    meanA.save[i] <- meanA
}

#Acceptance rate
n.accept/iters
```

```
## [1] 0.43537
```

```
#Trace plot
plot(meanA.save, type='l')
```



Acceptance rate of 0.431.

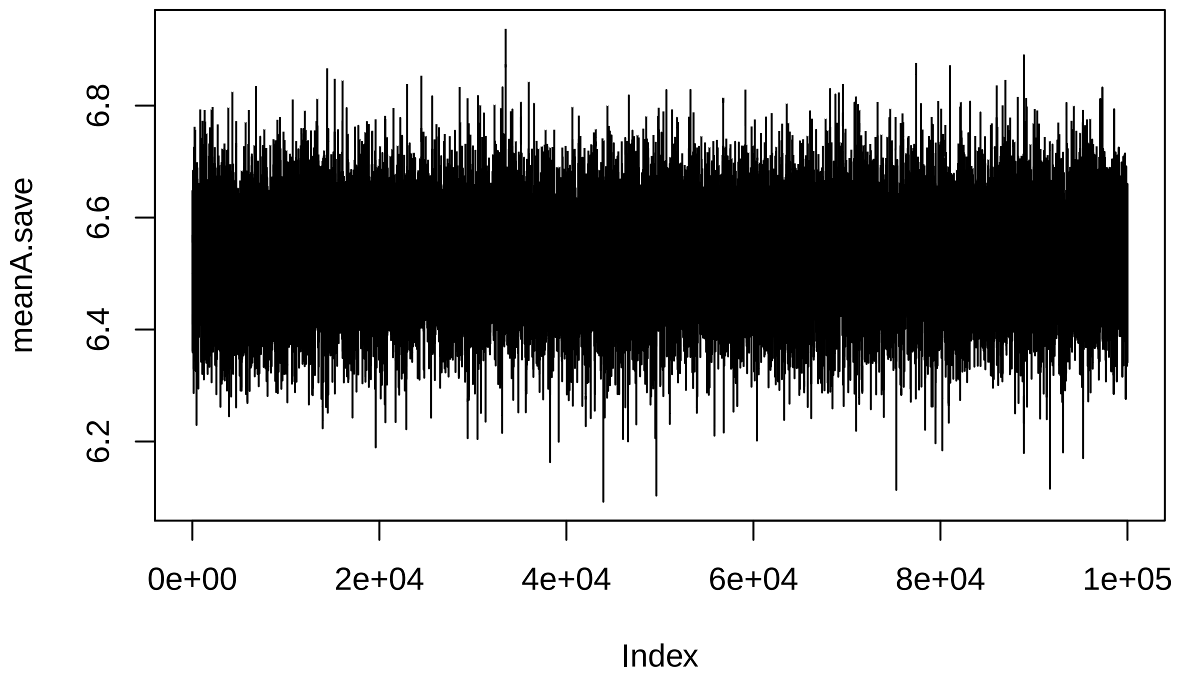
#2b

```
iters <- 1e5 #number of iterations
meanA.save <- rep(0, iters)
varA.save <- rep(0, iters)
meanA <- 6.5 #starting value of parameter
varA <- 0.7
n.accept <- 0 #how many times do we accept the proposed value?
c <- 0.2 #standard deviation of the proposal distribution
```

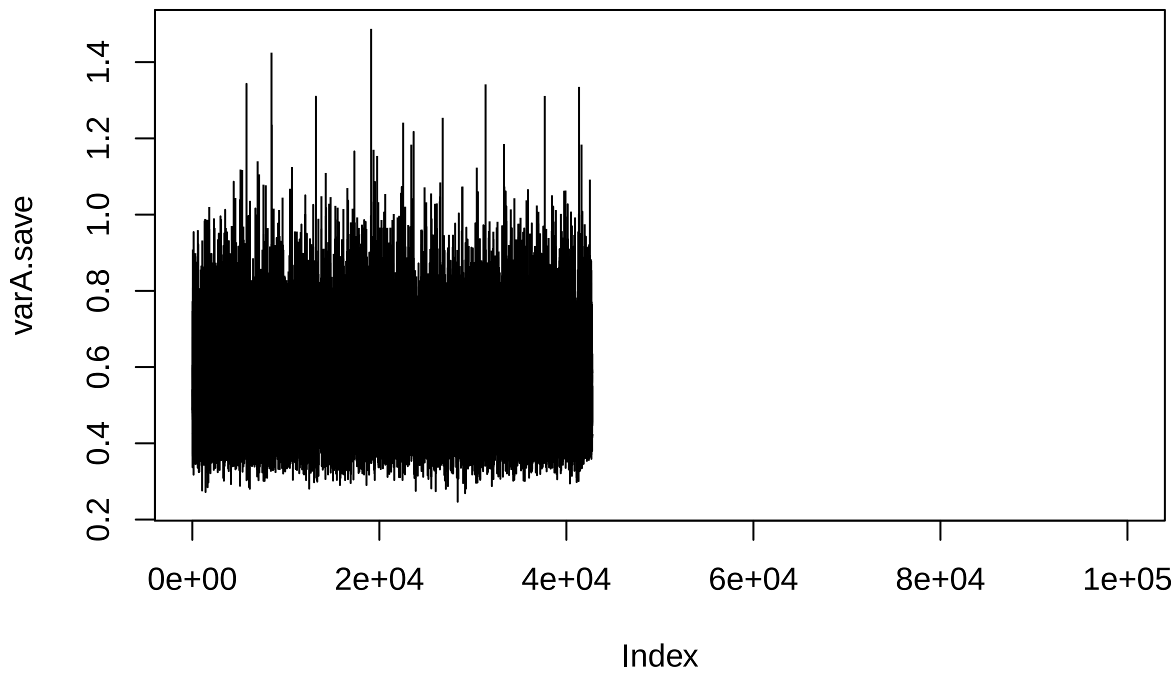
```
#MCMC algorithm (metropolis random walk)
for(i in 1:iters){
    meanA.dot <- rnorm(1, meanA, c)

    gamma1 <- gamma + length(ladder.aboveSS)/2
  phi1 <- phi + sum((ladder.aboveSS-meanA)^2)/2
  varA.dot <- 1/rgamma(1,gamma1,phi1)

    if(meanA.dot >=0){
        r <- prod(dnorm(ladder.aboveSS,meanA.dot,varA.dot))*dnorm(meanA.dot, lambda, tau)/(prod(dnorm(la
        if(r > 1){
            r <- 1
        }
        accept <- sample(c(T, F), 1, prob=c(r, 1-r))
        if(accept==T){
            meanA <- meanA.dot
            n.accept <- n.accept+1
        }
    }
    meanA.save[i] <- meanA
    varA.save[i] <- varA.dot
}

#Acceptance rate
n.accept/iters
```

```
## [1] 0.42305
```

```
#Trace plot of the mean
plot(meanA.save, type='l')
```



```r
#Trace plot of the variance
plot(varA.save, type='l')
```



Acceptance rate is 0.425.

#2c

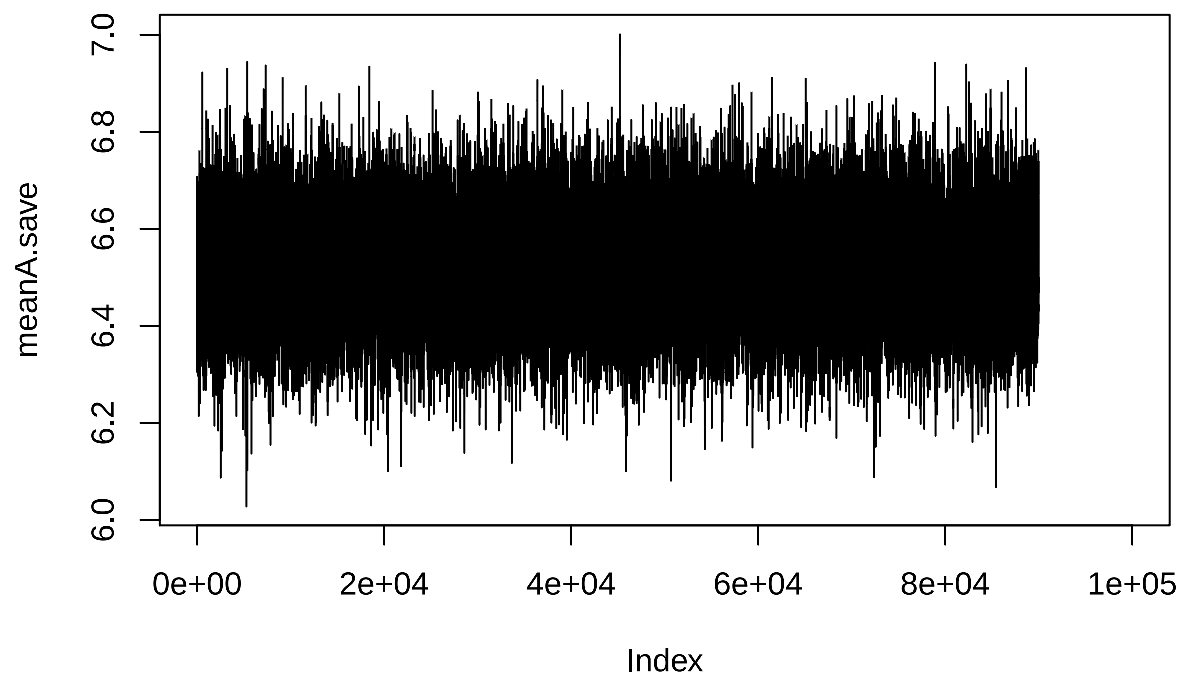
```r
iters <- 1e5 #number of iterations
meanA.save <- rep(0, iters)
varA.save <- rep(0, iters)
meanA <- 6.5 #starting value of parameter
varA <- 0.7
n.accept <- 0 #how many times do we accept the proposed value?
n.acceptv <- 0
c <- 0.25 #standard deviation of the proposal distribution
cv <- 0.2


#MCMC algorithm (metropolis random walk)
for(i in 1:iters){
    meanA.dot <- rnorm(1, meanA, c)
    varA.dot <- rnorm(1, varA, cv)
  if(varA.dot >=0){
        rV <- prod(dnorm(ladder.aboveSS,meanA,varA.dot))*dgamma(1/varA.dot, gamma, phi)/(prod(dnorm(lad
        if(rV > 1){
            rV <- 1
        }
        accept <- sample(c(T, F), 1, prob=c(rV, 1-rV))
        if(accept==T){
            varA <- varA.dot
            n.acceptv <- n.acceptv+1
        }
    }

    if(meanA.dot >=0){
        r <- prod(dnorm(ladder.aboveSS,meanA.dot,varA))*dnorm(meanA.dot, lambda, tau)/(prod(dnorm(ladde
        if(r > 1){
            r <- 1
        }
        accept <- sample(c(T, F), 1, prob=c(r, 1-r))
        if(accept==T){
            meanA <- meanA.dot
            n.accept <- n.accept+1
        }
    }
    meanA.save[i] <- meanA
    varA.save[i] <- varA
}

#Acceptance rate of the mean
n.accept/iters
```

```
## [1] 0.4386
```

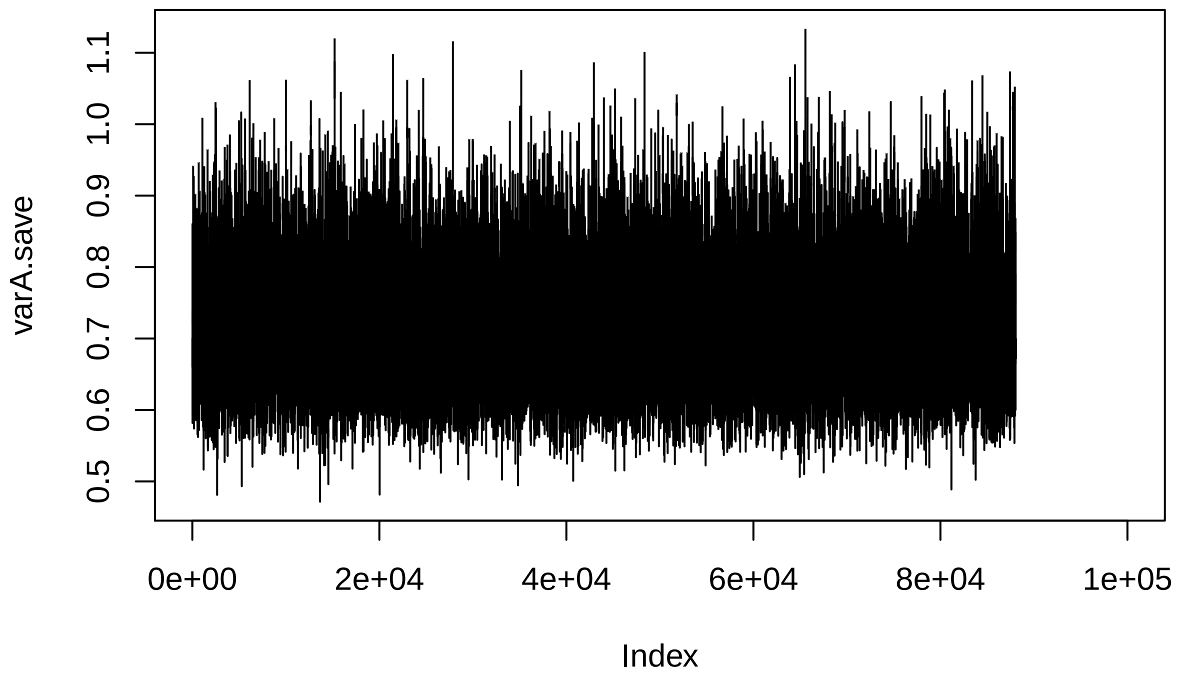
```
#Acceptance rate of the variance
n.acceptv/iters
```

```
## [1] 0.40695
```

```
#Trace plot of the mean
plot(meanA.save, type='l')
```



```r
#Trace plot of the variance
plot(varA.save, type='l')
```



#2d

Two parameter general Metropolis-within-Gibbs Algorithm

Step 0: Select two known proposal distribution for two different parameters, $g(\theta_1|\theta_1^{[t-1]}, s_1)$ and $h(\theta_2|\theta_2^{[t-1]}, s_2)$. Set a starting value for θ_1 and θ_2, call them $\theta_1^{[0]}$ and $\theta_2^{[0]}$ respecively.

Step 1: Set t = 1

Step 2: Draw values of θ_1 and θ_2 from $g(\theta_1|\theta_1^{[t-1]}, s_1)$ and $h(\theta_2|\theta_2^{[t-1]}, s_2)$. Call these values θ_1^* and θ_2^*.

Step 3: Compute the acceptance probabilities at θ_1^* and θ_2^*, where the acceptance probability for each parameter is the posterior probability of θ_1^* or θ_2^* divided by the acceptance probability of θ_1 or θ_2 respectively. The posterior probabilities are calculated as the prior times the likelihood, where the likelihood uses the previously set values of the other θ as the other parameter.

Step 4: For each parameter, set the new value of θ equal to θ^* with probability as the acceptance probability calculated previously; otherwise set θ to its previous value.

Step 5: Set t = t + 1

Step 6: Repeat Steps 2 - 5 many times