Ec/ACM/CS 112 - PS6

Ben Juarez

Question 1

preliminaries

```
rm(list=ls())
set.seed(123)
library(LaplacesDemon)
library(coda)
```

Step 2: Program the Gibbs sampler

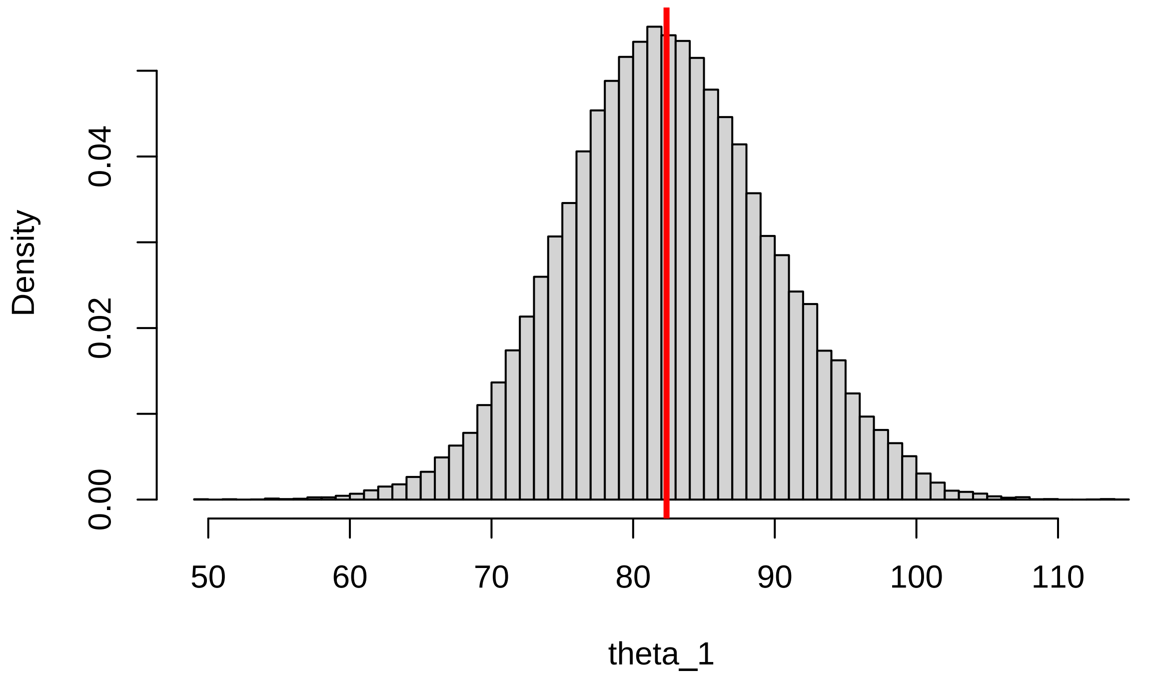
```
GibbsSampling = function(paramInit, y1, y2, y3, y4, y5, y6, nSamples) {
  n1 = length(y1)
  n2 = length(y2)
  n3 = length(y3)
  n4 = length(y4)
  n5 = length(y5)
  n6 = length(y6)
  nTot = n1 + n2 + n3 + n4 + n5 + n6
  nDim = length(paramInit)
  paramSamples = array(rep(NA, nSamples * nDim), dim = c(nSamples,nDim))
  theta_hat = function(mu, sigma, tau, n, y) {
    numerator = mu/tau^2 + (n*mean(y))/sigma^2
    denom = (1/tau^2) + (n/sigma^2)
    return(numerator/denom)
  }
  sd_theta = function(sigma, tau, n) {
    denom = (1/tau^2) + (n/sigma^2)
    return(sqrt(1/denom))
  }
  paramLast = paramInit
  for (t in 1:nSamples) {
    theta_1_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
                                      tau = paramLast[9],n = n1,y = y1),
                          sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n1))
    theta_2_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
                                      tau = paramLast[9],n = n2,y = y2),
                          sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n2))
    theta_3_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
                                      tau = paramLast[9],n = n3,y = y3),
                          sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n3))
    theta_4_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
```

```
                                         tau = paramLast[9],n = n4,y = y4),
                               sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n4))
    theta_5_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
                                         tau = paramLast[9],n = n5,y = y5),
                               sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n5))
    theta_6_Inner = rnorm(1,theta_hat(mu = paramLast[7],sigma = paramLast[8],
                                         tau = paramLast[9],n = n6,y = y6),
                               sd_theta(sigma = paramLast[8],tau = paramLast[9],n = n6))
    theta_Inner = c(theta_1_Inner, theta_2_Inner, theta_3_Inner, theta_4_Inner,
                    theta_5_Inner, theta_6_Inner)
    mu_Inner = rnorm(1, mean(theta_Inner), paramLast[9] / sqrt(nGroups))
    sampVar_1 = sum((y1 - theta_1_Inner)^2)
    sampVar_2 = sum((y2 - theta_2_Inner)^2)
    sampVar_3 = sum((y3 - theta_3_Inner)^2)
    sampVar_4 = sum((y4 - theta_4_Inner)^2)
    sampVar_5 = sum((y5 - theta_5_Inner)^2)
    sampVar_6 = sum((y6 - theta_6_Inner)^2)
    sigma_Hat_2 = (sampVar_1 + sampVar_2 + sampVar_3 + sampVar_4
                   + sampVar_5 + sampVar_6) / nTot
    sigma_2_Inner = rinvchisq(1, nTot, sigma_Hat_2)
    tau_Hat_2 = sum((theta_Inner - mu_Inner)^2) / (nGroups - 1)
    tau_2_Inner = rinvchisq(1, nGroups - 1, tau_Hat_2)
    paramSamples[t,] = c(theta_1_Inner, theta_2_Inner, theta_3_Inner, theta_4_Inner,
                         theta_5_Inner, theta_6_Inner, mu_Inner,
                         sqrt(sigma_2_Inner), sqrt(tau_2_Inner))
    paramLast = paramSamples[t,]
  }
  return(paramSamples)
}
```

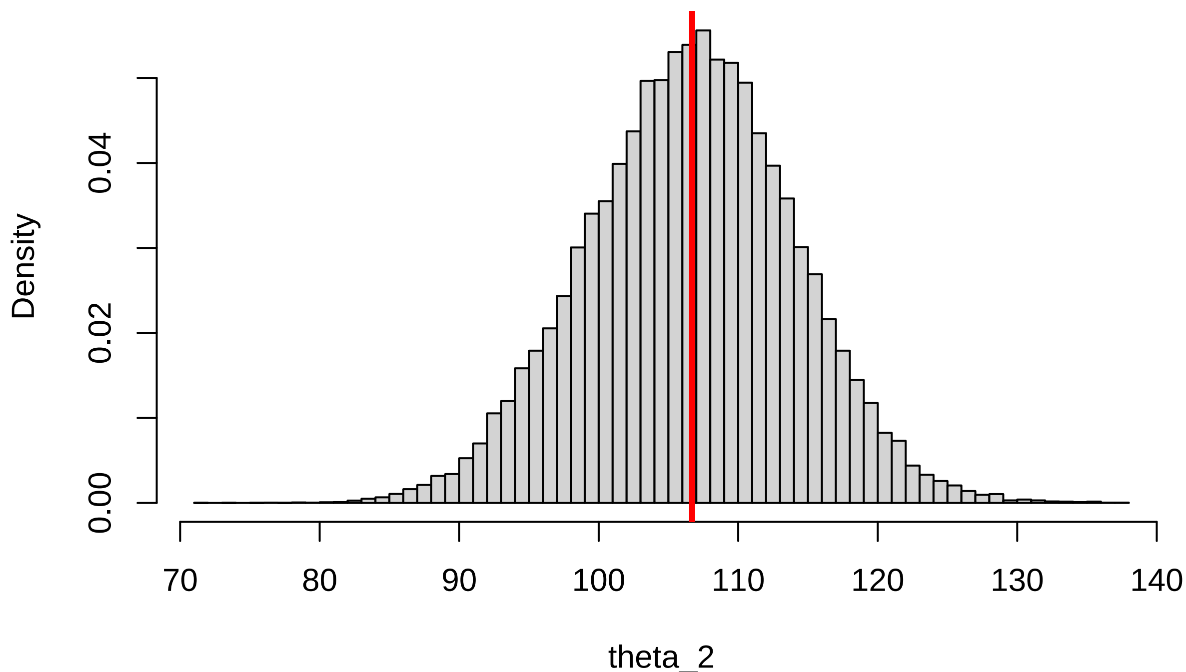
step 3: Sampling and diagnostics

```
nGroups = 6
y1 = c(83,92,92,46)
y2 = c(117,109,114,104)
y3 = c(101,93,92,86)
y4 = c(105,119,116,102)
y5 = c(79,97,103,79)
y6 = c(57,92,104,77)
nSamples = 100000
paramInit = function() {
  theta_1 = sample(y1, 1)
  theta_2 = sample(y2, 1)
  theta_3 = sample(y3, 1)
  theta_4 = sample(y4, 1)
  theta_5 = sample(y5, 1)
  theta_6 = sample(y6, 1)
  mu = mean(c(y1, y2, y3, y4, y5, y6))
  sigma = runif(1, 0.1, 5)
  tau = runif(1,0.1, 5)
  return(c(theta_1, theta_2, theta_3, theta_4, theta_5, theta_6, mu, sigma, tau))
}
postSamples = GibbsSampling(paramInit(), y1, y2, y3, y4, y5, y6, nSamples)
```

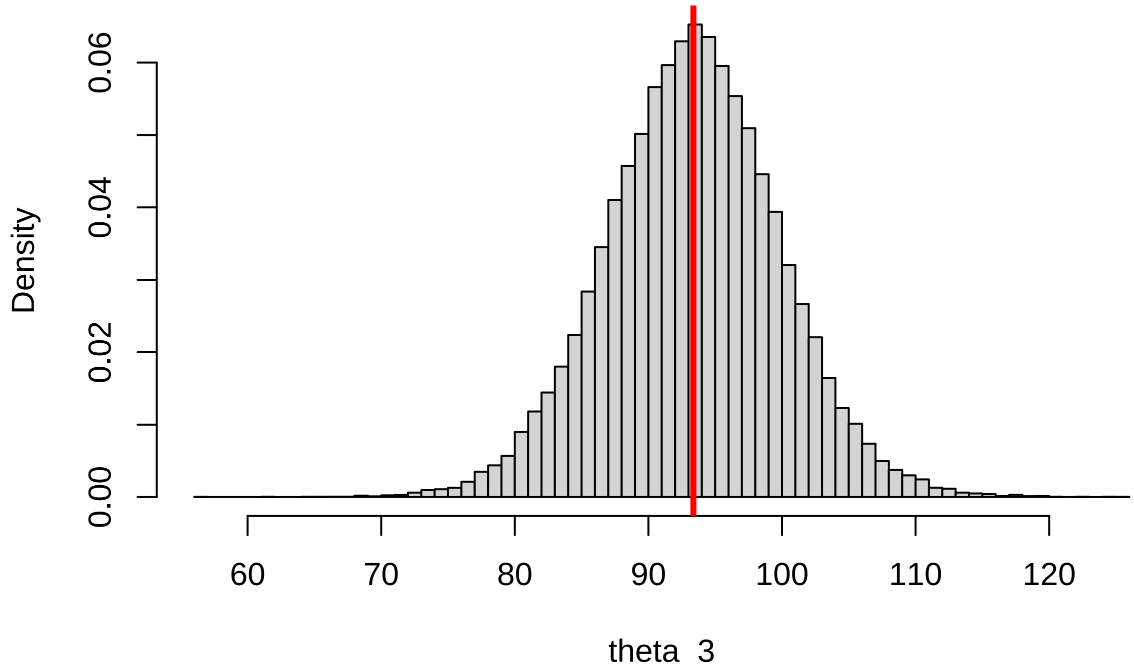
```
postSamples = postSamples[(nSamples/2 + 1):nSamples,]

hist(postSamples[,1], xlab="theta_1", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,1]), col="red", lwd=3)
```



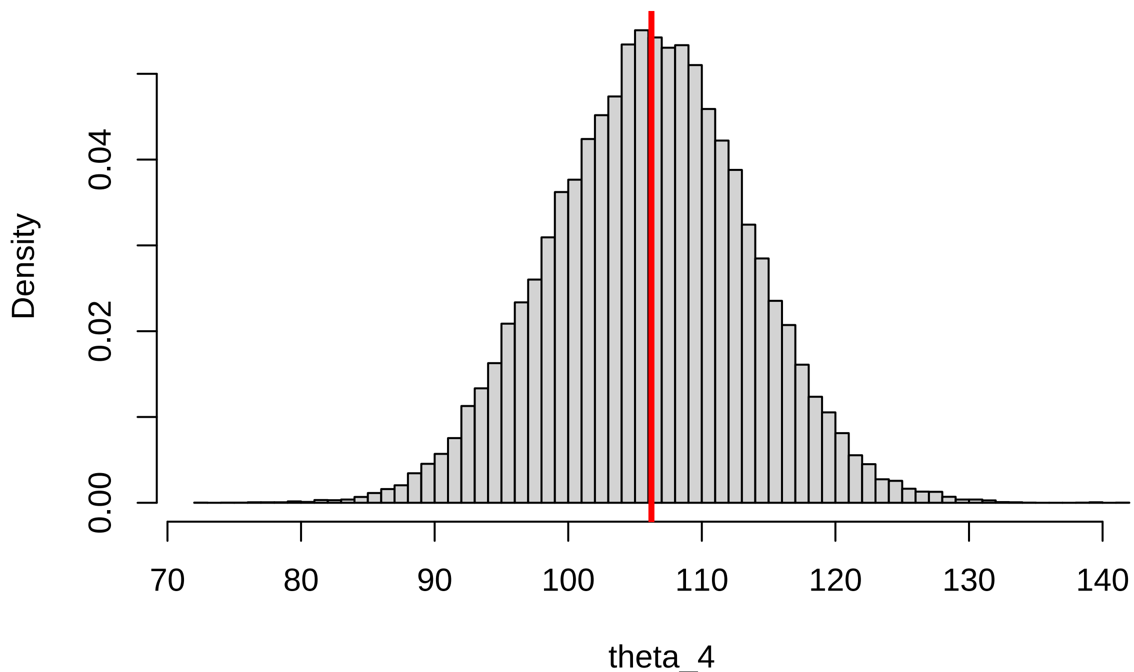

```
hist(postSamples[,2], xlab="theta_2", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,2]), col="red", lwd=3)
```



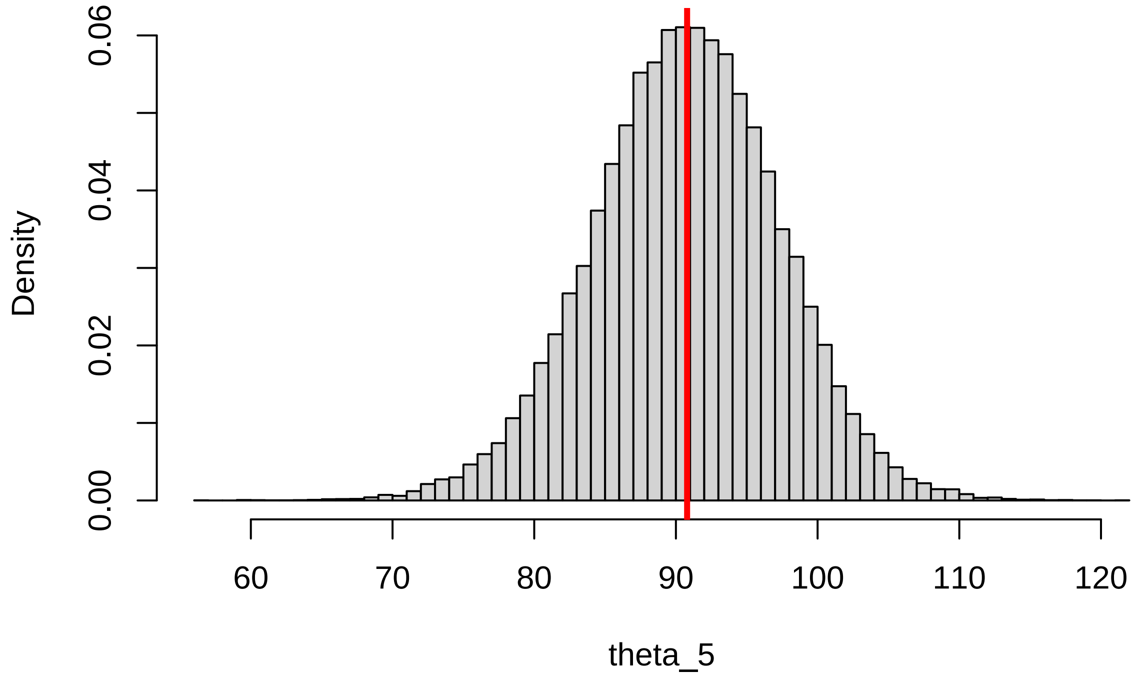

```
hist(postSamples[,3], xlab="theta_3", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,3]), col="red", lwd=3)
```



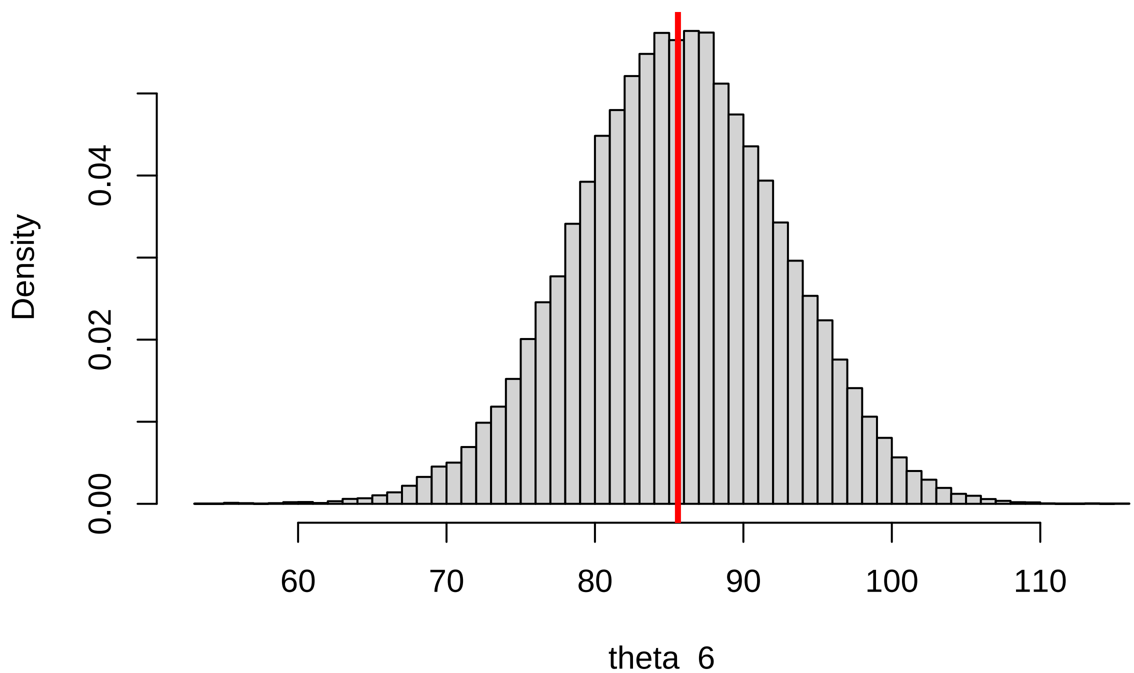

```r
hist(postSamples[,4], xlab="theta_4", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,4]), col="red", lwd=3)
```



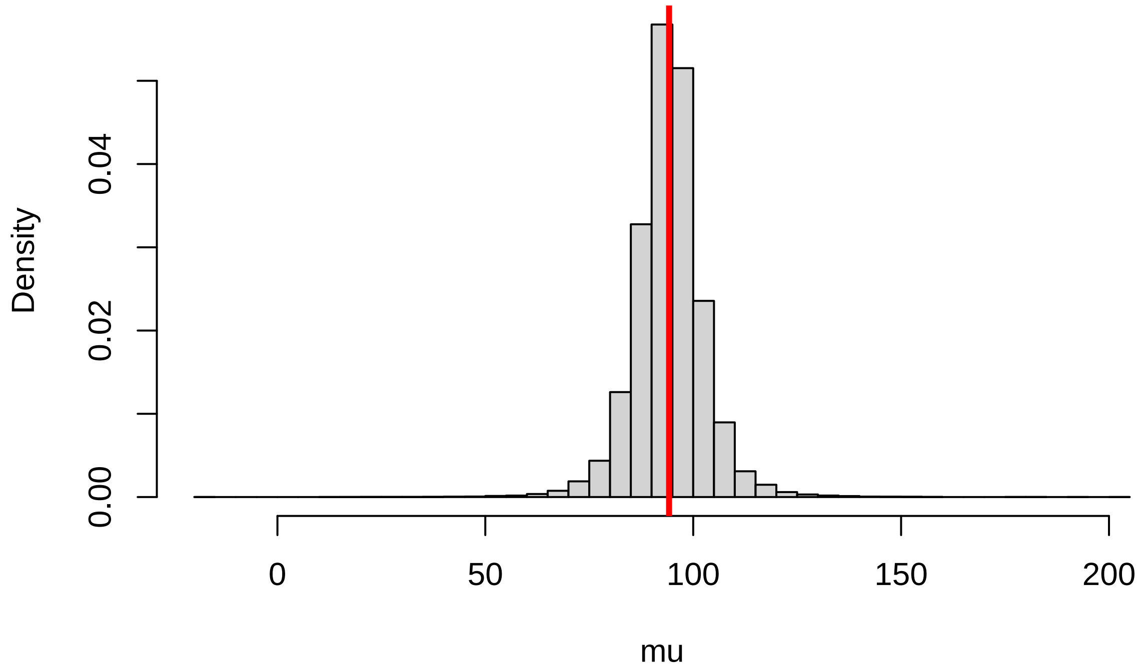

```r
hist(postSamples[,5], xlab="theta_5", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,5]), col="red", lwd=3)
```



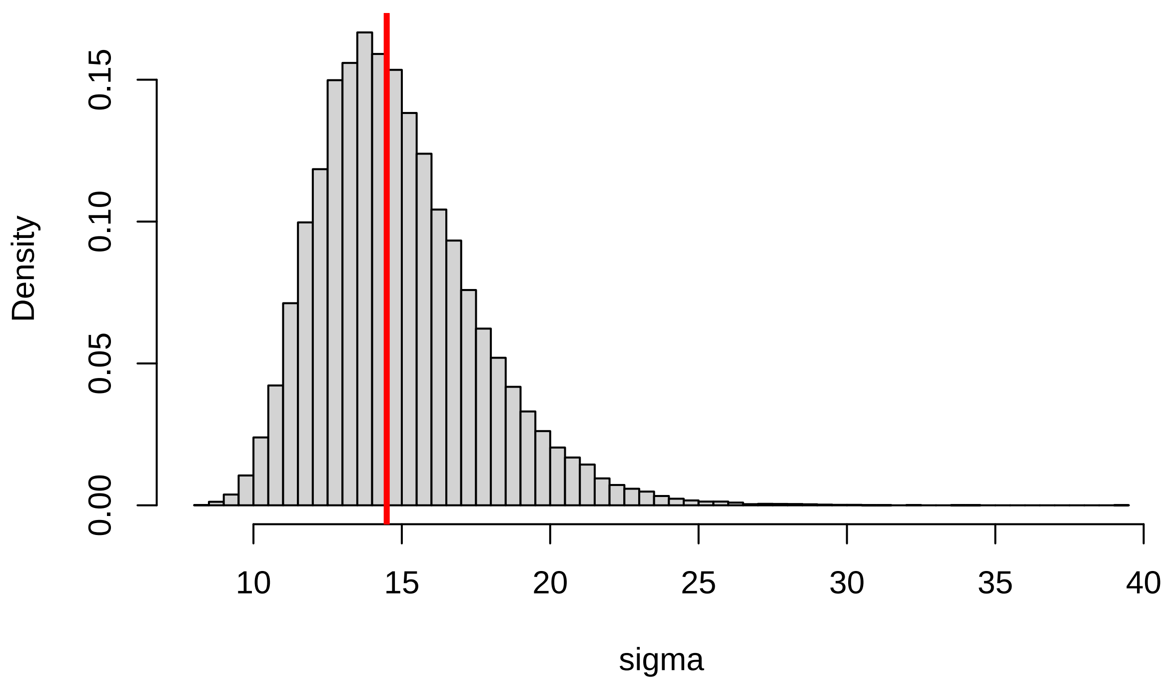

```
hist(postSamples[,6], xlab="theta_6", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,6]), col="red", lwd=3)
```



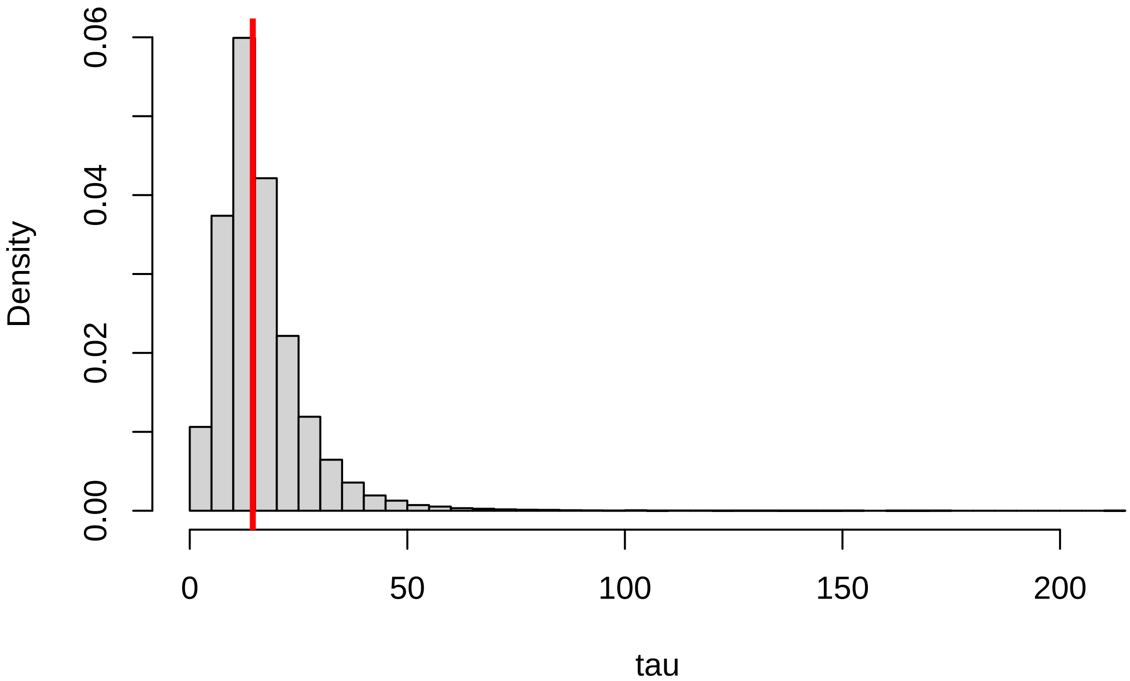

```
hist(postSamples[,7], xlab="mu", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,7]), col="red", lwd=3)
```




```r
hist(postSamples[,8], xlab="sigma", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,8]), col="red", lwd=3)
```




```r
hist(postSamples[,9], xlab="tau", breaks=50, prob=TRUE, main="")
abline(v = median(postSamples[,8]), col="red", lwd=3)
```



Looks like Markov chain is sampling properly. Theta parameters have converged quite well to normal distributions.

step 4: marginal posterior summary statistics

```
library(knitr)
data = matrix(c(quantile(postSamples[,1], 0.01),
                quantile(postSamples[,1], 0.05),
                mean(postSamples[,1]),median(postSamples[,1]),
                quantile(postSamples[,1], 0.95),
                quantile(postSamples[,1], 0.99),
                quantile(postSamples[,2], 0.01),
                quantile(postSamples[,2], 0.05),
                mean(postSamples[,2]),median(postSamples[,2]),
                quantile(postSamples[,2], 0.95),
                quantile(postSamples[,2], 0.99),
                quantile(postSamples[,3], 0.01),
                quantile(postSamples[,3], 0.05),
                mean(postSamples[,3]),median(postSamples[,3]),
                quantile(postSamples[,3], 0.95),
                quantile(postSamples[,3], 0.99),
                quantile(postSamples[,4], 0.01),
                quantile(postSamples[,4], 0.05),
                mean(postSamples[,4]),median(postSamples[,4]),
                quantile(postSamples[,4], 0.95),
                quantile(postSamples[,4], 0.99),
                quantile(postSamples[,5], 0.01),
                quantile(postSamples[,5], 0.05),
                mean(postSamples[,5]),median(postSamples[,5]),
                quantile(postSamples[,5], 0.95),
                quantile(postSamples[,5], 0.99),
                quantile(postSamples[,6], 0.01),
```

```r
                 quantile(postSamples[,6], 0.05),
                 mean(postSamples[,6]),median(postSamples[,6]),
                 quantile(postSamples[,6], 0.95),
                 quantile(postSamples[,6], 0.99),
                 quantile(postSamples[,7], 0.01),
                 quantile(postSamples[,7], 0.05),
                 mean(postSamples[,7]),median(postSamples[,7]),
                 quantile(postSamples[,7], 0.95),
                 quantile(postSamples[,7], 0.99),
                 quantile(postSamples[,8], 0.01),
                 quantile(postSamples[,8], 0.05),
                 mean(postSamples[,8]),median(postSamples[,8]),
                 quantile(postSamples[,8], 0.95),
                 quantile(postSamples[,8], 0.99),
                 quantile(postSamples[,9], 0.01),
                 quantile(postSamples[,9], 0.05),
                 mean(postSamples[,9]),median(postSamples[,9]),
                 quantile(postSamples[,9], 0.95),
                 quantile(postSamples[,9], 0.99)), ncol = 6, byrow = TRUE)
colnames(data) <- c('1%-quantile', '5%-quantile', 'mean', 'median', '95%-quantile',
                   '99%-quantile')
rownames(data) <- c('theta_1', 'theta_2', 'theta_3', 'theta_4', 'theta_5', 'theta_6',
                   'mu', 'sigma', 'tau')
t = as.table(data)
kable(t)
```

	1%-quantile	5%-quantile	mean	median	95%-quantile	99%-quantile
theta_1	65.287430	70.613362	82.52412	82.35844	95.03721	99.83041
theta_2	89.087264	94.134249	106.56796	106.69873	118.62369	123.86688
theta_3	77.835659	82.617471	93.34225	93.36741	103.91076	109.09174
theta_4	88.866137	93.790834	106.12940	106.22806	118.18398	123.51455
theta_5	74.428433	79.737991	90.71289	90.78935	101.28891	106.00111
theta_6	68.920311	74.137801	85.58360	85.59126	97.06357	101.56061
mu	71.443483	81.359559	94.17198	94.19404	107.17510	117.44062
sigma	10.118133	11.151167	14.86056	14.49052	19.85487	22.89024
tau	0.886097	4.816062	16.41065	14.30932	34.42551	53.97245

step 5: compute P(theta_5 < theta_1)

```r
print(paste("P(theta_5 < theta_1) =",
            sum(postSamples[,5] < postSamples[,1])/
              length(postSamples[,5] < postSamples[,1])))
```

```
## [1] "P(theta_5 < theta_1) = 0.18354"
```

Question 2

prelimns

```r
set.seed(123)
library(mvtnorm)
```

```
##
## Attaching package: 'mvtnorm'
## The following objects are masked from 'package:LaplacesDemon':
##
##     dmvt, rmvt
```

```r
library(LaplacesDemon)
library(coda)
```

step 1: function to simulate data

```r
simData = function(numBooks, numSubjects, mu, tau_sq, sigma_sq) {
  meanEffects = rnorm(n = numBooks, mean = mu, sd = tau_sq)
  scores = rmvnorm(n = numSubjects, mean = meanEffects, sigma = diag(numBooks)*sigma_sq)
  return(list(scores, meanEffects))
}
```

step 2: function to estimate posteriors usign Gibbs sampling

```r
GibbsSampling2 = function(paramInit, y, nSamples) {
  n = c()
  l = ncol(y)
  for (i in 1:l) {
    n[i] = length(y[,i])
  }
  nTot = sum(n)
  nDim = length(paramInit)
  paramSamples = array(rep(NA, nSamples * nDim), dim = c(nSamples, nDim))

  theta_hat = function(mu, sigma, tau, n, y) {
    numerator = mu/tau^2 + (n*mean(y))/sigma^2
    denom = (1/tau^2) + (n/sigma^2)
    return(numerator/denom)
  }
  sd_theta = function(sigma, tau, n) {
    denom = (1/tau^2) + (n/sigma^2)
    return(sqrt(1/denom))
  }
  paramLast = paramInit
  for (t in 1:nSamples) {
    theta_Inner = c()
    for (m in 1:l) {
      theta_Inner[m] = rnorm(1, theta_hat(mu = paramLast[l+1],
                                          sigma = paramLast[l+2],
                                          tau = paramLast[l+3],
                                          n = n[m],
                                          y = y[,m]),
                             sd_theta(sigma = paramLast[l+2],
                                      tau = paramLast[l+3],
                                      n = n[m]))
    }
    mu_Inner = rnorm(1, mean(theta_Inner), paramLast[l+3]/sqrt(nGroups))
    sampVar = c()
```

```
    for (m in 1:l) {
      sampVar[m] = sum((y[,m]-theta_Inner[m])^2)
    }
    sigma_Hat_2 = sum(sampVar)/nTot
    sigma_2_Inner = rinvchisq(1, nTot, sigma_Hat_2)
    tau_Hat_2 = sum((theta_Inner - mu_Inner)^2) / (nGroups - 1)
    tau_2_Inner = rinvchisq(1, nGroups - 1, tau_Hat_2)
    paramSamples[t,] = c(theta_Inner, mu_Inner, sqrt(sigma_2_Inner), sqrt(tau_2_Inner))
    paramLast = paramSamples[t,]
  }
  return(paramSamples)
}

pSamples = function(data) {
  nGroups = ncol(data)
  nSamples = 10000
  paramInit = function() {
    theta = c()
    for (i in 1:nGroups) {
      theta[i] = sample(data[,i],1)
    }
    mu = mean(data)
    sigma = runif(1,0.1,5)
    tau = runif(1,0.1,5)
    return(c(theta, mu, sigma, tau))
  }
  postSamples = GibbsSampling2(paramInit(), data, nSamples)
  postSamples = postSamples[(nSamples/2 + 1):nSamples,]
  return(postSamples)
}
```

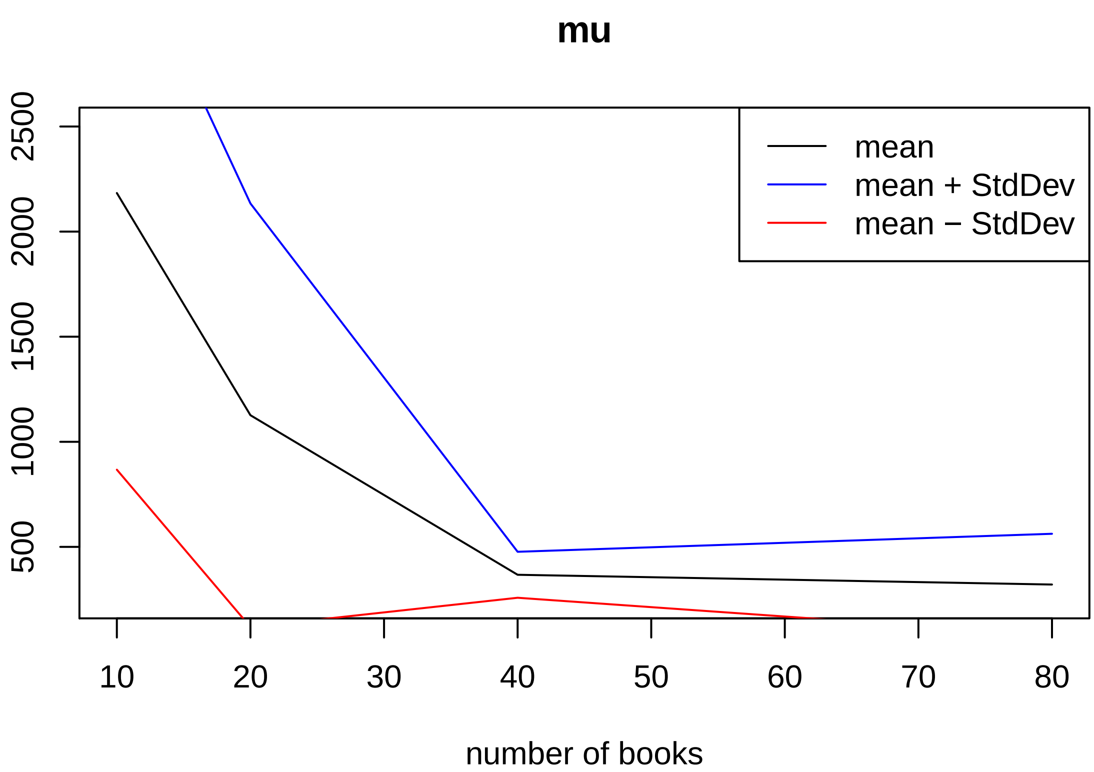
step 3: function to compute expected square errors

```
ESE = function(trueParams, postSamples) {
  true_means = trueParams[1:(length(trueParams)-3)]
  mu = trueParams[length(trueParams)-2]
  sigma_sq = trueParams[length(trueParams)-1]
  tau_sq = trueParams[length(trueParams)]

  ncol = ncol(postSamples)

  sample_mu = sample(postSamples[, ncol-2], 10000, replace=TRUE)
  sample_sigma_sq = sample(postSamples[, ncol-1], 10000, replace=TRUE)
  sample_tau_sq = sample(postSamples[, ncol], 10000, replace=TRUE)
  ese_mu = sum((sample_mu - mu)^2)/10000
  ese_tau_sq = sum((sample_tau_sq - tau_sq)^2)/10000
  ese_sigma_sq = sum((sample_sigma_sq - sigma_sq)^2)/10000

  avg_ese_theta = c()
  for (i in 1:length(true_means)) {
    theta = sample(postSamples[,i], 10000, replace=TRUE)
    avg_ese_theta[i] = sum((theta - true_means[i])^2)/10000
  }
```

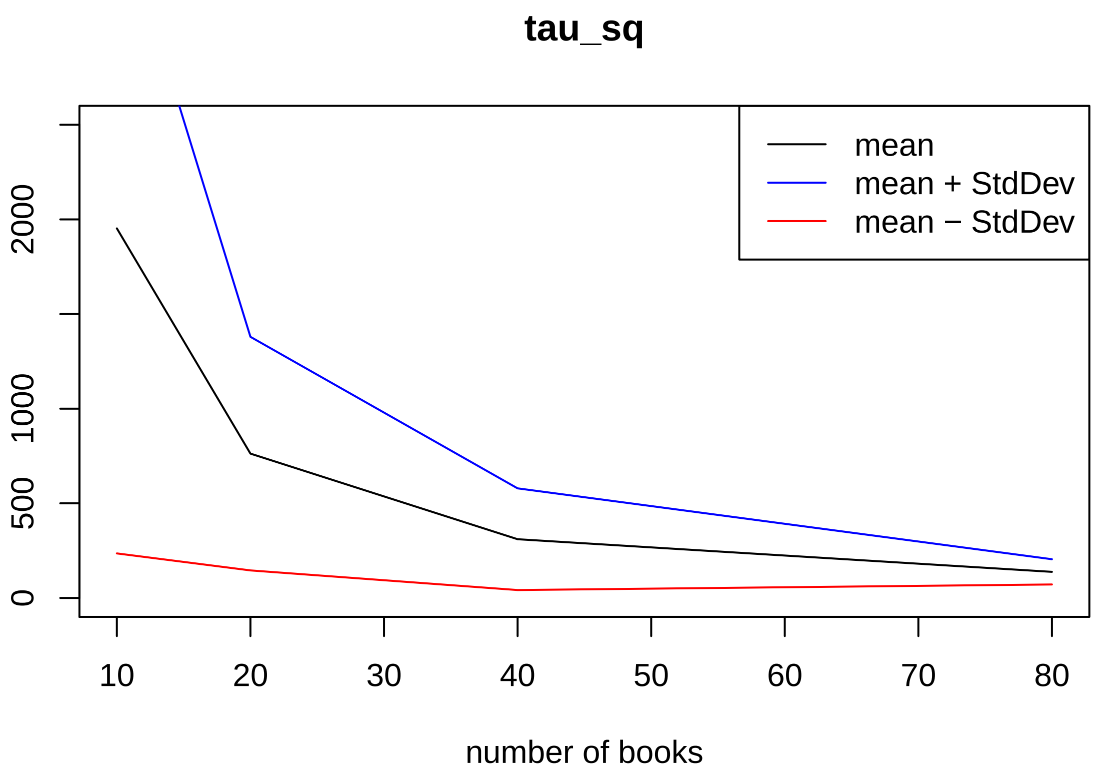
```
  return(list(ese_mu, ese_tau_sq, ese_sigma_sq, avg_ese_theta))
}
```

step 4: run simulations and plot results

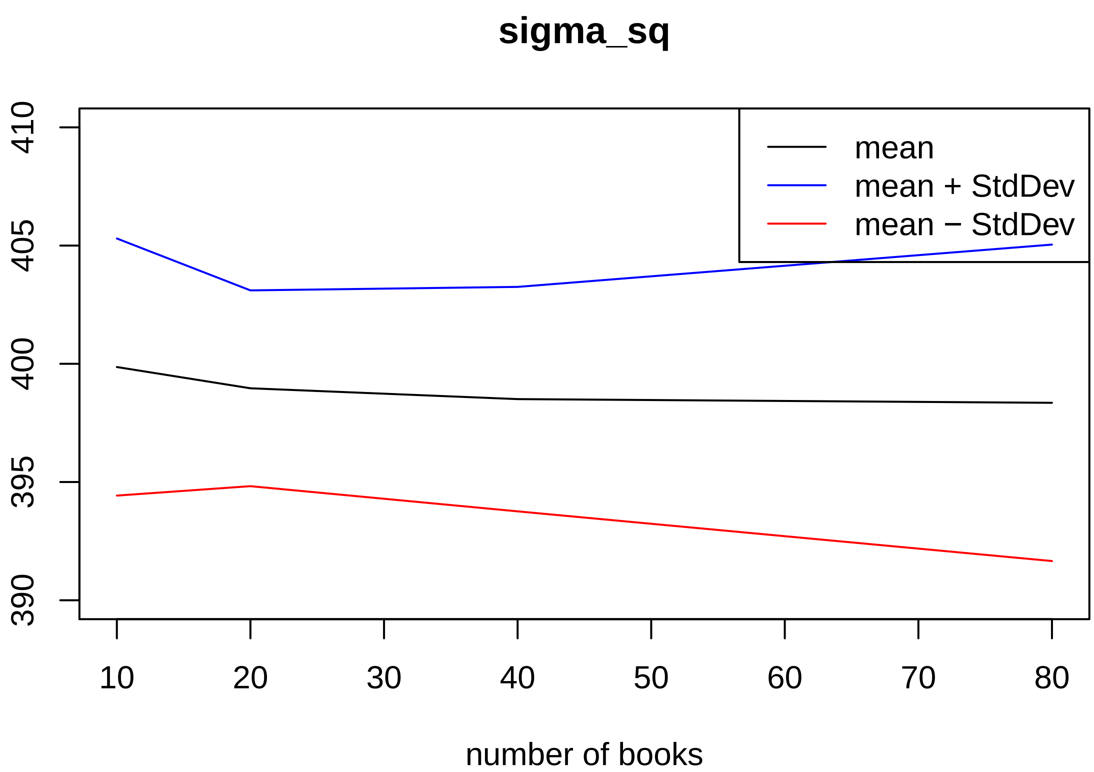
```
J = c(10, 20, 40, 80)
numDatasets = 10
mu = -10
tau_sq = 100
sigma_sq = 25

e1 = array(rep(-1, 40), dim = c(10,4))
e2 = array(rep(-1, 40), dim = c(10,4))
e3 = array(rep(-1, 40), dim = c(10,4))
e4 = array(rep(-1, 40), dim = c(10,4))

for (j in 1:length(J)) {
  for (n in 1:numDatasets) {
    sim_data = simData(numBooks = J[j], numSubjects = 960/J[j], mu = mu, tau_sq = tau_sq,
                 sigma_sq = sigma_sq)
    data = sim_data[[1]]
    nGroups = ncol(data)
    postSamples = pSamples(data = data)
    errors = ESE(trueParams = c(sim_data[[2]], mu, sigma_sq, tau_sq),
               postSamples = postSamples)
    ese_mu = errors[[1]]
    ese_tau_sq = errors[[2]]
    ese_sigma_sq = errors[[3]]
    avg_ese_theta = errors[[4]]

    if (j == 1) {
      e1[n, ] = c(ese_mu, ese_tau_sq, ese_sigma_sq, mean(avg_ese_theta))
    } else if (j == 2) {
      e2[n, ] = c(ese_mu, ese_tau_sq, ese_sigma_sq, mean(avg_ese_theta))
    } else if (j == 3) {
      e3[n, ] = c(ese_mu, ese_tau_sq, ese_sigma_sq, mean(avg_ese_theta))
    } else if (j == 4) {
      e4[n, ] = c(ese_mu, ese_tau_sq, ese_sigma_sq, mean(avg_ese_theta))
    }
  }
}
plot(J, c(mean(e1[,1]), mean(e2[,1]), mean(e3[,1]), mean(e4[,1])), type = "l",
     col = "black", ylab = "", xlab = "number of books", main = "mu", ylim = c(250, 2500))
lines(J, c(mean(e1[,1])-sd(e1[,1]),mean(e2[,1])-sd(e2[,1]),mean(e3[,1])-sd(e3[,1]),
           mean(e4[,1])-sd(e4[,1])), col = "red")
lines(J, c(mean(e1[,1])+sd(e1[,1]),mean(e2[,1])+sd(e2[,1]),mean(e3[,1])+sd(e3[,1]),
           mean(e4[,1])+sd(e4[,1])), col = "blue")
legend("topright", c("mean", "mean + StdDev", "mean - StdDev"), lty = c(1,1,1),
       col = c("black", "blue", "red"))
```



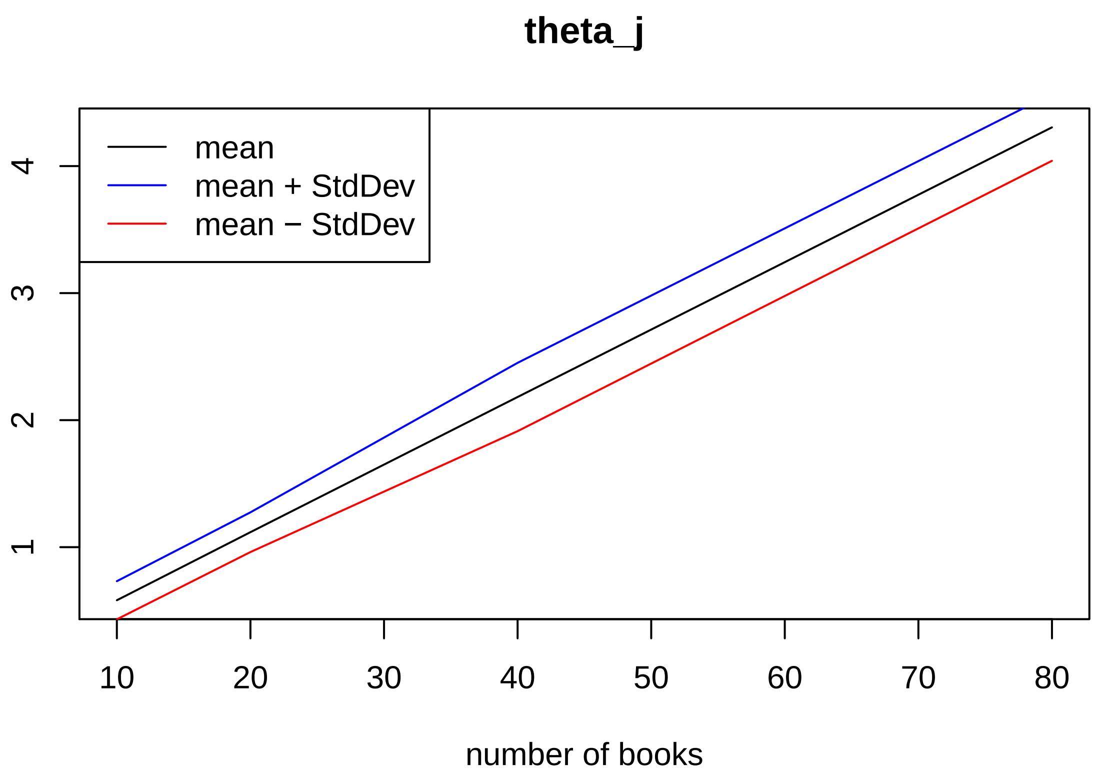

```r
plot(J, c(mean(e1[,2]), mean(e2[,2]), mean(e3[,2]), mean(e4[,2])), type = "l",
     col = "black", ylab = "", xlab = "number of books", main = "tau_sq",
     ylim = c(0, 2500))
lines(J, c(mean(e1[,2])-sd(e1[,2]),mean(e2[,2])-sd(e2[,2]),mean(e3[,2])-sd(e3[,2]),
           mean(e4[,2])-sd(e4[,2])), col = "red")
lines(J, c(mean(e1[,2])+sd(e1[,2]),mean(e2[,2])+sd(e2[,2]),mean(e3[,2])+sd(e3[,2]),
           mean(e4[,2])+sd(e4[,2])), col = "blue")
legend("topright", c("mean", "mean + StdDev", "mean - StdDev"), lty = c(1,1,1),
       col = c("black", "blue", "red"))
```




```
plot(J, c(mean(e1[,3]), mean(e2[,3]), mean(e3[,3]), mean(e4[,3])), type = "l",
     col = "black", ylab = "", xlab = "number of books", main = "sigma_sq",
     ylim = c(390, 410))
lines(J, c(mean(e1[,3])-sd(e1[,3]),mean(e2[,3])-sd(e2[,3]),mean(e3[,3])-sd(e3[,3]),
           mean(e4[,3])-sd(e4[,3])), col = "red")
lines(J, c(mean(e1[,3])+sd(e1[,3]),mean(e2[,3])+sd(e2[,3]),mean(e3[,3])+sd(e3[,3]),
           mean(e4[,3])+sd(e4[,3])), col = "blue")
legend("topright", c("mean", "mean + StdDev", "mean - StdDev"), lty = c(1,1,1),
       col = c("black", "blue", "red"))
```




```r
plot(J, c(mean(e1[,4]), mean(e2[,4]), mean(e3[,4]), mean(e4[,4])), type = "l",
     col = "black", ylab = "", xlab = "number of books", main = "theta_j")
lines(J, c(mean(e1[,4])-sd(e1[,4]),mean(e2[,4])-sd(e2[,4]),mean(e3[,4])-sd(e3[,4]),
           mean(e4[,4])-sd(e4[,4])), col = "red")
lines(J, c(mean(e1[,4])+sd(e1[,4]),mean(e2[,4])+sd(e2[,4]),mean(e3[,4])+sd(e3[,4]),
           mean(e4[,4])+sd(e4[,4])), col = "blue")
legend("topleft", c("mean", "mean + StdDev", "mean - StdDev"), lty = c(1,1,1),
       col = c("black", "blue", "red"))
```



step 5.

Looking at the previous step, we can see that as the number of books increases (and thus fewer subjects per book), the mean expected mean square errors of mu and tau squared squared decreases while it increases for the theta_j's. However, we see that these values remain somewhat constant for sigma squared. Looks like having more books is better since these mean values mu and tau are decreasing. Would choose 80 books.