BayesExam Report

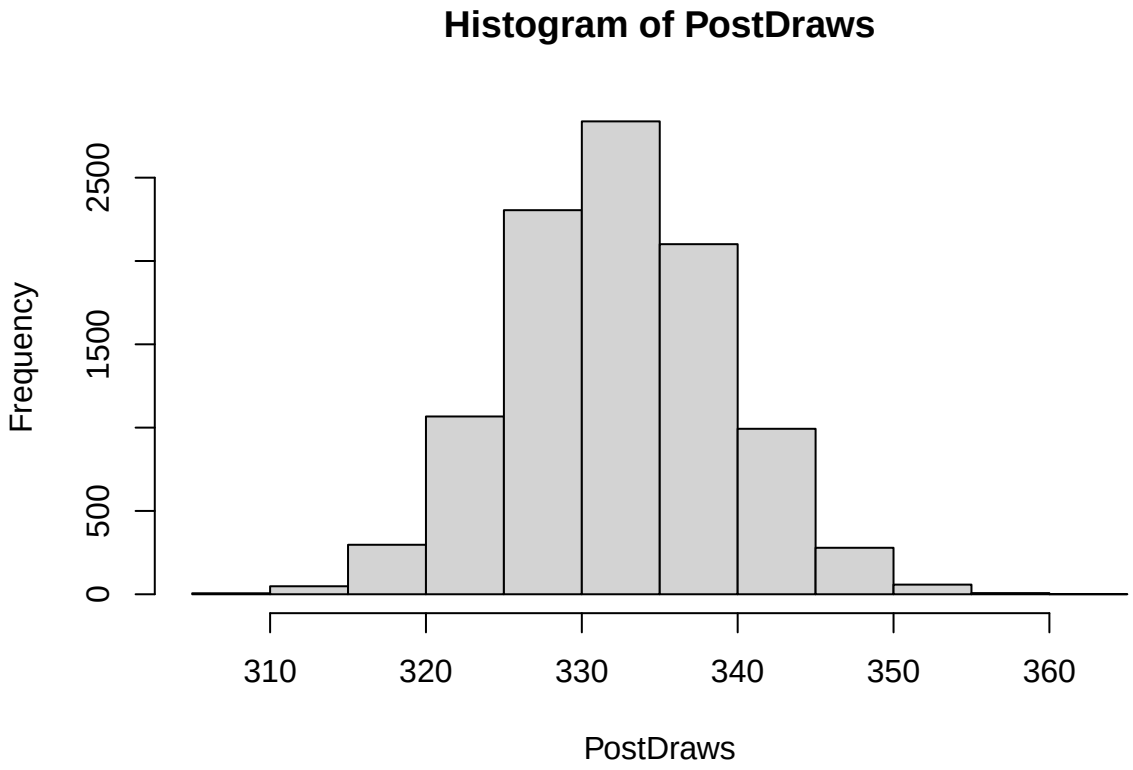
Problem 1

Task a

Hand written solution.

Task b

```
set.seed(12345)

q <- c(322,284,385,341,310)
n <- 5
an <- 2326
bn <- 7

N <- 10000

#generate thetas
thetas <- rgamma(n=N, shape = an-sum(q), rate = bn + n)

#generate posterior draws, the posterior is Gamma(an,bn)
PostDraws <- rgamma(n=N, shape = an, rate = bn)

hist(PostDraws)
```

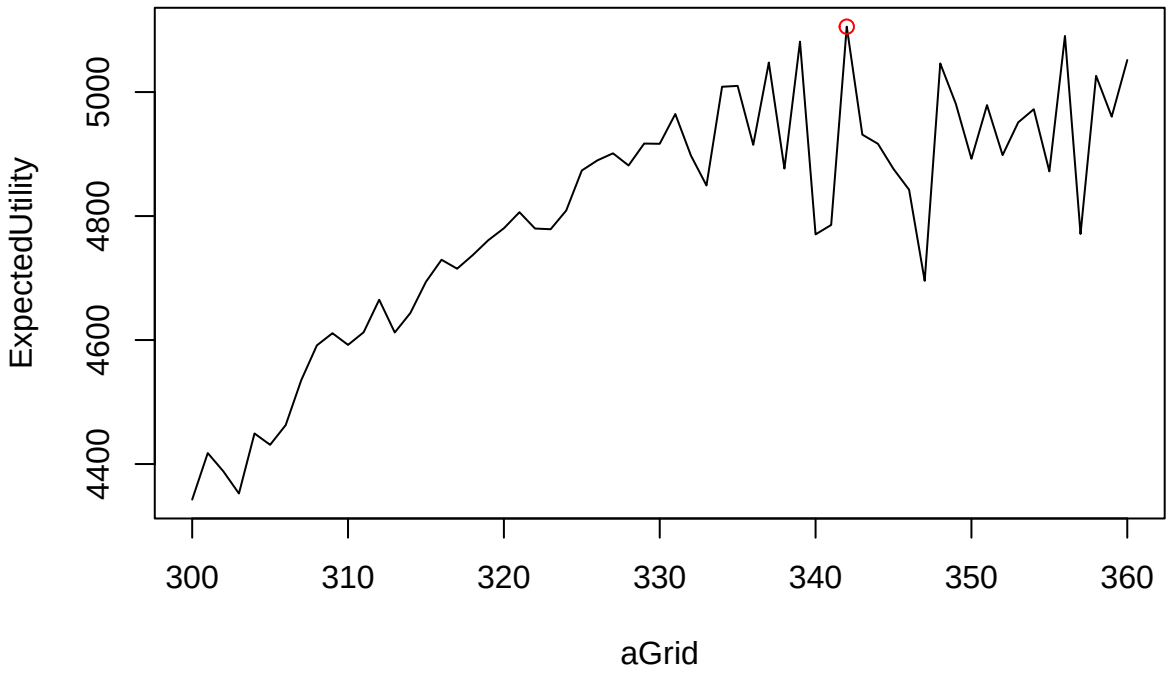



```
mean(PostDraws > 350)
```

```
## [1] 0.0066
```

Task C

```
utility_func <- function(a,Q6){
  util = rep(0,length(Q6))
  util[Q6<=a] = 15*Q6[Q6<=a]-(a-Q6[Q6<=a])
  util[Q6>a] = 15*a-0.1*(Q6[Q6>a]-a)^2
  return(util)
}

aGrid <- seq(300,360,1)
Q6 <- rgamma(n=length(aGrid), shape = an, rate = bn)

ExpectedUtility <- matrix(NA, nrow = length(aGrid), ncol = 1)

for (i in 1:length(aGrid)){
  ExpectedUtility[i,] <- utility_func(aGrid[i],Q6[i])
}

aOpt <- aGrid[which.max(ExpectedUtility)]

plot(aGrid,ExpectedUtility, type = "l")
points(aOpt,max(ExpectedUtility), col = "red")
```



The company needs to keep 349 products in stock for the sale in the next month.

Problem 2

```
load(file = 'Mollusc.RData')

rScaledInvChi2 <- function(n, df, scale){
  return((df*scale)/rchisq(n,df=df))
}

library(mvtnorm)

BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter){
  # Direct sampling from a Gaussian linear regression with conjugate prior:
  #
  # beta | sigma2 ~ N(mu_0, sigma2*inv(Omega_0))
  # sigma2 ~ Inv-Chi2(v_0,sigma2_0)
  #
  # INPUTS:
  #   y - n-by-1 vector with response data observations
  #   X - n-by-nCovs matrix with covariates, first column should be ones if you want an intercept.
  #   mu_0 - prior mean for beta
  #   Omega_0  - prior precision matrix for beta
  #   v_0      - degrees of freedom in the prior for sigma2
  #   sigma2_0 - location ("best guess") in the prior for sigma2
  #   nIter - Number of samples from the posterior (iterations)
  #
  # OUTPUTS:
  #   results$betaSample     - Posterior sample of beta.     nIter-by-nCovs matrix
  #   results$sigma2Sample   - Posterior sample of sigma2.   nIter-by-1 vector
```

```
  # Compute posterior hyperparameters
  n = length(y) # Number of observations
  nCovs = dim(X)[2] # Number of covariates
  XX = t(X)%*%X
  betaHat <- solve(XX,t(X)%*%y)
  Omega_n = XX + Omega_0
  mu_n = solve(Omega_n,XX%*%betaHat+Omega_0%*%mu_0)
  v_n = v_0 + n
  sigma2_n = as.numeric((v_0*sigma2_0 + ( t(y)%*%y + t(mu_0)%*%Omega_0%*%mu_0 - t(mu_n)%*%Omega_n%*%mu_n
  invOmega_n = solve(Omega_n)

  # The actual sampling
  sigma2Sample = rep(NA, nIter)
  betaSample = matrix(NA, nIter, nCovs)
  for (i in 1:nIter){

    # Simulate from p(sigma2 | y, X)
    sigma2 = rScaledInvChi2(n=1, df = v_n, scale = sigma2_n)
    sigma2Sample[i] = sigma2

    # Simulate from p(beta | sigma2, y, X)
    beta_ = rmvnorm(n=1, mean = mu_n, sigma = sigma2*invOmega_n)
    betaSample[i,] = beta_

  }
  return(results = list(sigma2Sample = sigma2Sample, betaSample=betaSample))
}
```

Task a

```
set.seed(12345)
#BayesLinReg <- function(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter)

mu_0 <- rep(0,6)
Omega_0 <- (1/100)*diag(6)
v_0 <- 1
sigma2_0 <- 100**2
nIter <- 1000

PostDraws <- BayesLinReg(y, X, mu_0, Omega_0, v_0, sigma2_0, nIter)

Betas <- PostDraws$betaSample

intervals <- matrix(NA, nrow = 6, ncol = 2)

for (i in 1:6){
  intervals[i,] <- quantile(Betas[,i], probs = c(0.005,0.995))
}

df_intervals <- data.frame(lower_bound = intervals[,1], upper_bound = intervals[,2])
colnames(df_intervals) <- c("lower bound", "upper bound")
rownames(df_intervals) <- c("b0", "b1","b2","b3","b4","b5")
knitr::kable(df_intervals, caption = "99% Equal Tail Credible Interval")
```

Table 1: 99% Equal Tail Credible Interval

	lower bound	upper bound
b0	-1460.5357688	403.1274144
b1	5.7456843	15.2044007
b2	-0.0358980	-0.0154597
b3	-31.8403534	108.5363835
b4	-2.0452711	0.6734511
b5	-0.2895864	0.0293873

The 99% probability of b1 is given by its interval.

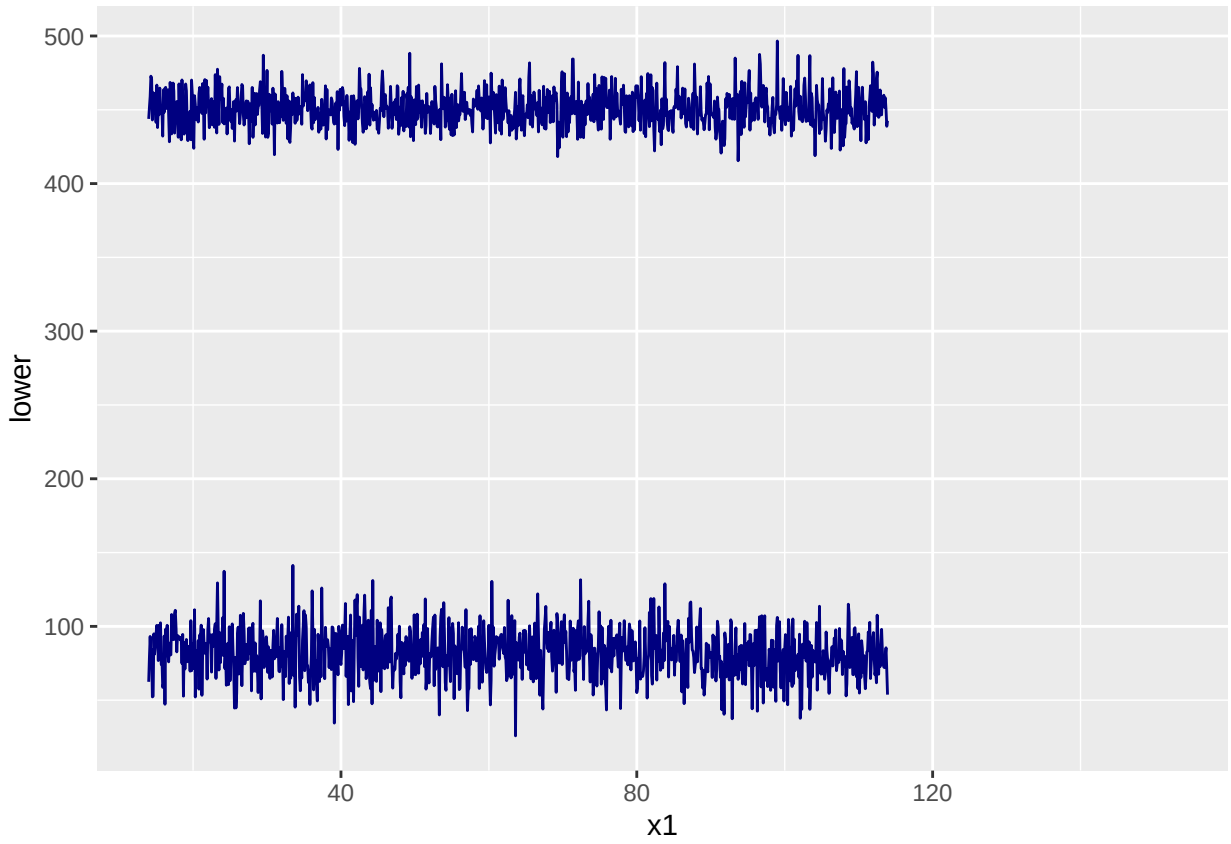
Task b

```
Sigmas2 <- PostDraws$sigma2Sample

MeanSD <- mean(sqrt(Sigmas2))
MedianSD <- median(sqrt(Sigmas2))
```

The posterior mean and posterior median of the standard deviation σ is approximately 39.93 and 39.45, respectively.

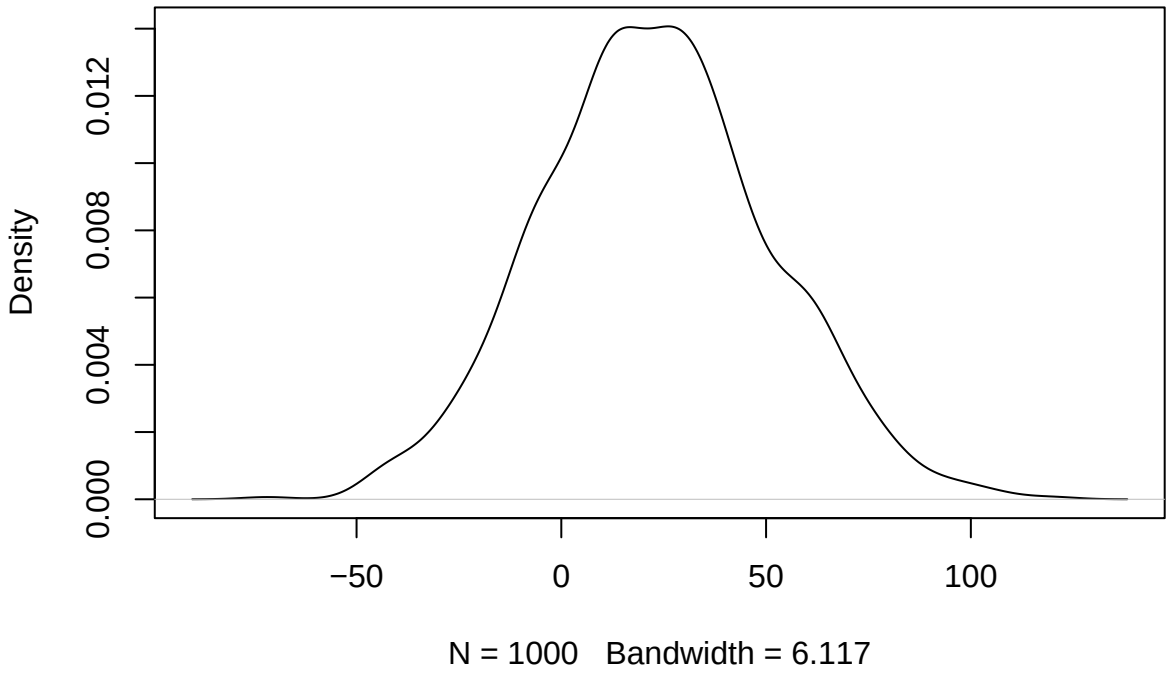
Task c

```
x1Grid <- seq(min(X[,2]), max(X[,2]), 0.1)
x1Grid2 <- x1Grid ^ 2
mu <- matrix(NA, nrow = length(x1Grid), ncol = 1000)

for (i in 1:length(x1Grid)){
  mu[i,] <- Betas[,1] + x1Grid[i]*Betas[,2] + x1Grid2[i]* Betas[,3] +
    Betas[,4] * 27 + Betas[,5] * (27^2) + x1Grid[i]*27*Betas[,6]
}

intervals <- matrix(NA, nrow = length(x1Grid), ncol = 2)

for (i in 1:1000){
  intervals[i,] <- quantile(mu[,i], probs = c(0.025,0.975))
}

df_intervals <- data.frame("x1" = x1Grid, "lower"=intervals[,1], "upper"=intervals[,2])

library(ggplot2)

ggplot(df_intervals)+
  geom_line(aes(x=x1,y=lower), color = "navy") +
  geom_line(aes(x=x1,y=upper), color = "navy")
```



Task d

```
Effect_x1 <- Betas[,2] + Betas[,3]
Effect_x2 <- Betas[,4] + Betas[,5]

Diff <- Effect_x2 - Effect_x1

DiffDens <- density(Diff)

plot(DiffDens, type = "l")
```



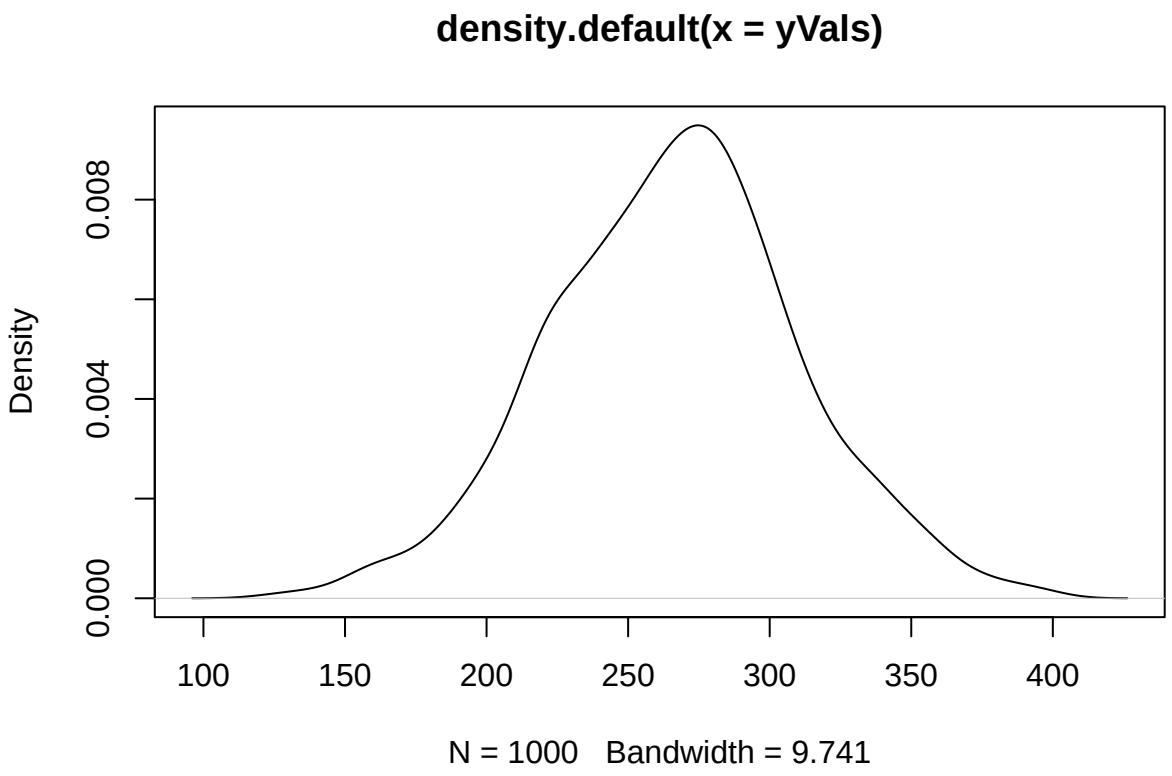

```
quantile(Diff, probs = c(0.025,0.975))
```

```
##      2.5%     97.5%
## -30.45664  78.82856
```

From the above plot there is a substantial mass probability that the effect of x2 is positive. However, the lower bound of the 95% equal tail credible interval is negative, thus there is not a certainty that the effect of x2 is always positive.

Task e

```
 mu <- Betas[,1] + 50*Betas[,2] + 2500* Betas[,3] + 25*Betas[,4] + 625*Betas[,5] + 50*25*Betas[,6]

yVals <- rnorm(1000, mean = mu , sd = sqrt(PostDraws$sigma2Sample))

yValsDens <- density(yVals)

plot(yValsDens, type = "l")
```



Task f

```
T_y_rep <- matrix(NA, nrow = 1000, ncol = 1)

for (i in 1:1000){
  T_y_rep[i,] <- rnorm(1, mean = Betas %*% t(X), sd = sqrt(PostDraws$sigma2Sample))

}

mean(T_y_rep >= max(T_y_rep))
```

```
## [1] 0.001
```

The posterior predictive value is 0.001, hence the model does not replicate the length of the largest mollusc in this data very well.

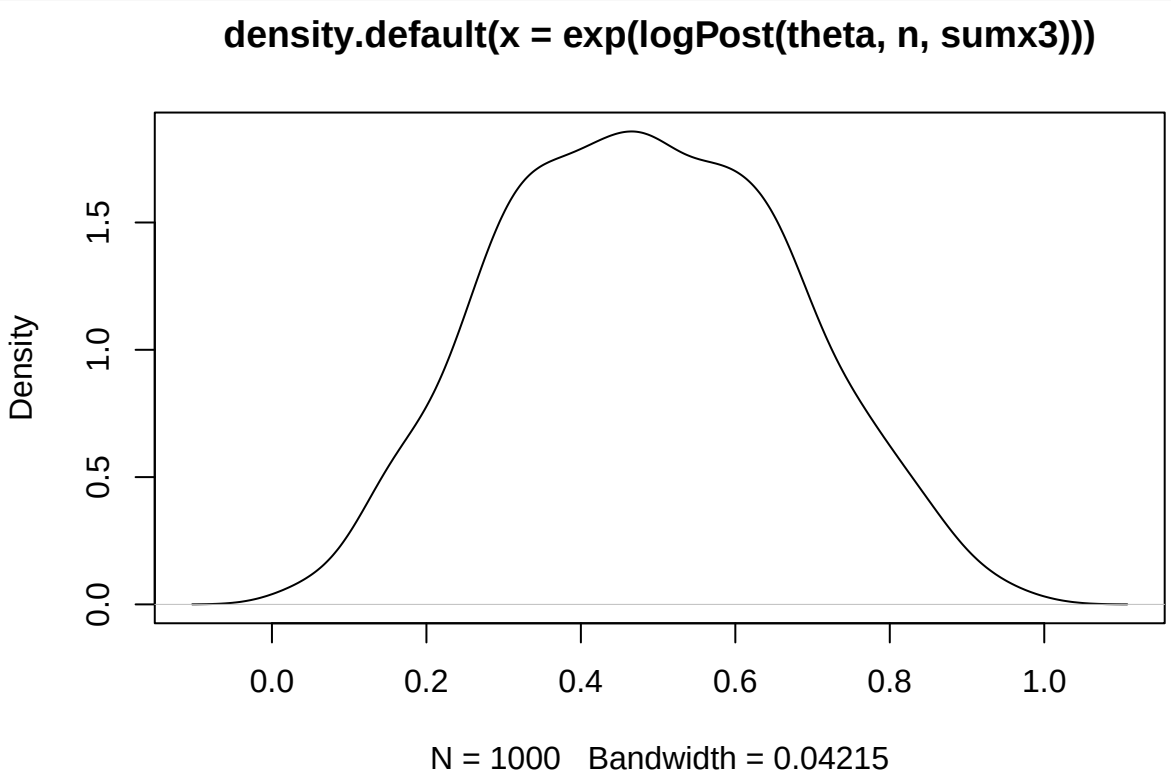
Problem 3

Tasks a,b and c are hand written.

Task d

```
set.seed(12345)

logPost <- function(theta,n,sumx3){
  res <- (2+n)*theta -(4 + sumx3)*theta
  return(res)
}
```

```r
N <- 1000
theta <- rgamma(N, shape = 3, rate = 4)
n <- 5
sumx3 <- (0.8)^3 + (1.1)^3 + (0.8)^3 + (0.9)^3 + 1

DensPosterior <- density(exp(logPost(theta,n,sumx3)))

plot(DensPosterior, type = "l")
```



Task e

```r
# I could not figure out why the optim function returned
#a Hessian matrix with value 0, thus I could not generate rnorm values.

OptimRes <- optim(0.5, logPost, gr = NULL, n, sumx3,
                  method = c("L-BFGS-B"),lower = 0.1,
                  control = list(fnscale = -1),hessian = TRUE)

#NormApprox <- dnorm(N, mean = OptimRes$par, sd = sqrt(diag(-solve(OptimRes$hessian))))
#plot(DensPosterior, type = "l", col = "red")
#lines(theta,NormApprox, type = "l", col = "blue")
```