Untitled

#1a)  
y = c(1690, 1790, 1760, 1750)  
sigma2 = 50^2  
n = length(y)  
meanY = mean(y)  
  
#Make draws from posterior using non informative prior  
yDraws = rnorm(1000, mean = meanY, sd = sqrt(sigma2 + sigma2/n))  
hist(yDraws, freq = FALSE, 50)  
  
#b  
  
#What is the prob that a weight in the next 365 days will have be larger than 230  
outcome = matrix(NA, 1000, 52)  
cases = c()  
for (i in 1:1000){  
 outcome[i,] = rnorm(52, mean = meanY, sd = sqrt(sigma2 + sigma2/n))  
 #draws\_larger\_than = sum(prediction\_52\_weeks > 1850)  
 #cases\_larger = append(cases\_larger, draws\_larger\_than)  
 cases = append(cases, sum(outcome[i,] > 1850))  
}  
mean =mean(cases)  
mean = mean(rowSums(outcome > 1850))  
  
#c   
aGrid = seq(0,10, length = 1000)  
  
  
#Important to use all the cases calculated => tae  
ExpectedLoss<-function(a){  
 EL = a + mean(rowSums(outcome > 1000\*log(a)))  
 return(EL)  
}  
  
loss = c()  
for(a in aGrid){  
 loss = append(loss, ExpectedLoss(a))  
}  
plot(aGrid, loss, type = 'l')  
aMode = aGrid[which.min(loss)]  
abline(h = aMode, col = "green")  
  
#2  
  
y = fish$length  
library(mvtnorm)  
data = matrix(0, 44, 6)  
data[,1] = fish$intercept  
data[,2] = fish$age  
data[,3] = fish$temp  
data[,4] = (fish$age)^2  
data[,5] = (fish$temp)^2  
data[,6] = (fish$age)\*(fish$temp)  
X = as.matrix(data)  
nCovs = dim(X)[2]  
mu\_0 = rep(0,nCovs)  
Omega\_0 = 0.01\*diag(nCovs)  
# Omega\_0[3,3] = 1000  
v\_0 = 1  
sigma2\_0 = 10000  
nIter = 5000  
  
Results = BayesLinReg(y, X, mu\_0, Omega\_0, v\_0, sigma2\_0, nIter)  
Results$betaSample  
Bmean = colMeans(Results$betaSample)  
Bq025 = apply(Results$betaSample,2,quantile,.025)  
Bq975 = apply(Results$betaSample,2,quantile,.975)  
print(data.frame(round(cbind(Bmean,Bq025,Bq975),3)),row.names= c("intercept", "age", "temp", "age^2", "temp^2", "ageXtemp"))  
  
  
median(sqrt(Results$sigma2Sample))  
mean(sqrt(Results$sigma2Sample))  
  
  
#b  
plot\_data = fish[1:11,]  
plot(plot\_data$age, plot\_data$length, ylim = c(90, 600))  
betas = Results$betaSample  
sigmas2 = Results$sigma2Sample  
  
  
function\_length = function(betas, sigmas2, X){  
 #length = betas%\*%X + rnorm(1, mean = 0, sd = sqrt(sigmas2))  
 length = betas%\*%X   
}  
  
ageGrid = seq(1, 160)  
length\_matrix = matrix(0, 5000, 160)  
mean\_length = c()  
for(i in 1:160){  
 length\_matrix[,i] = function\_length(betas, sigmas2, c(1,i, 25, i^2, 25^2, 25\*i ))  
 mean\_length = append(mean\_length, mean(length\_matrix[,i]))  
}  
lines(ageGrid, mean\_length)  
  
Length025 = apply(length\_matrix,2,quantile,.025)  
Length975 = apply(length\_matrix,2,quantile,.975)  
lines(ageGrid, Length025, col = "red")  
lines(ageGrid,Length975, col = "blue")  
  
#4  
log.posterior = function(params, x){  
 alpha = params[1]  
 beta = params[2]  
 log.likelihood = sum(dweibull(x, alpha, beta, log = TRUE))  
 log.alpha.prior = 2\*log(1/(alpha\*beta))  
 log.beta.prior = 2\*log(1/(alpha\*beta))  
 log.post = log.likelihood + log.alpha.prior + log.beta.prior  
 if(abs(log.post) == Inf){  
 log.likelihood = -20000  
}  
 return(log.post)  
}  
  
x = weibull  
  
initVal <- as.vector(rep(4,2));   
OptParams<-optim(initVal,log.posterior,gr=NULL,x,method=c("L-BFGS-B"),lower = c(0.00001,0.00001), upper = c(Inf,Inf), control=list(fnscale=-1),hessian=TRUE)  
  
params = OptParams$par  
  
# takes negative so that the posterior can be approx. as normal  
# J = -second derivate evaluated in theta\_hat  
hessian\_posterior = -solve(OptParams$hessian)  
  
  
  
  
  
  
#4b  
  
  
  
library(mvtnorm)  
metropolis = function(n, c, initval, hessian, posterior\_density, x){  
   
 # this step depends on previous position. Previous position becomes this turns mean.   
 acceptedDraws[i,] = initval  
 proposal\_draws\_previous = initval;  
 acceptedDraws = matrix(0, ncol=2,nrow=n)  
 accprobvec <- rep(0,n)  
   
 set.seed(12345)  
 for(i in 2:n){  
 # draws (theta\_p) from the proposal distribution ~ N(theta\_p-1, c\*hessian)  
 proposal\_draws = rmvnorm(1, proposal\_draws\_previous, c\*hessian)  
 proposal\_draws[proposal\_draws <= 0] = 1e-6  
 # create a ratio depending on if this draw is better than previous, take exp to remove logarithm (logposterior)  
 # posterior\_density = log.posterior => exp of the division => logA -logB   
 acceptance\_ratio = min(1,exp(posterior\_density(proposal\_draws, x)-posterior\_density(proposal\_draws\_previous, x)))  
 # draw a random uniformed variable to compare wiht acceptance ratio  
 random\_acceptance = runif(1,0,1)  
 # if acceptance ratio is bigger than random variable than we move to the new position, otherwise we stay  
 accprobvec[i] <- min(acceptance\_ratio,1)  
 if(acceptance\_ratio >= random\_acceptance){  
 proposal\_draws\_previous = proposal\_draws  
 params = proposal\_draws  
 }  
 acceptedDraws[i,] = params  
 }  
 return(list(draws = acceptedDraws, prob = accprobvec))  
}  
  
c = 0.1  
initval =c(1,1)  
hessian = hessian\_posterior  
x = weibull  
  
mp1 = metropolis(10000, c=0.1, initval, hessian, log.posterior, x)  
mp2 = metropolis(10000, c=4, initval, hessian, log.posterior, x)  
mp3 = metropolis(10000, c=100, initval, hessian, log.posterior, x)  
  
  
  
mean(mp1$prob)  
mean(mp2$prob)  
mean(mp3$prob)  
  
#bättre sätt  
c <- 0.1  
niter <- 2000  
warmup <- 500  
mp <- Metropolis(c,niter,warmup,initVal,Sigma,logPostWeibull,x)  
n <- length(initVal)  
theta\_mean <- rowMeans(mp$thetamat[,(warmup+1):(warmup+niter)])  
theta\_var<- rep(0,n)  
for(i in 1:n) {  
 theta\_var[i] <- var(mp$thetamat[i,(warmup+1):(warmup+niter)])   
}  
  
par(mfrow=c(2,1))  
plot(mp$thetamat[1,], type="l")  
plot(mp$thetamat[2,], type="l")  
mean(mp$accprobvec)