lecture 3.6 example 1

preliminaries

```r
# clear workspace
rm(list = ls())
#set random seed
set.seed(123)

#load data
data = read.csv("~/Desktop/data_height.csv")
dataM = data$height[data$sex =="Male"]
```

define key functions and objects

```r
#build parameter grid
nGridPoints = 1000
muGridMin = 0
muGridMax = 100
sigGridMin = 0
sigGridMax = 20
muGrid = seq(muGridMin, muGridMax,length.out = nGridPoints)
sigGrid = seq(sigGridMin, sigGridMax,length.out = nGridPoints)
muGridSize = (muGridMax - muGridMin) / nGridPoints
sigGridSize = (sigGridMax - sigGridMin) / nGridPoints

# generate prior matrix
buildPriors = function(meanMean, meanSD, sdMin, sdMax) {
  #initialize prior matrix
  priorM = matrix(rep(0, nGridPoints ^ 2 ),
                  nrow = nGridPoints,
                  ncol = nGridPoints,
                  byrow = TRUE)
  #fill out the prior matrix
  for (row in 1:nGridPoints) {
    for (col in 1:nGridPoints) {
      priorM[row,col] = muPrior[row] * sigPrior[col]
    }
  }
  #normalize the prior matrix
  priorM = priorM / (sum(priorM) * muGridSize * sigGridSize)
  #return the prior matrix
  return(priorM)
}

# compute posterior
computePost = function(data, priorM){
  #initialize posterior matrix
  postM = matrix(rep(-1, nGridPoints ^ 2 ),
                 nrow = nGridPoints,
                 ncol = nGridPoints,
                 byrow = TRUE)
  #fill out the posterior matrix
  for (row in 1:nGridPoints) {
    for (col in 1:nGridPoints) {
      muVal = muGrid[row]
      sigVal = sigGrid[col]
      #compute data likelihood
      loglike = sum(log(dnorm(data, muVal, sigVal)))
      # update posterior matrix cell
      postM[row,col] =  exp(loglike) * priorM[row,col]
    }
  }
  # normalize the posterior & return
  postM = postM / (sum(postM) * muGridSize * sigGridSize)
  return(postM)
}
```

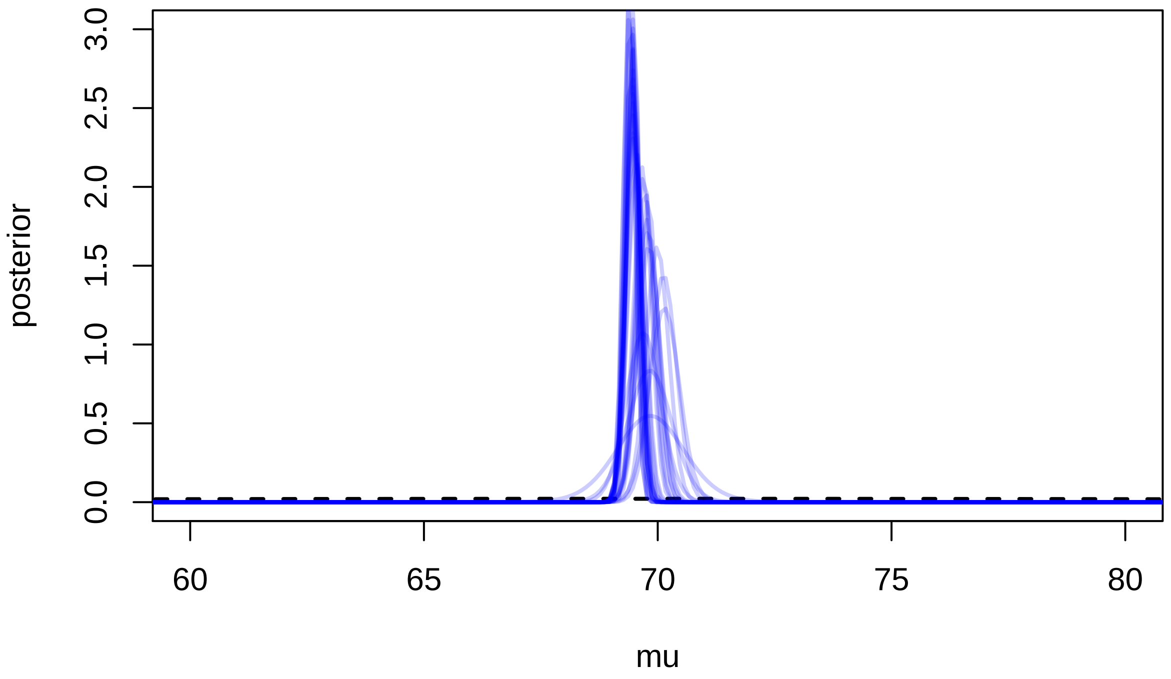
build prior matrix

```r
# set priors for mu
muMean = 70
muSd = 20
muPrior = dnorm(muGrid,muMean,muSd)
muPrior = muPrior / (sum(muPrior) * muGridSize)

#define priors for sigma
sigMin = 0
sigMax = 20
sigPrior = dunif(sigGrid,sigMin,sigMax)
sigPrior = sigPrior / (sum(sigPrior) * sigGridSize)

#build prior matrix
priorM = buildPriors(muMean, muSd, sigMin, sigMax)
```

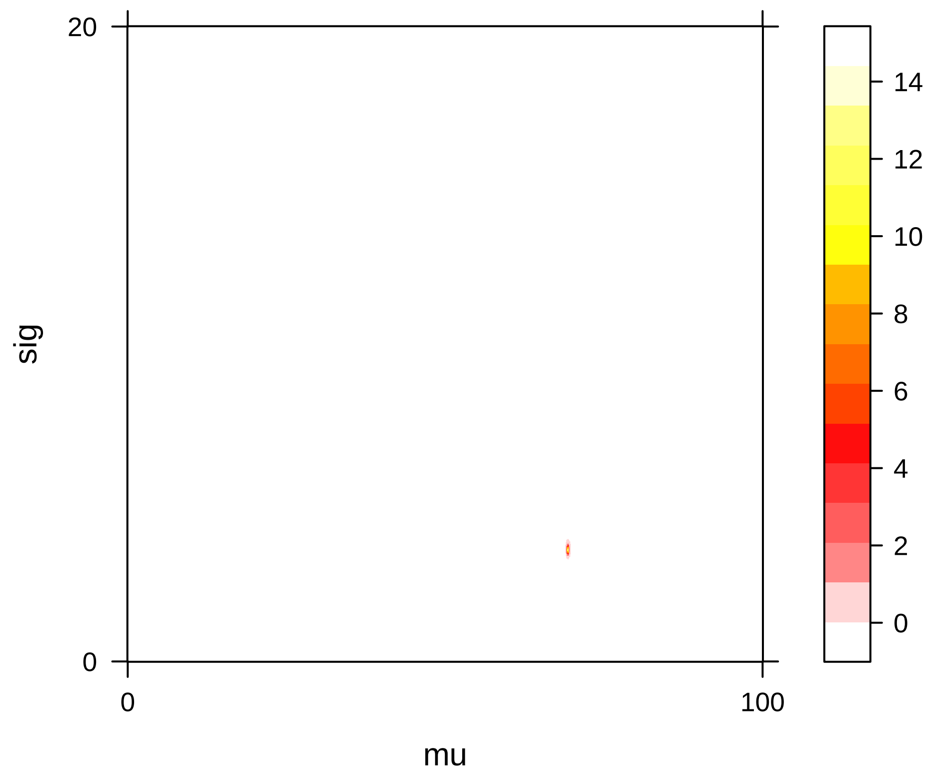
compute and visualize posteriors

NOTE: Decrease the number of grid points above to speed up the code

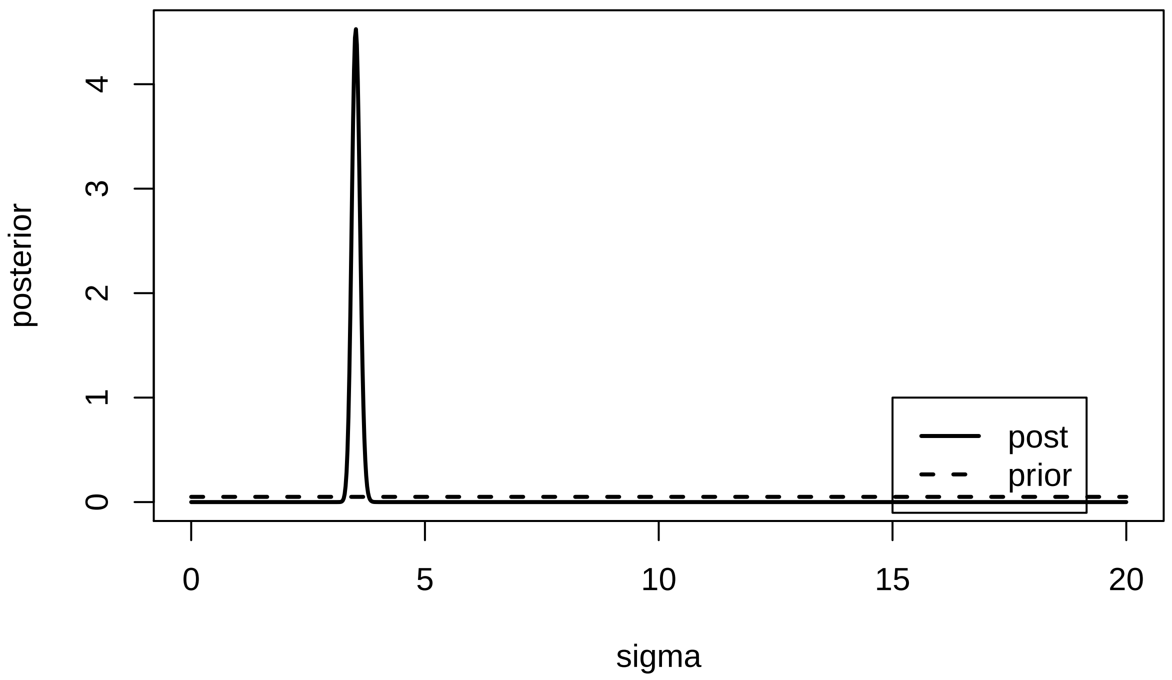
```r
# compute and plot posterior every K observations
# and visualize continuous updating
K = 25
plot(muGrid, muPrior, type="l", lwd=2, lty=2,
     xlab = "mu", ylab = "posterior",
     xlim=c(60,80), ylim = c(0,3))
for (t in 1:floor(length(dataM)/K)) {
  #choose data range
  range = (K*(t-1)+1):(K*(t-1)+K)
  #update posterior
  postM = computePost(dataM[range], priorM)
  # update prior
  priorM = postM
  #compute and plot marginal posterior Mu
  margMu = rowSums(postM * sigGridSize)
  points(muGrid,margMu, type="l", lwd=2,
         col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.2))
  Sys.sleep(1)
}
```



```
# visualize joint posterior
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(postM, col.regions=new.palette(20),
          xlab = "mu", ylab = "sig",,
          scales=list(x=list(at=c(1,nGridPoints), labels=c(muGridMin,muGridMax)),
                      y=list(at=c(1,nGridPoints), labels=c(sigGridMin,sigGridMax))))
```



```
# compute and visualize marginal posterior for sigma
margSig = colSums(postM * muGridSize)
plot(sigGrid,margSig, type="l", lwd=2,
     xlab = "sigma", ylab = "posterior")
points(sigGrid,sigPrior,
       type="l", col="black", lwd=2, lty=2)
legend(15,1, legend=c("post", "prior"),
       col=c("black","black"),
       lty=c(1,2), lwd=c(2,2))
```



posterior
sigma
0
1
2
3
4
0
5
10
15
20
post
prior