lecture 3.6 example 2

preliminaries

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

build parameter grid

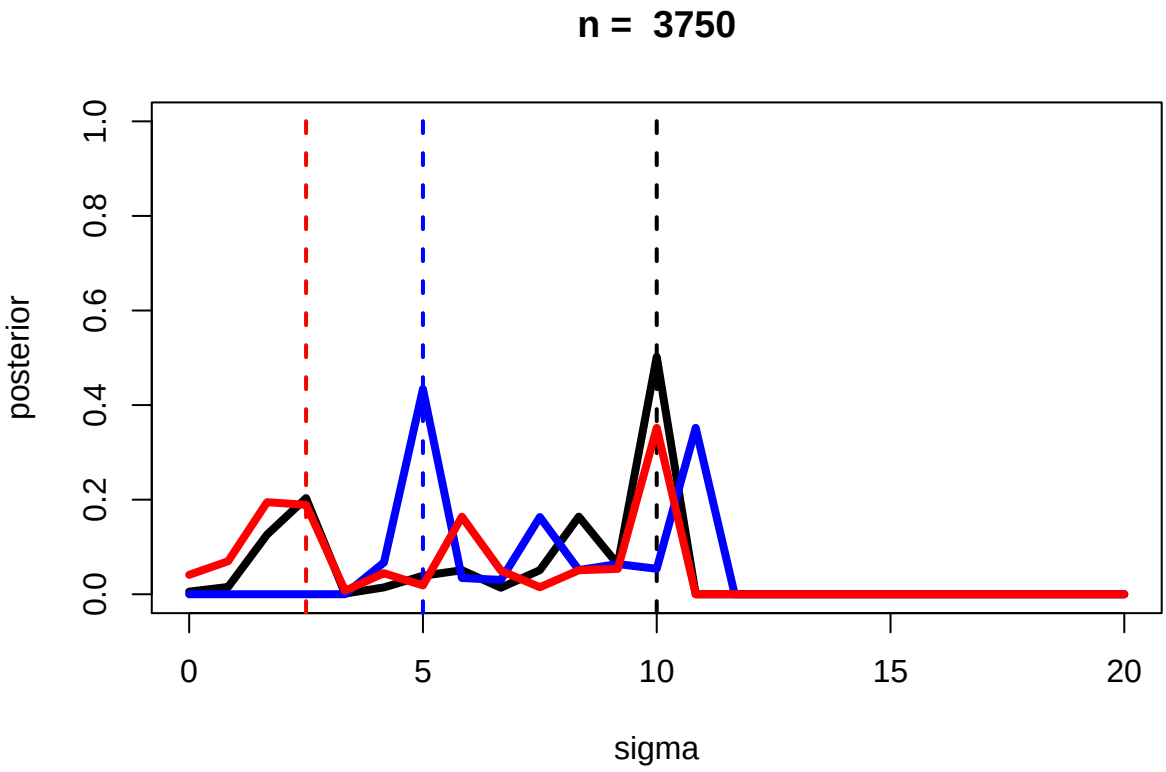
```
#build parameter grid
nGridPoints = 25

sigGridMin = 0
sigGridMax = 20
sigGrid = seq(sigGridMin, sigGridMax,length.out = nGridPoints)
sigGridSize = (sigGridMax - sigGridMin) / nGridPoints

sigAGridMin = 0
sigAGridMax = 20
sigAGrid = seq(sigAGridMin, sigAGridMax,length.out = nGridPoints)
sigAGridSize = (sigAGridMax - sigAGridMin) / nGridPoints
```

case of 2 instruments

NOTE: If it take too long to run, decrease the nObs used in the simulation

```
#simulation parameters
sig = 10
sigA = 5
sigB = 2.5
nObs = 3750

# uniform priors
prior = array(rep(1, nGridPoints ^ 3 ),
              dim = c(nGridPoints, nGridPoints, nGridPoints))

# define function to compute posterior
computePost = function(dataA,dataB,prior){
  #initialize posterior matrix
  postM = array(rep(-1, nGridPoints ^ 3 ),
                dim = c(nGridPoints, nGridPoints, nGridPoints))
  #fill out the posterior matrix
  for (dim1 in 1:nGridPoints) {
    for (dim2 in 1:nGridPoints) {
      for(dim3 in 1:nGridPoints) {
        sigVal = sigGrid[dim1]
        sigAVal = sigAGrid[dim2]
        sigBVal = sigAGrid[dim3]
```

```
        #compute data likelihood
        loglike = sum(log(dnorm(dataA, 0, sqrt(sigVal^2+sigAVal^2))))
        loglike = loglike + sum(log(dnorm(dataB, 0, sqrt(sigVal^2+sigBVal^2))))
        # update posterior matrix cell
        postM[dim1,dim2,dim3] =  exp(loglike) * prior[dim1,dim2,dim3]
      }
    }
  }
  # normalize the posterior & return
  postM = postM / sum(postM)
  postM = postM / (sigGridSize * sigAGridSize^2)
  return(postM)
}

#simulate data
dataA = rnorm(nObs, 0, sqrt(sig^2+sigA^2))
dataB = rnorm(nObs, 0, sqrt(sig^2+sigB^2))


#post = computePost(dataA[1:K],dataB[1:K],prior)
# computer posterior iteratively in groups of K observations
K = 50
for (t in 1:(nObs/K)) {
  range = (K*(t-1)+1):(K*(t-1)+K)
  post = computePost(dataA[range],dataB[range],prior)
  #compute marginal posteriors
  postSig = apply(post, c(1), sum)
  postSig = postSig / ( sum(postSig) * sigGridSize )

  postSigA = apply(post, c(2), sum)
  postSigA = postSigA / ( sum(postSigA) * sigAGridSize )

  postSigB = apply(post, c(3), sum)
  postSigB = postSigB / ( sum(postSigB) * sigAGridSize )

  # update prior
  prior = post
}

# visulize marginal posteriors
plot(sigGrid, postSig, type="l", lwd=4,
     xlab = "sigma", ylab = "posterior", ylim=c(0,1),
     main = paste("n = ", K*(t-1)+K))
abline(v=sig, lty=2, lwd=2)
points(sigAGrid, postSigA, type="l", lwd=4,
       col="blue",ylim=c(0,1))
abline(v=sigA, col="blue", lwd=2, lty=2)
points (sigAGrid, postSigB, type="l", lwd=4,col="red")
abline(v=sigB, col="red", lwd=2, lty=2)
```



case of 3 instruments

NOTE: If it take too long to run, decrease the nObs used in the simulation

```
#simulation parameters
sig = 10
sigA = 5
sigB = 2.5
sigC = 0.5
nObs = 2500

#set-up uniform priors
priorM = array(rep(1, nGridPoints ^ 4 ),
               dim = c(nGridPoints, nGridPoints, nGridPoints, nGridPoints))

# define function to compute posterior
computePost = function(dataA,dataB,dataC,prior){
  #initialize posterior matrix
  postM = array(rep(-1, nGridPoints ^ 4 ),
                dim = c(nGridPoints, nGridPoints, nGridPoints, nGridPoints))
  #fill out the posterior matrix
  for (dim1 in 1:nGridPoints) {
    for (dim2 in 1:nGridPoints) {
      for(dim3 in 1:nGridPoints) {
        for (dim4 in 1:nGridPoints) {
          sigVal = sigGrid[dim1]
          sigAVal = sigAGrid[dim2]
          sigBVal = sigAGrid[dim3]
          sigCVal = sigAGrid[dim4]
```

```
          #compute data likelihood
          loglike = sum(log(dnorm(dataA, 0, sqrt(sigVal^2+sigAVal^2))))
          loglike = loglike + sum(log(dnorm(dataB, 0, sqrt(sigVal^2+sigBVal^2))))
          loglike = loglike + sum(log(dnorm(dataC, 0, sqrt(sigVal^2+sigCVal^2))))
          # update posterior matrix cell
          postM[dim1,dim2,dim3,dim4] =  exp(loglike) * prior[dim1,dim2,dim3,dim4]
        }
      }
    }
  }
  # normalize the posterior & return
  postM = postM / sum(postM)
  postM = postM / (sigGridSize * sigAGridSize^3)
  return(postM)
}

#simulate data
dataA = rnorm(nObs, 0, sqrt(sig^2+sigA^2))
dataB = rnorm(nObs, 0, sqrt(sig^2+sigB^2))
dataC = rnorm(nObs, 0, sqrt(sig^2+sigC^2))

#set new prior to posterior and iterate in groups of K observations
K = 50
for (t in 1:(nObs/K)) {

  range = (K*(t-1)+1):(K*(t-1)+K)
  post = computePost(dataA[range],dataB[range],dataC[range],priorM)

  #compute marginal posteriors
  postSig = apply(post, c(1), sum)
  postSig = postSig / ( sum(postSig) * sigGridSize )

  postSigA = apply(post, c(2), sum)
  postSigA = postSigA / ( sum(postSigA) * sigAGridSize )

  postSigB = apply(post, c(3), sum)
  postSigB = postSigB / ( sum(postSigB) * sigAGridSize )

  postSigC = apply(post, c(4), sum)
  postSigC = postSigC / ( sum(postSigC) * sigAGridSize )

  priorM = post
}

# visulize marginal posterior for Sigma
plot(sigGrid, postSig, type="l", lwd=4,
     xlab = "sigma", ylab = "posterior", ylim=c(0,1),
     main = paste("n = ", K*(t-1)+K))
abline(v=sig, lty=2, lwd=2)
points(sigAGrid, postSigA, type="l", lwd=4,
       col="blue",ylim=c(0,1))
abline(v=sigA, col="blue", lwd=2, lty=2)
points (sigAGrid, postSigB, type="l", lwd=4,col="red")
```

```
abline(v=sigB, col="red", lwd=2, lty=2)
points (sigAGrid, postSigC, type="l", lwd=4,col="brown")
abline(v=sigC, col="brown", lwd=2, lty=2)
```