Lecture 4.8 example 5

preliminaries

```
#clear workspace
rm(list=ls())
#initalize random seed
set.seed(255)
```

specify simulation parameters

```
beta0 = 0
beta1 = 0.5
beta2 = 0.5
sigma = 1

alpha = 0.1   #controls degree of correlation between x1 and x2

nObs = 200

stepBeta0Grid = 0.1
stepBeta1Grid = 0.0125
stepBeta2Grid = 0.0125
stepSigmaGrid = 0.1
beta0Grid = seq(-0.5,0.5, by = stepBeta0Grid)
beta1Grid = seq(0,1, by = stepBeta1Grid)
beta2Grid = seq(0,1, by = stepBeta2Grid)
sigmaGrid = seq(stepSigmaGrid,2, by = stepSigmaGrid)
```

build model objects

```
# grid utility functions
stepSize = function(grid) {
  if (length(grid)==1) {
    step = 1
  }
  else {
    step = (max(grid) - min(grid)) / (length(grid) - 1)
  }
  return(step)
}

# build priors
buildPriorMultivar = function(beta0Grid,beta1Grid,beta2Grid,sigmaGrid) {
  # build useful grid objects
  nBeta0Grid = length(beta0Grid)
  nBeta1Grid = length(beta1Grid)
  nBeta2Grid = length(beta2Grid)
  nSigmaGrid = length(sigmaGrid)
```

```
  #
  prior = array( rep(1, nBeta0Grid * nBeta1Grid * nBeta2Grid * nSigmaGrid ),
                 dim = c(nBeta0Grid, nBeta1Grid, nBeta2Grid, nSigmaGrid ))
  #
  for (nB0 in 1:nBeta0Grid) {
    for (nB1 in 1:nBeta1Grid) {
      for (nB2 in 1:nBeta2Grid) {
        for (nSig in 1:nSigmaGrid) {
          # change next expression to set different priors
          prior[nB0,nB1,nB2, nSig] = 1 / nSig^2
        }
      }
    }
  }
  return(prior)
}


#likelihood
likelihoodMultivar = function(y,x1, x2, b0L, b1L, b2L, sL){
  loglike = sum(log(dnorm(y-b0L-b1L*x1-b2L*x2, mean = 0, sd=sL)))
  like = exp(loglike)
  return(like)
}


#compute posterior function
compPostMultivar = function(y,x1, x2, prior, beta0Grid,beta1Grid,beta2Grid,sigmaGrid) {
  # build useful grid objects
  nBeta0Grid = length(beta0Grid)
  nBeta1Grid = length(beta1Grid)
  nBeta2Grid = length(beta2Grid)
  nSigmaGrid = length(sigmaGrid)
  #initialize local posterior
  post = array( rep(-1, nBeta0Grid * nBeta1Grid * nBeta2Grid * nSigmaGrid ),
                dim = c(nBeta0Grid, nBeta1Grid, nBeta2Grid, nSigmaGrid ))
  # compute posterior
  for (nBeta0 in 1:nBeta0Grid) {
    b0 = beta0Grid[nBeta0]
    print(paste("bo = ", b0))
    for (nBeta1 in 1:nBeta1Grid) {
      b1 = beta1Grid[nBeta1]
      for (nBeta2 in 1:nBeta2Grid) {
        b2 = beta2Grid[nBeta2]
        for (nSigma in 1:nSigmaGrid) {
          s = sigmaGrid[nSigma]
          post[nBeta0,nBeta1,nBeta2,nSigma] = likelihoodMultivar(y,x1,x2,b0,b1,b2,s) * prior[nBeta0,nBe
        }
      }
    }
  }
  # normalize posterior
  post = post / ( sum(post) * stepSize(beta0Grid) * stepSize(beta1Grid) * stepSize(beta2Grid) * stepSiz
```

```
  # return
  return(post)
}
```

simulate dataset

```
z  = rnorm(nObs, 0.5,2)
x1 = alpha * z + (1-alpha) * rnorm(nObs, 0.5,2)
x2 = alpha * z + (1-alpha) * rnorm(nObs, 0.5,2)
y = rnorm(nObs, beta0 + beta1 * x1 + beta2 * x2, sigma )
```

fit model

```
# build priors
prior = buildPriorMultivar(beta0Grid,beta1Grid,beta2Grid,sigmaGrid)

#compute posterior function iteratively using batches of 100 observations
for (k in 1:floor(nObs/100)) {
  print('++++++++++++++++++++++')
  print(paste("k = ", k))
  print('++++++++++++++++++++++')
  y_batch = y[(1 + (k-1)* 100) : (k * 100)]
  x1_batch = x1[(1 + (k-1)* 100) : (k * 100)]
  x2_batch = x2[(1 + (k-1)* 100) : (k * 100)]
  post = compPostMultivar(y_batch,x1_batch,x2_batch,prior,
                          beta0Grid,beta1Grid,beta2Grid,sigmaGrid)
  prior = post
}
```

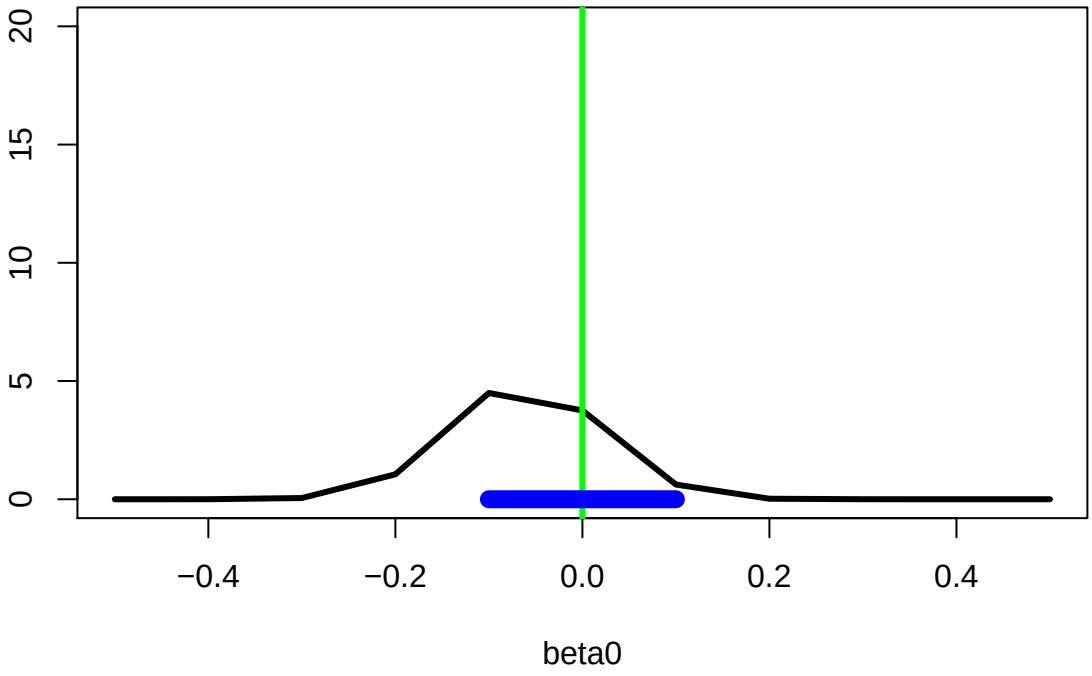
```
## [1] "++++++++++++++++++++"
## [1] "k =  1"
## [1] "++++++++++++++++++++"
## [1] "bo =  -0.5"
## [1] "bo =  -0.4"
## [1] "bo =  -0.3"
## [1] "bo =  -0.2"
## [1] "bo =  -0.1"
## [1] "bo =  0"
## [1] "bo =  0.1"
## [1] "bo =  0.2"
## [1] "bo =  0.3"
## [1] "bo =  0.4"
## [1] "bo =  0.5"
## [1] "++++++++++++++++++++"
## [1] "k =  2"
## [1] "++++++++++++++++++++"
## [1] "bo =  -0.5"
## [1] "bo =  -0.4"
## [1] "bo =  -0.3"
## [1] "bo =  -0.2"
## [1] "bo =  -0.1"
```

```
## [1] "bo =  0"
## [1] "bo =  0.1"
## [1] "bo =  0.2"
## [1] "bo =  0.3"
## [1] "bo =  0.4"
## [1] "bo =  0.5"
```

```
#compute marginal posteriors
margPostBeta0 = apply(post,c(1),sum)
margPostBeta0 = margPostBeta0 / (sum(margPostBeta0) * stepSize(beta0Grid))

margPostBeta1 = apply(post,c(2),sum)
margPostBeta1 = margPostBeta1 / (sum(margPostBeta1) * stepSize(beta1Grid))

margPostBeta2 = apply(post,c(3),sum)
margPostBeta2 = margPostBeta2 / ( sum(margPostBeta2) * stepSize(beta2Grid))

margPostSigma = apply(post,c(4),sum)
margPostSigma = margPostSigma / (sum(margPostSigma) * stepSize(sigmaGrid))
```

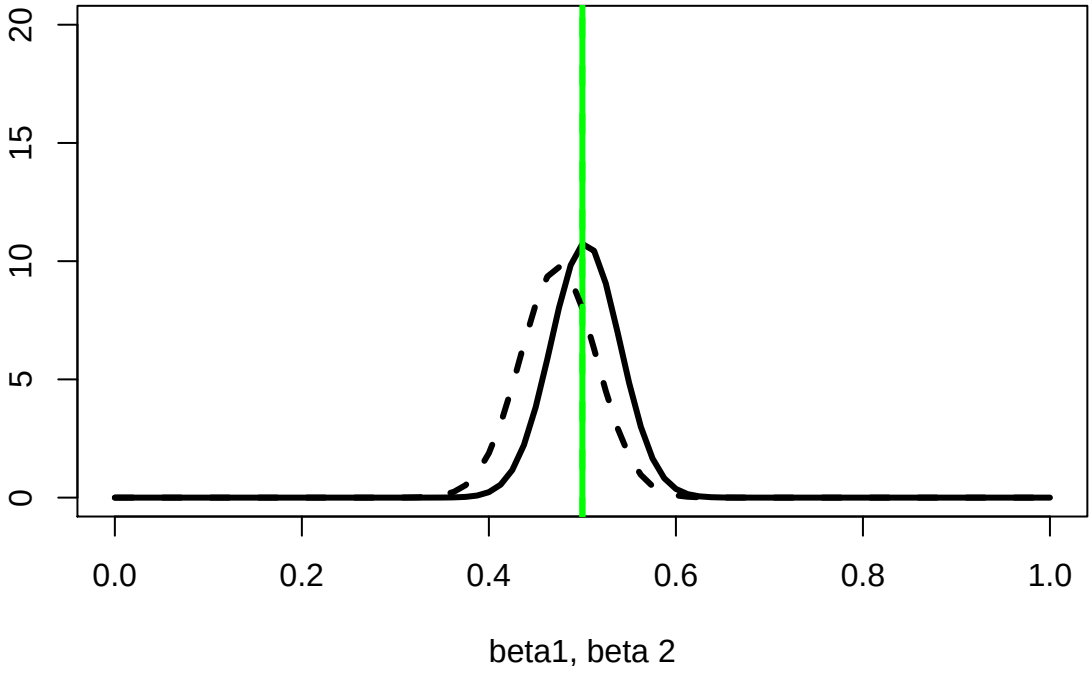
visualize results

```
plot(beta0Grid, margPostBeta0,
     xlab = "beta0", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,20))
abline(v=beta0, lwd=3, col="green")
segments(beta0-stepSize(beta0Grid), 0,beta0+stepSize(beta0Grid),0, lwd=9, col="blue" )
```

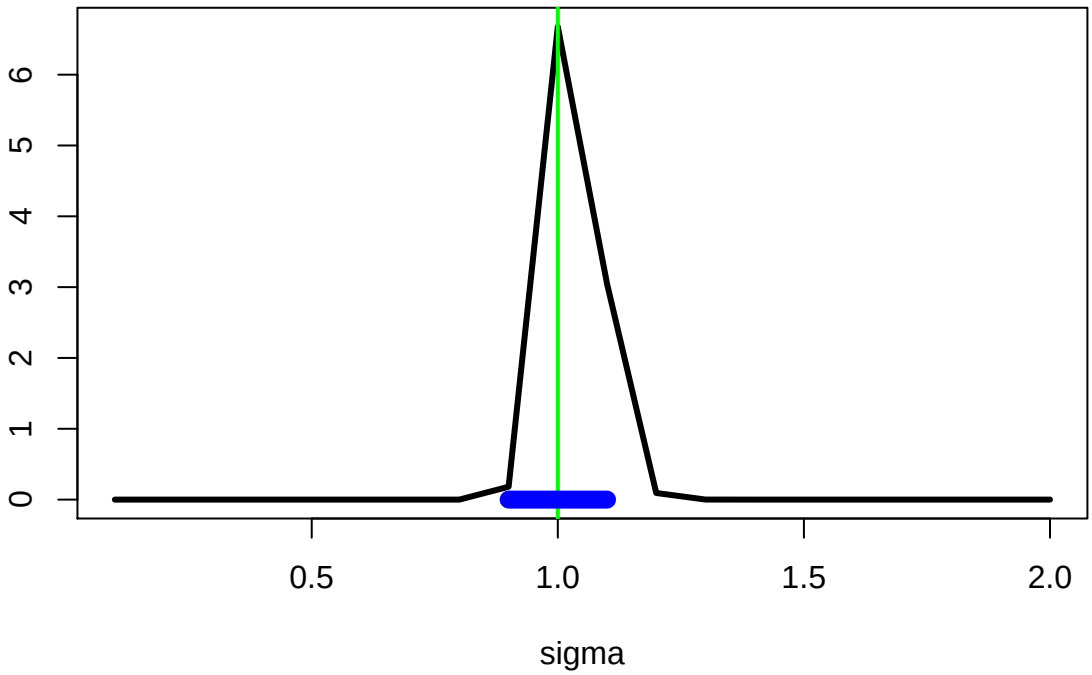



```
plot(beta1Grid, margPostBeta1,
     xlab = "beta1, beta 2", ylab="",
     type = "l", lwd = 3,
```

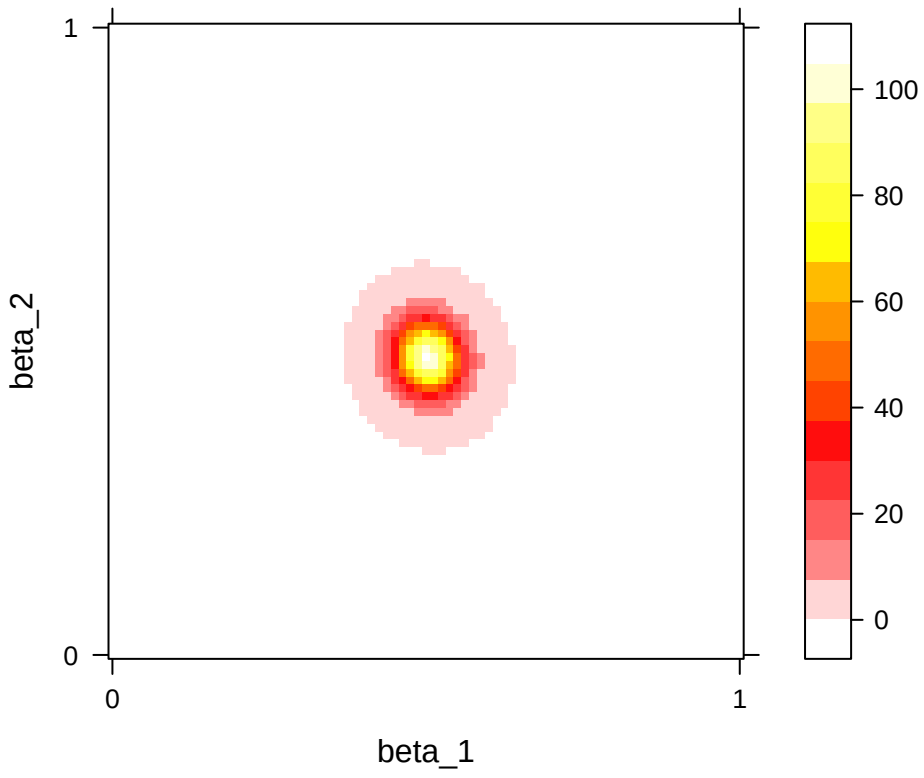
```
     ylim=c(0,20))
abline(v=beta1, lwd=3, col="green")
points(beta1Grid, margPostBeta2,
     type = "l", lwd = 3,lty=2)
abline(v=beta2, lwd=3, col="green", lty=2)
```




```
plot(sigmaGrid, margPostSigma,
     xlab = "sigma", ylab="",
     type = "l", lwd = 3)
abline(v=sigma, lwd=2, col="green")
segments(sigma-stepSize(sigmaGrid), 0,sigma+stepSize(sigmaGrid),0, lwd=9, col="blue" )
```




```
jointPost = apply(post,c(2,3),sum)
jointPost = jointPost / ( sum(jointPost) * stepSize(beta1Grid) * stepSize(beta2Grid))
library(lattice)
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
levelplot(jointPost, col.regions=new.palette(20),
          xlab = "beta_1", ylab = "beta_2",
          scales=list(x=list(at=c(1,length(beta1Grid)),
                             labels=c(0,1)),
                      y=list(at=c(1,length(beta2Grid)),
                             labels=c(0,1))))
```


1
0
beta_2
0
1
beta_1
100
80
60
40
20
0