lecture 4.1 example 3

preliminaries

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

set simulation parameters

```
nObs = c(10,20,40,80)
xGrid = seq(-1,1, by=0.1)

beta0 = 0
beta1 = 1
sigma = 1

boundBetaGrid = 2
sizeBetaGrid = 0.025
boundSigmaGrid = 2
sizeSigmaGrid = 0.025

beta0Grid = seq(-boundBetaGrid,boundBetaGrid, by = sizeBetaGrid)
beta1Grid = seq(-boundBetaGrid,boundBetaGrid, by = sizeBetaGrid)
sigmaGrid = seq(sizeSigmaGrid,boundSigmaGrid, by = sizeSigmaGrid)
nBeta0Grid = length(beta0Grid)
nBeta1Grid = length(beta1Grid)
nSigmaGrid = length(sigmaGrid)

# prior
priorSigma = 1 / sigmaGrid^2
```

build model objects

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

#compute posterior
compPost = function(y,x){
  #initialize local posterior
  post = array( rep(-1, nBeta0Grid * nBeta1Grid * nSigmaGrid ),
                dim = c(nBeta0Grid, nBeta1Grid, nSigmaGrid ))
  # compute posterior
```

```
  for (nBeta0 in 1:nBeta0Grid) {
    b0 = beta0Grid[nBeta0]
    for (nBeta1 in 1:nBeta1Grid) {
      b1 = beta1Grid[nBeta1]
      for (nSigma in 1:nSigmaGrid) {
        s = sigmaGrid[nSigma]
        post[nBeta0,nBeta1,nSigma] = likelihood(y,x, b0, b1, s) * priorSigma[nSigma]
      }
    }
  }
  # normalize posterior
  post = post / ( sum(post) * sizeBetaGrid^2 * sizeSigmaGrid )
  # return
  return(post)
}
```

simulations

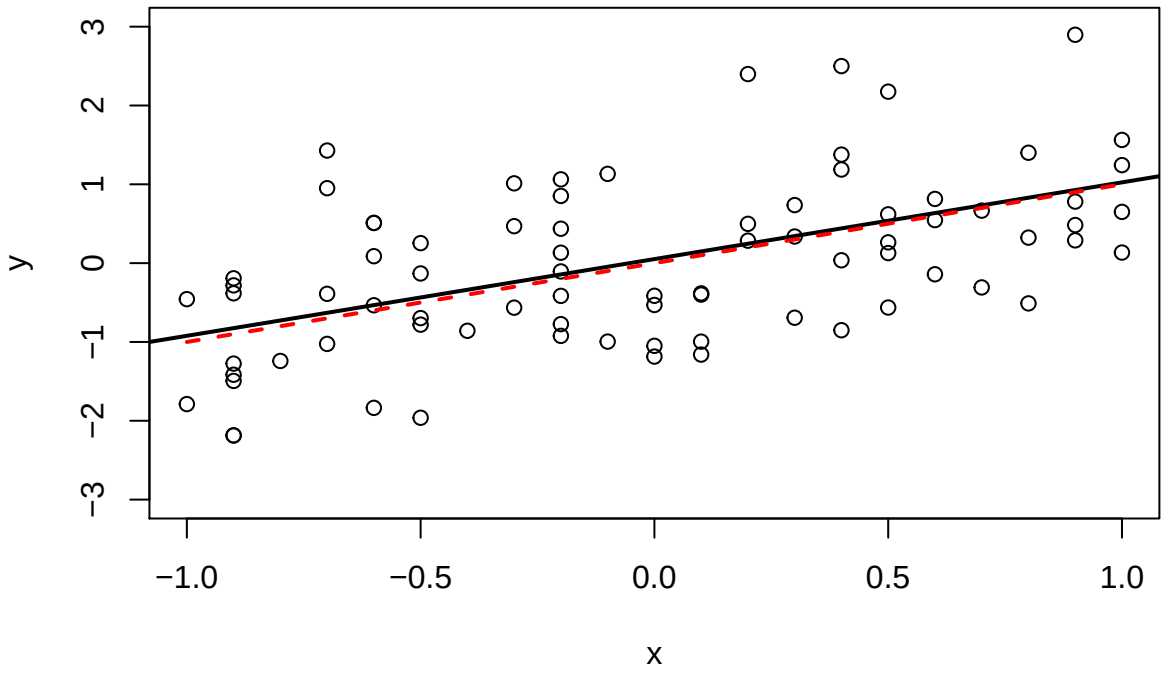
```
#initialize arrays
postFinal = array( rep(-1, length(nObs) * nBeta0Grid * nBeta1Grid * nSigmaGrid ),
              dim = c(length(nObs), nBeta0Grid, nBeta1Grid, nSigmaGrid ))

#main loop
for (n in 1:length(nObs)) {
  print(n)
  # generate data
  x = sample(xGrid, nObs[n], replace = TRUE )
  y = rnorm(nObs[n], mean = beta0 + beta1 * x, sd = sigma)
  # compute and store posterior
  postFinal[n,,,] = compPost(y,x)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

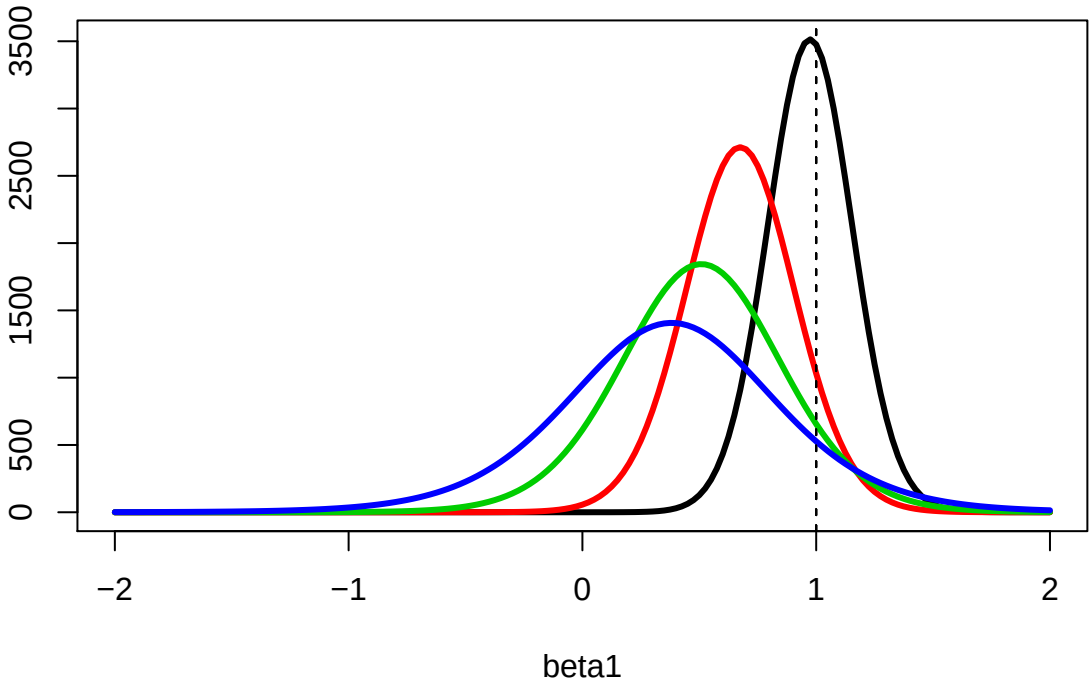
```
#compute marginal posteriors
margPostBeta0 = apply(postFinal,c(1,2),sum)
margPostBeta1 = apply(postFinal,c(1,3),sum)
margPostSigma = apply(postFinal,c(1,4),sum)
```

visualize results

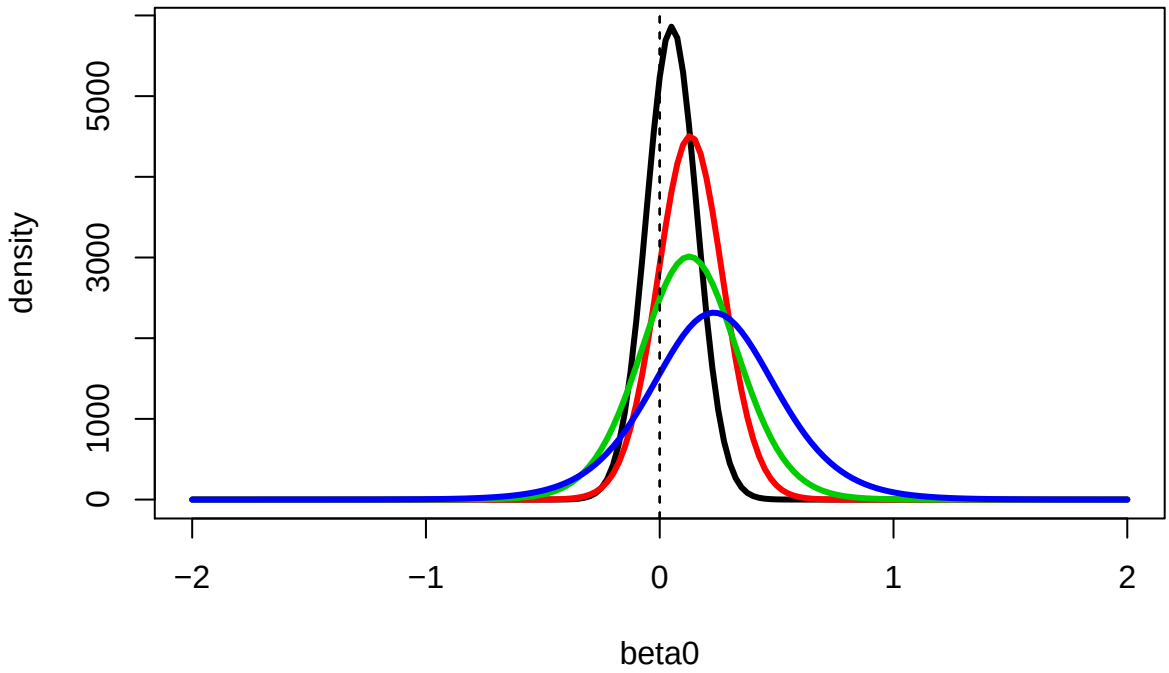
```
#plot data & beta
plot(x,y,xlim=c(-1,1),ylim=c(-3,3))
#plot beta-hat line
abline(lm(y~x), lwd=2)
points(xGrid, beta0 + beta1*xGrid, lwd=2, col=2, lty=2, type="l")
```



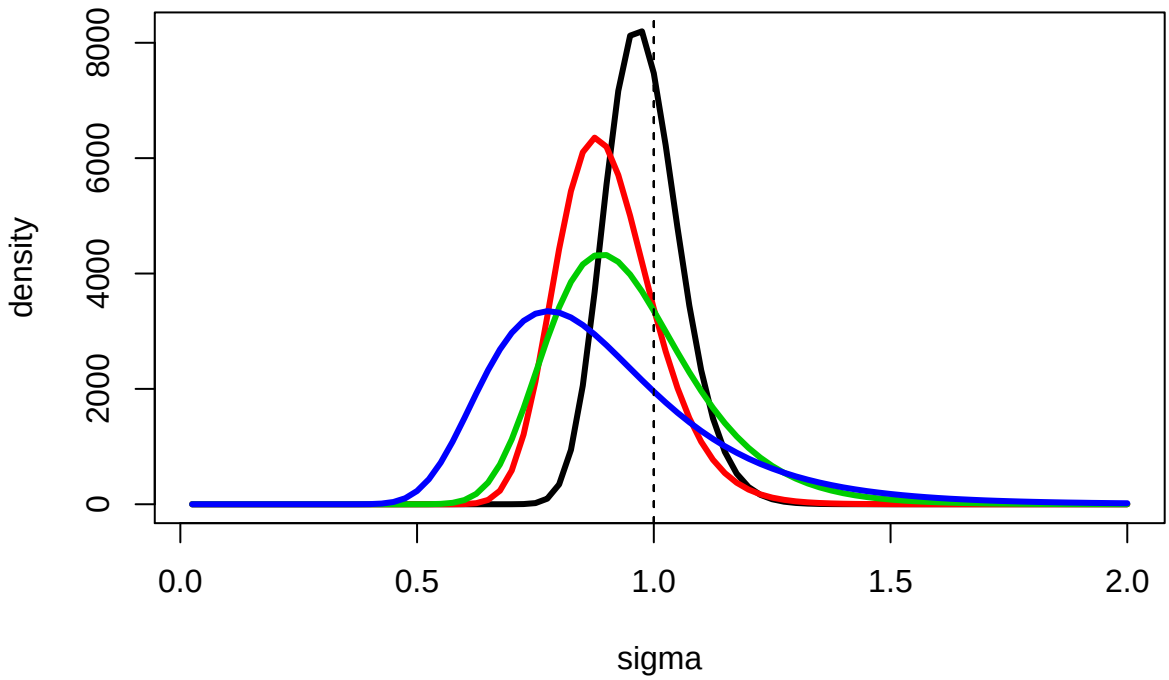

```
#posteriors beta1
for (n in 1:length(nObs)) {
  if (n==1) {
    plot(beta1Grid, margPostBeta1[length(nObs)-n+1,],
         xlab = "beta1", ylab="",
         type = "l", lwd = 3, col=n)
  }
  else {
    points(beta1Grid, margPostBeta1[length(nObs)-n+1,],
           type = "l", lwd = 3, col=n)
  }
  abline(v=beta1, lty=2)
}
```



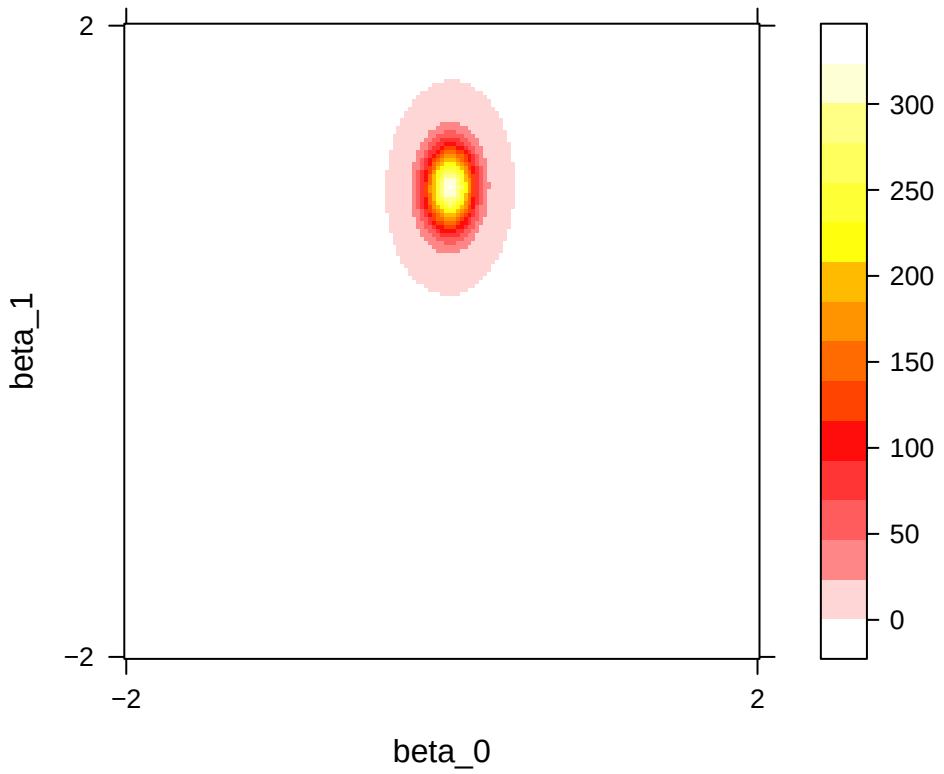

```
#posteriors beta0
for (n in 1:length(nObs)) {
  if (n==1) {
    plot(beta0Grid, margPostBeta0[length(nObs)-n+1,],
         xlab = "beta0", ylab="density",
         type = "l", lwd = 3, col=n)
  }
  else {
    points(beta0Grid, margPostBeta0[length(nObs)-n+1,],
           type = "l", lwd = 3, col=n)
  }
  abline(v=beta0, lty=2)
}
```



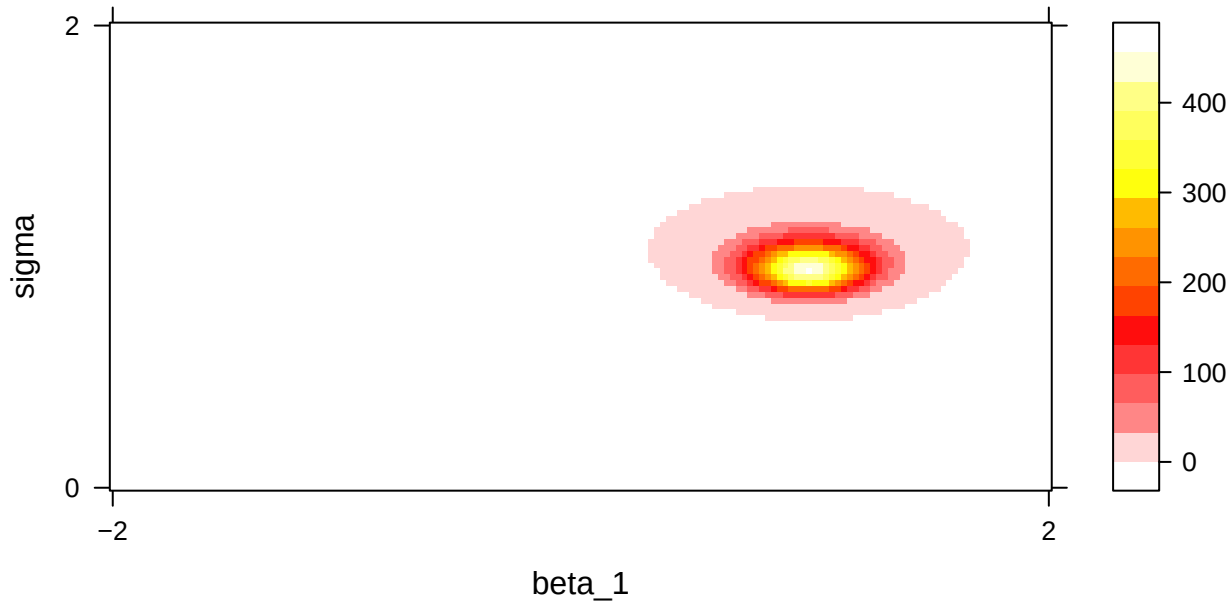

```
#posteriors sigma
for (n in 1:length(nObs)) {
  if (n==1) {
    plot(sigmaGrid, margPostSigma[length(nObs)-n+1,],
         xlab = "sigma", ylab="density",
         type = "l", lwd = 3, col=n)
  }
  else {
    points(sigmaGrid, margPostSigma[length(nObs)-n+1,],
           type = "l", lwd = 3, col=n)
  }
  abline(v=sigma, lty=2)
}
```



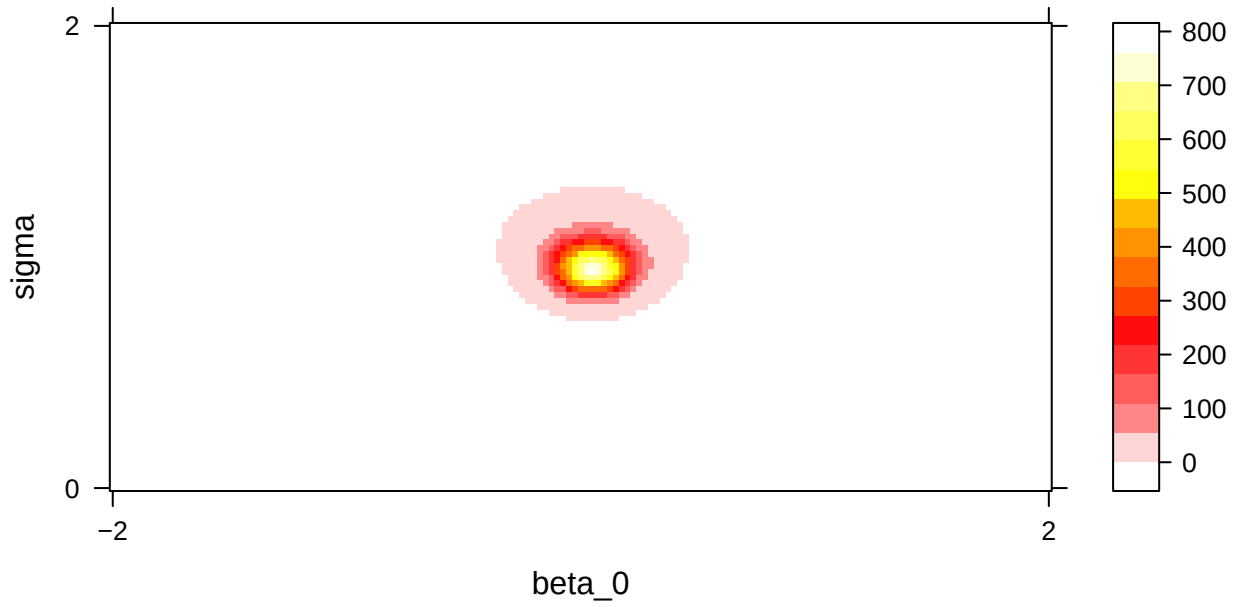

```
#posteriors beta0-beta1
# NOTE: change first dimention of postFinal array to index in nObs vector
#       desired to change num obs in heat maps
library(lattice)
postLocal = postFinal[4,,,]
new.palette=colorRampPalette(c("white","red","yellow","white"),space="rgb")
jointPost = apply(postLocal,c(1,2),sum)
levelplot(jointPost, col.regions=new.palette(20),
          xlab = "beta_0", ylab = "beta_1",
          scales=list(x=list(at=c(1,nBeta0Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nBeta1Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid))))
```



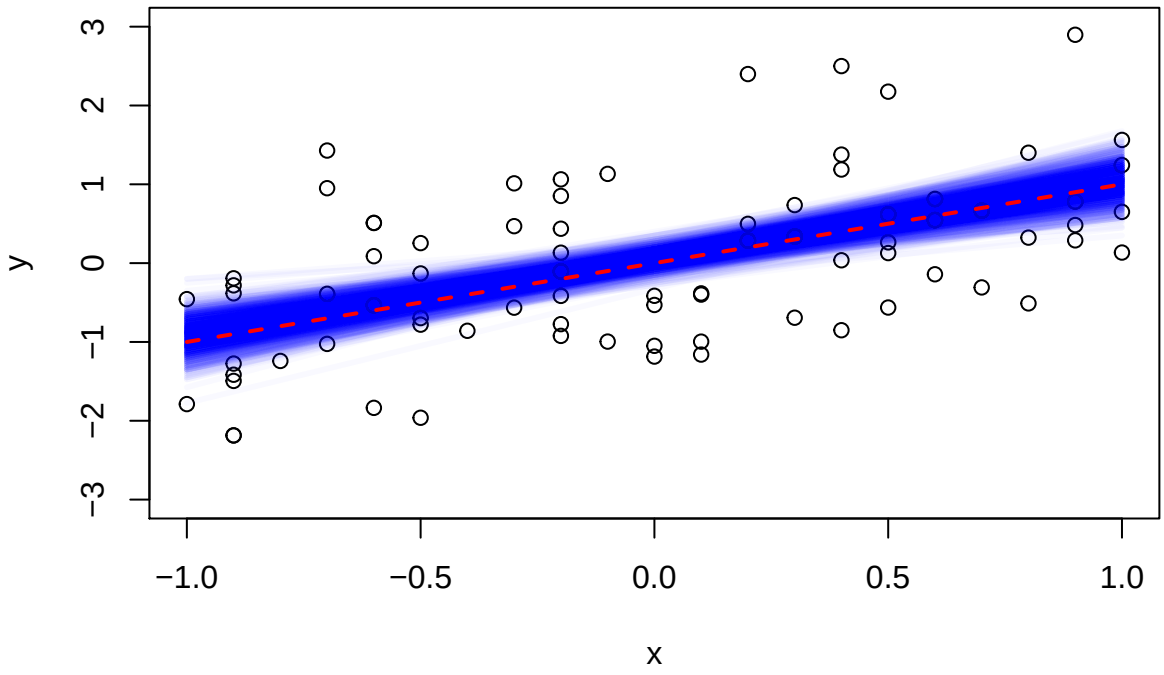

```
#posteriors beta1-sigma
postLocal = postFinal[4,,,]
jointPost = apply(postLocal,c(2,3),sum)
levelplot(jointPost, col.regions=new.palette(20),
          xlab = "beta_1", ylab = "sigma",
          scales=list(x=list(at=c(1,nBeta1Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nSigmaGrid),
                             labels=c(0,boundSigmaGrid))))
```



```
#posteriors beta0-sigma
postLocal = postFinal[4,,,]
jointPost = apply(postLocal,c(1,3),sum)
levelplot(jointPost, col.regions=new.palette(20),
          xlab = "beta_0", ylab = "sigma",
          scales=list(x=list(at=c(1,nBeta0Grid),
                             labels=c(-boundBetaGrid,boundBetaGrid)),
                      y=list(at=c(1,nSigmaGrid),
                             labels=c(0,boundSigmaGrid))))
```




```
#visualize uncertainty in posterior regression lines

#build conditional posteriors
margBeta0 = apply(postFinal[4,,,],c(1),sum)

margBeta1GivenBeta0 = array(rep(-1,nBeta0Grid * nBeta1Grid),
                            dim= c(nBeta0Grid, nBeta1Grid))
for (nBeta0 in 1:nBeta0Grid) {
  margBeta1GivenBeta0[nBeta0,] = apply(postFinal[4,nBeta0,,],c(1),sum)
}

# initialize plot
plot(x,y,xlim=c(-1,1),ylim=c(-3,3))
#plot posterior reg lines
for (sim in 1:1000) {
    b0Index = sample(1:nBeta0Grid, 1, prob=margBeta0)
    b1Index = sample(1:nBeta1Grid, 1, prob=margBeta1GivenBeta0[b0Index,])
    b0Sample = beta0Grid[b0Index]
    b1Sample = beta1Grid[b1Index]
    points(xGrid, b0Sample + b1Sample*xGrid, type="l",lwd=3,
           col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.025))
}
points(xGrid, beta0 + beta1*xGrid, lwd=2, col=2, lty=2, type="l")
```


y
3
2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
x