lecture 4.12 example 4

preliminaries

```
rm(list=ls())
#initalize random seed
set.seed(1563)
```

set simulation parameters

```
nObs = 100
muRange = seq(-1,1,by=0.01)
sigma = 0.25

nPredictPerX = 100
```

build model objects

```
# parameter grid
stepBeta0Grid = 0.05
stepBeta1Grid = 0.05
stepSigmaGrid = 0.05
beta0Grid = seq(-1,1, by = stepBeta0Grid)
beta1Grid = seq(0,2, by = stepBeta1Grid)
sigmaGrid = seq(stepSigmaGrid,2, by = stepSigmaGrid)

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

buildPriorUnivar = function(beta0Grid,betaPGrid,sigmaGrid) {
  # build useful grid objects
  nBeta0Grid = length(beta0Grid)
  nBetaPGrid = length(betaPGrid)
  nSigmaGrid = length(sigmaGrid)
  #
  prior = array( rep(1, nBeta0Grid * nBetaPGrid * nSigmaGrid ),
                 dim = c(nBeta0Grid, nBetaPGrid, nSigmaGrid ))
  #
  for (nB0 in 1:nBeta0Grid) {
    for (nBP in 1:nBetaPGrid) {
      for (nSig in 1:nSigmaGrid) {
```

```
        # change next expression to set different priors
        prior[nB0,nBP, nSig] = 1 / nSig^2
      }
    }
  }
  return(prior)
}

likelihoodUnivar = function(y,xP, b0L, bPL, sL){
  loglike = sum(log(dnorm(y-b0L-bPL*xP, mean = 0, sd=sL)))
  like = exp(loglike)
  return(like)
}

compPostUnivar = function(y,xP, prior, beta0Grid,betaPGrid,sigmaGrid) {
  # build useful grid objects
  nBeta0Grid = length(beta0Grid)
  nBetaPGrid = length(betaPGrid)
  nSigmaGrid = length(sigmaGrid)
  #initialize local posterior
  post = array( rep(-1, nBeta0Grid * nBetaPGrid * nSigmaGrid ),
                dim = c(nBeta0Grid, nBetaPGrid, nSigmaGrid ))
  # compute posterior
  for (nBeta0 in 1:nBeta0Grid) {
    b0 = beta0Grid[nBeta0]
    #print(paste("bo = ", b0))
    for (nBetaP in 1:nBetaPGrid) {
      bP = betaPGrid[nBetaP]
      for (nSigma in 1:nSigmaGrid) {
        s = sigmaGrid[nSigma]
        post[nBeta0,nBetaP,nSigma] = likelihoodUnivar(y,xP,b0,bP,s) * prior[nBeta0,nBetaP,nSigma]
      }
    }
  }
  # normalize posterior
  post = post / (sum(post) * stepSize(beta0Grid) * stepSize(betaPGrid) * stepSize(sigmaGrid))
  # return
  return(post)
}
```

simulate data

```
mus = sample(muRange, nObs, replace = TRUE)
y1 = rnorm(nObs, mean = mus, sd = sigma)
y2 = rnorm(nObs, mean = mus, sd = sigma)
```

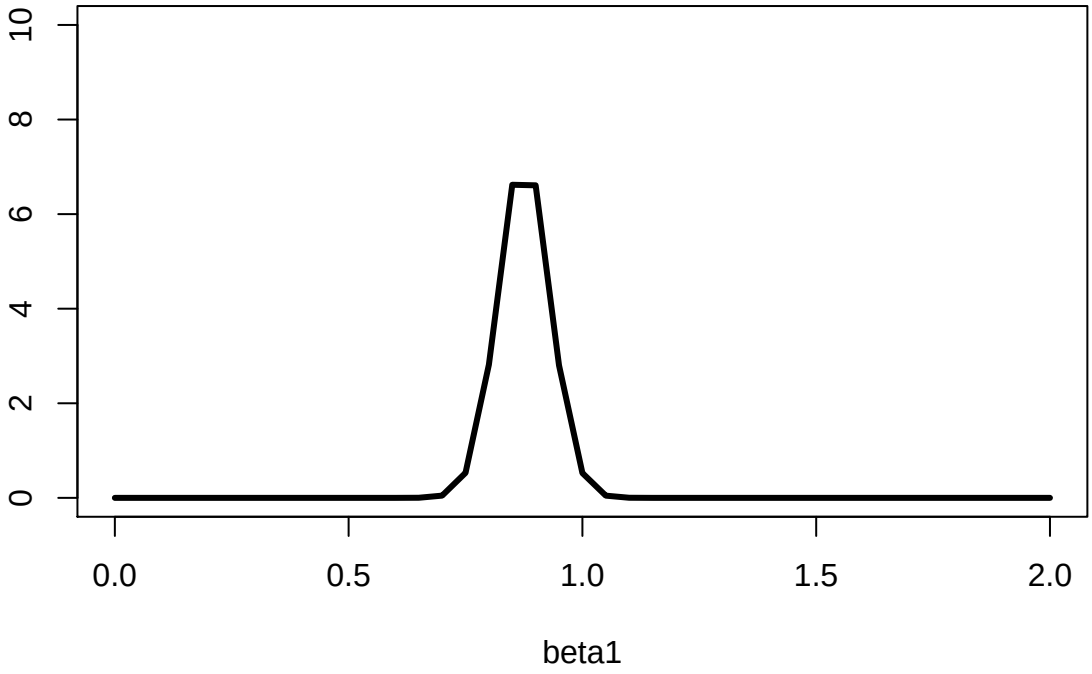
fit model: yDiff ~ N(b0 + b1y1, sigma^2)

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

```r
#compute posterior function iteratively using batches of 100 observations
for (k in 1:floor(nObs/100)) {
  #print(k)
  post = compPostUnivar(y2,y1,prior, beta0Grid,beta1Grid,sigmaGrid)
  prior = post
}

#compute and plot marginal posterior for slope
margPostBeta1 = apply(post,c(2),sum)
margPostBeta1 = margPostBeta1 / (sum(margPostBeta1) * stepSize(beta1Grid))

plot(beta1Grid, margPostBeta1,
     xlab = "beta1", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,10))
```



Use fitted model to predict Y2 based on Y1

```r
#initialize storage arrays
predicted = array(rep(-1, nObs * nPredictPerX),
                  dim = c(nObs, nPredictPerX))

#build conditional posteriors for sampling purposes
margBeta0 = apply(post,c(1),sum)

margBeta1GivenBeta0 = array(rep(-1,length(beta0Grid) * length(beta1Grid)),
                            dim= c(length(beta0Grid), length(beta1Grid)))
for (nBeta0 in 1:length(beta0Grid)) {
  margBeta1GivenBeta0[nBeta0,] = apply(post[nBeta0,,],c(1),sum)
}
```

```
# make predictions
for (t in 1:nObs) {
  xNew = y1[t]
  for (sim in 1:nPredictPerX) {
    b0Index = sample(1:length(beta0Grid), 1, prob=margBeta0)
    b1Index = sample(1:length(beta1Grid), 1, prob=margBeta1GivenBeta0[b0Index,])
    sigIndex = sample(1:length(sigmaGrid), 1, prob=post[b0Index,b1Index,])
    b0Sample = beta0Grid[b0Index]
    b1Sample = beta1Grid[b1Index]
    sigSample = sigmaGrid[sigIndex]
    predicted[t,sim] = rnorm(1,b0Sample + xNew * b1Sample, sigSample)
  }
}

# make plot
meanPredict = apply(predicted,c(1),mean)
plot(y1, meanPredict,
     pch = 19,
     col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25),
     xlab = "y1", ylab = "predicted y2",
     xlim = c(-2,2), ylim = c(-2,2))
for (t in 1:nObs) {
  tenPer = quantile(predicted[t,],probs=0.1)
  ninetyPer = quantile(predicted[t,],probs=0.9)
  segments(y1[t],tenPer , y1[t], ninetyPer ,
           col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25),
           lwd = 3)
}
abline(0,1, lwd=2)
abline(lm(meanPredict ~ y1), lty=2, lwd=2, col="blue")
```