lecture 4.6 example 3

preliminaries

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

specify simulation parameters

```
beta0 = 0.25
beta1 = 0.5
beta2 = 0.5
sigma = 1

nObs = 200

nPredictPerX = 100
```

define functions and 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)
}

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)
}

#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)
}

likelihoodUnivar = function(y,xP, b0L, bPL, sL){
  loglike = sum(log(dnorm(y-b0L-bPL*xP, 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]
    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) * stepSize(s
  # return
  return(post)
}


#compute posterior function
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]
    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)
}

stepBeta0Grid = 0.05
stepBeta1Grid = 0.05
stepBeta2Grid = 0.05
stepSigmaGrid = 0.05
beta0Grid = seq(-1,1, by = stepBeta0Grid)
beta1Grid = seq(-1,1, by = stepBeta1Grid)
beta2Grid = seq(-1,1, by = stepBeta2Grid)
```

```
sigmaGrid = seq(stepSigmaGrid,2, by = stepSigmaGrid)
```

simulate dataset

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

fit models

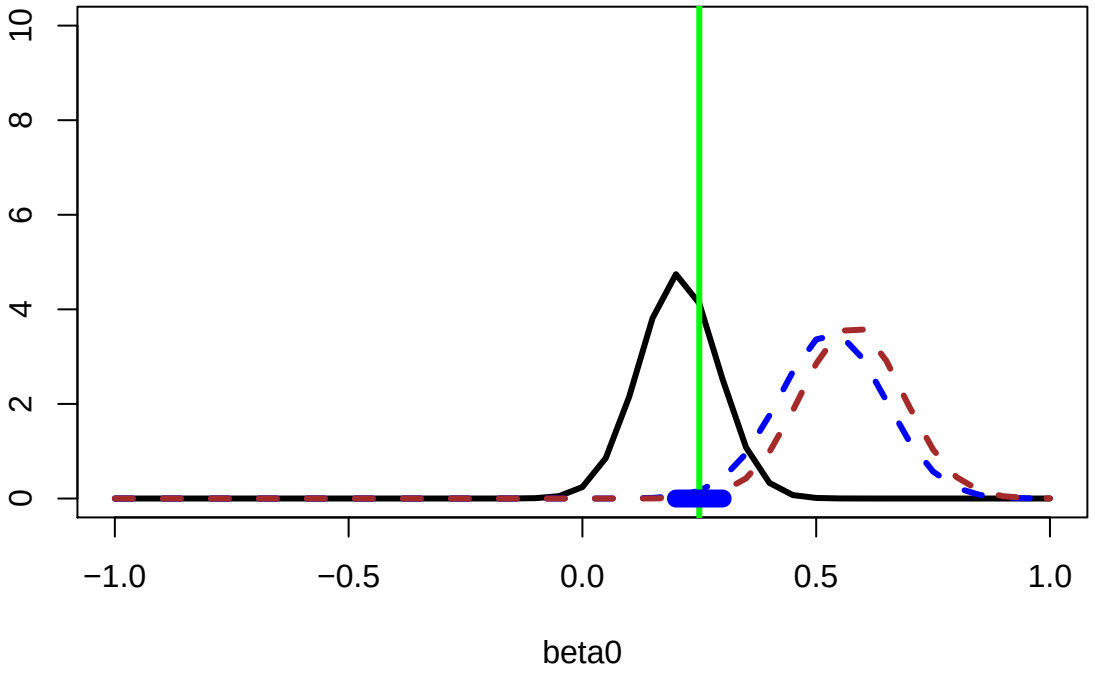
```
# FIT MODEL_1 y ~ bo + b1 * x1
############################

# specify parameter grid for this model
beta1Grid = seq(-1,1, by = stepBeta1Grid)
beta2Grid = c(0)

# build priors
priorM1 = buildPriorUnivar(beta0Grid,beta1Grid,sigmaGrid)

#compute posterior function iteratively using batches of 100 observations
for (k in 1:floor(nObs/100)) {
  y_batch = y[(1+(k-1)*100) : (k * 100)]
  x1_batch = x1[(1+(k-1)*100) : (k * 100)]
  postM1 = compPostUnivar(y_batch,x1_batch,priorM1,
                          beta0Grid,beta1Grid,sigmaGrid)
  priorM1 = postM1
}


#compute marginal posteriors
margPostBeta0M1 = apply(postM1,c(1),sum)
margPostBeta0M1 = margPostBeta0M1 / (sum(margPostBeta0M1) *stepSize(beta0Grid))

margPostBeta1M1 = apply(postM1,c(2),sum)
margPostBeta1M1 = margPostBeta1M1 / (sum(margPostBeta1M1) * stepSize(beta1Grid))

margPostSigmaM1 = apply(postM1,c(3),sum)
margPostSigmaM1 = margPostSigmaM1 / (sum(margPostSigmaM1) * stepSize(sigmaGrid))


# FIT MODEL_2 y ~ bo + b2 * x2
############################

# specify parameter grid for this model
beta1Grid = c(0)
beta2Grid = seq(-1,1, by = stepBeta2Grid)

# build priors
priorM2 = buildPriorUnivar(beta0Grid,beta2Grid,sigmaGrid)
```

```
#compute posterior function iteratively using batches of 100 observations
for (k in 1:floor(nObs/100)) {
  y_batch = y[(1+(k-1)*100) : (k * 100)]
  x2_batch = x2[(1+(k-1)*100) : (k * 100)]
  postM2 = compPostUnivar(y_batch,x2_batch,
                          priorM2, beta0Grid,beta2Grid,sigmaGrid)
  priorM2 = postM2
}


#compute marginal posteriors
margPostBeta0M2 = apply(postM2,c(1),sum)
margPostBeta0M2 = margPostBeta0M2 / (sum(margPostBeta0M2) * stepSize(beta0Grid))

margPostBeta2M2 = apply(postM2,c(2),sum)
margPostBeta2M2 = margPostBeta2M2 / (sum(margPostBeta2M2) * stepSize(beta2Grid))

margPostSigmaM2 = apply(postM2,c(3),sum)
margPostSigmaM2 = margPostSigmaM2 / (sum(margPostSigmaM2) * stepSize(sigmaGrid))


# FIT FULL MODEL_3 y ~ bo + b1 * x1 + b2 * x2
############################

#specify paramter grid for this model
beta1Grid = seq(-1,1, by = stepBeta1Grid)
beta2Grid = seq(-1,1, by = stepBeta2Grid)

# build priors
priorM3 = buildPriorMultivar(beta0Grid,beta1Grid,beta2Grid,sigmaGrid)

#compute posterior function iteratively using batches of 100 observations
for (k in 1:floor(nObs/100)) {
  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)]
  postM3 = compPostMultivar(y_batch,x1_batch,x2_batch,
                            priorM3,beta0Grid,beta1Grid,
                            beta2Grid,sigmaGrid)
  priorM3 = postM3
}

#compute marginal posteriors
margPostBeta0M3 = apply(postM3,c(1),sum)
margPostBeta0M3 = margPostBeta0M3 / (sum(margPostBeta0M3) * stepSize(beta0Grid))

margPostBeta1M3 = apply(postM3,c(2),sum)
margPostBeta1M3 = margPostBeta1M3 / (sum(margPostBeta1M3) * stepSize(beta1Grid))

margPostBeta2M3 = apply(postM3,c(3),sum)
margPostBeta2M3 = margPostBeta2M3 / (sum(margPostBeta2M3) * stepSize(beta2Grid))

margPostSigmaM3 = apply(postM3,c(4),sum)
```

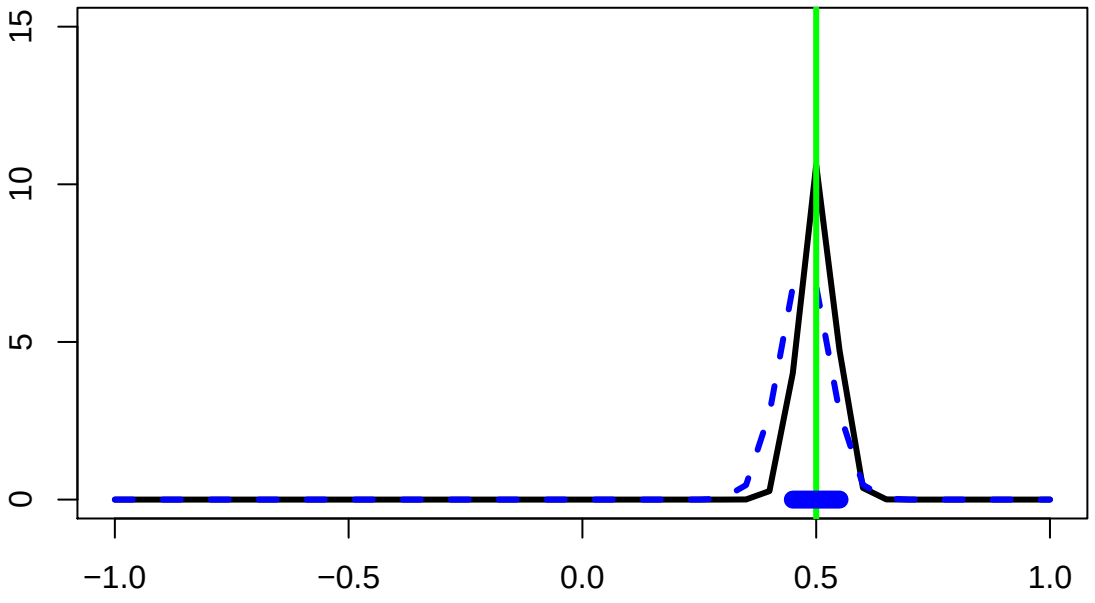
```
margPostSigmaM3 = margPostSigmaM3 / (sum(margPostSigmaM3) * stepSize(sigmaGrid))
```

compare marginal posteriors across models

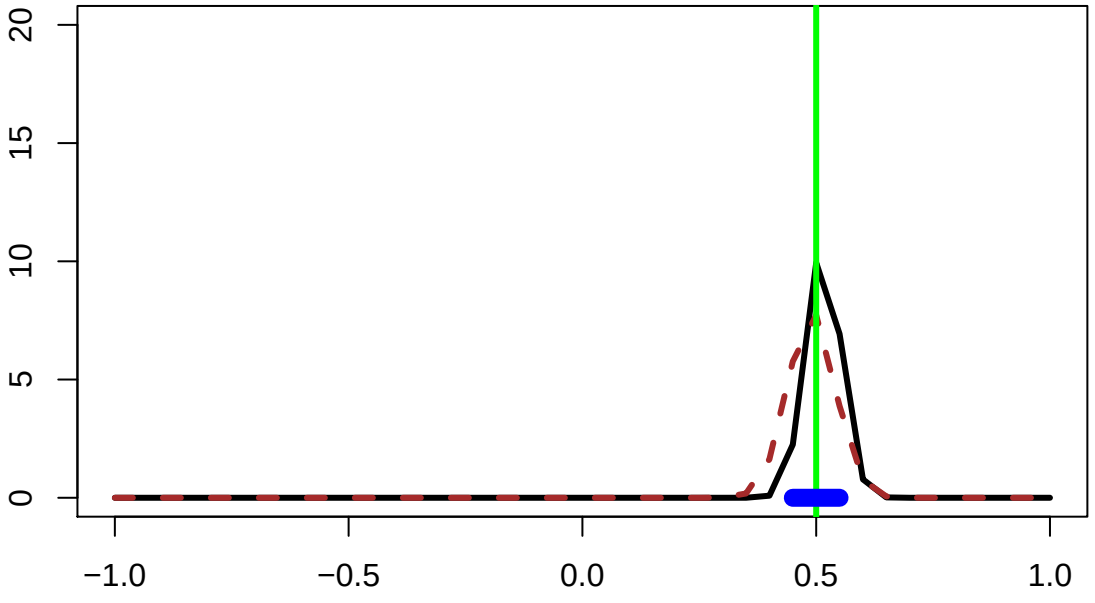
```
plot(beta0Grid, margPostBeta0M3,
     xlab = "beta0", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,10))
points(beta0Grid, margPostBeta0M1,
     type = "l", lwd = 3, col = "blue", lty=2)
points(beta0Grid, margPostBeta0M2,
       type = "l", lwd = 3, col = "brown",lty=2)
abline(v=beta0, lwd=3, col="green")
segments(beta0-stepSize(beta0Grid), 0,beta0+stepSize(beta0Grid),0, lwd=9, col="blue" )
```




```
plot(beta1Grid, margPostBeta1M3,
     xlab = "beta1", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,15))
points(beta1Grid, margPostBeta1M1,
       type = "l", lwd = 3, col = "blue", lty=2)
abline(v=beta1, lwd=3, col="green")
segments(beta1-stepSize(beta1Grid), 0,beta1+stepSize(beta1Grid),0, lwd=9, col="blue" )
```

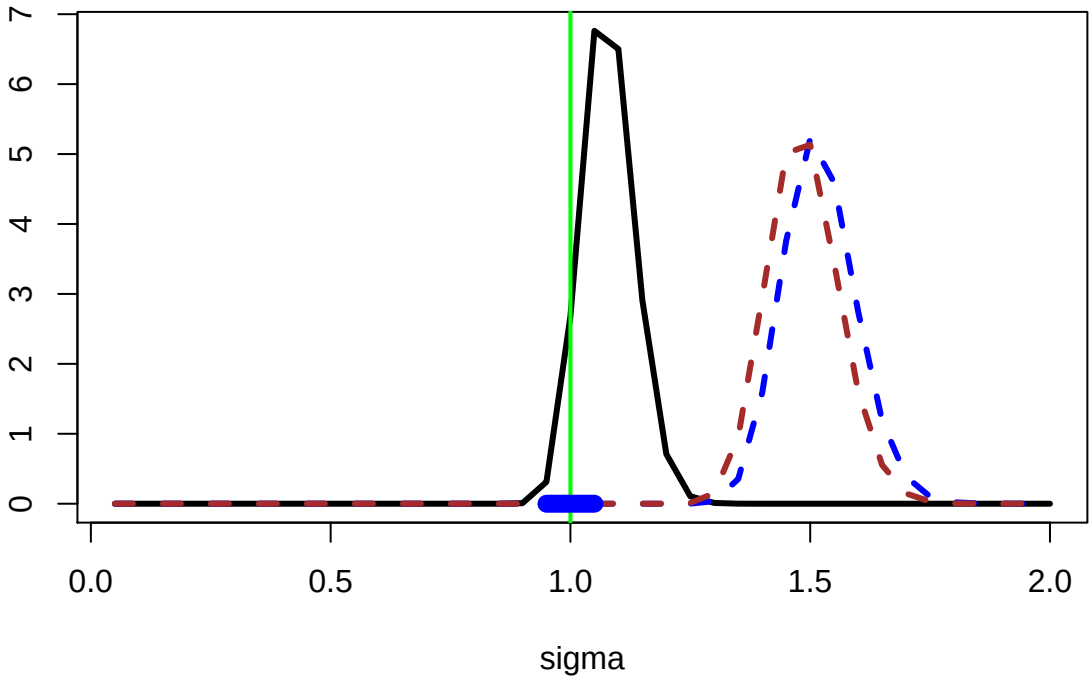



```
plot(beta2Grid, margPostBeta2M3,
     xlab = "beta2", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,20))
points(beta2Grid, margPostBeta2M2,
       type = "l", lwd = 3, col = "brown",lty=2)
abline(v=beta2, lwd=3, col="green")
segments(beta2-stepSize(beta2Grid), 0,beta2+stepSize(beta2Grid),0, lwd=9, col="blue" )
```




```
plot(sigmaGrid, margPostSigmaM3,
     xlab = "sigma", ylab="",
```

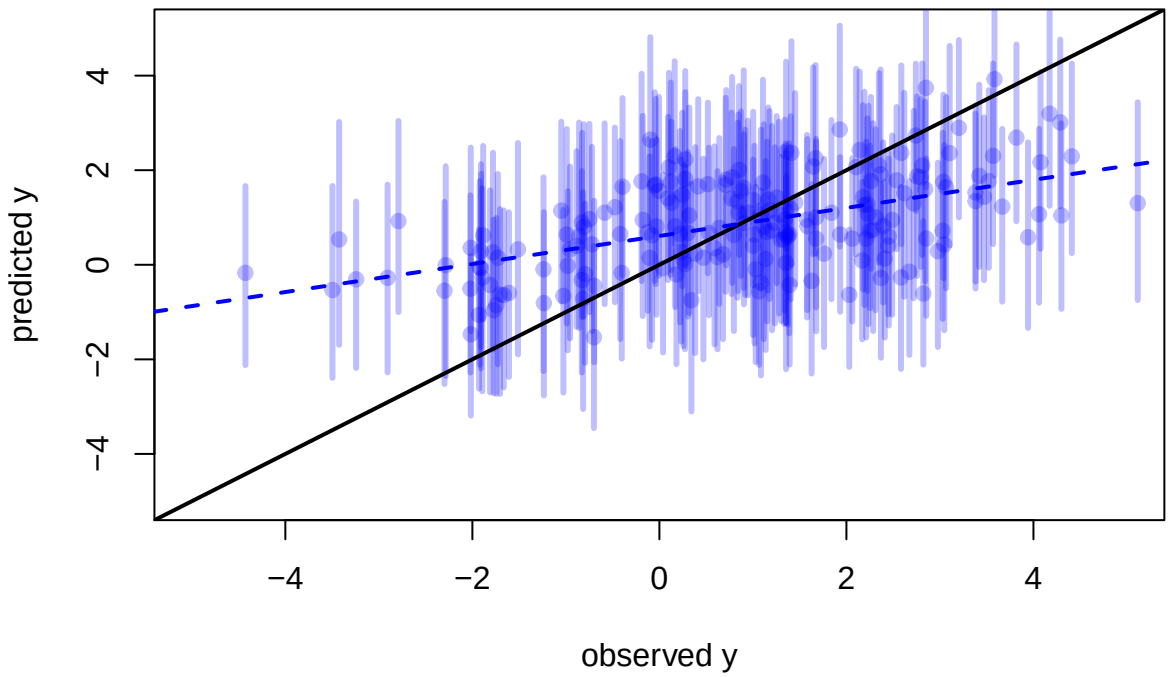
```
     type = "l", lwd = 3)
points(sigmaGrid, margPostSigmaM1,
       type = "l", lwd = 3, col = "blue", lty=2)
points(sigmaGrid, margPostSigmaM2,
       type = "l", lwd = 3, col = "brown",lty=2)
abline(v=sigma, lwd=2, col="green")
segments(sigma-stepSize(sigmaGrid), 0,sigma+stepSize(sigmaGrid),0, lwd=9, col="blue" )
```



Make and plot predictions

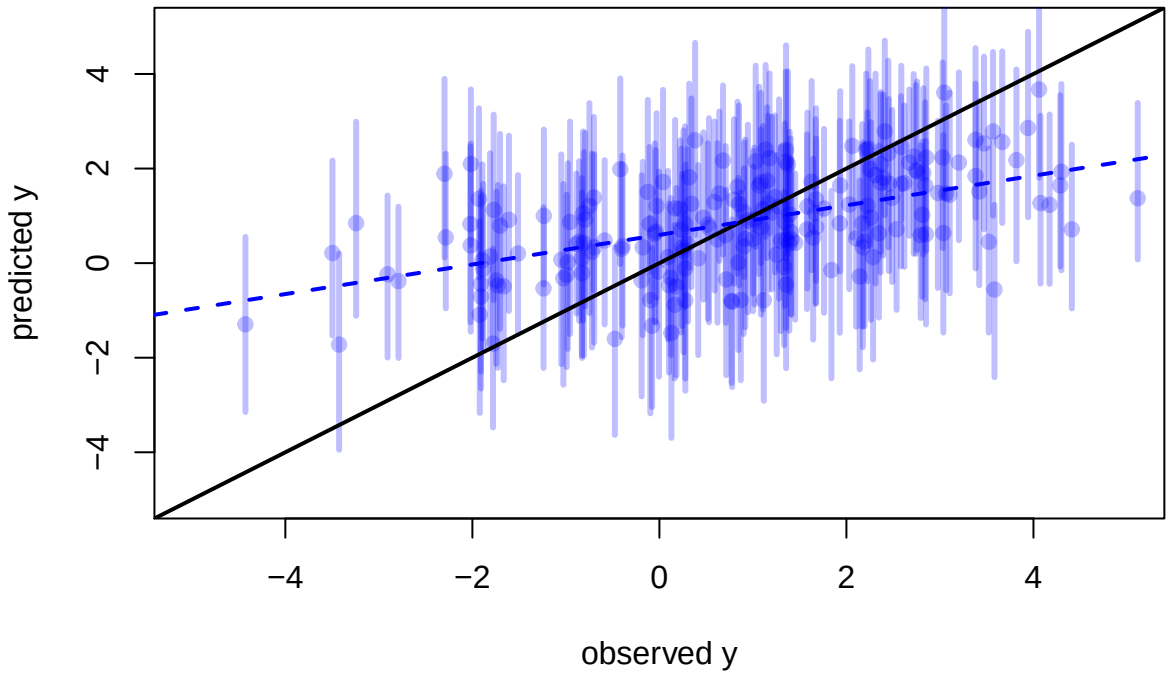
```
# MODEL 1

#initialize storage arrays
predictedM1 = array(rep(-1, nObs * nPredictPerX),
                    dim = c(nObs, nPredictPerX))

#build conditional posteriors for sampling purposes
margBeta0 = apply(postM1,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(postM1[nBeta0,,],c(1),sum)
}

# make predictions
for (t in 1:nObs) {
  xNew = x1[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=postM1[b0Index,b1Index,])
    b0Sample = beta0Grid[b0Index]
    b1Sample = beta1Grid[b1Index]
    sigSample = sigmaGrid[sigIndex]
    predictedM1[t,sim] = rnorm(1,b0Sample + xNew * b1Sample, sigSample)
  }
}

# make plot
meanPredict = apply(predictedM1,c(1),mean)
plot(y, meanPredict,
     pch = 19,
     col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25),
     xlab = "observed y", ylab = "predicted y",
     xlim = c(-5,5), ylim = c(-5,5))
for (t in 1:nObs) {
  tenPer = quantile(predictedM1[t,],probs=0.1)
  ninetyPer = quantile(predictedM1[t,],probs=0.9)
  segments(y[t],tenPer , y[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 ~ y), lty=2, lwd=2, col="blue")
```



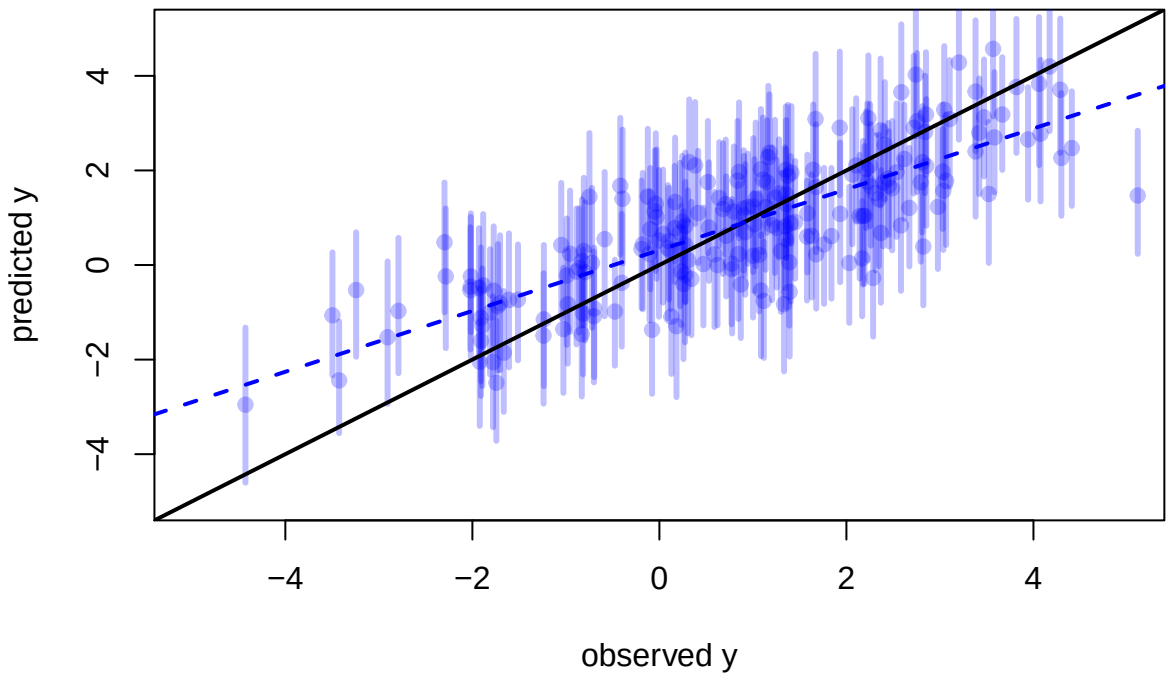

```
# MODEL 2

#initialize storage arrays
predictedM2 = array(rep(-1, nObs * nPredictPerX),
                    dim = c(nObs, nPredictPerX))
```

```
#build conditional posteriors for sampling purposes
margBeta0 = apply(postM2,c(1),sum)

margBeta2GivenBeta0 = array(rep(-1,length(beta0Grid) * length(beta2Grid)),
                            dim= c(length(beta0Grid), length(beta2Grid)))
for (nBeta0 in 1:length(beta0Grid)) {
  margBeta2GivenBeta0[nBeta0,] = apply(postM2[nBeta0,,],c(1),sum)
}

# make predictions
for (t in 1:nObs) {
  xNew = x2[t]
  for (sim in 1:nPredictPerX) {
    b0Index = sample(1:length(beta0Grid), 1, prob=margBeta0)
    b1Index = sample(1:length(beta1Grid), 1, prob=margBeta2GivenBeta0[b0Index,])
    sigIndex = sample(1:length(sigmaGrid), 1, prob=postM2[b0Index,b1Index,])
    b0Sample = beta0Grid[b0Index]
    b1Sample = beta1Grid[b1Index]
    sigSample = sigmaGrid[sigIndex]
    predictedM2[t,sim] = rnorm(1,b0Sample + xNew * b1Sample, sigSample)
  }
}

# make plot
meanPredict = apply(predictedM2,c(1),mean)
plot(y, meanPredict,
     pch = 19,
     col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25),
     xlab = "observed y", ylab = "predicted y",
     xlim = c(-5,5), ylim = c(-5,5))
for (t in 1:nObs) {
  tenPer = quantile(predictedM2[t,],probs=0.1)
  ninetyPer = quantile(predictedM2[t,],probs=0.9)
  segments(y[t],tenPer , y[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 ~ y), lty=2, lwd=2, col="blue")
```




```
# MODEL 3

#initialize storage arrays
predictedM3 = array(rep(-1, nObs * nPredictPerX),
                    dim = c(nObs, nPredictPerX))

#build conditional posteriors for sampling purposes
margBeta0 = apply(postM3,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(postM3[nBeta0,,,],c(1),sum)
}

margBeta2GivenBeta0Beta1 = array(rep(-1,length(beta0Grid) * length(beta1Grid)) * length(beta2Grid),
                            dim= c(length(beta0Grid), length(beta1Grid), length(beta2Grid)))
for (nBeta0 in 1:length(beta0Grid)) {
  for (nBeta1 in 1:length(beta1Grid))
    margBeta2GivenBeta0Beta1[nBeta0,nBeta1,] = apply(postM3[nBeta0,nBeta1,,],c(1),sum)
}

# make predictions
for (t in 1:nObs) {
  x1New = x1[t]
  x2New = x2[t]
  for (sim in 1:nPredictPerX) {
    b0Index = sample(1:length(beta0Grid), 1, prob=margBeta0)
    b1Index = sample(1:length(beta1Grid), 1, prob=margBeta1GivenBeta0[b0Index,])
    b2Index = sample(1:length(beta2Grid), 1, prob=margBeta2GivenBeta0Beta1[b0Index,b1Index,])
    sigIndex = sample(1:length(sigmaGrid), 1, prob=postM3[b0Index,b1Index,b2Index,])
    b0Sample = beta0Grid[b0Index]
```

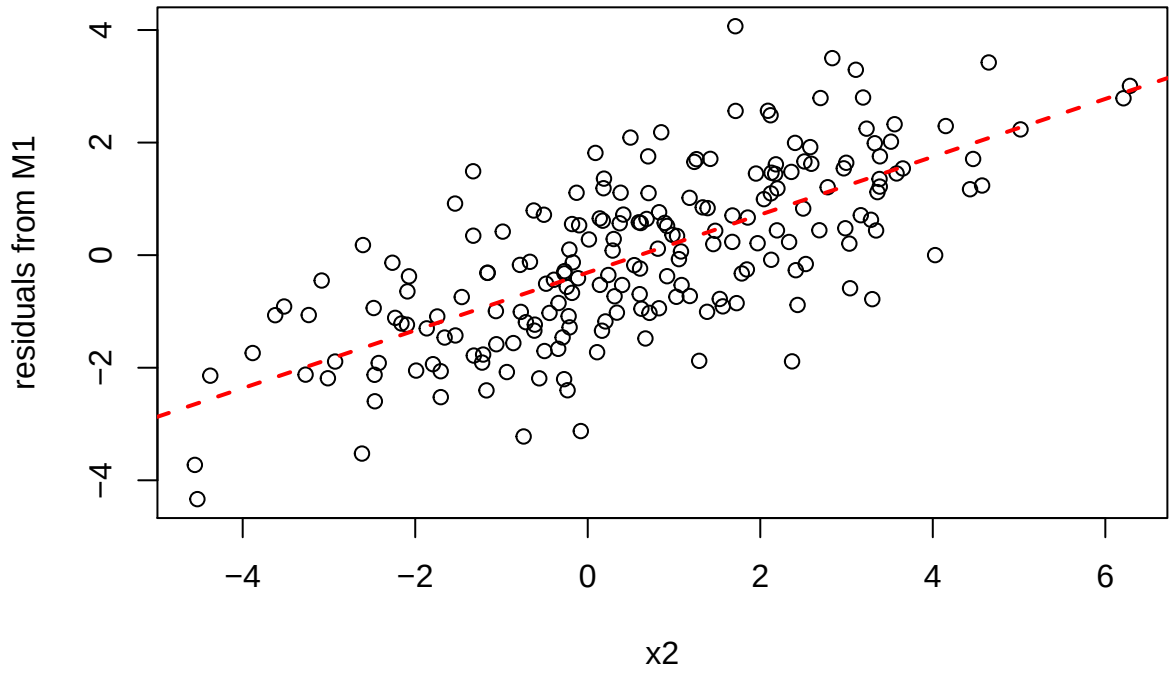
```
    b1Sample = beta1Grid[b1Index]
    b2Sample = beta2Grid[b2Index]
    sigSample = sigmaGrid[sigIndex]
    predictedM3[t,sim] = rnorm(1,b0Sample + x1New * b1Sample + x2New * b2Sample, sigSample)
  }
}

# make plot yobserves vs y predicted
meanPredict = apply(predictedM3,c(1),mean)
plot(y, meanPredict,
     pch = 19,
     col=rgb(red=0.0, green=0.0, blue=1.0, alpha=0.25),
     xlab = "observed y", ylab = "predicted y",
     xlim = c(-5,5), ylim = c(-5,5))
for (t in 1:nObs) {
  tenPer = quantile(predictedM3[t,],probs=0.1)
  ninetyPer = quantile(predictedM3[t,],probs=0.9)
  segments(y[t],tenPer , y[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 ~ y), lty=2, lwd=2, col="blue")
```



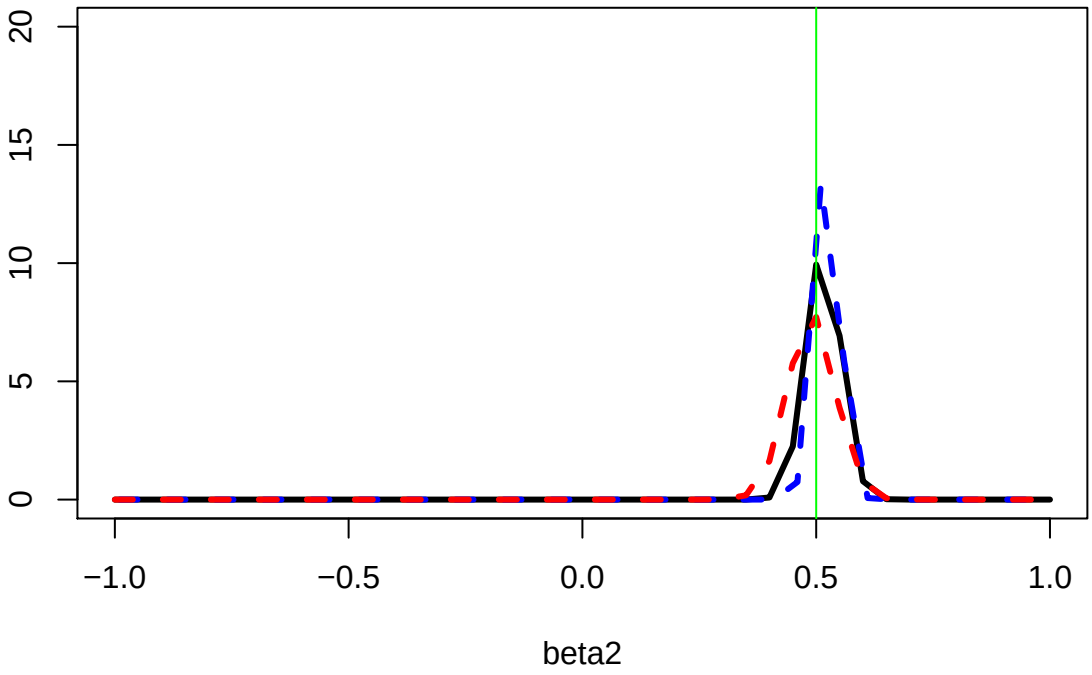
Two stage regression: M1, then residuals on X2

```
# compute residuals for M1 (using classical regression method)
m1 = lm(y ~ x1)

residM1 = resid(m1)
```

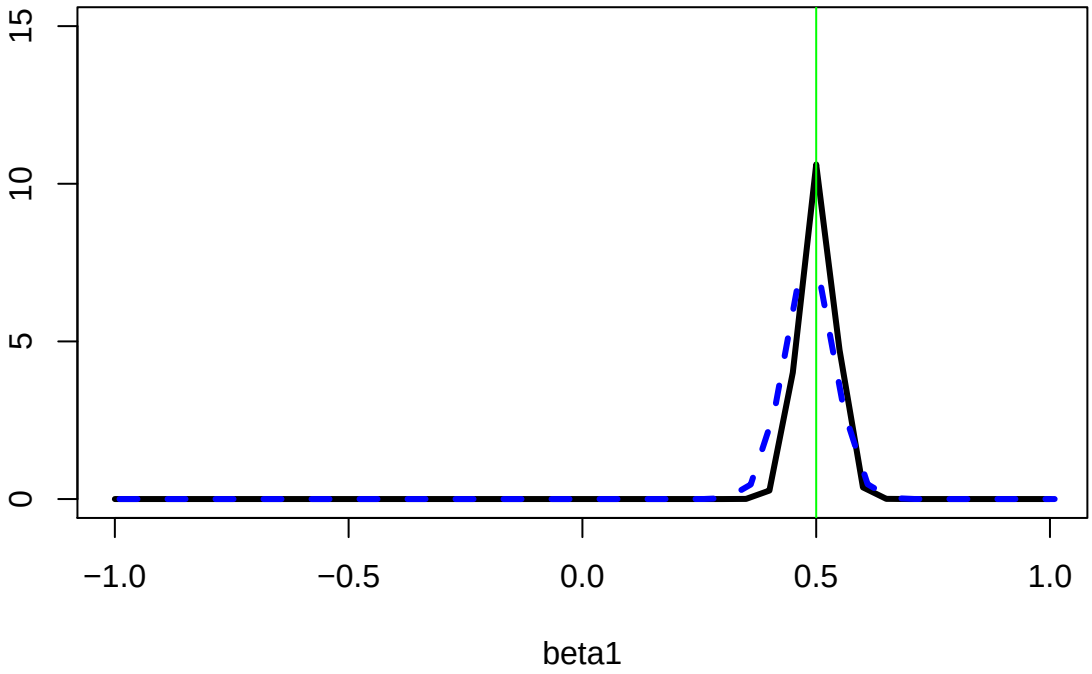
```r
# FIT residM1 ~ bo + b2 * xs
prior1Then2 = buildPriorUnivar(beta0Grid,beta2Grid,sigmaGrid)

for (k in 1:floor(nObs/100)) {
  post1Then2 = compPostUnivar(residM1,x2,prior1Then2, beta0Grid,beta2Grid,sigmaGrid)
  prior1Then2 = post1Then2
}

margPostBeta0_1Then2 = apply(post1Then2,c(1),sum)
margPostBeta0_1Then2 = margPostBeta0_1Then2 / (sum(margPostBeta0_1Then2) * stepSize(beta0Grid))

margPostBeta2_1Then2 = apply(post1Then2,c(2),sum)
margPostBeta2_1Then2 = margPostBeta2_1Then2 / (sum(margPostBeta2_1Then2) *stepSize(beta2Grid))

margPostSigma_1Then2 = apply(post1Then2,c(3),sum)
margPostSigma_1Then2 = margPostSigma_1Then2 / (sum(margPostSigma_1Then2) * stepSize(sigmaGrid))


plot(x2,residM1,
     ylab = "residuals from M1")
abline(lm(residM1~x2), col="red", lwd=2, lty=2)
```




```r
plot(beta2Grid, margPostBeta2M3,
     xlab = "beta2", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,20))
points(beta2Grid+0.01, margPostBeta2_1Then2,
       type = "l", lwd = 3, col = "blue", lty=2)
points(beta2Grid, margPostBeta2M2,
       type = "l", lwd = 3, col = "red",lty=2)
abline(v=beta2, lwd=1, col="green")
```

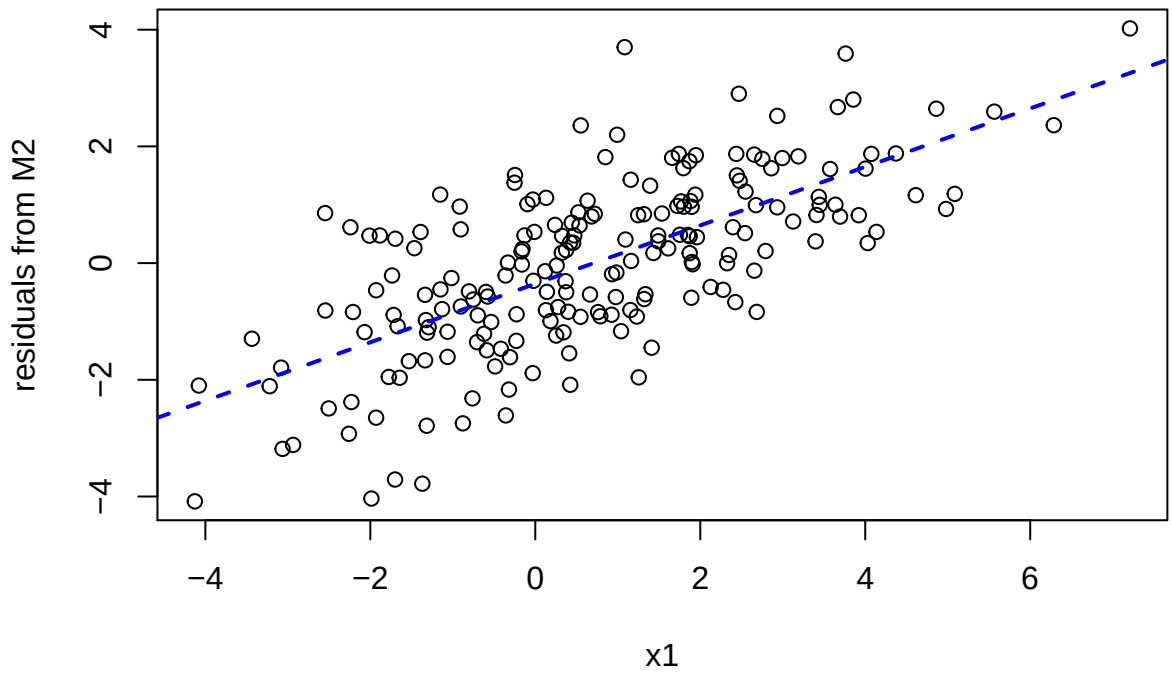



```
plot(beta1Grid, margPostBeta1M3,
     xlab = "beta1", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,15))
points(beta1Grid + 0.01, margPostBeta1M1,
       type = "l", lwd = 3, col = "blue", lty=2)
abline(v=beta1, lwd=1, col="green")
```



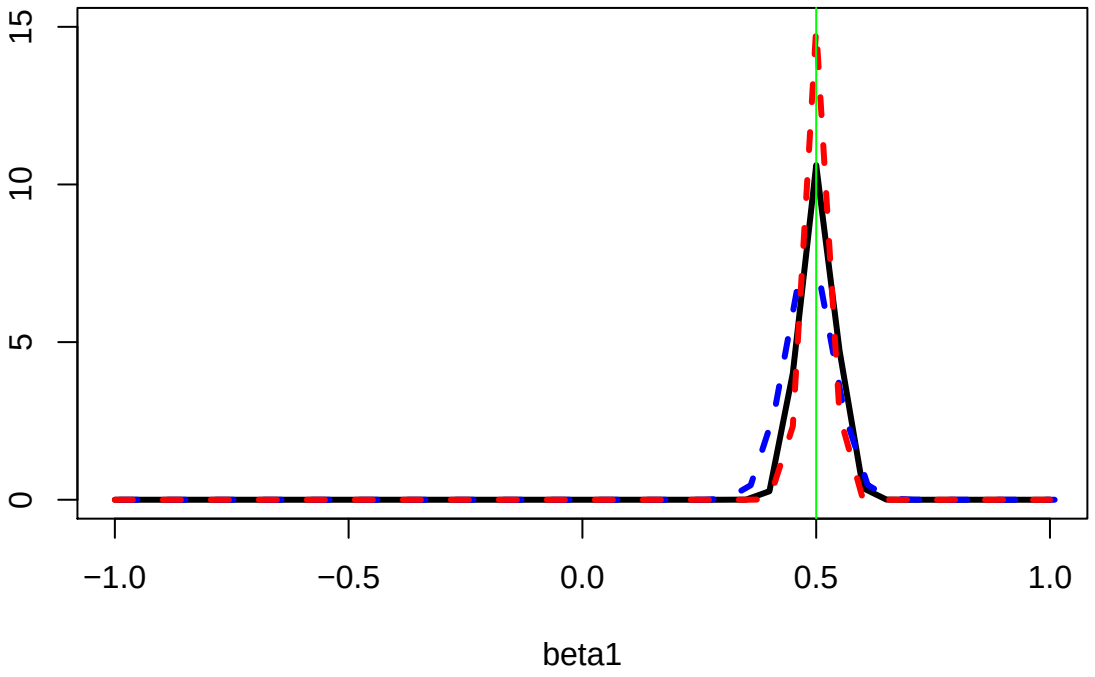
Two stage regression: M2, then residuals on x1

```
# compute residuals for M2 (using classical regression method)
m2 = lm(y ~ x2)
residM2 = resid(m2)

# FIT residM2 ~ bo + b1 * xs
prior2Then1 = buildPriorUnivar(beta0Grid,beta1Grid,sigmaGrid)

for (k in 1:floor(nObs/100)) {
  post2Then1 = compPostUnivar(residM2,x1,prior2Then1, beta0Grid,beta1Grid,sigmaGrid)
  prior2Then1 = post2Then1
}

margPostBeta0_2Then1 = apply(post2Then1,c(1),sum)
margPostBeta0_2Then1 = margPostBeta0_2Then1 / (sum(margPostBeta0_2Then1) * stepSize(beta0Grid))

margPostBeta1_2Then1 = apply(post2Then1,c(2),sum)
margPostBeta1_2Then1 = margPostBeta1_2Then1 / (sum(margPostBeta1_2Then1) * stepSize(beta1Grid))

margPostSigma_2Then1 = apply(post2Then1,c(3),sum)
margPostSigma_2Then1 = margPostSigma_2Then1 / (sum(margPostSigma_2Then1) *stepSize(sigmaGrid))

plot(x1,residM2,
     ylab = "residuals from M2")
abline(lm(residM2~x1), col="blue", lwd=2, lty=2)
```

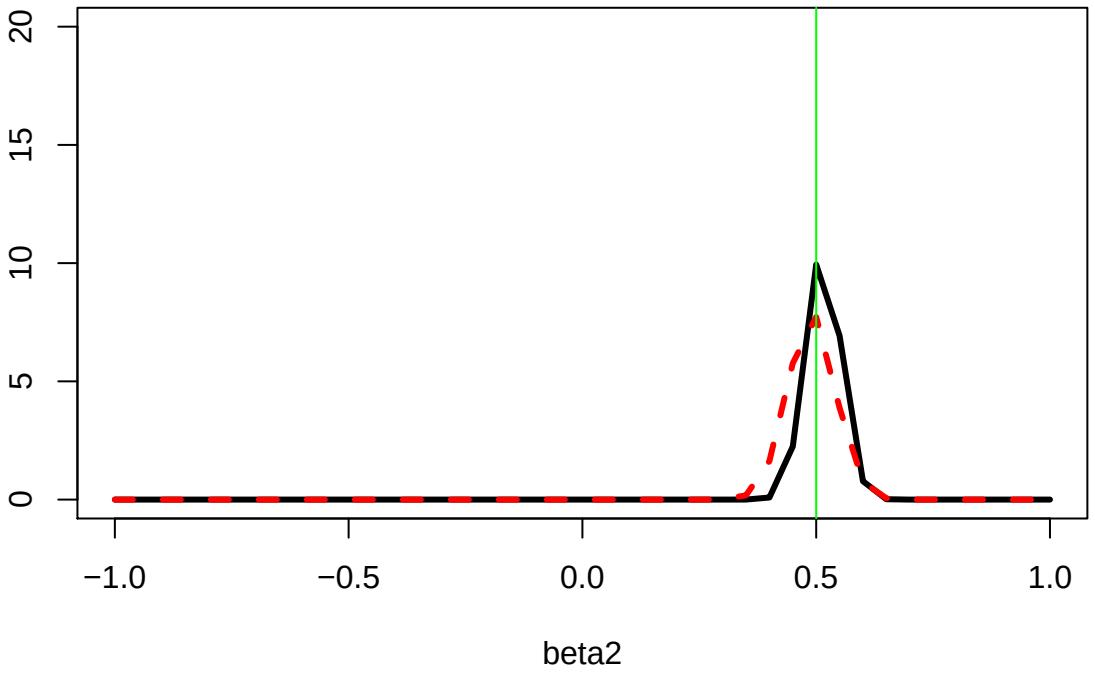



```
plot(beta1Grid, margPostBeta1M3,
     xlab = "beta1", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,15))
points(beta1Grid + 0.01, margPostBeta1M1,
```

```
       type = "l", lwd = 3, col = "blue", lty=2)
points(beta1Grid, margPostBeta1_2Then1,
       type = "l", lwd = 3, col = "red", lty=2)
abline(v=beta1, lwd=1, col="green")
```




```
plot(beta2Grid, margPostBeta2M3,
     xlab = "beta2", ylab="",
     type = "l", lwd = 3,
     ylim=c(0,20))
points(beta2Grid, margPostBeta2M2,
       type = "l", lwd = 3, col = "red",lty=2)
abline(v=beta2, lwd=1, col="green")
```


20
15
10
5
0
−1.0
−0.5
0.0
0.5
1.0
beta2