HW08

Zach White

November 4, 2016

Problem 9.3

Part A

```
g = n = nrow(data)

nu.0 = 2
sigma2.0 = 1
y = data$y
X = as.matrix(data[,-1])
n.samp = 10000

## Sigma2
p = ncol(X)
Hg = (g/(g+1)) * X %*% solve(t(X) %*% X) %*% t(X)
In = diag(1,n)
SSRg = t(y) %*% (In - Hg) %*% y

sigma2 = 1 / rgamma(n.samp, (nu.0 + n)/2, (nu.0*sigma2.0 + SSRg)/2)
## Betas
var.beta = (g/(g+1)) * ( solve(t(X) %*% X) )
mean.beta = (g/(g+1)) *(solve(t(X) %*% X) %*% t(X) %*% y)
E = matrix(rnorm(n.samp*p,0,sqrt(sigma2)),n.samp,p)
beta = t( t(E %*% chol(var.beta)) + c(mean.beta))

names(beta) = colnames(X)
apply(beta,2,quantile,c(.025,.975))
```

```
##                M         So        Ed         Po1        Po2         LF
## 2.5%  0.02576758 -0.3362082 0.2186987 -0.01151208 -2.3153924 -0.3501962
## 97.5% 0.53255221  0.3333508 0.8637961  2.91958434  0.7630941  0.2133887
##              M.F        Pop        NW          U1         U2        GDP
## 2.5%  -0.1516793 -0.2970966 -0.2012595 -0.61661756 0.03408779 -0.2404916
## 97.5%  0.4086671  0.1613657  0.4217334  0.08161821 0.69125145  0.7064579
##             Ineq        Prob       Time
## 2.5%  0.2876513 -0.52491239 -0.2967608
## 97.5% 1.1284366 -0.03910257  0.1750205
```

```
apply(beta,2,mean)
```

```
##            M           So           Ed          Po1          Po2
##  0.280807775 -0.001602821  0.535518546  1.453187855 -0.779230042
##           LF          M.F          Pop           NW           U1
## -0.065466532  0.128363241 -0.068386707  0.108737104 -0.266305757
##           U2          GDP         Ineq         Prob         Time
##  0.362487544  0.231037700  0.710221168 -0.281442701 -0.062973185
```

```r
ols = lm(y~.,data= data)
coef(ols)
```

```
##   (Intercept)             M            So            Ed           Po1
## -0.0004581088  0.2865181425 -0.0001140461  0.5445140840  1.4716210675
##           Po2            LF           M.F           Pop            NW
## -0.7817801428 -0.0659645619  0.1312980228 -0.0702919266  0.1090566856
##            U1            U2           GDP          Ineq          Prob
## -0.2705364468  0.3687303043  0.2380594756  0.7262919898 -0.2852264319
##          Time
## -0.0615768625
```

```r
beta.hat.ols = solve(t(X) %*% X) %*% t(X) %*% y
#longhand.pred = as.matrix(test.X) %*% beta.hat.ols
```

Some comments on the comparison. Specifically, which variabes seem very predictive. Look at the credible intervals.

Part B

```r
train = sample(n,n/2)
train.set = data[train,]
test.set = data[-train,]
train.y = train.set$y
train.X = train.set[,-1]
test.y = test.set$y
test.X = test.set[,-1]

# OLS on training, predicted values
train.lm = lm(y~., data = train.set)
pred.y.ols =predict.lm(train.lm,test.X)
# OR longhand
train.lm.beta0 = train.lm$coefficients[1]
train.lm.beta = train.lm$coefficients[-1]
long.pred = train.lm.beta0+ (as.matrix(test.X) %*% train.lm.beta)
train.X = as.matrix(train.X)
test.X = as.matrix(test.X)
beta.hat.ols = solve(t(train.X) %*% train.X) %*% t(train.X) %*% train.y
longhand.pred = as.matrix(test.X) %*% beta.hat.ols

long.ols.error = sum((test.y-longhand.pred)^2)
ols.error = sum((test.y-pred.y.ols)^2)

# Bayesian on training,
g = n = nrow(train.set)
train.X = as.matrix(train.X)
test.X = as.matrix(test.X)

nu.0 = 2
sigma2.0 = 1

n.samp = 10000
```

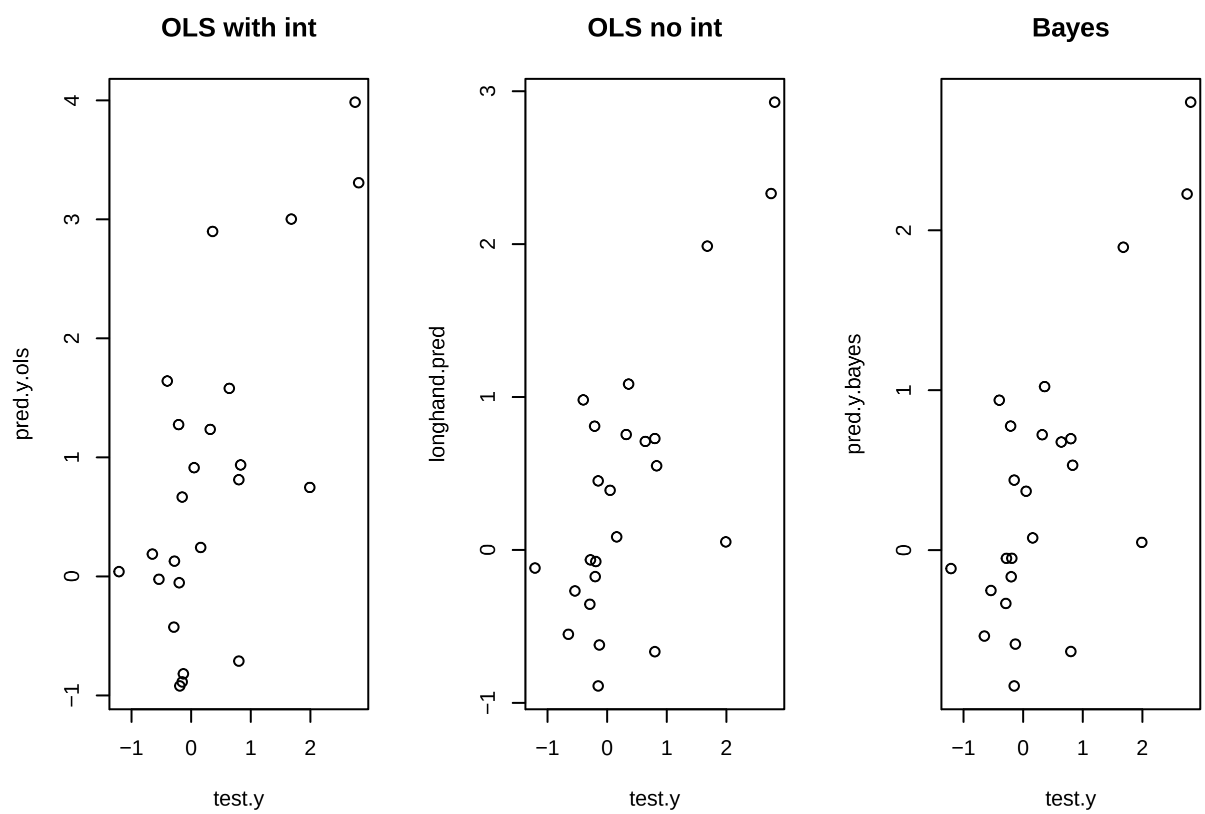
```
## Sigma2
p = ncol(train.X)
Hg = (g/(g+1)) * train.X %*% solve(t(train.X) %*% train.X) %*% t(train.X)
In = diag(1,n)
SSRg = t(train.y) %*% (In - Hg) %*% train.y

sigma2 = 1 / rgamma(n.samp, (nu.0 + n)/2, (nu.0*sigma2.0 + SSRg)/2)
## Betas
var.beta = (g/(g+1)) * ( solve(t(train.X) %*% train.X) )
mean.beta = (g/(g+1)) *(solve(t(train.X) %*% train.X) %*% t(train.X) %*% train.y)
E = matrix(rnorm(n.samp*p,0,sqrt(sigma2)),n.samp,p)
beta = t( t(E %*% chol(var.beta)) + c(mean.beta))
mean.beta = apply(beta,2,mean)

pred.y.bayes = test.X %*% mean.beta

bayes.error = sum((test.y - pred.y.bayes)^2)

par(mfrow = c(1,3))
plot(test.y,pred.y.ols, main = "OLS with int")
plot(test.y,longhand.pred, main = "OLS no int")
plot(test.y,pred.y.bayes,main = "Bayes")
```




```
long.ols.error
```

```
## [1] 12.54439
```

```
ols.error
```

```
## [1] 27.61405
```

```
bayes.error
```

```
## [1] 12.12994
```

Note: I report three errors because in the initial data, we assume that the data has been centered and scaled so it has a mean of zero and a variance of 1. When data are used like this, a regression model doesn't include a term for β_0 because it should be 0 in theory. However, even in the initial model, the coefficient for $\beta_0 > 0$, and so I chose to include it to see the discrepancies. Also, something of note is that since we are taking random samples in our cross validation, there is a relatively high probability that each sample won't have a mean of 0, and thus not including β_0 isn't necessarily a great idea. For this reason, I will report all three errors both in this problem and the next.

When we perform this analysis, we get our Bayes coefficients and the ones from OLS without an intercept are pretty similar, and they are much lower than if we were to include an intercept. However, this might just be due to our random sample, which is why we have our next question.

Part C

```
n.tot = nrow(data)
n.iter = 10000
error.int.ols = error.noint.ols = error.bayes = rep(0,n.iter)
# Prior Values
g = n = nrow(data) /2
nu.0 = 2
sigma2.0 = 1
n.samp = 10000

for(i in 1:n.iter){
  train = sample(n.tot,n.tot/2)
  train.set = data[train,]
  test.set = data[-train,]
  train.y = train.set$y
  train.X = as.matrix(train.set[,-1])
  test.y = test.set$y
  test.X = as.matrix(test.set[,-1])
  # OLS with intercept
  train.lm = lm(y~.,data = train.set)
  y.int.ols = predict.lm(train.lm,as.data.frame(test.X))
  # OLS without intercept
  beta.hat.ols = solve(t(train.X) %*% train.X) %*% t(train.X) %*% train.y
  y.noint = test.X %*% beta.hat.ols
  # Bayes
  ## Sigma2
  p = ncol(train.X)
  Hg = (g/(g+1)) * train.X %*% solve(t(train.X) %*% train.X) %*% t(train.X)
  In = diag(1,n)
  SSRg = t(train.y) %*% (In - Hg) %*% train.y

  sigma2 = 1 / rgamma(n.samp, (nu.0 + n)/2, (nu.0*sigma2.0 + SSRg)/2)
  ## Betas
```

```r
  var.beta = (g/(g+1)) * ( solve(t(train.X) %*% train.X) )
  mean.beta = var.beta %*% t(train.X) %*% train.y
  E = matrix(rnorm(n.samp*p,0,sqrt(sigma2)),n.samp,p)
  beta = t( t(E %*% chol(var.beta)) + c(mean.beta))
  mean.beta = apply(beta,2,mean)

  pred.y.bayes = test.X %*% mean.beta

  error.bayes[i] = sum((test.y - pred.y.bayes)^2)
  error.int.ols[i] = sum((test.y - y.int.ols)^2)
  error.noint.ols[i] = sum((test.y - y.noint)^2)

}

mean(error.bayes)
```

```
## [1] 22.44567
```

```r
mean(error.int.ols)
```

```
## [1] 29.45686
```

```r
mean(error.noint.ols)
```

```
## [1] 23.68114
```

```r
quantile(error.bayes,c(.025,.975))
```

```
##      2.5%     97.5%
##  9.826914 53.606254
```

```r
quantile(error.int.ols,c(.025,.975))
```

```
##     2.5%    97.5%
## 11.57757 77.14694
```

```r
quantile(error.noint.ols,c(.025,.975))
```

```
##     2.5%    97.5%
## 10.20691 57.58951
```

AFter repeated sampling, we can see that the Bayesian regression actually has the lowest predictive error, followed by the OLS without the intercept and then the OLS with the intercept.