XIAO_Lin_Solutions_lab9

Lin XIAO

November 12, 2015

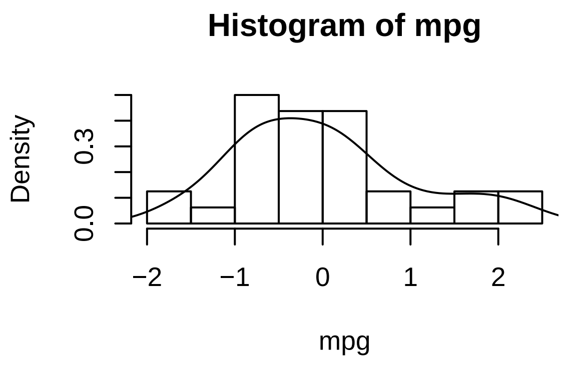
The main goal of the lab today is to fit a model with using the LASSO and the Bayesian LASSO and making a comparison. Please complete the following:

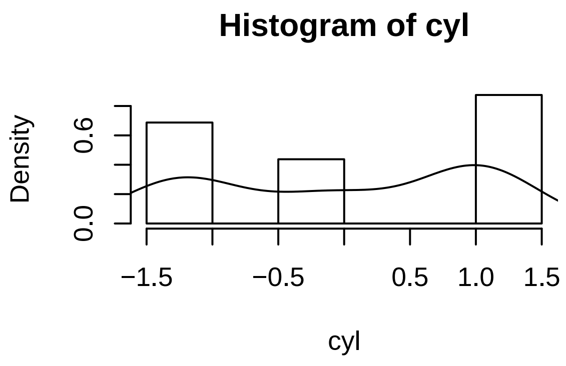
1. Read in the mtcars data from the **datasets** library. Next use the scale function around this dataset so that we can make an even comparison between the two methods. Doing so means that any normalization/ standardization parameter in the functions later should be **FALSE**. For the rest of the analysis use this scaled dataset.

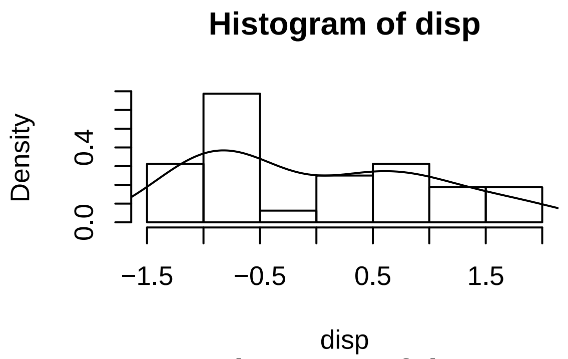
```
data(mtcars)

# Scale the dataset and keep it
mtcars <- as.matrix(as.data.frame(scale(mtcars)))
```

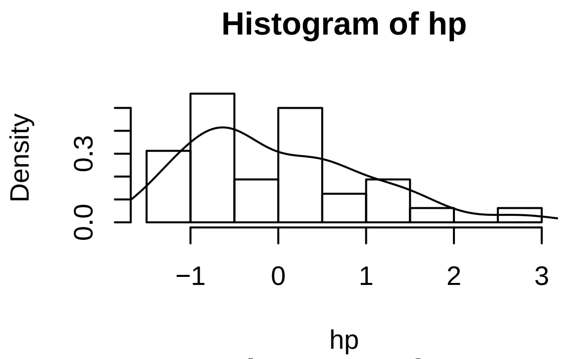
2. Perform univariate and bi-variate (including scatter plots) EDA and describe any features that you might want to investigate in the analysis. Your goal is to build a prediction model for **mpg** based on the entire dataset.

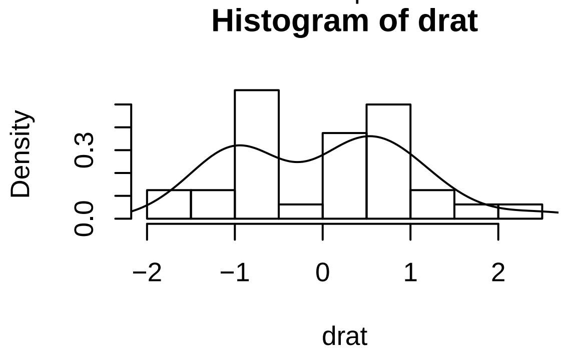
```
par(mfrow=c(2,2))

# Draw histogram and density plot for each variable
for(i in 1:ncol(mtcars)){
   hist(mtcars[,i], freq=F,
        main = paste("Histogram of", colnames(mtcars)[i], sep = " "),
        xlab = colnames(mtcars)[i])
   lines(density(mtcars[,i]))
}
```

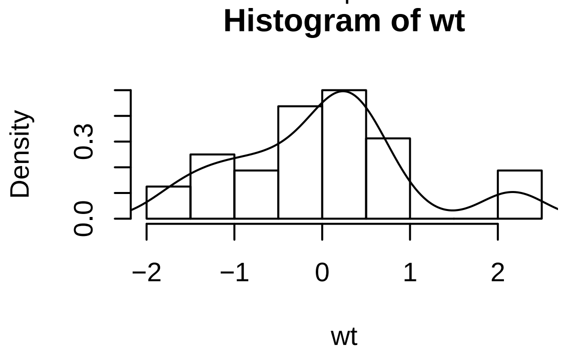


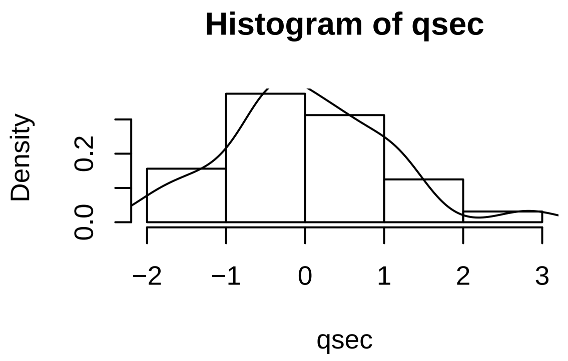


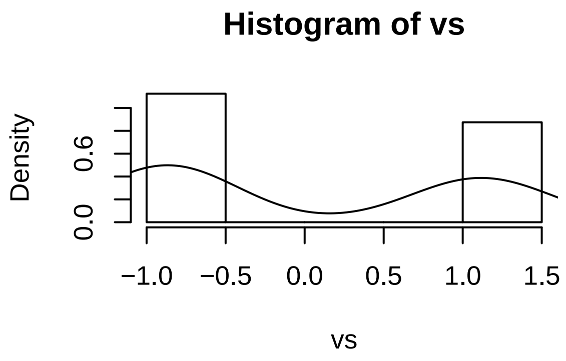




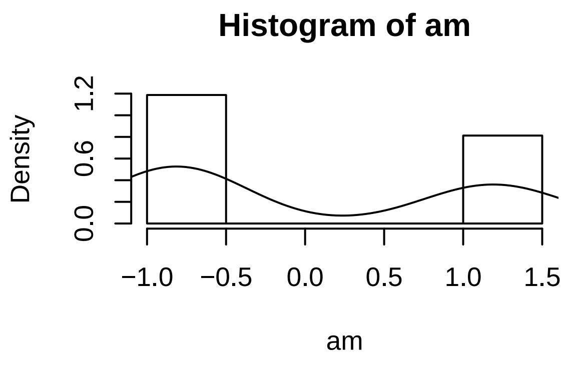


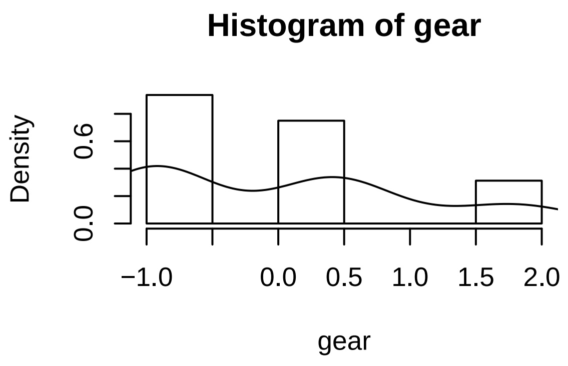


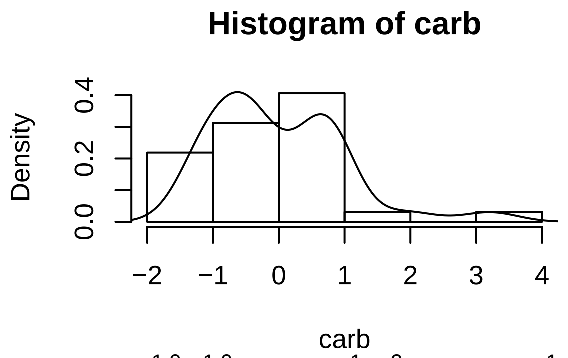


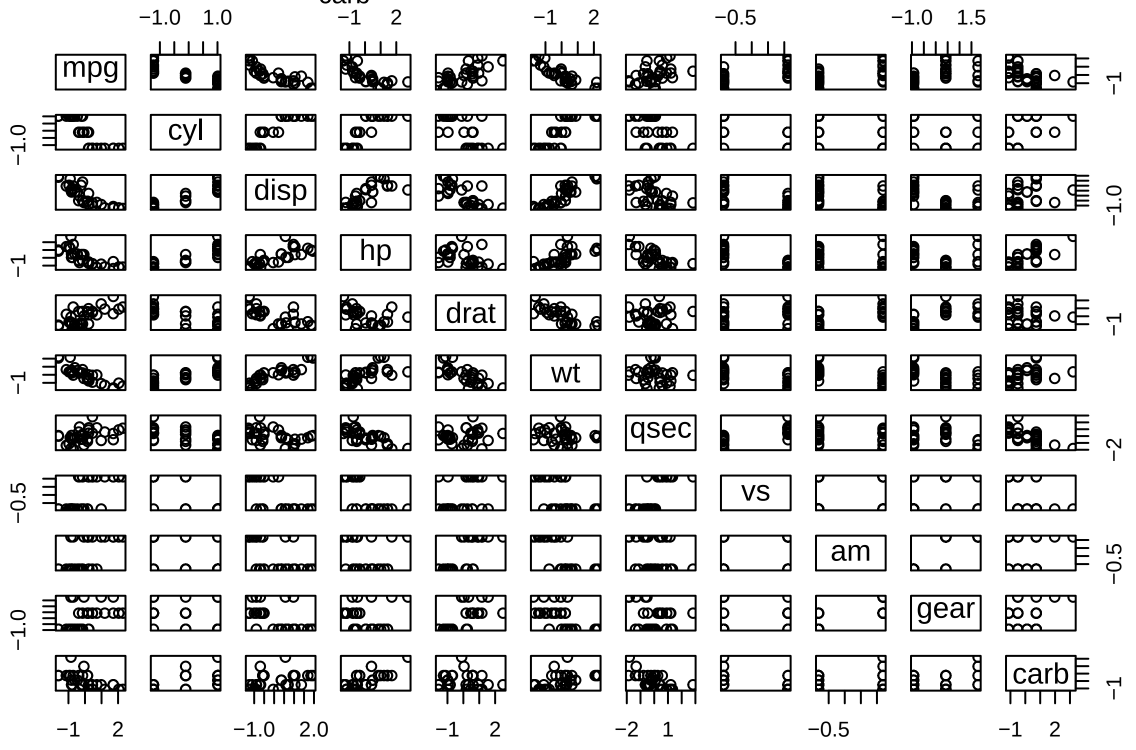


```
# Plot correlation matrix
pairs(mtcars)
```









Based on these plots, there are correlations among some predictors. With the goal of predicting mpg, we can see that mpg is negatively related to disp, hp, wt and so on while positively related to drat and qsec.

3. Fit a LASSO model using **glmnet**. Present the cross validation error curve, along with the choice of lambda set to the MSE minimizing value. Present your coefficients for this model using **xtable**

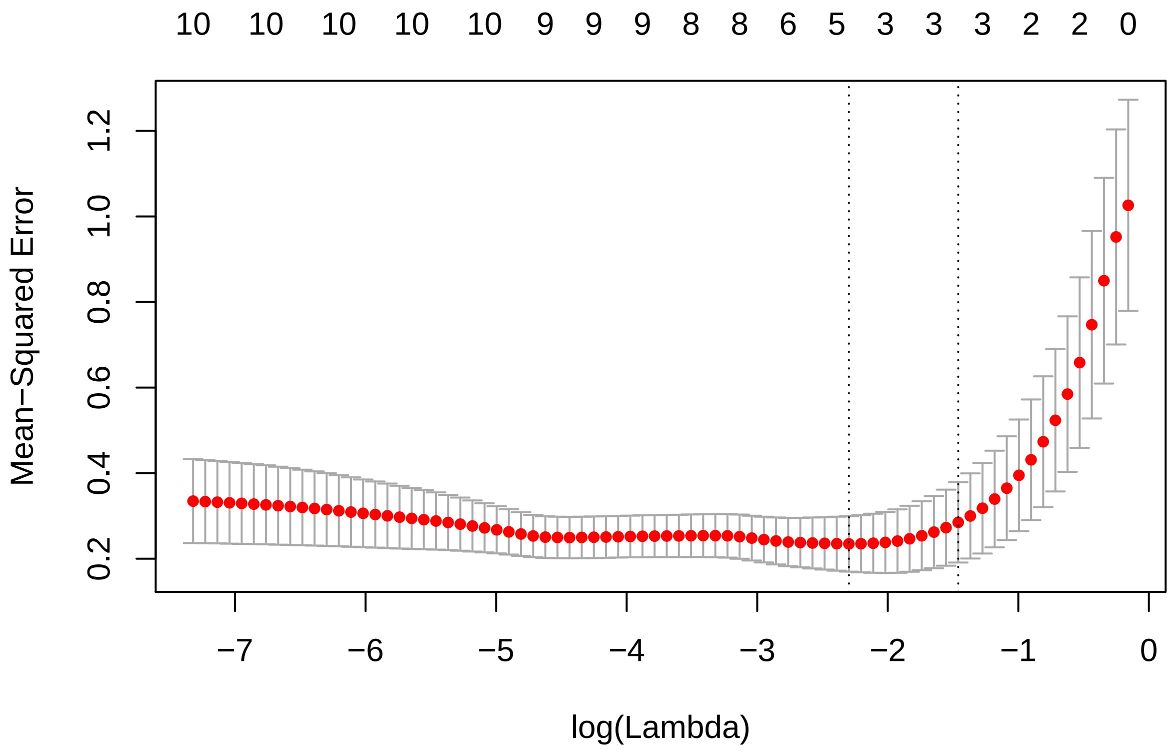
```r
require(glmnet)
```

```
## Loading required package: glmnet
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-2
```

```r
require(xtable)
```

```
## Loading required package: xtable
```

```r
cv_lasso_model <- cv.glmnet(x = as.matrix(mtcars[,-1]),
                      y = as.numeric(mtcars[,1]), alpha = 1)

# Cross validation error curve with MSE.min
plot(cv_lasso_model)
```



```r
lasso_model <- glmnet(x = as.matrix(mtcars[,-1]),
            y = as.numeric(mtcars[,1]), alpha = 1, lambda = cv_lasso_model$lambda.min)

# Coefficients for this model
coef_lasso <- as.vector(coef(lasso_model))


# XTABLE
dat <- data.frame(coef_lasso)
names(dat) <- c("Coefficient of cv lasso")
tb <- xtable(dat)
```

```
digits(tb)[2] <- 6
print(tb, comment=F)
```

	Coefficient of cv lasso
1	0.000000
2	-0.259658
3	0.000000
4	-0.156745
5	0.000000
6	-0.447953
7	0.000000
8	0.000000
9	0.013905
10	0.000000
11	-0.003704

4. Fit a bayesian LASSO using monomvn, with the blasso function. Return the posterior means of your coefficients. (**Hint**: Remember to include the unpenalized intercept estimate.)What do you notice between the LASSO and the Bayesian LASSO in terms of coefficient estimates - (Please present these using xtable as well). Why do you think this is the case?

```
require(monomvn)
```

```
## Loading required package: monomvn
## Loading required package: pls
##
## Attaching package: 'pls'
##
## The following object is masked from 'package:stats':
##
##     loadings
##
## Loading required package: lars
## Loaded lars 1.2
##
## Loading required package: MASS
```

```
# Bayesian LASSO
blasso_model <- blasso(X = as.matrix(mtcars[,-1]),
                       y = as.numeric(mtcars[,1]), T = 5000)
```

t=100, m=8 t=200, m=6 t=300, m=6 t=400, m=6 t=500, m=8 t=600, m=6 t=700, m=7 t=800, m=8 t=900, m=6 t=1000, m=5 t=1100, m=4 t=1200, m=7 t=1300, m=6 t=1400, m=5 t=1500, m=7 t=1600, m=7 t=1700, m=6 t=1800, m=6 t=1900, m=6 t=2000, m=6 t=2100, m=6 t=2200, m=7 t=2300, m=3 t=2400, m=5 t=2500, m=5 t=2600, m=8 t=2700, m=8 t=2800, m=6 t=2900, m=8 t=3000, m=7 t=3100, m=8 t=3200, m=6 t=3300, m=5 t=3400, m=5 t=3500, m=6 t=3600, m=6 t=3700, m=5 t=3800, m=7 t=3900, m=4 t=4000, m=7 t=4100, m=7 t=4200, m=7 t=4300, m=6 t=4400, m=7 t=4500, m=5 t=4600, m=6 t=4700, m=6 t=4800, m=5 t=4900, m=8

```
# Set the number of samples we want to burn in
burn <- 1000
```

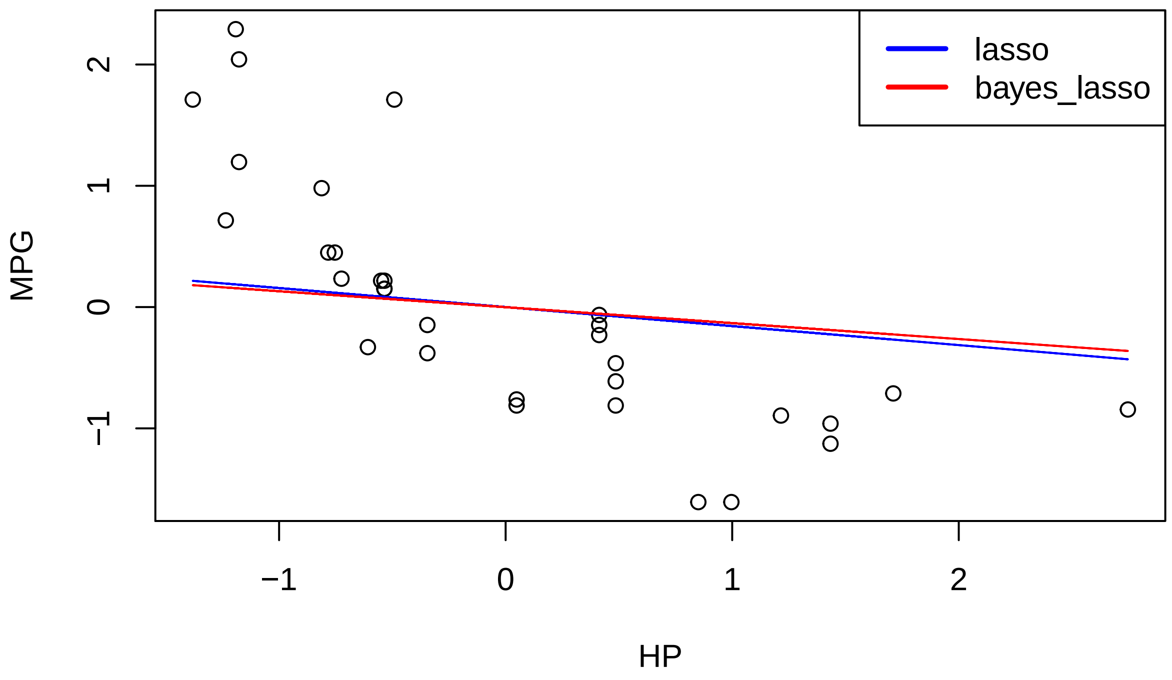
```r
# Use posterior mean of the burn-in samples as both
# intercept and penalized coefficient estimates
interc <- mean(blasso_model$mu[-(1:burn)])
beta <- apply(blasso_model$beta[-(1:burn),], 2, mean)
coef_blasso <- as.vector(c(interc, beta))

# XTABLE
dat <- data.frame(coef_lasso, coef_blasso)
names(dat) <- c("Coefficient of cv lasso","Coefficient of bayesian lasso")
tb <- xtable(dat)
digits(tb)[2:3] <- c(6,6)
print(tb, comment=F)
```

	Coefficient of cv lasso	Coefficient of bayesian lasso
1	0.000000	-0.000833
2	-0.259658	-0.134995
3	0.000000	-0.069233
4	-0.156745	-0.131374
5	0.000000	0.051123
6	-0.447953	-0.364277
7	0.000000	0.046818
8	0.000000	0.034822
9	0.013905	0.086981
10	0.000000	0.029323
11	-0.003704	-0.086513

LASSO is shrinking some of the coefficients to 0, but it's not the case for Bayesian LASSO. LASSO gives an estimated variance of 0 for predictors with $\beta = 0$. The "Bayesian lasso" of Park and Casella provides valid standard errors for β and provides more stable point estimates. The Bayesian Lasso estimates seem to be a compromise between the Lasso and ridge regression estimates.

5. Holding all other values at their mean levels (and this may not make sense for some variables, but do so anyway), create a plot of **MPG** on the vertical axis and **HP** on the horizontal. Holding all other variables at their mean levels, create a regression line based on your two models. Visually what looks better?

```r
# Create a column of 1's for intercept
x <- cbind(rep(1,nrow(mtcars)))

# Binding all the predictor columns and setting them as mean values except HP
for(i in c(2,3,5:ncol(mtcars))){
  x <- cbind(x, rep(mean(mtcars[,i]),nrow(mtcars)))
}
x <- data.frame(x[,1:3], mtcars[,4], x[,4:10])


# Plot the dots and create the regression lines
par(mfrow = c(1,1))
plot(mtcars[,4], mtcars[,1], xlab = "HP", ylab = "MPG")
lines(mtcars[,4], as.matrix(x)%*%coef_lasso, col = "blue")
lines(mtcars[,4], as.matrix(x)%*%coef_blasso, col = "red")
legend("topright", c("lasso","bayes_lasso"),lty=c(1,1),
       lwd=c(2.5,2.5), col=c("blue", "red"))
```



The two lines are basically the same, but if we have to give an answer of which is better I think Lasso is a little better than Bayesian Lasso.