STAT 534 Statistical Data Science I: Exam I (Take-Home Portion)

Erick Calderon-Morales

Fall 2021

Due Date: October 17 (Sunday)

Instructions: There are 50 points plus 10 extra credits on this portion of the exam. You may use any reference material for your exam. However, you are not to discuss any questions about the exam with anyone other than yourself. Please contact me with any questions and I will respond as soon as possible within the same day. Be sure to write/type your answers in complete sentences and show your work. Any items including R output not commented on will receive no credit. Attach your R code as an appendix or as a separate file. (If you use R markdown, you may keep the code inserted inline.)

```r
# Packages
library(tidyverse)
# For cleaning names
library(janitor)
# Best subset selection
library(leaps)
# For tidy function
library(broom)
# For mse plot
library(ggvis)
# For UScrimes dataset
library(MASS)
# For correlations
library(GGally)
# For joining plots
library(cowplot)
# For lasso and ridge regressions
library(glmnet)
# For pls and pcr regressions
library(pls)
# For tables
library(gt)
# For xyplot
library(lattice)
```

1. (20+10 pts) In this problem, we will generate simulated data, and will then apply best subset selection to this data using the validation method. The goal is to explore that as model size increases, the training error will necessarily decrease, but the testing error may not.

(a) (5 pts) Generate a data set with p = 20 predictors and n = 1000 observations according to the model:

$$Y = \beta_0 \ + \ \beta_1 X_1 \ ... \ + \ \beta_p X_p \ + \ \epsilon$$

where $X_j \sim N(0,1)$ and $\epsilon \ \sim N(0,1)$ independently. *Randomly select your β* values *but let some elements to* *be exactly zero. Then split your data set into a training set containing 100 observations and a testing set containing 900 observations.*

```
set.seed(123)

# Generate e values with mean 0 and sd 1
epsilon <- rnorm(1000, mean = 0, sd = 1)

# Generate x values with mean 0 and sd 1
n = 1000
variables = 20

# Create empty data frame
empty_data_set <- matrix(numeric(variables * n),
                                   ncol = variables,
                                   nrow = n)

for (each_variable in seq(along = 1:variables)){

     # Get random data and append to data frame
     empty_data_set[,each_variable] <- rnorm(1000, mean = 0, sd = 1)
}

# Clean data set
x_variables <- empty_data_set
```

```
# Select randomly the betas and set some to zero
random_betas <- rnorm(variables, mean = 5, sd = 10)

# Set some to Zero
random_betas[5]  <- 0
random_betas[15] <- 0
random_betas[17] <- 0


# Generate Y using my betas and simulated data
y <-  x_variables %*% random_betas + epsilon
colnames(y) <- "y"
```

```
# Join data'
data_set_full <-
  cbind(y,as.data.frame(x_variables)) %>%
  clean_names()
```

```
# slip data into train and test

# Get index
train_index <- sample(1:nrow(data_set_full),900)

# Test set
data_set_train <- data_set_full[train_index,]
nrow(data_set_train)
```

```
[1] 900
```

```
# Train set
data_set_test <-  data_set_full[-train_index,]
nrow(data_set_test)
```

```
[1] 100
```

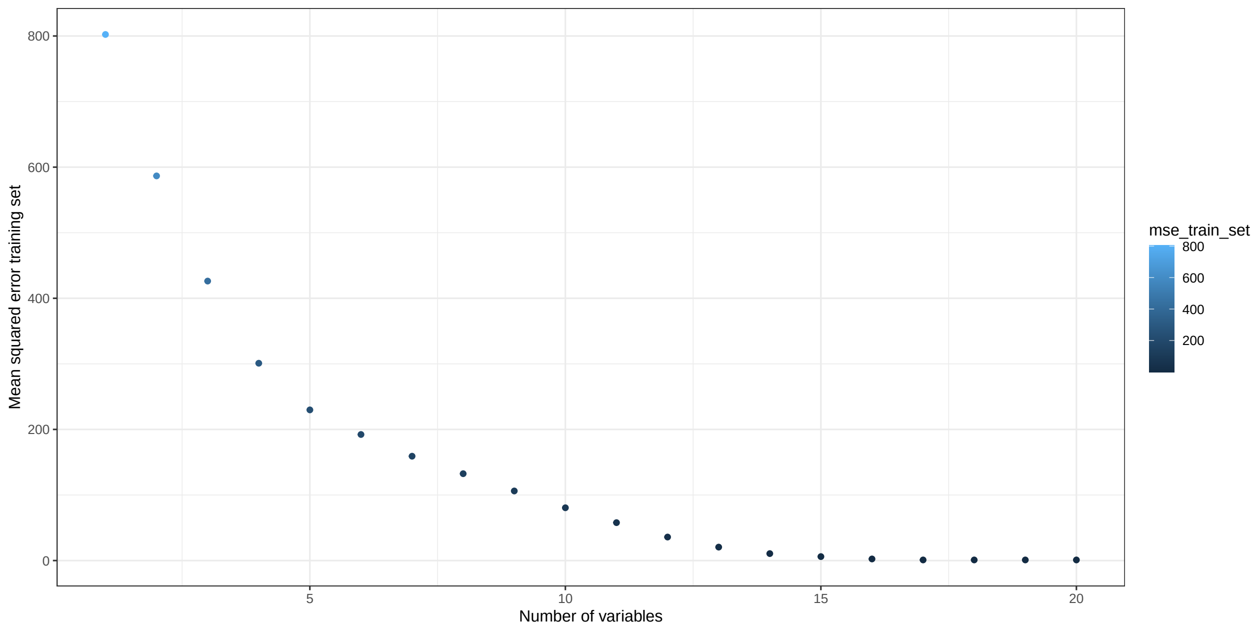
```
model_matrix_train <- model.matrix(y ~ ., data = data_set_train)
model_matrix_test  <- model.matrix(y ~ ., data = data_set_test)
```

(b) (3 pts) Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size. (Hint: regsubsets() returns error (or residual) sum of squares (rss) for each model and MSE = RSS/n.)

```
# Model on train data
best_subset_sel <- regsubsets(y ~.,
                              nvmax  = 20,
                              method = "exhaustive",
                              nbest  = 1,
                              data   = data_set_train)

best_subset_sel_summary <- summary(best_subset_sel)
```

```
# Get rss and change the colname
models_rss <- as.data.frame(best_subset_sel_summary$rss)
colnames(models_rss) <- "rss"

# Calculate MSE
mse_train_set <- models_rss$rss/nrow(data_set_train)

# Join data
models_errors <- cbind(models_rss,mse_train_set)
```

```
ggplot(data = models_errors, aes(x = c(1:20), y = mse_train_set,
                                 color = mse_train_set))+
    geom_point() +
    ylab("Mean squared error training set") + xlab("Number of variables") +
    theme_bw()
```

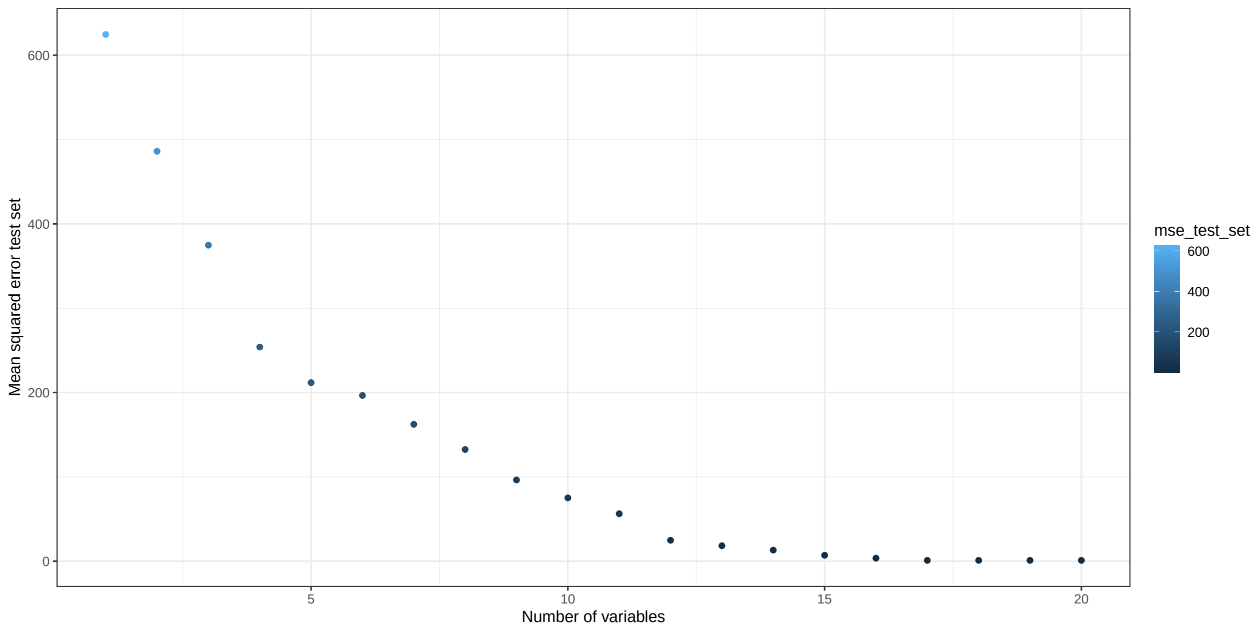


(c) (6pts) Plot the testing set MSE associated with the best model of each size. For which model size does the testing set MSE take on its minimum value? (Hint: For each model, obtain the predicted values for the testing set and then compute associated MSE.)

```
# Create empty vector
mse_test_set <- rep (NA , 20)

for (each_model in 1:20) {

    # Get the coefficients of each model build with the train set
    coefs <- coef(best_subset_sel, id = each_model)

    # get model variables and multiply then for their coef for getting the pred
    pred <- model_matrix_test[,names(coefs)] %*% coefs

    # get mse
    mse_test_set[each_model]<- mean((data_set_test$y - pred)^2)
}

# Add vector to a mse errors data frame
mse_test_set <- as.data.frame(mse_test_set)

models_errors <- cbind(models_errors,mse_test_set)
```

In this case the testing set MSE take its minimum value at 17 predictors

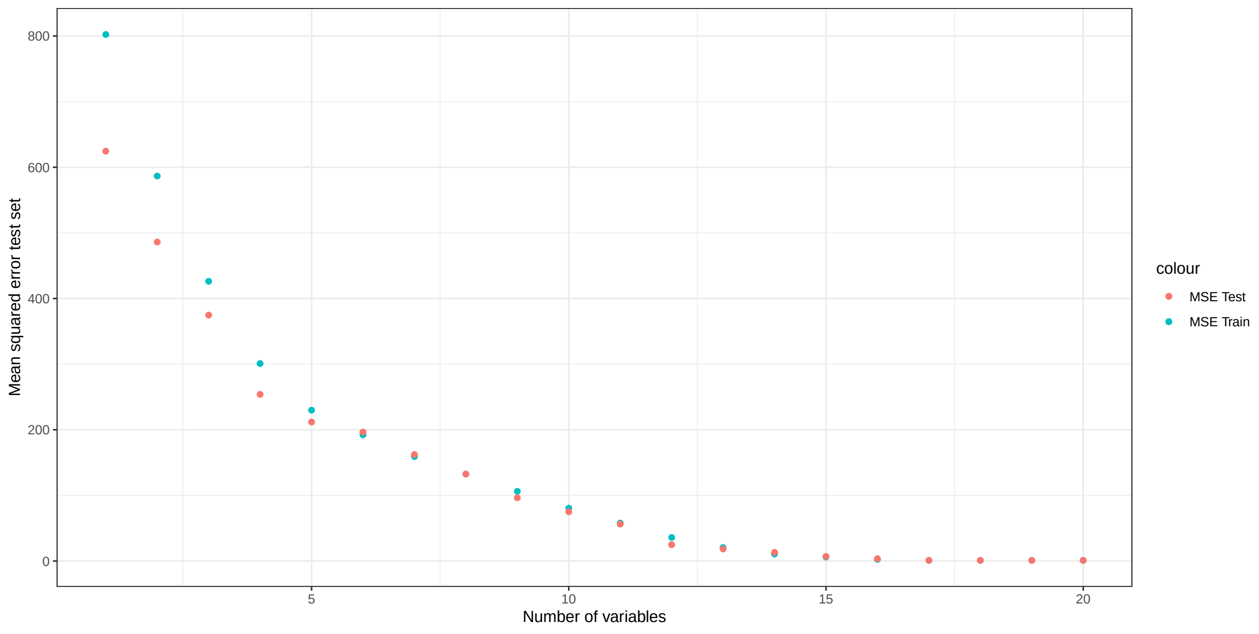
```
ggplot(data = models_errors, aes(x = c(1:20), y = mse_test_set,
                                  color = mse_test_set)) +
    geom_point() +
    ylab("Mean squared error test set") + xlab("Number of variables") +
    theme_bw()
```



(d) (3 pts) What do you observe about the changes in training MSE and testing MSE as model size increases?

In this particular case, as the model size increase, the difference between MSE train and MSE test gets lower and lower until are almost the same.

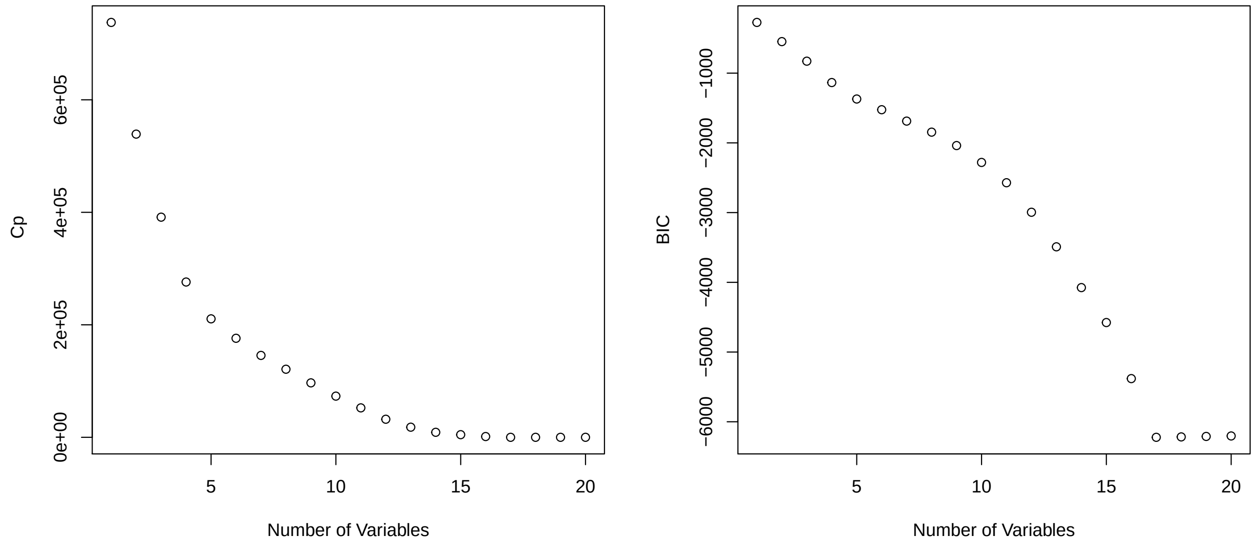
```
ggplot(data = models_errors) +
    geom_point(aes(x = c(1:20), y = mse_train_set, color = "MSE Train")) +
    geom_point(aes(x = c(1:20), y = mse_test_set, color =  "MSE Test")) +
    ylab("Mean squared error test set") + xlab("Number of variables") +
    theme_bw()
```



(e) (3 pts) How does the model at which the testing MSE is minimized compare to the true model used to generate the data? (Hint: You want to refit the regression model to the entire data set using the selected predictors.)

- Select the best model

```r
par(mfrow = c(1,2))
plot(best_subset_sel_summary$cp, xlab =" Number of Variables ", ylab =" Cp",type="p")
plot(best_subset_sel_summary$bic,xlab =" Number of Variables ", ylab =" BIC ",type="p")
```



When I compare the coefficients predicted by the best subset model and the one used for generate the data, I observe almost no difference between each other. The best subset model is the one with 17 variables, meaning that in order to explaining the data the other three coefficients are unnecessary. This coincides with the true model since those other coefficients are the ones that equals zero.

```r
coef(best_subset_sel, id = 17)
```

```
  (Intercept)            v1            v2            v3            v4 
-0.0008683453 -6.1280640597 -5.1718577540 17.5723321548 -5.0126861058 
           v6            v7            v8            v9           v10 
-1.2645632969  2.2240361175  3.1823324517  1.9377047542  5.5319291284 
          v11           v12           v13           v14           v16 
-3.8367443366  8.2467941325  5.1564036340  5.7607061500 11.2468914059 
          v18           v19           v20 
 5.0040284766 13.1144894587 13.6292706082 
```

```r
random_betas
```

```
 [1] -6.199920 -5.158191 17.580527 -5.012317  0.000000 -1.251544  2.255450
 [8]  3.161513  1.910718  5.549457 -3.768211  8.238624  5.132494  5.738987
[15]  0.000000 11.302823  0.000000  5.030707 13.045700 13.602677
```

Meanwhile, when I compared the MSE of the true model with the MSE of the best subset I observed small difference, which indicates that the model with 17 variables is able to predict the y values generated by the true model. The value of MSE for the true model is virtually zero while the value of the MSE for the best subset is 0.95. This difference is mainly due to the error term epsilon included in the true model for generating the data.

```r
# Refit the regression model to the entire data using the selected predictors

# True model
model_true <- lm(y ~ . + epsilon, data = data_set_full)

# Get MSE from original model
```

```r
model_true_pred <- predict(model_true)

# Model chosen
model_selected_17 <- lm(y ~ v1  + v2  + v3  + v4  + v6  + v7  +
                          v8  + v9  + v10 + v11 + v12 + v13 +
                          v14 + v16 + v18 + v19 + v20,
                     data = data_set_full)

# Get MSE from model chosen
model_selected_pred <- predict(model_selected_17)
```

```r
(mse_model_true <- mean((y - model_true_pred)^2))
```

```
[1] 4.410781e-27
```

```r
(mse_model_selected <- mean((y - model_selected_pred)^2))
```

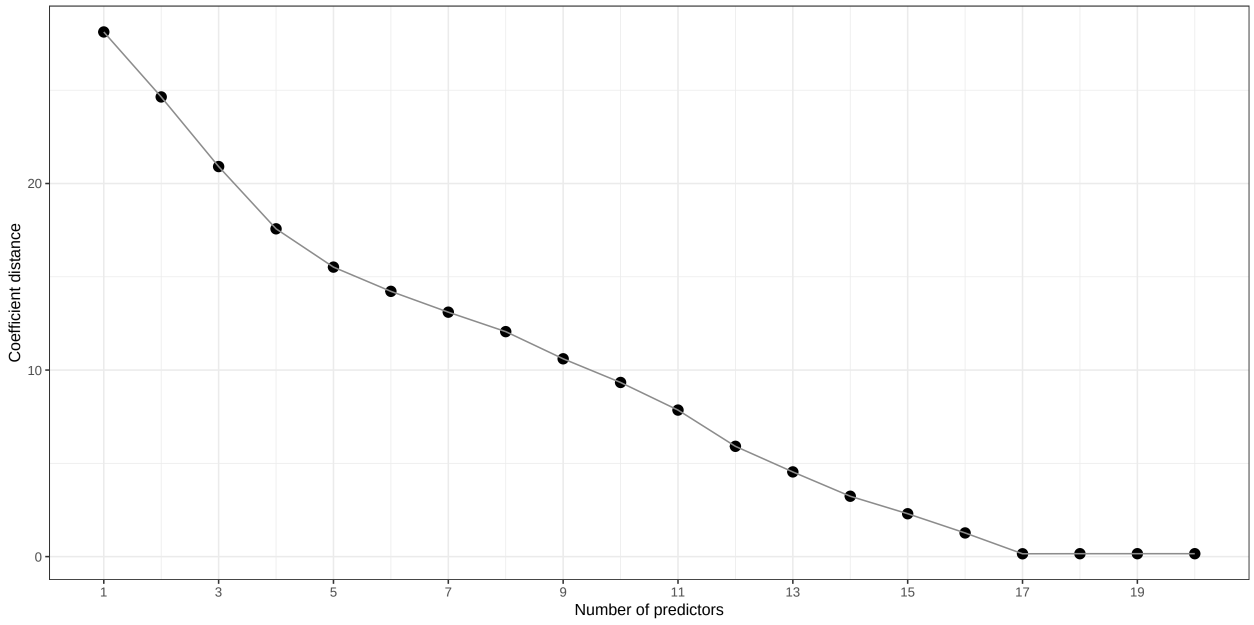
```
[1] 0.9553708
```

(f) (+10 pts) Create a plot displaying $\sqrt{\sum_{j=0}^{p}(\beta_j - \hat{\beta}_j^s}^2$ where $\hat{\beta}_j^s$ is the jth coefficient estimate for the best model of size s using the entire data set. Comment on what you observe. How does this compare to the testing MSE plot from part (c)?_

As the number of predictors increases the distance between the coefficients decreases. This happens because as the number of predictors increases the predictive capacity of the model increases too.

Also, as the number of predictors increases the bias in the model decrease since the true relationship between the y-variable is best estimated when the number of coefficients increase.

```r
param_dist <- c()

for (each_model in 1:20) {
  # Get the original betas
  params <- data.frame(parameter = colnames(best_subset_sel_summary$which),
                           actual = c(0, random_betas)) %>%

  # Get estimated betas from each model
  # Get names
  left_join(data.frame(parameter = names(coef(best_subset_sel,
                                          id = each_model)),
                       # Get values
                       estimated = coef(best_subset_sel, id = each_model)),
            by = "parameter") %>%

  # Set to 0 the estimated betas that were not estimated
  mutate(estimated = case_when(is.na(estimated) ~ 0, TRUE ~ estimated)) %>%
  filter(!parameter == "(Intercept)")

  # Calculate the parameter distance for each single model
  param_dist[each_model] <- sqrt(sum((params$actual - params$estimated)^2))
}

# Plot
as.data.frame(param_dist) %>%
    mutate(number = c(1:20)) %>%
      ggplot(.,aes(x = number, y = param_dist)) +
      geom_point(size = 3) +
      scale_x_continuous(breaks = seq(1, 20, 2)) +
      scale_y_continuous(labels = scales::comma_format()) +
      geom_line(col = "grey55") +
      theme_bw() +
      ylab("Coefficient distance") + xlab("Number of predictors")
```



2. (30 pts) Use the UScrime data set in the MASS library to study the effect of punishment regimes on crime rates.

```
set.seed(123)

# Load data
data("UScrime")

uscrime <-
  UScrime %>%
    clean_names()

summary(uscrime)
```

```
       m               so               ed              po1
 Min.   :119.0   Min.   :0.0000   Min.   : 87.0   Min.   : 45.0
 1st Qu.:130.0   1st Qu.:0.0000   1st Qu.: 97.5   1st Qu.: 62.5
 Median :136.0   Median :0.0000   Median :108.0   Median : 78.0
 Mean   :138.6   Mean   :0.3404   Mean   :105.6   Mean   : 85.0
 3rd Qu.:146.0   3rd Qu.:1.0000   3rd Qu.:114.5   3rd Qu.:104.5
 Max.   :177.0   Max.   :1.0000   Max.   :122.0   Max.   :166.0
      po2              lf             m_f              pop
 Min.   : 41.00   Min.   :480.0   Min.   : 934.0   Min.   :  3.00
 1st Qu.: 58.50   1st Qu.:530.5   1st Qu.: 964.5   1st Qu.: 10.00
 Median : 73.00   Median :560.0   Median : 977.0   Median : 25.00
 Mean   : 80.23   Mean   :561.2   Mean   : 983.0   Mean   : 36.62  
 3rd Qu.: 97.00   3rd Qu.:593.0   3rd Qu.: 992.0   3rd Qu.: 41.50  
 Max.   :157.00   Max.   :641.0   Max.   :1071.0   Max.   :168.00  
       nw              u1               u2             gdp       
 Min.   :  2.0   Min.   : 70.00   Min.   :20.00   Min.   :288.0  
 1st Qu.: 24.0   1st Qu.: 80.50   1st Qu.:27.50   1st Qu.:459.5  
 Median : 76.0   Median : 92.00   Median :34.00   Median :537.0  
 Mean   :101.1   Mean   : 95.47   Mean   :33.98   Mean   :525.4  
 3rd Qu.:132.5   3rd Qu.:104.00   3rd Qu.:38.50   3rd Qu.:591.5  
 Max.   :423.0   Max.   :142.00   Max.   :58.00   Max.   :689.0  
      ineq            prob              time             y         
 Min.   :126.0   Min.   :0.00690   Min.   :12.20   Min.   : 342.0  
 1st Qu.:165.5   1st Qu.:0.03270   1st Qu.:21.60   1st Qu.: 658.5  
 Median :176.0   Median :0.04210   Median :25.80   Median : 831.0  
 Mean   :194.0   Mean   :0.04709   Mean   :26.60   Mean   : 905.1  
 3rd Qu.:227.5   3rd Qu.:0.05445   3rd Qu.:30.45   3rd Qu.:1057.5  
 Max.   :276.0   Max.   :0.11980   Max.   :44.00   Max.   :1993.0  
```

(a) (5 pts) Explore the variables using appropriate graphics and summary statistics. Comment on your observations.

From the plot above I observed high correlation between variables. Especially the percentage of males aged 14–24 (m), the indicator variable for a Southern state (so), gross domestic product per capita (gdp) and income inequality (ineq) are highly correlated with other variables.

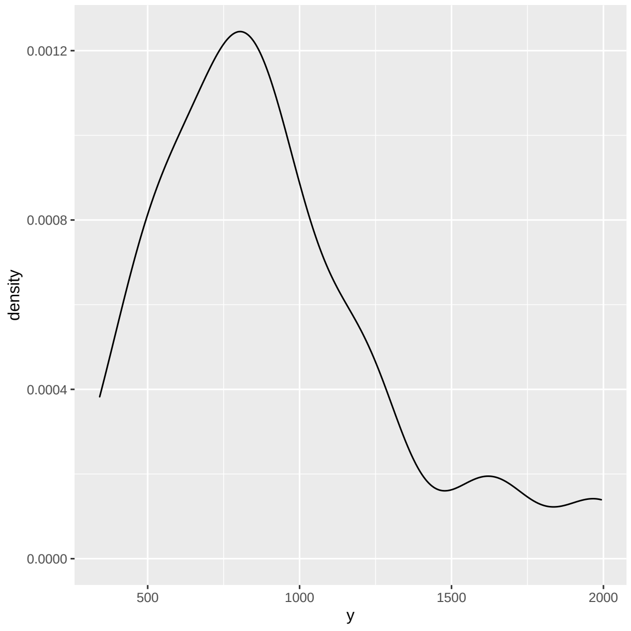
From the variables above its worth to mention that the percentage of males aged 14-24 (m) is positively correlated with the variables Southern States (so),number of non-whites per 1000 people (nw) and income inequality. While is negatively correlated with the variables mean years of schooling (Ed), police expenditure in 1960 (Po1) police expenditure in 1959 (Po2) and gdp. While the variable income inequality is negatively correlated with gdp, Po1, Po2 and ed and positively correlated with nw, so and m.

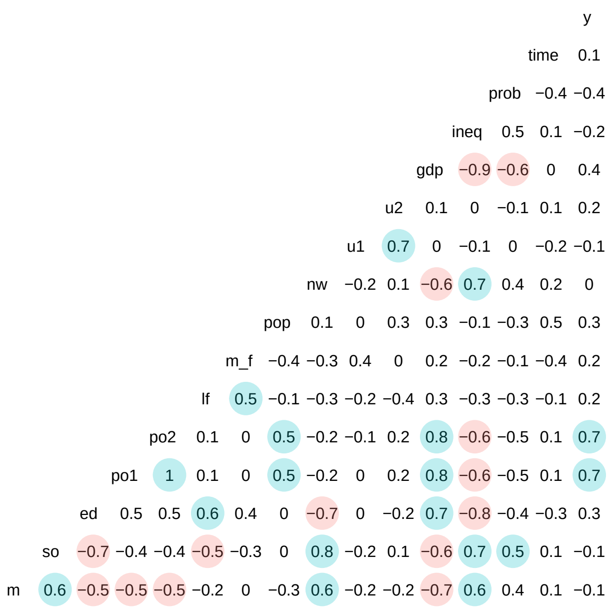
Finally I noticed that the y-variable is skew so in order to correct this I log transform this variable.

```r
# Visualization of correlations
correlations <-
  uscrime %>%
    ggcorr(geom = "blank", label = TRUE, hjust = 0.75) +
    geom_point(size = 10, aes(color = coefficient > 0,
                              alpha = abs(coefficient) > 0.5)) +
    scale_alpha_manual(values = c("TRUE" = 0.25, "FALSE" = 0)) +
    guides(color = FALSE, alpha = FALSE)
```

```r
y_density <- ggplot(uscrime, aes(x = y)) + geom_density()
```

```r
cowplot::plot_grid(y_density,correlations)
```





```
uscrime <-
  uscrime %>%
    mutate(so = factor(so),
           # log transform y variable
           log_y = log(y)) %>%
    dplyr::select(-y)
```

(b) (12 pts) Split the data into 75% training and 25% testing and build the following models using the training set:

```
# slip data into train and test

# Get index
train_index <- sample(1:nrow(uscrime),(nrow(uscrime)*75)/100)

# Train set
uscrime_train <- uscrime[train_index,]
(nrow(uscrime_train)*100)/nrow(uscrime)
```

```
[1] 74.46809
```

```
# This is done for fittin ridge and lasso
x_vars_train <- model.matrix(log_y ~., data = uscrime_train)[,-1]
y_var_train <- uscrime_train$log_y

# Test set
uscrime_test <-  uscrime[-train_index,]
(nrow(uscrime_test)*100)/nrow(uscrime)
```

```
[1] 25.53191
```

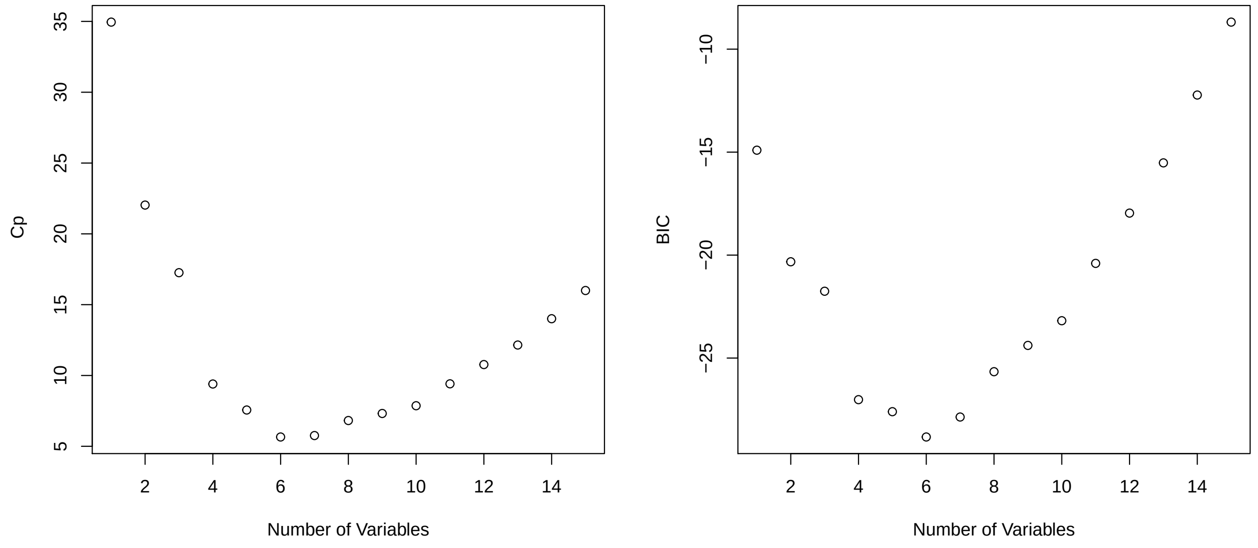
```
y_var_test <- uscrime_test$log_y


# This is done for fitting ridge and lasso
x_vars_test <- model.matrix(log_y ~., data = uscrime_test)[,-1]
```

- Multiple linear regression

The best subset model is the one with 6 variables

```
mult_reg_train <- regsubsets(log_y ~., nvmax = 16,
                             nbest = 1,
                             method="exhaustive",
                             data = uscrime_train)

mult_reg_train_summary <- summary(mult_reg_train)
```

```
# Select best multiple regression model
par(mfrow = c(1,2))
plot(mult_reg_train_summary$cp , xlab =" Number of Variables ", ylab =" Cp",type="p")
plot(mult_reg_train_summary$bic, xlab =" Number of Variables ", ylab =" BIC ",type="p")
```



```
coef(mult_reg_train, id = 6)
```

```
 (Intercept)            m          so1           ed          po1         ineq 
-0.035388076  0.011000106  0.262084275  0.025510474  0.011454884  0.005612907 
        time 
 0.013785523 
```

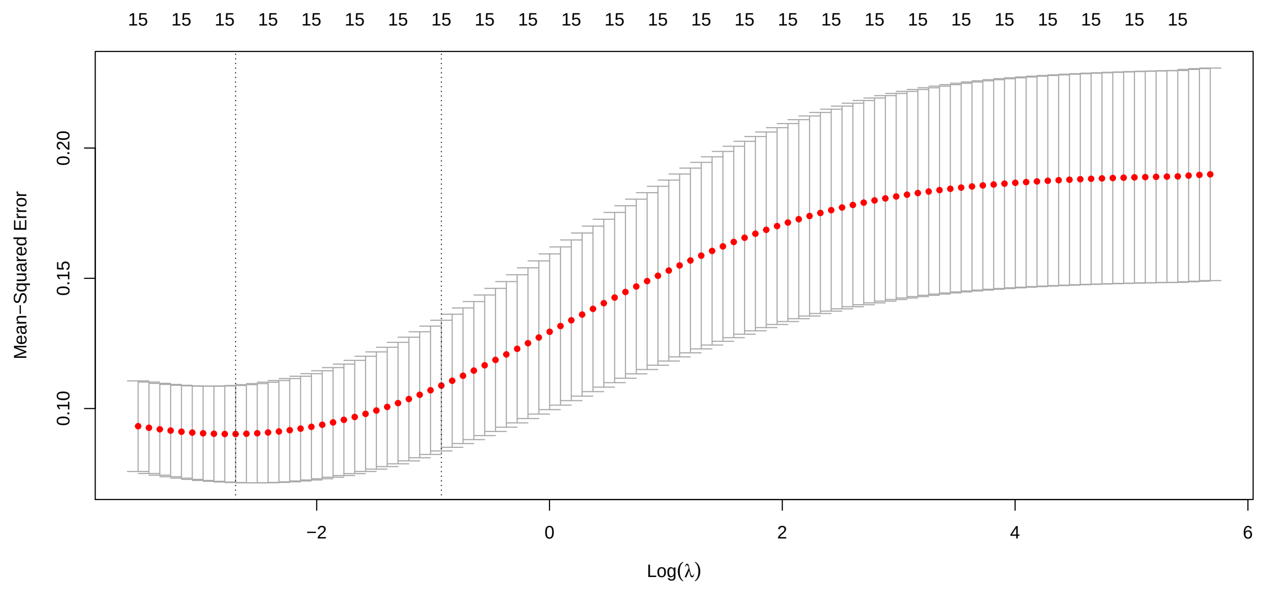
```
# Refit the best subset model
best_mult_reg_train <- lm(log_y ~ m + so + ed + po1 + ineq + time,
                          data = uscrime_train)
```

- Ridge regression

The best Ridge model is the one with 15 variables

```
ridge_uscrimes_train <- cv.glmnet(x_vars_train, y_var_train, alpha = 0)
```

```
plot(ridge_uscrimes_train)
```

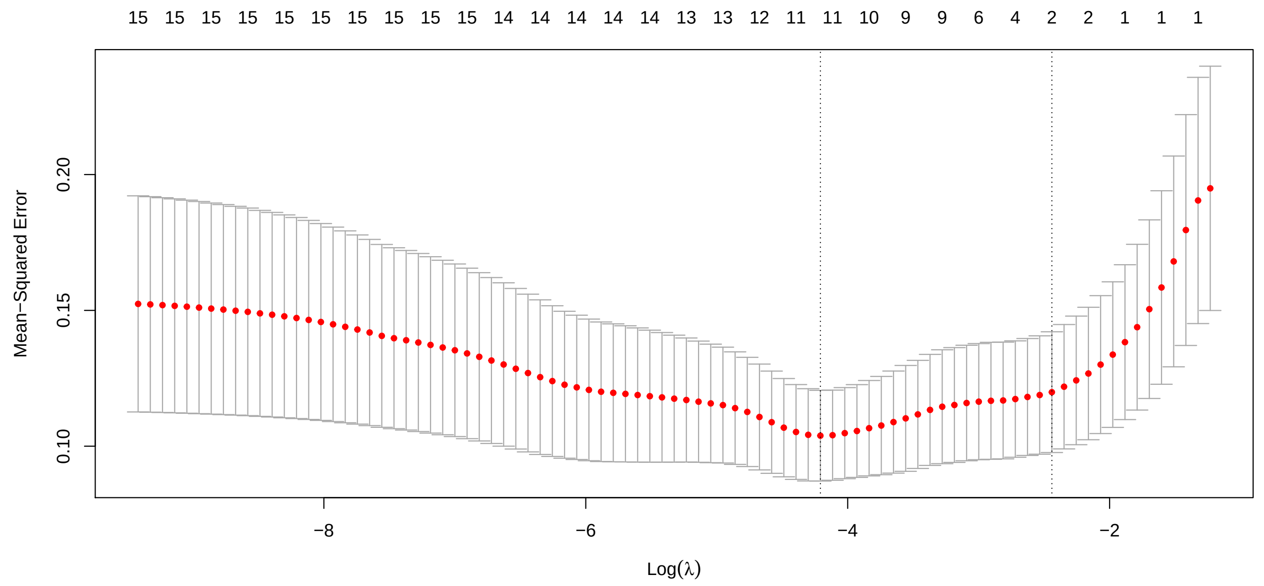


- Lasso regression

The best LASSO model is the one with 2 variables

```r
lasso_uscrimes_train <- cv.glmnet(x_vars_train, y_var_train, alpha = 1)
```

```r
plot(lasso_uscrimes_train)
```

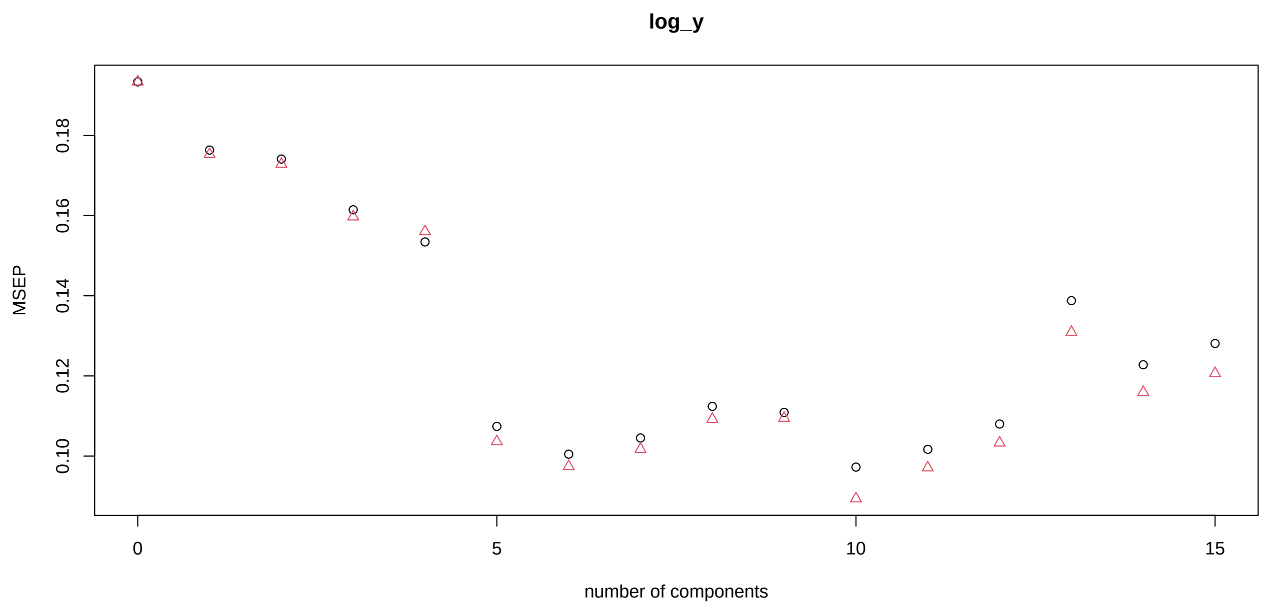


- Principal components regression (justify how many principal components should be used)

I chose 5 components for the principal component regression since the value of MSEP start to stabilize after 5 components

```r
pcr_uscrimes_train <- pcr(log_y ~ ., data = uscrime_train,
                          scale = TRUE ,
                          validation = "CV")
```

```r
validationplot(pcr_uscrimes_train, val.type = "MSEP", type = "p")
```

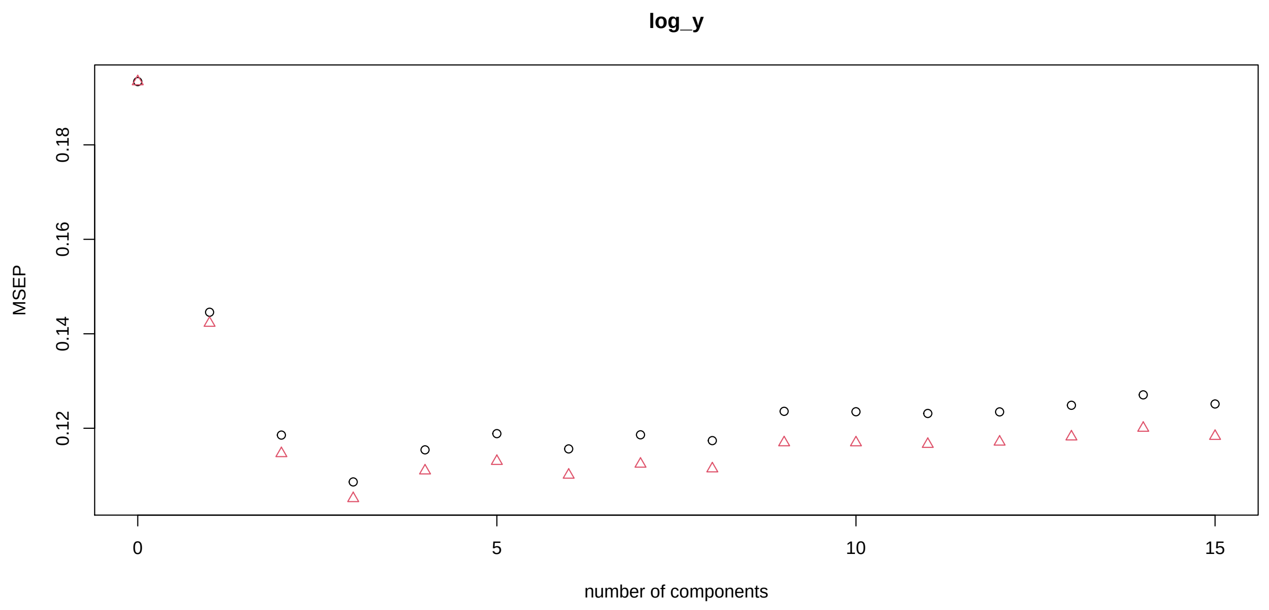


- Partial least squares (justify how many directions should be used)

I chose 3 components for the partial least squared regression since it has the lowest MSEP value.

```
pls_uscrimes_train <- plsr(log_y ~ ., data = uscrime_train,
                           scale = TRUE ,
                           validation = "CV")
```

```
validationplot(pls_uscrimes_train, val.type = "MSEP", type = "p")
```



(c) (5 pts) Compare the effectiveness of each model on training vs. testing data.

- Multiple linear regression MSE

```
# Training error
mult_mse_train <- summary(best_mult_reg_train)$sigma^2
```

```r
# Testing error
mult_pred <- predict(best_mult_reg_train, newx = uscrime_test)

mult_mse_test <- mean((mult_pred - y_var_test)^2)
```

- Ridge regression MSE

```r
# MSE train
ridge_pred_train <- predict(ridge_uscrimes_train,
                            s = ridge_uscrimes_train$lambda.1se,
                            newx = x_vars_train)

# Training error
ridge_mse_train <- mean((ridge_pred_train - y_var_train)^2)

# MSE test
ridge_pred_test <- predict(ridge_uscrimes_train,
                            s = ridge_uscrimes_train$lambda.1se,
                            newx = x_vars_test)
# Test error
ridge_mse_test <- mean((ridge_pred_test - y_var_test)^2)
```

- Lasso regression MSE

```r
# MSE train
lasso_pred_train <- predict(lasso_uscrimes_train,
                            s = lasso_uscrimes_train$lambda.1se,
                            newx = x_vars_train)

# Training error
lasso_mse_train <- mean((lasso_pred_train - y_var_train)^2)

# MSE test
lasso_pred_test <- predict(lasso_uscrimes_train,
                            s = lasso_uscrimes_train$lambda.1se,
                            newx = x_vars_test)
# Test error
lasso_mse_test <- mean((lasso_pred_test - y_var_test)^2)
```

- PCR MSE

```r
# Train error
pcr_pred_train <- predict(pcr_uscrimes_train, data = uscrime_train, ncomp = 5)
pcr_mse_train <- mean((pcr_pred_train - y_var_train)^2)

# Test error
pcr_pred_test <- predict(pcr_uscrimes_train, uscrime_test, ncomp = 5)
pcr_mse_test <- mean((pcr_pred_test - y_var_test)^2)
```

- PLSR MSE

```r
# Train error
pls_pred_train <- predict(pls_uscrimes_train, data = uscrime_train, ncomp = 3)
pls_mse_train <- mean((pls_pred_train - y_var_train)^2)

# Test error
```

```
pls_pred_test <- predict(pls_uscrimes_train, uscrime_test, ncomp = 3)
pls_mse_test <- mean((pls_pred_test - y_var_test)^2)
```

Table 1: Models MSE

Based on which model has the lowest mse test value and the lowest difference between mse test and mse train I decided to choose the Ridge model and the Principal component regression model with 5 components.

```
tribble(
~model, ~mse_train, ~mse_test,~difference,
"MLR",   mult_mse_train,  mult_mse_test,  abs(mult_mse_test  - mult_mse_train),
"RIDGE", ridge_mse_train, ridge_mse_test, abs(ridge_mse_test - ridge_mse_train),
"LASSO", lasso_mse_train, lasso_mse_test, abs(lasso_mse_test - lasso_mse_train),
"PCR",   pcr_mse_train,   pcr_mse_test,   abs(pcr_mse_test   - pcr_mse_train),
"PLS",   pls_mse_train,   pls_mse_test,   abs(pls_mse_test   - pls_mse_train)
) %>% gt()
```

model	mse_train	mse_test	difference
MLR	0.04915594	0.29402745	0.244871515
RIDGE	0.06709524	0.08756203	0.020466792
LASSO	0.09804007	0.09443654	0.003603525
PCR	0.06409633	0.08486221	0.020765882
PLS	0.04719103	0.09040896	0.043217926

(d) (8 pts) Select the best two models from above.

- Refit Ridge model to the entire data set

```
x_variables_uscrime <- model.matrix(log_y ~., data = uscrime)[,-1]
y_variable_uscrime <- uscrime$log_y

ridge_uscrimes <- cv.glmnet(x_variables_uscrime, y_variable_uscrime, alpha = 0)
```

- Refit PCR model to the entire data set

```
pcr_uscrimes <- pcr(log_y ~ ., ncomp = 5,
                        scale = TRUE ,
                        data = uscrime,
                        validation = "CV")
summary(pcr_uscrimes)
```

```
Data:   X dimension: 47 15
    Y dimension: 47 1
Fit method: svdpc
Number of components considered: 5

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
CV          0.4155   0.3917   0.3799   0.3879   0.4001   0.3070
adjCV       0.4155   0.3910   0.3784   0.3863   0.4056   0.3035

TRAINING: % variance explained
       1 comps  2 comps  3 comps  4 comps  5 comps
```

```
X        40.13    58.81    72.17    79.92    86.31
log_y    13.99    24.15    25.47    29.83    60.12
```

```
pcr_uscrimes[["coefficients"]][,,1]
```

```
           m          so1           ed          po1          po2           lf 
-0.019031445 -0.020733953  0.021281638  0.019339882  0.019487688  0.011039783 
         m_f          pop           nw           u1           u2          gdp 
 0.007292837  0.007085809 -0.018396955  0.002537930  0.001135593  0.023793278 
        ineq         prob         time 
-0.022921919 -0.016222563 -0.001292651 
```

(d.1) Interpret and compare their respective final fitted models

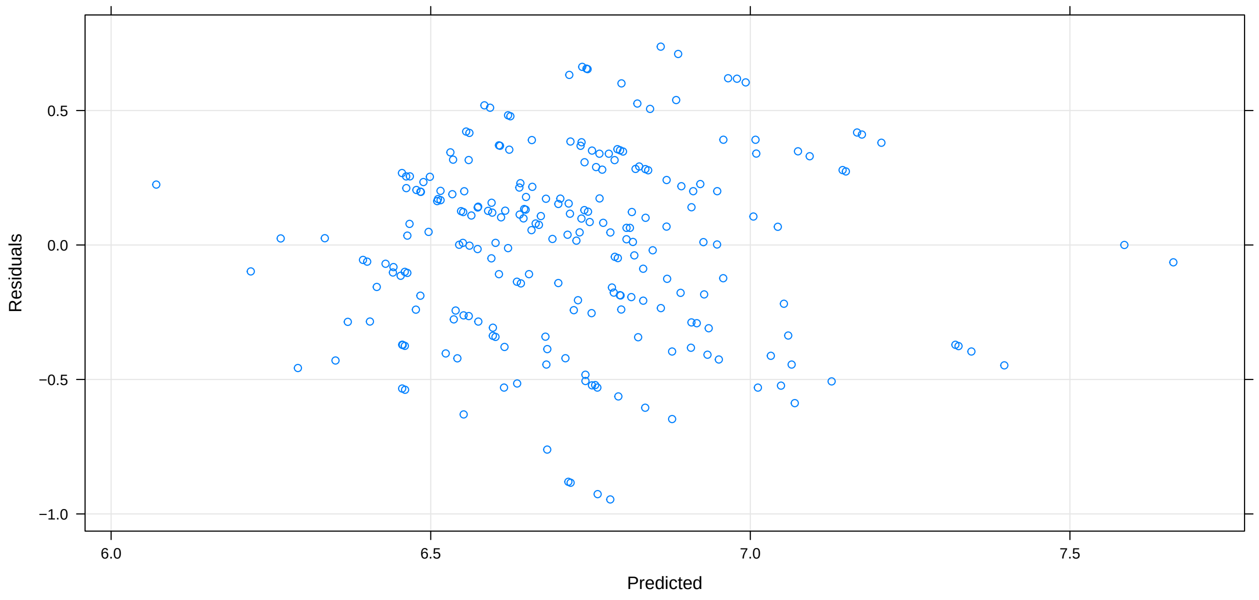
- PCR

Overall, the 5 principal components explains 86% of the variability in the data and 60% of the variability in the y-variable. I do not observe any violation of the model's assumptions since I do not detect any pattern in the residuals vs predicted plot.

```
summary(pcr_uscrimes)
```

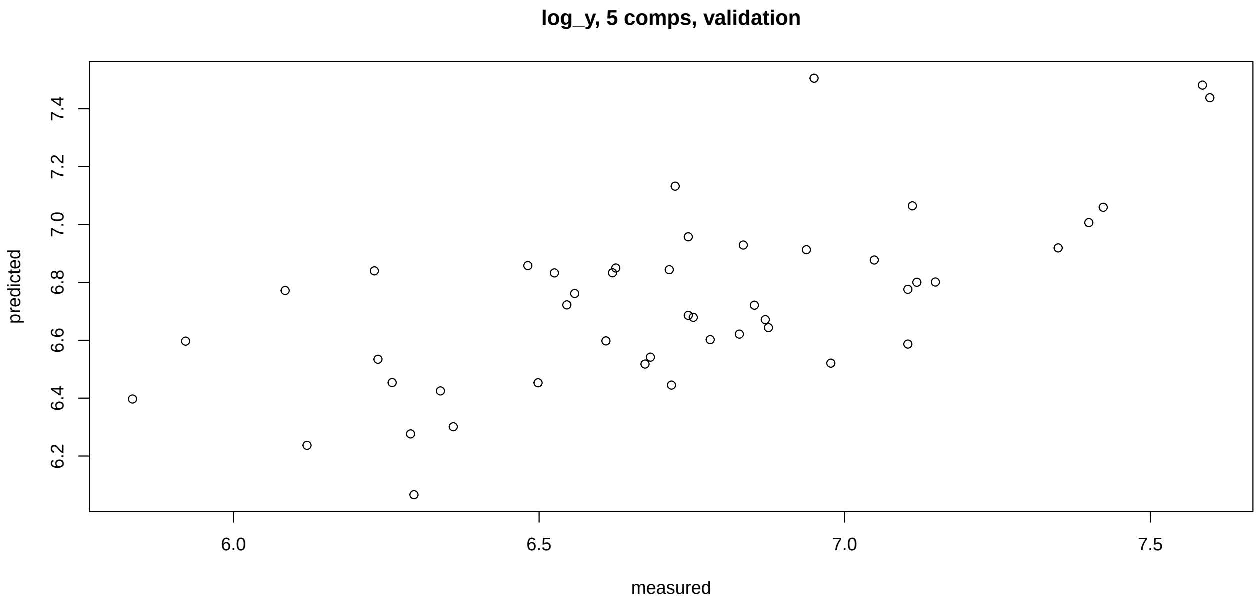
```
Data:   X dimension: 47 15 
    Y dimension: 47 1
Fit method: svdpc
Number of components considered: 5

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
CV          0.4155   0.3917   0.3799   0.3879   0.4001   0.3070
adjCV       0.4155   0.3910   0.3784   0.3863   0.4056   0.3035

TRAINING: % variance explained
       1 comps  2 comps  3 comps  4 comps  5 comps
X        40.13    58.81    72.17    79.92    86.31
log_y    13.99    24.15    25.47    29.83    60.12
```

```
xyplot(resid(pcr_uscrimes) ~ predict(pcr_uscrimes), type = c("p", "g"),
       xlab = "Predicted", ylab = "Residuals")
```



The predicted vs observed shows a linear relation which I considered is not good enough so I question the actual predictive power of the model. Also, when the y-variable is plot against each principal component, it seems that there is not relationship between the rate of crimes and any principal component.

```r
predplot(pcr_uscrimes)
```



```r
pc1 <- xyplot(y_variable_uscrime ~ pcr_uscrimes$projection[,1], type = c("p", "g"))
```

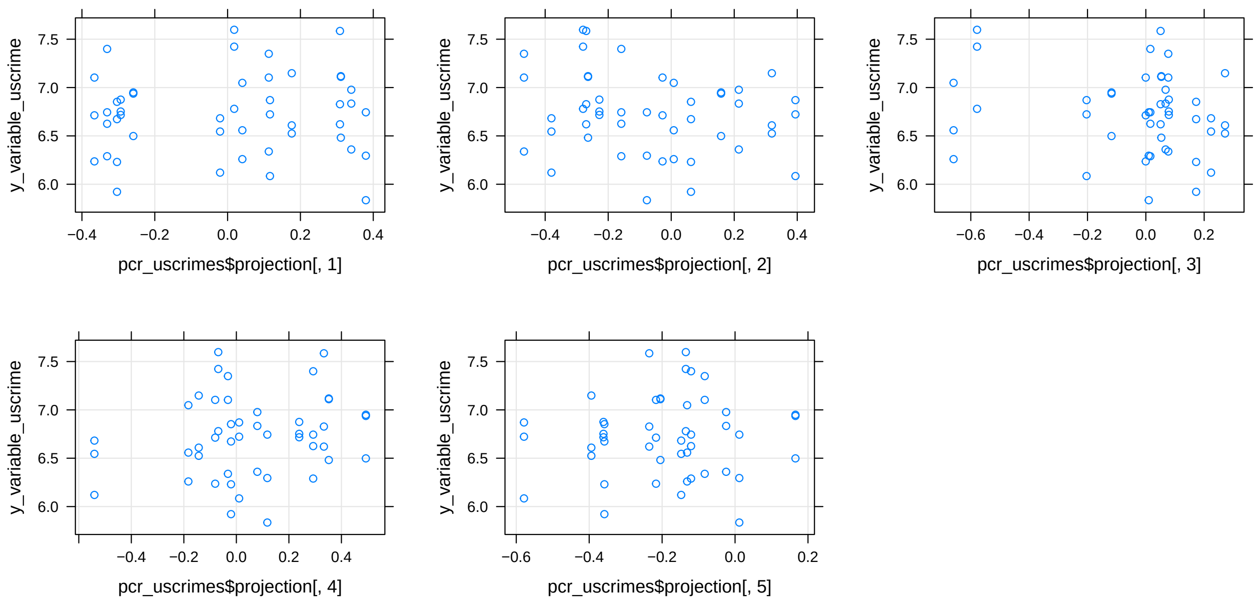
```r
pc2 <- xyplot(y_variable_uscrime ~ pcr_uscrimes$projection[,2], type = c("p", "g"))
```

```r
pc3 <-xyplot(y_variable_uscrime ~ pcr_uscrimes$projection[,3], type = c("p", "g"))
```

```r
pc4 <-xyplot(y_variable_uscrime ~ pcr_uscrimes$projection[,4], type = c("p", "g"))
```

```r
pc5 <- xyplot(y_variable_uscrime ~ pcr_uscrimes$projection[,5], type = c("p", "g"))
```

```
plot_grid(pc1,pc2,pc3,pc4,pc5)
```



Given the social nature of the data set and the importance of drawing conclusions for understanding the effect of punishment regimes on crime rates in United States of America, I consider that the PCR model is inadequate for understanding this problem. The main drawbacks of this model reside in the difficulty for drawing any conclusions. As showed above, it seems that none of the principal components relate to the y-variable. Also I consider this model inadequate for this data since the number of predictors (15) is not larger than the number of observation(47).

- Ridge regression

Even though the PCR has a lower MSE error (0.065) I consider the Ridge regression a better solution to understanding the effect of punishment regimes on crime rates compared to PCR. This because in this data set is more important to infer which predictors are related with crime rates than predict future values.

```
ridge_pred_full <- predict(ridge_uscrimes,
                           s = ridge_uscrimes$lambda.1se,
                           newx = x_variables_uscrime)

pcr_pred_full <- predict(pcr_uscrimes, data = uscrime, ncomp = 5)


# MSE
(pcr_mse_full <- mean((pcr_pred_full - y_variable_uscrime)^2))
```

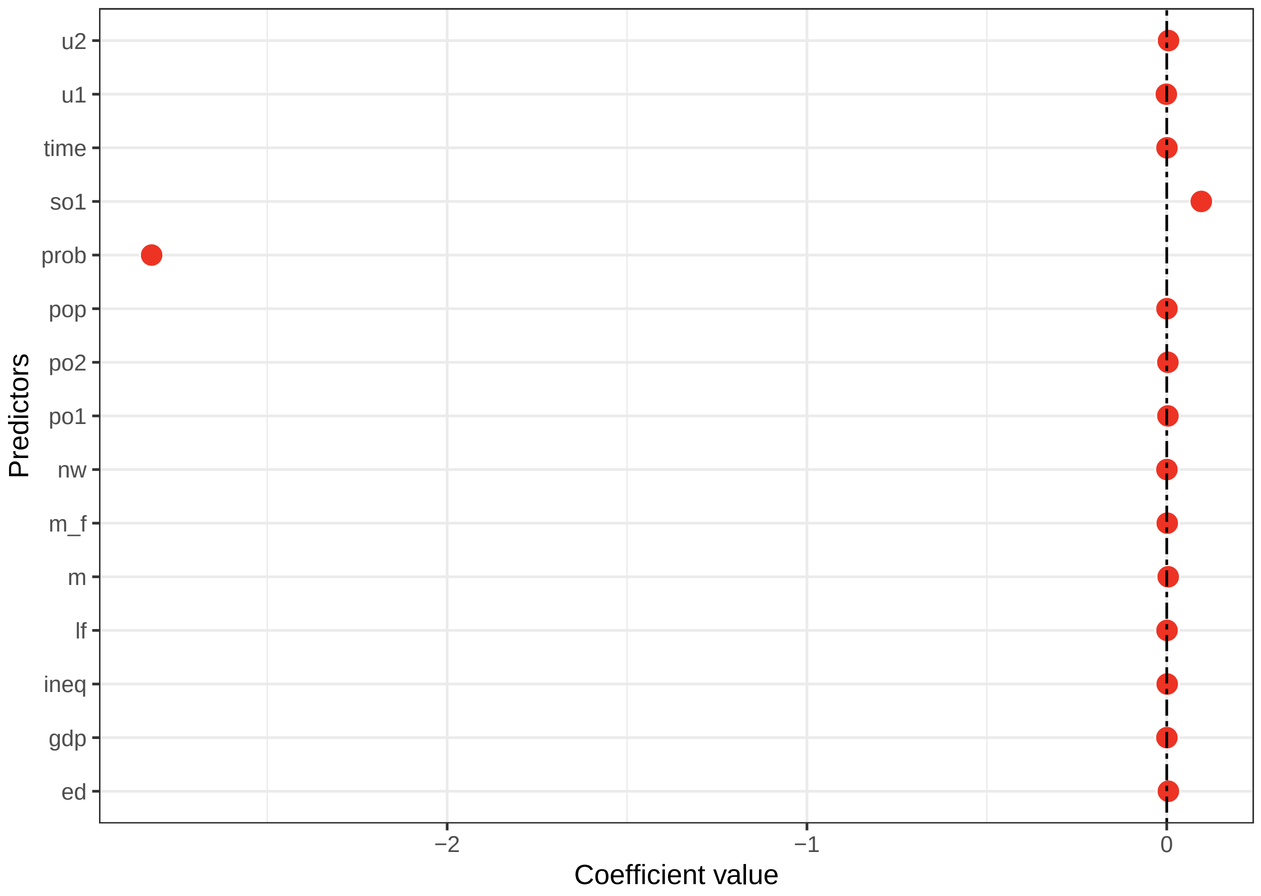
```
[1] 0.06595715
```

```
(ridge_mse_full <- mean((ridge_pred_full - y_variable_uscrime)^2))
```

```
[1] 0.06369761
```

The main problem of this data set is that some predictors showed high correlation between each other. This problem is solved in the Ridge regression by adding a penalty term that shrinks towards zero some coefficients.

```
coefs_ridge <- as.data.frame(coef(ridge_uscrimes)[-1,])
colnames(coefs_ridge) <- "coef_value"

coefs_ridge %>%
  rownames_to_column(var = "coefficient") %>%
    ggplot(data = ., aes(x = factor(coefficient), y = coef_value)) +
    geom_point(fill = "#ed3324", color = "white", size = 4, shape = 21) +
    geom_hline(aes(yintercept = 0),linetype = 6) +
    theme_bw() +
    coord_flip() +
    ylab("Coefficient value") + xlab("Predictors")
```



From all the variables in the model it seems that only the probability of imprisonment(prob) and indicator variable for a Southern state(so) are important to explain the crime rates.

It seems that a lower probability of imprisonment leads to higher rates of crime. Specifically for each unit increase in the probability of imprisonment the crime rate decreases by 90%, while being or not in a Southern state increase by 7% the crime rates.

```
# Percent increase or decrease in the response for every one-unit increase in the independent variable.
# For every one-unit increase in the independent variable
(exp(-2.3772447459)-1)*100
```

```
[1] -90.71941
```

```
# Percent increase or decrease in the response for every one-unit increase in the independent variable.
# For every one-unit increase in the independent variable
(exp(0.0765351706)-1)*100
```

```
[1] 7.954016
```