Project 1 - Modeling for Lag of 1, Horizon of 1

Group: Medha Dhir, Abdul R Latif, Tamara Hahn , Lisheng Zou

```r
library(readr)
library(faraway)
library(dplyr)
library(glmnet)
library(tidyr)

bit_data = read_csv('bitcoin.csv')
price_0 = bit_data[bit_data[2]==0,]
bit_data = bit_data[bit_data[2]!=0,] # 2745 non zeros for the response variable
na_rows = which(is.na(bit_data))
clean=na.omit(bit_data)
```

Based on the pair plots in the file "Data_Exploration.Rmd", we will first exclude predictors that do not seem to have a clear or useful relationship with the response when running Ridge Regression. However, first, for reference, we will make a "full model" (sans Date) without penalization and a model using Lasso Regression that leaves the predictor exclusion up to the Lasso Regression.

We add in the previous day as a predictor for the present day. This means a lag of 1, and a horizon of 1.

```r
# Use previous day price to predict next day. First day gets as dummy value its own price
clean$prev_price = 0
clean$prev_price[1] = clean$btc_market_price[1]

for(row_ind in 2:nrow(clean)){
  clean$prev_price[row_ind] = clean$btc_market_price[row_ind-1]
}
```

We sort by Date and split into train and test data. The test data for time series data needs to be sequential and from the chronological end of the dataset.

```r
test_data_fraction = 0.2
sorted = clean %>% arrange((Date))
test_data_size = round(nrow(sorted) * test_data_fraction)
tr_end_ind = nrow(sorted) - test_data_size
clean_train = sorted[1:tr_end_ind,]
clean_test = sorted[(tr_end_ind+1):nrow(clean),]
```

First, we build a full multiple linear regression model for reference. This is a simple model without any predictor transformations or penalty terms.

```r
full_model = lm(btc_market_price ~. -Date, data=clean_train)
```

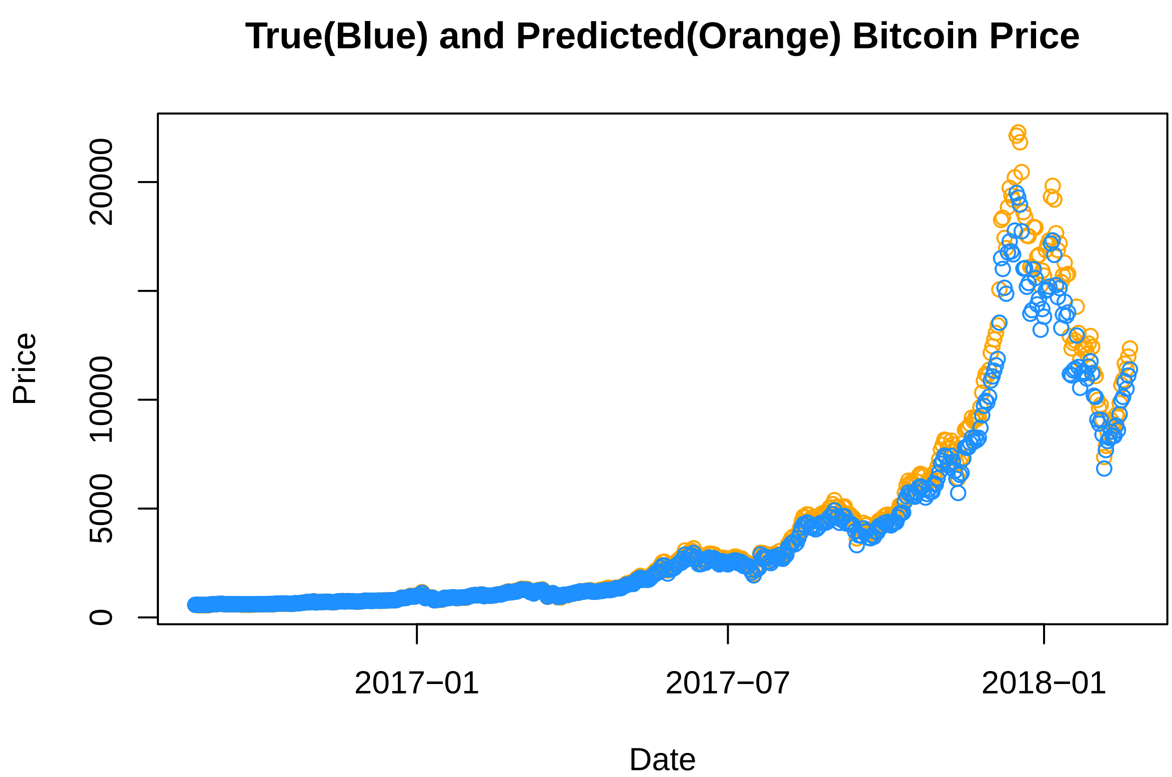
A helper function to calculate Mean Squared Error (MSE):

```r
get_mse = function(actual_y, predicted_y){
  num_obs = length(actual_y)
  result = sum((actual_y - predicted_y)^2)/num_obs
  result
}
```

```r
ytest_pred = predict(full_model, newdata = clean_test[, names(clean_test) != "btc_market_price"])

full_model_test_mse = get_mse(clean_test[2], ytest_pred)
full_model_train_mse = get_mse(clean_train[2], fitted(full_model))

full_model_test_mse
```

```
## [1] 285237284
```

```r
plot(x=clean_test$Date, y= ytest_pred, col = "orange",
     xlab = "Date",
     ylab = "Price",
     main = "True(Blue) and Predicted(Orange) Bitcoin Price")
points(x=clean_test$Date, y=clean_test$btc_market_price, col = "dodgerblue")
```



LASSO

We try Lasso regression on the cleaned training data, and do not transform any predictors, nor eliminate any predictors before hand. Lasso Regression can reduce the coefficients in front of predictors to zero, and thereby eliminate them from the model. We therefore let Lasso Regression find which predictors are best eliminated.

Data Preparation (Lasso and Ridge use the glmnet function which requires matrices).

```
# remove btc_market_price and Date from the features

rem_price_date = c("btc_market_price", "Date")
#glm_train_x = as.matrix(clean_train[, names(clean_train) != "btc_market_price"])
glm_train_x = as.matrix(clean_train[, !(names(clean_train) %in% rem_price_date)])
glm_train_y = clean_train$btc_market_price

#glm_test_x = as.matrix(clean_test[,names(clean_test) != "btc_market_price"])
glm_test_x = as.matrix(clean_test[, !(names(clean_test) %in% rem_price_date)])
glm_test_y = clean_test$btc_market_price
```

```
full_model_lasso_cv = cv.glmnet(glm_train_x, glm_train_y, alpha=1)
```

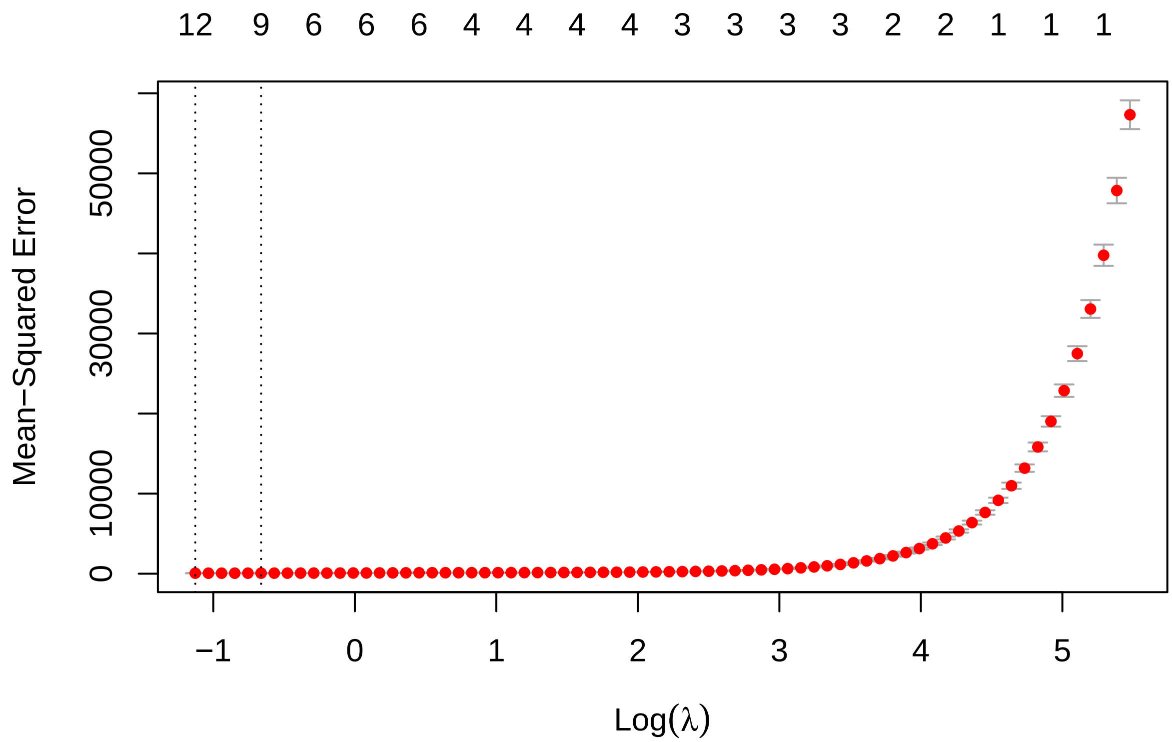
```
full_model_lasso_cv$lambda.min
```

```
## [1] 0.3237868
```

```
full_model_lasso_cv$lambda.1se
```

```
## [1] 0.5155601
```

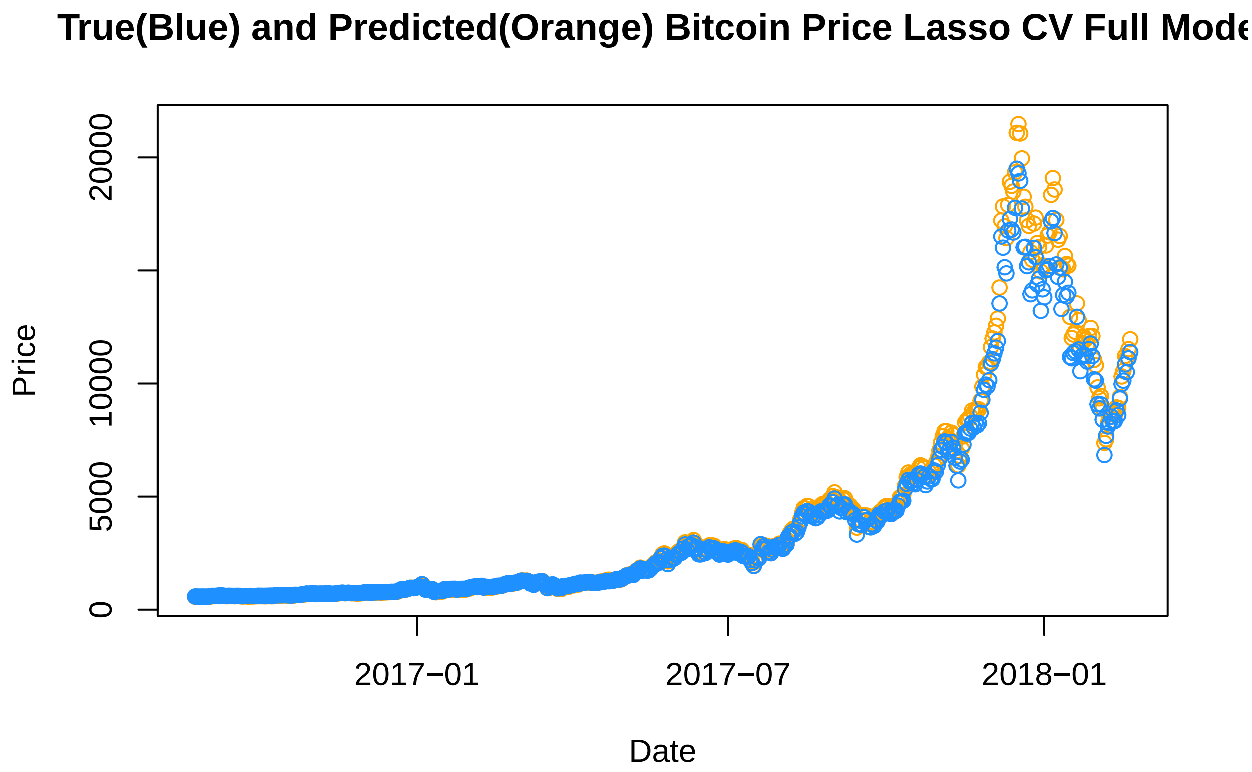
```
plot(full_model_lasso_cv)
```




```
full_model_lasso_cv_pred = predict(full_model_lasso_cv, s = full_model_lasso_cv$lambda.min, newx=glm_te
full_model_lasso_cv_test_mse = get_mse(glm_test_y, full_model_lasso_cv_pred)
full_model_lasso_cv_test_mse
```

```
## [1] 264426.1
```

```
plot(x=clean_test$Date, y= full_model_lasso_cv_pred, col = "orange",
     xlab = "Date",
     ylab = "Price",
     main = "True(Blue) and Predicted(Orange) Bitcoin Price Lasso CV Full Model")
points(x=clean_test$Date, y=clean_test$btc_market_price, col = "dodgerblue")
```



Which coefficients were eliminated? And how do they compare to the ones we were going to eliminate ahead of time?

```
full_model_lasso_cv_coef = predict(full_model_lasso_cv, s=full_model_lasso_cv$lambda.min, type="coeffic
full_model_lasso_cv_coef
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                                                    s1
## (Intercept)                             -1.709692e+00
## btc_total_bitcoins                       7.296767e-08
## btc_market_cap                           4.607216e-08
## btc_trade_volume                         .
## btc_blocks_size                         -1.297996e-07
## btc_avg_block_size                       .
## btc_n_orphaned_blocks                   -4.103936e-01
## btc_n_transactions_per_block             .
## btc_median_confirmation_time             9.405760e-02
## btc_hash_rate                           -3.791282e-05
## btc_difficulty                           .
## btc_miners_revenue                       1.738199e-05
## btc_transaction_fees                     2.452499e-02
## btc_cost_per_transaction_percent        -1.045224e-02
## btc_cost_per_transaction                 5.241233e-01
## btc_n_unique_addresses                   .
## btc_n_transactions                       .
## btc_n_transactions_total                 .
```

```
## btc_n_transactions_excluding_popular                    .
## btc_n_transactions_excluding_chains_longer_than_100     .
## btc_output_volume                                       .
## btc_estimated_transaction_volume                        .
## btc_estimated_transaction_volume_usd                    9.731340e-10
## prev_price                                              3.141402e-01
```

RIDGE

Note: We start with the default lambdas provided by the glmnet function, and subsequently try our own custom sequence of 100 lambdas to narrow in on the area where the first approach (with default lambdas) found the minimum lambda. In this case, the minimum lambda was at the lowest end of the range of default lambdas, which tells us to include even smaller values for lambda.

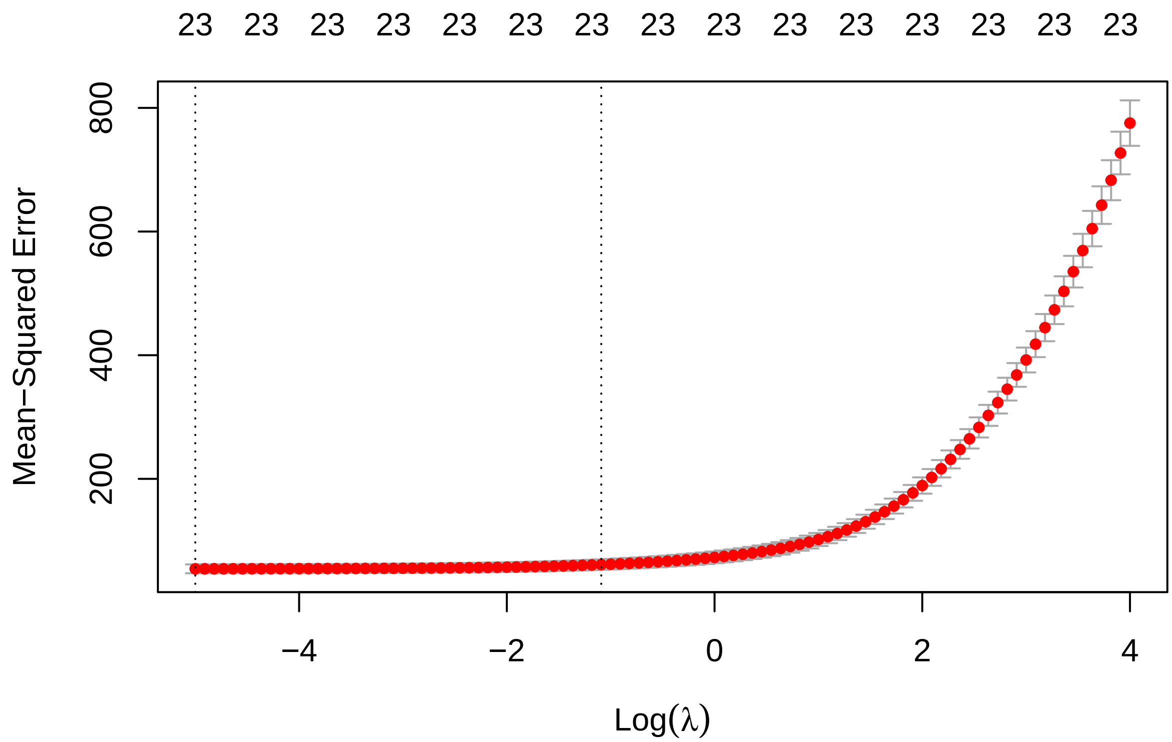
```r
lambda_seq_r = exp(seq(-5,4, length = 100))
full_model_ridge_cv = cv.glmnet(glm_train_x, glm_train_y, alpha = 0, lambda = lambda_seq_r)
print(full_model_ridge_cv)
```

```
##
## Call:  cv.glmnet(x = glm_train_x, y = glm_train_y, lambda = lambda_seq_r,      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0067   100   54.52 7.166      23
## 1se 0.3359    57   61.35 8.408      23
```

```r
full_model_ridge_cv$lambda.min
```

```
## [1] 0.006737947
```

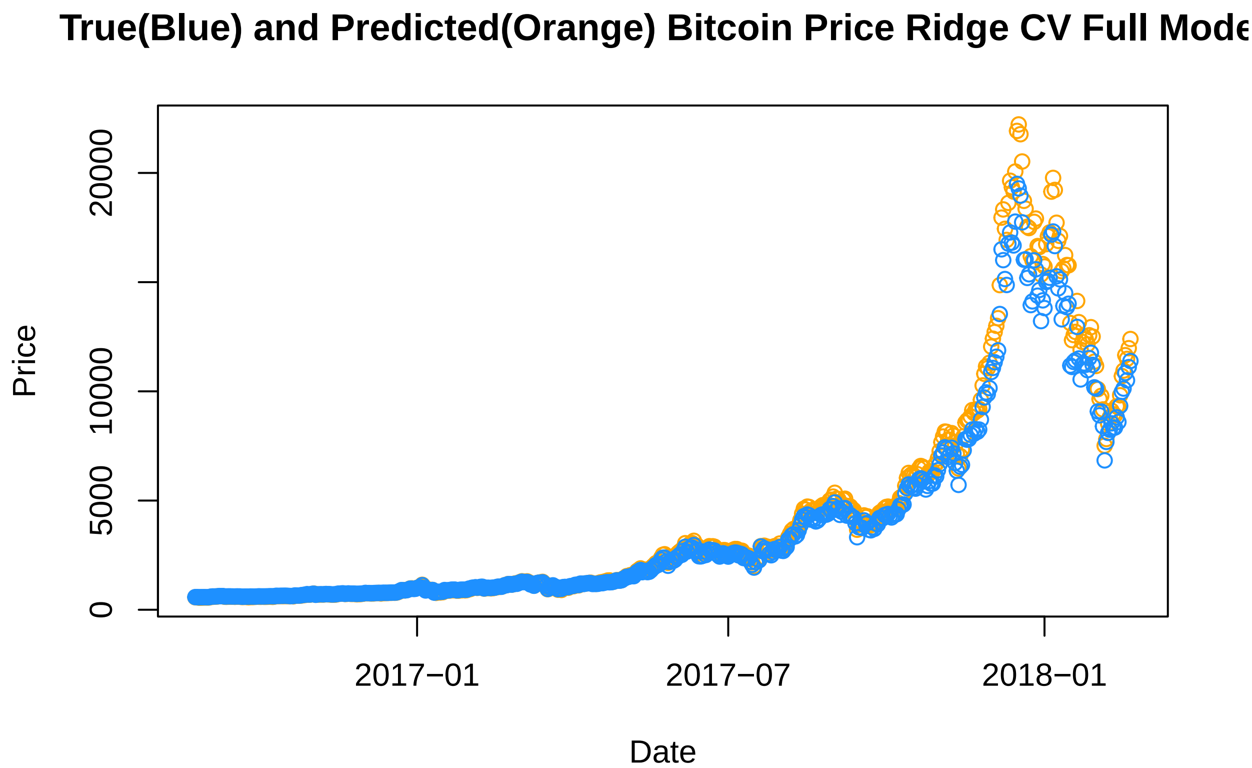
```r
plot(full_model_ridge_cv)
```




```
full_model_ridge_cv_pred = predict(full_model_ridge_cv, s = full_model_ridge_cv$lambda.min, newx=glm_te
full_model_ridge_cv_test_mse = get_mse(glm_test_y, full_model_ridge_cv_pred)
full_model_ridge_cv_test_mse
```

```
## [1] 511647.7
```

```
plot(x=clean_test$Date, y= full_model_ridge_cv_pred, col = "orange",
     xlab = "Date",
     ylab = "Price",
     main = "True(Blue) and Predicted(Orange) Bitcoin Price Ridge CV Full Model")
points(x=clean_test$Date, y=clean_test$btc_market_price, col = "dodgerblue")
```



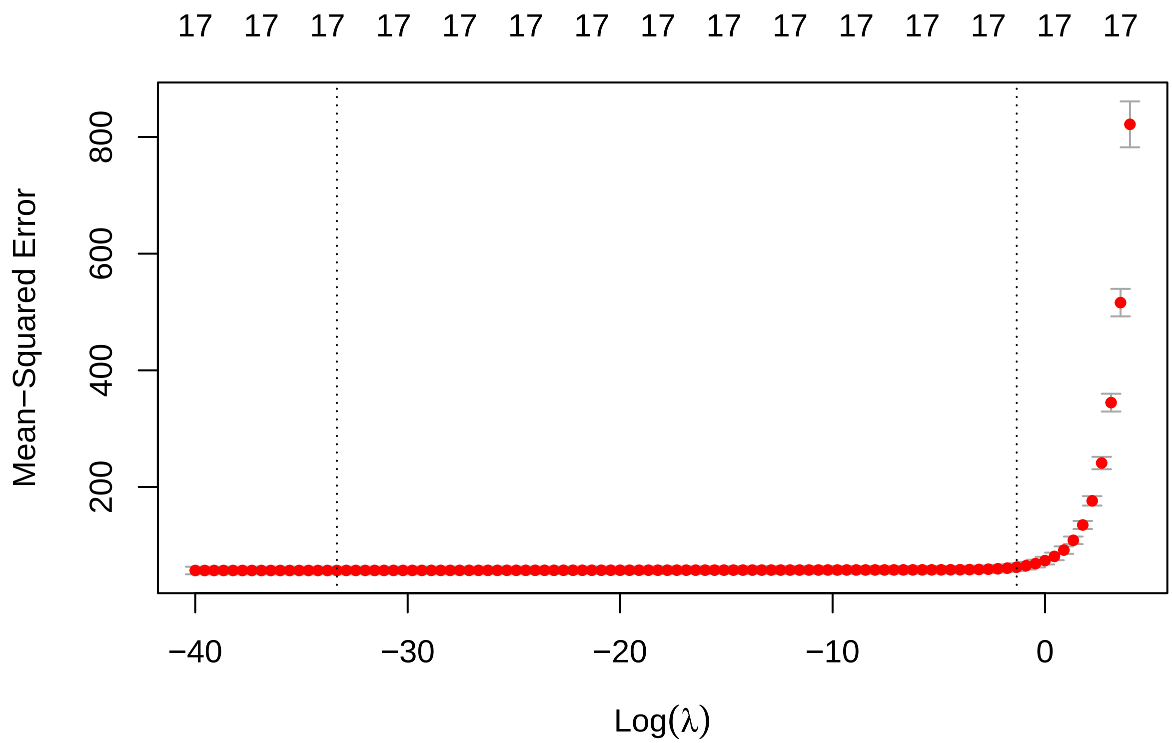
Ridge with reduced variables:

Category 1: Unlikely to be added in later btc_n_orphaned_blocks btc_median_confirmation_time btc_cost_per_transaction_percent btc_cost_per_transaction btc_n_transactions_excluding_chains_longer_than_100 btc_output_volume

First Reduction:

```
# remove the set of predictors listed in Category 1 + btc_n_orphaned_blocks

col_rem1 = c("btc_n_orphaned_blocks", "btc_median_confirmation_time", "btc_cost_per_transaction_percent

red1_train = clean_train[, !(names(clean_train) %in% col_rem1)]
red1_test = clean_test[, !(names(clean_test) %in% col_rem1)]

# as before, convert to matrix and remove Date and btc_market_price
glm_train_x_red1 = as.matrix(red1_train[, !(names(red1_train) %in% rem_price_date)])
glm_test_x_red1 = as.matrix(red1_test[, !(names(red1_test) %in% rem_price_date)])

# the y-values are of course unchanged, and we keep using glm_train_y and glm_test_y
```

```r
lambda_seq_r1 = exp(seq(-40,4, length = 100))
red1_ridge_cv = cv.glmnet(glm_train_x_red1, glm_train_y, alpha = 0, lambda = lambda_seq_r1)
plot(red1_ridge_cv)
```




```r
log(red1_ridge_cv$lambda.min)
```

```
## [1] -33.33333
```

```r
red1_ridge_cv_pred = predict(red1_ridge_cv, s = red1_ridge_cv$lambda.min, newx=glm_test_x_red1)
red1_ridge_cv_test_mse = get_mse(glm_test_y, red1_ridge_cv_pred)
red1_ridge_cv_test_mse
```

```
## [1] 596330.9
```

Ridge Regression with second reduction: Now we remove Category 1 and 0

Category 1: Unlikely to be added in later btc_n_orphaned_blocks btc_median_confirmation_time btc_cost_per_transaction_percent btc_cost_per_transaction btc_n_transactions_excluding_chains_longer_than_100 btc_output_volume

Category 0: Might be added in later btc_n_transactions_per_block btc_n_unique_addresses btc_n_transactions btc_n_transactions_excluding_popular

```r
# remove the set of predictors listed in Category 1 +  Category 0

col_rem2 = c("btc_n_orphaned_blocks", "btc_median_confirmation_time", "btc_cost_per_transaction_percent


red2_train = clean_train[, !(names(clean_train) %in% col_rem2)]
red2_test = clean_test[, !(names(clean_test) %in% col_rem2)]

# as before, convert to matrix and remove Date and btc_market_price
glm_train_x_red2 = as.matrix(red2_train[, !(names(red2_train) %in% rem_price_date)])
glm_test_x_red2 = as.matrix(red2_test[, !(names(red2_test) %in% rem_price_date)])

# the y-values are of course unchanged, and we keep using glm_train_y and glm_test_y
```

```r
lambda_seq_r2 = exp(seq(-2,4, length = 100))
red2_ridge_cv = cv.glmnet(glm_train_x_red2, glm_train_y, alpha = 0, lambda = lambda_seq_r2)
print(red2_ridge_cv)
```

```
##
## Call:  cv.glmnet(x = glm_train_x_red2, y = glm_train_y, lambda = lambda_seq_r2,      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.1353   100   59.62 3.907      13
## 1se 0.3569    84   63.45 4.110      13
```

```r
red2_ridge_cv_pred = predict(red2_ridge_cv, s = red2_ridge_cv$lambda.min, newx=glm_test_x_red2)
red2_ridge_cv_test_mse = get_mse(glm_test_y, red2_ridge_cv_pred)
red2_ridge_cv_test_mse
```

```
## [1] 445882.3
```

This model has a lower MSE than the first Ridge Model, but still a higher MSE than the Lasso Model, which currently outperforms all. Let us see the coefficients:

```r
print("The coefficients for the red2_ridge_cv model:")
```

```
## [1] "The coefficients for the red2_ridge_cv model:"
```

```r
print(coef(red2_ridge_cv))
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                                         s1
## (Intercept)                  -4.418270e+00
## btc_total_bitcoins            7.959108e-07
## btc_market_cap                4.429114e-08
## btc_trade_volume             -7.422972e-08
## btc_blocks_size              -5.897005e-04
## btc_avg_block_size            7.209418e-01
```

```
## btc_hash_rate                           -4.263976e-05
## btc_difficulty                           6.540133e-11
## btc_miners_revenue                       2.279869e-05
## btc_transaction_fees                     2.712505e-02
## btc_n_transactions_total                 1.211464e-07
## btc_estimated_transaction_volume        -1.035295e-06
## btc_estimated_transaction_volume_usd  2.604877e-08
## prev_price                               3.492943e-01
```

```
plot(x=clean_test$Date, y= red2_ridge_cv_pred, col = "orange",
     xlab = "Date",
     ylab = "Price",
     main = "True(Blue) and Predicted(Orange) Bitcoin Price Ridge CV Red2 Model")
points(x=clean_test$Date, y=clean_test$btc_market_price, col = "dodgerblue")
```