Final Project

Katelyn Liu, Reinhard Wilmer, Vivian Qian Yu Wong, Veronica Zhao

11/13/2020

Introduction

In this data report, we attempt to predict the forward return of a target asset based on the historical price series of the target asset and two other assets. In order to build a good model, we first derive features such as the backward return and forward return of each asset at different times (minutes).

We then fit and compare the following statistical models: linear regression, KNN regression, ridge regression, lasso, and principal components regression (PCR). For each model, we look at the validation MSE as well as the in-sample and out-of-sample correlation.

The dataset used for this analysis includes the given minutely prices of the three assets over a year. In other words, there are 524,160 rows.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(FNN)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.0-2
```

```
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
br_return <- function(Asset_i,h){
    ((Asset_i - Asset_i[h])/Asset_i[h])
}

fr_return <- function(Asset_i,h){
    ((Asset_i[h] - Asset_i)/Asset_i)
}

bret <- read.csv('bret.csv')
data <- read.csv("final_project-1.csv")
data$X <- data$X + 1
```

Exploratory Analysis

A. Backward Rolling Correlation

B. Linear Regression

The first analysis that we are going to look at is the linear regression model. We first calculate the 10-minute forward return of Asset 1 by creating a function that incorporates the 3-minute, 10-minute, and 30-minute backward returns of all three assets for each minute throughout the year. The function we used is

$$r_f(t,h) = \frac{s(min(t+h,T)) - s(t)}{s(t)}$$

where t = time and T = 524,160.

```r
bret_fwd <- bret %>% mutate(t = row_number(),
    h_fwd = ifelse((h_fwd = t+10)<= nrow(bret), h_fwd, nrow(bret))) %>%
    mutate(Asset_1_Fwd_10 = fr_return(data$Asset_1,h_fwd)) %>% select(-c(t,h_fwd))

train_id = c(1:(0.7*nrow(bret_fwd)))
train <- bret_fwd[train_id, ]
test <- bret_fwd[-train_id, ]
```

After creating the new response variable, we choose the first 70% of the new dataset to be the training set and the last 30% to be the testing set. We will use the training set to fit and train our model. After fitting the linear regression model on the training set, we obtain the results that the 3-minute, 10-minute, and 30-minute backward returns of Asset 1 and Asset 2, and the 3-minute backward return of Asset 3 are statistically significant at the alpha level of 0.001, while the 10-minute backward return of Asset 3 is statistically significant at the alpha level of 0.05. The 30-minute backward return of Asset 3 is not statistically significant.

```r
m1 <- lm(Asset_1_Fwd_10 ~ ., data = train)
summary(m1)
```

```
##
## Call:
## lm(formula = Asset_1_Fwd_10 ~ ., data = train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.147289 -0.000919  0.000010  0.000928  0.085484
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.036e-05  4.758e-06  -2.178 0.029412 *
## Asset_1_BRet_3   4.071e-02  4.076e-03   9.987  < 2e-16 ***
## Asset_1_BRet_10  1.706e-02  2.537e-03   6.725 1.75e-11 ***
## Asset_1_BRet_30  6.268e-03  1.261e-03   4.972 6.63e-07 ***
## Asset_2_BRet_3   2.593e-02  2.228e-03  11.636  < 2e-16 ***
## Asset_2_BRet_10 -5.400e-03  1.435e-03  -3.764 0.000168 ***
## Asset_2_BRet_30  8.880e-03  7.609e-04  11.672  < 2e-16 ***
## Asset_3_BRet_3   1.835e-02  2.247e-03   8.167 3.17e-16 ***
## Asset_3_BRet_10  3.429e-03  1.441e-03   2.379 0.017359 *
## Asset_3_BRet_30 -9.689e-04  7.295e-04  -1.328 0.184136
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002882 on 366902 degrees of freedom
## Multiple R-squared:  0.004606,   Adjusted R-squared:  0.004581
## F-statistic: 188.6 on 9 and 366902 DF,  p-value: < 2.2e-16
```

Furthermore, to analyze the relationship between the predicted response of the model and the values of the true response variable, we will measure both the in-sample correlation and the out-of-sample correlation between the predicted and true response. The in-sample correlation we obtained is 0.0678, and the out-of-sample correlation is 0.04068. To look at this visually, we can observe the three-week backward rolling correlation between the true response and the predicted response of the 10-min forward return of Asset 1. We can see that the correlation structure seems to be stationary over the year. The vertical red line represents the boundary between the training backward correlation and testing backward correlation.

```
train_pred <- predict.lm(m1, train)
lrin <- cor(train$Asset_1_Fwd_10, train_pred)
test_pred <- predict.lm(m1, test)
lrout <- cor(test$Asset_1_Fwd_10, test_pred)
lrin
```

```
## [1] 0.06786533
```

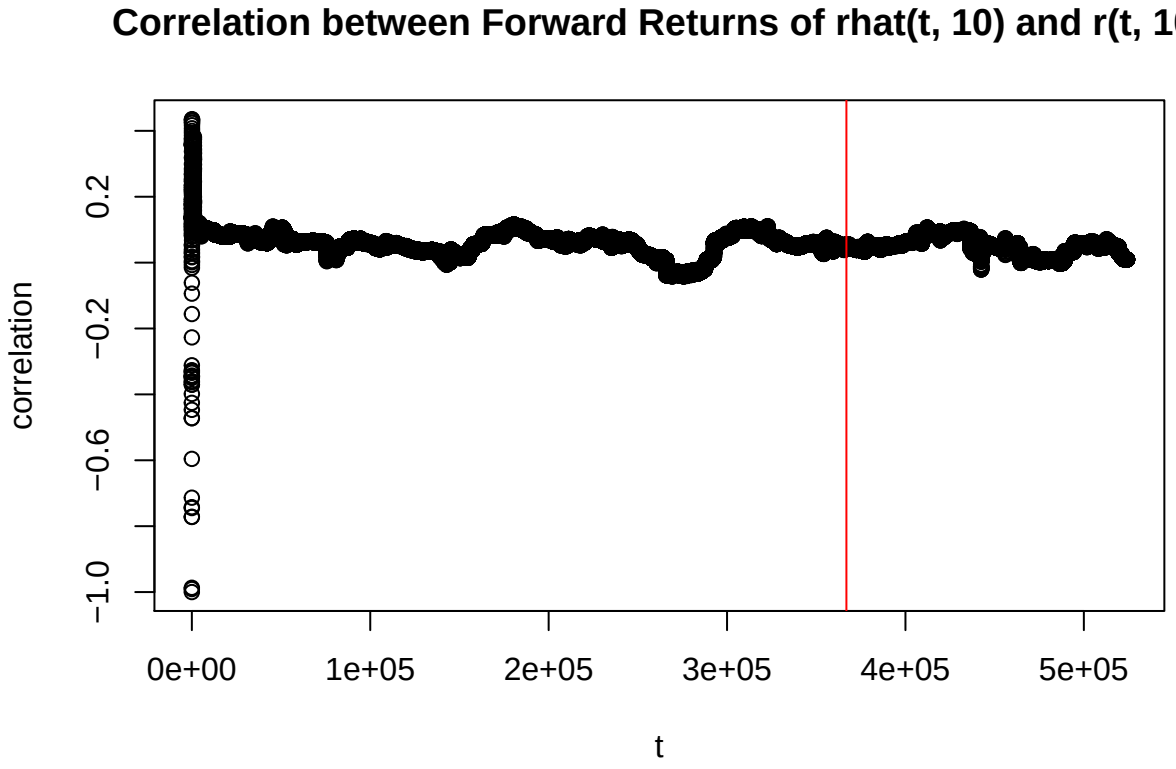
```
lrout
```

```
## [1] 0.04068153
```

```
preds = c()
preds = append(preds, as.vector(train_pred))
preds = append(preds, as.vector(test_pred))

rolling = data.frame(pred = preds, actual = bret_fwd$Asset_1_Fwd_10)
rolling$X = 1:nrow(rolling)

w = 21*24*60
rolling = rolling %>% mutate(index = ifelse((X-w)<=1,1,(X-w)))
roll_cor <- function(i){
    x = rolling$pred[rolling$index[i]:rolling$X[i]]
```

```
    y = rolling$actual[rolling$index[i]:rolling$X[i]]
    return(cor(x, y))
}

X = c(1:nrow(rolling))
r = vector()
rolling$correlation = sapply(X, roll_cor)
```

```
plot(rolling$correlation ~ rolling$X, xlab = 't', ylab = 'correlation', main = 'Correlation between Forw
abline(v=nrow(train), col='red')
```



C. KNN Regression

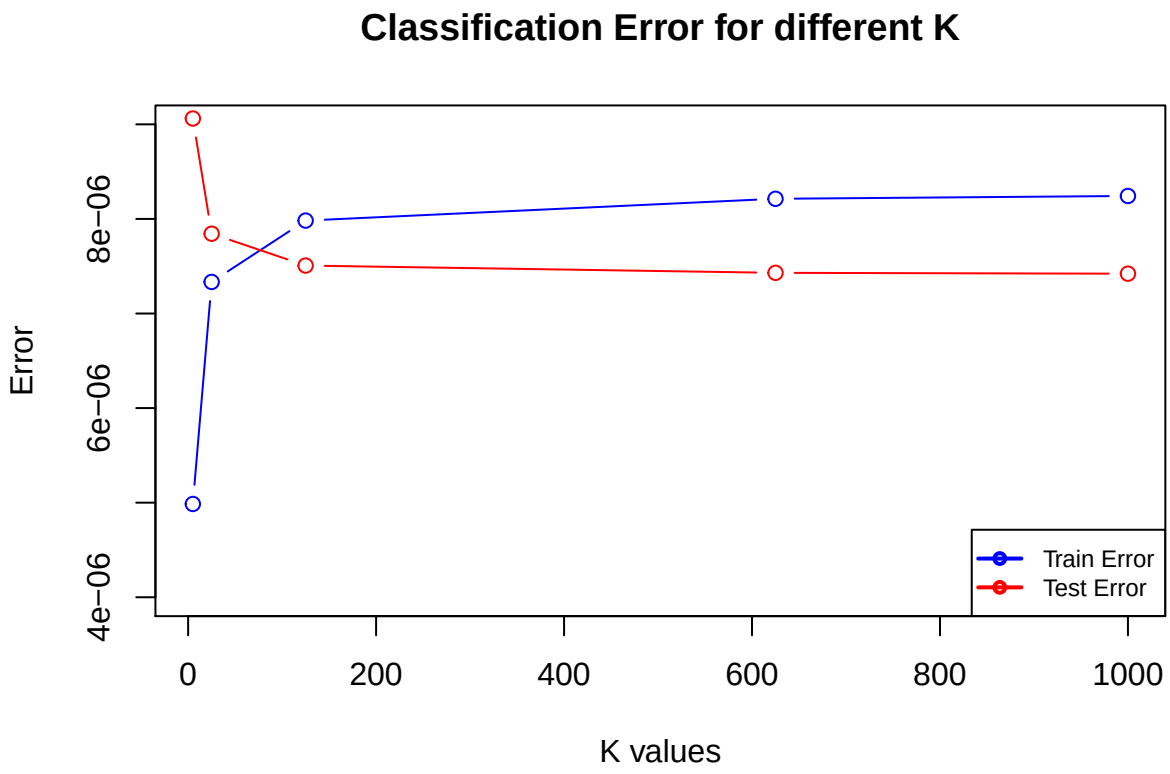
The next method that we are going to use is the KNN method. We will run KNN regression with the same features and response variables as above, along with the same testing and training set. We are going to observe the training and validation MSE for each K = 5, 25, 125, 625, 1000.

```
X_train <- train %>% select(-c(Asset_1_Fwd_10))
X_test <- test %>% select(-c(Asset_1_Fwd_10))

trainMSE <- c()

knnTrain <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 5)
trainMSE[1] <-mean((train$Asset_1_Fwd_10-knnTrain$pred)^2)
```

```
knnTrain <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 25)
trainMSE[2] <-mean((train$Asset_1_Fwd_10-knnTrain$pred)^2)
knnTrain <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 125)
trainMSE[3] <-mean((train$Asset_1_Fwd_10-knnTrain$pred)^2)
knnTrain <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 625)
trainMSE[4] <-mean((train$Asset_1_Fwd_10-knnTrain$pred)^2)
knnTrain <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 1000)
trainMSE[5] <-mean((train$Asset_1_Fwd_10-knnTrain$pred)^2)
k_range = c(5, 25, 125, 625, 1000)
testMSE =c()
knnTest <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 5)
testMSE[1] <-mean((test$Asset_1_Fwd_10-knnTest$pred)^2)
knnTest <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 25)
testMSE[2] <-mean((test$Asset_1_Fwd_10-knnTest$pred)^2)
knnTest <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 125)
testMSE[3] <-mean((test$Asset_1_Fwd_10-knnTest$pred)^2)
knnTest <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 625)
testMSE[4] <-mean((test$Asset_1_Fwd_10-knnTest$pred)^2)
knnTest <-knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 1000)
testMSE[5] <-mean((test$Asset_1_Fwd_10-knnTest$pred)^2)
```

After calculating and plotting the range of K values as well as the associated training MSE and testing MSE for each value, we can see that the K-value with the lowest validation MSE is K=1000. Therefore, we will use this K-value to make predictions for the whole year and analyze the correlation between the predictions and true responses.

```
plot(k_range, trainMSE, type = "b", xlab = "K values", ylab = "Error", col = "blue", main = "Classifica
lines(k_range, testMSE, type = "b", col = "red")
legend("bottomright", legend = c("Train Error", "Test Error"), cex = 0.75, col = c("blue", "red"), lwd =
pch = c(1, 1), lty = c(1, 1))
```



The in-sample correlation between the true and predicted response is 0.118332, and the out-of-sample correlation is 0.04320432. Compared to the correlation coefficients of linear regression, those of the KNN regression with K = 1000 are slightly higher.

```
knn_m <- knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_train, k = 1000)
train_pred <- knn_m$pred
knnin <- cor(train$Asset_1_Fwd_10, train_pred)
knn_m <- knn.reg(train = X_train, y = train$Asset_1_Fwd_10, test = X_test, k = 1000)
test_pred <- knn_m$pred
knnout <- cor(test$Asset_1_Fwd_10, test_pred)
knnin
```

```
## [1] 0.118332
```

```
knnout
```

```
## [1] 0.04320432
```

C. Ridge and LASSO

To extend our analysis, we will consider backward returns in more time horizons. Similar to the previous problems, we will calculate the backward return for all three assets in the t-minute ranges from t = 3,10,30,60,120,180,240,360,480,600,720,960,1200,1440. We will use the first 70% of the final dataset as training data and the last 30% as validation data.

```
table <- data %>% mutate(h3 = ifelse((h3 = X-3)<=1,1,h3),
                h10 = ifelse((h10 = X-10)<=1,1,h10),
                h30 = ifelse((h30 = X-30)<=1,1,h30),
                h60 = ifelse((h60 = X-60)<=1,1,h60),
                h120 = ifelse((h120 = X-120)<=1,1,h120),
                h180 = ifelse((h180 = X-180)<=1,1,h180),
                h240 = ifelse((h240 = X-240)<=1,1,h240),
                h360 = ifelse((h360 = X-360)<=1,1,h360),
                h480 = ifelse((h480 = X-480)<=1,1,h480),
                h600 = ifelse((h600 = X-600)<=1,1,h600),
                h720 = ifelse((h720 = X-720)<=1,1,h720),
                h960 = ifelse((h960 = X-960)<=1,1,h960),
                h1200 = ifelse((h1200 = X-1200)<=1,1,h1200),
                h1440 = ifelse((h1440 = X-1440)<=1,1,h1440)) %>%
mutate(Asset_1_BRet_3 = br_return(Asset_1,h3),
       Asset_1_BRet_10 = br_return(Asset_1, h10),
       Asset_1_BRet_30 = br_return(Asset_1, h30),
       Asset_1_BRet_60 = br_return(Asset_1, h60),
       Asset_1_BRet_120 = br_return(Asset_1, h120),
       Asset_1_BRet_180 = br_return(Asset_1, h180),
       Asset_1_BRet_240 = br_return(Asset_1, h240),
       Asset_1_BRet_360 = br_return(Asset_1, h360),
       Asset_1_BRet_480 = br_return(Asset_1, h480),
       Asset_1_BRet_600 = br_return(Asset_1, h600),
       Asset_1_BRet_720 = br_return(Asset_1, h720),
       Asset_1_BRet_960 = br_return(Asset_1, h960),
       Asset_1_BRet_1200 = br_return(Asset_1, h1200),
       Asset_1_BRet_1440 = br_return(Asset_1, h1440),
       Asset_2_BRet_3 = br_return(Asset_2,h3),
       Asset_2_BRet_10 = br_return(Asset_2, h10),
       Asset_2_BRet_30 = br_return(Asset_2, h30),
       Asset_2_BRet_60 = br_return(Asset_2, h60),
       Asset_2_BRet_120 = br_return(Asset_2, h120),
       Asset_2_BRet_180 = br_return(Asset_2, h180),
       Asset_2_BRet_240 = br_return(Asset_2, h240),
       Asset_2_BRet_360 = br_return(Asset_2, h360),
       Asset_2_BRet_480 = br_return(Asset_2, h480),
       Asset_2_BRet_600 = br_return(Asset_2, h600),
       Asset_2_BRet_720 = br_return(Asset_2, h720),
       Asset_2_BRet_960 = br_return(Asset_2, h960),
       Asset_2_BRet_1200 = br_return(Asset_2, h1200),
       Asset_2_BRet_1440 = br_return(Asset_2, h1440),
       Asset_3_BRet_3 = br_return(Asset_3,h3),
       Asset_3_BRet_10 = br_return(Asset_3, h10),
       Asset_3_BRet_30 = br_return(Asset_3, h30),
       Asset_3_BRet_60 = br_return(Asset_3, h60),
       Asset_3_BRet_120 = br_return(Asset_3, h120),
       Asset_3_BRet_180 = br_return(Asset_3, h180),
       Asset_3_BRet_240 = br_return(Asset_3, h240),
       Asset_3_BRet_360 = br_return(Asset_3, h360),
       Asset_3_BRet_480 = br_return(Asset_3, h480),
       Asset_3_BRet_600 = br_return(Asset_3, h600),
       Asset_3_BRet_720 = br_return(Asset_3, h720),
```

```
       Asset_3_BRet_960 = br_return(Asset_3, h960),
       Asset_3_BRet_1200 = br_return(Asset_3, h1200),
       Asset_3_BRet_1440 = br_return(Asset_3, h1440))
table = table %>% dplyr::select(c(19:ncol(table)))

table$X = 1:nrow(table)

bret_RL <- table %>% mutate(h10 = ifelse((h10 = X+10)<=nrow(table),h10,nrow(table))) %>% mutate(Asset_1

train_id = c(1:(0.7*nrow(bret_RL)))
train_rl <- bret_RL[train_id, ]
test_rl <- bret_RL[-train_id, ]
```

First, we will fit a ridge regression model. In order to find a the best tuning parameter choice of lambda, we will calculate the validation MSE of a range of lambdas and choose the lambda associated with the lowest validation MSE. The results show that the lambda that provides the smallest validation MSE is 0.01. Using this lambda t fit the ridge regression model, we are able to calculate the correlation coefficients. The in-sample correlation between the true response and the predicted response is 0.071036, while the out-of-sample correlation is 0.036476. Compared to previously, the in-sample correlation is slightly higher than that of the linear regression model, while the out-of-sample correlation is lower than both the linear regression and the KNN model.

```
grid = 10^seq(10, -2, length=100)
X_train_RL = model.matrix(Asset_1_Fwd_10 ~ ., train_rl)[, -1]
y_train_RL = train_rl$Asset_1_Fwd_10
X_test_RL = model.matrix(Asset_1_Fwd_10  ~ ., test_rl)[, -1]
y_test_RL = test_rl$Asset_1_Fwd_10
ridge.mod = glmnet(X_train_RL, y_train_RL, alpha=0, lambda=grid)
mse = vector()
for(i in 1:length(grid)){
    ridge.pred_test = predict(ridge.mod, s=grid[i], newx=X_test_RL)
    mse[i] = mean((ridge.pred_test - y_test_RL)^2)
}
bestlam = grid[which.min(mse)]
bestlam
```

```
## [1] 0.01
```

```
ridge.pred_train = predict(ridge.mod, s=bestlam, newx=X_train_RL)
ridgein <- cor(ridge.pred_train, y_train_RL)
ridge.pred_test = predict(ridge.mod, s=bestlam, newx=X_test_RL)
ridgeout <-cor(ridge.pred_test, y_test_RL)
ridgein
```

```
##          [,1]
## 1 0.07103612
```

```
ridgeout
```

```
##          [,1]
## 1 0.03647633
```

Now, we are going to take a look at LASSO regression with the same feature and response variables as that of the ridge regerssion model. Using the same process as before, the lambda associated with the lowest validation error is 4.008806e-05.

```
grid_lass = 10^seq(10, -5, length=200)
lasso.mod = glmnet(X_train_RL, y_train_RL, alpha=1, lambda=grid_lass)
mse = vector()
for(i in 1:length(grid_lass)){
    lasso.pred_test = predict(lasso.mod, s=grid_lass[i], newx=X_test_RL)
    mse[i] = mean((lasso.pred_test - y_test_RL)^2)
}
bestlam = grid_lass[which.min(mse)]
bestlam
```

```
## [1] 4.008806e-05
```

Using this lambda, the in-sample correlation is 0.06867927 while the out-of-sample correlation is 0.03929488.

```
lasso.pred_train = predict(lasso.mod,s=bestlam,newx=X_train_RL)
lassoin <- cor(lasso.pred_train, y_train_RL)
lasso.pred_test = predict(lasso.mod, s=bestlam, newx=X_test_RL)
lassoout <- cor(lasso.pred_test, y_test_RL)
lassoin
```

```
##          [,1]
## 1 0.06867927
```

```
lassoout
```

```
##          [,1]
## 1 0.03929488
```

E. Principal Components Regression (PCR)

Finally, we are going to run Principal Components Regression (PCR) with the same features and response variable as from the previous section's analysis. Once again, we are using the validation MSE approach to find the optimal number of components to fit the PCR model.

```
PCR <- pcr(Asset_1_Fwd_10 ~ ., data = train_rl, scale = TRUE, validation = "none")
```

After calculating the validation MSE for all PCR predictions, ranging from 1 component to 42 components, we obtain the results that the number of components associated with the lowest validation error is 6. Using 6 components to perform PCR on the training set of the data, the calculated in-sample correlation between the true response and predicted response is 0.06698, and the out-of-sample correlation is 0.04155. As compared to the ridge regression model and the linear regression model, the out-of-sample correlation of PCR is higher. However, it is only slightly lower than that of KNN regression.

```
n = c(1:42)
pcr_mse = vector()
for(i in 1:length(n)){
    PCR.pred_test<- predict(PCR, X_test_RL, ncomp = n[i])
```

```
    pcr_mse[i] = mean((PCR.pred_test - y_test_RL)^2)
}
n[which.min(pcr_mse)]
```

```
## [1] 6
```

```
PCR.pred_train <- predict(PCR, X_train_RL, ncomp = 6)
pcrin <- cor(PCR.pred_train, y_train_RL)
PCR.pred_test<- predict(PCR, X_test_RL, ncomp = 6)
pcrout <- cor(PCR.pred_test, y_test_RL)
pcrin
```

```
## [1] 0.06698264
```

```
pcrout
```

```
## [1] 0.04155347
```

Results

To compare all of our results, let's take a look at the various in-sample and out-of-sample correlations of the models below. We can see that PCR gave us the lowest in sample correlation, while KNN gave us the highest. The linear regression, lasso, and PCR models all had approximately the same values of in sample correlation. However, we also observe that KNN gave us the highest out of sample correlation, while ridge gave us the lowest. Similarly, the PCR, KNN, and linear regression models all gave correlation values greater than 0.04. With the out of sample correlaiton, it seems as if these models have predictions that are closer to the true response than compared to the ridge and lasso models.

```
d <-data.frame("In-Sample Correlation"=c(lrin, knnin, ridgein, lassoin, pcrin),
               "Out-of-Sample Correlation" = c(lrout, knnout, ridgeout, lassoout, pcrout))
rownames(d) <-c("Linear Regression","KNN","Ridge", "Lasso","PCR")
knitr::kable(d)
```

	In.Sample.Correlation	Out.of.Sample.Correlation
Linear Regression	0.0678653	0.0406815
KNN	0.1183320	0.0432043
Ridge	0.0710361	0.0364763
Lasso	0.0686793	0.0392949
PCR	0.0669826	0.0415535