Assignment #5:

By: Valon Tika

Introduction:

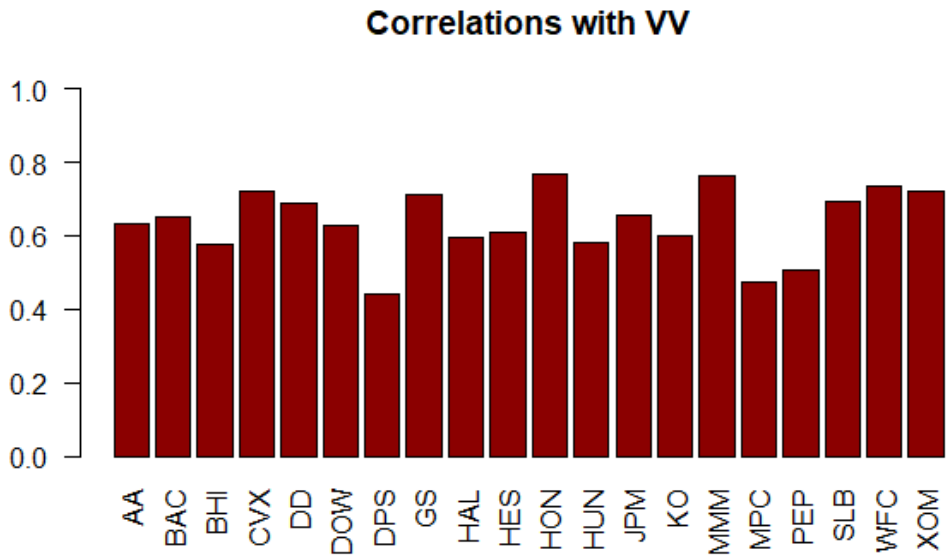
The goal of this assignment is to perform more detailed analysis of certain methods of variable selection in a multivariate analysis. The assignment will use the principal component analysis (PCA) as a method of dimension reduction to try and eliminate multicollinearity in a multivariate multiple linear regression analysis. We will also present methods of graphical representations of the given results of the PCA analysis of the data. We will be using a dataset that consists of the daily closing price of 20 stocks along with the daily closing price of a large-cap index fund, the Vanguard. This assignment will consist of 6 sections, each section detailing the purpose and outcome of the analysis.

Section 1:

Data that was provided was imported into the model and cleansed as needed for more detailed exploratory data analysis. Upon initial analysis of the data, 20 stocks were provided along with the Vanguard (VANGUARD) as our total list of available variables. The closing prices for the 21 variables ranged from 1/3/2012 to 12/31/1013. This spans to 2 years' worth of market performance for the 20 stocks and the VANGUARD index fund. The logarithmic return of each individual stock per day will be created to show the variation of the logarithmic returns of the indexed fund. These values will be used for both the linear regression analysis and PCA and compared to the actual values of the closing price. The VANGUARD index fund will be considered as the response variable for this analysis while we will try to use any (or all, if applicable) the 20 stocks as predictor variables.

Section 2:

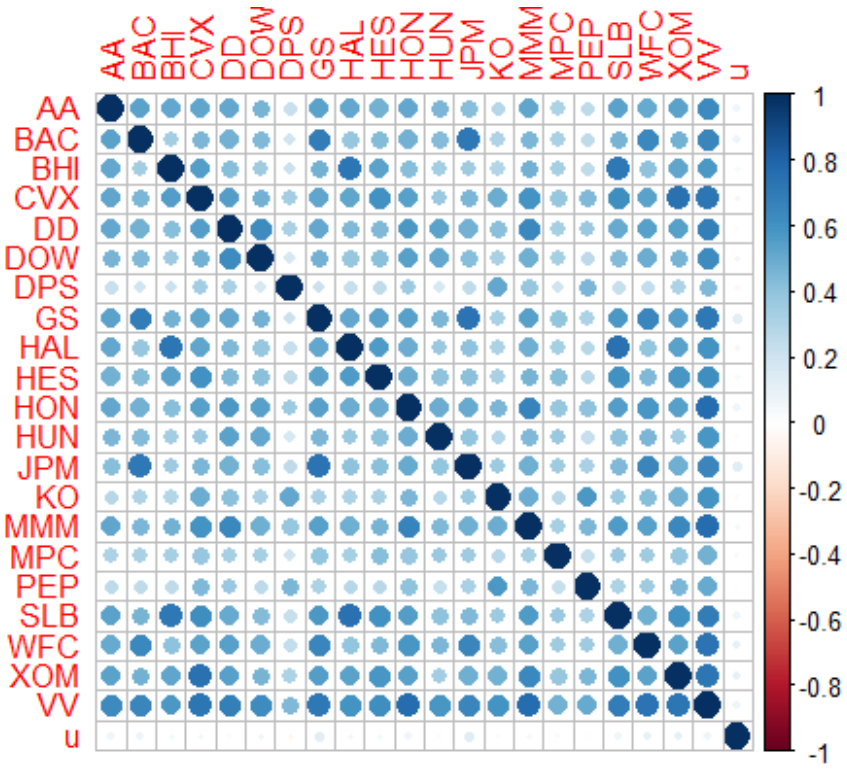
The next step is to perform more exploratory data analysis of the 20 variables and their correlation to the Vanguard. A correlation matrix was created to show whether the stock price for the twenty stocks are correlated with the Vanguard. Below is a representation of the correlation of the 20 stocks to the Vanguard.



From the initial plot above, it appears we could consider a list of stocks that have a correlation percentage that seem somewhat high relative to the Vanguard. We will need to plot this in a different manner to get more details about this correlation matrix.

Section 3:

Another way to represent the correlation of the 20 stocks to the Vanguard is to use a correlation plot. Below is the correlation plot results.



The information conveyed by the correlation plot is almost the same as the information conveyed by the bar plot in the part above. In correlation plot, we see the pair correlations between each and every column of the dataset, while in the bar plot we see the correlation of all stock prices with the Vanguard.

A statistical graph is a way to represent quantitative statistical data while data visualization is the means to get visual insights which may or may not be quantitative. A way to measure the effect of multicollinearity within a MLR is by observing the variance inflation factor (VIF) of each variable in a model. A low VIF values means essentially that there is no multicollinearity effect of the predictor variable. It seems as though DPS, MPC and PEP have low correlation with the Vanguard index and thus have a low VIF value. Stocks SLB, HON and MMM have high correlation and thus higher VIF values.

Section 4:

In this section, we will explore the variable inflation factor of two models, the full model of 20 variables as predictor variables and some random model that was selected to interpret. Below are the two explicit models:

Model 1:

$$VV = B_0 + GS * B_1 + DD * B_2 + DOW * B_3 + HON * B_4 + HUN * B_5 + JPM * B_6 + KO * B_7 + MMM * B_8 + XON * B_9$$

Model 2:

$$VV = B_0 + AA * B_1 + BAC * B_2 + GS * B_3 + JPM * B_4 + WFC * B_5 + BHI * B_6 + CVX * B_7 + DD * B_8 + DOW * B_9 + DPS * B_{10} + HAL * B_{11} + HES * B_{12} + HON * B_{13} + HUN * B_{14} + KO * B_{15} + MMM * B_{16} + MPC * B_{17} + PEP * B_{18} + SLB * B_{19} + XOM * B_{20}$$

We will interpret the VIF model results of the two models to understand and compare between the two. For interpretation of VIF for each variable, a value of 2.5 can be set as a threshold for multicollinearity.

(4) In weaker models a VIF value of 2.5 can be threshold for multicollinearity.

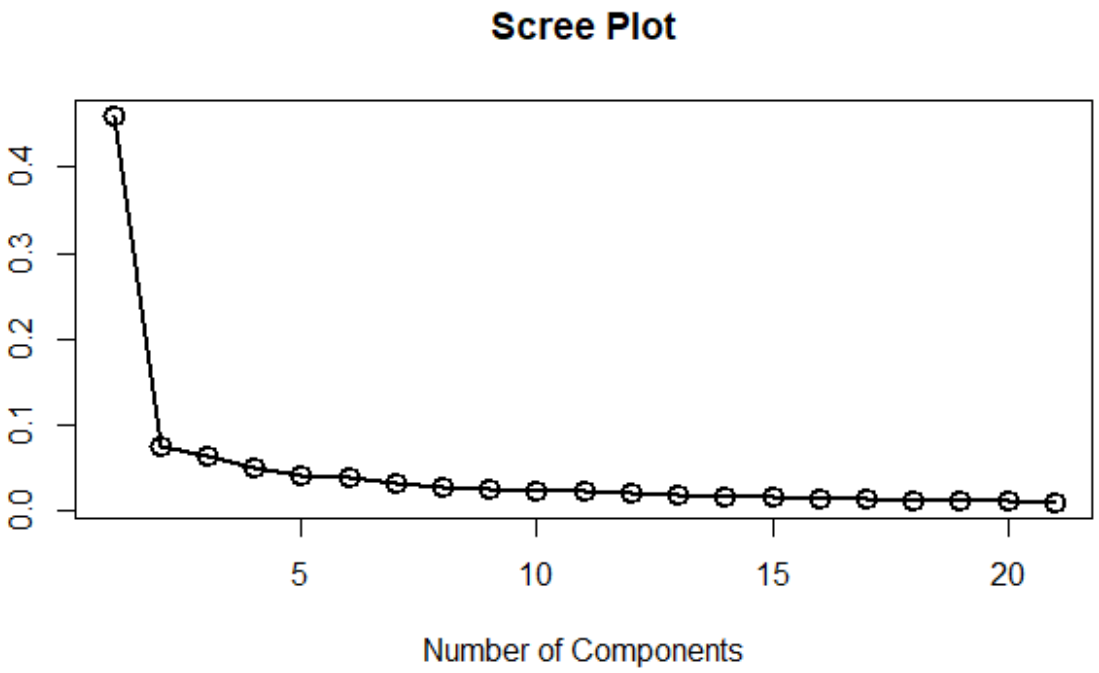
Section 5:

(5) 51

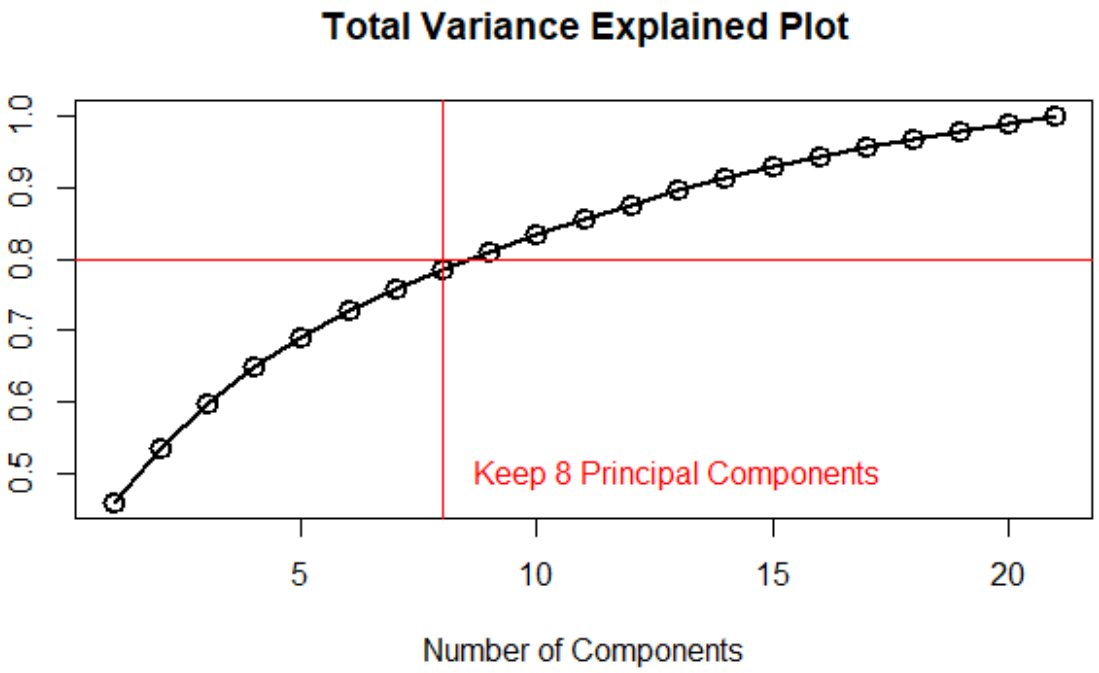
In PCA, you split covariance (or correlation) matrix into scale part (eigenvalues) and direction part (eigenvectors). You may then endow eigenvectors with the scale, which are loadings. So, loadings thus become comparable by magnitude with the covariances/correlations observed between the variables, because what had been drawn out from the variables' covariation now returns in the form of the covariation between the variables and the principal components. Loadings, more specifically, are the covariances/correlations between the original variables and the unit-scaled components.

Section 6:

Moving forward, there needs to be a method to better determine which principal components the model should select to explain enough of the variance that you are comfortable with. After removing multicollinearity features and further plotting, it seems as though we need to keep 8 principal components within this model. Let’s first take a look at the scree plot:



We see from the scree plot that after 8 components there is no meaningful increase in the variance explained by the model. Thus, we keep 8 components. This is considered using the Kaiser Rule in some context, since we are trying to keep the minimum number of components needed to represent the maximum number variance within the construct that is represented. Let us look further by plotting this in a different way:



By examining this plot, we see that roughly 80% of the total variance is preserved by maintaining 8 principal components within the model.

Section 7:

Let's say you have a cloud of N points in, say, 3D (which can be listed in a 100x3 array). Then, the principal components analysis (PCA) fits an arbitrarily oriented ellipsoid into the data. The principal component score is the length of the diameters of the ellipsoid.

With that, let us next take a testing and training set of the data and build out a PCA score model to see the performance of the principal components selected. Let's then fit them within an MLR both in sample and out of sample to assess the mean absolute error of the model.

The data was split randomly, assigning 70% of the data as the training set and 30% as the testing set. The RSE and R squared results of the linear model compared closely to the other models using the data earlier in this assignment. For the 8 principal components, the R squared (adjusted) was .8588 with an RSE of .002715. upon observation of the p-values of the 8 chosen principal components, only 4 were shown as statistically significant.

The VIF values of predictor variables in our model is 1 since these are principal components and are chosen such that variance explained is the maximum.

The calculation of the MAE for model "pca1" for the 8 components were .00205 for the testing and .00207 for the training sets.

Section 8:

To compare the two models, the below table will show the model results comparison of each:

Model	RSE	R-Squared	MAE Train	MAE Test
Model 1	0.003077	0.8346	0.00221327	0.002193018
Model 2	0.002766	0.8703	0.001989718	0.002007787

The comparison between both models are similar at first glance. Model 2 has the lowest MAE on test data when comparing the two. The R-squared and RSE for Model 2 seems to perform better as well. When wanting to determine the best model to use, ideally it comes down to optimal model performance with as few variables as needed to reach to that goal. This will help us streamline and performance tune processes when putting a model into production.

Section 9:

Unsupervised decision rule does not always translate to producing the best predictive model, but in most cases it can be. It can and should try to address issues of multicollinearity and help address selecting predictor variables across a wide array of possible variables to choose from. The overall goal should be to select the minimum number of variables that explains most of the variance of any given construct. It also tries to maintain the overall R-Squared and adjusted R-Squared during the process, which makes it that much nicer to work with. It also tends to have lower MAE's which helps exhibit better model behavior as well. Consideration for supervised modeling tends to be a hassle when trying to deal with many variables at once and trying to select the best variables and having to maintain that model using supervision methods can be somewhat tedious.

When using the model provided, a different number of components were provided compared to the model approach from before. Based on the backward selection method, the number of principal components that were selected were 7 components compared to the previous efforts of 8. Below is the table results of the backwards selection model to the PCA model:

Model	RSE	R-Squared	MAE Train	MAE Test
PCA	0.003077	0.8346	0.00221327	0.002193018
Backwards	0.002715	0.8625	0.001980847	0.002053967

For the backwards model, the MAE for both in-sample and out-of-sample are lower with a higher R-Squared and lower RSE. This model seems to perform better than the PCA model that was created. From the models created, it seems as though backwards selection method performed the best with the fewest number of variables selected.

APPENDIX:

```
##install packages
install.packages('corrplot', dependencies=TRUE)
library(corrplot)
library(car)
library(MASS)
## (1)
my.data <- read.csv('stock_portfolio.csv', sep = ',', header = TRUE)
head(my.data)
str(my.data)
# Note Date is a string of dd-Mon-yy in R this is '%d-%B-%y';
my.data$RDate <- as.Date(my.data$Date,'%d-%B-%y');
sorted.df <- my.data[order(my.data$RDate),];
head(sorted.df)
AA <- log(sorted.df$AA[-1]/sorted.df$AA[-dim(sorted.df)[1]]);
# Manually check the first entry: log(9.45/9.23)
# Type cast the array as a data frame;
returns.df <- as.data.frame(AA);
returns.df$BAC <- log(sorted.df$BAC[-1]/sorted.df$BAC[-dim(sorted.df)[1]]);
returns.df$BHI <- log(sorted.df$BHI[-1]/sorted.df$BHI[-dim(sorted.df)[1]]);
returns.df$CVX <- log(sorted.df$CVX[-1]/sorted.df$CVX[-dim(sorted.df)[1]]);
returns.df$DD <- log(sorted.df$DD[-1]/sorted.df$DD[-dim(sorted.df)[1]]);
returns.df$DOW <- log(sorted.df$DOW[-1]/sorted.df$DOW[-dim(sorted.df)[1]]);
returns.df$DPS <- log(sorted.df$DPS[-1]/sorted.df$DPS[-dim(sorted.df)[1]]);
returns.df$GS <- log(sorted.df$GS[-1]/sorted.df$GS[-dim(sorted.df)[1]]);
returns.df$HAL <- log(sorted.df$HAL[-1]/sorted.df$HAL[-dim(sorted.df)[1]]);
returns.df$HES <- log(sorted.df$HES[-1]/sorted.df$HES[-dim(sorted.df)[1]]);
returns.df$HON <- log(sorted.df$HON[-1]/sorted.df$HON[-dim(sorted.df)[1]]);
```

```
returns.df$HUN <- log(sorted.df$HUN[-1]/sorted.df$HUN[-dim(sorted.df)[1]]);
returns.df$JPM <- log(sorted.df$JPM[-1]/sorted.df$JPM[-dim(sorted.df)[1]]);
returns.df$KO <- log(sorted.df$KO[-1]/sorted.df$KO[-dim(sorted.df)[1]]);
returns.df$MMM <- log(sorted.df$MMM[-1]/sorted.df$MMM[-dim(sorted.df)[1]]);
returns.df$MPC <- log(sorted.df$MPC[-1]/sorted.df$MPC[-dim(sorted.df)[1]]);
returns.df$PEP <- log(sorted.df$PEP[-1]/sorted.df$PEP[-dim(sorted.df)[1]]);
returns.df$SLB <- log(sorted.df$SLB[-1]/sorted.df$SLB[-dim(sorted.df)[1]]);
returns.df$WFC <- log(sorted.df$WFC[-1]/sorted.df$WFC[-dim(sorted.df)[1]]);
returns.df$XOM <- log(sorted.df$XOM[-1]/sorted.df$XOM[-dim(sorted.df)[1]]);
returns.df$VV <- log(sorted.df$VV[-1]/sorted.df$VV[-dim(sorted.df)[1]]);

## (2)

# Compute correlation matrix for returns;
returns.cor <- cor(returns.df)
returns.cor[,c('VV')]
# Barplot the last column to visualize magnitude of correlations;
barplot(returns.cor[1:20,c('VV')],las=2,ylim=c(0,1.0), col = 'darkred')
title('Correlations with VV')

## (3)
# Make correlation plot for returns;
# If you need to install corrplot package; Note how many dependencies this package has;

corrplot(returns.cor)

## (4)
# load car package
```

```
# Fit some model
model.1 <- lm(VV ~ GS+DD+DOW+HON+HUN+JPM+KO+MMM+XOM, data=returns.df)
summary(model.1)
vif(model.1)
# Fit the full model
model.2 <- lm(VV ~

AA+BAC+GS+JPM+WFC+BHI+CVX+DD+DOW+DPS+HAL+HES+HON+HUN+KO+MMM+MPC+PEP+SLB+XOM,
        data=returns.df)
summary(model.2)
vif(model.2)


## (5)
returns.pca <- princomp(x=returns.df[,-21],cor=TRUE)
# See the output components returned by princomp();
names(returns.pca)
pc.1 <- returns.pca$loadings[,1];
pc.2 <- returns.pca$loadings[,2];
names(pc.1)
plot(-10,10,type='p',xlim=c(-0.27,-0.12),ylim=c(-0.27,0.6),xlab='PC 1',ylab='PC 2')
text(pc.1,pc.2,labels=names(pc.1),cex=0.75)


## (6)
# Plot the default scree plot;
plot(returns.pca, main = 'PCAs of Returns', col = 'darkblue')


# Make Scree Plot
```

```
scree.values <- (returns.pca$sdev^2)/sum(returns.pca$sdev^2);
plot(scree.values,xlab='Number of Components',ylab='',type='l',lwd=2)
points(scree.values,lwd=2,cex=1.5)
title('Scree Plot')

# Make Proportion of Variance Explained
variance.values <- cumsum(returns.pca$sdev^2)/sum(returns.pca$sdev^2);
plot(variance.values,xlab='Number of Components',ylab='',type='l',lwd=2)
points(variance.values,lwd=2,cex=1.5)
abline(h=0.8,lwd=1.5,col='red')
abline(v=8,lwd=1.5,col='red')
text(13,0.5,'Keep 8 Principal Components',col='red')
title('Total Variance Explained Plot')

## (7)
# Create the data frame of PCA predictor variables;
return.scores <- as.data.frame(returns.pca$scores);
return.scores$VV <- returns.df$VV;
return.scores$u <- runif(n=dim(return.scores)[1],min=0,max=1);
head(return.scores)
# Split the data set into train and test data sets;
train.scores <- subset(return.scores,u<0.70);
test.scores <- subset(return.scores,u>=0.70);
dim(train.scores)
dim(test.scores)
dim(train.scores)+dim(test.scores)
dim(return.scores)
# Fit a linear regression model using the first 8 principal components;
```

```
pca1.lm <- lm(VV ~ Comp.1+Comp.2+Comp.3+Comp.4+Comp.5+Comp.6+Comp.7+Comp.8,
          data=train.scores);
summary(pca1.lm)
# Compute the Mean Absolute Error on the training sample;
pca1.mae.train <- mean(abs(train.scores$VV-pca1.lm$fitted.values));
vif(pca1.lm)
# Score the model out-of-sample and compute MAE;
pca1.test <- predict(pca1.lm,newdata=test.scores);
pca1.mae.test <- mean(abs(test.scores$VV-pca1.test));

## (8)
# Let's compare the PCA regression model with a 'raw' regression model;
# Create a train/test split of the returns data set to match the scores data set;
returns.df$u <- return.scores$u;
train.returns <- subset(returns.df,u<0.70);
test.returns <- subset(returns.df,u>=0.70);
dim(train.returns)
dim(test.returns)
dim(train.returns)+dim(test.returns)
dim(returns.df)
# Fit model.1 on train data set and score on test data;
model.1 <- lm(VV ~ GS+DD+DOW+HON+HUN+JPM+KO+MMM+XOM, data=train.returns)
model1.mae.train <- mean(abs(train.returns$VV-model.1$fitted.values));
model1.test <- predict(model.1,newdata=test.returns);
model1.mae.test <- mean(abs(test.returns$VV-model1.test));
# Fit model.1 on train data set and score on test data;
model.2 <- lm(VV ~
```

```
BAC+GS+JPM+WFC+BHI+CVX+DD+DOW+DPS+HAL+HES+HON+HUN+KO+MMM+M
PC+PEP+SLB+XOM,
        data=train.returns)
model2.mae.train <- mean(abs(train.returns$VV-model.2$fitted.values));
model2.test <- predict(model.2,newdata=test.returns);
model2.mae.test <- mean(abs(test.returns$VV-model2.test));


## (9)
full.lm <- lm(VV ~ ., data=train.scores);
summary(full.lm)


backward.lm <- stepAIC(full.lm,direction=c('backward'))
summary(backward.lm)
backward.mae.train <- mean(abs(train.scores$VV-backward.lm$fitted.values));
vif(backward.lm)
backward.test <- predict(backward.lm,newdata=test.scores);
backward.mae.test <- mean(abs(test.scores$VV-backward.test));
```