ASSIGNMENT 6

Predict 410 Fall 2017

Zeeshan Latifi

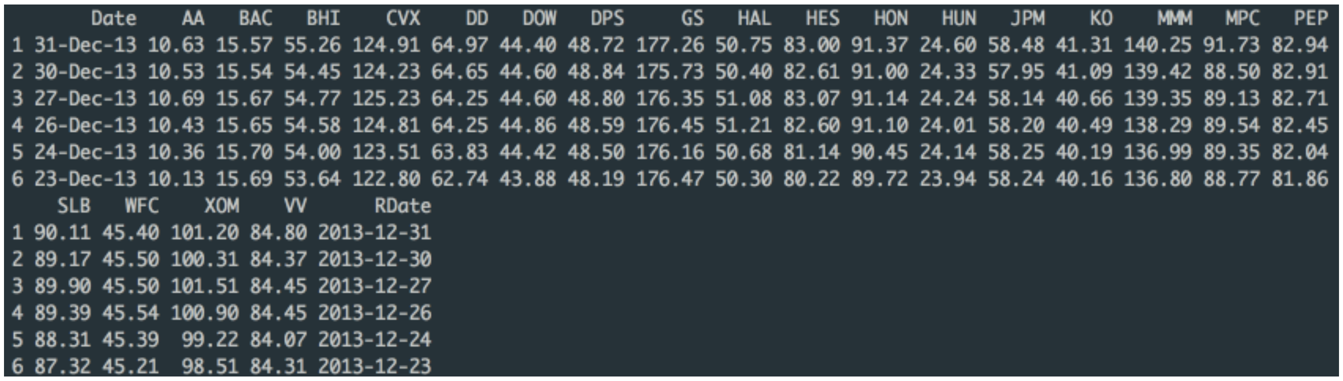
Northwestern University

Introduction:

In this analysis we will work through a data set that consists of daily closing stock prices for twenty stocks and a large-cap index fund from Vanguard (VV). We will use the log-returns of the individual stocks to explain the variation in the log-returns of the market index. We will explore this concept using both linear regression and principal components analysis.

Section 1 – Data Prep:

We begin with formatting a data field in R and typecasting our data into a data frame. Below are the first five observations for each variable in our data set.

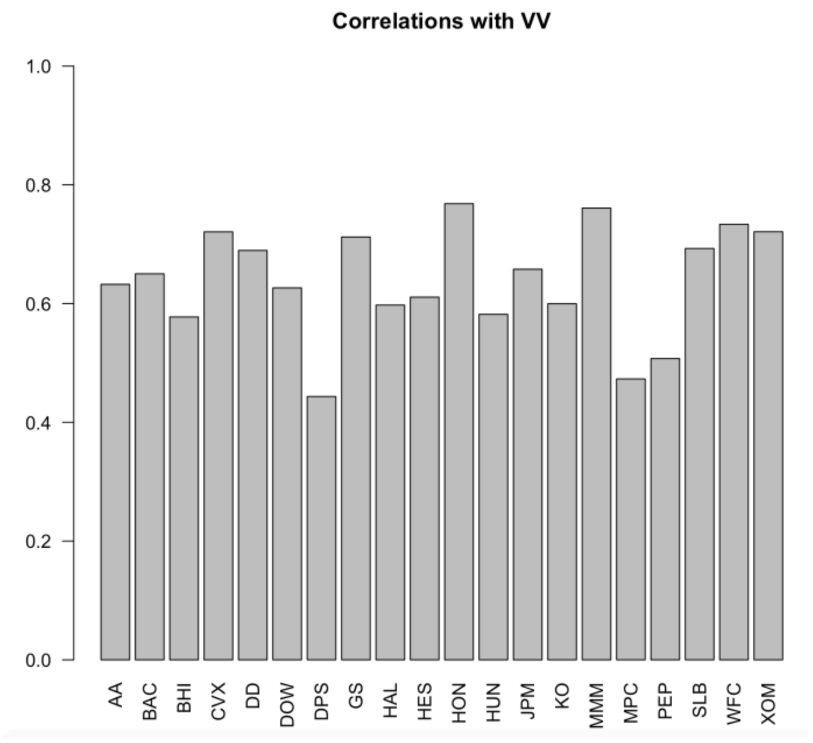


```
       Date    AA   BAC   BHI    CVX    DD   DOW   DPS     GS   HAL   HES   HON   HUN   JPM    KO    MMM   MPC   PEP
1 31-Dec-13 10.63 15.57 55.26 124.91 64.97 44.40 48.72 177.26 50.75 83.00 91.37 24.60 58.48 41.31 140.25 91.73 82.94
2 30-Dec-13 10.53 15.54 54.45 124.23 64.65 44.60 48.84 175.73 50.40 82.61 91.00 24.33 57.95 41.09 139.42 88.50 82.91
3 27-Dec-13 10.69 15.67 54.77 125.23 64.25 44.60 48.80 176.35 51.08 83.07 91.14 24.24 58.14 40.66 139.35 89.13 82.71
4 26-Dec-13 10.43 15.65 54.58 124.81 64.25 44.86 48.59 176.45 51.21 82.60 91.10 24.01 58.20 40.49 138.29 89.54 82.45
5 24-Dec-13 10.36 15.70 54.00 123.51 63.83 44.42 48.50 176.16 50.68 81.14 90.45 24.14 58.25 40.19 136.99 89.35 82.04
6 23-Dec-13 10.13 15.69 53.64 122.80 62.74 43.88 48.19 176.47 50.30 80.22 89.72 23.94 58.24 40.16 136.80 88.77 81.86
    SLB   WFC    XOM    VV      RDate
1 90.11 45.40 101.20 84.80 2013-12-31
2 89.17 45.50 100.31 84.37 2013-12-30
3 89.90 45.50 101.51 84.45 2013-12-27
4 89.39 45.54 100.90 84.45 2013-12-26
5 88.31 45.39  99.22 84.07 2013-12-24
6 87.32 45.21  98.51 84.31 2013-12-23
```

Section 2 – Exploratory Data Analysis:

After our data prep, we will now evaluate the correlations with our large-cap index fund from Vanguard (VV). From the figure below we can see the largest correlation is with the stock 'HON'. When compared to HON, it seems the stock MMM has very similar correlation. CVX and WFC and XOM also look to be very similar. We are most interested in evaluating these further.

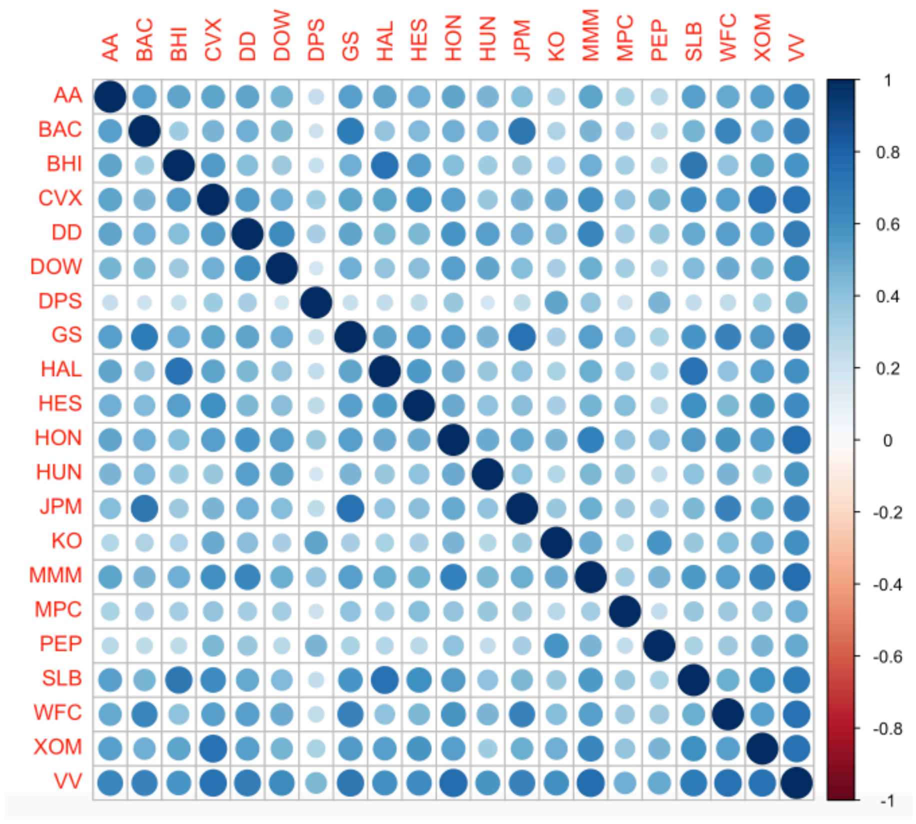
Figure 1:



Section 3 – Pairwise Correlations:

We now evaluate the pairwise correlations of each variable in our data set. Figure 2 below is a graphical representation of the correlations in R. The larger and darker the circle, the greater correlation is. As you can see, we don't have an exact number in regards how much the data is correlated; this is a relative visualization. This is the difference between statistical graphing and data visualization. Data visualization is used to tell a story about the data and may not always provide the actual information but helps the consumer understand what the data is representing. Statistical graphing informs the consumer of the actual numerical values and the underlying numbers. The user can then infer the information being represented. Not all visualizations will work when using data visualizations. However, you can replace the underlying data on a statistical graphic and still infer information from the result.

Figure 2:



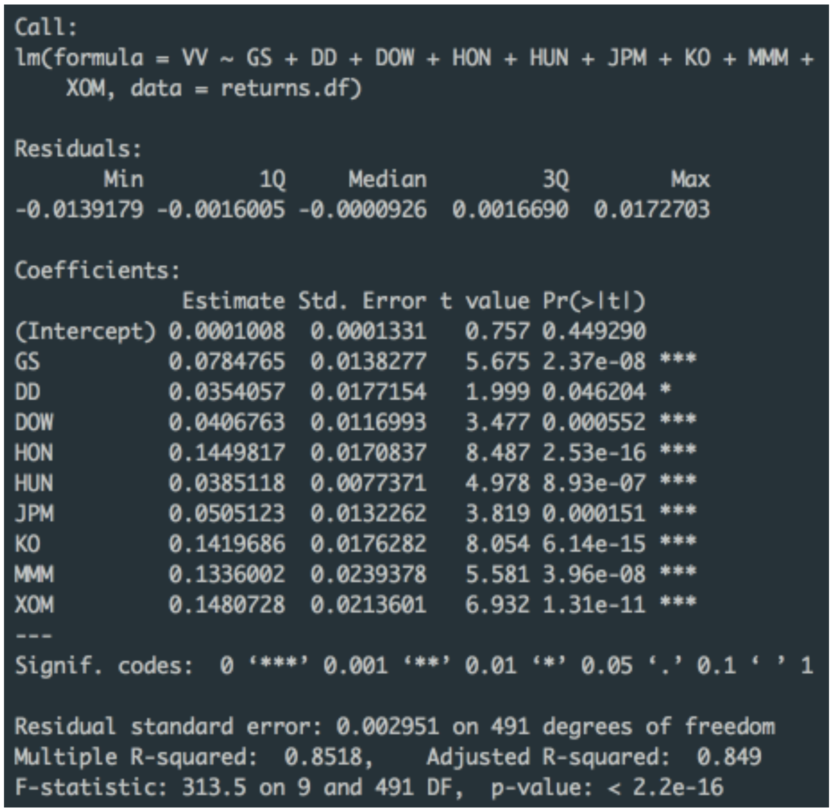
With respect to multicollinearity we can infer that DPS, MPC, and PEP should have the lowest VIF values with respect to VV. Alternatively, GS, HON, and MMM should have the highest VIF values.

Section 4 – VIF Values Naïve Models:

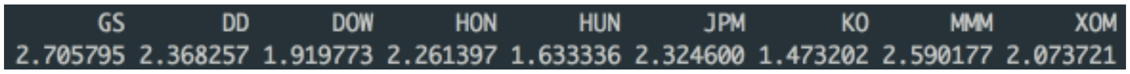
We will now create two naïve models; a partial model and a full model. The small model is what we believe to be a good starting point, however we'll also calculate the full model to gather VIF values for all variables. Figures 3 and 4 below outline the small model and full models with their respective VIF values.

Figure 3:

Small Model



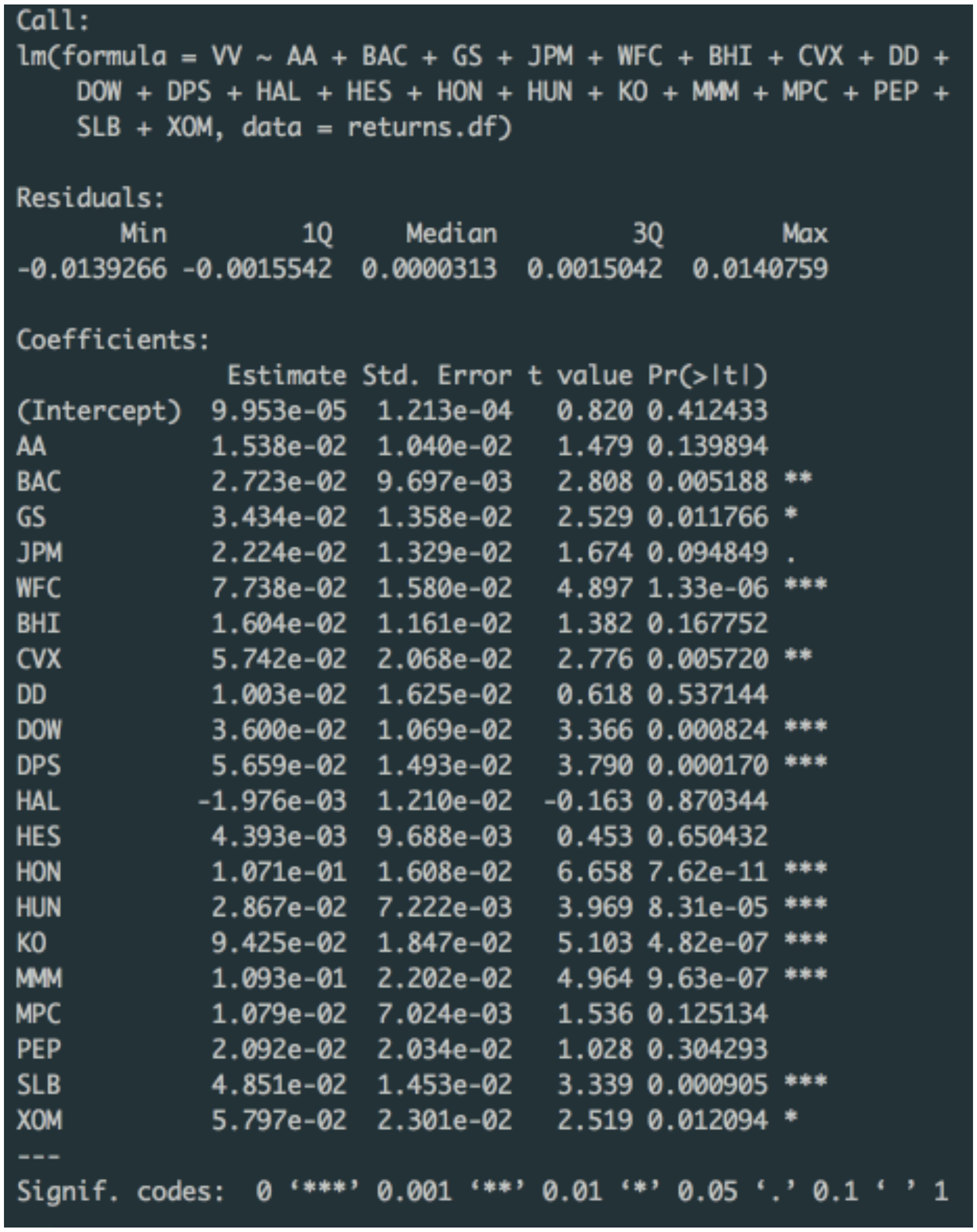
```
Call:
lm(formula = VV ~ GS + DD + DOW + HON + HUN + JPM + KO + MMM +
    XOM, data = returns.df)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0139179 -0.0016005 -0.0000926  0.0016690  0.0172703

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.0001008  0.0001331   0.757 0.449290
GS          0.0784765  0.0138277   5.675 2.37e-08 ***
DD          0.0354057  0.0177154   1.999 0.046204 *
DOW         0.0406763  0.0116993   3.477 0.000552 ***
HON         0.1449817  0.0170837   8.487 2.53e-16 ***
HUN         0.0385118  0.0077371   4.978 8.93e-07 ***
JPM         0.0505123  0.0132262   3.819 0.000151 ***
KO          0.1419686  0.0176282   8.054 6.14e-15 ***
MMM         0.1336002  0.0239378   5.581 3.96e-08 ***
XOM         0.1480728  0.0213601   6.932 1.31e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.002951 on 491 degrees of freedom
Multiple R-squared:  0.8518,    Adjusted R-squared:  0.849
F-statistic: 313.5 on 9 and 491 DF,  p-value: < 2.2e-16
```



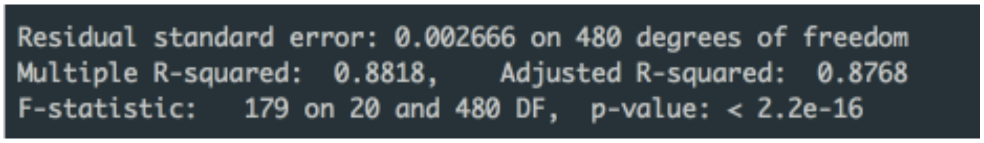
```
      GS       DD      DOW      HON      HUN      JPM       KO      MMM      XOM
2.705795 2.368257 1.919773 2.261397 1.633336 2.324600 1.473202 2.590177 2.073721
```

Figure 4:

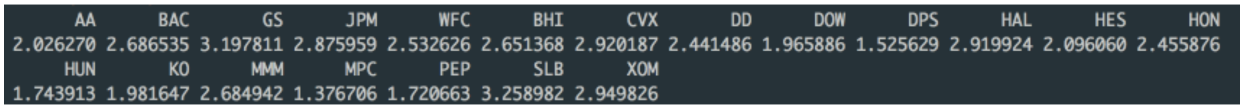
Full Model



```
Call:
lm(formula = VV ~ AA + BAC + GS + JPM + WFC + BHI + CVX + DD +
    DOW + DPS + HAL + HES + HON + HUN + KO + MMM + MPC + PEP +
    SLB + XOM, data = returns.df)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0139266 -0.0015542  0.0000313  0.0015042  0.0140759

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.953e-05  1.213e-04   0.820 0.412433
AA           1.538e-02  1.040e-02   1.479 0.139894
BAC          2.723e-02  9.697e-03   2.808 0.005188 **
GS           3.434e-02  1.358e-02   2.529 0.011766 *
JPM          2.224e-02  1.329e-02   1.674 0.094849 .
WFC          7.738e-02  1.580e-02   4.897 1.33e-06 ***
BHI          1.604e-02  1.161e-02   1.382 0.167752
CVX          5.742e-02  2.068e-02   2.776 0.005720 **
DD           1.003e-02  1.625e-02   0.618 0.537144
DOW          3.600e-02  1.069e-02   3.366 0.000824 ***
DPS          5.659e-02  1.493e-02   3.790 0.000170 ***
HAL         -1.976e-03  1.210e-02  -0.163 0.870344
HES          4.393e-03  9.688e-03   0.453 0.650432
HON          1.071e-01  1.608e-02   6.658 7.62e-11 ***
HUN          2.867e-02  7.222e-03   3.969 8.31e-05 ***
KO           9.425e-02  1.847e-02   5.103 4.82e-07 ***
MMM          1.093e-01  2.202e-02   4.964 9.63e-07 ***
MPC          1.079e-02  7.024e-03   1.536 0.125134
PEP          2.092e-02  2.034e-02   1.028 0.304293
SLB          4.851e-02  1.453e-02   3.339 0.000905 ***
XOM          5.797e-02  2.301e-02   2.519 0.012094 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



```
Residual standard error: 0.002666 on 480 degrees of freedom
Multiple R-squared:  0.8818,    Adjusted R-squared:  0.8768
F-statistic:   179 on 20 and 480 DF,  p-value: < 2.2e-16
```



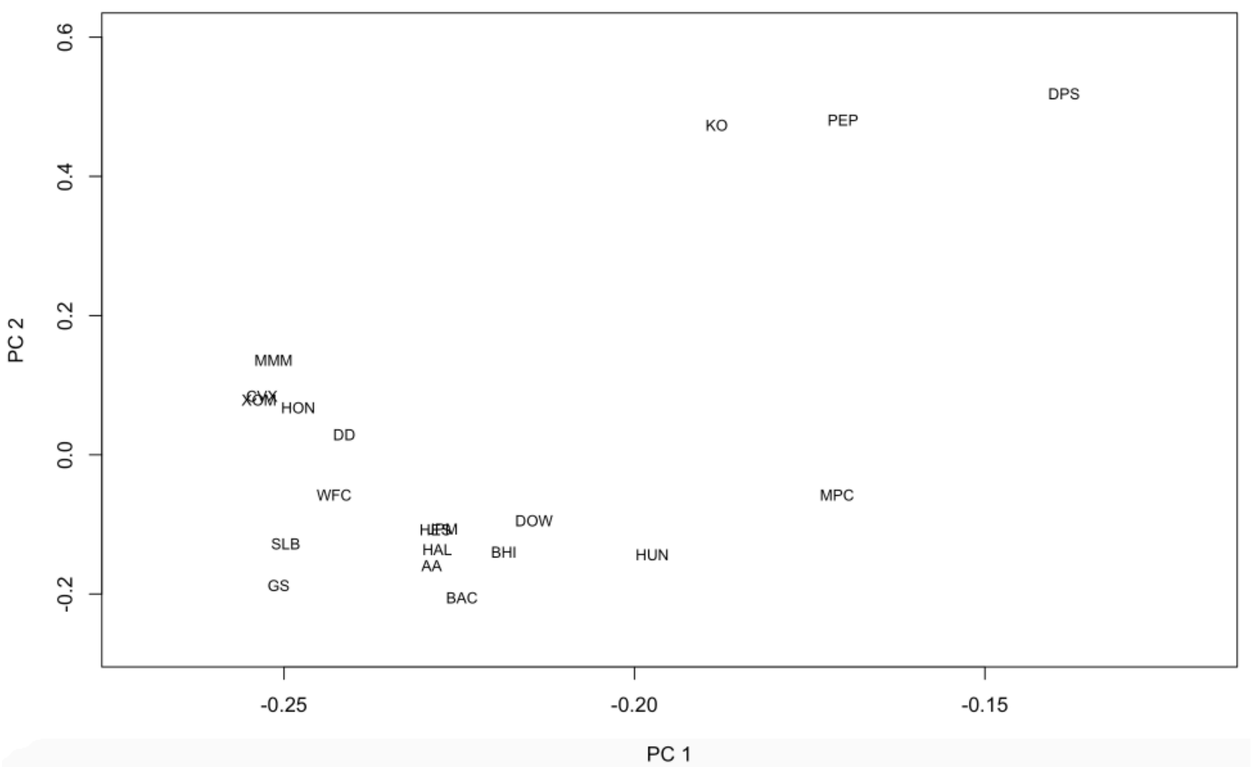
```
      AA      BAC       GS      JPM      WFC      BHI      CVX       DD      DOW      DPS      HAL      HES      HON
2.026270 2.686535 3.197811 2.875959 2.532626 2.651368 2.920187 2.441486 1.965886 1.525629 2.919924 2.096060 2.455876
     HUN       KO      MMM      MPC      PEP      SLB      XOM
1.743913 1.981647 2.684942 1.376706 1.720663 3.258982 2.949826
```

Based on our calculations, some variables should be of concern with regards to multicollinearity. We will be taking a more conservative approach and anything over a VIF value of 2.5 should be removed from our assessment. As mentioned above, GS and CVX seem to be the two largest values of VIF.

Section 5 – Principal Component Analysis:

In order to remedy multicollinearity, we can utilize principal components. We'll compute the principal components for each of the predictor variables. We will not be standardizing the data, as we already used the log-return transformation for each security; this will be our standardization. Figure 5 below illustrates the loadings for the first two principal components. Here, loadings are the correlation coefficients between the variables and factors.

Figure 5:



Some surprises arise here. Although having very similar VIF values at 2.68, BAC and MMM are not clustered together. KO, PEP, and DPS are all off in one area since they did have the lowest correlation. There are also several clusters such as JPM, HAL, AA, and HES as well as CVX, HON and XOM.

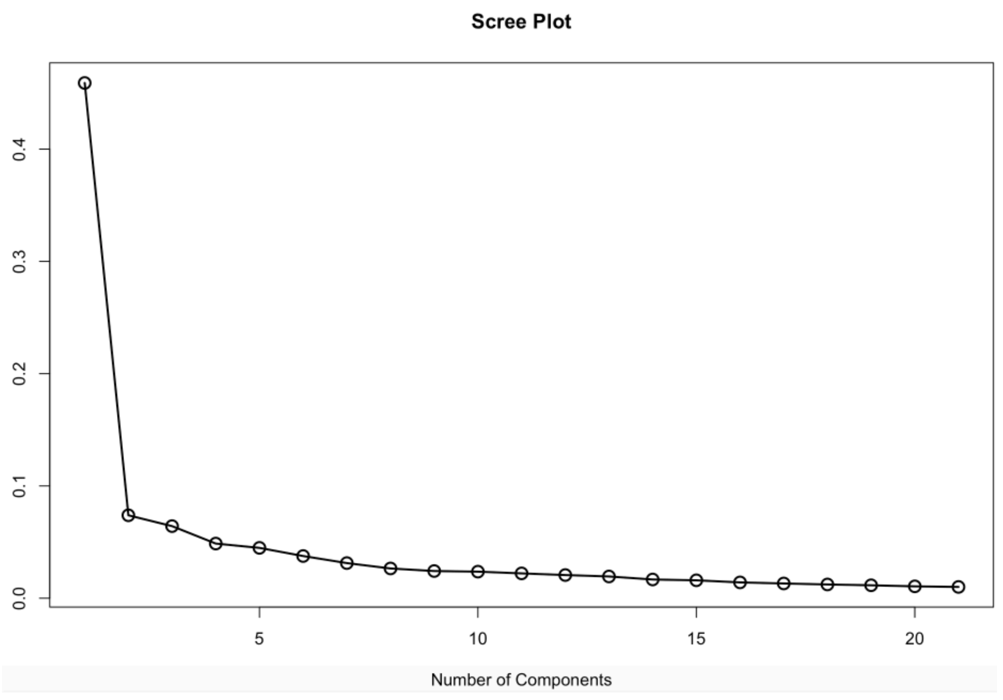
Section 6 – Dimension Reduction with PCA:

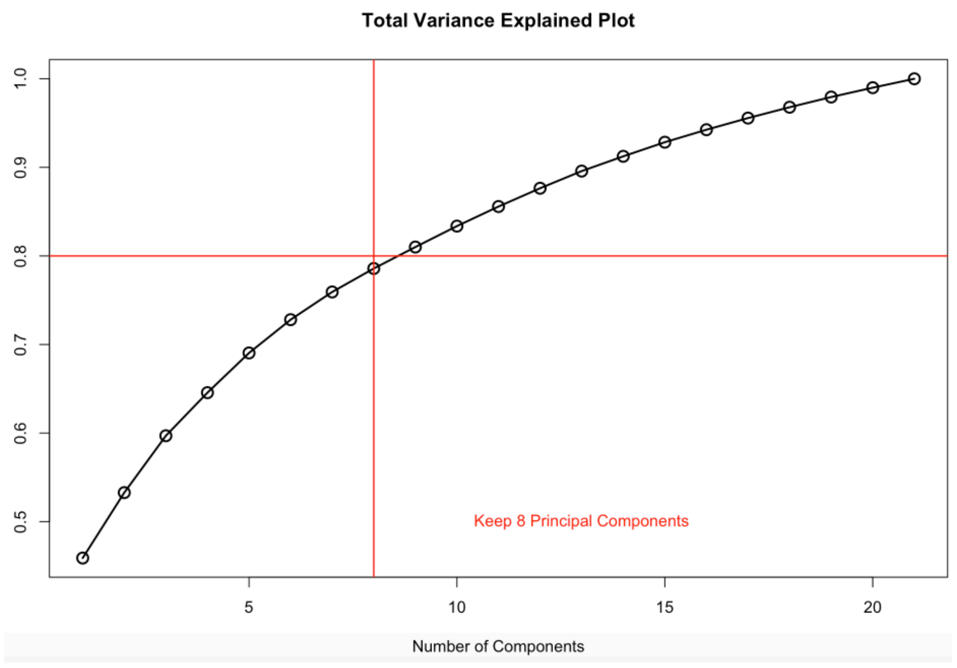
Dimension reduction is mainly used to reduce the redundancy in the data. There are multiple ways to determine how many principal components to keep. Ideally, we'd like to keep enough of the principal components so that it represents the majority of the sample. To do this, we use decision rules. Some are described below:

- Retain enough components to explain some large percentage of the total variation of the original variables, ~70%-90%
- Compute the eigenvalues and exclude the principal components whose eigenvalues are less than the average.
- Utilize a scree diagram, exclude the values when the graph begins to flatten out

To start, we should look to keep about 80% of the principal components. This way we'll have enough representation from the data. Figure 6 below outlines the scree plot, which can also help determine how many principal components to keep. The second graph in Figure 6 shows the proportions of the principal component values. From this, we can see that in order to keep 80% of the values, we must keep 8 variables.

Figure 6:

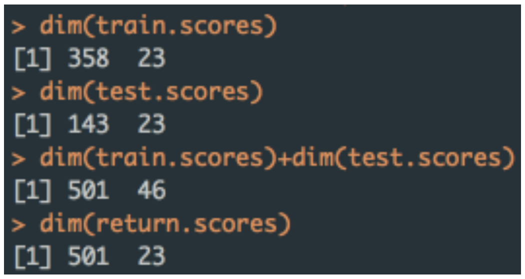




Section 7 – Principal Components in Predictive Modeling:

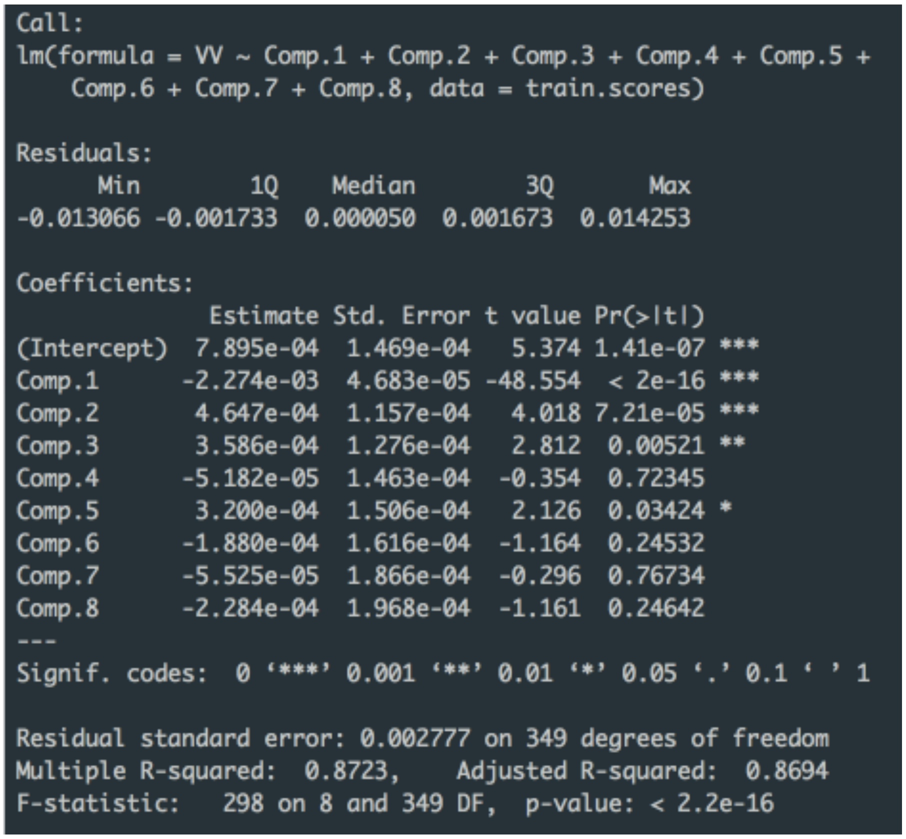
In order to build a predictive model, we'll need to use the PCA scores. These will be the predictor variables for our model. PCA scores are the scores of each row on each column. To compute the score, take the case's standardized score and multiply it by the corresponding factor loading for the given factor. The sum is the score. We will have a train data set and a test data set. The data will be split manually into train and test data sets by generating a uniform(0,1) random variable and attaching it to the data frame. We will compute these in R. Figure 7 is the breakdown of each data set as well. Figure 8 below is the linear regression model using the first eight principal components and computed (MAE) for that model. We also scored that model out-of-sample on our test data set and computed the out-of-sample MAE.

Figure 7:



```
> dim(train.scores)
[1] 358  23
> dim(test.scores)
[1] 143  23
> dim(train.scores)+dim(test.scores)
[1] 501  46
> dim(return.scores)
[1] 501  23
```

Figure 8:



```
Call:
lm(formula = VV ~ Comp.1 + Comp.2 + Comp.3 + Comp.4 + Comp.5 +
    Comp.6 + Comp.7 + Comp.8, data = train.scores)

Residuals:
      Min        1Q    Median        3Q       Max
-0.013066 -0.001733  0.000050  0.001673  0.014253

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.895e-04  1.469e-04   5.374 1.41e-07 ***
Comp.1      -2.274e-03  4.683e-05 -48.554  < 2e-16 ***
Comp.2       4.647e-04  1.157e-04   4.018 7.21e-05 ***
Comp.3       3.586e-04  1.276e-04   2.812  0.00521 **
Comp.4      -5.182e-05  1.463e-04  -0.354  0.72345
Comp.5       3.200e-04  1.506e-04   2.126  0.03424 *
Comp.6      -1.880e-04  1.616e-04  -1.164  0.24532
Comp.7      -5.525e-05  1.866e-04  -0.296  0.76734
Comp.8      -2.284e-04  1.968e-04  -1.161  0.24642
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.002777 on 349 degrees of freedom
Multiple R-squared:  0.8723,    Adjusted R-squared:  0.8694
F-statistic:   298 on 8 and 349 DF,  p-value: < 2.2e-16
```

In Sample MAE: 0.002035149

Out of Sample MAE: 0.002058605

Notice that our MAE’s are very similar indicating a good predictor model. The VIF values associated with every predictor variable in any principal components regression model should all be one. As was the case with our model. This is because no factors are correlated within our model.

Section 8 – Predictive Accuracy:

We’ll now compare the model above with a raw regression model. We’ll be using the returns data to match the scores data set. Model 1 will be the partial model and Model 2 will be the full model. Figure 9 below compares the MAE values from Figure 8 above with the newly conducted models.

Figure 9:

	PCA 1	Model 1	Model 2
MAE Train	0.002035149	0.002181089	0.001961861
MAE Test	0.002058605	0.002155727	0.001992463

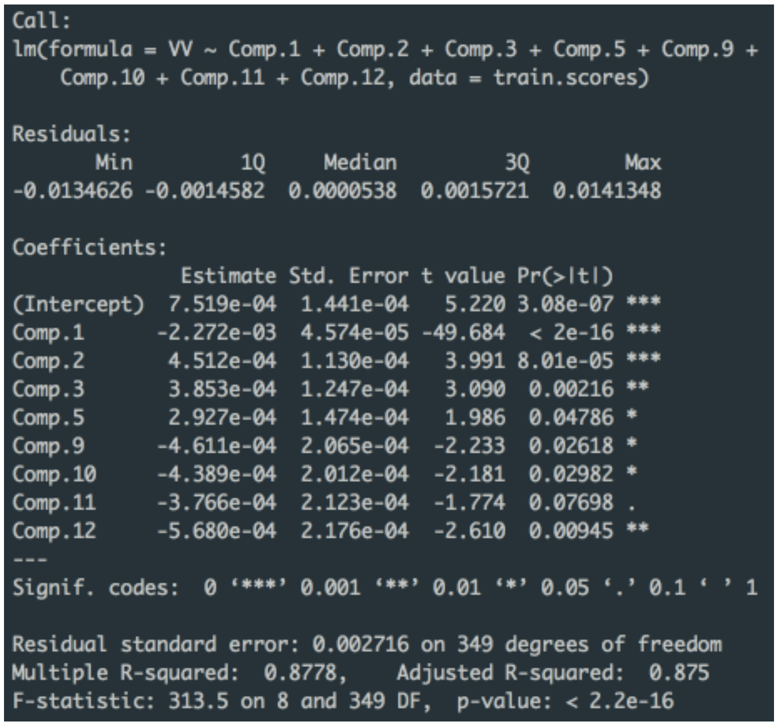
According to the results above, Model 2 seems to be the best fit. It has the lowest MAE value. However, overall Model 1 would work best. This is due to the multicollinearity that exists within Model 2.

Section 9 – PCA Supervised Learning:

So far we've taken a look at PCA from an unsupervised method. Now, we will attempt a supervised method to better assist with predictive modeling. We will use backwards elimination for variable selection and determine the number of principal components based on the selection.

After our evaluation the model selected 8 principal components as well. The components however were different than our original model, and were specified. Figure 10 below shows summary of the model and the components with their VIF values.

Figure 10:



```
Call:
lm(formula = VV ~ Comp.1 + Comp.2 + Comp.3 + Comp.5 + Comp.9 +
    Comp.10 + Comp.11 + Comp.12, data = train.scores)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0134626 -0.0014582  0.0000538  0.0015721  0.0141348

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.519e-04  1.441e-04   5.220 3.08e-07 ***
Comp.1      -2.272e-03  4.574e-05 -49.684  < 2e-16 ***
Comp.2       4.512e-04  1.130e-04   3.991 8.01e-05 ***
Comp.3       3.853e-04  1.247e-04   3.090  0.00216 **
Comp.5       2.927e-04  1.474e-04   1.986  0.04786 *
Comp.9      -4.611e-04  2.065e-04  -2.233  0.02618 *
Comp.10     -4.389e-04  2.012e-04  -2.181  0.02982 *
Comp.11     -3.766e-04  2.123e-04  -1.774  0.07698 .
Comp.12     -5.680e-04  2.176e-04  -2.610  0.00945 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002716 on 349 degrees of freedom
Multiple R-squared:  0.8778,    Adjusted R-squared:  0.875
F-statistic: 313.5 on 8 and 349 DF,  p-value: < 2.2e-16
```

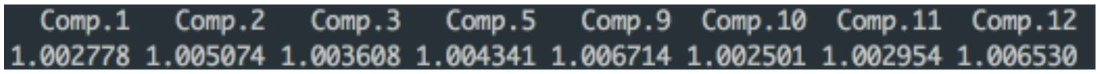

```
  Comp.1   Comp.2   Comp.3   Comp.5   Comp.9  Comp.10  Comp.11  Comp.12 
1.002778 1.005074 1.003608 1.004341 1.006714 1.002501 1.002954 1.006530 
```

Figure 11:

	PCA 1	**Model 1**	**Model 2**	**Full Model (Backward)**
MAE Train	0.002035149	0.002181089	0.001961861	0.001982944
MAE Test	0.002058605	0.002155727	0.001992463	0.001969093

Figure 11 above, outlines the breakdown of MAE's for each model we have evaluated. According to the MAE's the best model fit is the backward elimination variable selection model. It has the lowest MAE for our test data set, indicating a better predictor model when compared to the rest. This is because of the variable selection; we've specifically chosen variables that will best help predict out-of-sample observations for our large-cap index.

Code Appendix:

```
#Assignment 6
#Zeeshan Latifi
#Predict 410 Fall 2017


#Section 1 ----------------------------------------------------------------------------------------------------
my.path <- '/Users/Zeeshan/Desktop/PREDICT 410/Week 6/';
my.data <- read.csv(paste(my.path,'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]])
```

```
#Section 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))
title('Correlations with VV')
```

```
#Section 3 ------------------------------------------------------------------------------------------------------
# Make correlation plot for returns;
# If you need to install corrplot package; Note how many dependencies this package has;
install.packages('corrplot', dependencies=TRUE)
library(corrplot)
corrplot(returns.cor)
```

```
#Section 4 ------------------------------------------------------------------------------------------------------
# load car package
```

```
library(car)
# 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)




#Section 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)




#Section 6 ----------------------------------------------------------------------------------------------------------
# Plot the default scree plot;
plot(returns.pca)

# 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')
```

```
#Section 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))




#Section 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+MPC+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))




#Section 9 ------------------------------------------------------------------------------------------------------------
```

```
full.lm <- lm(VV ~ ., data=train.scores);
summary(full.lm)
library(MASS)
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))
```