Oscar Hernandez
MSDS410
Summer 2018
Assignment 6

Contents

Introduction

The purpose of this report is to provide results from the steps involved in fitting multiple linear regression models that use the Stock Portfolio data. The Stock Portfolio raw data contains 502 observations spread across 22 variables which are mainly all numerical variables except for one factor variable. The raw data contains the daily closing prices of twenty stocks and a large-cap index fund from Vanguard (VV). The report will summarize key aspects of the model building process. Specifically, the report will cover how we initially manipulated the raw data, results from exploratory data analysis, results from principal component analysis, comparison of different fitted models and results from using automated variable selection. The final goal is to determine which model is most appropriate at predicting the returns of the large-cap index fund from Vanguard (VV).

Data Description

As noted, the Stock Portfolio raw data contains the daily closing prices. We are concerned with the relationship of the twenty stocks' returns and the large-cap index fund returns. Therefore, we will manipulate the raw data to calculate daily return percentages. If t = Day 1, then a daily return percentage is calculated by t+1/t. A dataframe called "returns.df" was created based on this logic. Furthermore, since we are working with asset prices and returns, it is best to apply a log transformation to all our calculations. The Code section of this report has the R script that outlines how the returns.df was created in more detail. Figure 1 is a table that displays partial output from returns.df.

AA	BAC	BHI	CVX	DD	DOW
0.023556	0.001723	0.009946	-0.00172	0.010906	0.005357
-0.00957	0.082555	-0.01387	-0.00985	-0.00683	0.006324
-0.0216	-0.02082	0.008621	-0.00727	-0.01423	0.005954
0.027989	0.014458	0.006223	0.010836	0.008435	-0.00033
0.002121	0.055828	0.007148	-0.00394	0.015176	0.021864
0.019927	0.035559	-0.03407	-0.0119	0.003388	0.014421

Figure 1: returns.df (partial output)

Initial Exploratory Data Analysis

Statistical Graphics

Our next sub-section will focus on using statistical graphics to conduct initial exploratory data analysis. Specifically, we are concerned with the correlation of each of the twenty stocks returns to the index fund returns. Figure 2 is a bar chart that easily allows for comparison across the twenty stocks.

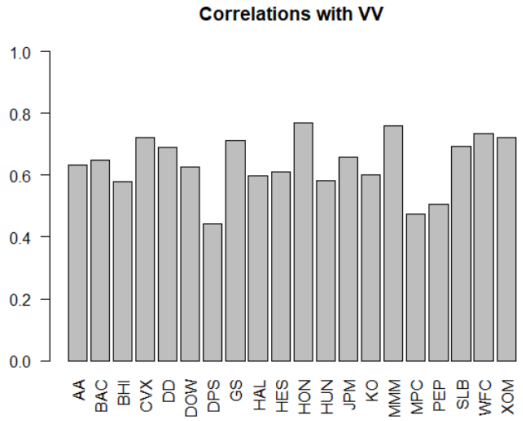

Figure 2: VV Correlation Bar Chart

The Code section of this report outlines the R script that was used to create the correlation matrix. The correlation matrix was then used to create Figure 2. With regards to the correlation matrix, we are most concerned with the VV column which shows the correlation it has with each of the twenty stocks. From Figure 2, we see that HON and MMM appear to be most correlated with VV. On the contrary, DSP and MPC appear to have the least correlation with VV.

Data Visualization

Figure 3 allows us to display all the pairwise correlations in our data using data visualization.

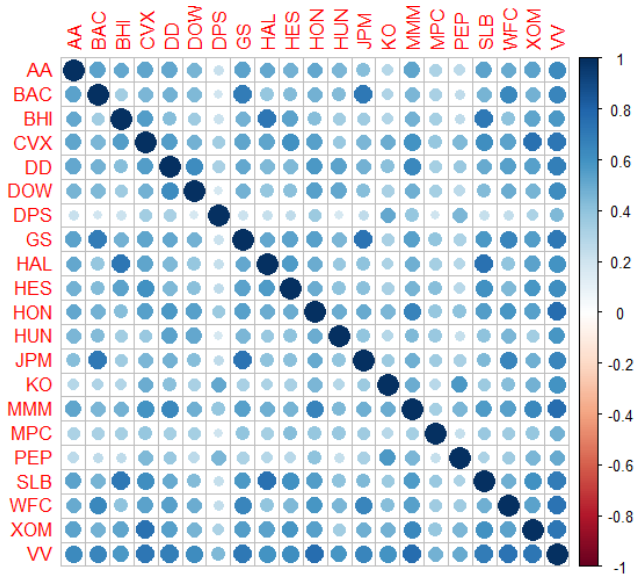

Figure 3: Pairwise Correlation Visual

From Figure 3, we can see which stocks are correlated with each other and each of their correlations with VV. This is an added benefit from Figure 3 that Figure 2 lacked. Figure 3 is a lot more useful than Figure 2 because we can get an initial sense of possible multicollinearity within our data. With regards to initial exploratory data analysis, Figure 3 is a lot more insightful than Figure 2. In general, the difference between a "statistical graph" and a "data visualization" is that a data visualization is designed to provide more informative than a statistical graph because of its enhanced pictorial appeal.

Using Figure 3, we can also begin to understand the extent to which multicollinearity exists within our data. Based on the number of light circles (using the color scale on the right) across the rows, we can state that DPS, KO and MPC should have low VIFs. On the contrary, we can expect CVX, XOM and SLB to have high VIFs.

Naïve Models

Another important tool to consider in exploratory data analysis is the use of models. Our small model will include nine stocks and the full model will include all twenty, which were fitted using returns.df. The mathematical equations for each model are as follows:

Small Model: $VV = \beta_0 + \beta_1 GS + \beta_2 DD + \beta_3 DOW + \beta_4 HON + \beta_5 HUN + \beta_6 JPM + \beta_7 KO + \beta_8 MMM + B_9\, XOM$

Full Model: $VV = \beta_0 + \beta_1 AA + \beta_2 BAC + \beta_3 GS + \beta_4 JPM + \beta_5 WFC + \beta_6 BHI + \beta_7 CVX + \beta_8 DD + B_9 DOW + \beta_{10} DPS + \beta_{11} HAL + \beta_{12} HES + \beta_{13} HON + \beta_{14} HUN + \beta_{15} KO + \beta_{16} MMM + \beta_{17} MPC + \beta_{18} PEP + \beta_{19}\, SLB + \beta_{20} XOM$

Both models contain a mix of statistically significant variables and both are overall statistically significant as highlighted by the summary outputs in Figure 4 and 5, respectively.

Figure 6 and 7 are VIF tables for each of our models. Based on Montgomery and Peck's *Introduction to Linear Regression Analysis*, we should be concerned when VIFs exceed 5 or 10. Others have said that we should be concerned with VIF values that are 20 or larger. Using either decision rule, these models do not appear to be suffering from multicollinearity. However, that doesn't mean multicollinearity does not exist as it also suggested that weak models with VIF values above 2.5 may be cause for concern.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.00	0.00	0.76	0.45
GS	0.08	0.01	5.68	0.00
DD	0.04	0.02	2.00	0.05
DOW	0.04	0.01	3.48	0.00
HON	0.14	0.02	8.49	0.00
HUN	0.04	0.01	4.98	0.00
JPM	0.05	0.01	3.82	0.00
KO	0.14	0.02	8.05	0.00
MMM	0.13	0.02	5.58	0.00
XOM	0.15	0.02	6.93	0.00
Model F-statistic: 313.5 (**p-value**: < 2.2e-16) **R-squared**: 0.8518**; Adjusted R-squared**: 0.849				

Figure 4: Small Model Output

	Estimate	Std.Error	t value	Pr(>|t|)
(Intercept)	0.00	0.00	0.82	0.41
AA	0.02	0.01	1.48	0.14
BAC	0.03	0.01	2.81	0.01
GS	0.03	0.01	2.53	0.01
JPM	0.02	0.01	1.67	0.09
WFC	0.08	0.02	4.90	0.00
BHI	0.02	0.01	1.38	0.17
CVX	0.06	0.02	2.78	0.01
DD	0.01	0.02	0.62	0.54
DOW	0.04	0.01	3.37	0.00
DPS	0.06	0.01	3.79	0.00
HAL	0.00	0.01	-0.16	0.87
HES	0.00	0.01	0.45	0.65
HON	0.11	0.02	6.66	0.00
HUN	0.03	0.01	3.97	0.00
KO	0.09	0.02	5.10	0.00
MMM	0.11	0.02	4.96	0.00
MPC	0.01	0.01	1.54	0.13
PEP	0.02	0.02	1.03	0.30
SLB	0.05	0.01	3.34	0.00
XOM	0.06	0.02	2.52	0.01
Model F-statistic: 179 (**p-value**: < 2.2e-16) **R-squared**: 0.8818**; Adjusted R-squared**: 0.8768				

Figure 5: Full Model Output

GS	DD	DOW	HON	HUN
2.7058	2.368257	1.919773	2.261397	1.633336
JPM	**KO**	**MMM**	**XOM**	
2.3246	1.473202	2.590177	2.073721	

Figure 6: Small Model VIF Table

AA	BAC	GS	JPM	WFC
2.02627	2.686535	3.197811	2.875959	2.532626
BHI	**CVX**	**DD**	**DOW**	**DPS**
2.651368	2.920187	2.441486	1.965886	1.525629
HAL	**HES**	**HON**	**HUN**	**KO**
2.919924	2.09606	2.455876	1.743913	1.98165
MMM	**MPC**	**PEP**	**SLB**	**XOM**
2.684942	1.376706	1.720663	3.258982	2.949826

Figure 7: Full Model VIF Table

Principal Component Analysis

Dimension Reduction

For models that do suffer from multicollinearity, we can use principal component analysis as a remedy. The Code section of this report outlines the R script utilized to compute the principal components using the returns.df we created. Figure 8 is a plot of the loadings for the first two principal components from the principal components analysis. Loading plots are useful because they help identify which variables have the largest effect on each component. Loadings can be described as the product of eigenvectors and the square root of eigenvalues. Loadings range from -1 to 1 and loadings close to either end signify that the variable has a strong influence on the component.

From Figure 8, we see two clusters formed which are circled on the plot. Strongly correlated variables tend to have the same weight value when they have positive correlation. We can see that from the cluster that includes MMM. We can also see the KO, PEP and DPS have a strong, positive influence on the second principal component. There doesn't appear to be any surprises from this plot.

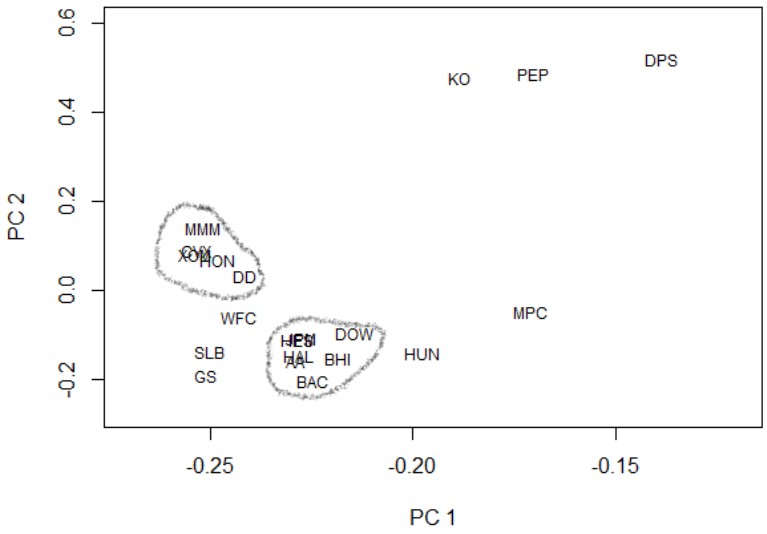

Figure 8: PC 1 vs. PC 2 Loadings Plot

As noted earlier, we can use principal component analysis for dimension reduction. Essentially, our goal is to determine how many principal components to keep. There doesn't exist a definitive answer that states how many principal components to keep. However, some popular decision rules are as follows:

- Use a minimum eigenvalue rule such as the Kaiser Rule, which recommends keeping the number of principal components which have an eigenvalue greater than one
- Keep at least as many principal components needed to explain at least 70% of the total variation in the data
- Exclude principal components extracted from a correlation matrix whose associated eigenvalues are less than 0.7

Figure 9 is a table output of the importance of components. The decision rule that will be used is to keep as many principal components that explain at least 70% of the total variation in the data. Therefore, we will keep five principal components since the cumulative proportion from five is at least 70%. The decision to keep 8 principal components would be based on the idea that we want our minimum threshold to be 80%. Interestingly enough, each additional principal components tends to add a decent amount of explanation even after the 8th principal component.

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**	**Comp.5**
Standard deviation	3.104	1.240	1.161	0.973	0.892
Proportion of Variance	0.482	0.077	0.067	0.047	0.040
Cumulative Proportion	0.482	0.559	0.626	0.674	0.713
	Comp.6	**Comp.7**	**Comp.8**	**Comp.9**	**Comp.10**
Standard deviation	0.816	0.747	0.716	0.705	0.681
Proportion of Variance	0.033	0.028	0.026	0.025	0.023
Cumulative Proportion	0.747	0.775	0.800	0.825	0.848
	Comp.11	**Comp.12**	**Comp.13**	**Comp.14**	**Comp.15**
Standard deviation	0.658	0.639	0.593	0.579	0.545
Proportion of Variance	0.022	0.020	0.018	0.017	0.015
Cumulative Proportion	0.870	0.890	0.908	0.925	0.939
	Comp.16	**Comp.17**	**Comp.18**	**Comp.19**	**Comp.20**
Standard deviation	0.526	0.509	0.492	0.471	0.460
Proportion of Variance	0.014	0.013	0.012	0.011	0.011
Cumulative Proportion	0.953	0.966	0.978	0.989	1.000

Figure 9: PCA Importance of Components

Furthermore, we can use different plots to determine how many principal components to keep. Figure 10 and Figure 11 are two common PCA plots that can help us make that determination. Using Figure 10, we would want to keep the number of principal components where we see the scree plot start to flatten out. We can see this trend start to occur between 5 and 10 principal components. A common issue with this type of plot is that it can be ambiguous. Figure 11 is a little more helpful in the sense that it simply depicts the data from Figure 9. If we were to use a threshold of 80%, then by using

this plot, we would keep 8 principal components. Overall, taking the two plots into consideration, we would want to keep 8 principal components.

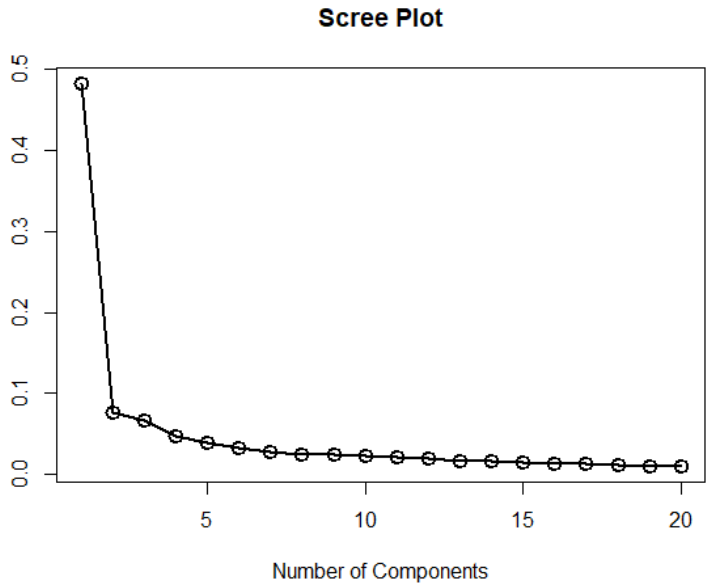

Figure 10: Scree Plot

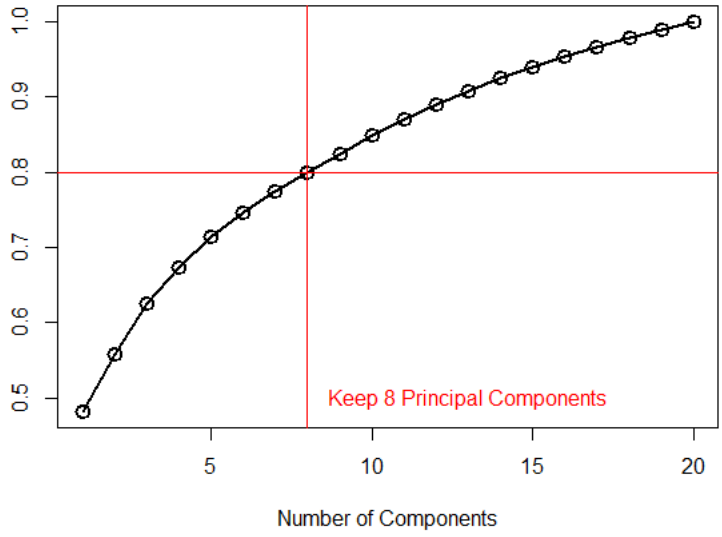

Figure 11: Proportion of Variance Explained

Predictive Modeling

Our next sub-section will be dedicated to using principal components in predictive modeling. We will use PCA scores as our predictor variables. Scores are the transformed variable values corresponding to a particular data point. We will build a "return.scores" dataframe and as always, prior to building our predictive model, we will split this dataframe into train/test data sets. The Code section of this report outlines the R script utilized to build our latest model. Our PCA Model 1 will include the first eight principal components. The mathematical equation for our model is as follows:

PCA Model 1: *$VV = \beta_0 + \beta_1 Comp.1 + \beta_2\ Comp.2 + \beta_3\ Comp.3 + \beta_4\ Comp.4 + \beta_5\ Comp.5 + \beta_6\ Comp.6 + \beta_7\ Comp.17 + \beta_8\ Comp.8$*

An output summary of our model is included in Figure 12. We can see that the model has a mix of statistically significant variables. Overall, the model is statistically significant as measured by the p-value from the F-statistic. Furthermore, it is necessary to compute the MAE for in the train and the test data to compare predictive accuracy. Figure 13 shows the results for both calculations. We can see that the model had higher predictive accuracy on the in-sample data since the MAE is lower than the out-of-sample MAE. Finally, we will display the VIF values for this model in Figure 14. Figure 14 shows that all the VIF values are one which make sense since we used principal components scores as predictor variables which is a remedy for multicollinearity.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0007	0.0001	4.92	0.00
Comp.1	-0.0023	0.0000	-51.74	< 2e-16
Comp.2	0.0005	0.0001	4.34	0.00
Comp.3	0.0006	0.0001	4.61	0.00
Comp.4	-0.0001	0.0002	-0.34	0.73
Comp.5	0.0001	0.0002	0.35	0.73
Comp.6	-0.0002	0.0002	-1.29	0.20
Comp.7	0.0000	0.0002	0.05	0.96
Comp.8	-0.0005	0.0002	-2.56	0.01
Model F-statistic: 344.3 (**p-value**: < 2.2e-16) **R-squared**: 0.8898**; Adjusted R-squared**: 0.8872				

Figure 12: PCA Model 1 Output

	MAE
In-Sample	0.00195
Out-of-Sample	0.00226

Figure 13: MAE Calculation Table – PCA Model 1

Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8
1.004179	1.006966	1.007726	1.010096	1.008543	1.004551	1.006747	1.002987

Figure 14: PCA Model 1 VIF Values

Model Comparison

It is too early to stated whether our PCA Model is a "better" model or the "best" model. Therefore, it is necessary to do some model comparison in this section of the report. Generally speaking and with regards to predictive accuracy, a hypothetical Model A is better Model B when it has a higher degree of predictive accuracy as measured by some standard measurement (e.g. MAE) and it doesn't egregiously violate some standard statistical assumptions (e.g. normality).

In this section, we will compare Model 1 and Model 2 from earlier in this report to our PCA Model 1. To do so, we need to create the matching train/test split of the raw log returns data. The Code section of this report outlines the R script utilized to complete this important step. Also, the mathematical equations for Model 1 and Model 2 have not changed—only the data used has been altered for comparison purposes with PCA Model 1. Figure 15 and Figure 16 are the summary outputs after fitting Model 1 and Model 2. From the outputs, we can see that models are statistically significant

based on the p-values associated with their respective F-statistic. Furthermore, both models have an adjusted R-squared over 86% which is relatively high.

Moreover, we are truly concerned with the predictive accuracy of each model as measured by MAE. Figure 17 is a summary table of the MAEs for the in-sample and out-of-sample data of each model. We can see that all models performed better on the in-sample data. Model 2 appears to be the best model in terms of predictive accuracy since it has the lowest MAE for the out-of-sample data. Model 1 can be considered the worst in comparison to the others.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.00	0.00	0.02	0.98
GS	0.08	0.02	4.88	0.00
DD	0.02	0.02	0.89	0.37
DOW	0.05	0.02	3.09	0.00
HON	0.14	0.02	7.47	0.00
HUN	0.03	0.01	3.75	0.00
JPM	0.08	0.02	4.84	0.00
KO	0.14	0.02	6.51	0.00
MMM	0.11	0.03	4.08	0.00
XOM	0.15	0.03	5.85	0.00
Model F-statistic: 253.1 (**p-value**: < 2.2e-16) **R-squared**: 0.8701**; Adjusted R-squared**: 0.8667				

Figure 15: Model 1 Output

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.00	0.00	0.08	0.94
BAC	0.03	0.01	2.80	0.01
GS	0.04	0.02	2.22	0.03
JPM	0.05	0.02	3.00	0.00
WFC	0.05	0.02	2.89	0.00
BHI	0.02	0.01	1.33	0.18
CVX	0.05	0.02	2.11	0.04
DD	0.00	0.02	0.06	0.95
DOW	0.03	0.01	2.46	0.01
DPS	0.07	0.02	3.79	0.00
HAL	0.01	0.01	0.50	0.61
HES	0.01	0.01	1.12	0.26
HON	0.11	0.02	6.17	0.00
HUN	0.03	0.01	3.29	0.00
KO	0.10	0.02	4.37	0.00
MMM	0.09	0.02	3.73	0.00
MPC	0.01	0.01	1.84	0.07
PEP	0.02	0.02	0.91	0.36
SLB	0.04	0.02	2.18	0.03
XOM	0.06	0.03	2.22	0.03
Model F-statistic: 149.8 (**p-value**: < 2.2e-16) **R-squared**: 0.8961**; Adjusted R-squared**: 0.8901				

Figure 16: Model 2 Output

	Model 1	Model 2	PCA Model 1
In-Sample	0.002139	0.001899	0.001946
Out-of-Sample	0.002403	0.002146	0.002255

Figure 17: MAE Output Summary Table – All models

Automated Variable Selection – PCA

The last section of this report will incorporate automated variable selection to select which principal components to use in a model. Specifically, we will fit a model using backward selection process. To do so, we will fit a model using the train data set used to fit PCA Model 1. The Code section outlines the R Script used to fit our new model which we will call Backward Model. The mathematical question for Backward Model is as follows:

Backward Model: $VV = \beta_0 + \beta_1 Comp.1 + \beta_2\ Comp.2 + \beta_3\ Comp.3 + \beta_4\ Comp.8 + \beta_5\ Comp.9 + \beta_6\ Comp.10 + \beta_7\ Comp.11 + \beta_8\ Comp.14$

Figure 18 is a summary output of the Backward Model. We can see that the overall model is statistically significant due to the p-value associated with the F-statistic. Also, the model appears to have one predictor variable (Comp.9) that is not statistically significant based on a significance level of 0.05. Also, as expected since we used principal components, the VIF values for the model are approximately one which means the model does not suffer from multicollinearity. We will see if our previous unsupervised decision rule produced the best predictive model. Currently, we cannot say that PCA Model 1 is the best predictive model. As noted already, we can use an automated variable selection (e.g. stepwise) to select the "best" number of principal components to keep. We should definitely consider a supervised approach to help us select which principal components to keep and not to keep.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0007	0.0001	4.90	0.00
Comp.1	-0.0023	0.0000	-53.02	< 2e-16
Comp.2	0.0005	0.0001	4.31	0.00
Comp.3	0.0006	0.0001	4.71	0.00
Comp.8	-0.0005	0.0002	-2.60	0.01
Comp.9	-0.0003	0.0002	-1.54	0.12
Comp.10	0.0005	0.0002	2.43	0.02
Comp.11	0.0004	0.0002	2.06	0.04
Comp.14	0.0006	0.0002	2.31	0.02
Model F-statistic: 362.3 (**p-value**: < 2.2e-16) **R-squared**: 0.8947**; Adjusted R-squared**: 0.8923				

Figure 18: Summary Output for Backward Model

Furthermore, the backward selection process suggested that we keep eight principle components which are displayed in Figure 18. This differs from PCA Model 1 since the Backward Model did not keep the first eight principal components. Figure 19 is an updated MAE summary table that compares the results from the Backward Model with the previous three models using both in-sample and out-of-sample data.

We can see that the Backward Model performed the best on the out-sample-data compared to all the other models. Given that we are concerned with predictive accuracy, the Backward Model appears to be the best model since it has the lowest MAE on the out-sample-data. The other statistical

measurement such as adjusted R-squared, p-values for the F-statistic, etc. are all relatively the same for all the models so using MAE as the "tiebreaker" seems appropriate.

	Model 1	Model 2	PCA Model 1	Backward Model
In-Sample	0.002139	0.001899	0.001946	0.001903584
Out-of-Sample	0.002403	0.002146	0.002255	0.002111469

Figure 19: MAE Comparison – All models (updated)

Conclusions

Overall, the aim of this report was to provide the results from various parts of the model building process. We covered the use of data visualization in exploratory data analysis, discussed how to remedy multicollinearity using principle component analysis, and did some model comparison to arrive at the "best" model with regards to predictive accuracy. Using an automated variable selection process in conjunction with principle component analysis allowed us to determine that the Backward Model is the "best" model. This model will help us at predicting the returns of the large-cap index fund from Vanguard (VV).

Code

```
#Load our data into R
my.data <- read.csv(file='stock_portfolio.csv',header=TRUE, sep=',')
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]]);


#Now we will conduct EDA by examining the correlations
# 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')
```

```
# 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)

#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+XO
M, data=returns.df)
summary(model.2)
vif(model.2)

#Start doing PCA
returns.pca <- princomp(x=returns.df[,-21],cor=TRUE)
# See the output components returned by princomp();
names(returns.pca)

returns.pca

print(summary(returns.pca), loadings = TRUE)

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)

#Use PCA for dimension reduction
# 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')

#Use PCA in predictive modeling
# 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))
pca1.mae.train
vif(pca1.lm)

# Score the model out-of-sample and compute MAE
pca1.test <- predict(pca1.lm,newdata=test.scores)
pca1.mae.test
pca1.mae.test <- mean(abs(test.scores$VV-pca1.test))



# 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. 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))

summary(model.1)
summary(model.2)

model1.mae.train
model1.mae.test

model2.mae.train
model2.mae.test

# Fit model.2 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))

#Predictive modeling using automated variable selection with PCA
full.lm <- lm(VV ~ ., data=train.scores)
summary(full.lm)

library(MASS)
#Backward variable selectiom
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.mae.train
backward.test <- predict(backward.lm,newdata=test.scores)
backward.mae.test <- mean(abs(test.scores$VV-backward.test))

backward.mae.test
```