# Assignment #6: Principal Components in Predictive Modeling

**Prabhat Thakur**

**Data:** The data for this assignment is the stock portfolio data set. This data is posted in Canvas.

**Introduction:**

The goal of this assignment is to perform multivariate analysis on stock portfolio data set using sample R code and demonstrate learned knowledge of the concepts from this analysis. In this assignment we will use Principal Components Analysis technique as a method of dimension reduction and as a remedial measure for multicollinearity in linear regression. Along the way we will also learn how to manipulate data in R and make some R graphics useful for these topics.

**Assignment Tasks:**

**(1) Data Preparation:**

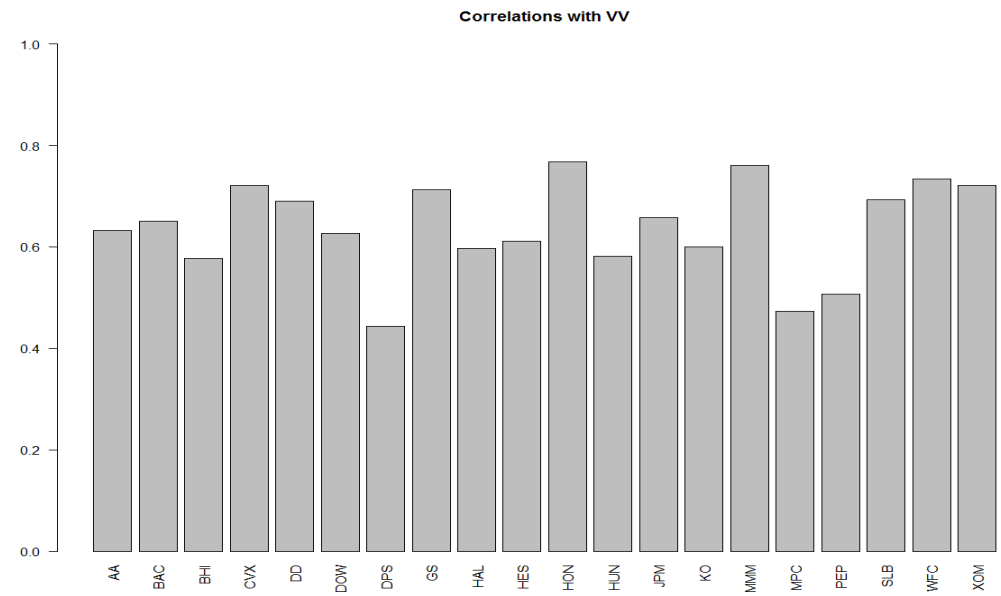
The stock price data is loaded in a data frame. A single row in our data frame consists of date (working weekdays), daily closing stock prices for twenty stocks and a large-cap index fund from Vanguard (VV). All variables with the exception of the date field contain continuous numeric data with two years of stock data (502 rows) starting with Jan 3rd, 2012.

The date column in our data frame is a factor variable and converted to date variable. Observations are sorted by date in ascending order. The stock price data we have is a time series data of incremental dates and suffers from the highly correlated prices for subsequent days. We can perform transformation of stock price by taking log of today's price versus yesterday's price ratio. By doing this for each stock we remove the autocorrelation that is inherent in the time series data. We do the same for the index fund to keep it consistent. We created a new data frame with ordered and log transformed observations.

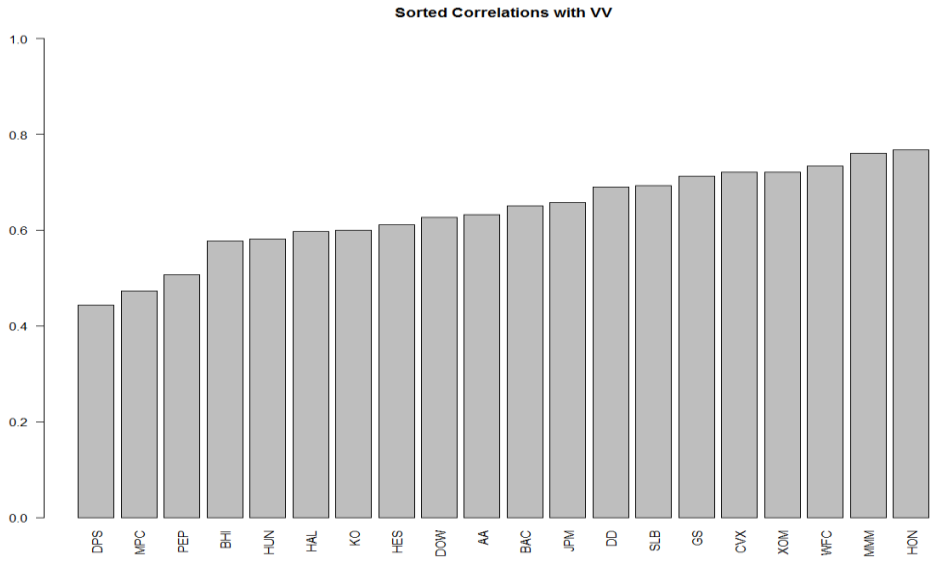
**(2) Exploratory Data Analysis:**

We have our data ordered by date and log transformed to remove the autocorrelation. We will now begin our exploratory data analysis by examining the correlations. First we want to compute complete correlation matrix, and then consider a visualizing a subset of those correlations for quick and easy comparison.

We begin with checking how VV (index fund) variable is correlated with all other stocks. We calculated correlation matrix and took out the VV row from it, next we generated a barplot of stocks price correlation with VV (index fund).

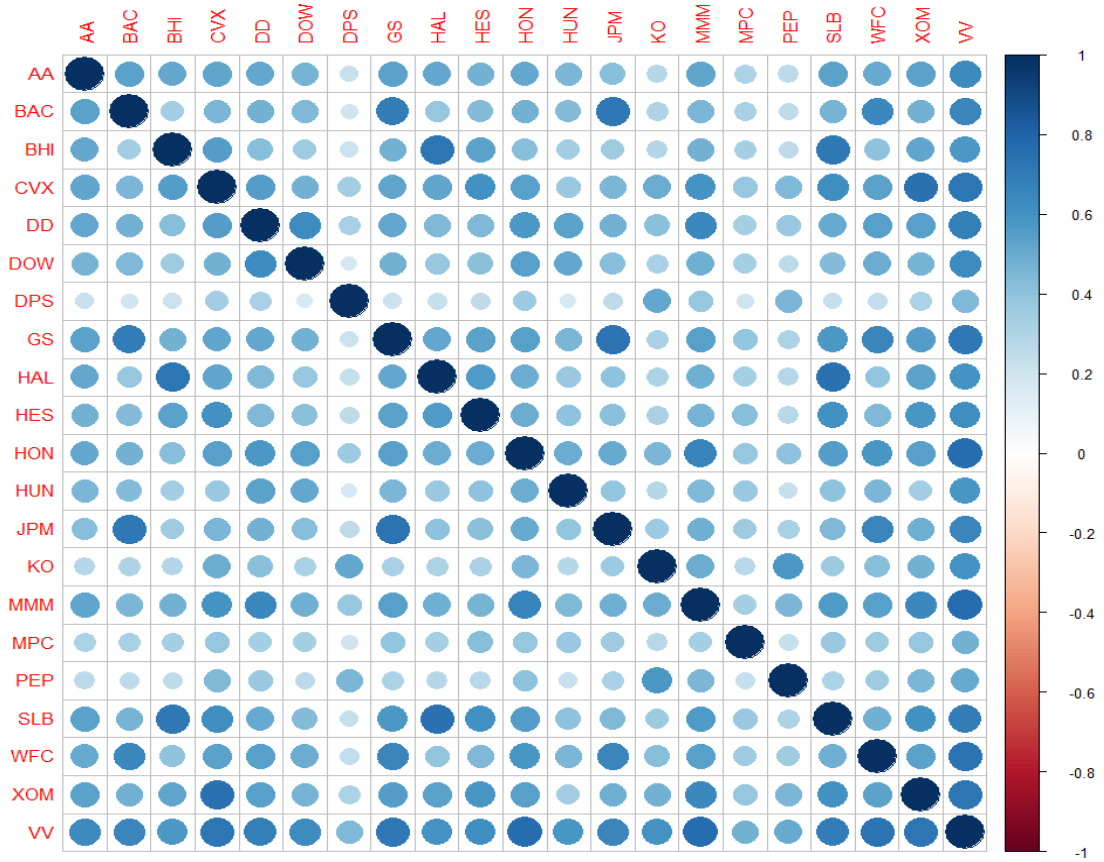



Barchart of sorted correlation factor in ascending order will be even easier to compare each stock correlation with VV. From the plot we can easily observe that HON and MMM shows the highest correlation with VV and DPS and MPC shows the lowest. Except the first three stocks rest of the stocks correlation with VV is very close to each other, in another words variation in the correlation factors do not differ much.




**(3) Data Visualization:**

In next step we will analyze pairwise correlations in the data. Using corrplot, we can get a better visual representation of the correlation for each pair of stocks. The plot is shown below.




From this pairwise correlations matrix plot, we can observer few things very quickly.

1. No variable is negatively correlated with other variable.
2. Stocks with the lowest pairwise correlations to the other stocks show up as lighter circles for almost all other stocks.
3. DPS, MPC and PEP stocks are not closely correlated with most other stocks
4. Looking at VV row we will find the stocks that have the highest correlation with the index fund. Based on the visual clues alone, going with CVX, DD, GS, HON, MMM, SLB, WFC, and XOM may be a good selection of variables for the model.

It is clear that corrplot is more insightful than our simple barplot. From a single plot we could observe so many important correlation related information. This is one of many differences between 'statistical graphic' and a 'data visualization'. Data visualization can reveal more insight from data.

This correlations matrix plot can also help validate if our data has multicollinearity issue. From the plot we can say that DPS, MPC and PEP should have lowest VIF values (a measure of multicollinearity in a model). Similarly, based on the plot above I would expect GS, HAL, and SLB to have the highest VIF values.

We will check these variables VIF values in next section when we build models using them.

**(4) VIF validation through Models:**

In addition to statistical graphics and data visualizations we can use models as tools for exploratory data analysis. Modeling is an inherently iterative process, and we typically begin the modeling process by fitting some 'naïve models' that are nothing more than models that we believe are good starting points for the modeling process. Typically we might start with a small model and the full model as our two initial naïve models. The full model also allows us to compute the VIF value for every predictor variable.

Below is comparison of initial model and full model.

| Model | #of Predictors | RSE | R-squared | F-stat | Highest VIF |
|---|---|---|---|---|---|
| Initial | 9 | 0.002951 | 0.8518 | 313.5 | 2.705795 |
| Full | 19 | 0.002669 | 0.8812 | 187.8 | 3.257595 |

VIF values from full model is listed below. Lowest and highets VIFs are preety much same as we suspeted from the correlations matrix plot.

```
> sort(vif(model.2))
     MPC      DPS      PEP      HUN      DOW       KO      HES       DD      HON      WFC      BAC      BHI
1.376185 1.524399 1.719788 1.721319 1.961953 1.967512 2.095666 2.432674 2.447013 2.528808 2.558097 2.603510
     MMM      JPM      HAL      CVX      XOM       GS      SLB
2.670404 2.844537 2.902240 2.909686 2.924084 3.190808 3.257595
```

From this table data, we notice that characteristics (such as RSE and R-squared) of the model improves as we add more variables. However, we also see that the VIF scores increase. I've only listed the highest VIF values however the values across the board for the min, median and max scores overall increase as we add more variables to the model moving from Initial to full model. The F-statistic continues to decrease as variables are added as well.

Ideal value for VIF is 1 which means both variables are orthogonal. In both model we do have some multicollinearity issue but not serious. It is suggested that a VIF in excess of 10 is an indication that collinearity may be causing problems in estimation.

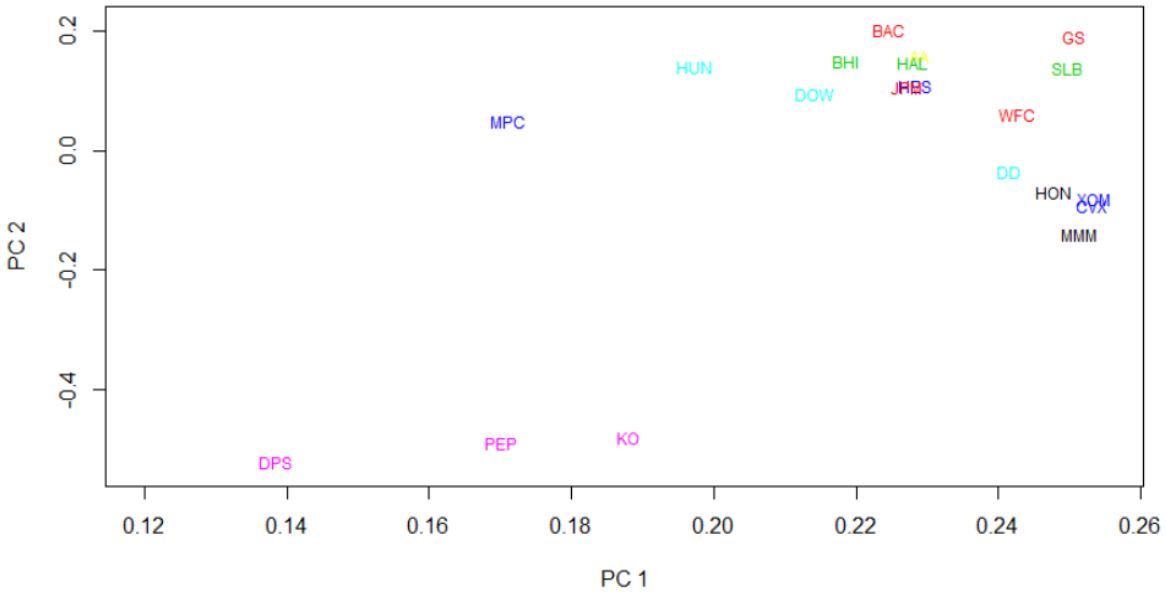
**(5) Principal Component Analysis:**

One remedy for multicollinearity is to transform the predictor variables using principal component analysis. PCA transforms predictor variables to orthogonal variables which removes the multicollinearity from data if any. PCA is also used for clustering predictor variables based on underling characteristics.

For PCA analysis, Variables should be standardize. However we are not going to explicitly standardize the data to mean zero and unit variance. The log-return transformation that we computed for each security is our standardization.

The variable loadings is the matrix of the eigenvectors for each component and from the first column we can form the first principal component for each stock. When we plot the first component versus the

second component, we can see these eigenvector values and the stocks with higher correlation between the respective components. Below is plot of first two principal components loadings values, color coded by industry. For example ticker symbol in red color are Banking and green ones are Oil Field Services.



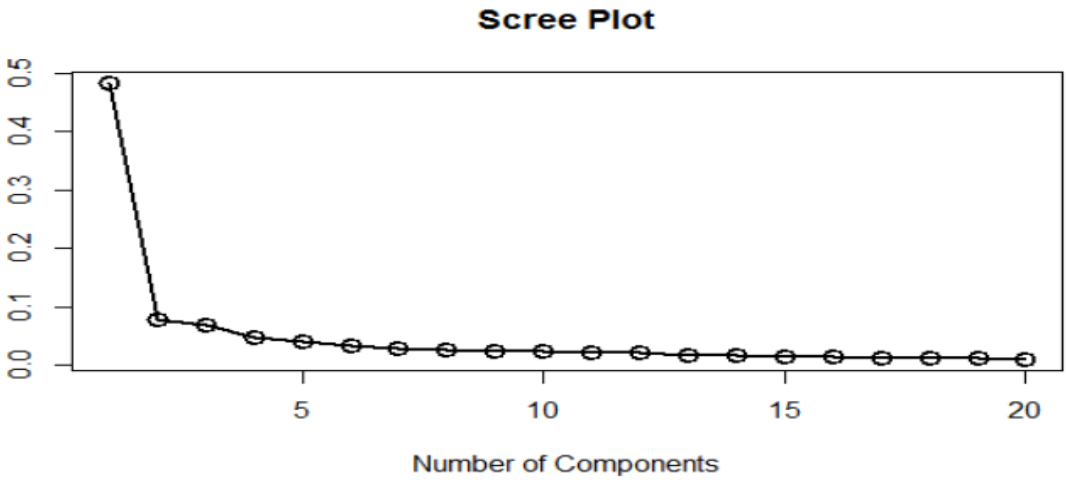

In above plot, we can clearly see stocks are grouped in two clusters, lower left and upper right. Also it is surprising to see stocks from same industry falling in same clusters. For example all three stocks from Soft Drinks industry are grouped in lower left corner. Also the first 6 stocks with lowest VIF value (lower correlation) are in the left side of plot and stocks with higher VIF (high correlation) values are in right side of the plot. The first two components show weaker correlation for soft drink beverage companies whereas the banking and oil field services companies show a higher correlation between them.

**(6) Dimension reduction / Component Selection:**

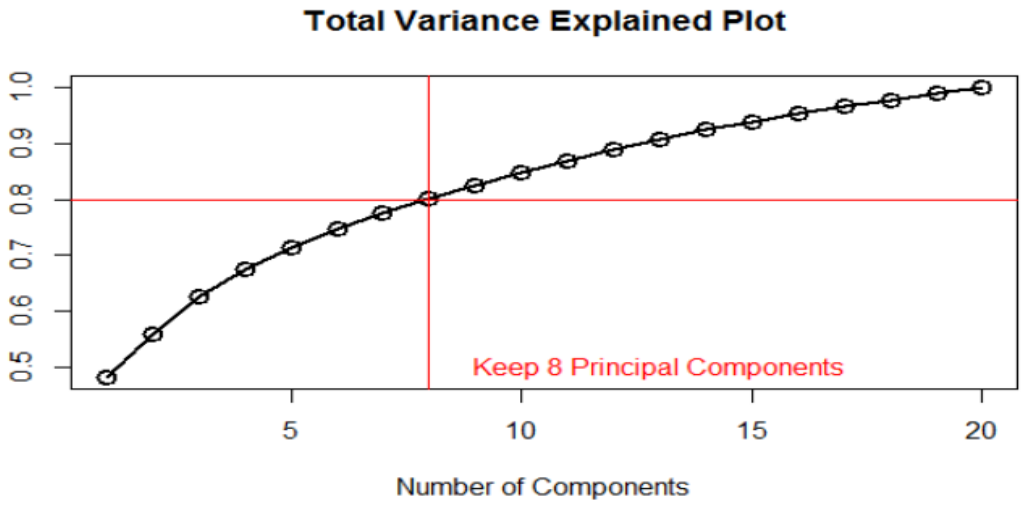
When we want to use PCA for dimension reduction, then after we compute the principal components we need to decide how many principal components to keep. One tool for deciding the number of principal components to keep is to make a scree plot.

We can create two scree plots a) Variance explained by components and b) Cumulative Variance Explained by components. Scree plot which gives us a clear indication of which principal components provide the most variance in the component data. From below plot, we can see that pc1 explains almost 50% of the variation before a big drop down to pc2 at less than 8%. Over 70 percent of the total

variation is covered by the first five principal components.




If we want to determine a cutoff percentage, the total variance plot can show the cumulative variance for each successive component.




The number of principal components to keep is largely a decision that must be made with goals of the particular problem we are working on. From above keeping eight principal components is a good first choice. We can see that they explain 80% of the total variance.

**(7) Principal components in predictive modeling:**

Now let's use principal components in predictive modeling. The predictor variables for our predictive model will be the PCA scores. The principal component scores contain the mapping of the original data into their principal components. The score for each component is essentially a numerical value indicating the strength of contribution to that component. Component scores are calculated using eigenvector values and the original data. In order for us to be building predictive models, we need to

have a train data set and a test data set so let's split our data into train and test data sets manually by generating a uniform(0,1) random variable and attaching it to the data frame.

In order to simplify this assignment we will compute and score the principal components on the whole data set, and then we will split it into training and test data sets. In a production environment we would have estimated our principal components on our training data set, stored our eigenvectors, and then scored them on each new data set (out-of-sample) as needed. However, the mechanics of those computations would distract us from the important parts of this assignment. If you understand the difference between what we are doing in this assignment, and what we would do in a production environment, then you are welcome to try the latter as a separate practice exercise for your own personal development.

```
# 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));
```

Using our train and test data sets, let's fit a linear regression model using the first eight principal components and compute the Mean Absolute Error (MAE) for that model. We will call this model pca1.lm. Let's also score that model out-of-sample on our test data set and compute the out-of-sample MAE.

Note: The VIF values associated with every predictor variable in any principal components regression model should all be one. Why?

The data was split into train and test samples using random assignment with 70% in the training set. The RSE and R2 results of the linear model are very comparable to the models using original data above in section 4. Using the first eight principal components, we've computed the linear model with a RSE of 0.002792 and an R-squared value of 0.8693, however looking at the p-values we see that only the first four are principal components are statistically significant. The VIF values are all approximately 1 as we

expected from the PCA which now means we have no collinearity. By definition, all principal components are orthogonal and thus give the lowest possible VIF value of one.

We also calculate the mean absolute error for the pca1 model with the first eight components and we get 0.002052123 and 0.001970655 for the train (in-sample) and test (out-of-sample) sets respectively which seems reasonable.

**(8) Model Comparison and Best Model:**

Is our principal components regression model a 'better' model or the 'best' model? What does it mean for Model A to be better than Model B? How about in predictive modeling? When we compare models in predictive modeling we need to compare the models on an objective performance criterion.

Let's compute and compare the in-sample and out-of-sample predictive accuracy of our principal components regression model to Model #1 and Model #2 from earlier. We will create the matching train/test split of the raw log returns data, fit and score Model #1 and Model #2, and compute the appropriate MAE values to compare with pca1.lm.

```
# 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 ~
AA+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));
```

Present the MAE values from models pca1.lm, model.1, and model.2 in a table so that they are easily compared and discussed in your paper. Discuss these results. Which model is the best model? Why?

Below is the table comparing model performance (predictive accuracy) of principal components regression model, Model #1 and Model #2 on in-sample and out-of-sample data.

| Model | #of Predictors | RSE Train | R-squared Train | MAE Train | MAE Test |
|---|---|---|---|---|---|
| Model1 | 9 | 0.00304 | 0.8454 | 0.002243 | 0.002073 |
| Model2 | 19 | 0.002758 | 0.8764 | 0.001997 | 0.001903 |
| PCA | 8 | 0.002792 | 0.8693 | 0.002052 | 0.001971 |

One important thing to note here is that the RSE, R-squared, and MAE values are all very similar for Model 2 and PCA model. From model 2 and PCA model we see that similar predictive performance can be achieved with fewer variables using PCA.
From above comparison and based on predictive performance (MAE) on train and test data set, Model 2 (full model) is the best model however our PCA model is certainly a better model due to less number of predictors and absence of multicollinearity effect.

**(9) Principal Components Analysis – Unsupervised and Supervised Learning**

Principal components is an example of a class of Statistical Learning methods called 'Unsupervised Learning'. They are called unsupervised learning methods because they have no response variable. Statistical Learning methods that have a response variable are called 'Supervised Learning'. In this assignment we have looked at PCA from the traditional unsupervised learning perspective. Now we want to look at PCA in a supervised learning perspective, which will be more useful when using principal components in predictive modeling.

In predictive modeling we frequently wrap unsupervised learning methods into a supervised learning problem as a means to improve our predictive models. One example where we wrap an unsupervised learning method into a supervised learning problem is using PCA to reduce the dimension of predictor variables in a linear regression model. Earlier in the assignment we selected eight principal components as the 'correct' or 'best' number of principal components to keep, and we fit the regression model pca1.lm using the first eight principal components. The decision to keep eight principal components was made using a decision rule from the standard unsupervised learning problem. Does our unsupervised decision rule translate to producing the best predictive model? More directly, is the eight principal components model the best predictive model? How else might we select the 'best' number of principal components to keep in our predictive modeling problem? Why don't we consider a supervised approach of using variable selection to select the number of principal components to keep, and which principal components to keep?

```
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));
```

Does our unsupervised decision rule translate to producing the best predictive model? More directly, is the eight principal components model the best predictive model? How else might we select the 'best' number of principal components to keep in our predictive modeling problem? Why don't we consider a supervised approach of using variable selection to select the number of principal components to keep, and which principal components to keep?

How many principal components does the variable selection approach suggest that we keep? Which ones? How does this differ from our previous discussions about the number of principal components to keep? Create a new table where you add these new MAE results to the previous MAE results. How does our backward.lm model compare to all of the other models that we have fit in this assignment? Which model is best and why?

The 8 principal component model is not necessarily be the best model however it is a useful transformation to have for addressing the multicollinearity issue. The unsupervised learning aspect of PCA allows us to choose a number of predictors based on the amount of variation explained where we are simply interested in cumulative percentage of variance. By selecting the PCA model that has similar fit performance as the original, we can verify that the PCA model achieves the intended goal of accuracy and R2 values but with less variables. Using the opposite approach of starting with the analysis of which original variables to keep would be difficult with collinearity present and the pairwise analysis of variables in a visual way could be prohibitive as the number of predictors grows.

Our goal for choosing the 'best' model should be to obtain a reduced-variable model that achieves similar predictive performance as a full model using original data. Based on our models above, the decision to include eight is based on the fact that we can account for 80% of the variation by choosing the first eight principal components. This variable reduction technique not only resolves the multicollinearity problem, it gives us confidence in our analysis that we are not missing important data in higher dimensional problems. This is important because as the size of the variables grows, our ability to properly perform pairwise comparisons using traditional univariate methods becomes costly, time consuming or outright impossible to complete in a structured manner.

Instead of using total variance explained threshold, we will now use automated variable selection techniques like forward selection, backward elimination and stepwise selection. We used backward elimination technique on principal components. It suggest to keep following 9 principle components

```
Comp.1 + Comp.2 + Comp.3 + Comp.4 + Comp.5 + Comp.10 + Comp.11 + Comp.14 + Co
mp.16
```

Total variance explained by these principle component is 78.9% which is less than our 8 pc model, however the this model perform slightly better than the 8 pc model. I added two more model in the earlier model comparison table. Full PCA model and backward elimination PCA model.

| Model | #of Predictors | RSE Train | R-squared Train | MAE Train | MAE Test |
|---|---|---|---|---|---|
| Model1 | 9 | 0.00304 | 0.8454 | 0.002243 | 0.002073 |

| | | | | | |
|---|---|---|---|---|---|
| Model2 | 19 | 0.002758 | 0.8764 | 0.001997 | 0.001903 |
| PCA | 8 | 0.002792 | 0.8693 | 0.002052 | 0.001971 |
| PCA Full | 20 | 0.002751 | 0.8774 | 0.001998 | 0.001921 |
| PCA Backward | 9 | 0.002729 | 0.8754 | 0.002006 | 0.001939 |

The backward PCA model is the best model out of all the models we created in this assignment. It is based on PCA hence brings all the benefits we already discussed above. Its predictive performance is almost same as Model 2 and PCA Full but it uses only 9 predictors as compared to 19 and 20 in Model2 and PCA Full.

It seems clear that the supervised approach of using variable selection like backward elimination to select the number of principal components to keep, and which principal components to keep provide the most benefits and should be selected as the 'best model' to use as compared to arbitrary selecting the number of components based on % of explained variance.