Robin Singh

***Answers in images and blue text.

Assignment #1: Principal Components (PCA) and Stochastic Neighbor Embedding (t-SNE)
MSDS 411

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

Assignment Instructions:

In this assignment we will use Principal Components Analysis and t-SNE as methods of dimension reduction. The idea is that PCA is a good strategy to use as a remedial measure for multicollinearity in OLS linear regression or Logistic Regression, while t-SNE gives a non-linear mechanism to reduce dimensionality. 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) Let's begin our analysis with some data prep. Our raw data 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.

The following code will get your started. Note that you need to define the variable `my.path`.

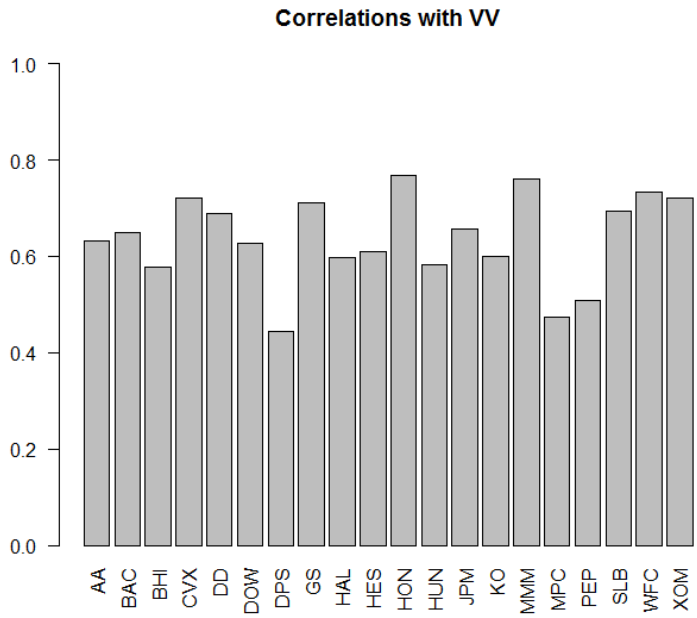
```
my.data <- read.csv(paste(my.path,'stock_portfolio.csv',sep=''),header=TRUE);
head(my.data)
str(my.data)

# 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]]);
```

(2) Now that we have our data defined, let's begin our exploratory data analysis by examining the correlations. We will want to compute the complete correlation matrix, and then consider visualizing a subset of those correlations for quick and easy comparison. Which row or column of the correlation matrix are we most interested in?

Along with analyzing the correlations of the stocks to each other, we are also highly interested in comparing the correlation between each individual stock log return with the Vanguard index fund (VV). This could help us determine which of them is most heavily contributing to the Vanguard Index Fund returns.

```
# 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')
```

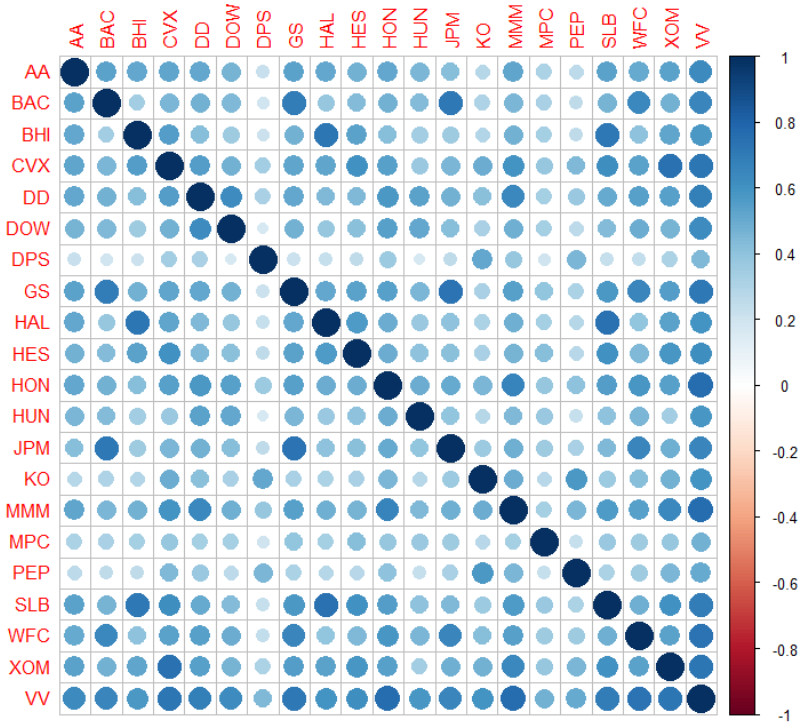


(3) How about we make an even fancier data visualization of the correlations? The corrplot will allow us to visualize all pairwise correlations in the data.

- Is the corrplot more useful or insightful than our simple barplot?
  - Yes, because we can now more clearly analyze the correlations of individual stocks to themselves as well and more accurately see which ones relate more to the VV index fund.
- What is the difference between a 'statistical graphic' and a 'data visualization'?
  - Statistical graphic is a more simplified visualization of the data, perhaps comparing a handful of elements to one other element while data visualization is a grander scale of data visualized and compared to each other in one visual illustration. With data visualization you can see more trends across the data.

Note: You might need to install the `corrplot` package.

```
# 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)
```



With respect to the concept of multicollinearity, look at the corrplot and identify three stocks that should have low VIF values? Similarly, pick three stocks that should have high VIF values?

3 low VIF value stocks: DPS, MPC, PEP

3 High VIF value stocks: WFC, CVX, MMM

(4) 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.

Note: You will need the `car` package in order to use the `vif()` function.

```
# 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,dat
a=returns.df)
summary(model.2)
```

```
vif(model.2)
```

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         0.0585118  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
```

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      HUN       KO      MMM      MPC      PEP      SLB      XOM
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 1.743913 1.981647 2.684942 1.376706 1.720663 3.258982 2.949826
```

Is multicollinearity a concern for either of these models?

When comparing the first model with the full model, you can see some of the VIF values increase significantly, especially those of GS, JPM, KO, and XOM. This means multicollinearity could be an issue.

What value of VIF should make you concerned about multicollinearity?

VIF values over 2.5 should be concerning. This includes GS and MMM in model 1 and BAC, GS, JPM, WFC, BHI, CVX, HAL, MMM, SLB, and XOM.

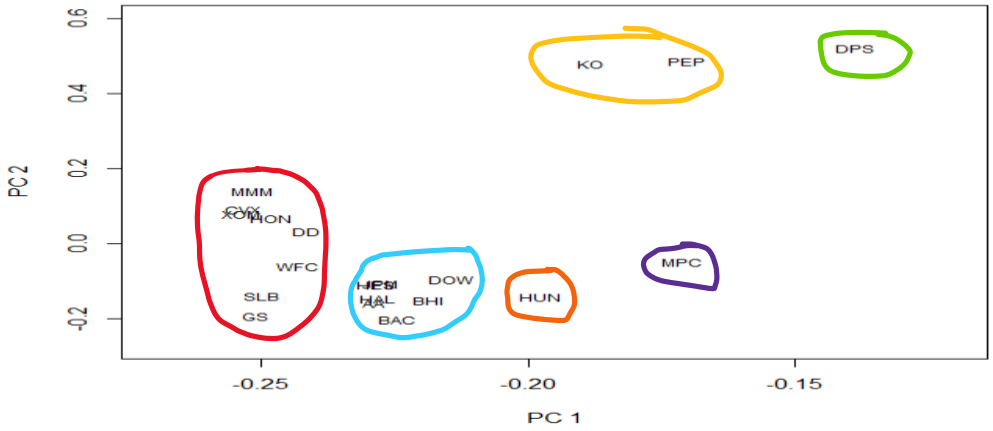
(5) One remedy for multicollinearity is to transform the explanatory or predictor variables using Principal Component Analysis (PCA). Let's compute the principal components for the return data. We will use the `princomp()` function that is available in Base R.

Note that 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.

First, plot the loadings for first two principal components from the principal components analysis. (What are the loadings?) When we plot the loadings, we can see relationships in the data.

- What groupings (or clusters) do you see in the plot of the first two principal components? Any surprises?
    - The groupings have been circled in the diagram below.

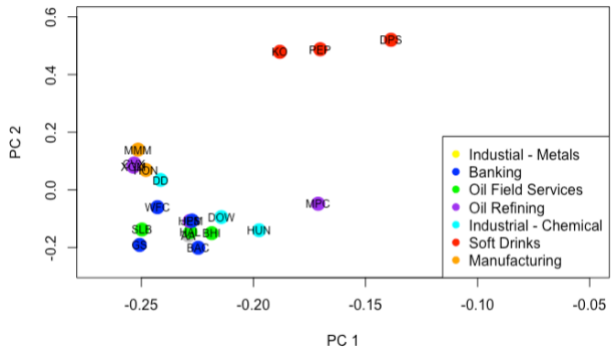
```
returns.pca <- princomp(x=returns.df[,-21],cor=TRUE)
# See the output components returned by princomp();
names(returns.pca)

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)
```



Color code this plot by industry? Here are the ticker symbols and their industry. You do not need to use the colors that used in the table.

	Ticker	Name	Industry	
1	AA	Alcoa Aluminum	Industrial - Metals	7
2	BAC	Bank of America	Banking	2
3	BHI	Baker Hughes Incorporated	Oil Field Services	3
4	CVX	Chevron	Oil Refining	4
5	DD	Dupont	Industrial - Chemical	5
6	DOW	Dow Chemical	Industrial - Chemical	5
7	DPS	DrPepper Snapple	Soft Drinks	6
8	GS	Goldman Sachs	Banking	2
9	HAL	Halliburton	Oil Field Services	3
10	HES	Hess Energy	Oil Refining	4
11	HON	Honeywell International	Manufacturing	1
12	HUN	Huntsman Corporation	Industrial - Chemical	5
13	JPM	JPMorgan Chase	Banking	2
14	KO	The Coca-Cola Company	Soft Drinks	6
15	MMM	3M Company	Manufacturing	1
16	MPC	Marathon Petroleum Corp	Oil Refining	4
17	PEP	Pepsi Company	Soft Drinks	6
18	SLB	Schlumberger	Oil Field Services	3
19	WFC	Wells Fargo	Banking	2
20	XOM	Exxon-Mobile	Oil Refining	4

21	VV	Vanguard Large Cap Index	Market Index	9

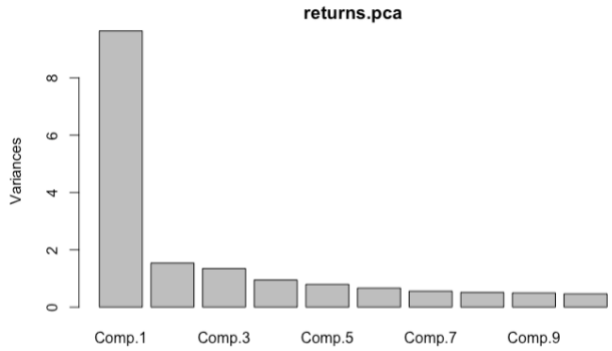


(6) In using PCA for dimension reduction, after we compute the principal components, we need to decide how many principal components to keep or retain.

- How many principal components do you think that we should keep? Why? (Hint: What decision rules should we use to determine the number of principal components to keep?)
  - I believe we should go along with the number of groupings we had found to determine the number of Principal components, which for me would be 6.
- Later in the assignment we will use the first eight principal components. Why eight?
  - This could be the estimated amount of groupings determined by the professor, which differs from my estimate of 6.

One tool for deciding the number of principal components to keep is to make a scree plot. Note that R will make a scree plot by default when plotting a PCA object.

```
# Plot the default scree plot;
plot(returns.pca)
```



The default scree plot is a scree plot, but it is not that great. Maybe we should know how to make our own plots and make plots that are better than the default scree plot. Let's make two common plots associated with PCA, and let's make them look nicer than the default scree plot.

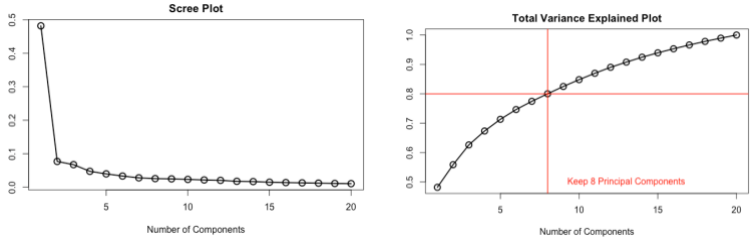
Note that we need to understand the PCA output and how to make plots in R in order to make these plots.

```
# 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 these two plots to decide how many principal components to keep.

According to the plot, we should keep 8 principal components as that is when the change in variance becomes very small with each incremental principal component kept.



(7) Now let's use principal components in predictive modeling. The predictor variables for our predictive model will be the PCA scores. (What are the scores?)

The PCA scores are created values that represent the relationship or effect of that feature from the origin.

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 VIF values are all quite close to 1, which means it is not correlated to other variables. We had tried to reduce multicollinearity through PCA analysis.

(8) 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.2 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+X
OM, 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?

Models	Train Score	Test score
PCA.lm	0.002023279	0.002122295
Model 1	0.00218203	0.002161644
Model 2	0.001903272	0.002115065

From this table you can see the train and test MAE scores for each of the models. The highest MAE results from Model 1, which makes sense as is accounts for the least amount of stock prices against VV, so it should be the least accurate. That is then followed by the PCA LM which has a lower MAE score for both the Train and Test datasets. This means that doing a PCA analysis with 8 components allowed the model to account for more of the variance than Model 1, but not as much as model 2 which considers all the various stocks and the linear relationship to the VV index fund. Model 2, hence is the best model, having the smallest MAE score, but the PCA LM still did well for just using 8 components and simplifying the data so much.

(9) 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));
```

How many principal components does the variable selection approach suggest that we keep?
The variable selection approach suggests we keep 9 principal components.
Which ones?
Comp.1 + Comp.2 + Comp.3 + Comp.8 + Comp.10 + Comp.11 + Comp.12 + Comp.14 + Comp.16
How does this differ from our previous discussions about the number of principal components to keep?
This states that we keep one more Principal component than our last PCA linear model as it suggests that this is the optimal number of principal components for this data which may account for most of the variance. It also analyzes all of the principal components and selects which ones may contribute the most to our model only choosing those (hence why it skips some numbers.
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?

Models	Train Score	Test score
PCA.lm	0.002023279	0.002122295
Model 1	0.00218203	0.002161644
Model 2	0.001903272	0.002115065
Backwards.lm*	0.001915709	0.002110214

Our backwards.lm model now is even closer to the train and test scores of model 2, with the training MAE being only slightly above, but or test MAE being lower than that of model 2, indicating it is even better and is our best model so far. The reason for this may be that is no longer overfitting as much as the full model may have on the test data.

(10) Let's go back to the beginning with this data set, and use the t-SNE approach to reduce and visualize this data. Conduct a t-SNE analysis on the Stock Portfolio data. Interpret the results, then compare and contrast the t-SNE results with that of the Principal Components analysis you've conducted previously.
I chose to run t-SNE with three dimensions (as that is the maximum number of dimensions I could use) on this dataset and have to say the results were not as good as those conducted with PCA.

```r
# using rtsne
set.seed(1) # for reproducibility
tsne <- Rtsne(returns.df[,-21], dims = 3, perplexity=30, verbose=TRUE, max_iter = 5000, learning = 200)

```

```
Call:
lm(formula = VV ~ V1 + V2 + V3, data = train.scores)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0139107 -0.0027801  0.0000532  0.0030810  0.0222382

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 8.384e-04  2.604e-04    3.22  0.00141 **
V1          9.663e-04  5.027e-05   19.22  < 2e-16 ***
V2          3.229e-03  1.821e-04   17.73  < 2e-16 ***
V3          1.639e-03  8.355e-05   19.62  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.004781 on 334 degrees of freedom
Multiple R-squared:  0.5363,    Adjusted R-squared:  0.5322
F-statistic: 128.8 on 3 and 334 DF,  p-value: < 2.2e-16

      V1       V2       V3
15.66084  6.27518 20.41849
[1] 0.003625234
[1] 0.00429303
```

Here is a chart comparing the results below:

Models	Train Score	Test score
PCA.lm	0.002023279	0.002122295
Model 1	0.00218203	0.002161644
Model 2	0.001903272	0.002115065
Backwards.lm*	0.001915709	0.002110214
t-SNE.lm	0.003625234	0.00429303

As you can see, both the training and test MAE values were higher in this case. This to me seems to say that three dimensions is not enough to account for most of the variance in the data and hence needs more components to account for more of the variance. Hence, for my case, the PCA outperformed t-SNE due to more parameter adjustment in my case.

(11)Please write a reflection on your modeling experiences during this first assignment.
This assignment allowed me to become more familiar with PCA analysis and exposed me to t-SNE as an alternative to create components to reduce the dimensionality of the complex stock dataset while analyzing it against the VV index fund. I would have liked a little more explanation on how to code the t-SNE to visualize the data, but I believe I had figured it out a little. Overall, this was a great assignment to learn more about the field of unsupervised learning through R.

Assignment Document:

All assignment reports should answer each of the questions separately. Please be sure to clearly indicate which question is being addressed. Results should be presented and discussed in an organized

manner with the discussion in close proximity of the results. The report should not contain unnecessary R-code, intermediary computations, R-results, or non-essential information. The document should be submitted in pdf format. Name your file Assign1_LastName.pdf.