Principal Components Analysis (PCA) and Stochastic Neighbor Embedding (t-SNE)

Serra Uzun, MSDS_411 FALL 2020
09/27/2020

Introduction

The Stock dataset we will be using for our Principal Components Analysis (PCA) and Stochastic Neighbor Embedding (t-SNE) consists of the daily closing stock price, the return amount of Stock for the Stock, and the date of the recorded information. The PCA will be performed on the Stock dataset (aka Stock.Data) for linear dimensional reduction, whereas we will also be conducting a t-SNE analysis for nonlinear dimensionality reduction. The PCA will be used for more mathematical approaches, while t-SNE will also be performed for more probabilistic analysis. We will be conducting an Exploratory Data Analysis (EDA), examine correlations and multicollinearity between variables, and select principal components and predictor variables for our linear regression model.

The Dataset

As the initial step, we study the dataset's properties to better understand the data we currently hold. The Stock dataset consists of 502 observations and 43 variables. The 43 variables in the dataset consist of one variable for 'Date', 20 predictor stock variables, one response variable, and 21 variables with return on all stocks, both predictor and response variables. All variables mentioned earlier are numeric except for the variable 'Date. The variables with ticker names suggest the information on stock price, and the variable names that begin with 'return_' suggest the return made on the stock. The variables of the dataset are listed below:

- Date
- AA
- BAC
- BHI
- CVX
- DD
- DOW
- DPS
- GS
- HAL
- HES
- HON
- HUN
- JPM
- KO
- MM
- MPC
- PEP
- SLB
- WFC
- XOM
- VV
- return_AA
- return_BAC
- return_BHI
- return_CVX
- return_DD
- return_DOW
- return_DPS
- return_GS
- return_HAL
- return_HES
- return_HON
- return_HUN
- return_JPM
- return_KO
- return_MMM
- return_MPC
- return_PEP
- return_SLB
- return_WFC
- return_XOM
- return_VV

Correlation Analysis for Returns *(Questions #2 and #3)*

The correlation matrix and plots give us a better overview of the relationships between the stock return response and predictor variables, excluding the stock returns. The correlation table and bar plot suggest that all predictor variables are positively correlated with VV's response variable. As all correlation values are above 0, any increase in a predictor stock return variable will indicate an increase in VV stock return, yet in different levels.

	VV
AA	0.63

	VV
HON	0.77

BAC	0.65	HUN	0.58
BHI	0.58	JPM	0.66
CVX	0.72	KO	0.60
DD	0.69	MMM	0.76
DOW	0.63	MPC	0.47
DPS	0.44	PEP	0.51
GS	0.71	SLB	0.69
HAL	0.60	WFC	0.73
HES	0.61	XOM	0.72

Table 1: Correlations between VV the Response Variable and Predictor Variables

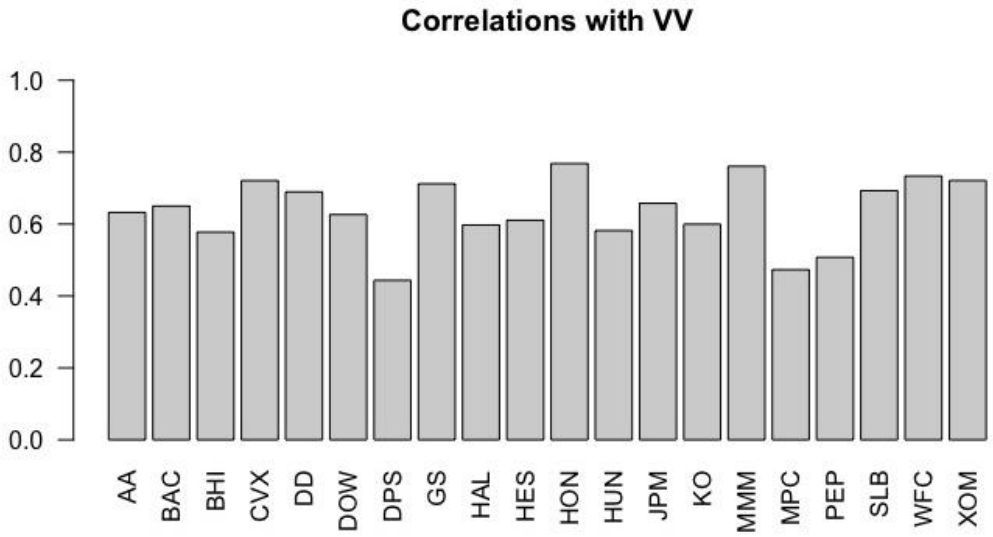

Figure 1: Bar Plot of the Correlations between VV the Response Variable and Predictor Variables

Per Table 1 and Figure 1, HON, MMM, and WFC are the predictor variables that are most positively correlated with VV, with 0.77, 0.76, and 0.73, respectively. The below Correlation Matrix, in Figure 2, helps us better visualize the relationships between predictor variables and response variables. The results presented in the correlation matrix plot in Figure 2 indicate the same correlations we have observed in Table 1, and Figure 1 yet presents it in a more audience-friendly format. The most left and the very bottom column and row (indicated with arrows in Figure 2) are the ones that we are interested in as they are the ones that show the correlation between VV, response variable, and all other stock return variables.

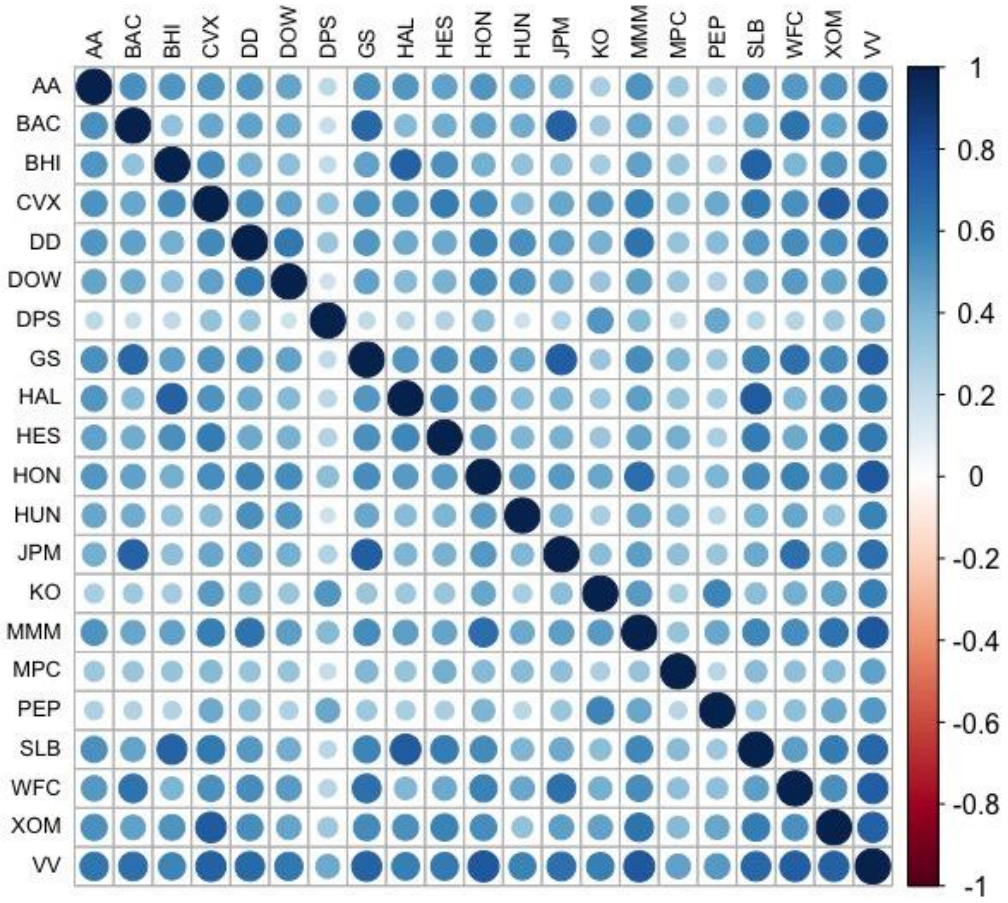

Figure 2: Correlation Matrix Plot of Stock Returns

The correlation matrix plot in Figure 2 helps us get a glimpse of multicollinearity and guess variables that are likely to get VIF (Variance Inflation Factor) on the lower and higher end of the spectrum. Scanning through the plot in Figure 2, we can expect GS, XOM, and SLB to have the highest VIF. Through the correlation matrix, we see that these predictor variables have relatively strong correlations will various and numerous other variables in the dataset. On the other hand, we can see from Figure 2 that it is quite likely that we will get a low VIF for DPS, PEP, and MPC. These three variables have a noticeably weak correlation with the other variables in the Stock return dataset.

(Naïve) Linear Regression Modelling *(Questions #4)*

The process of running naïve models is useful to obtain VIF scores for each variable that is chosen for the model and provide us with a more detailed view on multicollinearity within our dataset. The random selection of predictor variables of GS, DD, DOW, HON, HUN, JPM, KO, MMM, and XOM for the naïve linear regression model (Linear Model #1, LM #1) we use as a part of our iterative process shows no concerning results. (See appendix for LM #1 Summary).

The Linear Model #1 presents an R-squared value of 0.85, suggesting that the 85 percent of the dataset variance can be explained through the variables in the LM #1 and an overall p-value below 0.01, which indicate statistical significance. Below are the VIF scores of the variables chosen for LM #1.

GS	DD	DOW	HON	HUN	JPM	KO	MMM	XOM
2.7058	2.3683	1.9198	2.2614	1.6333	2.3246	1.4732	2.5902	2.0737

Table 2: VIF scores of variables selected for and used in Linear Model #1

Upon reviewing Table 2, we can see that the selected variables' VIF scores range roughly between 1.4 and 2.8. If we were to observe any VIF score equal to 1 or greater than five and less than ten we would be concerned with variable having either no multicollinearity, meaning just very weak correlations with other variables in the dataset, or too much multicollinearity, meaning the variable is correlated with too many of the other variables; thus it is not a good predictor variable.

The second Linear Model, LM #2 gave us an R-squared value of 0.88, an improvement from LM #1 that had 0.85 R-squared, while having the p-value less than 0.01. (See appendix for LM #1 Summary)

The below table presents the VIF scores of all predictor variables in the dataset. The Linear Model (Linear Model #2, LM #2) ran with all predictor variables modeled against the response variables show an increase in VIF scores of variables that we selected and used in LM #1 in Table 2. When we look at Table 3 below, we see that SLB got the highest VIF score of 3.2576 followed by GS with 3.1908 and XOM with 2.9241. On the other hand, the variables with the lowest VIF scores are MPC, DPS, and PEP with 1.3762, 1.5244 and 1.7198, respectively. ((See appendix for LM #2 Summary)

BAC	GS	JPM	WFC	BHI	CVX	DD	DOW	DPS	HAL
2.5581	3.1908	2.8445	2.5288	2.6035	2.9097	2.4327	1.9620	1.5244	2.9022

HES	HON	HUN	KO	MMM	MPC	PEP	SLB	XOM
2.0957	2.4470	1.7213	1.9675	2.6704	1.3762	1.7198	3.2576	2.9241

Table 2: VIF scores of all variables in Stock Return Dataset

The VIF scores we obtained for the predictor variables through Linear Model #1 and #2 do not present any alarming results as we do not see any VIF scores that are too close to 1, which indicate weak or none multicollinearity and any value that is too close or above five that would indicate strong multicollinearity. Moving forward, these VIF scores show us that SLB, GS, and XOM are likely to correlate with quite a few variables in the dataset in addition to VV, and MPC, DPS, and PEP are likely to correlate with few other variables in the dataset including VV. For model and result accuracy, these variables should be in our watch to not skew or distort any outcome.

Principal Component Analysis *(Questions #5 and #6)*

The plot of loadings of the initial two principal components (PC1 and PC2) of our principal component analysis present clusters that we expected to see through the outputs we observed in the previous section. Shown in Figure 3, the predictor variables with low VIF scores such as DPS, PEP, and KO, are spread further away from the cluster ranging between 0.20 and 0.10 in PC1 and between -0.4 and -0.6 in PC2. Similarly, we see that the variables that had high VIF scores in the previous section, such as GS, SLB, and WFC, are clusters on the other side of the spectrum around 0.25 in PC1 and 0.2 in PC2. So, per the loadings plots for PC1 and PC2, we see that the weights of variables with high VIF scores are higher than those with a low VIF score.

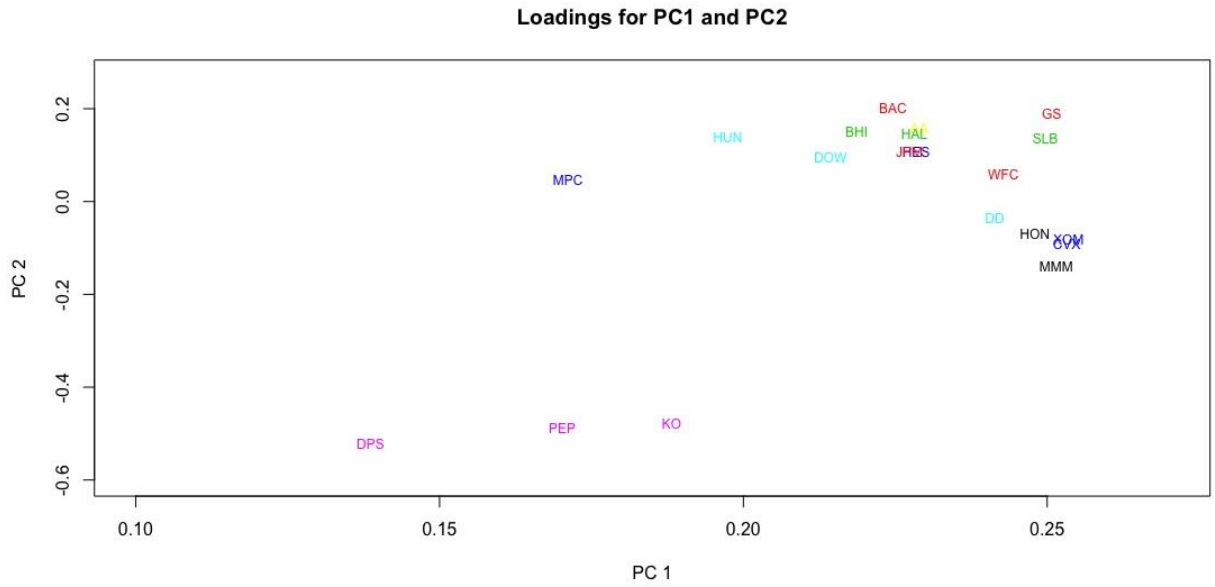

Figure 3: Loadings for PC1 and PC2

Ticker	Name	Industry	
AA	Alcoa Aluminum	Industrial - Metals	**7**
BAC	Bank of America	Banking	**2**
BHI	Baker Hughes Incorprated	Oil Field Services	**3**
CVX	Chevron	Oil Refining	**4**
DD	Dupont	Industrial - Chemical	**5**
DOW	Dow Chemical	Industrial - Chemical	**5**
DPS	DrPepper Snapple	Soft Drinks	**6**
GS	Goldman Sachs	Banking	**2**
HAL	Halliburton	Oil Field Services	**3**
HES	Hess Energy	Oil Refining	**4**
HON	Honeywell International	Manufacturing	**1**
HUN	Huntsman Corporation	Industrial - Chemical	**5**
JPM	JPMorgan Chase	Banking	**2**
KO	The Coca-Cola Company	Soft Drinks	**6**
MMM	3M Company	Manufacturing	**1**
MPC	Marathon Petroleum Corp	Oil Refining	**4**
PEP	Pepsi Company	Soft Drinks	**6**
SLB	Schlumberger	Oil Field Services	**3**
WFC	Wells Fargo	Banking	**2**
XOM	Exxon-Mobile	Oil Refining	**4**
VV	Vanguard Large Cap Index	Market Index	**9**

Table 3: Industry, Company Name and Ticker of variables

Per Table 3, the variables that had the lowest VIF score that was also in the lower part of the spectrum in our PC1 and PC2 loadings plot all belong to Soft Drinks industry. This shows that the Soft Drink industry companies' returns show low multicollinearity, whereas Banking industry companies' returns show moderate multicollinearity and, therefore, higher VIF scores.

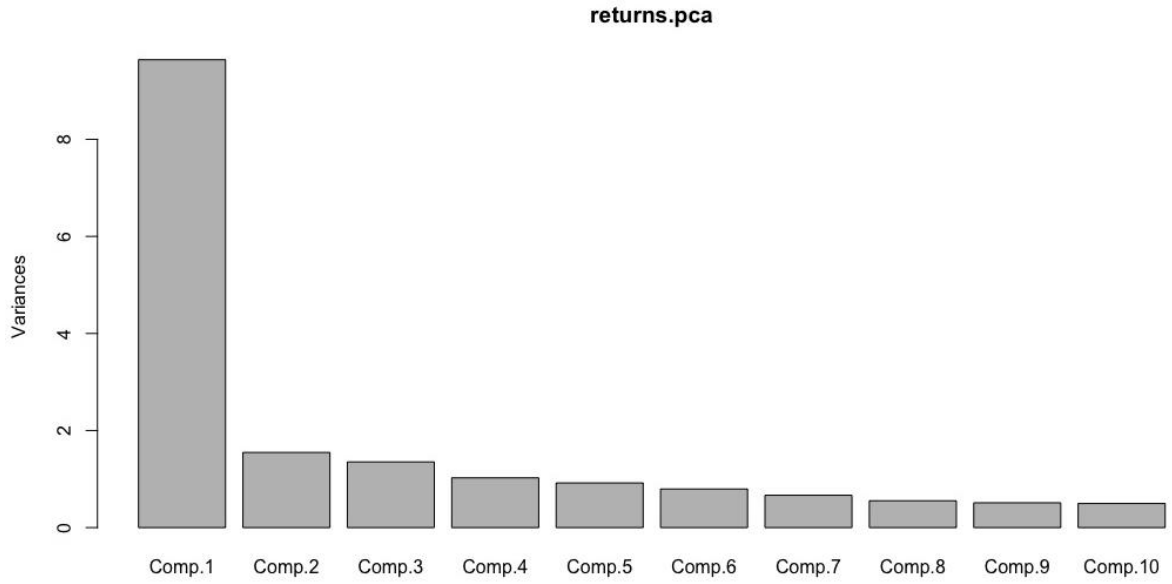

Figure 4: Scree Plot in Bar Chart Format

Scree plot plots Eigenvalue against Principal Component, and Eigenvalue gives us the directional variance in each component. By looking at both Figure 4 and Figure 5, we can see that variance is exceptionally high in Component 1 yet decrease significantly moving from Comp 1 to Comp 2, then incrementally to Comp 21. Both charts suggest that Component 1 explains a considerable amount of the stock returns dataset variance. To determine the number of components, we will keep through our PCA; we look at the 'elbow' in both charts. Both Scree plots in Figures 4 and 5 indicate that we can capture 80 percent of the dataset's variance and information by using the first eight components.

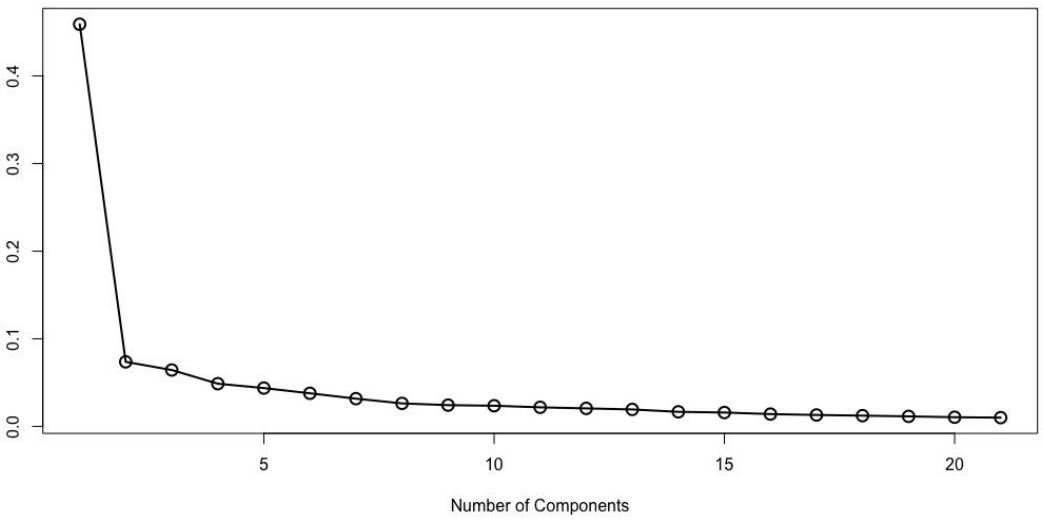

Figure 5: Scree Plot

After Comp 8 we see minimal addition to the information and variance in our dataset; thus we accept Comp 8 as the 'elbow' and keep these components moving forward. In parallel with the scree plots, the chart of variance per component in Figure 6 supports our decision to keep the initial 8 components as 80 percent of the variance captured is sufficient for our analysis.

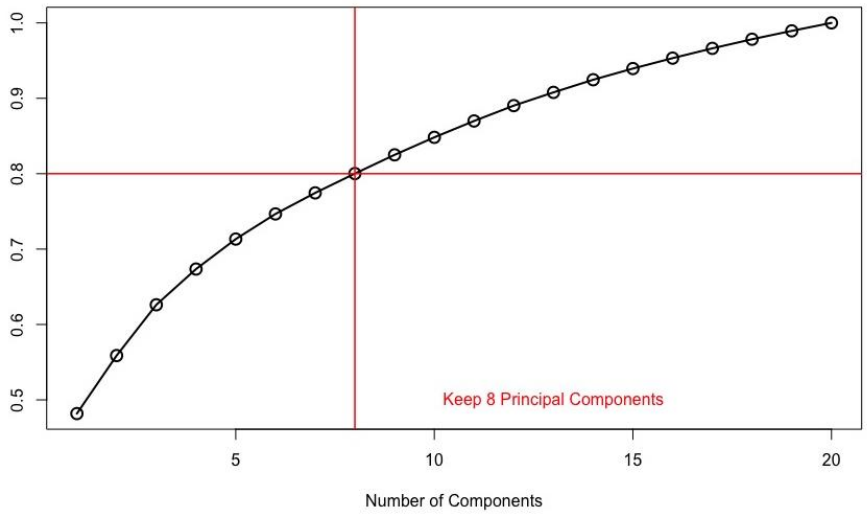

Figure 6: Total Variance Explained

Predictive Modelling with PCA Results *(Questions #7, #8 and #9)*

The Linear Regression model that was run with Components 1 through 8 that we obtained from our principal component analysis shows statistical significance (p-value <0.01) for the first three component as well as the overall model. In addition, through the 343 observations used through the training set, we see that the model achieved an R-squared value of 0.87, suggesting that 87 percent of the dataset variance can be explained through these eight components (see Figure 7).

PCA Model 1

	Dependent variable:
	VV
	pca1.lm
Constant	0.001***
	(0.0001)
Comp.1	0.002***
	(0.0000)
Comp.2	-0.001***
	(0.0001)
Comp.3	-0.001***
	(0.0001)
Comp.4	0.0002
	(0.0001)
Comp.5	0.0002
	(0.0002)
Comp.6	-0.0001
	(0.0002)
Comp.7	-0.0002
	(0.0002)
Comp.8	-0.0004**
	(0.0002)
Observations	343
R^2	0.87
Adjusted R^2	0.86
Residual Std. Error	0.003 (df = 334)
F Statistic	273.37*** (df = 8; 334)
Note:	*p<0.1; **p<0.05; ***p<0.01

Figure 7: Summary Statistics of PCA1 Linear Reg. Model

Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8
1.002989	1.003828	1.004474	1.006376	1.007698	1.005468	1.007198	1.003799

Table 4: VIF Score of each Component in PCA1 LM

Table 4 presents the VIF score of each of the eight components we used in the PCA1 LM. Per the results, we can see that VIF scores range roughly between 1.002 and 1.008, peaking at Component 5. We do not observe any high VIF score that suggests multicollinearity that we did with our predictor variables previously, yet we see that all components do have multicollinearity even if it is weak. So, even though we did not get the exactly identical VIF score for each Component, the results are incredibly close and not concerning.

The Mean Absolute Error (MAE) is a measure of errors between two datasets, in our case, between train and test datasets, and our original Stock dataset. This will show us (a) the extent of the errors between these two data sets and the stock data and (b) which of them performed better. The PCA1 LM that was conducted with our training dataset gave an MAE of 0.001983, and the model conducted with the test dataset gave an MAE of 0.002177. These results are not concerning. It is expected that the training dataset has a lower MAE value than the test data MAE as the model recognizes the training data more, thus resulting in higher accuracy.

To compute and compare our PCA regression models' in-sample and out-of-sample predictive accuracy, we will conduct the same LM as Model 1 and Model 2 with our PCA train/test datasets.

PCA LM 1 & 2

	Dependent variable:	
	VV	
	PCA LM #1	PCA LM #2
	(1)	(2)
Constant	0.0001	0.0001
	(0.0002)	(0.0001)
BAC		0.02**
		(0.01)
GS	0.05***	0.02
	(0.02)	(0.02)
DD	0.07***	0.03
	(0.02)	(0.02)
DOW	0.04***	0.04***
	(0.01)	(0.01)
DPS		0.03*
		(0.02)
HAL		-0.01
		(0.01)
HES		0.004
		(0.01)
HON	0.13***	0.10***
	(0.02)	(0.02)
HUN	0.03***	0.02***
	(0.01)	(0.01)
JPM	0.08***	0.04**
	(0.02)	(0.02)
WFC		0.08***
		(0.02)
BHI		0.01
		(0.01)
CVX		0.06**
		(0.03)
KO	0.14***	0.11***
	(0.02)	(0.02)
MMM	0.13***	0.12***
	(0.03)	(0.03)
MPC		0.01*
		(0.01)
PEP		0.02
		(0.02)
SLB		0.06***
		(0.02)
XOM	0.14***	0.05*
	(0.02)	(0.03)
Observations	343	343
R^2	0.85	0.88
Adjusted R^2	0.85	0.87
Residual Std. Error	0.003 (df = 333)	0.003 (df = 323)
F Statistic	214.70*** (df = 9; 333)	120.63*** (df = 19; 323)
Note:		*p<0.1; **p<0.05; ***p<0.01

Figure 8: PCA LM #1 and #2 Summary

The Linear Model conducted with PCA train dataset (343 observations instead of 502 in the raw dataset) shows extreme similarities with the same linear models conducted with the raw dataset in 'Naïve Linear Regression Modelling' Section. (See results of the referred model in the appendix) Presented in Figure 8, the R-squared of both PCA Linear Models are identical with raw Linear Models, and the difference in the value of coefficients is extremely minimal. Let's now compare the MAE for each of the models we ran in the PCA section, PCA Linear Model with 8 PCA Components (PCA1 LM), Linear Model with PCA train data and same variables as LM #1 (PCA LM#1), and Linear Model with PCA train data and same variables as LM #2 (PCA LM#2).

		PCA1 LM	**PCA LM #1**	**PCA LM #2**
Mean Absolute Error (MAE)	Train	0.001983	0.002037	0.001895
	Test	0.002074	0.002561	0.002216

Table 5: MAE Value of each PCA LM Model

Upon reviewing Table 5, we see that PCA LM #1 has the highest MAE, meaning error in values between the train/test dataset and actual dataset, compared to the other two models also on the table. As we eliminate PCA LM #1we now focus on PCA1 LM, we had Components 1 through 8 as our independent variables and PCA LM #2 where we used all predictor variables and their associated information dataset as independent

variables. While PCA LM #2 performs best, the meaning has the lowest MAE, when the only training dataset is taken to account but performs moderately when the test dataset is used instead. PCA1 LM shows the opposite result, having the best MAE for the test dataset, yet has a moderate MAE with the training dataset. Upon this analysis, we would pick PCA1 LM because it has the least difference between the train and test MAE, suggesting that it is the best fit compared to the other two linear models in Table 5.

Finally, in contrary to the process we followed so far where we conducted the PCA followed by Linear Regression Models, as the last step in this section we will be conducting a Backward Linear Regression Model (BLRM) to double check the correctness of our PC selections. The BLRM will be conducting the Linear Modelling first, then followed by a PCA where it recommends the number of components to be chosen.

The BLRM with all predictor variables as independent variables has given us results that are no in parallel with our previous findings. The Backward Linear Model has 'recommended' is that we keep 11 Components instead of 8 as we previously did. The BLRM, instead of consecutive components, chose a more varied range of components from 20. While we do not see a drastic change in the lm statistics summary (please see the appendix), with BLRM we observe overall higher VIF values for its chosen components than we did in our previous models.

Comp.1	Comp.2	Comp.3	Comp.5	Comp.8	Comp.10	Comp.11	Comp.12
1.01008	1.00972	1.00580	1.01077	1.00759	1.00938	1.01175	1.01525

Comp.16	Comp.18	Comp.19
1.01141	1.00718	1.00588

Table 6: VIF Value of each Component per BLRM

Compared to our previous models, the VIF values of the components are higher, indicating more multicollinearity than our previous attempts. When the MAE for BLRM assessed against the MAE for train and test data that we obtained through our previous models, we see that the BLRM had the lowest train MAE and a mid-spectrum test MAE. While these are tangible results and BLRM performed better than PCA LM#1 and PCA LM#2, the model with the best train and test MAE remains PCA1 LM. So, with this outcome, we can state that BLRM with 11 components instead of 8 in PCA1 LM did not generate a worthy improvement in model performance.

		PCA1 LM	PCA LM #1	PCA LM #2	Backward LM
Mean Absolute Error (MAE)	Train	0.001983	0.002037	0.001895	0.001831
	Test	0.002074	0.002561	0.002216	0.002109

Table 7: MAE Value for All Models

Stochastic Neighbor Embedding (t-SNE) Analysis *(Questions #10)*

Opposite to Principal Component Analysis' linear transformation, dimensional reduction, and mapping, Stochastic Neighbor Embedding, aka t-SNE, offers a nonlinear and high dimensional approach to data exploration, visualization, and dimensional reduction. The main goal of running a t-SNE analysis on the Stock dataset is to visualize our data embedded in a lower number of dimensions, 2 or 3, to determine patterns and trends.

The t-SNE models were run multiple times with various perplexity values, learning value, and dimensions, both 2 and 3. Figure 9 and Figure 10 present four plots, each where different perplexity and learning values were tested. In both attempts, we chose to have 5000 iterations with the aim of high accuracy and a default dim of 2. The perplexity values we are testing are 1, 2, 3, 4, and 5. Due to our dataset's size, we should not require a high perplexity value and the t-SNE plots with the listed perplexity values re-iterate this approach as perplexity above 5 has caused errors in our analysis.

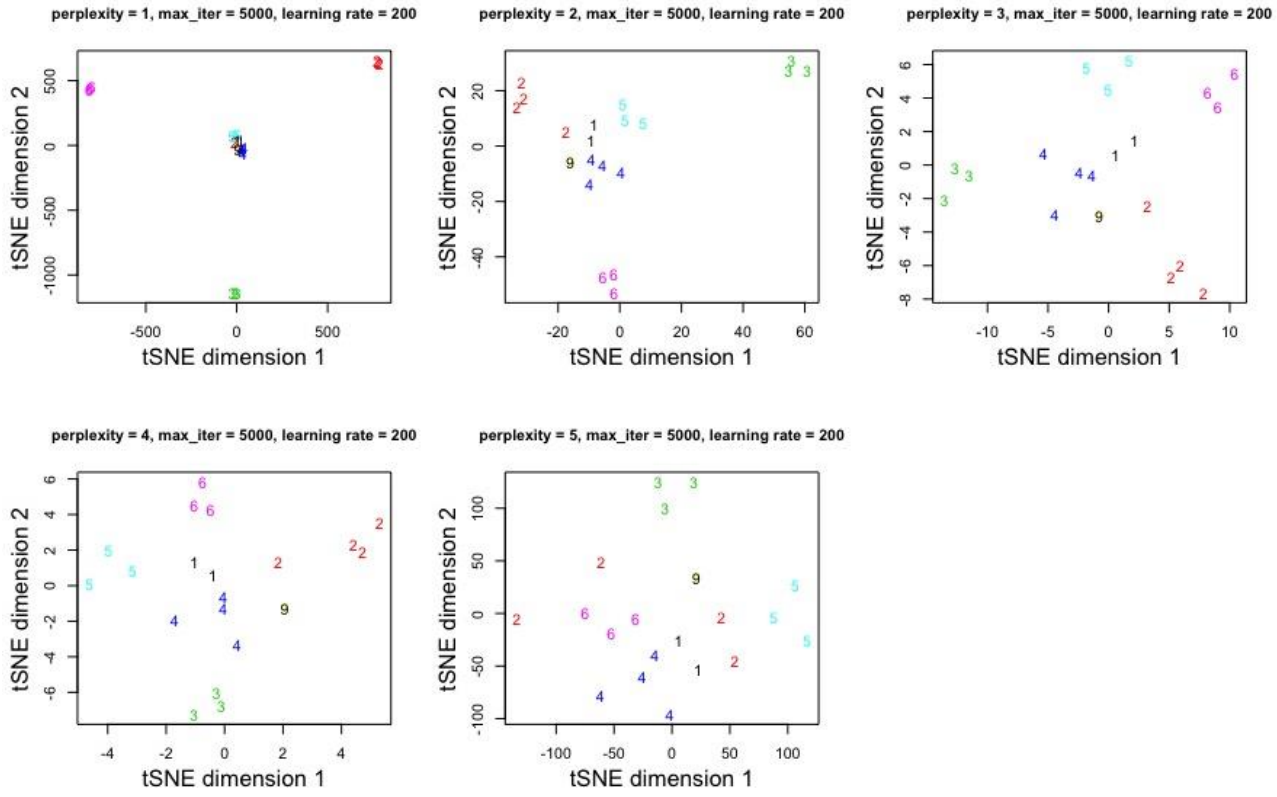

Figure 9: t-SNE Plots with Perplexity Values 1, 2, 3, 4 and 5

Figure 9 above presents the t-SNE plots with perplexity value from 1 through 5, with a fixed learning value of 200, max iterations of 5000 and dim of 2. From the plots, in Figure 9, we can see that perplexity value between 2 and 4 is ideal for our t-SNE as the model is able to differentiate the data by industry.

The impact of the changing learning rate is presented in Figure 10 below. Upon reviewing the t-SNE plots with changing learning value and fixed perplexity value of 3, we determined that the average of our ideal range in the previous paragraph determined that the learning value between 100 and 1000 all shown viable plots. With the learning value of 2000, we see that the model struggles to differentiate the industries in the dataset while with learning values 100, 200, 500, and 1000, it can.

Through our various plots of t-SNE with changing perplexity and learning value, we conclude that for our dataset, perplexity of 3 and learning rate of 200 give us the most optimal outcome.

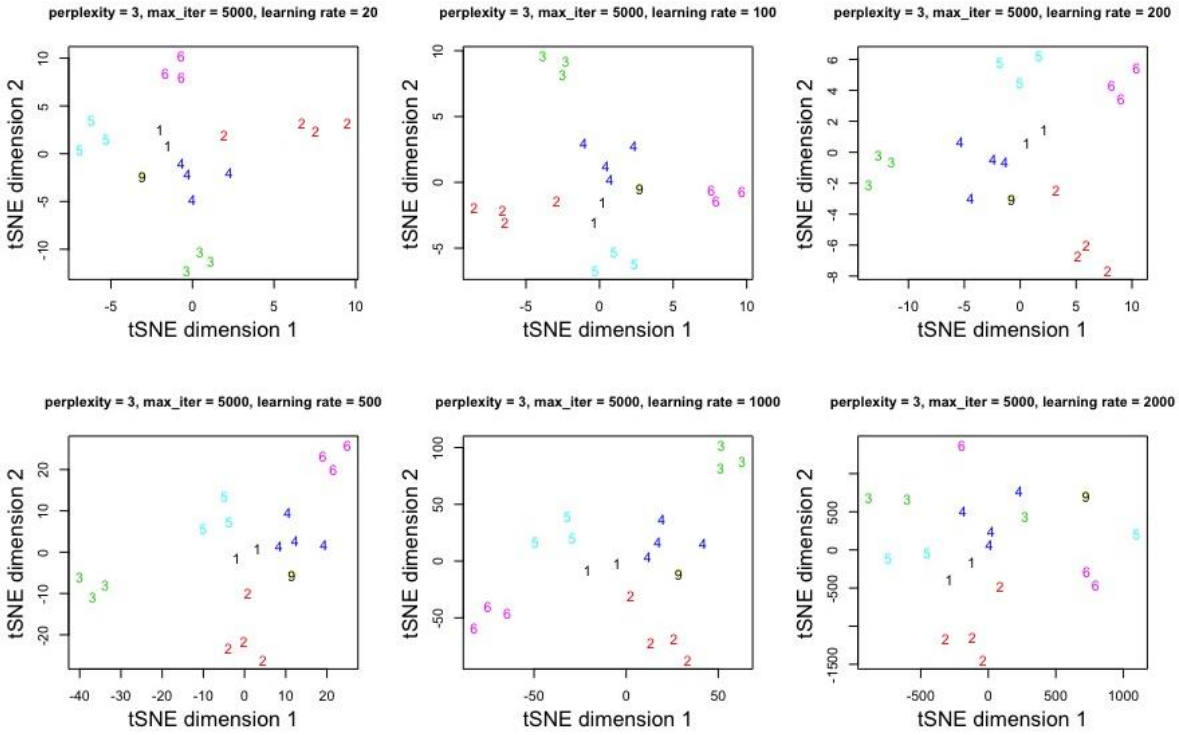

Figure 10: t-SNE Plots with Learning Values 20, 100, 200, 500, 1000 and 2000

Finally, in Figure 11, we clearly see neighboring points that our model identified in the high dimensional space, which then transferred to 'low' dimensional space, the 2D plane. Through this, we are able to determine underlying patterns and neighboring. The t-SNE plot in Figure 11 suggests that the data points in our dataset that is of industries 3, 5, 6, and 2 are relatively more distinguishable, whereas the data points for industries 1, 4, and 9, and in one case, industry 2, are scattered more near with each other than the rest.

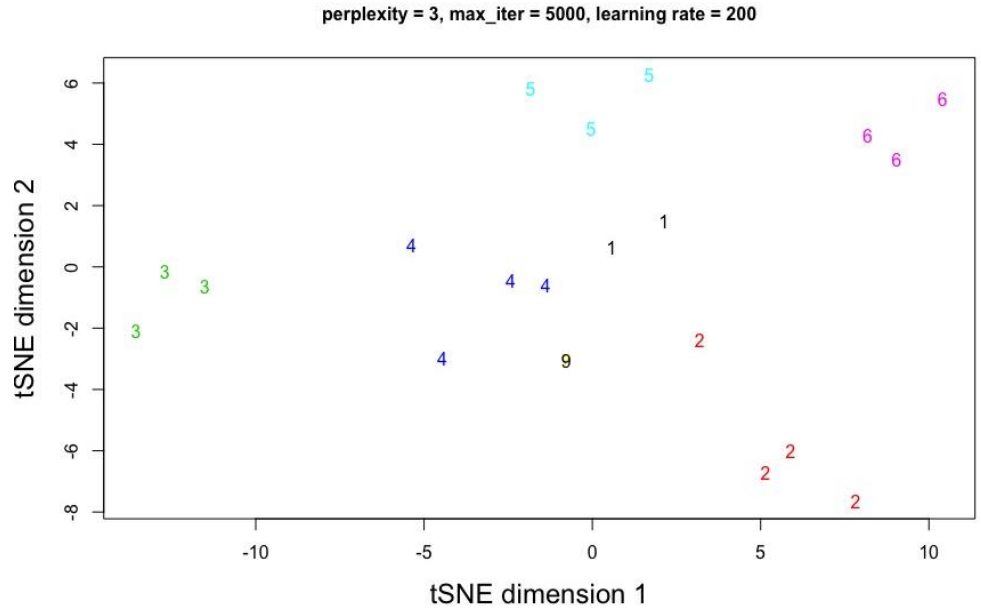

Figure 11: t-SNE Plots with Learning Value of 200 and Perplexity of 3

Comparison *(Questions #11)*

As we conducted both a Principal Component Analysis, PCA, and a Stochastic Neighbor Embedding, t-SNE, we were able to run a mathematical linear dimensionality reduction with PCA and probabilistic high dimensional reduction through t-SNE. While both models are good for a dimensional reduction in one form or another, the process of conducting them and the results that we achieve through both are not entirely comparable. As PCA is mathematical, it gives us a more quantitative understanding of the underlying relationship, patterns, trends, and neighbors in the dataset. In contrast, with t-SNE, due to its probabilistic nature, we are able to generate t-SNE plots with numerous variations in perplexity, learning level, number of iterations, and dimensions, which aids us to visualize the neighboring in our dataset.

Our analysis has given us useful insights into the core patterns and trends in the Stock dataset. PCA has provided us with a more quantitative understanding of multicollinearity. It is a linear dimensional reduction model, and t-SNE has shown us the clustered industries and their neighbors. Our exploration with these two unsupervised learning methods shown us that in an ideal scenario we would use both models to understand the underlying characteristics of a large dataset and that it is not feasible to directly compare their outputs as they essentially provided different yet both valuable insights.

Reflection

Assignment #1 has challenged me as this was my first true introduction to both of the unsupervised learning methods, PCA, and t-SNE. The Skeleton R Code has helped drastically, and interpreting results have given a great deal of understanding on both of these concepts/models, yet I find PCA and t-SNE to be extensive subjects that I personally need to further study and practice. After this assignment, my impression is that both models conducted in this assignment are crucial and complement each other; therefore, both should be performed if we want to get a thorough initial understanding of a large and complex dataset.

APPENDIX

LM Model #1 and #2

	Dependent variable:	
	VV	
	LM Model #1	LM Model #2
	(1)	(2)
Constant	0.0001	0.0001
	(0.0001)	(0.0001)
BAC		0.03***
		(0.01)
GS	0.08***	0.04***
	(0.01)	(0.01)
DD	0.04**	0.01
	(0.02)	(0.02)
DOW	0.04***	0.04***
	(0.01)	(0.01)
DPS		0.06***
		(0.01)
HAL		-0.001
		(0.01)
HES		0.005
		(0.01)
HON	0.14***	0.11***
	(0.02)	(0.02)
HUN	0.04***	0.03***
	(0.01)	(0.01)
JPM	0.05***	0.02
	(0.01)	(0.01)
WFC		0.08***
		(0.02)
BHI		0.02
		(0.01)
CVX		0.06***
		(0.02)
KO	0.14***	0.09***
	(0.02)	(0.02)
MMM	0.13***	0.11***
	(0.02)	(0.02)
MPC		0.01
		(0.01)
PEP		0.02
		(0.02)
SLB		0.05***
		(0.01)
XOM	0.15***	0.06***
	(0.02)	(0.02)
Observations	501	501
R^2	0.85	0.88
Adjusted R^2	0.85	0.88
Residual Std. Error	0.003 (df = 491)	0.003 (df = 481)
F Statistic	313.48*** (df = 9; 491)	187.80*** (df = 19; 481)
Note:		*p<0.1; **p<0.05; ***p<0.01

Backward Linear Model

	Dependent variable:
	VV
	Backward LM
Constant	0.001^{***}
	(0.0002)
Comp.1	0.002^{***}
	(0.0000)
Comp.2	-0.0005^{***}
	(0.0001)
Comp.3	-0.001^{***}
	(0.0001)
Comp.5	0.0003^{*}
	(0.0002)
Comp.8	-0.0003
	(0.0002)
Comp.10	-0.0004^{*}
	(0.0002)
Comp.11	0.001^{***}
	(0.0002)
Comp.12	0.0003
	(0.0002)
Comp.16	-0.001^{*}
	(0.0003)
Comp.18	0.0005
	(0.0003)
Comp.19	-0.001
	(0.0003)
Observations	322
R^2	0.88
Adjusted R^2	0.87
Residual Std. Error	0.003 (df = 310)
F Statistic	204.66^{***} (df = 11; 310)
Note:	*p<0.1; **p<0.05; ***p<0.01

The R Code for Assignment #1

```
#set directory and import data
getwd()
setwd('/Users/serrauzun/Desktop/MSDS_411_Unsupervised/Assignment #1')
install.packages("readxl",repos = "http://cran.us.r-project.org")
library(readxl)
install.packages("readxl",repos = "http://cran.us.r-project.org")
install.packages("tidyverse",repos = "http://cran.us.r-project.org")
install.packages("Rtsne",repos = "http://cran.us.r-project.org")

library(readxl)
library(tidyverse)
library(Rtsne)

Stock.Data <- read_excel("/Users/serrauzun/Desktop/MSDS_411_Unsupervised/Assignment
#1/stockdata.xlsx")
raw.data <- read_excel("/Users/serrauzun/Desktop/MSDS_411_Unsupervised/Assignment
#1/stockdata.xlsx")

str(raw.data)
head(raw.data)
names(raw.data)
dim(raw.data)

str(Stock.Data)
head(Stock.Data)
names(Stock.Data)
dim(Stock.Data)


# Note Date is a string of dd-Mon-yy in R this is '%d-%B-%y';
Stock.Data$RDate <- as.Date(Stock.Data$Date,'%d-%B-%y');
sorted.df <- Stock.Data[order(Stock.Data$RDate),];
head(sorted.df)

AA <- log(sorted.df$AA[-1]/sorted.df$AA[-dim(sorted.df)[1]]);
head(AA)
# Manually check the first entry: log(9.45/9.23)
# Type cast the array as a data frame;
returns.df <- as.data.frame(AA);
str(returns.df)
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]]);
str(returns.df)

# Compute correlation matrix for returns;
returns.cor <- cor(returns.df)
corr_ <- as.data.frame(returns.cor[,c('VV')])


# Barplot the last column to visualize magnitude of correlations;
par(mfrow = c(1,1))
barplot(returns.cor[1:20,c('VV')],las=2,ylim=c(0,1.0), col = "light gray")
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, repos = "http://cran.us.r-project.org")

require(corrplot)
par(mfrow = c(1,1))
#corrplot(returns.cor, tl.cex = 0.8, tl.col = "black")
corrplot(returns.cor, tl.cex = 0.8, cl.cex = 1, tl.col = "black")

# load car package
require(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 ~
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)

install.packages('stargazer',repos = "http://cran.us.r-project.org")
library(stargazer)



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

# check the range of the components
```

```
range(pc.1)
range(pc.2)

dev.off()
plot(-10,10,type='p',xlim=c(0.10,0.27),ylim=c(-0.6,.27),xlab='PC 1',ylab='PC 2', main = "Loadings for
PC1 and PC2")
text(pc.1,pc.2,labels=names(pc.1),cex=0.75, col = stock_data_industry$IndustryCode)

pc.1 <- returns.pca$loadings[,1];
pc.2 <- returns.pca$loadings[,2];
names(pc.1)



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

# Make Scree Plot
scree.values <- (returns.pca$sdev^2)/sum(returns.pca$sdev^2);
scree.values
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);
variance.values
#80 percent of the variance is covered in first 8 components

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')

# 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)

install.packages('stargazer',dependencies=TRUE, repos = "http://cran.us.r-project.org")
library(MASS)
library(stargazer)
out.path <- '/Users/serrauzun/Desktop/MSDS_411_Unsupervised/Assignment #1/'

file.name <- 'pca1.lm .html';
stargazer(pca1.lm, type=c('html'),out=paste(out.path,file.name,sep=''),
          title=c('PCA Model 1'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE,
          column.labels=c("pca1.lm"), intercept.bottom=FALSE )

# 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));


pca1.mae.train
pca1.mae.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.1 on train data set and 'test' on test data;
model.1_pca <- lm(VV ~ GS+DD+DOW+HON+HUN+JPM+KO+MMM+XOM, data=train.returns)
model1_pca.mae.train <- mean(abs(train.returns$VV-model.1_pca$fitted.values));
model1_pca.test <- predict(model.1_pca,newdata=test.returns);
model1_pca.mae.test <- mean(abs(test.returns$VV-model1_pca.test));


# Fit model.2 on train data set and 'test' on test data;
model.2_pca <- 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_pca.mae.train <- mean(abs(train.returns$VV-model.2_pca$fitted.values));
model2_pca.test <- predict(model.2_pca,newdata=test.returns);
model2_pca.mae.test <- mean(abs(test.returns$VV-model2_pca.test));
summary(model.2)
```

```
file.name1 <- 'model 1 and 2.lm .html';
stargazer(model.1_pca,model.2_pca, type=c('html'),out=paste(out.path,file.name1,sep=''),
          title=c('PCA LM 1 & 2'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE,
          column.labels=c("PCA LM #1","PCA LM #2"), intercept.bottom=FALSE )

#MAE comparison
pca1.mae.train
model1_pca.mae.train
model2_pca.mae.train

pca1.mae.test
model1_pca.mae.test
model2_pca.mae.test

# remove u

train.scores <- train.scores[c(-22)]

# Fit full.lm on PCA scores of train data
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));

file.name2 <- 'backward.html';
stargazer(backward.lm, type=c('html'),out=paste(out.path,file.name2,sep=''),
          title=c('Backward Linear Model'),
          align=TRUE, digits=2, digits.extra=2, initial.zero=TRUE,
          column.labels=c("Backward LM"), intercept.bottom=FALSE )

vif(backward.lm)

backward.test <- predict(backward.lm,newdata=test.scores);
backward.mae.test <- mean(abs(test.scores$VV-backward.test))

pca1.mae.train
model1_pca.mae.train
model2_pca.mae.train
backward.mae.train

pca1.mae.test
model1_pca.mae.test
model2_pca.mae.test
backward.mae.test

##############
##############

# t-Distributed Stochastic Neighbor Embedding [t-SNE]
```

```
stock_data_industry <- read_excel("/Users/serrauzun/Desktop/MSDS_411_Unsupervised/Assignment
#1/stock_data_industry.xlsx")


############################
# Data load and prep
############################
return.df_tsne <- returns.df[,c(1:20)]
colnames(return.df_tsne)

tsne_transpose <- as.data.frame(t(as.matrix(return.df_tsne)))
dim(tsne_transpose)
colnames(tsne_transpose)

############################
# Exploratory Data Analysis
# Data Prep
############################

# review range of variables and ensure no N/As exist
summary(tsne_data)
sapply(tsne_transpose, function(x) sum(is.na(x)))


tsne_plot <- function(perpl=3,iterations=5000,learning=200){
  set.seed(1) # for reproducibility
  tsne <- Rtsne(tsne_transpose , dims = 2, perplexity=perpl, verbose=TRUE, max_iter=iterations,
eta=learning)
  plot(tsne$Y, t='n', main = print(paste0("perplexity = ",perpl, ", max_iter = ",iterations, ", learning rate =
",learning)), xlab="tSNE dimension 1", ylab="tSNE dimension 2", cex.main=1, cex.lab=1.5)
  text(tsne$Y, labels = stock_data_industry$IndustryCode, col = stock_data_industry$IndustryCode)
}

par(mfrow = c(2,3))
perplexity_values <- c(1,2,3,4,5)
sapply(perplexity_values, function(i){tsne_plot(perpl=i)})

par(mfrow = c(2,3))
learning_values <- c(20,100,200,500,1000,2000)
sapply(learning_values,function(i){tsne_plot(learning=i)})

par(mfrow = c(1,1))
learning_values <- c(200)
sapply(learning_values,function(i){tsne_plot(learning=i)})
```