Assignment #1: Principal Components Analysis and Stochastic Neighbor Embedding

Harini Anand

1. Introduction

Data for financial assets such as stocks, bonds, commodities, currencies, and cash tend to be highly correlated (that is, they move in relation to each other). For example, the performance of small-cap stocks relative to large-cap stocks is correlated by the overall market sentiment.

In this report, we present the results of dimension reduction conducted on the stock portfolio data using the Principal Component Analysis (PCA) and T-distributed Stochastic Neighbor Embedding (t-SNE) methods. Principal component analysis (PCA) is a statistical method of dimension reduction that allows us to understand if there are a small number of uncorrelated parts of the data that can explain a large part of the data. It is used as a remedial measure for multicollinearity in linear regression. T-distributed Stochastic Neighbor Embedding approaches the dimension reduction using a nonlinear probabilistic approach. The results of dimension reduction from the two methods were compared and contrasted.

In addition to that, the principal components obtained using the PCA method were used to predict the value of the Vanguard large-cap index fund using the linear regression method. The Predictive modeling framework with the 70/30 train/test split of the component score data was used to perform the fitting. The accuracy of this model obtained through the fitting of the principal components is compared with two naïve models (a small model and a full model). Furthermore, the backward variable selection method (automatic variable selection) was used to select the best principal components, and the accuracy of that model is compared with all the previous models.

2. Data

The dataset used for this analysis is the stock portfolio data. It contains the time-series data of daily closing stock prices for twenty stocks and a large-cap index fund from Vanguard from 3-Jan-2012 to 31-Dec-2013 for 21 stocks (502 observations). The input dataset has a total of 43 attributes. These include the date, the daily closing stock values, and the return values for the 21 stocks. The daily closing stock and the return values are continuous values.

3. Data preparation

As part of the data preparation, the date attribute in the raw dataset was transformed to the R Date format (YYYY-mm-dd). The dataset was then sorted using the transformed date. We then generated log-returns of the individual stocks. Log-return for a stock is obtained by taking the natural logarithm of the future value to the present value. Since stock prices behave similar to exponential functions, to make them close to a normally distributed variable, log returns are used, which have more stable distributions than the arithmetic returns. Code Snippet 01 in section 13.1 contains the R code for data preparation.

4. Exploratory data analysis

In this section, we present the results of the exploratory data analysis conducted by examining the correlations among all log-return values of the stocks. For that, we computed the full correlation matrix using all the log-return stock variables, including the VV index fund. From the full correlation matrix, **we extracted the last column, which contained the correlation coefficients of the log-return**

stock variables computed against the log-return values of the VV index fund (shown in Table 01). The same is also plotted using as a bar graph in FIG 01.

Log-return variable	Correlation Coefficient
AA	0.6324106
BAC	0.6501877
BHI	0.5774988
CVX	0.7209041
DD	0.6895190
DOW	0.6264550
DPS	0.4435005
GS	0.7121620
HAL	0.5974989
HES	0.6107960
HON	0.7683784
HUN	0.5819449
JPM	0.6578478
KO	0.5997988
MMM	0.7608489
MPC	0.4731198
PEP	0.5075264
SLB	0.6928534
WFC	0.7335731
XOM	0.7211079
VV	1.0000000

Table 01: Correlation coefficient for each of the log-return stocks against the log-return VV index fund.

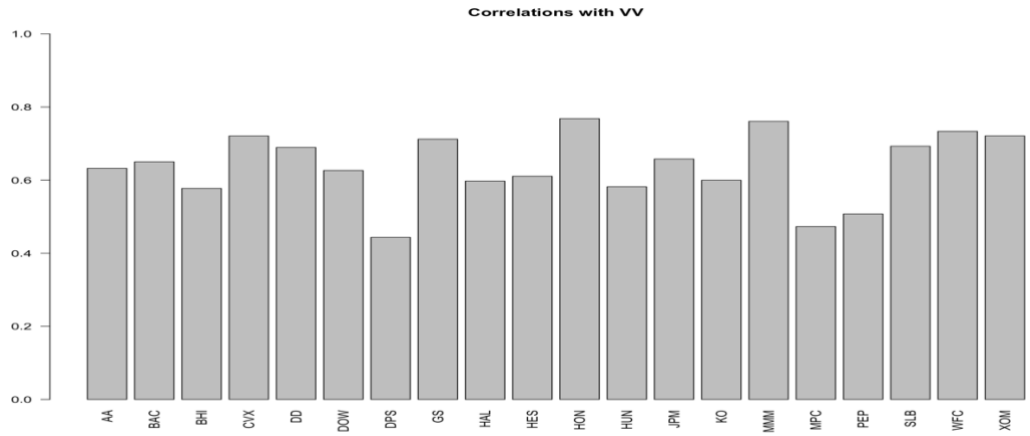

FIG 01: A plot of the correlation coefficients for each of the log-return stocks against the log-return VV index fund.

From Table 01 and FIG 01, we can gather that HON and MMM have the top two high correlation coefficients with respect to VV, with the values 0.7683784 and 0.7608489, respectively.

Corrplot Visualization:

To be able to view all the pairwise correlations in the data, we created a corrplot. From the corrplot, we can determine not only the pairwise correlations with the VV index fund, but also the correlation coefficients among the other stock variables. That is the advantage of the corrplot (FIG 02) over the simple bar chart (FIG 01). We can detect the multicollinearity by examining the corrplot for

correlation coefficients that are closers to -1 or +1. In the stock portfolio data, all the correlation coefficients are positive values.

DPS, MPC, and PEP are three variables that have low VIF values (most of the correlation coefficient are below or about 0.50).

SLB, WFC, and XOM (as part from VV) are some variables that have high VIF values (most of the correlation coefficients are above 0.50)

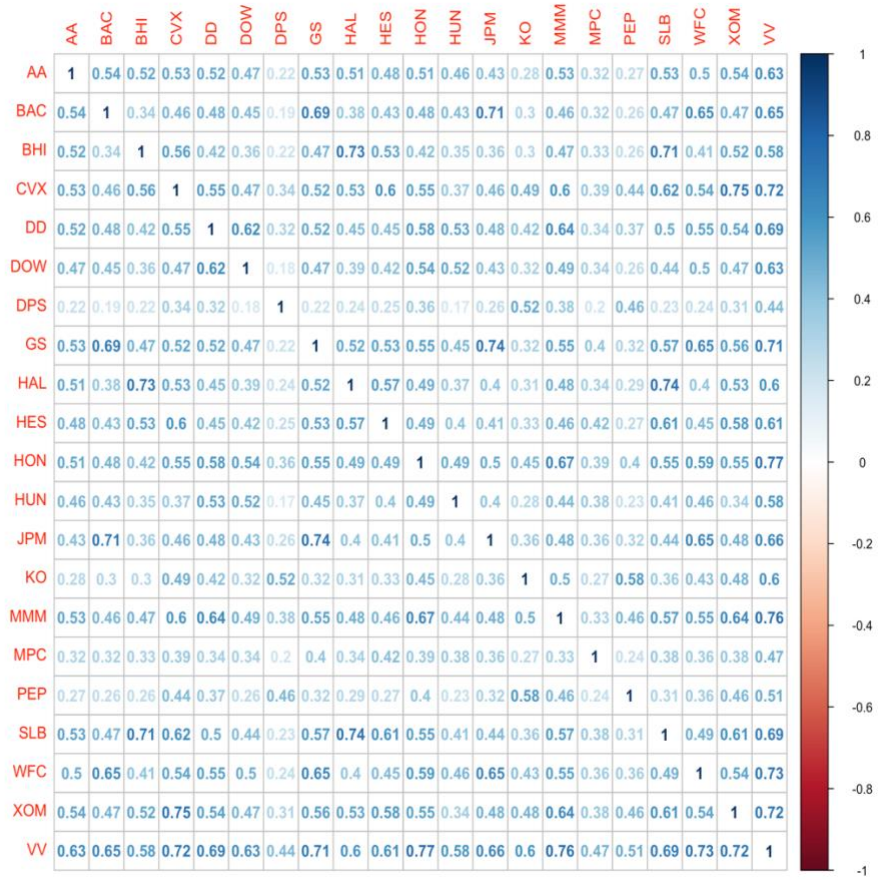

FIG 02: A plot of the correlation coefficients among all the log-return stock variables (including the log-return VV index fund)

A statistical graph is a pictorial representation to describe and quantify the characteristics of data (e.g., bar charts, pie charts, histograms, etc.) for statistical evaluation. On the other hand, data visualization is a broader term that is used to present data in a visual context for the exploration of the patterns, trends, outliers, and correlations in the data set.

Code Snippet 02 in section 13.2 contains the R code for EDA using statistical graphs and visualization tools.

5. Exploratory data analysis using models

In this section, we present two naïve models – a small model and a full model. The purpose of the two models is to compute the VIF values of all the variables for the examination of the multicollinearity among the variables.

The small model is obtained by fitting the log return stock values of the variables GS, DD, DOW, HON HUN, JPM, KO, MMM, XOM to predict the log return value of the VV index fund.

The full model is obtained by fitting all the log return stock values (namely AA, BAC, GS, JPM, WFC, BHI, CVX, DD, DOW, DPS, HAL, HES, HON, HUN, KO, MMM, MPC, PEP, SLB, and XOM) to predict the log return value of the VV index fund.

The fitting for the **small model** resulted in the following equation:

VV = 0.0001008 + 0.0784765 * GS + 0.0354057 * DD + 0.0406763 * DOW + 0.1449817 * HON + 0.0385118 * HUN + 0.0505123 * JPM + 0.1419686 * KO + 0.1336002 * MMM + 0.1480728 * XOM

Based on the summary table in Table 02, we can conclude at the level of significance of 0.001, we reject that the joint null hypothesis that all the coefficients have no effect on the overall regression. This is based on the p-value (with a corresponding F-statistic value of 313.5) of less than 2.2e-16. Also, it can be gleaned from Table 02 that all the individual regression coefficients but for the intercept and the coefficient for log return variable DD are strongly significant at the level of significance of 0.001 meaning we can reject the null hypothesis for these individual regression coefficients to be zero at the level of significance of 0.001. The regression coefficient for the log return variable DD is significant at a level of significance of 0.05. The intercept is NOT significant. The coefficient of determination (Adjusted R-squared) for the small model is 84.9% indicating that 84.9 percent of the variation in the log-return of VV index fund can be explained by variations in the predictor variables. Among the regression coefficients, HON, KO, MMM, and XOM provide considerably more increase to the response variable (VV) for a change unit to the variables, when compared to all the other variables.

Term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0001008	0.0001331	0.757	0.44929
GS	0.0784765	0.0138277	5.675	2.37e-08 ***
DD	0.0354057	0.0177154	1.999	0.046204 *
DOW	0.0406763	0.0116993	3.477	0.000552 ***
HON	0.1449817	0.0170837	8.487	2.53e-16 ***
HUN	0.0385118	0.0077371	4.978	8.93e-07 ***
JPM	0.0505123	0.0132262	3.819	0.000151 ***
KO	0.1419686	0.0176282	8.054	6.14e-15 ***
MMM	0.1336002	0.0239378	5.581	3.96e-08 ***
XOM	0.1480728	0.0213601	6.932	1.31e-11 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

Residuals:

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

Table 02: Regression summary table for the small model - lm(formula = VV ~ GS + DD + DOW + HON + HUN + JPM + KO + MMM + XOM, data = returns.df)

Next, we computed the VIF values for the predictor variables used in the small model. The relationship between the predictor variables can be judged by examining a quantity called the VIF (Variance Inflation Factor). We used a cutoff value of 3 to determine if there is any multicollinearity among the variables. VIF is related to the square of the multiple correlation coefficient (the coefficient of determination, R-squared value) that results when the predictor variable is regressed against all the other predictor variables**. A VIF value of 3 implies the coefficient of determination for the predictor variable when regressed against all the other predictor variables is 66.7%, meaning 66.7 percent of**

the variation in the predictor variable is explained by the other predictor variables (an indication of the presence of multicollinearity).

The VIF values for the predictor variables used in the small model in decreasing order are shown in Table 03.

Predictor Variable	VIF value
GS	2.705795
MMM	2.590177
DD	2.368257
JPM	2.324600
HON	2.261397
XOM	2.073721
DOW	1.919773
HUN	1.633336
KO	1.473202

Table 03: VIF values for the predictor variables in the small model.

Since none of the VIF values are greater than 3, there is no indication of multicollinearity among the predictor variables used in the small model.

The fitting for the **full model** results in the following equation:

VV = 0.00009953 + 0.01538 * AA + 0.02723 * BAC + 0.03434 * GS + 0.02224 * JPM + 0.07738 * WFC + 0.01604 * BHI + 0.05742* CVX + 0.01003 * DD + 0.036 * DOW + 0.05659 * DPS - 0.001976 * HAL + 0.004393 * HES + 0.1071 * HON + 0.02867 * HUN + 0.09425 * KO + 0.1093 * MMM + 0.01079 * MPC + 0.02092 * PEP + 0.04851 * SLB + 0.05797 * XOM

Table 04 shows the regression summary table for the fitted full model.

From Table 04, we can conclude that at a level of significance of 0.001, we can reject the joint null hypothesis that all the coefficients have no effect on the overall regression. This is based on the p-value (F-statistic is 179) of less than 2.2e-16. For the individual coefficient tests, the variables for the log returns of the stocks WFC, DOW, DPS, HON, HUN, KO, MMM, and SLB are strongly statistically significant at the level of significance of 0.001. So, at a level of significance of 0.001, we can reject the null hypothesis based on the test using Student's T-distribution that the individual regression coefficient for the above variables is zero.

The coefficients for BAC and CVX are significant at only at a level of significance of 0.01. The coefficients for GS and XOM are significant at a level of significance of 0.05. JPM is significant at a level of significance 0.1. The rest of the coefficients for the variables AA, BHI, DD, HAL, HES, MPC, PEP, and the intercept are NOT significant.

Also, from the summary table, Table 04, we can determine that the coefficient of determination for the full model is 87.68% indicating 87.68 percent of the variation in the log return of VV (Vanguard index fund) can be explained by the variations in all the predictor variables used in the full model.

Based on the coefficients, for a unit change in the predictor variables, HON and MMM result in more change in the response variable (VV) compared to all the other variables.

Term	Estimate	Std. Error t	Value	Pr(>|t|)
(Intercept)	0.00009953	1.21E-04	0.82	0.412433
AA	0.01538	1.04E-02	1.479	0.139894
BAC	0.02723	9.70E-03	2.808	0.005188 **
GS	0.03434	1.36E-02	2.529	0.011766 *
JPM	0.02224	1.33E-02	1.674	0.094849 .
WFC	0.07738	1.58E-02	4.897	1.33e-06 ***
BHI	0.01604	1.16E-02	1.382	0.167752
CVX	0.05742	2.07E-02	2.776	0.005720 **
DD	0.01003	1.63E-02	0.618	0.537144
DOW	0.036	1.07E-02	3.366	0.000824 ***
DPS	0.05659	1.49E-02	3.79	0.000170 ***
HAL	- 0.001976	1.21E-02	-0.163	0.870344
HES	0.004393	9.69E-03	0.453	0.650432
HON	0.1071	1.61E-02	6.658	7.62e-11 ***
HUN	0.02867	7.22E-03	3.969	8.31e-05 ***
KO	0.09425	1.85E-02	5.103	4.82e-07 ***
MMM	0.1093	2.20E-02	4.964	9.63e-07 ***
MPC	0.01079	7.02E-03	1.536	0.125134
PEP	0.02092	2.03E-02	1.028	0.304293
SLB	0.04851	1.45E-02	3.339	0.000905 ***
XOM	0.05797	2.30E-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

Residuals:

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

Table 04: Regression summary table for the full model 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)

After fitting the model, we computed the VIF (Variance Inflation Factor) values for all the predictor variables in the full model. The objective is to identify the multicollinearity among these variables**. If any of the VIF values are greater 3 (as explained in the case of the small model), it indicates the presence of multicollinearity among the variables**. Table 05 shows the VIF values sorted in the decreasing order for all the predictor variables in the full model.

In Table 05, **the variables SLB and GS have VIF values greater than 3**. **Therefore, there are concerns about multicollinearity in the data.**

We next used the statistical method of principal component analysis to reduce the dimensions of the data. This also helped to remediate the multicollinearity identified in the data.

Code Snippet 03 in section 13.3 contains the R code for EDA using models.

Predictor Variable	VIF value
SLB	**3.258982**
GS	**3.197811**
XOM	2.949826
CVX	2.920187
HAL	2.919924
JPM	2.875959
BAC	2.686535
MMM	2.684942
BHI	2.651368
WFC	2.532626
HON	2.455876
DD	2.441486
HES	2.09606
AA	2.02627
KO	1.981647
DOW	1.965886
HUN	1.743913
PEP	1.720663
DPS	1.525629
MPC	1.376706

Table 05: VIF values for all the predictor variables

6. Dimension reduction using the principal component analysis

In this section, we present the results of applying the principal component analysis on the log return values of the predictor variables representing the stock prices. The principal components are NOT standardized (i.e., scaling and mean centering) to mean zero and unit variance. However, in our case, the scale invariance is achieved by utilizing the log-return transformation, which has already been applied to the stock prices. The principal component calculation is done by directly using the ***princomp()*** R function with the ***cor*** logical value set to TRUE (meaning by using the correlation matrix).

The loading in the principal component analysis provides the correlation between a principal component and the original variable. It is weight by which each variable should be multiplied to get the principal component score.

Table 06 shows the loadings of the first and the second principal components obtained by using the **princomp()** function on the log-return data.

Predictor variable	PC1 loading value	PC2 loading value
AA	0.2289477	0.15741300
BAC	0.2246401	0.20081224
BHI	0.2186159	0.15017188
CVX	0.2532034	-0.09115728
DD	0.2414028	-0.03433672
DOW	0.2143027	0.09442208
DPS	0.1386150	-0.52040357
GS	0.2507545	0.19111046
HAL	0.2281190	0.14600696
HES	0.2284285	0.10831601
HON	0.2479589	-0.06892393
HUN	0.1974712	0.14011003
JPM	0.2273526	0.10656303
KO	0.1881999	-0.47871023
MMM	0.2514868	-0.13960716
MPC	0.1711442	0.04823903
PEP	0.1702056	-0.48759946
SLB	0.2497330	0.13757414
WFC	0.2428197	0.06010682
XOM	0.2535416	-0.08084532

Table 06: shows the loading values of pc.1 and pc.2 for each of the predictor variable

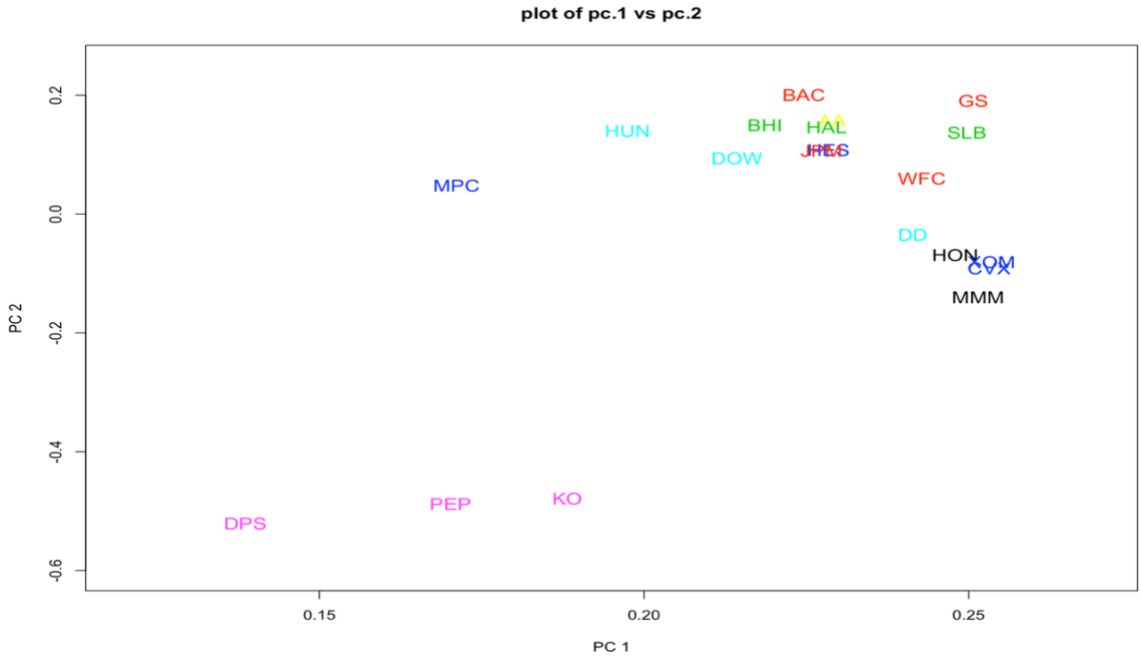

FIG 03: Plot of pc.1 and pc.2 loadings for each of the predictor variable with the labeled stock ticker. (The color of the font for the predictor variable is based on the industry the stock belongs to)

In FIG 03, we notice a few clusters/groupings in the PC1 vs PC2 loadings plot. The clusters reveal how the variables are related in terms of assigned loadings (weights) to PC1 and PC2. **The clusters noticed are:**

- **A cluster with the stock tickers AA, HAL, HES, and JPM. Based on the colors, these correspond to the Industrial - Metals, Oil Field services, Oil Refining, and Banking industries.**
- **A cluster with the tickers HON, XOM, and CVX. These correspond to the industrial – Chemical, Manufacturing, and Oil Refining industries.**

Code Snippet 04 in section 13.4 contains the R code for dimension reduction using principal component analysis (PCA).

7 Principal component selection

In this section, we present the decision rules that were evaluated for keeping the principal components. Among them, we selected the scree plot decision rule and applied it to select the principal components from the twenty principal components created by the **princomp()** R function.

The decision rules for the principal component selection that were evaluated are:

1. Just enough principal components are kept to explain some percentage of the total variation of the variables.
2. Only those principal components are kept whose eigenvalues are less than the average eigenvalue. The average eigenvalue is the average variance of all the original variables. This rule allows keeping those components that account for more variance than the average for the observed variables.
3. When the components are extracted from the correlation matrix (the data is standardized (transformed) to mean zero and unit variance), the average variance is one. Principal components with eigenvalues less than one are excluded. Typically, a cutoff or threshold (e.g., 0.7) is used.
4. Keep the principal components based on the examination of the plot of the eigenvalue of the principal component against the index of the principal component called scree plot. The scree plot shows the fraction of the total variance in the data as represented by each principal component.
5. A variation of the scree diagram is the log-eigenvalue diagram consisting of a plot of the logarithm value of the eigenvalue of the principal component for each principal component.

For our analysis, among the decision rules, we utilized rule #4 (the scree plot rule) for the selection of the principal components. We selected eight principal components because that explains 80% of the total variation in the data based on the visual inspection of the elbow in the scree plot.

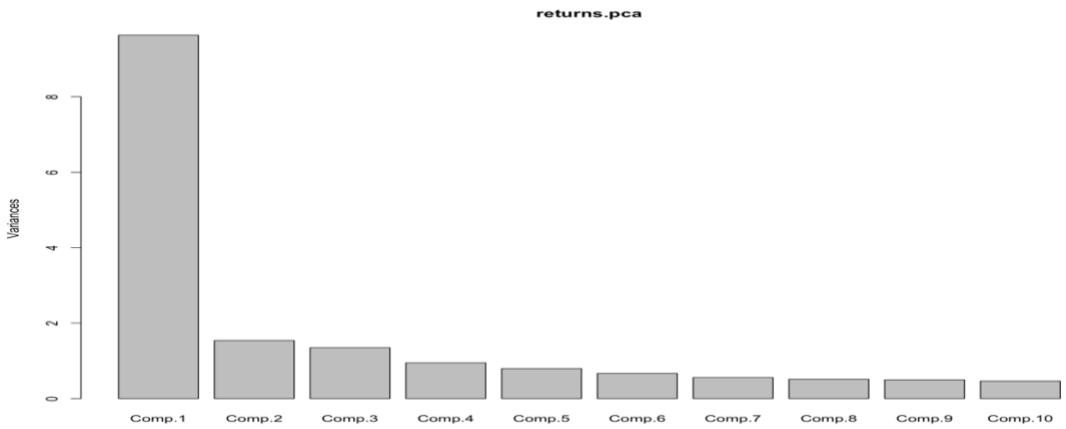

FIG 04: Default scree plot showing the amount of variance that each principal component contributes to the total, arranged from principal component 1 to principal component 10.

In FIG 04, we can confirm that the first principal component (the linear combination of the log-return variables) accounts for the highest sample variance in the data among all the linear combinations (principal components). It accounts for 48.18% of the total variance. The second principal component is defined as the linear combination of the log-return variables that accounts for a maximal proportion of the remaining variance subject to being uncorrelated with the first principal component. From Table 07, we can gather that the second principal component accounts for 7.69% of the remaining variance subject to being uncorrelated to the first principal component. The remaining principal components are defined in the same way.

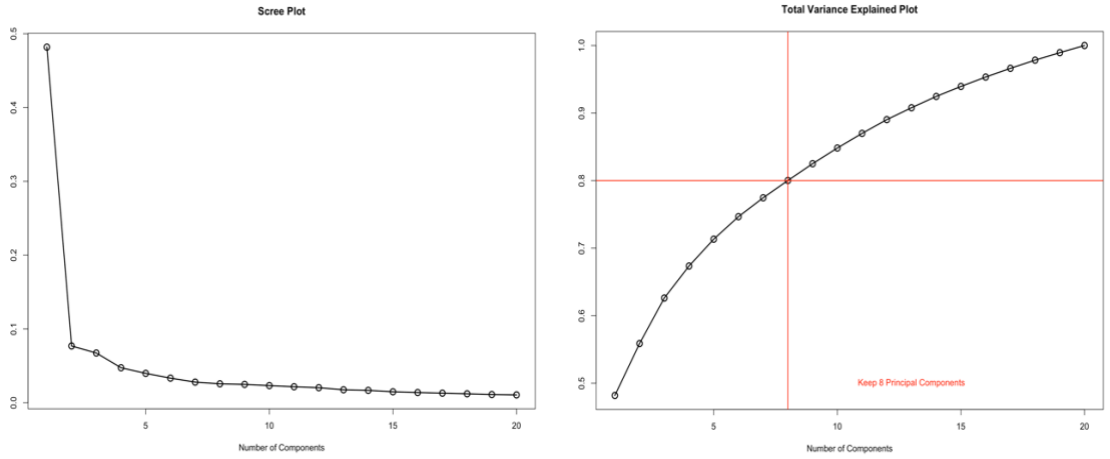

FIG 05: Two scree plots – a) plot of scree values (proportion of variance to total sum of the variance of all the data) for all components and b) line segment plot of cumulative variance for each principal component

From FIG 05 (a), we notice the elbow in the curve (a change of slope from “steep” to “shallow”) occurs when the number of the principal components is 8. Also, from FIG 05 (b), it is evident based on the intersection of the y(cum. variance)=0.8 with the scree plot line that to be able to explain 80% of the variance in the variables, we need to **keep the first eight principal components.** The same is also confirmed from the tabulation of the standard deviation, proportion of variance, and cumulative proportion of each principal component in Table 07.

Principal Component	Standard deviation	Proportion of Variance	Cumulative Proportion
Comp.1	3.1042633	0.4818225	0.4818225
Comp.2	1.24037473	0.07692647	0.55874901
Comp.3	1.16079159	0.06737186	0.62612087
Comp.4	0.97348817	0.04738396	0.67350483
Comp.5	0.89191173	0.03977533	0.71328015
Comp.6	0.8163381	0.0333204	0.7466005
Comp.7	0.7472754	0.02792103	0.77452158
Comp.8	**0.71606462**	**0.02563743**	**0.800159**
Comp.9	0.70486968	0.02484206	0.82500107
Comp.10	0.68141987	0.02321665	0.84821772
Comp.11	0.65836107	0.02167196	0.86988968
Comp.12	0.6385577	0.0203878	0.8902775
Comp.13	0.592525	0.0175543	0.9078318
Comp.14	0.57888423	0.01675535	0.92458713
Comp.15	0.54494939	0.01484849	0.93943562
Comp.16	0.52563057	0.01381437	0.95324999
Comp.17	0.50927436	0.01296802	0.96621801
Comp.18	0.491994	0.0121029	0.9783209
Comp.19	0.4710181	0.0110929	0.9894138
Comp.20	0.46013436	0.01058618	1.000000

Table 07: Table shows the standard deviation, proportion of variance and cumulative proportion that each principal component explains

Code Snippet 05 in section 13.5 contains the R code for principal component selection.

8 Principal components in predictive modeling

In this section, we present the results of fitting a linear regression model using the PCA scores by employing the first eight principal components (output of the previous section). The component scores or the PCA scores are the values of transformed variables based on the associated principal component loadings value for each data point. We employed the predictive modeling framework to split the PCA scores into train and test data partitions based on the 70/30 train/test (the basic form of cross-validation) split. We then computed the VIF values and the predictive accuracy metric (MAE) (using the train and test data partitions) for the fitted model.

Table 08 shows the PCA scores split into train and test partitions. The train/test split of the PCA scores is done using a uniform (0,1) random variable.

Data Partition	Number of records in the data partitions	Percentage of the records to the total records in PCA scores data set
Train	370	73.9%
Test	131	26.1%

Table 08: Table show record counts of the 70/30 training/test data partition of the PCA scores.

Next, we fitted a linear regression model using the first eight principal components to predict the log-return value of VV index fund using the train PCA scores partition.

The resulting fitted regression equation with the first eight principal components is

VV = 0.0006651 + 0.002265 * Comp.1 - 0.0003804 * Comp.2 - 0.0004832 * Comp.3 + 0.0001368 * Comp.4 + 0.0001707 * Comp.5 + 0.00004258 * Comp.6 - 0.0001445 * Comp.7 - 0.0003958* Comp.8

Table 09 shows the regression summary table for the fitted model (we refer to it as pca.lm)

Term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0006651	0.000143	4.652	4.62e-06 ***
Comp.1	0.002265	0.00004608	49.152	< 2e-16 ***
Comp.2	-0.0003804	0.000115	-3.308	0.00103 **
Comp.3	-0.0004832	0.0001219	-3.964	8.90e-05 ***
Comp.4	0.0001368	0.0001424	0.961	0.33738
Comp.5	0.0001707	0.0001555	1.098	0.27301
Comp.6	0.00004258	0.0001694	0.251	0.80165
Comp.7	-0.0001445	0.0001841	-0.785	0.43307
Comp.8	-0.0003958	0.000203	-1.95	0.052

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002743 on 361 degrees of freedom
Multiple R-squared: 0.8736, Adjusted R-squared: 0.8708
F-statistic: 312 on 8 and 361 DF, p-value: < 2.2e-16

Residuals:

Min	1Q	Median	3Q	Max
-0.0078374	-0.0016205	-0.0000776	0.0016213	0.0114812

Table 09: Regression summary table for the pca.lm model lm(formula = VV ~ Comp.1 + Comp.2 + Comp.3 + Comp.4 + Comp.5 + Comp.6 + Comp.7 + Comp.8, data = train.scores)

From Table 09, we can determine that at a level of significance of 0.001, the overall regression is strongly statistically significant with a p-value < 2.2e-16 (with a F-statistic value of 312). Therefore, at a level of significance of 0.001, we can reject the overall F-test (joint) null hypothesis that all the predictor variables have a zero coefficient and have no effect on the regression. Regarding the individual regression coefficients, only the intercept, Comp.1, and Comp.3 are strongly statistically significant at a level of significance of 0.001 (meaning at level of significance of 0.001, we can reject the null hypothesis that the individual regression coefficient for these components is zero). Comp.2 is strongly statistically significant at a level of significance of 0.01. All the rest of the coefficients, namely Comp.4, Comp.5, Comp.6, Comp.7and Comp.8 are NOT statistically significant. Interestingly only the first few principal components used in the regression are significant.

Also, from the Table 09, we can determine that the coefficient of determination (adjusted-R-squared) is 87.08% indicating that 87.08 percent of variation in log-return of VV can be explained variation in the principal components.

We then computed the VIF values for the principal components utilized in the fitting of the model. Table 10 shows the VIF values for the first 8 principal components.

Variable (Principal Component)	VIF value
Comp.1	1.006324
Comp.2	1.003969
Comp.3	1.007207
Comp.4	1.003926
Comp.5	1.000998
Comp.6	1.003192
Comp.7	1.004690
Comp.8	1.006958

Table 10: VIF values for all the first 8 principal components used in the fitting of the pca.lm model

None of the VIF values are greater than 3. All the values are approximately 1 (~ 1). Multicollinearity is not an issue with the principal components. The VIF values associated with every predictor variable in any principal components regression model should be one. This is because in the formula to compute the VIF value,

$$VIF_j = \frac{1}{(1 - Rj^2)}$$

In the absence of any linear relationship between the predictor variables (the principal components in the case of pca.lm), Rj^2 would be zero. As a result, VIF_j would be one for all the predictor variables in any principal components regression models.

We next computed the predictive accuracy metric, MAE (Mean Absolute Error), using the train and test partitions of the PCA scores data. Table 11 shows the in-sample (train) and out-of-sample (test) MAE values. The MAE value computed for the test data is only slightly smaller than the MAE value computed using the train data.

Model	MAE value using the train data	MAE value using the test data
pca.lm	0.002073546	0.00190761

Table 11: MAE values for pca.lm using train and test data partitions

Code Snippet 06 in section 13.6 contains the R code for fitting the linear regression model using the first eight principal components to predict the log-return VV index fund using PCA scores data.

9. Model comparison

In this section, we present the results of the comparison of the models pca.lm (the model from previous section that is fitted with the first eight principal components to predict the log-return VV index fund value), model.1 (the small model which is fitted with the log-return stock variables for GS, DD, DOW, HON, HUN, JPM, KO, MMM and XOM to predict the log-return VV stock value) and model.2 (the full model which is fitted with all the log-return stock variables for the prediction of the log-return VV stock value). Both the model.1 (small model) and model.2 (full model) were fitted in section 5 using the log-return data as part of the exploratory data analysis to examine the VIF values. However, in this section, we present the results of the model.1 (small model) and model.2 (full model), which were re-fitted using the return scores train data partition. We employed the predictive modeling framework technique to split the returns data into train/test data partitions based on the 70/30 train/test split.

The returns data is split using the same random (0,1) sampling that was employed for the splitting of the PCA scores data in section 8. As a result, the number of records in the train and the test

data partitions shown in Table 12 are exactly the same as for the PCA scores data split shown in section 8.

Data Partition	Number of records in the data partitions	Percentage of records to the total records in returns data set
Train	370	73.9%
Test	131	26.1%

Table 12: Table show record counts of the 70/30 training/test data partition of the returns dataset.

The re-fitting for the **small model (model.1)** using returns train partition results in the following equation:

VV = -0.00001764 + 0.08708 * GS + 0.02239 * DD + 0.04845 * DOW + 0.1491 * HON + 0.03808 * HUN + 0.04238 * JPM + 0.1583* KO + 0.1126 * MMM + 0.1495 * XOM

Based on the summary table in Table 13, we can conclude at the level of significance of 0.001, and we reject that the joint null hypothesis that all the coefficients have no effect on the overall regression. This is based on the p-value (with a corresponding F-statistic value of 237.1) of less than 2.2e-16. Also, it can be gleaned from Table 13 is that the individual regression coefficients for GS, DOW, HON, HUN, KO, MMM, and XOM are strongly significant at the level of significance of 0.001 meaning we can reject the null hypothesis for these individual regression coefficients to be zero at the level of significance of 0.001. The regression coefficient for JPM is significant at a level of significance of 0.01. The intercept and the regression coefficient for DD are NOT statistically significant. The coefficient of determination for the small model is 85.2% indicating that 85.2 percent of the variation in the log-return of VV index fund can be explained by variations in the predictor variables.

Term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.00001764	0.0001562	-0.113	0.910108
GS	0.08708	0.01569	5.549	5.57e-08 ***
DD	0.02239	0.02001	1.119	0.26405
DOW	0.04845	0.0137	3.536	0.000460 ***
HON	0.1491	0.01928	7.731	1.08e-13 ***
HUN	0.03808	0.008908	4.274	2.46e-05 ***
JPM	0.04238	0.01533	2.764	0.005997 **
KO	0.1583	0.01989	7.959	2.28e-14 ***
MMM	0.1126	0.02991	3.764	0.000195 ***
XOM	0.1495	0.02405	6.217	1.40e-09 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002935 on 360 degrees of freedom
Multiple R-squared: 0.8557, Adjusted R-squared: 0.852
F-statistic: 237.1 on 9 and 360 DF, p-value: < 2.2e-16

Residuals:

Min	1Q	Median	3Q	Max
-0.0095595	-0.0017119	-0.0000419	0.0017396	0.0087306

Table 13: Regression summary table after re-fitting the model.1 (Small) model lm (formula = VV ~ GS + DD + DOW + HON + HUN + JPM + KO + MMM + XOM, data = train.returns)

The re-fitting for the **full model (model.2)** using returns train data partition results in the following equation:

VV = 0.000006859 + 0.009675 * AA+ 0.03362 * BAC + 0.04189 * GS + 0.01092 * JPM + 0.07899 * WFC + 0.01257 * BHI + 0.05805 * CVX + 0.003387 * DD + 0.04254 * DOW + 0.04595 * DPS + 0.01139 * HAL + 0.001522 * HES + 0.1124 * HON + 0.03064 * HUN + 0.1124 * KO + 0.08597 * MMM + 0.008625 * MPC + 0.005828 * PEP + 0.04192 * SLB + 0.06689 * XOM

Using Table 14, we can conclude that at a level of significance of 0.001, we can reject the joint null hypothesis that all the coefficients have no effect on the overall regression. This is based on the p-value (F-statistic is 135.2) of less than 2.2e-16.

For the individual coefficient tests, the variables for the log returns of the stocks WFC, DOW, HON, HUN, and KO are strongly statistically significant at the level of significance of 0.001. The coefficients for BAC, GS, DPS, XOM, and MMM, are significant at only at a level of significance of 0.01. The coefficients for CVX and SLB are significant at level of significance 0.05. The rest of the coefficients for the variables AA, JPM, BHI, DD, HAL, HES, MPC, PEP, and the intercept are not significant.

Also, from the summary table Table 14, we can determine that the coefficient of determination for the full model (model.2) is 87.91% indicating 87.91 percent of the variation in the log return of VV can be explained by the variations in the predictor variables used in the full model.

Term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.000006859	0.0001432	0.048	0.96182
AA	0.009675	0.01231	0.786	0.432304
BAC	0.03362	0.01089	3.087	0.002183 **
GS	0.04189	0.01556	2.692	0.007437 **
JPM	0.01092	0.01542	0.708	0.479361
WFC	0.07899	0.01784	4.428	1.28e-05 ***
BHI	0.01257	0.01317	0.955	0.340441
CVX	0.05805	0.02367	2.453	0.014667 *
DD	0.003387	0.01843	0.184	0.85434
DOW	0.04254	0.01266	3.36	0.000866 ***
DPS	0.04595	0.01682	2.732	0.006619 **
HAL	0.01139	0.01429	0.797	0.426107
HES	0.001522	0.01106	0.138	0.890583
HON	0.1124	0.0183	6.141	2.23e-09 ***
HUN	0.03064	0.008312	3.687	0.000263 ***
KO	0.1124	0.02077	5.413	1.16e-07 ***
MMM	0.08597	0.02758	3.117	0.001981 **
MPC	0.008625	0.008164	1.057	0.291461
PEP	0.005828	0.02412	0.242	0.809214
SLB	0.04192	0.01638	2.56	0.010886 *
XOM	0.06689	0.02574	2.599	0.009756 **

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002653 on 349 degrees of freedom
Multiple R-squared: 0.8857, Adjusted R-squared: 0.8791
F-statistic: 135.2 on 20 and 349 DF, p-value: < 2.2e-16

Residuals:

Min	1Q	Median	3Q	Max
-0.009177	-0.001602	0.000034	0.001611	0.008301

Table 14: Regression summary table after re-fitting the model.2 (full) model lm(formula = VV ~ AA+BAC + GS + JPM + WFC + BHI + CVX + DD + DOW + DPS + HAL + HES + HON + HUN + KO + MMM + MPC + PEP + SLB + XOM, data = train.returns)

Next, we computed the predictive accuracy metric MAE for model.1 and model.2 using the in-sample and out-of-sample data. MAE measures the average magnitude of the errors in the predictions, without considering the direction. Table 15 shows the MAE values for model.1 and model.2 next to pca.lm for comparison. Smaller MAE values are preferred. The test MAE errors are lower than the train MAE errors. As a result, overfitting is not a concern.

Model	MAE value using the train data	MAE value using the test data
pca.lm	0.002073546	0.00190761
Model.1 (small model)	0.00223857	0.002043099
Model.2 (full model)	**0.001992983**	**0.001874203**

Table 15: MAE values for the three models – pca.lm, model.1, model.2 using in-sample and out-of-sample data

From the table, we can gather that the model with the smallest MAE computed using the train data and test data is Model.2 (full model).

Code Snippet 07 in section 13.7 contains the R code for model comparison using pca.lm, model.1 (small model), and model.2 (full model).

10. Automatic variable selection for the principal components to keep

In this section, we present the results of employing the automatic variable selection method to select the principal components to keep. We used the backward variable selection method for this. We utilized the stepAIC function in R with "backward" direction and a starter regression equation containing a full model with all the principal components to predict the log-return VV variable. The train partition for the PCA scores is used. The stepAIC function performed 12 iterations based on the AIC value. The backward variable selection method selected a regression equation with 10 terms (9 principal components and an intercept term).

The fitted regression equation selected by the backward variable selection (we refer to it as backward.lm) is

VV = 0.0006666 + 0.002259 * Comp.1 - 0.0003652 * Comp.2 - 0.000503 * Comp.3 -0.0004221* Comp.8 + 0.0002834 * Comp.9 - 0.0005123* Comp.10 + 0.0007687 * Comp.11 + 0.00035 * Comp.12 - 0.0005107* Comp.14

Compared to the first 8 principal components used in the pca.lm model, the backward variable selection suggests 9 principal components. Among the first 8, the backward variable selection does not suggest Comp.5, Comp.6, and Comp.7.

On the principal components picked out by the stepAIC function using the backward variable selection, we computed the VIF values (Variance Inflation Factors) to determine if any of them are collinear. Typically, VIF values greater than 3 indicate the presence of multicollinearity among the predictor variables. From Table 16, we can gather the VIF values are approximately about ~ 1.

Principal Component	VIF value
Comp.1	1.017908
Comp.2	1.004115
Comp.3	1.011744
Comp.8	1.013439
Comp.9	1.007720
Comp.10	1.007588
Comp.11	1.010018
Comp.12	1.010143
Comp.14	1.011550

Table 16: VIF values for the principal components picked out the backward variable selection model

From the Table 17 below, we can conclude that at a level of significance of 0.001, the overall regression is strongly statistically significant with a p-value < 2.2e-16 (with a F-statistic value of 302.3). So, at a level of significance of 0.001, we can reject the null joint hypothesis that all the predictor variables have a zero coefficient and have no effect on the regression.
Regarding the individual regression coefficients, the coefficients for the principal components the intercept, Comp.1, Comp.3, and Comp.11 are strongly statistically significant at the level of significance 0.001. The coefficient for Comp.2 is statistically significant at the level of significance 0.01. The coefficients for the principal components Comp.8, Comp.10, and Comp.14 are statistically significant at a level of significance 0.05. The coefficient for the component Comp.12 is significant at a level of significance of 0.1. The adjusted-r-squared value is 0.8802 indicating 88.02% of the variation in the log-return of VV is explained by the variations in the principal components.

Term	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0006666	0.0001379	4.834	1.99e-06 ***
Comp.1	0.002259	0.00004463	50.614	< 2e-16 ***
Comp.2	-0.0003652	0.0001107	-3.298	0.001072 **
Comp.3	-0.000503	0.0001177	-4.275	2.45e-05 ***
Comp.8	-0.0004221	0.0001961	-2.152	0.032063 *
Comp.9	0.0002834	0.0001935	1.464	0.144008
Comp.10	-0.0005123	0.0002044	-2.506	0.012652 *
Comp.11	0.0007687	0.0002016	3.814	0.000161 ***
Comp.12	0.00035	0.0002103	1.665	0.096828 .
Comp.14	-0.0005107	0.0002355	-2.169	0.030757 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002641 on 360 degrees of freedom
Multiple R-squared: 0.8831, Adjusted R-squared: 0.8802
F-statistic: 302.3 on 9 and 360 DF, p-value: < 2.2e-16

Residuals:

Min	1Q	Median	3Q	Max
-0.0091099	-0.0015776	0.0000265	0.0016262	0.0097865

Table 17: Regression output for the estimated model using the backward variable selection (backward.lm)

Next, we computed the predictive accuracy metric MAE for the model estimated using the backward variable selection. This is done using both the in-sample (train) and out-of-sample (test) data partitions. Table 18 shows the MAE values for the four models (pca.lm, model.1 (small model), model.2 (full model), and backward.lm) using the train and test data partitions.

Model	MAE value using the train data	MAE value using the test data
pca.lm	0.002073546	0.00190761
Model.1 (small model)	0.00223857	0.002043099
Model.2 (full model)	**0.001992983**	0.001874203
backward.lm	0.002014638	**0.001863317**

Table 18: MAE values for the four models – pca.lm, model.1 (small model), model.2 (full model), backward.lm using in-sample and out-of-sample data

From Table 18, we can determine that Model.2 (the full model) has the lowest MAE value among all the four models using the in-sample data. Among the MAE values computed using the out-of-sample data partition, the backward.lm (the linear regression model returned by backward variable selection) has the lowest MAE value. Out-of-sample accuracy is more than in-sample accuracy. As a result, we conclude the following:

Based on the comparison, backward.lm (the linear model to predict the log-return value of Vanguard (VV) index fund using the principal components selected using automated variable selection) is the best model among all the fitted models.

Code Snippet 08 in section 13.8 contains the R code to select the principal components using automatic variable selection.

11. Dimension reduction using T-distributed Stochastic Neighbor Embedding (t-SNE)

In this section, we present the results of dimension reduction using T-distributed Stochastic Neighbor Embedding (t-SNE), a non-linear dimensionality reduction algorithm, on the original stock data dataset. As explained in section 2, the stock data dataset has 21 variables representing the daily closing stock values. The values are represented by continuous variables. t-SNE takes this data in the high-dimensional space and iteratively transforms them to represent them in a 2D space (map). The 2D map captures the local structure of the high-dimensional data while also revealing the global structure (clusters) at several scales.

We used the **Rtsne** package for this analysis. To the **Rtsne**() function, we passed in the 20 variables in the dataset. We have excluded the VV variable since it is the response variable. VV variable ranges from 58.18 to 84.80. To be able to visualize the clusters, we divided the VV variable into 7 bins. A unique color was assigned to each bin.

Bin	VV variable range
1	>= 58.0 and < 60.0
2	>= 60.0 and < 65.0
3	>= 65.0 and < 70.0
4	>= 70.0 and < 75.0
5	>= 75.0 and < 80.0
6	>= 80.0 and < 85.0
7	All other

Table 19: Table show the bin number allocation based on the VV variable range.

t-SNE has several tuning parameters:

a) **dims** - Output dimensionality; the dimensionality of the output map. For a 2D map, dims value is 2.
b) **Initial dims** - the number of dimensions that should be retained in the initial PCA step; PCA may be used as a first step to bring the dimensions down before t-SNE algorithm is applied. The default is 50. In our case, the number of dimensions is 20 (excluding the response variable VV). Hence, we do not adjust this parameter.
c) **perplexity** – it is a cost function parameter. It is a smooth measure of the effective number of neighbors. Larger datasets require a larger perplexity. A typical value is between 5 and 50.
d) **max iteration** - number of iterations for t-SNE ; it is an optimization parameter. It should be at least 250 to achieve optimization.
e) **learning rate** – it controls how t-SNE learns after every iteration. It is an optimization parameter. The usual range is from 10.0 to 1000.0.

For our analysis, we controlled the perplexity, learning rate, and max iterations parameters. The base values employed for perplexity, learning rate, and max iterations are 30, 200, and 500, respectively. Each parameter is tested one at a time while keeping the other parameters at base values.

For the perplexity, we tested with the values 2, 5, 30, 50, 65, and 80. FIG 06 shows the visualization results using t-SNE of the stock data for different values of perplexity in a 2D map (dims = 2).

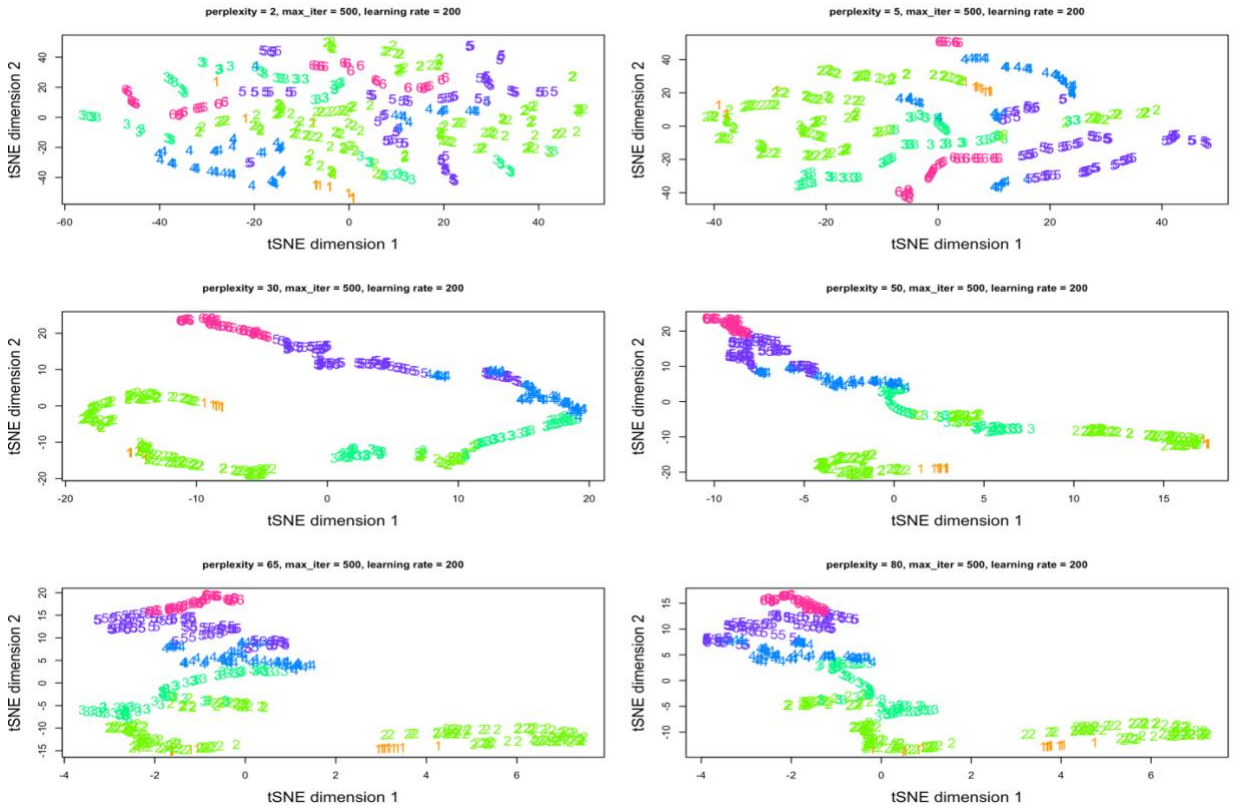

FIG 06: Visualization of the clusters generated by t-SNE corresponding to different perplexity values - 2,5,30,50,65,80 with max_iter set to 500 and learning rate set to 200

From FIG 06, we determined that the 2D maps produced by the perplexity values are able to differentiate the VV values (bins). However, we noticed in the structures produced, there are some overlaps among bins 1 and 2, bins 2 and 3, and bins 4 and 5. These overlaps or structures are evident in the visualizations produced by perplexity values 30, 50, 65, and 80. For our data, a perplexity value of 30 is ideal.

Next, we trained t-SNE with different learning rates. We applied different values - 20, 100, 200, 1000, and 2000, with perplexity set to 30 and maximum iterations set to 500. Refer to FIG 07 for the clusters generated by t-SNE for the stock data using the different learning rate values.

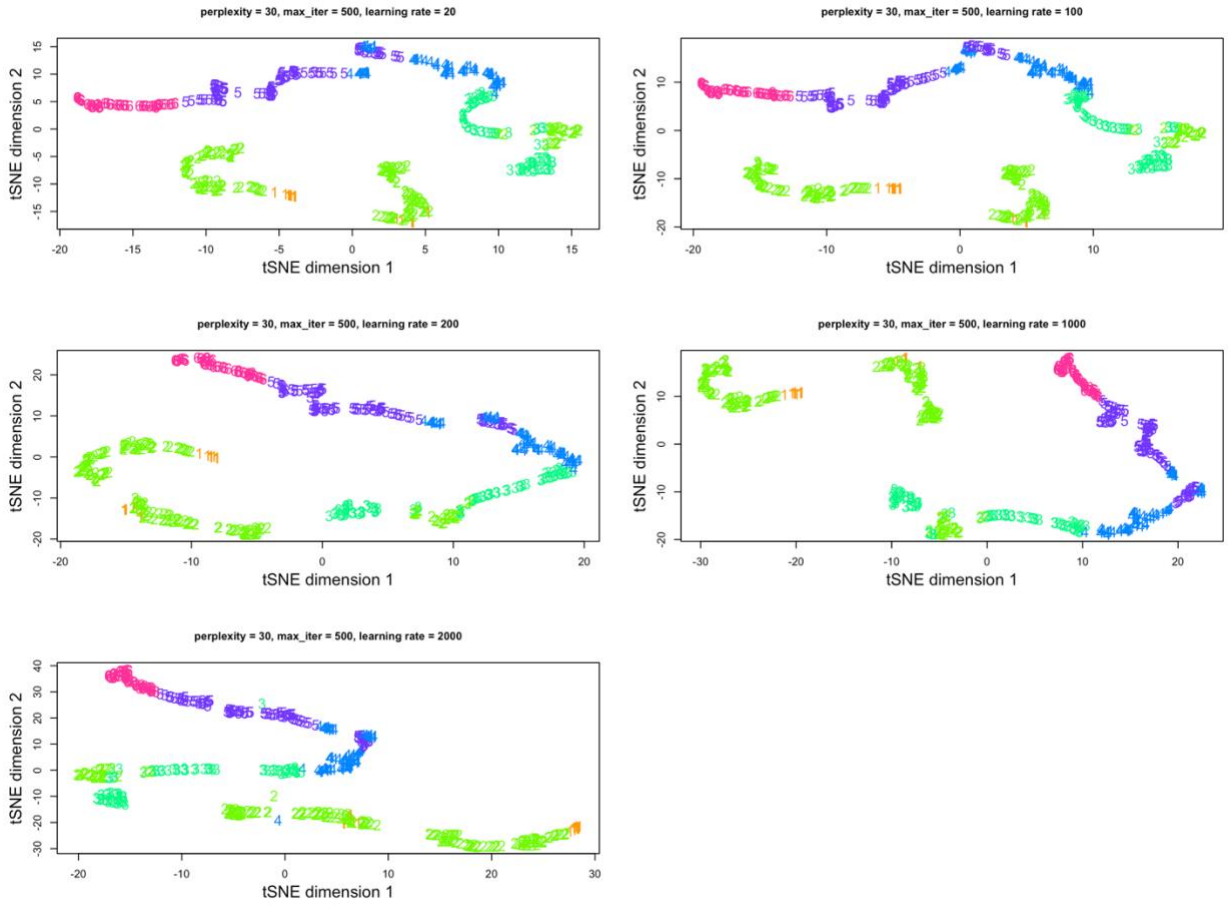

FIG 07: Visualization of the clusters generated by t-SNE corresponding to different learning rate values – 20, 100, 200, 1000, 2000 with max_iter set to 500 and perplexity set to 30

In FIG 07, the same overlaps noted in the perplexity 2D maps – namely among bins 1 and 2, bins 2 and 3, and bins 4 and 5 are also seen in the structures produced with different learning rates. All five values capture the local and global structures. Since the learning rate refers to the amount of adjustment made in each iteration and hence decides how fast the optimization is achieved, we used the middle of the ground approach and selected 200 for our dataset.

We also trained t-SNE model with different values for the max iterations. We tried with the values 250, 500, 1000, 2000, 3000, and 4000. FIG 08 shows the visualizations of the structures generated by t-SNE with different max iterations values. In these graphs too, the overlaps noted previously among bins 1 and 2, bins 2 and 3, and bins 4 and 5 are clearly evident. It can be seen at 500 max iterations, the optimization has been achieved. All the max iterations values greater than 500 result in the same structure. So, max iterations of 500 is enough for this data.

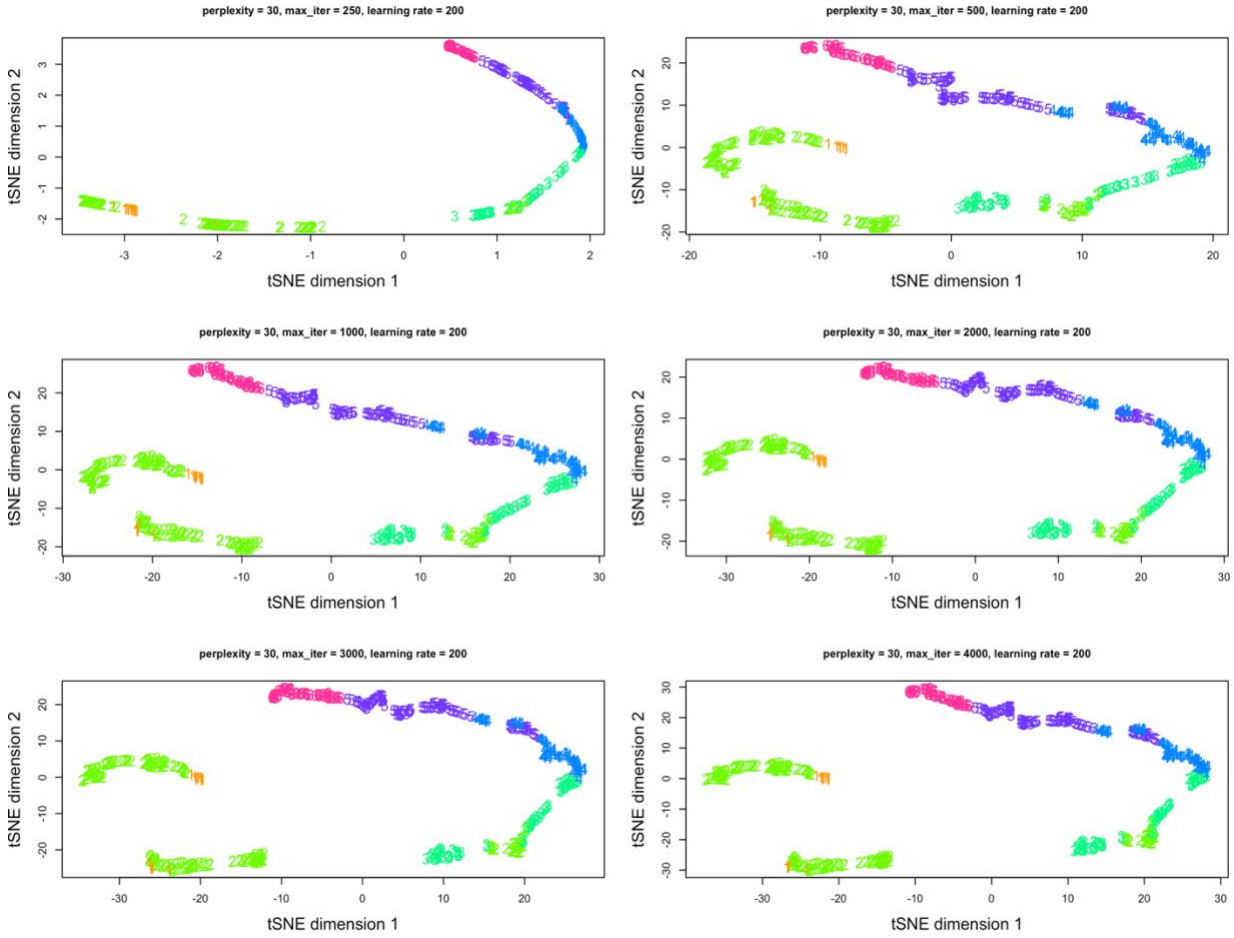

FIG 08: Visualization of the clusters generated by t-SNE corresponding to different max iteration values – 250, 500, 1000, 2000, 3000, 4000 with learning rate set to 200 and perplexity set to 30

t-SNE vs PCA plots:

Though PCA and t-SNE are both dimensionality reduction methods, the approach to the projection of the high dimensional data to low dimensional space is different. PCA performs a **linear combination** of the original variables in such a way that the sample variance is the maximum among all possible linear combinations. The resultant linear combinations of the original variables (which are the principal components) are also orthogonal to each other. Additionally, the principal components are derived in the decreasing order of importance. Hence, PCA is a **deterministic** approach to dimension reduction. Hence, the results of PCA presented in section 6 obtained using the ***princomp()*** function on the log-return data with the correlation matrix are deterministic regardless of how many times the function was run. The PCA results can be interpreted as a rotation of the axes represented by the

principal components. FIG 03 in section 6 is a PC1 vs PC2 loadings plot, and it reveals some clusters/relationships among variables in terms loadings for principal components. FIG 09 shows the PCA score plot computed using the return.scores generated as part of section 8. No color codes were assigned to the return.scores based on the log-return VV values. Even then, the scatter plot of PC1 vs PC2 score plot does not reveal any clusters.

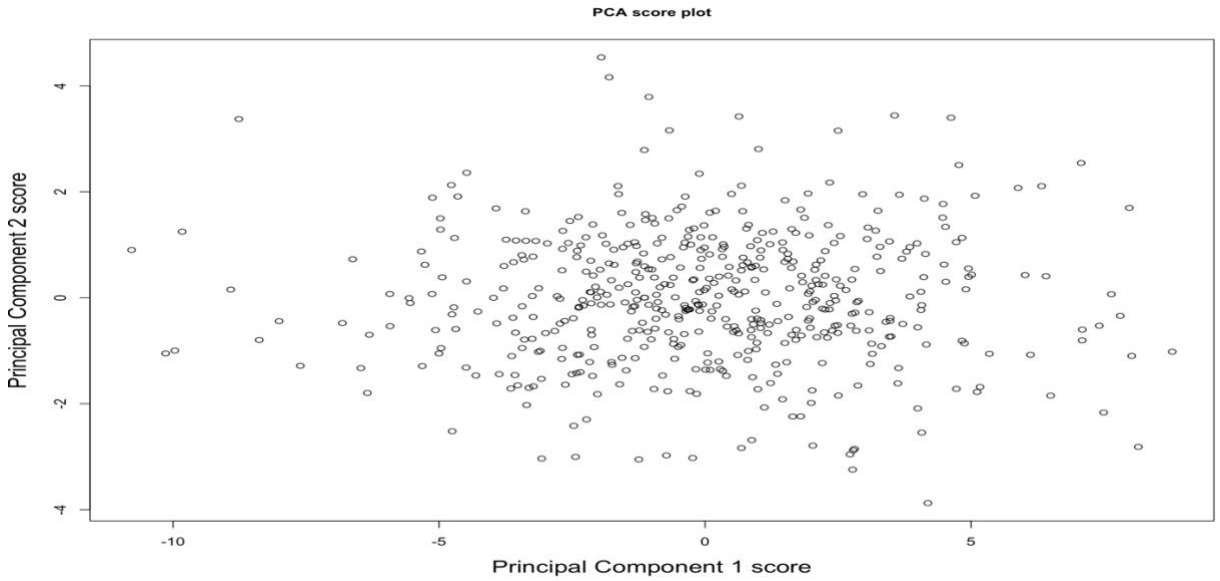

FIG 09: PCA Score plot

On the contrary, t-SNE uses a **non-linear approach** to the projection of the high dimensional data to the low dimensional space. t-SNE uses a **probabilistic approach** with two stages:

Stage 1: A probabilistic distribution is constructed for pairs of high-dimensional data points so that similar data points have a high probability of being picked, while dissimilar data points have an extremely small probability of being picked.

State 2: A similar probability distribution over the points in the low-dimensional map is obtained, then the Kullback-Leibler divergence between the two distributions is minimized with respect to the locations of the points in the map.

This is where another difference arises between PCA and t-SNE. **PCA preserves the global structure**. **t-SNE is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales.** The t-SNE plots in FIG 06, FIG 07, and FIG 08 are reveal these clusters or structures. But t-SNE does not preserve the global inter-cluster structure or distance. t-SNE can handle non-linearities in the data.

Due to the probabilistic nature of t-SNE, it is only useful for exploration and/or visualization. PCA, on the other hand, due to its deterministic output can be used as the first step in other analysis.

Code Snippet 09 in section 13.9 contains the R code for dimension reduction using t-SNE and the PCA score plot for comparison.

12. Summary and reflection

We began our analysis with PCA. We used log-return transformations on the stock data dataset. This also served as the standardization for PCA to ensure each variable has a mean zero and unit variance. Our review of the correlation plots has shown evidence of multicollinearity in the data.

Using princomp() R function, we computed the principal components for the return data. We ran the PCA using the correlation matrix. The PC1 vs PC2 loadings plot has revealed some clusters among the log-return variables. Since a loading in a principal component is the weight by which each original variable should be multiplied to get the principal component score, the clusters in the PC1 vs PC2 loadings plot reveal how the variables are related in terms of assigned loadings to PC1 and PC2.

After obtaining the principal components, we then used several techniques to determine how many PCs to keep to explain 80% of the total variance in the data. We used scree plots, total variance explained plot, and cutoff/threshold method. Our criteria resulted in us selecting the first eight principal components.

We next proceeded to perform linear regression using the first eight principal components to predict the log-return VV variable. The data used for this model is the return scores data. We also used the predictive framework with 70/30 train/test data to train and test the model. The resultant model is compared to two naïve models (small and full) using the accuracy metric, MAE. Among those three, the full model fared the best based on in-sample and out-of-sample MAE values. Next, we used a backward variable selection to select the principal components to use in the linear regression model to predict the log-return value of VV. The resultant model, when compared to the previous models, presented the best MAE value on the out-of-sample data among all the four models. Based on that, we concluded the backward variable selection model as the best model to predict the log-return VV value.

The next task was to start again with the original stock portfolio data set and use the T-distribution Stochastic Neighbor Embedding (t-SNE) to perform the dimension reduction. Based on the VV values, the data were divided into 7 bins. The bins were used to color the structure provided by t-SNE. t-SNE has several tuning parameters – perplexity, learning rate, max_iterations. Different values were used to train t-SNE model, and based on the testing, for the stock portfolio data set, we selected 30 as the perplexity, 200 for the learning rate and 500 for the max_iterations. The visualization produced by t-SNE shows overlapping clusters among bins 1 and 2, bins 2 and 3, and bins 4 and 5.

Lastly, we compared and contrasted the PCA and t-SNE plots. From the PCA plots and the t-SNE plots, we determined PCA preserves the global structure well, but t-SNE is well-suited for understanding local structures in the data. We understood that t-SNE is a stochastic algorithm, whereas PCA is a deterministic algorithm. PCA projections to low dimensional space are based on linear combinations of the original variables. t-SNE is a non-linear dimensionality reduction algorithm. Due to the probabilistic nature of the t-SNE, we learned that the output of t-SNE could not be used for inference. It serves only as a data exploration and visualization tool. PCA output, on the other hand, can be used with other analytical methods.

13. Code

13.1 Data preparation

```
library(readxl)
setwd("/Users/harini-mac/Desktop/Northwestern University/MSDS-411/Week1/Assignment1/")

my.data <- read_excel(path="stockdata.xlsx")

str(my.data)
head(my.data)
names(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]]);
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)
```

Code Snippet 01: R code for the data preparation

13.2 Exploratory data analysis using statistical graphs and data visualization tools

```
# 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)

# Draw the corrplot
require(corrplot)
corrplot(returns.cor, method="number")
```

Code Snippet 02: R code for the Exploratory Data Analysis (EDA) using statistical graphs and data visualization plots.

13.3 Exploratory data analysis using models

```
# 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 ~
AA+BAC+GS+JPM+WFC+BHI+CVX+DD+DOW+DPS+HAL+HES+HON+HUN+KO+MMM+MPC+PEP+SLB+XOM,data=retu
rns.df)
summary(model.2)
vif(model.2)
```

Code Snippet 03: R code for the Exploratory Data Analysis (EDA) using models

13.4 Dimension reduction using the principal component analysis

```
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(0.12,0.27),ylim=c(-0.60,0.25),xlab='PC 1',ylab='PC 2')
text(pc.1,pc.2,labels=names(pc.1),cex=1.25,col=c(7,2,3,4,5,5,6,2,3,4,1,5,2,6,1,4,6,3,2,4,9))
title("plot of pc.1 vs pc.2")
```

Code Snippet 04: R code for the principal component analysis using the log-return data

13.5 Principal component selection

```
# 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')
```

Code Snippet 05: R code for the principal component selection using scree plots

13.6 Principal components 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));
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));
```

Code Snippet 06: R code for fitting the linear regression model using the first eight principal components to predict the log-return VV index fund using PCA scores data.

13.7 Model comparison

```
# 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 <- 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)

# Fit model.2 on train data set and 'test' 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));
summary(model.2)
```

Code Snippet 07: R code for model comparison using pca.lm, model.1 (small model), and model.2 (full model)

13.8 Automatic variable selection to pick the principal components

```
# 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)

# use stepAIC to perform backward variable selection
library(MASS)
backward.lm <- stepAIC(full.lm,direction=c('backward'))
summary(backward.lm)

# Compute the MAE on training data and the VIF values
backward.mae.train <- mean(abs(train.scores$VV-backward.lm$fitted.values));
vif(backward.lm)

# Compute the MAE on test data
backward.test <- predict(backward.lm,newdata=test.scores);
backward.mae.test <- mean(abs(test.scores$VV-backward.test))
```

Code Snippet 08: R code for automatic variable selection to pick the principal components

13.9 Dimension reduction using t-SNE

```
# t-Distributed Stochastic Neighbor Embedding [t-SNE]
# import libraries
library(tidyverse)
library(Rtsne)
library(readxl)

setwd("/Users/harini-mac/Desktop/Northwestern University/MSDS-411/Week1/Assignment1/")
my.data <- read_excel(path="stockdata.xlsx")

# Exclude the returns values from the tsne_data
tsne_data <- my.data %>% select(-Date, -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, -response_VV)
str(tsne_data)
head(tsne_data)
names(tsne_data)
#summary(tsne_data)

# Assign bins based on the VV index
tsne_data$bin <- ifelse((tsne_data$VV >= 58.0 & tsne_data$VV < 60.0),1,
ifelse((tsne_data$VV >= 60.0 & tsne_data$VV < 65.0),2,
ifelse((tsne_data$VV >= 65.0 & tsne_data$VV < 70.0),3,
ifelse((tsne_data$VV >= 70.0 & tsne_data$VV < 75.0),4,
ifelse((tsne_data$VV >= 75.0 & tsne_data$VV < 80.0),5,
ifelse((tsne_data$VV >= 80.0 & tsne_data$VV < 85.0),6,7))))));

# Assign colors based on the bin values
colors = rainbow(length(unique(tsne_data$bin)),start=0.1,end=0.9)
names(colors) = unique(tsne_data$bin)

# Function to plot the t-sNE plots
tsne_plot <- function(perpl=30,iterations=500,learning=200){
  set.seed(1) # for reproducibility
  tsne <- Rtsne(tsne_data %>% select(-VV, -bin), 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 = tsne_data$bin, col = colors[tsne_data$bin],cex=1.35)
}

# Test with different perplexity values
par(mgp=c(3,1,0), mfrow=c(3,2))
perplexity_values <- c(2,5,30,50,65,80)
sapply(perplexity_values, function(i){tsne_plot(perpl=i)})

# Test with different learning rate values
par(mgp=c(2.5,1,0), mfrow=c(3,2))
learning_values <- c(20,100,200,1000,2000)
sapply(learning_values,function(i){tsne_plot(learning=i)})

# Test with different max iteration values
par(mgp=c(3,1,0), mfrow=c(3,2))
max_iters_values <- c(250, 500, 1000, 2000, 3000, 4000)
sapply(max_iters_values, function(i){tsne_plot(iterations=i)})

# for comparison, plot the PC1 vs PC2 scores
par(mgp=c(3,1,0), mfrow=c(1,1))
plot(return.scores$Comp.1, return.scores$Comp.2, xlab="Principal Component 1 score", ylab="Principal Component 2 score", main =
"PCA score plot", cex.main=1, cex.lab=1.5)
```

Code Snippet 09: R code for dimension reduction using t-SNE and comparison of results with PCA