Dow Jones Case Study

Cynthia Farias, Maria Morinigo, Robert Perez, Rubina Saya

4/09/2021

Executive Summary

The `Support Vector Regression (SVR)` was able to predict the stocks with the highest rate of return and the best accuracy for the Dow Jones case study dataset, with an `average mean squared error (MSE)` of 8.93. Comparatively, the `linear regression` and `decision tree` resulted in average `MSEs` of 64.08 and 9.57, respectively.

Even though the `linear regression` resulted in the highest average `MSE`, the process of eliminating collinearity issues between independent variables helped determine the inputs used for generating the final three models. (`Week`, `percent_change_price`, `percent_change_volume_over_last_wk`, `open.lag`, `percent_change_volume_over_last_wk.lag`).

Although `SVR` is not as interpretable as a `decision tree` or `linear regression` models, in this case the interpretability is not as important as the accuracy of prediction. If a final model is stable enough to predict stock market trends relatively accurately compared to other models, the significance of identifying inputs is not as critical.

The `Capital Asset Pricing Model (CAPM)` was also utilized to assess the given stocks' risk compared to the market. Based on the `CAPM` and `SVR` model predictions, we selected `IBM`, `DuPont`, and `Caterpillar` as the three best stocks. These stocks had `Beta` values of less than 1, indicating less volatility than the market, making them less risky. At the same time, `SVR` predicted these stocks to have the highest rate of return.

Problem

The stock market prediction is an extensively researched and studied area of finance, mainly due to its impact on the investment banking realm. In this case study, the focus is on forecasting the rate of return for the given stocks based on historical patterns in earnings. The assumption is that the past earnings are reflective of the near future earnings.

The goal of this case study is to answer the following questions:

- Which stocks will produce the highest rate of return in the following week?
- Based on `CAPM`, which stocks are riskier?
- Which model between `Linear`, `Decision Tree`, and `Support Vector Regression` will provide the highest accuracy for predicting stock prices?

The remaining sections of this report will look at the existing literature related to stock pricing prediction models, discuss the data exploration and methodologies applied to develop the best model, and present the results and recommendations.

Review of Related Literature

Given the enormous potential for profit that can be achieved by identifying a prediction model for the stock market performance that is even marginally better than the competition, it comes as no surprise that there has been significant research in this area and attempts to apply machine learning to improve upon traditional models. Kumbhare, Makhija, and Raichandani chose to evaluate two years of `S&P 500` market data using a `Support Vector Machine Algorithm (SVM)` model with Decision Trees (2007). They note their decision to exclude a Neural Network approach because of its difficulty to program and train, and visually interpret the data. Utilizing a 75/25 split of training and test data, they were able to achieve accuracies ranging from 73% to 87% in predicting stock prices. It should be noted that these were prices of specific stocks, including the highest accuracy (Facebook) and not the overall index.

Hegazy, Soliman, and Salam, also utilized an SVM model to analyze S&P 500 stocks rather than an algorithm based on Neural Networks, citing the over-fitting issues common to Neural Networks because of the large number of parameters to fix (2013). Alternatively, Qian and Rasheed chose to analyze the Dow Jones Industrial Average (DJIA) index dataset using `Neural Network`, `K-Nearest Neighbor`, and `Decision Tree algorithms`. They achieved an accuracy of 60% through combining these models (2007). They noted that because time series prediction of stock prices above 50% is rare, this can be considered a success as even a 1% improvement in predictive accuracy can be lucrative.

Finally, Nguyen and Yoon focused exclusively on a Neural Network based algorithm achieving similar accuracy of just over 60% in analyzing both the S&P 500 and KOSPI 200 indices (2019). This accuracy was achieved by using the one-day stock returns of the index for a company compared to companies in a similar industry. They also demonstrated that their model's accuracy exceeded an SVM-based model, showing that neural networks can be more accurate in certain situations. To overcome the common overfitting problem noted above due to insufficient sample data, they utilized a framework called Deep Transfer with Related Stock Information (DTRSI), which uses a large amount of data from different but related stocks.

Methods

This case study aims to build models to predict stock prices and evaluate the risks. We used `linear regression` , `decision trees` , and `support vector machines` to predict the stock prices and test for prediction accuracy using average `mean squared error (MSE)` . We also used `CAPM (Capital Asset Pricing Model)` to evaluate the risks of the different stocks. The `dow_jones_index.data` is a subset of weekly stock data from the `Dow Jones Index` , and it comprises 750 observations and 16 variables reported by major stock exchanges. The `percent_change_next_weeks_price` is the response variable. The independent variables are `quarter` , `stock` , `date` , `open` , `high` , `low` , `close` , `volume` , `percent_change_price` , `percent_change_volume_over_last_wk` , p `revious_weeks_volume` , `next_weeks_open` , `next_weeks_close` , `percent_change_next_weeks_price` , `days_to_next_dividend` , `percent_return_next_dividend` . Since the information is quantifiable, and it is reasonable to assume the past pattern will continue, a forecast was developed using a time series method. Before creating the models, the dataset was preprocessed to ensure that the missing values and multicollinearity amongst the independent variables would not create issues in the model creation process. Additionally, the categorical variables were coded as factors to properly run the models. An additional variable, Week, was created to be able to create lag variables.

We used `regression analysis` as a forecasting tool to view the time series that we wanted to forecast as the dependent variable. In the `dow_jones_index.data` , the measurements were taken every week, and a `Week` variable was created to account for the regular time interval. The next step was to create a lagged residual plot to detect autocorrelation in residuals from the time series. Lag variables were created and added to the regression model as explanatory variables; we used lag = 1 as predictors to the model. We also were able to identify a good set of related independent variables and developed a regression equation for forecasting the time series. For this, we use `VIF (Variance inflation factor)` to detect multicollinearity between predictors in the model. We kept variables that had `VIF` less than 5, which indicates that the predictors are moderately correlated. The resulting regression equation was used as formulas in the other predicting models used in this case study. Then we ran the model, made predictions, and calculated the `MSE` for each of the stocks. We use the average `MSE` to compare against the average `MSE` of the other model techniques. As with any data modeling technique, each of the model techniques has advantages and inherent constraints. One of the main advantages of a `linear model` is the simplicity and the interpretability of the model coefficients and the effect of independent variables on the target variable. However, one major disadvantage of `linear models` is sensitivity to outliers.

A `Decision Tree` is a supervised learning algorithm that can be used in regression problems. It is the second predictive modeling technique used in this case study. For this model technique, we used the same predictors used in the linear regression model. We grew our tree using the training set and obtained our predictions on the test dataset, and finally computed the MSE for each stock. One of the advantages of a `Decision Tree` is that the output is easy to interpret. Decision Tree is also helpful in data exploration because it identifies the most significant variables and relations between two or more variables. Another advantage is that it requires less data cleaning and outliers do not influence it, and it can handle both numerical and categorical variables. However, overfitting can be a problem when using decision trees. And small changes can have a significant impact on the decisions. Some techniques can be performed to improve the performance of decision trees like bagging, random forest, and boosting (Arka Roy, DA 6813. DATA ANALYTICS APPLICATIONS SPRING 2021. PowerPoint).

The third machine learning technique utilized in this case study was `Support Vector Machines (SVM)` . The idea behind this type of modeling is to find the optimal hyperplane that gives the best separation boundaries between the data points; therefore, allowing correct classification of the points. We used `tune.svm()` to perform `cross_validation` to select the best choice for `gamma` and `cost` . After tuning the model and finding the best parameters for cost and gamma, we made our predictions and computed the MSE. Benefit of SVM includes its ability to deal with both regression and classification tasks. `SVM` can lead to higher accuracy in large dimensional problems where the data is skewed, or the distribution is unknown, compared to parametric models such as `linear` or `logistic` models. They can also adapt to various types of data through the usage of different kernel methods, making them very versatile. One of the downsides of SVM is that the models are harder to interpret, as they do not have a direct probabilistic interpretation. Additionally, `SVMs` suffer from scaling issues when scaling to a large amount of training samples.

Finally, used `CAPM` to predict individual stock `betas` of investment and measure the systematic risk, which are risks like interest rates, recessions, wars, and pandemics (COVID 19) that cannot be diversified away. `CAPM` can also be used for pricing securities and generating expected returns. The beta obtained by the `CAPM` model is a statistical measure that captures the relationship between the returns of security and the returns of the market. If the `beta` is greater than 1, the stock is riskier than the market. On the contrary, if the `beta` is less than 1, the stock is less riskier and therefore, reduces the risk of a portfolio. As mentioned before, CAPM can also be used to determine the expected return of the investment. Investors use these results to find the value of an asset (Arka Roy, DA 6813. DATA ANALYTICS APPLICATIONS SPRING 2021. PowerPoint).

Data

The data utilized relates to the Dow Jones Index(DJI). The dataset contains 750 observations, and 16 variables. The data was collected from the timeframe of January 1, 2011, until June 24, 2011. It should be noted that the variable `stock` consists of 30 individual stocks. The `stock` variable contained missing values for the week of January 07, 2011, for each stock; therefore, those observations were omitted. The date column was then converted into a YYYY-MM-DD format, prior to processing data. Although the option of utilizing the variable `date` was considered, a new variable, `Week`, was created for a more simplistic analysis. As the `Week` variable uses the date variable as a basis, the sequence of time intervals was preserved for the new variable. This resulted in a weekly increment of 23 weeks, with the week of January 01, 2011, to January 07, 2011, being the first week.

- **quarter**: the yearly quarter (1 = Jan-Mar; 2 = Apr=Jun).
- **stock**: the stock symbol.
- **date**: the last business day of the work (this is typically a Friday)
- **open**: the price of the stock at the beginning of the week
- **high**: the highest price of the stock during the week
- **low**: the lowest price of the stock during the week
- **close**: the price of the stock at the end of the week
- **volume**: the number of shares of stock that traded hands in the week
- **percent_change_price**: the percentage change in price throughout the week
- **percent_change_volume_over_last_week**: the percentage change in the number of shares of stock that traded hands for this week compared to the previous week
- **previous_weeks_volume**: the number of shares of stock that traded hands in the previous week
- **next_weeks_open**: the opening price of the stock in the following week
- **next_weeks_close**: the closing price of the stock in the following week
- **percent_change_next_weeks_price**: the percentage change in price of the stock in the following week
- **days_to_next_dividend**: the number of days until the next dividend
- **percent_return_next_dividend**: the percentage of return on the next dividend

In order to easier process the data, various changes were made to the data set. The quarter and stock variables were made into factors. The levels of the stock variables were then stored for later use with the functions' for-loop. Dollar signs were then removed from the `open`, `high`, `low`, `close`, `next_weeks_close`, and `next_weeks_open` variables so that there would be no issues when processing the data and using it to make predictions.

All variables that contained data from the following week were removed from the data set. As the goal of our model is to be predictive, information from the following week shouldn't be used as it would not be known when making a real-time prediction. `Lag` variables were then created for each numerical variable for each stock in the data set. These lag variables were referred to as their base variable's name with `.lag` added to the end.

In order to create predictive models, all observations with `NA` values were removed from the dataset. This resulted in the removal of observations for the first and second week for each stock, as the first week of each stock has no data for variables that rely on the prior week, and the missing values in the first week create `NA` values in the `lag` variables for the second week of each stock.

One goal was to look at `CAPM` analysis to understand the relationship between systematic risk and expected return for stocks compared to market. Accordingly, the `S&P 500` data was downloaded from yahoo finance for respective weeks as the Dow Jones Index. The `beta values` for each stock were calculated via `linear regression`, with the stock as the `response` variable and the `S&P 500` as the `input` variable. In order to better understand the stocks, the abbreviated names of each stock were converted into their proper names.

Results

The final input formula that was selected and used for generating the three models used the `Week`, `percent_change_price`, `percent_change_volume_over_last_week`, `open.lag`, and `percent_change_volume_over_last_week.lag`. The first step to determining this formula was removing independent variables with a correlation to the y variable that was less than the average correlation the independent variables had with the y variable. Independent variables were then systematically removed from the input formula until the `VIF` values for all inputted variables reached acceptable amounts, less than 5.

Of the three models tested on the data, the `SVR` model was selected to be the final model. The average `mean squared errors` was used as the evaluation metric for the three models. Each model type was run for each individual stock, using the same input formula,with the mean square error for each "run" being recorded. The average for each model type's summed mean square errors was then taken.

The linear model produced an average mean square value of roughly 64.08, much greater than the amount produced by the `Tree` and `SVR` models. The `Tree` and `SVR` models produced average mean square values of roughly 9.57 and 8.93, respectively. The `SVR` model was chosen to proceed with, as it had the lowest average `mean square error`. From this point, the model was then used in order to make a prediction for the response variable for the last week of every stock.

```
include_graphics("https://i.imgur.com/MjMlV01.png") #image of final Dataframe
```

	VariableNames	Percent Change Prediction for Next Week	Beta Value
1	Alcoa	0.6720748	0.6528
2	American Express	-1.1867045	0.5341
3	Boeing	0.2800098	0.6655
4	Bank of America	-0.5080646	0.4177
5	Caterpillar	2.3461575	0.7797
6	Cisco	-1.5366819	0.4959
7	Chevron	1.8740887	1.0543
8	DuPont	3.0893889	0.6350
9	Walt Disney	0.4114028	0.6848
10	General Electric	1.0051864	0.7048
11	Home Depot	0.1720837	0.7302
12	HP	-0.4980126	0.0817
13	IBM	2.1269750	0.5859
14	Intel	-1.0811523	1.2194
15	J&J	0.2248138	0.6424
16	JPMorgan	0.2164145	0.2473
17	Coca-Cola	1.3309479	-0.0072
18	KRFT	0.6944458	4.3939
19	McDonald's	1.2560624	0.1980
20	3M	0.4053232	0.3144
21	Merck	-0.3907161	-0.2242
22	Microsoft	-0.8900265	0.2837
23	Pfizer	-0.1468189	-0.1069
24	Procter & Gamble	-0.5502820	-0.2463
25	AT&T	0.4839640	0.3558
26	Travelers	0.5525911	0.4186
27	Raytheon	0.2955521	1.0773
28	Verizon	0.8612156	0.1234
29	Walmart	0.1890913	0.4348
30	Exxon	0.3216285	0.9247

Conclusions and Recommendations

There is no guaranteed way in which to use Dow Jones Index data to make future predictions about the market. As mentioned in the research section, even small improvements in predictive modeling can bring great rewards in an industry with so much capital involved and leverage used. Our analysis found that a model using `Support Vector Regression` was the most accurate at predicting next-week stock prices. The `linear` model, while much more inaccurate than our models based on `SVR` or `Decision Tree`, still proved to be useful in identifying our five predictors.

Based on our final results, we identified that the `percent_change_price`, `percent_change_volume_over_last_wk`, `open.lag`, and `percent_change_volume_over_last_wk.lag` variables have a significant impact on our response variable. Due to the `SVR` model's lack of interpretability, it is difficult to determine which variable is more significant than others.

Based on the `Beta` values generated using `CAPM` analysis combined with the predictions generated for individual stocks using the `SVR` model, we recommend investment in `DuPont`, `IBM`, and `Caterpillar` stocks. The three companies have `Beta` of less than 1, indicating these as low risk as compared to the market. Additionally, these stocks have the highest `Percentage Change Prediction for Next Week` variable, indicating an overall higher rate of return compared to other stocks in the portfolio. Even though the actual percentage change for all three stocks was higher compared to the prediction, none of the three stocks resulted in lower change than predicted. We believe adding these stocks to theportfolio will result in overall positive earnings for the investors.

Predicting the percentage change in a stock is only half of the equation. A stock that is predicted to gain 100 percent, but runs the risk of losing 200 percent in any given week may not be a profitable investment long-term. This is where the CAPM model will help the potential investor identify both low-beta (low-risk) stocks and stocks with potential for high performance. Our combined model assists investors in first identifying stocks with the highest predicted return, then choosing stocks with the lowest beta values for the most effective investment strategy.

References

Hegazy, O., Soliman, O., & Salam M. (2013). A Machine Learning Model for Stock Market Prediction. International Journal of Computer Science
and Telecommunications, 4, 17-23.

Kumbhare, P., Makhija, R., & Raichandani, H. (2019). Stock Market Prediction. Santa Clara University.

Nguyen, T., & Yoon, S. (2019). A Novel Approach to Short-Term Stock Price Movement Prediction using Transfer Learning. Applied Sciences,
9(22), 4745–.

Sert, O., Şahin, S., Özyer, T., & Alhajj, R. (2020). Analysis and prediction in sparse and high dimensional text data: The case of Dow Jones stock
market. Physica A, 545, 123752–.

Qian, B., Qian, B., Rasheed, K., & Rasheed, K. (2007). Stock market prediction with multiple classifiers. Applied Intelligence (Dordrecht
, Netherlands), 26(1), 25–33.

Appendix

Importing the Data

```
DJI <- read.csv("dow_jones_index.data") #25 of each stock in the dataset
```

Cleaning Data

```
DJI1 <- DJI

DJI1$quarter <- as.factor(DJI1$quarter)

DJI1$date <- as.Date(DJI1$date, "%m/%d/%Y")
DJI1$open = as.numeric(gsub("\\$", "", DJI1$open))
DJI1$high = as.numeric(gsub("\\$", "", DJI1$high))
DJI1$low = as.numeric(gsub("\\$", "", DJI1$low))
DJI1$close = as.numeric(gsub("\\$", "", DJI1$close))
DJI1$next_weeks_close = as.numeric(gsub("\\$", "", DJI1$next_weeks_close))
DJI1$next_weeks_open = as.numeric(gsub("\\$", "", DJI1$next_weeks_open))
DJI1$stock <- as.factor(DJI1$stock)
stocknames = levels(DJI1$stock)
```

```
Week = strftime(DJI1$date, format = "%V")
DJI1 = cbind(Week, DJI1)
DJI1$Week <- as.numeric(DJI1$Week)
DJI1 <- DJI1[order(DJI1$stock),]
str(DJI1)
```

```
## 'data.frame':    750 obs. of  17 variables:
##  $ Week                             : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ quarter                          : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ stock                            : Factor w/ 30 levels "AA","AXP","BA",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ date                             : Date, format: "2011-01-07" "2011-01-14" ...
##  $ open                             : num  15.8 16.7 16.2 15.9 16.2 ...
##  $ high                             : num  16.7 16.7 16.4 16.6 17.4 ...
##  $ low                              : num  15.8 15.6 15.6 15.8 16.2 ...
##  $ close                            : num  16.4 16 15.8 16.1 17.1 ...
##  $ volume                           : int  239655616 242963398 138428495 151379173 154387761 114691279 80023895 1
32981863 109493077 114332562 ...
##  $ percent_change_price             : num  3.79 -4.43 -2.47 1.64 5.93 ...
##  $ percent_change_volume_over_last_wk: num  NA 1.38 -43.02 9.36 1.99 ...
##  $ previous_weeks_volume            : int  NA 239655616 242963398 138428495 151379173 154387761 114691279 8002389
5 132981863 109493077 ...
##  $ next_weeks_open                  : num  16.7 16.2 15.9 16.2 17.3 ...
##  $ next_weeks_close                 : num  16 15.8 16.1 17.1 17.4 ...
##  $ percent_change_next_weeks_price  : num  -4.428 -2.471 1.638 5.933 0.231 ...
##  $ days_to_next_dividend            : int  26 19 12 5 97 90 83 76 69 62 ...
##  $ percent_return_next_dividend     : num  0.183 0.188 0.19 0.186 0.175 ...
```

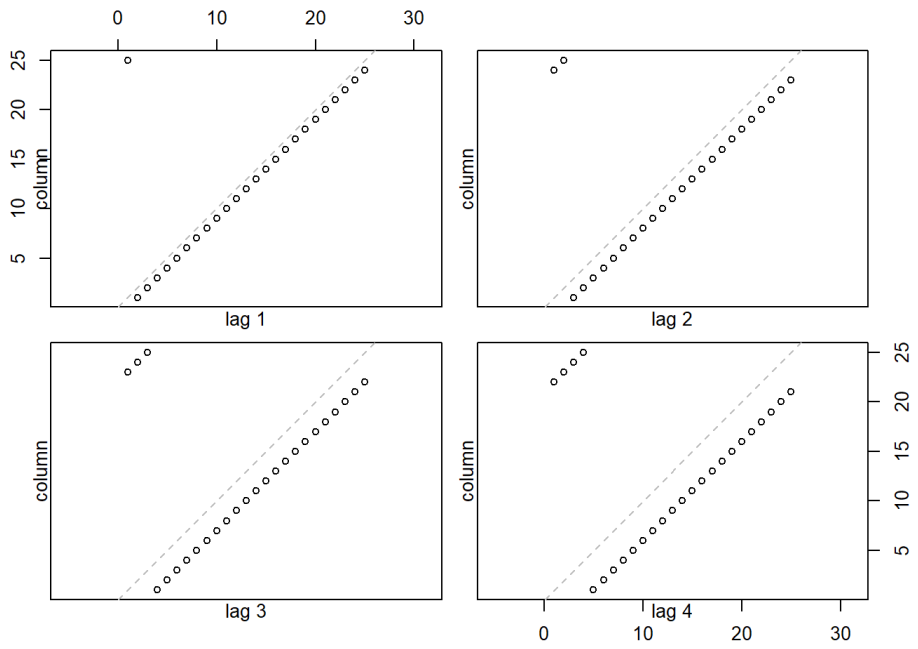
```
DJI1 <- DJI1[-c(13:14,16:17)] #Not using "next" columns as predictors
```

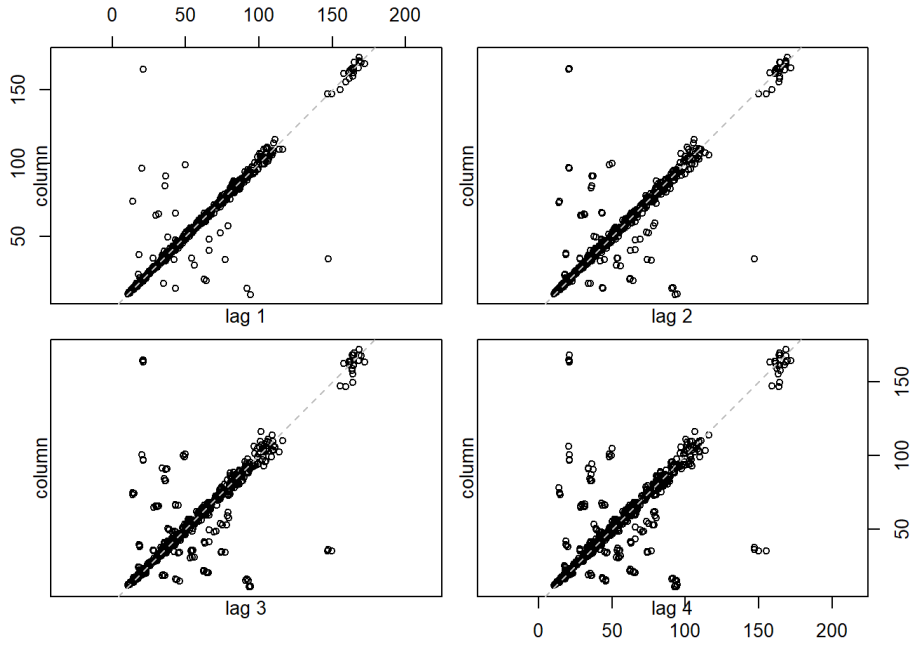
Creating lag plots

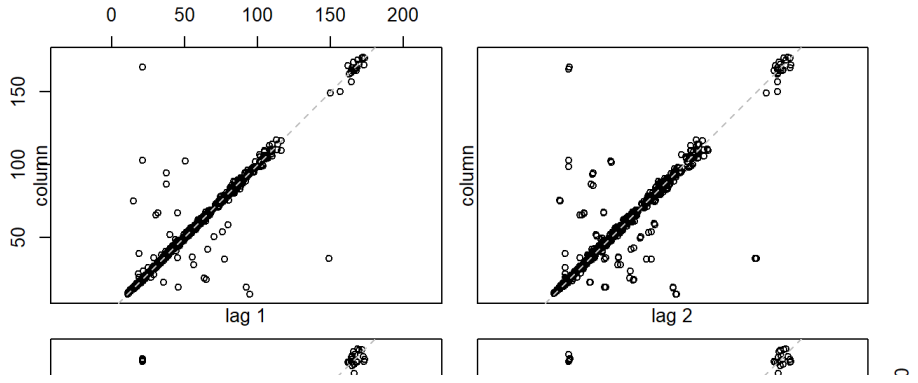
```
CNamesFunction <- function(dataframe)
{
  ColumnNames = colnames(dataframe)
}
```

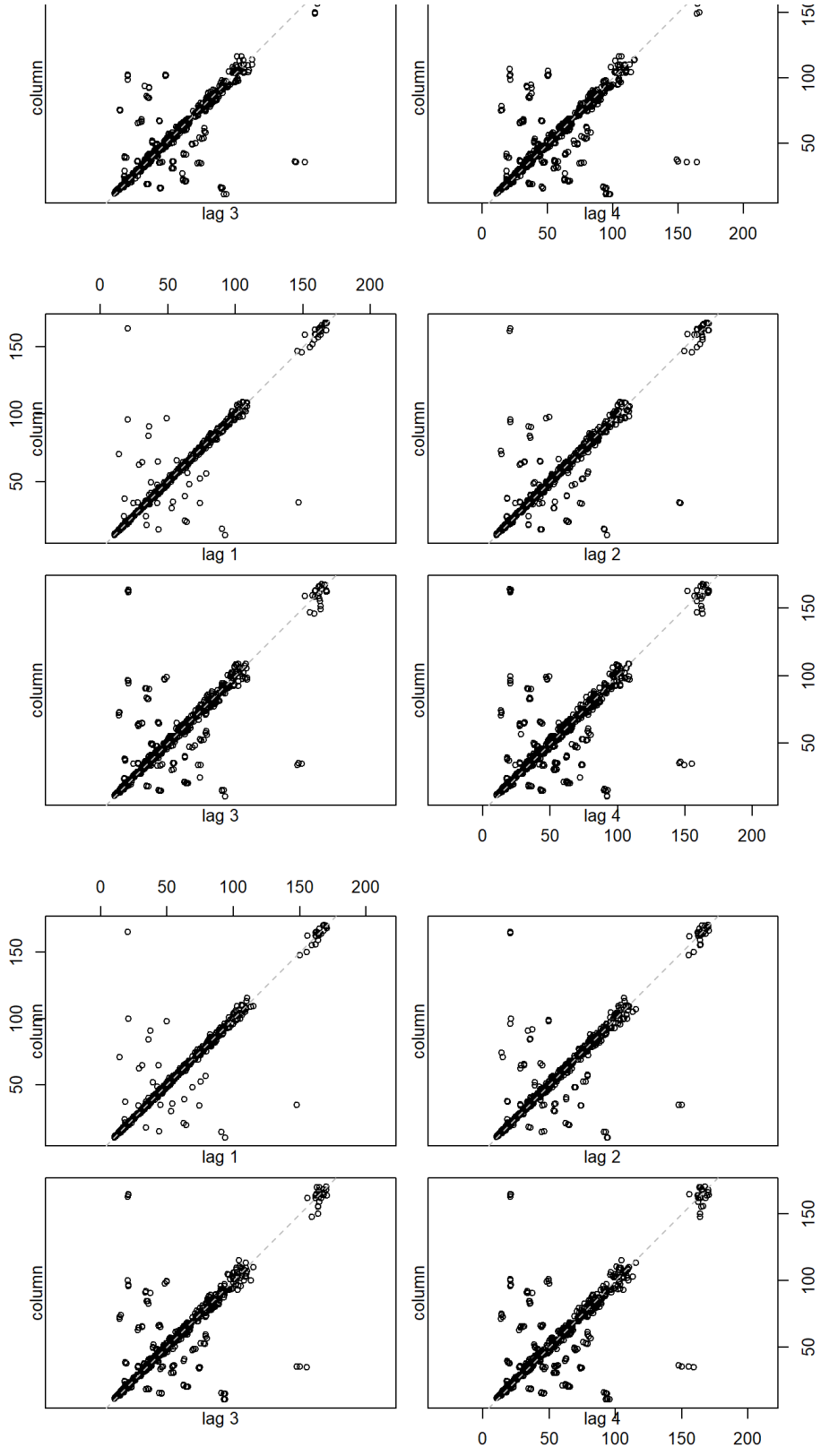
```
LagPlotFunc <- function(column, lagnumber)
{
  lag.plot(column, set.lags = 1:lagnumber)
}
```

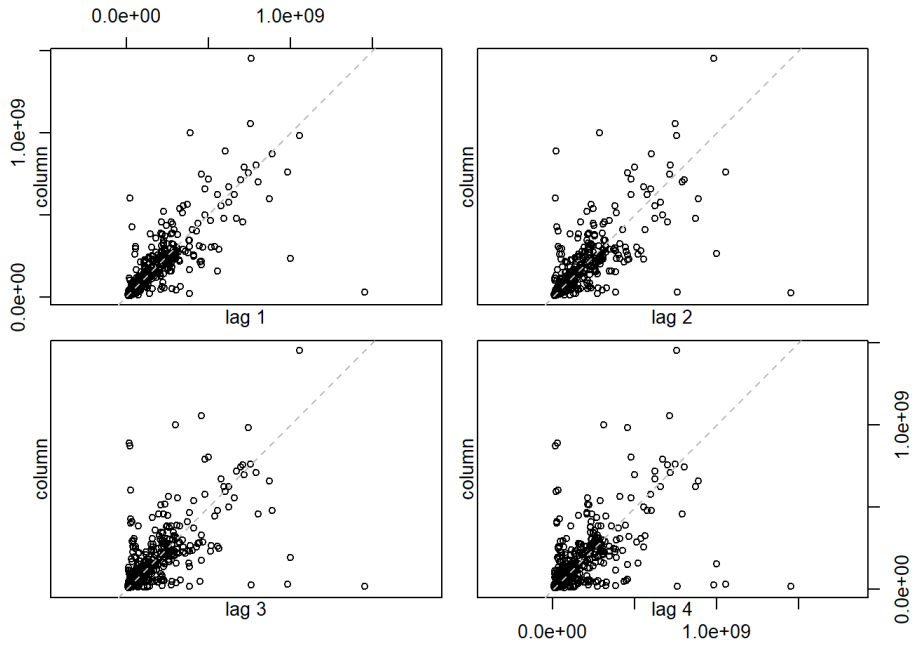
```
LagVars = CNamesFunction(Filter(is.numeric, DJI1)[,1:10]) # only first 10 columns, all other numeric variables are la
g variables

for(i in LagVars)
{
data = DJI1[[i]]
 LagPlotFunc(na.omit(data),4)
}
```

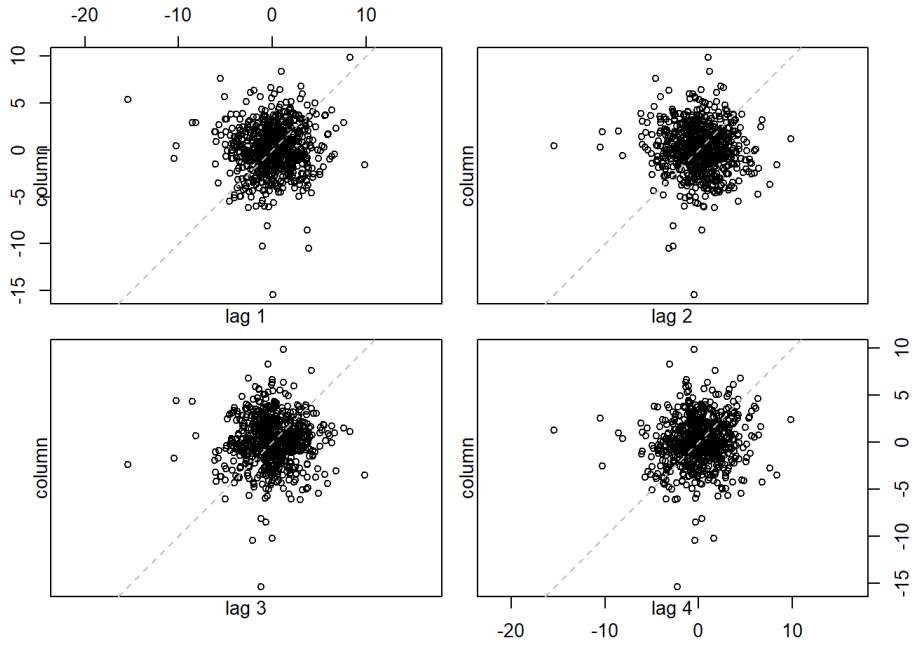

0
10
20
30
5
10
15
20
25
column
lag 1
lag 2
lag 3
lag 4

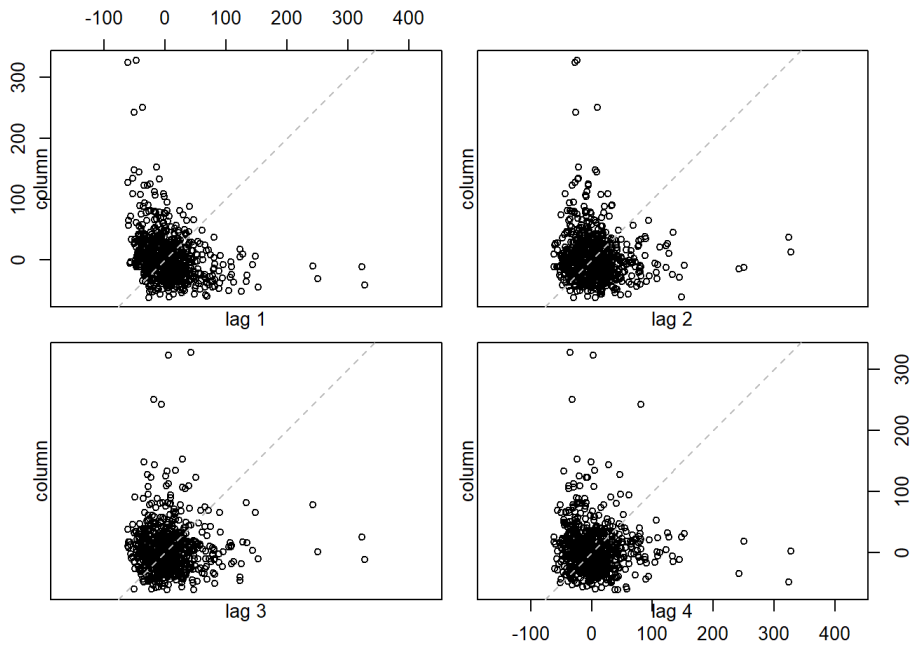

0
50
100
150
200
50
100
150
column
lag 1
lag 2
lag 3
lag 4

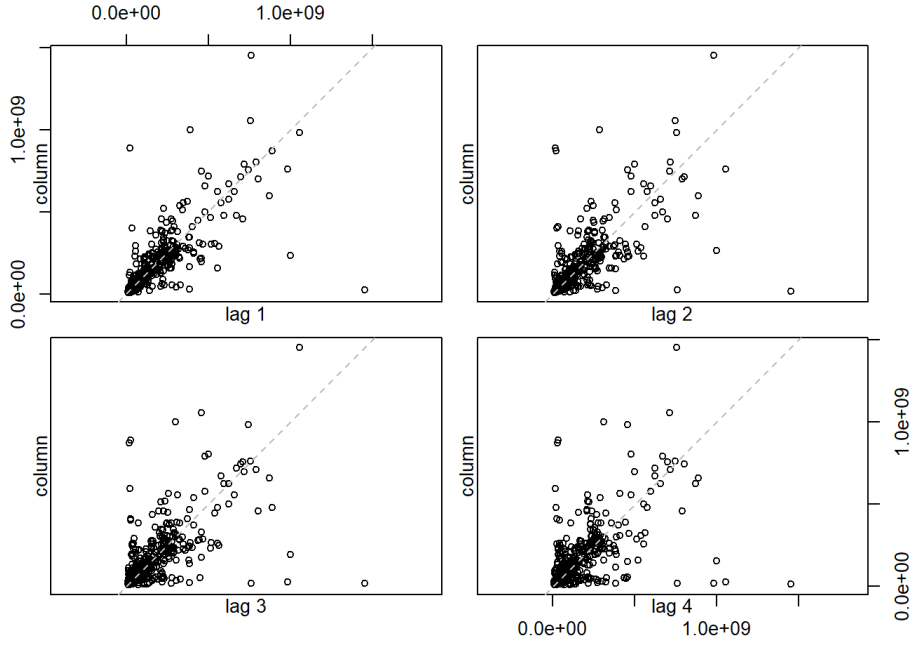

0
50
100
150
200
50
100
150
column
lag 1
lag 2

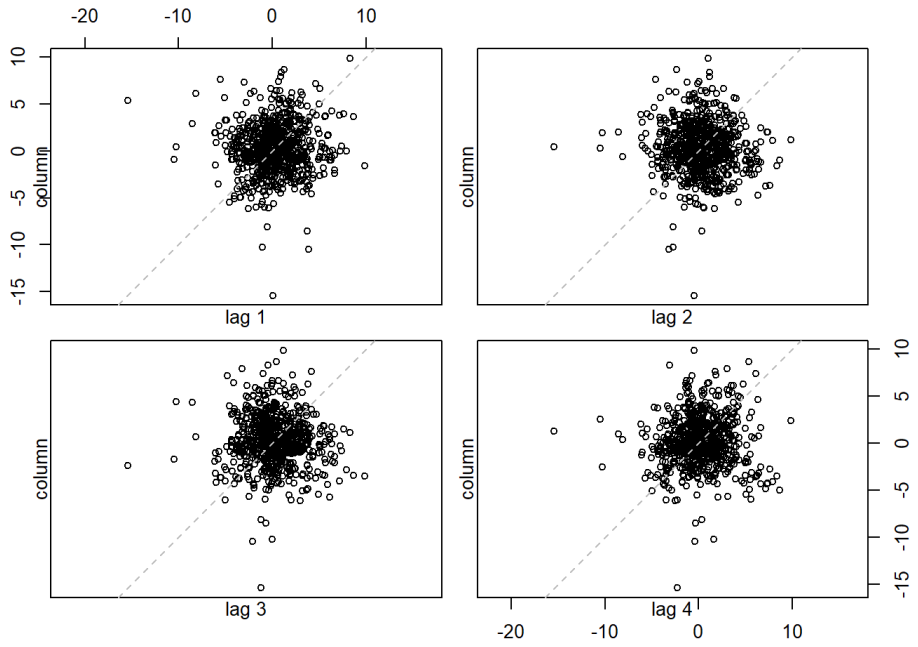

column
lag 3
lag 4
0
50
100
150
200
lag 1
lag 2


0.0e+00
1.0e+09
column
lag 1
lag 2
lag 3
lag 4


-20
-10
0
10
column
lag 1
lag 2
lag 3
lag 4


-100
0
100
200
300
400
0
100
200
300
column
lag 1
column
lag 2
column
lag 3
column
lag 4


0.0e+00
1.0e+09
0.0e+00
1.0e+09
column
lag 1
column
lag 2
column
lag 3
column
lag 4



Creating lag variables

```
LagValFunction <-  function(variname)
{
  print(lag(variname, 1))
}
```

```
open.lag <- c()
high.lag <- c()
low.lag <- c()
close.lag <- c()
volume.lag <- c()
percent_change_price.lag <- c()
percent_change_volume_over_last_wk.lag <- c()
previous_weeks_volume.lag <- c()
percent_change_next_weeks_price.lag <- c()
```

```
for(i in stocknames)
{
  x = filter(DJI1,stock == i)

  open.lag <- c(open.lag,LagValFunction(x$open))
  high.lag <- c(high.lag,LagValFunction(x$high))
  low.lag <- c(low.lag,LagValFunction(x$low))
  close.lag <- c(close.lag,LagValFunction(x$close))
  volume.lag <- c(volume.lag,LagValFunction(x$volume))
  percent_change_price.lag <- c(percent_change_price.lag,LagValFunction(x$percent_change_price))
  percent_change_volume_over_last_wk.lag <- c(percent_change_volume_over_last_wk.lag,LagValFunction(x$percent_change_
volume_over_last_wk))
  previous_weeks_volume.lag <- c(previous_weeks_volume.lag,LagValFunction(x$previous_weeks_volume))
  percent_change_next_weeks_price.lag <- c(percent_change_next_weeks_price.lag,LagValFunction(x$percent_change_next_w
eeks_price))

}

DJI1 <- cbind(DJI1,open.lag, high.lag, low.lag, close.lag,volume.lag,percent_change_price.lag,percent_change_volume_o
ver_last_wk.lag,previous_weeks_volume.lag,percent_change_next_weeks_price.lag)
```

Splitting the Data

```
DJI1 <- na.omit(DJI1)  #Removes the 1/7/2011 week from all stocks, because the there are missing percent change varia
bles.  This is because they are the first variable in the time series.
head(DJI1)
```

```
##   Week quarter stock       date  open  high   low close    volume
## 3    3       1    AA 2011-01-21 16.19 16.38 15.60 15.79 138428495
## 4    4       1    AA 2011-01-28 15.87 16.63 15.82 16.13 151379173
## 5    5       1    AA 2011-02-04 16.18 17.39 16.18 17.14 154387761
## 6    6       1    AA 2011-02-11 17.33 17.48 16.97 17.37 114691279
## 7    7       1    AA 2011-02-18 17.39 17.68 17.28 17.28  80023895
## 8    8       1    AA 2011-02-25 16.98 17.15 15.96 16.68 132981863
##   percent_change_price percent_change_volume_over_last_wk previous_weeks_volume
## 3            -2.470660                       -43.024959             242963398
## 4             1.638310                         9.355500             138428495
## 5             5.933250                         1.987452             151379173
## 6             0.230814                       -25.712195             154387761
## 7            -0.632547                       -30.226696             114691279
## 8            -1.766780                        66.177694              80023895
##   percent_change_next_weeks_price open.lag high.lag low.lag close.lag
## 3                        1.638310    16.71    16.71   15.64     15.97
## 4                        5.933250    16.19    16.38   15.60     15.79
## 5                        0.230814    15.87    16.63   15.82     16.13
## 6                       -0.632547    16.18    17.39   16.18     17.14
## 7                       -1.766780    17.33    17.48   16.97     17.37
## 8                       -1.368230    17.39    17.68   17.28     17.28
##   volume.lag percent_change_price.lag percent_change_volume_over_last_wk.lag
## 3  242963398                -4.428490                                1.380223
## 4  138428495                -2.470660                              -43.024959
## 5  151379173                 1.638310                                9.355500
## 6  154387761                 5.933250                                1.987452
## 7  114691279                 0.230814                              -25.712195
## 8   80023895                -0.632547                              -30.226696
##   previous_weeks_volume.lag percent_change_next_weeks_price.lag
## 3                 239655616                           -2.470660
## 4                 242963398                            1.638310
## 5                 138428495                            5.933250
## 6                 151379173                            0.230814
## 7                 154387761                           -0.632547
## 8                 114691279                           -1.766780
```

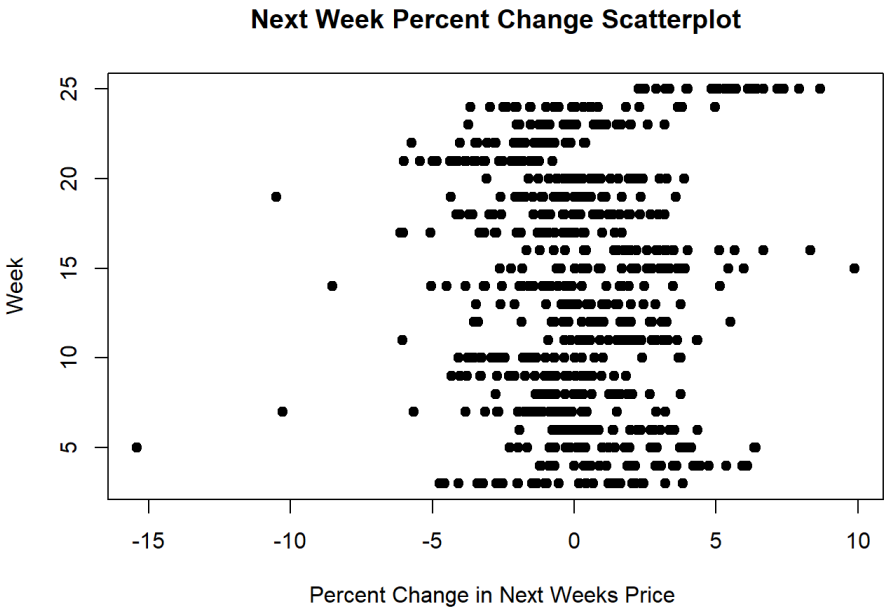
```
DJI1Training <- DJI1[ which(DJI1$quarter=='1'),]
DJI1Test <- DJI1[ which(DJI1$quarter=='2'),]
```

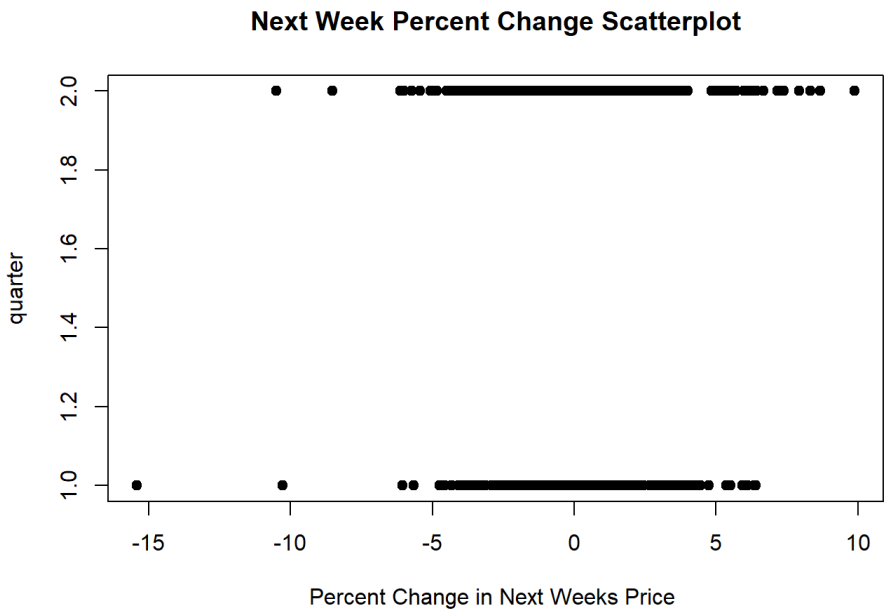
Scatter Plots

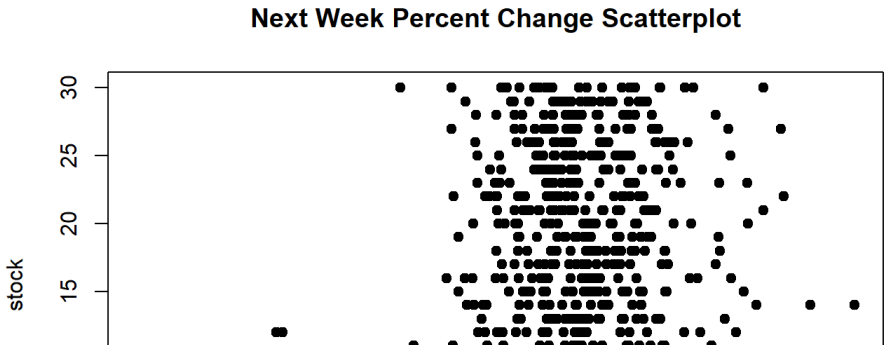
```
PlotAxis = CNamesFunction(DJI1)
```

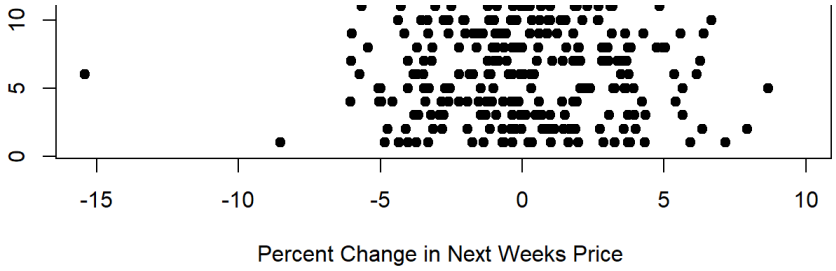
```
ScatterFunction <- function(column, var_name)
{
  plot(DJI1$percent_change_next_weeks_price, column, main="Next Week Percent Change Scatterplot",
   xlab="Percent Change in Next Weeks Price ", ylab = var_name, pch=19)
}
```

```
for(i in PlotAxis)
{
data = DJI1[[i]]
 ScatterFunction(data,i)
}
```


Next Week Percent Change Scatterplot
Week
5
10
15
20
25
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

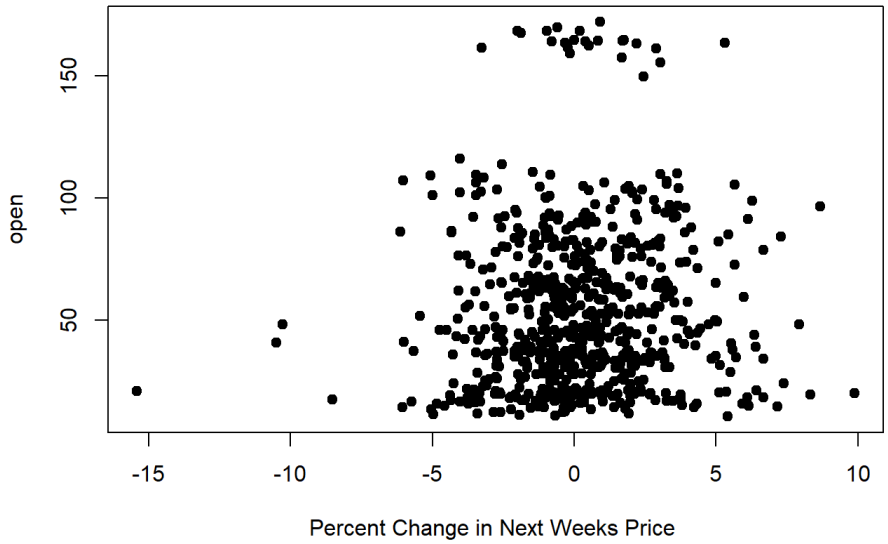

Next Week Percent Change Scatterplot
quarter
1.0
1.2
1.4
1.6
1.8
2.0
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

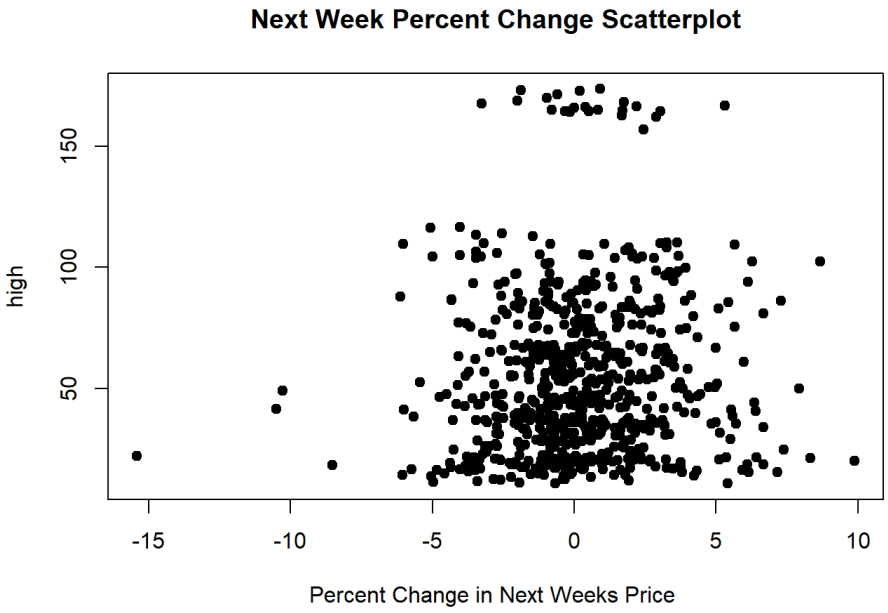

Next Week Percent Change Scatterplot
stock
15
20
25
30

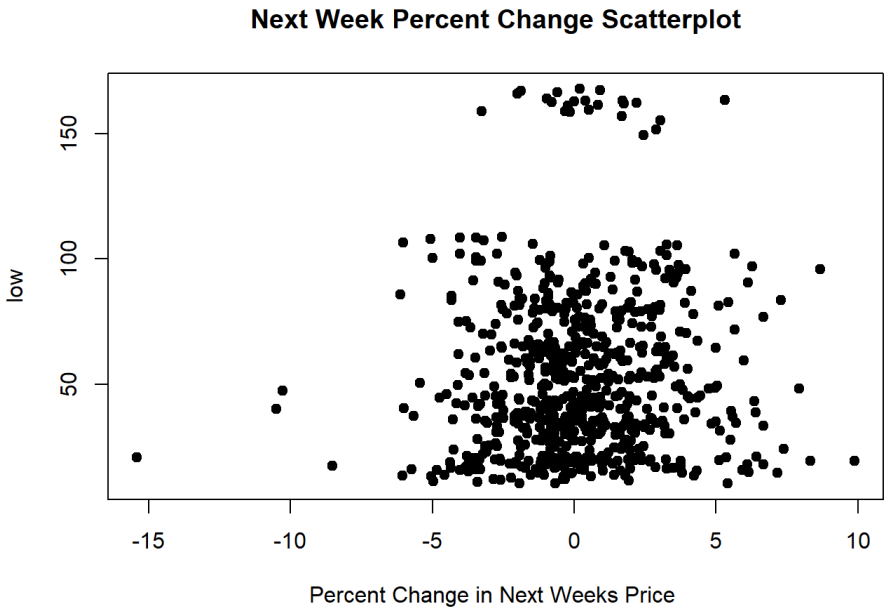


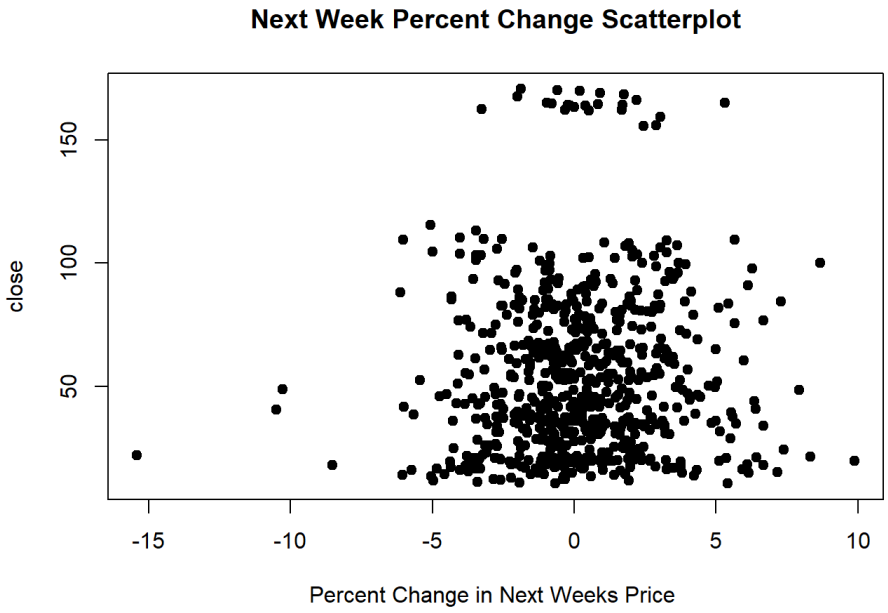
Next Week Percent Change Scatterplot

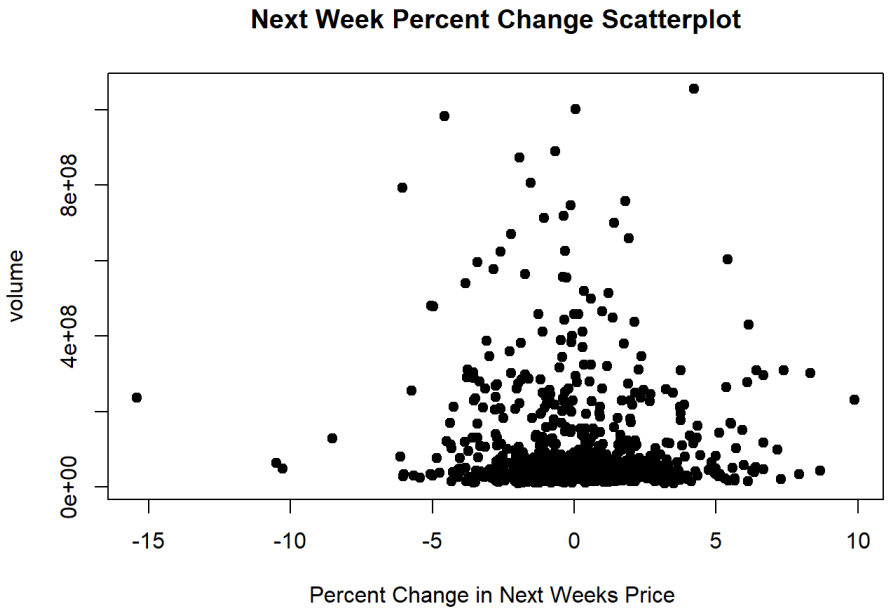
Next Week Percent Change Scatterplot

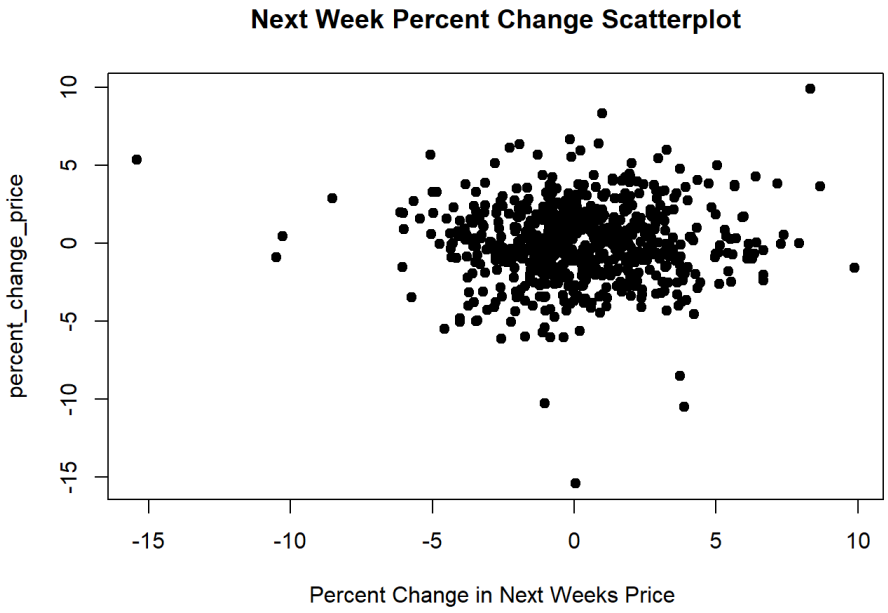


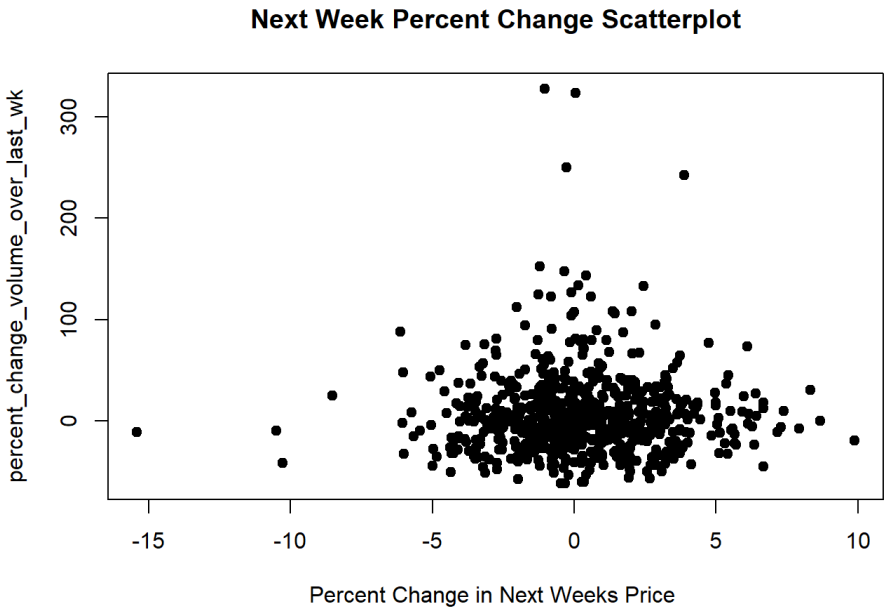

Next Week Percent Change Scatterplot
high
150
100
50
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

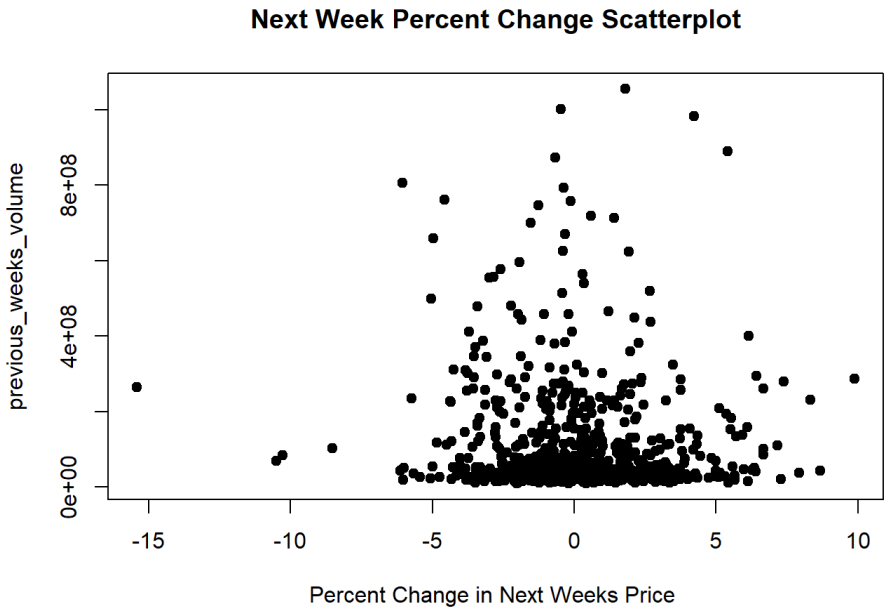

Next Week Percent Change Scatterplot
low
150
100
50
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

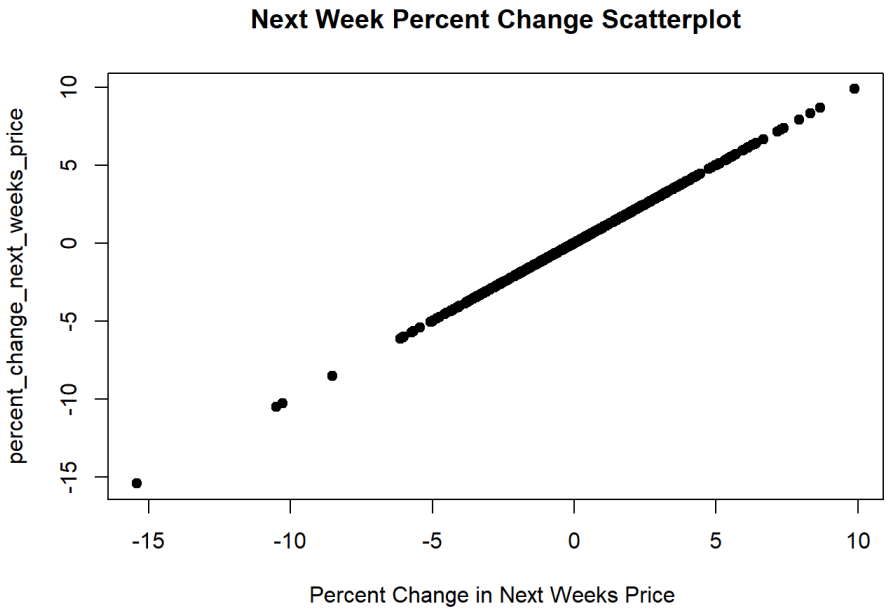

Next Week Percent Change Scatterplot
close
150
100
50
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

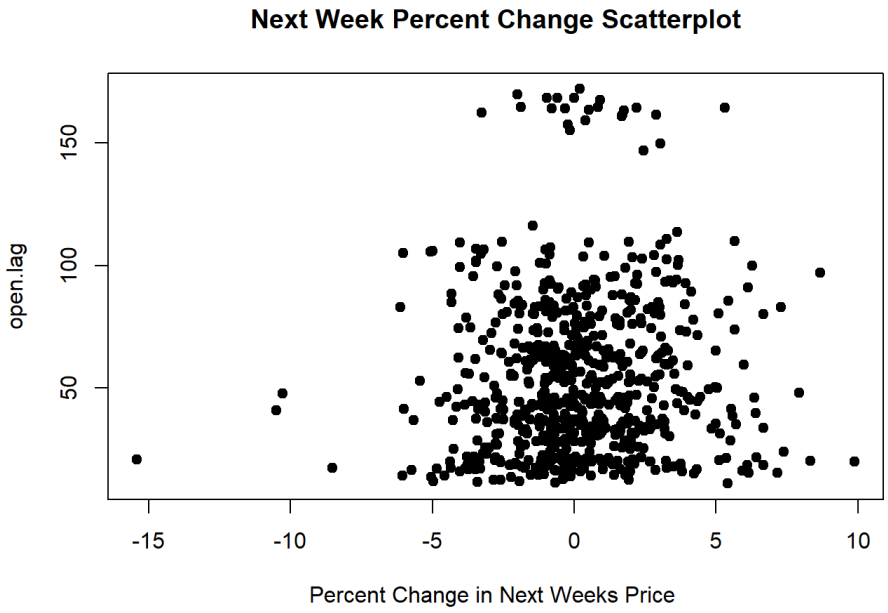

Next Week Percent Change Scatterplot
volume
8e+08
4e+08
0e+00
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

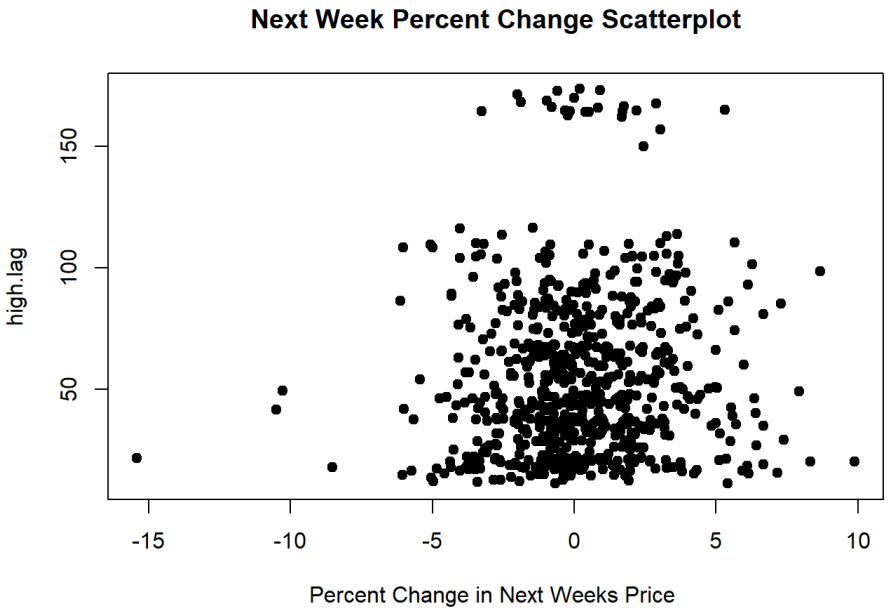

Next Week Percent Change Scatterplot
percent_change_price
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

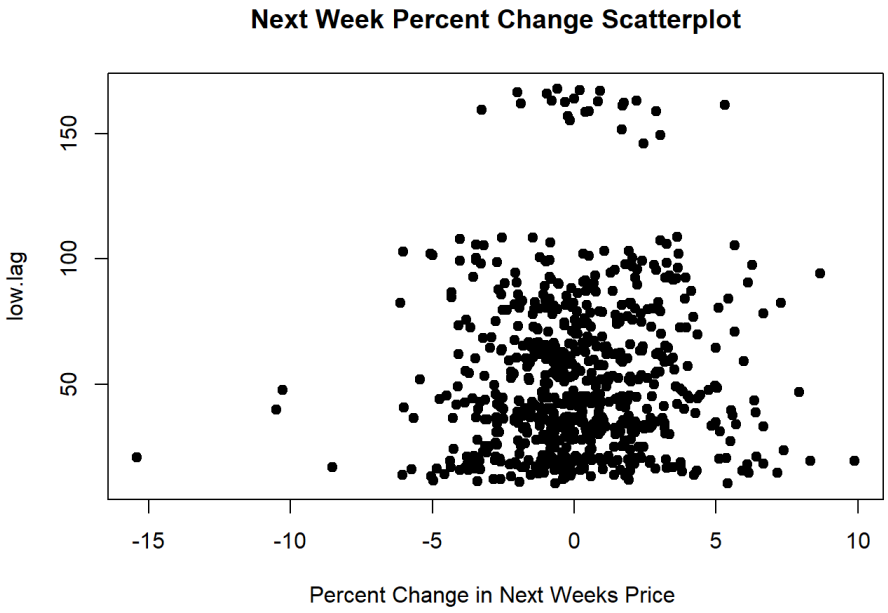

Next Week Percent Change Scatterplot
percent_change_volume_over_last_wk
300
200
100
0
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

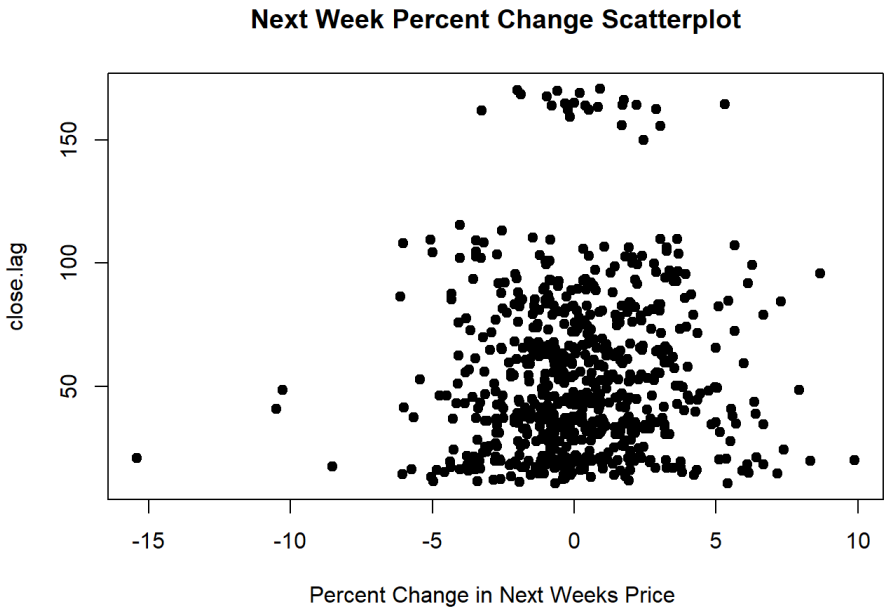

Next Week Percent Change Scatterplot
previous_weeks_volume
0e+00
4e+08
8e+08
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

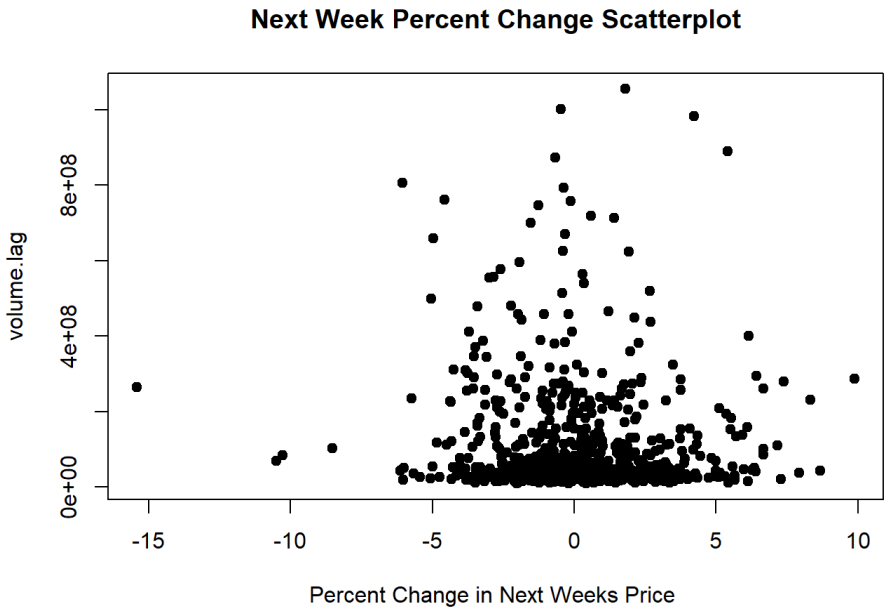

Next Week Percent Change Scatterplot
percent_change_next_weeks_price
-15
-10
-5
0
5
10
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

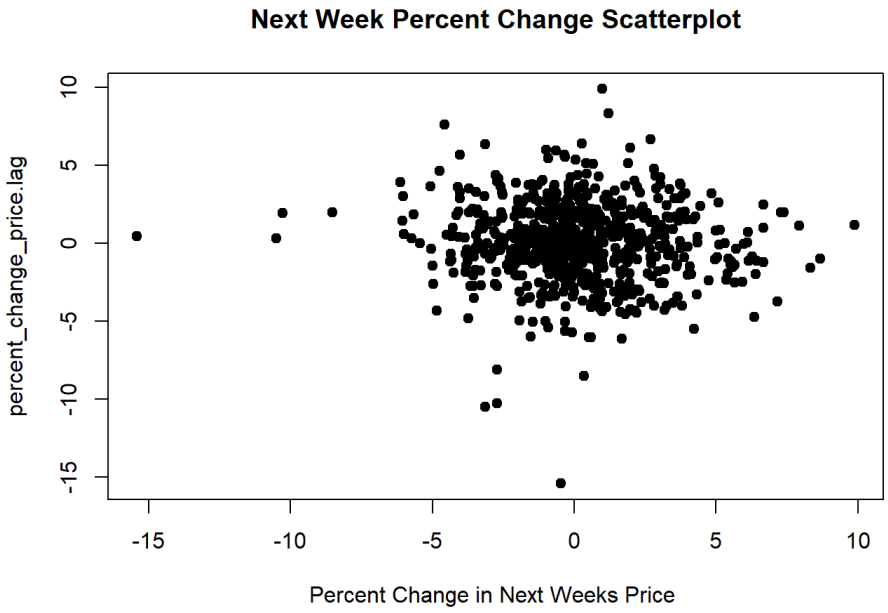

Next Week Percent Change Scatterplot
open.lag
50
100
150
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

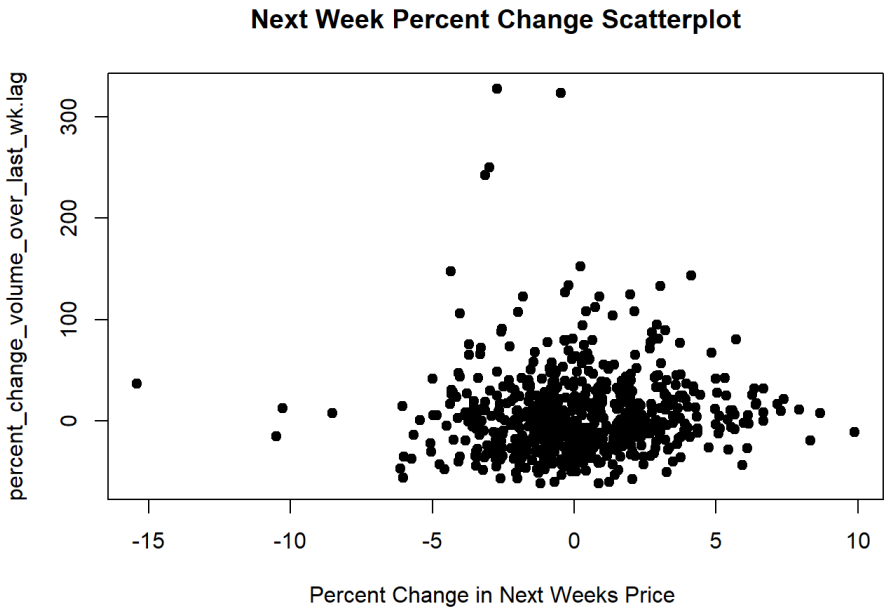

Next Week Percent Change Scatterplot
high.lag
50
100
150
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

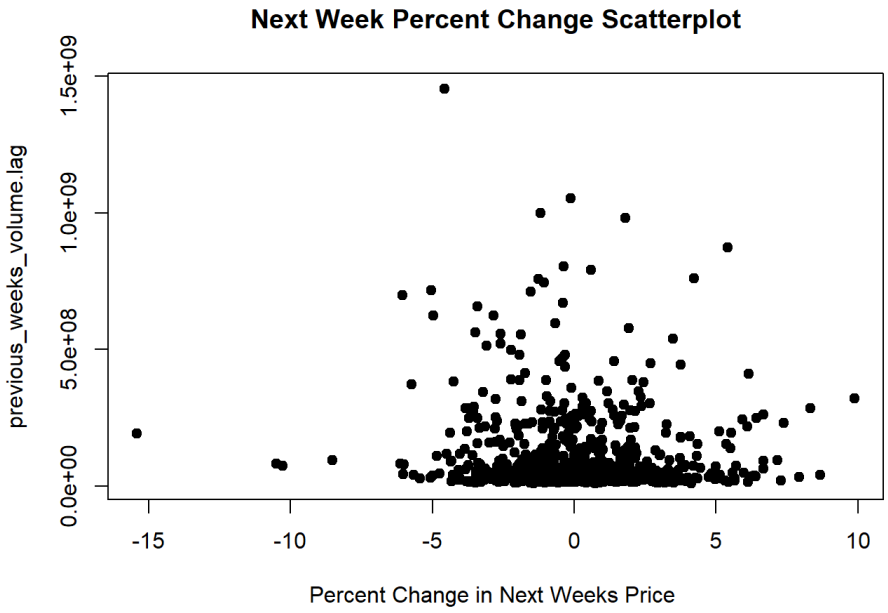

Next Week Percent Change Scatterplot
low.lag
50
100
150
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

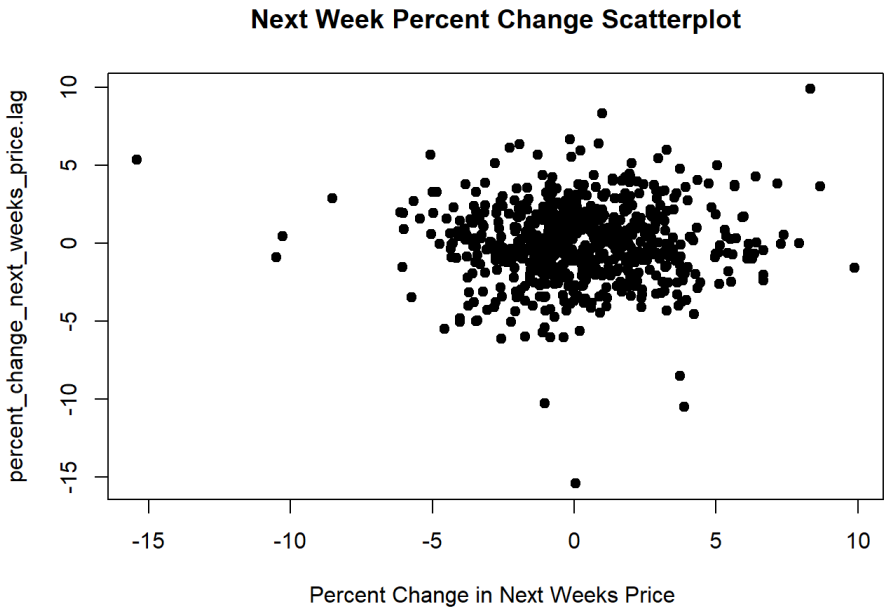

Next Week Percent Change Scatterplot
close.lag
50
100
150
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price


Next Week Percent Change Scatterplot
volume.lag
0e+00
4e+08
8e+08
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

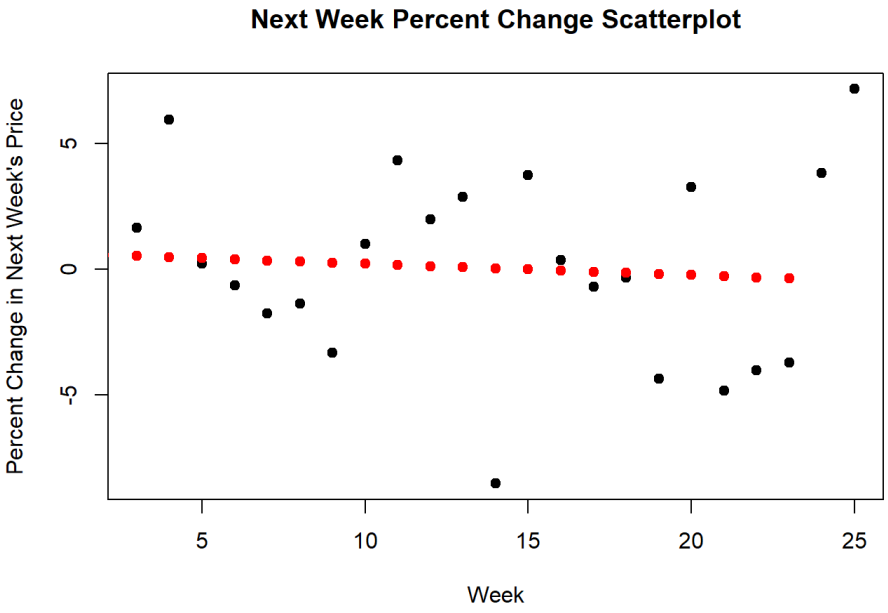

Next Week Percent Change Scatterplot
percent_change_price.lag
-15
-10
-5
0
5
10
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

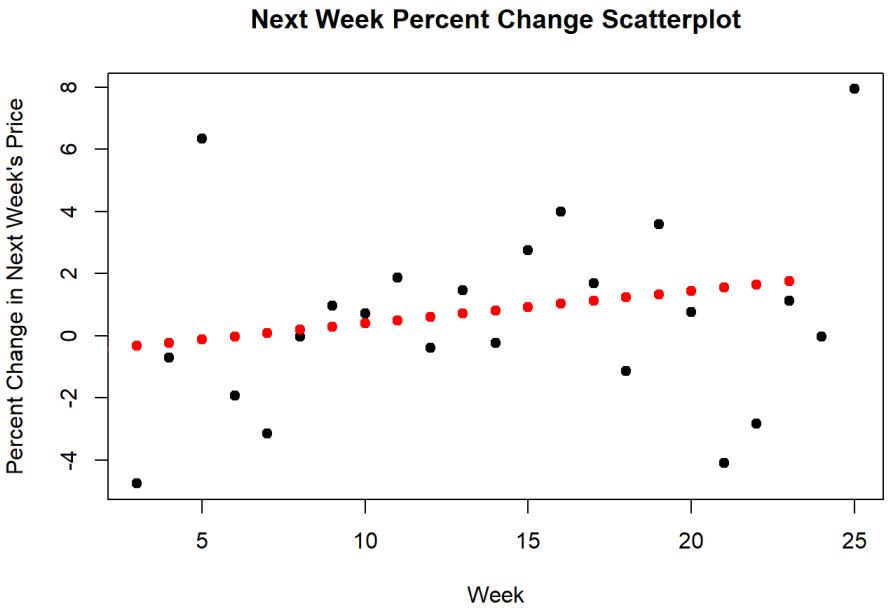

Next Week Percent Change Scatterplot
percent_change_volume_over_last_wk.lag
0
100
200
300
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

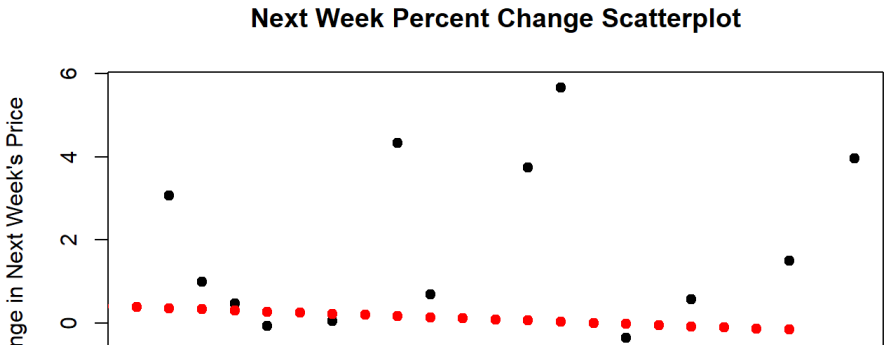

Next Week Percent Change Scatterplot
previous_weeks_volume.lag
0.0e+00
5.0e+08
1.0e+09
1.5e+09
-15
-10
-5
0
5
10
Percent Change in Next Weeks Price

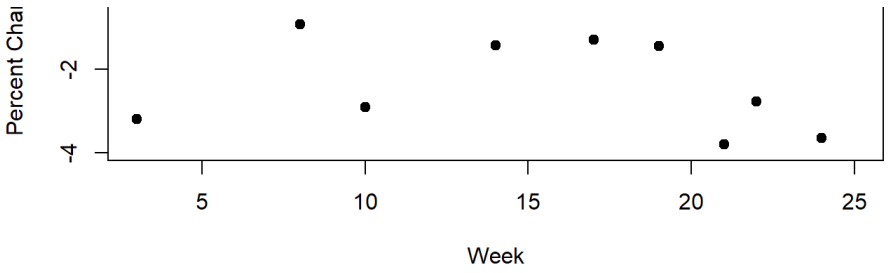


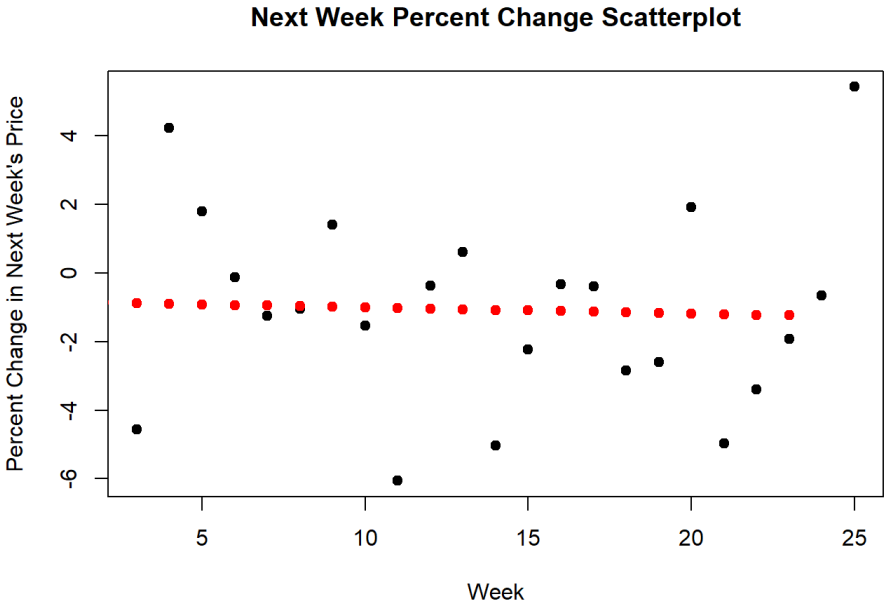

```
ScatterFunction <- function(response, column, xlabel, ylabel)
{
  plot(response, column, main="Next Week Percent Change Scatterplot",
   xlab= xlabel , ylab = ylabel, pch=19)

  points(predict(lm(column ~ response)), col = "red", pch = 19)
}


for (i in stocknames)
{
  x = filter(DJI1, stock == i)

  ScatterFunction(x$Week, x$percent_change_next_weeks_price,  "Week", "Percent Change in Next Week's Price")
}
```

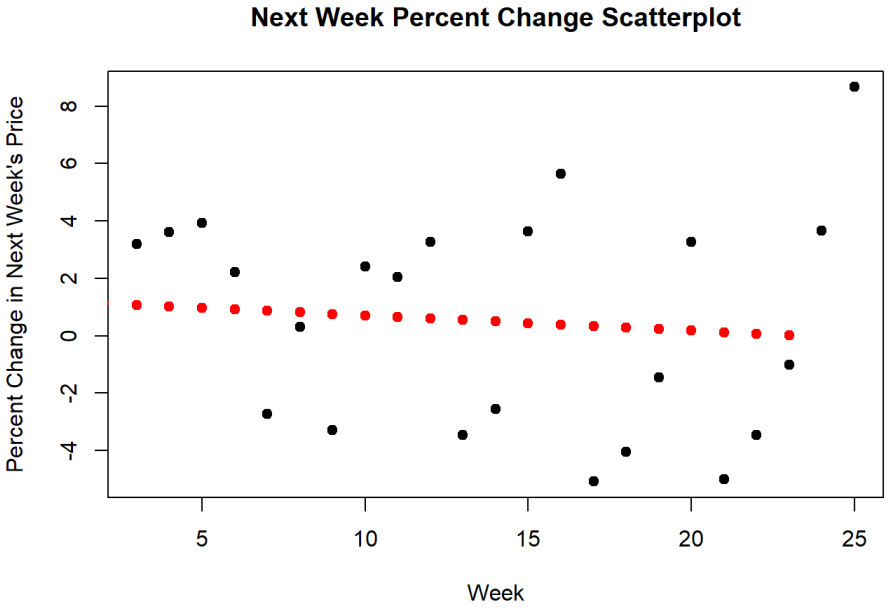

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-5
0
5
5
10
15
20
25
Week

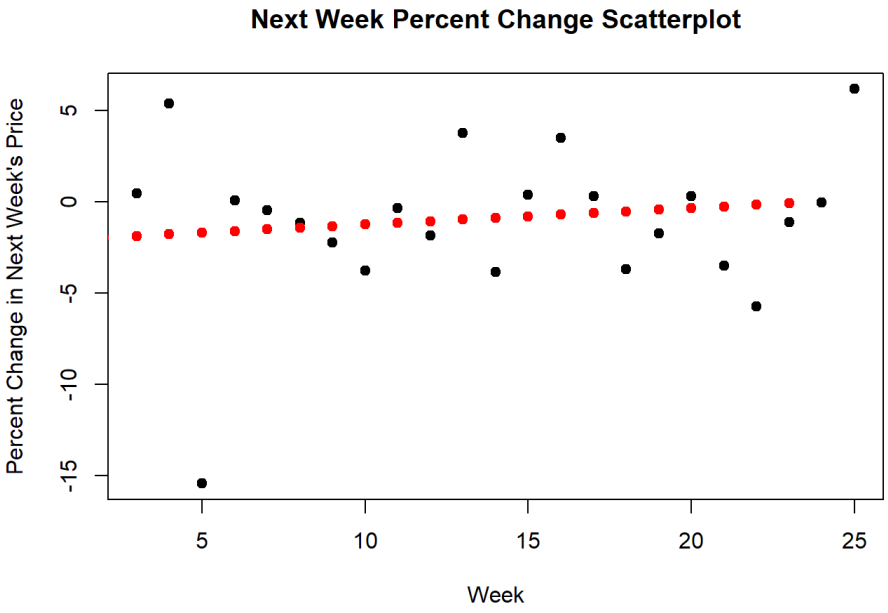

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
8
5
10
15
20
25
Week

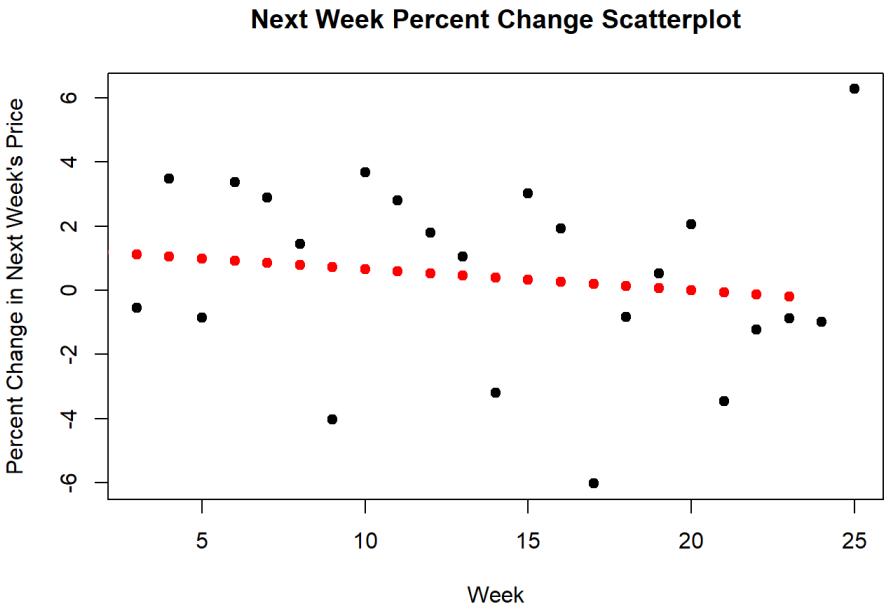

Next Week Percent Change Scatterplot
nge in Next Week's Price
0
2
4
6

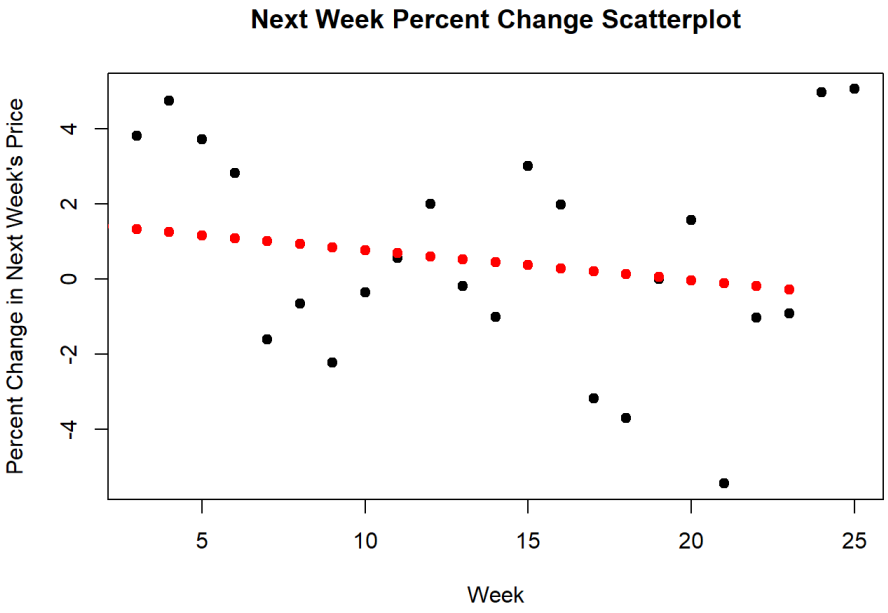

Percent Chai
-4
-2
5
10
15
20
25
Week

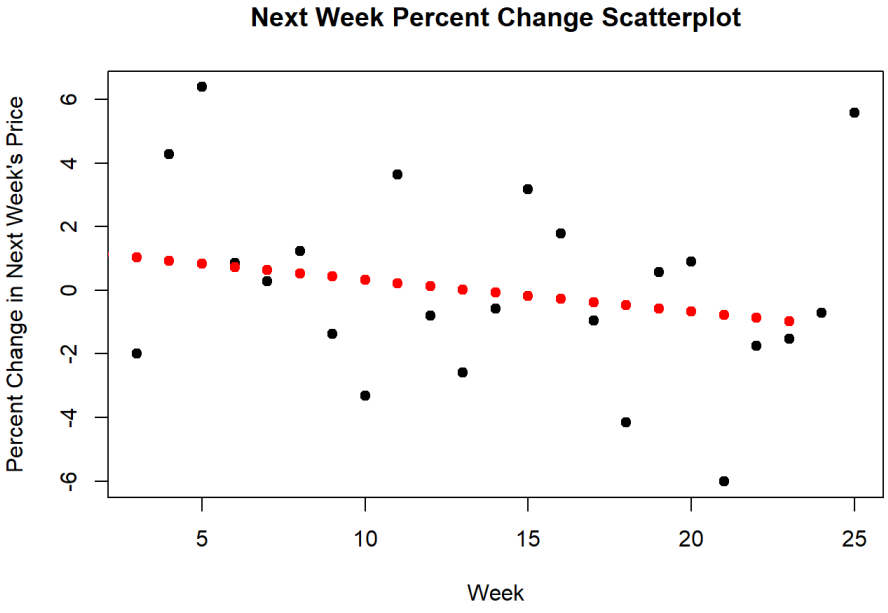

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-6
-4
-2
0
2
4
5
10
15
20
25
Week

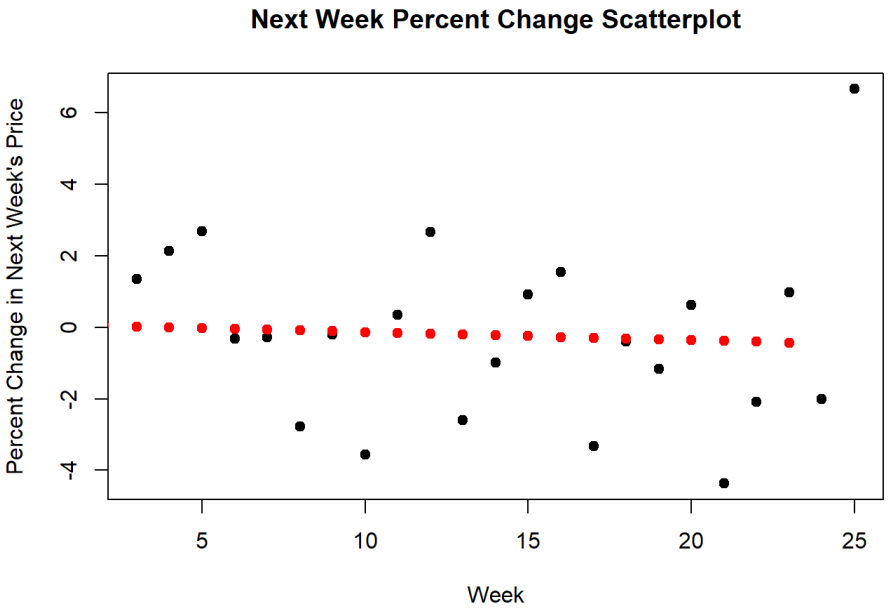

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
8
5
10
15
20
25
Week

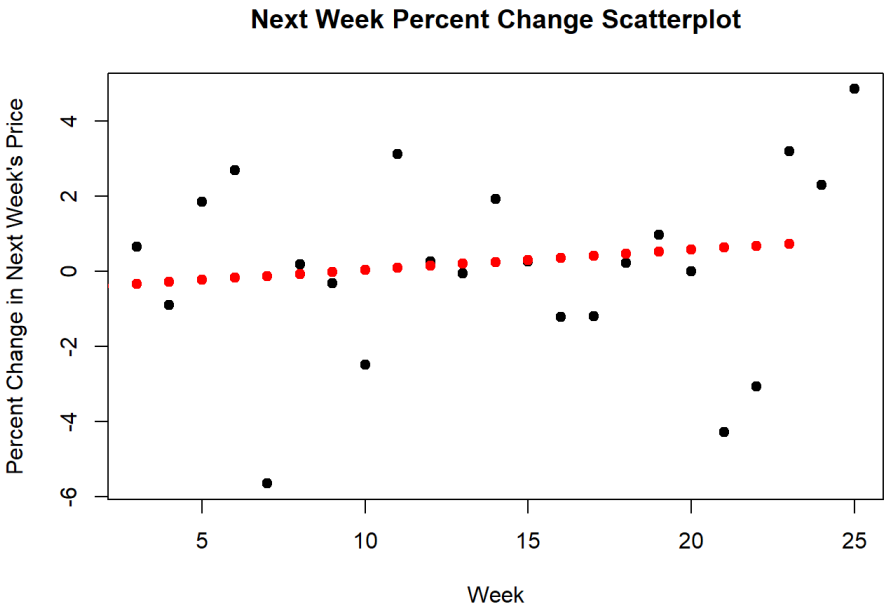

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-15
-10
-5
0
5
5
10
15
20
25
Week

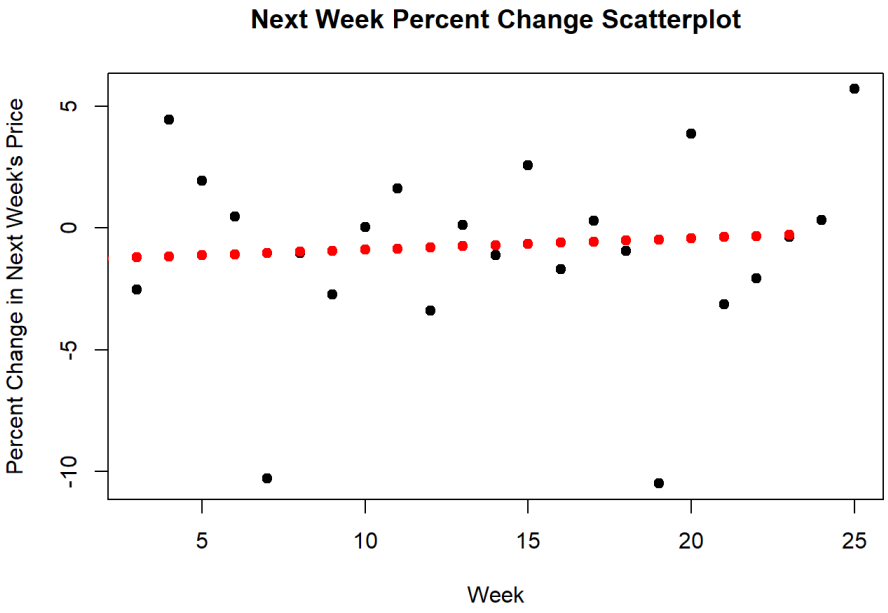

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-6
-4
-2
0
2
4
6
5
10
15
20
25
Week

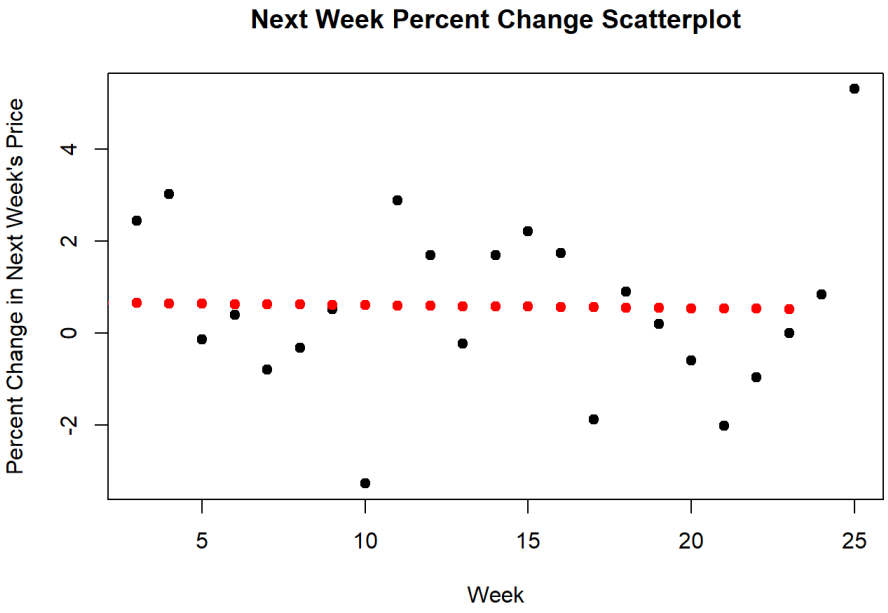

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
5
10
15
20
25
Week

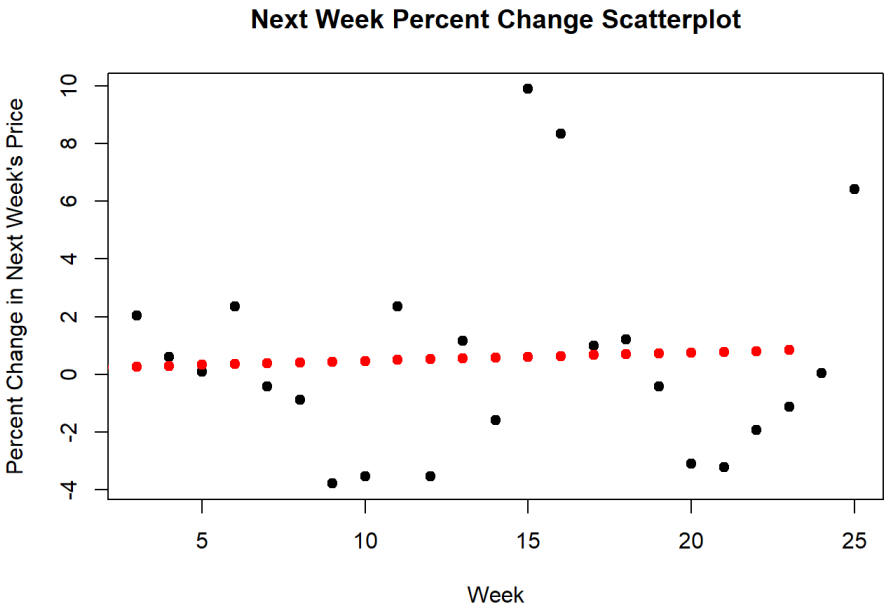

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-6
-4
-2
0
2
4
6
5
10
15
20
25
Week

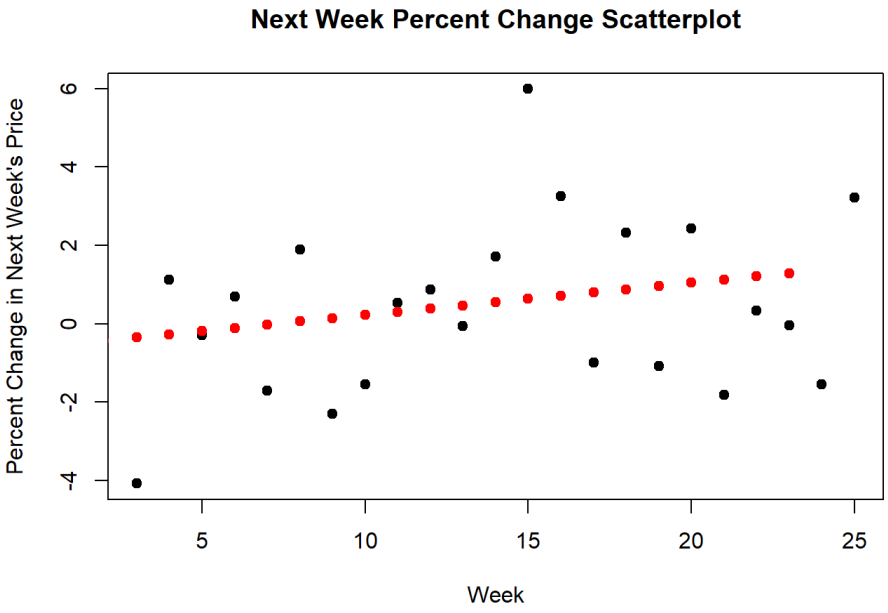

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
5
10
15
20
25
Week

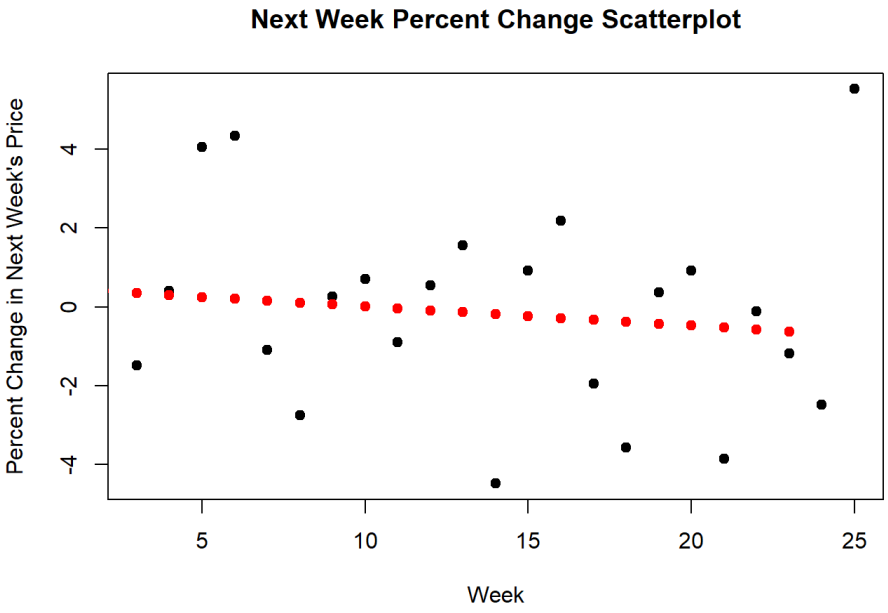

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-6
-4
-2
0
2
4
5
10
15
20
25
Week

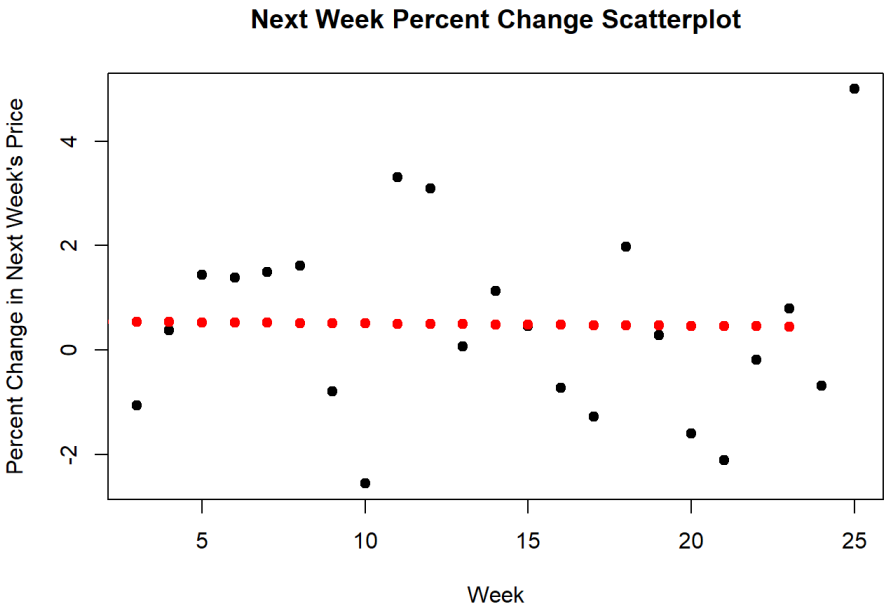

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
5
0
-5
-10
5
10
15
20
25
Week

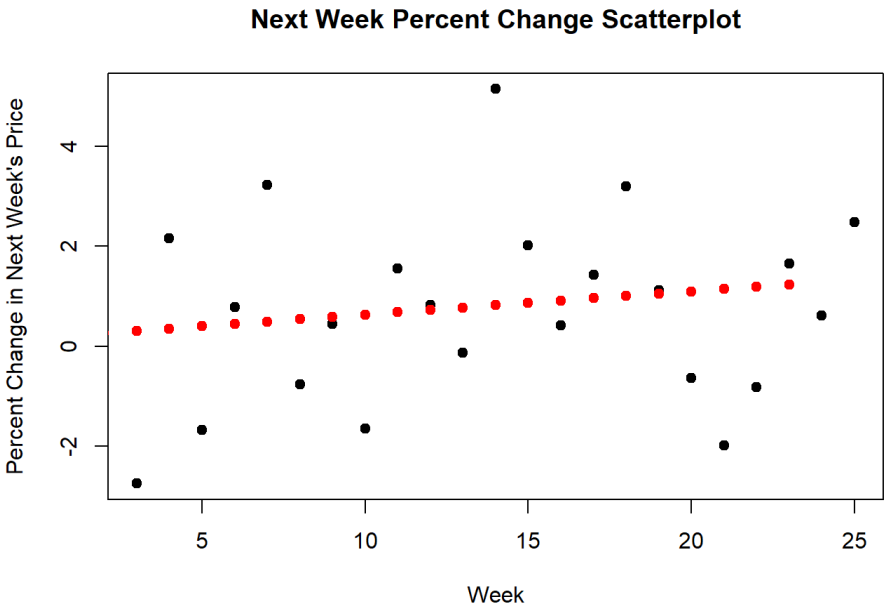

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
4
2
0
-2
5
10
15
20
25
Week

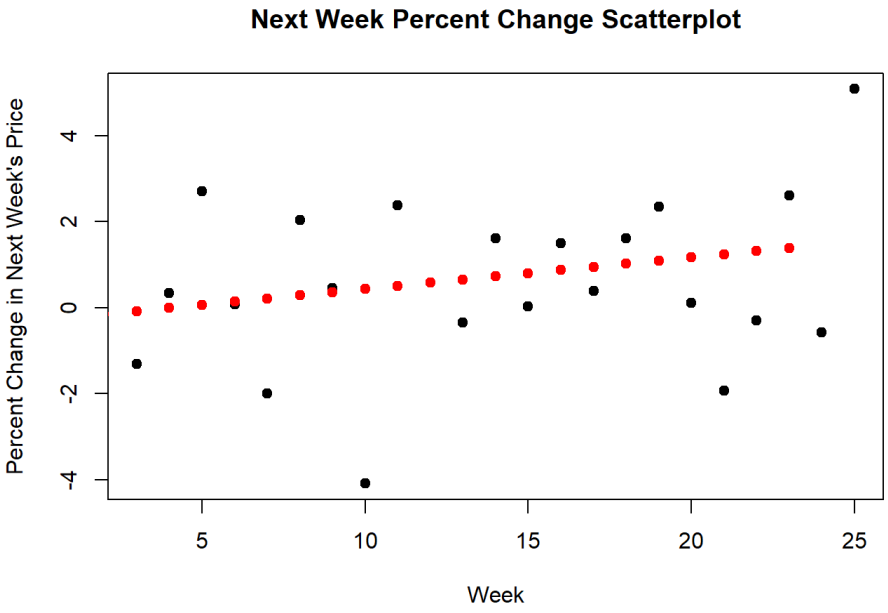

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
8
10
5
10
15
20
25
Week

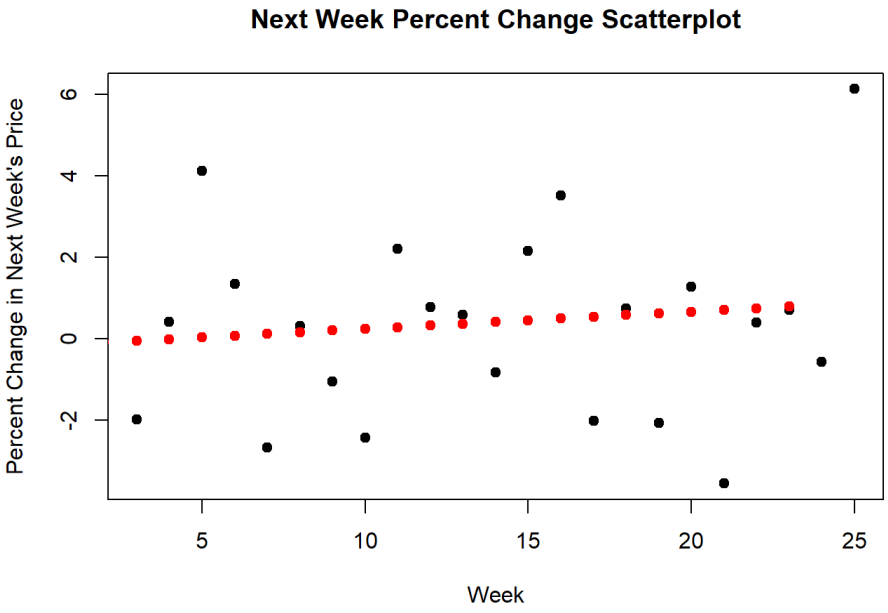

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
5
10
15
20
25
Week

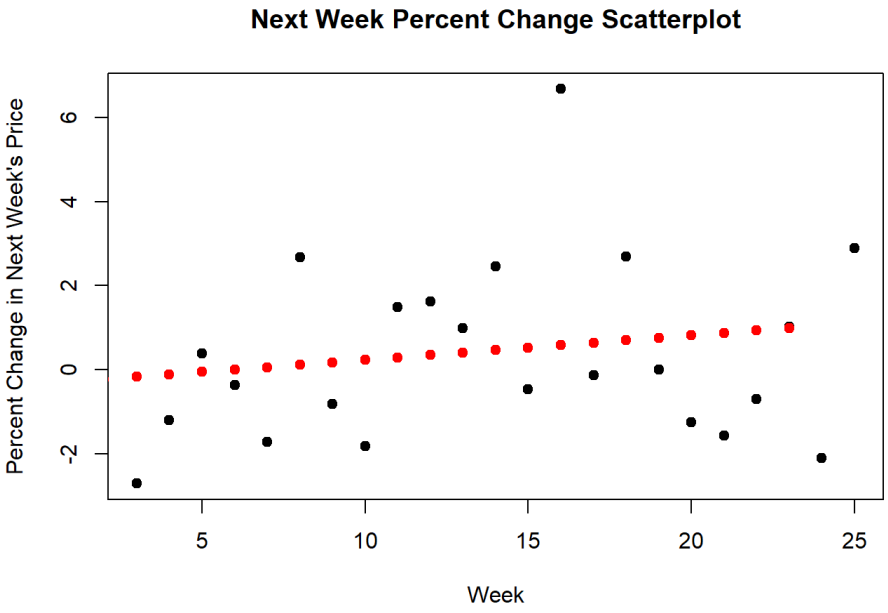

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
5
10
15
20
25
Week

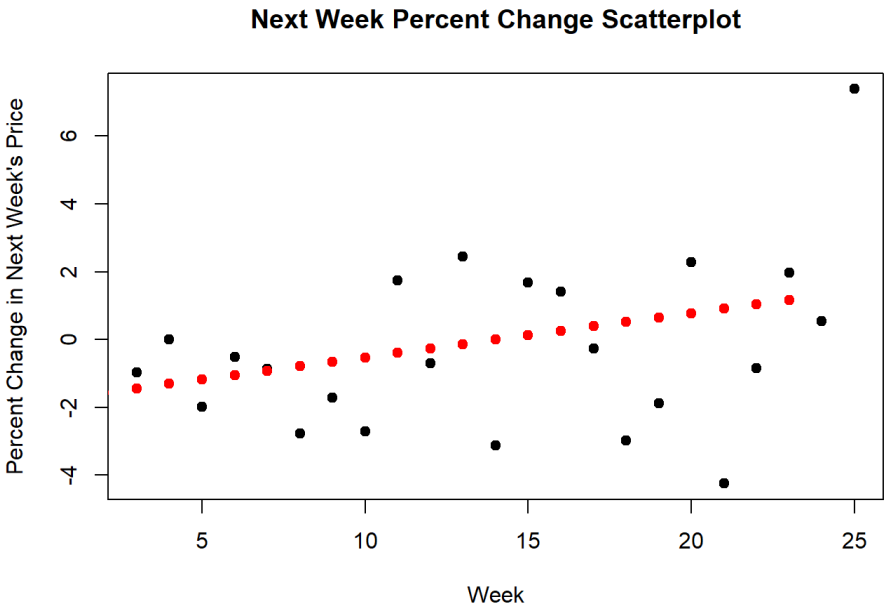

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
5
10
15
20
25
Week

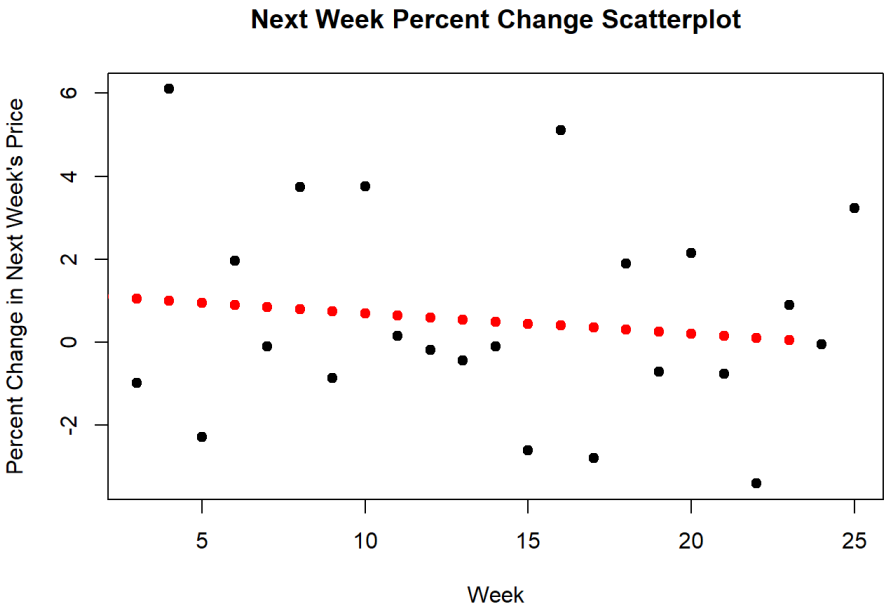

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
5
10
15
20
25
Week

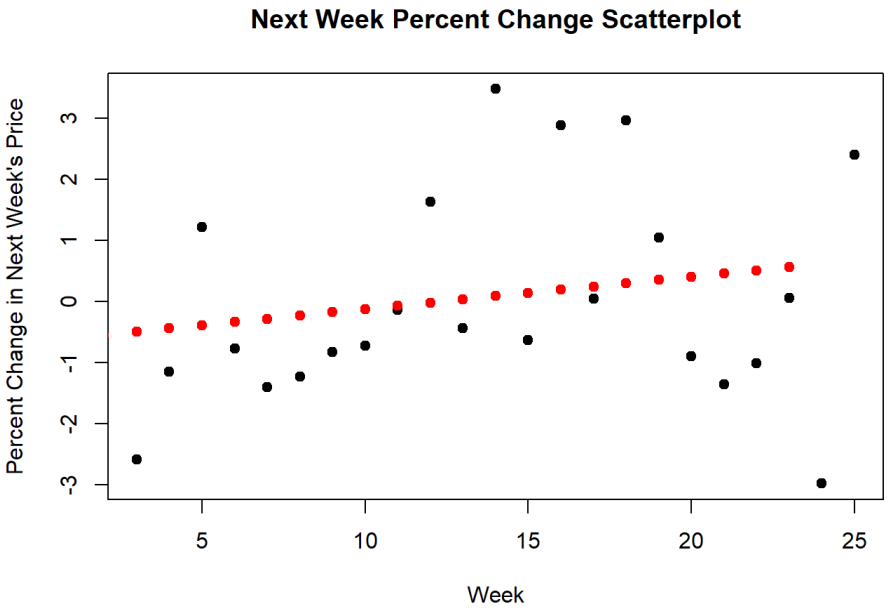

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
5
10
15
20
25
Week

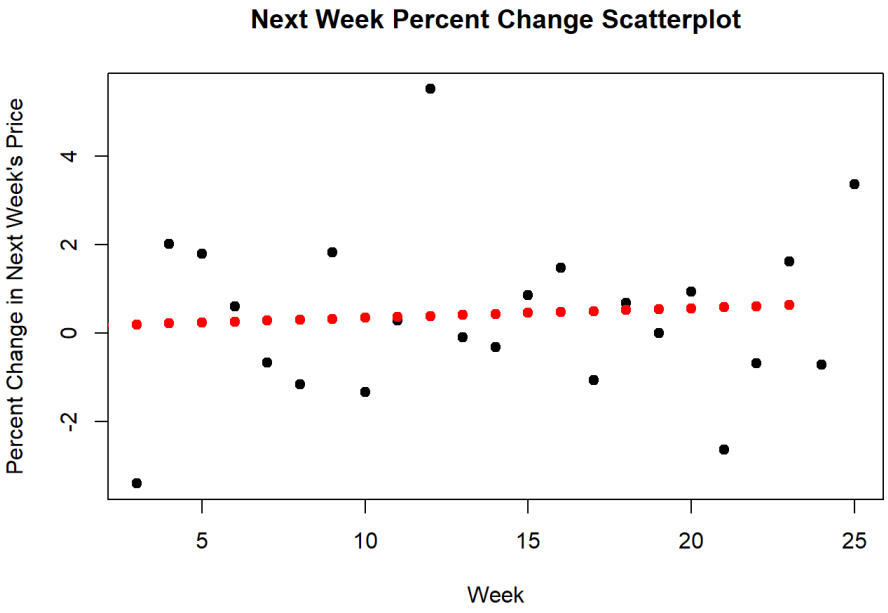

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
6
5
10
15
20
25
Week

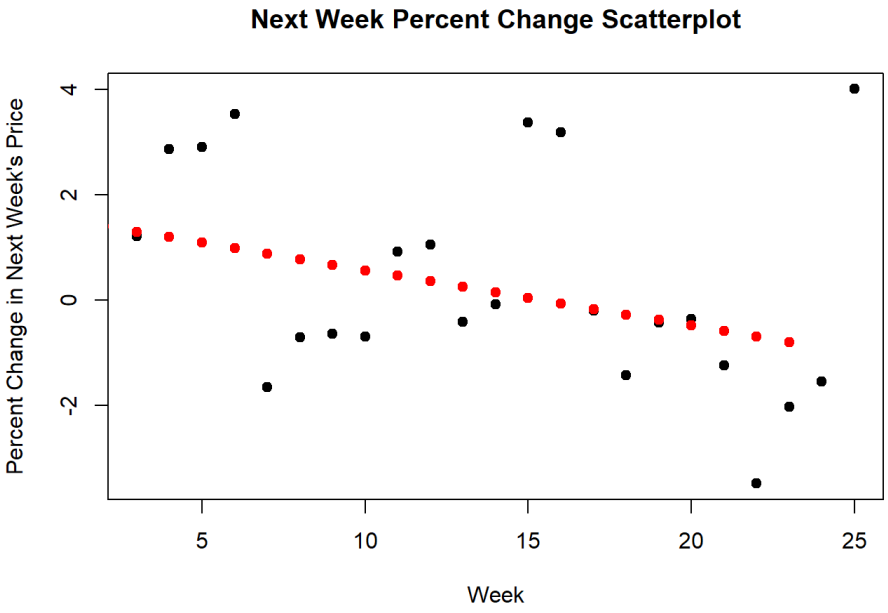

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
6
5
10
15
20
25
Week

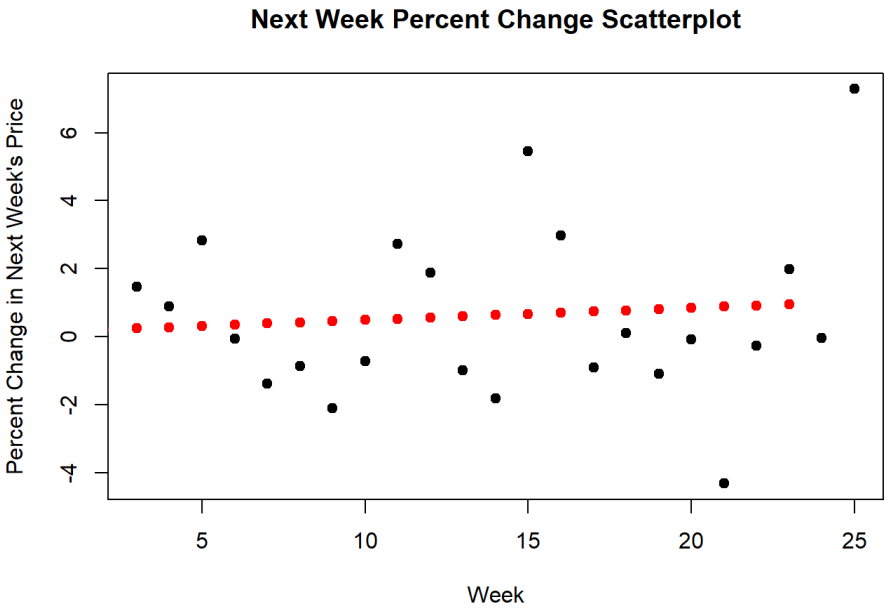

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-2
0
2
4
6
5
10
15
20
25
Week

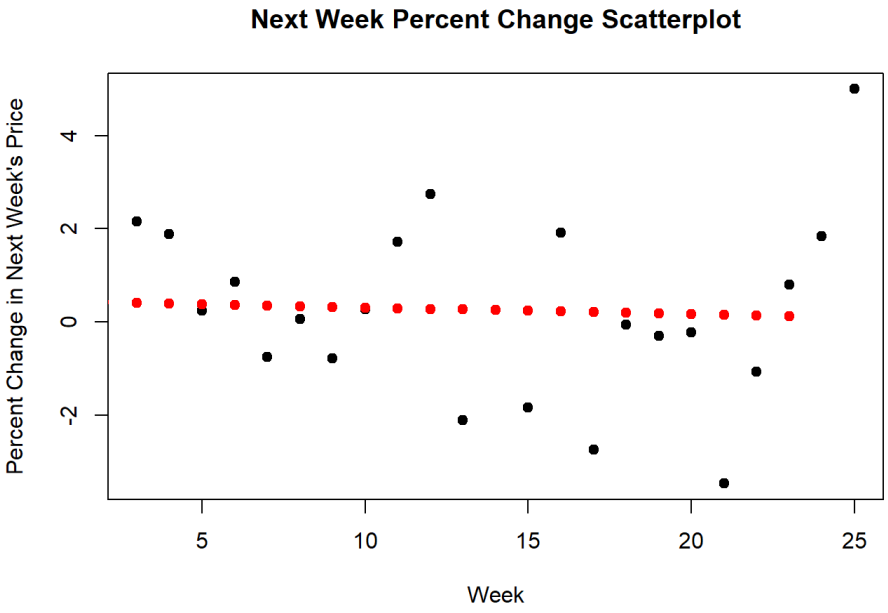

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
6
5
10
15
20
25
Week

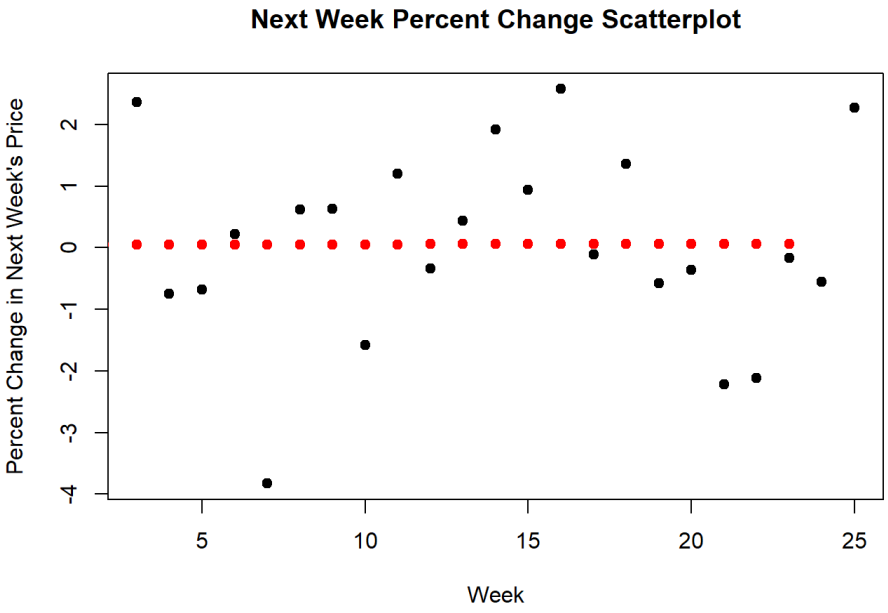

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-3
-2
-1
0
1
2
3
5
10
15
20
25
Week

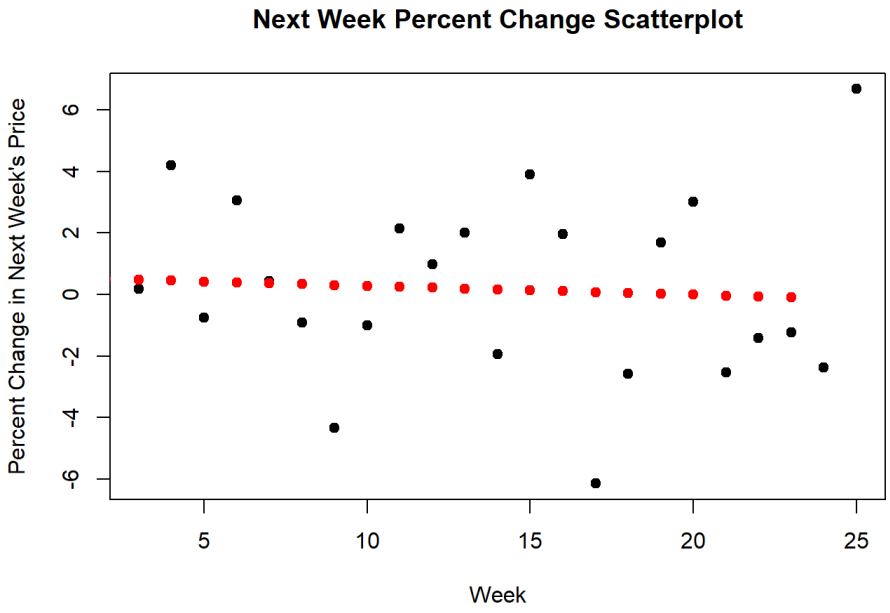

Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
5
10
15
20
25
Week


Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
4
2
0
-2
5
10
15
20
25
Week


Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
6
4
2
0
-2
-4
5
10
15
20
25
Week


Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-2
0
2
4
5
10
15
20
25
Week


Next Week Percent Change Scatterplot
Percent Change in Next Week's Price
-4
-3
-2
-1
0
1
2
5
10
15
20
25
Week



Determing the formula for regression

```
DJI2 <- na.omit(DJI1) #DJI2 omits NAs in DJI1 in order for "cor" function to generate correlation values for lag vari
ables.
num_cols <- unlist((lapply(DJI2, is.numeric)))
CorValues <- data.frame(cor(DJI2[, na.omit(num_cols)]))
PerChangeNextCorVals <- CorValues[10]
PerChangeNextCorVals <- PerChangeNextCorVals %>% rownames_to_column('Variable') #preserves variable names for ease of
access
PerChangeNextCorVals <- PerChangeNextCorVals[-c(10),] #removing the correlation between Next Weeks Price and itself
```

```
AveragePerCorVal <- mean(PerChangeNextCorVals$percent_change_next_weeks_price)
AveragePerCorVal
```

```
## [1] 0.0102227
```

```
FormulaCorVals <- (PerChangeNextCorVals[PerChangeNextCorVals$percent_change_next_weeks_price > AveragePerCorVal,])

View(FormulaCorVals)
```

```
(FormulaCorVals$Variable)
```

```
##  [1] "Week"
##  [2] "open"
##  [3] "high"
##  [4] "low"
##  [5] "close"
##  [6] "percent_change_price"
##  [7] "percent_change_volume_over_last_wk"
##  [8] "open.lag"
##  [9] "high.lag"
## [10] "low.lag"
## [11] "close.lag"
## [12] "percent_change_volume_over_last_wk.lag"
## [13] "percent_change_next_weeks_price.lag"
```

```
LM1 = lm(percent_change_next_weeks_price ~ Week + open + high + low + close + percent_change_price + percent_change_v
olume_over_last_wk + open.lag + close.lag + high.lag + low.lag + percent_change_volume_over_last_wk.lag + percent_cha
nge_next_weeks_price.lag, data = DJI1)
#car::vif(LM1) #<- cannot run VIF due to aliased coefficients
```

```
ld.vars <- attributes(alias(LM1)$Complete)$dimnames[[1]]
print(ld.vars) # Outputs aliased coefficients
```

```
## [1] "percent_change_next_weeks_price.lag"
```

```
LM2 = lm(percent_change_next_weeks_price ~ Week + open + high + low + close + percent_change_price + percent_change_v
olume_over_last_wk + open.lag + close.lag + high.lag + low.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM2)
```

```
##                                Week                                open
##                            1.081757                            8542.095172
##                                high                                 low
##                         3310.066341                            1739.034896
##                               close                   percent_change_price
##                         3440.839504                               3.416006
##     percent_change_volume_over_last_wk                            open.lag
##                            1.361819                            1810.721193
##                           close.lag                            high.lag
##                         8631.138061                            3468.003613
##                             low.lag percent_change_volume_over_last_wk.lag
##                         1783.023338                               1.294589
```

```
LM3 = lm(percent_change_next_weeks_price ~ Week + open + high + low + close + percent_change_price + percent_change_v
olume_over_last_wk + open.lag + high.lag + low.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM3)
```

```
##                                Week                                open
##                            1.081031                            4324.020832
##                                high                                 low
##                         3298.187054                            1737.489581
##                               close                   percent_change_price
##                         3395.537697                               3.410060
##     percent_change_volume_over_last_wk                            open.lag
##                            1.360782                            1726.960329
##                            high.lag                             low.lag
##                         3085.562121                            1542.122772
## percent_change_volume_over_last_wk.lag
##                            1.265054
```

```
LM4 = lm(percent_change_next_weeks_price ~ Week + high + low + close + percent_change_price + percent_change_volume_o
ver_last_wk + open.lag + high.lag + low.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM4)
```

```
##                                Week                                high
##                            1.070661                            2594.969048
##                                 low                               close
##                         1631.094192                            3352.238153
##                   percent_change_price     percent_change_volume_over_last_wk
##                            2.478717                               1.342796
##                            open.lag                            high.lag
##                         1671.060518                            2822.547059
##                             low.lag percent_change_volume_over_last_wk.lag
##                         1354.670644                               1.255092
```

```
LM5 = lm(percent_change_next_weeks_price ~ Week + high + low + percent_change_price + percent_change_volume_over_last
_wk + open.lag + high.lag + low.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM5)
```

```
##                                Week                                high
##                            1.054540                            1532.372739
##                                 low                   percent_change_price
##                          1124.297274                               1.437698
##     percent_change_volume_over_last_wk                            open.lag
##                            1.331592                            1669.386107
##                            high.lag                                low.lag
##                          2804.802079                            1354.615198
## percent_change_volume_over_last_wk.lag
##                            1.244611
```

```
LM6 = lm(percent_change_next_weeks_price ~ Week + high + low + percent_change_price + percent_change_volume_over_last
_wk + open.lag + low.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM6)
```

```
##                                Week                                high
##                            1.054486                            1205.308005
##                                 low                   percent_change_price
##                          1123.119048                               1.355684
##     percent_change_volume_over_last_wk                            open.lag
##                            1.311802                             839.138988
##                             low.lag percent_change_volume_over_last_wk.lag
##                          1329.527195                               1.185363
```

```
LM7 = lm(percent_change_next_weeks_price ~ Week + high + low + percent_change_price + percent_change_volume_over_last
_wk + open.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM7)
```

```
##                                Week                                high
##                            1.053641                            1183.874741
##                                 low                   percent_change_price
##                           994.231141                               1.237311
##     percent_change_volume_over_last_wk                            open.lag
##                            1.310973                             418.565080
## percent_change_volume_over_last_wk.lag
##                            1.142156
```

```
LM8 = lm(percent_change_next_weeks_price ~ Week + low + percent_change_price + percent_change_volume_over_last_wk + o
pen.lag + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM8)
```

```
##                                Week                                 low
##                            1.052410                            341.424857
##                   percent_change_price     percent_change_volume_over_last_wk
##                            1.227056                               1.177392
##                            open.lag percent_change_volume_over_last_wk.lag
##                          340.220432                               1.138327
```

```
LM9 = lm(percent_change_next_weeks_price ~ Week + percent_change_price + percent_change_volume_over_last_wk + open.la
g + percent_change_volume_over_last_wk.lag, data = DJI1)
car::vif(LM9)
```

```
##                                  Week                  percent_change_price
##                              1.026779                              1.080845
##     percent_change_volume_over_last_wk                              open.lag
##                              1.171368                              1.007114
## percent_change_volume_over_last_wk.lag
##                              1.116324
```

```
formulafinal = percent_change_next_weeks_price ~ Week + percent_change_price + percent_change_volume_over_last_wk + o
pen.lag + percent_change_volume_over_last_wk.lag
```

Creating the Linear Model and MSE

```
StockFunctionL <- function (training, testing, formula)  {
  set.seed(111)

  LinearModel <- lm(formula, data = training)

  LinearPreds <- predict(LinearModel, newdata = testing)

  LinearSumm <- (summary(LinearModel))

  MSE.Linear <- (mean((testing$percent_change_next_weeks_price - LinearPreds)^2))

  print(MSE.Linear)
}
```

```
MSE.L <- c()

for (i in stocknames)
{x = filter((DJI1Training), stock == i)

  y = filter(DJI1Test, stock == i)

  MSE.L <- c(MSE.L,StockFunctionL(x, y, formulafinal))



}
```

```
## [1] 37.63047
## [1] 166.8408
## [1] 9.549699
## [1] 106.5506
## [1] 210.1657
## [1] 861.7221
## [1] 35.12052
## [1] 60.19373
## [1] 18.96346
## [1] 14.84058
## [1] 13.17321
## [1] 15.61936
## [1] 5.881939
## [1] 59.49513
## [1] 8.979624
## [1] 39.80836
## [1] 23.45129
## [1] 17.31594
## [1] 14.64322
## [1] 9.020035
## [1] 19.51418
## [1] 16.03365
## [1] 7.304041
## [1] 13.80023
## [1] 36.05449
## [1] 8.480527
## [1] 9.643165
## [1] 25.2066
## [1] 36.04095
## [1] 21.46854
```

```
view(MSE.L)

df <- data.frame(stocknames, MSE.L)
```

Creating the Tree Model and MSE

```
StockFunctionTree <- function(training, testing, formula) {
  set.seed(111)

  TreeModel <- tree(formula, data = training)

  TreePreds <- predict(TreeModel, newdata = testing)

  MSE.Tree <- (mean((testing$percent_change_next_weeks_price - TreePreds)^2))

  print(MSE.Tree)
}
```

```
MSE.T <- c()

for (i in stocknames)
{x = filter(DJI1Training, stock == i)

  y = filter(DJI1Test, stock == i)

  MSE.T <- c(MSE.T,StockFunctionTree(x, y, formulafinal))

}
```

```
## [1] 16.30036
## [1] 9.525216
## [1] 7.947067
## [1] 7.931907
## [1] 23.24086
## [1] 18.72477
## [1] 13.65708
## [1] 9.411393
## [1] 16.5827
## [1] 11.69934
## [1] 4.867015
## [1] 14.46336
## [1] 4.288812
## [1] 26.73175
## [1] 7.932704
## [1] 7.394828
## [1] 3.346866
## [1] 4.738038
## [1] 2.879354
## [1] 5.718057
## [1] 8.663407
## [1] 11.75154
## [1] 8.189972
## [1] 4.33191
## [1] 3.804891
## [1] 4.649902
## [1] 10.32846
## [1] 4.523278
## [1] 1.273753
## [1] 12.10326
```

```
view(MSE.T)
```

Creating the SVR Model and MSE

```
StockFunctionSVR <- function(training, testing, formula) {
  set.seed(111)

  svmtune <- tune.svm(formula, data = training, gamma =seq(.01, 0.1, by = .01), cost = seq(0.1,1, by = 0.1))

  SVRModel <- svm(formula, data = training, kernel = "radial", cost = svmtune$best.parameters$cost, gamma = svmtune$b
est.parameters$gamma)

  SVRPreds <- predict(SVRModel, newdata = testing)

  MSE.SVR <- (mean((testing$percent_change_next_weeks_price - SVRPreds)^2))

  print(MSE.SVR)
}
```

```
MSE.SVR <- c()

for (i in stocknames)
{x = filter(DJI1Training, stock == i)

  y = filter(DJI1Test, stock == i)

  MSE.SVR <- c(MSE.SVR,StockFunctionSVR(x, y, formulafinal))

}
```

```
## [1] 19.65936
## [1] 12.03012
## [1] 8.131715
## [1] 8.106007
## [1] 24.51668
## [1] 11.80605
## [1] 16.14803
## [1] 14.86331
## [1] 9.173087
## [1] 9.070629
## [1] 5.667518
## [1] 13.90082
## [1] 4.447285
## [1] 19.53144
## [1] 5.56073
## [1] 7.861922
## [1] 4.351632
## [1] 3.474229
## [1] 2.885961
## [1] 5.912115
## [1] 5.919042
## [1] 10.29288
## [1] 5.367814
## [1] 4.198298
## [1] 2.08079
## [1] 4.738434
## [1] 9.20122
## [1] 5.534084
## [1] 2.321339
## [1] 11.04375
```

```
view(MSE.SVR)
```

Calculating MSE Averages

```
MSEDataFrame <- (cbind(df, MSE.T, MSE.SVR))
MSEDataFrame <- data.frame(MSEDataFrame)
```

```
MSE.LAvg <- mean(MSEDataFrame$MSE.L)
MSE.TAvg <- mean(MSEDataFrame$MSE.T)
MSE.SVRAvg <- mean(MSEDataFrame$MSE.SVR)
```

```
MSE.LAvg
```

```
## [1] 64.08374
```

```
MSE.TAvg
```

```
## [1] 9.566728
```

```
MSE.SVRAvg
```

```
## [1] 8.926543
```

SVR Model with all variables, for comparison

```
SVRALL <- c()

for (i in stocknames)
{x = filter(DJI1Training, stock == i)

  y = filter(DJI1Test, stock == i)

  SVRALL <- c(SVRALL,StockFunctionSVR(x, y, percent_change_next_weeks_price ~ Week + open + high + low + close + perc
ent_change_price + percent_change_volume_over_last_wk + open.lag + close.lag + high.lag + low.lag + percent_change_vo
lume_over_last_wk.lag))

}
```

```
## [1] 17.85271
## [1] 16.51706
## [1] 8.513247
## [1] 7.990682
## [1] 24.43615
## [1] 11.50231
## [1] 15.31846
## [1] 8.769717
## [1] 9.54957
## [1] 9.134985
## [1] 5.68309
## [1] 13.92852
## [1] 3.88957
## [1] 19.07377
## [1] 6.040683
## [1] 7.648957
## [1] 4.07934
## [1] 3.561194
## [1] 3.394678
## [1] 6.514753
## [1] 8.517038
## [1] 10.33157
## [1] 5.655647
## [1] 4.791393
## [1] 2.086325
## [1] 5.402838
## [1] 9.46905
## [1] 7.0819
## [1] 2.283613
## [1] 10.96047
```

```
view(SVRALL)
```

```
SVRComparison <- cbind(SVRALL, MSE.SVR)
mean(SVRALL)
```

```
## [1] 8.999309
```

Creating the CAPM Model and Beta Values

```
DJICAPM1 <-split(DJI1Test, DJI1Test$stock)
SP500Data <-read.csv("S&P_500.csv", header = TRUE, sep = ",")
SP500Data1 = na.omit(SP500Data)
SP500Data1$Date <- as.Date(SP500Data1$Date, "%m/%d/%Y")
```

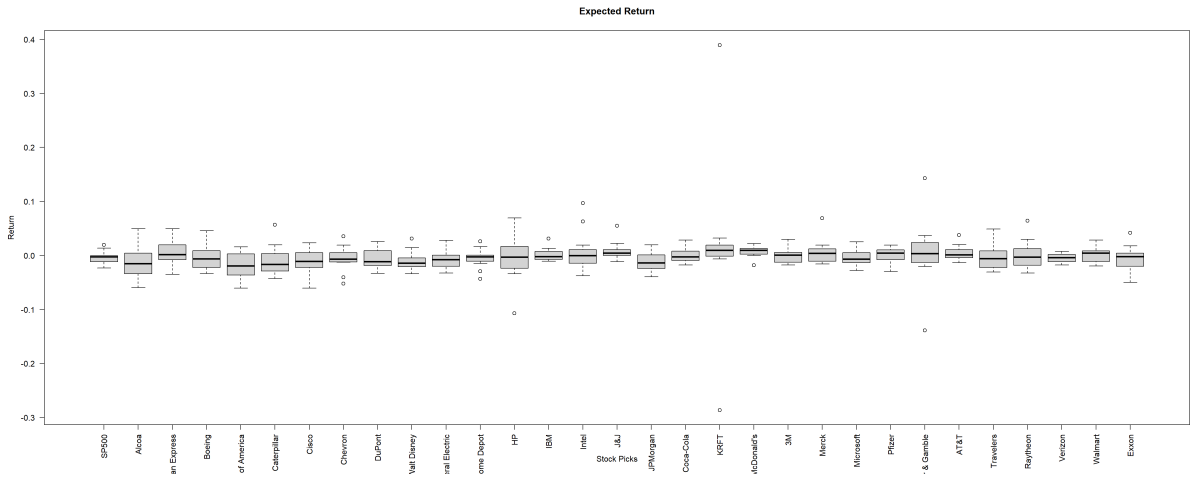
```
ReturnSP500 = na.omit(Delt(SP500Data1[,5]))
ReturnAA = na.omit(Delt(DJICAPM1$AA[,7]))
ReturnAXP = na.omit(Delt(DJICAPM1$AXP[,7]))
ReturnBA = na.omit(Delt(DJICAPM1$BA[,7]))
ReturnBAC = na.omit(Delt(DJICAPM1$BAC[,7]))
ReturnCAT = na.omit(Delt(DJICAPM1$CAT[,7]))
ReturnCSCO = na.omit(Delt(DJICAPM1$CSCO[,7]))
ReturnCVX = na.omit(Delt(DJICAPM1$CVX[,7]))
ReturnDD = na.omit(Delt(DJICAPM1$DD[,7]))
ReturnDIS = na.omit(Delt(DJICAPM1$DIS[,7]))
ReturnGE = na.omit(Delt(DJICAPM1$GE[,7]))
ReturnHD = na.omit(Delt(DJICAPM1$HD[,7]))
ReturnHPQ = na.omit(Delt(DJICAPM1$HPQ[,7]))
ReturnIBM = na.omit(Delt(DJICAPM1$IBM[,7]))
ReturnINTC = na.omit(Delt(DJICAPM1$INTC[,7]))
ReturnJNJ = na.omit(Delt(DJICAPM1$JNJ[,7]))
ReturnJPM = na.omit(Delt(DJICAPM1$JPM[,7]))
ReturnKO = na.omit(Delt(DJICAPM1$KO[,7]))
ReturnKRFT = na.omit(Delt(DJICAPM1$KRFT[,7]))
ReturnMCD = na.omit(Delt(DJICAPM1$MCD[,7]))
ReturnMMM = na.omit(Delt(DJICAPM1$MMM[,7]))
ReturnMRK = na.omit(Delt(DJICAPM1$MRK[,7]))
ReturnMSFT = na.omit(Delt(DJICAPM1$MSFT[,7]))
ReturnPFE = na.omit(Delt(DJICAPM1$PFE[,7]))
ReturnPG = na.omit(Delt(DJICAPM1$PG[,7]))
ReturnT = na.omit(Delt(DJICAPM1$T[,7]))
ReturnTRV = na.omit(Delt(DJICAPM1$TRV[,7]))
ReturnUTX = na.omit(Delt(DJICAPM1$UTX[,7]))
ReturnVZ = na.omit(Delt(DJICAPM1$VZ[,7]))
ReturnWMT = na.omit(Delt(DJICAPM1$WMT[,7]))
ReturnXOM = na.omit(Delt(DJICAPM1$XOM[,7]))
```

```
DataCAPM = cbind(ReturnSP500, ReturnAA, ReturnAXP, ReturnBA, ReturnBAC, ReturnCAT, ReturnCSCO, ReturnCVX, ReturnDD,
               ReturnDIS, ReturnGE, ReturnHD, ReturnHPQ, ReturnIBM, ReturnINTC, ReturnJNJ, ReturnJPM, ReturnKO,
               ReturnKRFT, ReturnMCD, ReturnMMM, ReturnMRK, ReturnMSFT, ReturnPFE, ReturnPG, ReturnT, ReturnTRV,
               ReturnUTX, ReturnVZ, ReturnWMT, ReturnXOM)
```

```
colnames(DataCAPM) = c("SP500", "Alcoa", "American Express", "Boeing","Bank of America","Caterpillar", "Cisco",
                     "Chevron","DuPont","Walt Disney", "General Electric", "Home Depot", "HP", "IBM", "Intel",
                     "J&J", "JPMorgan", "Coca-Cola", "KRFT", "McDonald's", "3M", "Merck", "Microsoft", "Pfizer",
                     "Procter & Gamble", "AT&T", "Travelers", "Raytheon", "Verizon", "Walmart", "Exxon")
head(DataCAPM)
```

```
##             SP500        Alcoa American Express       Boeing Bank of America
## [1,] -0.003182204  0.023501763    -0.0002216803 -0.004381761     0.015957447
## [2,] -0.006392254 -0.059701493     0.0110864745 -0.025168478    -0.041136874
## [3,]  0.013412342 -0.030525031    -0.0072368421  0.012274266    -0.052262090
## [4,]  0.019612975  0.049118388     0.0331345262  0.040836237     0.006584362
## [5,] -0.017167665  0.010204082     0.0493906350  0.045661489    -0.002452984
## [6,] -0.001813162 -0.004753417     0.0028524857  0.004994237    -0.023770492
##       Caterpillar         Cisco      Chevron       DuPont  Walt Disney
## [1,]  0.002485959  0.0005885815  0.018858984  0.024208566 -0.022662890
## [2,] -0.032421014 -0.0017647059 -0.040740396 -0.033272727 -0.013526570
## [3,] -0.030280019 -0.0265173836 -0.000872685  0.000376152 -0.009549461
## [4,]  0.056675803  0.0230024213  0.035326087  0.025380711  0.031149567
## [5,]  0.004075961  0.0224852071 -0.052212223 -0.017418409  0.014145289
## [6,] -0.021865486 -0.0370370370 -0.006923153 -0.019406606 -0.032151300
##      General Electric    Home Depot           HP           IBM         Intel
## [1,]      0.027649770  0.0161114145  0.069148936  0.0126613704 -0.0102249489
## [2,]     -0.023418037 -0.0010749798 -0.001990050 -0.0052708997  0.0025826446
## [3,]     -0.004591837  0.0008071025 -0.023928215 -0.0006777572  0.0020607934
## [4,]      0.019477191 -0.0069892473  0.021195097  0.0310746655  0.0966580977
## [5,]     -0.004022122 -0.0046020574 -0.004751188  0.0016145428  0.0628223160
## [6,]     -0.002523978 -0.0019037259  0.010552764  0.0019104478 -0.0008822232
##              J&J     JPMorgan    Coca-Cola        KRFT  McDonald's
## [1,] 0.005430171  0.011865524  0.028140858 0.009336768 0.010551623
## [2,] 0.004050633 -0.026492942 -0.006730482 0.010845295 0.002246894
## [3,] 0.010253824 -0.028998439  0.003463334 0.031871253 0.007516814
## [4,] 0.054908486  0.019526763 -0.012454982 0.009174312 0.001832461
## [5,] 0.022082019  0.008562416  0.008965203 0.006666667 0.021688006
## [6,] 0.004166667 -0.039767650  0.006475904 0.017459362 0.011253197
##                3M       Merck    Microsoft       Pfizer Procter & Gamble
## [1,]  0.0076169750 0.019141710  0.004347826  0.001989060     0.0141890777
## [2,] -0.0091792657 0.007876401 -0.012593467  0.008933002     0.0069952823
## [3,] -0.0127520436 0.007213706 -0.014746911 -0.026561731     0.0221324717
## [4,]  0.0296975050 0.003581021  0.025080906  0.005558363    -0.0105895369
## [5,]  0.0121153640 0.068986024  0.005524862  0.008542714     0.0367412141
## [6,]  0.0004237288 0.004172462 -0.020800628  0.018933732    -0.0006163328
##               AT&T    Travelers     Raytheon      Verizon      Walmart
## [1,]  0.0377162630  0.012677745  0.011049724 -0.004275788  0.003276161
## [2,]  0.0006668890 -0.001860937 -0.017581373  0.002147075  0.006530926
## [3,] -0.0033322226 -0.015762712 -0.019347037 -0.011515801  0.009732824
## [4,]  0.0173854898  0.048906492  0.063995068 -0.002438364  0.006048006
## [5,]  0.0197173842  0.031850271 -0.005678526  0.002172732 -0.000563592
## [6,]  0.0006445375 -0.010660302  0.029720280  0.001355014  0.028195489
##              Exxon
## [1,]  0.0176734052
## [2,] -0.0201022957
## [3,]  0.0007283321
## [4,]  0.0418486172
## [5,] -0.0500640354
## [6,] -0.0265963966
```

```
boxplot(DataCAPM,main="Expected Return", xlab="Stock Picks", ylab="Return", las = 2)
```




```
DataMean=apply(DataCAPM, 2, mean)
DataSD=apply(DataCAPM, 2, sd)
```

```
cbind(DataMean,DataSD)
```

```
##                      DataMean      DataSD
## SP500            -0.0040165238 0.012734511
## Alcoa            -0.0124984548 0.030445572
## American Express  0.0056848799 0.022759720
## Boeing           -0.0028762726 0.025917564
## Bank of America  -0.0185116675 0.024669288
## Caterpillar      -0.0098467145 0.027465207
## Cisco            -0.0104974089 0.024098526
## Chevron          -0.0067360208 0.023462501
## DuPont           -0.0066566315 0.019483678
## Walt Disney      -0.0106329613 0.018420645
## General Electric -0.0067725531 0.017520537
## Home Depot       -0.0053326395 0.018284563
## HP               -0.0061917147 0.042012358
## IBM               0.0013352893 0.012128694
## Intel             0.0068050946 0.037731703
## J&J               0.0079359622 0.017341273
## JPMorgan         -0.0117102291 0.017907032
## Coca-Cola        -0.0006947390 0.013026295
## KRFT              0.0165980980 0.145348970
## McDonald's        0.0070410005 0.010251248
## 3M               -0.0010283347 0.013634258
## Merck             0.0055135310 0.023257689
## Microsoft        -0.0037198677 0.015841822
## Pfizer           -0.0000078646 0.015874197
## Procter & Gamble  0.0042924047 0.062597942
## AT&T              0.0040178668 0.014532605
## Travelers        -0.0029976830 0.024514074
## Raytheon          0.0007454031 0.026503509
## Verizon          -0.0050556201 0.007795722
## Walmart           0.0008217758 0.013711913
## Exxon            -0.0058274012 0.023442281
```

```
DataCAPM1 = data.frame(DataCAPM)
```

```
CAPMFunction <- function (stockname)  {
  set.seed(111)

  LinearModel <- lm(stockname ~ SP500, data = DataCAPM1)

  LinearSumm <- (summary(LinearModel))

  BetaVal <- (LinearSumm)$coefficients[2, 1]

  print(BetaVal)
}
```

```
CAPMNames <- colnames(DataCAPM1[,2:31])
CAPMNames
```

```
##  [1] "Alcoa"            "American.Express" "Boeing"           "Bank.of.America"
##  [5] "Caterpillar"      "Cisco"            "Chevron"          "DuPont"
##  [9] "Walt.Disney"      "General.Electric" "Home.Depot"       "HP"
## [13] "IBM"              "Intel"            "J.J"              "JPMorgan"
## [17] "Coca.Cola"        "KRFT"             "McDonald.s"       "X3M"
## [21] "Merck"            "Microsoft"        "Pfizer"           "Procter...Gamble"
## [25] "AT.T"             "Travelers"        "Raytheon"         "Verizon"
## [29] "Walmart"          "Exxon"
```

```
BetaVals <- c()

for(i in CAPMNames)
{
data = DataCAPM1[[i]]
  BetaVals <- c(BetaVals,CAPMFunction((data)))



}
```

```
## [1] 0.6528356
## [1] 0.5340911
## [1] 0.6654864
## [1] 0.4177411
## [1] 0.7797351
## [1] 0.4958597
## [1] 1.054348
## [1] 0.6350098
## [1] 0.6848398
## [1] 0.7047665
## [1] 0.730166
## [1] 0.08174864
## [1] 0.5858814
## [1] 1.219414
## [1] 0.642415
## [1] 0.247314
## [1] -0.007214267
## [1] 4.393856
## [1] 0.1979787
## [1] 0.3144422
## [1] -0.2242097
## [1] 0.2837018
## [1] -0.1068793
## [1] -0.2463233
## [1] 0.3557808
## [1] 0.4186005
## [1] 1.077343
## [1] 0.1233665
## [1] 0.4347628
## [1] 0.924724
```

```
view(BetaVals)
```

```
VariableNames<- c("Alcoa", "American Express", "Boeing","Bank of America","Caterpillar", "Cisco",
                    "Chevron","DuPont","Walt Disney", "General Electric", "Home Depot", "HP", "IBM", "Intel",
                    "J&J", "JPMorgan", "Coca-Cola", "KRFT", "McDonald's", "3M", "Merck", "Microsoft", "Pfizer",
                    "Procter & Gamble", "AT&T", "Travelers", "Raytheon", "Verizon", "Walmart", "Exxon")

BetaVals <- cbind(VariableNames, BetaVals)
BetaVals <- (data.frame(BetaVals))
```

```
BetaVals
```

```
##       VariableNames             BetaVals
## 1             Alcoa    0.652835561410909
## 2  American Express    0.534091085628767
## 3            Boeing    0.665486413652409
## 4   Bank of America    0.417741081143332
## 5       Caterpillar    0.779735136121435
## 6             Cisco    0.495859673970679
## 7           Chevron     1.05434829072847
## 8            DuPont    0.635009846326277
## 9       Walt Disney    0.684839829193602
## 10 General Electric    0.704766470520885
## 11       Home Depot     0.73016602074518
## 12               HP    0.081748644849603
## 13              IBM    0.585881415105085
## 14            Intel     1.21941448523545
## 15              J&J    0.642414985243703
## 16         JPMorgan    0.247314047118315
## 17        Coca-Cola -0.00721426653681537
## 18             KRFT     4.39385602443855
## 19       McDonald's    0.197978650151929
## 20               3M    0.314442236384186
## 21            Merck   -0.224209721906038
## 22        Microsoft    0.283701759473491
## 23           Pfizer   -0.106879283067782
## 24 Procter & Gamble   -0.246323281224192
## 25             AT&T    0.355780775179573
## 26        Travelers    0.418600517699297
## 27         Raytheon     1.07734292816457
## 28          Verizon    0.123366529296826
## 29          Walmart    0.434762848389391
## 30            Exxon    0.924723978318247
```

```
BetaVals$BetaVals <- round(as.numeric(BetaVals$BetaVals),4)
BetaVals
```

```
##       VariableNames BetaVals
## 1             Alcoa   0.6528
## 2  American Express   0.5341
## 3            Boeing   0.6655
## 4   Bank of America   0.4177
## 5       Caterpillar   0.7797
## 6             Cisco   0.4959
## 7           Chevron   1.0543
## 8            DuPont   0.6350
## 9       Walt Disney   0.6848
## 10 General Electric   0.7048
## 11       Home Depot   0.7302
## 12               HP   0.0817
## 13              IBM   0.5859
## 14            Intel   1.2194
## 15              J&J   0.6424
## 16         JPMorgan   0.2473
## 17        Coca-Cola  -0.0072
## 18             KRFT   4.3939
## 19       McDonald's   0.1980
## 20               3M   0.3144
## 21            Merck  -0.2242
## 22        Microsoft   0.2837
## 23           Pfizer  -0.1069
## 24 Procter & Gamble  -0.2463
## 25             AT&T   0.3558
## 26        Travelers   0.4186
## 27         Raytheon   1.0773
## 28          Verizon   0.1234
## 29          Walmart   0.4348
## 30            Exxon   0.9247
```

Predicting using the SVR Model

```
SVRPredsValues <- function(training, testing, formula) {
  set.seed(111)

  svmtune <- tune.svm(formula, data = training, gamma =seq(.01, 0.1, by = .01), cost = seq(0.1,1, by = 0.1))

  SVRModel <- svm(formula, data = training, kernel = "radial", cost = svmtune$best.parameters$cost, gamma = svmtune$b
est.parameters$gamma)

  SVRPreds <- predict(SVRModel, newdata = testing)

  print(SVRPreds)
}
```

```
SVRNextWeek <- c()

for (i in stocknames)
{
x = filter(DJI1Training, stock == i)

y = filter(DJI1Test, stock == i & Week == 25)

SVRNextWeek <- c(SVRNextWeek,SVRPredsValues(x, y, formulafinal))
}
```

```
##         1
## 0.6720748
##         1
## -1.186704
##         1
## 0.2800098
##          1
## -0.5080646
##        1
## 2.346157
##         1
## -1.536682
##        1
## 1.874089
##        1
## 3.089389
##         1
## 0.4114028
##        1
## 1.005186
##         1
## 0.1720837
##          1
## -0.4980126
##        1
## 2.126975
##         1
## -1.081152
##         1
## 0.2248138
##         1
## 0.2164145
##        1
## 1.330948
##         1
## 0.6944458
##        1
## 1.256062
##         1
## 0.4053232
##          1
## -0.3907161
##          1
## -0.8900265
##          1
## -0.1468189
##         1
## -0.550282
##        1
## 0.483964
##         1
## 0.5525911
##         1
## 0.2955521
##         1
## 0.8612156
##         1
## 0.1890913
##         1
## 0.3216285
```

```
view(SVRNextWeek)
```

Combining Values

```
SVRNextWeek1 <- data.frame(SVRNextWeek)
SVRNextWeek1 <- cbind(VariableNames,SVRNextWeek1)
```

```
ComparisonDF <- cbind(SVRNextWeek1, BetaVals$BetaVals)
ComparisonDF <- ComparisonDF %>%
  rename("Percent Change Prediction for Next Week"  = SVRNextWeek,
         "Beta Value" = "BetaVals$BetaVals")
```

```
head(ComparisonDF,30)
```

```
##       VariableNames Percent Change Prediction for Next Week Beta Value
## 1             Alcoa                               0.6720748     0.6528
## 2  American Express                              -1.1867045     0.5341
## 3            Boeing                               0.2800098     0.6655
## 4   Bank of America                              -0.5080646     0.4177
## 5       Caterpillar                               2.3461575     0.7797
## 6             Cisco                              -1.5366819     0.4959
## 7           Chevron                               1.8740887     1.0543
## 8            DuPont                               3.0893889     0.6350
## 9       Walt Disney                               0.4114028     0.6848
## 10 General Electric                               1.0051864     0.7048
## 11       Home Depot                               0.1720837     0.7302
## 12               HP                              -0.4980126     0.0817
## 13              IBM                               2.1269750     0.5859
## 14            Intel                              -1.0811523     1.2194
## 15              J&J                               0.2248138     0.6424
## 16         JPMorgan                               0.2164145     0.2473
## 17        Coca-Cola                               1.3309479    -0.0072
## 18             KRFT                               0.6944458     4.3939
## 19       McDonald's                               1.2560624     0.1980
## 20               3M                               0.4053232     0.3144
## 21            Merck                              -0.3907161    -0.2242
## 22        Microsoft                              -0.8900265     0.2837
## 23           Pfizer                              -0.1468189    -0.1069
## 24 Procter & Gamble                              -0.5502820    -0.2463
## 25             AT&T                               0.4839640     0.3558
## 26        Travelers                               0.5525911     0.4186
## 27         Raytheon                               0.2955521     1.0773
## 28          Verizon                               0.8612156     0.1234
## 29          Walmart                               0.1890913     0.4348
## 30            Exxon                               0.3216285     0.9247
```