Casestudy 3 – Dow Jones Index

Vishwa Koppisetti

February 12, 2019

I. Executive Summary

In predicting stock prices, we collect data over a period of time. But we cannot take advantage of data from a time period until the next increment of the time period. For example, we can use the data from Monday to invest on Tuesday although not on Monday itself.

Each record in the dataset provided is data for a week. Each record also has the percentage of return that stock has in the following week.
Objectives:

- Build different models to predict stock prices
- Identify stocks with greatest return in the following week
- Evaluate risks of stocks

The dataset has information for 2 quarters pertaining to 30 stocks with 16 variables
Summary of Data: Appendix C

The variables open, high, close, low, next_weeks_open and next_weeks_close are changed from factor to numeric and their dollar symbols where removed.
Quarter variable is changed from integer to factor and date variable is formatted accordingly. Appendix A
Created lag plots between each variable and its previous 3 lags to see if the variables have any correlation with their previous lags.
Lags of open, close, high, low, next_weeks_open, next_weeks_close and percent_return_next_dividend had correlations with their previous 2 or 3 lags but strongly correlated with their previous lag 1. Appendix B
Hence created lags for the variables open, close, high, low and percent_return_next_dividend to be used in model but not for next_weeks_open, next_weeks_close because in real scenario we wouldn't know the future information which these two variables represent.
We initially split the data by quarter and then by stock.
Train: list of 30 stocks for quarter 1 (Jan 2011-Mar 2011)
Test: list of same 30 stocks for quarter 2 (Apr 2011-Jun 2011) Appendix C

- Linear Regression: We run linear Regression to model percent_change_price with lags for the variables open, close, high, low and percent_return_next_dividend as predictors for 30 stocks each. We do not use

the actual variables since they would be highly correlated as we should not have multicollinearity in which case the information a variable provides about the response is redundant in the presence of the other variables

This model helps in easy interpretation of how each variable effects the response however we get a very high mean squared error (approx. 350 by averaging all the respective mean squared errors of each stock) hence conclude that linear regression may not be preferable in predicting the percent_change_price accurately. One possible reason could be that linear model does not successfully capture the complete variability in response resulting in high error rates. We look at other alternatives. Appendix D

Decision Trees:

We get a mean square error of 10 from averaging mean squared errors of all 30 stocks. This is much better than that of linear regression and can be used to make better prediction than that of linear regression, provides easy interpretation but we get only 1 split due to small sample size of each stock. Appendix E

Support Vector Regression:

SVM has SVR for modelling continuous response that we use in our analysis. The model has an average mean squared error of 0.9 which is much better when compared to linear model and considerably better than decision tree. This can be used to predict the percentage change in price of stocks for the following weeks relatively accurately. Appendix F

SVR model is the best model and uses the last week of the test data set to predict the following week percentage change in price. According to the predictions, out of the 30 stocks, VZ, IBM, HD, JPM, MMM are the top 5 stocks with maximum percent change price. Appendix G

For evaluating risk of stocks, we calculated percentage change of the price of stocks during end of the week and ran a linear regression against that of SP500 data for same time period. The coefficient estimates/Beta of VZ, IBM, HD, JPM, MMM were 0.6, 0.64, 0.495, 0.721, 1.03 respectively.

If Beta is 1 then it replicates the market, is as risky as the market (SP500).

Since the Beta of all these stocks except MMM was below 1 it means they are less volatile hence less risky. But MMM on the other hand being greater than 1 is volatile and risky.

II. The Problem

A. Introduction/Background

Forecasting the price movements of stocks is a difficult task. The possible financial reward of picking the correct direction that a stock will move has created much interest in developing systems to predict such behavior. Early work on formal financial forecasting began in the early 1900's and continues to this day. In the last 20 years a variety of techniques have been used to predict stock movement. These include Genetic Algorithms, Neural Networks and other artificial intelligence techniques.

A variety of methods use GAs and Genetic Programming to predict stock and security movements. These methods take different approaches. Some research uses GAs and GPs to develop classification rules, while others use GAs in hybrid approaches. Much research has been done using these evolutionary approaches to perform black-box investing. (Brown, M. S., Pelosi, M. & Dirska, H. (2013). n.d.)

Each record in the dataset provided is data for a week. Each record also has the percentage of return that stock has in the following week.
Objectives:

- Build different models to predict stock prices
- Identify stocks with greatest return in the following week
- Evaluate risks of stocks

B. Purpose of study/importance of study/statement of problem

- Build different models to predict stock prices
- Identify stocks with greatest return in the following week
- Evaluate risks of stocks

C. Questions to be answered/conceptual statement of hypotheses

- Which of the three; linear regression, decision tree, and support vector machines is best suited for predicting percent change price
- What are the advantages of one model over another; Which model provides better prediction accuracy, appropriateness of models
- Evaluate risks of stocks

D. Outline of remainder of report (brief)

- Procedure followed to model the response variable on train set using for the different models.
- Assessing the performance of the model using test set
- Predicting the percentage change in price of stocks for the following weeks with the best model and finding out the risks of top 5 stocks with maximum return

III. Review of Related Literature

A. Acquaint reader with existing methodologies used in this area.

Over the last few decades, hundreds of papers have been written on GAs. But only a small number of these studies use GAs to classify and forecast stock market data.

While sparse, research that uses genetic algorithms for this purpose validates its effectiveness and value for future research. Medical research also supports using classifiers based on GAs.
This research uses a Niche Genetic Algorithm (NGA) called Dynamic-radius Species-conserving Genetic Algorithm (DSGA) to select stocks to purchase from the Dow Jones Index. DSGA uses a set of training data to produce a set of rules. These rules are then used to predict stock prices.
DSGA is an NGA that uses a clustering algorithm enhanced by a tabu list and radial variations. DSGA also uses a shared fitness algorithm to investigate different areas of the domain. This research applies the DSGA algorithm to training data which produces a set of rules. The rules are applied to a set of testing data to obtain results. The DSGA algorithm did very well in predicting stock movement. (Brown, M. S., Pelosi, M. & Dirska, H. (2013). n.d.)

IV. Methodology

A. Identification, classification and operationalization of variables.

- The dataset has information for 2 quarters pertaining to 30 stocks with 16 variables. Summary of Data: Appendix C
- We initially split the data by quarter and then by stock.
- Train: list of 30 stocks for quarter 1 (Jan 2011-Mar 2011)
- Test: list of same 30 stocks for quarter 2 (Apr 2011-Jun 2011)

B. Statements of hypotheses being tested and/or models being developed.

- To find a highly accurate model among Linear Model, Decision Tree, and support vector machine to better predict stock prices
- Identify stocks with greatest return in the following week
- Evaluate risks of stocks

C. Sampling techniques, if full data is not being used. Appendix C

- Train: list of 30 stocks for quarter 1 (Jan 2011-Mar 2011)
- Test: list of same 30 stocks for quarter 2 (Apr 2011-Jun 2011)

D. Data collection process, including data sources, data size, etc. Primary/secondary?

The dataset is secondary source of data from (Brown, M. S., Pelosi, M. & Dirska, H. (2013). n.d.), partitioned into two sets (training/testing) based on financial quarter, thus the 360 examples of the first quarter were utilized as training set and the rest 390 of the second quarter were utilized as testing set.

E. Modeling analysis/techniques used

Different modelling techniques such as Linear Regression, Decision Trees, and support vector machines are used in an attempt to test for prediction accuracy, appropriateness of each models. Risks of different stocks determined through CAPM. Appendix D - Appendix G

F. Methodological assumptions and limitations.

The linear model may not be preferable in predicting the percent_change_price accurately. One possible reason could be that linear model does not successfully capture the complete variability in response (that is sensitive to time, expected to have a highly irregular pattern) resulting in high error rates. We should try other alternatives though the interpretation is better with this model.

The decision Tree is a non-parametric method which does not require any assumptions and is easy to interpret but in our case we have only 2 terminal nodes since there are only 12 observations per stock. We do not prune the tree because there is only 1 split per stock when modelled with train set of 12 observations.

The SVM provides better accuracy than the other two though the interpretation is not as simple as that of linear model. We did not tune the model since its time consuming when running on 30 stocks

V. Data

A. Data cleaning Appendix A

There are no missing values in any of the datasets. There are some NAs entrees for some of the features which are expected as they would not be some data available for the first/initial observation when referring to the previous weeks data.

The variables open, high, close, low, next_weeks_open and next_weeks_close are changed from factor to numeric and their dollar symbols where removed. Quarter variable is changed from integer to factor and date variable is formatted accordingly.

B. Data preprocessing Appendix B

Created lag plots between each variable and its previous 6 lags to see if the variables have any correlation with their previous lags.
Lags of open, close, high, low, next_weeks_open, next_weeks_close and percent_return_next_dividend had correlations with their previous 2 or 3 lags but strongly correlated with their previous lag 1.
Hence created lags for the variables open, close, high, low and percent_return_next_dividend but not for next_weeks_open, next_weeks_close because in real scenario we wouldn't know the future information which these two variables represent.

We initially split the data by quarter and then by stock. Appendix C
Train: list of 30 stocks for quarter 1 (Jan 2011-Mar 2011)
Test: list of same 30 stocks for quarter 2 (Apr 2011-Jun 2011)

C. Data Limitations

One major limitation is that there are only 25 observations per stock for both quarters making it even less when split into test(13) and train sets(12 observations). This may not give the desired results for the different models.

VI. Findings (Results)

A. Results presented in tables or charts when appropriate
Qualitative analysis of relationship between explanatory variables and their lags where analysed through lagplots Appendix B. Lags of open, close, high, low, next_weeks_open, next_weeks_close and percent_return_next_dividend had correlations with their previous 2 or 3 lags but strongly correlated with their previous lag 1.
Hence created lags for the variables open, close, high, low and percent_return_next_dividend to be used in the model but not for next_weeks_open, next_weeks_close because in real scenario we wouldn't know the future information which these two variables represent. Hence we do not include them in modelling

B. Results reported with respect to hypotheses/models.

Linear Regression: We run linear Regression to model percent_change_price with lags for the variables open, close, high, low and percent_return_next_dividend as predictors for 30 stocks each. We do not use the actual variables since they would be highly correlated as we should not have multicollinearity in which case the information a variable provides about the response is redundant in the presence of the other variables. Appendix D

This model helps in easy interpretation of how each variable effects the response however we get a very high mean squared error (approx. 350 by averaging all the respective mean squared errors of each stock) hence conclude that linear regression may not be preferable in predicting the percent_change_price accurately. One possible reason could be that linear model does not successfully capture the complete variability in response resulting in high error rates. We should try other alternatives.

Decision Trees:

In our case we have only 2 terminal nodes since there are only 12 observations per stock. We do not prune the tree because there is only 1 split per stock when modelled with train set of 12 observations. Appendix E

We get a mean squared error of 10 from averaging mean squared errors of all 30 stocks. This is much better than that of linear regression and can be used to make better prediction than that of linear regression.

Support Vector Regression:

SVM has SVR for modelling continuous response which we use in our analysis.

The model has an average mean squared error of 0.9 which is much better when compared to linear model and considerably better than decision tree. This can be used to predict the percentage change in price of stocks for the following weeks relatively accurately. Appendix F

SVR model is the best model and is uses the last week of the test data set to predict the following week percentage change in price. According to the predictions, out of the 30 stocks, VZ, IBM, HD, JPM, MMM are the top 5 stocks with maximum percent change price. Appendix G

For evaluating risk of stocks, we calculated percentage change of the price of stocks during end of the week and ran a linear regression against that of SP500 data for same time period. The coefficient estimates/Beta of VZ, IBM, HD, JPM, MMM were 0.6, 0.64, 0.495, 0.721, 1.03 respectively.

If Beta is 1 then it replicates the market, is as risky as the market (SP500).

Since the Beta of all these stocks except MMM was below 1 it means they are less volatile hence less risky. But MMM on the other hand being greater than 1 is volatile and risky by 3%.

C. Factual information kept separate from interpretation, inference and evaluation.

> There have been hundreds of systems developed to forecast financial data. Most use some type of artificial intelligence technique from neural networks to GP. Atsalakis and Valavanis state that a goal of the research community is to produce the best results using the least amount of information about the stocks and by developing the least complex model. Many financial forecasting algorithms attempt to meet these two conditions.
>
> Genetic Algorithms are very useful algorithms that can locate optima within very complex domains. Early research in the subject began the 1950's. The 1970's was another period of important research in the area. Research in the subject has been going on continuously since the 1950's and with more powerful computing power we are beginning to see more applied uses for GAs.
> (Brown, M. S., Pelosi, M. & Dirska, H. (2013). n.d.)

VII. Conclusions and Recommendations

We can conclude that though interpretation is easiest among the three for linear regression, it is not suitable as we get a very high mean squared error (approx. 350 by averaging all the respective mean squared errors of each stock) hence conclude that linear regression may not be preferable in predicting the percent_change_price accurately. One possible reason could be that linear model does not successfully capture the complete variability in the time sensitive response, resulting in high error rates. We should try other alternatives.

Though SVM is not as simple to interpret as linear model it has very low error rate hence has much better prediction accuracy than the other two models. Since there is huge improvement in accuracy on switching from linear/Decision tree to SVM, I recommend SVM to be used. Also, regarding the number of observations, larger sample size per stock is recommended for the model to capture different behavior in data and get better prediction results

SVR model is the best model with a very low error rate of 0.9. According to the predictions, out of the 30 stocks, VZ, IBM, HD, JPM, MMM are the top 5 stocks with maximum percent change price.

For evaluating risk of stocks, we look at the coefficient estimates. They are 0.6, 0.64, 0.495, 0.721, 1.03 for VZ, IBM, HD, JPM, MMM respectively.

Since the Beta of all these stocks except MMM was below 1 it means they are less volatile hence less risky. But MMM on the other hand being greater than 1 is volatile and risky by 3%.

Appendix A

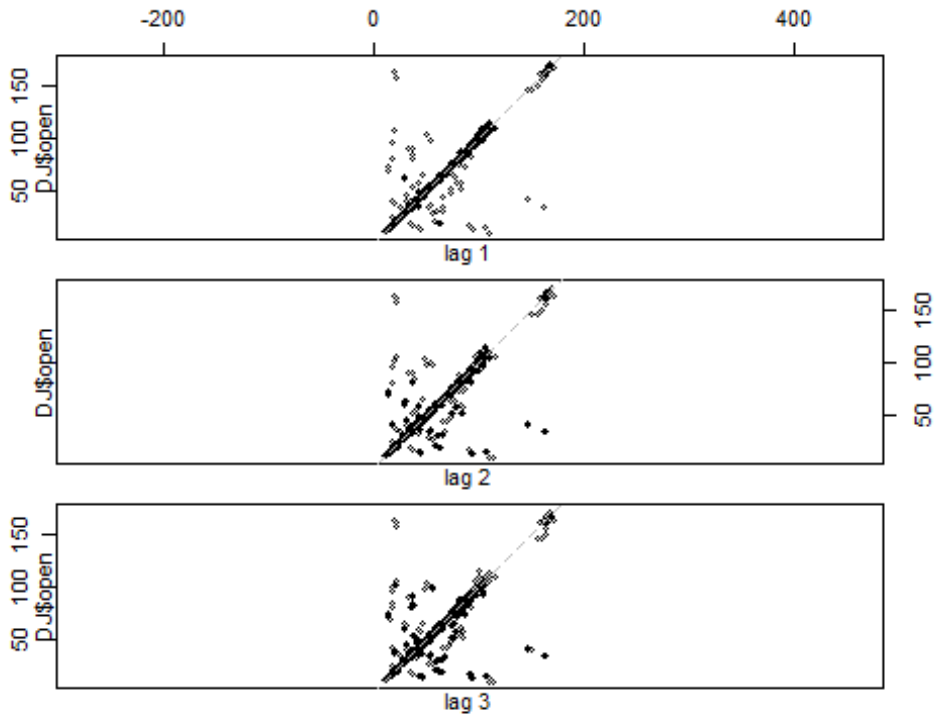
```
DJ1 = read.table("/Users/prady_000/Documents/Vishwa/MSDA/DAA/Dow Jones case s
tudy/dow_jones_index.data",sep = ',', header = T)
DJ = DJ1

#converting variable types
DJ$open = as.numeric(gsub("\\$","",DJ$open))
DJ$high = as.numeric(gsub("\\$","",DJ$high))
DJ$low = as.numeric(gsub("\\$","",DJ$low))
DJ$close = as.numeric(gsub("\\$","",DJ$close))
DJ$next_weeks_open = as.numeric(gsub("\\$","",DJ$next_weeks_open))
DJ$next_weeks_close = as.numeric(gsub("\\$","",DJ$next_weeks_close))
DJ$quarter = as.factor(DJ$quarter)
DJ$date = as.Date(DJ$date, format = "%m/%d/%Y")
```

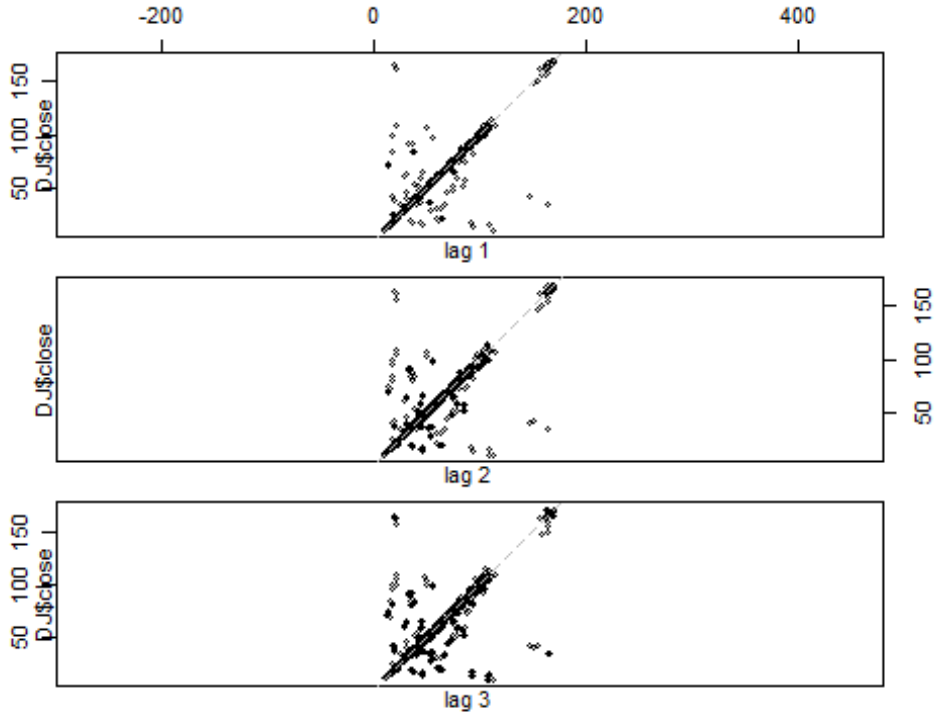
Appendix B

Lag PLots of explanatory variables

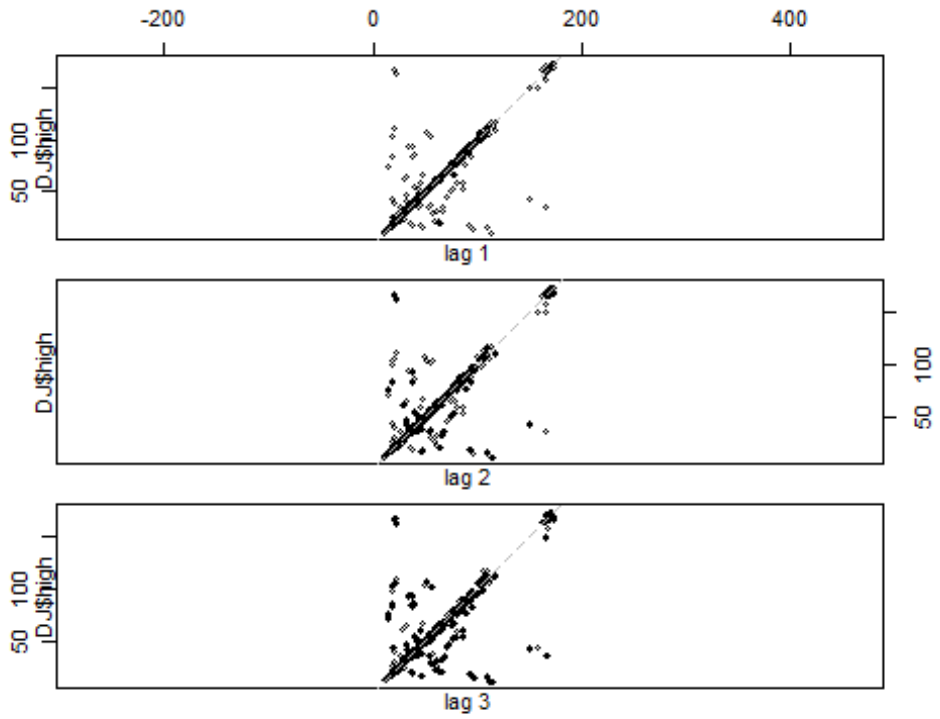
```
lag.plot(DJ$open, set.lags = 1:3)
```



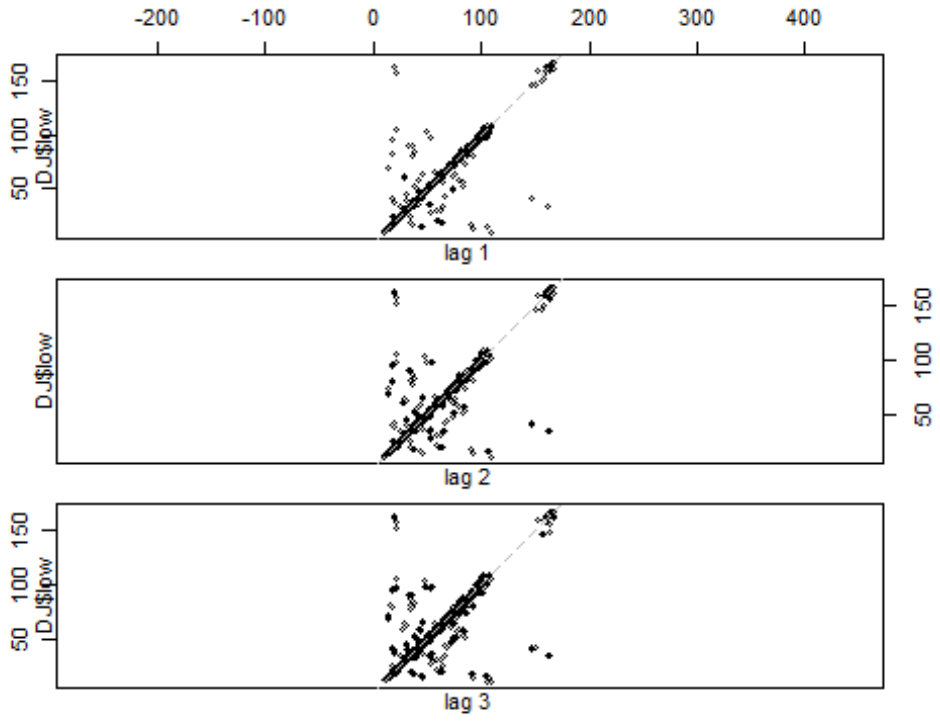

```
lag.plot(DJ$close, set.lags = 1:3)
```



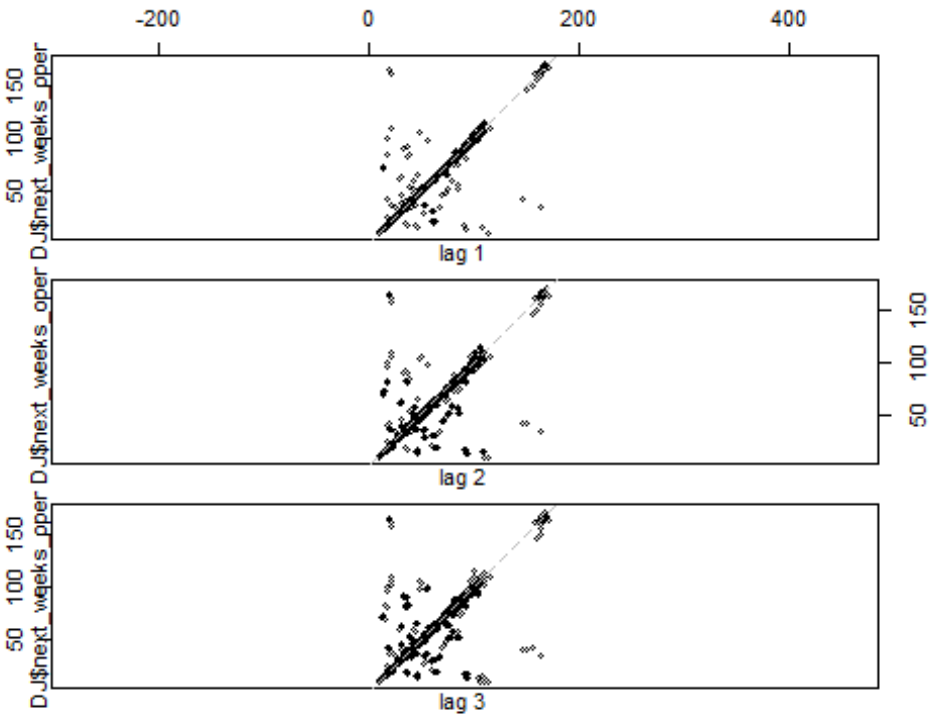

```
lag.plot(DJ$high, set.lags = 1:3)
```



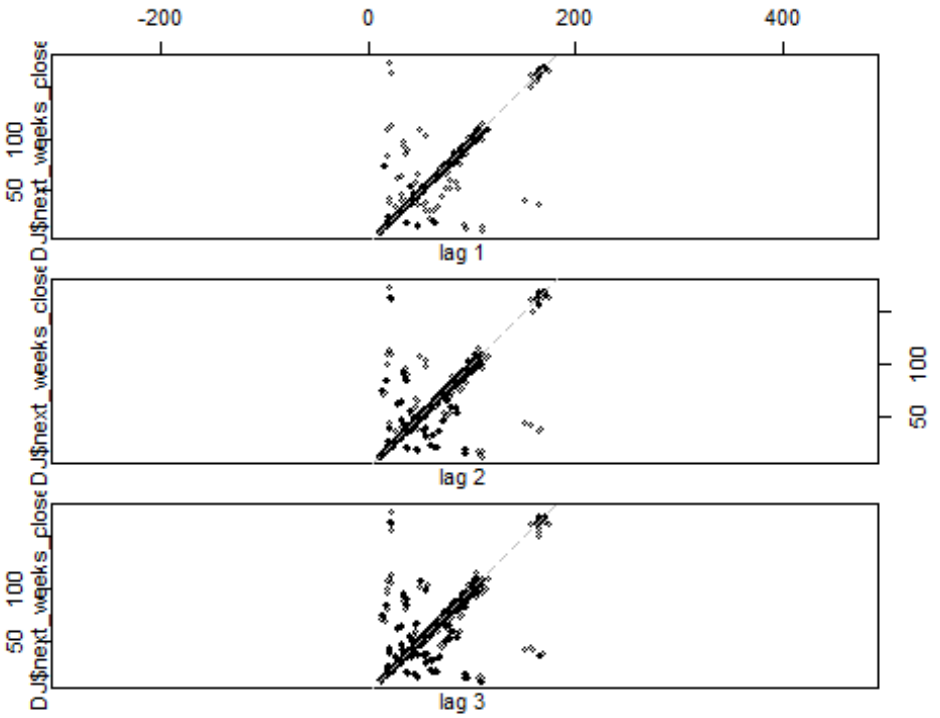

```
lag.plot(DJ$low, set.lags = 1:3)
```



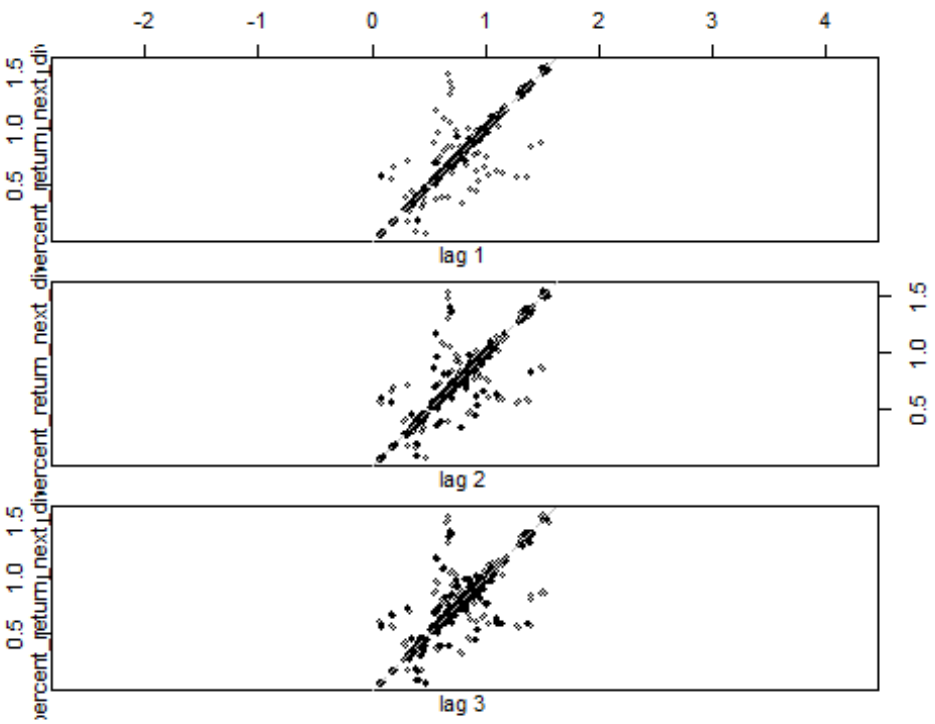

```
lag.plot(DJ$next_weeks_open, set.lags = 1:3)
```




```
lag.plot(DJ$next_weeks_close, set.lags = 1:3)
```




```
lag.plot(DJ$percent_return_next_dividend, set.lags = 1:3)
```




```
#creating lag variables
library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

DJ = DJ %>%
  group_by(stock)%>%
  mutate(open.lag = dplyr::lag(open,n=1))

DJ = DJ %>%
  group_by(stock)%>%
  mutate(close.lag = dplyr::lag(close,n=1))

DJ = DJ %>%
  group_by(stock)%>%
  mutate(high.lag = dplyr::lag(high,n=1))
```

```
DJ = DJ %>%
  group_by(stock)%>%
  mutate(low.lag = dplyr::lag(low,n=1))
DJ = DJ %>%
  group_by(stock)%>%
  mutate(next_weeks_open.lag = dplyr::lag(next_weeks_open,n=1))
DJ = DJ %>%
  group_by(stock)%>%
  mutate(next_weeks_close.lag = dplyr::lag(next_weeks_close,n=1))

DJ = DJ %>%
  group_by(stock)%>%
  mutate(percent_return_next_dividend.lag = dplyr::lag(percent_return_next_di
vidend,n=1))
```

Appendix C

```
summary(DJ)
```

```
##  quarter     stock          date                 open
##  1:360   AA     : 25   Min.   :2011-01-07   Min.   : 10.59
##  2:390   AXP    : 25   1st Qu.:2011-02-18   1st Qu.: 29.83
##          BA     : 25   Median :2011-04-01   Median : 45.97
##          BAC    : 25   Mean   :2011-03-31   Mean   : 53.65
##          CAT    : 25   3rd Qu.:2011-05-13   3rd Qu.: 72.72
##          CSCO   : 25   Max.   :2011-06-24   Max.   :172.11
##          (Other):600
##       high             low              close            volume
##  Min.   : 10.94   Min.   : 10.40   Min.   : 10.52   Min.   :9.719e+06
##  1st Qu.: 30.63   1st Qu.: 28.72   1st Qu.: 30.36   1st Qu.:3.087e+07
##  Median : 46.88   Median : 44.80   Median : 45.93   Median :5.306e+07
##  Mean   : 54.67   Mean   : 52.64   Mean   : 53.73   Mean   :1.175e+08
##  3rd Qu.: 74.29   3rd Qu.: 71.04   3rd Qu.: 72.67   3rd Qu.:1.327e+08
##  Max.   :173.54   Max.   :167.82   Max.   :170.58   Max.   :1.453e+09
##
##  percent_change_price percent_change_volume_over_last_wk
##  Min.   :-15.42290    Min.   :-61.4332
##  1st Qu.: -1.28805    1st Qu.:-19.8043
##  Median :  0.00000    Median :  0.5126
##  Mean   :  0.05026    Mean   :  5.5936
##  3rd Qu.:  1.65089    3rd Qu.: 21.8006
##  Max.   :  9.88223    Max.   :327.4089
##                       NA's   :30
##  previous_weeks_volume next_weeks_open  next_weeks_close
##  Min.   :9.719e+06     Min.   : 10.52   Min.   : 10.52
##  1st Qu.:3.068e+07     1st Qu.: 30.32   1st Qu.: 30.46
##  Median :5.295e+07     Median : 46.02   Median : 46.12
```

```
##  Mean   :1.174e+08     Mean   : 53.70   Mean   : 53.89
##  3rd Qu.:1.333e+08     3rd Qu.: 72.72   3rd Qu.: 72.92
##  Max.   :1.453e+09     Max.   :172.11   Max.   :174.54
##  NA's   :30
##  percent_change_next_weeks_price days_to_next_dividend
##  Min.   :-15.4229                Min.   :  0.00
##  1st Qu.: -1.2221                1st Qu.: 24.00
##  Median :  0.1012                Median : 47.00
##  Mean   :  0.2385                Mean   : 52.53
##  3rd Qu.:  1.8456                3rd Qu.: 69.00
##  Max.   :  9.8822                Max.   :336.00
##
##  percent_return_next_dividend    open.lag        close.lag
##  Min.   :0.06557              Min.   : 10.89   Min.   : 10.68
##  1st Qu.:0.53455              1st Qu.: 29.52   1st Qu.: 29.97
##  Median :0.68107              Median : 45.97   Median : 45.93
##  Mean   :0.69183              Mean   : 53.69   Mean   : 53.78
##  3rd Qu.:0.85429              3rd Qu.: 72.70   3rd Qu.: 72.77
##  Max.   :1.56421              Max.   :172.11   Max.   :170.58
##                               NA's   :30       NA's   :30
##     high.lag         low.lag       next_weeks_open.lag
##  Min.   : 11.12   Min.   : 10.40   Min.   : 10.59
##  1st Qu.: 30.48   1st Qu.: 28.53   1st Qu.: 29.96
##  Median : 46.88   Median : 44.80   Median : 46.02
##  Mean   : 54.69   Mean   : 52.67   Mean   : 53.75
##  3rd Qu.: 74.28   3rd Qu.: 71.11   3rd Qu.: 72.81
##  Max.   :173.54   Max.   :167.82   Max.   :172.11
##  NA's   :30       NA's   :30       NA's   :30
##  next_weeks_close.lag percent_return_next_dividend.lag
##  Min.   : 10.52       Min.   :0.06557
##  1st Qu.: 30.41       1st Qu.:0.53409
##  Median : 46.12       Median :0.67788
##  Mean   : 53.83       Mean   :0.69085
##  3rd Qu.: 72.77       3rd Qu.:0.85472
##  Max.   :170.58       Max.   :1.56421
##  NA's   :30           NA's   :30
```

```
#splitting into train and test sets
DJList=(split(DJ, DJ$quarter))
train=DJList[[1]]
test = DJList[[2]]
train1 = split(train, train$stock)
test1 = split(test, test$stock)
```

Appendix D

Linear Model

```
glmfxn <- function (trainstock, teststock,formula) {


    glmfit <- glm(formula,data = trainstock)

    linear.predict <- predict.glm(glmfit, newdata = teststock)
    DJ.test = teststock$percent_change_next_weeks_price
    err.rate = mean((linear.predict-DJ.test)^2)

  }
formula = percent_change_price ~ open.lag + high.lag + low.lag + close.lag +
percent_return_next_dividend.lag
err = NULL
for (l in names(train1))
{

  x=train1[[l]]
  y=test1[[l]]
  err = c(err,glmfxn(x,y,formula))
}
#calculating average of error rate for all 30 stocks
avg_glm = mean(err);avg_glm
```

```
## [1] 349.3643
```

Appendix E

Decision Tree

```
library(tree)
DTfxn <- function (trainstock, teststock,formula) {


    tree.DJ = tree(formula,data = trainstock)
    summary(tree.DJ)

    yhat = predict(tree.DJ, newdata = teststock)
    DJ.test = teststock$percent_change_next_weeks_price
    err.rate = mean((yhat-DJ.test)^2)
}
err_DT = NULL
for (l in names(train1))
{
  x=train1[[l]]
  y=test1[[l]]
  err_DT = c(err_DT,DTfxn(x,y,formula))
}
```

```
#calculating average of error rate for all 30 stocks
avg_DT = mean(err_DT);avg_DT

## [1] 10.06285
```

Appendix F

SVR

```
library(e1071)

SVRfxn <- function (trainstock, teststock,formula) {
#scaling the continuous variables before building svm model
trainstock[,4:23] = scale(trainstock[,4:23])
teststock[,4:23] = scale(teststock[,4:23])

linear_DJ = svm(formula = formula, data = trainstock)

    svm.test = predict(linear_DJ, newdata = teststock)

    DJ.test = teststock$percent_change_next_weeks_price
    err.rate = mean((svm.test-DJ.test)^2)

}
err_SVR = NULL
for (l in names(train1))
{
  x=train1[[l]]
  y=test1[[l]]
  err_SVR = c(err_SVR,SVRfxn(x,y,formula))
}
#calculating average of error rate for all 30 stocks
avg_SVR = mean(err_SVR);avg_SVR

## [1] 0.9767443
```

Appendix G

CAPM

```
SP500 <- read.csv("/Users/prady_000/Documents/Vishwa/MSDA/DAA/Dow Jones case
study/^GSPC.csv")
stock = read.table("/Users/prady_000/Documents/Vishwa/MSDA/DAA/Dow Jones case
study/dow_jones_index.data",sep = ',', header = T)
stock$close = as.numeric(gsub("\\$","",stock$close))
stockquat = split(stock,stock$quarter)
stockquat2 = stockquat[[2]]
capm_train = split(stockquat2,stockquat2$stock)
```

```r
library('quantmod')

library('tseries')

#calculating percentage change in close price
ReturnSP500 = na.omit(Delt(SP500[,5]))
ReturnAA = na.omit(Delt(capm_train$AA[,7]))
ReturnAXP = na.omit(Delt(capm_train$AXP[,7]))
ReturnBA = na.omit(Delt(capm_train$BA[,7]))
ReturnBAC = na.omit(Delt(capm_train$BAC[,7]))
ReturnCAT = na.omit(Delt(capm_train$CAT[,7]))
ReturnCSCO = na.omit(Delt(capm_train$CSCO[,7]))
ReturnCVX = na.omit(Delt(capm_train$CVX[,7]))
ReturnDD = na.omit(Delt(capm_train$DD[,7]))
ReturnDIS = na.omit(Delt(capm_train$DIS[,7]))
ReturnGE = na.omit(Delt(capm_train$GE[,7]))
ReturnHD = na.omit(Delt(capm_train$HD[,7]))
ReturnHPQ = na.omit(Delt(capm_train$HPQ[,7]))
ReturnIBM = na.omit(Delt(capm_train$IBM[,7]))
ReturnINTC = na.omit(Delt(capm_train$INTC[,7]))
ReturnJNJ = na.omit(Delt(capm_train$JNJ[,7]))
ReturnJPM = na.omit(Delt(capm_train$JPM[,7]))
ReturnKO = na.omit(Delt(capm_train$KO[,7]))
ReturnKRFT = na.omit(Delt(capm_train$KRFT[,7]))
ReturnMCD = na.omit(Delt(capm_train$MCD[,7]))
ReturnMMM = na.omit(Delt(capm_train$MMM[,7]))
ReturnMRK = na.omit(Delt(capm_train$MRK[,7]))
ReturnMSFT = na.omit(Delt(capm_train$MSFT[,7]))
ReturnPFE = na.omit(Delt(capm_train$PFE[,7]))
ReturnPG = na.omit(Delt(capm_train$PG[,7]))
ReturnT = na.omit(Delt(capm_train$T[,7]))
ReturnTRV = na.omit(Delt(capm_train$TRV[,7]))
ReturnUTX = na.omit(Delt(capm_train$UTX[,7]))
ReturnVZ = na.omit(Delt(capm_train$VZ[,7]))
ReturnWMT = na.omit(Delt(capm_train$WMT[,7]))
ReturnXOM = na.omit(Delt(capm_train$XOM[,7]))

MyData = cbind(ReturnSP500,ReturnAA,ReturnAXP,ReturnBA,ReturnBAC,ReturnCAT,Re
turnCSCO,ReturnCVX,ReturnDD,ReturnDIS,ReturnGE,ReturnHD,ReturnHPQ,ReturnIBM,
ReturnINTC,ReturnJNJ,ReturnJPM,ReturnKO,ReturnKRFT,ReturnMCD,ReturnMMM,Return
MRK,ReturnMSFT,ReturnPFE,ReturnPG,ReturnT,ReturnTRV,ReturnUTX,
ReturnVZ,ReturnWMT,ReturnXOM)

colnames(MyData) = c("SP500", "AA", "AXP","BA","BAC","CAT","CSCO","CVX","DD",
"DIS","GE","HD","HPQ","IBM","INTC","JNJ","JPM","KO","KRFT","MCD",
"MMM","MRK","MSFT","PFE","PG","T","TRV","UTX","VZ","WMT","XOM")

lm.AA = lm(AA ~ SP500, data = as.data.frame(MyData))
BetaAA = summary(lm.AA)$coefficients[2,1]
print(paste0("BetaAA:",BetaAA))
```

```
## [1] "BetaAA:1.05917746066696"
```

```
lm.AXP = lm(AXP ~ SP500, data = as.data.frame(MyData))
BetaAXP = summary(lm.AXP)$coefficients[2,1]
print(paste0("BetaAXP:",BetaAXP))
```

```
## [1] "BetaAXP:1.32055085168069"
```

```
lm.BA = lm(BA ~ SP500, data = as.data.frame(MyData))
BetaBA = summary(lm.BA)$coefficients[2,1]
print(paste0("BetaBA:",BetaBA))
```

```
## [1] "BetaBA:1.7920032506852"
```

```
lm.BAC = lm(BAC ~ SP500, data = as.data.frame(MyData))
BetaBAC = summary(lm.BAC)$coefficients[2,1]
print(paste0("BetaBAC:",BetaBAC))
```

```
## [1] "BetaBAC:0.37038904522729"
```

```
lm.CAT = lm(CAT ~ SP500, data = as.data.frame(MyData))
BetaCAT = summary(lm.CAT)$coefficients[2,1]
print(paste0("BetaCAT:",BetaCAT))
```

```
## [1] "BetaCAT:2.03800197167765"
```

```
lm.CSCO = lm(CSCO ~ SP500, data = as.data.frame(MyData))
BetaCSCO = summary(lm.CSCO)$coefficients[2,1]
print(paste0("BetaCSCO:",BetaCSCO))
```

```
## [1] "BetaCSCO:1.22938650665767"
```

```
lm.CVX = lm(CVX ~ SP500, data = as.data.frame(MyData))
BetaCVX = summary(lm.CVX)$coefficients[2,1]
print(paste0("BetaCVX:",BetaCVX))
```

```
## [1] "BetaCVX:1.16940931889395"
```

```
lm.DD = lm(DD ~ SP500, data = as.data.frame(MyData))
BetaDD = summary(lm.DD)$coefficients[2,1]
print(paste0("BetaDD:",BetaDD))
```

```
## [1] "BetaDD:1.24339360728926"
```

```
lm.DIS = lm(DIS ~ SP500, data = as.data.frame(MyData))
BetaDIS = summary(lm.DIS)$coefficients[2,1]
print(paste0("BetaDIS:",BetaDIS))
```

```
## [1] "BetaDIS:1.14125236849973"
```

```
lm.GE = lm(GE ~ SP500, data = as.data.frame(MyData))
BetaGE = summary(lm.GE)$coefficients[2,1]
print(paste0("BetaGE:",BetaGE))
```

```
## [1] "BetaGE:1.04423696907275"

lm.HD = lm(HD ~ SP500, data = as.data.frame(MyData))
BetaHD = summary(lm.HD)$coefficients[2,1]
print(paste0("BetaHD:",BetaHD))

## [1] "BetaHD:0.495107791734213"

lm.HPQ = lm(HPQ ~ SP500, data = as.data.frame(MyData))
BetaHPQ = summary(lm.HPQ)$coefficients[2,1]
print(paste0("BetaHPQ:",BetaHPQ))

## [1] "BetaHPQ:0.365618975358897"

lm.IBM = lm(IBM ~ SP500, data = as.data.frame(MyData))
BetaIBM = summary(lm.IBM)$coefficients[2,1]
print(paste0("BetaIBM:",BetaIBM))

## [1] "BetaIBM:0.642048559353456"

lm.INTC = lm(INTC ~ SP500, data = as.data.frame(MyData))
BetaINTC = summary(lm.INTC)$coefficients[2,1]
print(paste0("BetaINTC:",BetaINTC))

## [1] "BetaINTC:2.20732729470623"

lm.JNJ = lm(JNJ ~ SP500, data = as.data.frame(MyData))
BetaJNJ = summary(lm.JNJ)$coefficients[2,1]
print(paste0("BetaJNJ:",BetaJNJ))

## [1] "BetaJNJ:1.07593377308688"

lm.JPM = lm(JPM ~ SP500, data = as.data.frame(MyData))
BetaJPM = summary(lm.JPM)$coefficients[2,1]
print(paste0("BetaJPM:",BetaJPM))

## [1] "BetaJPM:0.72139828351022"

lm.KO = lm(KO ~ SP500, data = as.data.frame(MyData))
BetaKO = summary(lm.KO)$coefficients[2,1]
print(paste0("BetaKO:",BetaKO))

## [1] "BetaKO:0.115518754443971"

lm.KRFT = lm(KRFT ~ SP500, data = as.data.frame(MyData))
BetaKRFT = summary(lm.KRFT)$coefficients[2,1]
print(paste0("BetaKRFT:",BetaKRFT))

## [1] "BetaKRFT:0.275484936905726"

lm.MCD = lm(MCD ~ SP500, data = as.data.frame(MyData))
BetaMCD = summary(lm.MCD)$coefficients[2,1]
print(paste0("BetaMCD:",BetaMCD))
```

```
## [1] "BetaMCD:0.37450075291131"

lm.MMM = lm(MMM ~ SP500, data = as.data.frame(MyData))
BetaMMM = summary(lm.MMM)$coefficients[2,1]
print(paste0("BetaMMM:",BetaMMM))

## [1] "BetaMMM:1.03867349686881"

lm.MRK = lm(MRK ~ SP500, data = as.data.frame(MyData))
BetaMRK = summary(lm.MRK)$coefficients[2,1]
print(paste0("BetaMRK:",BetaMRK))

## [1] "BetaMRK:0.684800273623773"

lm.MSFT = lm(MSFT ~ SP500, data = as.data.frame(MyData))
BetaMSFT = summary(lm.MSFT)$coefficients[2,1]
print(paste0("BetaMSFT:",BetaMSFT))

## [1] "BetaMSFT:0.81390373939246"

lm.PFE = lm(PFE ~ SP500, data = as.data.frame(MyData))
BetaPFE = summary(lm.PFE)$coefficients[2,1]
print(paste0("BetaPFE:",BetaPFE))

## [1] "BetaPFE:1.03498786828371"

lm.PG = lm(PG ~ SP500, data = as.data.frame(MyData))
BetaPG = summary(lm.PG)$coefficients[2,1]
print(paste0("BetaPG:",BetaPG))

## [1] "BetaPG:0.330848591486319"

lm.T = lm(T ~ SP500, data = as.data.frame(MyData))
BetaT = summary(lm.T)$coefficients[2,1]
print(paste0("BetaT:",BetaT))

## [1] "BetaT:0.52432210810945"

lm.TRV = lm(TRV ~ SP500, data = as.data.frame(MyData))
BetaTRV = summary(lm.TRV)$coefficients[2,1]
print(paste0("BetaTRV:",BetaTRV))

## [1] "BetaTRV:0.929462269126195"

lm.UTX = lm(UTX ~ SP500, data = as.data.frame(MyData))
BetaUTX = summary(lm.UTX)$coefficients[2,1]
print(paste0("BetaUTX:",BetaUTX))

## [1] "BetaUTX:1.33267885913185"

lm.VZ = lm(VZ ~ SP500, data = as.data.frame(MyData))
BetaVZ = summary(lm.VZ)$coefficients[2,1]
print(paste0("BetaVZ:",BetaVZ))
```

```
## [1] "BetaVZ:0.602730504682449"
```

```
lm.WMT = lm(WMT ~ SP500, data = as.data.frame(MyData))
BetaWMT = summary(lm.WMT)$coefficients[2,1]
print(paste0("BetaWMT:",BetaWMT))
```

```
## [1] "BetaWMT:0.706458270909264"
```

```
lm.XOM = lm(XOM ~ SP500, data = as.data.frame(MyData))
BetaXOM = summary(lm.XOM)$coefficients[2,1]
print(paste0("BetaXOM:",BetaXOM))
```

```
## [1] "BetaXOM:1.27577431140436"
```