Yewno Assignment

Harsh Vardhan Tiwari

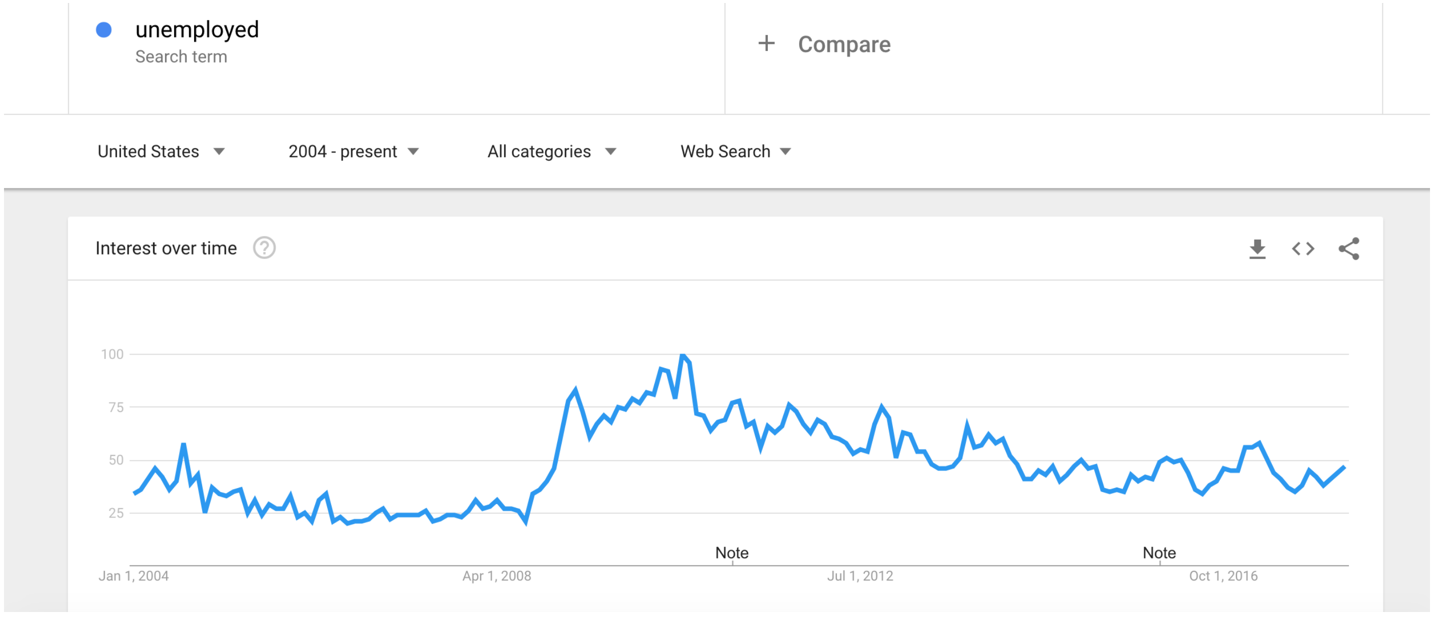
3/11/2018

Question 1

Predicting Initial Claims of Unemployment using Google correlate

The Initial Claims data describes the number of people who filed for unemployment benefits in the previous week and is released every Thursday for the week ending Saturday. The data is seen as a good leading indicator for the health of the US job market and the economy in general. This data also plays an important role in setting the inflation expectations and is therefore a key macroeconomic indicator in the financial markets. The 5-day lag in the release makes accurate prediction of this very valuable to portfolio managers and traders.

Most commonly used approaches of using alternative datasets for predicting macroeconomic trends like Initial Claims start with a baseline time series model like Autoregressive models, but these models work well only during trends and fail to identify turning points (or changes in trend regimes) leading to poor generalization capabilites. Some recent approaches have tried to add to the predictive power of the classic AR model by adding features from these alternative datasets. Google trends is a very commonly used data source for this purpose. Google trends provides a time series index of the volume of queries corresponding to a tag under a certain category users enter into Google in a given geographic area. When using Google trends, we need to identity the tags which might relate the most to our time series of interests, while this is very useful this process can be very cumbersome and tricky as a certain tag can be used in different contexts and there might be many alternative search tags relating to a certain topic. Most commonly used tags for someone unemployed could be "unemployed,"how to find a job" and a quick search for "unemployed" tag gives the following result, which tends to make sense highlighting the high unemployment rates during the financial crisis of 2008.

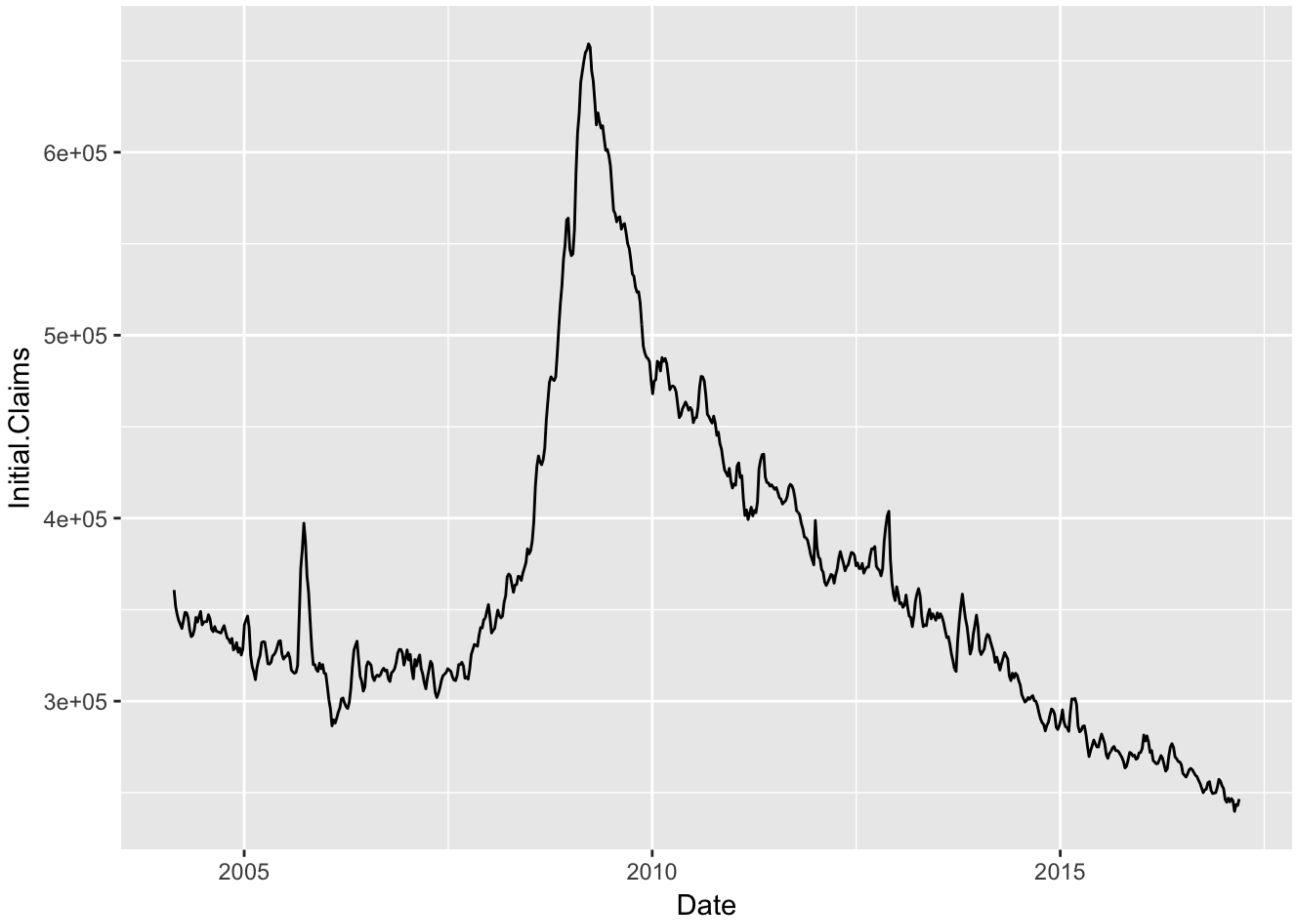


Google Trend search for tag 'unemployed'

I wanted to use a native approach that deviates from the standard AR modelling framework offering the potential of stronger out-sample predictive power and also solves the problem of hand-picking features in the beginning. Google Correlate comes to our rescue. Google correlate offers an API for uploading a weekly/monthly time series to find search patterns that correlate the most to a user entered time series. This solves the problem of handpicking a few search tags and provides us with carefully selected search tags that have high correlation to the time-series of our interest: Initial Claims.

Data Highlights 1) Time Period: 02/22/2004 to 03/12/2017 2) Feature Set: Top 100 most correlated search tags to Initial Claims weekly data are used as features, all having correlation > 0.9. The time series for each search tag is scaled to have mean 0 and variance 1 by default from google correlate API

```
library(ggplot2)
myData <- read.csv(file="/Users/Harsh/Documents/R/Yewno/Initial Claims Prediction/google
 correlate-Initial_Claims.csv", header=TRUE, stringsAsFactors=FALSE,
                   fileEncoding="latin1")
myData $Date <- as.Date( myData$Date, '%m/%d/%y')
ggplot( data = myData, aes( Date, Initial.Claims )) + geom_line()
```



Linear Regression Model: Train Data

We propose a multiple linear regression model using all the to start with to start with. We are cognizant that having so many correlated features make our model prone to overfitting, so we will test our model both in-sample and out-sample. We have 682 rows in our dataset, we use a 50-50 split for training and test datasets.

```
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
## define predictor and independent variable
x <- model.matrix(Initial.Claims ~ . - Date, myData)[,-1]
y <- myData$Initial.Claims

## set the seed to make your partition reproductible
set.seed(123)

## 50% of the sample size
nRows <- nrow(myData)
smp_size <- floor(0.50 * nRows)
train <- sample(seq_len(nRows), size = smp_size)
test <- -train

## training set
xtrain <- x[train,]
ytrain <- y[train]
## test set
xtest <- x[test,]
ytest <- y[test]
```

Model Statistics: In-Sample

```
myData <- subset(myData, select = -Date)
ols.model <- lm(Initial.Claims ~ ., data = myData, subset = train)
summary(ols.model)
```

```
##
## Call:
## lm(formula = Initial.Claims ~ ., data = myData, subset = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36128  -5511    -31   5355  60451
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      361880.4      794.3 455.622  < 2e-16 ***
## uncoached                         -6187.3    10079.5  -0.614 0.539897
## hear.it.first                       565.6     4915.9   0.115 0.908500
## laguna.beach.jeans                -3034.5     8144.2  -0.373 0.709780
## dare4distance                     22016.9    14244.8   1.546 0.123518
## mgmt.kids.lyrics                   2536.3     7379.4   0.344 0.731375
## craigslist.helper                  4720.3     9280.7   0.509 0.611494
## perfect.world                     -4965.6     8216.9  -0.604 0.546200
## nickasaur                          6223.5    12362.7   0.503 0.615136
## green.jobs                        -1180.9     5835.1  -0.202 0.839799
## dare4distance.lyrics                882.2    11358.0   0.078 0.938151
## paretologic                        5576.5     5845.2   0.954 0.341028
## the.moon.lyrics                    9162.0     5055.7   1.812 0.071202 .
## to.make.a.website                 30109.3    50806.9   0.593 0.553992
## ihiphop                           19969.9     7071.4   2.824 0.005141 **
## city.dreams                       17901.3    12402.2   1.443 0.150214
## used.blackberry.curve              8266.8     5942.1   1.391 0.165447
## ex3                                7194.4     4349.5   1.654 0.099416 .
## rtl8168c                          -1653.1     8730.9  -0.189 0.849989
## anytwat                            2649.8     6976.9   0.380 0.704437
## make.a.website                     9077.5    14913.3   0.609 0.543309
## how.to.make.a.website            -43404.7    47795.9  -0.908 0.364722
## gamesofdesire.com                -13863.8    10112.6  -1.371 0.171671
## fxcm.micro                         -570.1     5737.2  -0.099 0.920932
## big.city.dreams                  -12473.9    13781.4  -0.905 0.366305
## loan.modification                  4022.1    16099.8   0.250 0.802936
## att.webmail                       11180.2    15989.5   0.699 0.485090
## on.the.moon.lyrics                 1905.3     6232.9   0.306 0.760105
## naruto.wire                        -819.8     8643.8  -0.095 0.924515
## independent.producers.of.america    770.3     5064.9   0.152 0.879245
## view.pdf.files                     3154.5     4090.1   0.771 0.441321
## coomclips                         19170.2     7646.8   2.507 0.012840 *
## fail.blog                         -3645.9    11906.8  -0.306 0.759714
## dining.deals                       4893.3     4376.6   1.118 0.264666
## demotivate                        -5805.7     4902.6  -1.184 0.237504
## alone.in.this.bed                 -5008.3    11131.8  -0.450 0.653181
## loan.modifications               -16957.8    12026.3  -1.410 0.159817
## iphone.development               -11096.2     7370.7  -1.505 0.133521
## loan.mod                          10306.6    10673.2   0.966 0.335189
## driver.scanner                     1664.7     5368.4   0.310 0.756766
## zomganime                           889.6     9385.7   0.095 0.924568
## media.converter                    5311.5     7120.8   0.746 0.456451
## upgrade.blackberry.os               691.5     4575.1   0.151 0.879983
## soju.com                        -7711.8     9506.1  -0.811 0.418028    
## iphone.opengl                     1981.0     6652.3   0.298 0.766124    
## crackberry                       -7039.9    12238.4  -0.575 0.565676    
## makano.lyrics                    -1142.3     5354.8  -0.213 0.831254    
## realtek.rtl8168c                  2939.2     7776.7   0.378 0.705802    
## free.applications                -6123.0     7688.8  -0.796 0.426615    
## ich10r                           -8389.9     7571.3  -1.108 0.268918    
## ziza.es                          15359.0     6878.8   2.233 0.026485 *  
## alone.in.this.bed.lyrics          8280.1     8680.2   0.954 0.341087    
## snippet.and.ink                   7105.8     7149.8   0.994 0.321304    
## anivide                          16784.7     7191.5   2.334 0.020424 *  
## woome                             4970.3     8678.8   0.573 0.567384    
## drag.racer.v4                     4687.1     5770.6   0.812 0.417461    
## e74                               3400.2     7848.4   0.433 0.665234    
## mimo777                          -1755.5     8029.8  -0.219 0.827128    
## japan.lingzhi                   -13752.4    10077.0  -1.365 0.173616    
## lg.cu920                          -327.7     5728.2  -0.057 0.954423    
## ed.hardy.baby                     1217.0     5650.2   0.215 0.829641    
## nvidia.9500                      -4337.6     7507.3  -0.578 0.563954    
## applications.for.samsung          5595.5     5141.7   1.088 0.277573    
## loan.modification.help            4224.5     5221.2   0.809 0.419256    
## um175                           -10814.6    11050.6  -0.979 0.328740    
## applications.for                 25265.0     5580.6   4.527 9.41e-06 ***
## ww.hulu.com                       1470.5     5150.5   0.286 0.775507    
## yahoo.babel                      -8261.3     7333.8  -1.126 0.261089    
## attack..attack.                  -4084.0     4999.1  -0.817 0.414770    
## atlantica                        -4746.7     6213.6  -0.764 0.445663    
## driver.detective                  2605.8     9598.1   0.271 0.786245    
## a.beautiful.mess.lyrics           2746.0     3145.2   0.873 0.383483    
## big.city.dreams.lyrics          -12299.3     7136.7  -1.723 0.086107 .  
## winster.com                      -9365.7     5041.0  -1.858 0.064409 .  
## brokencyde                      -15819.6     9739.1  -1.624 0.105615    
## smcgui.exe                       -6002.4     7799.7  -0.770 0.442310    
## ich10                            -1549.5     6229.5  -0.249 0.803776    
## windows.internet.explorer.8      -2820.0     5842.8  -0.483 0.629783    
## blackberry.curve.battery         -9323.3     6809.1  -1.369 0.172205    
## X_æçÎÊ.                            169.8     7252.9   0.023 0.981342    
## mp3locker                        -5448.6     5586.6  -0.975 0.330396    
## smelyalata                       -9794.9     7695.5  -1.273 0.204319    
## thunderfap                       -1573.1     4571.7  -0.344 0.731073    
## irfree                           -5431.2     6121.7  -0.887 0.375859    
## mailtrust.com                     2554.7     6163.5   0.414 0.678887    
## perfect.world.database            7790.4     6285.8   1.239 0.216418    
## arthemia                          -969.4     6445.9  -0.150 0.880584    
## croco                            25689.0     7733.2   3.322 0.001034 ** 
## georgia.unemployment              4021.5     3020.0   1.332 0.184244    
## craigshelper                      5491.4     9674.2   0.568 0.570817    
## sm.bus.controller.driver         10306.8     2815.4   3.661 0.000309 ***
## colorado.sunrise.lyrics          -1497.5     8841.8  -0.169 0.865654    
## iphone.or.blackberry             -6899.0     6936.4  -0.995 0.320931    
## i.need.a.job                     -2614.4     3581.7  -0.730 0.466135    
## jackie.boyz                      -2819.4     5910.2  -0.477 0.633769    
## my.soju.com                      -2542.4     7207.0  -0.353 0.724567    
## X2.day.diet.japan               -55086.8    33400.6  -1.649 0.100400    
## dem.get.away.boyz                    12960.6     4840.3   2.678 0.007927 **
## X2.day.diet.japan.lingzhi            66293.6    32238.3   2.056 0.040829 *
## w350a                                -1889.6     6478.5  -0.292 0.770786
## parking.games.com                     -700.4     5131.6  -0.136 0.891545
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12820 on 240 degrees of freedom
## Multiple R-squared:  0.9855, Adjusted R-squared:  0.9795
## F-statistic: 163.3 on 100 and 240 DF,  p-value: < 2.2e-16
```

Model Interpretation and Analysis

A quick note: In the search tags from google correlate above dots were actually spaces, so “i.need.a.job” is actually the search query “i need a job”

We find a good fit achieving adjusted R-squared close to 0.9795. Considering that our universe of features is so big it only makes sense to separate the most important features. When we look for the most significant variables in our model, which correspond to the ones with the lowest p-values above and also highlighted with asterisk, we realize most of the tags really don’t make a lot of sense intuitively. I looked into a couple of tags like ihiphop, dare4distance.lyrics and soon realized that they represent search patterns that peaked around 2008-2010 and dropped after. Given that Initial Claims are is range bound in general in this period apart from the peak during 2008-2010 period, it is no surprise that high correlation alone is not very useful. We need some manual filtering to narrow down to meaningful search tags from the rest. I tried running a linear regression with just one search tag that makes 100% sense: “i need a job”.

```
ols.model1 <- lm(Initial.Claims ~ i.need.a.job, data = myData, subset = train)
summary(ols.model1)
```

```
##
## Call:
## lm(formula = Initial.Claims ~ i.need.a.job, data = myData, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -174528  -18540   -1279   16387  166531
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    363701       2078  175.00   <2e-16 ***
## i.need.a.job    79853       2055   38.86   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38370 on 339 degrees of freedom
## Multiple R-squared:  0.8167, Adjusted R-squared:  0.8161
## F-statistic:  1510 on 1 and 339 DF,  p-value: < 2.2e-16
```

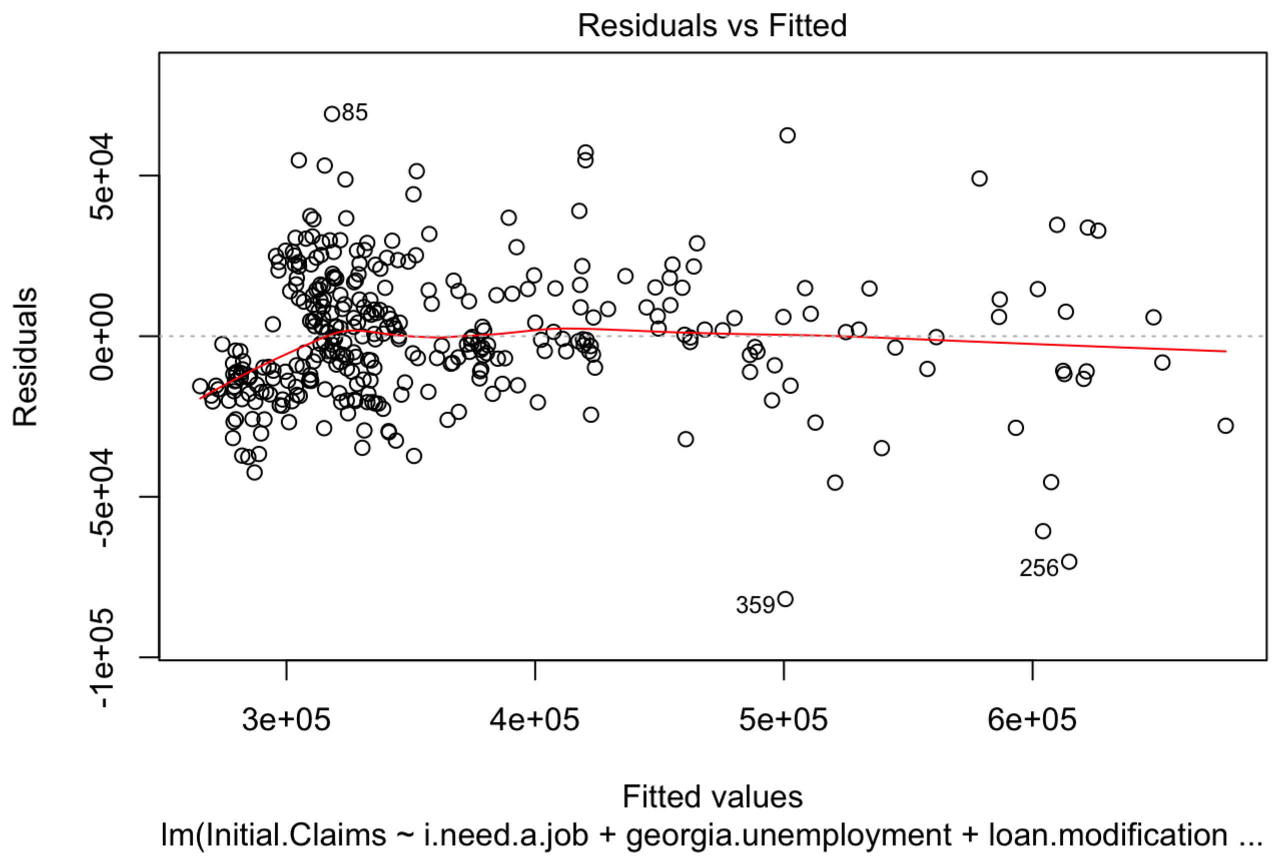
We quickly realize that even this feature alone has an adjusted R-squared of 0.8161. We try adding some more features to get our adjusted R-squared closer to previous value of 0.9795. Lets try a few more hand picked features. In particular we pick search tags that directly imply unemployment like “i need a job”, “georgia unemployment" and also that might be a consequence of unemployment like finding alternative sources of income like "craigslist helper" and modifying loan payment because of lack of income like "loan modification". Lets give these 4 features a shot.

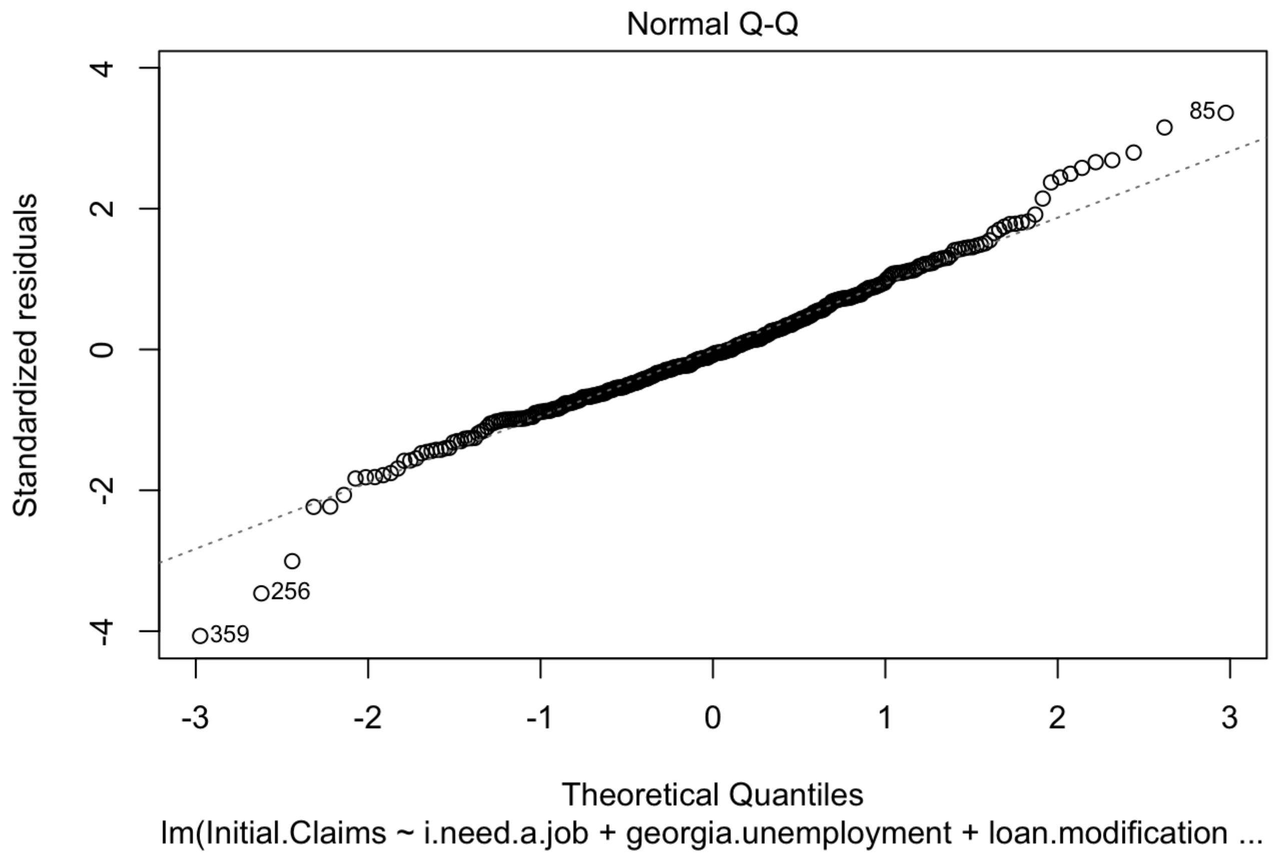
```
ols.model2 <- lm(Initial.Claims ~ i.need.a.job + georgia.unemployment + loan.modificatio
ns + craigslist.helper, data = myData, subset = train)
summary(ols.model2)
```

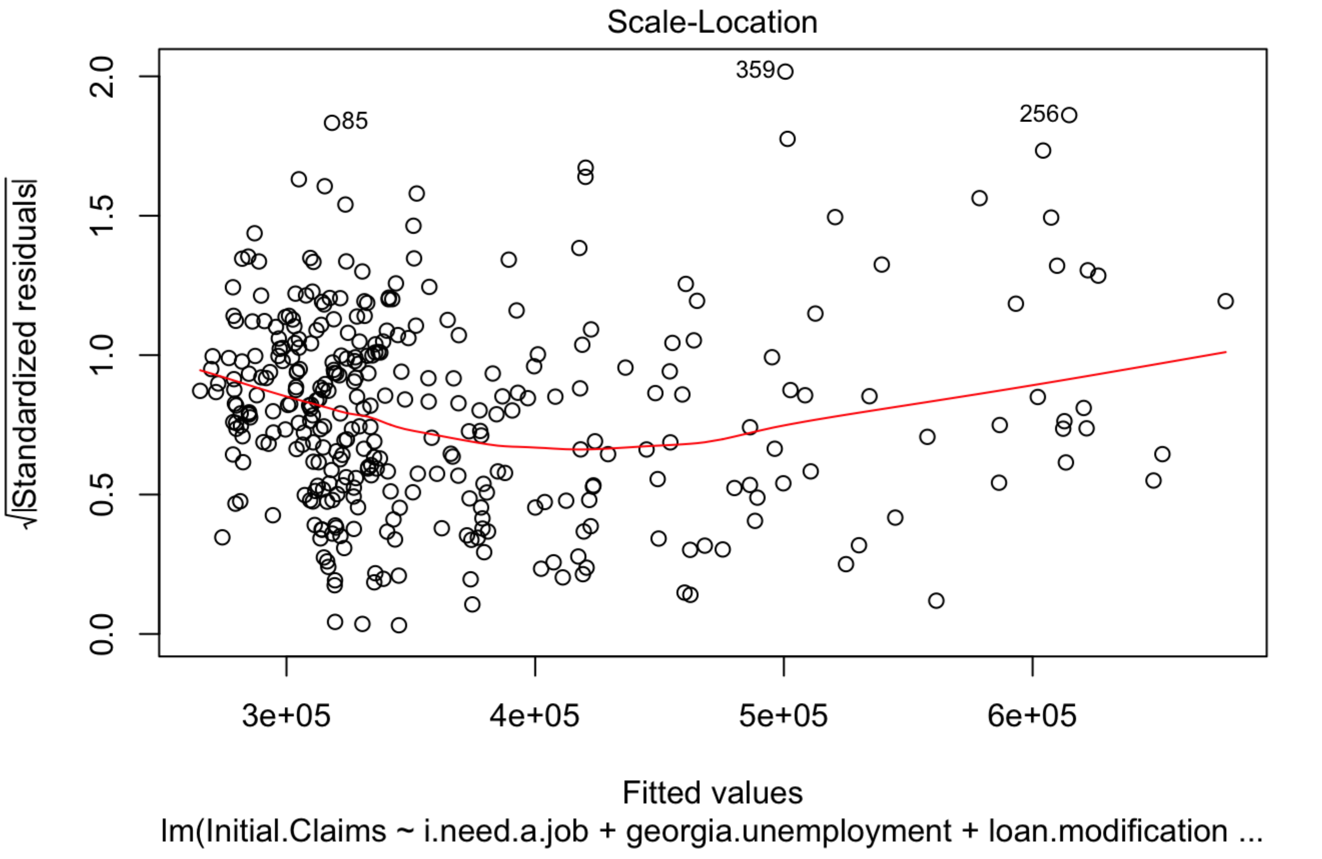
```
##
## Call:
## lm(formula = Initial.Claims ~ i.need.a.job + georgia.unemployment +
##     loan.modifications + craigslist.helper, data = myData, subset = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -81843 -13141  -1551  12861  69184
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            362798       1121 323.564  < 2e-16 ***
## i.need.a.job            27312       2458  11.113  < 2e-16 ***
## georgia.unemployment    15744       2718   5.793 1.59e-08 ***
## loan.modifications      23165       3844   6.027 4.39e-09 ***
## craigslist.helper       21911       4222   5.189 3.66e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20660 on 336 degrees of freedom
## Multiple R-squared:  0.9473, Adjusted R-squared:  0.9467
## F-statistic:  1511 on 4 and 336 DF,  p-value: < 2.2e-16
```

With just 4 search tags "i need a job", "georgia unemployment", "craigslist helper" and "loan modification" we get an adjusted R-squared of 0.9467 and also have a lot more confidence in our selected search tags. Lets see use this as our model. We will first view some residual plots to verify our model fit.

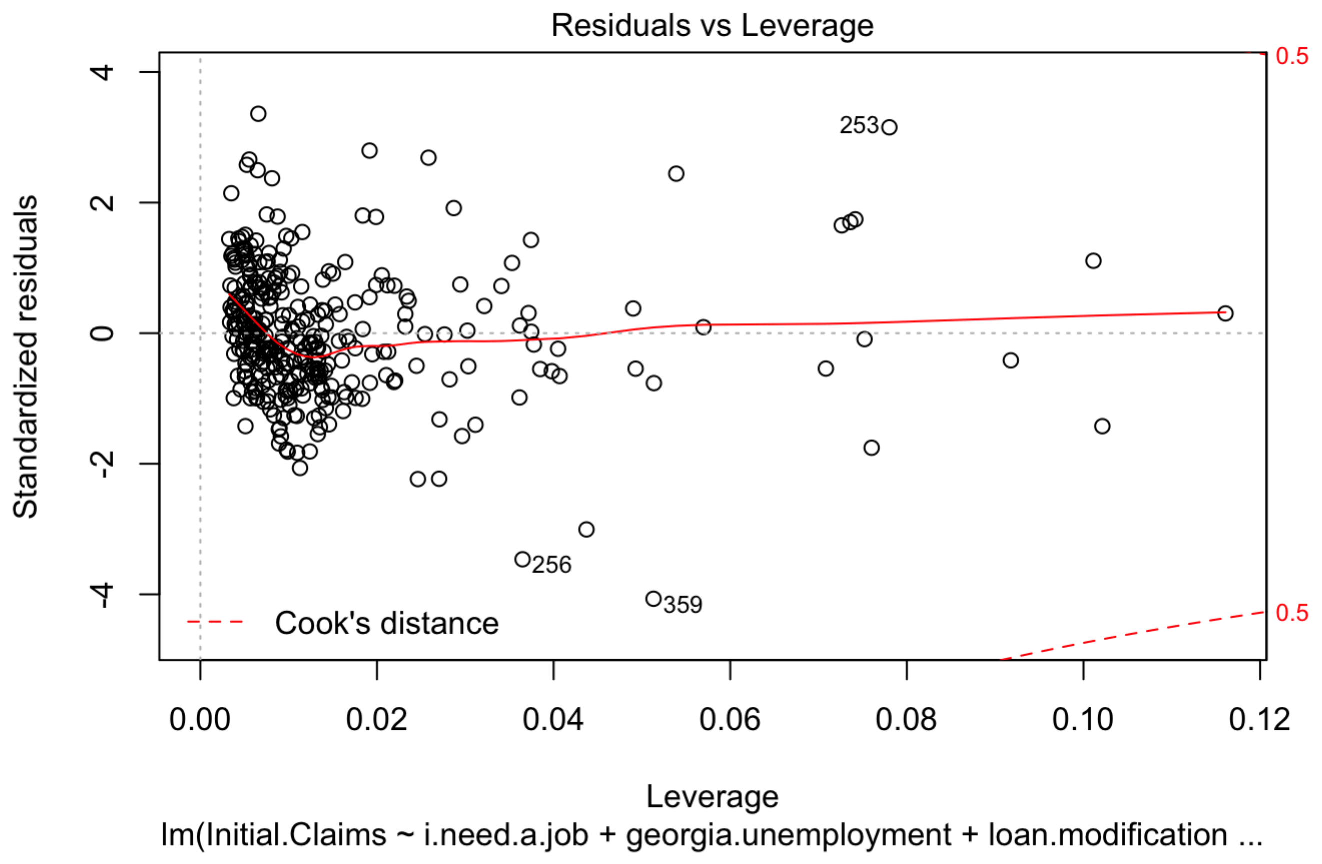
```
plot(ols.model2)
```


Residuals vs Fitted
Residuals
5e+04
0e+00
-5e+04
-1e+05
3e+05
4e+05
5e+05
6e+05
Fitted values
lm(Initial.Claims ~ i.need.a.job + georgia.unemployment + loan.modification ...
85
359
256


Normal Q-Q
Standardized residuals
4
2
0
-2
-4
-3
-2
-1
0
1
2
3
Theoretical Quantiles
lm(Initial.Claims ~ i.need.a.job + georgia.unemployment + loan.modification ...
85
256
359



Scale-Location
359
85
256
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
3e+05
4e+05
5e+05
6e+05
Fitted values
lm(Initial.Claims ~ i.need.a.job + georgia.unemployment + loan.modification ...



Residuals vs Leverage
253
256
359
0.5
0.5
Cook's distance
Standardized residuals
-4
-2
0
2
4
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Leverage
lm(Initial.Claims ~ i.need.a.job + georgia.unemployment + loan.modification ...

Our residuals seem to satify the normality assumptions well. At this point we are fairly confident of our in-sample fit. Lets look at our out-sample fit, in particular we will use RMSE (root mean squared error) as a preferred metric here.

```
# RMSE: training set
sqrt(mean((ols.model2$residuals)^2))
```

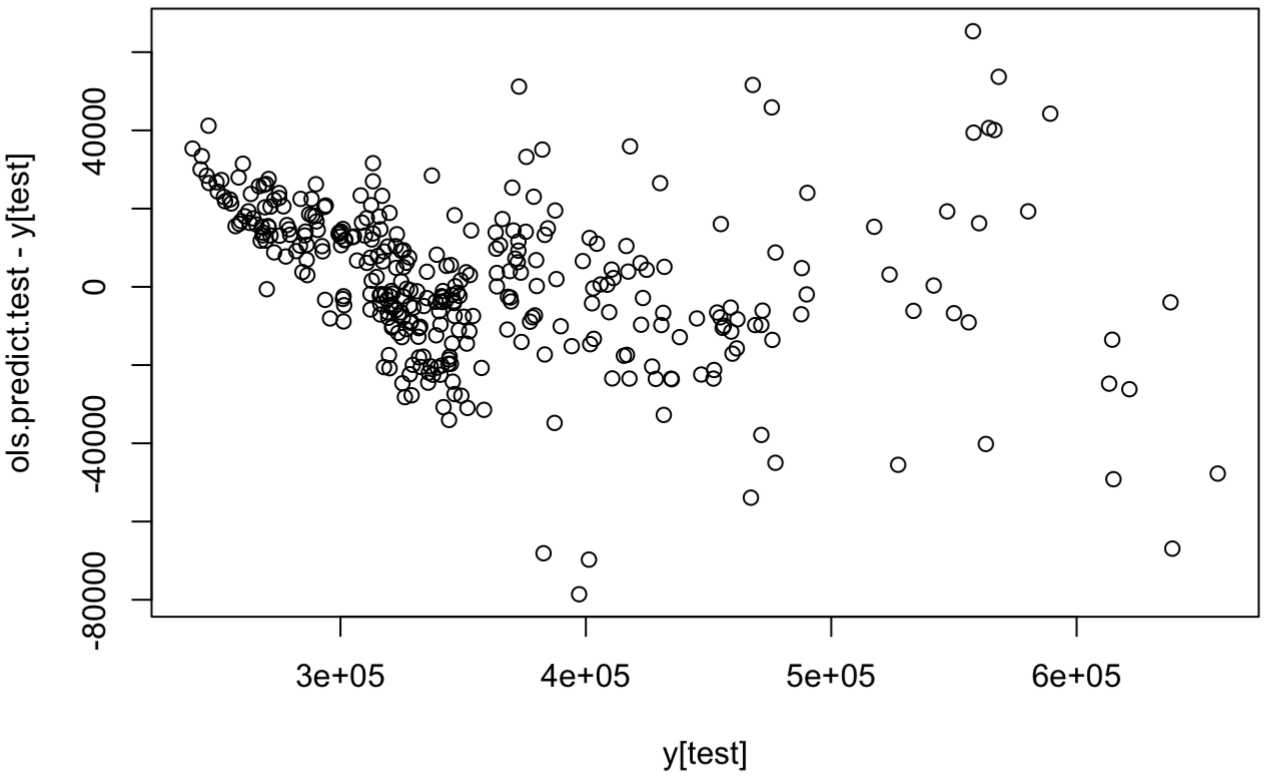
```
## [1] 20503.73
```

```
# RMSE: test set
ols.predict.test <- predict(ols.model2, newdata = data.frame(x[test,]))
sqrt(mean((y[test]-ols.predict.test)^2))
```

```
## [1] 20058.5
```

Our In-Sample and Out-Sample RMSE are very close, both around 20k. Given that only a random sample of 50% of all the data was used for developing the model and that it genealizes very well for the unobserved 50% of the data. We conclude this is our proposed model. Lastly lets see how our out-sample residuals look and if there are any signs of a bias in our prediction power. We plot true values against out-sample residuals to check for bias.

```
plot(y[test],ols.predict.test-y[test])
```



The residuals show no signs of bias, predicting true Initials Claims fairly well for low as well as high values. Our model can therefore be useful in predicting turning points as well.

One key advantage of our model is that it is state independent i.e. does not use traditional time series framework of lag orders, instead sees the problem as a purely predictive learning model and our sampling framework verifies the robustness of our model to out-of-sample data. This model is particularly useful in developed countries like the United States where almost everyone has access to the internet and every answer is just a google search away. In developing countries this model would not be that accurate.

Question 2

Smart Beta Strategy

Smart-beta strategies revolve around ideas of smarter asset allocation to beat a chosen benchmark. Most commonly used approaches involve the efficient market hypothesis and CAPM assumptions. Mean-Variance optimizations are used with the motivation of minimizing risk and maximing returns or equivalently maximising sharpe ratio. These Mean-Variance optimizations make assumptions of estimating risk/returns from purely historical price data. It is this assumption of Mean-Variance analysis that makes their effectiveness in real life relatively weak. These portfolios are still able to show outperformance to market capitalization or equally weighted indexes as various studies have show.

In this analysis, I will move away from the common ideas of Mean-Variance analysis. I aim to use come up with a pure alpha model i.e. we only care about maximising returns. The motivation revolves around the idea of value investing. Most common approaches to smart beta strategies using fundamental data involve ranking stocks based on certain factors and selecting stocks that fall within the top quantiles based on one or more factor values, trying to capture relative value between different stocks and sectors. I wanted to come up with a more general stock specific model. In particular I wanted to see how much of an impact does quarterly earnings data release have on the next quarter performance of the stock price and how this can be used in coming up wiht a smart beta strategy.

Data: Commonly used Value, Growth and Efficiency Ratios. I used this research paper “CSFB Quantitative Research, Alpha Factor Framework by P. N. Patel, S. Yao, R. Carlson, A. Banerji, J. Handelman” to select factors driving growth, value and profitability. Further we used Intrinio Api to pull historical fundamental data for a stock.

Problem: We attempt to solve a classification problem that tries to predict outperformance of stock price (or equivalently positive returns) in the next 3 months following the quarterly fundamental data release. The general idea is that the better our prediction accuracy is on a certain stock the more confident we are of its outperformance. This further can be used for stock selection or even rebalancing weights quarterly in a mean-variance analysis framework.

```r
library(httr)

## function to pull historical fundamental stock data for a specified data range and fre
quency
history <- function(ticker, item, start_date, end_date, frequency){
  history_base <- "https://api.intrinio.com/historical_data?ticker="
  username <- "541af5920e9416769122ddaefc2f8ec8"
  password <- "10a220e8fb9b9ca41f8ec088b7762bdd"

  historical <- paste(history_base, ticker, "&item=", item, "&start_date=", start_date,
"&end_date=", end_date, "&frequency=", frequency,sep="")
  tp <- GET(historical, authenticate(username, password, type = "basic"))
  z <- unlist(content(tp,"parsed"))

  n=length(z)
  if((n-5)<=0) {
    cat("n is",n,"\n")
    return("n becomes -ve")
  }
  b=as.data.frame(matrix(z[1:(n-5)],(n-5)/2, byrow = T))
  names(b)=names(z)[1:2]
  return(b)
}
```

We don’t claim that there is a unique model predicting outperformance for all the stocks. In the interest of time, we will just try to model the outperformance of 1 stock, AAPL and we will also provide some ideas on how this analysis can be used for creating customized pure alpha indexes, as we believe that a full fledged Machine learning based index would require a more involved backtesting and a lot more data.

```r
## chosen factors driving Value, Growth and Profitability
factorList <- list(price=c("close_price","weightedavedilutedsharesos"),
growth=c("ocfqoqgrowth","revenueqoqgrowth",
"netincomeqoqgrowth","epsqoqgrowth","ebitdaqoqgrowth","ebitda",
"bookvaluepershare"),
profitability=c("nopatqoqgrowth","ebitdamargin",
"nopatmargin","operatingmargin","opextorevenue","sgaextorevenue",
"arturnover","invturnover"))

## function to get chosen fundamental factors for a single stock
getHistoricFactors <- function(ticker, factors, start_date, end_date, frequency){
  result <- list()
  for (name in names(factors)) {
    cat("Getting factors for: ",name,"\n")
    for (factor in factors[[name]]){
      historicalFactorData <- history(ticker,factor,start_date,end_date,frequency)
      result[[factor]] <- historicalFactorData
    }
  }
  return(result)
}
```

Data: Quarterly fundamental data for AAPL from 2010-01-01 to 2018-01-01 (32 data points) including following fundamental factors as,

Growth Factors : “ocfqoqgrowth”,“revenueqoqgrowth”, “netincomeqoqgrowth”,“epsqoqgrowth”,“ebitdaqoqgrowth”,“pricetoearnings”,“pricetobook”

Profitability Factors: “nopatqoqgrowth”,“ebitdamargin”, “nopatmargin”,“operatingmargin”,“opextorevenue”,“sgaextorevenue”, “arturnover”,“invturnover”

We load AAPL data already saved as a csv and also do some data cleaning next.

```
##
## 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
```

We start with splitting the data into train and test dataset, we use an 80%-20% split

```
# training and test data
set.seed(124)
nRows <- nrow(AAPLModifiedData)
smp_size <- floor(0.8 * nRows)
train <- sample(seq_len(nRows), size = smp_size)
test <- -train
```

We try out multiple linear regression model to forecast the returns first.

```
## multiple linear regression
AAPLData1 <- select(AAPLModifiedData,-close_price.data.date,-close_price.data.value,
            -weightedavedilutedsharesos.data.value,-bookvaluepershare.data.value,-ebitd
a.data.value)


regressionModel <- lm(quarterlyReturn ~ ., data = AAPLData1[train,])
summary(regressionModel)
```

```
## 
## Call:
## lm(formula = quarterlyReturn ~ ., data = AAPLData1[train, ])
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -22.392  -4.197   1.817   5.614  30.029 
## 
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)  
## (Intercept)                    623.5631   846.1860   0.737   0.4822  
## ocfqoqgrowth.data.value       -240.8056   175.5727  -1.372   0.2074  
## revenueqoqgrowth.data.value     37.0993   438.3835   0.085   0.9346  
## netincomeqoqgrowth.data.value -427.9222  3145.9943  -0.136   0.8952  
## epsqoqgrowth.data.value         -4.9299     6.2436  -0.790   0.4525  
## ebitdaqoqgrowth.data.value     432.7528  1056.0939   0.410   0.6927  
## nopatqoqgrowth.data.value      197.6588  2413.5019   0.082   0.9367  
## ebitdamargin.data.value      -3863.0916  3631.6826  -1.064   0.3185  
## nopatmargin.data.value       -3046.0922  3037.0309  -1.003   0.3452  
## operatingmargin.data.value    5445.7392  4622.7630   1.178   0.2726  
## opextorevenue.data.value      3002.7554  1626.3375   1.846   0.1020  
## sgaextorevenue.data.value    -7309.2457  5531.8274  -1.321   0.2229  
## arturnover.data.value           -5.3074     2.3166  -2.291   0.0512 .
## invturnover.data.value           0.7314     1.0317   0.709   0.4985  
## priceToEarnings                  5.5738     5.5153   1.011   0.3418  
## priceToBook                    -13.7086    11.6689  -1.175   0.2739  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 20.06 on 8 degrees of freedom
## Multiple R-squared:  0.7366, Adjusted R-squared:  0.2426 
## F-statistic: 1.491 on 15 and 8 DF,  p-value: 0.2901
```

The regression model has an adjuted R squared of 0.2426 and also identified accounts receivables turnover being significant at 90% confidence level.

The fit doesn’t seem statistically significant. Given our small dataset of just 32 data points this is not surprising. For us to come up with a regression model we’ll surely need a lot more data. For our dataset where data frequency is quarterly this is clearly not a good approach.

Let’s rephrase the problem to a classification problem instead, to classify returns being positive (denoted by 1’s) or not.

We start with logistic regression model for classification. We also decide to use a 60-40% split as our dataset is quite small and such a split would ensure we have enough data points in the test set to see how well our model generalizes to unseen data.

```
# creating varible for classification problem
AAPLData2<- cbind(AAPLData1,
isReturnPositive = sapply(AAPLData1$quarterlyReturn, function(x)
  {if (x>0) return(1) else return(0)}))

AAPLData2 <- select(AAPLData2,-quarterlyReturn)

set.seed(125)
nRows <- nrow(AAPLModifiedData)
smp_size <- floor(0.6 * nRows)
train <- sample(seq_len(nRows), size = smp_size)
test <- -train

# use the same train and test spit as above
AAPLData2_train <- AAPLData2[train,]
AAPLData2_test <- AAPLData2[test,]
```

Training Logistic Regression classifier

```
logisticModel <- glm(isReturnPositive ~.,family=binomial(link='logit'),data=AAPLData2_tr
ain)
summary(logisticModel)
```

```
##
## Call:
## glm(formula = isReturnPositive ~ ., family = binomial(link = "logit"),
##     data = AAPLData2_train)
##
## Deviance Residuals:
##          7          29          24          23           6           2
##  9.160e-06   2.110e-08  -3.348e-06   1.798e-06   2.110e-08   4.653e-06
##         19          25          17           8          32          10
##  7.754e-06   7.870e-06  -6.509e-06  -6.975e-06   2.110e-08  -5.462e-06
##         26          31           5          14          13          11
##  2.110e-08   7.932e-06   4.083e-06   8.466e-06   3.032e-06  -7.557e-06
##
## Coefficients:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    3.181e+03  7.309e+08       0        1
## ocfqoqgrowth.data.value       -1.715e+02  8.349e+07       0        1
## revenueqoqgrowth.data.value    3.704e+02  4.223e+08       0        1
## netincomeqoqgrowth.data.value -7.675e+03  4.231e+09       0        1
## epsqoqgrowth.data.value       -5.696e+02  7.662e+08       0        1
## ebitdaqoqgrowth.data.value    -6.902e+02  2.505e+08       0        1
## nopatqoqgrowth.data.value      8.258e+03  4.342e+09       0        1
## ebitdamargin.data.value       -1.712e+04  4.443e+09       0        1
## nopatmargin.data.value        -9.636e+03  6.159e+09       0        1
## operatingmargin.data.value     1.962e+04  8.444e+09       0        1
## opextorevenue.data.value       8.499e+03  4.779e+09       0        1
## sgaextorevenue.data.value     -2.726e+04  1.205e+10       0        1
## arturnover.data.value         -5.484e+00  9.726e+05       0        1
## invturnover.data.value         1.457e+00  6.610e+05       0        1
## priceToEarnings               -4.409e+00  3.407e+06       0        1
## priceToBook                    3.213e+00  9.065e+06       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.1270e+01  on 17  degrees of freedom
## Residual deviance: 5.8051e-10  on  2  degrees of freedom
## AIC: 32
##
## Number of Fisher Scoring iterations: 24
```

```
# Analyzing training set classification accuracy

# calculate fitted probabilities
train_pred_prob <- predict(logisticModel, newdata = AAPLData2_train, type='response')

# classify returns as postive or not
train_pred <- ifelse(train_pred_prob > 0.5,1,0)

# training set accuracy
logisticModel_train_accuracy <- 100*mean(train_pred == AAPLData2_train$isReturnPositive)

logisticModel_train_accuracy
```

```
## [1] 100
```

```
# Analyzing test set classification accuracy

# calculate fitted probabilities
test_pred_prob <- predict(logisticModel, newdata = AAPLData2_test, type='response')

# classify returns as postive or not
test_pred <- ifelse(test_pred_prob > 0.5,TRUE,FALSE)

# test set accuracy
logisticModel_test_accuracy <- 100*mean(test_pred == AAPLData2_test$isReturnPositive)

logisticModel_test_accuracy
```

```
## [1] 61.53846
```

The logisitic regression model above clearly overfits the data, with training set accuracy being 100% while test set accuracy being just 61%. This is not surprising as we have only 32 data points and 12 features.

We try Support Vector Machine algorithm next. We will only use Linear Kernel as any complex kernel will make the overfitting problem more likely to reoccur.

```
# Setting up SVM Model
AAPLData3<- cbind(AAPLData1,
isReturnPositive = sapply(AAPLData1$quarterlyReturn, function(x) as.factor(x>0)))

AAPLData3 <- select(AAPLData3,-quarterlyReturn)

AAPLData3_train <- AAPLData3[train,]
AAPLData3_test <- AAPLData3[test,]
```

Training SVM classifier with Linear Kernel

```
# setting trainControl() which species resampling method as repeated cross validation,
# sets no of folds of cross validation as 10
# the cross validation is repeated 3 times
library('caret')
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:httr':
##
##     progress
```

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

svmLinear <- train(isReturnPositive ~., data = AAPLData3_train, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)
svmLinear
```

```
## Support Vector Machines with Linear Kernel
##
## 18 samples
## 15 predictors
##  2 classes: 'TRUE', 'FALSE'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 17, 16, 17, 16, 17, 17, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5555556  -0.1238095
##
## Tuning parameter 'C' was held constant at a value of 1
```

Analysing training set accuracy with SVM

```
train_pred <- predict(svmLinear, newdata = AAPLData3_train)

confusionMatrix(train_pred, AAPLData3_train$isReturnPositive)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction TRUE FALSE
##      TRUE    11     1
##      FALSE    2     4
##                                           
##                Accuracy : 0.8333          
##                  95% CI : (0.5858, 0.9642)
##     No Information Rate : 0.7222          
##     P-Value [Acc > NIR] : 0.22            
##                                           
##                   Kappa : 0.6087          
##  Mcnemar's Test P-Value : 1.00            
##                                           
##             Sensitivity : 0.8462          
##             Specificity : 0.8000          
##          Pos Pred Value : 0.9167          
##          Neg Pred Value : 0.6667          
##              Prevalence : 0.7222          
##          Detection Rate : 0.6111          
##    Detection Prevalence : 0.6667          
##       Balanced Accuracy : 0.8231          
##                                           
##        'Positive' Class : TRUE            
## 
```

```
confusionMatrix
```

```
## function (data, ...) 
## {
##     UseMethod("confusionMatrix")
## }
## <environment: namespace:caret>
```

Analysing test set accuracy with SVM

```
# test set accuracy
test_pred <- predict(svmLinear, newdata = AAPLData3_test)

confusionMatrix(test_pred, AAPLData3_test$isReturnPositive)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction TRUE FALSE
##      TRUE     7     3
##      FALSE    1     2
##                                           
##                Accuracy : 0.6923          
##                  95% CI : (0.3857, 0.9091)
##     No Information Rate : 0.6154          
##     P-Value [Acc > NIR] : 0.3966          
##                                           
##                   Kappa : 0.2973          
##  Mcnemar's Test P-Value : 0.6171          
##                                           
##             Sensitivity : 0.8750          
##             Specificity : 0.4000          
##          Pos Pred Value : 0.7000          
##          Neg Pred Value : 0.6667          
##              Prevalence : 0.6154          
##          Detection Rate : 0.5385          
##    Detection Prevalence : 0.7692          
##       Balanced Accuracy : 0.6375          
##                                           
##        'Positive' Class : TRUE            
## 
```

```
confusionMatrix
```

```
## function (data, ...) 
## {
##     UseMethod("confusionMatrix")
## }
## <environment: namespace:caret>
```

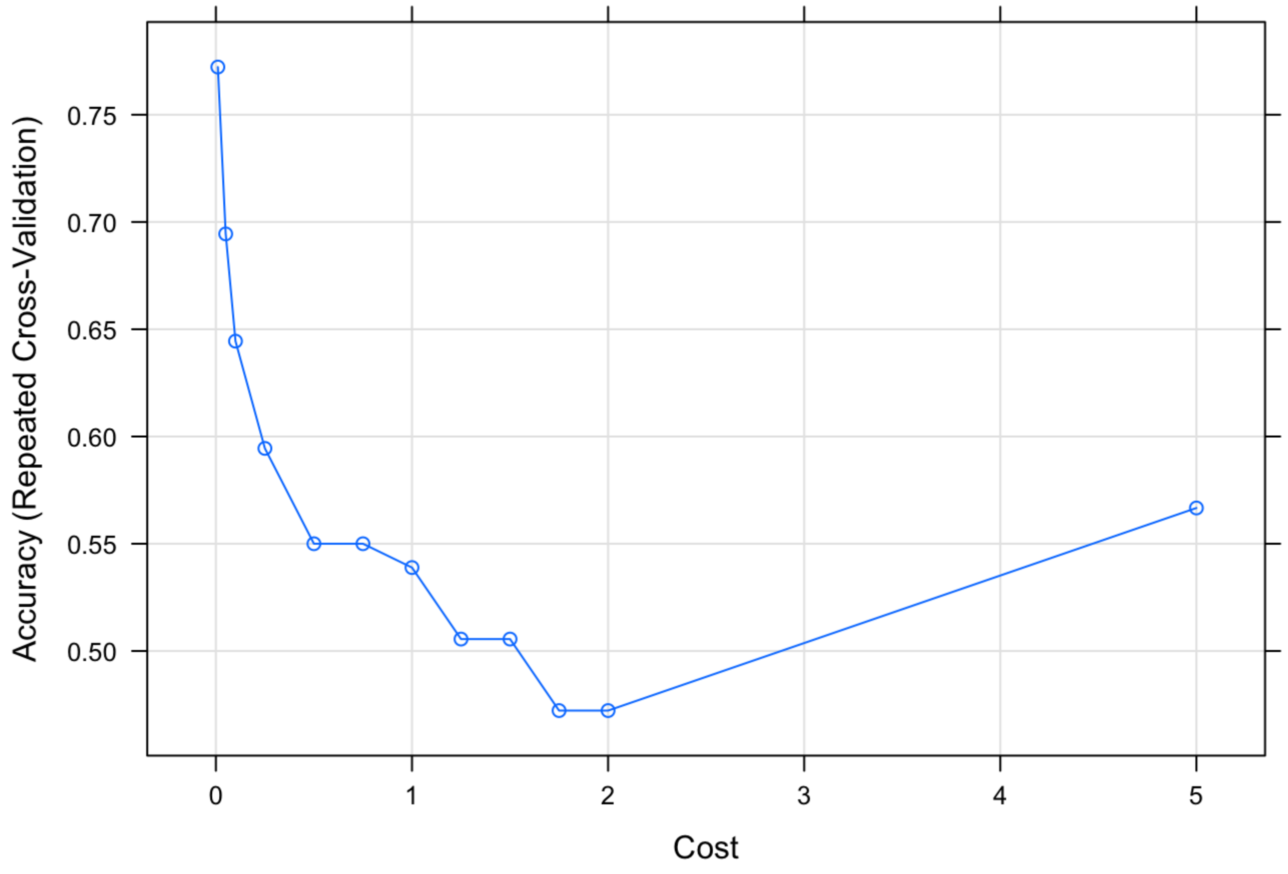
The training set accuracy is 83% while the test set accuracy is 69%. This model generalizes much better and therefore SVM with linear kernal is our preferrend model.

We next try to tune parameter C which is a regularization parameter, with lower values of C likely reducing overfitting but also making the model prone to underfitting. In the previous model C was set to 1. We use the grid search method to find the value of C that maximises our cross validation accuracy.

```
# Tuning parameter C
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))
svmLinearGrid <- train(isReturnPositive ~., data = AAPLData3_train, method = "svmLinear"
,
                           trControl=trctrl,
                           preProcess = c("center", "scale"),
                           tuneGrid = grid,
                           tuneLength = 10)
svmLinearGrid
```

```
## Support Vector Machines with Linear Kernel
##
## 18 samples
## 15 predictors
##  2 classes: 'TRUE', 'FALSE'
##
## Pre-processing: centered (15), scaled (15)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 15, 16, 16, 16, 16, 17, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00        NaN          NaN
##   0.01  0.7722222   0.00000000
##   0.05  0.6944444  -0.02777778
##   0.10  0.6444444  -0.02631579
##   0.25  0.5944444  -0.07500000
##   0.50  0.5500000  -0.09523810
##   0.75  0.5500000  -0.09523810
##   1.00  0.5388889  -0.10952381
##   1.25  0.5055556  -0.10454545
##   1.50  0.5055556  -0.10454545
##   1.75  0.4722222  -0.19545455
##   2.00  0.4722222  -0.19545455
##   5.00  0.5666667  -0.12000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```
plot(svmLinearGrid)
```



Lets see if tuned the paramter C value of 0.01 has increased our test accuracy or not.

```
test_pred_grid <- predict(svmLinearGrid, newdata = AAPLData3_test)
confusionMatrix(test_pred_grid, AAPLData3_test$isReturnPositive)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction TRUE FALSE
##      TRUE     8     5
##      FALSE    0     0
##
##                Accuracy : 0.6154
##                  95% CI : (0.3158, 0.8614)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : 0.61942
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6154
##          Neg Pred Value :    NaN
##              Prevalence : 0.6154
##          Detection Rate : 0.6154
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : TRUE
##
```

Our test set accuracy had infact reduced to around 61.5% from 69%. Indicating that this value of C might have infact led to underfitting of the model. Lets look at our training set accuracy to check if this is the case.

```
train_pred_grid <- predict(svmLinearGrid, newdata = AAPLData3_train)
confusionMatrix(train_pred_grid, AAPLData3_train$isReturnPositive)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction TRUE FALSE
##      TRUE    13     5
##      FALSE    0     0
##
##                Accuracy : 0.7222
##                  95% CI : (0.4652, 0.9031)
##     No Information Rate : 0.7222
##     P-Value [Acc > NIR] : 0.61751
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7222
##          Neg Pred Value :    NaN
##              Prevalence : 0.7222
##          Detection Rate : 0.7222
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : TRUE
##
```

Our training set accuracy reduced from 83% to 72%, making the case for underfitting for this tuned value of C. We therefore conclude that the SVM model with linear kernel and C = 1 is the best classification model.

We did this analysis for just one stock and showed the significance of growth and value fundamental factors in predicting quarterly returns. We next define a smart beta strategy based on our analysis. The intuition is based on changing the above mentioned classification problem from predicting absolute stock outperformance to predicting the relative stock outperformance to an index.

Predicting outperforming stocks in an index in the following quarter of the fundamental data release will involve collecting the quarterly fundamental data for the factors we have used in our analysis for all the constituent stocks of the index and then performing the following steps,

1. Calculating average factor values across all the index components
2. Adding certain factors reflecting relative value of these fundamental factors of a stock as compared to the index, we propose calculating the ratio (stock specific factor value/average index factor value) for all the above factors
3. Changing the classification label to indicate relative outperformance to index, 1 if stock will outperform the index, 0 if stock will underperform the index.

Our hope is that SVM with a linear kernel should work for this classification problem too.

Unfortunately we don’t have any returns, risk or sharpe ratio metrics to report our performance, though we sucessfully suggest a novel smart beta model that attempts to take a purely value based fundamental approach to quarterly investing.

Question 3

Predicting FX Markets

We propose modelling the relationship between a basket of currencies correlated with oil prices namely AUDUSD, NZDUSD, USDMXN and USDCAD, Oil prices and DXY. We use DXY as a factor in addition to the oil prices as we need a proxy for relative strength of the dollar as all these currency pairs are denominated in dollars. The idea was that using a multi-factor model with any one of them as a predictor, and regressing against the others could lead to a statistically significant model for prediction. In theory we could have modelled just a pair of these currency pairs against each other too for eg. from inital analysis I found a strong negative correlation of -0.87 between AUDUSD and USDCAD which could be modelled as a pairs trading strategy, but we believe that using a multi factor relative value model should lead to better fit as well as stronger signal for trading. If the regression residuals of the proposed model are stationary, we can further do relative value trading based on the mean reversion of the residuals.

Data: We use Quandl Api to get prices starting from 2017-01-01 for AUDUSD, NZDUSD, USDMXN, USDCAD and WTI Crude Oil prices from FRED database. For DXY we have used data from yahoo finance pulled using getSymbols method in library quantmod.

Proposed Model:

log(AUDUSD_t) ~ beta0 + beta1(log(NZDUSD_t)) + beta2(log(USDMXN_t)) + beta3(log(USDCAD_t)) + beta4*log(Oilprice_t) + beta5(DXY_t) + epsilon_t

Trading Model:

If we take the first order differencing of the above model we get,

epsilon_t - epsilon_t-1 = log(AUDUSD_t/AUDUSD_t-1) - {beta0 + beta1(log(NZDUSD_t/NZDUSD_t-1)) + beta2(log(USDMXN_t/USDMXN_t-1)) + beta3(log(USDCAD_t/USDCAD_t-1)) + beta4(log(Oilprice_t/Oilprice_t-1)) + beta5log((DXY_t/DXY_t-1))}, where x_t denotes x(t)

For simiplicity, lets call everything within {} as a 'basket'.

Each log term above just represents the return on each of the assets and the financial interpretation of the above model follows as: A portfolio that is long 1 dollar of AUDUSD and short the basket (where short the basket involves shorting beta1 dollar of NZDUSD, beta3 dollar of USDCAD, beta4 barrels of Oil and beta5 units of DXY) will have profit in dollars equal to the order 1 lag of our model error. So our trading signal will be as follows:

If epsilon_t > threshold, Short 1 dollar of AUDUSD and Long the basket If epsilon_t < -threshold, Long 1 dollar of AUDSUD and Short the basket

```
library(Quandl)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
library(quantmod)
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
# Getting historical price data from 2017-01-01 to present
Quandl.api_key("HD_sSceQtLjyiEa9LCJM")
AUDUSD <- Quandl("FRED/DEXUSAL", type="xts", start_date='2017-01-01')
NZDUSD <- Quandl("FRED/DEXUSNZ", type="xts", start_date='2017-01-01')
USDMXN <- Quandl("FRED/DEXMXUS", type="xts", start_date='2017-01-01')
USDCAD <- Quandl("FRED/DEXCAUS", type="xts", start_date='2017-01-01')
Oil <- Quandl("FRED/DCOILWTICO", type="xts", start_date='2017-01-01')
getSymbols("DXY",src="yahoo", startDate = '2017-01-01')
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## [1] "DXY"
```

Proppposed Model:

```r
# log normalization of the price data
AUDUSDFactors <- na.omit((merge(AUDUSD,NZDUSD,USDMXN,USDCAD,Oil,DXY$DXY.Close)))

AUDUSDFactorModel <- lm(AUDUSD ~ .,data=log(AUDUSDFactors))
summary(AUDUSDFactorModel)
```

```
##
## Call:
## lm(formula = AUDUSD ~ ., data = log(AUDUSDFactors))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.023918 -0.006070 -0.000022  0.006224  0.025427
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.336615   0.082719   4.069 6.11e-05 ***
## NZDUSD       0.358637   0.034114  10.513  < 2e-16 ***
## USDMXN      -0.119287   0.015802  -7.549 6.00e-13 ***
## USDCAD      -0.425693   0.028748 -14.808  < 2e-16 ***
## Oil          0.029077   0.007127   4.080 5.87e-05 ***
## DXY.Close   -0.018948   0.008349  -2.270    0.024 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009074 on 284 degrees of freedom
## Multiple R-squared:  0.8665, Adjusted R-squared:  0.8642
## F-statistic: 368.8 on 5 and 284 DF,  p-value: < 2.2e-16
```

The proposed model seems to fit the data well with an adjusted R squared of 0.8642 and all the chosen variables being significant in our model. NZDUSD, USDMXN, USDCAD and Oil at 99.9% confidence level and DXY at 95% confidence level.

We started with using Johanssen test which aims to find a linear combination of the factors that is cointegrated. It basically performs eigen value decomposition of the factor matrix and provides factor loadings that produce a stationary times series explaining the maximum possible variance in the data. Cointegration is a more reliable measure of mean reversion vs correlation in time series analysis and therefore try to see is these loadings from Johanssen Test can be used by us.

```r
library(urca)
JOTest=ca.jo(AUDUSDFactors, type="trace", K=2, ecdet="none", spec="longrun")
summary(JOTest)
```

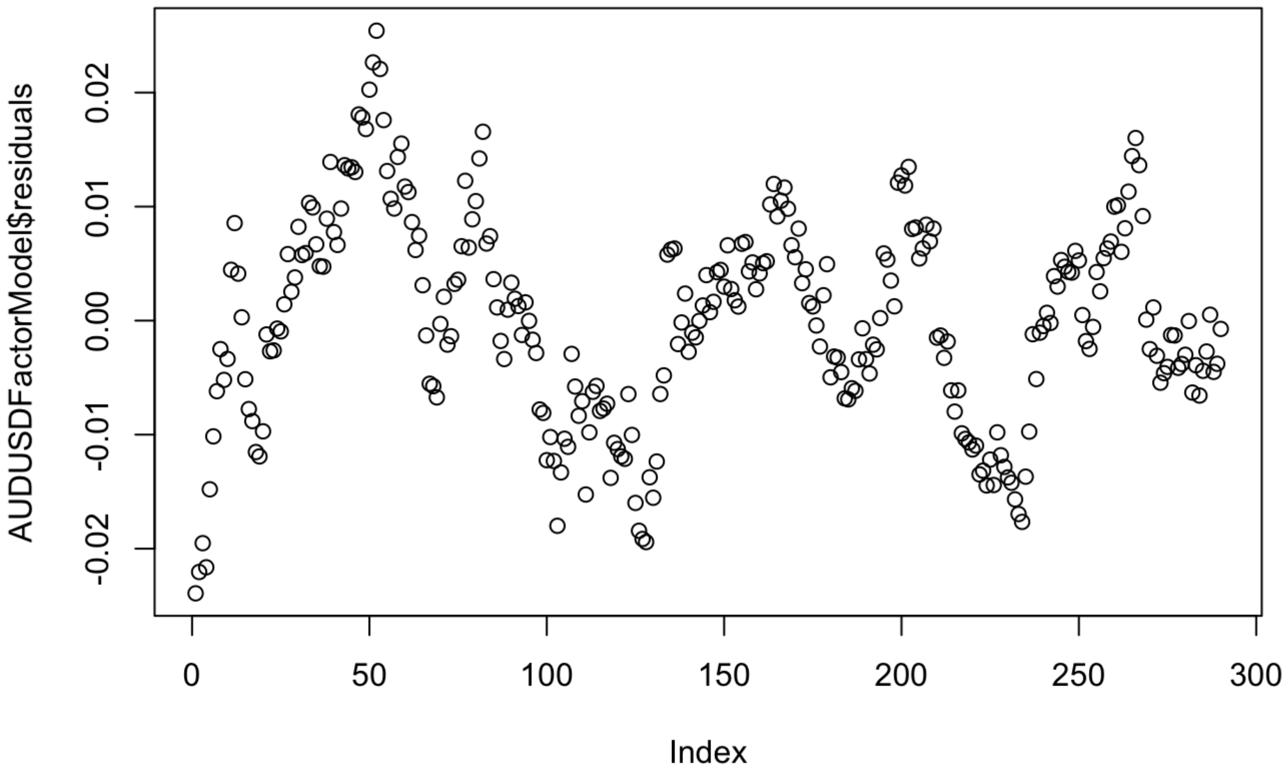
```
## 
## ###################### 
## # Johansen-Procedure # 
## ###################### 
## 
## Test type: trace statistic , with linear trend 
## 
## Eigenvalues (lambda):
## [1] 0.11519251 0.05976814 0.05116405 0.03557416 0.02564327 0.01087120
## 
## Values of teststatistic and critical values of test:
## 
##           test 10pct  5pct   1pct
## r <= 5 |  3.15  6.50  8.18  11.65
## r <= 4 | 10.63 15.66 17.95  23.52
## r <= 3 | 21.06 28.71 31.52  37.22
## r <= 2 | 36.19 45.23 48.28  55.43
## r <= 1 | 53.94 66.49 70.60  78.87
## r = 0  | 89.18 85.18 90.39 104.20
## 
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
## 
##                  AUDUSD.l2     NZDUSD.l2     USDMXN.l2     USDCAD.l2
## AUDUSD.l2     1.000000000  1.000000e+00  1.000000e+00  1.000000e+00
## NZDUSD.l2     0.509330871 -5.121161e-01 -1.130627e+00  2.028504e-01
## USDMXN.l2    -0.009217523  6.184218e-03  2.133823e-03  9.033932e-03
## USDCAD.l2     0.586076610  5.596375e-01  5.463892e-03  2.487770e-01
## Oil.l2        0.001564724  5.018172e-04 -1.032156e-03 -2.894685e-03
## DXY.Close.l2 -0.000169522  4.611516e-05  7.828182e-05  7.138113e-05
##                    Oil.l2 DXY.Close.l2
## AUDUSD.l2     1.000000e+00  1.000000000
## NZDUSD.l2    -4.113381e-01 -7.651042889
## USDMXN.l2     2.373675e-02  0.147593561
## USDCAD.l2    -9.882147e-02 -3.373092431
## Oil.l2       -9.071780e-04 -0.057262366
## DXY.Close.l2  5.055087e-05 -0.001747955
## 
## Weights W:
## (This is the loading matrix)
## 
##                AUDUSD.l2     NZDUSD.l2   USDMXN.l2     USDCAD.l2
## AUDUSD.d    -0.07691157  -0.003313495 -0.02561067  -0.002073592
## NZDUSD.d    -0.08070761   0.015866466  0.01856206  -0.011075025
## USDMXN.d     0.97761398  -1.318032705  0.72293608  -0.413392044
## USDCAD.d     0.08230992  -0.018002576 -0.02065694  -0.029304892
## Oil.d       -3.41381311  -3.957770376  6.33132531   1.501773560
## DXY.Close.d 59.44947743 126.890260327 19.52228027 -67.777745025
##                   Oil.l2  DXY.Close.l2
## AUDUSD.d     7.004382e-04  0.0002224666
## NZDUSD.d     3.707444e-03 -0.0001832372
## USDMXN.d    -5.594831e-01 -0.0072042206
## USDCAD.d     1.661488e-02 -0.0004143618
## Oil.d        3.808977e+00  0.1329896003
## DXY.Close.d -4.888596e+01  3.5606634037
```

The test statistic values for Johanssen test imply we cannot reject the r<=1 hypothesis at any confidence level, not providing strong support to our analysis. We try the Augmented Dickey Fuller test next.

```
library('tseries')
adf.test(AUDUSDFactorModel$residuals)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  AUDUSDFactorModel$residuals
## Dickey-Fuller = -3.5796, Lag order = 6, p-value = 0.03547
## alternative hypothesis: stationary
```

```
plot(AUDUSDFactorModel$residuals)
```



ADF test statistic is a large negative value implying that the regression residuals of our model is stationary. We trading strategy specifications are described below:

1. Long the spread: Goes Long 1 dollar worth of AUDUSD and shorts the basket described in the initial model description with the betas now being simply our regression coefficents when spread less than -1.5 standard deviations from the mean (which is 0)

2. Short the spread: Goes Short 1 dollar worth of AUDUSD and shorts the basket described in the initial model description with the betas now being simply our regression coefficents when spread greater than 1.5 standard deviations from the mean (which is 0)
3. We also implement stop losses at levels 2.5 standard deviations on either side. So if we are long the spread and spread becomes less than -2.5 standard deviations or moves further away from the mean rather than mean reverting we close the trade to prevent further losses. This happens when are are short the spread and spread becomes greater than 2.5 standard deviations.
4. We use a training period of 60 days to estimate our regression model and use the estimated parameters for trading in the next 30 days. We use a rolling window method for the same. In our data starting from 2017-01-01 we had 7 such test periods which will help us evaluate the performance of our trading strategy.
5. At any points in time there can only be 1 active trade and any open position is closed at the end of the test period window.

```
# function that implements the trading strategy
tradeSignal <- function(data, coeff, spread){

  N <- nrow(testData)
  sdev <-sd(spread)
  zeroprices <- rep(0,ncol(data))
  factorCoeff <- coeff[2:length(coeff)]
  prices <- list()
  exposures <- list()
  costOfTrade <- vector(mode="numeric", length=0)
  returns <- list()
  spreadPosition <- 'no'
  countTrades <- 0

  for (i in 1:N) {

    basespreadprices <- c(-1,1,1,1,1,1)
    if(spreadPosition == 'no') {

      if(spread[i] > 1.5*sdev){
        # go short the spread
        cat("Going short the spread at index level", i, " and spread level ", spread[i],
 "\n")
        countTrades <- countTrades + 1
        prices[[countTrades]] <- data[i,]
        exposures[[countTrades]] <- c(1,-factorCoeff) * c(1/coredata(data[i,'AUDUSD']),1
/coredata(data[i,'NZDUSD']),1,1,1,1)
        costOfTrade[countTrades] <- abs(sum(c(1,-factorCoeff)*c(1,1,1,1,coredata(data[i,
'Oil']),coredata(data[i,'DXY.Close']))))
        spreadPosition <- 'short'
      }
      else if(spread[i] < -1.5*sdev){
        # go long the spread
        cat("Going long the spread at index level", i, " and spread level ", spread[i],
"\n")
        countTrades <- countTrades + 1
        prices[[countTrades]] <- data[i,]
        exposures[[countTrades]] <- c(1,-factorCoeff) * c(1/coredata(data[i,'AUDUSD']),1
/coredata(data[i,'NZDUSD']),1,1,1,1)
        costOfTrade[countTrades] <- abs(sum(c(1,-factorCoeff)*c(1,1,1,1,coredata(data[i,
'Oil']),coredata(data[i,'DXY.Close']))))
        spreadPosition <- 'long'
      }

      next
    }

    if(spreadPosition == 'short') {

      if(spread[i] <= 0.1*sdev){
        # close the spreadPosition
        cat("Closing long spreadPosition at index level", i, " and spread level ", sprea
d[i], "\n")
```

```
        returns[[countTrades]] <- exposures[[countTrades]]*(coredata(prices[[countTrade
s]]) - coredata(data[i,]))*c(1,1,1/coredata(data[i,'USDMXN']),1/coredata(data[i,'USDCAD'
]),1,1)
        spreadPosition <- 'no'
      }
      else if(spread[i] > 2.5*sdev){
        # stop loss
        cat("Stop Loss Trigger: closing long spreadPosition at index level", i, " and sp
read level ", spread[i], "\n")
        returns[[countTrades]] <- exposures[[countTrades]]*(coredata(prices[[countTrade
s]]) - coredata(data[i,]))*c(1,1,1/coredata(data[i,'USDMXN']),1/coredata(data[i,'USDCAD'
]),1,1)
        spreadPosition <- 'no'
      }

      next
    }


    if(spreadPosition == 'long') {

      if(spread[i] >= -0.1*sdev){
        # close the spreadPosition
        cat("Closing long spreadPosition at index level", i, " and spread level ", sprea
d[i], "\n")
        returns[[countTrades]] <- exposures[[countTrades]]*(coredata(data[i,]) - coredat
a(prices[[countTrades]]))*c(1,1,1/coredata(data[i,'USDMXN']),1/coredata(data[i,'USDCAD'
]),1,1)
        spreadPosition <- 'no'

      }
      else if(spread[i] < -2.5*sdev){
        # stop loss
        cat("Stop Loss Trigger: closing long spreadPosition at index level", i, " and sp
read level ", spread[i], "\n")
        returns[[countTrades]] <- exposures[[countTrades]]*(coredata(data[i,]) - coredat
a(prices[[countTrades]]))*c(1,1,1/coredata(data[i,'USDMXN']),1/coredata(data[i,'USDCAD'
]),1,1)
        spreadPosition <- 'no'
      }

      next
    }
  }

  # close any open spreadprices on the last day of the test period
  if(spreadPosition != 'close') {
    cat("End of Trading Period: Closing open spreadPosition at index level", i, " and sp
read level ", spread[i], "\n")
    if(spreadPosition == "long") returns[[countTrades]] <- exposures[[countTrades]]*(cor
edata(data[N,]) - coredata(prices[[countTrades]]))*c(1,1,1/coredata(data[i,'USDMXN']),1/
coredata(data[i,'USDCAD']),1,1)
    else returns[[countTrades]] <- exposures[[countTrades]]*(coredata(prices[[countTrade
s]]) - coredata(data[N,]))*c(1,1,1/coredata(data[i,'USDMXN']),1/coredata(data[i,'USDCAD'
]),1,1)
  }

  allReturns <- do.call(rbind, returns)
  returnsTrades <- apply(allReturns,1,sum)/costOfTrade
  return(100*((prod(sapply(returnsTrades,function(x) (1+x)))) - 1))

}
```

Trading signals generated from the trading strategy as described above in the first 30 day test period are shown below. It is clear that we are able to bet on spread mean reversion correctly.

```
i <- 0
train <- 60
test <- 30
noOfSamples<-nrow(AUDUSDFactors)        #NO OF DATA POINTS
noOfPeriods<-floor((noOfSamples-train)/test)
trainData <- AUDUSDFactors[(1 + i*test) : (train + i*test),]
# log scaling is necessary to interpret spread reversion as returns
linearModel <- lm(AUDUSD ~ .,data = log(trainData))
testData <- AUDUSDFactors[(train + i*test + 1) : (train + test + test*i),]
meanRevertingspread <- apply(log(testData)*
                                 c(1,-linearModel$coefficients[2:length(linearModel$coef
ficients)]) - linearModel$coefficients[1],1,sum)
 returns <- tradeSignal(testData, linearModel$coefficients, meanRevertingspread)
```

```
## Going long the spread at index level 2  and spread level  -12.97147
## Closing long spreadPosition at index level 3  and spread level  0.4989728
## Going long the spread at index level 8  and spread level  -13.02373
## Closing long spreadPosition at index level 9  and spread level  0.5075611
## Going long the spread at index level 14  and spread level  -12.97899
## Closing long spreadPosition at index level 15  and spread level  0.4873737
## Going long the spread at index level 20  and spread level  -12.98421
## Closing long spreadPosition at index level 21  and spread level  0.4974719
## Going long the spread at index level 26  and spread level  -12.98911
## Closing long spreadPosition at index level 27  and spread level  0.5032364
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -4.269486
```

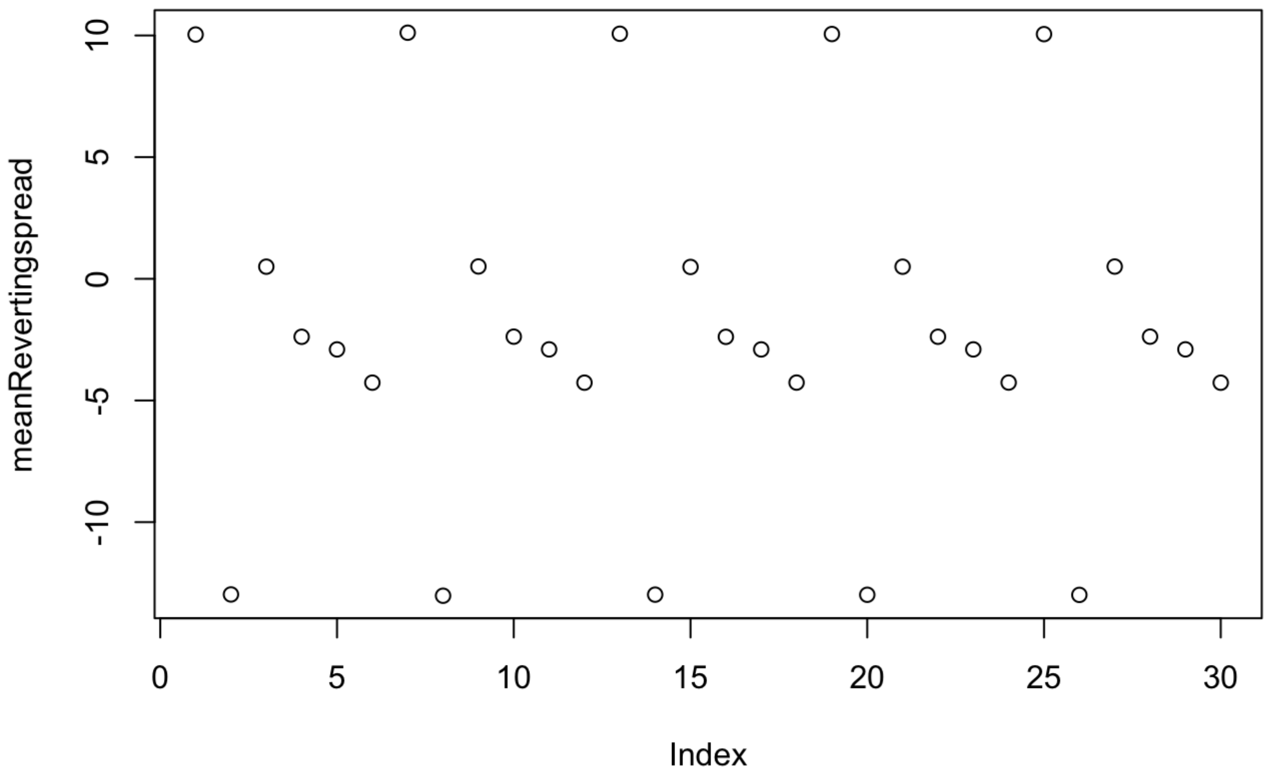
```
returns
```

```
## [1] 8.195649
```

We see that our first test period generates return of 8.19% validating the profitability of the trading strategy based on reversion.

When we plot our estimated residuals for this period based on the estimated regression coefficents from the past 60 days training data, we realize that our profitability is because of the stationarity of these residuals which is accurately captured in our trading signals above.

```
plot(meanRevertingspread)
```



Let report our model performance for all 7 test periods

```
rain <- 60
test <- 30
noOfSamples<-nrow(AUDUSDFactors)        #NO OF DATA POINTS
noOfPeriods<-floor((noOfSamples-train)/test)

osReturns <- vector(mode="numeric", length=0)

for( i in 0:(noOfPeriods-1) ) {
  cat("Starting backtesting period ", i+1, "\n")
  trainData <- AUDUSDFactors[(1 + i*test) : (train + i*test),]
  # log scaling is necessary to interpret spread reversion as returns
  linearModel <- lm(AUDUSD ~ .,data = log(trainData))
  testData <- AUDUSDFactors[(train + i*test + 1) : (train + test + test*i),]
  meanRevertingspread <- apply(log(testData)*
                                 c(1,-linearModel$coefficients[2:length(linearModel$coef
ficients)]) - linearModel$coefficients[1],
                               1,sum)
  osReturns[i+1] <- tradeSignal(testData, linearModel$coefficients, meanRevertingspread)
}
```

```
## Starting backtesting period  1
## Going long the spread at index level 2  and spread level  -12.97147
## Closing long spreadPosition at index level 3  and spread level  0.4989728
## Going long the spread at index level 8  and spread level  -13.02373
## Closing long spreadPosition at index level 9  and spread level  0.5075611
## Going long the spread at index level 14  and spread level  -12.97899
## Closing long spreadPosition at index level 15  and spread level  0.4873737
## Going long the spread at index level 20  and spread level  -12.98421
## Closing long spreadPosition at index level 21  and spread level  0.4974719
## Going long the spread at index level 26  and spread level  -12.98911
## Closing long spreadPosition at index level 27  and spread level  0.5032364
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -4.269486
## Starting backtesting period  2
## Going long the spread at index level 2  and spread level  -12.21151
## Closing long spreadPosition at index level 7  and spread level  6.73028
## Going long the spread at index level 8  and spread level  -12.21592
## Closing long spreadPosition at index level 13  and spread level  6.64161
## Going long the spread at index level 14  and spread level  -12.19699
## Closing long spreadPosition at index level 19  and spread level  6.601747
## Going long the spread at index level 20  and spread level  -12.18705
## Closing long spreadPosition at index level 25  and spread level  6.585038
## Going long the spread at index level 26  and spread level  -12.1694
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -5.270405
## Starting backtesting period  3
## Going long the spread at index level 2  and spread level  -10.50296
## Stop Loss Trigger: closing long spreadPosition at index level 5  and spread level  -1
2.00171
## Going long the spread at index level 6  and spread level  -10.7379
## Closing long spreadPosition at index level 7  and spread level  0.963873
## Going long the spread at index level 8  and spread level  -10.50424
## Stop Loss Trigger: closing long spreadPosition at index level 11  and spread level  -
12.00103
## Going long the spread at index level 12  and spread level  -10.73629
## Closing long spreadPosition at index level 13  and spread level  1.021874
## Going long the spread at index level 14  and spread level  -10.49478
## Stop Loss Trigger: closing long spreadPosition at index level 17  and spread level  -
11.99998
## Going long the spread at index level 18  and spread level  -10.732
## Closing long spreadPosition at index level 19  and spread level  1.089714
## Going long the spread at index level 20  and spread level  -10.48987
## Stop Loss Trigger: closing long spreadPosition at index level 23  and spread level  -
11.99917
## Going long the spread at index level 24  and spread level  -10.73606
## Closing long spreadPosition at index level 25  and spread level  1.033965
## Going long the spread at index level 26  and spread level  -10.4988
## Stop Loss Trigger: closing long spreadPosition at index level 29  and spread level  -
12.0012
## Going long the spread at index level 30  and spread level  -10.73494
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -10.73494
## Starting backtesting period  4
## Going long the spread at index level 2  and spread level  -9.51673
## Closing long spreadPosition at index level 3  and spread level  -0.5253557
## Going long the spread at index level 5  and spread level  -8.672654
## Closing long spreadPosition at index level 7  and spread level  6.436968
## Going long the spread at index level 8  and spread level  -9.511082
## Closing long spreadPosition at index level 9  and spread level  -0.5284122
## Going long the spread at index level 11  and spread level  -8.668278
## Closing long spreadPosition at index level 13  and spread level  6.40024
## Going long the spread at index level 14  and spread level  -9.506359
## Closing long spreadPosition at index level 15  and spread level  -0.5488304
## Going long the spread at index level 17  and spread level  -8.66556
## Closing long spreadPosition at index level 19  and spread level  6.39111
## Going long the spread at index level 20  and spread level  -9.507266
## Closing long spreadPosition at index level 25  and spread level  6.37023
## Going long the spread at index level 26  and spread level  -9.504604
## Closing long spreadPosition at index level 27  and spread level  -0.5511114
## Going long the spread at index level 29  and spread level  -8.669881
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -6.387667
## Starting backtesting period  5
## Going long the spread at index level 2  and spread level  -16.04274
## Closing long spreadPosition at index level 7  and spread level  2.01389
## Going long the spread at index level 8  and spread level  -16.035
## Closing long spreadPosition at index level 13  and spread level  2.041137
## Going long the spread at index level 14  and spread level  -16.04046
## Closing long spreadPosition at index level 19  and spread level  2.021339
## Going long the spread at index level 20  and spread level  -16.04492
## Closing long spreadPosition at index level 25  and spread level  2.066821
## Going long the spread at index level 26  and spread level  -16.04858
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  -10.30041
## Starting backtesting period  6
## Going short the spread at index level 1  and spread level  16.58409
## Stop Loss Trigger: closing long spreadPosition at index level 7  and spread level  1
6.45772
## Going short the spread at index level 10  and spread level  10.42552
## Stop Loss Trigger: closing long spreadPosition at index level 13  and spread level  1
6.52719
## Going short the spread at index level 16  and spread level  10.44889
## Stop Loss Trigger: closing long spreadPosition at index level 19  and spread level  1
6.48662
## Going short the spread at index level 22  and spread level  10.43515
## Stop Loss Trigger: closing long spreadPosition at index level 25  and spread level  1
6.528
## Going short the spread at index level 28  and spread level  10.51678
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  2.145155
## Starting backtesting period  7
## Going short the spread at index level 1  and spread level  18.46967
## Stop Loss Trigger: closing long spreadPosition at index level 7  and spread level  1
8.58029
## Going short the spread at index level 10  and spread level  13.79863
## Stop Loss Trigger: closing long spreadPosition at index level 13  and spread level  1
8.69512
```

```
## Going short the spread at index level 16  and spread level  13.84672
## Stop Loss Trigger: closing long spreadPosition at index level 19  and spread level  1
8.68205
## Going short the spread at index level 22  and spread level  13.86568
## Stop Loss Trigger: closing long spreadPosition at index level 25  and spread level  1
8.80098
## Going short the spread at index level 28  and spread level  13.88304
## End of Trading Period: Closing open spreadPosition at index level 30  and spread leve
l  3.180543
```

Test period returns

```
osReturns
```

```
## [1]  8.195649  4.452018 -4.606145 -7.030562 -2.371616 -1.866206  6.991498
```

As we can see out our trading strategy is profitable in only 3 out of the 7 test period returns. Lets calculate the sharpe ratio of our analysis.

```
# annual share ratio: sqrt(12) as our test set returns are monthly
sqrt(12)*mean(osReturns)/sd(osReturns)
```

```
## [1] 0.3122427
```

We do come up with a trading strategy exploiting the stationarity of the regression residuals but also realize that this strategy is not always profitable. I am proposing a couple of ideas to make to more it profitable:

1. One could try to model the spread using an OU mean reverting process and try to estimate the half life of mean reversion, the test period can then be chosen allowing for enough time for reversion
2. Instead of keeping the estimated regression coefficents in the model constant, use a rolling regression or kalman filter model. We think that Kalman filtering will be more useful at it is quicker to absorb any new information.