Stocks Data Exploration

Josh Kong

8/12/2020

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------- t

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------------------------- tidyve
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

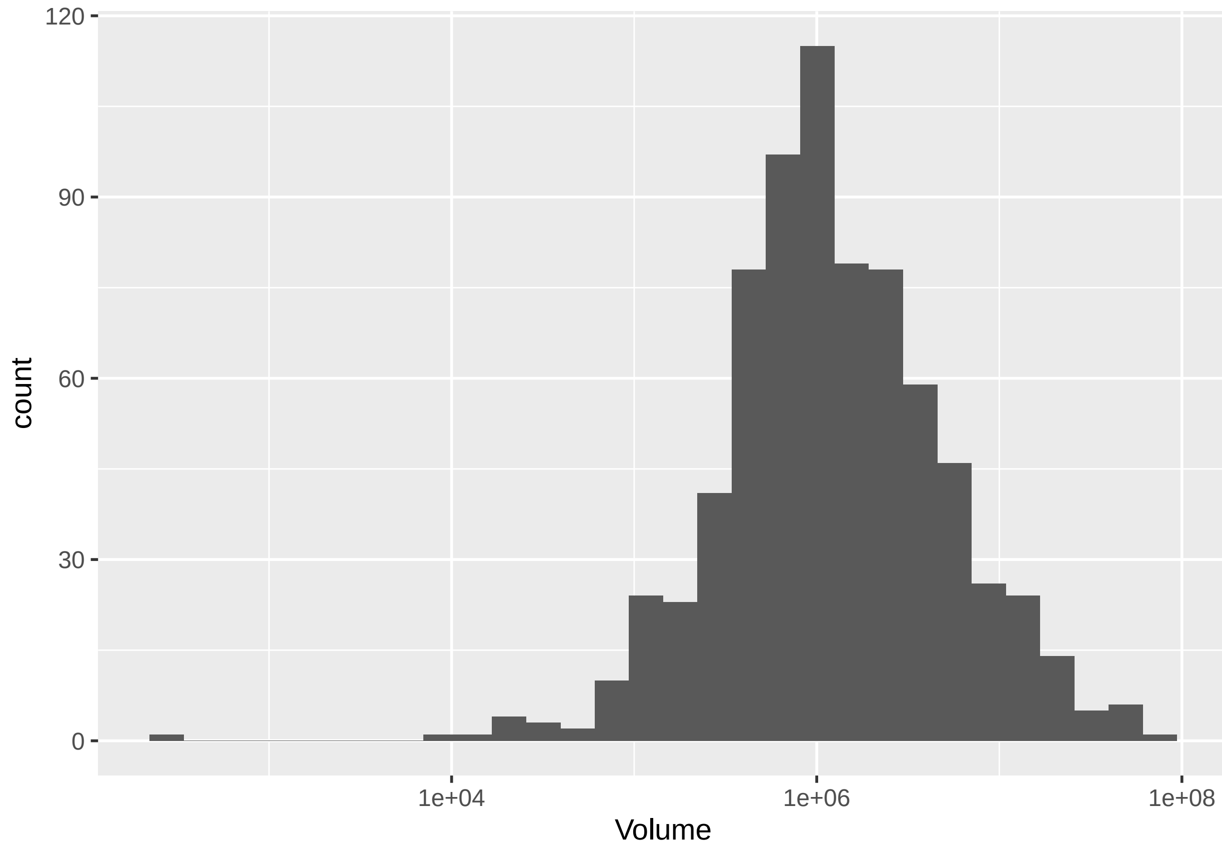
```
library(broom)
stocks <- read_csv("stocks_final.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Ticker = col_character(),
##   Company = col_character(),
##   Sector = col_character(),
##   Industry = col_character(),
##   Country = col_character()
## )

## See spec(...) for full column specifications.
```

Looking at the distributions of the data

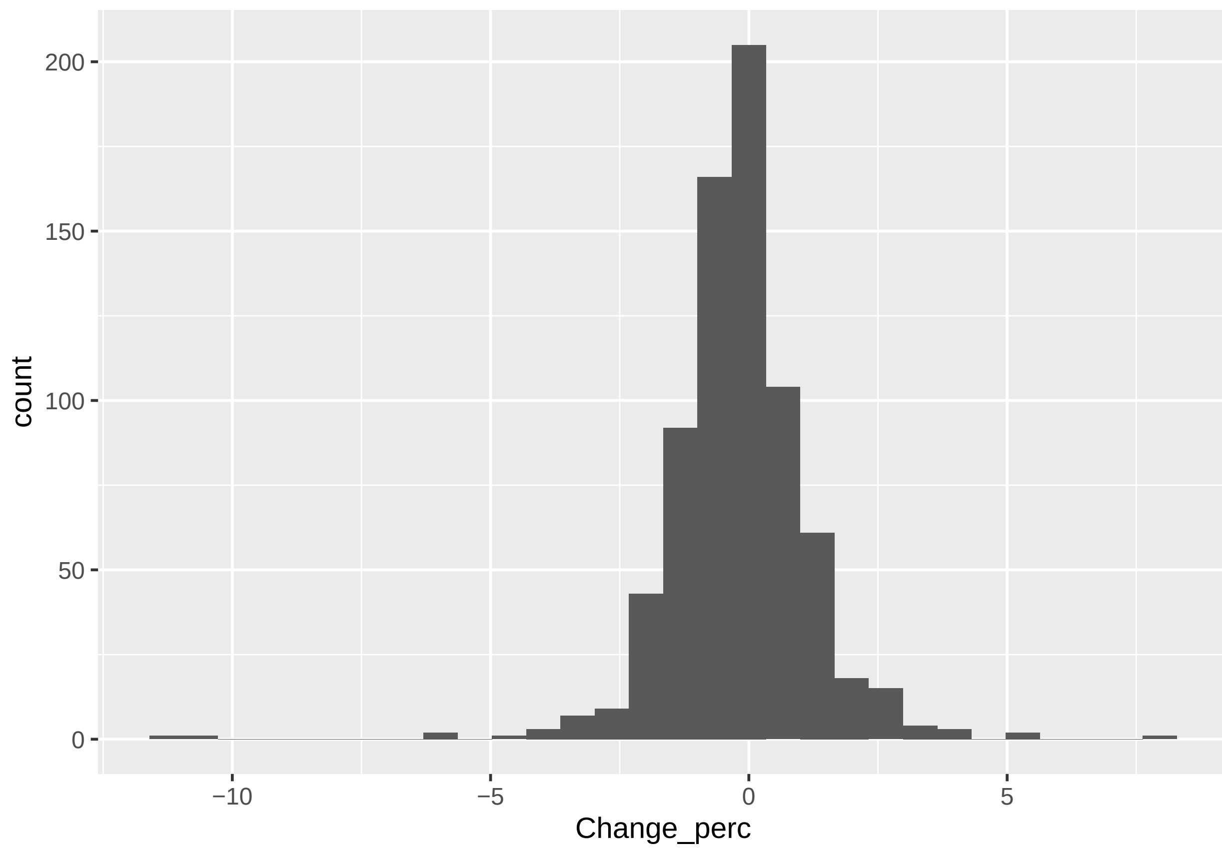
```
stocks %>%
  ggplot(aes(Volume)) +
  geom_histogram()+
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
stocks %>%
  ggplot(aes(Change_perc)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Exploring the categorical variables

```
#looking at the distribution
stocks %>%
  count(Country, sort= TRUE)
```

```
## # A tibble: 34 x 2
##    Country            n
##    <chr>          <int>
##  1 USA              534
##  2 Canada            34
##  3 China             24
##  4 United Kingdom    24
##  5 Brazil            14
##  6 Ireland           11
##  7 Japan             11
##  8 Netherlands       11
##  9 Switzerland       11
## 10 Bermuda            5
## # ... with 24 more rows
```

```
stocks %>%
  count(Industry, sort = TRUE)
```

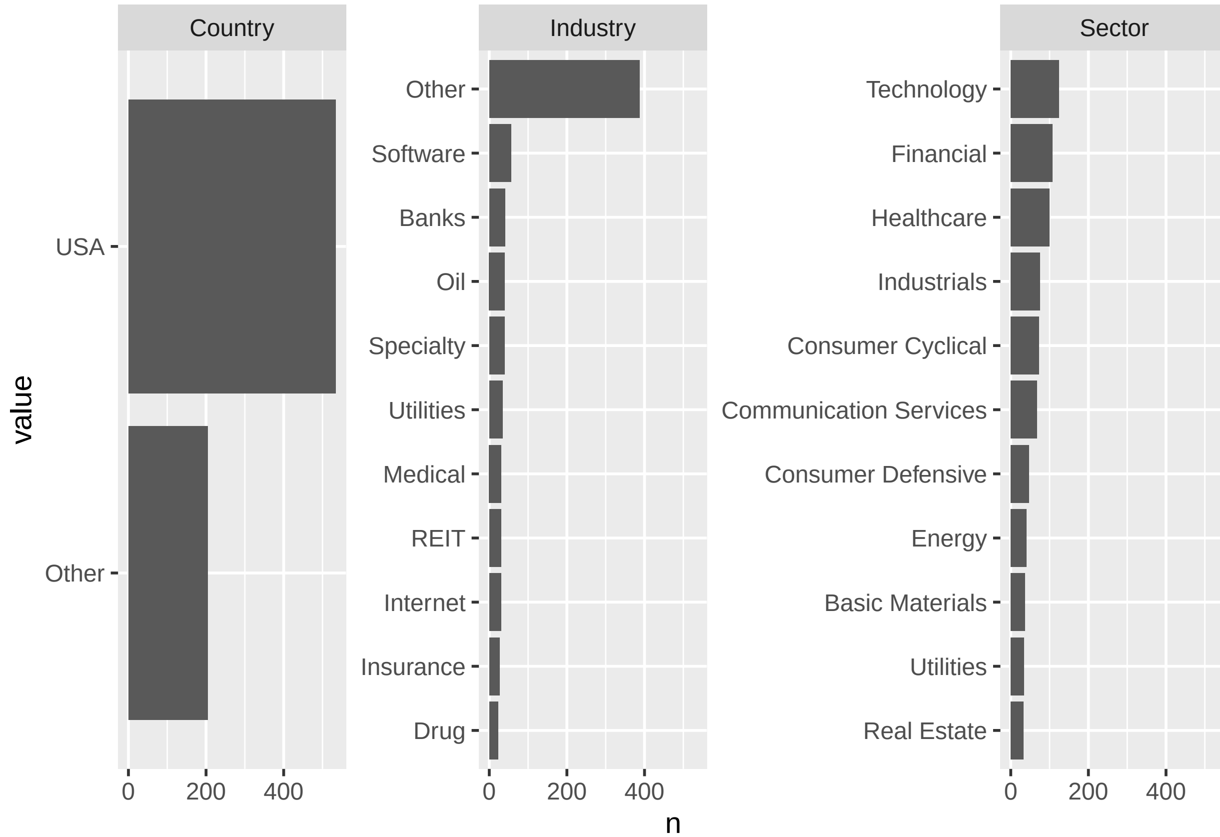
```
## # A tibble: 117 x 2
##    Industry                          n
##    <chr>                         <int>
##  1 Software - Application           34
##  2 Software - Infrastructure        22
##  3 Banks - Regional                 21
##  4 Specialty Industrial Machinery   21
##  5 Telecom Services                 21
##  6 Utilities - Regulated Electric   21
##  7 Banks - Diversified              20
##  8 Semiconductors                   20
##  9 Internet Content & Information   19
## 10 Biotechnology                    17
## # ... with 107 more rows
```

```
stocks %>%
  count(Sector, sort = TRUE)
```

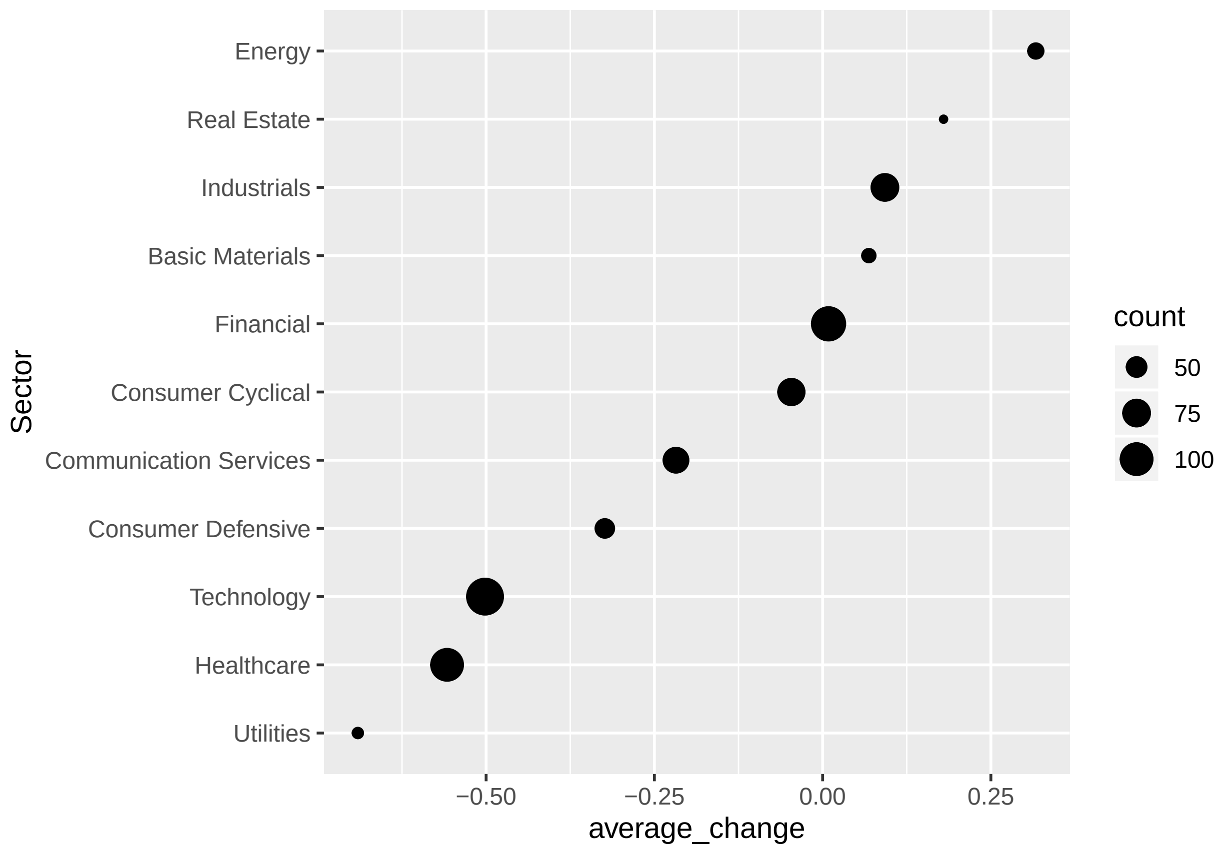
```
## # A tibble: 11 x 2
##    Sector                     n
##    <chr>                  <int>
##  1 Technology               124
##  2 Financial                108
##  3 Healthcare               100
##  4 Industrials               75
##  5 Consumer Cyclical         73
##  6 Communication Services    67
##  7 Consumer Defensive        47
##  8 Energy                    40
##  9 Basic Materials           37
## 10 Utilities                 34
## 11 Real Estate               33
```

```
#for country, I'm going to be looking at USA countries vs every other country
#there are too many different industries. Going to do some feature engineering to bring down the distin
#i'm going to leave Sector alone
stocks_category <- stocks%>%
  separate(Industry, c("Industry", "Other"), extra = "merge", sep = " ", fill = "right") %>%
  select(Company,Sector, Industry, Country, Change_perc) %>%
  mutate(Country = fct_lump(Country,1),
         Industry = fct_lump(Industry,10))

#looking at the distributions of the categorical data
stocks_category %>%
  gather(category,value,-Change_perc,-Company) %>%
  count(category,value, sort = TRUE) %>%
  group_by(category) %>%
  top_n(20,n) %>%
  ungroup() %>%
  mutate(value = fct_reorder(value, n)) %>%
  ggplot(aes(value, n)) +
  geom_col() +
  facet_wrap(~category, scale = "free_y") +
  coord_flip()
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```



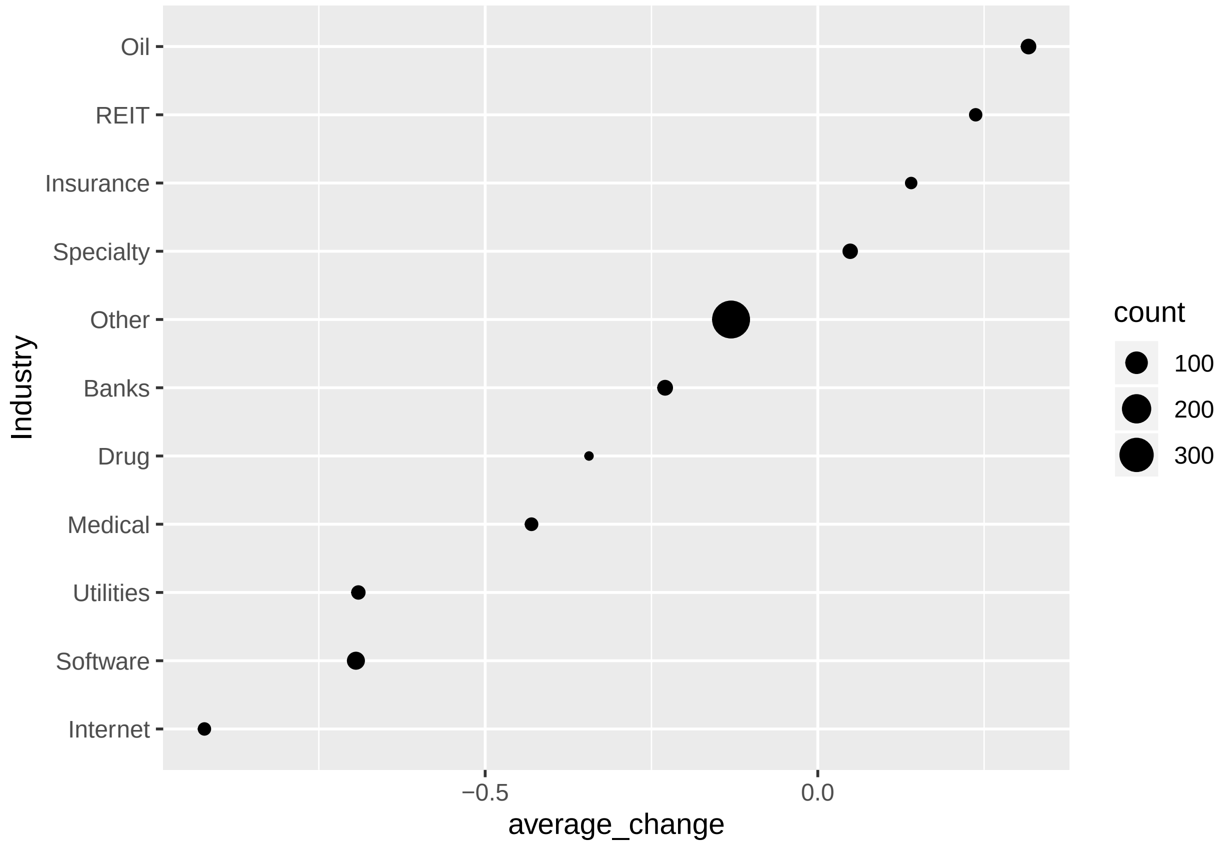

```r
#taking a closer look at Sector
stocks %>%
  group_by(Sector) %>%
  summarise(average_change = mean(Change_perc), count = n()) %>%
  ungroup() %>%
  mutate(Sector = fct_reorder(Sector,average_change)) %>%
  ggplot(aes(Sector, average_change, size = count)) +
  geom_point() +
  coord_flip()
```




```r
stocks %>%
  lm(Change_perc ~ Sector, data = .) %>%
  anova()
```

```
## Analysis of Variance Table
##
## Response: Change_perc
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## Sector     10   63.97  6.3971  3.3561 0.0002736 ***
## Residuals 727 1385.76  1.9061
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

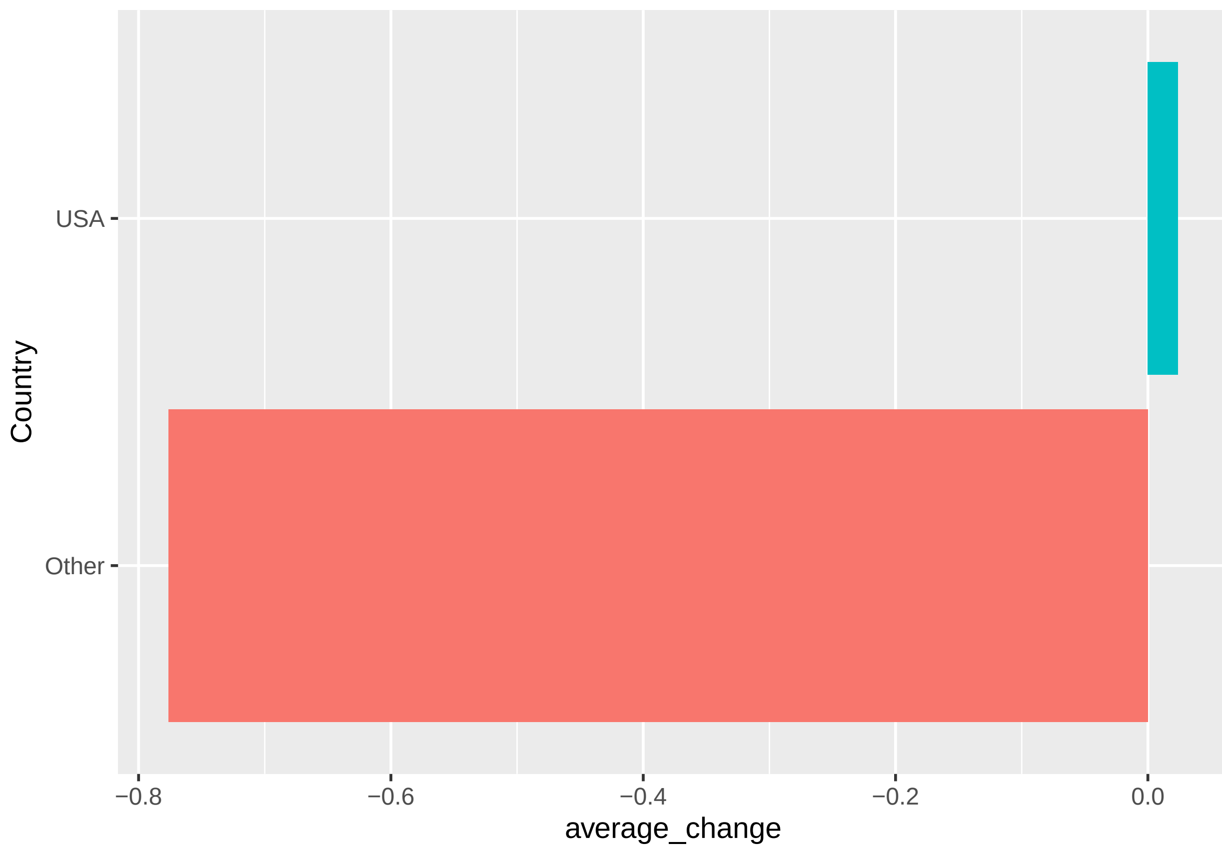
```r
#something that we notice is that technology sectors had an drop in their stocks in the month of July
# looks like that Sector is significant in determining the price change


#taking a closer look at industry
stocks_category %>%
  group_by(Industry) %>%
  summarise(average_change = mean(Change_perc), count = n()) %>%
  mutate(Industry = fct_reorder(Industry, average_change)) %>%
  ggplot(aes(Industry,average_change, size = count)) +
  geom_point() +
  coord_flip()
```




```
stocks_category %>%
  lm(Change_perc ~ Industry, data =.) %>%
  anova()
```

```
## Analysis of Variance Table
##
## Response: Change_perc
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## Industry   10   63.49  6.3491  3.3297 0.0003017 ***
## Residuals 727 1386.25  1.9068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#taking a closer look at country
stocks_category %>%
  group_by(Country) %>%
  summarise(average_change = mean(Change_perc), count = n()) %>%
  mutate(Country = fct_reorder(Country, average_change)) %>%
  ggplot(aes(Country,average_change, fill = average_change> 0)) +
  geom_col() +
  coord_flip()+
  theme(legend.position = "none")
```




```r
#appears that US did better than most other countries
stocks_category %>%
  lm(Change_perc~Country, data = .) %>%
  anova()
```

```
## Analysis of Variance Table
##
## Response: Change_perc
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## Country     1   94.49  94.488  51.314 1.913e-12 ***
## Residuals 736 1355.25   1.841
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#whether the company was in the US or not is significant in determining the price change of the stocks
```

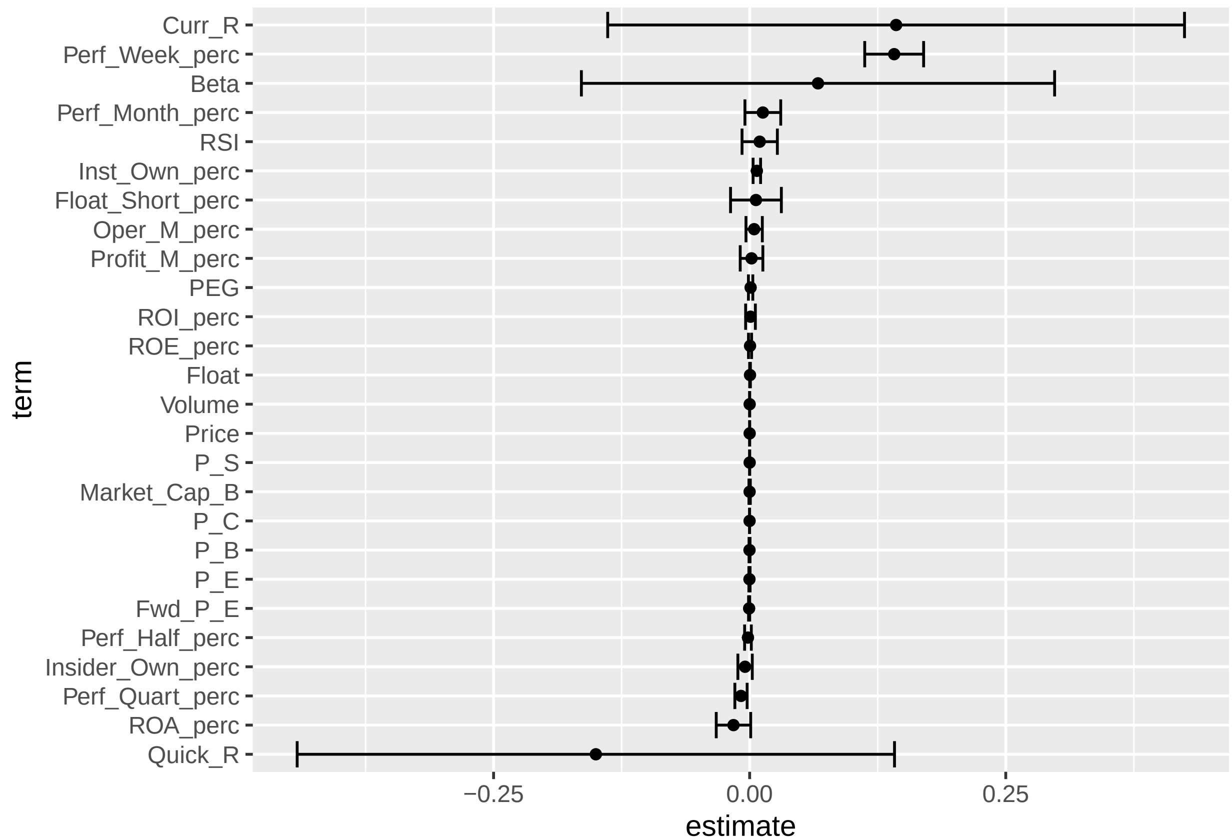
The three categorical variables that I looked closely at were country, industry and sector. Looking at the country, it seems that the majority of the companies were from the US, so I decided to make the country column have two options; either they were in the US or not. It seems that the stocks of the companies in the US tend to do better than countries that were not in the US.

After taking a closer look at industry and sector, it does appear that these categorical variables were significant in determining the performance of the stocks.

I'm not going to be using industry for my models due to there being too many different types of industries.

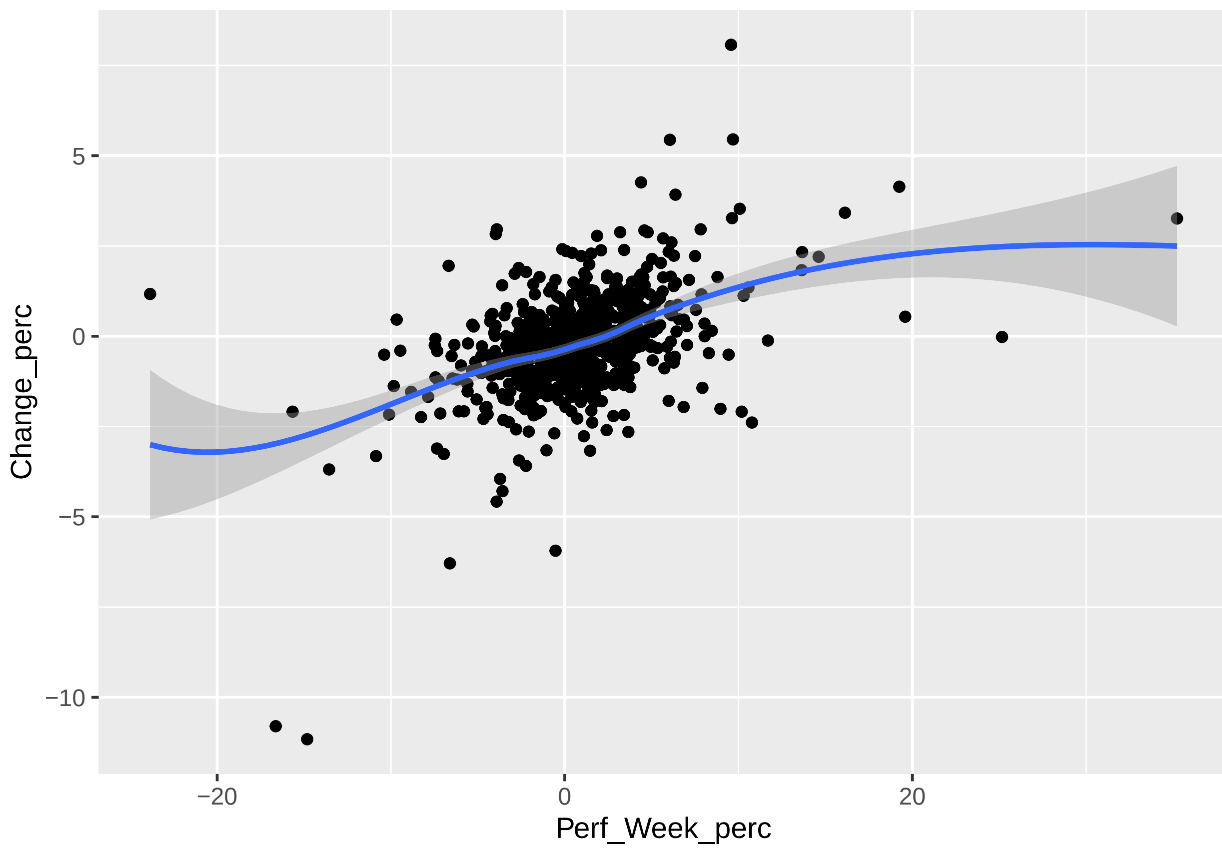
Exploring the numeric variables

```
#taking a look to see which numeric variables seem significant
#im going to filter for values that have a p-value smaller than 0.05, as they are the significant value
stocks_numeric <- stocks[7:33]
lm(Change_perc ~ ., data = stocks_numeric) %>%
  tidy(conf.int = TRUE) %>%
  filter(term != "(Intercept)") %>%
  arrange(desc(estimate)) %>%
  mutate(term = fct_reorder(term,estimate)) %>%
  ggplot(aes(term, estimate)) +
  geom_point() +
  coord_flip()+
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))
```



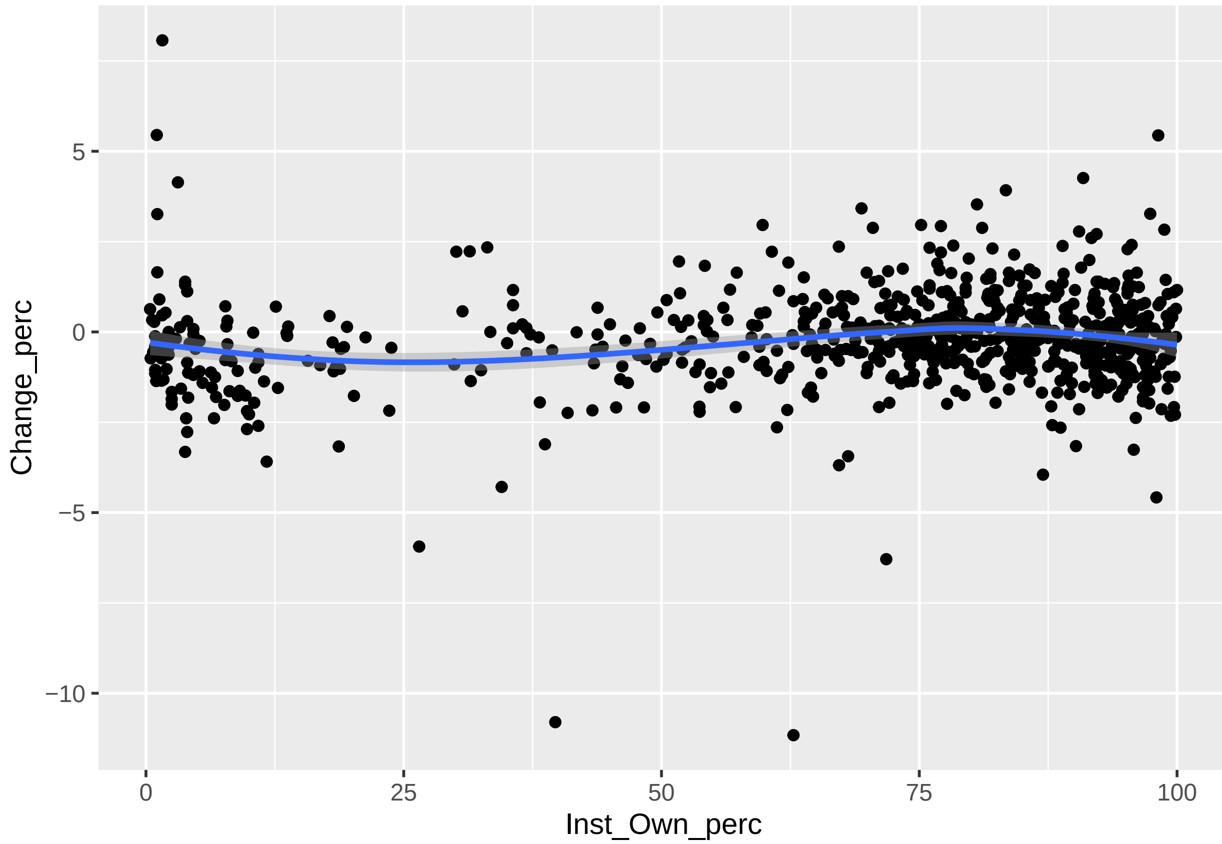

```
#Perf_week, inst_own, P_C, are the ones I will be analyzing

#looking closer at the weekly performance
stocks %>%
  ggplot(aes(Perf_Week_perc,Change_perc)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



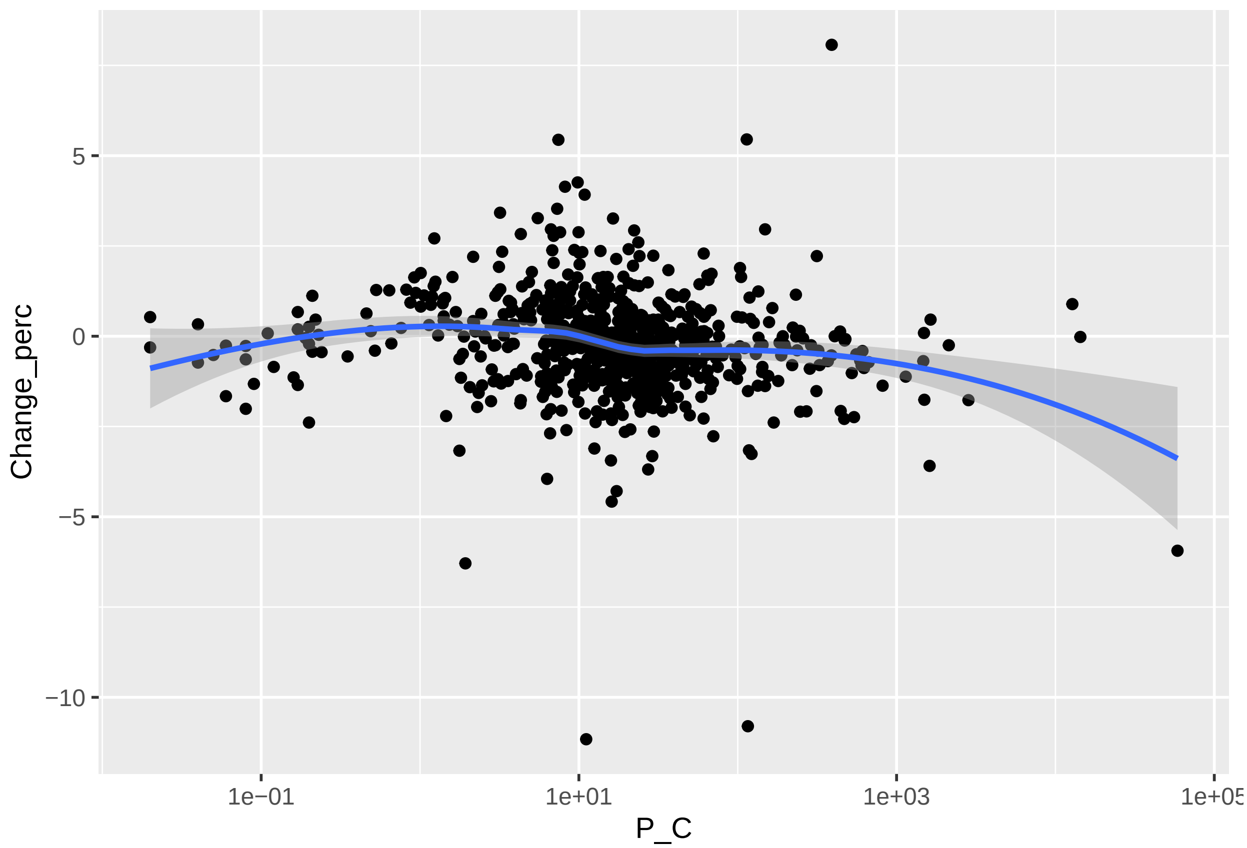

```r
#looking at Inst. Own.
stocks %>%
  ggplot(aes(Inst_Own_perc, Change_perc)) +
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
#P_C
stocks %>%
  ggplot(aes(P_C,Change_perc)) +
  geom_point() +
  geom_smooth()+
  scale_x_log10()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Principle Component Analysis

Due to the large number of numeric variables(27), I decided to use principle component analysis to bring down the number of variables.

```
stocks_pca <- prcomp(stocks_numeric[,-1],scale = TRUE)
summary(stocks_pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     1.9403 1.6272 1.50323 1.27126 1.22701 1.2142 1.14026
## Proportion of Variance 0.1448 0.1018 0.08691 0.06216 0.05791 0.0567 0.05001
## Cumulative Proportion  0.1448 0.2466 0.33355 0.39570 0.45361 0.5103 0.56032
##                            PC8     PC9    PC10    PC11   PC12    PC13    PC14
## Standard deviation     1.07280 1.06824 1.01402 1.00500 0.9458 0.93537 0.91671
## Proportion of Variance 0.04427 0.04389 0.03955 0.03885 0.0344 0.03365 0.03232
## Cumulative Proportion  0.60458 0.64847 0.68802 0.72687 0.7613 0.79492 0.82725
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.89739 0.84123 0.74374 0.73297 0.67519 0.58766 0.54811
## Proportion of Variance 0.03097 0.02722 0.02128 0.02066 0.01753 0.01328 0.01155
## Cumulative Proportion  0.85822 0.88544 0.90671 0.92737 0.94491 0.95819 0.96975
##                           PC22    PC23    PC24    PC25    PC26
## Standard deviation     0.52310 0.47126 0.38989 0.34539 0.13991
## Proportion of Variance 0.01052 0.00854 0.00585 0.00459 0.00075
## Cumulative Proportion  0.98027 0.98881 0.99466 0.99925 1.00000
```

```r
eig <- (stocks_pca$sdev)^2
vari <- eig*100/sum(eig)
cumvar <- cumsum(vari); cumvar
```

```
##  [1]  14.47955  24.66341  33.35452  39.57027  45.36090  51.03112  56.03188
##  [8]  60.45841  64.84739  68.80217  72.68686  76.12735  79.49239  82.72450
## [15]  85.82187  88.54364  90.67115  92.73749  94.49086  95.81911  96.97459
## [22]  98.02704  98.88123  99.46589  99.92471 100.00000
```

```r
#going to use the first 13 principle components to get around ~80% of the original variance
stocks_pca$rotation[,1:13]
```

```
##                         PC1         PC2         PC3          PC4           PC5
## Market_Cap_B     -0.06058325  0.03328492 -0.02425976  0.231010512 -0.0373733265
## Price            -0.00886355  0.02073151  0.04339093  0.050919086 -0.1620728891
## Volume            0.14081369 -0.05727193  0.12500436  0.312661770  0.1871812908
## P_E               0.23211077 -0.15044057 -0.08553854  0.080158127  0.2554719999
## Fwd_P_E           0.19215020 -0.03014007 -0.11999498 -0.007569089 -0.1299460138
## PEG               0.14194658 -0.17067484 -0.01003732 -0.041242656  0.2976000450
## P_B               0.03542454 -0.03971791 -0.03649445 -0.051639129 -0.0238870207
## P_C               0.05015673 -0.03924226  0.04248016  0.364270424  0.0794493079
## P_S               0.07307394  0.05541234  0.03595428  0.393597385  0.0254905843
## Curr_R            0.22454542  0.09855893 -0.46462866 -0.079681370 -0.2907807216
## Quick_R           0.23560780  0.07496806 -0.46348176 -0.065040180 -0.2998331410
## RSI              -0.04346742  0.50898541  0.16352631 -0.155745449  0.0127234973
## Beta              0.23428139  0.18742268  0.09530526 -0.133636358  0.0806638753
## Float            -0.02160728  0.02947642  0.04926483 -0.083358851  0.2690437616
## ROA_perc         -0.35425991  0.16226481 -0.24851353  0.116039431  0.0053578076
## ROE_perc         -0.10292194  0.10200170 -0.08525210  0.209075066  0.1414960245
## ROI_perc         -0.30698247  0.03618376 -0.16431090 -0.147005526 -0.0626618192
## Profit_M_perc    -0.35949127  0.14750089 -0.24603124  0.241868140  0.1040394269
## Oper_M_perc      -0.34624884  0.11949428 -0.16808669  0.212004433  0.0725023644
## Perf_Week_perc   -0.06214301  0.25697502  0.31944065 -0.111233096 -0.1129559482
## Perf_Month_perc   0.04277193  0.51107373  0.08234513 -0.116854241 -0.0004542538
## Perf_Quart_perc   0.22927880  0.39546286  0.10468729  0.196464861  0.0975582706
## Perf_Half_perc    0.26738683  0.20573319 -0.19709124  0.345717979 -0.0102451584
## Inst_Own_perc     0.02330027  0.10590217 -0.29943040 -0.297856226  0.4242810629
## Insider_Own_perc -0.04025602 -0.05559255  0.22824546  0.164962387 -0.5104311309
## Float_Short_perc  0.28603834  0.10767082 -0.03898108  0.076592388  0.0504853777
##                           PC6         PC7          PC8          PC9
## Market_Cap_B      0.0483178638  0.44096168 -0.204382303 -0.496637474
## Price            -0.0520139614  0.22809597 -0.301895041 -0.236517679
## Volume            0.2296976790  0.33218146  0.029401660 -0.253327833
## P_E              -0.5275738965  0.12941593 -0.082743851  0.091974265
## Fwd_P_E           0.0627766606 -0.20524678 -0.244965490  0.073431901
## PEG              -0.5521397665  0.26185895 -0.049694679  0.164844810
## P_B              -0.0022210805 -0.11786991 -0.137697305 -0.125765705
## P_C              -0.1153056427 -0.30900050  0.147161453 -0.236897937
## P_S              -0.2155783665 -0.37519521  0.272469180 -0.259537464
## Curr_R           -0.0576951392  0.15460894  0.283328937 -0.071678528
## Quick_R          -0.0438104209  0.15603655  0.274034156 -0.069843273
## RSI              -0.1582919212  0.06175405 -0.035234015 -0.078303304
```

```
## Beta             -0.0517799018  0.04278397  0.072193023 -0.053561835
## Float             0.1475043996  0.09673488  0.467525979  0.075359227
## ROA_perc         -0.1640871058 -0.06734876 -0.144909054 -0.006641007
## ROE_perc          0.0869325498  0.13156478  0.102029353  0.313827069
## ROI_perc         -0.2764963643 -0.20365032 -0.166678668 -0.209257363
## Profit_M_perc     0.0001948431  0.07428549  0.027592638  0.167503900
## Oper_M_perc      -0.0009756812  0.09152494  0.082395872  0.155113955
## Perf_Week_perc   -0.2444277149 -0.08046130  0.323185321 -0.194763434
## Perf_Month_perc  -0.0721582857  0.08031686 -0.152819422  0.062827023
## Perf_Quart_perc   0.0827002144  0.08040776 -0.043736867  0.172928398
## Perf_Half_perc    0.0124664349 -0.16240302 -0.154703422  0.073052714
## Inst_Own_perc     0.1269273926 -0.11716529 -0.070409187 -0.202266144
## Insider_Own_perc -0.1659860665  0.11003358 -0.006801648  0.300330520
## Float_Short_perc  0.1183490909 -0.23134992 -0.276125251  0.159839681
##                         PC10         PC11          PC12         PC13
## Market_Cap_B      0.02126866  0.003851162  0.0622725278  0.400253601
## Price            -0.48703721  0.257236510 -0.4748357519 -0.457870611
## Volume            0.29334347 -0.171242477 -0.0001516822 -0.007785842
## P_E              -0.05239522  0.018253154  0.0502716909  0.034219491
## Fwd_P_E          -0.10072568 -0.120830591 -0.4178668847  0.528884853
## PEG               0.04171869 -0.007256833 -0.0378742137  0.017035909
## P_B               0.47013720  0.792122401  0.0729887973 -0.049532142
## P_C              -0.09378980 -0.039537341 -0.0424345164 -0.012895843
## P_S              -0.08292624  0.056533122 -0.0265507314 -0.062569091
## Curr_R            0.03373677 -0.010280222  0.0029986588 -0.038071122
## Quick_R           0.02554524 -0.006631425  0.0044167860 -0.026450817
## RSI              -0.13374751  0.010407192  0.1481996163  0.076276706
## Beta              0.41952767 -0.094012586 -0.4687734246 -0.103138456
## Float            -0.26294910  0.270916188 -0.1124406939  0.101264004
## ROA_perc          0.08949856 -0.004221794 -0.0073115132  0.075148833
## ROE_perc         -0.10811478  0.364824944 -0.2214511918  0.423765164
## ROI_perc          0.13510616 -0.041646101 -0.0327803031  0.096323705
## Profit_M_perc     0.08850630 -0.114728157 -0.1385340311 -0.143801962
## Oper_M_perc       0.17829324 -0.078530261 -0.1899283004 -0.199304294
## Perf_Week_perc    0.08161224 -0.014308310 -0.2374305193  0.174858480
## Perf_Month_perc  -0.06451133 -0.007592255  0.2075304929  0.031666043
## Perf_Quart_perc   0.08892794  0.016613266  0.0374379661 -0.151259287
## Perf_Half_perc   -0.16005807  0.098238531  0.2566633462 -0.004262082
## Inst_Own_perc    -0.11467878 -0.003574762 -0.0777331310 -0.039138060
## Insider_Own_perc  0.05318111  0.052895490 -0.0348112342  0.023268595
## Float_Short_perc  0.14909975 -0.087274150 -0.2246229394 -0.045432306
```

```
stocks_numeric_pca <- stocks_pca$x[,1:13]
```

After using principle component analysis, I reduced the number of numeric variables from 27 to 13 while keeping ~80& of the original variance.

Creating the machine learning model

Training the data

```r
stocks <- cbind(stocks_category[,-c(1,3)], stocks_numeric_pca)
stocks$Sector <- as.factor(stocks$Sector)
set.seed(1)
n <- nrow(stocks)
index <- sample(1:n, round(0.8*n))
train_stocks <- stocks[index,]
test_stocks <- stocks[-index,]
```

Multiple Linear Regression model

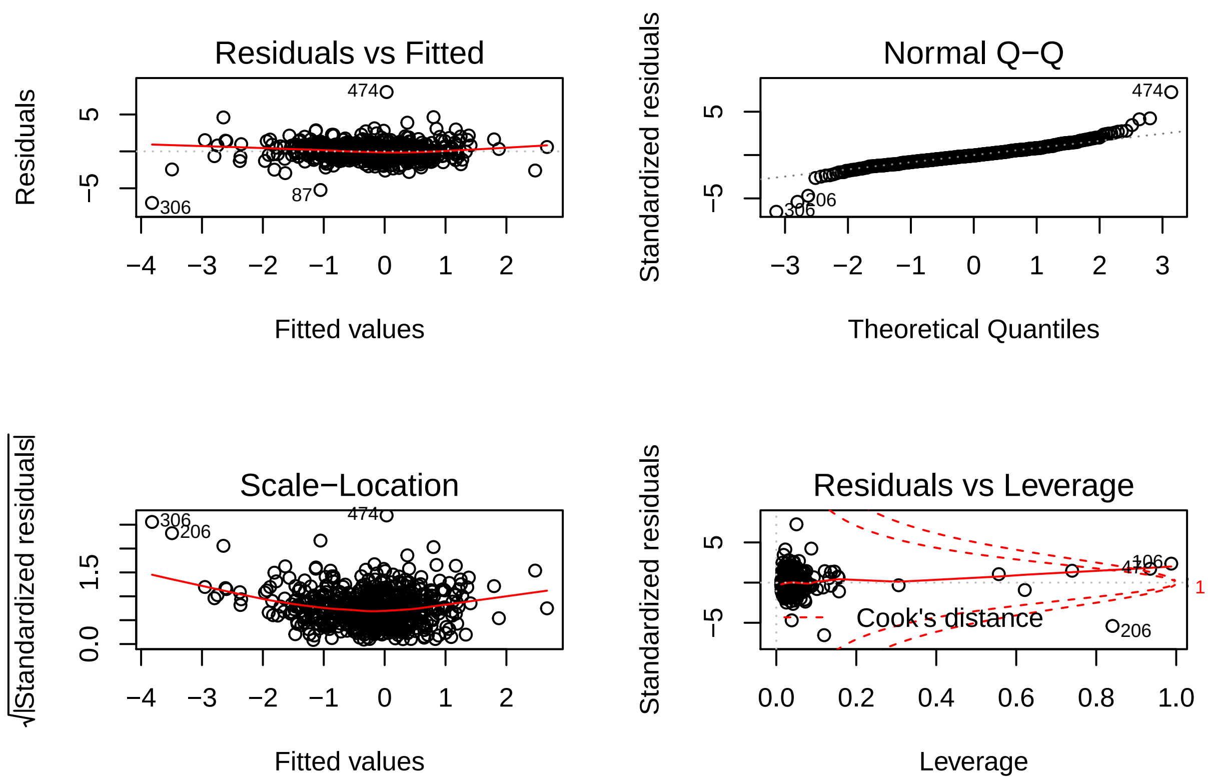
```r
lin_model <- lm(Change_perc ~ ., data = train_stocks)
summary(lin_model)
```

```
##
## Call:
## lm(formula = Change_perc ~ ., data = train_stocks)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9794 -0.6298 -0.0583  0.5939  8.0378
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     0.24932    0.22449   1.111 0.267201
## SectorCommunication Services -0.03326    0.27070  -0.123 0.902259
## SectorConsumer Cyclical      -0.29250    0.26884  -1.088 0.277065
## SectorConsumer Defensive     -0.30418    0.29219  -1.041 0.298306
## SectorEnergy                 -0.18747    0.31025  -0.604 0.545924
## SectorFinancial              -0.13253    0.25343  -0.523 0.601218
## SectorHealthcare             -0.22653    0.25532  -0.887 0.375339
## SectorIndustrials            -0.45275    0.26485  -1.709 0.087917 .
## SectorReal Estate             0.11934    0.30784   0.388 0.698393
## SectorTechnology             -0.40922    0.24754  -1.653 0.098852 .
## SectorUtilities              -0.72420    0.31902  -2.270 0.023578 *
## CountryOther                 -0.81881    0.14063  -5.822 9.73e-09 ***
## PC1                          -0.10250    0.02608  -3.931 9.51e-05 ***
## PC2                           0.12461    0.03245   3.840 0.000137 ***
## PC3                           0.25314    0.03565   7.101 3.74e-12 ***
## PC4                          -0.29481    0.04956  -5.948 4.76e-09 ***
## PC5                           0.02615    0.04312   0.607 0.544384
## PC6                          -0.04945    0.04199  -1.178 0.239375
## PC7                           0.14092    0.04914   2.868 0.004291 **
## PC8                           0.20138    0.04972   4.050 5.83e-05 ***
## PC9                          -0.06029    0.05259  -1.147 0.252060
## PC10                          0.07351    0.04684   1.569 0.117109
## PC11                         -0.05786    0.04462  -1.297 0.195283
## PC12                         -0.12350    0.05145  -2.400 0.016706 *
## PC13                          0.08864    0.05008   1.770 0.077257 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 565 degrees of freedom
## Multiple R-squared:  0.3328, Adjusted R-squared:  0.3045
```

```
## F-statistic: 11.74 on 24 and 565 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2,2))
plot(lin_model)
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```

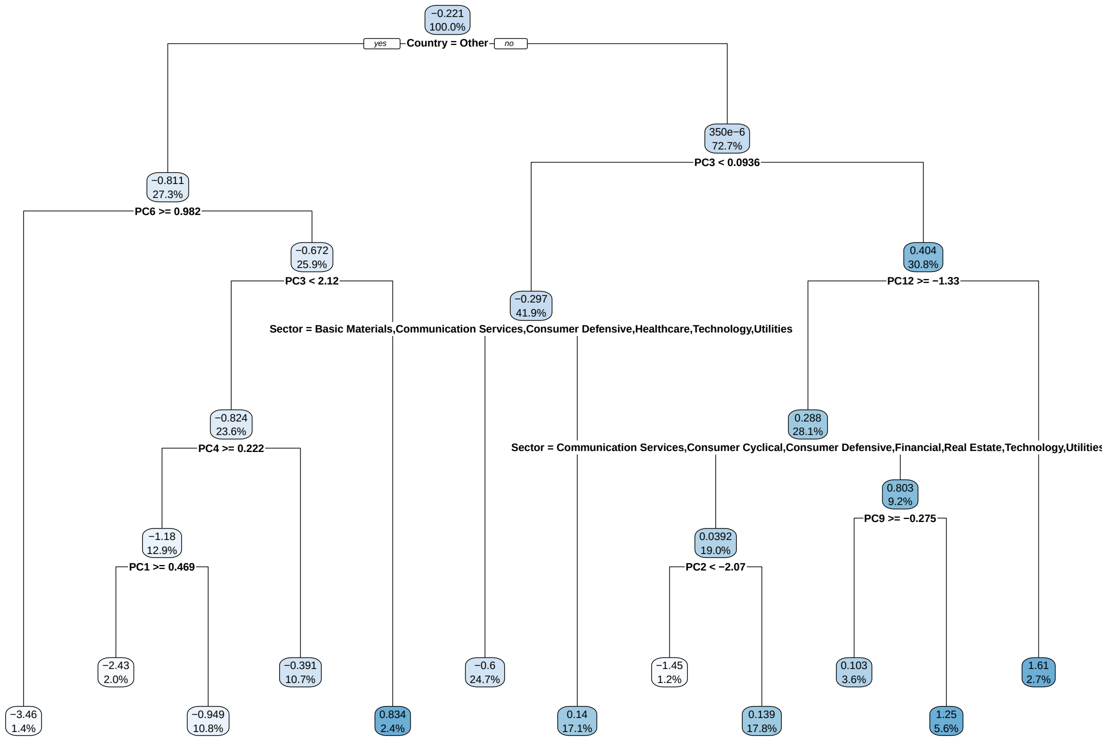



```r
#model seems to have constant variance and is approximately normal, so the model seems to be valid
```

```r
#creating a function that will assess our predictions. I'm going to be using the mean absolute error
MAE <- function(actual,predicted)
{
  mean(abs(actual-predicted))
}
```

Regression tree

```r
library(rpart)
library(rpart.plot)
tree_model <- rpart(Change_perc ~ ., data = train_stocks)
rpart.plot(tree_model, digit = 3)
```




```r
tree_predict <- predict(tree_model, test_stocks)
MAE(test_stocks$Change_perc,tree_predict)
```

```
## [1] 0.9303565
```

Using the regression tree model, I got a mean absolute error of 0.93.

Random Forest Regression model

```r
set.seed(12)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf_model <- randomForest(Change_perc ~ ., data = train_stocks, importance = TRUE, mtry = 3, na.action =
                         ntree = 500, proximity = TRUE)
rf_predict <- predict(rf_model, test_stocks)
MAE(test_stocks$Change_perc, rf_predict)
```

```
## [1] 0.8784454
```

Using a random forest model, I got a mean absolute error of 0.878.