Valuation Analysis

Moustafa Abdelaziz, Erik Wong, Luke Hong

Professor Kathleen Durant

Data Collection, Analysis, and Integration

Project Description

Our goal is to provide financially aware individuals with insight on company valuations for educated recommendations on stock purchases and sales. In order to do so, we created a prediction model using equity data of all companies in the S&P 500. Our model was built using analysis of the key statistics of each company in the S&P 500 index. We scraped the key statistics of each company in the index from Yahoo finance, cleaned the data, stored it in a database, and finally used the stored data to conduct multiple linear regression analyses. Our prediction model compares the estimated stock price against the current market price and instructs the user on whether or not to buy the current stock. In addition to our recommendation, we provided the user with key information of the S&P 500 index. This includes financial statements, plots of each financial statement for each company, and each stock's respective industry. Ultimately our user interface provides the user with all necessary information to inform their investment decisions.

Data Collection

To create our model, we initially scraped ticker names and industry information to establish a reference table for the individual stocks. We then pulled the equity statistics and ran two regression models to predict the price of each stock. We also pulled the three financial statements – the income statement, balance sheet, and statement of cash flows – for each company ticker. Finally, we stored our data in three csv files; one for each financial statement. This design ensured that we would be able to collate and analyze our data effectively.

Issues

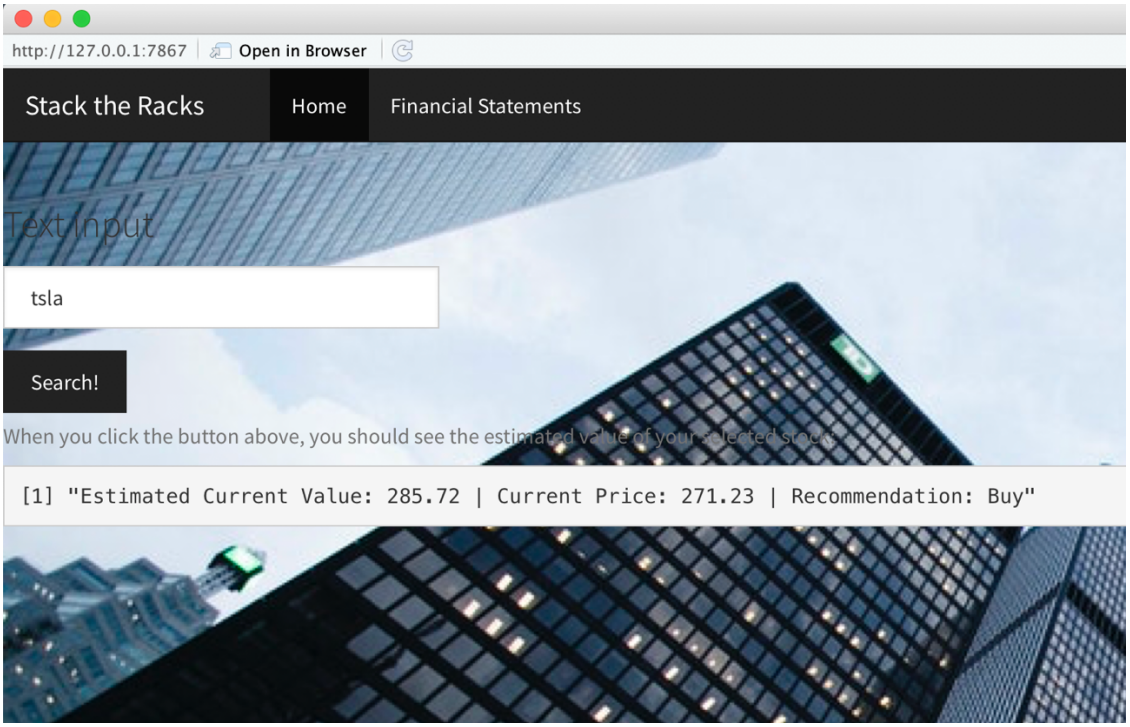
Our main issue was developing a model that accurately represented the value of each stock's price. This issue was comprised of ensuring the accuracy and uniformity of the collected data, determining where to retrieve the data, and determining how to accurately analyze the data.

To resolve these issues, we had to locate a website that stored the data uniformly. Once we located the source, we were able to ensure the uniformity. We then removed the NA values and were able to ensure accuracy.

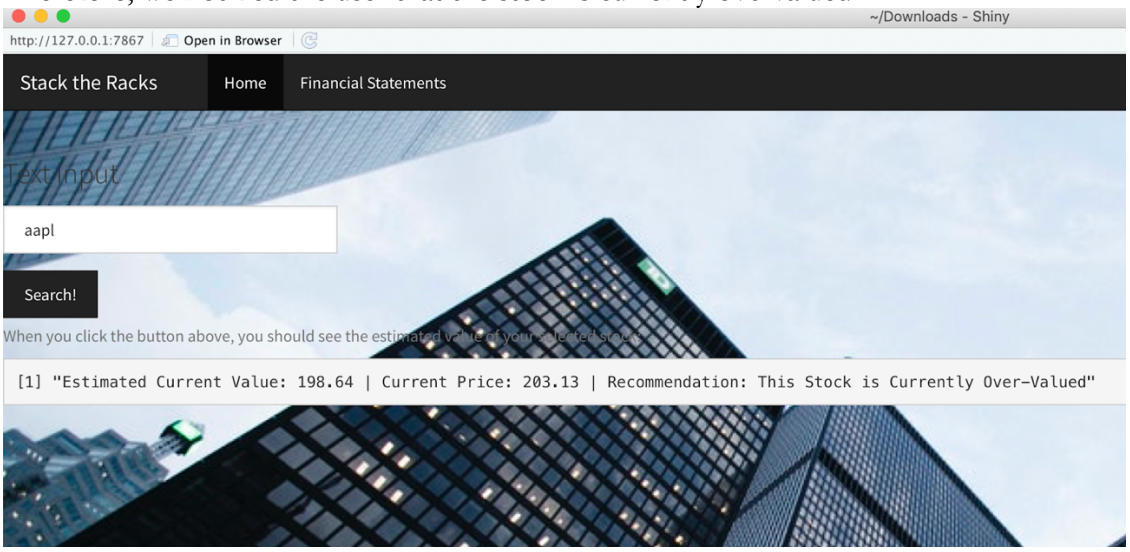
Our initial goal was to use the dividend discount model, the free cash flow model, and the free cash flow to equity model as valuation models. Nonetheless, after determining that that approach would not accurately provide us with an estimated price for each stock, we leaned towards scraping key statistics for each company and creating linear regression prediction models. Our models take into account statistics like moving averages, earnings per share (EPS), and other statistically significant variables. Doing our initial planned analysis would require analysis techniques that extend beyond the scope of this course, which will be expanded on in the Insights section.

Inputs and Outputs

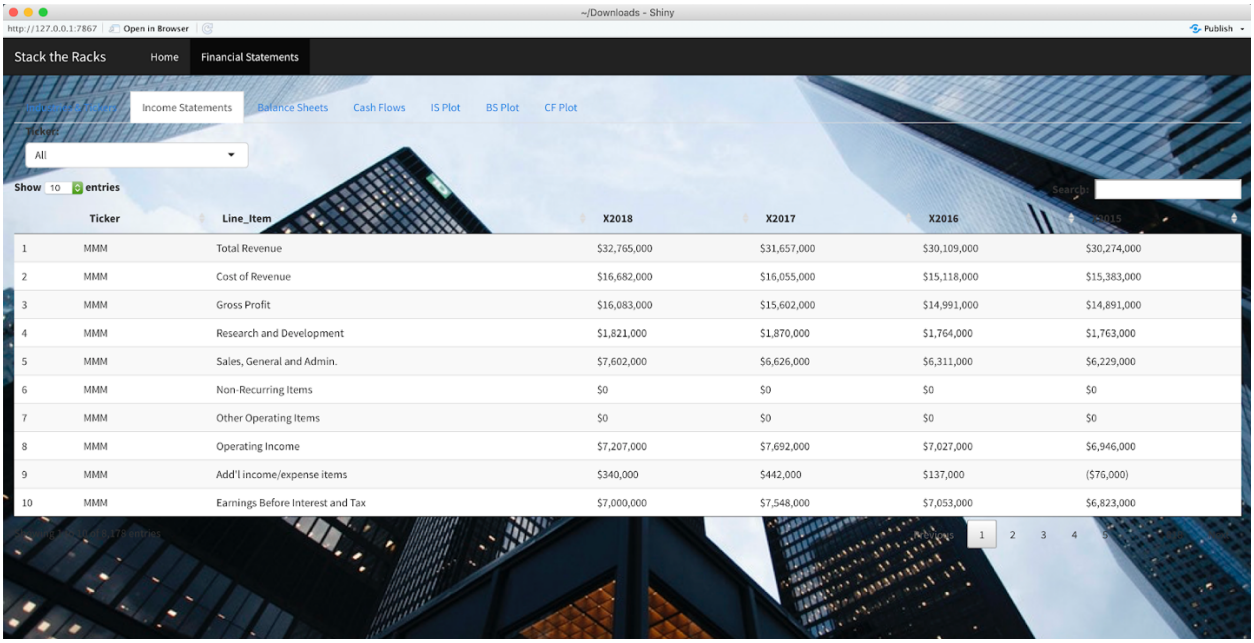
As seen below, the user interface provides the user with a recommendation based on the ticker input. In this case our prediction model valued TSLA at a higher price than the current market price. Therefore, we notified the user that they should purchase the stock.



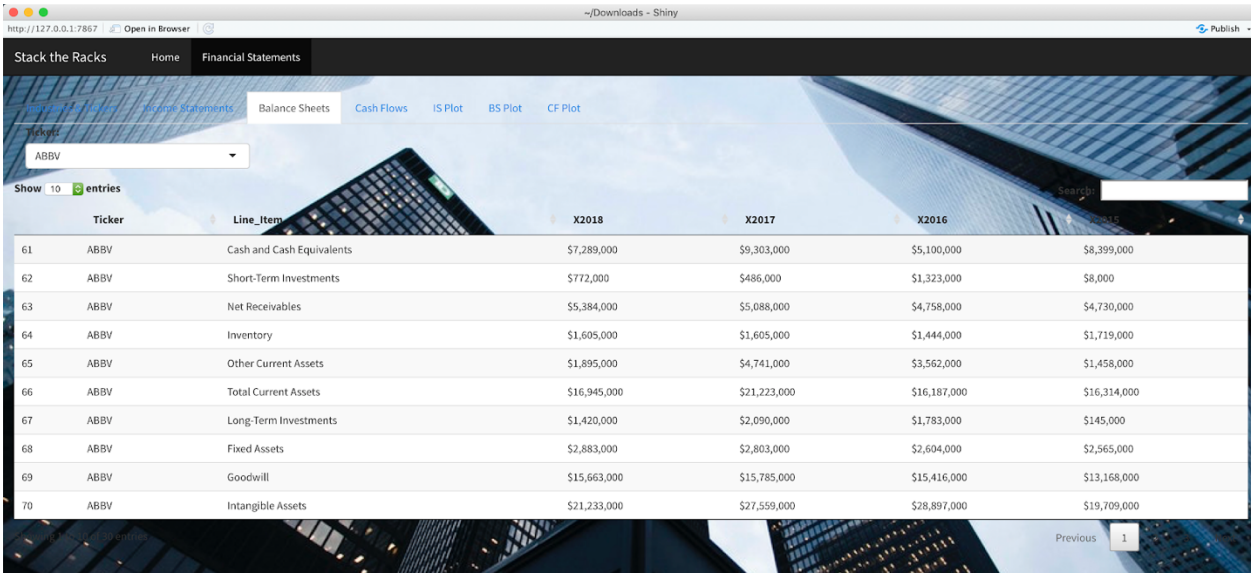
In this case, our prediction model valued AAPL at a lower price than the current market price. Therefore, we notified the user that the stock is currently overvalued.



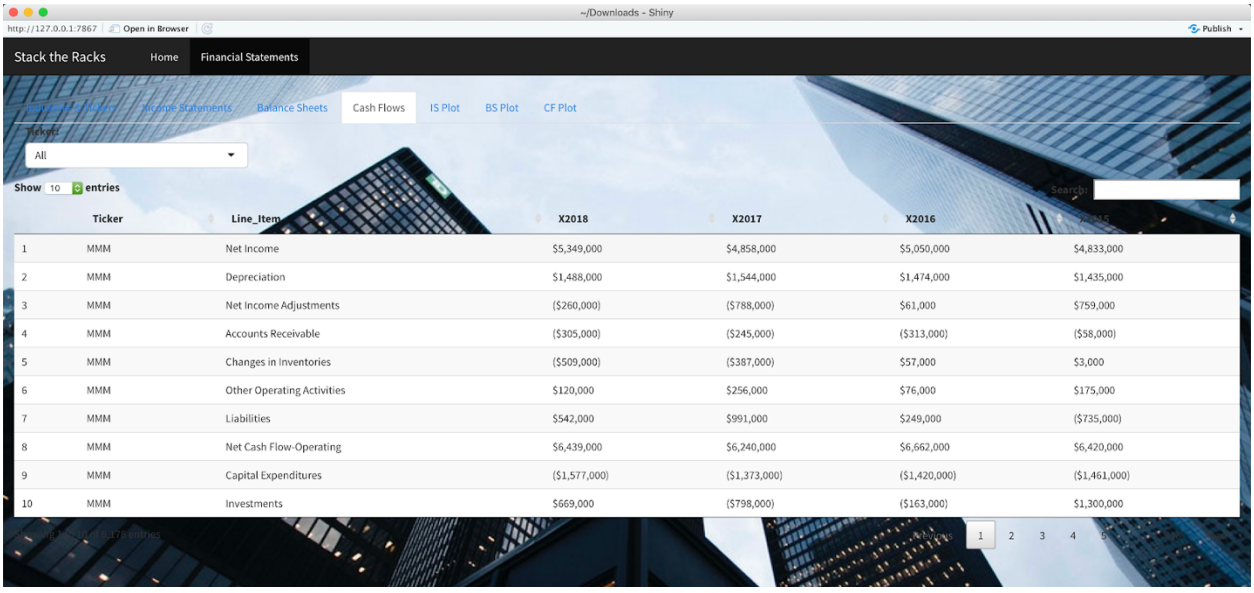
The below shows the income statements for all companies in the S&P 500. The user can choose which stock to analyze by using the drop-down menu.



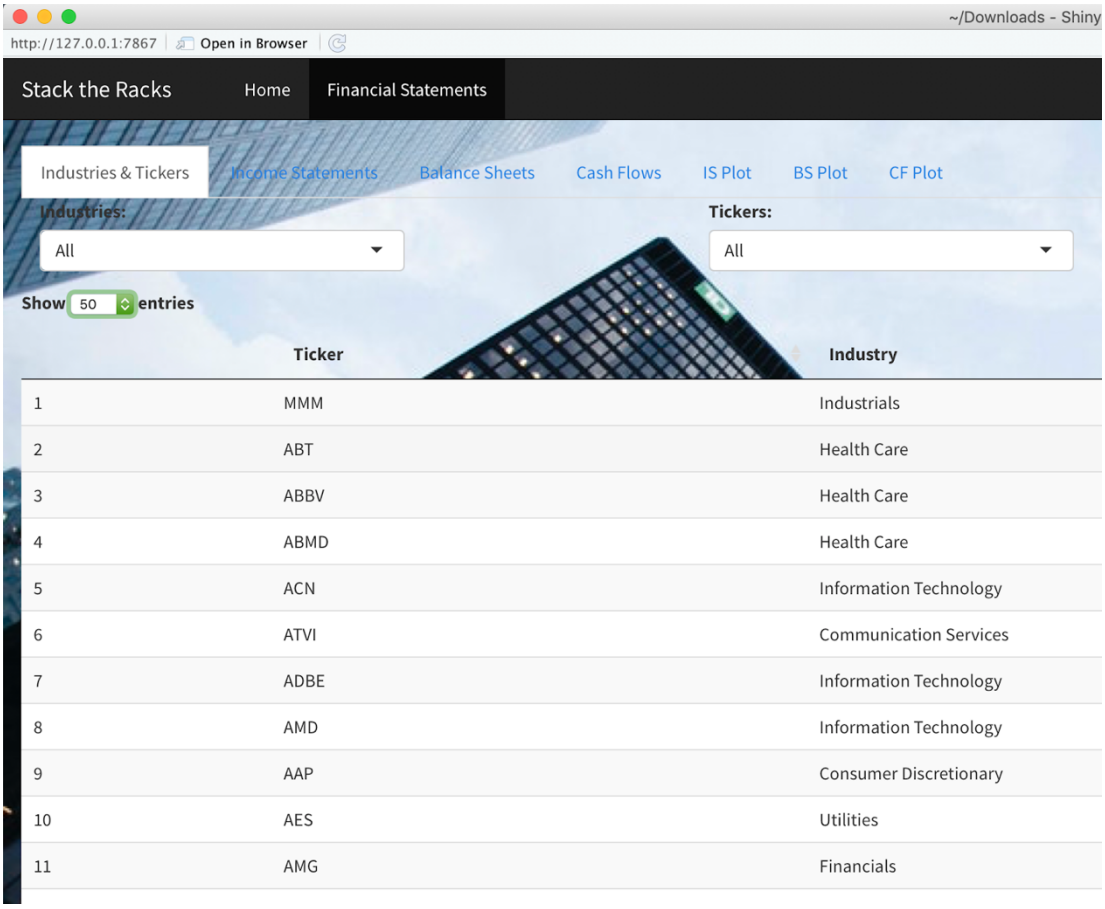
The below shows the balance sheets for all companies in the S&P 500. The user can choose which stock to analyze by using the drop-down menu.



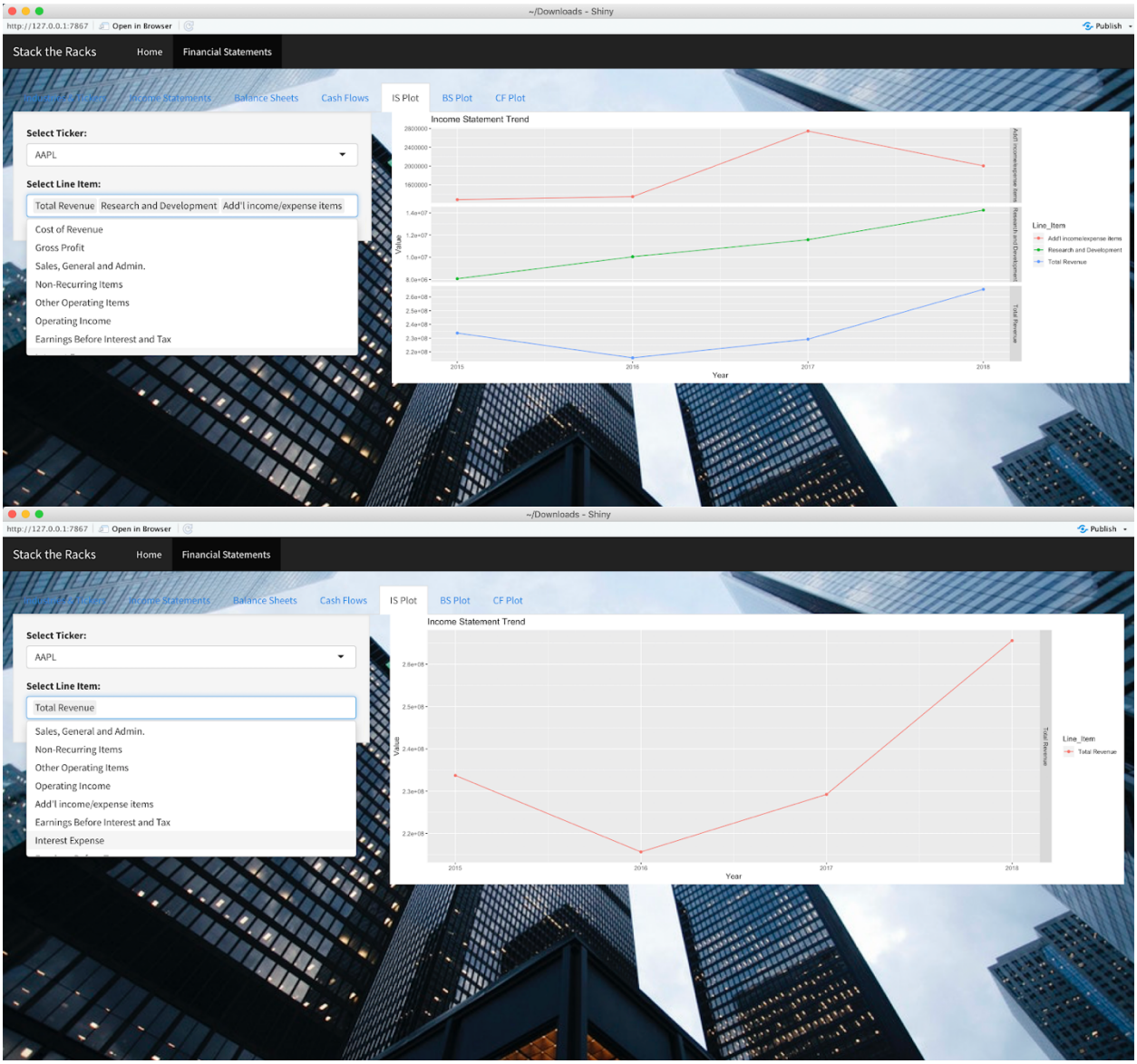
The below shows the cash flow statements for all companies in the S&P 500. The user can choose which stock to analyze by using the drop-down menu.

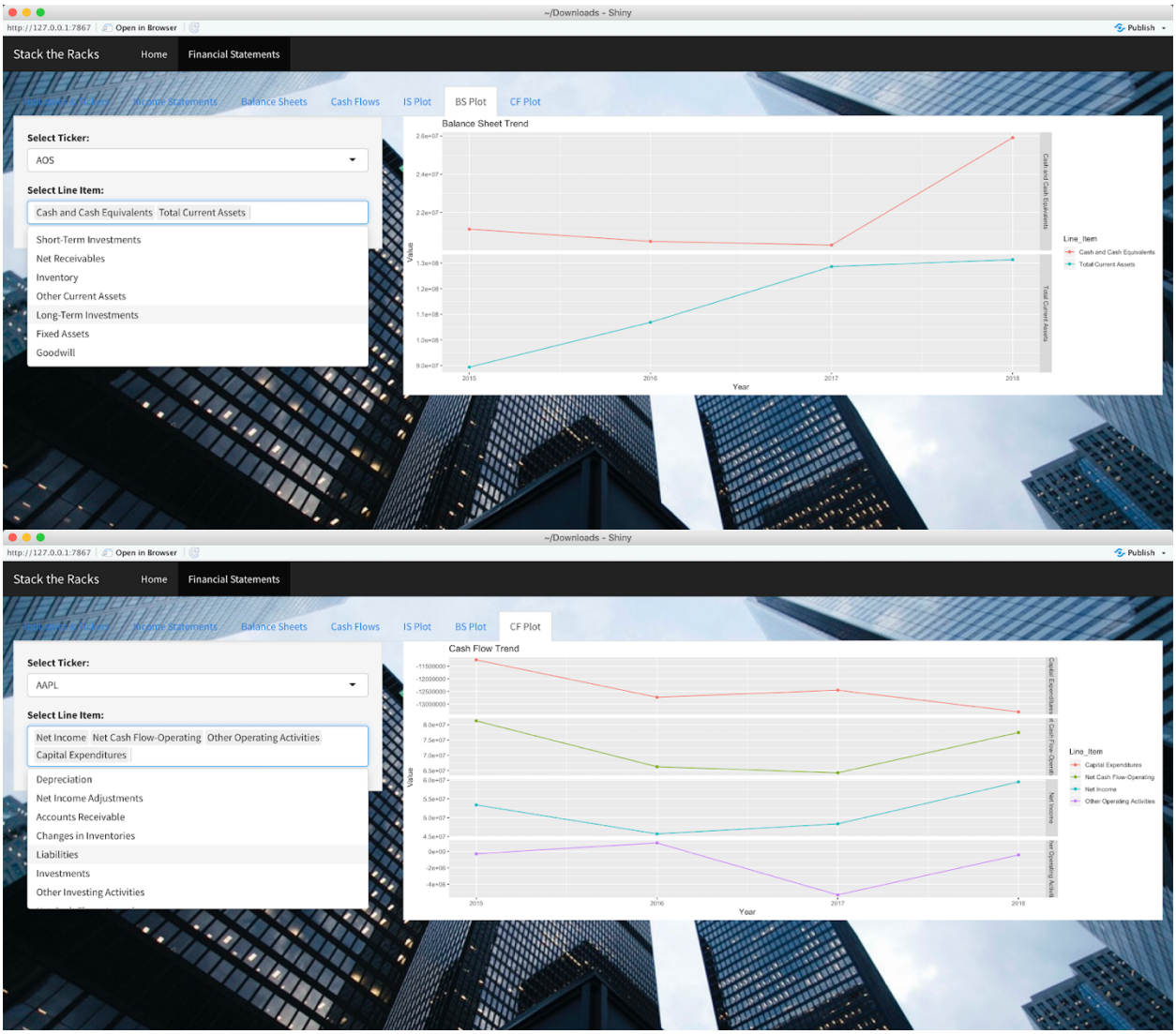


The below shows the relationships between industries and tickers for all companies in the S&P 500. The user can choose which stocks by using the drop-down menu to choose industries or tickers.



The below display the linear graphs of each chosen line item against each financial statement.





Insights

This project allowed us to take a glimpse into the data science process, and how proper cleaning, scraping, and analysis procedures leads to more concise and accurate results. Nonetheless, we realize that there is huge room for improvements.

Given that a valuation on a stock has much more to do with the financial statement data from Yahoo Finance and Nasdaq, we realize that there is a huge room for improvement for this project in terms of accuracy and scale. Many of these factors involve news and politics, which can develop at speeds that a quarterly report cannot suffice. The price of a stock is not a static value, and a more in-depth analysis on the data would bring this project to the next level, onto the level of professional stocker traders. These include finding the income statement reports from companies themselves and analyzing sentiments of the most recent news articles for each company. Such a scale of analysis is beyond the scope of this course, but would definitely be a good project to work on in the future.

Scraper Functions

Moustafa Abdelaziz, Erik Wong, Luke Hong

4/16/2019

Justification for Techniques Used through out the project:

There are several things of discussion: Our use of both a file system and sqlite database, our linear regression model, and explanation of data visuals.

We used a file system to keep track of data because it took much to long to have to gather data each time, and it was easy to see our data within a csv file rather than loading the database in Rmd and using a query to see the data.

Our linear regression model uses a complex and simple model. This is because N/A's will render the complex model useless. We suspect that some companies may not have all the data for certain columns and some room for improvement would be to look at some more places that have NA values and exclude them when we build up our model again. This may include Dividend data, as some companies do not provide dividends.

Our data visuals will track selected items over a period of time. This is to help the user see how a company has been doing. It is hard to have precise explanations, since each company may have had different product releases, news, or general economic trends that may effect these historic data.

```
library(rvest)
```

```
## Loading required package: xml2
```

```
library(dplyr)
```

```
##
## 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
```

```
library(RSQLite)
library(distr)
```

```
## Warning: package 'distr' was built under R version 3.5.2
```

```
## Loading required package: startupmsg
```

```
## Warning: package 'startupmsg' was built under R version 3.5.2
```

```
## Utilities for Start-Up Messages (version 0.9.6)
```

```
## For more information see ?"startupmsg", NEWS("startupmsg")
```

```
## Loading required package: sfsmisc
```

```
##
## Attaching package: 'sfsmisc'
```

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

```
## Object Oriented Implementation of Distributions (version 2.8.0)

## Attention: Arithmetics on distribution objects are understood as operations on corresponding random v
## Some functions from package 'stats' are intentionally masked ---see distrMASK().
## Note that global options are controlled by distroptions() ---c.f. ?"distroptions".

## For more information see ?"distr", NEWS("distr"), as well as
##   http://distr.r-forge.r-project.org/
## Package "distrDoc" provides a vignette to this package as well as to several extension packages; try

##
## Attaching package: 'distr'

## The following objects are masked from 'package:dplyr':
##
##     location, n

## The following objects are masked from 'package:stats':
##
##     df, qqplot, sd
```

```
income_statement_by_ticker <- function(ticker) {

  page <- read_html(paste(paste("https://www.nasdaq.com/symbol/", ticker, sep=""), "/financials", sep="
  income_statement <- html_text(html_nodes(page, "td+ td , .td_genTable+ td , th"))

  income_df <- data.frame(income_statement)
  income_df <- data.frame(income_df[2:nrow(income_df),])
  names(income_df) <- c("Values")
  income_df <- income_df[income_df != ""]
  income_df <- income_df[income_df != "Operating Expenses"]

  company_financials <- data.frame(ticker = double(),
                                   "2018" = double(),
                                   "2017" = double(),
                                   "2016" = double(),
                                   "2015"= double())

  if (length(income_df) > 2) {
    for (val in seq(from=6, to=length(income_df), by=5)) {
      row <- c(as.character(income_df[val]), income_df[val+1], income_df[val+2], income_df[val+3], incon
      company_financials[nrow(company_financials)+1,] <- row }

    colnames(company_financials)[1] <- ticker
  }

  company_financials
}

balance_sheet_by_ticker <- function(ticker) {

  page <- read_html(paste(paste("https://www.nasdaq.com/symbol/", ticker, sep=""), "/financials?query=ba
  balance_sheet <- html_text(html_nodes(page, "td+ td , .td_genTable+ td , th"))

  balance_sheet_df <- data.frame(balance_sheet)
  balance_sheet_df <- data.frame(balance_sheet_df[2:nrow(balance_sheet_df),])
  names(balance_sheet_df) <- c("Values")
```

```
  balance_sheet_df <- balance_sheet_df[balance_sheet_df != ""]
  balance_sheet_df <- balance_sheet_df[balance_sheet_df != "Current Assets"]
  balance_sheet_df <- balance_sheet_df[balance_sheet_df != "Long-Term Assets"]
  balance_sheet_df <- balance_sheet_df[balance_sheet_df != "Current Liabilities"]
  balance_sheet_df <- balance_sheet_df[balance_sheet_df != "Stock Holders Equity"]


  company_financials <- data.frame(ticker = double(),
                                   "2018" = double(),
                                   "2017" = double(),
                                   "2016" = double(),
                                   "2015"= double())

  if (length(balance_sheet_df) > 2) {
    for (val in seq(from=6, to=length(balance_sheet_df), by=5)) {
      row <- c(as.character(balance_sheet_df[val]), balance_sheet_df[val+1], balance_sheet_df[val+2], ba
      company_financials[nrow(company_financials)+1,] <- row }

    colnames(company_financials)[1] <- ticker
  }

  company_financials
}

cash_flow_by_ticker <- function(ticker) {

  page <- read_html(paste(paste("https://www.nasdaq.com/symbol/", ticker, sep=""), "/financials?query=ca
  cash_flow <- html_text(html_nodes(page, "td+ td , .td_genTable+ td , th"))

  cash_flow_df <- data.frame(cash_flow)
  cash_flow_df <- data.frame(cash_flow_df[2:nrow(cash_flow_df),])
  names(cash_flow_df) <- c("Values")
  cash_flow_df <- cash_flow_df[cash_flow_df != ""]
  cash_flow_df <- cash_flow_df[cash_flow_df != "Cash Flows-Operating Activities"]
  cash_flow_df <- cash_flow_df[cash_flow_df != "Changes in Operating Activities"]
  cash_flow_df <- cash_flow_df[cash_flow_df != "Cash Flows-Investing Activities"]
  cash_flow_df <- cash_flow_df[cash_flow_df != "Cash Flows-Financing Activities"]


  company_financials <- data.frame(ticker = double(),
                                   "2018" = double(),
                                   "2017" = double(),
                                   "2016" = double(),
                                   "2015"= double())

  if (length(cash_flow_df) > 2) {
    for (val in seq(from=6, to=length(cash_flow_df), by=5)) {
      row <- c(as.character(cash_flow_df[val]), cash_flow_df[val+1], cash_flow_df[val+2], cash_flow_df[v
      company_financials[nrow(company_financials)+1,] <- row }

    colnames(company_financials)[1] <- ticker
  }
```

```
  company_financials
}

# Results of scraping data for Apple
income_statement_by_ticker("aapl")
```

```
##                                           aapl        X2018
## 1                                Total Revenue $265,595,000
## 2                              Cost of Revenue $163,756,000
## 3                                Gross Profit $101,839,000
## 4                      Research and Development  $14,236,000
## 5                     Sales, General and Admin.  $16,705,000
## 6                           Non-Recurring Items           $0
## 7                       Other Operating Items           $0
## 8                            Operating Income  $70,898,000
## 9                  Add'l income/expense items   $2,005,000
## 10            Earnings Before Interest and Tax  $72,903,000
## 11                           Interest Expense           $0
## 12                        Earnings Before Tax  $72,903,000
## 13                                 Income Tax  $13,372,000
## 14                           Minority Interest           $0
## 15 Equity Earnings/Loss Unconsolidated Subsidiary           $0
## 16                 Net Income-Cont. Operations  $59,531,000
## 17                                 Net Income  $59,531,000
## 18   Net Income Applicable to Common Shareholders  $59,531,000
##           X2017        X2016        X2015
## 1  $229,234,000 $215,639,000 $233,715,000
## 2  $141,048,000 $131,376,000 $140,089,000
## 3   $88,186,000  $84,263,000  $93,626,000
## 4   $11,581,000  $10,045,000   $8,067,000
## 5   $15,261,000  $14,194,000  $14,329,000
## 6            $0           $0           $0
## 7            $0           $0           $0
## 8   $61,344,000  $60,024,000  $71,230,000
## 9    $2,745,000   $1,348,000   $1,285,000
## 10  $64,089,000  $61,372,000  $72,515,000
## 11           $0           $0           $0
## 12  $64,089,000  $61,372,000  $72,515,000
## 13  $15,738,000  $15,685,000  $19,121,000
## 14           $0           $0           $0
## 15           $0           $0           $0
## 16  $48,351,000  $45,687,000  $53,394,000
## 17  $48,351,000  $45,687,000  $53,394,000
## 18  $48,351,000  $45,687,000  $53,394,000
```

```
balance_sheet_by_ticker("aapl")
```

```
##                                         aapl        X2018
## 1                 Cash and Cash Equivalents  $25,913,000
## 2                    Short-Term Investments  $40,388,000
## 3                           Net Receivables  $48,995,000
## 4                                 Inventory   $3,956,000
## 5                     Other Current Assets  $12,087,000
## 6                     Total Current Assets $131,339,000
```

```
## 7                                Long-Term Investments $170,799,000
## 8                                         Fixed Assets  $41,304,000
## 9                                             Goodwill           $0
## 10                                    Intangible Assets           $0
## 11                                         Other Assets  $22,283,000
## 12                                   Deferred Asset Charges           $0
## 13                                         Total Assets $365,725,000
## 14                                   Accounts Payable  $55,888,000
## 15 Short-Term Debt / Current Portion of Long-Term Debt  $20,748,000
## 16                             Other Current Liabilities  $40,230,000
## 17                             Total Current Liabilities $116,866,000
## 18                                       Long-Term Debt  $93,735,000
## 19                                    Other Liabilities  $45,180,000
## 20                               Deferred Liability Charges   $2,797,000
## 21                                         Misc. Stocks           $0
## 22                                    Minority Interest           $0
## 23                                    Total Liabilities $258,578,000
## 24                                        Common Stocks  $40,201,000
## 25                                      Capital Surplus           $0
## 26                                    Retained Earnings  $70,400,000
## 27                                       Treasury Stock           $0
## 28                                         Other Equity ($3,454,000)
## 29                                         Total Equity $107,147,000
## 30                               Total Liabilities & Equity $365,725,000
##           X2017        X2016        X2015
## 1   $20,289,000  $20,484,000  $21,120,000
## 2   $53,892,000  $46,671,000  $20,481,000
## 3   $35,673,000  $29,299,000  $30,343,000
## 4    $4,855,000   $2,132,000   $2,349,000
## 5   $13,936,000   $8,283,000  $15,085,000
## 6  $128,645,000 $106,869,000  $89,378,000
## 7  $194,714,000 $170,430,000 $164,065,000
## 8   $33,783,000  $27,010,000  $22,471,000
## 9            $0   $5,414,000   $5,116,000
## 10           $0   $3,206,000   $3,893,000
## 11  $18,177,000   $8,757,000   $5,422,000
## 12           $0            $0           $0
## 13 $375,319,000 $321,686,000 $290,345,000
## 14  $44,242,000  $59,321,000  $60,671,000
## 15  $18,473,000  $11,605,000  $10,999,000
## 16  $38,099,000   $8,080,000   $8,940,000
## 17 $100,814,000  $79,006,000  $80,610,000
## 18  $97,207,000  $75,427,000  $53,329,000
## 19  $40,415,000  $36,074,000  $33,427,000
## 20   $2,836,000   $2,930,000   $3,624,000
## 21           $0            $0           $0
## 22           $0            $0           $0
## 23 $241,272,000 $193,437,000 $170,990,000
## 24  $35,867,000  $31,251,000  $27,416,000
## 25           $0            $0           $0
## 26  $98,330,000  $96,364,000  $92,284,000
## 27           $0            $0           $0
## 28   ($150,000)     $634,000   ($345,000)
## 29 $134,047,000 $128,249,000 $119,355,000
```

```
## 30 $375,319,000 $321,686,000 $290,345,000
```

```
cash_flow_by_ticker("aapl")
```

```
##                          aapl         X2018         X2017         X2016
## 1                  Net Income   $59,531,000   $48,351,000   $45,687,000
## 2                Depreciation   $10,903,000   $10,157,000   $10,505,000
## 3      Net Income Adjustments ($27,694,000)   $10,640,000    $9,634,000
## 4         Accounts Receivable ($13,332,000)  ($6,347,000)      $476,000
## 5      Changes in Inventories      $828,000  ($2,723,000)      $217,000
## 6  Other Operating Activities    ($423,000)  ($5,318,000)    $1,055,000
## 7                 Liabilities   $47,621,000    $9,465,000  ($1,343,000)
## 8     Net Cash Flow-Operating   $77,434,000   $64,225,000   $66,231,000
## 9        Capital Expenditures ($13,313,000) ($12,451,000) ($12,734,000)
## 10                Investments   $30,845,000 ($33,542,000) ($32,022,000)
## 11 Other Investing Activities  ($1,466,000)    ($453,000)  ($1,221,000)
## 12   Net Cash Flows-Investing   $16,066,000 ($46,446,000) ($45,977,000)
## 13 Sale and Purchase of Stock ($72,069,000) ($32,345,000) ($29,227,000)
## 14             Net Borrowings      $432,000   $29,014,000   $22,057,000
## 15 Other Financing Activities  ($2,527,000)  ($1,874,000)  ($1,570,000)
## 16   Net Cash Flows-Financing ($87,876,000) ($17,974,000) ($20,890,000)
## 17    Effect of Exchange Rate            $0            $0            $0
## 18              Net Cash Flow    $5,624,000    ($195,000)    ($636,000)
##            X2015
## 1    $53,394,000
## 2    $11,257,000
## 3     $5,353,000
## 4   ($3,318,000)
## 5     ($238,000)
## 6     ($283,000)
## 7    $15,101,000
## 8    $81,266,000
## 9  ($11,247,000)
## 10 ($44,417,000)
## 11    ($610,000)
## 12 ($56,274,000)
## 13 ($34,710,000)
## 14   $29,305,000
## 15  ($1,499,000)
## 16 ($17,716,000)
## 17            $0
## 18    $7,276,000
```

Compiles all Tickers from the S&P 500 and their relative industries.

```
ticker_url <- read_html("https://en.wikipedia.org/wiki/List_of_S%26P_500_companies")
ticker_and_industry_list <- html_text(html_nodes(ticker_url, "#constituents td:nth-child(2) , #constitu

ticker_and_industry <- data.frame("Ticker" = double(),
                                  "Industry" = double())

for (val in seq(from=1, to=length(ticker_and_industry_list), by=2)) {

  row <- c(as.character(ticker_and_industry_list[val]), ticker_and_industry_list[val+1])
  ticker_and_industry[nrow(ticker_and_industry)+1,] <- row }
```

```
#ticker_and_industry
```

```
db <- dbConnect(SQLite(), dbname = "financial_statements.sqlite")


d <- Truncate(Weibull(shape=2, scale=30), lower=0, upper=5)

income_statement_df <- data.frame("Ticker" = double(),
                                  "Line_Item" = double(),
                                  "X2018" = double(),
                                  "X2017" = double(),
                                  "X2016" = double(),
                                  "X2015" = double())

balance_sheet_df <- data.frame("Ticker" = double(),
                               "Line_Item" = double(),
                               "X2018" = double(),
                               "X2017" = double(),
                               "X2016" = double(),
                               "X2015" = double())

cash_flow_df <- data.frame("Ticker" = double(),
                           "Line_Item" = double(),
                           "X2018" = double(),
                           "X2017" = double(),
                           "X2016" = double(),
                           "X2015" = double())



for (val in ticker_and_industry$Ticker) {

  if (file.exists(paste(val, ".csv", sep=""))) {

    income_statement <- read.csv(paste(val, ".csv", sep=""), TRUE, ",")
    new_income <- cbind(rep(toupper(val), nrow(income_statement)), income_statement)

    names(new_income) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    income_statement_df <- rbind(income_statement_df, new_income)

    dbWriteTable(conn = db, name = paste(val, "_income_statement", sep=""), value = new_income, row.name
  }
  else {
    income_statement <- income_statement_by_ticker(val)
    # write.csv(income_statement, file = paste(val, ".csv", sep=""), row.names=FALSE)
    new_income <- cbind(rep(toupper(val), nrow(income_statement)), income_statement)

    names(new_income) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    income_statement_df <- rbind(income_statement_df, new_income)

    dbWriteTable(conn = db, name = paste(val, "_income_statement", sep=""), value = new_income, row.name
  }
```

```
  if (file.exists(paste(val, "_balance.csv", sep=""))) {
    balance_sheet <- read.csv(paste(val, "_balance.csv", sep=""), TRUE, ",")
    new_balance <- cbind(rep(toupper(val), nrow(balance_sheet)), balance_sheet )

    names(new_balance) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    balance_sheet_df <- rbind(balance_sheet_df, new_balance)


    dbWriteTable(conn = db, name = paste(val, "_balance_sheet", sep=""), value = new_balance, row.names

  }
  else {
    balance_sheet <- balance_sheet_by_ticker(val)
    # write.csv(balance_sheet, file = paste(val, "_balance.csv", sep=""), row.names=FALSE)
    new_balance <- cbind(rep(toupper(val), nrow(balance_sheet)), balance_sheet)

    names(new_balance) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    balance_sheet_df <- rbind(balance_sheet_df, new_balance)

    dbWriteTable(conn = db, name = paste(val, "_balance_sheet", sep=""), value = new_balance, row.names
  }


  if (file.exists(paste(val, "_cashflow.csv", sep=""))) {

    cash_flows <- read.csv(paste(val, "_cashflow.csv", sep=""), TRUE, ",")
    new_cash <- cbind(rep(toupper(val), nrow(cash_flows)), cash_flows )
    names(new_cash) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    cash_flow_df <- rbind(cash_flow_df, new_cash)

    dbWriteTable(conn = db, name = paste(val, "_cash_flows", sep=""), value = new_cash, row.names = FALS

  }
  else {
    cash_flows <- cash_flow_by_ticker(val)
    # write.csv(cash_flows, file = paste(val, "_cashflow.csv", sep=""), row.names=FALSE)
    new_cash <- cbind(rep(toupper(val), nrow(cash_flows)), cash_flows )
    names(new_cash) <- c("Ticker", "Line_Item", "X2018", "X2017", "X2016", "X2015")
    cash_flow_df <- rbind(cash_flow_df, new_cash)

    dbWriteTable(conn = db, name = paste(val, "_cash_flows", sep=""), value = new_cash, row.names = FALS
  }



  #Wrote csv files
  #write.csv(new_df, file = paste(val, ".csv", sep=""), row.names=FALSE)
  #Sys.sleep(d@r(1))

  #created tables based on the written files

  # }
}
```

```
dbWriteTable(conn = db, name = "income_statements", value = income_statement_df, row.names = FALSE, hea
dbWriteTable(conn = db, name = "balance_sheets", value = balance_sheet_df, row.names = FALSE, header = 
dbWriteTable(conn = db, name = "cash_flows", value = cash_flow_df, row.names = FALSE, header = TRUE, ov

write.csv(dbGetQuery(db, "SELECT * FROM income_statements"), "compiled_income_statements.csv", row.name
write.csv(dbGetQuery(db, "SELECT * FROM balance_sheets"), "compiled_balance_sheets.csv", row.names = FA
write.csv(dbGetQuery(db, "SELECT * FROM cash_flows"), "compiled_cash_flows.csv", row.names = FALSE)
```

```
key_stats <- data.frame(matrix(ncol=0, nrow=0))

for (val in ticker_and_industry$Ticker) {

  stats_url <- read_html(paste(paste(paste("https://finance.yahoo.com/quote/", val, sep=""), "/key-stati

  stats <- html_nodes(stats_url, "td")
  clean_stats <- html_text(stats)

  price_url <- read_html(paste("https://www.nasdaq.com/symbol/", val, sep=""))
  price <- clean_number(html_text(html_nodes(price_url, "#qwidget_lastsale")))

  stats_df <- data.frame("Stat" = double(),
                         "Value" = double())

  for (tick in seq(1, length(clean_stats), 2)) {
    row <- c(as.character(clean_stats[tick]), clean_value(clean_stats[tick+1]))
    stats_df[nrow(stats_df)+1,] <- row
  }

  stats_df <- stats_df[stats_df$Stat != "Fiscal Year Ends ",]
  stats_df <- stats_df[stats_df$Stat != "Most Recent Quarter (mrq)",]
  stats_df <- stats_df[stats_df$Stat != "Dividend Date 3",]
  stats_df <- stats_df[stats_df$Stat != "Ex-Dividend Date 4",]
  stats_df <- stats_df[stats_df$Stat != "Last Split Date 3",]

  spread_df <- spread(stats_df, "Stat", "Value")
  if (nrow(key_stats) > 0) {
    if (same_names(spread_df, key_stats)) {
      new_stats_df <- cbind(ticker = val, cbind(price, spread_df))
      key_stats <- rbind(key_stats, new_stats_df)
    }
  } else {
    new_stats_df <- cbind(ticker = val, cbind(price, spread_df))
    key_stats <- rbind(key_stats, new_stats_df)
  }
}

for (val in 2:ncol(key_stats)) {

  for (row in 1:nrow(key_stats)) {
    key_stats[[val]][[row]] <- clean_value(as.character(key_stats[[val]][[row]]))
  }
}
```

```
key_stats
```

Gets the Tickers for the Russel 1000

```
russell_ticker_url <- read_html("https://en.wikipedia.org/wiki/Russell_1000_Index")
russell_ticker_list <- html_text(html_nodes(russell_ticker_url, "td+ td"))

russell_ticker_list <- russell_ticker_list[51:length(russell_ticker_list)]
for (val in 1:length(russell_ticker_list)) {
  russell_ticker_list[val] <- substr(russell_ticker_list[val], 0, nchar(russell_ticker_list[val])-1)
}


russell_df <- data.frame("Ticker" = double())
for (val in russell_ticker_list) {
  russell_df[nrow(russell_df)+1,] <- c(val)
}
```

Removes Less common data

```
write.csv(key_stats, file = "russell_clean_key_stats.csv", row.names = FALSE)


library(dplyr)
key_stats <- dplyr::select(key_stats, -c("5 Year Average Dividend Yield 4", "Forward Annual Dividend Ra



# russel_ticker_list
# key_stats
```

Project

Moustafa Abdelaziz, Erik Wong, Luke Hong

4/7/2019

```
options(warn=-1)
library(rvest)
```

```
## Loading required package: xml2
```

```
library(xml2)
library(magrittr)
library(dplyr)
```

```
##
## 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
```

```
library(stringr)
require(distr)
```

```
## Loading required package: distr
```

```
## Loading required package: startupmsg
```

```
## Utilities for Start-Up Messages (version 0.9.6)
```

```
## For more information see ?"startupmsg", NEWS("startupmsg")
```

```
## Loading required package: sfsmisc
```

```
##
## Attaching package: 'sfsmisc'
```

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

```
## Object Oriented Implementation of Distributions (version 2.8.0)
```

```
## Attention: Arithmetics on distribution objects are understood as operations on corresponding random v
## Some functions from package 'stats' are intentionally masked ---see distrMASK().
## Note that global options are controlled by distroptions() ---c.f. ?"distroptions".
```

```
## For more information see ?"distr", NEWS("distr"), as well as
##   http://distr.r-forge.r-project.org/
## Package "distrDoc" provides a vignette to this package as well as to several extension packages; try
```

```
##
## Attaching package: 'distr'
```

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

```
##     location, n
```

```
## The following objects are masked from 'package:stats':
##
##     df, qqplot, sd
```

```
library(RSQLite)
library(tidyverse)
```

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

```
## v ggplot2 3.1.0     v readr   1.3.1
## v tibble  2.0.0     v purrr   0.2.5
## v tidyr   0.8.2     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x tidyr::extract()        masks magrittr::extract()
## x dplyr::filter()         masks stats::filter()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## x sfsmisc::last()         masks dplyr::last()
## x distr::location()       masks dplyr::location()
## x distr::n()              masks dplyr::n()
## x purrr::pluck()          masks rvest::pluck()
## x purrr::set_names()      masks magrittr::set_names()
```

```
library(caret)
```

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

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

```
## The following object is masked from 'package:purrr':
##
##     lift
```

Function that creates income statement by ticker by scraping data from Nasdaq.

```
income_statement_by_ticker <- function(ticker) {

  #page used to scrape data regarding income statement values
  page <- read_html(paste(paste("https://www.nasdaq.com/symbol/", ticker, sep=""), "/financials", sep="
  income_statement <- html_text(html_nodes(page, "td+ td , .td_genTable+ td , th"))

  #
  income_df <- data.frame(income_statement)
  income_df <- data.frame(income_df[2:nrow(income_df),])
  names(income_df) <- c("Values")
  income_df <- income_df[income_df != ""]
  income_df <- income_df[income_df != "Operating Expenses"]

  company_financials <- data.frame(ticker = double(),
                                   "2018" = double(),
                                   "2017" = double(),
                                   "2016" = double(),
                                   "2015"= double())

  if (length(income_df) > 2) {
```

```
    for (val in seq(from=6, to=length(income_df), by=5)) {
      row <- c(as.character(income_df[val]), income_df[val+1], income_df[val+2], income_df[val+3], inco
      company_financials[nrow(company_financials)+1,] <- row }

    colnames(company_financials)[1] <- ticker
  }
  company_financials
}
```

The three compiled financial statements of all S&P 500 companies were scraped from Nasdaq and stored in a csv for easy access.

```
final_income <- read.csv("compiled_income_statements.csv", sep=",", TRUE)
final_balance <- read.csv("compiled_balance_sheets.csv", sep=",", TRUE)
final_cash <- read.csv("compiled_cash_flows.csv", sep=",", TRUE)
```

Functions used for cleaning our data from the compiled financial statements.

```
clean_value <- function(x) {

  if (!is.na(x)) {
    if (substr(x, nchar(x), nchar(x)) == "B") {
      x <- as.numeric(substr(x, 0, nchar(x) -1)) * 1000000000
    }

    else if (substr(x, nchar(x), nchar(x)) == "M") {
      x <- as.numeric(substr(x, 0, nchar(x) -1)) * 1000000
    }

    else if (substr(x, nchar(x), nchar(x)) == "k") {
      x <- as.numeric(substr(x, 0, nchar(x) -1)) * 1000
    }

    else if (substr(x, nchar(x), nchar(x)) == "%") {
      x <- as.numeric(substr(x, 0, nchar(x) -1)) / 100
    }
  }
  x
}

clean_number <- function(x) {
  x <- gsub("[,]", "", x)
  x <- gsub("[(]", "-", x)
  x <- gsub("[)]", "", x)
  x <- gsub("[$]", "", x)
  x <- as.double(x)
  x
}

same_names <- function(df1, df2) {

  boolean <- TRUE

  for (name1 in names(df1)) {
    if (!(name1 %in% names(df2))) {
```

```
      boolean <- FALSE
    }
  }

  boolean
}
```

Compiles all Tickers from the S&P 500 and their relative industries.

```
ticker_url <- read_html("https://en.wikipedia.org/wiki/List_of_S%26P_500_companies")
ticker_and_industry_list <- html_text(html_nodes(ticker_url, "#constituents td:nth-child(2) , #constitue

ticker_and_industry <- data.frame("Ticker" = double(),
                                  "Industry" = double())

for (val in seq(from=1, to=length(ticker_and_industry_list), by=2)) {

  row <- c(as.character(ticker_and_industry_list[val]), ticker_and_industry_list[val+1])
  ticker_and_industry[nrow(ticker_and_industry)+1,] <- row }

#ticker_and_industry
```

Complex linear regression model using data from the S&P 500 for both train and test sets.

```
clean_key_stats <- read.csv("clean_key_stats.csv", sep=",", stringsAsFactors = FALSE, check.names = FALS

clean_key_stats <- dplyr::select(clean_key_stats, -c("5 Year Average Dividend Yield 4", "Forward Annual

clean_key_stats <- clean_key_stats[complete.cases(clean_key_stats),]
clean_key_stats[clean_key_stats == "N/A"]  <- NA
clean_key_stats <- na.omit(clean_key_stats)
clean_key_stats <- dplyr::select(clean_key_stats,  -c("ticker"))
clean_key_stats <- dplyr::select(clean_key_stats, -c("S&P500 52-Week Change 3"))

for (i in 2:ncol(clean_key_stats)) {
  for (j in 1:nrow(clean_key_stats)) {
    clean_key_stats[[i]][[j]] <- clean_number(clean_key_stats[[i]][[j]])
  }
}
#clean_key_stats

clean_key_stats <- clean_key_stats%>% mutate_all(funs(type.convert(as.double(.))))

norm_clean_key_stats <- clean_key_stats

for (i in 3:ncol(norm_clean_key_stats)) {
  max1 <- as.numeric(max(norm_clean_key_stats[[i]]))
  min1 <- as.numeric(min(norm_clean_key_stats[[i]]))
  for (j in 1:nrow(norm_clean_key_stats)) {
    norm_clean_key_stats[[i]][[j]] <- (as.numeric(norm_clean_key_stats[[i]][[j]]) - min1) / (max1 - min
  }
}


#clean_key_stats
```

```r
sample_size <- floor(0.70 * nrow(norm_clean_key_stats))

classes <- sapply(norm_clean_key_stats, class)
set.seed(120)

new_set <- sample(seq_len(nrow(norm_clean_key_stats)), size = sample_size)
non_norm_train <- clean_key_stats[new_set, ]
initial_test <- norm_clean_key_stats[-new_set, ]
initial_train <- norm_clean_key_stats[new_set, ]
```

Complex model cont. Removes the least significant variables one at a time to ensure that only the most significant variables are considered and are not removed prematurely.

```r
updated_train <- initial_train
updated_test <- initial_test
train_control <- trainControl(method="repeatedcv", number=5, repeats=3)
has_insignificant_vars <- TRUE
while(has_insignificant_vars) {
  test_model <- caret::train(price~., data=updated_train, trControl=train_control, method="lm")

  prob <- data.frame(summary(test_model)$coefficients[,"Pr(>|t|)"])
  prob <- prob[1:nrow(prob),]
  prob <- data.frame(prob)

  variables <- summary(test_model)$coefficients[,0]

  variables <- as.data.frame(row.names(variables))

  prob <- cbind(variables, prob)

  names(prob) <- c("Variable", "Probability")
  prob
  prob <- data.frame(prob[2:nrow(prob),])
  max_p <- max(prob[,"Probability"])
  if (max_p <= 0.05) {
    has_insignificant_vars <- FALSE
  }
  else {
    insig_var <- prob[prob[,"Probability"] == max_p,1]
    updated_train <- dplyr::select(updated_train, -c(insig_var))
    updated_test <- dplyr::select(updated_test, -c(insig_var))
  }
}

#updated_train


# train the model
complex_model <- caret::train(price~., data=updated_train, trControl=train_control, method="lm")
summary(complex_model)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.790  -2.338  -0.379   2.190  21.884
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          14.241      1.765   8.067 3.65e-14
## `\\`% Held by Insiders 1\\``         -8.109      2.988  -2.714 0.007139
## `\\`200-Day Moving Average 3\\``  -1114.357    115.131  -9.679  < 2e-16
## `\\`50-Day Moving Average 3\\``    2114.407     61.080  34.617  < 2e-16
## `\\`52 Week High 3\\``              571.824     58.685   9.744  < 2e-16
## `\\`52 Week Low 3\\``               304.127     57.009   5.335 2.24e-07
## `\\`52-Week Change 3\\``             14.961      4.938   3.030 0.002722
## `\\`Book Value Per Share (mrq)\\``  -14.768      4.506  -3.278 0.001204
## `\\`Diluted EPS (ttm)\\``           -42.695      8.719  -4.897 1.81e-06
## `\\`Enterprise Value 3\\``         -232.597    117.782  -1.975 0.049455
## `\\`Enterprise Value/EBITDA 6\\``   -33.306      6.551  -5.084 7.53e-07
## `\\`Float \\``                      -14.508      5.596  -2.592 0.010129
## `\\`Market Cap (intraday) 5\\``     244.361    114.571   2.133 0.033971
## `\\`Operating Margin (ttm)\\``      -11.779      3.451  -3.413 0.000755
## `\\`Price/Sales (ttm)\\``            14.095      3.988   3.534 0.000492
## `\\`Return on Assets (ttm)\\``       -7.844      2.944  -2.665 0.008235
## `\\`Total Cash (mrq)\\``            -43.831     18.225  -2.405 0.016945
## `\\`Total Debt (mrq)\\``             54.562     22.995   2.373 0.018456
## `\\`Trailing P/E \\``               -19.398      9.171  -2.115 0.035459
##
## (Intercept)                         ***
## `\\`% Held by Insiders 1\\``        **
## `\\`200-Day Moving Average 3\\``    ***
## `\\`50-Day Moving Average 3\\``     ***
## `\\`52 Week High 3\\``              ***
## `\\`52 Week Low 3\\``               ***
## `\\`52-Week Change 3\\``            **
## `\\`Book Value Per Share (mrq)\\`` **
## `\\`Diluted EPS (ttm)\\``           ***
## `\\`Enterprise Value 3\\``          *
## `\\`Enterprise Value/EBITDA 6\\``   ***
## `\\`Float \\``                      *
## `\\`Market Cap (intraday) 5\\``     *
## `\\`Operating Margin (ttm)\\``      ***
## `\\`Price/Sales (ttm)\\``           ***
## `\\`Return on Assets (ttm)\\``      **
## `\\`Total Cash (mrq)\\``            *
## `\\`Total Debt (mrq)\\``            *
## `\\`Trailing P/E \\``               *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.11 on 236 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9994
## F-statistic: 2.179e+04 on 18 and 236 DF,  p-value: < 2.2e-16
```

```
prediction <- complex_model %>% predict(updated_test)
#prediction
```

The simple model takes the most significant variables in the initial screenining of the model.

```
library(caret)
library(tidyverse)
library(dplyr)

clean_key_stats <- read.csv("clean_key_stats3.csv", sep=",", stringsAsFactors = FALSE, check.names = FA
clean_key_stats <- clean_key_stats[complete.cases(clean_key_stats),]
clean_key_stats <- na.omit(clean_key_stats)
clean_key_stats[clean_key_stats == "N/A"]  <- NA
clean_key_stats <- na.omit(clean_key_stats)

for (i in 2:ncol(clean_key_stats)) {
  for (j in 1:nrow(clean_key_stats)) {
    clean_key_stats[[i]][[j]] <- clean_number(clean_key_stats[[i]][[j]])
  }
}

clean_key_stats$`200-Day Moving Average 3` <- as.numeric(as.character(clean_key_stats$`200-Day Moving A
clean_key_stats$`50-Day Moving Average 3` <- as.numeric(as.character(clean_key_stats$`50-Day Moving Aver
clean_key_stats$`52 Week High 3` <- as.numeric(as.character(clean_key_stats$`52 Week High 3`))
clean_key_stats$`52 Week Low 3` <- as.numeric(as.character(clean_key_stats$`52 Week Low 3`))
clean_key_stats$`Total Debt/Equity (mrq)` <- as.numeric(as.character(clean_key_stats$`Total Debt/Equity

for (i in 3:ncol(clean_key_stats)) {
  max1 <- as.numeric(max(clean_key_stats[[i]]))
  min1 <- as.numeric(min(clean_key_stats[[i]]))
  for (j in 1:nrow(clean_key_stats)) {
    clean_key_stats[[i]][[j]] <- (as.numeric(clean_key_stats[[i]][[j]]) - min1) / (max1 - min1)
  }
}
clean_key_stats <- dplyr::select(clean_key_stats, -c("S&P500 52-Week Change 3"))

clean_key_stats <- dplyr::select(clean_key_stats,  -c("ticker"))


sample_size <- floor(0.70 * nrow(clean_key_stats))

classes <- sapply(clean_key_stats, class)
set.seed(100)

new_set <- sample(seq_len(nrow(clean_key_stats)), size = sample_size)

key_stats_significant <- dplyr::select(clean_key_stats, c("price", "200-Day Moving Average 3", "50-Day

train <- key_stats_significant[new_set, ]
test <- key_stats_significant[-new_set, ]

simple_model <- lm(price ~., data = train)
```

Predicts a value using the complex model. If that value cannot be predicted due to missing data, the function

will use the simple model to predict the value.

```
value <- function(x) {

  key_stats <- data.frame(matrix(ncol=0, nrow=0))

  stats_url <- read_html(paste(paste(paste("https://finance.yahoo.com/quote/", x, sep=""), "/key-statis

  stats <- html_nodes(stats_url, "td")
  clean_stats <- html_text(stats)

  price_url <- read_html(paste("https://www.nasdaq.com/symbol/", x, sep=""))
  price <- clean_number(html_text(html_nodes(price_url, "#qwidget_lastsale")))

  stats_df <- data.frame("Stat" = double(), "Value" = double())

  for (tick in seq(1, length(clean_stats), 2)) {
    row <- c(as.character(clean_stats[tick]), clean_value(clean_stats[tick+1]))
    stats_df[nrow(stats_df)+1,] <- row
  }

  stats_df <- stats_df[stats_df$Stat != "Fiscal Year Ends ",]
  stats_df <- stats_df[stats_df$Stat != "Most Recent Quarter (mrq)",]
  stats_df <- stats_df[stats_df$Stat != "Dividend Date 3",]
  stats_df <- stats_df[stats_df$Stat != "Ex-Dividend Date 4",]
  stats_df <- stats_df[stats_df$Stat != "Last Split Date 3",]

  spread_df <- spread(stats_df, "Stat", "Value")
  if (nrow(key_stats) > 0) {
    if (same_names(spread_df, key_stats)) {
      new_stats_df <- cbind(ticker = x, cbind(price, spread_df))
      key_stats <- rbind(key_stats, new_stats_df)
    }
  } else {
    new_stats_df <- cbind(ticker = x, cbind(price, spread_df))
    key_stats <- rbind(key_stats, new_stats_df)
  }

  test_value <- key_stats
  for (val in 2:ncol(test_value)) {

    for (row in 1:nrow(test_value)) {
      test_value[[val]][[row]] <- clean_number(as.character(test_value[[val]][[row]]))
    }
  }


  test_value <- test_value[colnames(updated_train)]
  test_value <- test_value%>% mutate_all(funs(type.convert(as.double(.))))

  normalized_data <- rbind(non_norm_train[colnames(updated_train)]%>% mutate_all(funs(type.convert(as.d

  for (i in 2:ncol(normalized_data)) {
    max1 <- as.numeric(max(normalized_data[[i]]))
```

```
    min1 <- as.numeric(min(normalized_data[[i]]))
    for (j in 1:nrow(normalized_data)) {
      normalized_data[[i]][[j]] <- (as.numeric(normalized_data[[i]][[j]]) - min1) / (max1 - min1)
    }
  }

  new_val <- normalized_data[nrow(normalized_data),]


  c_predict <- predict(complex_model, new_val)

  if (length(c_predict) == 0) {

    c_predict <- predict(simple_model, new_val)
  }
  c_predict
}
```

Tests results to ensure validity.

```
#returns 183.41 compared to value at approx 180 (would issue a buy recommendation)
value("fb")
```

```
##      256
## 180.1412
```

```
#returns 199.45 compared to value at approx 199 (would issue a sell/hold recommendation)
value("aapl")
```

```
##      256
## 198.6363
```

```
#returns 32.54 compared to value at approx 35 (would issue a sell/hold recommendation)
value("twtr")
```

```
##      256
## 32.50376
```

```
#returns 286.68 compared to value at approx 267 (would issue a sell/hold recommendation)
value("tsla")
```

```
##      256
## 285.7245
```

Serves as a dictionary in conjunction with the S&P ticker list for RShiny queries.

```
russell_ticker_url <- read_html("https://en.wikipedia.org/wiki/Russell_1000_Index")
russell_ticker_list <- html_text(html_nodes(russell_ticker_url, "td+ td"))

russell_ticker_list <- russell_ticker_list[51:length(russell_ticker_list)]
for (val in 1:length(russell_ticker_list)) {
  russell_ticker_list[val] <- substr(russell_ticker_list[val], 0, nchar(russell_ticker_list[val])-1)
}


russell_df <- data.frame("Ticker" = double())
for (val in russell_ticker_list) {
  russell_df[nrow(russell_df)+1,] <- c(val)
```

```
}
```

Compares our predicted value against the current market price to provide the user awith a recommendation.

```
compare_vals <- function(x) {

  price_url <- read_html(paste("https://www.nasdaq.com/symbol/", x, sep=""))
  price <- clean_number(html_text(html_nodes(price_url, "#qwidget_lastsale")))
  recommendation <- if (value(x) > price) {
    recommendation <- "Buy"
  }
  else {
    recommendation <- "This Stock is Currently Over-Valued"
  }

  paste(paste(paste(paste("Estimated Current Value: ", round(value(x), 2), sep =""), paste(" | Current
}
```