Equity Research Tool

Nicholas Colonna, Nicole Lange, Kristina Redmond

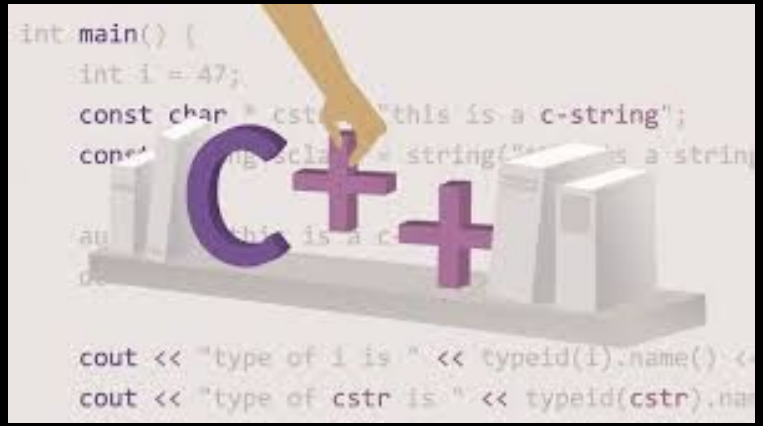
FE 522 - Spring 2019

Project Overview

- We developed a program that allows a user to quickly and easily generate a brief equity research report
- This program can help investors make informed investment decisions
- Students can utilize this tool to quickly pull data and research companies for their projects
- Input database is comprised of stocks in the S&P 500 that have existed for the past 5 years and Treasury yields
- Outputs:
  - Ratios
  - Chart of historical prices
  - Stock recommendation based on ratios

Utilizing What We Learned

- Import data from a .csv file
- Output information to a text file
- Output statements using cout
- Used different data types
- Utilized vectors to bring in the prices
- Used for and while loops
- Applied .h and C++ formulas
  - Pow()
  - Sqrt()
  - Sort()



Data Input

- C++ Yahoo Finance API was unfortunately discontinued in 2017
- Our alternative approach was an R script that pulls 5yrs of historical prices for the S&P 500 index, the stocks in that index, and Treasury yields
    - This script outputs a price_data.csv file that is then used in our C++ program. A sample from the database is seen below.

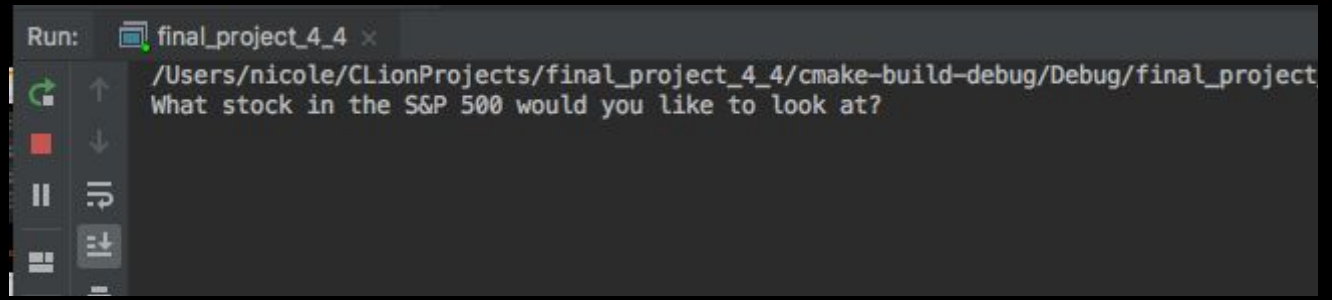
date	^GSPC	MMM	AOS	ABT	ABBV
4/30/2014	1883.95	122.6085	21.49004	34.83312	43.29713
5/1/2014	1883.68	124.1247	21.69686	34.77017	42.90639
5/2/2014	1881.14	123.5164	21.95882	34.67127	42.5489
5/5/2014	1884.66	123.9483	21.77958	34.92303	42.69854
5/6/2014	1867.72	122.7847	21.41651	34.68926	42.32445
5/7/2014	1878.21	124.4155	21.4211	34.78815	43.88739
5/8/2014	1875.63	124.1423	21.58196	34.82412	43.35532

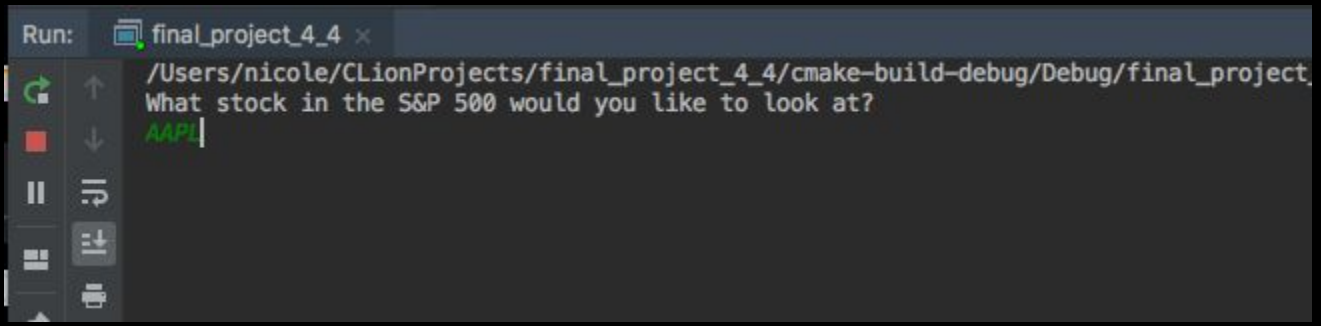
```r
#read ticker data in from csv as dataframe, then extract just the tickers
symbols <- read.csv(file="tickers.csv", header=TRUE, sep=",")
tickers <- symbols[,1]

#counts number of tickers used
numTickers <- length(tickers)

#start and end date of data pulled
startDate <- "2014-04-30"
endDate <- "2019-04-30"

#create zoo object to store historical data by doing just for the first ticker
price_data <- get.hist.quote(instrument = as.character(tickers[1]), start = startDate,
                             end = endDate, quote = "AdjClose", retclass = "zoo", quiet = T)

volume_data <- get.hist.quote(instrument = as.character(tickers[1]), start = startDate,
                              end = endDate, quote = "Volume", retclass = "zoo", quiet = T)

#set column name as the ticker
dimnames(price_data)[[2]] <- as.character(tickers[1])
dimnames(volume_data)[[2]] <- as.character(tickers[1])

#now that zoo object is created, loop through all tickers, saving historical data
for (i in 2:numTickers) {
  #Displays what number ticker its on
  cat("On company ", i, " out of ", numTickers , "\n")

  #try to pull price data, if error, try again. If 2 errors, skip completely
  p_data <- try(x <- get.hist.quote(instrument = as.character(tickers[i]), start = startDate,
                                    end = endDate, quote = "AdjClose", retclass = "zoo", quiet = T))
  if(class(p_data) == "try-error") {
    next
  }
  else {
    dimnames(x)[[2]] <- as.character(tickers[i])  #set column name to ticker
    #add the new stock to the database
    price_data <- merge(price_data, x)

    #try to pull volume data, if error, try again. If 2 errors, skip completely
    v_data <- try(y <- get.hist.quote(instrument = as.character(tickers[i]), start = startDate,
                                      end = endDate, quote = "Volume", retclass = "zoo", quiet = T))
    if(class(v_data) == "try-error") {
      next
    }
    else {
      dimnames(y)[[2]] <- as.character(tickers[i])  #set column name to ticker
      #add the new stock to the database
      volume_data <- merge(volume_data, y)

    }
  }
}

#write data to a csv file
write.zoo(price_data, file = "price_data.csv", index.name = "date")
write.zoo(volume_data, file = "volume_data.csv", index.name = "date")
```

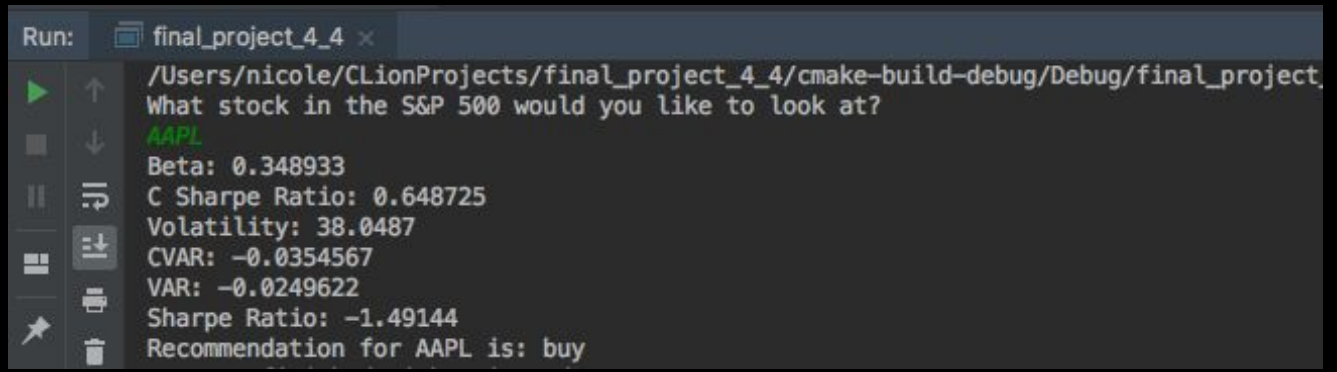
Usage

1. User runs the program
2. Enters desired stock's ticker symbol in all capital letters in terminal
3. Presses enter

Program outputs ratios & recommendation in terminal & also exports ratios to a .txt file







Functions Created in Program

- Mean
- Median
- Standard Deviation
- Stock Returns (daily, monthly, annual)
- Volatility
- Beta (daily)
- Sharpe Ratio (annual)
- C-Sharpe Ratio (annual)
- 95% VaR (daily, monthly)
- 95% cVaR (daily, monthly)

$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} = B(y,x) \qquad x>0, y>0$$

$$= \frac{R_p - R_f}{\sigma_p}$$

Function Example: Beta

```
double beta(vector<double> prices, vector<double> SP){
    vector<double> perc_change_stock;
    vector<double> perc_change_sp;
    double hold;
    for (int i = 1; i < prices.size(); i++){
        double temp = (prices[i] - prices[i-1])/prices[i-1];
        perc_change_stock.push_back(temp);
    }
    for (int i = 1; i < SP.size(); i++){
        double temp = (SP[i] - SP[i-1])/SP[i-1];
        perc_change_sp.push_back(temp);
    }
    double covar = covariance(perc_change_stock, perc_change_sp);
    double var = variance(perc_change_stock);
    hold = covar/var;
    return hold;
}
```

- Created a “for” loop to store stock and S&P returns in a double vector
- Utilized covariance and variance helper functions to calculate covariance of stock and S&P and variance of stock returns

Buy/Sell Recommendation

BUY: Sharpe ratio is greater than the average for all stocks in the S&P 500 **AND** if the average annual return is greater than the median for all stocks in the S&P 500.

SELL: otherwise

```
vector<double> stock_specific = load_new(database, stock_position);

double avg_sharpe = 0;
vector<double> annual_rets(database[0].size()-3);
for (int i=2; i< database[0].size()-1; i++){
    vector<double> stock = load_new(database, i);
    avg_sharpe += sharpe_ratio(stock);
    annual_rets[i-2] = expected_return(annual_returns(stock));
}
avg_sharpe = avg_sharpe / (database[0].size() - 3);
double median_rets = median(annual_rets);

string rec;
if(sharpe_ratio(stock_specific) >= avg_sharpe && expected_return(annual_returns(stock_specific)) >= median_rets){
    rec = "Buy";
}else{
    rec = "Sell";
}
return rec;
```

Program Output

```
What S&P 500 stock would you like to research?
AAPL
Ratios and Recommendations for AAPL
Closing Price on 4/11/2019: $198.95
Average Annual Return: 28.306%
Beta: 1.19147
Volatility: 38.0487
Sharpe Ratio: 0.880176
C Sharpe Ratio: -0.00399099
VAR: -0.0249622
CVAR: -0.0354567
Recommendation for AAPL is: Buy
Research report file AAPL.txt has been created.

Would you like to research another stock? (0 for Yes, 1 for No): 0

What S&P 500 stock would you like to research?
WFC
Ratios and Recommendations for WFC
Closing Price on 4/11/2019: $47.74
Average Annual Return: 2.97645%
Beta: 1.04837
Volatility: 4.3074
Sharpe Ratio: 0.0588433
C Sharpe Ratio: -5.87242e-05
VAR: -0.0203715
CVAR: -0.0288562
Recommendation for WFC is: Sell
Research report file WFC.txt has been created.

Would you like to research another stock? (0 for Yes, 1 for No): 1
```

AAPL.txt

```
Ratios and Recommendations for AAPL
Closing Price on 4/11/2019: $198.95
Average Annual Return: 28.306%
Beta: 1.19147
Volatility: 38.0487
Sharpe Ratio: 0.880176
C Sharpe Ratio: -0.00399099
VAR: -0.0249622
CVAR: -0.0354567
Recommendation for AAPL is: Buy
```

WFC.txt

```
Ratios and Recommendations for WFC
Closing Price on 4/11/2019: $47.74
Average Annual Return: 2.97645%
Beta: 1.04837
Volatility: 4.3074
Sharpe Ratio: 0.0588433
C Sharpe Ratio: -5.87242e-05
VAR: -0.0203715
CVAR: -0.0288562
Recommendation for WFC is: Sell
```

Limitations

- Only used historical prices to calculate ratios
  - Future versions could take in .csv file databases with accounting data and produce ratios using this information (P/E, current ratio, ROA, ROE)
- Plotting in C++ is challenging due to lack of an easy plotting library (i.e. Matplotlib.pyplot in python)
- Recommendations function only uses two ratios
  - Adding financial statement data would provide additional ratios that would enhance the value of our recommendation

Conclusion

- Easily gather and calculate a variety of information on a given stock
- Quick report generation
- Gives baseline recommendation for users to go off of
- Internet is not a requirement to calculate information, since database is local
    - Internet access only needed to run R script to pull stock data
- No cost or fees

Questions?