**241 Project Report**

Caleb Kipp, Svens Daukss, Ujjwal Ujjwal, James Mullins

**Repository:** [SvensDaukss/SPY-Options-Analysis (github.com)](https://github.com/SvensDaukss/SPY-Options-Analysis)

**Source Code Raw:**

**Data analysis:**

#include <stdio.h>

#include "dataAnalysis.h"

// ################################### Put Volumes | Call Volumes | Average Put/Call Ratio ###################################

void analyzeData(MarketData\* data, int dataSize) {

if (dataSize <= 0) return;

// Average Put Volume 2010-2019

float totalPutVolume2010To2019 = 0;

for (int i = 0; i <= dataSize; i++) {

totalPutVolume2010To2019 += data[i].putVolume;

}

float averageTotalPutVolume2010To2019 = totalPutVolume2010To2019 / dataSize;

printf("\nAverage Total Put Volume 2010-2019: %.0f\n", averageTotalPutVolume2010To2019);

// Average Call Volume 2010-2019

float totalCallVolume2010To2019 = 0;

for (int i = 0; i <= dataSize; i++) {

totalCallVolume2010To2019 += data[i].callVolume;

}

float averageTotalCallVolume2010To2019 = totalCallVolume2010To2019 / dataSize;

printf("Average Total Call Volume 2010-2019: %.0f\n", averageTotalCallVolume2010To2019);

// Average Put/Call Ratio 2010-2019

float totalPutCallRatio2010To2019 = 0;

for (int i = 0; i <= dataSize; i++) {

totalPutCallRatio2010To2019 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2010To2019 = totalPutCallRatio2010To2019 / dataSize;

printf("Average Put/Call Ratio From 2010-2019: %.3f\n", averageTotalPutCallRatio2010To2019);

// Average Put Volume 2010

float totalPutVolume2010 = 0;

for (int i = 0; i <= 126; i++) {

totalPutVolume2010 += data[i].putVolume;

}

float averageTotalPutVolume2010 = totalPutVolume2010 / 126;

printf("\nAverage Total Put Volume in 2010: %.0f\n", averageTotalPutVolume2010);

// Average Call Volume 2010

float totalCallVolume2010 = 0;

for (int i = 0; i <= 126; i++) {

totalCallVolume2010 += data[i].callVolume;

}

float averageTotalCallVolume2010 = totalCallVolume2010 / 126;

printf("Average Total Call Volume in 2010: %.0f\n", averageTotalCallVolume2010);

// Average Put/Call Ratio 2010

float totalPutCallRatio2010 = 0;

for (int i = 0; i <= 126; i++) {

totalPutCallRatio2010 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2010 = totalPutCallRatio2010 / 126;

printf("Average Put/Call Ratio in 2010: %.3f\n", averageTotalPutCallRatio2010);

// Average Put Volume 2011

float totalPutVolume2011 = 0;

for (int i = 127; i <= 378; i++) {

totalPutVolume2011 += data[i].putVolume;

}

float averageTotalPutVolume2011 = totalPutVolume2011 / 251;

printf("\nAverage Total Put Volume in 2011: %.0f\n", averageTotalPutVolume2011);

// Average Call Volume 2011

float totalCallVolume2011 = 0;

for (int i = 127; i <= 378; i++) {

totalCallVolume2011 += data[i].callVolume;

}

float averageTotalCallVolume2011 = totalCallVolume2011 / 251;

printf("Average Total Call Volume in 2011: %.0f\n", averageTotalCallVolume2011);

// Average Put/Call Ratio 2011

float totalPutCallRatio2011 = 0;

for (int i = 127; i <= 378; i++) {

totalPutCallRatio2011 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2011 = totalPutCallRatio2011 / 251;

printf("Average Put/Call Ratio in 2011: %.3f\n", averageTotalPutCallRatio2011);

// Average Put Volume 2012

float totalPutVolume2012 = 0;

for (int i = 379; i <= 628; i++) {

totalPutVolume2012 += data[i].putVolume;

}

float averageTotalPutVolume2012 = totalPutVolume2012 / 249;

printf("\nAverage Total Put Volume in 2012: %.0f\n", averageTotalPutVolume2012);

// Average Call Volume 2012

float totalCallVolume2012 = 0;

for (int i = 379; i <= 628; i++) {

totalCallVolume2012 += data[i].callVolume;

}

float averageTotalCallVolume2012 = totalCallVolume2012 / 249;

printf("Average Total Call Volume in 2012: %.0f\n", averageTotalCallVolume2012);

// Average Put/Call Ratio 2012

float totalPutCallRatio2012 = 0;

for (int i = 379; i <= 628; i++) {

totalPutCallRatio2012 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2012 = totalPutCallRatio2012 / 249;

printf("Average Put/Call Ratio in 2012: %.3f\n", averageTotalPutCallRatio2012);

// Average Put Volume 2013

float totalPutVolume2013 = 0;

for (int i = 629; i <= 880; i++) {

totalPutVolume2013 += data[i].putVolume;

}

float averageTotalPutVolume2013 = totalPutVolume2013 / 251;

printf("\nAverage Total Put Volume in 2013: %.0f\n", averageTotalPutVolume2013);

// Average Call Volume 2013

float totalCallVolume2013 = 0;

for (int i = 629; i <= 880; i++) {

totalCallVolume2013 += data[i].callVolume;

}

float averageTotalCallVolume2013 = totalCallVolume2013 / 251;

printf("Average Total Call Volume in 2013: %.0f\n", averageTotalCallVolume2013);

// Average Put/Call Ratio 2013

float totalPutCallRatio2013 = 0;

for (int i = 629; i <= 880; i++) {

totalPutCallRatio2013 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2013 = totalPutCallRatio2013 / 251;

printf("Average Put/Call Ratio in 2013: %.3f\n", averageTotalPutCallRatio2013);

// Average Put Volume 2014

float totalPutVolume2014 = 0;

for (int i = 881; i <= 1132; i++) {

totalPutVolume2014 += data[i].putVolume;

}

float averageTotalPutVolume2014 = totalPutVolume2014 / 251;

printf("\nAverage Total Put Volume in 2014: %.0f\n", averageTotalPutVolume2014);

// Average Call Volume 2014

float totalCallVolume2014 = 0;

for (int i = 881; i <= 1132; i++) {

totalCallVolume2014 += data[i].callVolume;

}

float averageTotalCallVolume2014 = totalCallVolume2014 / 251;

printf("Average Total Call Volume in 2014: %.0f\n", averageTotalCallVolume2014);

// Average Put/Call Ratio 2014

float totalPutCallRatio2014 = 0;

for (int i = 881; i <= 1132; i++) {

totalPutCallRatio2014 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2014 = totalPutCallRatio2014 / 251;

printf("Average Put/Call Ratio in 2014: %.3f\n", averageTotalPutCallRatio2014);

// Average Put Volume 2015

float totalPutVolume2015 = 0;

for (int i = 1133; i <= 1384; i++) {

totalPutVolume2015 += data[i].putVolume;

}

float averageTotalPutVolume2015 = totalPutVolume2015 / 251;

printf("\nAverage Total Put Volume in 2015: %.0f\n", averageTotalPutVolume2015);

// Average Call Volume 2015

float totalCallVolume2015 = 0;

for (int i = 1133; i <= 1384; i++) {

totalCallVolume2015 += data[i].callVolume;

}

float averageTotalCallVolume2015 = totalCallVolume2015 / 251;

printf("Average Total Call Volume in 2015: %.0f\n", averageTotalCallVolume2015);

// Average Put/Call Ratio 2015

float totalPutCallRatio2015 = 0;

for (int i = 1133; i <= 1384; i++) {

totalPutCallRatio2015 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2015 = totalPutCallRatio2015 / 251;

printf("Average Put/Call Ratio in 2015: %.3f\n", averageTotalPutCallRatio2015);

// Average Put Volume 2016

float totalPutVolume2016 = 0;

for (int i = 1385; i <= 1636; i++) {

totalPutVolume2016 += data[i].putVolume;

}

float averageTotalPutVolume2016 = totalPutVolume2016 / 251;

printf("\nAverage Total Put Volume in 2016: %.0f\n", averageTotalPutVolume2016);

// Average Call Volume 2016

float totalCallVolume2016 = 0;

for (int i = 1385; i <= 1636; i++) {

totalCallVolume2016 += data[i].callVolume;

}

float averageTotalCallVolume2016 = totalCallVolume2016 / 251;

printf("Average Total Call Volume in 2016: %.0f\n", averageTotalCallVolume2016);

// Average Put/Call Ratio 2016

float totalPutCallRatio2016 = 0;

for (int i = 1385; i <= 1636; i++) {

totalPutCallRatio2016 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2016 = totalPutCallRatio2016 / 251;

printf("Average Put/Call Ratio in 2016: %.3f\n", averageTotalPutCallRatio2016);

// Average Put Volume 2017

float totalPutVolume2017 = 0;

for (int i = 1637; i <= 1887; i++) {

totalPutVolume2017 += data[i].putVolume;

}

float averageTotalPutVolume2017 = totalPutVolume2017 / 250;

printf("\nAverage Total Put Volume in 2017: %.0f\n", averageTotalPutVolume2017);

// Average Call Volume 2017

float totalCallVolume2017 = 0;

for (int i = 1637; i <= 1887; i++) {

totalCallVolume2017 += data[i].callVolume;

}

float averageTotalCallVolume2017 = totalCallVolume2017 / 250;

printf("Average Total Call Volume in 2017: %.0f\n", averageTotalCallVolume2017);

// Average Put/Call Ratio 2017

float totalPutCallRatio2017 = 0;

for (int i = 1637; i <= 1887; i++) {

totalPutCallRatio2017 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2017 = totalPutCallRatio2017 / 250;

printf("Average Put/Call Ratio in 2017: %.3f\n", averageTotalPutCallRatio2017);

// Average Put Volume 2018

float totalPutVolume2018 = 0;

for (int i = 1888; i <= 2138; i++) {

totalPutVolume2018 += data[i].putVolume;

}

float averageTotalPutVolume2018 = totalPutVolume2018 / 250;

printf("\nAverage Total Put Volume in 2018: %.0f\n", averageTotalPutVolume2018);

// Average Call Volume 2018

float totalCallVolume2018 = 0;

for (int i = 1888; i <= 2138; i++) {

totalCallVolume2018 += data[i].callVolume;

}

float averageTotalCallVolume2018 = totalCallVolume2018 / 250;

printf("Average Total Call Volume in 2018: %.0f\n", averageTotalCallVolume2018);

// Average Put/Call Ratio 2018

float totalPutCallRatio2018 = 0;

for (int i = 1888; i <= 2138; i++) {

totalPutCallRatio2018 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2018 = totalPutCallRatio2018 / 250;

printf("Average Put/Call Ratio in 2018: %.3f\n", averageTotalPutCallRatio2018);

// Average Put Volume 2019

float totalPutVolume2019 = 0;

for (int i = 2139; i <= dataSize; i++) {

totalPutVolume2019 += data[i].putVolume;

}

float averageTotalPutVolume2019 = totalPutVolume2019 / (dataSize - 2139);

printf("\nAverage Total Put Volume in 2019: %.0f\n", averageTotalPutVolume2019);

// Average Call Volume 2019

float totalCallVolume2019 = 0;

for (int i = 2139; i <= dataSize; i++) {

totalCallVolume2019 += data[i].callVolume;

}

float averageTotalCallVolume2019 = totalCallVolume2019 / (dataSize - 2139);

printf("Average Total Call Volume in 2019: %.0f\n", averageTotalCallVolume2019);

// Average Put/Call Ratio 2019

float totalPutCallRatio2019 = 0;

for (int i = 2139; i <= dataSize; i++) {

totalPutCallRatio2019 += data[i].putCallRatio;

}

float averageTotalPutCallRatio2019 = totalPutCallRatio2019 / (dataSize - 2139);

printf("Average Put/Call Ratio in 2019: %.3f\n", averageTotalPutCallRatio2019);

}

**dataAnalysis.h:**#ifndef DATA\_ANALYSIS\_H

#define DATA\_ANALYSIS\_H

#include "dataParsing.h"

void analyzeData(MarketData\* data, int dataSize);

#endif

**dataParsing.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "dataParsing.h"

#define MAX\_ROW\_LENGTH 2332

void parseLine(char\* line, MarketData\* data) {

sscanf(line, "%10[^,],%f,%d,%d,%d", data->date, &data->putCallRatio, &data->putVolume, &data->callVolume, &data->totalOptionsVolume);

}

void printMarketData(const MarketData\* data, int rowNumber) {

printf(" %d | %s | %.2f | %d | %d | %d\n-----------------------------------------------------------------------------------------------------------------\n", rowNumber, data->date, data->putCallRatio, data->putVolume, data->callVolume, data->totalOptionsVolume);

}

void printMarketDataRange(const MarketData\* data, int dataSize, int startRow, int numberOfRowsToPrint) {

printf("-----------------------------------------------------------------------------------------------------------------\n # | Date | Put/Call Ratio | Put Volume | Call Volume | Total Options Volume \n-----------------------------------------------------------------------------------------------------------------\n");

for (int i = startRow; i < startRow + numberOfRowsToPrint && i < dataSize; i++) {

printMarketData(&data[i], i + 1);

}

}

void processLine(char\* line, MarketData\* data, int rowNumber) {

parseLine(line, data);

printMarketData(data, rowNumber);

}

void readCSVFile(const char\* filepath, MarketData\*\* data, int\* dataSize) {

FILE\* file = fopen(filepath, "r");

if (!file) {

perror("Error Opening File");

return;

} else {

printf("File Opened Successfully: %s\n\n", filepath);

}

// This is the logic to ignore the first 2 rows in spxpc.csv

char line[MAX\_ROW\_LENGTH];

fgets(line, MAX\_ROW\_LENGTH, file);

fgets(line, MAX\_ROW\_LENGTH, file);

int count = 0;

while (fgets(line, MAX\_ROW\_LENGTH, file)) {

\*data = realloc(\*data, (count + 1) \* sizeof(MarketData));

if (\*data == NULL) {

fprintf(stderr, "Memory reallocation failed\n");

fclose(file);

return;

} else {

parseLine(line, &((\*data)[count])); // Parse the literal numbers and data line into the array

count++;

}

}

fclose(file);

\*dataSize = count; // Update dataSize with the actual amount of lines instead of 0.

}

**dataParsing.h:**#ifndef DATA\_PARSING\_H

#define DATA\_PARSING\_H

typedef struct {

char date[11];

float putCallRatio;

int putVolume;

int callVolume;

int totalOptionsVolume;

} MarketData;

void readCSVFile(const char\* filepath, MarketData\*\* data, int\* dataSize);

void printMarketData(const MarketData\* data, int rowNumber);

void printMarketDataRange(const MarketData\* data, int dataSize, int startRow, int numberOfRowsToPrint);

#endif

**main.c:**

#include <stdio.h>

#include <stdlib.h>

#include "dataParsing.h"

#include "dataAnalysis.h"

int main() {

const char\* filepath = "/mnt/c/Users/ComputerName/Desktop/SPY-Options-Analysis/data/spxpc.csv"; // Adjust this to where your spxpc.csv is actually located

int dataSize = 0;

MarketData\* data = NULL;

data = malloc(dataSize \* sizeof(MarketData));

if (data == NULL) {

fprintf(stderr, "Memory allocation failed\n");

return 1;

}

readCSVFile(filepath, &data, &dataSize);

int startRow = 0; // Adjust this for the row number you want printed first

int numberOfRowsToPrint = dataSize; // Adjust this for the number of rows to print after the startRow

printMarketDataRange(data, dataSize, startRow, numberOfRowsToPrint);

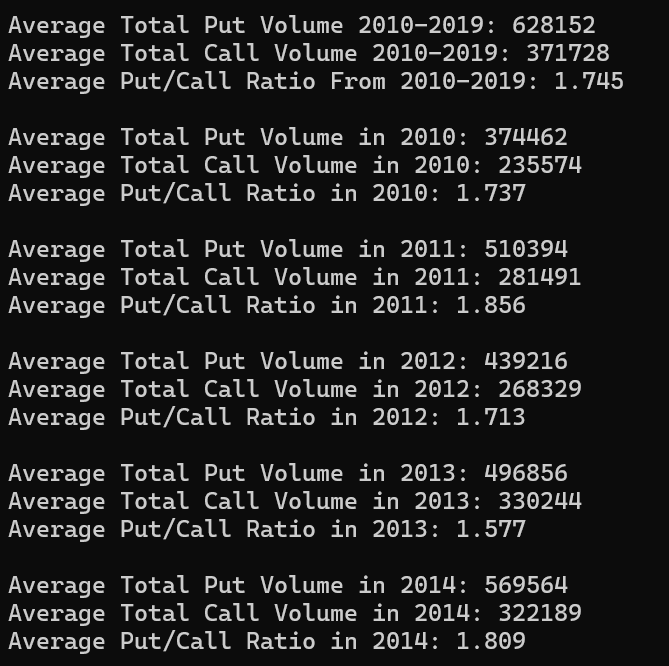
analyzeData(data, dataSize);

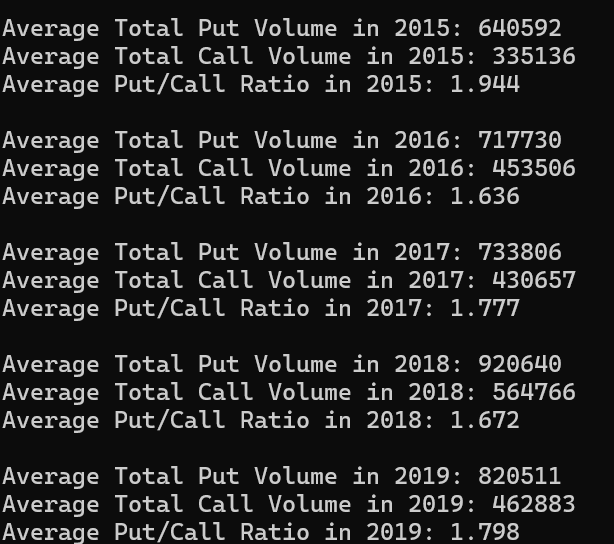
free(data);

return 0;

}

**Output Supporting Conclusions Drawn:**

UPDATED FIGURES:



Put to call Ratio displays how well the market is doing as a whole. Puts allow you to sell a specific stock based on the date or if it were to drop to a specific price. Whereas a call is the exact opposite and you choose to buy a stock when it hits a desired price with an expiration date of when you would no longer like to buy a stock. Therefore if the amount of puts on a stock is higher than the calls, people are looking to pull more money out than put in, so if the ratio of puts to calls is higher, the market is doing worse, the lower and the market is doing better. Knowing this helps determine conclusions of the stocks per year. It is also important to note that puts will normally always be higher than calls because in general people tend to sell their stocks more than buy them.

After the big 2008 crisis the stock market was on a comeback journey which is evident between 2010 and 2013. After 2013 however the stock market got worse, marking the worst it's been by 2015 and eventually it evened out between 2016 and 2019, swaying closely between to relative points.

**Video Demo: https://github.com/SvensDaukss/SPY-Options-Analysis/blob/3417ffd3c82b4f0ad392efb6a3936a607b19f6c1/documentation/VideoDemo.mp4**

**Contributions of Team Members:**

Caleb: Report and Drawing Conclusions from code

Ujjwal: Mathematics and Test Checking

James: Helped with Code

Svens: Main Coding Solutions