Research on Stock Earning Impact

FRE-GY 6883 Financial Computing Team Project

Murray Wang
Zhuoran Ma
Yiyao Li
Lulin Wang
Qinkai Yang
Kaiyun Kang

2022/05/11

Summary

In this project, our team investigated the relationship between Russell 3000 stocks' earning surprise percentage and their CAAR (Cumulative Average Abnormal Return) based on IWV benchmark before and after the earning announcement date.

We divided all Russell 3000 stocks equally into 3 groups (beat, meet and miss) according to their earning surprise percentage and implemented bootstrapping 40 times with batch size 80 to all 3 groups. We found that in general, stocks with higher surprise percentage would have higher CAAR after earning announcement date.

Task Allocation

Murray Wang: Implement menu, group stocks, multithreading retrieve data

Zhouran Ma: Multithreading retrieve data, create Benchmark classes

Yiyao Li: Calculate and pull data of individual stock, create class Stock

Lulin Wang: Implementation of bootstrapping, created relative classes

Qinkai Yang: Create class Vector, class Onecalcul, class SampleCalcul

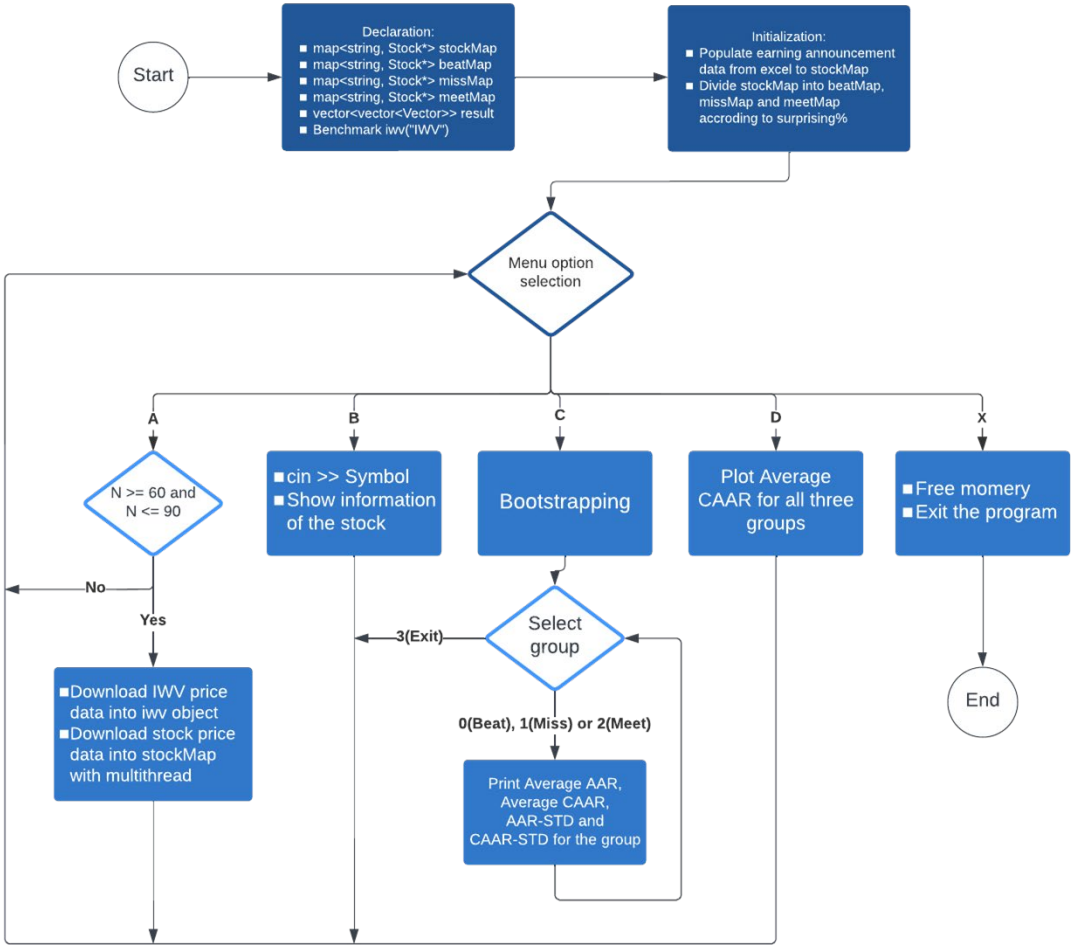
Kaiyun Kang: Plot average CAARs using Gnuplot and make conclusions

PART 01

Project Design Diagrams

PART 01

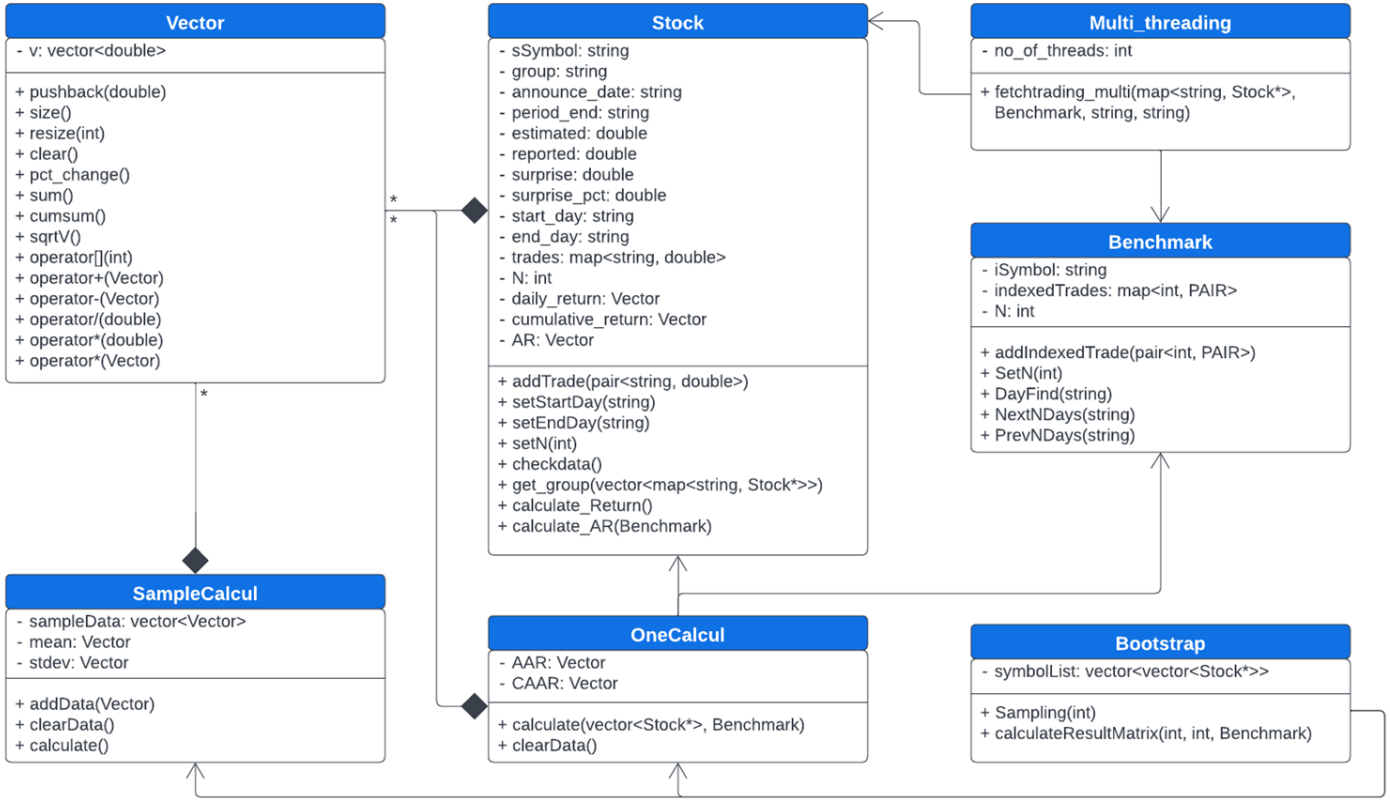
Flow Chart



NYU

PART 01

Class UML Diagram



NYU

Group Selected Stocks

- We created three individual map containers to group stock pointers.
- We used **vector<pair<string, double>> stockSurp** to realize the sorting according to surprise% since map is always sorted by keys.

Function defining the sorting rule

```
bool cmp_by_value(const PAIR& lhs, const PAIR& rhs) {
    return lhs.second < rhs.second;
}
```

Sorting symbols by surprise% in ascending order

```
sort(stockSurp.begin(), stockSurp.end(), cmp_by_value);
```

- We created **map<string, Stock*> beatMap, meetMap, missMap** based on **stockSurp** and **stockMap**.

PART 02

Class declaration & Data structures

Class: Benchmark

- Typedef pair<string, double> PAIR

Private Data:

- String iSymbol
- map <int, PAIR> indexedTrades
- int N

Public Member Function:

- getSymbol( )
- getIndexedTrades( )
- getN( )
- addIndexedTtade( )
- DayFind( )
- NextNDays( )
- PrevNDays( )

Data and Function related to benchmark

User defined data type to store pair of ticker and surprise %

Store the benchmark data in the map

Get private data

Some firms release report on weekends (eg: CE, ON), so we will need to adjust the day0 to next trading day.

Find for the previous N days before the announcement date and Next N days after the announcement date

Class Benchmark: DayFind( )

- Used to adjust day0 if the announcement date is in the weekend

```
int DayFind(string date) const
{
    // some firms release report on weekends (eg: CE, ON), so we will need to adjust the day0 to next trading day.

    for (auto iter = indexedTrades.begin(); iter != indexedTrades.end(); iter++)
    {
        if (iter->second.first >= date) return iter->first;
    }
    return (int)indexedTrades.size();
}
```

Go through the iterator to check whether we need to adjust date

Class Muti_threading: Retrieve price data

int no_of_threads;
void fetchtrading_multi(map<string, Stock*>& stockMap, const Benchmark& benchmark, const string &url_common, const string &api_token)

01	stockmap vector<string> stockTickers	• Populate startday, endday, N using benchmark.PrevNDays(annouceday), NextNDays, GetN() • Record the stock symbols in stockTickers in the same order as stockMap
02	vector<map<string, Stock*>> sub_stockMap_list; vector<thread> threads;	• sub_stockmap_list store all sliced stock map • threads.emplace_back(thread(fetchtrading, ref(sub_stockMap_list[i]), url_common, api_token))

NYU

Single Thread

int fetchtrading(map<string, Stock*>& stockMap, string url_common, string api_token)

Iterate over stockmap

Get startday/endday from stock pointer

Use Libcurl to fetch data and store in the stock object

```
iter->second->addTrade(make_pair(sDate, dValue));
```

warning for stocks with less than 2N+1 price data

```
if (iter->second->getTrades().size() != 2 * iter->second-> getN() + 1) {
        cout << iter->first << '\t' << iter->second->getTrades().size() << endl;
```

Class: Stock

Private Data:

- Symbol
- Group
- Announce_date
- Period_end
- estimated
- reported
- surprise
- surprise_pct
- start_day
- end_day
- map<string, double> trades
- Vector daily_return
- Vector cumulative_return
- Vector AR

Public Member function:

- getSymbol( )
- getAnnounceDay( )
- .....
- Vector getReturn( )
- Vector getCumulativeReturn( )
- Vector getAR( )
- .....
- .....
- ostream& operator<<(ostream& ostr, const Stock& stock)

Pull information for one stock from one group

Store stock data in the map. Key is the trading date, and value is the daily price

User-defined data type Vector: to store vector <double>

Get Private data

Overload cout to get stock information

Class Stock: Calculate_Return( )

Use our own Vector class to store daily price

We use some Vector calculation function inside the class Vector to calculate the percentage change and cumulative return

```
//Get Stock Daily Returns
void Stock::calculate_Return(){
    Vector daily_price;
    for (auto itr = trades.begin(); itr != trades.end(); itr++)
    {
        daily_price.pushback(itr->second);

    }
    daily_return = daily_price.pct_change();
    cumulative_return = daily_return.cumsum();
}
```

Class Vector

Compared with STL vector, the Vector class stores a vector<double>.

It has some extended advanced vector calculation functions.

For example:

```
Vector pct_change()const;     // percentage change can be used to calculate the daily return

Vector cumsum()const;     //  cumulative sum can be used to calculate the CAAR
```

We also did operator overloading for common vector calculations (+, -, *, /)

Class Stock: Calculate_AR( )

```
Vector calculate_AR(const Benchmark& benchmark);
```

This member function returns the Abnormal Return vector of the stock.

How it works:

Step 1: Get the daily price and then calculate the daily return of IWV of the corresponding time period

```
Vector iwvPrice;

Vector iwvReturn = iwvPrice.pct_change();
```

Step 2: Calculate the Abnormal Return

```
AR = daily_return - iwvReturn;

return AR;
```

Class OneCalcul

Use this class to calculate the AAR and CAAR of a group of stocks.

Structure of this class:

```
Vector AAR;

Vector CAAR;

void calculate(vector<Stock*> stockList, const Benchmark& iwv);
```

Given a vector of pointers to stock objects in a group and the benchmark, this function will calculate the AAR and CAAR of this stock group and store the result in Vevtor AAR and Vector CAAR.

Class SampleCalcul

Used to store the result of AARs or CAARs of 40 sampling of 1 group and calculate mean & std

Private data member:

vector<Vector> sampleData → Store AAR(CAAR) of 40 sampling

Vector mean
Vector stdev

Public member function:

addData(Vector v) → Called after each sampling to store results

calculate() → Called after all the sampling to use sampleData to calculate Ave AAR(CAAR) & AAR std(CAAR std) and store in mean & stdev

getMean()
getStd() → Return mean(stdev)

Class Bootstrap

Used to generate random number to do the sampling and calculate result matrix of 3 group

Private data member:

vector<vector<Stock*>> symbolList

Store all the stock pointers that have enough data
symbolList.size() = 3
symbolList[i].size() = num of stocks with enough data in each group

Public member function:

constructor with a parameter (parameter: vector of map of 3 group)

Set seed for random number generator

go through 3 map & store pointer into symbolList

Class Bootstrap

Public member function:

vector<vector<Stock*>> Sampling(int size)

Generate a sample for 3 group

size: num of sample, in this project 80

Use while loop to ensure 80 different stocks for each group

Size of return: 3x80

vector<vector<Vector>> calculateResultMatrix(int size, int times, const Benchmark& iwv)

size: used when calling the function Sampling

times: times of sampling, in this project 40

1) do the sampling, call the member function of class OneCalcul to get AAR & CAAR and store.
2) call the member function of class SampleCalcul to calculate and store the result

Size of return: 3x4x2N

PART 03

Visualization & Gnuplot

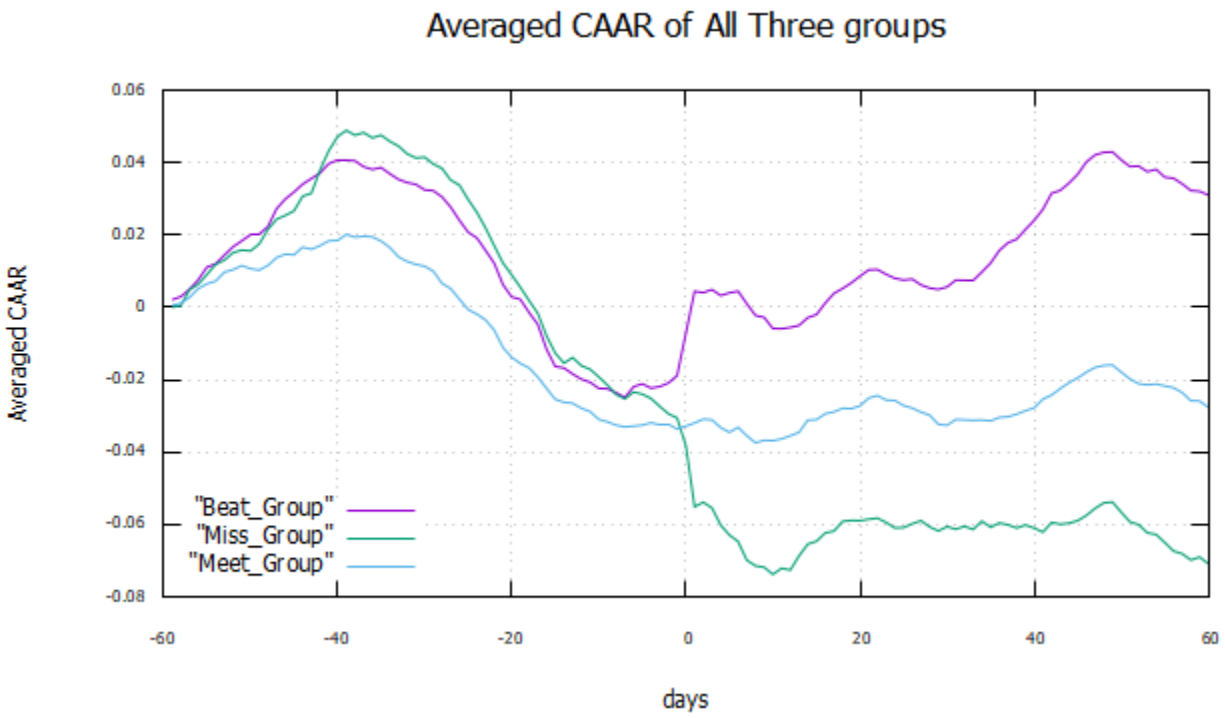
Gnuplot - Averaged CAAR for Beat, Meet, Miss Groups

Set Up Gnuplot

Function	*plot_CAAR(vector<double> xData, vector<double> yData1, vector<double> yData2, vector<double> yData3)*	
Data type	*vector<double>*	
yData	*Averaged CAAR for 40 samplings*	*yData1: Beat group*
		yData2: Miss group
		yData3: Meet group
xData	*2N days (x=0 means announcement day)*	
Operation	*gnuplotPipe & 3 tempDataFiles*	

Back to Menu

Gnuplot - Averaged CAAR for Beat, Meet, Miss Groups



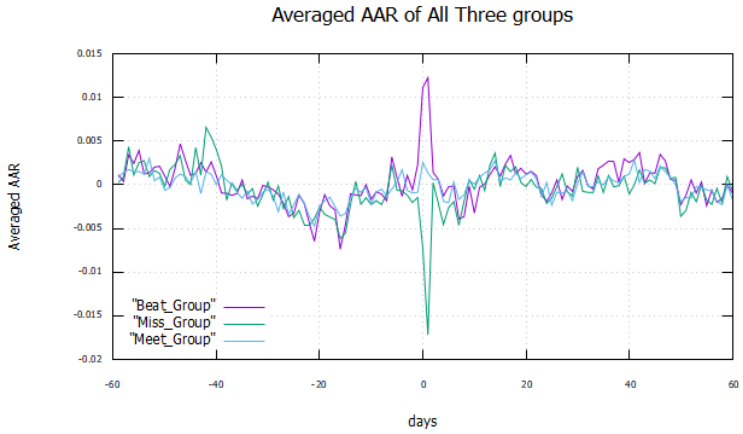
PART 04

Conclusion

Conclusion

Three groups are sorted by their earnings surprise of 2021 Q2. Earnings reveal the financial health and economic conditions of businesses. An earnings surprise occurs when a company's reported quarterly or annual profits are above or below analysts' expectations. When a company's profit performance fails to match the expectations set by the investment community, investors often express their disappointment by selling shares.

Our group concluded that: **earning releases of Russell 3000 stocks have a huge impact on their future stock prices, and stocks with higher surprise will have higher CAAR after earning announcement date.** This conclusion can be explained in three aspects from graphs CAAR and AAR as following table:



Conclusion

1. Comparison between two periods	Before day 0	Three groups have similar fluctuation patterns. **miss**>beat>meet.
	After day 0	The CAAR result shows obvious **gaps**: beat>meet>miss This follows their different features of surprises values, which reflect the effects of surprises release.
2. Suddenly changes after the announcement day	A positive surprise release will often lead to a sharp increase in the company's stock price, while a negative surprise release to a rapid decline. Beating the guidance causes more investors to jump on the bandwagon and buy more stock. ***(more obvious in AAR graph)***	Beat group: CAAR rises suddenly
		Meet group: CAAR doesn't change much
		Miss group: CAAR drops suddenly
3. Relative longer-term effect of earning surprise release	Beat group: Several studies suggest that positive earnings surprises not only lead to an immediate hike in a stock's price, but also to a gradual increase over time. After day 0, the CAAR graph shows an **upward** tendency. As a result, the beat group **outperforms** the benchmark IWV by approximated 3%.	
	Meet & Miss group: As time passes by, the CAAR tends to be stable, and the effect of "earning surprises release" is not apparent. As a result, they both **underperform** the benchmark IWV.	

PART 05

Enhancement & Enrichment

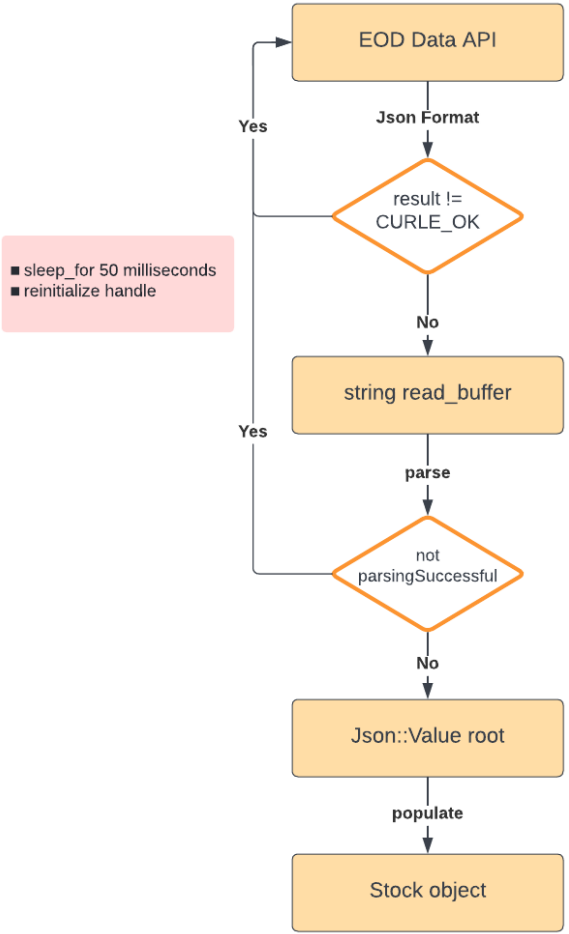
Enhancement & Enrichment

- We downloaded json format data from EOD and parsed them into our stock objects for program stability and efficiency.

- Fixed issues which might occur when retrieving data from EOD on PC:

  - Libcurl connection failed

  - 429 Too Many Request error



PART 06

Reference

Reference

Multithreading in C++:
https://www.geeksforgeeks.org/multithreading-in-cpp/

How Earnings Affect Stock Prices:
https://money.usnews.com/investing/investing-101/articles/how-earnings-affect-stock-prices

The Impact of Earnings Announcements on Stock Prices:
https://finance.zacks.com/impact-earnings-announcements-stock-prices-4265.html

How to Fix 429 Too Many Requests Error:
https://kinsta.com/knowledgebase/429-too-many-requests/