FRE6883 FINAL PROJECT

Xinhao Zhu, Wen Teng and Lewei Peng

Outline

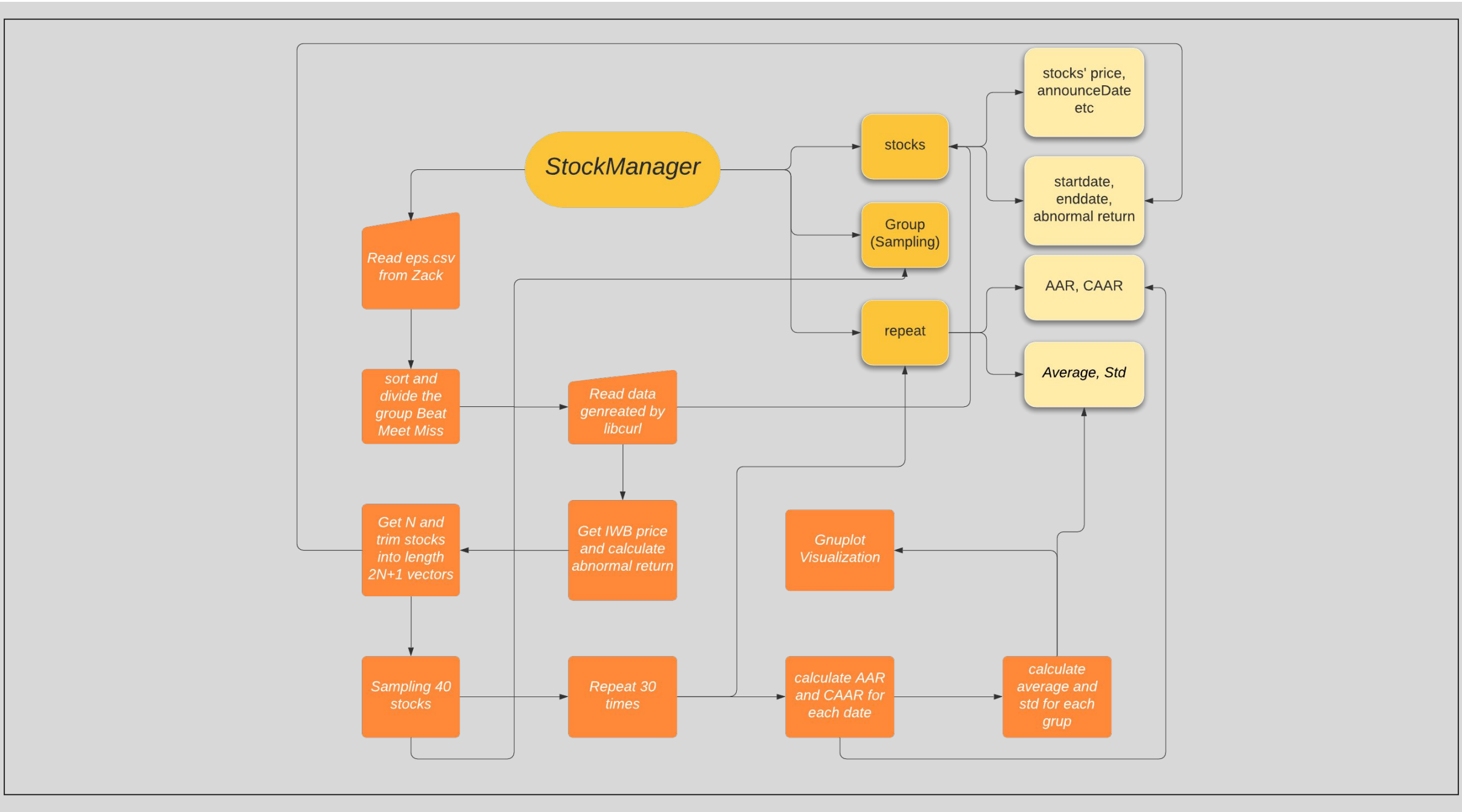
STRUCTURE

DETAILS

RESULT

Derek

STRUCTURE


StockManager
Read eps.csv from Zack
sort and divide the group Beat Meet Miss
Get N and trim stocks into length 2N+1 vectors
Sampling 40 stocks
Read data genreated by libcurl
Get IWB price and calculate abnormal return
Repeat 30 times
calculate AAR and CAAR for each date
calculate average and std for each grup
Gnuplot Visualization
stocks
Group (Sampling)
repeat
stocks' price, announceDate etc
startdate, enddate, abnormal return
AAR, CAAR
Average, Std

DETAILS

Contribution

Xinhao Zhu: StockManager, Stock, Main, Zack

Derek Peng: Visualization, multi-thread, libcurl

Wen Teng: Repeat, Group, Sampling.

Extract data from Zacks

Xinhao

```
C=C{1};
C(1)=[];
disp(C)

N = length(C);
A=randperm(N);
B = sort(A);

earnings={};
for i = B
    disp(C{i})
    earning = scrapeEarningsZacks(C{i});
    [m,n] = find(earning=='Mar 2020');
    if(isempty(m))
       [m,n] = find(earning=='Apr 2020');
       if(isempty(m))
            continue
       end
    end
    earning=earning(m(1),:);
    earning(7)={char(C{i})};
    earnings=[earnings;strrep(earning,',','')];
end
```

Main function Part I

```
{
    cout << "For Convenience, the author defaulted N = 3, SamplingTimes = 30." << endl
        << "The user can change the setting by menu later."<<endl
        << "***********  Please Press Any Key to Continue: **********" << endl;
    system("pause");

    int choice = 0;//Used to store user options

    stockManager sm;

    bool OneClicked = false;
    vector<vector<double>> plotmatrix;
    string plotchoice;
    while (true) {
        sm.showMenu();
        cout << "Please enter your options: " << endl;
        cin >> choice;
        switch (choice) {
        case 1:
            OneClicked = true;
            cout << "Please enter N: " << endl;
            int N; cin >> N;
            sm.slice_N(N);
            sm.GetData();
            sm.updateStocksReturn();
            sm.calculate_abret();
            sm.enter_N_retrieve_data();
            break;
```

Constructor

```
//default constructor
stockManager::stockManager() {


    this->load_IWB();//RetrieveIWB(IWB, datelist);

    //Load "eps1.csv" file and load eps, surprise and ticker information.
    this->load_eps_csv();


}
```

Download data using Multi-thread

```
void stockManager::GetData() {

    thread t1(ref(RetrieveMulti), ref(beat));
    thread t2(ref(RetrieveMulti), ref(meet));
    thread t3(ref(RetrieveMulti), ref(miss));
    t1.join();
    t2.join();
    t3.join();


    stocks.swap(beat);
    stocks.insert(meet.begin(), meet.end());
    stocks.insert(miss.begin(), miss.end());

    //RetrieveAll(stocks);
}
```

Libcurl

```
string url = urlA + symbol + urlB + startTime + urlC + endTime + urlD + sCrumb;
const char* cURL = url.c_str();
const char* cookies = sCookies.c_str();
curl_easy_setopt(handle, CURLOPT_COOKIE, cookies);
curl_easy_setopt(handle, CURLOPT_URL, cURL);

curl_easy_setopt(handle, CURLOPT_WRITEFUNCTION, write_data2);
curl_easy_setopt(handle, CURLOPT_WRITEDATA, (void*)&data);
result = curl_easy_perform(handle);

if (result != CURLE_OK)
{
    // if errors have occurred, tell us what is wrong with result
    fprintf(stderr, "2 curl_easy_perform() failed: %s\n", curl_easy_strerror(result));
    return 1;
}

cout << itr->first;

stringstream sData;
sData.str(data.memory);

string sValue;
double dValue = 0;
vector<double> adj;
string line;
getline(sData, line);
while (getline(sData, line)) {
    size_t found = line.find("-");
    if (found != std::string::npos) {
        line.erase(line.find_last_of(','));
        sValue = line.substr(line.find_last_of(',') + 1);
        dValue = strtod(sValue.c_str(), NULL);
        adj.push_back(dValue);
    }
}
```

Xinhao

```
class stockManager {
public:
    map<string, stock> stocks;
    int N;
    void enter_N_retrieve_data();
    void pull_one_stock_data();
    stockManager(); //constructor
    ~stockManager(); //destructor
    vector<stock> Stocks;
    void showMenu();    //show menu
    void exitSystem(); //exit program
    void trim_N(int N);
    void load_result_txt();
    void load_eps_csv();
    void load_IWB_csv_and_update_AbnormalRet();
    void updateStocksReturn();
    void SetStocks(vector<stock> & Stocks1000);
    repeat r;
    void updateRepeat();
    void updateStocks();
    double GetN() {return N;}
};
```

```
class stock {
public:
    string ticker;
    vector<string> Date;
    vector<double> Price;
    string announcementDate;
    double surprise;
    bool PriceIsEmpty; //mark useless data
    string group;
    vector<double> Return;
    vector<double> AbnomalRet;
    vector<double> AbnomalRet_trimmed;
    vector<double> Price_trimmed;
    vector<string> Date_trimmed;
    string startDate;
    string endDate;
    void calculateReturn();
    void showInfo();
    stock(string ticker_, string announcementDate_,
        ticker = ticker_;
        announcementDate = announcementDate_;
        surprise = surprise_;
        group = group_;
        PriceIsEmpty = false;
    }
};
```

Trim all the vector to 2N+1 length

Xinhao

```
void stockManager::trim_N(int N) {
    for (map<string, stock>::iterator it = this->stocks.begin(); it != this->stocks.end(); it++) {
        string annDate = it->second.announcementDate;
        string annDate2 = transform_Format(annDate);
        vector<string>::iterator itr;
        itr = find(it->second.Date.begin(), it->second.Date.end(), annDate2);
        int nPosition = distance(it->second.Date.begin(), itr);
        if (itr != it->second.Date.end()) {
            if (nPosition < N || it->second.PriceIsEmpty == true) {
                it->second.PriceIsEmpty = true;
                continue;
            }
            cout << "Historical Price Information for Stock: " << it->second.ticker << " " << endl;
            vector<double>temp(it->second.AbnomalRet.begin() + nPosition - N,
                it->second.AbnomalRet.begin() + nPosition + N + 1);
            vector<string>temp2(it->second.Date.begin() + nPosition - N, it->second.Date.begin() + nPosition + N + 1);
            vector<double>temp3(it->second.Price.begin() + nPosition - N,
                it->second.Price.begin() + nPosition + N + 1);
            it->second.AbnomalRet_trimmed = temp;
            it->second.Date_trimmed = temp2;
            it->second.endDate = it->second.Date_trimmed.back();
            it->second.startDate = it->second.Date_trimmed.front();
            it->second.Price_trimmed = temp3;
            cout << it->second.Price_trimmed;
            cout << it->second.Date_trimmed;
        }
        else {
            cout << "Failure!! Data missing or incomplete for stock: " << it->second.ticker << endl;
        }
    }
```

Sampling & Calculation WT

Independent function: sampling

```
inline vector<stock> sampling(string GroupType_, int StockAmount_, const vector<stock> Stocks1000_)
{
    vector<stock> sample;
    size_t n = Stocks1000_.size();
    for (int i = 0; i < StockAmount_; i++)
    {
        srand(time(NULL));
        int s = rand() % n;
        while (Stocks1000_[s].group != GroupType_ || Stocks1000_[s].PriceIsEmpty == 1)
        {
            if (s >= n - 1) s = rand() % (n - 1);
            s += 1;
        }
        sample.push_back(Stocks1000_[s]);
    }
    return sample;
};
```

Class: group

```
class group
{
private:
    string GroupType;
    int StockAmount;
    vector<stock> StockSet;
    vector<double> AAR; //1*60
    vector<double> CAAR; //1*60
public:
    group() :GroupType("beat"), StockAmount(40)
    {};
    group(string GroupType_, int StockAmount_, const vector<stock>& Stocks1000_);
    vector<double> calAAR();
    vector<double> calCAAR();
    friend class repeat;
};
```

Class: repeat

```
class repeat
    //Sampling: 40 stocks for each of the three groups; pass sampling times during initialization;
{
private:
    int times;
    //Repeat sampling for a certain times and record the AAR and CAAR of each time
    //Size: times * (2N+1)
    vector<vector<double>> BeatAAR;
    vector<vector<double>> MeetAAR;
    vector<vector<double>> MissAAR;
    vector<vector<double>> BeatCAAR;
    vector<vector<double>> MeetCAAR;
    vector<vector<double>> MissCAAR;

    //Calculate average of AAR for each group. Ave[0] is for beat. Ave[1] is for meet. Ave[2] is for miss.
    //Size: 3*(2N+1)
    vector<vector<double>> AveAAR;
    vector<vector<double>> AveCAAR;
    vector<vector<double>> StdAAR;
    vector<vector<double>> StdCAAR;
    vector<vector<vector<double>>> MatrixAveStd;
```

Class: repeat

```
public:
    repeat() : times(30) //Default constructor with default sampling times 30
    {};
    repeat(int times_, const vector<stock>& Stocks1000_);
    vector<vector<double>> GetAveAAR();
    vector<vector<double>> GetAveCAAR();
    vector<vector<double>> GetStdAAR();
    vector<vector<double>> GetStdCAAR();
    void showInfo(); //Print mean and variance information.
};
```

Main function Part II

```
case 2:
    if (OneClicked == false) {
        cout << "You must enter \"1\" first to kick off the sampling process!!! ";
        system("pause");
        system("cls");
        break;
    }
    sm.pull_one_stock_data();
    break;
case 3:
    if (OneClicked == false) {
        cout << "You must enter \"1\" first to kick off the sampling process!!! ";
        system("pause");
        system("cls");
        break;
    }
    sm.r.showInfo();
    break;
```

Pull one stock

```
void stockManager::pull_one_stock_data() {
    cout << "Please Enter one stock ticker: " << endl;
    string target; cin >> target;
    map<string, stock>::iterator it= this->stocks.find(target);
    if (it != stocks.end()) {
        (*it).second.showInfo();//stock
    }
    else {
        cout << "This stock's info is missing in our database" << endl
    }
    system("pause");
    system("cls");

}
```

Print four variables

```
void repeat::showInfo() {

    string arr[3] = {"Beat","Meet","Miss"};
    cout << "AveARR: " << endl;
    for (int i = 0; i < AveAAR.size(); i++) {
        cout  << arr[i] << endl;
        for (int j = 0; j < AveAAR[i].size()-1; j++) {
            cout << AveAAR[i][j] << " ";
        }
        cout << endl;
    }
```

Main function Part III

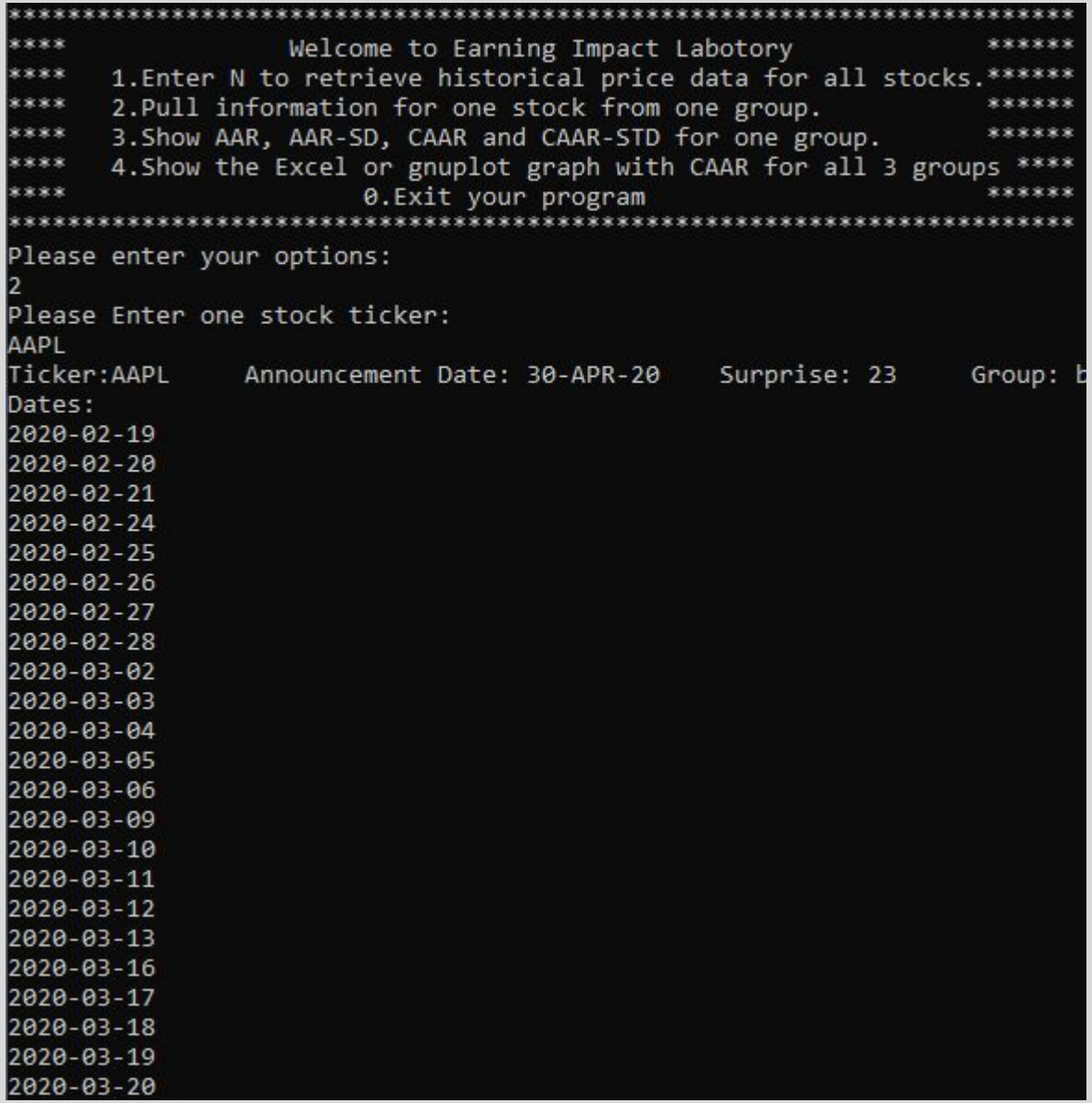
```
case 4:
    if (OneClicked == false) {
        cout << "You must enter \"1\" first to kick off the sampling process!!! ";
        system("pause");
        system("cls");
        break;
    }

    cout << "Choose AveAAR, AveCAAR, StdAAR or StdCAAR" << endl;
    cin >> plotchoice;
    if (plotchoice == "AveAAR") plotmatrix = sm.r.GetAveAAR();
    else if (plotchoice == "AveCAAR") plotmatrix = sm.r.GetAveCAAR();
    else if (plotchoice == "StdAAR") plotmatrix = sm.r.GetStdAAR();
    else if (plotchoice == "StdCAAR") plotmatrix = sm.r.GetStdCAAR();
    else {
        cout << "wrong input!" << endl;
        break;
    }
    plotResults(plotmatrix, sm.GetN());
```

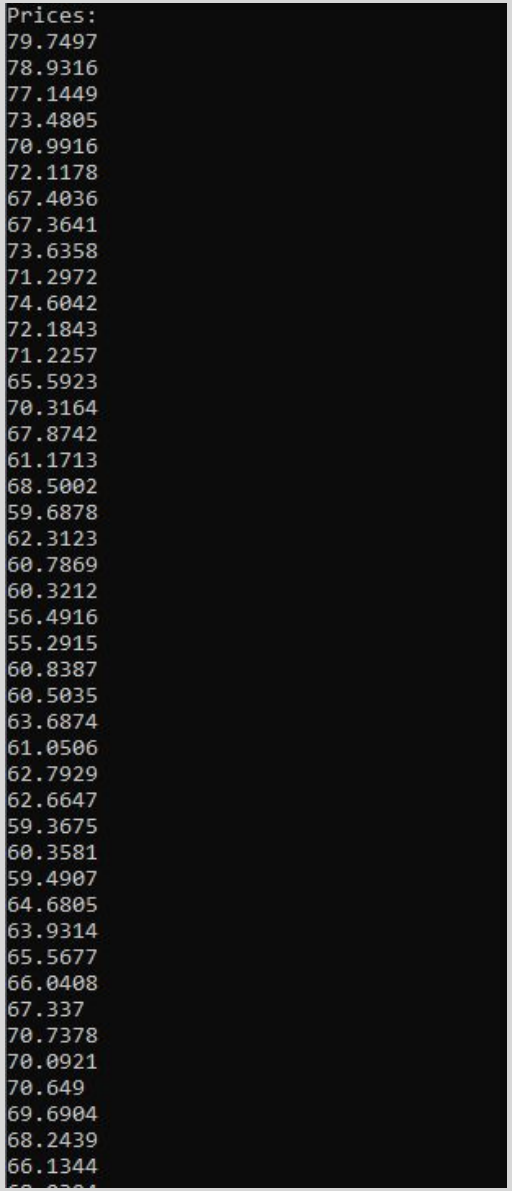
Gnuplot —Derek

```
void plotResults(vector<vector<double>> repeat, int N = 30) {
    FILE* gnuplotPipe, * tempDataFile;
    vector<double> beat1 = repeat[0];
    vector<double> meet1 = repeat[1];
    vector<double> miss1 = repeat[2];
    size_t dataSize = beat1.size() - 1;

    const char* tempDataFileBeat = "Beat";
    const char* tempDataFileMeet = "Meet";
    const char* tempDataFileMiss = "Miss";

    double x1, y1, x2, y2, x3, y3;
    int i;
    // gnuplotPipe = popen("/opt/local/bin/gnuplot","w");
    //gnuplotPipe = _popen("C:\\PROGRA~1\\gnuplot\\bin\\gnuplot.exe", "w");
    gnuplotPipe = _popen("C:\\PROGRA~1\\gnuplot\\bin\\gnuplot.exe", "w");
    if (gnuplotPipe) {
        fprintf(gnuplotPipe, "plot \"%s\" with lines, \"%s\" with lines, \"%s\" with lines\n",
        fflush(gnuplotPipe);
```

```
tempDataFile = fopen(tempDataFileBeat, "w");
for (i = 0; i <= dataSize; i++) {
    x1 = i - N;
    y1 = beat1[i];
    fprintf(tempDataFile, "%lf %lf\n", x1, y1);
}
fclose(tempDataFile);


tempDataFile = fopen(tempDataFileMeet, "w");
for (i = 0; i <= dataSize; i++) {
    x2 = i - N;
    y2 = meet1[i];
    fprintf(tempDataFile, "%lf %lf\n", x2, y2);
}
fclose(tempDataFile);

tempDataFile = fopen(tempDataFileMiss, "w");
for (i = 0; i <= dataSize; i++) {
    x3 = i - N;
    y3 = miss1[i];
    fprintf(tempDataFile, "%lf %lf\n", x3, y3);
}
fclose(tempDataFile);
```

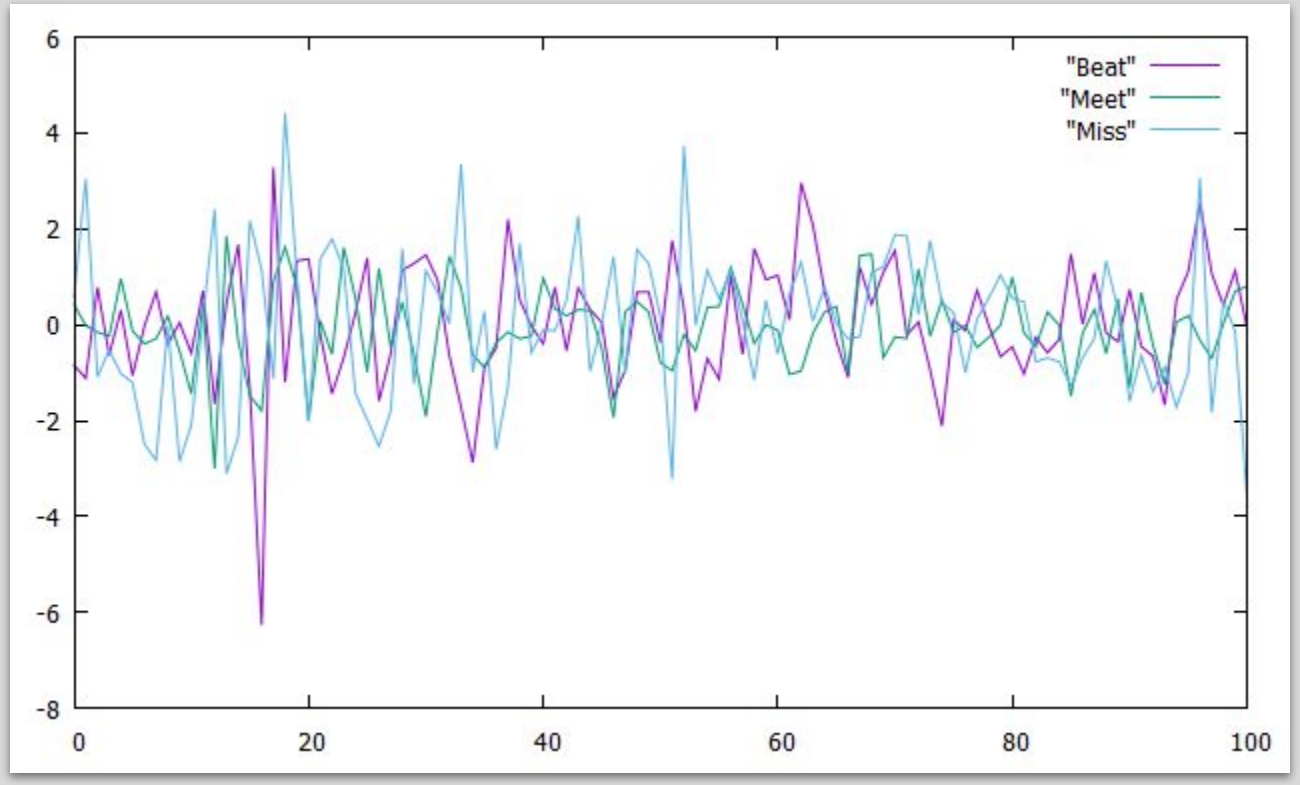
WT
Xinhao


****
**** Welcome to Earning Impact Labotory ******
**** 1.Enter N to retrieve historical price data for all stocks.******
**** 2.Pull information for one stock from one group. ******
**** 3.Show AAR, AAR-SD, CAAR and CAAR-STD for one group. ******
**** 4.Show the Excel or gnuplot graph with CAAR for all 3 groups ****
**** 0.Exit your program ******
****
Please enter your options:
2
Please Enter one stock ticker:
AAPL
Ticker:AAPL Announcement Date: 30-APR-20 Surprise: 23 Group: b
Dates:
2020-02-19
2020-02-20
2020-02-21
2020-02-24
2020-02-25
2020-02-26
2020-02-27
2020-02-28
2020-03-02
2020-03-03
2020-03-04
2020-03-05
2020-03-06
2020-03-09
2020-03-10
2020-03-11
2020-03-12
2020-03-13
2020-03-16
2020-03-17
2020-03-18
2020-03-19
2020-03-20

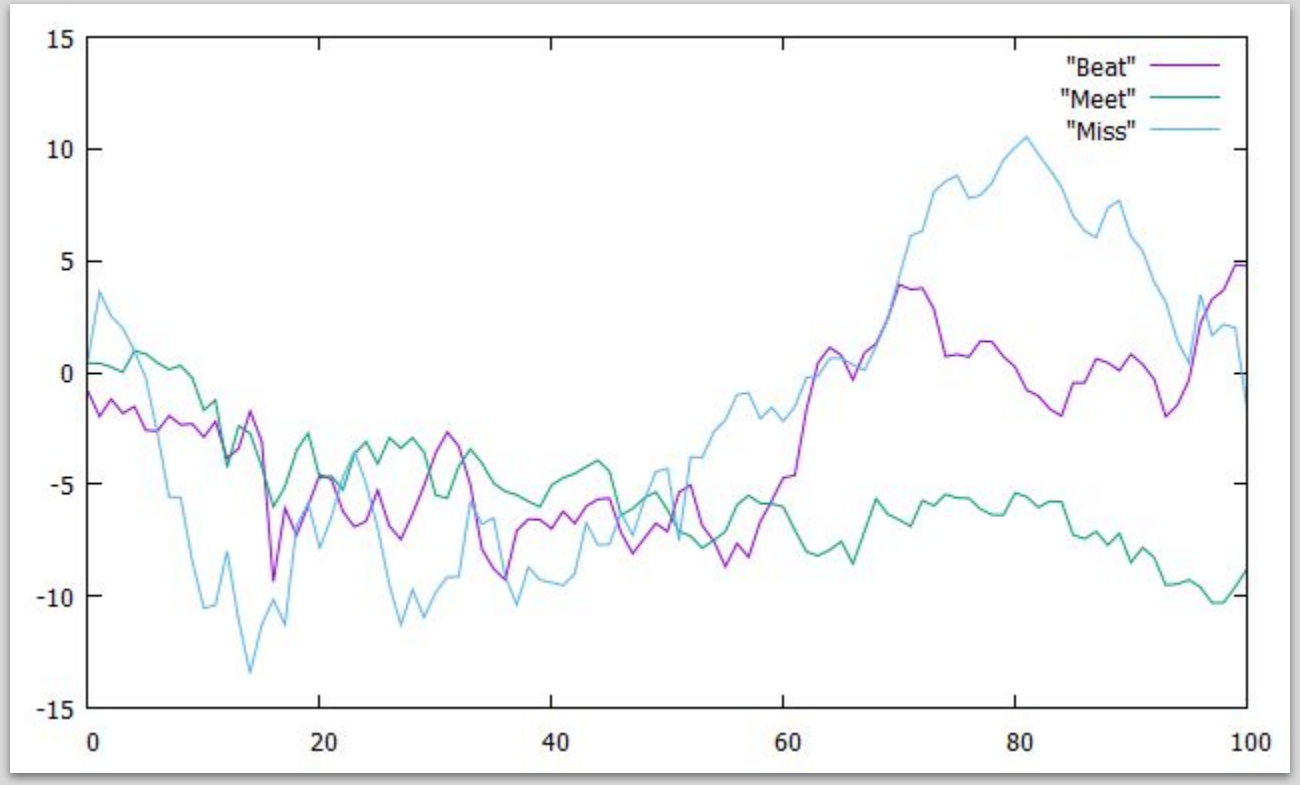

Prices:
79.7497
78.9316
77.1449
73.4805
70.9916
72.1178
67.4036
67.3641
73.6358
71.2972
74.6042
72.1843
71.2257
65.5923
70.3164
67.8742
61.1713
68.5002
59.6878
62.3123
60.7869
60.3212
56.4916
55.2915
60.8387
60.5035
63.6874
61.0506
62.7929
62.6647
59.3675
60.3581
59.4907
64.6805
63.9314
65.5677
66.0408
67.337
70.7378
70.0921
70.649
69.6904
68.2439
66.1344

```
**********************************************************************
****                Welcome to Earning Impact Labotory            ******
****   1.Enter N to retrieve historical price data for all stocks.******
****   2.Pull information for one stock from one group.           ******
****   3.Show AAR, AAR-SD, CAAR and CAAR-STD for one group.       ******
****   4.Show the Excel or gnuplot graph with CAAR for all 3 groups ****
****                    0.Exit your program                       ******
**********************************************************************
Please enter your options:
3
AveARR:
Beat
-0.686896 -1.11627 0.794684 -0.599417 0.31762 -1.02923 0.0161614 0.395215 -0.503928 -0.0408249 -0.679177 0.696106 -1.6346 0.265391 1.61406 -0.699856 -6.95536 3.27259 -1.53884 1.04212 1.44955 -0.406292 -1.25537 -0.700798 0.401347
1.40858 -1.68812 -0.697649 0.994187 1.2013 1.81882 1.20531 -0.906733 -1.70812 -3.15534 -0.749849 -0.631881 2.17148 0.487234 0.0198085 -0.176494 0.639807 -0.434136 0.837401 0.338914 -0.0267723 -1.4797 -1.02085 0.653305 0.492467 -0
.283467 2.04199 0.373042 -1.98374 -0.707608 -1.28887 0.909306 -0.494238 1.54467 1.30291 1.04479 0.162564 3.18008 2.02651 0.887622 -0.323651 -1.11818 1.14793 0.246311 1.12735 1.66345 0.00936869 -0.00602456 -0.988191 -2.36974 0.100
305 -0.369617 0.698797 -0.00807912 -0.690119 -0.520378 -1.19394 -0.322507 -0.624076 -0.321688 1.46207 -0.00270212 1.13977 -0.208589 -0.305232 0.666439 -0.541966 -0.633827 -1.69497 0.539972 1.1682 2.60303 1.25626 0.494258 1.11394
Meet
0.136006 -0.184561 0.0806447 -0.487949 0.946592 -0.0355925 -0.0275798 -0.485638 -0.0989507 -0.524981 -1.10988 0.00812179 -3.02455 1.71903 -0.879579 -1.3365 -2.37476 0.603698 1.09088 1.13515 -2.55754 0.33625 0.00567446 1.18541 0.5
75188 0.150609 0.0814965 -0.218286 0.10518 -0.298915 -1.53012 -0.212642 1.31406 0.44939 -0.0884385 -0.608394 -0.46885 0.0300943 -0.372426 -0.551984 0.652014 0.342597 0.646324 0.489884 0.104903 -0.36717 -1.81263 0.251821 0.482217
0.734598 -1.11444 -0.453333 -0.378875 -0.855394 -0.207861 0.241309 1.68108 0.285613 -0.189694 -0.109545 -0.250727 -0.887729 -0.650814 0.164209 0.26811 0.401896 -1.30043 1.59784 1.62659 -0.486425 -0.0869688 -0.214304 0.760841 -0.4
70686 0.790753 -0.223515 0.558437 -0.865187 -0.251485 -0.233148 0.912858 -0.201119 -0.578067 0.335677 -0.261733 -1.0018 0.0604321 0.265392 -0.442458 -0.0188249 -1.34882 0.375685 -0.736168 -1.34654 -0.208965 0.191852 -0.170788 -0
584036 -0.0876538 0.33048
Miss
0.637125 3.49232 -0.680287 0.105498 -0.841955 -1.04403 -2.51121 -3.61119 1.06144 -3.26201 -2.63216 0.483531 3.06152 -3.87815 -3.29874 3.98865 2.01713 -1.17332 5.71033 2.49626 -2.04594 0.554977 1.70803 1.4048 -1.82063 -3.62933 -3
82364 -2.70013 1.92616 -1.40712 2.36848 1.41582 1.32016 2.6887 -1.82115 0.165763 -2.70569 -2.08487 0.568224 -1.25827 1.1972 0.540681 0.431183 4.09765 -1.49405 -1.19883 0.967321 -2.005 2.34294 0.827191 0.478153 -3.60125 3.39938 -0
.928239 1.67641 1.33808 2.62205 -0.689816 -1.04973 -0.0762886 -0.992171 0.496011 1.8331 0.48188 -0.176211 -0.262336 -1.7989 0.750576 1.31155 1.75033 1.64027 1.22083 -0.723116 0.062864 1.3115 0.545573 -0.668723 -0.813891 1.73973 0
.726429 1.01525 1.0484 0.116168 0.101625 -1.30146 -1.52222 -1.11737 0.598288 1.69138 0.491279 -2.24832 -0.721182 -1.69088 -0.625002 -2.0465 -1.0342 4.23512 -2.32352 0.528711 -0.235733
AveCARR:
Beat
-0.686896 -1.80316 -1.00848 -1.6079 -1.29028 -2.3195 -2.30334 -1.90813 -2.41205 -2.45288 -3.13206 -2.43595 -4.07055 -3.80516 -2.19109 -2.89095 -9.84631 -6.57372 -8.11256 -7.07044 -5.62089 -6.02718 -7.28256 -7.98335 -7.58201 -6.17
343 -7.86155 -8.5592 -7.56502 -6.36372 -4.5449 -3.33959 -4.24632 -5.95444 -9.10978 -9.85963 -10.4915 -8.32003 -7.83279 -7.81298 -7.98948 -7.34967 -7.78381 -6.9464 -6.60749 -6.63426 -8.11396 -9.13481 -8.48151 -7.98904 -8.27251 -6
23052 -5.85748 -7.84122 -8.54883 -9.8377 -8.92839 -9.42263 -7.87796 -6.57505 -5.53026 -5.3677 -2.18762 -0.161105 0.726518 0.402866 -0.71531 0.432617 0.678928 1.80628 3.46973 3.4791 3.47308 2.48489 0.115151 0.221457 -0.14816 0.550
637 0.542558 -0.147561 -0.667939 -1.86187 -2.18438 -2.80846 -3.13015 -1.66808 -1.67078 -0.531008 -0.739597 -1.04483 -0.37839 -0.920356 -1.55418 -3.24916 -2.70918 -1.54098 1.06204 2.3183 2.81256 3.9265
Meet
0.136006 -0.0485547 0.03209 -0.455859 0.490734 0.455141 0.427561 -0.058077 -0.157028 -0.682009 -1.79189 -1.78377 -4.80831 -3.08928 -3.96886 -5.30537 -7.68012 -7.07643 -5.98555 -4.8504 -7.40793 -7.07168 -7.06601 -5.8806 -5.30542 -
5.15481 -5.07331 -5.2916 -5.18642 -5.48533 -7.01546 -7.2281 -5.91404 -5.46465 -5.55308 -6.16148 -6.63033 -6.60023 -6.97266 -7.52464 -6.87263 -6.53003 -5.88371 -5.39383 -5.28892 -5.65609 -7.46872 -7.2169 -6.73468 -6.00008 -7.11452
 -7.56785 -7.94673 -8.80212 -9.00998 -8.76867 -7.0876 -6.80198 -6.99168 -7.10122 -7.35195 -8.23968 -8.89049 -8.72629 -8.45818 -8.05628 -9.35671 -7.75888 -6.13229 -6.61871 -6.70568 -6.91998 -6.15914 -6.62983 -5.83907 -6.06259 -5.5
0415 -6.36934 -6.62082 -6.85397 -5.94111 -6.14223 -6.7203 -6.38462 -6.64636 -7.64816 -7.58772 -7.32233 -7.76479 -7.78361 -9.13243 -8.75674 -9.49291 -10.8395 -11.0484 -10.8566 -11.0274 -11.6114 -11.699 -11.3686
Miss
0.637125 4.12944 3.44916 3.55465 2.7127 1.66867 -0.842543 -4.45374 -3.3923 -6.65431 -9.28647 -8.80294 -5.74142 -9.61956 -12.9183 -8.92966 -6.91253 -8.08584 -2.37551 0.120751 -1.92519 -1.37021 0.337818 1.74261 -0.0780103 -3.70734
-7.53098 -10.2311 -8.30496 -9.71208 -7.34359 -5.92777 -4.60761 -1.91892 -3.74007 -3.57431 -6.28 -8.36487 -7.79665 -9.05492 -7.85772 -7.31704 -6.88586 -2.78821 -4.28226 -5.48109 -4.51376 -6.51876 -4.17583 -3.34863 -2.87048 -6.4717
3 -3.07235 -4.00059 -2.32418 -0.986099 1.63595 0.946134 -0.103596 -0.179884 -1.17206 -0.676045 1.15705 1.63893 1.46272 1.20038 -0.598515 0.152061 1.46362 3.21395 4.85422 6.07505 5.35194 5.4148 6.7263 7.27187 6.60315 5.78926 7.528
99 8.25541 9.27067 10.3191 10.4352 10.5369 9.2354 7.71318 6.59581 7.1941 8.88548 9.37676 7.12844 6.40726 4.71638 4.09138 2.04488 1.01068 5.2458 2.92228 3.45099 3.21526
StdARR:
Beat
1.22011 1.15716 1.9669 1.14448 1.66462 0.816733 1.14488 2.55453 2.60662 1.2505 0.969684 1.29993 2.93061 1.40045 3.8401 7.5296 6.22769 11.0303 3.36152 1.47945 3.46347 3.30672 3.59145 1.38366 2.34031 1.8341 1.64874 1.98489 1.52815
2.16222 3.76733 3.72616 3.31204 1.61038 2.92192 1.21206 3.87005 3.63764 0.761027 1.13601 2.1536 0.999837 1.17263 1.85798 2.97983 2.62713 1.74679 1.29389 2.11852 2.43926 1.77122 2.82302 1.28764 2.70009 2.45902 1.29686 1.5285 1.720
64 2.76433 2.725 1.8434 1.18103 3.42131 4.58476 0.991893 0.984089 2.52805 1.52804 1.38355 1.90848 2.19235 1.57724 3.10755 1.55924 3.03905 2.199 2.62078 1.43197 1.11283 1.5326 1.02251 1.87206 1.12636 1.15408 0.732646 1.39361 1.098
68 1.55038 1.43733 2.29225 1.09985 1.86435 0.987221 0.778977 1.96269 2.28527 2.70319 2.22885 1.80246 1.34831
Meet
1.20669 1.82122 1.34992 2.11935 1.96947 2.53751 2.09159 1.71984 2.46947 2.56885 4.8297 2.93462 5.77074 3.63804 4.25884 1.78414 6.21934 7.71197 4.87664 4.78463 4.18775 5.13366 3.85632 2.44055 2.65857 9.65737 3.29133 4.704 4.20843
2.85676 3.74754 3.74486 4.94635 1.84322 2.47739 3.15093 2.86521 2.14332 1.78354 2.01309 1.82871 2.04696 3.06529 2.57553 1.72268 1.2284 1.80178 2.17177 1.69036 3.57621 1.70219 2.99123 1.99784 1.56765 3.45068 3.56503 2.41708 2.2461
4 2.19755 2.34214 2.19649 2.14797 2.13455 1.96619 1.41672 2.13379 2.62682 2.01632 2.32349 2.01627 2.40552 1.64306 1.78171 1.60727 2.6931 2.32123 1.84789 1.96568 1.9985 1.72116 2.05985 1.16908 1.6615 1.33512 1.32889 2.27683 1.2381
8 1.14548 1.63505 2.1357 1.53917 1.33842 1.43952 2.1766 1.64777 1.13496 1.70644 1.44951 1.99899 1.54784
Miss
2.25064 4.4718 1.39018 1.03657 2.24678 2.77374 3.5765 5.07711 2.07222 1.6549 6.81197 6.09975 8.30551 11.0696 9.4426 14.3529 8.00023 5.3177 12.344 5.96975 5.12508 6.39164 4.13753 4.46301 2.30796 5.54328 2.08225 7.92008 4.76604 2.1
1758 2.55962 4.87613 5.74568 4.34719 3.94024 3.90569 3.39708 2.45094 5.04281 2.2104 3.78001 2.62336 2.84434 5.42102 2.83712 4.18396 3.78003 2.55404 4.71235 1.84971 4.22003 3.08679 4.70124 3.91975 1.97638 4.11698 6.50841 3.48752 1
.12765 2.79012 1.01872 3.68777 2.91158 1.48742 3.23253 1.5269 1.91237 4.33674 2.5527 3.45739 2.54281 2.90505 3.50954 3.27201 3.03885 4.23596 1.77428 3.27766 2.46953 2.54193 3.01571 1.99402 2.46267 2.77077 1.04719 3.83448 1.76592
3.65217 1.46864 1.64963 4.60571 1.38787 2.37122 2.10063 3.60823 2.3735 3.71401 2.15684 1.73166 2.32423
StdCARR:
Beat
1.22011 1.9943 2.00956 2.62774 3.31545 3.78609 4.14562 5.54245 4.21592 3.9089 3.69818 2.96032 5.34717 6.20781 6.15686 10.3396 14.8344 11.6927 12.0645 12.0636 10.0097 10.4377 11.38 12.316 12.3934 12.4716 13.7809 15.2385 15.6772 16
.372 12.886 11.1839 10.7318 11.6382 13.1448 13.2056 15.1159 14.5421 15.015 15.1019 15.0954 15.1766 14.7783 14.169 12.7966 13.4572 14.8563 15.5594 16.8762 18.7005 17.9329 17.7449 16.9407 18.7957 18.389 19.1965 18.9969 18.5594 16.5
802 16.0592 14.9369 14.077 14.1027 11.7335 11.8613 11.8111 12.6209 11.3914 11.4354 9.71925 7.76926 7.27576 5.26583 6.17829 8.46843 9.5608 8.94395 9.80789 10.5362 11.6665 11.4944 11.7472 11.6112 11.1528 11.4819 11.6739 12.1071 11
5757 12.0605 12.6349 12.5574 11.8946 12.3116 12.7242 13.7164 13.5783 12.7982 12.4669 11.6322 11.9051
```

AveAAR (N = 50)



AveCAAR (N = 50)



Improvements

1. Improve the stability of threading
2. Improve the speed of bootstrap
3. Use more complex but convenience structure
4. Try to use Excel to plot

Thank You!