Fall 2021
FRE-GY 6883 Financial Computing
Course Team Projects

General Requirement

You are required to do class projects in teams with 6 members. We will have total 5 teams in our class. You should elect one member of your team to be the team leader. Teams once formed cannot be changed midway through the project. The team leader is responsible to facilitate the planning of the project, and the entire team will plan the project under the guidance of your team leader. Planning involves identifying what should be done (tasks), who should do it (resources), when tasks should be done (time frames) and how tasks are best sequenced (dependencies).

Team leader is required to provide a brief weekly update of your team progress.

Each team will submit PowerPoint slides and all the project files including source codes and executables tar/zipped to our course Web site 3 days before the presentation day (12/18/2021 11:55pm for Tuesday Section, and 12/15/2021 11:55pm for Saturday Section). The PowerPoint presentation should include your research on Russell 1000 stocks for their earnings, a drawing of project design (UML), class declaration and data structures, program outputs. All the teams are requested to present and demonstrate their projects. Each team can resubmit one time on the presentation day. Your project will be judged by program efficiency, complexity, and the success of your demonstration and presentation. All the submission will be done by team leader.

Project Description
Evaluate the impact of quarterly earnings report on stock price movement

Programming Requirements:

- Use **liburl** to retrieve historical price data from eodhistoricaldata.com: A function retrieves the adjusted close prices for selected **Russell 1000 stocks** and **IWB** (Russell 1000 ETF used as market benchmark) into memory.
- Create a set of classes such as class for stock to handle EPS (earnings per share) estimate and price information.
- Use member functions or independent functions for all calculation. Overload a few arithmetic operators for vector/matrix.
- The stocks and their corresponding price information for each group should be stored in a STL map, with stock symbol as its keys.

- The expected AAR, AAR STD, and expected CAAR and CAAR STD for 3 groups are presented in a matrix. The row of the matrix is the group#, matrix columns are for AAR, AAR-STD, CAAR, CAAR-STD
- Use gnuplot to show the CAAR from all 3 groups in one graph.
- Your program should be able to:
    - Retrieve historical price data for all selected stocks. Parse the retrieved data for dates and adjusted closing prices.
    - Calculate AAR, AAR-STD, CAAR, CAAR-STD for each group
    - Populate the stock maps and AAR/CAAR matrix.
    - Show the gnuplot graph with CAAR for all 3 groups.
- Your program should have a menu of 5 options:
    - Enter N to retrieve 2N+1 days of historical price data for all stocks (you need to validate user input to make sure N >= 60).
    - Pull information for one stock from one group:
        - Daily Prices
        - Cumulative Daily Returns
        - The group the stock belongs to
        - Earning Announcement Date, Period Ending, Estimated, Reported Earnings, Surprise and Surprise %.
    - Show AAR, AAR-STD, CAAR and CAAR-STD for one group.
    - Show the gnuplot graph with CAAR for all 3 groups.
    - Exit your program.

Calculation Details:

1. Based on the 1st quarter of 2021 earnings announcement for Russell 1000 stocks (See the Earnings Announcements sheet), sort all the surprise% in ascending order, and split all the stocks into 3 groups with **relatively equivalent numbers** of stocks:
    i. Highest surprise group: Beat Estimate Group
    ii. Lowest surprise group: Miss Estimate Group
    iii. The rest stocks in between: Meet Estimate Group

2. Define day “zero” for a stock as the day the earning is announced.

3. Implement Bootstrapping:
    a. Randomly selecting 80 stocks from each group, total 240 stocks.

    b. Use libcurl lib to retrieve 2N+1 days of historical prices for Russell 1000 stocks and ETF ticker IWB (used as market benchmark) around the date of earning release (You could enhance our class example for this purpose). N is integer which must be greater or equal to 60, will be entered by users. Users will be warned if there are no enough historical prices for 2N+1.

c. For each stock calculate the daily returns R_{it} for N days before the day "zero" and N days after, such as t = -60, -59,…-1, 0, 1,…, 59, 60:

   $R_{it} = (Price_t – Price_{t-1})/ Price_{t-1}$

   Using adjusted daily closing price for your calculation

d. Calculate the corresponding daily return R_{mt} for IWB for the same days.

e. Define abnormal returns as the difference $AR_{it}= R_{it}–R_{mt.}$

f. Calculate average daily abnormal returns for each group of stocks (with M stocks, M = 80 in our case) for all 2N reference days:

$$AARt = \frac{1}{M}\sum_{i=1}^{M} ARit$$

g. Cumulate the returns on the first T days to CAAR:

   $CAAR = \sum_{t=-N+1}^{T} AARt$, T could be -N+1, -N+2, …, N-2, N-1, N

h. Repeat steps a to g 40 times to create 40 samplings and then calculate
   i. Average AAR and CAAR for 40 samplings for each group
   ii. Standard Deviation of AAR and CAAR for each group

4. Generate a gnuplot chart show the averaged CAAR of all three groups and discuss the impact the earning releases on their stock prices. Is there any conclusion you could draw from your project?

Project Tasks:

Task 1: Earnings research: sort stocks from Russell 1000 into 3 groups based on their earnings and EPS Estimate based Zacks.

Task 2: Project Design:

a) Create classes and data structure such as vectors, matrix, and maps.
b) Figure out how to handle historical price retrieval from eodhistoricaldata.com for all IWB 1000 stocks and the benchmark IWB, and parse the retrieved data?
c) Figure out how to implement your Bootstrap algorithm?
d) Write member function or independent functions (with operator overloading) for all the calculation.
e) Design and implement menu

Task 3: Figure out how to implement menu options and graphing your results with gnuplot.

Task 4: Divide the project into modules and assign team members working on each module.

Task 5: Module Integration and Testing

Task 6: Presentation Preparation.