## IEOR 4727 Project: Missing Historical Data and Insert it Back

Group Members: Guoan Wang, Ellen Sun

1. I. List of all library files and include files
   1. Missing Historical Data and Insert it Back
      1. main.cpp
      2. AbstractData.cpp, AbstractData.h
      3. TradeData.cpp, TradeData.h
      4. TickData.cpp, TickData.h
      5. loadRData.cpp, loadRData.h
      6. RetrieveRData.cpp, RetrieveRData.h
      7. RHashTable.cpp, RHashTable.h
      8. InsertMissingData.cpp, InsertMissingData.h
      9. TestCases.cpp, TestCases.h
      10. elasticsearch.cpp, elasticsearch.h (provided by Kai Fang)
      11. http.cpp, http.h (provided by Kai Fang)
      12. json.cpp, json.h (provided by Kai Fang)
   2. Simulate RData (optional, see already simulated RData samples used for testing in folder RData)
      1. main.cpp
      2. ran2\_mod.cpp (provided in class sample code)
      3. functions.h (provided in class sample code)
2. II. Project Summary
   1. Project Mission: Compare trading data provided in RData files and in ElasticSearch, find missing and inaccurate data on ElasticSearch and make appropriate corrections. Specifically, in Part B and C, we list purposes of classes mentioned above, together with general working flow of the program
   2. Program work flow
      1. Get access to RData files in computer based on user input;
      2. Store all RData files as Rcpp matrix/vector;
      3. Iterate over each RData files using a for loop, for each RData fIles:
      4. Store each RData files information in a hashtable, iterate over the hashtable to compare with ElasticSearch data.
      5. Repeat Step 4.
   3. Project detailed explained
      1. RetrieveRData.cpp: Get user input for desired starting date (yyyy/mm/dd) and ending date (yyyy/mm/dd), as well as product type needed for comparison. For product type names, user can choose to enter manually or provide a .txt file under the same directory as the RData files. Get access to all RData files in the computer according to user requirements.
      2. loadRData.cpp: Assume each RData file represents the Tick/Trade Data of one product in on day, and assume the file has n rows and 6 columns (time in milliseconds, Bid.Price, Bid.Size, Ask.Price, Ask.Size, Trade.Price, Volume). Use the RInside library and Rcpp library to load the file, and store the time column in a Rcpp vector, and the rest of the data in a Rcpp matrix.
      3. AbstractData.cpp, TradeData.cpp, TickData.cpp: Read the Rcpp Matrix and Vector row by row. Each row is converted into a TradeData Object, which is a derived class of AbstractData class, and AbstractData class is derived from json class. AbstractData class is the base class for TradeData and TickData class. TickData (Bid&Ask data) class is currently not in use, but if later we need to check TickData as well, this could be done easily.
      4. RHashTable.cpp: Store all TradeData object, each representing a row in RData files, into a hashtable (unordered\_map). The hashtable uses timestamp of each transaction as key, and each TradeData object as corresponding value.
      5. InsertMissingData.cpp: Utilized ElasticSerach C++ API (elasticsearch.cpp) to retrieve & insert data to ElasticSearch as JSON object. Looped over hashtable to search and compare each TradeData object.
3. III. Appendix: SimulateRData
   1. Mission: Simulate multiple RData files for testing.
   2. Method: Simulate the price of a product over one day, and serialize into an RData file. The model we use is Black-Scholes.
   3. To compile:
      1. you need to link RInside library manually. Refer to “<https://www.youtube.com/watch?v=sjiSaaNA8BY>” to link RInside package.
      2. If you are using XCode and no txt files or Rdata files are created, follow this link: <https://www.quora.com/Why-is-Xcode-not-writing-me-a-txt-file>