Algorithmic Trading

This problem is aimed at giving a flavor of various events and controls involved in real world trading. With the goal to design an optimal trading strategy for fulfilling client orders, while satisfying the constraints posed by marketplace.

Devise a trading algorithm to get the **Aggregate-best-trade-executions** (cheapest buys and highest sells) to fulfill the given client orders under certain constraints. Real time market data will be provided for the products to be traded. A Client comes in with the objective of buying / selling certain quantity on a basket of stocks [probably multiple requests during the trading time window]. Your goal is to help them get the best possible trades based on the market data available to you.

The scenario is as follows:

- You are given 3 different tradable products P1, P2, P3; such that P1 can be traded on exchange E1; P2 can be traded on exchanges E1 and E2; and P3 can be traded on exchanges E1, E2, E3, E4.

- Real time bid and ask prices and corresponding bid and ask quantities are available for these products from different exchanges as applicable. This information is available at an interval of one second. There may be intervals where this data is missing for one or more products.

- The client orders are given in the form of product to be traded, quantity to buy or sell, and time window during which the order has to be completed. You may split the original client order into chunks of orders for trading and may trade on any of the available exchanges.
  Note: If a client order is to buy a product, you can only buy that product till the order is completed, and not sell in between. Similarly if a client order is to sell a product, you cannot buy that product till the order is complete.

- You have to write a program for trading the given products on behalf of the client with the objective of getting cheapest buys and highest sells in such a way that the constraints given below are met.

Constraints

- ***Final Position:*** Net positions in all products should be equal to the net client orders at the end of the trading window, that is, all client orders should get completely filled.
- ***Order size limit:*** At any time interval you cannot sell more than the bid quantity, and you cannot buy more than the ask quantity; this holds for all exchanges and products.
- ***Trade size limit:*** At any interval, the number of stocks bought or sold on an exchange cannot exceed 1,000,000 for P1; 20,000 for P2; 20,000 for P3.
- ***Exchange position limit:*** The absolute number of stocks of a product cannot exceed 70,000 for P2; 140,000 for P3 (No such limit for P1).
- ***Restricted product:*** Typically, right after an important public announcement related to a company, its stock prices tend to move very rapidly. In order to avoid a possibility of undue advantage to someone closely related to the announcement, a list of products is maintained that cannot be traded for a specified time window. P1 is such a product that becomes restricted for certain time during the trading window (details given in **Input** section). P1 cannot be traded (bought or sold) during the time that it remains in the restricted trading list (RTL).

Evaluation

Your output should be submitted as a csv file which will be used to check for all the applicable constraints and calculate the total execution price for each buy and sell client order. The trade orders in your output should satisfy all the given constraints and fulfill the client orders in the specified time duration. The evaluation will be based on the approach and total execution price achieved for each order.
Note: For making a trade decision at a timestamp T, your solution can use any market information till timestamp T

only; and no market information after timestamp T. Any forward-looking solutions will be disqualified.

Dataset

- At the start of the competition, you will be given a sample dataset with market data for a duration of 120 minutes (t = 0 to t = 120 minutes for each interval of one second), and client order to be fulfilled in each interval of 30 minutes (starting from t = 30 minutes). You are free to use this data to set up you trading strategy/algorithm and write/test your code.
- Your code then needs to be run for five different datasets that will be provided 3 hours before the end of the competition. Each of these datasets will contain market data for a duration of 120 minutes (t = 0 to t = 120 minutes), and client orders to be fulfilled in each interval of 30 minutes (starting from t = 30 minutes). **Each of the five datasets and the initial sample dataset are independent from each other.**

Input

Filename	Fields*	Available from
Market_Data_Sample.csv	Timestamp, Exchange, Product, Bid Price, Bid Quantity, Ask Price, Ask Quantity	9pm, Sep 28, 2013
Client_Orders_Sample.csv	Start_time, End_time, Product, Quantity	9pm, Sep 28, 2013
Market_Data_A.csv	Timestamp, Exchange, Product, Bid Price, Bid Quantity, Ask Price, Ask Quantity	6pm, Sep 30, 2013
Market_Data_B.csv		
Market_Data_C.csv		
Market_Data_D.csv		
Market_Data_E.csv		
Client_Orders_A.csv	Start_time, End_time, Product, Quantity	6pm, Sep 30, 2013
Client_Orders_B.csv		
Client_Orders_C.csv		
Client_Orders_D.csv		
Client_Orders_E.csv		

Market Data

- Timestamp is given as Date hh:mm:ss.000 am/pm
- Bid and Ask Prices are given as decimal numbers rounded to 4 decimal places
- Bid and Ask quantities are integers
- *Market data will occasionally have RTL lines specified as below:
  - ***Timestamp1, RTL_begin, P1, 0, 0, 0, 0***
  - ***Timestamp2, RTL_end, P1, 0, 0, 0, 0***
  - From Timestamp1 to Timestamp2, P1 remains under RTL and thus cannot be traded

Client Orders

- Start_time and End_time are given as hh:mm; the client order has to be completed within this time window
- Quantity is positive for a buy order and negative for a sell order

Output

Zipped file to be submitted by 9pm, Sep 30, 2013 with the following contents:

Filename	Fields
ModelDocumentation	-
SourceCode	(zipped folder with all relevant files)
output_sample.csv	Timestamp, Exchange, Product, Quantity
output _A.csv	
output _B.csv	
output _C.csv	
output _D.csv	
output _E.csv	

- At each interval of one second during the trading time period, your code should give a trade order in the format: ***Timestamp, Exchange, Product, Quantity***

- Quantity is positive for a buy order and negative for a sell order. For example: a line from your output {*Timestamp3, E2, P3, -250}* will be interpreted as: Sell 250 stocks of product P3 on exchange E2 at Timestamp3 at the last available price; bid for sell order and ask for buy order.
- **Note:** If at an interval you do not wish to trade, you should still specify the timestamp in the output but with other fields as empty NULL)

- Model Documentation should contain following parts:
    - Description of the approach
    - Algorithm or pseudo code
    - Assumptions (if any)

Error Detection in Prices

You are given a history of end of day prices for 100 securities. For the purpose of this problem, a security can be taken to be either a stock or a bond. Each security is uniquely indicated by the ID number.
They are valued in various currencies as indicated by the 'currency' field. Please use publicly available Foreign Exchange Rates if required (mention the exact sources used in the model documentation). In addition, you are given the dollar worth of every security held in a portfolio (**holdings.csv**) as of Aug 26, 2013.

You are also given a fresh set of prices coming into the pricing database for the next day i.e. August 27, 2013. You are to:

a) Finding all breaks - potentially erroneous prices in the fresh set of prices.
b) Are all the breaks equally important? How does one measure their relative importance?

In addition:

a) Are all historical prices accurate? Briefly describe the approach you will take to classify all historical prices as accurate or inaccurate. Also produce a file listing out all the historical prices that are potentially inaccurate (**output_history.csv)**
b) What happens when there are stock splits? There is precisely one security that has undergone a stock split, can you identify which one? Does this change what the other files should look like? If so how? (Provide answer as part of Model Documentation)
c) Types A, B and C of securities are different in a fundamental way. A represents stocks whereas B and C represent bonds. C is more likely to have "jumps" in prices compared to B. Can you devise a statistical test that will help you "classify" a new product into A, B or C given only a history of prices? Describe your approach. (Provide answer as part of Model Doc)
d) What happens when there are notable market wide moves in prices on any given new day? Briefly describe the approach you will take. You do not need to produce a file. Provide answer as part of the model documentation

Input

Filename	Fields	Available from
history.csv	ID, date, price, currency, type	9pm, Sep 28, 2013
holding.csv	ID, USD value held	9pm, Sep 28, 2013
prices_A.csv	ID, price on Aug-27-2013	9pm, Sep 28, 2013
prices_B.csv	ID, price on Aug-27-2013	6pm, Sep 30, 2013
prices_C.csv	ID, price on Aug-27-2013	6pm, Sep 30, 2013

Output

Zipped file to be submitted by **9pm, Sep 30, 2013** with the following contents:

Filename	Fields
ModelDocumentation	-
SourceCode	(zipped folder with all relevant files)
output_A.csv	ID, Min Price, Max Price, Error Rank
output_B.csv	ID, Min Price, Max Price, Error Rank
output_C.csv	ID, Min Price, Max Price, Error Rank

output_history.csv	ID, Date

- Min Price and Max Price will be double values in the range beyond which the price is considered an Error.

- Date in output_history.csv should be in the same format as available in history.csv.
- Error Rank should be between 0 & N, inclusive. 0 will indicate no error. 1 (lowest importance) through N (highest importance) will stand for the relative importance of the N errors identified.

- Partial submissions i.e., without one/more of the output_* files will be accepted. However, there should be at least one output_* file in the submission. ModelDocumentation & SourceCode are mandatory.

Monte Carlo in Derivative Pricing

Derivatives are financial products which derive their price from simpler financial instruments (called underlying) such as stocks. The price of a derivative is usually a complicated function of the future price of the underlying instrument. To find the price of the derivative, we need to find the future distribution of the price of the underlying instrument.

Product

Let us assume we have a derivative called an option whose underlying is a stock. This derivative pays the following amount at a future fixed time T (which is called expiration time):

$V = 1 \; if \; S(T) > K$

$V = 0 \; otherwise$

where K is called the strike. This is a specific kind of option called "binary option". Note that S(T) is a random variable – we do not know its future value.

Model

To find the expected value of the option in the future, we need to know the distribution of the stock in the future. Let us assume that the price of the stock evolves in the following way:

The initial value of the stock is S(0). The value of the stock at any future time t is given by,

$S(t) = S(0)\exp(X(t))$

where X(t) is a auxiliary time dependent variable whose initial value X(0) = 0.

$dX(t) = \epsilon_t v dt$

$$\text{where } \epsilon_t = \epsilon_{t-dt} \; with \; probability \; 1 - \lambda dt$$

$$\text{or } \epsilon_t = -\epsilon_{t-dt} \; with \; probability \; \lambda dt$$

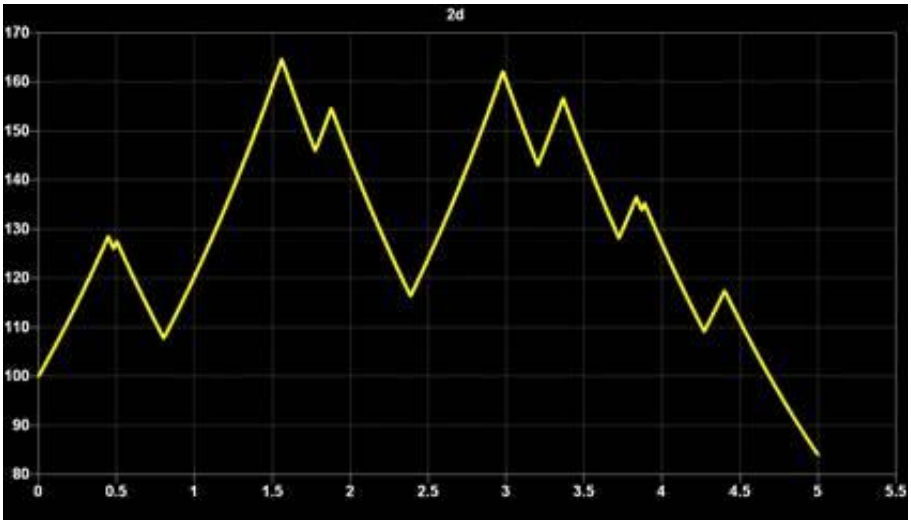
Basically the direction of "motion" changes at every time with probability λdt. Initial condition for dX(t) is as follows:

$dX(0) = vdt \; or - vdt \; with \; equal \; probability$

This means $\epsilon_0 = \pm 1$ with probability ½ each.

Example of the process

Consider the process for $X(t)$. X(t) can be thought of as position of a particle with initial velocity ±V with equal probability and flipping its direction of motion at any time t with probability λdt.



Suppose initial value of $\epsilon_t = 1$. Lets say that the first flip happens after n time steps. The value of $X(ndt)$ will be $v * ndt$. The flip happens with probability λdt and the process $X(t)$ has speed $-v$. It will continue with $-v$ after each dt with probability $1 - \lambda dt$ till a flip happens again with probability λdt, and so on. A sample "path" is shown in the above figure.

Assume that initial value of S(t) is S0 = 100. Note that initial values for X(t) and dX(t) are specified in the model description above.

Given this information, answer the following questions:

1. Given values of v and λ, find the mean, std dev, skew and excess kurtosis of the distribution of X(t) at time t=1. Also find the expected value of the binary option with strike K and expiration time T=1 in this case.
2. Given values of v and λ, find the mean, std dev, skew and excess kurtosis of the distribution of X(t) at time t=1, given that S(t) < B for all 0 < t < 1. Also find the expected value of the binary option with strike K and expiration time T=1 in this case.
3. You are given stock_data.csv containing stock prices for a fictitious stock for 5 years. Find values of v and λ with which you can fit the data to the above process.

Inputs to part 1 and 2 are given in inputs.csv
[Note: You can use Monte Carlo technique to find the above values.]

Input

Initial Values:

- S(0) = 100
- X(0) = 0
- dX(0) = ±v with equal probability

For part 1 and 2, the inputs will be v, λ, K, B. You have to ignore B for part 1. The inputs will be contained in the following files:

Filename	Fields	Available From
stock_data.csv	Time (yrs), Stock Price	9pm, Sep 28, 2013
input_A.csv	v, λ, K, B	9pm, Sep 28, 2013
input_B.csv	v, λ, K, B	6pm, Spe 30, 2013

A sample output file (sample_output_A.csv) will also be provided for first 6 inputs in input_A.csv.

Output

Zipped file to be submitted by **9pm, Sep 30, 2013** containing:

Filename*	Fields
ModelDocumentation	Format as described below
SourceCode	(zipped folder with all relevant files)
output_A.csv	ID, v, λ, K, B, Mean, Std Dev, Skewness, Kurtosis, Expected Value
output_B.csv	ID, v, λ, K, B, Mean, Std dev, Skewness, Kurtosis, Expected Value

*First line of **output_A.csv** should contain 3,v, λ where v and λ are the calculated parameters for part 3.

- ID should be 1 or 2 corresponding to output for parts 1 and 2 respectively. You have to submit an output for part 1 and 2 for each set of v, λ, K, B.

- Partial submissions, without one or more outputs will be accepted. Model documentation & SourceCode is mandatory.

Model Documentation Format

The model documentation should include the following parts:

1. Description of the approach
2. Analysis
3. Assumptions (if any)

Definitions

- Mean:

$$\bar{X} = \sum_{i=1}^{N} \frac{X_i}{N}$$

- Std Dev σ:

$$\sigma^2 = \sum_{i=1}^{N} \frac{(X_i - \bar{X})^2}{N}$$

- Skew:

$$Skewness = \sum_{i=1}^{N} \frac{(X_i - \bar{X})^3}{N\sigma^3}$$

- Kurtosis:

$$Kurtosis = \sum_{i}^{N} \frac{(X_i - \bar{X})^4}{N\sigma^4} - 3$$

References

[1]http://en.wikipedia.org/wiki/Derivative_(finance)

[2]http://en.wikipedia.org/wiki/Monte_Carlo_methods_in_finance