Final Project
Due: Dec 10, 2015

Choose one of the following two topics to implement for your final project.

- **Topic 1: Gamma Scalping Trading Strategy (100 points)**

  Design a C++ or a Python program to implement the Gamma scalping trading strategy involving one underlying stock and two at-the-money (ATM) options (one Call and one Put) as described in the posted PPT lecture file.

  Let the Gamma scalping testing period be from start date t_0 to end date t_N. At the start date t_0, an initial capital in the amount of \$1,000,000.00 is deposited in a trading account. For simplicity, assume the trading account does not earn interest from the cash in the account, and additional cash can be borrowed from bank at risk free interest rate if needed.

  The implemented strategy is expected to contain the following three main components.

  1. **Entry/Exit Signal Generator**
     - At t_0, instead of gamma scalping right away, we start observing the moving average of volatility to see when the underlying volatility is about to explode, so that we can find profitable trading periods to do gamma scalping. We automate such a process by implementing an Entry/Exit Signal Generator.
     - Pick an integer $w > 0$ to be the length of time window over which the moving average of volatility is computed. Let $MA_w(\sigma_t)$ denote the average volatility at time t over periods $t-1$, $t-2$, $\ldots$, $t-w$.
     - Trade entry/exit signal generator.
       * Pick two positive parameters: lower threshold σ_{entry} and higher threshold σ_{exit} for the signal generator. $\sigma_{entry} < \sigma_{exit}$.
       * Entry/exit signal rules:
         · Entry signal: If the moving average of volatility $MA(\sigma_t)$ goes down to the lower threshold, namely, $MA(\sigma_t) < \sigma_{entry}$, then the signal generator will output an entry signal. It indicates the volatility is in the low range and may subsequently go up, and so a gamma scalping trade is initiated in this period t_{entry}.
         · Exit signal: After a gamma scalping position is opened, if the moving average of volatility reaches up to the higher threshold, namely, $MA(\sigma_t) > \sigma_{exit}$, or expiration date is no more than 20 days later, the a trade exit signal is generated. All the positions of this gamma scalp trade are closed immediately (namely, close the underlying positions and sell call and put) to lock our profits. After that, we wait until the next entry signal.
       * The entry/exit signal generator might use different sources of volatilities to calculate σ_t. We test two sources:
         · Historical volatility: sample standard deviation of last m underlying stock prices (or, log-returns):
         $$\sigma_t = \sigma_m(S_t)$$
         · The average of implied volatility of at-the-money call and put options:
         $$\sigma_t = (IV_t(Call_{ATM}) + IV_t(Put_{ATM}))/2$$
     - **Note**: Between t_0 and t_N, the gamma scalping trade may be carried out for multiple times, depending on the choice of the entry/exit signal thresholds.
  2. **Portfolio Rebalancing Engine**

- Select one trading parameter, $\bar{\Delta}$, which represents the threshold for rebalancing the total Delta position of the gamma scalping trade to 0. For example, $\bar{\Delta} = 0.15$.
- After a trade entry signal is generated in period t_{entry}, the gamma scalping trade is opened by purchasing K shares of ATM straddles: namely, simultaneously purchasing K shares of Call and K shares of Put which have the same strike price equaling the current underlying stock price, and the identical expiration date T.
- Compute the Deltas of 1 share of call ($\Delta_{Call}(t)$) and 1 share of put ($\Delta_{Put}(t)$) at time t (using Black-Scholes implied volatility function).
- If $-\bar{\Delta} \leq K * (\Delta_{Call}(t) + \Delta_{Put}(t)) \leq \bar{\Delta}$, then do nothing. If otherwise, long ($\Delta_{stock}(t)$*100 shares) of the underlying stocks to make the overall position delta neutral, where $\Delta_{stock}(t) \equiv -K * (\Delta_{Call}(t) + \Delta_{Put}(t))$ (If the sign of $\Delta_{stock}(t)$ is negative, then the trading action is to short). Such a portfolio is gamma positive.
- For each passing period t, we calculate the portfolio Delta $\Delta_t = \Delta_{Call}(t)+\Delta_{Put}(t)+\Delta_{stock}(t)$. If $|\Delta_t| > \bar{\Delta}$, then buy/sell Δ_t shares of stock to make the portfolio Delta equal 0. The position of the underlying stock is updated to $-1 * \Delta_t + \Delta_{stock}(t)$.
- We keep adjusting the number of shares of the stocks in the gamma scalping trade portfolio until the exit signal or expiration date, while recording all of the long/short actions and the corresponding trading gain/loss in each period t.
- **Transactions costs**: $TC_{cost} = 0.001$ of the transaction dollar amount. Note that the value of 1 share of Call/Put is equal to 100*(Price per share of Call/Put).
- **Note**: When selecting the expiration date of the Call and the Put in forming the straddle position, one can choose the options with the smallest time to expiration (T) among all available options with $T_{min} \leq T \leq T_{max}$, where T_{min} and T_{max} are pre-specified parameters representing the range of expiration days, say, 60 days to 120 days, respectively. When using ATM implied volatility as signal generator, follow the same rules when selecting option to calculate implied volatility.
- **Note**: After t_{entry}, when calculating implied volatility of ATM calls and puts for signal generator, the ATM straddle position we already purchased on t_{entry} might not be ATM anymore during the trading period, because S_t keeps changing and ATM option is the option with strike price closest to current underlying price. So for each day after t_{entry}, we need to find which option is at the money.

3. **Backtesting and Performance Evaluator**

   During the entire testing period from t_0 to t_N, we calculate the annualized daily return of gamma scalping strategy R_t and evaluate the performance of current parameter setting by the following criteria:
   - Total profit/loss
   - Average daily return (annualized)
   - Sharpe Ratio
   - Maximum Drawdowns (MDD)

Data and Requirements

- Data

  Same format as midterm. An extra underlying (SPY) is provided.
- Input

  There are three ways to pre-specify input parameters: given at the beginning of the program, prompt user to input, read a configuration file. The input parameters are:
  * General setting: choices of GOOG or SPY, t_0, t_N, K (initially set to 1).
  * Signal Generator: σ_{entry},σ_{exit}, w, m, choices of historical or implied volatility
  * Portfolio Rebalancing Engine: $\bar{\Delta}$, T_{min}, T_{max}

- Output

  Write all the output in a table form into a csv file "result.csv". The first column of the csv file is the date ranging from t_0 to t_N.

  The header of the table shall contain date, underlying price, σ_t, $MA_k(\sigma_t)$, signal, strike price, time to expiration, call price, put price, call IV, put IV, call Delta, put Delta, call Delta+put Delta, call Gamma, put Gamma, stock shares, Position Delta, Action, Realized profit, Position Total Value, Return, Total Wealth. If ATM implied volatility is selected as signal generator, include ATM call and put price in the header. The row index is from t_0 to t_N. During the non-trading period, only need to present signal related data.

  Output the performance statistics to a csv file "performance.csv". It contains the total profit/loss, average daily return, Sharpe ratio and maximum drawdowns.

- **Bonus Problem (20 points)**

  - Collect minute-by-minute option prices and underlying stock prices for GOOG and SPY between 9:30 to 16:00 from 12/1/2015 to 12/10/2015. (This part can be done by Python script in HW5).
  - Run your implementation of the Gamma Scalping Strategy over the minute-by-minute data and report results.

- **Deliverables**

  A final report (in original Word or latex format, do not submit pdf file) is expected and it shall contain the following parts.

  - Problem description.
  - Details on the implementation framework and code structure description
  - The performance of the trading strategy under the best set of trading parameters and how the best trading parameter set is obtained.
  - Use tables to illustrate the percentage of trades being winning/losing ones, the average profit/loss per trade, the annualized rate of return and the standard deviation of the daily return, and the Sharpe ratio. Use figures to show the cumulative growth rate of account value (with starting value being $1 and the distribution of the daily returns of the account value.

  Submit all the program codes/documentations as ONE zip file containing the final report and a sub folder containing all the codes and input configuration file. In each folder, write a readme file to record the compiler and library you use and any other necessary information to run your codes, i.e. how to fill in your parameter configuration file. Use relative path when reading data file.

  Also document the estimated time to run your program if you search all combinations of parameters to find the best trading parameters by brute force. If the running time is more than one hour, a relatively small test case of a parameter domain shall be provided which contains the best trading parameter set.

- **Topic 2: Multi-factor Portfolio Trading Strategy (100 points)**

  Consider the problem of investing a total amount of $10,000,000 in a large universe of stocks. The Chinese stocks traded on the Shanghai Stock Exchange and Shenzhen Stock Exchange (over 2000 stocks) are chosen to be the entire universe.

  Suppose a subset of 1000 (or 1500) stocks are selected from the entire universe. At every rebalancing time point, a small group of 100 to 150 stocks are selected out of the 1000 (or 1500) stocks during the in-sample backtesting period based on the following criterion for long position.

  - **Stock selection criteria**
    - Criterion 1: The market capitalization needs to be no less than 500,000,000 (RMB)
    - Criterion 2: The average daily trading volume over the past 15 business-days needs to be no less than 1,000,000.
    - Criterion 3: Computing M-score based on a group of factors for each stock i which passes Criterion 1 and 2. Let $C^i(t)$ denote the price of stock i in period t. The group of factors are described below.
      - Factor F_1: Price to Book ratio (PB)
      - Factor F_2: Price to CashFlow ratio (PCF)
      - Factor F_3: Price to Earning ratio (PE)
      - Factor F_4: Price to Sales ratio: PS
      - Factor F_5: n-period momentum factor (PM): $PM^i(t,) = \ln \frac{C^i(t-1)}{C^i(t-n)}$; (n = 5)
      - Factor F_6: m-period reversion factor (PRev): $PR^i(t) = \ln \frac{C^i(t-m)}{C^i(t-1)}$; (m = 20)
      - Factor F_7: L-period log-return volatility (Vol): $Vol^i(t) = \sigma_L^i(r(t-1))$ where $\sigma_L^i(r(t-1))$ denotes the annualized standard deviation of the log-return of stock i over time window $[t-L, t-1]$.

      Choose a weight vector $w \equiv (w_1, w_2, \cdots, w_7)$. For example, $w = (0, 0, 0, 0, 0.5, 0.5, 0)$ or $w = (1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7)$. M-score for stock i is calculated as:

      $$M_w^i(t) \equiv w_1 \cdot F_1^i(t) + w_2 \cdot F_2^i(t) + \cdots + w_7 \cdot F_7^i(t).$$

  - **Portfolio construction**
    - At each time point t, select the best K stocks with highest M-score. Initially, choose $K = 100$.
    - Put 1% of total account value at time t into each of the 100 stocks.
    - Transaction cost is $TC_{cost} = 0.001$ of total dollar amount of transactions.
  - **Backtesting and Performance Evaluator** During the entire testing period from t_0 to t_N, calculate the annualized daily return of this factor-based portfolio strategy R_t and evaluate the performance of a given set of trading parameters by the following criteria:
    - Total profit/loss
    - Average daily return (annualized)
    - Sharpe Ratio
    - Maximum Drawdowns (MDD)
  - **Data and Implementation Expectation**
    - Ticker universe and the fundamental data are provided as csv files in the Resources on $T-Squares$.
    - You are expected to download all the price data from Yahoo finance and save them in ./data folder in your project directory.
    - Your implementation shall contain at least the following classes.
      1. SecurityData Class: contain all the relevant time series data for a given universe of tickers.
      2. Strategy Class: generate buy/sell signals for a universe of securities and their corresponding data

3. Portfolio Class: generate portfolio of securities based on the buy/sell signals generated in Strategy Class.

* **Strategy input parameters**:
  - Score weight vector $w = (w_1, w_2, \cdots, w_7)$ for computing the M-score. This can be determined by running cross-sectional regression of daily (or weekly) return over the 7 factors.
  - n value for PM, m value for PR and L value for Vol.

– **Bonus Problem (20 points)**

* At every rebalance time t, use the values of 000300.ss as market index in the past 100 days to compute the beta of each stock with respect to this market index using the CAPM model.
* Calculate the total beta of the 100 stocks with the highest M-scores at time t, denoted by $\beta_{port}(t)$. Add a short position of -1*$\beta_{port}(t)$ shares of 000300.ss to the portfolio of 100 stocks so that the overall portfolio has a 0-beta with respect to 000300.ss. Report the performance of this new portfolio and compare it with the original portfolio.

– **Deliverables**

A final report (submit original Word or latex files, donnot submit pdf file) shall be submitted. It shall contain the following parts.

* Problem description.
* Details on the implementation framework and code structure description
* The performance of the trading strategy under the best set of **strategy input parameters** and how the best trading parameter set is obtained.
* Use tables to illustrate the percentage of trades being winning/losing ones, the average profit/loss per trade, the annualized rate of return and the standard deviation of the daily return, and the Sharpe ratio. Use figures to show the cumulative growth rate of account value (with starting value being $1 and the distribution of the daily returns of the account value.

Submit all the program codes/documentations as ONE zip file containing the final report and a sub folder containing all the codes and input configuration file. In each folder, write a readme file to record the compiler and library you use and any other necessary information to run your codes, i.e. how to fill in your parameter configuration file. Use relative path when reading data file.

Also document the estimated time to run your program if you search all combinations of parameters to find the best trading parameters by brute force. If the running time is more than one hour, provide a relatively small test case of a parameter domain which contains the best trading parameter set.