# Assignment-1

Given the daily asset value of portfolio A and B (portfolio.csv), calculate the normalized annual return, annual volatility, max drawdown, sharp ratio, sortino ratio, VaR of each portfolio. Analyze the relation between two portfolios, using at least three methods.

## Answer:

The complete solution is in ./Problem\_1/; running ./Problem\_1/main.py will generate all the required results. This report summarizes the same results from the outputs[[1]](#footnote-1) that can be found in ./Problem\_1/outputs/.

## Normalized annual return, volatility, max drawdown, sharpe, sortino, VaR of each portfolio

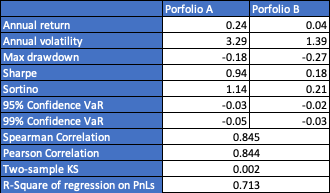
Trading day = 260 is applied to all calculations, since the data has 520 trading prices in total.

To understand the relation between two portfolios, three analyses are done on the daily PnL of those two portfolios: correlation (Spearman, Pearson), two-sample KS test (H0: two samples are drawn from the same distribution), and linear regression.

There is a strong correlation between daily PnLs of those two portfolios. The linear regression gives similar but more information, indicating a strong “linear” relation between the two portfolios.

A simple linear regression is thus conducted on those two PnLs. The goodness-of-fit measure R square is ~0.7, telling us the linear model can be a reasonably good choice for explaining the relation between the two portfolio. In addition, other metrics, e.g., F-statistics, T-statistics also indicate us the effectiveness of such a simple linear model.

Additionally, a p-value = 0.002 from a two sample KS test shows us the samples are not from the same distribution (the null is rejected, since p-value is less than the common 0.05 significance level).



## Scatter Plot

The scatter plot and linear regression fitted straight line are shown below.

Chart, scatter chart

Description automatically generated

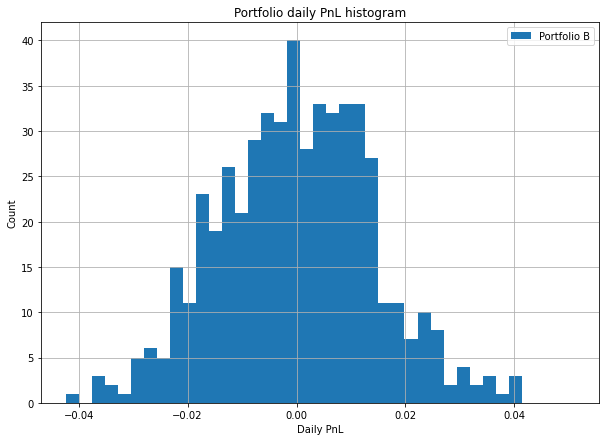
Table

Description automatically generated

## Daily PnL Histogram

Chart, histogram

Description automatically generated



## Max drawdown

Graphical user interface, chart, line chart

Description automatically generated

Figure Max drawdown: prices, cumulative max, through value of Portfolio A

Chart, histogram

Description automatically generated

Figure Max drawdown: prices, cumulative max, through value of Portfolio B

# Assignment-2

Give the hdf5 file, please load the data from the file (sample.h5) and store in a pandas DataFrame.

## Answer:

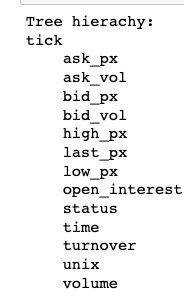
The solution is in the jupyter file ./Problem\_2/main.py and the output /outputs/sample.csv.

The sample format is shown below.

Table, Excel

Description automatically generated

The file hierarchy can printed by the Converter::\_str\_, shown as below:



# Assignment-3

Write function/class to calculate the vanilla option price and Greeks (Delta, Gamma, Rho, Vega), using Black-Scholes and Trinomial-Tree models.

## Results

All the results can be generated by running ./Problem\_3/main.py.

1. outputs/euro\_call\_put.csv
2. outputs/am\_call\_put.csv
3. outputs/implied\_vol.csv
4. Convergence plots:
   1. outputs/S\_20\_T\_3\_r\_0.05\_sigma\_0.3\_q\_0.02\_K\_25\_payoff\_call\_n\_200.jpg
   2. outputs/S\_20\_T\_3\_r\_0.05\_sigma\_0.3\_q\_0.02\_K\_25\_payoff\_put\_n\_200.jpg
   3. outputs/S\_42\_T\_0.75\_K\_40\_r\_0.04\_q\_0.08\_sigma\_0.35\_payoff\_call\_n\_200.jpg
   4. outputs/S\_100\_T\_3\_r\_0.15\_sigma\_0.5\_q\_0.04\_K\_65\_payoff\_call\_n\_200.jpg
   5. outputs/S\_127\_T\_1\_r\_0.1\_sigma\_0.2\_q\_0.03\_K\_130\_payoff\_put\_n\_200.jpg

## European Call and Put Option

The results for vanilla option price, Greeks (Delta, Gamma, Theta, Rho and Vega) are summarized in table 1, 2.

The code which generates the Black-Scholes formula results are saved in in ./utitls/BlackScholesModel.py, the analytical formula used in the implementation can be found in *“Section2.13, The Complete Guide to Option Pricing Formulas (2nd Edition), by Espen Gaarder Haug, PhD”.*

The column names start with “Tree\_” , e.g., Tree\_Euro(call) in the tables are made by the trinomial model, which can be found in the class defined in ./utils/TrinomidalModel.py.

The gamma and theta have same values, as expected. As a reference, more tests, e.g., call-put parity, can be found in jupyter notebook Question-3.ipynb.

Table

Description automatically generated

Table Euro Call and Put (BS, Tree Model, R result)

## American Call and Put Option (with optimal early exercise setup)

The American option prices, and Greeks are generated by the same file ./utils/TrinomidalModel.py, but with different set up, i.e., by changing payoff and exercise condition (as two parameters passed by user).

Unfortunately, it’s not easy to find reliable, open-source American option calculator to compare with the Trinomial Model results. The red number in the table indicates some R packages may have some unexpected results.

Table

Description automatically generated

Table American Option vs R package result(call option only)[[2]](#footnote-2)

## Implied Volatility

To obtain implied volatility, two methods (Newton’s, and bisection) are used for BS and Tree model. There are 6 tests are set up, with different parameters to include all models (call, put, Eruo, American). The meta data can be found in ./utils/WriteOut.py::save\_implied\_vol.

The implied volatilities **all match** with their true sigma well and their convergence speed are fast.

Table

Description automatically generated

Table

Description automatically generated

Table Implied Volatility from Tree Model and BS.

## Convergence Plots (with different steps in Trinomial Model)

In the convergence test, the Trinomial Model results with different steps, i.e., from 0 ~ 200 steps (1 step=T/dt), are used to compute initial option price and the results are compared with BS formula.

The tree model convergence results, and overall look close to BS model even with small steps and it quickly give pretty good results, after around 100~200 steps.

Note only Euro option results are compared in the convergence test.

Chart, line chart

Description automatically generated

Equation S=20, T=3, r=0.05, sigma=0.3, q=0.02, K=25, Call

Chart, line chart

Description automatically generated

Equation S=20, T=3, r=0.05, sigma=0.3, q=0.02, K=25, Put

Chart

Description automatically generated

Equation S=42, T=0.75, r=0.04, sigma=0.35, q=0.08, K=40, Call

Chart, line chart

Description automatically generated

Equation S=100, T=3, r=0.15, sigma=0.5, q=0.04, K=65, Call

Chart, line chart

Description automatically generated

Equation S=127, T=1, r=0.1, sigma=0.2, q=0.03, K=130, Put

# Assignment-4

Write a TCP client and host class, which send heart beat messages to each other. The host shall allows multiple client connections simultaneous. The client and host shall be properly constructed and closed, and managed by different processes.

## Explains:

Two modules are created: MyMiniClient.py and MyMiniServer.py.

They do the following things:

1. MyMiniServer::connect creates a TCP/IP socket and bind the socket to the port given by user.
2. The main part of the server calls, is the select method, in which the server uses to monitor communication channels. Three arguments are passed to select. The first is the a list of object to be checked for incoming data, the second checks the objects that are meant to receive outgoing data from the server. The last list may be used for error checking.
3. A class variable last\_write\_time is created to record the last write time for each individual client that connects to the server. If there is no response from the server for a certain period (default 10s), then server starts to send a “heart beat test” message to the corresponding client. If the client hasn’t replied for more than 10 times (max\_failure), then server will remove the client form its monitoring list.
4. Multiple clients can be started in parallel, the server will be able to distribute the message from all clients to queues, waiting for the server to reply to the clients. If the server does not reply back to the client, the client will stop.
5. Client can shutdown themselves, and server will just remove the clients from it’s monitoring list.

The log.txt shows the logger info from the server for one test. The red parts explain the log info.

2021-12-22 04:49:27.078 INFO MyMiniServer - connect: starting up on localhost port 8090 --> start communication  
2021-12-22 04:49:27.078 INFO MyMiniServer - run: waiting for connect...  
2021-12-22 04:49:45.128 INFO MyMiniServer - run: connection from ('127.0.0.1', 62121) ---> client1 connected  
2021-12-22 04:49:47.130 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62121) --> got msg from 1  
2021-12-22 04:49:47.130 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62121) --> send back to 1  
2021-12-22 04:49:50.135 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62121) --> got msg from 1 again  
2021-12-22 04:49:50.135 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62121) --> send back again to 1  
2021-12-22 04:50:02.143 INFO MyMiniServer - run: haven't receive data from ('127.0.0.1', 62121) for more than 10s --> no msg from 1 anymore  
2021-12-22 04:50:02.143 INFO MyMiniServer - run: send msg [heart beat test from server] to ('127.0.0.1', 62121) --> start to send heartbeat to 1  
2021-12-22 04:50:05.145 INFO MyMiniServer - run: connection from ('127.0.0.1', 62124) --> in the same time, client 2 connectes  
2021-12-22 04:50:08.150 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62124) --> got msg from 2  
2021-12-22 04:50:11.151 INFO MyMiniServer - run: client ('127.0.0.1', 62121) disconnect, closing. --> client 1 shutdown itself  
2021-12-22 04:50:11.151 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62124) --> send msg to 2  
2021-12-22 04:50:14.154 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62124) --> got second msg from 2  
2021-12-22 04:50:14.154 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62124) --> send back to 2  
2021-12-22 04:50:26.165 INFO MyMiniServer - run: haven't receive data from ('127.0.0.1', 62124) for more than 10s --> no more msg from 2  
2021-12-22 04:50:26.166 INFO MyMiniServer - run: send msg [heart beat test from server] to ('127.0.0.1', 62124) --> send heart beat to 2  
2021-12-22 04:50:35.173 INFO MyMiniServer - run: connection from ('127.0.0.1', 62134) --> client 3 connectes  
2021-12-22 04:50:35.173 INFO MyMiniServer - run: client ('127.0.0.1', 62124) disconnect, closing. --> client 2 disconnect  
2021-12-22 04:50:36.426 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62134) --> got msg from client 3  
2021-12-22 04:50:36.427 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62134) --> send msg back to client 3  
2021-12-22 04:50:39.432 INFO MyMiniServer - run: received msg[hello~] from ('127.0.0.1', 62134) --> got second msg from 3  
2021-12-22 04:50:39.432 INFO MyMiniServer - run: send msg [hello~] to ('127.0.0.1', 62134) --> send second msg back to 3  
2021-12-22 04:50:42.433 INFO MyMiniServer - run: client ('127.0.0.1', 62134) disconnect, closing. --> all clients disconnect from their sides, server waiting...

Client 1:  
connecting to localhost port 8090  
('127.0.0.1', 62121): received from server, msg [hello~] --> received the hello msg back from the server  
('127.0.0.1', 62121): received from server, msg [hello~] --> received the second hello msg back  
('127.0.0.1', 62121): received from server, msg [heart beat test from server] --> the heart beat msg from server, the client will be shutdown by ctrl+c after this  
  
Client 2:  
connecting to localhost port 8090  
('127.0.0.1', 62124): received from server, msg [hello~]  
('127.0.0.1', 62124): received from server, msg [hello~]  
('127.0.0.1', 62124): received from server, msg [heart beat test from server]  
  
Client 3:  
connecting to localhost port 8090  
('127.0.0.1', 62134): received from server, msg [hello~]  
('127.0.0.1', 62134): received from server, msg [hello~]

1. A jupyter notebook ./Problem\_1/Question-1.ipynb is saved as a reference, I found it come into handy when look through the plots. [↑](#footnote-ref-1)
2. The table compares the TreeModel results with a R package called “AmericanCallOpt”, the theta, which defined as , should be negative for American option, but the result from the R package is a relatively big positive number. [↑](#footnote-ref-2)