NYU

Pair Trading Project

FRE 7831 Topic in Financial and Risk Engineering

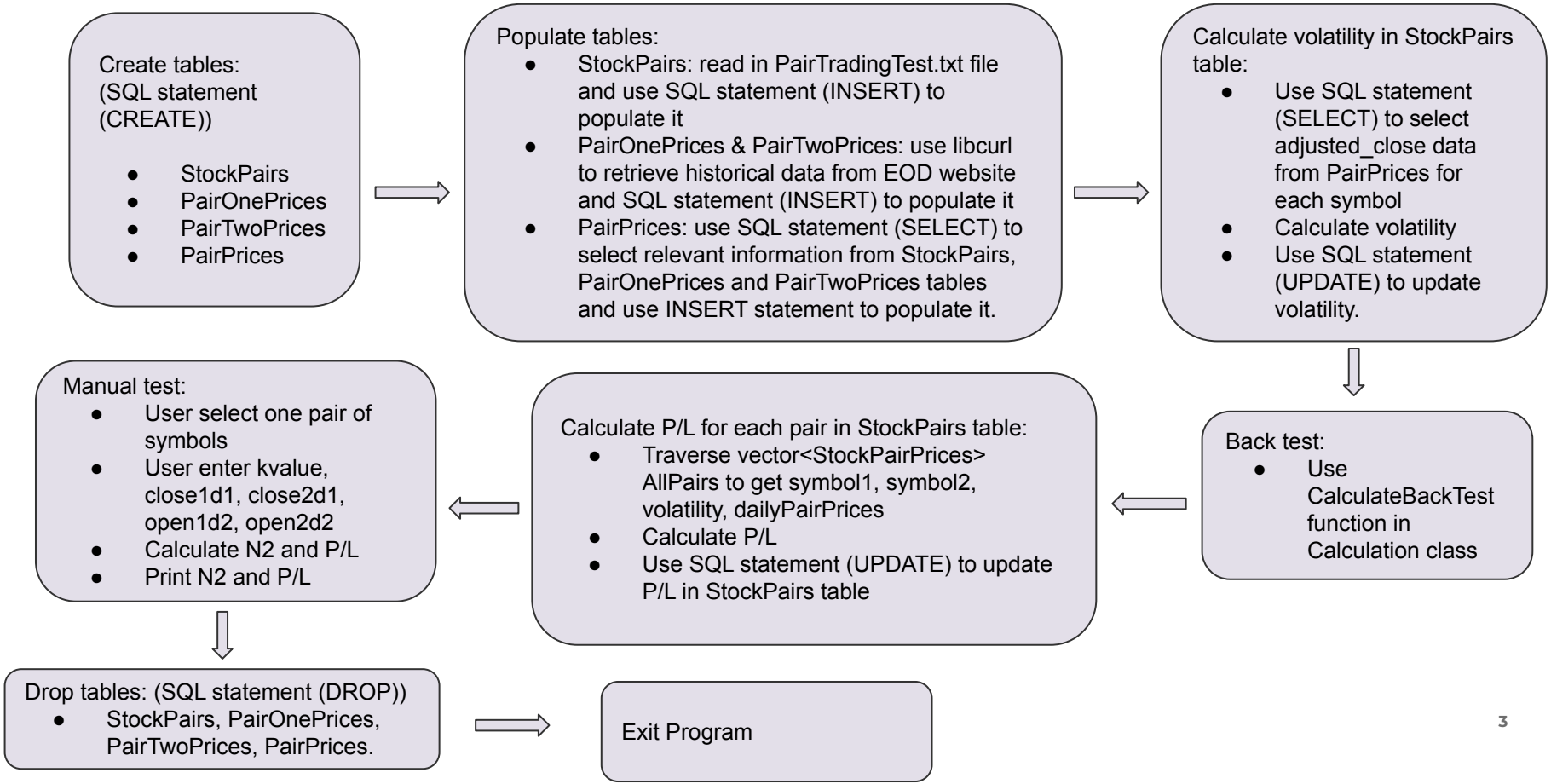
Spring 2022

PRESENTED BY Team 2 Jie Yu, Tianjing Cao, Kuiming Wang, Teresa Duan

Summary

Pair trading is a popular trading strategy. One of the easy version of it is to assume that the ratio of two prices is a mean-reverting process. In this project, we test the performance of this strategy using 18 pairs of US stocks. We use the market data retrieved from EOD and stored in the database to calculate the total P&L of 2022. We will see that most pairs can not generate a positive P&L. Hence, the project shows that pair trading by simple ratio can not guarantee a positive profit.

Program Flow Chart



Task Allocation

- A - Create and populate Pair table
- B - Retrieve and populate historical data for each stock
- C - Create PairPrices table
- D - Calculate Volatility
- E - Back Test
- F - Calculate profit and loss for each pair
- G - Manual testing
- H - Drop all the tables
- X - Exit

Jie Yu:ABCDH, Tianjing Cao:F, Kuiming Wang:G, Teresa Duan:E

NYU

SQL Statements

Create, Drop, Insert

- StockPairs
- PairOnePrices
- PairTwoPrices
- PairPrices

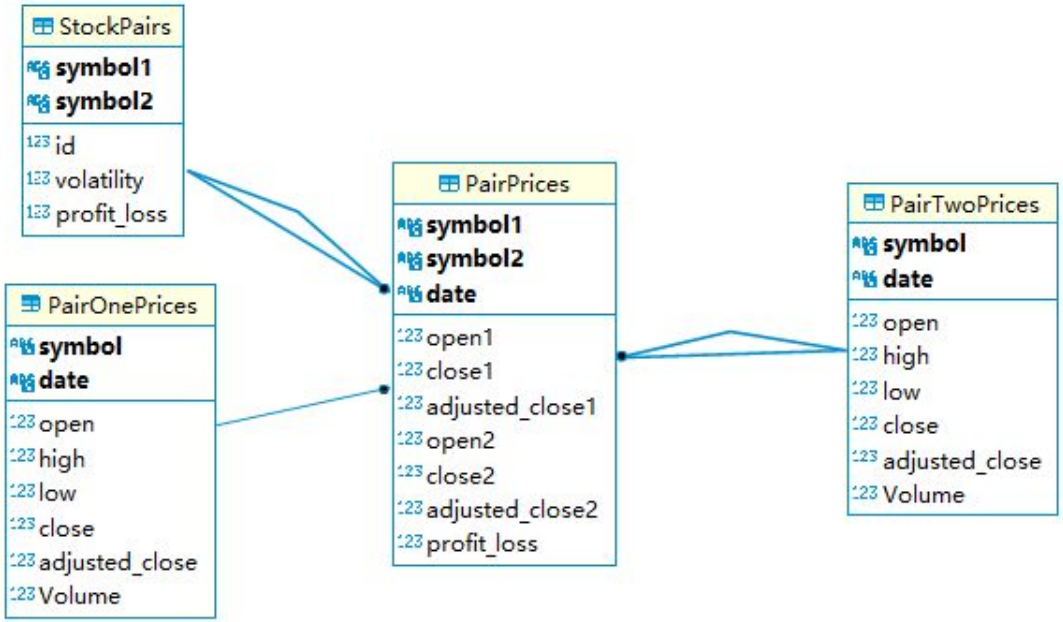
SELECT

- StockPairs
- PairOnePrices
- PairTwoPrices
- PairPrices

Update

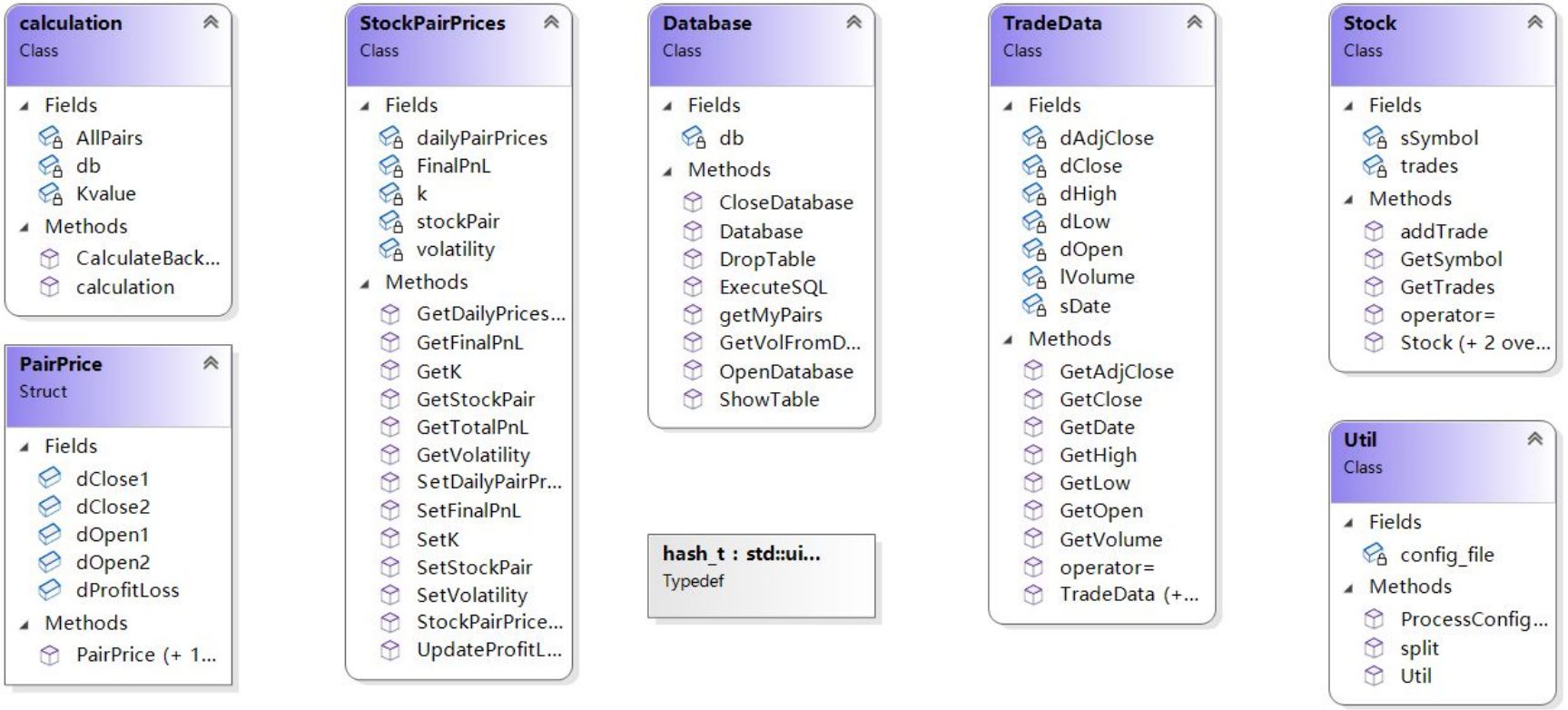
- StockPairs
- PairPrices

E-R diagram



NYU

UML Design


calculation
Class
Fields
AllPairs
db
Kvalue
Methods
CalculateBack...
calculation
PairPrice
Struct
Fields
dClose1
dClose2
dOpen1
dOpen2
dProfitLoss
Methods
PairPrice (+ 1...
StockPairPrices
Class
Fields
dailyPairPrices
FinalPnL
k
stockPair
volatility
Methods
GetDailyPrices...
GetFinalPnL
GetK
GetStockPair
GetTotalPnL
GetVolatility
SetDailyPairPr...
SetFinalPnL
SetK
SetStockPair
SetVolatility
StockPairPrice...
UpdateProfitL...
Database
Class
Fields
db
Methods
CloseDatabase
Database
DropTable
ExecuteSQL
getMyPairs
GetVolFromD...
OpenDatabase
ShowTable
hash_t : std::ui...
Typedef
TradeData
Class
Fields
dAdjClose
dClose
dHigh
dLow
dOpen
lVolume
sDate
Methods
GetAdjClose
GetClose
GetDate
GetHigh
GetLow
GetOpen
GetVolume
operator=
TradeData (+...
Stock
Class
Fields
sSymbol
trades
Methods
addTrade
GetSymbol
GetTrades
operator=
Stock (+ 2 ove...
Util
Class
Fields
config_file
Methods
ProcessConfig...
split
Util

Added Functions and Container

- Database.cpp
  - int GetVolFromDatabase (sqlite3* db, vector<double>& vols);
  - vector<StockPairPrices> getMyPairs(sqlite3* db);
- Calculation.cpp
  - int CalculateBackTest(sqlite3* db, vector <StockPairPrices>& AllPairs,double Kvalue);

- container
  - Map: map<string, Stock> stockMap
  - Set: set<string> symbol1, symbol2(no duplicate in table pairprice)
  - Vector: vector<string> symbolVec1, symbolVec2
  - Vector: vector<StockPairPrices> AllPairs

NYU

A - Create and populate Pair table

- SQL create statements for StockPairs, PairOnePrices, PairTwoPrices(drop table if existed)
- Read information from PairTrading.txt, for every line insert into StockPairs
- set<string> symbol1, symbol2(ensure no duplication when inserting into PairOnePrices and PairTwoPrices later)

```
"CREATE TABLE IF NOT EXISTS StockPairs(" \
 "id INT NOT NULL, "
 "symbol1 CHAR(20) NOT NULL," \
 ....
```

B - Retrieve and populate historical data for each stock

- TradeData: contain prices, getter
- Stock: contain symbol and a vector of TradeData, add TradeData and getter

- Build a stock map, map<string, Stock>
- Get data in buffer, build a TradeData, add into stock using stock.addTrade()
- For symbol in set symbol1, symbol2, get stock from stock map
- Insert into PairOnePrices, PairTwoPrices

"INSERT INTO PairTwoPrices(symbol, date, open, high, low, close, adjusted_close, volume) VALUES(\"%s\", \"%s\", %f, %f, %f, %f, %f, %d)"

C - Create PairPrices table

- Use SQL to create and insert data from PairOnePrices, PairTwoPrices

```
"PRIMARY KEY(symbol1, symbol2, date),"
                    "FOREIGN KEY(symbol1, date) REFERENCES PairOnePrices(symbol, date) ON DELETE CASCADE ON UPDATE CASCADE,"
                    "FOREIGN KEY(symbol2, date) REFERENCES PairTwoPrices(symbol, date) ON DELETE CASCADE ON UPDATE CASCADE,"
                    "FOREIGN KEY(symbol1, symbol2) REFERENCES StockPairs(symbol1, symbol2) ON DELETE CASCADE ON UPDATE CASCADE);";
```

D - Calculate Volatility

- PairPrice: contain prices and pnl for a day
- StockPairPrices: contain pair, a map of date, pariPrice, backtest setting, getter and setter

- Use sql to update variance for each pair in StockPairs
- StockPairPrices.SetVolatility()

- vector<StockPairPrices> AllPairs
- Function one:to get AllPairs from table StockPairs
- Function two: GetVolFromDatabase vector<double>& vols (sqrt)

- Use vols to StockPairPrices.SetVolatility()

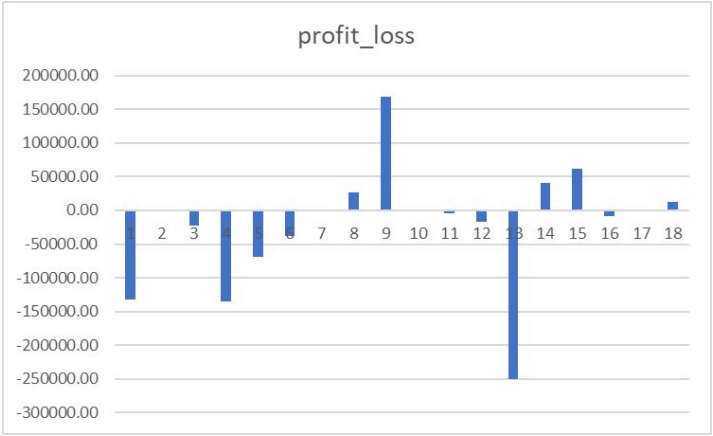
E - Back Test (From 2022-01-01 to 2022-03-11)

- **int CalculateBackTest(sqlite3* db, vector <StockPairPrices>& AllPairs,double Kvalue) function in Calculation.cpp**
  - Get symbol1, symbol2, kvalue and volatility of each pair by traversing vector <StockPairPrices> AllPairs;
  - Use SQL statement to retrieve date, open, close data for each pair from PairPrices table:
    - SELECT symbol1, symbol2, date, open1, close1, open2, close2 FROM PairPrices ") + "WHERE symbol1 = \'" + Stock1 + "\' AND symbol2 = \'" + Stock2 + "\' AND date >= \'" + BackTestStartDate + "\';";
  - Define variable LongShort:
    - if (abs(Close1d1 / Close2d1 - Open1d2 / Open2d2) > (VolPair * kPair)) LongShort = -1; else LongShort = 1;
  - Calculate N2:
    - N2 = N1 * (Open1d2 / Open2d2);
  - Calculate P/L:
    - ProfitLoss = (-LongShort * N1 * (Open1d2 - Close1d2)) + (LongShort * N2 * (Open2d2 - Close2d2));
  - Use SQL statement to update P/L for each pair in PairPrices table:
    - "Update PairPrices SET profit_loss = %f WHERE date = \"%s\" AND symbol1 = \"%s\" AND symbol2 = \"%s\";", ProfitLoss, results[(rowCtr * columns) + 2], results[(rowCtr * columns) + 0], results[(rowCtr * columns) + 1])

F-Calculate PNL for Each Pair

- Calculate total profit and loss
  - For each pair of stocks in vector AllPairs, use GetStockPairs() function in StockPairPrices class to get symbols of the pair
  - For each pair, use GetDailyPrices() function to get map dailypairprices, in which the index is date and value is PairPrice structure.
  - Get daily PNL from the PairPrice structure for each date and sum them up to get total PNL for each pair.
- Update SQL table

  Then we update the profit and loss in SQL StockPairs table according to two symbols of the pair



G- Manual test

- Retrieve available stock pairs from database and print it on the screen
- Enter the id of stock pairs and retrieve the volatility of that pair from database, by calling the function “GetVolFromDatabase”
- Enter k, and the open prices for day 1 and day 2 and close prices for day 1
- Get N2 and P&L for that trade

Conclusion

- The pair trading strategy using the ratio of two prices cannot guarantee to get a positive profit. In other words, the ratio of two prices is not a good indicator for pair trading
- This strategy does not consider the mean of the price ratio. A $k\sigma$ interval around mean may be a good improvement

Enhancement

By default, SQLite will evaluate every INSERT statement within a unique transaction. If performing many inserts, it's better to wrap your operation in a transaction

Insert for two symbols: 27.68seconds to 0.54 seconds

sqlite3_exec(db, "BEGIN TRANSACTION", NULL, NULL, &sErrMsg);

Many inserts...

sqlite3_exec(db, "END TRANSACTION", NULL, NULL, &sErrMsg);

References

- *High-Frequency Trading: A Practical Guide to Algorithmic Strategies & Trading Systems, 2nd Ed, 2013, Irene Aldridge, ISBN: 1-118-34350-6*
- *Improve INSERT-per-second performance of SQLite* https://stackoverflow.com/questions/1711631/
- *NYU Polytechnic School, FRE7831, Topics in Financial and Risk Engineering, Lecture Notes, Spring 2022*
- *Pairs Trading, Quantitative Methods and Analysis, Ganapathy Vidyamurthy, Wiley, 2004, ISBN 0-471-46067-2*
- *Pairs Trading: A Bayesian Example, Stefan Hools & J.Richard Hollos, Abrazol Publishing, 2012*

Thank you!

NYU