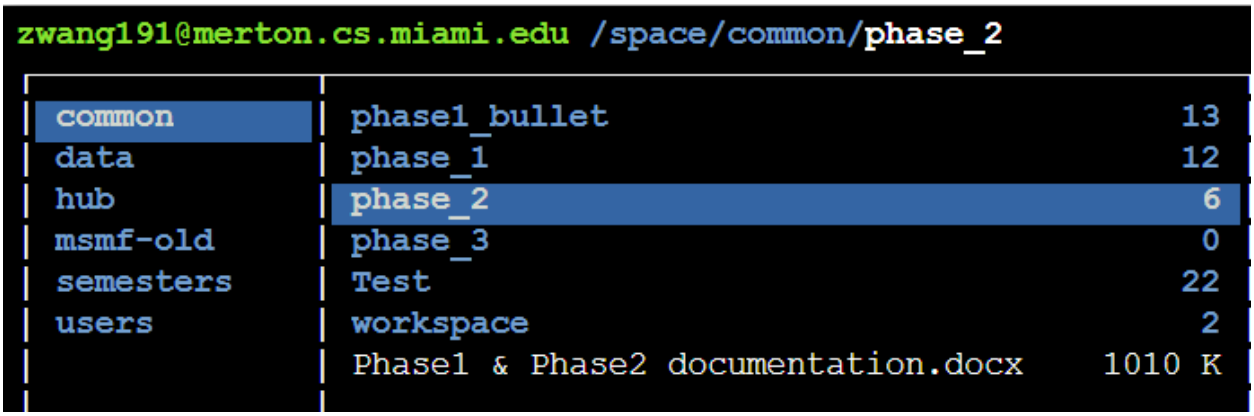
About the files

Team members：

Zirui Wang, Yiming Zhang, Kangjun Cheng, Junyu Wang.

We develop 2 sets of programs, one of which is normal input (phase_1) and the other is input by bullet function (phase1_bullet).



The programs are in :

phase_1, phase_2, phase_3 and phase1_bullet.

The Test file is used for testing programs.

The /space/common/workspace/ is the output path.

Phase 1 Program Documentation

Location of programs

/space/common/phase_1/

Notes

All programs are designed to keep asking for input until a valid one is inputted. To end the program at anytime, you can press CTRL+C to raise KeyboardInterrupt and stop the program.

This documentation is written for the merton server. Different initial work is required if ICE is running this.

Running the Programs

Before running the program please make sure you have downloaded all the required packages in your environment.

To run a python script called myprogram.py, type the following:

python myprogram.py

Program Descriptions

All programs have the input constraint that you can only enter one input, not a list of inputs. For example, if the program asks the user to input a ticker, you cannot enter

X:SXBTUSD, X:SBCHEUR

Program 2

Filename: problem2.py

Description:

Plot the trade data across all venues for one pair and one date. If a pair has no trade data, then program will notify the user that there is no trade data, and no output file will be produced

Inputs:

filetype: A (1619 Spot data 1) or B (1356 Spot data 2) or C(666 Future data 1) or D(680 Future data 2)

date: the date in yyyymmdd format

ticker: the pair to plot

option: plotly (for an html interactive plot) or matplotlib (png image)

Output: An html file (if plotly) or png file (if matplotlib) in workspace/phase1_output/program2_out

Program 3

Filename: problem 3.py

Description:

Plot the quote data (both bid and ask) across all venues for one pair and one date. If a pair has no quote data, then program will notify the user that there is no quote data, and no output file will be produced.

Inputs:

filetype: A (1619 Spot data 1) or B (1356 Spot data 2) or C(666 Future data 1) or D(680 Future data 2)

date: the date in yyyymmdd format

ticker: the pair to plot

option: plotly (for an html interactive plot) or matplotlib (png image)

Output:

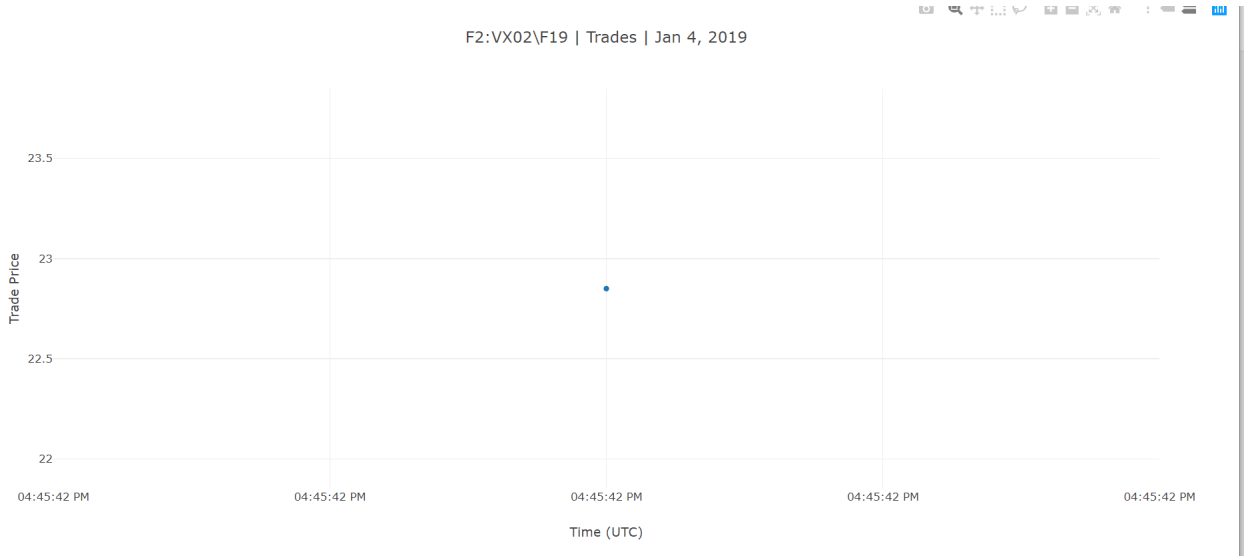
An html file (if plotly) or png file (if matplotlib) in workspace/phase1_output/program3_out

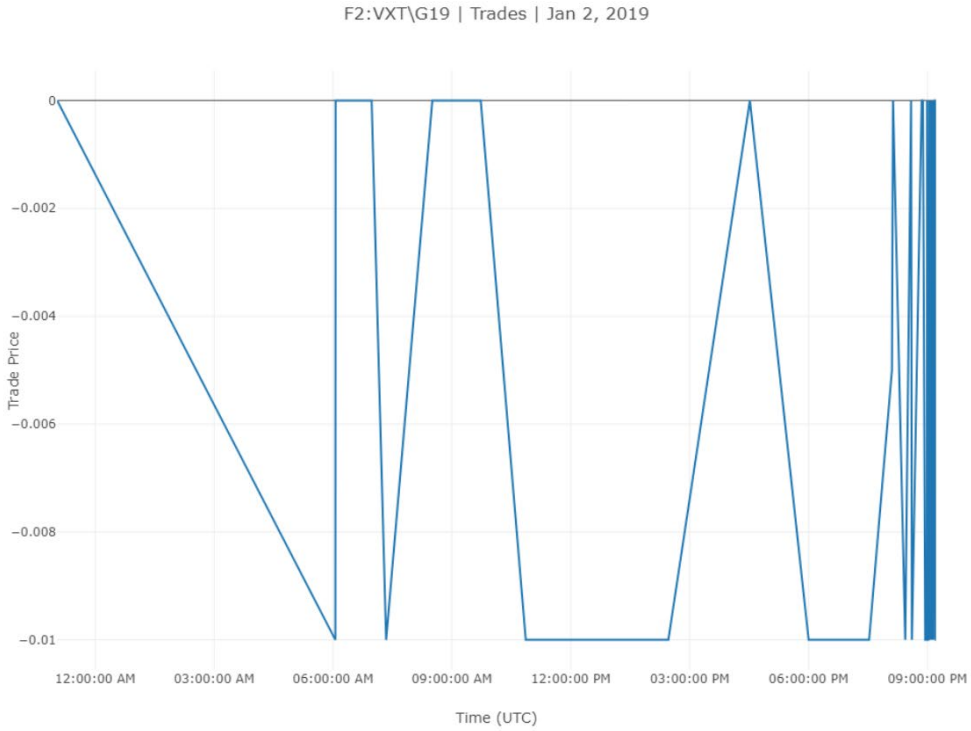
Comment：

The sample plots show in file A and B there is nothing unusual in the output of pro2 and pro3, just like the plots in the market.

Most of the future data don’t have enough trade or quote data, which leads to the plots be flat or huge fluctuation.

Samples:


F2:VX02\F19 | Trades | Jan 4, 2019
23.5
23
22.5
22
Trade Price
04:45:42 PM
04:45:42 PM
04:45:42 PM
04:45:42 PM
04:45:42 PM
Time (UTC)


F2:VXT\G19 | Trades | Jan 2, 2019
0
−0.002
−0.004
−0.006
−0.008
−0.01
Trade Price
12:00:00 AM
03:00:00 AM
06:00:00 AM
09:00:00 AM
12:00:00 PM
03:00:00 PM
06:00:00 PM
09:00:00 PM
Time (UTC)

Program 4

Filename: problem 4.py

Description:

Create a csv file that contains the number of trades and quotes and trade volume for all pairs in that date. For each pair, the csv provides both the total number quote and trade orders and total traded volume of that pair at the exchange level, region level, and combined level.

Inputs:

filetype: A or B or C or D

date: the date in yyyymmdd format

Output:

A csv file that takes pairs as its row names and venues as its column names.

Comment：

In this part the sample outputs are csv files, nothing particular. This file makes it easier for us to check the missing values.

Program 5a

Filename: problem 5a.py

Description:

Given a quote type (bid or ask), plot that quote type for each exchange for one pair and one date. If a pair has no quote data, then program will notify the user that there is no quote data, and no output file will be produced.

Inputs:

filetype: A or B or C or D

date: the date in yyyymmdd format

ticker: the pair to plot

quotetype: Bid or Ask

Output:

An html file (if plotly) or png file (if matplotlib) in workspace/phase1_output/program5a_out

Comment：

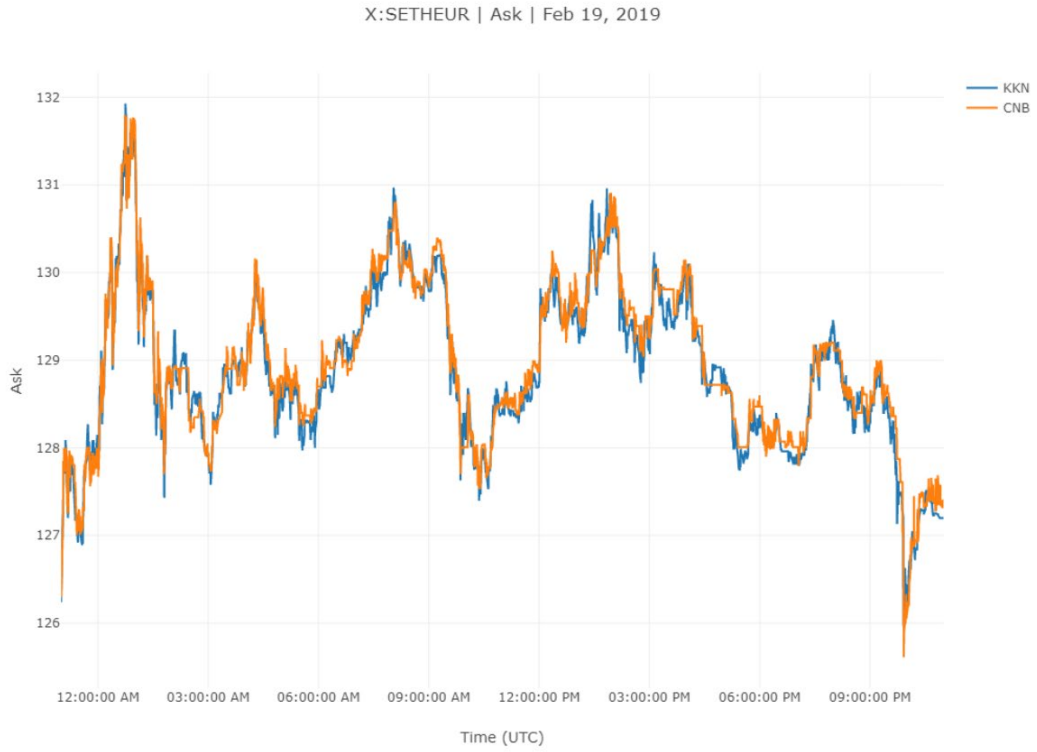
Some of the data files' venue name are missing, for these files all the missing venue are grouped into a single venue "un".

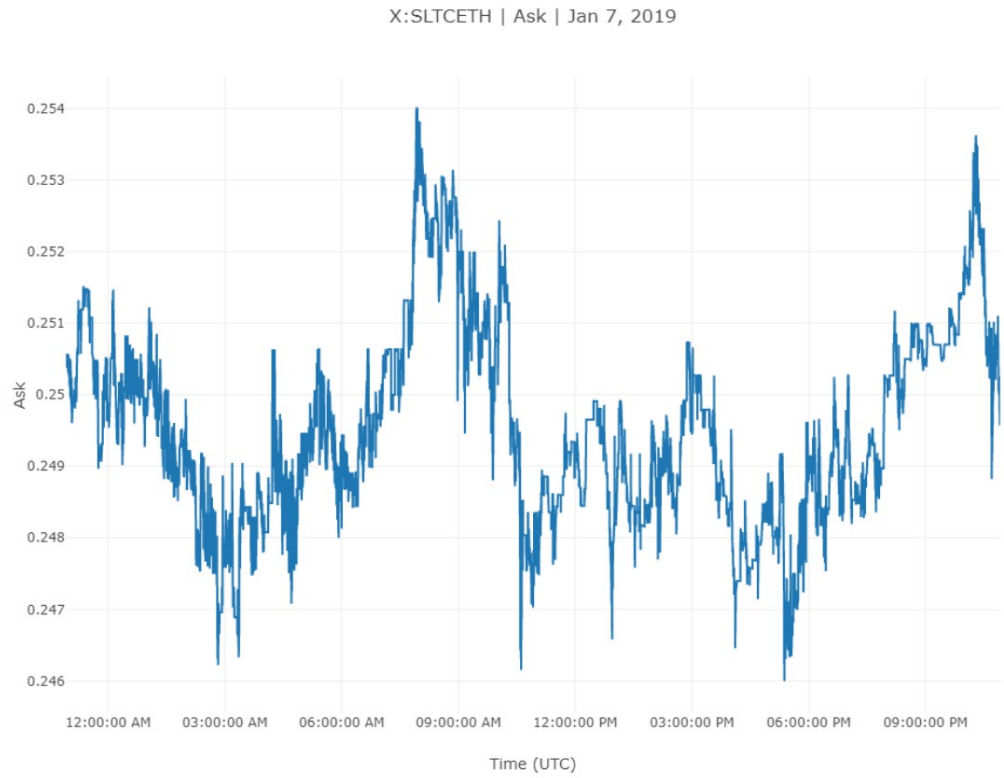
Most of A files have venue-wise data and the venues are "KKN" and "CNB", the data from these two venues have less spread.

As for B files, the missing venue are the majority. The plots are mostly not different from the aggregate plots.

When it comes to C and D files things get worse. Most of them have few trade or quote data, which leads to the empty plots or plots with few points.

Samples:


X:SETHEUR | Ask | Feb 19, 2019
KKN
CNB
132
131
130
129
128
127
126
Ask
12:00:00 AM
03:00:00 AM
06:00:00 AM
09:00:00 AM
12:00:00 PM
03:00:00 PM
06:00:00 PM
09:00:00 PM
Time (UTC)


X:SLTCETH | Ask | Jan 7, 2019
0.254
0.253
0.252
0.251
0.25
0.249
0.248
0.247
0.246
Ask
12:00:00 AM
03:00:00 AM
06:00:00 AM
09:00:00 AM
12:00:00 PM
03:00:00 PM
06:00:00 PM
09:00:00 PM
Time (UTC)

Program 5b

filename: problem 5b.py

Description:

Plot the trade data for each exchange for one pair and one date. If a pair has no trade data, then program will notify the user that there is no quote data, and no output file will be produced.

Inputs:

filetype: A or B or C or D

date: the date in yyyymmdd format

ticker: the pair to plot

Output:

An html file (if plotly) or png file (if matplotlib) in output/program5b_out

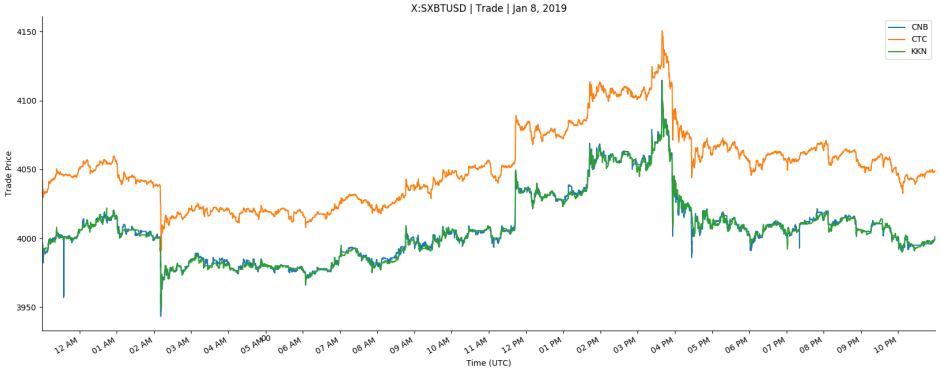
Comment：

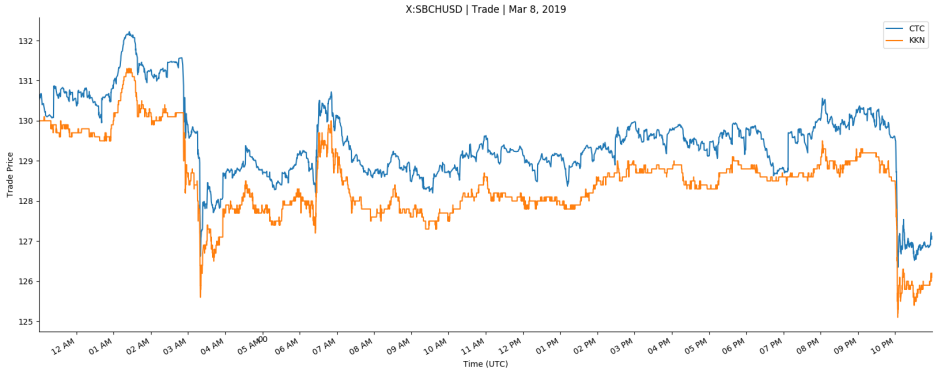
Based on the problem5b, we can find that for some specific ticker, the price spread between different venues could be utilized. Take ‘A | X:SXBTUSD’ & ‘A | X_SBCHUSD’ as example.

It’s easy to see that in the period of low volatility, the spread is going to be wide; and during the period of high volatility, the spread is going to be narrow. Since one possible trade strategy

is to focus on the volatility of the price, practicing arbitrage with spread change.

Samples:





Program 6

Filename: problem 6.py

Filename: problem 6-Quote.py

Description:

Given the file types of Crypto Future and Crypto Spot, select an available date and the program will ask for a ticker symbol and plot that spread. If the selected files have no available data, then program will notify the user that there is no ticker data, and no output file will be produced.

Inputs:

Filetype1: A or B

Filetype2: C or D

date: the date in yyyymmdd format

ticker: the ticker symbol to plot

option: plotly (for an html interactive plot) or matplotlib (png image)

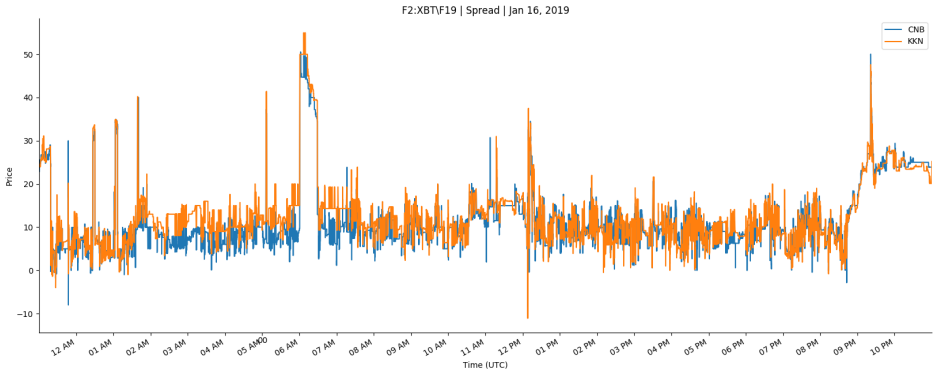
Output:

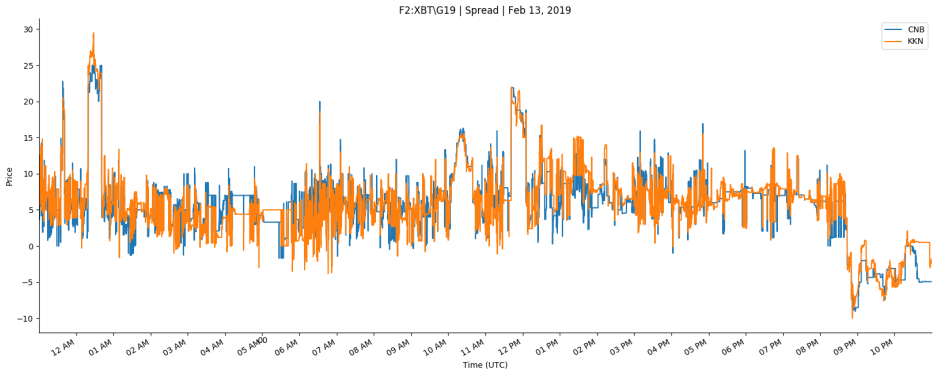
An html file (if plotly) or png file (if matplotlib) in workspace/phase1_output/program6_out

Based on the problem6 quotes, we can find that spread (Future bid price – spot bid price(X:SXBTUSD)) fulfill the following observation:

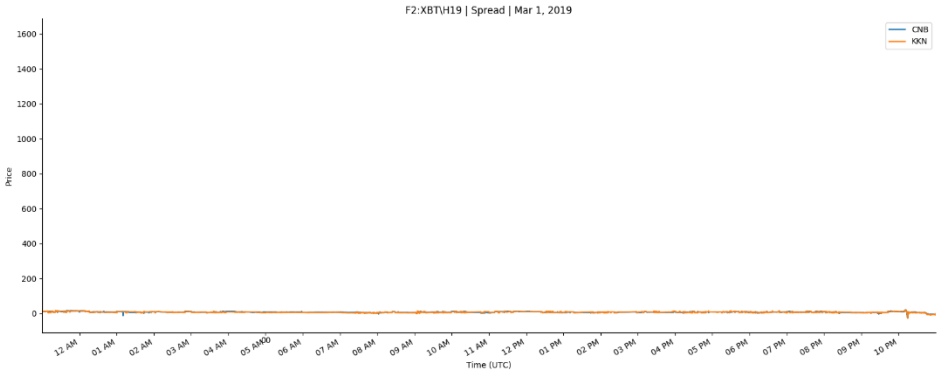
1.Every beginning of the week the spread will be very fluctuation either future price will over the spot price or on the opposite side and then it will become more staple in the median period and after that will keep on fluctuating.

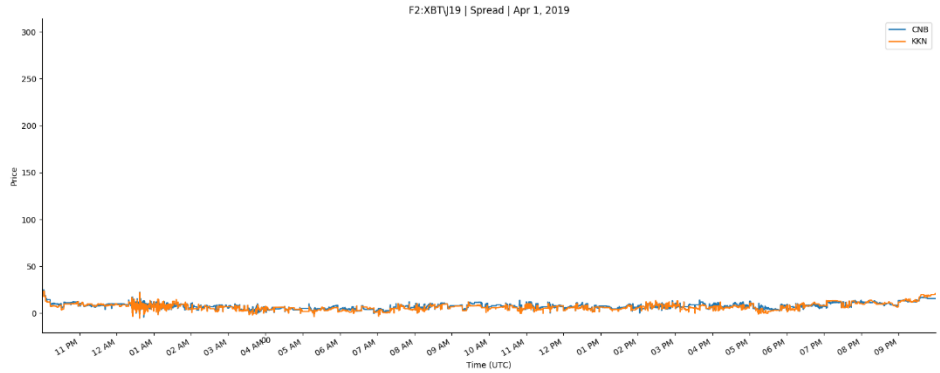
2.The expiration date of the futures are different for different months or tickers, like F19 will expirate on Jan 16th, G19 will expirate on Feb 13th, which I conclude from the plot that at these days the spread will be extreme fluctuations.



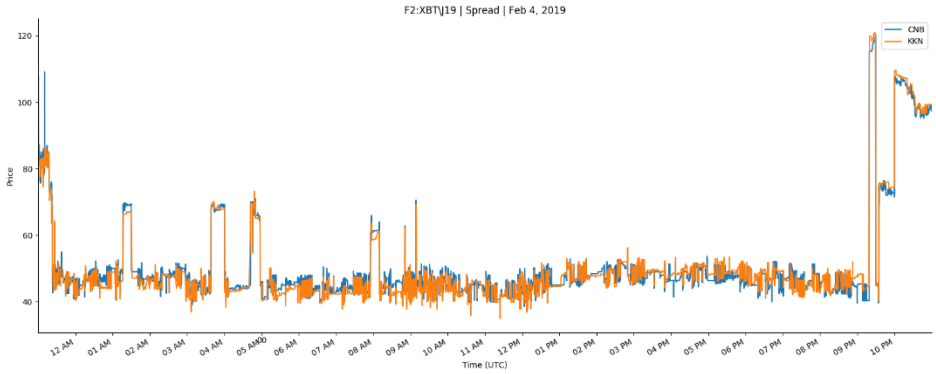


3.And comparing to the tickers that will expirate soon the tickers have more time to expire will have more fluctuations in the beginning and become less fluctuations when near the expiration and the ticker at the very beginning of the month the fluctuations are really small.





4.from the plots I made, I find that the spread for bitcoin can be very unstable which means the average price for spot and future are near 3000 to 4000, but the spread the difference can be over 1000 many times during a period or a single day which stands for investing opportunity and also risk.



5.From the plot of two different venue, the CNB and KKN ticker spread nearly keep moving at the same trend and have little difference.

Program 7

Filename: problem 7.py

Description:

Given the file types of Crypto Future and Crypto Spot, no need to select a date. And the program will ask for a ticker symbol and plot that spread. If the selected files have no available data, then program will notify the user that there is no ticker data, and no output file will be produced.

Inputs:

Filetype1: A or B

Filetype2: C or D

ticker: the ticker symbol to plot

option: plotly (for an html interactive plot) or matplotlib (png image)

Output:

An html file (if plotly) or png file (if matplotlib) in workspace/phase1_output/program7_out

Comment：

For the plots of C and D files, we have the same issue as the program 2 and 3 due to the lack of trade and quote future data.

Because of the lack of trade data in C and D, a large amount of the plot in program 6 and 7 cannot be shown.

Advanced Usage:

The programs are designed to take in command line inputs, so that you can input the parameters in the same line as the python command, rather than inputting the parameters after you start the program. For example, you can type:

python program2.py A 20180420 X:SXBTUSD plotly

rather than type python program2.py, then input those parameters in by hand. If you have the parameters saved in a file called input.txt, then you can type:

python program2.py $(cat input.txt)

Note:

The programs assume correct input if you use command line arguments. For example, if you run program2.py, if you enter 4 parameters in the wrong order, the program will probably crash. If you enter fewer than 4 arguments, the program will run as if you entered no arguments. If you enter more than 5 arguments, any arguments after the 4th argument will be ignored.

The bullet function programs are easier to get input from testers. The logic of the two sets of programs is same. Pro6 and pro7 are not available for bullet function.

Phase 2 Program Documentation

Location of program

/space/common/phase2_programs

Notes

All programs are designed to keep asking for input until a valid one is inputted. To end the program at anytime, you can press CTRL+C to raise KeyboardInterrupt and stop the program.

Running the Program

To run a python script called myprogram.py, type the following:

python myprogram.py

Program Description

All output will go to the common workspace folder, i.e. /space/common/workspace

Program 8

File name: problem8.py

Description:

Develop a program to plot volume and price trends together to check the correlation between the two, make plots to show the difference between quotes' bid volume and quotes' ask volume and the trade price .

Inputs:

Filetype: B (1356 Spot data 2)

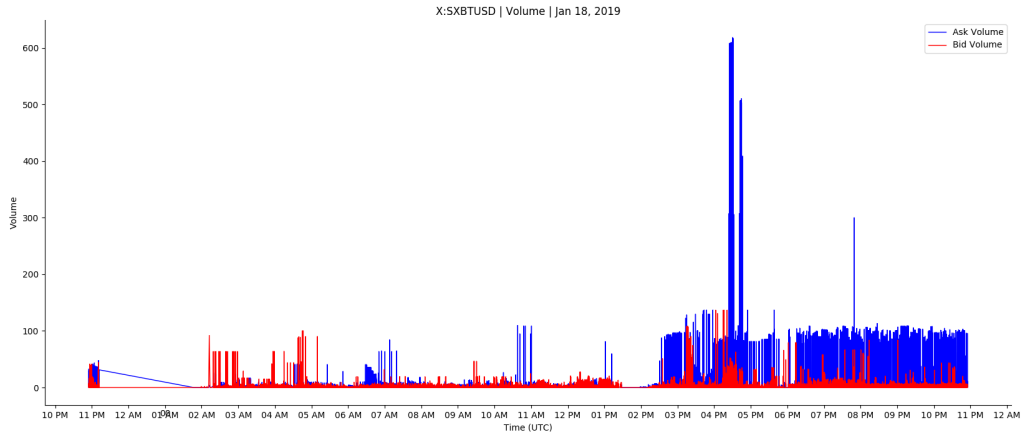
date: the date in yyyymmdd format

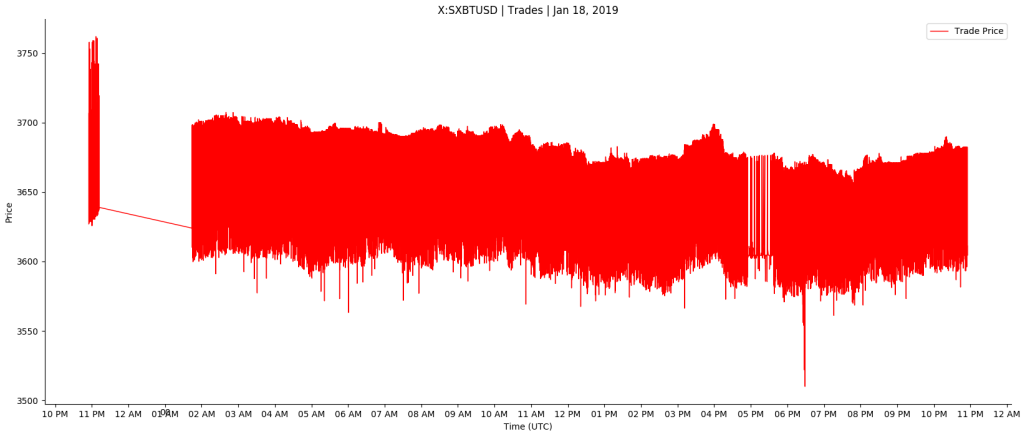
ticker: the pair to plot

plot: plotly or matplotlib

Output: For the selected database B, filetype and the date, the output is a png file or a html file with ticker name, files and date in the name.

Sample：


X:SXBTUSD | Volume | Jan 18, 2019
Ask Volume
Bid Volume
Volume
600
500
400
300
200
100
0
10 PM
11 PM
12 AM
01 AM
02 AM
03 AM
04 AM
05 AM
06 AM
07 AM
08 AM
09 AM
10 AM
11 AM
12 PM
01 PM
02 PM
03 PM
04 PM
05 PM
06 PM
07 PM
08 PM
09 PM
10 PM
11 PM
12 AM
Time (UTC)


X:SXBTUSD | Trades | Jan 18, 2019
Trade Price
Price
3750
3700
3650
3600
3550
3500
10 PM
11 PM
12 AM
01 AM
02 AM
03 AM
04 AM
05 AM
06 AM
07 AM
08 AM
09 AM
10 AM
11 AM
12 PM
01 PM
02 PM
03 PM
04 PM
05 PM
06 PM
07 PM
08 PM
09 PM
10 PM
11 PM
12 AM
Time (UTC)

Program 9

Filename: problem 9.py

Description:

This program gets the statistics (num outliers, percent outliers, etc), outliers and its “strength” as a csv, and a plot of where are the outliers.

Notes about strength:

k-gamma: the larger the number, the more likely it is an outlier

Inputs:

filetype: A or B or C or D

date: the date in yyyymmdd format

ticker: the pair to plot

option1: Trade, Bid, or Ask

Venue: the venue (missing venue as unknown)

Parameters:

k (int): window size

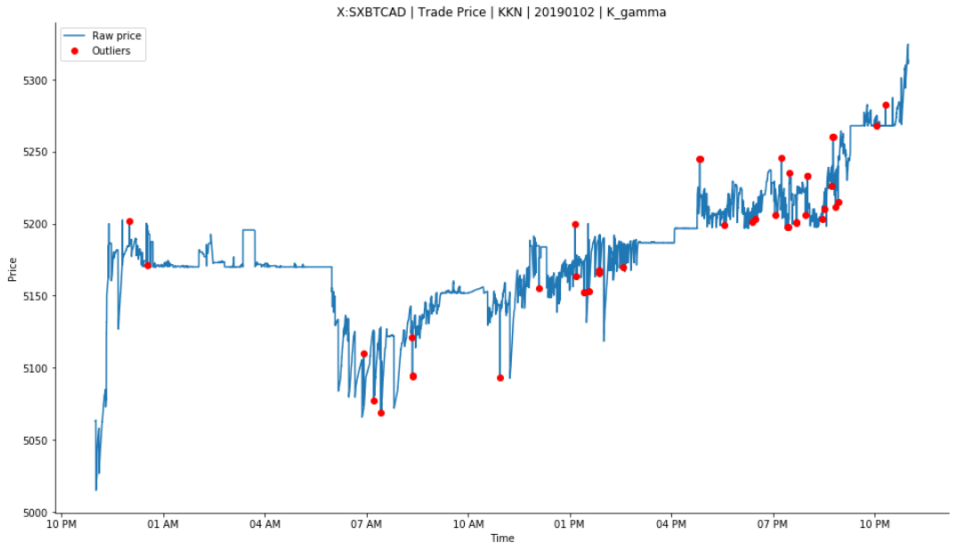
gamma (float): granularity parameter

Output:

For all outlier detection methods. A .txt file with the statistics of the method and the outliers. A .csv file with all of the detected outliers and their “strength”. A .png file with the location of outliers.

Sample：

```
Method: K_gamma
Parameters: 20_1
Number of outliers: 42
Total number of observations: 16458
Percentage outliers: 0.0025519504192489974
```



Program 10

Filename: problem_10.py

Description:

Make plots in such a way that the difference between quotes' bid volume and quotes' ask volume and the trade price are shown on the same plot.

There're two sets of trade prices data: raw and cleaned data (remove the outliers venue-wise).

Plot the price series for all venues/regions in same graph using different colors.

Along with the program, gather the outliers data and data description in a .CSV file.

The data description includes the following information:

1. Venue
2. Parameters: value of k and gamma
3. Number of outliers
4. Total number of observations
5. Percentage outliers
6. Cleaned price range
7. Raw price range

Inputs:

filetype: A or B or C or D

ticker: the pair to plot spread

Parameters:

k (int): window size

gamma (float): granularity parameter

Output:

For the selected filetype and the date, the output is a png file with ticker name, filetype and date in the name. All available venues are plotted together on one plot.

Other Program

Filename: problem_other.py

Description:

This program makes use of the outliers get from program 9 and show the plot of the filtered spread.

Notes about strength (same as the parameters in program 9):

For k-gamma: the larger the number, the more likely it is an outlier

Inputs:

Filetype1: A (1619 Spot data 1) or B (1356 Spot data 2)
Filetype2: C (666 Future data 1) or D(680 Future data 2)

date: the date in yyyymmdd format

ticker: the pair to plot spread

Parameters: depending on the method chosen

K-gamma

k (int): window size

gamma (float): granularity parameter

Output:

For the selected pair of the filetype and the date, the output is a png file with ticker name, files and date in the name. Each available venue is plotted separately on one plot.

Sample：