Advanced Risk Management, William Cheung, Assignment 1, June 14, 2021
Due date: June 28 23:59, 2021

Arthur Bonetti

Disclaimer: This work was done using Python 3.0 and the pandas and matplotlib libraries. Because of the size of the data, I decided to drop some data and estimate the daily realized spread and the Amihud price impact with only the transaction with volume 1. I used the function iterrows() which is very inefficient and therefore the program takes a long time to do the computations and give the wanted results. This work can be optimized in term of computation time. I shared the codes with some fellow students that lacked the technical skills to implement the homework using a programming language. Nonetheless, they still reviewed the code and understood the concepts behind it. The code can be run in any python environment, I personally use Pycharm.

1. **Compute daily realized spread**

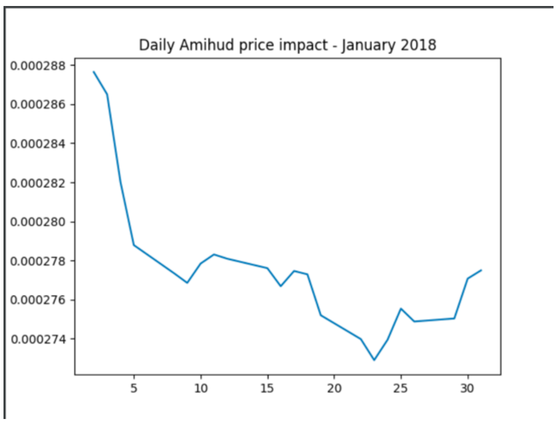
   The code to compute the realized spread can be reviewed in the ARM_HW1_drs.py file. First, we import the data. Then, we clean the data a bit. Because we want the daily realized spread, we check if the date is the same as the previous transaction and if it's the case, we add the realized spread to a variable. We put a condition to only keep track of the market orders. To obtain the price 5 minutes after each transaction, we use the datetime column, add 300 seconds and look at the closest transaction. We compute the realized spread for every transaction and add them together. When we arrive at the end of the day, we divide the total number of each transaction by the number of transactions to obtain the daily realized spread. We repeat the process for each day and plot the data in a graph. The results and the final graph can be found in the ARM_HW1_plot_drs.py file.

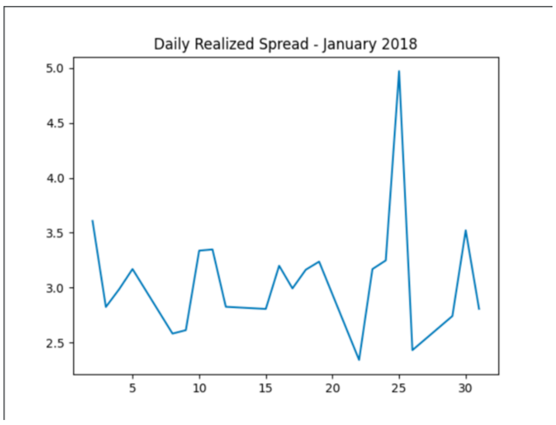
2. **Compute daily Amihud price impact by constructing minute-by-minute return and dollar volume.**

   The code to compute the Amihud price impact can be reviewed in the ARM_HW1_ap.py file. The code is quite similar to the daily realized spread in term of data management. The difference is in the formulas used. This time because we want the minute-by-minute return, we look for the first transaction in a +60 datetime horizon. We also look at all the transactions and not only the market orders. Then, we compute the Amihud price for each transaction and when we arrive at a new day, we divide by the total number of transactions and add it to a list to keep track of each daily Amihud price. The results and the final graph can be found in the ARM_HW1_plot_ap.py file.

3. **Plot your daily measures over time. Has liquidity improved or deteriorated over time in the sample? Discuss you results.**

The plots can be reproduced using the data and the code in the ARM_HW1_plot_drs.py and ARM_HW1_plot_ap.py files.





When we look at both graphs, the liquidity seems to improve over time in the sample. Indeed, the daily Amihud price clearly has a descending trend from 2nd January until the 24th of January. The daily realized spread variates a lot more, nonetheless from the 2nd of January to the 24th of January the daily realized spread has also decreased. We can see that for the 25th of January, the illiquidity increased, this can be seen in bot graph with an uptick for this date. The uptick is more pronounced in the daily realized spread. From the 24th of January and onward, the illiquidity seems to jump back at previous level. To summarize, I would say that the liquidity has improved from the beginning of the month until the 24th. Then is has deteriorated from the 24th until the end of the month.

Bonus. Suggest how one could compute daily *effective* spread from the current data set. Show your results if you could please.

To compute the daily effective spread from the current data set, we need the traded price and the midpoint. In our data set we can easily get the traded price by looking at the adequate column (PRICE). Nonetheless, we still must estimate the midpoint. To do so, I suggest looking only at market orders. Then, if the transaction is a buy, we look at the next sale value and compute the midprice: (ask+bid)/2. If it's a sale, we look at the next buy value and compute the midprice in the same way. Then we compute the effective spread with the following formula: 2*(traded price – midprice). We add the values for each transaction together and when we arrive at the end of the day, we divide the total value by the total number of transactions to find the daily effective spread. The code for the daily effective spread can be found in the ARM_HW1_des.py file. The code is working, nonetheless, it is so inefficient in term of computation time that I decided to give up on waiting and did not computed all the daily values nor plotted the data.