NYU Courant MATH-GA 2708

Assignment 1 Written Report

Yuanzhe Yang yy3460

Chien-Yueh Shih cs6380

Zedi Qiu zq2015

Introduction

This project mainly focused on processing and analyzing tick-size NYSE TAQ data during 2007-06-20 and 2007-09-20.

The functionalities include:

1) TAQAdjust: filter out only S&P 500 listed tickers and adjust the stock price and size for stock splitting/ stock buyback.
2) TAQClearner: clean the tick data according to historical rolling window to remove outlier
3) TAQStats: computing different statistical measures on both cleaned and uncleaned data and compare
4) TAQAutoCorr_analysis: test the Autocorrelation between stock returns to compute minimum return resample frequency that will hardly be affected by bid-ask bounce
5) TAQPlot: visualize the statistical results for previous functionalities. Result visualization by matplotlib for all previous functionalities
6) PortfolioOptimization: Portfolio optimization through CVXOPT using historical data during 20070620 - 20070920, and calculate portfolio turnover rate

In order to successfully deploy our code on your local device, we strongly suggest you follow following instruction:

1) If you are a mac user, you can use the script file named Unzip.py in our code to help you unzip all the zipped date files for both trades and quotes. If you are a Windows user, you can alternatively do it by unzip all the files in a batch using file unzipper.
2) Be careful about the directory of data files used in our code. You should have both trades and quotes folders sitting inside a main folder named data, and data should be set under your working directory.
3) In order to make it easier for user/grader to test our code, we use relative path in all places of our code whenever we use directory path. Namely, you just need to have the correct structure of storing data, you should be fine
4) The basic workflow of our code is: TAQ_Filter → TAQ_Adjust → TAQ_Cleaner → TAQ_Stat → TAQ_AutoCorrelation → TAQ_Plot. Note that you do not need to run all the script to see the result. We provide a main.py which is the main file for you to go through all the process and exhibit the result.
5) Since it is a very large dataset, the whole project might blow up your storage space. Prepare more than 100 GB storage space in order to run the project. If you do not have enough space, you can delete all the raw data files (E.g. data/trades/20070620 …) and only save the Daily_trades and Daily_quotes folder which include concatenated parquet.gzip file after your ran TAQ_Adjust. Since all the following steps will not depend on original data but instead on the parquet.gzip data.
6) During our coding, we found some minor abnormality exists in our dataset. JAVA and SUMW represent the exact same company, and SUMW is the previous ticker name for JAVA. Ticker CMCSA exist in. sp500 list but we do not have any trades and quotes data for it.

1. Preparing TAQ Data:

   1) Download the data

   TAQ Data which saved the in the gzip streams form were downloaded and unzipped from google drive to the local disc. The file contains all the quotes and trades data from June 20 to September 20, 2007.

Each profile. Daily Trade and Quote provides users to access all trades and quotes for all issues in NYSE. It's a comprehensive history of daily activity from NYSE markets.

2) Data Filtering

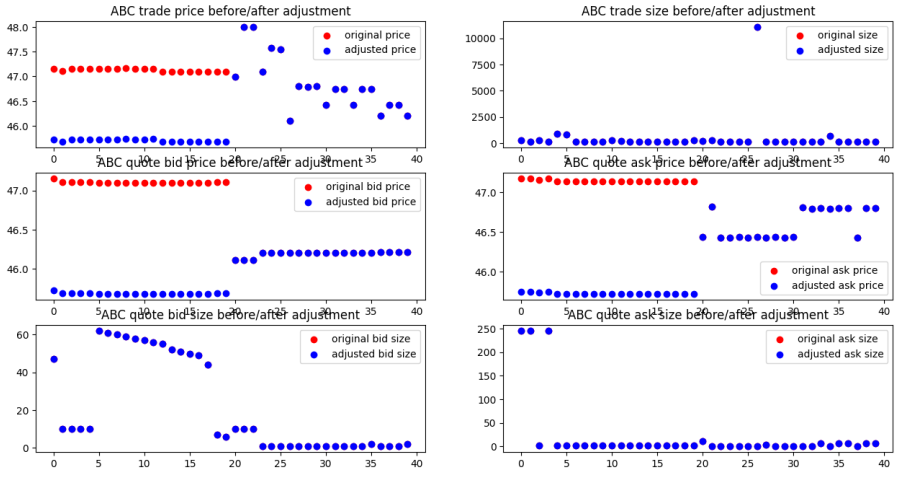
With the TAQFilter script, it will find the splitting days and extract the list for stocks lists that we would like to analyze. In this case, we filter out the S&P 500 stocks with S&P 500 tickers file. Since we know S&P 500 changes its constituent tickers all the time. It is not guaranteed that the constituent tickers will all be the same for every day in given period. We thus provide a function to help us quickly find the S&P 500 tickers name list given a distinctive date. We further utilize this script to filter out all the daily return and turnover rate of each individual ticker, so that we can exploit CVXOPT to do a portfolio optimization later in the last part.

3) Adjustment for corporate actions

The corporate actions are analyzed based on the factors including "Cumulative Factor to Adjust Prices" and "Cumulative Factor to Adjust Shares". With these factor values, we perform price and volume adjustment on trade data and bid-ask price and volume adjustment on quote data.

After the adjustment, for each individual ticker, we will store each day's ticker information in pickle file. Then we will concatenate all the information across the whole trading periods in the parquet file. As we mentioned above, if you want to save storage space, you can delete raw data after this step, meaning you do not need both raw binRQ files and pickle files for further analysis. Parquet file provides efficient data compression and encoding schemes with enhanced performance to handle complex data in bulk. Through concatenating data into one file, it will increase the efficiency of data throughput and performance. We end up having one zipped parquet file for each ticker after factor adjustment. With TAQAdjust script, our code will automatically analyze all stocks and compress the parquet file for each individual ticker.

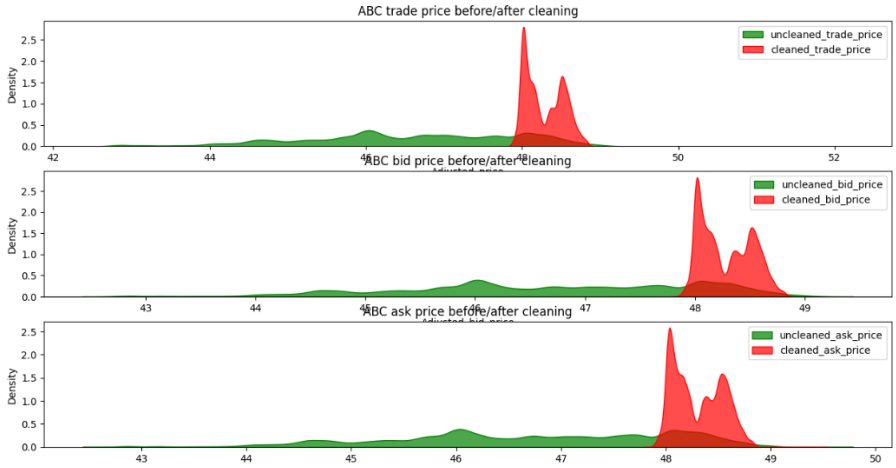
Based on the split adjusted data, we plot graphs for trade price, bid/ask price, trade size, and quote/ask size for ABC before and after adjustment. We choose 20 days before and after adjustment. Blue points correspond to the series after adjustment, while the red represent before adjustment. Ideally, the price before and after adjustment should be close and should be lower after adjustment. We do observe this in out graph below.



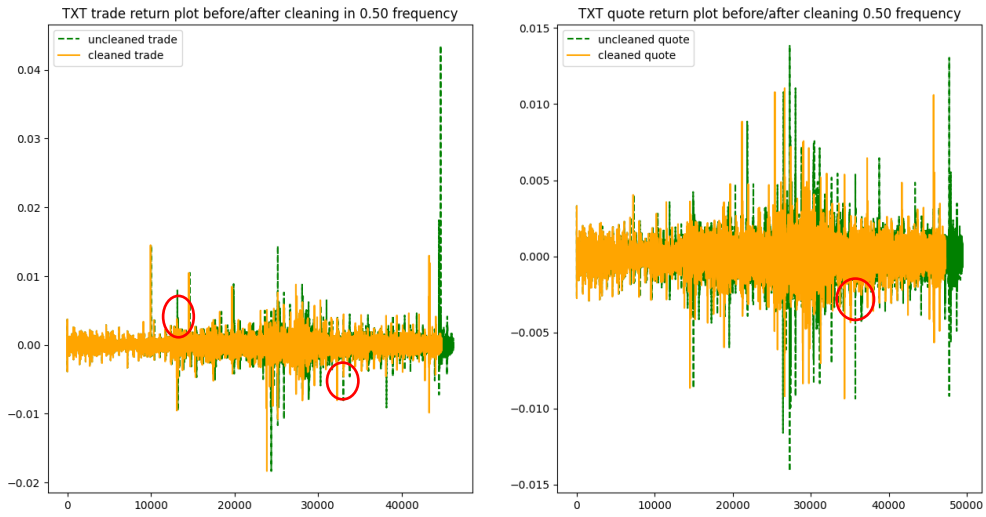
4) TAQ Data cleaning

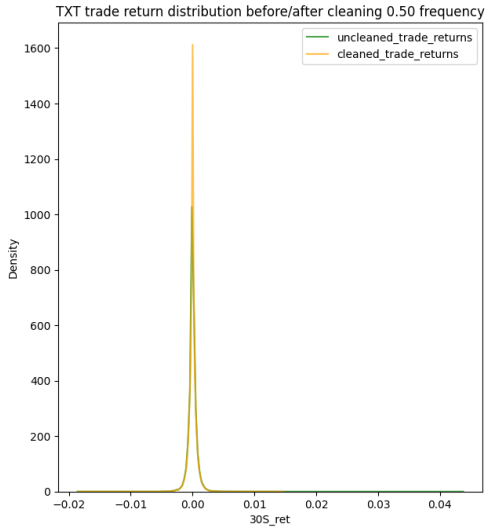
The TAQCleaner script will be used for the clean the outliers in the profile. For trade data and quote data, if values are more than two standard deviations away from K-days rolling. If the value is more than two standard deviations plus threshold error from the K-days rolling means, we will mark it as NA values. In this case, the K will be set at five days. The output from this script will be cleaned trade and quote profile for each individual ticker and save it in the parquet file.

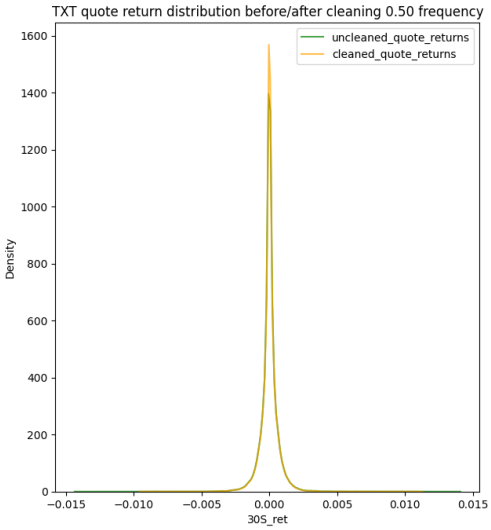
To check our results, we plot the following graphs



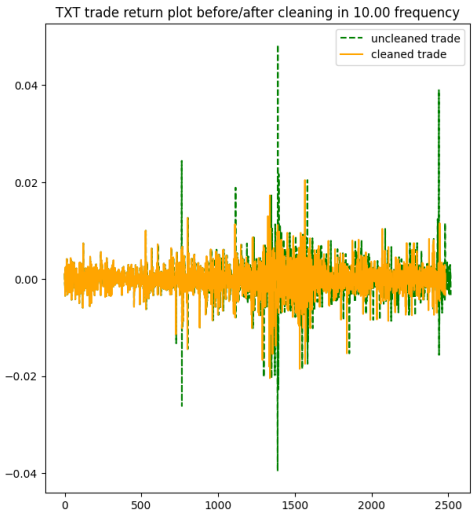
From the above graph we can tell that we have cleaned the data falling in two standard deviations away resulting the red distributions (cleaned price) are narrower than the green distributions (uncleaned price). As from returns perspective, we provide two returns with different frequencies, one in 30 second and the other in 10 min.

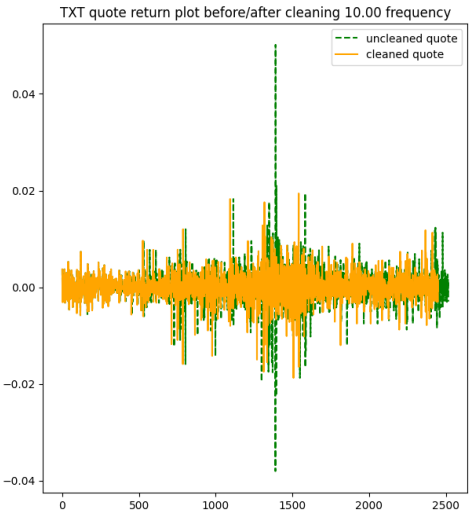


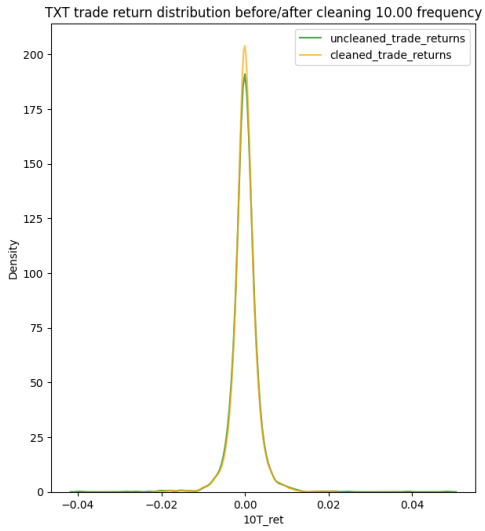


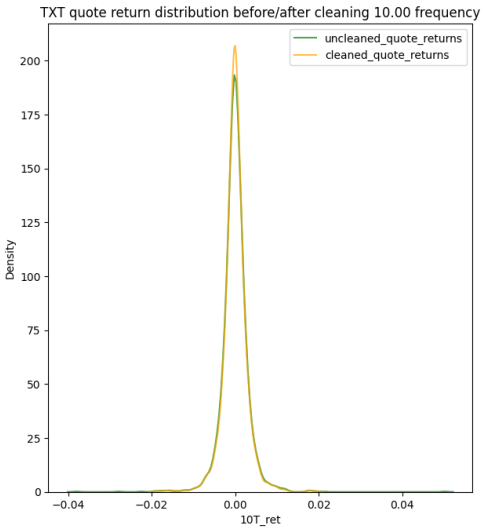


To see the impact of cleaning data in terms of return, we randomly choose TXT and plot the returns with respect to time and its return distribution. As we can see from the graphs, there are some places, marked in red circle, that the cleaning function successfully removes the outlier. Moreover, we can see that the uncleaned distribution(green line) has fatter tail and lower peak which follows our intution of the return from cleaned and uncleaned data.









Basically, the two graphs with different frequencies have the same features. However, the distribution for 10-minute moving average window has lower peak and the difference between cleaned return and uncleaned return is lower than 30-second moving average window. It is because the longer the moving average window is, the smoother the curve is which cause the difference to be smaller. That is to say, the effect of bid-ask bounce is smaller.

2. Compute Summary Statistics of TAQ Data.

1) Compute X-minutes returns of trade and mid-quotes

In the TAQStats script, we will calculate the basic statistics characteristics of the TAQ data. In the TAQUtilis function, we resample the data with 10 minutes level. After grouping the data into different time interval, we will compute the average price within each bucket. For returns in our case, we just use the simple returns.

$$(P_t - P_{t-1})/P_{t-1}$$

2) Basic Statistics

Based on the trade and mid-quote returns, we calculated the mean return, median return, standard deviation of return, median absolute deviation, skew, kurtosis, 10 largest returns, 10 smallest returns, maximum drawdown. Formula are listed below.

a) Mean Return Annualized

$$AnnualizedReturn\_Mean = (1 + Mean(return))^{NumFreq} - 1$$

b) Median Return Annualized

$$AnnualizedReturn\_Median = (1 + Median(return)^{NumFreq}) - 1$$

c) Median Absolute Deviation Return

$$AbsoluteReturn_{Median} = ABS\big(Median(return)\big) * \sqrt{NumFreq}$$

d) Skewness of Return

$$Skewness = E[(\frac{Return - Return\_Mean}{Return\_STD})^3]$$

e) Kurtosis of Return

$$Kurtosis = E[(\frac{Return - Return\_Mean}{Return\_STD})^4]$$

f) Ten largest Return

$$Top\_ten = Largest(returns, 10)$$

g) Ten Smallest Return

$$Bottom\_ten = Smallest(returns, 10)$$

h) Maximum Dropdown

$$MDD = \frac{Trough\ Value - Peak\ Value}{Peak\ Value}$$

3) Analyze the impact of cleaning data.

Following from Sec. 5 TAQ data cleaning, we also calculated some of the basic statistics. To be coherent of our analysis, we also provide the 30-second moving window and 10-minute moving window. These statistics not only prove that the previously mentioned intuition is correct, but they also show that cleaned data are close to normal distribution. As for 10-minute window, due to the reason that it is already smoothed by bigger window, the difference isn't that big.

Time frequency: 0.5 min

Statistic_measures	uncleaned	cleaned
day_length	6.500000e+01	6.500000e+01
quote_number	2.400165e+07	2.225144e+07
trade_number	1.398507e+08	1.302854e+08
fraction	5.826710e+00	5.855142e+00
mean_ret	-2.482394e-04	-3.529157e-04
median_ret	0.000000e+00	0.000000e+00
std_ret	1.198971e-02	1.069850e-02
absolute_dev	2.428461e-04	1.874549e-04
skew	4.210728e+00	-7.309164e-02
kurtosis	2.821060e+02	4.081148e+01
maximum_drawdown	2.396597e-01	1.905893e-01

Time frequency: 10 min

Statistic_measures	uncleaned	cleaned
day_length	6.500000e+01	6.500000e+01
quote_number	2.400165e+07	2.225144e+07
trade_number	1.398507e+08	1.302854e+08
fraction	5.826710e+00	5.855142e+00
mean_ret	-1.033601e-02	-1.005193e-02
median_ret	0.000000e+00	0.000000e+00
std_ret	5.564778e-02	4.578545e-02
absolute_dev	1.351487e-03	1.310660e-03
skew	5.417201e-01	-3.455797e-01
kurtosis	3.503668e+01	7.227399e+00
maximum_drawdown	2.433234e-01	1.949343e-01

3. Analysis of Autocorrelation

1) Optimal X-min window selection

TAQAutoCorr_analysis will determine the trade-off between different window length. For shorter window, it will result in return autocorrelation but for longer window, it will leave us with smaller number of datapoints.

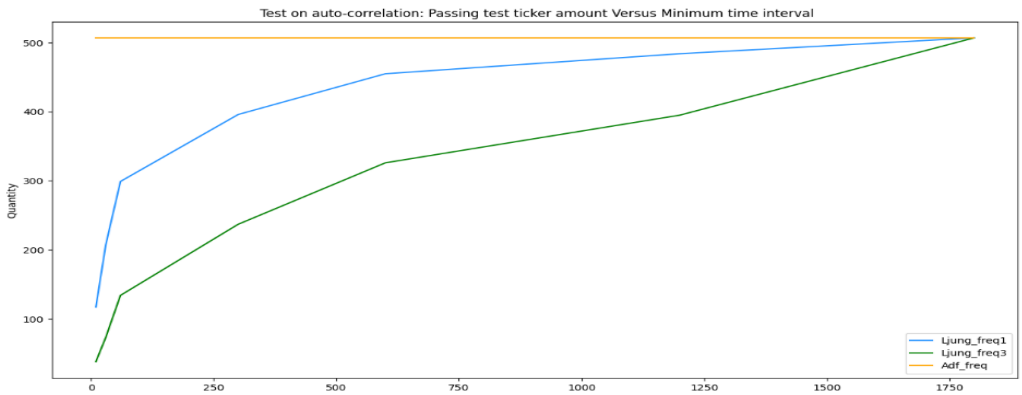
2) Ljung-Box Test

Ljung-Box Test is a type of statistical tests of whether any of a group of autocorrelations of a time series are different from zero. In our case, we try performs Ljung_Box Test on the stock return data and conduct grid search across different frequency level includes 10, 20, 60 minutes.

For the time lag in our case, we just try one-time interval lag and three-time interval lags and focus main parts on optimize X-min. By the end, we try to find the smallest X-min that there are no autocorrelation patterns.

3) Dickey-Fuller test

In statistics, the Dickey-Fuller test tests the null hypothesis that a unit root is presented in an autoregressive time series model. Below is the plot of the Dickey- Fuller test and LjungBox test with x-axis represents frequency in second and y-axis is number of tickers that pass the test. We can see that with 30-minute window every ticker has passed the test, but this information doesn't help a lot since it will cause us with less data points. So, we propose the choice of parameter to be depended on models. That is to say, we will choose 30 seconds or 10 minutes if the model requires daily settlement since these two points act as the reflection points in the graph.



5. Mean-Variance Optimization in Python

1) The CVXOPT Example

The CVXOPT solves a quadratic programming problem in the form of:

$$f(x) = q^T x + \frac{1}{2} x^T Q x$$

In this case CVXOPT find optimal x to minimize $f(x)$ with the following constraints:

$$Ax = a$$
$$Bx \leq b$$
$$x \geq 0$$

Specifically, for this problem,

$$g(x) = \frac{1}{2}\boldsymbol{\mu}(\boldsymbol{x^T \Sigma x}) - \boldsymbol{p^T x}$$

with constraints:

$$Ax = 1$$
$$Gx \leq 0$$

In this case, $\boldsymbol{\mu}$ is the risk aversion coefficients which reflects investor risk tolerance. Higher $\boldsymbol{\mu}$ means investors are strong risk averse. Σ is the corresponding covariance matrix of the S&P 500 stocks.
The optimizer has absolute tolerance level of 1e-6.

2) Optimize the market level portfolio

With the market profile represented by the S&P 500 index, we tried to compute our whole portfolio with for people with different risk aversion levels with respect to the holding constraints. For this task, we utilized the stock returns, volume, Bid-Ask average price, Shares outstanding for portfolio calculation.

For the optimization constraints, in this case, we restrict our holdings to be larger than zero which means there will be no short positions. In addition, the sum of composite of each stock is one which denotes that there is no shortage or leverage. Those constraint are very similar to what we have in our previous example, but we expand the security pool to the whole S&P 500 list.

In order to mimic the real-life situation, we need to add risk-free asset into our security pool, since we know that risk-free asset have annual return approximately 2% and it have no variance and covariance with any other securities. In order to perform portfolio optimization, we need return matrix and covariance matrix. We can easily extract that information from daily returns of S&P 500 tickers. Here we annualized all the numbers to make it consistent. As we mentioned before, we have a special condition of JAVA and SUMW. We notice that the covariance between these two stocks is infinity as it should be because they are essentially the same exact company. We solved this problem by simply discard these two tickers. The PortfolioOptimization script will use the CVXOPT package to generate the optimized holdings of each stock in our market portfolio. Note that since we should make an assumption that the holdings are only effective when we use this holding ratio as a percentage of total market value of portfolio instead of number of shares for each ticker, because different ticker have different price and return is a price-normalized measure. Thus, our holdings should also be price normalized.

Finally, we calculate the turnover ratio to reflect how much trading activity took place on a given a given

business day in the market for individual stocks.

$$Turnover\_Ratio = \frac{Total\ Traded\ Shares}{Total\ Shares\ Outstanding}$$

Our portfolio turnover ratio should be the sum of individual stock's turnover ratio weighted by our portfolio holding on corresponding stock.

In this case, from June 20, 2007, to September 20, 2007, given the risk aversion parameter to be 223.8, our optimized portfolio turnover ratio is 0.39.

```
the risk-aversion parameter is 223.872113856834
The 5-largest portion ticker in our portfolio:
Index(['BIIB', 'PLMD', 'ISRG', 'JBL', 'FLR'], dtype='object', name='Ticker')
Corresponding holding percentage:
63     0.265287
376    0.224486
243    0.161840
247    0.160006
187    0.074390
dtype: float64
Total portfolio optimized return: 1.2297405913049746
Total optimized portfolio risk: 0.04898559713162763
Total portfolio turnover is [0.39325787]
```

We also tested different parameters for risk-aversion parameters, they all yield similar tickers combinations but different holding weights. We think it might because we are choosing returns based on a relative short period of time. It is not representative enough that only good-performed stock in that special period got selected by our optimizer. In our further work, we probably want to use historical 10-year data to compute a more consistent result.

Supplement note:

We test our code on smaller data set named trades_test and quotes_test, which have fewer days than our orginal trades and quotes folders.

With Test_TAQAdjust file, we perform unit test for the adjusted stocks to make sure that the individual code compile correctly and generate the correct output.

With TAQ_Clean and TAQ_Stats, we do not find a very effective way to set up a unit test for those two files; However, through our plot and our data table before and after cleaning, we can easily validate that the cleaning match up with our expectation.

With TAQ_Autocorrelation, we again find the plot matches up with our expectation, meaning our code did not go wrong.