A Temporal Graph Neural Network for Volatility Prediction

Shu Bin *sbin6@gatech.edu*
Xinyue Ma *xinyue.ma@gatech.edu*
Zhongyun Zhang *jasonzhongyunzhang@gatech.edu*

April 30 2022

Abstract

This project seeks to construct a temporal graph neural network (TGN) model to predict realized volatility of S&P 500 component stocks during 10-minute intervals. Price and return data of 502 S&P 500 component stocks from March 15, 2022 to April 15, 2022 are extracted from Yahoo Finance. Lists of the companies' subsidiary relationships are downloaded from Wikidata to construct relationship edges between the companies. Furthermore, the stocks' correlation coefficients and their industries are also used to construct more relationship edges. The TGN model as well as other traditional models (such as ARIMA, multi linear regression, neural network, and LightBGM) are then fitted to the dataset. The models' accuracy and performance are then compared. It is found that, while the ARIMA model has a very low mean RMSE, it treats a major portion of the data as white noise, and thus does not give out any useful prediction. In contrast, the TGN model, having been able to analyze a large amount of company relationship data, has a RMSE of 0.14367, lower than all other models tested excluding ARIMA. It can be concluded that the TGN is an effective model when it comes to predicting stock return volatility as long as large amount of accurate data is available in order to construct complete relationship edge matrices.

1 Introduction

In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated to periods of market turbulence and large price swings, while low volatility describes more calm and quiet markets. As a result, accurately predicting volatility is essential for the trading of stocks and their derivatives, whose price is directly related to the volatility of the underlying product.[10] However, traditional prediction methods such as linear regression and ARIMA model often have a hard time taking into account the complexity of the relationships between multiple companies. On the other hand, graph neural network (GNN) models are relative newcomers to the finance world, but they have the ability to incorporate graph representation with regression and classification methods. This project seeks to build temporal graph neural network (TGN) models to predict short-term volatility for S&P 500 component companies' stocks. To be specific, realized volatility within ten-minute trading windows are predicted using the stocks' historical return and volatility data. Traditional methods such as ARIMA, linear regression, and LightGBM are also employed to see which model performs better.

2 Literature Review

A significant amount of studies have been done to predict volatility of the financial market.[1]

There are two major indicators commonly used to predict volatility: implied volatility and realized volatility. Specifically, realized volatility is referred to as the assessment of variation in returns for an investment product by analyzing its historical returns within a defined time period. Meanwhile, one frequently used implied volatility indicator, CBOE VIX, is based on the 30-day expected volatility resulted from the bid and ask prices of options whose expiry period lie between 23 and 37 days.[2] Both types of volatility can be used to conduct hedging or arbitrage and to obtain profit in the market.

Moreover, some previous studies on volatility predictions are based on single assets while other studies are based on multiple assets. Some studies also predict volatilities for different asset classes, such as stocks, commodities, ETF Indices, etc.[3]

Meanwhile, while econometrics models such as ARIMA and GARCH have been used to predict volatility, new deep learning models have been used stock prices and volatility as well, including MLP, CNN, RNN, and LSTM.[6] Especially, Graph Neural Network (GNN) tends to predict stock prices very well.[7] This model structure allows us to include a large number of relations. Many studies also explore other graph based neural network models.[4] Some ideas include utilizing state-of-the-art models adapted for time series data, forecasting a distribution to reflect confidence in the prediction, and also incorporating an attention mechanism into the model.[8]

3 Data and Experiments

3.1 Data Collection

3.1.1 Sector and industry relations

We use a web scraper to download list of S&P 500 company list from wikipedia.[11] The S&P 500 stock market index is maintained by S&P Dow Jones Indices. It comprises 504 common stocks which are issued by 500 large-cap companies traded on American stock exchanges. Among them, two companies which have no trading price data on Yahoo Finance are removed. The companies' sector and sub-industry classifications are based on the Global Industry Classification Standard

(GICS), which is an industry taxonomy developed by MSCI and Standard & Poor's (S&P) for use by the global financial community. The GICS structure consists of 11 sectors and 158 sub-industries.

3.1.2 Wikidata company-based relations

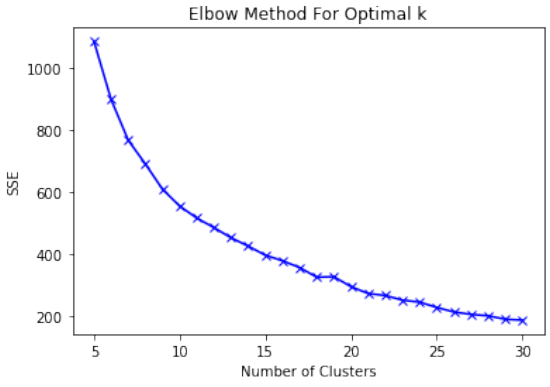
Wikidata is a collaboratively edited multilingual knowledge graph hosted by the Wikimedia Foundation. To collect data of the companies' relationships, the companies' tickers are used to look up their Wikidata IDs. Query requests are then set up to get the companies' subsidiary (P355) relationship data from *query.wikidata.org*. The following is an example of a query for the subsidiary relationship of Microsoft. Then we search the S&P 500 company list to see if any company is included in the returned subsidiary lists.

```
Query for SUBSIDIARY(P355) of MICROSOFT(Q2283):
SELECT DISTINCT ?item ?itemLabel WHERE {
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE]". }
  {
    SELECT DISTINCT ?item WHERE {
      ?item p:P355 ?statement0.
      ?statement0 (ps:P355/(wdt:P279*)) wd:Q2283.
    }
    LIMIT 100
  }
}
```

3.1.3 Stock correlation relations

To model the stock correlation relations, we calculate a correlation matrix based on the historical market closing price return data. This correlation matrix provides very valuable information about the inter-stock relations.[9] Then, we apply the KMeans clustering method to the correlation matrix. According to the elbow method, we set the optimal number of clusters to 18. The companies within the same clusters are then related to each other.

Figure 1: Eblow Method for Optimal k



3.1.4 Sequential data for feature engineering

We collect data on the 502 stocks from Yahoo Finance between 03/15/2022 and 4/15/2022. The prediction frequency is set to 2-minute level (4485 time points). Under this problem formulation, we aim to predict the realized volatilities in 10-minute intervals for each stock based on historical data. The following data preprocessing is conducted.

Data Preprocessing:

1. Calculate percentage returns of open price, high price, low price, closing price, and adjusted closing price.
2. Calculate 10-minute realized volatility based on adjusted closing return.
3. Get the response variable, which is the the realized volatility of the next ten-minute time window.
4. Check outliers and process missing data.
5. Encode categorical variables (ticker name) using label encoding.

3.2 Experiments

To model the relationship among stocks, we build a stock relationship graph. In the graph, each node represents a stock. Each node is attached with some price data. The nodes are then connected with a combination of the aforementioned stock relations.

Figure 2: Stock Relation Graph Based on 250 Sampled Tickers

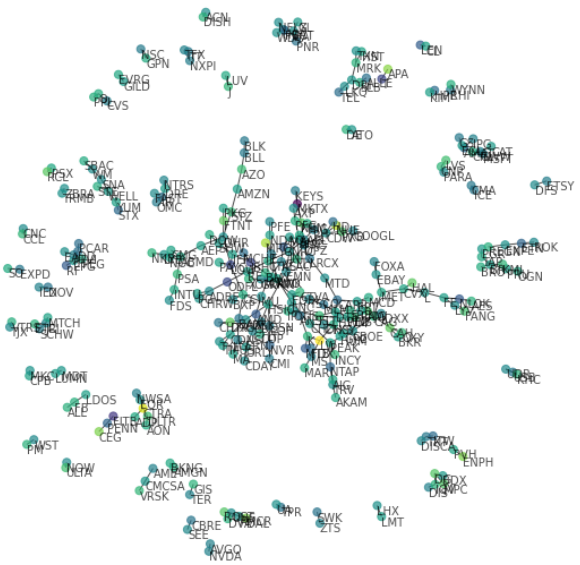
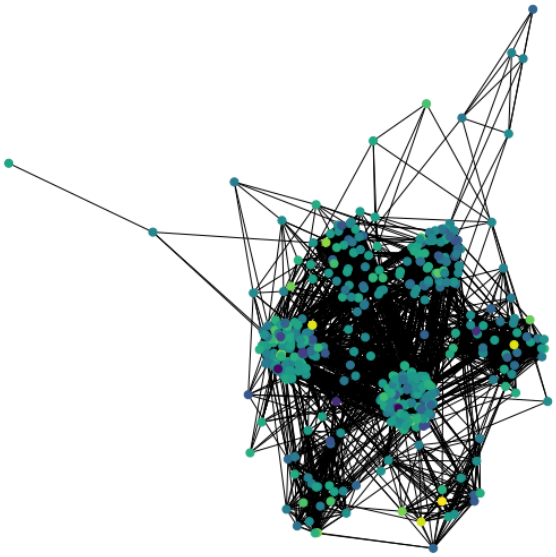


Figure 3: Complete Stock Relation Graph



4 Methodology

4.1 Temporal Graph Network (TGN)

The structure of our TGN model is based on the T-GCN traffic prediction model proposed by Zhao et al..[5] As a result, the process described below closely follow their work. We use the return of closing prices as the features.

Definition 1: Stock network G. We use an unweighted graph G = (V, E) to describe the relationship of the S&P 500 companies. We treat each stock as a node, where V is a set of stock nodes, $V = \{v_1, v_2, ..., v_N\}$, N is the number of the nodes, and E is a set of edges. The adjacency matrix A is used to represent the relationship between stocks, where $A \in \mathbb{R}^{N \times N}$. Each element of the adjacency matrix is either 0 or 1. The element 0 means there is no connection between stock i and stock j, while 1 denotes there is a connection.

Definition 2: We use price information as the feature of the nodes in the graph and represented as a feature matrix $X \in \mathbb{R}^{N \times P}$. P represents the dimension of the node features (the length of the historical time series).

Thus, we consider the mapping function f on the stock relation graph G and feature matrix X. Our goal is to predict the returns in the next 5 moments so that we can calculate the volatility based on returns, as shown in equation (1).

$$[X_{t+1}, \cdots, X_{t+T}] = f(G; (X_{t-n}, \cdots, X_{t-1}, X_t)) \tag{1}$$

There are two parts of the TGN model: the graph convolutional network and the gated recurrent unit. First, we use the historical n time series data as input and the graph neural network to obtain the features. Second, we input the obtained time time series into the gated recurrent unit model. Finally, we get results through the fully connected layer. One layer of the

GCN model can be expressed as:

$$f(X, A) = \sigma(\hat{A}XW) \tag{2}$$

The GRU model can be expressed as the following, where h_t denotes the output at time t; u_t and r_t are the update gate and reset gate at time t. $f(A, X_t)$ represents the output from the previous process. W and b represent the weights and deviations in the training process.

$$u_t = \sigma(W_u[f(A, X_t), h_{t-1}] + b_u) \tag{3}$$

$$r_t = \sigma(W_r[f(A, X_t), h_{t-1}] + b_r) \tag{4}$$

$$c_t = tanh(W_c[f(A, X_t), (r_t h_{t-1})] + b_c)] \tag{5}$$

$$h_t = u_t h_{t-1} + (1 - u_t)c_t \tag{6}$$

In the training process, the goal is to minimize the error between the target and the predicted value. We use Y_t and $\hat{Y}_t$ to denote the these two values respectively. The loss function of the TGN model is shown in equation (7). The first term is used to minimize the root mean square error between the real data and prediction. In the second term, λ is a hyperparameter. L_{reg} is an L2 regularization term, which helps to avoid overfitting.

$$loss = \|Y_t - \hat{Y}_t\|^2 + \lambda L_{reg} \tag{7}$$

4.2 ARIMA

Another method that this project seeks to examine is the autoregressive integrated moving average (ARIMA) model. An ARIMA model is a combination of an autoregressive (AR) model and a moving average (MA) model. For a time series dataset $X = \{x_1, x_2, \cdots, x_n\}$, the AR model uses lagged previous observations to predict the series' current value. Let x_t be the value of X at time t, then an p_{th} order AR model is structured as

$$x_t = \alpha_{AR} + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \cdots + \beta_p x_{t-p} + \epsilon.$$

Similarly, the MA model uses lagged errors of previous predictions to forecast the current value. A q_{th} order MA model is structured as

$$x_t = \alpha_{MA} + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \cdots + \phi_q \epsilon_{t-q}$$

where ϵ_t is the error at the t_{th} observation. An ARIMA model is then a combination of the two. An ARIMA(p,d,q) model is structured as follows

$$y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \cdots + \phi_q \epsilon_{t-q}$$

where $Y = \{y_t\}(2 \leq t \leq n)$ is the d_{th} order difference of X.

The ARIMA model is prevalent when it comes to finding patterns in time series data, making it a popular tool for analyzing stock market data sets. To compare the commonly used ARIMA model to the TGN model, the ARIMA model is applied onto the 2-minute return data and 10-minute volatility data. The models are trained and tested on Python using the *pmdarima* module[1], which uses a method similar to *R*'s *autoarima* and would automatically select the best

[1] https://pypi.org/project/pmdarima

ARIMA fit for a dataset based on the AIC and BIC scores. The ratio of the number of training data points to the number of testing data points is $\frac{0.8}{0.2}$. Since ARIMA models are univariate by nature, each company's data would have their own separate ARIMA model fitted. The mean of all of the models' RMSE are then taken in order to evaluate the model as a whole.

5 Results and Discussion

5.1 TGN

The TGN model gives us a RMSE of 0.14367, which is much better than most other models tested. This may because it captures a portion of the very complex relationships between companies. In summary, the TGN model works well to deal with the spatial dependence and temporal dynamics.

The result of prediction for the test dataset is shown as follows. For sequence modeling, LSTM has three gates including the input, output and forget gate; meanwhile, GRU has the reset and update gate. This means that GRU uses fewer training parameters and use less memory. So it can execute faster and train faster than LSTM. However, LSTM tends to more accurate than the GRU model. To achieve better results, we may consider to use LSTM model in a future study.

Figure 4: Test Result Visualization

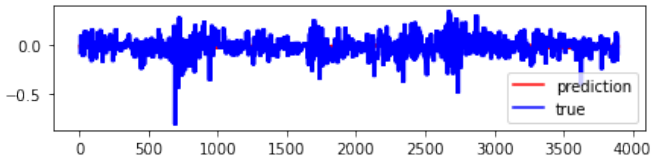
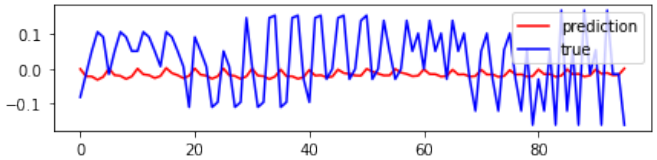


Figure 5: Oneday Test Result Visualization



5.2 ARIMA

ARIMA models are fitted for all 502 companies' 10-minute volatility data. The mean RMSE of all 502 ARIMA models is low at 0.0669. Purely judging from $RMSE$ alone, it seems the models do a fairly good job of predicting the stocks' realized volatility. However, among the 502 models, 280 of them are fitted to be $ARIMA(0, 0, 0)$. This means that ARIMA considers a majority of the dataset to be a stationary series (constant mean and variance) with no pattern to be observed in time. For these models, the testing data is predicted to be a constant straight line as a result. An example of this can be seen in figure 8, where the predicted volatility of 3M stock is simply a horizontal straight line. Upon further examination, the 10-minute volatility of stocks with similar straight line predictions have negligible first difference terms. In other words, the value of $x_t - x_{t-1}$ for stock volatility X is considered to zero at a statistically significant

degree. An example of this can be seen in figure 9 as a histogram of 3M stock's volatility's first difference term, where over 99% of the first difference terms are extremely close to 0.

Based on these observations, although the models' RMSEs are low overall, they do not do a good job fitting the dataset. Since the return data is taken at very short 2-minute intervals, for a lot of the stocks sampled, there is simply too much market fluctuation at a micro scale to discern any meaningful pattern. As another example, a time series plot of 3M stock's 2-minute return can be seen in fugure 10; it is visually apparent that the stock's return resembles white noise. Even though the models' average RMSE is low, no useful information can be extracted from them as long as the predicted volatility is a constant. Thus, the ARIMA model is a poor fit for predicting stock return volatility.

5.3 Other Baseline Models

We also used other models as our baseline model, such as multiple linear Rrgression, LightGBM, and neural network model. The RMSE for these three models are 1.453, 0.462, and 0.659, respetively; their RMSEs are much higher than the RMSE of the TGN model. Overall, TGN model has the best performance.

6 Conclusion

Overall, the temporal graph network model has the best performance among all of the models tested. This is because it works well with a spatial dependent and temporally dynamic problem such as stock return volatility prediction. Compared to baseline models, the TGN model is also able to take into consideration more information.

However, the TGN model has its limitation. There exists the graph structure limitation, as the model highly dependant on the relationship data it is fed. If we fail to create an accurate network structure between nodes, the model will not return satisfactory results. Because of the large volume of parameters to train, the TGN may not be the most efficient model when obtaining large amount of accurate data is challenging.

References

[1] Ge, Wenbo, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen, *"Neural Network–Based Financial Volatility Forecasting: A Systematic Review."*, ACM Computing Surveys (CSUR) 55, no. 1 (2022): 1-30.

[2] Whaley, Robert E. *"Understanding the VIX." The Journal of Portfolio Management 35*, no. 3 (2009): 98-105.

[3] Qinkai Chen, Christian-Yann Robert (2021), *Multivariate Realized Volatility Forecasting with Graph Neural Network*, arXiv, https://arxiv.org/abs/2112.09015, version 2.

[4] Fuli Feng et al. (2019), *Temporal Relational Ranking for Stock Prediction*, arXiv, https://arxiv.org/abs/1809.09441v2, version 2.

[5] Ling Zhao et al. (2020), *T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction*, IEEE Transactions on Intelligent Transportation Systems, Vol. 21, No. 9, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8809901&tag=1.

[6] Jianian Wang et al. (2022), *A Review on Graph Neural Network Methods in Financial Applications*, arXiv, https://arxiv.org/abs/2111.15367, version 2.

[7] Ramit Sawhney et al. (2022), *Time Evolving Spatio-Temporal Hypergraph for Stock Forecasting*, WSDM, Association for Computing Machinery, New York, NY.

[8] Emanuele Rossi et al. (2020), *Temporal Graph Networks for Deep Learning on Dynamic Graphs*, arXiv, https://arxiv.org/abs/2006.10637, version 3.

[9] Wei Li et al. (2022) *Modeling the Stock Relation with Graph Network for Overnight Stock Movement Prediction*, Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20).

[10] *Optiver Realized Volatility Prediction* (2022), Kaggle, https://www.kaggle.com/competitions/optiver-realized-volatility-prediction/overview/description.

[11] *List of S&P 500 Companies*, Wikiepdia, https://en.wikipedia.org/wiki/List_of_S%26P_500_companies.

Appendix: Plots and Tables

Figure 6: RMSE for TGN Model

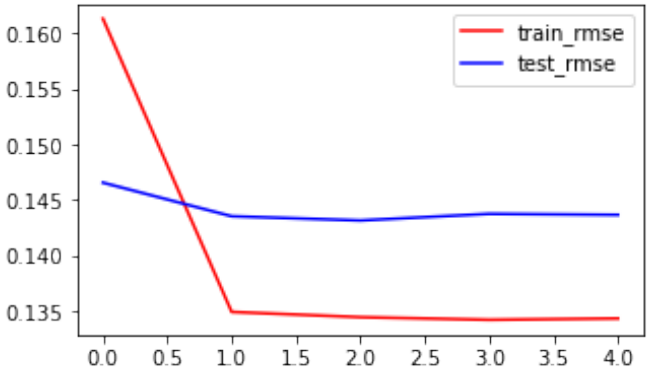


Figure 7: Loss for TGN Model

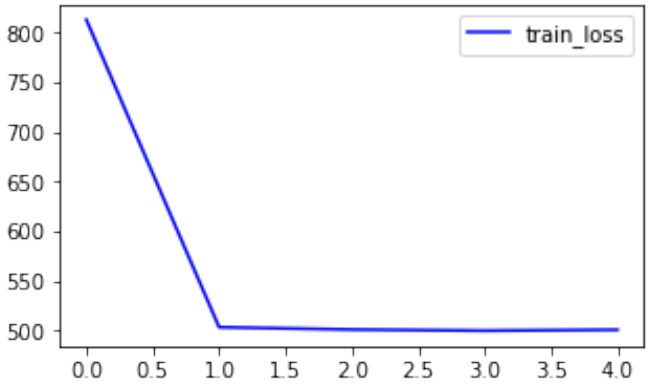


Figure 8: Actual and predicted data of 3M stock return's 10-minute volatility.

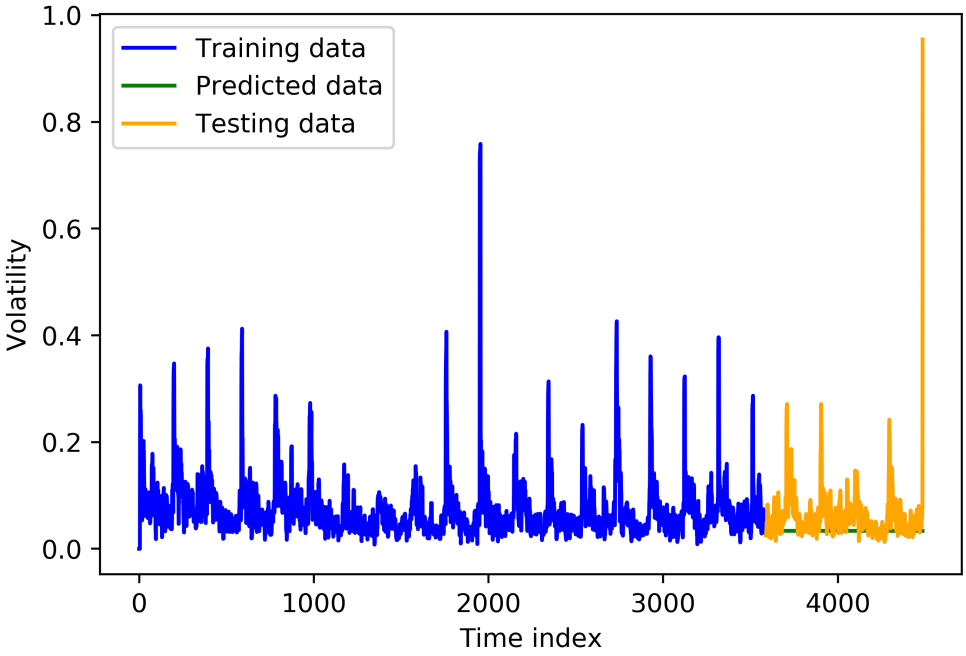


Figure 9: Histogram of 3M stock return's 10-minute volatility's first difference.

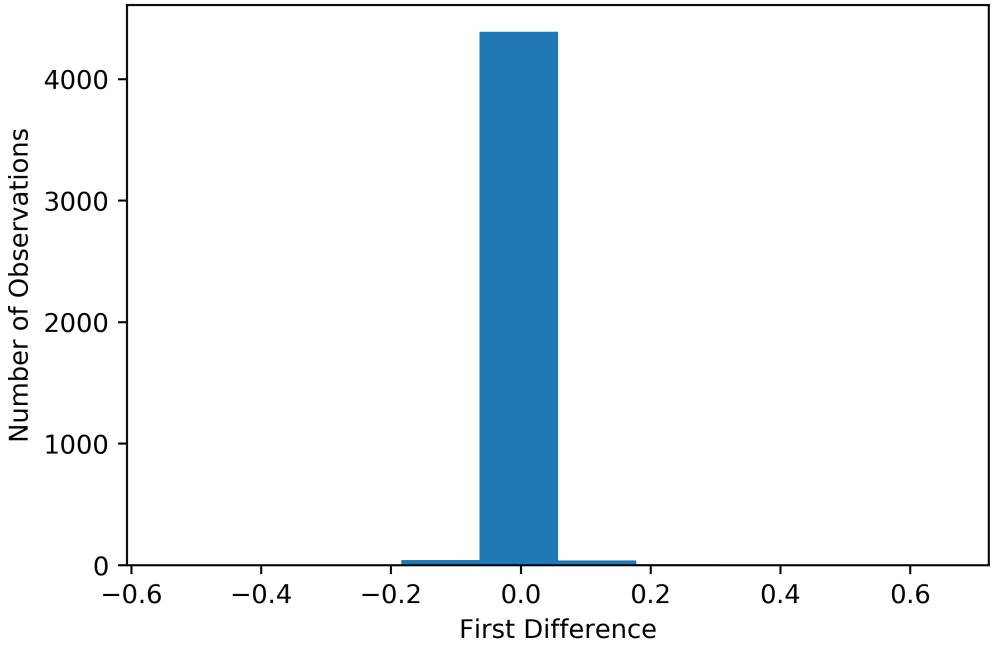


Figure 10: Time series plot of 3M stock 2-minute return.