FE 570 – Market Microstructure and Trading Strategies

Fall’ 2022

Empirical Analysis of Microstructure Data

Final Project

Team 6: Sandeep Ranjan

Introduction:

The purpose of this project is to perform Empirical analysis of microstructure data. The stock chosen was Tesla Inc. (TSLA). The stock returns for the last few years were analyzed and it was observed that on 3rd February 2020, TSLA had one of the biggest jumps when the stock gained almost 20%.

As part of this project, the following tasks were performed:

- Retrieve tick level dataset, perform data cleansing, and organize in Trades & Quotes (TAQ) format
- Evaluate various Spread measures to study Liquidity dynamics
- Estimate Volatility using intraday data
- Estimate Probability of Informed Trading (PIN) measure

Data Retrieval:

Tick data for TSLA stock for 02/03/2020 from Refinitiv was obtained from the Hanlon Lab at Stevens. The raw data set had about 1.19 million records just for one day.

1	#RIC	Alias Und	Domain	Date-Time	GMT Offset	Type	Ex/Cntrb.	LOC	Price	Volume	Market V	Buyer ID	Bid Price	Bid Size	No. Buyer	Seller ID	Ask Price	Ask Size	No. Seller	Qualifiers	Seq. No.	Exch Time	Blo
47	TSLA.O		Market Price	2020-02-03T10:21:08.544020242Z	-5	Quote						NAS	647.05	2		NAS	649	4		[PRC_QL_CD]; [F		21:08.5	
48	TSLA.O		Market Price	2020-02-03T10:21:08.552021356Z	-5	Quote						NAS	647.05	1		NAS	649	4		[PRC_QL_CD]; [F		21:08.5	
49	TSLA.O		Market Price	2020-02-03T10:21:45.075573575Z	-5	Trade	NAS		647.36	4			647.05	1			649	4		@ TI[GV4	2931	21:45.1	
50	TSLA.O		Market Price	2020-02-03T10:22:02.710304871Z	-5	Quote						NAS	647.05	1		NAS	649	5		[PRC_QL_CD]; [F		22:02.7	
51	TSLA.O		Market Price	2020-02-03T10:22:11.034349884Z	-5	Quote						NAS	647.05	1		NAS	649	6		[PRC_QL_CD]; [F		22:11.0	
52	TSLA.O		Market Price	2020-02-03T10:22:39.001652957Z	-5	Quote						NAS	647.05	1		NAS	648.5	6		[PRC_QL_CD]; [F		22:39.0	
53	TSLA.O		Market Price	2020-02-03T10:22:39.002290334Z	-5	Trade	NAS		648.5	2			647.05	1			648.5	6		@ TI[GV4	2933	22:39.0	
54	TSLA.O		Market Price	2020-02-03T10:22:42.153534499Z	-5	Trade	PSE		648.5	3			647.05	1			648.5	6		@ TI[GV4	2934	22:42.1	
55	TSLA.O		Market Price	2020-02-03T10:22:44.077660477Z	-5	Quote						NAS	647.1	1		NAS	648.5	6		[PRC_QL_CD]; [F		22:44.1	
56	TSLA.O		Market Price	2020-02-03T10:24:42.403133906Z	-5	Quote						NAS	647.1	1		PSE	647.81	1		[PRC_QL_CD]; [F		24:42.4	
57	TSLA.O		Market Price	2020-02-03T10:26:16.789393793Z	-5	Trade	NAS		647.79	1			647.1	1			647.81	1		@ TI[GV4	2960	26:16.8	
58	TSLA.O		Market Price	2020-02-03T10:27:01.108584461Z	-5	Quote						NAS	647.1	1		NAS	648.5	7		[PRC_QL_CD]; [F		27:01.1	
59	TSLA.O		Market Price	2020-02-03T10:27:01.109469815Z	-5	Trade	PSE		647.81	20			647.1	1			648.5	7		@ TI[GV4	2968	27:01.1	
60	TSLA.O		Market Price	2020-02-03T10:27:03.064445651Z	-5	Quote						NAS	647.1	1		PSE	647.81	1		[PRC_QL_CD]; [F		27:03.0	
61	TSLA.O		Market Price	2020-02-03T10:28:07.526939852Z	-5	Trade	PSE		647.81	10			647.1	1			647.81	1		@ TI[GV4	2976	28:07.5	
62	TSLA.O		Market Price	2020-02-03T10:29:23.733585422Z	-5	Trade	PSE		647.73	1			647.1	1			647.81	1		@ TI[GV4	2986	29:23.7	
63	TSLA.O		Market Price	2020-02-03T10:29:32.621968299Z	-5	Trade	NAS		647.79	1			647.1	1			647.81	1		@FTI[GV4	2987	29:32.6	
64	TSLA.O		Market Price	2020-02-03T10:30:17.619667675Z	-5	Trade	PSE		647.51	1			647.1	1			647.81	1		@FTI[GV4	3000	30:17.6	
65	TSLA.O		Market Price	2020-02-03T10:30:18.220123379Z	-5	Quote						NAS	647.1	1		NAS	649	6		[PRC_QL_CD]; [F		30:18.2	

Figure 1. Tick data for TSLA

Data Cleansing:

The raw dataset was analyzed and cleaned up using R programming language. Below are some of the key steps that were performed -

- Filter trades for NASDAQ exchange (Ex.Cntrb.ID = 'ADF')
- Few trades records were having Price = 0. Such records were filtered out.
- Any duplicate trades/quotes records were ignored
- Any Trading Activity outside of the normal US market hours were ignored, I.e., we only considered trades between 9.30 AM – 4 PM EST
- Some of the key functions used from the *highfrequency* R package were – *mergeQuotesSameTimestamp, mergeTradesSameTimestamp*, *matchTradesQuotes, aggregateTrades, getTradeDirection* & *getLiquidityMeasures*
- Trades and quotes data set were grouped in time buckets of 1 sec, 10 sec, 30 sec and 1 minute.

Cleaned up data in Trades & Quotes (TAQ) format:

```
> head(tqdata.xts,10)
                        SYMBOL   BID       OFR       OFRSIZ BIDSIZ QUOTEEX MIDQUOTE  PRICE      NUMTRADES SIZE    EX    TRADE_DIRECTION
2020-02-03 09:30:00.937 "TSLA.O" "674.065" "674.880" "   4" "   5" ""      "674.475" "674.8800" "3"       "  599" "ADF" " 1"
2020-02-03 09:30:00.946 "TSLA.O" "674.070" "674.880" "   1" "   1" ""      "674.475" "674.2450" "4"       "   32" "ADF" "-1"
2020-02-03 09:30:00.947 "TSLA.O" "674.070" "674.880" "   1" "   1" ""      "674.475" "674.8800" "1"       "   25" "ADF" " 1"
2020-02-03 09:30:00.955 "TSLA.O" "674.070" "674.880" "   1" "   1" ""      "674.475" "674.8800" "1"       "    5" "ADF" " 1"
2020-02-03 09:30:01.032 "TSLA.O" "674.070" "674.880" "   1" "   1" ""      "674.475" "674.0700" "1"       "  100" "ADF" "-1"
2020-02-03 09:30:01.164 "TSLA.O" "674.070" "674.880" "   2" "   4" ""      "674.475" "673.6900" "1"       "  193" "ADF" "-1"
2020-02-03 09:30:01.274 "TSLA.O" "674.070" "674.875" "   2" "   2" ""      "674.475" "674.8600" "1"       "   50" "ADF" " 1"
2020-02-03 09:30:01.311 "TSLA.O" "674.480" "674.900" "   2" "   1" ""      "674.690" "674.8950" "1"       "    1" "ADF" " 1"
2020-02-03 09:30:01.340 "TSLA.O" "674.480" "674.900" "   1" "   1" ""      "674.690" "674.6900" "1"       "  100" "ADF" "-1"
2020-02-03 09:30:01.384 "TSLA.O" "674.590" "674.980" "   2" "   2" ""      "674.840" "674.7050" "1"       "   10" "ADF" "-1"
```

```
> tail(tqdata.xts,10)
                        SYMBOL   BID       OFR       OFRSIZ BIDSIZ QUOTEEX MIDQUOTE  PRICE      NUMTRADES SIZE    EX    TRADE_DIRECTION
2020-02-03 15:59:59.410 "TSLA.O" "780.000" "780.290" "   3" "1703" ""      "780.145" "780.0500" "1"       "    4" "ADF" "-1"
2020-02-03 15:59:59.418 "TSLA.O" "780.000" "780.290" "   3" "1703" ""      "780.145" "780.2800" "1"       "   11" "ADF" " 1"
2020-02-03 15:59:59.470 "TSLA.O" "780.000" "780.290" "   3" "1703" ""      "780.145" "780.1601" "1"       "  100" "ADF" " 1"
2020-02-03 15:59:59.566 "TSLA.O" "780.000" "780.290" "   3" "1702" ""      "780.145" "780.2700" "1"       "   40" "ADF" " 1"
2020-02-03 15:59:59.578 "TSLA.O" "780.000" "780.290" "   3" "1702" ""      "780.145" "780.0500" "1"       "    1" "ADF" "-1"
2020-02-03 15:59:59.590 "TSLA.O" "780.000" "780.290" "   3" "1702" ""      "780.145" "780.0934" "2"       "   90" "ADF" "-1"
2020-02-03 15:59:59.622 "TSLA.O" "780.000" "780.290" "   3" "1702" ""      "780.145" "780.0001" "1"       "   24" "ADF" "-1"
2020-02-03 15:59:59.874 "TSLA.O" "780.010" "780.290" "  27" "   1" ""      "780.150" "780.1794" "1"       "   50" "ADF" " 1"
2020-02-03 15:59:59.902 "TSLA.O" "780.010" "780.290" "  27" "   1" ""      "780.150" "780.2620" "1"       "    1" "ADF" " 1"
2020-02-03 15:59:59.906 "TSLA.O" "780.010" "780.290" "  27" "   1" ""      "780.150" "780.1601" "1"       "  100" "ADF" " 1"
```

Figure 2. Cleaned up TAQ dataset for TSLA

No. of trades	No. Of quotes	No. Of Trades and Quotes
`> nrow(tdata.xts)` `[1] 276004`	`> nrow(qdata.xts)` `[1] 221462`	`> nrow(tqdata.xts)` `[1] 276004`

Cleaned up Trades data set:

```
> head(tdata,10)
                        DT SYMBOL   PRICE NUMTRADES SIZE  EX
 1: 2020-02-03 09:30:00.937 TSLA.O 674.880         3  599 ADF
 2: 2020-02-03 09:30:00.946 TSLA.O 674.245         4   32 ADF
 3: 2020-02-03 09:30:00.947 TSLA.O 674.880         1   25 ADF
 4: 2020-02-03 09:30:00.955 TSLA.O 674.880         1    5 ADF
 5: 2020-02-03 09:30:01.032 TSLA.O 674.070         1  100 ADF
 6: 2020-02-03 09:30:01.164 TSLA.O 673.690         1  193 ADF
 7: 2020-02-03 09:30:01.274 TSLA.O 674.860         1   50 ADF
 8: 2020-02-03 09:30:01.311 TSLA.O 674.895         1    1 ADF
 9: 2020-02-03 09:30:01.340 TSLA.O 674.690         1  100 ADF
10: 2020-02-03 09:30:01.384 TSLA.O 674.705         1   10 ADF
```

Cleaned up Quotes data set:

```
> head(qdata,10)
                        DT SYMBOL    BID    OFR OFRSIZ BIDSIZ EX MIDQUOTE
 1: 2020-02-03 09:30:00.245 TSLA.O 673.52 673.87      1      1     673.695
 2: 2020-02-03 09:30:00.262 TSLA.O 673.52 673.88      4      1     673.700
 3: 2020-02-03 09:30:00.572 TSLA.O 673.54 673.88      4      1     673.710
 4: 2020-02-03 09:30:00.616 TSLA.O 673.52 673.88      4      1     673.700
 5: 2020-02-03 09:30:00.859 TSLA.O 673.54 673.88      4      1     673.710
 6: 2020-02-03 09:30:00.883 TSLA.O 673.52 673.88      4      1     673.700
 7: 2020-02-03 09:30:00.911 TSLA.O 673.52 674.45      1      1     673.985
 8: 2020-02-03 09:30:00.911 TSLA.O 673.52 673.98      1      1     673.750
 9: 2020-02-03 09:30:00.912 TSLA.O 673.60 674.45      2      2     674.065
10: 2020-02-03 09:30:00.912 TSLA.O 673.68 674.21      2      2     674.065
```

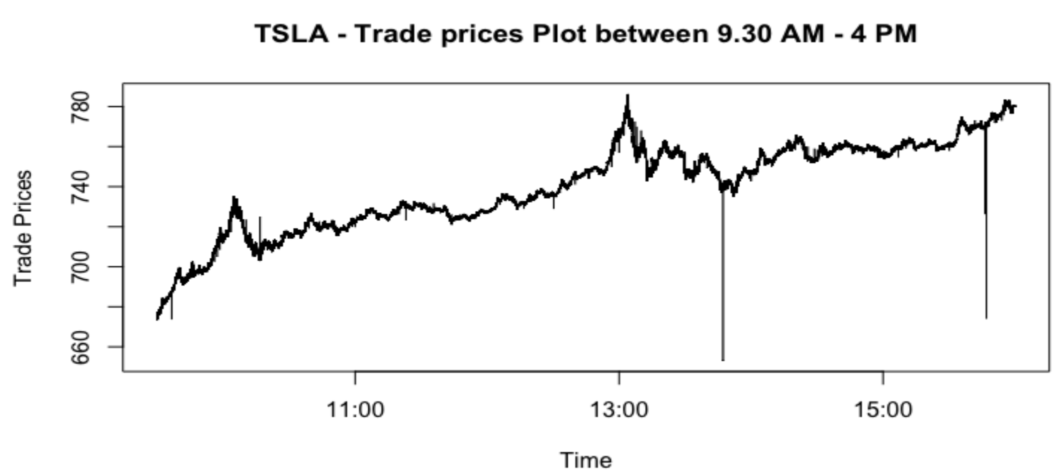


Figure 3. Plot of Trade Prices during exchange hours

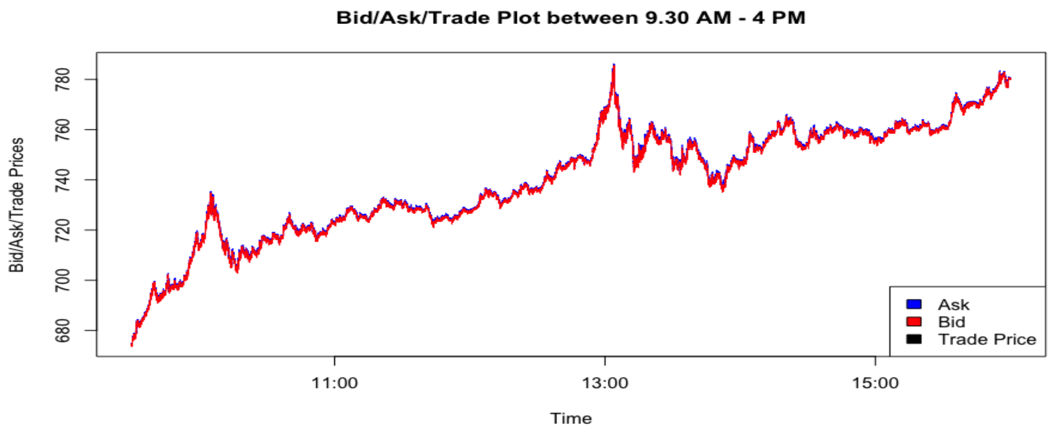


Figure 4. Plot of Bid/Ask/Trade prices during exchange hours

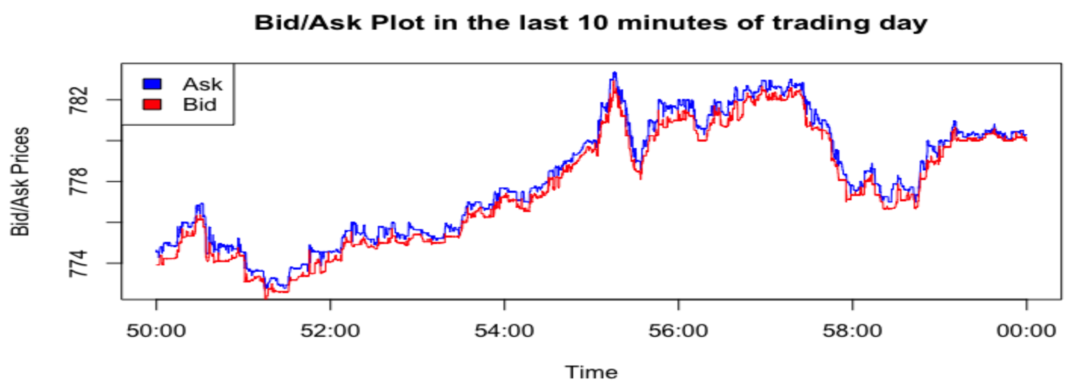


Figure 5. Plot of Bid/Ask prices during last 10 minutes of trading day

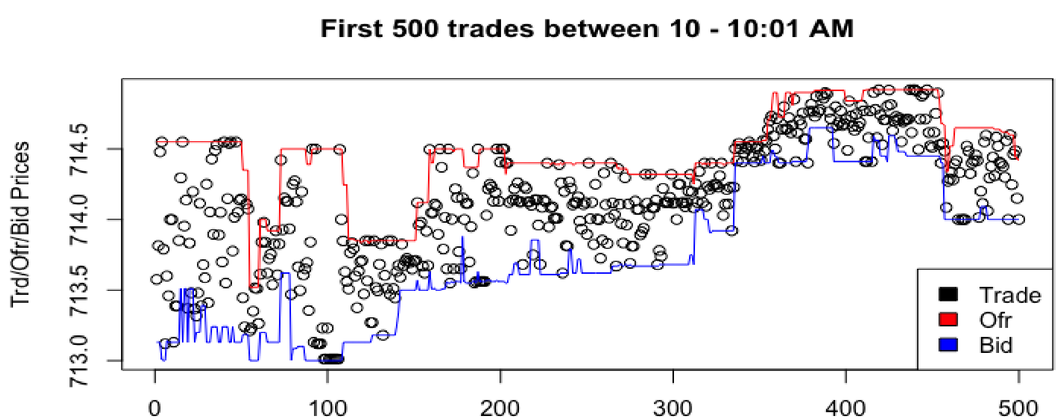


Figure 6. Plot showing how many trades were done within the bid/ask between 10 – 10:01 AM

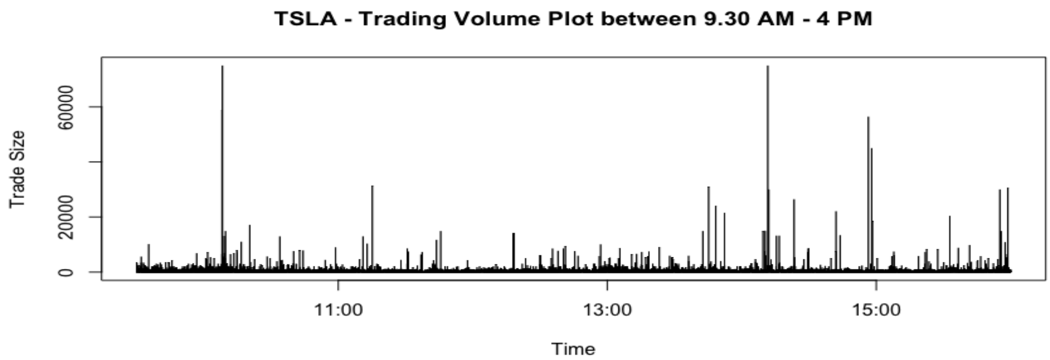


Figure 7. Trading Volume

Observations:

Looking at Figure 3, it appears that there were few trades done for which the trade prices were considerably lower than the other trades that happened during the same period. This is evident from the TAQ data set. This probably looks to be just bad data since these were SELL trades done at considerably low prices compared to the Bid Prices

```
> tqdata.xts[tqdata.xts$PRICE <= 680 & tqdata$DT >= '2020-02-03 13:00:00',]
                        SYMBOL   BID       OFR       OFRSIZ BIDSIZ QUOTEEX MIDQUOTE  PRICE      NUMTRADES SIZE    EX    TRADE_DIRECTION
2020-02-03 13:47:14.326 "TSLA.O" "740.820" "741.940" "   1" "   3" ""      "741.380" "653.0000" "1"       "   28" "ADF" "-1"
2020-02-03 15:47:00.138 "TSLA.O" "770.000" "770.660" "   4" "   7" ""      "770.330" "674.1650" "1"       "    1" "ADF" "-1"
> |
```

Figure 8. TAQ data after 1 PM where trade price <= $ 680

Below we can see how many trades (for the entire day) were done within the spread, at bid, at ask and outside the spread.

```
  n.trades_within_spread n.trades_at_bid n.trades_at_ask n.trades_outside_spread
1                 241964           15091           12437                    6515
```

```
  %_trades_within_spread %_trades_at_bid %_trades_at_ask %_trades_outside_spread
1                87.6668         5.46767         4.50609                 2.36047
```

Trade stats for TSLA for 02/03/2020:

```
> trade_stats
  mean_price mean_bid mean_ask total_volume_traded
1    737.263  736.862  737.605            22231941
```

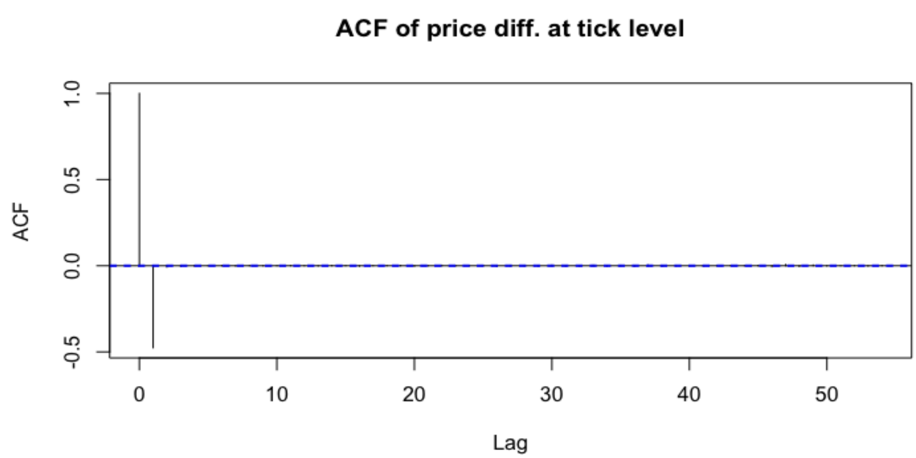
Summary statistics of price changes (p(t) - p(t-1)) at different sampling frequencies:

Return Statistics	tqdata	tqdata.1sec	tqdata.10sec	tqdata.30sec	tqdata.1min
N(Obs.)	276003	23322	2340	780	390
Range	[-96.1650, 96.4886]	[-7.2600, 6.8924]	[-9.7690, 6.9651]	[-13.1385, 6.7043]	[-12.6596, 8.8179]
Mean	0.0003814455	0.004514197	0.0449915	0.1349745	0.269949
Std. Deviation	0.5334836	0.450891	1.027228	1.694026	2.52278
Kurtosis	13472.82	14.77001	7.070126	6.331224	3.937051

Table 1. Summary Statistics of price change

The Mean of price changes is approximately 0 for tick level and 1 sec aggregation. However, as the # of observations decreases, the mean of price changes tends to show a non-zero value.

Autocorrelation Plot ACF of price differences (returns):



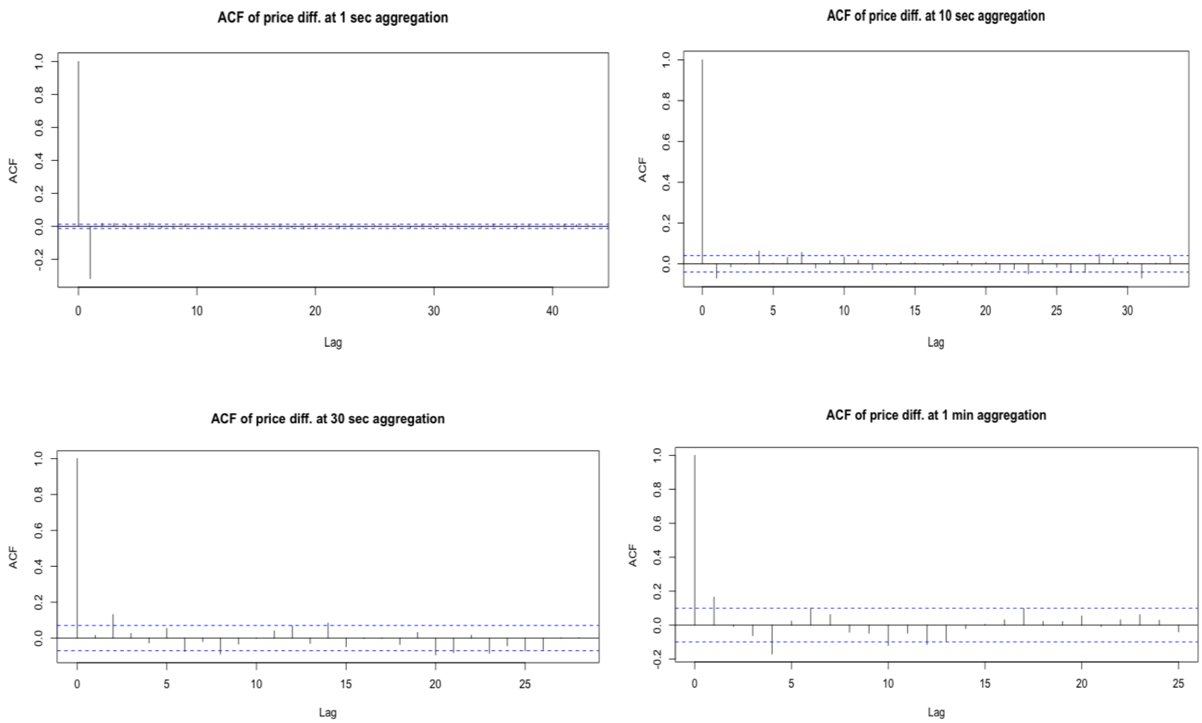


Figure 9. Auto Correlation Plots of price change

Observations:

- The price returns have mean close to zero
- For trades aggregated at 10 sec or more interval, we see autocorrelations.
- For the tick level data there is noticeable negative autocorrelation at lag 1.

Liquidity

Liquidity is the property of markets which allows for rapid and cheap trade execution. It is the most important characteristic of a well-functioning market. It has several dimensions – time, size & cost.

The main liquidity measures used in microstructure are spread based measures as defined below:

Quoted Spread - is defined in terms of best bid and best ask prices

$$s^Q = \frac{1}{T}\sum_{t=1}^{T}(a_t - b_t)$$

Effective Spread – measures the cost of immediate execution. It can also be referred to as the true cost of round-trip trade. It is defined as twice the difference between the trade price and the fundamental value. Since the fundamental value is not known, it is proxied as the mid-point price which is nothing but the average of best bid and best ask price.

$$ES = \frac{1}{T}\sum_{t=1}^{T} 2q_t(p_t - m_t)$$

Realized Spread – Effective spread does not consider price movements induced by trading. The realized spread adds a delay to the mid-price of ~ 5 min, allowing the price impact to be absorbed into prices. Here the proxy value used in the delayed mid-price

$$RS = \frac{1}{T}\sum_{t=1}^{T} 2q_t(p_t - m_{t+\delta})$$

We have the relation **Effective Spread = Price Impact + Realized Spread**

This relation expresses the point of view of a market maker submitting limit orders (liquidity). The ES is the expected profit of the MM, of which RS is a more realistic estimate, net of losses to informed traders (PI). PI is a measure of the Adverse Selection cost to the MM from Informed Traders.

```
> spread_measures
                   tickData tqdata.1sec tqdata.10sec tqdata.30sec tqdata.1min
Quoted_spread     0.7438347   0.7365459    0.7394511    0.7349808   0.7308824
Effective_Spread  0.4022092   0.3901718    0.3876833    0.3805618   0.3544478
Realized_Spread   0.2654133  -0.3049043   -3.9812144   -8.8000297 -10.0211606
```

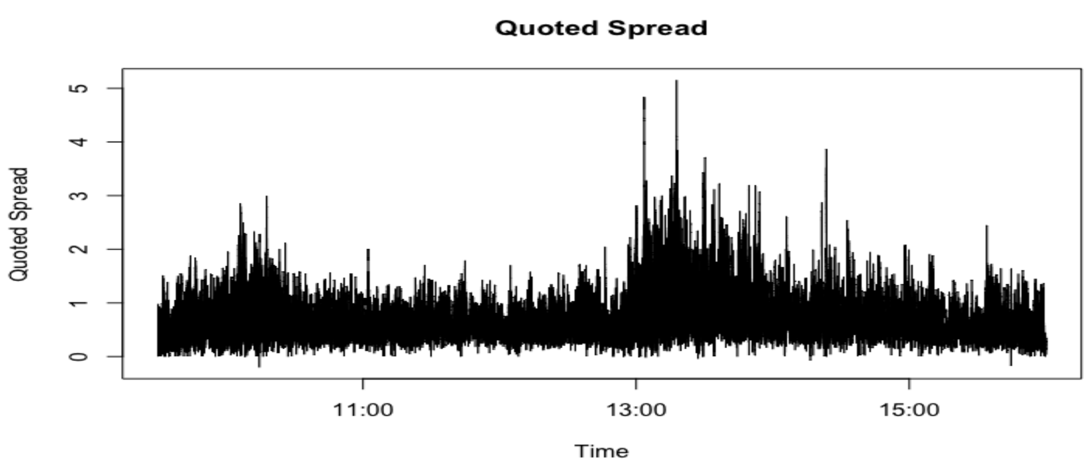


Figure 10. Plot of Quoted Spread

The quoted spread is $0.74 and effective spread is $0.40, which indicates high volatility of TSLA stock

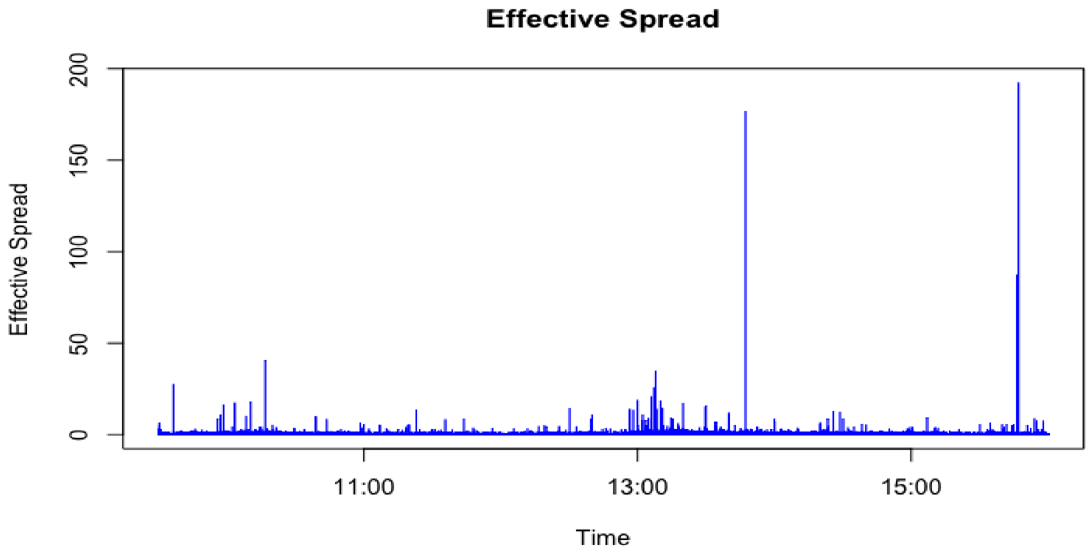


Figure 11. Plot of Effective Spread

From the above plot of Effective spread, the 2 peaks are in line with the 2 trades that were done where the trade price was off compared to the bid/ask price.

```
> tqdata.xts[tqdata.xts$PRICE <= 680 & tqdata$DT >= '2020-02-03 13:00:00',]
                        SYMBOL    BID       OFR       OFRSIZ BIDSIZ QUOTEEX MIDQUOTE  PRICE      NUMTRADES SIZE    EX    TRADE_DIRECTION
2020-02-03 13:47:14.326 "TSLA.O" "740.820" "741.940" "   1" "    3" ""      "741.380" "653.0000" "1"       "   28" "ADF" "-1"
2020-02-03 15:47:00.138 "TSLA.O" "770.000" "770.660" "   4" "    7" ""      "770.330" "674.1650" "1"       "    1" "ADF" "-1"
> 
```

Volatility Estimation

Volatility is a measure of the variability of returns of a traded asset. Intuitively, an asset with larger volatility is expected to have a larger price change over the same time-period.

There are 2 ways of estimating the Volatility from microstructure data

Method 1:

Sampling the trade prices at frequency q: The daily volatility at lag q is given by -

$$\sigma^2_{Day}(q)(q\Delta t) = Var(\Delta p_q)$$

where

$$\Delta p_q := p_{t+q} - p_t$$

Typically, a lag of 5 minutes is sufficient for the noise term to average out. The required lag can be estimated visually from the signature plot which is a graphical representation of daily volatility vs lag q. This plot plateaus where the daily volatility becomes independent of the lag q.

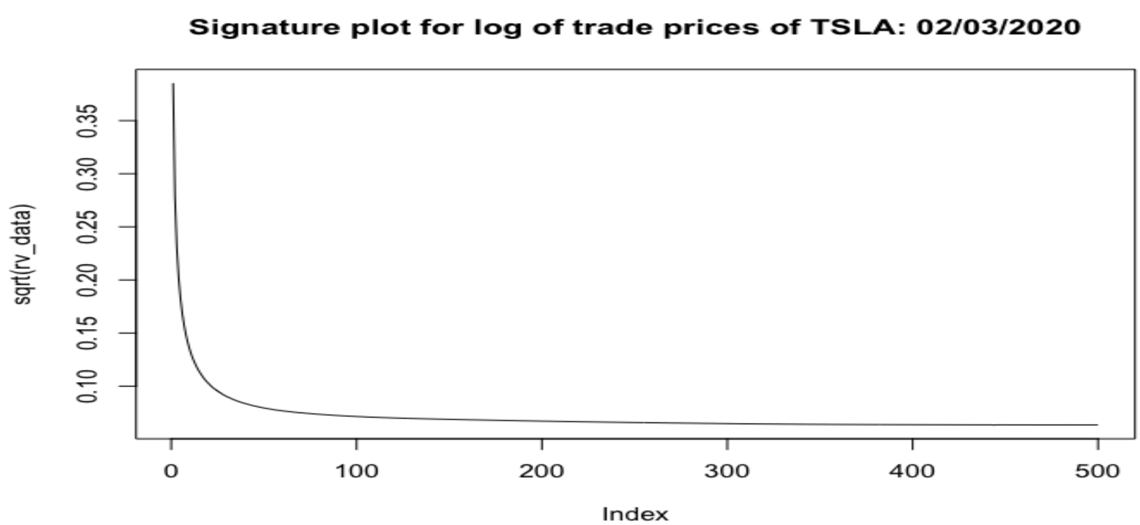


Figure 12. Signature Plot of TSLA

Observations about Signature Plot -

- At the highest sampling frequency (lag = 1), the estimated volatility is very large. This is due to the bid-ask bounce noise.
- The bid-ask bounce noise averages out at large lags and at lags ~ 50 the estimated volatility is only due to fluctuations of the efficient price

Below we show calculated values of realized Volatility at lags 1, 2, 5, 50, 100 & 500 using **log of trade prices**

realized_Vol1	realized_Vol2	realized_Vol5	realized_Vol50	realized_Vol100	realized_Vol500
0.385334	0.278684	0.182318	0.0794923	0.071435	0.0635368

Hence the volatility can be estimated as 0.0635368. Note that this was calculated using log of trade prices.

Method 2:

Roll Model - Assuming the independence of the trading signs dt, the Roll model gives an estimate for the Volatility of the efficient price

$$\sigma_u^2 = var(\Delta m_t) = \gamma_0 + 2\gamma_1$$

where

$$\gamma_0 = var(\Delta p_t), \gamma_1 = cov(\Delta p_t, \Delta p_{t-1})$$

In Roll Model, the trades prices are decomposed into 2 components:

1. Efficient Price: This is the slow-moving component. It is the fundamental value of the asset and embeds information about future earnings of the stock.
2. Noise: This is the rapidly changing up-down component which is responsible for the bid-ask bounce. Normally denoted by q(t), it's possible value is {+1, -1} and it shows the trade direction.

This can be converted to daily volatility by multiplying with total trades in a day:

$$(\sigma_{day}^{Roll})^2 = \sigma_u^2 n_{trades}$$

For TSLA, **Roll model estimate of Volatility comes out to be 0.0827528 (**calculated using log of prices**).** Clearly this is larger than the value obtained in Method 1 using the sampling approach because Roll model estimate of Volatility also includes contribution from the trading activity.

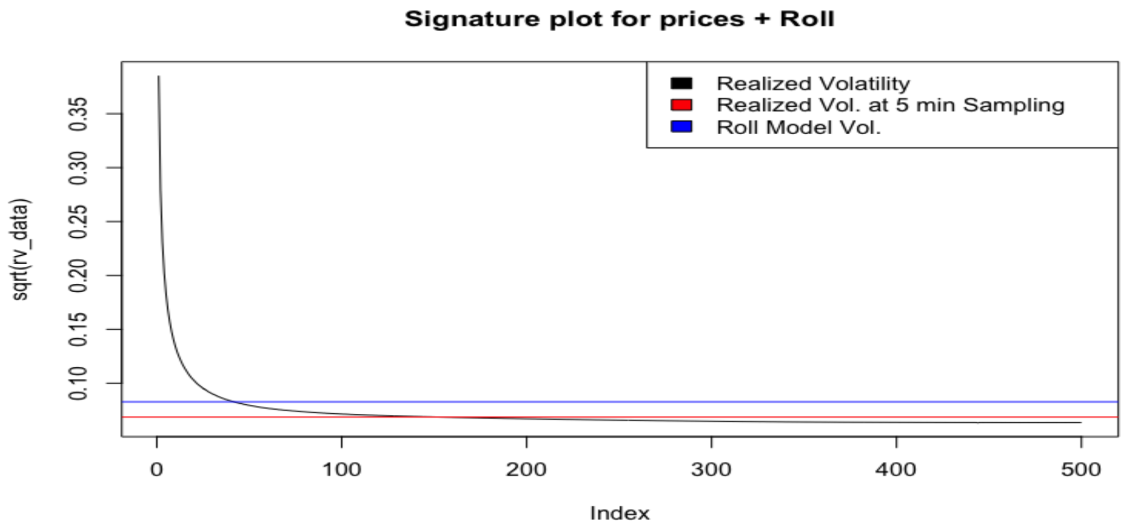


Figure 13. Signature Plot + Roll Model Volatility

The Roll model estimate is less reliable than the sampling method because the Roll model estimate is biased since the autocorrelation of trade signs is non-vanishing.

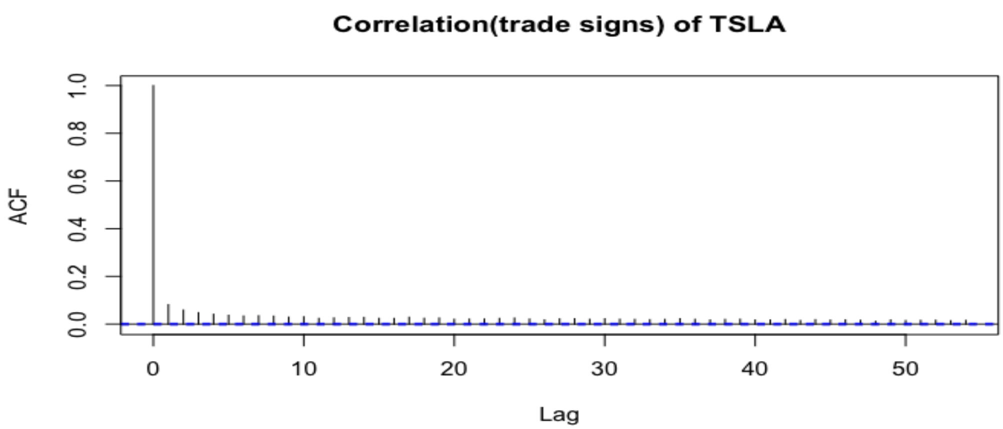


Figure 14. Autocorrelation of Trade Signs

Probability of Informed Trading (PIN)

The PIN is the unconditional probability that a randomly chosen trader on a randomly chosen day is informed. PIN is one of the primary measures of proxy information asymmetry in the market. The structural model is driven from maximum likelihood estimation (MLE). However, estimating PIN using MLE algorithms has been shown to be problematic, resulting in biased or unavailable estimates.

Here we make use of the “InfoTrad” package in R to calculate PIN. We will use the different factorizations available – EHO, LK through the YZ, GAN and EA algorithms. These algorithms help overcome the bias introduced due to boundary estimates.

$$PIN = \frac{\alpha\mu}{\alpha\mu+\epsilon_B+\epsilon_S}$$

Where:

News arrives at rate α
Probability of good news is δ
Probability of bad news is $1-\delta$
Informed trades arrive at rate μ
Uninformed buys/sells arrive with intensities ϵ_B/ϵ_S

We calculated the # of buys and sells per minute:

```
          time_interval TRADE_DIRECTION count
1  2020-02-03 09:30:00              -1   623
2  2020-02-03 09:30:00               1   675
3  2020-02-03 09:31:00              -1   440
4  2020-02-03 09:31:00               1   797
5  2020-02-03 09:32:00              -1   634
6  2020-02-03 09:32:00               1   958
7  2020-02-03 09:33:00              -1   456
8  2020-02-03 09:33:00               1   759
9  2020-02-03 09:34:00              -1   418
10 2020-02-03 09:34:00               1   777
11 2020-02-03 09:35:00              -1   449
```

```
> head(data_buy_sell)
     buys sells
[1,]  675   623
[2,]  797   440
[3,]  958   634
[4,]  759   456
[5,]  777   418
[6,]  741   449
```

We chose initial parameter values as below:

```
# Initial parameter values
# par0 = (alpha, delta, mu, epsilon_b, epsilon_s)
par0 = c(0.5,0.5,300,400,500)
```

After executing the optimization (using optim function in stats package), the estimated parameter values are -

```
      alpha     delta       mu       eb       es
1 0.4366017 0.6499111 335.4736 758.5145 485.7301
```

And the Probability of Informed Trading is: **0.1053**

```
pin
[1] 0.1053189
```

Observation:

The calculated PIN measure doesn't seem to be too high, which implies that there were few informed traders trading TSLA on this day. Probably most of the news about the stock was public.

We also calculated PIN and the other parameter values using the other 3 algorithms in the InfoTrad package – YZ, GAN and EA using each of the likelihood functions: LK and EHO.

	Method	Factorization	PIN	alpha	delta	mu	epsilon-b	epsilon-s
1	YZ	LK	0.112152015253059	0.360790741714478	0.445585495100842	402.906885490443	747.871259367155	498.423311184494
2	YZ	EHO	0.20785220708056	0.900000071525574	0.700000238418579	306.607626710114	745.056532905579	370.657598448118
3	GAN	LK	0.104456661254423	0.360798678500515	0.445601204125618	402.90723301958	747.870703683595	498.423196979575
4	GAN	EHO	0.167443134790805	0.536000137329102	0.323310187767292	430.138175159666	837.455577421189	308.900267170823
5	EA	LK	0.105947966206357	0.360878690162715	0.388132928157913	408.567325432714	737.270006201926	506.943869955558
6	EA	EHO	0.207650352484322	0.589915075303249	0.0371688852891233	508.899395470732	645.385258979443	500.142522185665

References

1. Financial Markets and Trading – *Anatoly B. Schmidt*
2. Market Liquidity: Theory, Evidence and Policy – *Thierry Foucault, Marco Pagano and Alisa Roell*
3. InfoTrad: An R Package for estimating the probability of Informed Trading – *Duygu Celik and Murat Tinic*
4. PIN: Measuring Asymmetric Information in Financial Markets with R – *Paolo Zagaglia*

R code

```
################################################################
################

## FE 570 - Final Project ##

## Sandeep Ranjan ##

## Empirical Analysis of Micro structure Data ##

# Project Ask:

# Empirical analysis of micro structure data.
```

```
#

# Download a tick level data set from Refinitiv (Trade and Quote data set). Clean it up and organize it as a TAQ format data. Analyze the resulting dataset:

#

# Perform a study of liquidity: compute the spread measures (quoted spread, effective spread, realized spread) in time buckets and study the intra-day liquidity dynamics

# Estimate the volatility using intraday data

# Estimate the probability of informed trading (PIN measure)



############################################################
###############


# load packages

library(highfrequency)

library(xts)

library(data.table)

library(ggplot2)

library(TTR)

library(timeDate)

library(quantmod)
```

```
library(InfoTrad)

Sys.setenv(TZ='EST')
options(digits.secs=3)

# print the time zone
Sys.timezone()

mkt_open <- '2020-02-03 09:30:00'
mkt_close <- '2020-02-03 16:00:00'

taq_data.raw <-
read.csv('/Users/sandeepranjan/Documents/Stevens/FE570/Final
Project/code/taq_data/OneDrive_1_11-30-2022/tsla_02_03_20.csv')
taq_data.raw[is.na(taq_data.raw)] <- 0

taq_data.raw$Date.Time <- as.POSIXct(taq_data.raw$Date.Time,
format = "%Y-%m-%dT%H:%M:%OS",tz = "GMT")
attr(taq_data.raw$Date.Time,"tzone") <- "EST" ## Convert to EST
timezone
```

```
tdata <- taq_data.raw[taq_data.raw$Type ==
'Trade',c('Date.Time','Ex.Cntrb.ID','X.RIC','Price','Volume')]

qdata <- taq_data.raw[taq_data.raw$Type ==
'Quote',c('Date.Time','Ex.Cntrb.ID','X.RIC','Bid.Price','Bid.Size','Ask.Price'
,'Ask.Size')]


## Change the column names

colnames(tdata)[colnames(tdata) == "Date.Time"] <- 'DT'

colnames(tdata)[colnames(tdata) == "Ex.Cntrb.ID"] <- 'EX'

colnames(tdata)[colnames(tdata) == "X.RIC"] <- 'SYMBOL'

colnames(tdata)[colnames(tdata) == "Price"] <- 'PRICE'

colnames(tdata)[colnames(tdata) == "Volume"] <- 'SIZE'


colnames(qdata)[colnames(qdata) == "Date.Time"] <- 'DT'

colnames(qdata)[colnames(qdata) == "X.RIC"] <- 'SYMBOL'

colnames(qdata)[colnames(qdata) == "Bid.Price"] <- 'BID'

colnames(qdata)[colnames(qdata) == "Ask.Price"] <- 'OFR'

colnames(qdata)[colnames(qdata) == "Ask.Size"] <- 'OFRSIZ'

colnames(qdata)[colnames(qdata) == "Bid.Size"] <- 'BIDSIZ'

colnames(qdata)[colnames(qdata) == "Ex.Cntrb.ID"] <- 'EX'
```

```
qdata$MIDQUOTE <- (qdata$BID + qdata$OFR)/2

########### Clean up trades data - Start #############
tdata <- as.data.table(tdata)

#ignore trades having 0 prices
tdata <- noZeroPrices(tdata)
tdata <- mergeTradesSameTimestamp(tdata)

# find # of trades grouped by Exchanges, then pick the one that has
highest # of trades
as.data.frame(table(tdata$EX))
tdata <- tdata[tdata$EX=='ADF'] ## For NASDAQ exchange

# get trades & quotes for exchange hours
#tdata <- tdata[tdata$DT >= mkt_open & tdata$DT <= mkt_close,]
tdata <- exchangeHoursOnly(tdata)
tdata <- tdata[ ! duplicated( index(tdata), fromLast = TRUE ), ]

tdata.xts <- xts(tdata[,-1],order.by=as.POSIXct(tdata$DT, format = "%Y-
%m-%d %H:%M:%OS"))
nrow(tdata)
```

```
########### Clean up trades data - End #############

########### Clean up quotes data - Start #############
qdata <- as.data.table(qdata)
#remove quotes with 0 prices
qdata <- noZeroQuotes(qdata)
qdata <- mergeQuotesSameTimestamp(qdata)

#qdata <- qdata[qdata$DT >= mkt_open & qdata$DT <= mkt_close,]
qdata <- exchangeHoursOnly(qdata)

# remove duplicates
qdata <- qdata[ ! duplicated( index(qdata), fromLast = TRUE ), ]

##convert to xts objects
qdata.xts <- xts(qdata[,-1],order.by=as.POSIXct(qdata$DT, format =
"%Y-%m-%d %H:%M:%OS"))

nrow(qdata)
########### Clean up quotes data - End #############
```

```
######################  Prepare TAQ data set
######################################

tqdata <- matchTradesQuotes(tdata, qdata)


## uniq rows

tqdata.uniq <- tqdata[ ! duplicated( index(tqdata), fromLast = TRUE ), ]


#get trade direction : -1 indicates SELL, +1 indicates BUY

tqdata$TRADE_DIRECTION  <- getTradeDirection(tqdata)

#convert to xts object

tqdata.xts <- xts(tqdata[,-1], order.by=as.POSIXct(tqdata$DT, format =
"%Y-%m-%d %H:%M:%OS"))


head(tqdata.xts)

tail(tqdata.xts)


nrow(tqdata.xts)

############################################################
#
```

```
################################################################
##
#aggregate trades and quotes every 1 second
tdata.1sec <- aggregateTrades(as.data.table(tdata), alignBy="seconds", alignPeriod=1)
qdata.1sec <- aggregateQuotes(as.data.table(qdata), alignBy="seconds", alignPeriod=1)
#remove duplicates
tdata.1sec <- tdata.1sec[ ! duplicated( index(tdata.1sec), fromLast = TRUE ), ]
qdata.1sec <- qdata.1sec[ ! duplicated( index(qdata.1sec), fromLast = TRUE ), ]
#match trades & quotes
tqdata.1sec <- matchTradesQuotes(as.data.table(tdata.1sec), as.data.table(qdata.1sec))
#convert tqdata.1sec to a data frame
tqdata.1sec.df <- as.data.frame(tqdata.1sec)
#now convert it to xts object
tqdata.1sec.xts <- as.xts(tqdata.1sec.df[,-1], order.by = as.POSIXct(tqdata.1sec.df[,1],format = "%Y-%m-%dT%H:%M:%OS",tz = "EST"))
################################################################
##
```

```
##################################################################
##
#aggregate trades and quotes every 10 second
tdata.10sec <- aggregateTrades(as.data.table(tdata),
alignBy="seconds", alignPeriod=10)
qdata.10sec <- aggregateQuotes(as.data.table(qdata),
alignBy="seconds", alignPeriod=10)
#remove duplicates
tdata.10sec <- tdata.10sec[ ! duplicated( index(tdata.10sec), fromLast =
TRUE ), ]
qdata.10sec <- qdata.10sec[ ! duplicated( index(qdata.10sec), fromLast
= TRUE ), ]
#match trades & quotes
tqdata.10sec <- matchTradesQuotes(as.data.table(tdata.10sec),
as.data.table(qdata.10sec))
#convert tqdata.1sec to a data frame
tqdata.10sec.df <- as.data.frame(tqdata.10sec)
#now convert it to xts object
tqdata.10sec.xts <- as.xts(tqdata.10sec.df[,-1], order.by =
as.POSIXct(tqdata.10sec.df[,1],format = "%Y-%m-%dT%H:%M:%OS",tz =
"EST"))
##################################################################
##
```

```
###############################################################
##

#aggregate trades and quotes every 30 second

tdata.30sec <- aggregateTrades(as.data.table(tdata),
alignBy="seconds", alignPeriod=30)

qdata.30sec <- aggregateQuotes(as.data.table(qdata),
alignBy="seconds", alignPeriod=30)

#remove duplicates

tdata.30sec <- tdata.30sec[ ! duplicated( index(tdata.30sec), fromLast =
TRUE ), ]

qdata.30sec <- qdata.30sec[ ! duplicated( index(qdata.30sec), fromLast
= TRUE ), ]

#match trades & quotes

tqdata.30sec <- matchTradesQuotes(as.data.table(tdata.30sec),
as.data.table(qdata.30sec))

#convert tqdata.1sec to a data frame

tqdata.30sec.df <- as.data.frame(tqdata.30sec)

#now convert it to xts object

tqdata.30sec.xts <- as.xts(tqdata.30sec.df[,-1], order.by =
as.POSIXct(tqdata.30sec.df[,1],format = "%Y-%m-%dT%H:%M:%OS",tz =
"EST"))

###############################################################
##
```

```
###################################################################
#aggregate trades and quotes every 1 minute
tdata.1min <- aggregateTrades(as.data.table(tdata), alignBy="minutes", alignPeriod=1)
qdata.1min <- aggregateQuotes(as.data.table(qdata), alignBy="minutes", alignPeriod=1)
#remove duplicates
tdata.1min <- tdata.1min[ ! duplicated( index(tdata.1min), fromLast = TRUE ), ]
qdata.1min <- qdata.1min[ ! duplicated( index(qdata.1min), fromLast = TRUE ), ]
#match trades & quotes
tqdata.1min <- matchTradesQuotes(as.data.table(tdata.1min), as.data.table(qdata.1min))
#convert tqdata.1sec to a data frame
tqdata.1min.df <- as.data.frame(tqdata.1min)
#now convert it to xts object
tqdata.1min.xts <- as.xts(tqdata.1min.df[,-1], order.by = as.POSIXct(tqdata.1min.df[,1],format = "%Y-%m-%dT%H:%M:%OS",tz = "EST"))
###################################################################
```

```
####################################################################
# Plot trade/Bid/Ask Prices


#Trade Prices plot between 09:30 AM - 4 PM
plot(x = index(tqdata.xts), y =
as.numeric(tqdata.xts$PRICE),col="black",type = "l",
    xlab = "Time",ylab = "Trade Prices",
    main = "TSLA - Trade prices Plot between 9.30 AM - 4 PM")


#Bid/Ask/Trade plot during 09:30 AM - 4 PM


plot(x = index(tqdata.xts), y = as.numeric(tqdata.xts$OFR),xlab =
"Time",ylab = "Bid/Ask/Trade Prices", type="l",
    col="blue", main = "Bid/Ask/Trade Plot between 9.30 AM - 4 PM")
lines(x = index(tqdata.xts), y = as.numeric(tqdata.xts$BID), col="red")
lines(x = index(tqdata.xts, y = as.numeric(tqdata.xts$PRICE), col =
"black"))
legend("bottomright",c("Ask","Bid","Trade Price"),fill =
c("blue","red","black"))
```

```
#Bid/Ofr between 3.50 PM - 4 PM

last.10min <- "2020-02-03 15:50:00/2020-02-03 16:00:00"

tqdata.xts.last10min <- tqdata.xts[last.10min]


plot(x = index(tqdata.xts.last10min), y =
as.numeric(tqdata.xts.last10min$OFR),xlab = "Time",ylab = "Bid/Ask
Prices",

    type="l",col="blue", main = "Bid/Ask Plot in the last 10 minutes of
trading day")

lines(x = index(tqdata.xts.last10min), y =
as.numeric(tqdata.xts.last10min$BID), col="red")

legend("topleft",c("Ask","Bid"),fill = c("blue","red"))


# Trading volume plot

plot(x = index(tqdata.xts), y =
as.numeric(tqdata.xts$SIZE),col="black",type = "l",

    xlab = "Time",ylab = "Trade Size",

    main = "TSLA - Trading Volume Plot between 9.30 AM - 4 PM")


##############################################################
####


#### @@ ###
```

##

Explore data

(ii) Plot trade Price with best bid and best ask for entire data set

```
plot( tqdata$DT,as.numeric(tqdata$PRICE),xlab = "",ylab = "Trd/Ofr/Bid Prices", type="l",
    col="yellow")
lines(as.numeric(tqdata$OFR), col="red")
lines(as.numeric(tqdata$BID), col="blue")
legend("topright",c("Trade","Ofr","Bid"),fill = c("yellow","red","blue"))

# (iii) Plot trade Price with best bid and best ask for rows with counts 1 : 500
df_1min <- data.frame(tqdata[tqdata$DT >= '2020-02-03 10:00:00' & tqdata$DT <= '2020-02-03 10:01:00',])

plot(c(1:500),df_1min$PRICE[1:500],xlab = "", ylab = "Trd/Ofr/Bid Prices",type="p",
    col="black",main = "First 500 trades between 10 - 10:01 AM")
```

```
lines(c(1:500),df_1min$OFR[1:500], col="red")

lines(c(1:500),df_1min$BID[1:500], col="blue")

legend("bottomright",c("Trade","Ofr","Bid"),fill =
c("black","red","blue"))




# 2. Count how many trades take place: i) within the spread, ii) at bid,
iii) at ask


n.trades <- nrow(tqdata)

df_trd_within_spread <- subset(tqdata, PRICE > BID & PRICE < OFR)

df_trd_at_bid <- subset(tqdata,PRICE==BID)

df_trd_at_ask <- subset(tqdata,PRICE==OFR)


df_trd_outside_spread <- subset(tqdata, PRICE < BID | PRICE > OFR)


n.trades_within_spread <- nrow(df_trd_within_spread)

n.trades_at_bid <- nrow(df_trd_at_bid)

n.trades_at_ask <- nrow(df_trd_at_ask)


n.trades_outside_spread <- nrow(df_trd_outside_spread)
```

```
n.trades_within_spread

n.trades_at_bid

n.trades_at_ask


n.trades_outside_spread


n.trades.stats <-
data.frame(n.trades_within_spread,n.trades_at_bid,n.trades_at_ask,n.
trades_outside_spread)

n.trades.stats


pct.trades.stats <-
data.frame((n.trades_within_spread/n.trades)*100,(n.trades_at_bid/n.
trades)*100,


(n.trades_at_ask/n.trades)*100,(n.trades_outside_spread/n.trades)*10
0)

colnames(pct.trades.stats) <-
c("%_trades_within_spread","%_trades_at_bid","%_trades_at_ask","%
_trades_outside_spread")

pct.trades.stats

### @@ ###
```

```
################################################################
###
## Liquidity - Calculate Spread Measures
#Use the getLiquidityMeasures function in the highfrequency package

spread_measures <- data.frame(row.names =
c('Quoted_spread','Effective_Spread','Realized_Spread'))

## spread measures for tick level data
liquidity_measures <- getLiquidityMeasures(tqdata,win = 300)
liquidity_measures[is.na(liquidity_measures)] <- 0

quoted_spread <- mean(liquidity_measures$quotedSpread)
eff_spread <- mean(liquidity_measures$effectiveSpread)
realized_spread <- mean(liquidity_measures$realizedSpread)

spread_measures$tickData <- c(quoted_spread, eff_spread,
realized_spread)

################
# Plot the effective, quoted and realized spreads for tick data
```

```
liquidity_measures.df <- as.data.frame(liquidity_measures)

liquidity_measures.xts <- as.xts(liquidity_measures.df[,-1], order.by =
as.POSIXct(liquidity_measures.df[,1],format = "%Y-%m-
%dT%H:%M:%OS",tz = "EST"))


# TODO

# plot(x = index(liquidity_measures.xts), y =
as.numeric(liquidity_measures.xts$quotedSpread),xlab = "Time",ylab =
"Quoted/Eff/Realized Spread",

#     type="l",col="blue", main = "Quoted/Eff/Realized Spread for tick
data")

# lines(x = index(liquidity_measures.xts), y =
as.numeric(liquidity_measures.xts$effectiveSpread), col="red")

# lines(x = index(liquidity_measures.xts), y =
as.numeric(liquidity_measures.xts$realizedSpread), col="green")

# legend("topright",c("Quoted","Effective","Realized"),fill =
c("blue","red","green"))


################


## spread measures for 1 sec data

liquidity_measures <- getLiquidityMeasures(tqdata.1sec,win = 300)

liquidity_measures[is.na(liquidity_measures)] <- 0
```

```
quoted_spread <- mean(liquidity_measures$quotedSpread)
eff_spread <- mean(liquidity_measures$effectiveSpread)
realized_spread <- mean(liquidity_measures$realizedSpread)

spread_measures$tqdata.1sec <- c(quoted_spread, eff_spread,
realized_spread)

## spread measures for 10 sec data
liquidity_measures <- getLiquidityMeasures(tqdata.10sec,win = 300)
liquidity_measures[is.na(liquidity_measures)] <- 0

quoted_spread <- mean(liquidity_measures$quotedSpread)
eff_spread <- mean(liquidity_measures$effectiveSpread)
realized_spread <- mean(liquidity_measures$realizedSpread)

spread_measures$tqdata.10sec <- c(quoted_spread, eff_spread,
realized_spread)

## spread measures for 30 sec data
liquidity_measures <- getLiquidityMeasures(tqdata.30sec,win = 300)
liquidity_measures[is.na(liquidity_measures)] <- 0
```

```
quoted_spread <- mean(liquidity_measures$quotedSpread)
eff_spread <- mean(liquidity_measures$effectiveSpread)
realized_spread <- mean(liquidity_measures$realizedSpread)

spread_measures$tqdata.30sec <- c(quoted_spread, eff_spread,
realized_spread)

## spread measures for 1 min of data
liquidity_measures <- getLiquidityMeasures(tqdata.1min,win = 300)
liquidity_measures[is.na(liquidity_measures)] <- 0

quoted_spread <- mean(liquidity_measures$quotedSpread)
eff_spread <- mean(liquidity_measures$effectiveSpread)
realized_spread <- mean(liquidity_measures$realizedSpread)

spread_measures$tqdata.1min <- c(quoted_spread, eff_spread,
realized_spread)

spread_measures

#Plot Spreads for tick level data
liquidity_measures.tick <- getLiquidityMeasures(tqdata,win = 300)
```

```
liquidity_measures.tick[is.na(liquidity_measures.tick)] <- 0

plot(liquidity_measures.tick$DT,
liquidity_measures.tick$quotedSpread,
    type='l', main = "Quoted Spread",xlab = "Time",ylab = "Quoted
Spread")

plot(liquidity_measures.tick$DT,
liquidity_measures.tick$effectiveSpread,
    type='l', main = "Effective Spread",xlab = "Time",ylab = "Effective
Spread",col='blue')

###############################################################
####

###############################################################
####
### Statistics of price returns

#no. of observations
length(diff(tqdata$PRICE))
length(diff(tqdata.1sec$PRICE))
length(diff(tqdata.10sec$PRICE))
```

```
length(diff(tqdata.30sec$PRICE))
length(diff(tqdata.1min$PRICE))

#min/max range of price differences for trades
range(diff(tqdata$PRICE))
range(diff(tqdata.1sec$PRICE))
range(diff(tqdata.10sec$PRICE))
range(diff(tqdata.30sec$PRICE))
range(diff(tqdata.1min$PRICE))

#Calculate Mean of price differences for trades
PerformanceAnalytics::Mean.arithmetic(diff(tqdata$PRICE))
PerformanceAnalytics::Mean.arithmetic(diff(tqdata.1sec$PRICE))
PerformanceAnalytics::Mean.arithmetic(diff(tqdata.10sec$PRICE))
PerformanceAnalytics::Mean.arithmetic(diff(tqdata.30sec$PRICE))
PerformanceAnalytics::Mean.arithmetic(diff(tqdata.1min$PRICE))

#Calculate Std. dev. of price differences for trades
PerformanceAnalytics::StdDev(diff(tqdata$PRICE))
PerformanceAnalytics::StdDev(diff(tqdata.1sec$PRICE))
PerformanceAnalytics::StdDev(diff(tqdata.10sec$PRICE))
```

```
PerformanceAnalytics::StdDev(diff(tqdata.30sec$PRICE))

PerformanceAnalytics::StdDev(diff(tqdata.1min$PRICE))


#Calculate kurtosis of price differences for trades

PerformanceAnalytics::kurtosis(diff(tqdata$PRICE))

PerformanceAnalytics::kurtosis(diff(tqdata.1sec$PRICE))

PerformanceAnalytics::kurtosis(diff(tqdata.10sec$PRICE))

PerformanceAnalytics::kurtosis(diff(tqdata.30sec$PRICE))

PerformanceAnalytics::kurtosis(diff(tqdata.1min$PRICE))

################################################################
####



# Auto correlation of trade price differences

acf.tick <- acf(diff(tqdata$PRICE), main = "ACF of price diff. at tick
level")

acf.1sec <- acf(diff(tqdata.1sec$PRICE), main = "ACF of price diff. at 1
sec aggregation")

acf.10sec <- acf(diff(tqdata.10sec$PRICE), main = "ACF of price diff. at
10 sec aggregation")

acf.30sec <- acf(diff(tqdata.30sec$PRICE), main = "ACF of price diff. at
30 sec aggregation")
```

```
acf.1min <- acf(diff(tqdata.1min$PRICE),  main = "ACF of price diff. at 1 min aggregation")


#Auto correlation  of log returns
log.p.tick <- log(as.numeric(tqdata.xts$PRICE))
d.log.p.tick <- diff(log.p.tick)

acf.ret <- acf(d.log.p.tick,main="ACF  of the tick level log-returns")



log.p <- log(as.numeric(tqdata.1sec.xts$PRICE))
d.log.p <- diff(log.p)

acf.ret <- acf(d.log.p,main="ACF  of the 1 sec log-returns")

## ACF of price diff.
ret.tsla <- diff(tqdata$PRICE)
ret.tsla

ret.tsla <- ret.tsla[!is.na(ret.tsla)]   # Remove missing values
ret.tsla <- ret.tsla  - mean(ret.tsla)
acf.ret <- acf(ret.tsla,main="ACF  of the Price returns")
```

```
ret.tsla.30sec <- log(tqdata.1min$PRICE)/lag(tqdata.1min$PRICE)
ret.tsla.30sec <- ret.tsla.30sec[!is.na(ret.tsla.30sec)]  # Remove missing values
acf.ret.30sec <- acf(ret.tsla.30sec)




##############################################
## Volatility Estimation
##############################################

## Method 1 : Using Trades Prices Sampled at lag q

p <- as.numeric(tqdata$PRICE)
realizedVar <- function(q){rCov(diff(p, lag=q, differences=1))/q}

# vol. at lag 1
realized_Vol1 <- sqrt(realizedVar(1))
realized_Vol1
```

```r
# vol. at lag 2
realized_Vol2  <- sqrt(realizedVar(2))
realized_Vol2

# vol. at lag 5
realized_Vol5  <- sqrt(realizedVar(5))
realized_Vol5

# vol. at lag 50
realized_Vol50  <- sqrt(realizedVar(50))
realized_Vol50

# vol. at lag 100
realized_Vol100  <- sqrt(realizedVar(100))
realized_Vol100

# vol. at lag 500
realized_Vol500  <- sqrt(realizedVar(500))
realized_Vol500
```

```
rv <-
data.frame(realized_Vol1,realized_Vol2,realized_Vol5,realized_Vol50,r
ealized_Vol100,realized_Vol500)

rv


## Signature plot


rv_data <- NULL

for(q in 1:500){

  rv_data <- c(rv_data, realizedVar(q))


}


plot(sqrt(rv_data), type ="l", main="Signature plot for TSLA:
02/03/2020")


q5min <- n.trades*5/390


rv5 = realizedVar(q5min)


## Method 2 : Roll Model estimate of Volatility
```

```
dp = diff(p)

# compute the covariance of the price changes, for the Roll model analysis
covdp <- acf(dp, lag.max=10,
          type="covariance", plot=TRUE,
          main="Autocovariance of price changes")

gamma0 <- covdp$acf[1]
gamma1 <- covdp$acf[2]

sig2u = gamma0 + 2*gamma1

rvRoll <- sig2u*n.trades

sigRoll <- sqrt(sig2u*n.trades)

plot(sqrt(rv_data),  type ="l",
    main="Signature plot for prices + Roll",col = "black"
    )
abline(h=sqrt(rv5),col="red")
abline(h=sigRoll,col="blue")
```

```
legend("topright",c("Realized Volatility","Realized Vol. at 5 min Sampling","Roll Model Vol."),fill = c("black","red","blue"))

tradeSigns <- getTradeDirection(tqdata)

acf(tradeSigns,main = "Correlation(trade signs) of TSLA", type = "correlation")

######################### Estimate PIN ######################################

pin_stats <- data.frame(matrix(ncol = 8, nrow = 0))

colnames(pin_stats) <- c('Method','Factorization','PIN','alpha','delta','mu','epsilon-b','epsilon-s')

# count B/S events

x <- getTradeDirection(tqdata)

tradeDirection <- matrix(x)

buy_side <- which(tradeDirection >0)
```

```
num_buy_side <- length(matrix(buy_side))

num_sell_side <- length(tradeDirection) - length(matrix(buy_side))


## group by 1 min time interval and find the # of buys and sells in each of those intervals

buy_sell_count <- tqdata %>%

  mutate(time_interval = cut(DT,seq(from = as.POSIXct("2020-02-03 09:30:00",tz="EST"),

                              to = as.POSIXct("2020-02-03 16:00:00",tz = "EST"),by = "1 min"))) %>%

  group_by(time_interval,TRADE_DIRECTION) %>%

  dplyr::summarize(count = length(TRADE_DIRECTION)) %>%

  as.data.frame()


head(buy_sell_count,15)


buys <- buy_sell_count[buy_sell_count$TRADE_DIRECTION==1,c("count")][1:50]

sells <- buy_sell_count[buy_sell_count$TRADE_DIRECTION==-1,c("count")][1:50]
```

```
# Initial parameter values
# par0 = (alpha, delta, mu, epsilon_b, epsilon_s)
par0 = c(0.5,0.5,300,400,500)
options(warn = -1)
data_buy_sell = cbind(buys,sells)
LK_out = LK(data_buy_sell)

model = optim(par0, LK_out, gr = NULL,method = c("Nelder-Mead"),
hessian = FALSE)

## Parameter Estimates
alpha <- model$par[1] # Estimate for alpha
delta <- model$par[2] # Estimate for delta
mu <- model$par[3] # Estimate for mu
eb <- model$par[4] # Estimate for eb
es <- model$par[5] # Estimate for es

## Estimate for PIN
pin <- (alpha * mu)/(alpha*mu + eb + es)
#(model$par[1]*model$par[3])/((model$par[1]*model$par[3])+model$
par[4]+model$par[5])
pin
```

```
parameter_values <- data.frame(alpha,delta,mu,eb,es)
parameter_values

### using EHO factorization method

EHO_out = EHO(data_buy_sell)
par0 = c(0.5,0.5,200,400,500)

model = optim(par0, EHO_out, gr = NULL,method = c("Nelder-Mead"),
hessian = FALSE)

## Parameter Estimates
alpha <- model$par[1] # Estimate for alpha
delta <- model$par[2] # Estimate for delta
mu <- model$par[3] # Estimate for mu
eb <- model$par[4] # Estimate for eb
es <- model$par[5] # Estimate for es


## Estimate for PIN
pin <- (alpha * mu)/(alpha*mu + eb + es)
```

```
#(model$par[1]*model$par[3])/((model$par[1]*model$par[3])+model$
par[4]+model$par[5])

pin


parameter_values  <- data.frame(alpha,delta,mu,eb,es)

parameter_values


### Using YZ, GAN, EA algorithms

result <- YZ(data_buy_sell,likelihood  = 'LK')

pin_stats[nrow(pin_stats)+1,]  <-
c('YZ','LK',result$PIN,result$alpha,result$delta,

                              result$mu,result$epsilon_b,result$epsilon_s)


result <- YZ(data_buy_sell,likelihood  = 'EHO')

pin_stats[nrow(pin_stats)+1,]  <-
c('YZ','EHO',result$PIN,result$alpha,result$delta,

                              result$mu,result$epsilon_b,result$epsilon_s)


## GAN

result <- GAN(data_buy_sell,likelihood  = 'LK')

pin_stats[nrow(pin_stats)+1,]  <-
c('GAN','LK',result$PIN,result$alpha,result$delta,

                              result$mu,result$epsilon_b,result$epsilon_s)
```

```
result <- GAN(data_buy_sell,likelihood  = 'EHO')

pin_stats[nrow(pin_stats)+1,]  <-
c('GAN','EHO',result$PIN,result$alpha,result$delta,

                          result$mu,result$epsilon_b,result$epsilon_s)

## EA

result <- EA(data_buy_sell,likelihood  = 'LK')

pin_stats[nrow(pin_stats)+1,]  <-
c('EA','LK',result$PIN,result$alpha,result$delta,

                          result$mu,result$epsilon_b,result$epsilon_s)


result <- EA(data_buy_sell,likelihood  = 'EHO')

pin_stats[nrow(pin_stats)+1,]  <-
c('EA','EHO',result$PIN,result$alpha,result$delta,

                          result$mu,result$epsilon_b,result$epsilon_s)


pin_stats


############################################################
##################
```