final

July 9, 2019

1 Algo trading :execute order now or rest this order within the Bid/Offer spread for a period time ?

Kefei Liu (kl3295) Jiangguangyu Xue (jx1021)

Goal: The purpose of this project is to find out if an order should be immediately executed by paying (half) the Bid/Offer spread or it would be economically better to rest this order withing the Bid/Offer spread for a period time to achieve possibly a better execution.

2 1. Introduction

In this project,we test different execution approaches to find the best strategy in acquiring best Excution PnL. There are three approaches: - **Market Taking (MT)** - **Opportunity Market Taking Mid(OMMMid)** - **Opportunity Market Taking Side (OMMSide)**

MT here is serving like a baseline model, a benchmark that always get negative half of ask-bid spread, While OMMMid's and OMMSide's execution PnL will vary according to market flow. We also try to explore the relations between several parameters in the model and execution PnL, which includes **max Time to Execution (TTE)** and **max Stop Loss (SL)**. What's more, we are eager to find out whether there will be difference in our strategy selection in term of **different alpha engine**.

3 2. Getting data and having a brief look

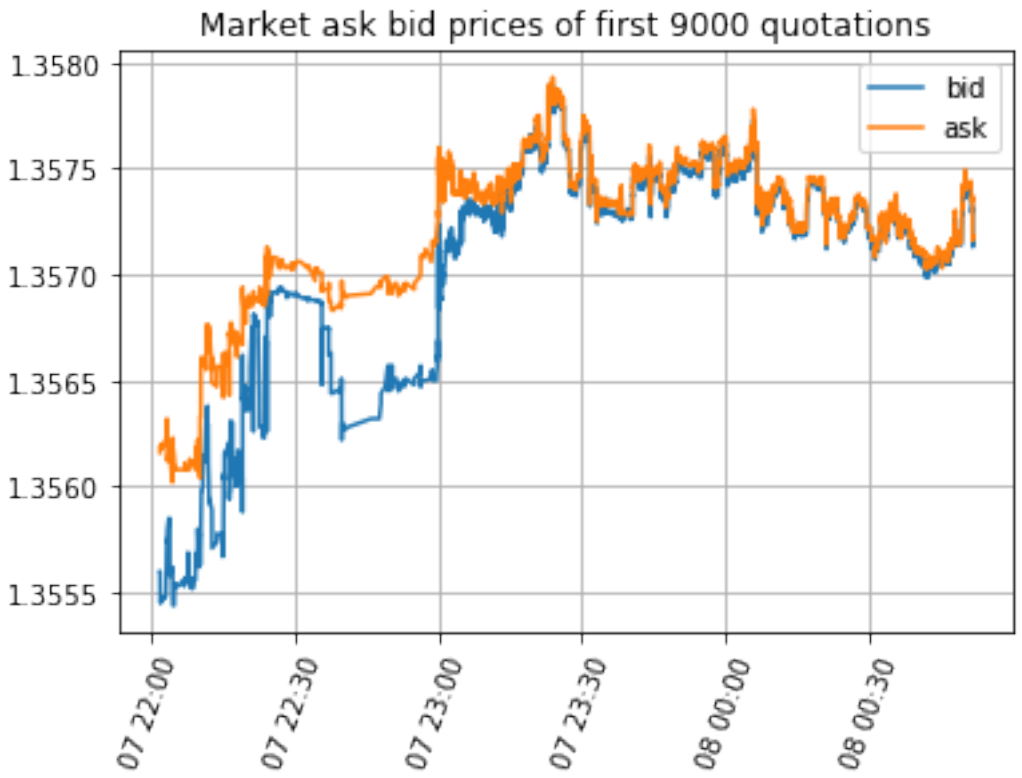
We are given two CSV Sheets to start with. After we import the these two files, let's have a brief look at the ask-bid data and the order data.

3.0.1 1) Market ask and bid prices

For **Market Ask Bid** time series, we want to see the spread in different time periods. However, the data set is quite large, we will just plot the first 9000 quotations of the market to have a brief understanding about the spread and ask-bid prices.

```
In [37]: figure_ask_bid
```

```
Out[37]:
```

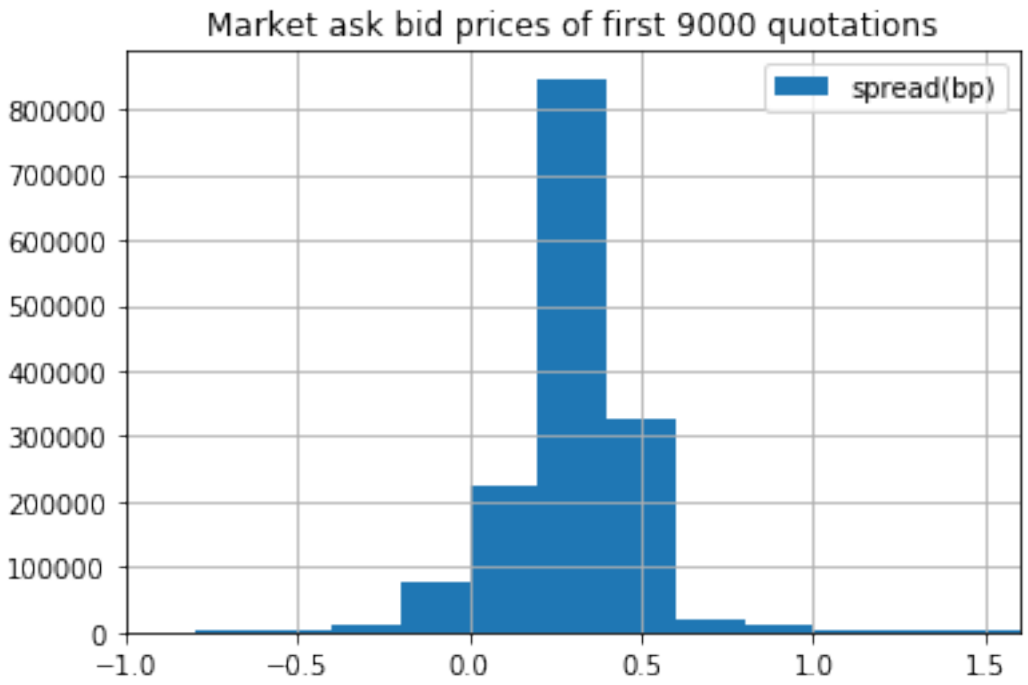


We can find from the graph above that the spread is quite wide during mid night but siginificantly narrow in the morning.

Also, we have a look at the **spread distribution** before implement the strategies.

```
In [38]: figure_spread
```

```
Out[38]:
```



Suprisingly, we find some of the spreads even nagetive. The median of spread is less than half of a bp and around 0.4bp. These information can be very helpful when we implement the model later, since we need to setup approperiate max Stop Loss (SL) for the model.

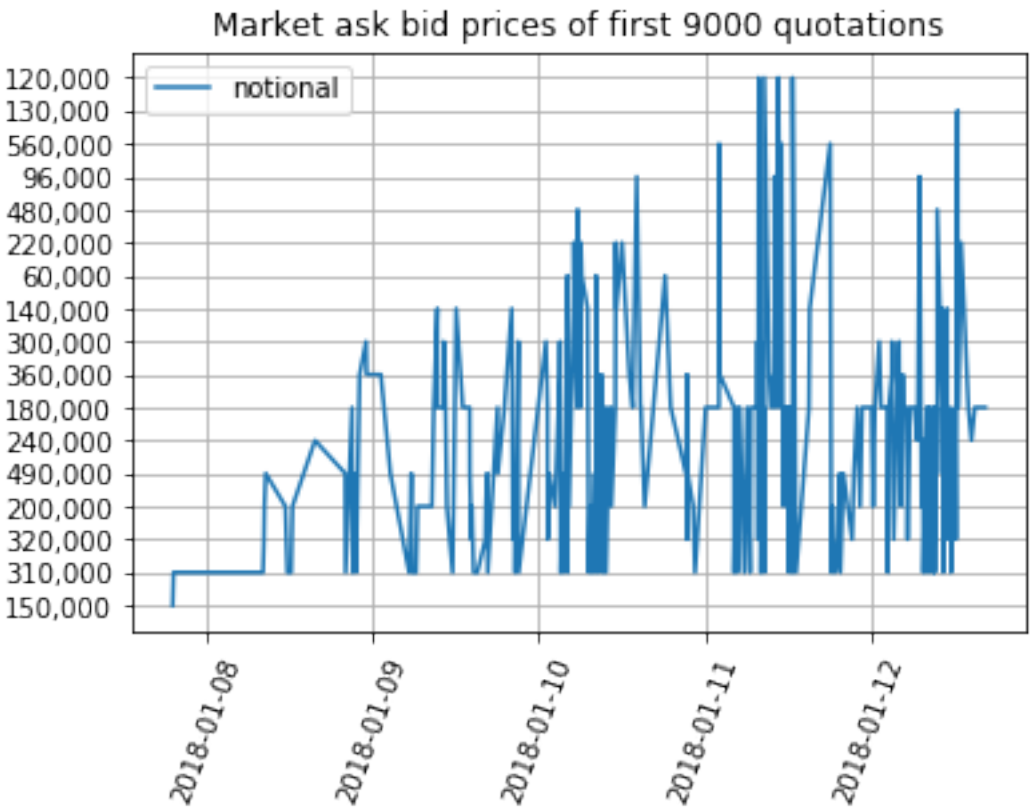
3.0.2 2) order time and buy/sell numbers

We look at the number of buy or sell orders and also the notional of the order in different time.

In terms of the order types, we have 208 sell orders and 162 buy orders. In other words, the sell orders are more than buy orders.

```
In [39]: figure_notional
```

```
Out[39]:
```



What's more, there are more orders in the last three days and the volume is going up as time goes by.

4 3. Comparison of three different strategies

At first, we initialize our parameters as following: max Time to Execution(TTE) is 10 seconds, max Stop Loss(SL) is 0.0003.

4.0.1 1) Mean, median and min of three strategies

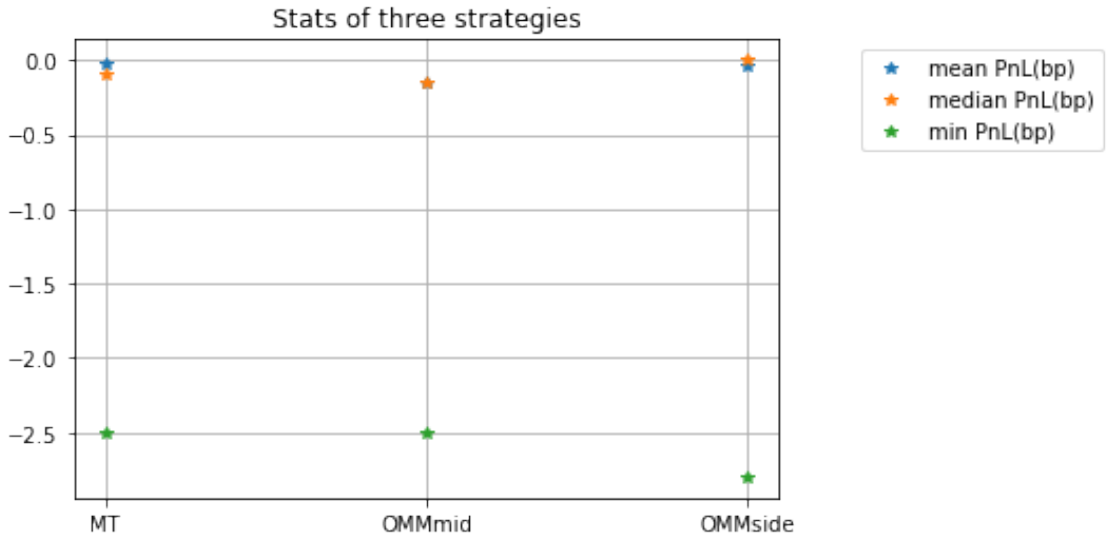
After calculation of execution times and prices, we can see that for all approaches,we lost money because mean PnLs are negative. OMMmid and OMMside lose less than half of the start spread, but OMMmid and OMMside strategies work worse than MT strategy.

```
In [40]: stats_.iloc[:,0:3].round(3)
```

```
Out[40]:           mean PnL(bp)  median PnL(bp)  min PnL(bp)
         MT              -0.021           -0.10         -2.5
         OMMmid          -0.150           -0.15         -2.5
         OMMside         -0.037           -0.00         -2.8
```

```
In [41]: fig1
```

Out[41]:



What's more, we can see that OMMmid approach is the worst strategy and all strategies will loss money with all order executed .

4.0.2 2) Sum of PnL, triggered condition of three strategies

We list the sum of PnL of the three strategies and also dive into analyze when the orders are excuted.

```
In [42]: stats
```

```
Out[42]:           sum PnL(bp)
         MT              -7.65
         OMMmid         -55.50
         OMMside        -13.65
```

In OMMmid approach, orders marked as SL triggered are most, which means most orders are executed at a worse price than start mid price and thus loss money. In OMMside approach, most orders are executed at SL and 116 orders are executed at end meet price which is less than that of OMMmid because end_meet_price trigger has a larger interval compared to OMMmid approach.

```
In [43]: stop_type_number_OMMmid
```

```
Out[43]: SL triggered      174
         end_meet_price    146
         TTE triggered      49
         Name: End condition, dtype: int64
```

```
In [44]: stop_type_number_OMMside
```

```
Out[44]: SL triggered      183
         end_meet_price    116
         TTE triggered      70
         Name: End condition, dtype: int64
```

4.0.3 3) Sensitivity to TTE

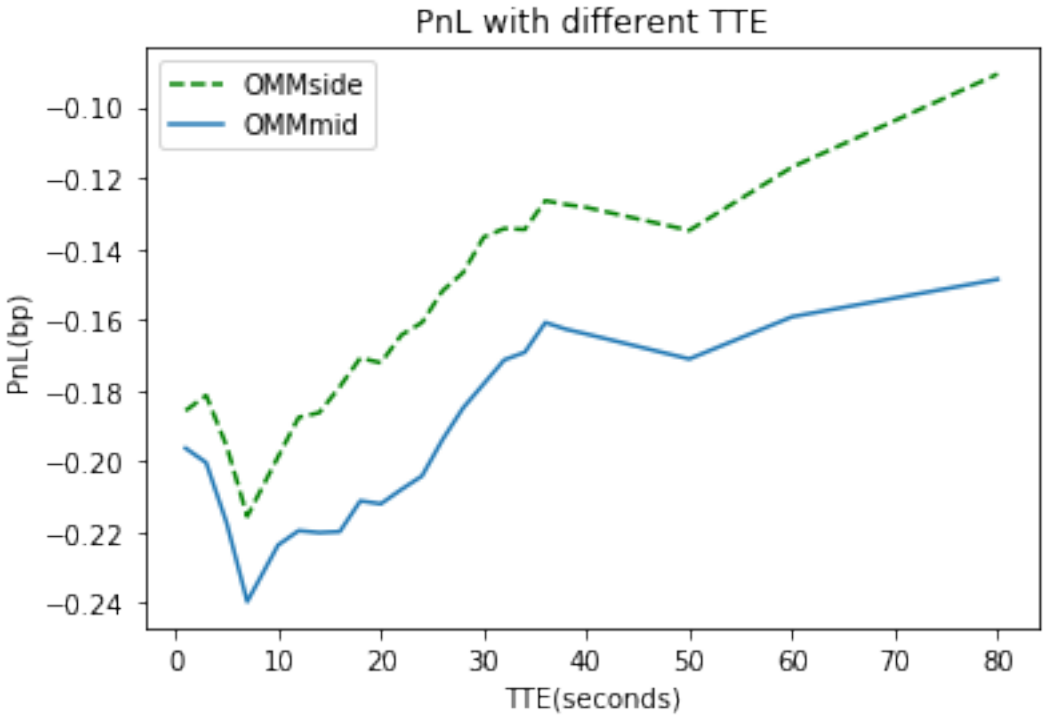
From the anlysis above, we can see that TTE might be small for this case. We want to give a series of TTE to see how the two stratgies react to the change of TTE.

When max Stop Loss(SL) is stable(0.005)(we pick a large Stop Loss, so we can solely focus on TTE), as TTE increasing, the means of both strategies are going up.OMMside is a better strategy when we give a TTE that is long enough because it can earn more money than OMMmid strategy.

What's more, OMMside and OMMmid have about the same sensitivity to TTE.

```
In [74]: fig2
```

Out[74]:



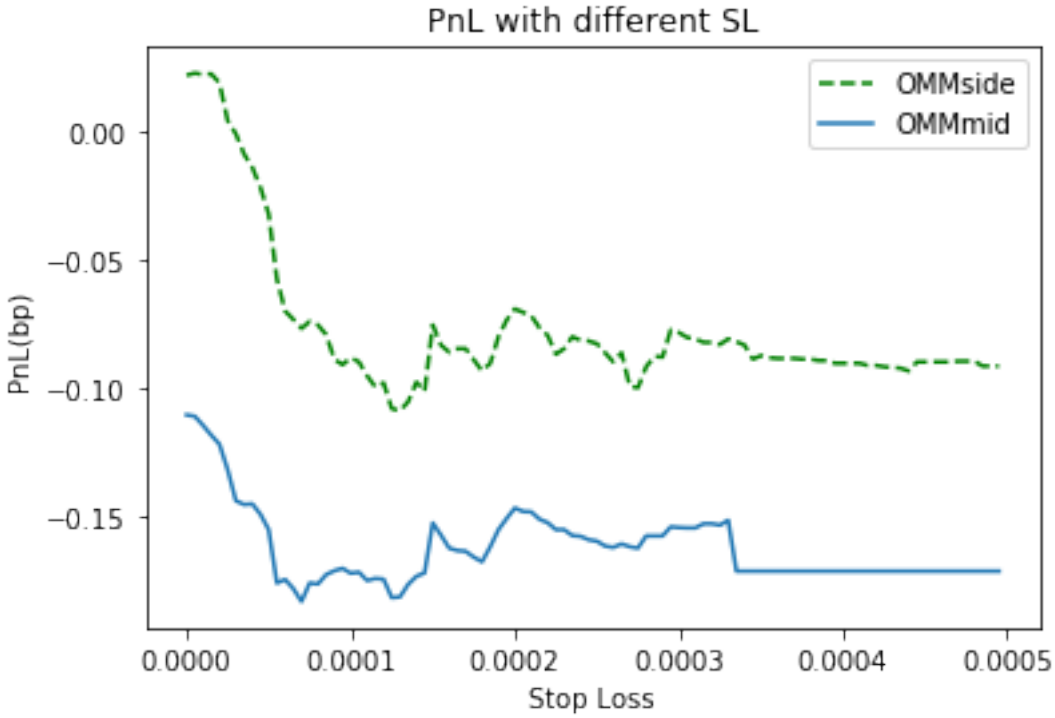
For a safety choice, we will pick `TTE = 50` here.

4.0.4 4) Sensitivity to Stop Loss

We picked TTE = 50 seconds, and try to find the relations between Stop Loss and PnL.

```
In [79]: fig4
```

Out[79]:



We can again find from graph above that OMMSide beats OMMMid all time and execution PnL goes down as SL goes up. And if we pick SL=0.00002, OMMside will beat MT(MT has a mean PnL of -0.02). OMMMid will never beat MT.

However, when we are doing this, we might have `overfitting risk`.

4.0.5 5) Different alpha engines

We can see from the above anlysis that OMMside is a better choice than OMMmid with TTE = 50 and SL = 0.00002. We will go on our exploration about the aphla engines with these parameters.

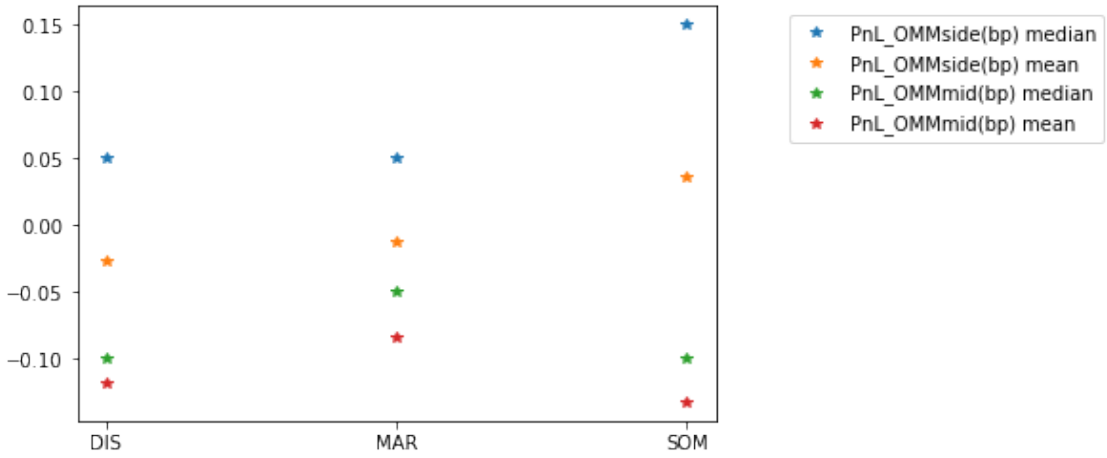
In [96]:
```
alpha_.round(4)
```

Out[96]:

Alpha	PnL_OMMside(bp) median	PnL_OMMside(bp) mean	PnL_OMMmid(bp) median	PnL_OMMmid(bp) mean
DIS	0.05	-0.0267	-0.10	-0.1175
MAR	0.05	-0.0127	-0.05	-0.0831
SOM	0.15	0.0356	-0.10	-0.1320

```
In [97]: fig3
```

Out[97]:



As we can see from the graphes above: - For SOM, OMMSide is abtter choice and it can even bring positive execution PnL. - For MAR, mean PnL of OMMSide(-0.0127) is slightly MT(-0.0207), but median is much better. We will go for OMMSide either. - For DIS, mean PnL of OMMSide is worse than MT. So we will use MT here without taking risk.

5 4. Conclusion

1. OMMMid performs worse than the other two strategies given any TTE and SL.
2. If we pick TTE aroungd 50 seconds and SL around 2bp, OMMSide will beat MT in both mean and median of PnL.
3. With approperiate SL and TTE in last conclusion, we recommend OMMSide for MAR and SOM and MT for DIS.

6 5. Coding part

Below is our coding part, there are some comments and explainations step by step in our coding part, which introduces how excatly we design our algorithms in details.

7 Data input

```
In [56]: import pandas as pd
         import os
         import sys
         import matplotlib.pyplot as plt
         import numpy as np
         import datetime
```

```
import time
import math
```

In [57]:
```
ba=pd.read_csv('2.csv',encoding='utf-16-le',index_col=0)#read csv in utf-16
ba.head()
```

Out[57]:

DateTime	Bid	Ask
01/07/2018 22:02:03.832	1.35560	1.35616
01/07/2018 22:02:03.845	1.35560	1.35617
01/07/2018 22:02:03.851	1.35554	1.35617
01/07/2018 22:02:03.859	1.35554	1.35618
01/07/2018 22:02:04.114	1.35554	1.35619

In [58]:
```
order = pd.read_csv("Assignment#3_Orders.csv")
order.head()
```

Out[58]:

	trade_id	buy/sell	notional	executed_price	order_datetime	instrument	alpha
0	107414	1	150,000	1.35724	1/7/2018 19:07:49.749	GBP/USD	SOM
1	107427	1	310,000	1.35725	1/7/2018 19:15:23.1523	GBP/USD	SOM
2	107638	-1	310,000	1.35453	1/8/2018 7:38:48.3848	GBP/USD	SOM
3	107649	-1	310,000	1.35371	1/8/2018 8:10:55.1055	GBP/USD	SOM
4	107654	-1	320,000	1.35394	1/8/2018 8:19:10.1910	GBP/USD	SOM

8 Approaches introduction

First of all, we import price-time data and data of orders and decide the execution times of each order.

In market taking approach, we agress the market immediately and therefore incur the (half) Bid/Offer spread.

In Opportunistic Market Making approach, we have two strategies. The first is OMMSide, in this part, we go through every quotations after the order time in the following 10 seconds(max Time to Execution)for each order. Then we decide whether the order should be executed at each step following the rule below:

1)For 'buy' order:

If ask price at the step is smaller than or equals to bid price at the start time, we execute the order and mark it as "end_meet_price".

If ask price at the step minus start mid price is larger than max stop loss, we should execute to aviod larger loss and mark the order as "SL triggered".

If the order is not executed until the end time, we execute it at the end and mark the order as "TTE triggered". All orders are executed at the ask price.

2)For 'sell' order:

If bid price at the step is bigger than or equals to ask price at the start time, we execute the order and mark it as "end_meet_price".

If start mid price minus bid price is larger than max stop loss, we should execute to aviod larger loss and mark the order as "SL triggered".

If the order is not executed until the end time, we execute it at the end and mark the order as "TTE triggered". All orders are executed at the bid price.

The second is OMMMid strategy.In this part, we go through every quotations after the order time in the following 10 seconds(max Time to Execution)for each order.Then we decide whether the order should be executed at each step following the rule below:

1)For 'buy' order:

If ask price at the step is smaller than or equals to start mid price, we execute the order at the ask price and mark it as "end_meet_price".

If start mid price minus ask price at the step is larger than max stop loss, we should execute at the mid price immediately to aviod larger loss and mark the order as "SL triggered".

If the order is not executed until the end time, we execute it at the end ask price and mark the order as "TTE triggered".

2)For 'sell' order:

If bid price at the step is bigger than or equals to start mid price at the start time, we execute the order at the bid price and mark it as "end_meet_price".

If bid price at the step minus start mid price is larger than max stop loss, we should execute at the mid price immediately to aviod larger loss and mark the order as "SL triggered".

If the order is not executed until the end time, we execute it at the end bid price and mark the order as "TTE triggered".

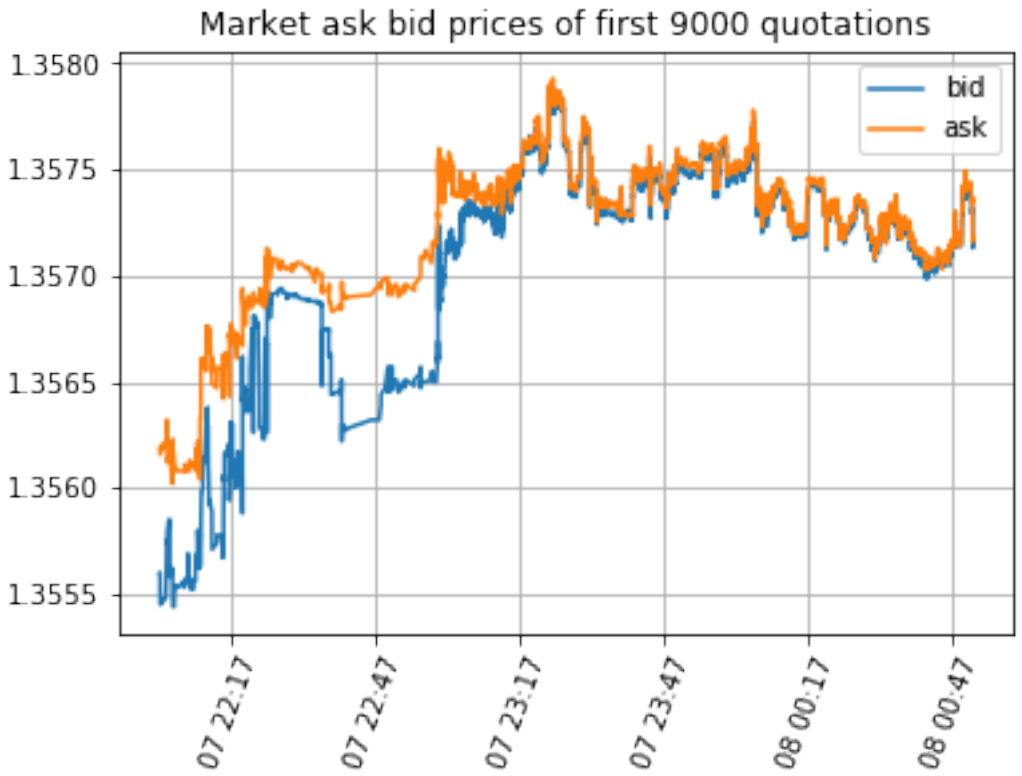
8.1 1)Market Taking (MT)

For each order, you will agress the market immediately and therefore incur the (half) Bid/Offer spread.

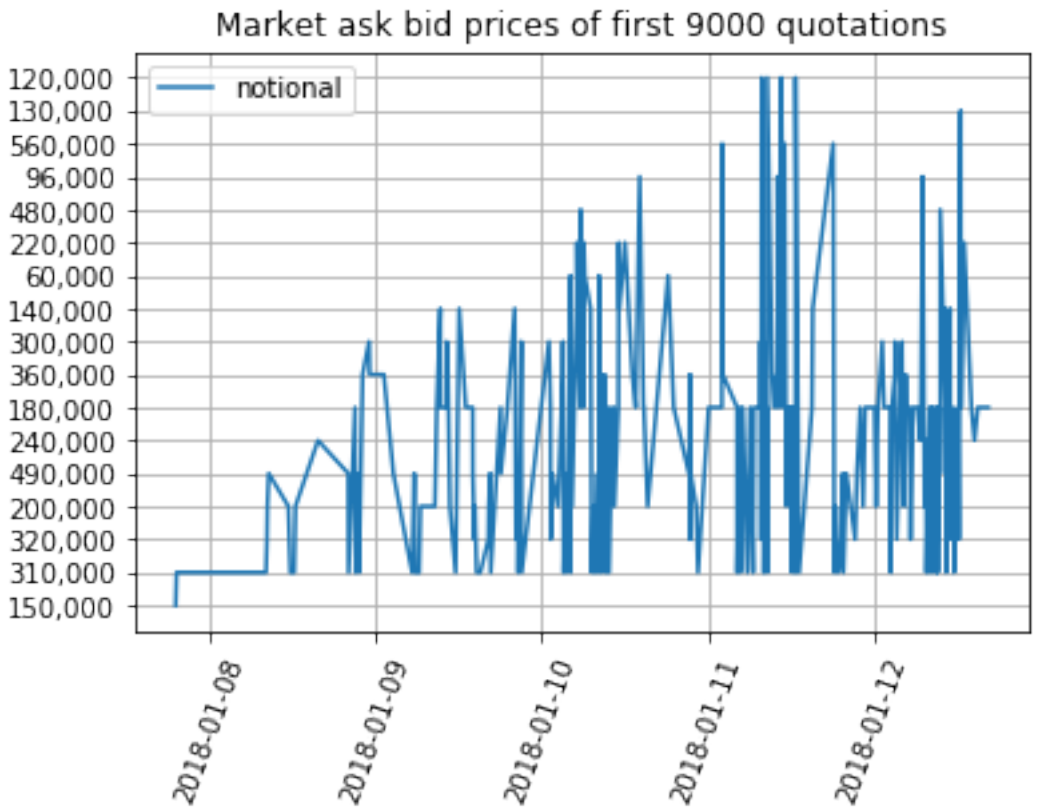
```
In [59]: ba.index=pd.to_datetime(ba.index,format='%m/%d/%Y %H:%M:%S.%f')# change time in '%m/%
         order_timestamp=pd.to_datetime(order['order_datetime'],format='%m/%d/%Y %H:%M:%S.%f')
         order_timestamp.head()

Out[59]: 0   2018-01-07 19:07:49.749000
         1   2018-01-07 19:15:23.152300
         2   2018-01-08 07:38:48.384800
         3   2018-01-08 08:10:55.105500
         4   2018-01-08 08:19:10.191000
         Name: order_datetime, dtype: datetime64[ns]

In [60]: figure_ask_bid = plt.figure()
         plt.plot(ba[0:9000])
         plt.title("Market ask bid prices of first 9000 quotations")
         plt.legend(["bid","ask"])
         plt.xticks(rotation = 70)
```

```
plt.grid()
plt.show
```

Out[60]: <function matplotlib.pyplot.show(*args, **kw)>



In [61]:
```
figure_notional = plt.figure()
plt.plot(np.array(order_timestamp),np.array(order["notional"]))
plt.title("Market ask bid prices of first 9000 quotations")
plt.legend(["notional"])
plt.xticks(rotation = 70)
plt.grid()
_ = plt.show
```




```
In [62]: def find_pos(order_timestamp,ba):    #given order time and the trading time, find the
             out=[]
             for i,time in enumerate(order_timestamp):
                 temp = ba.index.get_loc(time,method ='backfill')
                 out.append(temp)
             return out
```

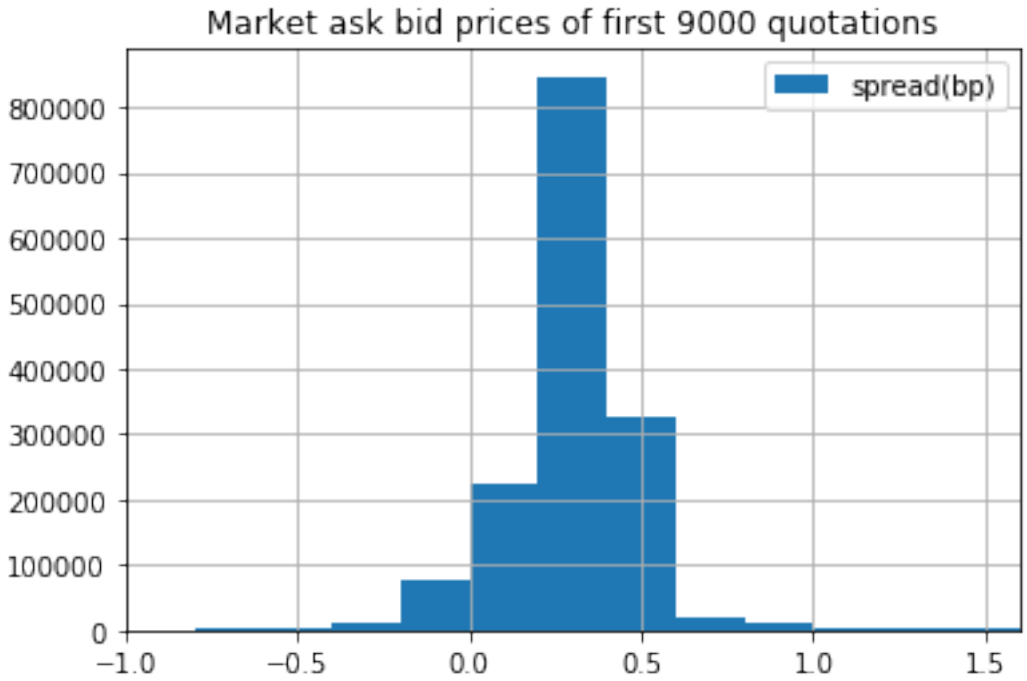
```
In [63]: start_pos = find_pos(order_timestamp,ba)
         h_spread =(ba["Ask"]-ba["Bid"])/2
         PnL_MT =pd.DataFrame({"Excution Time":h_spread[start_pos].index,"PnL_MT(bp)":order["bu
         PnL_MT.set_index(["Excution Time"], inplace=True)
         PnL_MT.head()
```

Out[63]:

Excution Time	PnL_MT(bp)
2018-01-07 22:02:03.832	2.80
2018-01-07 22:02:03.832	2.80
2018-01-08 07:38:49.708	-0.10
2018-01-08 08:10:55.233	-0.20
2018-01-08 08:19:10.599	-0.15

```
In [64]: spread= ba["Ask"]-ba["Bid"]
         spread = spread*10000
```

```
In [65]: figure_spread = plt.figure()
         plt.hist(spread,bins = 50)
         plt.title("Market ask bid prices of first 9000 quotations")
         plt.legend(["spread(bp)"])
         plt.xlim((-1, 1.6))
         plt.grid()
         _ = plt.show
```



8.2 2) Opportunistic Market Making (OMM)

For each order, you will rest it within the Bid/Offer spread for a given amount of time awaiting for its (possible) opportunsitic execution.

```
In [66]: TTE =10 # 10 secs max Time to Execution (TTE)
         mid_price = (ba["Ask"]+ba["Bid"])/2
         start_mid = mid_price[start_pos]
         SL =0.00003 # max Stop Loss
```

8.2.1 First by joining "your side" of the market (ie the Bid price if you are buying, the Offer if selling) -- OMMSide

In this part, we go through every quotations after the order time in the following 10 seconds(max Time to Execution)for each order.

Then we decide whether the order should be executed at each step following the rule below in comments.

```
In [12]: bid_start = ba["Bid"][start_pos]
         ask_start =ba["Ask"][start_pos]

In [13]: def OMMside(ba,start_pos,TTE,order,start_mid,SL,PnL_MT):
             end_time = PnL_MT.index + datetime.timedelta(seconds = TTE)
             end_pos = find_pos(end_time,ba)

             execution_time_side =[]
             execution_price_side = []
             end_condition_side = []

             for i in np.arange(0,len(order)):
                 if(order["buy/sell"][i]==1):
                     for j,ask_price in enumerate(ba["Ask"][start_pos[i]:(end_pos[i]+1)]):
                         #ask price at the step is smaller than or equals to bid price at the
                         #execute the order and mark it as "end_meet_price".
                         if(ask_price == bid_start[i] or ask_price <bid_start[i]):
                             execution_price_side.append(ask_price)
                             execution_time_side.append(ba.index[start_pos[i]+j])
                             end_condition_side.append("end_meet_price")#
                             break;

                         #ask price at the step minus start mid price is larger than max stop
                         #execute to aviod larger loss and mark the order as "SL triggered".
                         if ((ask_price - start_mid[i])>SL):
                             execution_price_side.append(ask_price)
                             execution_time_side.append(ba.index[start_pos[i]+j])
                             end_condition_side.append("SL triggered")
                             break;
                         #execute at the end and mark the order as "TTE triggered"
                         if (j == ((end_pos[i]+1)-start_pos[i]-1)):
                             execution_price_side.append(ask_price)
                             execution_time_side.append(ba.index[start_pos[i]+j])
                             end_condition_side.append("TTE triggered")
                 else:
                     for j,bid_price in enumerate(ba["Bid"][start_pos[i]:(end_pos[i]+1)]):
                         #bid price at the step is bigger than or equals to ask price at the s
                         #execute the order and mark it as "end_meet_price".
                         if(bid_price == ask_start[i] or bid_price >ask_start[i]):
                             execution_price_side.append(bid_price)
                             execution_time_side.append(ba.index[start_pos[i]+j])
                             end_condition_side.append("end_meet_price")
                             break;

                         if ((start_mid[i] - bid_price)>SL):#execute to aviod larger loss and
                             execution_price_side.append(ba["Ask"][start_pos[i]+j])
                             execution_time_side.append(ba.index[start_pos[i]+j])
                             end_condition_side.append("SL triggered")
```

```
                            break;

                        if (j == ((end_pos[i]+1)-start_pos[i]-1)):#execute at the end and mar
                            execution_price_side.append(bid_price)
                            execution_time_side.append(ba.index[start_pos[i]+j])
                            end_condition_side.append("TTE triggered")
                return [execution_time_side,execution_price_side,end_condition_side]
```

In [14]:
```
[execution_time_side,execution_price_side,end_condition_side] = OMMside(ba,start_pos,
PnL_OMMside = (start_mid - np.array(execution_price_side))*np.array(order["buy/sell"])
PnL_OMMside_df = pd.DataFrame({"PnL_OMMside(bp)":PnL_OMMside*10000,
                               "End condition":end_condition_side,
                                "Execution time":execution_time_side,
                                "Mid price at starting pos":start_mid,
                              "Excution price":execution_price_side})
PnL_OMMside_df.index.names =[ "Order Time"]
annex_side = PnL_OMMside_df
PnL_OMMside_df.head()
```

Out[14]:

Order Time	PnL_OMMside(bp)	End condition	Execution time	Mid price at starting pos	Excution price
2018-01-07 22:02:03.832	-2.80	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35616
2018-01-07 22:02:03.832	-2.80	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35616
2018-01-08 07:38:49.708	0.20	end_meet_price	2018-01-08 07:38:50.407	1.354220	1.35424
2018-01-08 08:10:55.233	0.20	end_meet_price	2018-01-08 08:11:01.082	1.354140	1.35416
2018-01-08 08:19:10.599	-0.05	SL triggered	2018-01-08 08:19:11.096	1.353785	1.35378

8.2.2 Secondly by resting your order at mid market -- OMMMid

In this part, we go through every quotations after the order time in the following 10 seconds(max Time to Execution)for each order.

Then we decide whether the order should be executed at each step following the rule in comments.

```
In [15]: def OMMmid(ba,start_pos,TTE,order,start_mid,SL,PnL_MT):
             end_time = PnL_MT.index + datetime.timedelta(seconds = TTE)
             end_pos = find_pos(end_time,ba)

             execution_time =[]
             execution_price = []
             end_condition = []

             for i in np.arange(0,len(order)):
                 if(order["buy/sell"][i]==1):
                     for j,ask_price in enumerate(ba["Ask"][start_pos[i]:(end_pos[i]+1)]):
                         if(ask_price == start_mid[i] or ask_price <start_mid[i]):
                             #execute the order at the ask price and mark it as "end_meet_pric
                             execution_price.append(ask_price)
                             execution_time.append(ba.index[start_pos[i]+j])
                             end_condition.append("end_meet_price")
                             break;

                         if (ask_price-(start_mid[i])>SL):
                             #execute at the mid price immediately to aviod larger loss and ma
                             execution_price.append(mid_price[start_pos[i]+j])
                             execution_time.append(ba.index[start_pos[i]+j])
                             end_condition.append("SL triggered")
                             break;

                         if (j == ((end_pos[i]+1)-start_pos[i]-1)):
                             execution_price.append(ask_price)
                             execution_time.append(ba.index[start_pos[i]+j])
                             end_condition.append("TTE triggered")
                 else:
                     for j,bid_price in enumerate(ba["Bid"][start_pos[i]:(end_pos[i]+1)]):
                         if(bid_price == start_mid[i] or bid_price >start_mid[i]):
                             #execute the order at the bid price and mark it as "end_meet_pric
                             execution_price.append(bid_price)
                             execution_time.append(ba.index[start_pos[i]+j])
                             end_condition.append("end_meet_price")
                             break;

                         if ((start_mid[i] - bid_price)>SL):
                             #execute at the mid price immediately to aviod larger loss and ma
                             execution_price.append(bid_price)
                             execution_time.append(ba.index[start_pos[i]+j])
                             end_condition.append("SL triggered")
                             break;

                         if (j == ((end_pos[i]+1)-start_pos[i]-1)):
                             execution_price.append(bid_price)
                             execution_time.append(ba.index[start_pos[i]+j])
```

```
                        end_condition.append("TTE triggered")
            return [execution_time,execution_price,end_condition]
```

In [16]:
```
[execution_time,execution_price,end_condition] = OMMmid(ba,start_pos,TTE,order,start_m
PnL_OMMmid = (start_mid - np.array(execution_price))*np.array(order["buy/sell"])
PnL_OMMmid_df = pd.DataFrame({"PnL_OMMmid(bp)":PnL_OMMmid*10000,
                              "End condition":end_condition,
                              "Execution time":execution_time,
                              "Mid price at starting pos":start_mid,
                             "Excution price":execution_price})
PnL_OMMmid_df.index.names =[ "Order Time"]
annex_mid = PnL_OMMmid_df
PnL_OMMmid_df.head()
```

Out[16]:

Order Time	PnL_OMMmid(bp)	End condition	Execution time	Mid price at starting pos	Excution price
2018-01-07 22:02:03.832	0.00	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35588
2018-01-07 22:02:03.832	0.00	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35588
2018-01-08 07:38:49.708	-0.00	end_meet_price	2018-01-08 07:38:49.787	1.354220	1.35422
2018-01-08 08:10:55.233	0.20	end_meet_price	2018-01-08 08:11:01.082	1.354140	1.35416
2018-01-08 08:19:10.599	-0.35	SL triggered	2018-01-08 08:19:11.096	1.353785	1.35375

8.3 3) Analysis

1)For each of your 3 approaches (MT, OMMSide and OMMMid), provide a recapitulating table which will show at the minimum the average and median execution PnL for each order (taken from Mid Market), average and median time to execution, the number of times when your Stop Loss was triggered, the number of times when your order was executed on Time Limit.

In [17]:
```
mean_PnL= []
mean_PnL.append(PnL_MT[ "PnL_MT(bp)" ].mean())
mean_PnL.append(PnL_OMMmid_df["PnL_OMMmid(bp)"].mean())
mean_PnL.append(PnL_OMMside_df["PnL_OMMside(bp)"].mean())
```

```
mean_execution_time=[]
mean_execution_time.append("/")
mean_execution_time.append((PnL_OMMmid_df["Execution time"]-PnL_OMMmid_df.index).mean
mean_execution_time.append((PnL_OMMside_df["Execution time"]-PnL_OMMside_df.index).mea
```

In [18]:
```
median_PnL = []
median_PnL.append(PnL_MT[ "PnL_MT(bp)" ].median())
median_PnL.append(PnL_OMMmid_df["PnL_OMMmid(bp)"].median())
median_PnL.append(PnL_OMMside_df["PnL_OMMside(bp)"].median())
median_execution_time=[]
median_execution_time.append("/")
median_execution_time.append((PnL_OMMmid_df["Execution time"]-PnL_OMMmid_df.index).med
median_execution_time.append((PnL_OMMside_df["Execution time"]-PnL_OMMside_df.index).m
```

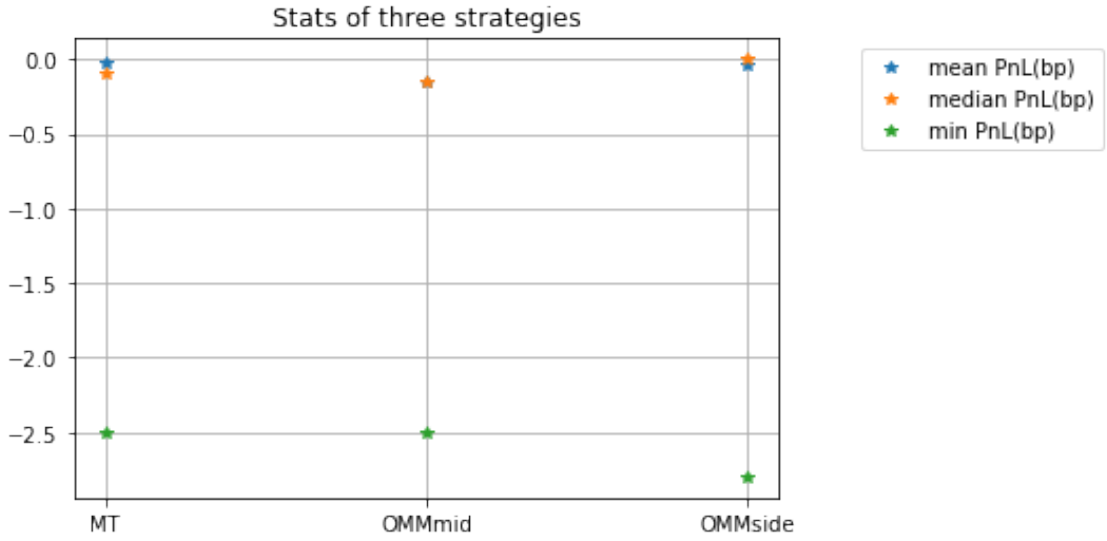
In [19]:
```
min_PnL = []
min_PnL.append(PnL_MT[ "PnL_MT(bp)" ].min())
min_PnL.append(PnL_OMMmid_df["PnL_OMMmid(bp)"].min())
min_PnL.append(PnL_OMMside_df["PnL_OMMside(bp)"].min())
min_execution_time=[]
min_execution_time.append("/")
min_execution_time.append((PnL_OMMmid_df["Execution time"]-PnL_OMMmid_df.index).min()
min_execution_time.append((PnL_OMMside_df["Execution time"]-PnL_OMMside_df.index).min
```

In [20]:
```
stats_ = pd.DataFrame({"mean PnL(bp)":mean_PnL,"median PnL(bp)":median_PnL,"min PnL(bp
                       "mean Execution time":mean_execution_time,"median Execution time
stats_.index = ["MT","OMMmid","OMMside"]
stats_
```

Out[20]:

	mean PnL(bp)	median PnL(bp)	min PnL(bp)	mean Execution time	median Execution time	min Execution time
MT	-0.020732	-0.10	-2.5	/	/	/
OMMmid	-0.150407	-0.15	-2.5	0 days 00:00:03.821918	0 days 00:00:02.041000	0 days 00:00:00
OMMside	-0.036992	-0.00	-2.8	0 days 00:00:04.680368	0 days 00:00:02.936000	0 days 00:00:00

In [21]:
```
fig1 = plt.figure()
plt.plot(["MT","OMMmid","OMMside"],stats_[stats_.columns[0:3]],'*')
L=plt.legend(np.arange(1,4),bbox_to_anchor=(1.1, 1))
for i in range(1,4):
    L.get_texts()[i-1].set_text(stats_.columns[i-1])
plt.grid()
plt.title("Stats of three strategies")
_ = plt.show
```



For all approaches,we lost money because mean PnLs are negative. However OMMmid and OMMside lose less than half of the start spread.

OMMmid and OMMside strategy works **not better** than MT strategy.

The number of times when your Stop Loss was triggered, the number of times when your order was executed on Time Limit:

```
In [22]: stop_type_number_OMMmid=PnL_OMMmid_df["End condition"].value_counts()

In [23]: stop_type_number_OMMside=PnL_OMMside_df["End condition"].value_counts()
```

2)Add other statistics if you wish (for extra points)

```
In [24]: sum_PnL=[]
         sum_PnL.append(np.sum(PnL_MT[ "PnL_MT(bp)" ]))
         sum_PnL.append(np.sum(PnL_OMMmid_df["PnL_OMMmid(bp)"]))
         sum_PnL.append(np.sum(PnL_OMMside_df["PnL_OMMside(bp)"]))
         stats = pd.DataFrame({"sum PnL(bp)":sum_PnL})
         stats.index = ["MT","OMMmid","OMMside"]
         stats

Out[24]:          sum PnL(bp)
         MT             -7.65
         OMMmid        -55.50
         OMMside       -13.65
```

We can see that OMMside approach is the worst strategy and all strategies will loss money with all order executed if we set TTE as 10 seconds and SL as 0.00003.

3)List in annex each trade with its PnL, time to execution and whether its SL or TTE has been triggered

We can see in tables in the annex at the end of this file.

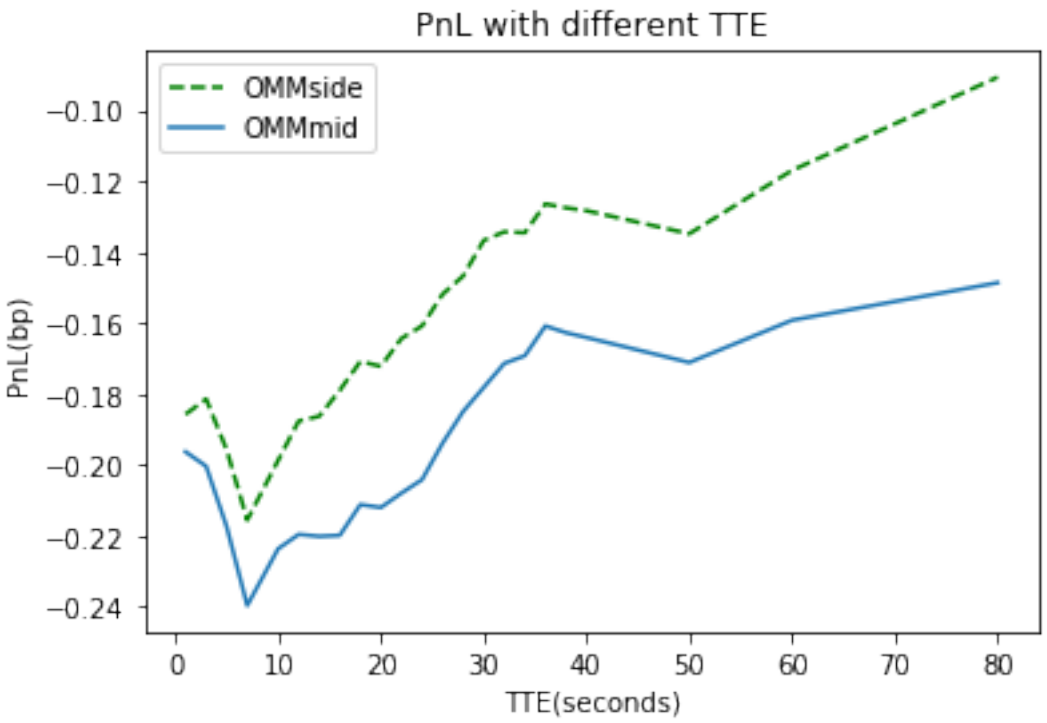
8.3.1 Further Analysis:

1)Show the influence of the length of TTE on the median execution PnL – present a graph

2)Each order was labelled as either DIS or MAR or SOM. These are three different apha generators. Can you find any significant differences in term of execution

8.3.2 1. TTE v.s. mean

```
In [54]: SL = 0.005
         TTE_list = [1,3,5,7,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,50,60,80]
         mean_mid =[]
         mean_side = []
         for tte in TTE_list:
             #calculation of PnL_mid
             [execution_time,execution_price,end_condition] = OMMmid(ba,start_pos,tte,order,sta
             PnL_OMMmid_temp = (start_mid - np.array(execution_price))*np.array(order["buy/sell
             mean_mid_temp = np.mean(np.array(PnL_OMMmid_temp))
             #calculation of PnL_side
             [execution_time_side,execution_price_side,end_condition_side] = OMMside(ba,start_p
             PnL_OMMside_temp = (start_mid - np.array(execution_price_side))*np.array(order["bu
             mean_side_temp = np.mean(np.array(PnL_OMMside_temp))

             mean_mid.append(mean_mid_temp)
             mean_side.append(mean_side_temp)

In [55]: fig2=plt.figure()
         plt.plot(TTE_list,mean_side,'g--',label="OMMside")
         plt.plot(TTE_list,mean_mid,label="OMMmid")
         plt.legend()
         plt.title('PnL with different TTE')# give plot a title
         plt.xlabel('TTE(seconds)')# make axis labels
         plt.ylabel('PnL(bp)')
         plt.show()
```



1) **Positive Margin**

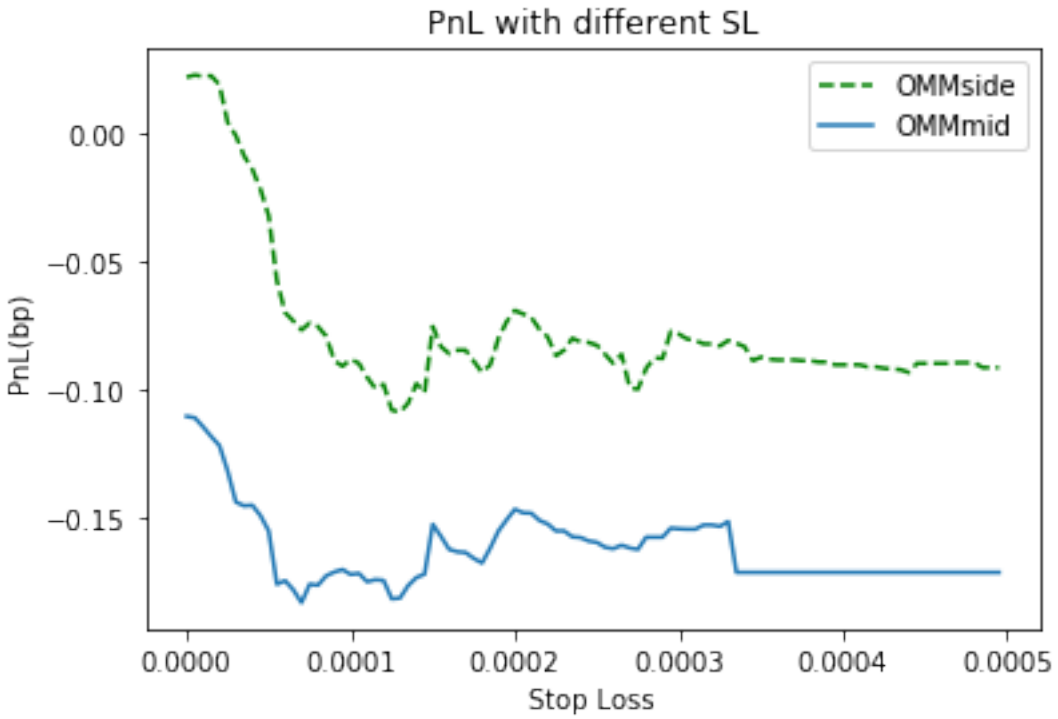
As TTE increasing, the median of both strategy are going up.OMMside is a better strategy when we give a TTE that is long enough because it can earn more money than OMMmid strategy.

2) **OMMside and OMMmid have about the same sensitivity to TTE**

8.3.3 2. SL vs. PnL mean

```
In [77]: SL = np.arange(0,0.0005,0.000005)
         TTE_list = 50
         mean_mid =[]
         mean_side = []
         for sl in SL:
             #calculation of PnL_mid
             [execution_time,execution_price,end_condition] = OMMmid(ba,start_pos,TTE,order,sta
             PnL_OMMmid_temp = (start_mid - np.array(execution_price))*np.array(order["buy/sell
             mean_mid_temp = np.mean(np.array(PnL_OMMmid_temp))
             #calculation of PnL_side
             [execution_time_side,execution_price_side,end_condition_side] = OMMside(ba,start_p
             PnL_OMMside_temp = (start_mid - np.array(execution_price_side))*np.array(order["bu
             mean_side_temp = np.mean(np.array(PnL_OMMside_temp))

             mean_mid.append(mean_mid_temp)
             mean_side.append(mean_side_temp)
```

```
In [78]: fig4=plt.figure()
         plt.plot(SL,mean_side,'g--',label="OMMside")
         plt.plot(SL,mean_mid,label="OMMmid")
         plt.legend()
         plt.title('PnL with different SL')# give plot a title
         plt.xlabel('Stop Loss')# make axis labels
         plt.ylabel('PnL(bp)')
         plt.show()
```



8.3.4 3.Anlysis about different Alphas

We can see from the conclusion above, 50 seconds for TTE might be a safe choice.

```
In [89]: TTE = 50
         SL = 0.00002
```

```
In [90]: [execution_time_side,execution_price_side,end_condition_side] = OMMside(ba,start_pos,
         PnL_OMMside = (start_mid - np.array(execution_price_side))*np.array(order["buy/sell"])
         PnL_OMMside =pd.DataFrame(PnL_OMMside*10000)
         PnL_OMMside.columns = ["PnL_OMMside(bp)"]
         PnL_OMMside.index.names =[ "Order Time"]
         PnL_OMMside_df = pd.DataFrame({"PnL_OMMside(bp)":PnL_OMMside["PnL_OMMside(bp)"],
                                        "End condition":end_condition_side,"Execution time":exec
                                         "Mid price at starting pos":start_mid,
```

```
                                    "Excution price":execution_price_side,
                                     "Alpha":np.array(order["alpha"])})
        PnL_OMMside_df.head()
```

Out[90]:

Order Time	PnL_OMMside(bp)	End condition	Execution time	Mid price at starting pos	Excution price	Alpha
2018-01-07 22:02:03.832	-2.80	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35616	SOM
2018-01-07 22:02:03.832	-2.80	SL triggered	2018-01-07 22:02:03.832	1.355880	1.35616	SOM
2018-01-08 07:38:49.708	0.20	end_meet_price	2018-01-08 07:38:50.407	1.354220	1.35424	SOM
2018-01-08 08:10:55.233	0.20	SL triggered	2018-01-08 08:10:55.233	1.354140	1.35416	SOM
2018-01-08 08:19:10.599	0.15	SL triggered	2018-01-08 08:19:10.619	1.353785	1.35380	SOM

```
In [91]: median_side = PnL_OMMside_df[["PnL_OMMside(bp)","Alpha"]].groupby(['Alpha']).median()
         mean_side = PnL_OMMside_df[["PnL_OMMside(bp)","Alpha"]].groupby(['Alpha']).mean()

In [92]: [execution_time,execution_price,end_condition] = OMMmid(ba,start_pos,TTE,order,start_
         PnL_OMMmid = (start_mid - np.array(execution_price))*np.array(order["buy/sell"])
         PnL_OMMmid =pd.DataFrame(PnL_OMMmid*10000)
         PnL_OMMmid.columns = ["PnL_OMMmid(bp)"]
         PnL_OMMmid.index.names =[ "Order Time"]
         PnL_OMMmid_df = pd.DataFrame({"PnL_OMMmid(bp)":PnL_OMMmid["PnL_OMMmid(bp)"],
                                       "End condition":end_condition,"Execution time":executio
                                        "Mid price at starting pos":start_mid,
                                      "Excution price":execution_price,
                                       "Alpha":np.array(order["alpha"])})
         PnL_OMMmid_df.head()
```

Out[92]:

Order Time	PnL_OMMmid(bp)	End condition
2018-01-07 22:02:03.832	0.00	SL triggered
2018-01-07 22:02:03.832	0.00	SL triggered
2018-01-08 07:38:49.708	-0.00	end_meet_price
2018-01-08 08:10:55.233	-0.20	SL triggered

```
2018-01-08 08:19:10.599           -0.25    SL triggered

                                  Execution time  Mid price at starting pos  \
Order Time
2018-01-07 22:02:03.832 2018-01-07 22:02:03.832                    1.355880
2018-01-07 22:02:03.832 2018-01-07 22:02:03.832                    1.355880
2018-01-08 07:38:49.708 2018-01-08 07:38:49.787                    1.354220
2018-01-08 08:10:55.233 2018-01-08 08:10:55.233                    1.354140
2018-01-08 08:19:10.599 2018-01-08 08:19:10.619                    1.353785

                          Excution price Alpha
Order Time
2018-01-07 22:02:03.832         1.35588   SOM
2018-01-07 22:02:03.832         1.35588   SOM
2018-01-08 07:38:49.708         1.35422   SOM
2018-01-08 08:10:55.233         1.35412   SOM
2018-01-08 08:19:10.599         1.35376   SOM
```

In [93]:
```
median_mid = PnL_OMMmid_df[["PnL_OMMmid(bp)","Alpha"]].groupby(['Alpha']).median()
mean_mid = PnL_OMMmid_df[["PnL_OMMmid(bp)","Alpha"]].groupby(['Alpha']).mean()
mean_mid
```

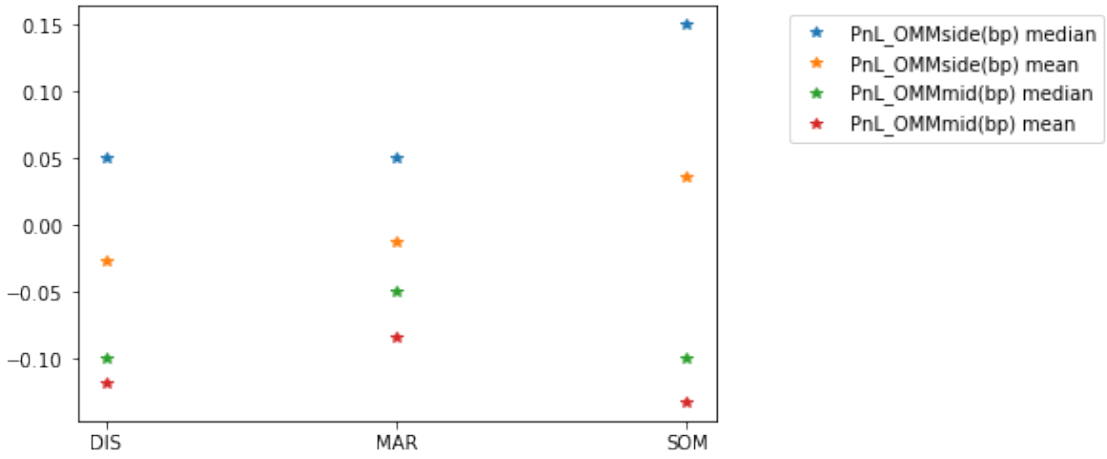
Out[93]:
```
       PnL_OMMmid(bp)
Alpha
DIS         -0.117500
MAR         -0.083051
SOM         -0.132000
```

In [94]:
```
alpha_=pd.DataFrame({"PnL_OMMside(bp) median":median_side["PnL_OMMside(bp)"],"PnL_OMMs
                     "PnL_OMMmid(bp) median":median_mid["PnL_OMMmid(bp)"],"PnL_OMMmid(bp) mea
alpha_
```

Out[94]:
```
       PnL_OMMside(bp) median  PnL_OMMside(bp) mean  PnL_OMMmid(bp) median  \
Alpha
DIS                      0.05             -0.026667                  -0.10
MAR                      0.05             -0.012712                  -0.05
SOM                      0.15              0.035600                  -0.10

       PnL_OMMmid(bp) mean
Alpha
DIS              -0.117500
MAR              -0.083051
SOM              -0.132000
```

In [95]:
```
fig3 = plt.figure()
plt.plot(["DIS","MAR","SOM"],alpha_,'*')
L=plt.legend(np.arange(1,5),bbox_to_anchor=(1.1, 1))
for i in range(1,5):
    L.get_texts()[i-1].set_text(alpha_.columns[i-1])
plt.show
```

Out[95]: <function matplotlib.pyplot.show(*args, **kw)>



We can find from the graph above that:

1) For all of these three strategies, OMMside is a better choice with TTE = 40 and SL = 0.01

2) To rank these three alpha strategies, we can say that MAR>SOM>DIS. However, the difference is not that huge.

9 6. Annex

In [34]: annex_side[:]

Out[34]:

Order Time	PnL_OMMside(bp)	End condition
2018-01-07 22:02:03.832	-2.80	SL triggered
2018-01-07 22:02:03.832	-2.80	SL triggered
2018-01-08 07:38:49.708	0.20	end_meet_price
2018-01-08 08:10:55.233	0.20	end_meet_price
2018-01-08 08:19:10.599	-0.05	SL triggered
2018-01-08 08:23:16.738	-0.10	TTE triggered
2018-01-08 08:34:40.821	0.05	end_meet_price
2018-01-08 11:25:14.548	-0.35	SL triggered
2018-01-08 11:42:08.435	-0.35	SL triggered
2018-01-08 12:15:36.284	0.25	end_meet_price
2018-01-08 12:15:36.284	0.25	end_meet_price
2018-01-08 12:20:20.564	0.20	end_meet_price
2018-01-08 12:26:08.344	0.10	end_meet_price
2018-01-08 15:41:57.641	-0.35	SL triggered
2018-01-08 15:41:57.641	-0.35	SL triggered
2018-01-08 20:03:50.183	0.20	end_meet_price
2018-01-08 20:06:52.786	-0.10	SL triggered

```
2018-01-08 21:02:55.856            0.10  end_meet_price
2018-01-08 21:07:41.214            0.20    SL triggered
2018-01-08 21:18:37.069            0.15    SL triggered
2018-01-08 21:21:10.383           -0.25   TTE triggered
2018-01-08 21:21:10.383           -0.25   TTE triggered
2018-01-08 21:33:54.637           -0.15   TTE triggered
2018-01-08 21:37:20.309           -0.00   TTE triggered
2018-01-08 22:09:00.566            0.45    SL triggered
2018-01-08 22:09:00.566            0.45    SL triggered
2018-01-08 23:04:53.132            0.15    SL triggered
2018-01-08 23:09:46.080            0.20    SL triggered
2018-01-09 01:12:38.162           -0.25   TTE triggered
2018-01-09 02:35:31.606           -0.10   TTE triggered
...                                 ...             ...
2018-01-12 09:36:53.944            0.05  end_meet_price
2018-01-12 10:22:17.907            0.15    SL triggered
2018-01-12 10:22:17.907            0.15    SL triggered
2018-01-12 10:29:20.069           -0.05   TTE triggered
2018-01-12 10:38:17.673            0.30  end_meet_price
2018-01-12 11:02:13.620           -0.35    SL triggered
2018-01-12 11:02:49.340            0.00   TTE triggered
2018-01-12 11:03:30.028            0.25  end_meet_price
2018-01-12 11:17:41.907            0.20  end_meet_price
2018-01-12 11:17:41.907            0.20  end_meet_price
2018-01-12 11:18:29.346            0.15  end_meet_price
2018-01-12 11:32:58.328           -0.35    SL triggered
2018-01-12 11:35:02.400            0.15  end_meet_price
2018-01-12 12:03:37.044            0.20  end_meet_price
2018-01-12 12:04:53.512            0.20  end_meet_price
2018-01-12 12:06:16.678            0.50  end_meet_price
2018-01-12 12:18:15.666            0.15  end_meet_price
2018-01-12 12:20:26.177           -0.35    SL triggered
2018-01-12 12:33:30.547            0.15  end_meet_price
2018-01-12 12:36:14.212           -0.35    SL triggered
2018-01-12 13:04:03.433            0.05    SL triggered
2018-01-12 14:06:24.667           -0.10    SL triggered
2018-01-12 14:33:14.631            0.15  end_meet_price
2018-01-12 15:02:01.285            0.35  end_meet_price
2018-01-12 15:17:02.234           -0.35    SL triggered
2018-01-12 15:32:15.121           -0.25   TTE triggered
2018-01-12 16:03:23.404           -0.35    SL triggered
2018-01-12 16:03:24.328           -0.45    SL triggered
2018-01-12 16:18:07.271            0.00  end_meet_price
2018-01-12 16:33:22.620           -0.30    SL triggered

                                Execution time  Mid price at starting pos  \
Order Time
2018-01-07 22:02:03.832 2018-01-07 22:02:03.832                   1.355880
```

```
2018-01-07 22:02:03.832 2018-01-07 22:02:03.832                 1.355880
2018-01-08 07:38:49.708 2018-01-08 07:38:50.407                 1.354220
2018-01-08 08:10:55.233 2018-01-08 08:11:01.082                 1.354140
2018-01-08 08:19:10.599 2018-01-08 08:19:11.096                 1.353785
2018-01-08 08:23:16.738 2018-01-08 08:23:29.753                 1.353530
2018-01-08 08:34:40.821 2018-01-08 08:34:40.878                 1.353825
2018-01-08 11:25:14.548 2018-01-08 11:25:16.331                 1.354385
2018-01-08 11:42:08.435 2018-01-08 11:42:08.527                 1.354385
2018-01-08 12:15:36.284 2018-01-08 12:15:38.333                 1.354535
2018-01-08 12:15:36.284 2018-01-08 12:15:38.333                 1.354535
2018-01-08 12:20:20.564 2018-01-08 12:20:30.394                 1.354540
2018-01-08 12:26:08.344 2018-01-08 12:26:11.494                 1.354620
2018-01-08 15:41:57.641 2018-01-08 15:42:01.325                 1.355845
2018-01-08 15:41:57.641 2018-01-08 15:42:01.325                 1.355845
2018-01-08 20:03:50.183 2018-01-08 20:03:51.483                 1.356430
2018-01-08 20:06:52.786 2018-01-08 20:06:56.714                 1.356520
2018-01-08 21:02:55.856 2018-01-08 21:03:02.366                 1.356700
2018-01-08 21:07:41.214 2018-01-08 21:07:49.366                 1.356710
2018-01-08 21:18:37.069 2018-01-08 21:18:43.395                 1.356595
2018-01-08 21:21:10.383 2018-01-08 21:21:29.194                 1.356595
2018-01-08 21:21:10.383 2018-01-08 21:21:29.194                 1.356595
2018-01-08 21:33:54.637 2018-01-08 21:34:04.692                 1.356615
2018-01-08 21:37:20.309 2018-01-08 21:37:41.626                 1.356600
2018-01-08 22:09:00.566 2018-01-08 22:09:00.566                 1.356585
2018-01-08 22:09:00.566 2018-01-08 22:09:00.566                 1.356585
2018-01-08 23:04:53.132 2018-01-08 23:05:01.742                 1.356945
2018-01-08 23:09:46.080 2018-01-08 23:09:58.118                 1.356990
2018-01-09 01:12:38.162 2018-01-09 01:12:49.854                 1.356145
2018-01-09 02:35:31.606 2018-01-09 02:35:41.857                 1.357590
...                                             ...                  ...
2018-01-12 09:36:53.944 2018-01-12 09:37:00.828                 1.361935
2018-01-12 10:22:17.907 2018-01-12 10:22:22.621                 1.361395
2018-01-12 10:22:17.907 2018-01-12 10:22:22.621                 1.361395
2018-01-12 10:29:20.069 2018-01-12 10:29:30.474                 1.361735
2018-01-12 10:38:17.673 2018-01-12 10:38:18.942                 1.361580
2018-01-12 11:02:13.620 2018-01-12 11:02:18.929                 1.361535
2018-01-12 11:02:49.340 2018-01-12 11:03:00.052                 1.361540
2018-01-12 11:03:30.028 2018-01-12 11:03:34.967                 1.361595
2018-01-12 11:17:41.907 2018-01-12 11:17:42.185                 1.362690
2018-01-12 11:17:41.907 2018-01-12 11:17:42.185                 1.362690
2018-01-12 11:18:29.346 2018-01-12 11:18:30.803                 1.362705
2018-01-12 11:32:58.328 2018-01-12 11:32:58.500                 1.363435
2018-01-12 11:35:02.400 2018-01-12 11:35:02.564                 1.363935
2018-01-12 12:03:37.044 2018-01-12 12:03:37.360                 1.363550
2018-01-12 12:04:53.512 2018-01-12 12:04:56.259                 1.363130
2018-01-12 12:06:16.678 2018-01-12 12:06:21.680                 1.363190
2018-01-12 12:18:15.666 2018-01-12 12:18:22.777                 1.363365
2018-01-12 12:20:26.177 2018-01-12 12:20:26.289                 1.363335
```

2018-01-12 12:33:30.547	2018-01-12 12:33:30.567	1.363465
2018-01-12 12:36:14.212	2018-01-12 12:36:17.952	1.362915
2018-01-12 13:04:03.433	2018-01-12 13:04:04.153	1.363825
2018-01-12 14:06:24.667	2018-01-12 14:06:25.487	1.365010
2018-01-12 14:33:14.631	2018-01-12 14:33:17.211	1.366875
2018-01-12 15:02:01.285	2018-01-12 15:02:01.610	1.368035
2018-01-12 15:17:02.234	2018-01-12 15:17:03.984	1.366895
2018-01-12 15:32:15.121	2018-01-12 15:32:25.244	1.367435
2018-01-12 16:03:23.404	2018-01-12 16:03:24.439	1.368575
2018-01-12 16:03:24.328	2018-01-12 16:03:26.369	1.368585
2018-01-12 16:18:07.271	2018-01-12 16:18:07.271	1.368410
2018-01-12 16:33:22.620	2018-01-12 16:33:22.639	1.367920

Order Time	Excution price
2018-01-07 22:02:03.832	1.35616
2018-01-07 22:02:03.832	1.35616
2018-01-08 07:38:49.708	1.35424
2018-01-08 08:10:55.233	1.35416
2018-01-08 08:19:10.599	1.35378
2018-01-08 08:23:16.738	1.35352
2018-01-08 08:34:40.821	1.35383
2018-01-08 11:25:14.548	1.35442
2018-01-08 11:42:08.435	1.35442
2018-01-08 12:15:36.284	1.35451
2018-01-08 12:15:36.284	1.35451
2018-01-08 12:20:20.564	1.35452
2018-01-08 12:26:08.344	1.35461
2018-01-08 15:41:57.641	1.35588
2018-01-08 15:41:57.641	1.35588
2018-01-08 20:03:50.183	1.35645
2018-01-08 20:06:52.786	1.35651
2018-01-08 21:02:55.856	1.35671
2018-01-08 21:07:41.214	1.35673
2018-01-08 21:18:37.069	1.35661
2018-01-08 21:21:10.383	1.35657
2018-01-08 21:21:10.383	1.35657
2018-01-08 21:33:54.637	1.35660
2018-01-08 21:37:20.309	1.35660
2018-01-08 22:09:00.566	1.35663
2018-01-08 22:09:00.566	1.35663
2018-01-08 23:04:53.132	1.35696
2018-01-08 23:09:46.080	1.35701
2018-01-09 01:12:38.162	1.35612
2018-01-09 02:35:31.606	1.35760
...	...
2018-01-12 09:36:53.944	1.36194
2018-01-12 10:22:17.907	1.36141

```
2018-01-12 10:22:17.907         1.36141
2018-01-12 10:29:20.069         1.36174
2018-01-12 10:38:17.673         1.36161
2018-01-12 11:02:13.620         1.36157
2018-01-12 11:02:49.340         1.36154
2018-01-12 11:03:30.028         1.36157
2018-01-12 11:17:41.907         1.36267
2018-01-12 11:17:41.907         1.36267
2018-01-12 11:18:29.346         1.36269
2018-01-12 11:32:58.328         1.36347
2018-01-12 11:35:02.400         1.36392
2018-01-12 12:03:37.044         1.36353
2018-01-12 12:04:53.512         1.36311
2018-01-12 12:06:16.678         1.36314
2018-01-12 12:18:15.666         1.36335
2018-01-12 12:20:26.177         1.36337
2018-01-12 12:33:30.547         1.36345
2018-01-12 12:36:14.212         1.36295
2018-01-12 13:04:03.433         1.36383
2018-01-12 14:06:24.667         1.36500
2018-01-12 14:33:14.631         1.36686
2018-01-12 15:02:01.285         1.36800
2018-01-12 15:17:02.234         1.36693
2018-01-12 15:32:15.121         1.36746
2018-01-12 16:03:23.404         1.36861
2018-01-12 16:03:24.328         1.36863
2018-01-12 16:18:07.271         1.36841
2018-01-12 16:33:22.620         1.36795

[369 rows x 5 columns]
```

```
In [35]: annex_mid[:]
```

```
Out[35]:                          PnL_OMMmid(bp)   End condition  \
         Order Time
         2018-01-07 22:02:03.832    0.000000e+00    SL triggered
         2018-01-07 22:02:03.832    0.000000e+00    SL triggered
         2018-01-08 07:38:49.708   -0.000000e+00  end_meet_price
         2018-01-08 08:10:55.233    2.000000e-01  end_meet_price
         2018-01-08 08:19:10.599   -3.500000e-01    SL triggered
         2018-01-08 08:23:16.738   -1.000000e-01   TTE triggered
         2018-01-08 08:34:40.821    5.000000e-02  end_meet_price
         2018-01-08 11:25:14.548   -1.000000e-01    SL triggered
         2018-01-08 11:42:08.435   -1.000000e-01    SL triggered
         2018-01-08 12:15:36.284    5.000000e-02  end_meet_price
         2018-01-08 12:15:36.284    5.000000e-02  end_meet_price
         2018-01-08 12:20:20.564    2.220446e-12  end_meet_price
         2018-01-08 12:26:08.344    0.000000e+00  end_meet_price
```

```
2018-01-08 15:41:57.641   -1.500000e-01    SL triggered
2018-01-08 15:41:57.641   -1.500000e-01    SL triggered
2018-01-08 20:03:50.183    1.000000e-01  end_meet_price
2018-01-08 20:06:52.786   -5.000000e-01    SL triggered
2018-01-08 21:02:55.856    1.000000e-01  end_meet_price
2018-01-08 21:07:41.214   -3.000000e-01    SL triggered
2018-01-08 21:18:37.069   -3.500000e-01    SL triggered
2018-01-08 21:21:10.383   -2.500000e-01   TTE triggered
2018-01-08 21:21:10.383   -2.500000e-01   TTE triggered
2018-01-08 21:33:54.637   -1.500000e-01   TTE triggered
2018-01-08 21:37:20.309   -0.000000e+00  end_meet_price
2018-01-08 22:09:00.566   -4.500000e-01    SL triggered
2018-01-08 22:09:00.566   -4.500000e-01    SL triggered
2018-01-08 23:04:53.132    5.000000e-02  end_meet_price
2018-01-08 23:09:46.080   -3.000000e-01    SL triggered
2018-01-09 01:12:38.162   -2.500000e-01   TTE triggered
2018-01-09 02:35:31.606    0.000000e+00  end_meet_price
...                                 ...             ...
2018-01-12 09:36:53.944    5.000000e-02  end_meet_price
2018-01-12 10:22:17.907   -3.500000e-01    SL triggered
2018-01-12 10:22:17.907   -3.500000e-01    SL triggered
2018-01-12 10:29:20.069   -5.000000e-02   TTE triggered
2018-01-12 10:38:17.673    3.000000e-01  end_meet_price
2018-01-12 11:02:13.620   -2.000000e-01    SL triggered
2018-01-12 11:02:49.340    0.000000e+00  end_meet_price
2018-01-12 11:03:30.028    5.000000e-02  end_meet_price
2018-01-12 11:17:41.907    2.220446e-12  end_meet_price
2018-01-12 11:17:41.907    2.220446e-12  end_meet_price
2018-01-12 11:18:29.346    5.000000e-02  end_meet_price
2018-01-12 11:32:58.328   -2.000000e-01    SL triggered
2018-01-12 11:35:02.400    1.500000e-01  end_meet_price
2018-01-12 12:03:37.044    0.000000e+00  end_meet_price
2018-01-12 12:04:53.512    0.000000e+00  end_meet_price
2018-01-12 12:06:16.678    5.000000e-01  end_meet_price
2018-01-12 12:18:15.666    5.000000e-02  end_meet_price
2018-01-12 12:20:26.177   -1.500000e-01    SL triggered
2018-01-12 12:33:30.547    5.000000e-02  end_meet_price
2018-01-12 12:36:14.212   -2.000000e-01    SL triggered
2018-01-12 13:04:03.433   -3.500000e-01    SL triggered
2018-01-12 14:06:24.667   -4.000000e-01    SL triggered
2018-01-12 14:33:14.631    5.000000e-02  end_meet_price
2018-01-12 15:02:01.285    5.000000e-02  end_meet_price
2018-01-12 15:17:02.234   -1.000000e-01    SL triggered
2018-01-12 15:32:15.121   -2.500000e-01   TTE triggered
2018-01-12 16:03:23.404   -1.500000e-01    SL triggered
2018-01-12 16:03:24.328   -3.000000e-01    SL triggered
2018-01-12 16:18:07.271    0.000000e+00  end_meet_price
2018-01-12 16:33:22.620   -5.000000e-02    SL triggered
```

Order Time	Execution time	Mid price at starting pos
2018-01-07 22:02:03.832	2018-01-07 22:02:03.832	1.355880
2018-01-07 22:02:03.832	2018-01-07 22:02:03.832	1.355880
2018-01-08 07:38:49.708	2018-01-08 07:38:49.787	1.354220
2018-01-08 08:10:55.233	2018-01-08 08:11:01.082	1.354140
2018-01-08 08:19:10.599	2018-01-08 08:19:11.096	1.353785
2018-01-08 08:23:16.738	2018-01-08 08:23:29.753	1.353530
2018-01-08 08:34:40.821	2018-01-08 08:34:40.878	1.353825
2018-01-08 11:25:14.548	2018-01-08 11:25:16.331	1.354385
2018-01-08 11:42:08.435	2018-01-08 11:42:08.527	1.354385
2018-01-08 12:15:36.284	2018-01-08 12:15:38.265	1.354535
2018-01-08 12:15:36.284	2018-01-08 12:15:38.265	1.354535
2018-01-08 12:20:20.564	2018-01-08 12:20:20.702	1.354540
2018-01-08 12:26:08.344	2018-01-08 12:26:09.809	1.354620
2018-01-08 15:41:57.641	2018-01-08 15:42:01.325	1.355845
2018-01-08 15:41:57.641	2018-01-08 15:42:01.325	1.355845
2018-01-08 20:03:50.183	2018-01-08 20:03:51.478	1.356430
2018-01-08 20:06:52.786	2018-01-08 20:06:56.714	1.356520
2018-01-08 21:02:55.856	2018-01-08 21:03:02.366	1.356700
2018-01-08 21:07:41.214	2018-01-08 21:07:49.366	1.356710
2018-01-08 21:18:37.069	2018-01-08 21:18:43.395	1.356595
2018-01-08 21:21:10.383	2018-01-08 21:21:29.194	1.356595
2018-01-08 21:21:10.383	2018-01-08 21:21:29.194	1.356595
2018-01-08 21:33:54.637	2018-01-08 21:34:04.692	1.356615
2018-01-08 21:37:20.309	2018-01-08 21:37:20.374	1.356600
2018-01-08 22:09:00.566	2018-01-08 22:09:00.566	1.356585
2018-01-08 22:09:00.566	2018-01-08 22:09:00.566	1.356585
2018-01-08 23:04:53.132	2018-01-08 23:05:01.020	1.356945
2018-01-08 23:09:46.080	2018-01-08 23:09:58.118	1.356990
2018-01-09 01:12:38.162	2018-01-09 01:12:49.854	1.356145
2018-01-09 02:35:31.606	2018-01-09 02:35:34.110	1.357590
...	...	...
2018-01-12 09:36:53.944	2018-01-12 09:37:00.828	1.361935
2018-01-12 10:22:17.907	2018-01-12 10:22:22.621	1.361395
2018-01-12 10:22:17.907	2018-01-12 10:22:22.621	1.361395
2018-01-12 10:29:20.069	2018-01-12 10:29:30.474	1.361735
2018-01-12 10:38:17.673	2018-01-12 10:38:18.942	1.361580
2018-01-12 11:02:13.620	2018-01-12 11:02:18.929	1.361535
2018-01-12 11:02:49.340	2018-01-12 11:02:53.791	1.361540
2018-01-12 11:03:30.028	2018-01-12 11:03:34.946	1.361595
2018-01-12 11:17:41.907	2018-01-12 11:17:42.178	1.362690
2018-01-12 11:17:41.907	2018-01-12 11:17:42.178	1.362690
2018-01-12 11:18:29.346	2018-01-12 11:18:29.972	1.362705
2018-01-12 11:32:58.328	2018-01-12 11:32:58.500	1.363435
2018-01-12 11:35:02.400	2018-01-12 11:35:02.564	1.363935
2018-01-12 12:03:37.044	2018-01-12 12:03:37.279	1.363550

2018-01-12 12:04:53.512	2018-01-12 12:04:54.192	1.363130
2018-01-12 12:06:16.678	2018-01-12 12:06:21.680	1.363190
2018-01-12 12:18:15.666	2018-01-12 12:18:22.680	1.363365
2018-01-12 12:20:26.177	2018-01-12 12:20:26.289	1.363335
2018-01-12 12:33:30.547	2018-01-12 12:33:30.552	1.363465
2018-01-12 12:36:14.212	2018-01-12 12:36:17.952	1.362915
2018-01-12 13:04:03.433	2018-01-12 13:04:04.153	1.363825
2018-01-12 14:06:24.667	2018-01-12 14:06:25.487	1.365010
2018-01-12 14:33:14.631	2018-01-12 14:33:17.150	1.366875
2018-01-12 15:02:01.285	2018-01-12 15:02:01.543	1.368035
2018-01-12 15:17:02.234	2018-01-12 15:17:03.984	1.366895
2018-01-12 15:32:15.121	2018-01-12 15:32:25.244	1.367435
2018-01-12 16:03:23.404	2018-01-12 16:03:24.439	1.368575
2018-01-12 16:03:24.328	2018-01-12 16:03:26.369	1.368585
2018-01-12 16:18:07.271	2018-01-12 16:18:07.271	1.368410
2018-01-12 16:33:22.620	2018-01-12 16:33:22.639	1.367920

Order Time	Excution price
2018-01-07 22:02:03.832	1.355880
2018-01-07 22:02:03.832	1.355880
2018-01-08 07:38:49.708	1.354220
2018-01-08 08:10:55.233	1.354160
2018-01-08 08:19:10.599	1.353750
2018-01-08 08:23:16.738	1.353520
2018-01-08 08:34:40.821	1.353830
2018-01-08 11:25:14.548	1.354395
2018-01-08 11:42:08.435	1.354395
2018-01-08 12:15:36.284	1.354530
2018-01-08 12:15:36.284	1.354530
2018-01-08 12:20:20.564	1.354540
2018-01-08 12:26:08.344	1.354620
2018-01-08 15:41:57.641	1.355860
2018-01-08 15:41:57.641	1.355860
2018-01-08 20:03:50.183	1.356440
2018-01-08 20:06:52.786	1.356470
2018-01-08 21:02:55.856	1.356710
2018-01-08 21:07:41.214	1.356680
2018-01-08 21:18:37.069	1.356560
2018-01-08 21:21:10.383	1.356570
2018-01-08 21:21:10.383	1.356570
2018-01-08 21:33:54.637	1.356600
2018-01-08 21:37:20.309	1.356600
2018-01-08 22:09:00.566	1.356540
2018-01-08 22:09:00.566	1.356540
2018-01-08 23:04:53.132	1.356950
2018-01-08 23:09:46.080	1.356960
2018-01-09 01:12:38.162	1.356120

```
2018-01-09 02:35:31.606        1.357590
...                                 ...
2018-01-12 09:36:53.944        1.361940
2018-01-12 10:22:17.907        1.361360
2018-01-12 10:22:17.907        1.361360
2018-01-12 10:29:20.069        1.361740
2018-01-12 10:38:17.673        1.361610
2018-01-12 11:02:13.620        1.361555
2018-01-12 11:02:49.340        1.361540
2018-01-12 11:03:30.028        1.361590
2018-01-12 11:17:41.907        1.362690
2018-01-12 11:17:41.907        1.362690
2018-01-12 11:18:29.346        1.362700
2018-01-12 11:32:58.328        1.363455
2018-01-12 11:35:02.400        1.363920
2018-01-12 12:03:37.044        1.363550
2018-01-12 12:04:53.512        1.363130
2018-01-12 12:06:16.678        1.363140
2018-01-12 12:18:15.666        1.363360
2018-01-12 12:20:26.177        1.363350
2018-01-12 12:33:30.547        1.363460
2018-01-12 12:36:14.212        1.362935
2018-01-12 13:04:03.433        1.363790
2018-01-12 14:06:24.667        1.364970
2018-01-12 14:33:14.631        1.366870
2018-01-12 15:02:01.285        1.368030
2018-01-12 15:17:02.234        1.366905
2018-01-12 15:32:15.121        1.367460
2018-01-12 16:03:23.404        1.368590
2018-01-12 16:03:24.328        1.368615
2018-01-12 16:18:07.271        1.368410
2018-01-12 16:33:22.620        1.367925

[369 rows x 5 columns]
```