In [1]:

```
import dolphindb as ddb
import talib as ta
import re
s=ddb.session()
import pandas as pd
import numpy as np
s.connect('10.0.60.55',8509,'admin','123456')
z=ddb.session()
z.connect('10.0.40.33',8505,'admin','123456')
```

Out[1]:

True

In [49]:

```
import dolphindb as ddb
import talib as ta
import re
s=ddb.session()
import pandas as pd
import numpy as np
s.connect('10.0.60.55',8509,'admin','123456')
z=ddb.session()
z.connect('10.0.40.33',8505,'admin','123456')

class ETFClassification:
    def __init__(self):
        # 511380 & 500060 is missing for Bond ETF
        self.etf_dict = {
            'BigCapIndex': ['510300', '510330', '510050', '159919', '510310'],
            'SmallCapIndex': ['159949', '510500', '159915', '512500', '159968'],
            'TotalMarketIndex': ['515800', '512990', '512380', '512160', '512090'],
            'TMTSector': ['159995', '512760', '515050', '159801', '512480'],
            'BioMedicalSector': ['512290', '159992', '512170', '512010', '159938'],
            'TechTopic': ['515000', '515750', '159807', '515860', '159987'],
            'NEVTopic': ['515030', '515700', '159806'],
            'RealEstateSector': ['512880', '512000', '512800', '512900', '159993'],
            'ConsumptionSector': ['159928', '512690', '515650', '159996', '510150'],
            'MilitarySector': ['512660', '512710'],
            'PeriodicSector': ['515210', '512400', '515220'],
            'SmartBeta': ['159966', '159905', '159967', '510880', '515180'],
            'Other': ['515680', '515900', '159976', '515600', '159978'],
            'Bond': ['511010', '511260', '159972'],
            'QDII': ['510900', '159920', '513050', '513090', '513500'],
            'Commodity': ['518880', '159934', '159937', '518800', '159980']
        }

    def etf_daily_avg_turnover(self, symbol):
        if symbol[:2] == '51':
            db = "'dfs://STOCK_SH_TRDMIN','STOCK_SH_TRDMIN'"
        elif symbol[:2] == '15':
            db = "'dfs://STOCK_SZ_TRDMIN','STOCK_SZ_TRDMIN'"
        else:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")

        sql = f"select sum(turnover) as turnover_daily \
                from loadTable({db})  \
                where symbol = `{symbol}, cycle = 1, tradingDay < 2018.02.01 \
                group by tradingDay"
        data = s.run(sql)

        sql2 = f"select last(turnover) as turnover_daily \
                from loadTable({db})  \
                where symbol = `{symbol}, cycle = 1, tradingDay >= 2018.02.01 \
                group by tradingDay"
        data2 = s.run(sql2)
        res = pd.concat([data, data2])
        if len(res) == 0:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")

        return res['turnover_daily'].mean()

    def etf_avg_spread(self, symbol):
        if symbol[:2] == '51':
```

```
            sql = f"select symbol, date, time, askPrice1, bidPrice1  \
                    from loadTable('dfs://STOCK_SHL2_TAQ','SHL2_TAQ')  \
                    where symbol = `{symbol}"
        elif symbol[:2] == '15':
            sql = f"select symbol, date, time, askPrice1, bidPrice1 \
                    from loadTable('dfs://STOCK_SZL2_TAQ','SZL2_TAQ')  \
                    where symbol = `{symbol}"
        else:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")
        data = s.run(sql)
        if len(data) == 0:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")
        len_ttl = len(data)
        data.drop(data[(data['askPrice1'] == 0) | (data['bidPrice1'] == 0) | (data['askPrice1'] < data['bidPrice1'])].index,
                  inplace = True)
        data.reset_index(drop = True, inplace = True)
        len_clean = len(data)
        data['spread'] = data['askPrice1'] - data['bidPrice1']
        return data['spread'].mean(), data['spread'].quantile(0.1), data['spread'].quantile(0.9), (len_ttl - len_clean)/len_ttl

    def etf_vol(self, symbol, tag = 'close', window = 10 * 4 * 60):
        if symbol[:2] == '51':
            sql = f"select symbol, tradingDay, time, {tag} \
                    from loadTable('dfs://STOCK_SH_TRDMIN','STOCK_SH_TRDMIN') \
                    where symbol = `{symbol}, cycle = 1 \
                    order by tradingDay, time"
        elif symbol[:2] == '15':
            sql = f"select symbol, tradingDay, time, {tag} \
                    from loadTable('dfs://STOCK_SZ_TRDMIN','STOCK_SZ_TRDMIN') \
                    where symbol = `{symbol}, cycle = 1 \
                    order by tradingDay, time"
        else:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")
        data = s.run(sql)
        if len(data) == 0:
            raise AttributeError(f"{symbol} is not identified as a valid ETF symbol")

        # replace tag value of 0 with previous value (dealing with missing values)
        idx = np.where(data[tag] == 0)[0]
        for i in idx:
            data[tag].iloc[i] = data[tag].iloc[i-1]
        # calculate return
        data['return'] = np.log(data[tag]) - np.log(data[tag].shift(1))
        data.dropna(inplace = True)
        vol = data['return'].std() * np.sqrt(252 * 4 * 60)
        data['vol_window'] = data['return'].rolling(window = window).std() * np.sqrt(252 * 4 * 60)
        return vol, data[['symbol', 'tradingDay', 'time', 'vol_window']].dropna()
```

In [50]:

```
c = ETFClassification()
res = pd.DataFrame(columns = ['Symbol', 'Type', 'Daily Turover', 'Avg Spread', 'Avg Spread 10%',
'Avg Spread 90%', 'Clean Rate', 'Annualize Vol'])
idx = 0
vol_dict = {}
for k, v in c.etf_dict.items():
    for symbol in v:
        try:
            vol = c.etf_vol(symbol)
            turnover = c.etf_daily_avg_turnover(symbol)
            spread = c.etf_avg_spread(symbol)
            res.loc[idx] = [
                symbol,
                k,
                turnover,
                spread[0],
                spread[1],
                spread[2],
                spread[3],
                vol[0]
            ]
            vol_dict[symbol] = vol[1]
            idx += 1
        except AttributeError as e:
            print(e)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexing.py:670: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_with_indexer(indexer, value)
```

In [51]:

```
pd.set_option('max_columns',1000)
pd.set_option('max_row', 300)
res
```

Out[51]:

	Symbol	Type	Daily Turover	Avg Spread	Avg Spread 10%	Avg Spread 90%	Clean Rate	Annualize Vo
0	510300	BigCapIndex	1.283632e+09	0.001246	0.001	0.002	0.000787	0.22264
1	510330	BigCapIndex	1.708635e+08	0.004252	0.001	0.009	0.000458	0.29850
2	510050	BigCapIndex	1.089774e+09	0.001084	0.001	0.001	0.000383	0.25808
3	159919	BigCapIndex	2.706228e+08	0.002080	0.001	0.004	0.001919	0.42427
4	510310	BigCapIndex	3.178554e+07	0.002061	0.001	0.004	0.000939	0.34568
5	159949	SmallCapIndex	2.931953e+08	0.001108	0.001	0.001	0.000281	0.32955
6	510500	SmallCapIndex	4.505183e+08	0.002770	0.001	0.005	0.001144	0.57279
7	159915	SmallCapIndex	5.279195e+08	0.001181	0.001	0.002	0.003000	0.32013
8	512500	SmallCapIndex	6.928739e+07	0.005472	0.001	0.015	0.001535	0.44294
9	159968	SmallCapIndex	1.183576e+08	0.002932	0.001	0.005	0.003826	0.24511
10	515800	TotalMarketIndex	6.585641e+07	0.001235	0.001	0.002	0.000915	0.25522
11	512990	TotalMarketIndex	1.505395e+07	0.001863	0.001	0.003	0.001384	0.35184
12	512380	TotalMarketIndex	5.897763e+07	0.001175	0.001	0.002	0.000350	0.19993
13	512160	TotalMarketIndex	3.915058e+07	0.001245	0.001	0.002	0.000304	0.23306
14	512090	TotalMarketIndex	7.066792e+07	0.001194	0.001	0.002	0.000146	0.22281
15	159995	TMTSector	1.837037e+09	0.000999	0.001	0.001	0.000084	0.47685
16	512760	TMTSector	7.595528e+08	0.001086	0.001	0.001	0.004258	0.42115
17	515050	TMTSector	1.249226e+09	0.000999	0.001	0.001	0.003220	0.38249
18	159801	TMTSector	4.098801e+08	0.001028	0.001	0.001	0.000199	0.46859
19	512480	TMTSector	1.734476e+08	0.001208	0.001	0.002	0.004413	0.44590
20	512290	BioMedicalSector	8.450179e+07	0.001180	0.001	0.002	0.000302	0.27232
21	159992	BioMedicalSector	2.088485e+08	0.001022	0.001	0.001	0.000251	0.23743
22	512170	BioMedicalSector	6.277956e+07	0.001200	0.001	0.002	0.000437	0.27045
23	512010	BioMedicalSector	1.627210e+07	0.003938	0.001	0.010	0.001398	0.54990
24	159938	BioMedicalSector	1.068875e+07	0.001868	0.001	0.003	0.005573	0.42309
25	515000	TechTopic	6.185341e+08	0.001011	0.001	0.001	0.000376	0.31569
26	515750	TechTopic	1.197412e+08	0.001096	0.001	0.001	0.001206	0.35935
27	159807	TechTopic	6.047199e+07	0.001050	0.001	0.001	0.000489	0.24182
28	515860	TechTopic	3.405269e+07	0.001240	0.001	0.002	0.001338	0.32028
29	159987	TechTopic	2.512842e+07	0.001188	0.001	0.002	0.001722	0.31274
30	515030	NEVTopic	3.662280e+08	0.001002	0.001	0.001	0.000245	0.37419
31	515700	NEVTopic	4.334103e+08	0.001013	0.001	0.001	0.000235	0.36853
32	159806	NEVTopic	6.429906e+07	0.001079	0.001	0.001	0.000621	0.33183
33	512880	RealEstateSector	3.670658e+08	0.001063	0.001	0.001	0.002279	0.31151
34	512000	RealEstateSector	1.653046e+08	0.001103	0.001	0.001	0.002663	0.30527
35	512800	RealEstateSector	7.107327e+07	0.001115	0.001	0.002	0.000083	0.21687

	Symbol	Type	Daily Turover	Avg Spread	Avg Spread 10%	Avg Spread 90%	Clean Rate	Annualize Vo
36	512900	RealEstateSector	1.736284e+07	0.001356	0.001	0.002	0.001015	0.313307
37	159993	RealEstateSector	6.600373e+07	0.001067	0.001	0.001	0.000580	0.328281
38	159928	ConsumptionSector	1.316177e+07	0.003509	0.001	0.008	0.006177	0.468945
39	512690	ConsumptionSector	3.652211e+07	0.001144	0.001	0.002	0.002835	0.307973
40	515650	ConsumptionSector	3.468043e+07	0.001085	0.001	0.001	0.000988	0.280296
41	159996	ConsumptionSector	2.835535e+07	0.001145	0.001	0.002	0.000989	0.268888
42	510150	ConsumptionSector	2.547955e+06	0.044377	0.002	0.115	0.005616	0.738931
43	512660	MilitarySector	5.470121e+07	0.001084	0.001	0.001	0.000518	0.287674
44	512710	MilitarySector	1.051233e+08	0.001131	0.001	0.002	0.000784	0.286002
45	515210	PeriodicSector	3.623689e+07	0.001235	0.001	0.002	0.000400	0.241556
46	512400	PeriodicSector	1.545050e+07	0.001297	0.001	0.002	0.000952	0.340469
47	515220	PeriodicSector	2.390241e+07	0.001143	0.001	0.002	0.000604	0.223874
48	159966	SmartBeta	7.737577e+07	0.001138	0.001	0.002	0.000827	0.274917
49	159905	SmartBeta	1.121384e+07	0.003200	0.001	0.007	0.003206	0.465420
50	159967	SmartBeta	1.631938e+07	0.001256	0.001	0.002	0.000840	0.278642
51	510880	SmartBeta	7.017952e+07	0.002179	0.001	0.004	0.000972	0.298606
52	515180	SmartBeta	2.542231e+07	0.001075	0.001	0.001	0.000221	0.232158
53	515680	Other	3.755231e+07	0.002053	0.001	0.004	0.007729	0.406336
54	515900	Other	3.196034e+07	0.002380	0.001	0.003	0.003108	0.463242
55	159976	Other	2.837369e+07	0.001256	0.001	0.002	0.001852	0.307331
56	515600	Other	3.226328e+07	0.001938	0.001	0.003	0.001208	0.291668
57	159978	Other	2.086215e+07	0.001371	0.001	0.002	0.003500	0.226262
58	511010	Bond	3.943010e+08	0.013265	0.001	0.031	0.000339	0.112861
59	511260	Bond	2.092162e+07	0.066502	0.001	0.177	0.001243	0.116911
60	159972	Bond	2.941660e+07	0.063862	0.003	0.101	0.006535	0.102515
61	510900	QDII	6.252838e+08	0.001093	0.001	0.001	0.000236	0.273296
62	159920	QDII	2.685282e+08	0.001071	0.001	0.001	0.001916	0.230216
63	513050	QDII	4.579768e+07	0.001078	0.001	0.001	0.000110	0.265179
64	513090	QDII	1.153698e+08	0.001052	0.001	0.001	0.000543	0.221764
65	513500	QDII	4.437837e+07	0.001452	0.001	0.002	0.000417	0.249802
66	518880	Commodity	7.774713e+08	0.001088	0.001	0.001	0.000037	0.142202
67	159934	Commodity	1.237793e+08	0.001326	0.001	0.002	0.002013	0.219691
68	159937	Commodity	1.085085e+08	0.002052	0.001	0.003	0.003522	0.270522
69	518800	Commodity	8.050204e+07	0.001810	0.001	0.002	0.000205	0.200601
70	159980	Commodity	2.702668e+07	0.001293	0.001	0.002	0.009899	0.243887

In [52]:

```
from datetime import datetime
# 对齐时间戳
def combine_time(df,tag1 = 'tradingDay', tag2 = 'time'):
    new_idx = []
    for idx, row in df.iterrows():
        new_idx.append(datetime(row[tag1].year, row[tag1].month, row[tag1].day, row[tag2].hour,
row[tag2].minute, row[tag2].second))
    df['datetime'] = new_idx
    df.set_index('datetime',inplace = True)
    return df['vol_window']
```

In [53]:

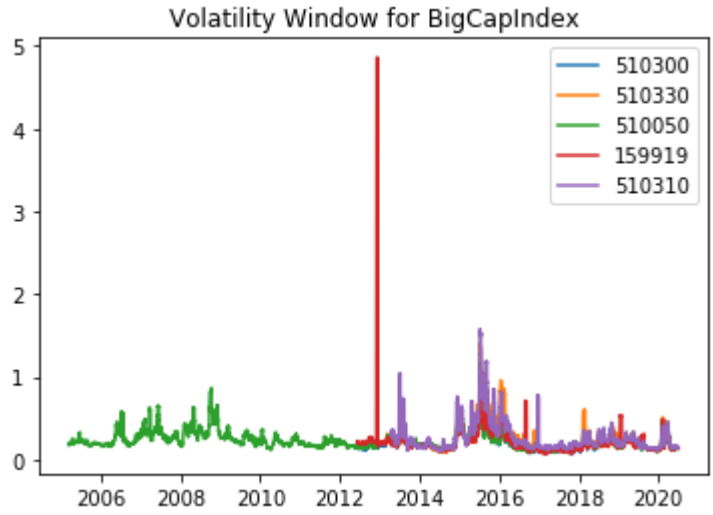
```
for k, v in c.etf_dict.items():
    for symbol in v:
        vol_dict[symbol] = combine_time(vol_dict[symbol])
```

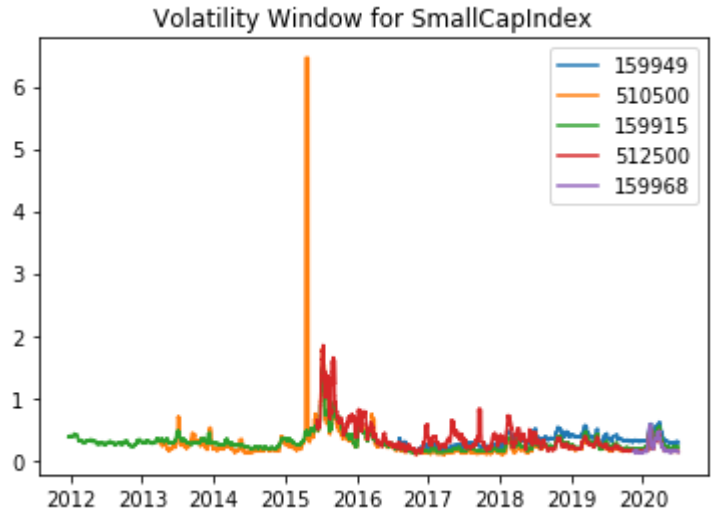
In [54]:

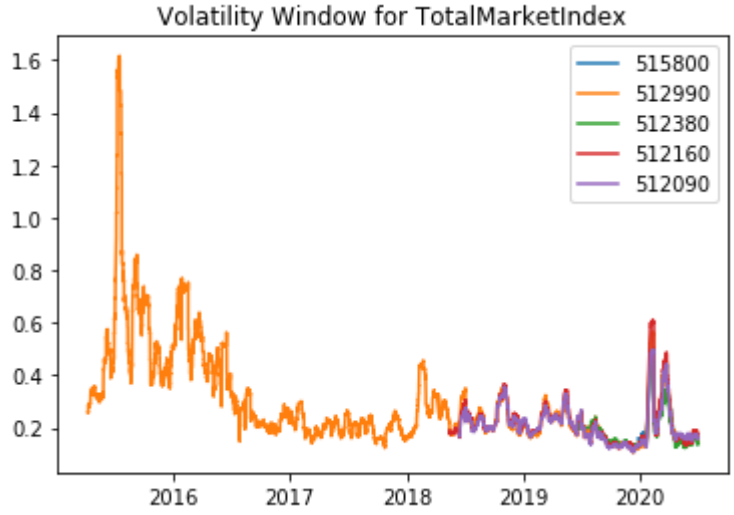
```
import matplotlib.pyplot as plt
```

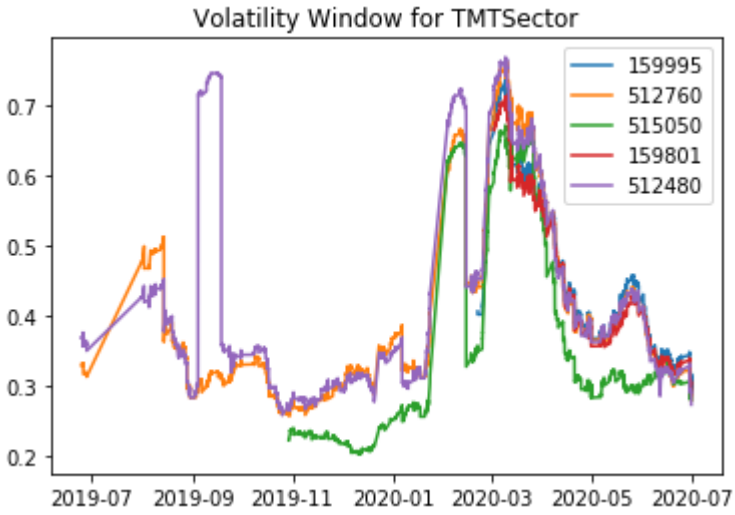
In [55]:

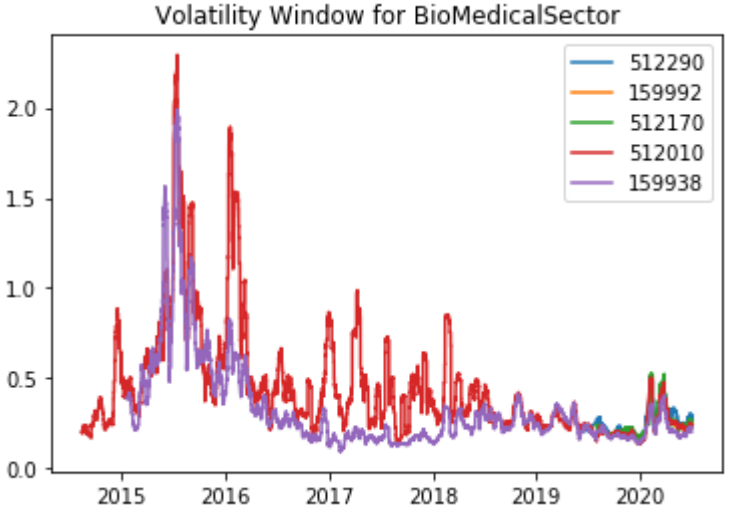
```
for k, v in c.etf_dict.items():
    for sym in v:
        plt.plot(vol_dict[sym], label = sym)
    plt.title(f"Volatility Window for {k}")
    plt.legend(loc = 'best')
    plt.show()
```

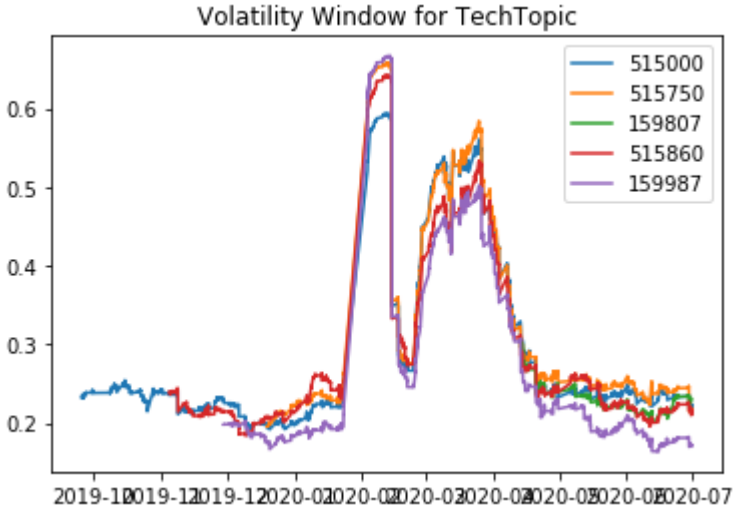

Volatility Window for BigCapIndex
510300
510330
510050
159919
510310
0
1
2
3
4
5
2006
2008
2010
2012
2014
2016
2018
2020

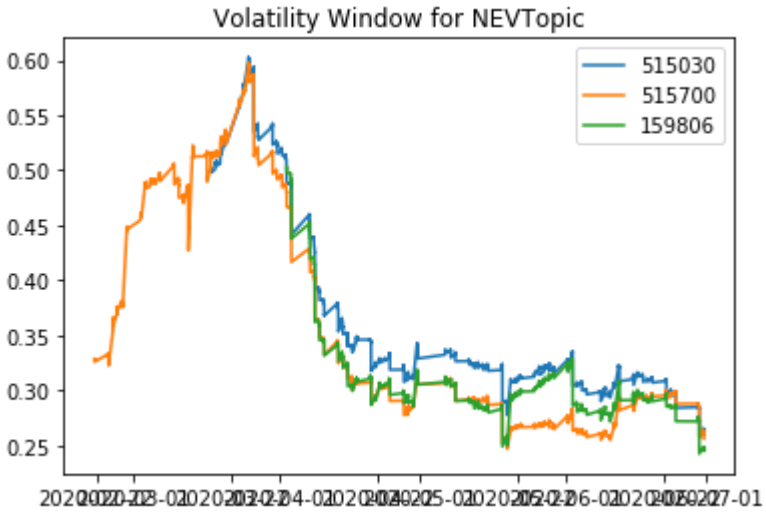

Volatility Window for SmallCapIndex
159949
510500
159915
512500
159968
0
1
2
3
4
5
6
2012
2013
2014
2015
2016
2017
2018
2019
2020

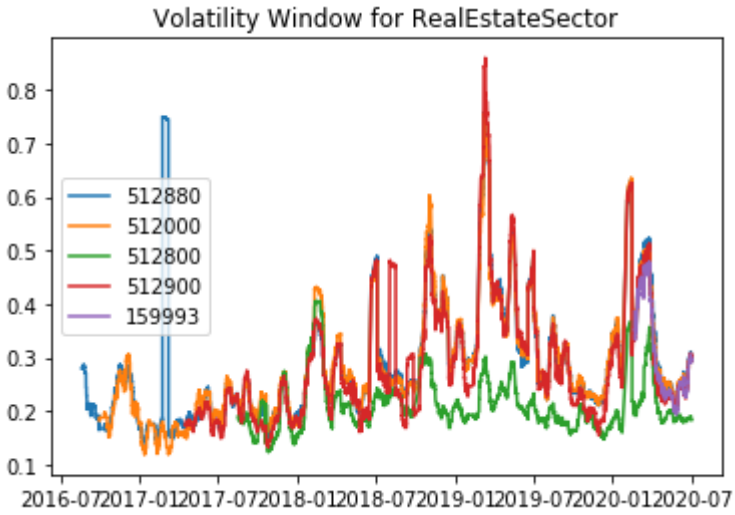

Volatility Window for TotalMarketIndex
515800
512990
512380
512160
512090
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
2016
2017
2018
2019
2020

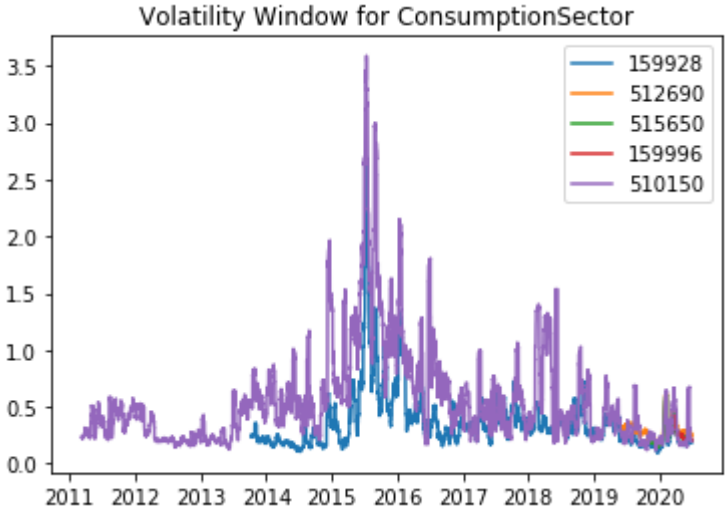

Volatility Window for TMTSector
159995
512760
515050
159801
512480
0.7
0.6
0.5
0.4
0.3
0.2
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07


Volatility Window for BioMedicalSector
512290
159992
512170
512010
159938
2.0
1.5
1.0
0.5
0.0
2015
2016
2017
2018
2019
2020

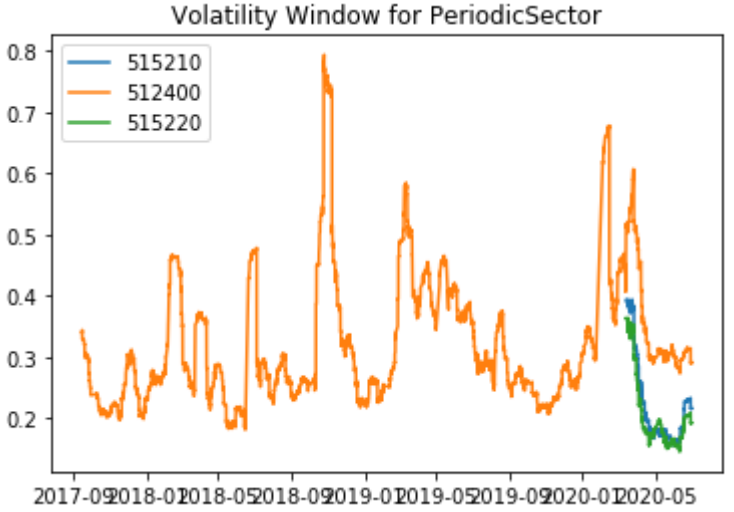

Volatility Window for TechTopic
515000
515750
159807
515860
159987
0.6
0.5
0.4
0.3
0.2

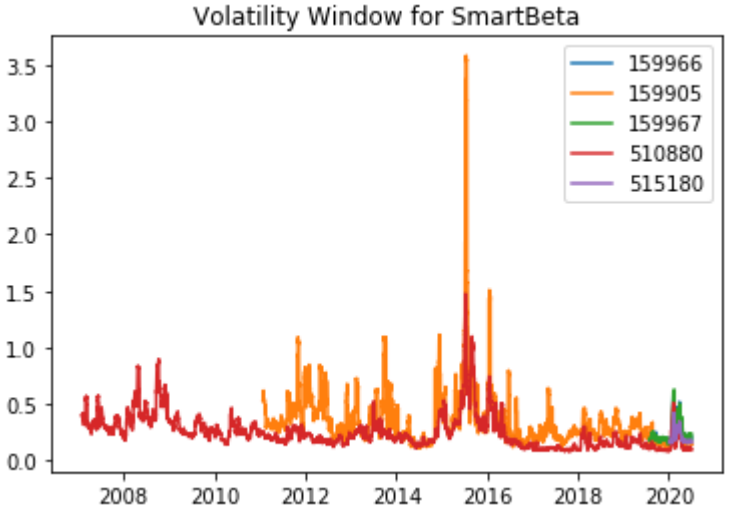

Volatility Window for NEVTopic
515030
515700
159806
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25

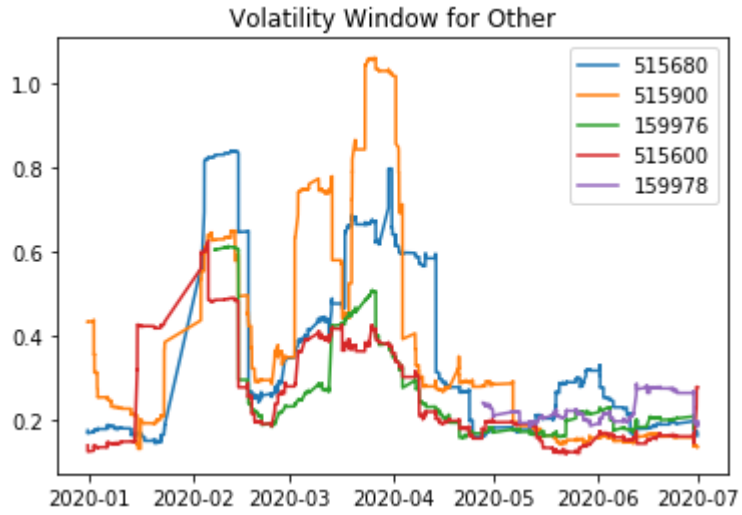

Volatility Window for RealEstateSector
512880
512000
512800
512900
159993
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07

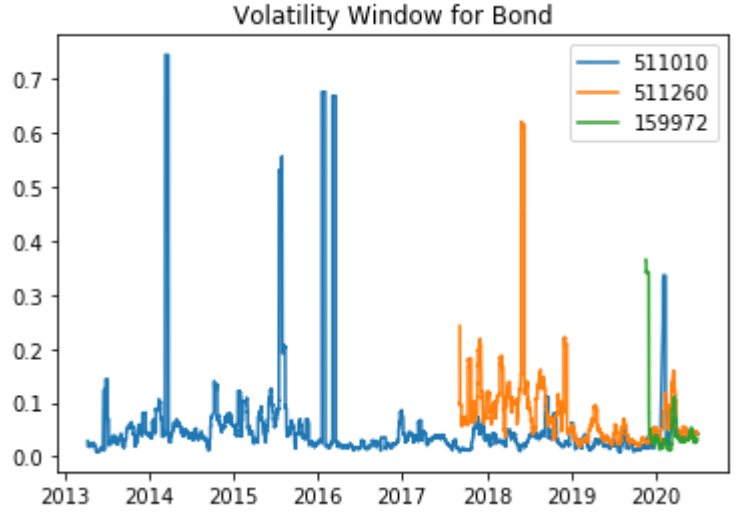

Volatility Window for ConsumptionSector
159928
512690
515650
159996
510150
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

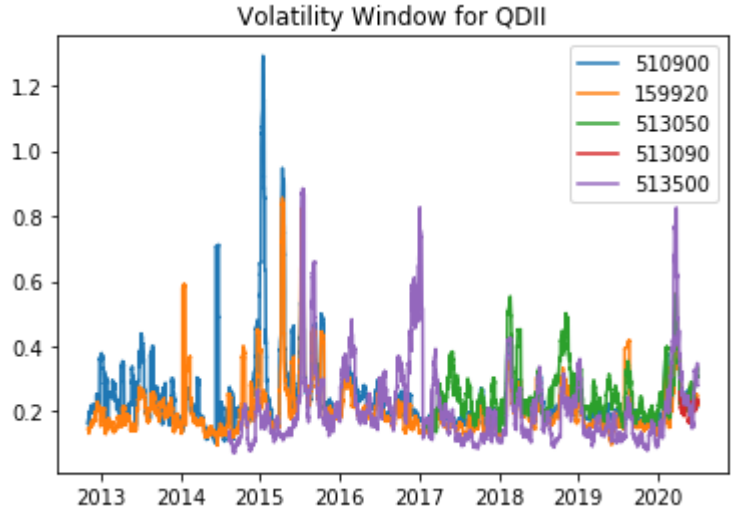

Volatility Window for MilitarySector
512660
512710
0.6
0.5
0.4
0.3
0.2
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07

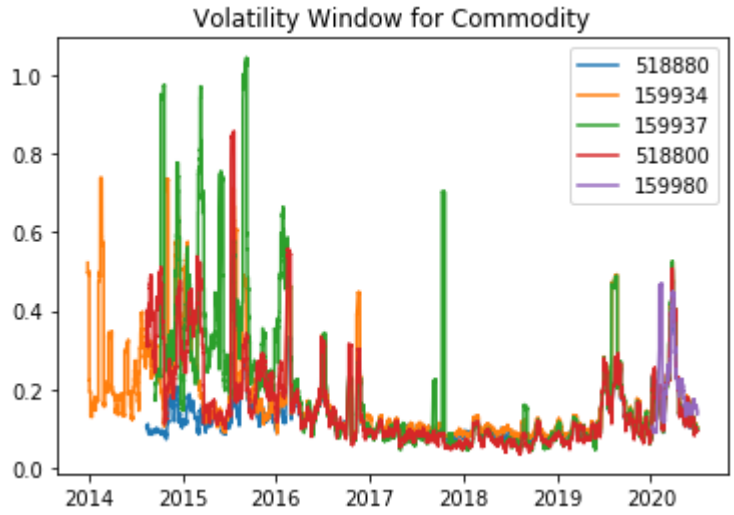

Volatility Window for PeriodicSector
515210
512400
515220
0.8
0.7
0.6
0.5
0.4
0.3
0.2
2017-09
2018-01
2018-05
2018-09
2019-01
2019-05
2019-09
2020-01
2020-05


Volatility Window for SmartBeta
159966
159905
159967
510880
515180
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2008
2010
2012
2014
2016
2018
2020


Volatility Window for Other
515680
515900
159976
515600
159978
1.0
0.8
0.6
0.4
0.2
2020-01
2020-02
2020-03
2020-04
2020-05
2020-06
2020-07


Volatility Window for Bond
511010
511260
159972
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2013
2014
2015
2016
2017
2018
2019
2020


Volatility Window for QDII
510900
159920
513050
513090
513500
1.2
1.0
0.8
0.6
0.4
0.2
2013
2014
2015
2016
2017
2018
2019
2020



In [56]:

```
np.save('vol_window.npy',vol_dict)
```

In [ ]:

```

```