**Project 1: High Frequency Data Analysis¶**

##### ***Shizheng Li¶***

In [1]:

**import** **pandas** **as** **pd**  
%**pylab** inline  
**import** **matplotlib.pyplot** **as** **plt**  
**import** **numpy** **as** **np**  
**import** **numba**

Populating the interactive namespace from numpy and matplotlib

In [2]:

**import** **sys**  
print(sys.version)  
print(pd.version.version)  
print(np.version.version)

3.4.3 |Anaconda 2.3.0 (64-bit)| (default, Mar 6 2015, 12:06:10) [MSC v.1600 64 bit (AMD64)]  
0.16.2  
1.9.2

In [3]:

rawcsv = pd.read\_csv('./sample.taq.csv', header = **None**, names = ['time', 'sym', 'type','d1','d2','d3','d4'],  
 nrows = 3139595, error\_bad\_lines=**True**, dtype={'d1': np.float64, 'd2': np.float64, 'd3':np.float64, 'd4':np.float64})

In [4]:

pd.isnull(rawcsv).any(0)

Out[4]:

time False  
sym False  
type False  
d1 False  
d2 False  
d3 False  
d4 False  
dtype: bool

No NaN in raw csv

In [5]:

rawcsv['sym'].unique()

Out[5]:

array(['SLB', 'JPM', 'NOV', 'WFC'], dtype=object)

starting from 3139596, four lines are exotic data, from 16:00:00, just ignore the remaining since we only care about 9:30 to 16:00

Caveat: Somehow the pandas CSV parser treats d1-d4 as string after a certain row, need to specify dtype to force casting. We need d2 to be floating point type because it could be ask price or trade size.

In [6]:

rawcsv

Out[6]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **time** | **sym** | **type** | **d1** | **d2** | **d3** | **d4** |
| **0** | 40003.448829 | SLB | Q | 101.00 | 128.37 | 200 | 100 |
| **1** | 40003.448927 | SLB | Q | 110.96 | 128.37 | 100 | 100 |
| **2** | 40014.967413 | SLB | Q | 110.95 | 128.37 | 100 | 100 |
| **3** | 40014.974791 | SLB | Q | 110.96 | 128.37 | 100 | 100 |
| **4** | 40014.986969 | SLB | Q | 110.95 | 128.37 | 100 | 100 |
| **5** | 40014.995795 | SLB | Q | 110.96 | 128.37 | 100 | 100 |
| **6** | 40014.996803 | SLB | Q | 110.95 | 128.37 | 100 | 100 |
| **7** | 40015.546104 | SLB | Q | 101.00 | 128.37 | 200 | 100 |
| **8** | 40015.546228 | SLB | Q | 110.96 | 128.37 | 100 | 100 |
| **9** | 40015.549694 | SLB | Q | 101.00 | 128.37 | 200 | 100 |
| **10** | 40019.519178 | SLB | Q | 101.00 | 128.37 | 300 | 100 |
| **11** | 40114.333994 | JPM | Q | 57.88 | 58.70 | 500 | 200 |
| **12** | 40134.683981 | JPM | Q | 57.88 | 58.70 | 1000 | 200 |
| **13** | 40135.239840 | JPM | Q | 57.26 | 58.70 | 500 | 200 |
| **14** | 40135.240799 | JPM | Q | 57.66 | 58.70 | 1000 | 200 |
| **15** | 40136.239031 | JPM | Q | 57.87 | 58.70 | 100 | 200 |
| **16** | 40136.240066 | JPM | Q | 57.88 | 58.70 | 1000 | 200 |
| **17** | 40218.479430 | SLB | Q | 104.95 | 128.37 | 100 | 100 |
| **18** | 40218.482962 | SLB | Q | 104.95 | 117.09 | 100 | 100 |
| **19** | 40218.547601 | SLB | Q | 104.95 | 117.09 | 200 | 100 |
| **20** | 40224.983465 | NOV | Q | 78.82 | 87.00 | 100 | 200 |
| **21** | 40426.256739 | WFC | Q | 50.96 | 53.20 | 300 | 5000 |
| **22** | 40427.205737 | WFC | Q | 50.96 | 52.37 | 300 | 100 |
| **23** | 41543.108368 | WFC | Q | 50.96 | 52.89 | 300 | 100 |
| **24** | 41543.156197 | WFC | Q | 50.96 | 53.20 | 300 | 5000 |
| **25** | 41544.196282 | WFC | Q | 50.96 | 52.91 | 300 | 100 |
| **26** | 41544.280252 | WFC | Q | 50.96 | 52.40 | 300 | 100 |
| **27** | 42721.788106 | JPM | Q | 57.88 | 58.70 | 500 | 200 |
| **28** | 42721.791973 | JPM | Q | 57.88 | 58.70 | 1000 | 200 |
| **29** | 42730.918512 | NOV | Q | 82.25 | 87.00 | 100 | 200 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **3139565** | 155959.849337 | JPM | Q | 58.90 | 58.91 | 41300 | 19500 |
| **3139566** | 155959.849339 | JPM | Q | 58.90 | 58.91 | 41300 | 19600 |
| **3139567** | 155959.849414 | JPM | Q | 58.90 | 58.91 | 41300 | 19100 |
| **3139568** | 155959.872641 | WFC | Q | 52.09 | 52.10 | 116200 | 10100 |
| **3139569** | 155959.888856 | JPM | Q | 58.90 | 58.91 | 41400 | 19100 |
| **3139570** | 155959.890287 | NOV | Q | 81.76 | 81.77 | 20500 | 5300 |
| **3139571** | 155959.890368 | NOV | Q | 81.76 | 81.77 | 20500 | 5400 |
| **3139572** | 155959.890382 | NOV | Q | 81.76 | 81.77 | 20500 | 5500 |
| **3139573** | 155959.890390 | NOV | Q | 81.76 | 81.77 | 20500 | 5600 |
| **3139574** | 155959.890426 | NOV | Q | 81.76 | 81.77 | 20500 | 6600 |
| **3139575** | 155959.890555 | NOV | Q | 81.76 | 81.77 | 20400 | 6600 |
| **3139576** | 155959.890605 | NOV | Q | 81.76 | 81.77 | 20400 | 6700 |
| **3139577** | 155959.890685 | NOV | Q | 81.76 | 81.77 | 20300 | 6700 |
| **3139578** | 155959.890746 | NOV | Q | 81.76 | 81.77 | 20400 | 6700 |
| **3139579** | 155959.891005 | NOV | Q | 81.76 | 81.77 | 20600 | 6700 |
| **3139580** | 155959.892217 | JPM | Q | 58.90 | 58.91 | 41500 | 19100 |
| **3139581** | 155959.895468 | NOV | Q | 81.76 | 81.77 | 20600 | 6800 |
| **3139582** | 155959.897000 | JPM | Q | 58.90 | 58.91 | 41600 | 19100 |
| **3139583** | 155959.901586 | SLB | Q | 110.32 | 110.34 | 26100 | 100 |
| **3139584** | 155959.906935 | WFC | Q | 52.09 | 52.10 | 135200 | 10100 |
| **3139585** | 155959.908438 | JPM | Q | 58.90 | 58.91 | 58300 | 19100 |
| **3139586** | 155959.908962 | NOV | Q | 81.76 | 81.77 | 32300 | 6800 |
| **3139587** | 155959.910291 | SLB | Q | 110.32 | 110.34 | 34900 | 100 |
| **3139588** | 155959.915648 | NOV | Q | 81.76 | 81.77 | 32400 | 6800 |
| **3139589** | 155959.923683 | JPM | Q | 58.90 | 58.91 | 58200 | 19100 |
| **3139590** | 155959.928341 | SLB | Q | 110.32 | 110.34 | 36700 | 100 |
| **3139591** | 155959.966927 | NOV | Q | 81.76 | 81.77 | 32300 | 6800 |
| **3139592** | 155959.994780 | JPM | Q | 58.90 | 58.91 | 58200 | 19200 |
| **3139593** | 155959.995358 | JPM | Q | 58.90 | 58.91 | 58300 | 19200 |
| **3139594** | 155959.995646 | NOV | Q | 81.76 | 81.77 | 32300 | 6900 |

3139595 rows × 7 columns

In [7]:

syms = rawcsv['sym'].unique() *# ['SLB', 'JPM', 'NOV', 'WFC']*

In [8]:

quotes = {sym: pd.DataFrame **for** sym **in** syms}  
trades = {sym: pd.DataFrame **for** sym **in** syms}

In [9]:

grouped = rawcsv.groupby(['sym','type'])

In [10]:

grouped.first() *#This removes sym and type column in grouped.get\_group*

Out[10]:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **time** | **d1** | **d2** | **d3** | **d4** |
| **sym** | **type** |  |  |  |  |  |
| **JPM** | **Q** | 40114.333994 | 57.88 | 58.70 | 500 | 200 |
| **T** | 43846.022625 | 58.70 | 200.00 | 5 | 0 |
| **NOV** | **Q** | 40224.983465 | 78.82 | 87.00 | 100 | 200 |
| **T** | 80001.738312 | 83.00 | 100.00 | 40 | 0 |
| **SLB** | **Q** | 40003.448829 | 101.00 | 128.37 | 200 | 100 |
| **T** | 85852.264515 | 111.12 | 48.00 | 5 | 0 |
| **WFC** | **Q** | 40426.256739 | 50.96 | 53.20 | 300 | 5000 |
| **T** | 72927.557545 | 51.62 | 1000.00 | 1 | 0 |

In [11]:

**for** sym **in** syms:  
 q = grouped.get\_group((sym, 'Q'))   
 *#only take between 9:30 and 16:00*  
 quotes[sym] = q[(q.time >= 93000) & (q.time <= 160000)] *#.drop(['sym', 'type'], axis = 1)*  
 t = grouped.get\_group((sym, 'T'))  
 trades[sym] = t[(t.time >= 93000) & (t.time <= 160000)] *#.drop(['sym', 'type'], axis = 1)*  
*#quotes and trades contains raw price, size data for each symbol*

In [12]:

quotes['NOV']

Out[12]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **d1** | **d2** | **d3** | **d4** | **time** |
| **1837** | 82.45 | 83.29 | 300 | 300 | 93000.089285 |
| **1838** | 82.46 | 83.29 | 300 | 300 | 93000.089287 |
| **1839** | 82.46 | 83.29 | 300 | 400 | 93000.090329 |
| **1840** | 82.46 | 83.29 | 400 | 400 | 93000.090331 |
| **1855** | 82.46 | 83.29 | 700 | 400 | 93000.188251 |
| **1856** | 82.46 | 83.29 | 700 | 700 | 93000.195435 |
| **1857** | 82.47 | 83.29 | 300 | 700 | 93000.201859 |
| **1858** | 82.47 | 83.29 | 400 | 700 | 93000.203892 |
| **1860** | 82.47 | 83.29 | 400 | 600 | 93000.218947 |
| **1899** | 82.47 | 83.27 | 400 | 300 | 93000.573910 |
| **1900** | 82.47 | 83.27 | 400 | 400 | 93000.574508 |
| **1917** | 82.47 | 83.26 | 400 | 300 | 93001.035896 |
| **1918** | 82.47 | 83.26 | 400 | 400 | 93001.039687 |
| **1919** | 82.47 | 83.25 | 400 | 300 | 93001.042070 |
| **1920** | 82.47 | 83.25 | 400 | 400 | 93001.042081 |
| **1921** | 82.48 | 83.25 | 300 | 400 | 93001.049981 |
| **1922** | 82.48 | 83.25 | 400 | 400 | 93001.050010 |
| **1923** | 82.48 | 83.24 | 400 | 300 | 93001.056186 |
| **1924** | 82.48 | 83.24 | 400 | 400 | 93001.057125 |
| **1931** | 82.49 | 83.24 | 300 | 400 | 93001.084838 |
| **1932** | 82.49 | 83.24 | 400 | 400 | 93001.085081 |
| **1935** | 82.49 | 83.24 | 400 | 100 | 93001.131615 |
| **1963** | 82.49 | 83.23 | 400 | 300 | 93002.001376 |
| **1965** | 82.49 | 83.23 | 400 | 400 | 93002.001959 |
| **1966** | 82.50 | 83.23 | 300 | 400 | 93002.001961 |
| **1967** | 82.50 | 83.22 | 300 | 300 | 93002.002327 |
| **1968** | 82.50 | 83.22 | 400 | 300 | 93002.002589 |
| **1971** | 82.51 | 83.21 | 300 | 300 | 93002.021220 |
| **1972** | 82.51 | 83.21 | 400 | 300 | 93002.021658 |
| **1973** | 82.51 | 83.21 | 400 | 400 | 93002.021658 |
| **...** | ... | ... | ... | ... | ... |
| **3139469** | 81.76 | 81.77 | 19500 | 6000 | 155959.699143 |
| **3139470** | 81.76 | 81.77 | 19500 | 5800 | 155959.699144 |
| **3139471** | 81.76 | 81.77 | 19500 | 5600 | 155959.699347 |
| **3139475** | 81.76 | 81.77 | 19600 | 5600 | 155959.699930 |
| **3139480** | 81.76 | 81.77 | 19600 | 5400 | 155959.700123 |
| **3139481** | 81.76 | 81.77 | 19800 | 5400 | 155959.700125 |
| **3139482** | 81.76 | 81.77 | 19900 | 5400 | 155959.701180 |
| **3139509** | 81.76 | 81.77 | 20100 | 5400 | 155959.716595 |
| **3139512** | 81.76 | 81.77 | 20100 | 5300 | 155959.717070 |
| **3139513** | 81.76 | 81.77 | 20200 | 5300 | 155959.717071 |
| **3139520** | 81.76 | 81.77 | 20200 | 5200 | 155959.724936 |
| **3139528** | 81.76 | 81.77 | 20000 | 5200 | 155959.760026 |
| **3139558** | 81.76 | 81.77 | 20200 | 5200 | 155959.837960 |
| **3139559** | 81.76 | 81.77 | 20400 | 5200 | 155959.838179 |
| **3139560** | 81.76 | 81.77 | 20500 | 5200 | 155959.838284 |
| **3139570** | 81.76 | 81.77 | 20500 | 5300 | 155959.890287 |
| **3139571** | 81.76 | 81.77 | 20500 | 5400 | 155959.890368 |
| **3139572** | 81.76 | 81.77 | 20500 | 5500 | 155959.890382 |
| **3139573** | 81.76 | 81.77 | 20500 | 5600 | 155959.890390 |
| **3139574** | 81.76 | 81.77 | 20500 | 6600 | 155959.890426 |
| **3139575** | 81.76 | 81.77 | 20400 | 6600 | 155959.890555 |
| **3139576** | 81.76 | 81.77 | 20400 | 6700 | 155959.890605 |
| **3139577** | 81.76 | 81.77 | 20300 | 6700 | 155959.890685 |
| **3139578** | 81.76 | 81.77 | 20400 | 6700 | 155959.890746 |
| **3139579** | 81.76 | 81.77 | 20600 | 6700 | 155959.891005 |
| **3139581** | 81.76 | 81.77 | 20600 | 6800 | 155959.895468 |
| **3139586** | 81.76 | 81.77 | 32300 | 6800 | 155959.908962 |
| **3139588** | 81.76 | 81.77 | 32400 | 6800 | 155959.915648 |
| **3139591** | 81.76 | 81.77 | 32300 | 6800 | 155959.966927 |
| **3139594** | 81.76 | 81.77 | 32300 | 6900 | 155959.995646 |

337970 rows × 5 columns

In [13]:

*#Find the crossed quotes and remove them, print out the row number and time stamp of the crossed quote*  
**for** sym **in** syms:  
 print(sym)  
 print("Before dropping:", quotes[sym].shape)  
 crossed\_idx = quotes[sym].index[(quotes[sym].d1 > quotes[sym].d2).nonzero()[0]]  
 print("Number of crossed quotes:", len(crossed\_idx))  
 print(quotes[sym].loc[crossed\_idx])   
 quotes[sym] = quotes[sym].drop(crossed\_idx)  
 print("After dropping:", quotes[sym].shape)

SLB  
Before dropping: (236957, 5)  
Number of crossed quotes: 350  
 d1 d2 d3 d4 time  
303767 111.04 111.02 100 200 100425.530288  
353529 111.07 111.06 200 100 100822.557583  
353530 111.07 111.06 300 100 100822.557590  
353546 111.07 111.06 400 100 100822.558821  
353549 111.07 111.06 500 100 100822.559653  
353551 111.07 111.06 400 100 100822.560250  
353553 111.07 111.06 300 100 100822.561251  
353555 111.07 111.06 400 100 100822.562725  
353556 111.07 111.06 500 100 100822.562729  
353557 111.07 111.06 400 100 100822.563297  
353558 111.07 111.06 300 100 100822.563375  
360254 111.13 111.12 100 600 100901.700467  
360255 111.13 111.12 100 500 100901.700472  
360256 111.13 111.12 200 500 100901.700483  
360262 111.13 111.12 200 200 100901.701793  
360264 111.13 111.12 200 100 100901.702337  
365014 111.24 111.23 100 100 100929.269700  
365015 111.24 111.23 300 100 100929.269720  
365016 111.24 111.23 200 100 100929.269721  
365017 111.24 111.23 100 100 100929.269727  
480902 111.23 111.22 100 100 102500.136328  
480907 111.23 111.22 200 100 102500.136522  
480908 111.23 111.22 300 100 102500.136523  
480909 111.23 111.22 400 100 102500.136558  
480910 111.23 111.22 500 100 102500.136565  
480912 111.23 111.22 600 100 102500.136692  
480920 111.24 111.23 100 100 102500.137137  
480922 111.24 111.23 100 100 102500.137252  
486193 111.23 111.22 100 300 102545.859756  
486209 111.23 111.22 100 100 102545.863408  
... ... ... ... ... ...  
2512722 110.39 110.38 300 100 145421.557327  
2512723 110.39 110.38 400 100 145421.557345  
2548593 110.43 110.42 200 100 145751.570361  
2548594 110.43 110.42 200 300 145751.570366  
2548595 110.43 110.42 200 400 145751.570370  
2548596 110.43 110.42 200 500 145751.570371  
2548597 110.43 110.42 200 400 145751.570372  
2548634 110.43 110.42 100 400 145751.571799  
2548677 110.43 110.42 100 500 145751.574112  
2548721 110.42 110.41 400 100 145751.577308  
2548736 110.42 110.41 100 100 145751.578104  
2704560 110.39 110.38 100 200 151116.755320  
2704561 110.39 110.38 100 300 151116.755334  
2773600 110.52 110.51 100 100 151748.199747  
2910630 110.63 110.62 100 200 153241.916667  
2910678 110.63 110.62 300 200 153241.917786  
2910830 110.64 110.62 100 200 153241.920768  
2910832 110.64 110.62 100 100 153241.920795  
2910862 110.64 110.63 100 300 153241.921609  
2910903 110.64 110.63 100 200 153241.923356  
2910947 110.64 110.63 100 100 153241.925108  
2910961 110.64 110.63 100 200 153241.926014  
2934907 110.63 110.62 100 100 153616.011595  
3001833 110.58 110.57 200 200 154619.366686  
3001841 110.58 110.57 200 100 154619.367504  
3040809 110.56 110.55 100 100 155058.518688  
3040815 110.56 110.55 300 100 155058.519032  
3040821 110.56 110.55 500 100 155058.519375  
3040822 110.56 110.55 600 100 155058.519377  
3040823 110.56 110.55 700 100 155058.519382  
  
[350 rows x 5 columns]  
After dropping: (236607, 5)  
JPM  
Before dropping: (1362550, 5)  
Number of crossed quotes: 93  
 d1 d2 d3 d4 time  
2542 58.98 58.97 300 100 93015.211617  
3224 59.04 59.03 1300 800 93033.032942  
3225 59.04 59.03 1100 800 93033.033017  
3226 59.04 59.03 900 800 93033.033058  
103625 59.14 59.13 4900 100 94443.295504  
103626 59.14 59.13 4800 100 94443.295508  
103633 59.14 59.13 3600 100 94443.295806  
226547 59.10 59.09 300 100 95842.499863  
226548 59.10 59.09 500 100 95842.499867  
226549 59.10 59.09 600 100 95842.499867  
226550 59.10 59.09 1000 100 95842.499868  
226551 59.10 59.09 1500 100 95842.499869  
226552 59.10 59.09 1100 100 95842.499872  
289929 58.94 58.93 200 100 100311.981249  
289930 58.94 58.93 200 200 100311.981255  
289931 58.94 58.93 200 300 100311.981260  
289932 58.94 58.93 200 200 100311.981269  
289938 58.94 58.93 200 300 100311.983354  
289939 58.94 58.93 200 700 100311.983365  
289940 58.94 58.93 200 800 100311.983367  
966117 58.91 58.90 100 200 112046.341151  
966123 58.91 58.90 200 200 112046.342016  
966124 58.91 58.90 600 200 112046.342020  
1187025 58.71 58.70 100 700 115446.807406  
1187026 58.71 58.70 100 500 115446.807412  
1187027 58.71 58.70 100 800 115446.807417  
1187028 58.71 58.70 100 700 115446.807429  
1187029 58.71 58.70 100 600 115446.807436  
1190524 58.69 58.68 1300 100 115513.296169  
1190560 58.69 58.68 1200 100 115513.298718  
... ... ... ... ... ...  
2609576 58.86 58.85 100 1800 150253.429889  
2609577 58.86 58.85 100 2000 150253.429892  
2609580 58.86 58.85 100 1800 150253.429967  
2609581 58.86 58.85 100 2000 150253.429977  
2609582 58.86 58.85 100 1800 150253.429978  
2609584 58.86 58.85 100 1600 150253.430046  
2609595 58.86 58.85 100 1800 150253.430515  
2609596 58.86 58.85 100 2000 150253.430516  
2609607 58.86 58.85 100 2100 150253.431362  
2609608 58.86 58.85 100 2200 150253.431367  
2609609 58.86 58.85 100 2400 150253.431367  
2609610 58.86 58.85 100 2200 150253.431369  
2609611 58.86 58.85 100 2400 150253.431372  
2609612 58.86 58.85 100 2300 150253.431376  
2609618 58.86 58.85 100 2200 150253.432200  
2609619 58.86 58.85 100 2100 150253.432205  
2609620 58.86 58.85 100 2000 150253.432208  
2609621 58.86 58.85 100 1900 150253.432210  
2609624 58.86 58.85 100 1800 150253.432648  
2609625 58.86 58.85 100 1700 150253.432657  
2609627 58.86 58.85 100 1600 150253.432663  
2609629 58.86 58.85 100 1200 150253.432765  
2609632 58.86 58.85 100 1000 150253.433088  
2609635 58.86 58.85 100 900 150253.433204  
2609637 58.86 58.85 100 800 150253.433245  
2609638 58.86 58.85 100 700 150253.433249  
2609639 58.86 58.85 100 500 150253.433309  
2609642 58.86 58.85 100 300 150253.433381  
2609646 58.86 58.85 100 200 150253.433457  
2664380 58.85 58.84 400 100 150730.262590  
  
[93 rows x 5 columns]  
After dropping: (1362457, 5)  
NOV  
Before dropping: (337970, 5)  
Number of crossed quotes: 39  
 d1 d2 d3 d4 time  
8285 82.92 82.89 100 100 93118.012438  
110171 83.09 83.08 100 100 94531.524723  
110175 83.09 83.08 100 200 94531.524838  
110176 83.09 83.08 100 300 94531.524840  
110177 83.09 83.08 100 400 94531.524842  
110178 83.09 83.08 100 500 94531.525242  
329670 82.51 82.50 200 100 100613.463772  
329671 82.51 82.50 100 100 100613.463898  
814333 82.16 82.15 500 100 110257.397480  
814334 82.16 82.15 500 200 110257.397491  
814336 82.16 82.15 500 300 110257.397718  
814337 82.16 82.15 500 200 110257.397731  
814338 82.16 82.15 500 300 110257.397837  
814339 82.16 82.15 500 200 110257.397924  
814340 82.16 82.15 500 300 110257.398850  
814341 82.16 82.15 500 400 110257.398853  
814342 82.16 82.15 500 500 110257.399224  
814343 82.16 82.15 500 600 110257.399598  
814344 82.16 82.15 500 500 110257.399601  
814345 82.16 82.15 500 400 110257.399727  
814346 82.16 82.15 500 500 110257.399730  
814347 82.16 82.15 500 400 110257.399784  
814348 82.16 82.15 500 300 110257.399788  
814349 82.16 82.15 500 200 110257.400402  
814351 82.16 82.15 500 100 110257.402114  
814352 82.16 82.15 300 100 110257.411223  
814353 82.16 82.15 100 100 110257.411261  
814355 82.16 82.15 100 100 110257.411557  
814356 82.16 82.15 200 100 110257.411604  
814357 82.16 82.15 100 100 110257.411656  
907039 82.17 82.16 100 100 111329.611246  
1219360 82.13 82.12 100 100 115902.989974  
1219361 82.13 82.12 100 300 115902.989979  
2501753 81.97 81.96 200 100 145310.785638  
2501754 81.97 81.96 200 300 145310.785885  
2501755 81.97 81.96 200 400 145310.785890  
2501781 81.97 81.96 200 600 145310.787547  
2501783 81.97 81.96 200 700 145310.787764  
2501804 81.97 81.96 200 500 145310.792816  
After dropping: (337931, 5)  
WFC  
Before dropping: (1002169, 5)  
Number of crossed quotes: 71  
 d1 d2 d3 d4 time  
59505 52.12 52.11 1400 100 93942.798144  
59507 52.12 52.11 1400 200 93942.798189  
1846010 51.94 51.93 1900 100 134250.744313  
1846011 51.94 51.93 1800 100 134250.744376  
1846012 51.94 51.93 1700 100 134250.744388  
1846014 51.94 51.93 1200 100 134250.744433  
1846015 51.94 51.93 1100 100 134250.744456  
1846017 51.94 51.93 1000 100 134250.744665  
1846018 51.94 51.93 900 100 134250.744762  
1846020 51.94 51.93 700 100 134250.744866  
1846021 51.94 51.93 600 100 134250.744906  
1846022 51.94 51.93 500 100 134250.744962  
1846023 51.94 51.93 400 100 134250.745002  
1915154 52.01 52.00 500 100 135827.099636  
1967288 52.01 52.00 900 100 140623.026594  
1967289 52.01 52.00 1300 100 140623.026599  
1967290 52.01 52.00 900 100 140623.026611  
1967291 52.01 52.00 500 100 140623.026616  
1967292 52.01 52.00 900 100 140623.026625  
1967300 52.01 52.00 800 100 140623.027492  
1967356 52.01 52.00 900 100 140623.032922  
1967357 52.01 52.00 1000 100 140623.032922  
1967358 52.01 52.00 1400 100 140623.032927  
1967572 52.02 52.01 800 100 140623.054859  
1972255 52.01 52.00 400 100 140633.061288  
1972256 52.01 52.00 1500 100 140633.061293  
1972257 52.01 52.00 1300 100 140633.061305  
1972258 52.01 52.00 1400 100 140633.061307  
1972259 52.01 52.00 2000 100 140633.061309  
1972264 52.01 52.00 1900 100 140633.062216  
... ... ... ... ... ...  
1972278 52.01 52.00 2400 100 140633.064062  
1972279 52.01 52.00 1900 100 140633.064063  
1972280 52.01 52.00 1800 100 140633.064066  
1972282 52.01 52.00 1900 100 140633.066690  
1972287 52.01 52.00 2100 100 140633.067531  
1972288 52.01 52.00 2500 100 140633.067539  
1972289 52.01 52.00 2700 100 140633.067548  
1972294 52.01 52.00 2800 100 140633.068336  
1972367 52.02 52.01 100 100 140633.073557  
1972368 52.02 52.01 1200 100 140633.073560  
1972398 52.02 52.01 1500 100 140633.077463  
1972417 52.02 52.01 1600 100 140633.081427  
1972421 52.02 52.01 1700 100 140633.081932  
1972422 52.02 52.01 1800 100 140633.081934  
1972423 52.02 52.01 1700 100 140633.081942  
1972427 52.02 52.01 1800 100 140633.082957  
1972444 52.02 52.01 1900 100 140633.086054  
2117658 52.23 52.22 4300 300 141717.392876  
2117674 52.24 52.23 100 600 141717.393539  
2117676 52.24 52.23 100 700 141717.393651  
2117677 52.24 52.23 100 500 141717.393651  
2117679 52.24 52.23 100 400 141717.393668  
2117680 52.24 52.23 100 300 141717.393672  
2121287 52.27 52.26 3400 100 141729.774294  
2121291 52.27 52.26 3500 100 141729.774523  
2121295 52.27 52.26 7200 100 141729.774685  
2121304 52.28 52.27 100 300 141729.775102  
2121306 52.28 52.27 200 300 141729.775166  
2203855 52.21 52.20 600 600 142516.948836  
2203856 52.21 52.20 1100 600 142516.949329  
  
[71 rows x 5 columns]  
After dropping: (1002098, 5)

In [14]:

**for** sym **in** syms:  
 print(sym)   
 crossed\_idx = quotes[sym].index[(quotes[sym].d1 > quotes[sym].d2).nonzero()[0]]  
 print(len(crossed\_idx))

SLB  
0  
JPM  
0  
NOV  
0  
WFC  
0

In [15]:

*#First take a look at five sigma outliers, note the index printed is the index in each individual data frame*  
**for** sym **in** syms:  
 print(sym)  
 print(np.abs(((quotes[sym].d1 - quotes[sym].d1.mean() ) > (5 \* quotes[sym].d1.std()))).nonzero())  
 print(np.abs(((quotes[sym].d2 - quotes[sym].d2.mean() ) > (5 \* quotes[sym].d2.std()))).nonzero())  
 print(np.abs(((trades[sym].d1 - trades[sym].d1.mean() ) > (5 \* trades[sym].d1.std()))).nonzero())

SLB  
(array([], dtype=int64),)  
(array([68869], dtype=int64),)  
(array([], dtype=int64),)  
JPM  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
(array([64699], dtype=int64),)  
NOV  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
WFC  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
(array([], dtype=int64),)

These two are clearly bad data. Highly suspect this is human manipulated :).

In [16]:

quotes['SLB'].iloc[68868:68871] *#focus on 68869*

Out[16]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **d1** | **d2** | **d3** | **d4** | **time** |
| **983105** | 110.60 | 110.62 | 400 | 100 | 112331.981147 |
| **983106** | 110.60 | 210.62 | 100 | 100 | 112331.981148 |
| **983111** | 110.61 | 110.62 | 100 | 100 | 112332.179731 |

In [17]:

quotes['SLB'] = quotes['SLB'].drop(quotes['SLB'].index[68869])

In [18]:

trades['JPM'].iloc[64698:64701] *#focus on 64699*

Out[18]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **d1** | **d2** | **d3** | **d4** | **time** |
| **3107167** | 58.92 | 21 | 61 | 0 | 155800.010907 |
| **3107168** | 78.92 | 1200 | 1 | 0 | 155800.010908 |
| **3107471** | 58.92 | 100 | 11 | 0 | 155800.611183 |

In [19]:

trades['JPM'] = trades['JPM'].drop(trades['JPM'].index[64699])

In [20]:

*#Now take a look at 3 sigma, we actaully accept them*  
**for** sym **in** syms:  
 print(sym)  
 print(np.abs(((quotes[sym].d1 - quotes[sym].d1.mean() ) > (3 \* quotes[sym].d1.std()))).nonzero())  
 print(np.abs(((quotes[sym].d2 - quotes[sym].d2.mean() ) > (3 \* quotes[sym].d2.std()))).nonzero())  
 print(np.abs(((trades[sym].d1 - trades[sym].d1.mean() ) > (3 \* trades[sym].d1.std()))).nonzero())

SLB  
(array([ 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597,  
 598, 599, 607, 608, 609, 610, 611, 612, 613, 614, 615,  
 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626,  
 627, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657,  
 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668,  
 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679,  
 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690,  
 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701,  
 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712,  
 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723,  
 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734,  
 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745,  
 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756,  
 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767,  
 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778,  
 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789,  
 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800,  
 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811,  
 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822,  
 838, 840, 916, 984, 1041, 1042, 1043, 1044, 1045, 1046, 1047,  
 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058,  
 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069,  
 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080,  
 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091,  
 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102,  
 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113,  
 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124,  
 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135,  
 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1183, 1184,  
 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195,  
 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206,  
 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217,  
 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1227, 1228, 1230, 1231,  
 1241, 1242, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295,  
 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306,  
 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317,  
 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328,  
 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339,  
 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350,  
 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361,  
 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372,  
 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383,  
 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1636, 1637, 1638,  
 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1662, 1663, 1664,  
 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675,  
 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 2643, 2644, 2645,  
 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659,  
 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2671, 2672,  
 2673, 2678, 2679, 2680, 2683, 2684, 2685, 2691, 2695, 2700, 2707,  
 2708, 2709, 2710, 2711, 2712, 2713, 2714, 2715, 2716, 2717, 2718,  
 2719, 2720, 2721, 2722, 2723, 2724, 2725, 2726, 2727, 2728, 2729,  
 2730, 2731, 2732, 2733, 2734, 2735, 2736, 2737, 2738, 2739, 2740,  
 2741, 2742, 2743, 2744, 2745, 2746, 2747, 2748, 2749, 2750, 2751,  
 2752, 2753, 2754, 2755, 2756, 2757, 2758, 2759, 2760, 2761, 2762,  
 2763, 2764, 2765, 2766, 2767, 2768, 2769, 2770, 2771, 2772, 2773,  
 2774, 2775, 2776, 2777, 2778, 2779, 2780, 2781, 2782, 2783, 2784,  
 2785, 2786, 2787, 2788, 2789, 2790, 2791, 2792, 2793, 2794, 2795,  
 2796, 2797, 2798, 2799, 2800, 2801, 2802, 2803, 2804, 2805, 2806,  
 2807, 2808, 2809, 2810, 2811, 2812, 2813, 2814, 2815, 2816, 2817,  
 2818, 2819, 2820, 2821, 2822, 2823, 2824, 2825, 2826, 2827, 2828,  
 2829, 2830, 2831, 2832, 2833, 2834, 2835, 2836, 2837, 2838, 2839,  
 2840, 2841, 2842, 2843, 2844, 2845, 2846, 2847, 2848, 2849, 2850,  
 2851, 2852, 2853, 2854, 2855, 2856, 2857, 2858, 2859, 2860, 2861,  
 2862, 2863, 2864, 2865, 2866, 2867, 2868, 2869, 2870, 2871, 2872,  
 2873, 2874, 2875, 2876, 2877, 2878, 2879, 2880, 2881, 2882, 2883,  
 2884, 2885, 2886, 2887, 2888, 2889, 2890, 2891, 2892, 2893, 2894,  
 2895, 2896, 2897, 2898, 2899, 2900, 2901, 2902, 2903, 2904, 2905,  
 2906, 2907, 2908, 2909, 2910, 2911, 2912], dtype=int64),)  
(array([ 0, 1, 2, ..., 2950, 2951, 2959], dtype=int64),)  
(array([151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163,  
 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176,  
 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189,  
 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202,  
 231, 232, 233, 238, 239, 240, 242, 243, 244, 245, 246, 247, 248,  
 251, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,  
 339, 340, 386, 754], dtype=int64),)  
JPM  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
NOV  
(array([ 1564, 1565, 1566, ..., 15389, 15390, 15391], dtype=int64),)  
(array([ 0, 1, 2, ..., 13628, 13629, 13630], dtype=int64),)  
(array([ 0, 1, 371, 373, 374, 375, 376, 377, 378, 379, 380,  
 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391,  
 392, 393, 394, 395, 396, 397, 398, 399, 401, 404, 407,  
 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418,  
 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,  
 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440,  
 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451,  
 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462,  
 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473,  
 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484,  
 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495,  
 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506,  
 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517,  
 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528,  
 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539,  
 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550,  
 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561,  
 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572,  
 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583,  
 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594,  
 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605,  
 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616,  
 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627,  
 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638,  
 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649,  
 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660,  
 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671,  
 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682,  
 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693,  
 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704,  
 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715,  
 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726,  
 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737,  
 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748,  
 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759,  
 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770,  
 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781,  
 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792,  
 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803,  
 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814,  
 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825,  
 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836,  
 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847,  
 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858,  
 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869,  
 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880,  
 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891,  
 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902,  
 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913,  
 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924,  
 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935,  
 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946,  
 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957,  
 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968,  
 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979,  
 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990,  
 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001,  
 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012,  
 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023,  
 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034,  
 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045,  
 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056,  
 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067,  
 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078,  
 1079, 1080, 1081, 1082, 1083, 1084], dtype=int64),)  
WFC  
(array([], dtype=int64),)  
(array([], dtype=int64),)  
(array([], dtype=int64),)

In [21]:

**def** plot3Sigma(series):  
 plt.plot(series)  
 m = series.mean()  
 plt.axhline(m, color = 'r')  
 std3 = series.std() \* 3  
 plt.axhline(m + std3, color = 'g')  
 plt.axhline(m - std3, color = 'g')

Based on plots below, we can see that the only short period time where bid, ask or trade prices go beyond three sigma is during market open, these are normal market behaviors and should not be excluded, we keep these samples.

In [22]:

plot3Sigma(quotes['SLB'].d1)

In [23]:

plot3Sigma(quotes['SLB'].d2)

In [24]:

plot3Sigma(trades['SLB'].d1)

In [25]:

plot3Sigma(quotes['NOV'].d1)

In [26]:

plot3Sigma(quotes['NOV'].d2)

In [27]:

plot3Sigma(trades['NOV'].d1)

In [21]:

*#Returns epoch time in us. The numba JIT compiler generates machine code, making it much faster than python*  
@numba.vectorize(['uint64(float64)'], target='cpu')  
**def** toEpochUFunc(ts):  
 ts = ts \* 1000000   
 h = ts // 10000000000  
 ts = ts % 10000000000  
 m = ts // 100000000  
 ts = ts % 100000000  
 s = ts // 1000000  
 us = ts % 1000000  
 *#assumes the date is Aug 21, 2015, since we do not know the date. Does not matter.*  
 **return** (1440115200 + h \* 3600 + m \* 60 + s) \* 1000000 + us

In [16]:

*#How fast to convert all 3 millions timestamps to pandas DateTimeIndex*  
%**time** epons = toEpochUFunc(rawcsv.time.values) \* 1000 #350 ms to convert to epoch ns  
%**time** pd.to\_datetime(epons.astype('datetime64[ns]')) #10ms to create pandas DateTimeIndex

Wall time: 350 ms  
Wall time: 10 ms

Out[16]:

DatetimeIndex(['2015-08-21 04:00:03.448829', '2015-08-21 04:00:03.448927',  
 '2015-08-21 04:00:14.967413', '2015-08-21 04:00:14.974791',  
 '2015-08-21 04:00:14.986969', '2015-08-21 04:00:14.995795',  
 '2015-08-21 04:00:14.996803', '2015-08-21 04:00:15.546104',  
 '2015-08-21 04:00:15.546228', '2015-08-21 04:00:15.549694',   
 ...  
 '2015-08-21 15:59:59.908438', '2015-08-21 15:59:59.908962',  
 '2015-08-21 15:59:59.910291', '2015-08-21 15:59:59.915648',  
 '2015-08-21 15:59:59.923683', '2015-08-21 15:59:59.928341',  
 '2015-08-21 15:59:59.966927', '2015-08-21 15:59:59.994780',  
 '2015-08-21 15:59:59.995358', '2015-08-21 15:59:59.995646'],  
 dtype='datetime64[ns]', length=3139595, freq=None, tz=None)

In [22]:

*#Note that we need return and price (for minutes price data)*  
quoteRet = {}  
quotePrice = {}  
**for** sym **in** syms:  
 quotePrice[sym] = pd.DataFrame(index = pd.to\_datetime(toEpochUFunc(quotes[sym].time.values) \* 1000),   
 data=(quotes[sym].d1.values + quotes[sym].d2.values)/2, columns=['Mid'])  
 quoteRet[sym] = quotePrice[sym].pct\_change().dropna()

In [23]:

quotePrice['JPM'] *#Just to give an example*

Out[23]:

|  |  |
| --- | --- |
|  | **Mid** |
| **2015-08-21 09:30:00.061424** | 59.025 |
| **2015-08-21 09:30:00.061426** | 59.025 |
| **2015-08-21 09:30:00.062633** | 59.015 |
| **2015-08-21 09:30:00.062657** | 59.010 |
| **2015-08-21 09:30:00.067675** | 59.010 |
| **2015-08-21 09:30:00.067682** | 59.010 |
| **2015-08-21 09:30:00.070331** | 59.010 |
| **2015-08-21 09:30:00.071750** | 59.010 |
| **2015-08-21 09:30:00.084090** | 59.035 |
| **2015-08-21 09:30:00.087116** | 59.040 |
| **2015-08-21 09:30:00.107481** | 59.040 |
| **2015-08-21 09:30:00.114949** | 59.040 |
| **2015-08-21 09:30:00.264041** | 59.040 |
| **2015-08-21 09:30:00.282607** | 59.040 |
| **2015-08-21 09:30:00.298378** | 59.040 |
| **2015-08-21 09:30:00.332400** | 59.040 |
| **2015-08-21 09:30:00.368959** | 59.040 |
| **2015-08-21 09:30:00.480720** | 59.035 |
| **2015-08-21 09:30:00.480722** | 59.030 |
| **2015-08-21 09:30:00.481771** | 59.030 |
| **2015-08-21 09:30:00.481771** | 59.030 |
| **2015-08-21 09:30:00.481773** | 59.030 |
| **2015-08-21 09:30:00.482698** | 59.030 |
| **2015-08-21 09:30:00.482698** | 59.030 |
| **2015-08-21 09:30:00.482700** | 59.030 |
| **2015-08-21 09:30:00.483698** | 59.025 |
| **2015-08-21 09:30:00.484557** | 59.025 |
| **2015-08-21 09:30:00.484558** | 59.025 |
| **2015-08-21 09:30:00.485480** | 59.025 |
| **2015-08-21 09:30:00.485489** | 59.025 |
| **...** | ... |
| **2015-08-21 15:59:59.768377** | 58.905 |
| **2015-08-21 15:59:59.768419** | 58.905 |
| **2015-08-21 15:59:59.768656** | 58.905 |
| **2015-08-21 15:59:59.768768** | 58.905 |
| **2015-08-21 15:59:59.769042** | 58.905 |
| **2015-08-21 15:59:59.769097** | 58.905 |
| **2015-08-21 15:59:59.769202** | 58.905 |
| **2015-08-21 15:59:59.769225** | 58.905 |
| **2015-08-21 15:59:59.769300** | 58.905 |
| **2015-08-21 15:59:59.769527** | 58.905 |
| **2015-08-21 15:59:59.769957** | 58.905 |
| **2015-08-21 15:59:59.769958** | 58.905 |
| **2015-08-21 15:59:59.791375** | 58.905 |
| **2015-08-21 15:59:59.797425** | 58.905 |
| **2015-08-21 15:59:59.800773** | 58.905 |
| **2015-08-21 15:59:59.810710** | 58.905 |
| **2015-08-21 15:59:59.842871** | 58.905 |
| **2015-08-21 15:59:59.843408** | 58.905 |
| **2015-08-21 15:59:59.843900** | 58.905 |
| **2015-08-21 15:59:59.848947** | 58.905 |
| **2015-08-21 15:59:59.849337** | 58.905 |
| **2015-08-21 15:59:59.849339** | 58.905 |
| **2015-08-21 15:59:59.849414** | 58.905 |
| **2015-08-21 15:59:59.888856** | 58.905 |
| **2015-08-21 15:59:59.892217** | 58.905 |
| **2015-08-21 15:59:59.897000** | 58.905 |
| **2015-08-21 15:59:59.908438** | 58.905 |
| **2015-08-21 15:59:59.923683** | 58.905 |
| **2015-08-21 15:59:59.994780** | 58.905 |
| **2015-08-21 15:59:59.995358** | 58.905 |

1362457 rows × 1 columns

In [24]:

tradeRet = {}  
**for** sym **in** syms:  
 tradeRet[sym] = pd.DataFrame(index = pd.to\_datetime(toEpochUFunc(trades[sym].time.values) \* 1000),   
 data=trades[sym].d1.values, columns=['Trd']).pct\_change().dropna()

In [25]:

*#Autocorrelation of mid quote returns*  
lags = range(1,10)  
**for** sym **in** syms:   
 ac = list(map(**lambda** lag: quoteRet[sym].Mid.autocorr(lag), lags))  
 plt.figure()  
 plt.plot(lags,ac)  
 plt.title(sym)

In [26]:

*#Autocorrelation of trade returns*  
**for** sym **in** syms:  
 ac = list(map(**lambda** lag: tradeRet[sym].Trd.autocorr(lag), lags))  
 plt.figure()  
 plt.plot(lags,ac)  
 plt.title(sym)

Trade return has stronger negative autocorrelation than mid quote returns. I suspect this is because trades happan between bid and ask prices, bouncing between bid and ask prices.

Quote return autocorrelation is a bit hard to explain to me... And I even suspect if it makes sense to compute quote by quote mid quote return autocorrelation, because lots of time the quote price does not change and return is just zero...

Minutes return and correlation

In [27]:

*#Here I sample mid quote to minutes. Note that we need to consider case when there is no data in that minute, here I just do simple things: just fill use previous minute price*  
min\_ret = {}  
**for** sym **in** syms:  
 *#This gives a Series, rather than a DF. dropna() is to drop the first time stamp NaN.*  
 min\_ret[sym] = quotePrice[sym].resample('1min', how='median').fillna(method='ffill').pct\_change().dropna()['Mid']

In [28]:

min\_retDF = pd.DataFrame(data=min\_ret)

In [29]:

min\_retDF

Out[29]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **JPM** | **NOV** | **SLB** | **WFC** |
| **2015-08-21 09:31:00** | 0.001947 | -0.001085 | 0.001568 | 0.003191 |
| **2015-08-21 09:32:00** | 0.000084 | 0.002111 | 0.000268 | 0.000867 |
| **2015-08-21 09:33:00** | 0.000169 | 0.003371 | -0.000134 | 0.001541 |
| **2015-08-21 09:34:00** | 0.000676 | 0.000360 | -0.000671 | 0.000288 |
| **2015-08-21 09:35:00** | -0.000169 | 0.000660 | -0.000045 | 0.000096 |
| **2015-08-21 09:36:00** | -0.000338 | -0.000240 | 0.000000 | 0.000000 |
| **2015-08-21 09:37:00** | 0.000000 | -0.000839 | 0.000582 | 0.000384 |
| **2015-08-21 09:38:00** | 0.000169 | -0.000660 | -0.000626 | 0.000961 |
| **2015-08-21 09:39:00** | 0.000507 | -0.001921 | -0.001521 | 0.000384 |
| **2015-08-21 09:40:00** | -0.000169 | -0.001203 | -0.001703 | -0.000576 |
| **2015-08-21 09:41:00** | -0.000338 | -0.000361 | -0.000359 | -0.000576 |
| **2015-08-21 09:42:00** | -0.000844 | 0.000060 | 0.001662 | -0.000192 |
| **2015-08-21 09:43:00** | -0.000338 | -0.000602 | 0.000471 | -0.000769 |
| **2015-08-21 09:44:00** | 0.000000 | 0.000723 | -0.000022 | -0.000192 |
| **2015-08-21 09:45:00** | -0.000169 | 0.001084 | 0.000493 | -0.000577 |
| **2015-08-21 09:46:00** | 0.000254 | 0.000602 | -0.000851 | 0.000577 |
| **2015-08-21 09:47:00** | 0.000592 | 0.000481 | -0.000538 | 0.000192 |
| **2015-08-21 09:48:00** | -0.000507 | -0.000120 | -0.000763 | -0.000577 |
| **2015-08-21 09:49:00** | 0.000676 | -0.000240 | -0.000584 | 0.000385 |
| **2015-08-21 09:50:00** | -0.000169 | -0.000962 | -0.000314 | 0.000000 |
| **2015-08-21 09:51:00** | -0.000338 | 0.000481 | 0.000000 | -0.000577 |
| **2015-08-21 09:52:00** | -0.001014 | -0.001203 | -0.001348 | -0.000962 |
| **2015-08-21 09:53:00** | 0.000508 | -0.000482 | 0.000585 | 0.000963 |
| **2015-08-21 09:54:00** | -0.000169 | -0.000362 | 0.000899 | 0.000385 |
| **2015-08-21 09:55:00** | 0.000338 | 0.000723 | 0.000674 | 0.000192 |
| **2015-08-21 09:56:00** | 0.000000 | -0.000181 | -0.000045 | -0.000192 |
| **2015-08-21 09:57:00** | 0.000000 | -0.000482 | -0.000135 | 0.000000 |
| **2015-08-21 09:58:00** | -0.001014 | -0.001206 | -0.000449 | -0.000962 |
| **2015-08-21 09:59:00** | -0.001015 | -0.000483 | -0.001348 | -0.001155 |
| **2015-08-21 10:00:00** | 0.000000 | -0.000906 | -0.000450 | 0.000386 |
| **...** | ... | ... | ... | ... |
| **2015-08-21 15:30:00** | 0.000509 | 0.000672 | 0.000317 | 0.000384 |
| **2015-08-21 15:31:00** | -0.000170 | 0.000427 | 0.000498 | -0.000192 |
| **2015-08-21 15:32:00** | 0.000509 | -0.000122 | 0.000543 | 0.000384 |
| **2015-08-21 15:33:00** | 0.000254 | 0.000122 | 0.000316 | 0.000192 |
| **2015-08-21 15:34:00** | 0.000254 | 0.000000 | -0.000271 | 0.000192 |
| **2015-08-21 15:35:00** | 0.000000 | 0.000000 | -0.000361 | 0.000000 |
| **2015-08-21 15:36:00** | -0.000169 | -0.000244 | 0.000181 | 0.000000 |
| **2015-08-21 15:37:00** | 0.000000 | -0.000366 | 0.000136 | -0.000192 |
| **2015-08-21 15:38:00** | 0.000000 | -0.000366 | -0.000181 | 0.000000 |
| **2015-08-21 15:39:00** | 0.000000 | -0.000122 | -0.000090 | 0.000192 |
| **2015-08-21 15:40:00** | 0.000169 | 0.000000 | -0.000181 | 0.000192 |
| **2015-08-21 15:41:00** | -0.000254 | 0.000000 | -0.000271 | -0.000383 |
| **2015-08-21 15:42:00** | 0.000085 | 0.000000 | 0.000995 | 0.000000 |
| **2015-08-21 15:43:00** | 0.000000 | 0.000000 | -0.000587 | 0.000000 |
| **2015-08-21 15:44:00** | 0.000000 | -0.000366 | 0.000000 | 0.000000 |
| **2015-08-21 15:45:00** | -0.000508 | 0.000122 | -0.000136 | -0.000383 |
| **2015-08-21 15:46:00** | -0.000424 | 0.000122 | -0.000181 | 0.000000 |
| **2015-08-21 15:47:00** | 0.000085 | 0.000000 | -0.000045 | 0.000192 |
| **2015-08-21 15:48:00** | 0.000170 | 0.000000 | -0.000226 | 0.000000 |
| **2015-08-21 15:49:00** | -0.000339 | -0.000244 | -0.000362 | 0.000000 |
| **2015-08-21 15:50:00** | -0.000254 | -0.000244 | 0.000090 | 0.000000 |
| **2015-08-21 15:51:00** | 0.000085 | 0.000000 | 0.000090 | 0.000192 |
| **2015-08-21 15:52:00** | 0.000000 | -0.000244 | -0.000181 | 0.000000 |
| **2015-08-21 15:53:00** | -0.000170 | -0.000367 | -0.000090 | -0.000192 |
| **2015-08-21 15:54:00** | 0.000000 | 0.000550 | -0.000181 | -0.000192 |
| **2015-08-21 15:55:00** | -0.000339 | 0.000183 | -0.000091 | -0.000192 |
| **2015-08-21 15:56:00** | 0.000339 | -0.000244 | 0.000000 | 0.000000 |
| **2015-08-21 15:57:00** | -0.000170 | -0.000244 | -0.000091 | -0.000192 |
| **2015-08-21 15:58:00** | -0.000170 | -0.000428 | -0.000091 | -0.000384 |
| **2015-08-21 15:59:00** | -0.000170 | -0.000183 | -0.000679 | -0.000192 |

389 rows × 4 columns

In [30]:

*#Correlation matrix of minute returns*  
min\_corr = min\_retDF.corr()  
min\_corr

Out[30]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **JPM** | **NOV** | **SLB** | **WFC** |
| **JPM** | 1.000000 | 0.418408 | 0.545826 | 0.750968 |
| **NOV** | 0.418408 | 1.000000 | 0.547980 | 0.398392 |
| **SLB** | 0.545826 | 0.547980 | 1.000000 | 0.502081 |
| **WFC** | 0.750968 | 0.398392 | 0.502081 | 1.000000 |

JPM and WFC both in finance industry, exhibits high correlation as expected. NOV and SLV both in oil/NG industry, but the correlation is not as high as JPM and WFC.