Day2

February 26, 2021

0.1 Day2

[8]:
```
import pandas as pd
import numpy as np

import plotly.offline as pyo
import plotly.graph_objects as go
```

0.1.1 Advances in Machine Learning - Ex 2.1

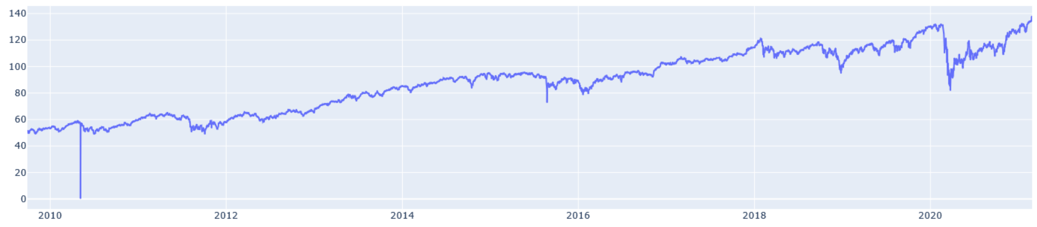
On a series of E-mini S&P 500 futures tick data: **(a)** Form tick, volume, and dollar bars. Use the ETF trick to deal with the roll. **(b)** Count the number of bars produced by tick, volume, and dollar bars on a weekly basis. Plot a time series of that bar count. What bar type produces the most stable weekly count? Why? **(c)** Compute the serial correlation of returns for the three bar types. What bar method has the lowest serial correlation? **(d)** Partition the bar series into monthly subsets. Compute the variance of returns for every subset of every bar type. Compute the variance of those variances. What method exhibits the smallest variance of variances? **(e)** Apply the Jarque-Bera normality test on returns from the three bar types. What method achieves the lowest test statistic?

The data can be downloaded from Kibot. It presents the format “Date,Time,Price,Bid,Ask,Size”

[10]:
```
data_path = "D:\Dataset\IVE_tickbidask.txt"

def preprocess_data(data_p):

    data = pd.read_csv(data_p, header=None,␣
↪names=['day','time','price','bid','ask','volume'])

    # we're interested on the date, so we can merge day/time
    data['date'] = pd.to_datetime(data['day'] + data['time'], format='%m/%d/
↪%Y%H:%M:%S')
    data.drop(columns=['day','time'], inplace=True)

    # needed for dollar bars
    data['dollar_volume'] = data['price']*data['volume']

    # as a time series, we should have the date as index
    data = data.set_index('date')
    data = data.drop_duplicates()

    return data

df = preprocess_data(data_path)
df.sample()
```

[10]:
```
                        price     bid     ask  volume  dollar_volume
date
2017-07-25 10:33:09  106.3556  106.35  106.36     483     51369.7548
```

[12]:
```
fig = go.Figure()

# Add traces
fig.add_trace(go.Scatter(x=df.index, y=df['price'], name='price'))

fig.show()
```



a. Form tick, volume, and dollar bars. Use the ETF trick to deal with the roll.

[14]:
```
"""
    tick bars:
        the sample variables will be extracted whenever a pre-defined number of␣
↪transactions takes place (e.g. 500 ticks)

    volume bars:
        the sample variables will be extracted whenever a pre-defined number of␣
↪units have been exchanged (e.g. 500 units)

    dollar bars:
        the sample variables will be extracted whenever a pre-defined market␣
↪value have been exchanged
"""
```

```python
def tick_bars(data, thresh):
    return data.iloc[::thresh]

def volume_bars(data, thresh):
    new_df = data.reset_index()

    ids = []
    vols = []
    cumulative_volume = 0

    for ind, volume in new_df.volume.items():
        cumulative_volume = cumulative_volume + volume

        if cumulative_volume > thresh:
            # we can sample
            ids.append(ind)
            vols.append(cumulative_volume)
            cumulative_volume = 0

    # take just the samples with their cumulative volume
    v_bars = new_df.loc[ids]
    v_bars.loc[ids, 'cum_volume'] = vols

    return v_bars.set_index('date')

def dollar_bars(data, thresh):

    new_df = data.reset_index()

    ids = []
    doll = []
    cumulative_dollars = 0

    for ind, dollars in new_df.dollar_volume.items():
        cumulative_dollars = cumulative_dollars + dollars

        if cumulative_dollars > thresh:
            # we can sample
            ids.append(ind)
            doll.append(cumulative_dollars)
            cumulative_dollars = 0

    # take just the samples with their cumulative volume
    d_bars = new_df.loc[ids]
    d_bars.loc[ids, 'cum_dollars'] = doll

    return d_bars.set_index('date')
```

```
[16]: tick_b = tick_bars(df, 10)
      volume_b = volume_bars(df, 1000)
      dollars_b = dollar_bars(df, 100000)
```

b. Count the number of bars produced by tick, volume, and dollar bars on a weekly basis. Plot a time series of that bar count. What bar type produces the most stable weekly count? Why?

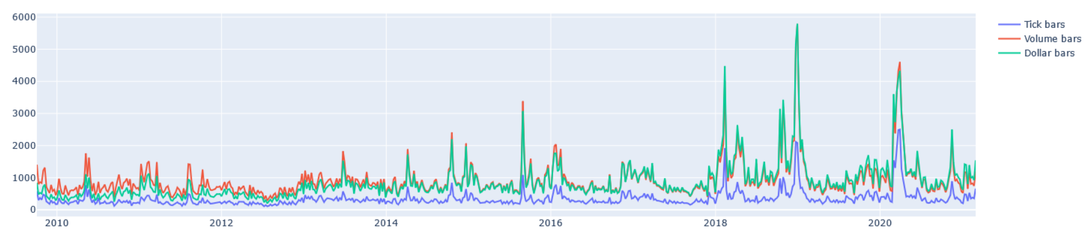
```
[18]: print(f"tick bars\t: {len(tick_b)}")
      print(f"volume bars\t: {len(volume_b)}")
      print(f"dollars bars\t: {len(dollars_b)}")
```

```
tick bars       : 203907
volume bars     : 551616
dollars bars    : 519252
```

```
[28]: tick_b_weekly = tick_b.resample('1w').count()
      volume_b_weekly = volume_b.resample('1w').count()
      dollars_b_weekly = dollars_b.resample('1w').count()
```

```
[29]: # Set notebook mode to work in offline
      pyo.init_notebook_mode()
```

```
[30]: fig = go.Figure()

      # Add traces
      fig.add_trace(go.Scatter(x=tick_b_weekly.index, y=tick_b_weekly['price'],␣
       ↪name='Tick bars'))
      fig.add_trace(go.Scatter(x=volume_b_weekly.index, y=volume_b_weekly['price'],␣
       ↪name='Volume bars'))
      fig.add_trace(go.Scatter(x=dollars_b_weekly.index, y=dollars_b_weekly['price'],␣
       ↪name='Dollar bars'))

      fig.show()
```



(c). Compute the serial correlation of returns for the three bar types. What bar method has the lowest serial correlation?

Benefit of using returns, versus prices, is normalization: measuring all variables in a comparable metric, thus enabling evaluation of analytic relationships amongst two or more variables despite originating from price series of unequal values > * https://quantivity.wordpress.com/2011/02/21/why-log-returns/ > * https://numpy.org/doc/stable/reference/generated/numpy.diff.html

[31]:
```
tick_log_returns = np.log(tick_b_weekly.price).diff().dropna()
vol_log_returns = np.log(volume_b_weekly.price).diff().dropna()
doll_log_returns = np.log(dollars_b_weekly.price).diff().dropna()

# Pearson correlation between the Series and its shifted self
print(f"tick\t: {tick_log_returns.autocorr()}")
print(f"volume\t: {vol_log_returns.autocorr()}")
print(f"dollars\t: {doll_log_returns.autocorr()}")
```

```
tick    : -0.3117047173249058
volume  : -0.3168114350161693
dollars : -0.326946524542962
```

- **(d)** Partition the bar series into monthly subsets. Compute the variance of returns for every subset of every bar type. Compute the variance of those variances. What method exhibits the smallest variance of variances?

[32]:
```
monthly_tick_var = tick_log_returns.resample('1M').var()
monthly_vol_var = vol_log_returns.resample('1M').var()
monthly_dollar_var = doll_log_returns.resample('1M').var()

print(f"Tick bars : {monthly_tick_var.var()}")
print(f"Volume bars   : {monthly_vol_var.var()}")
print(f"Dollar bars : {monthly_dollar_var.var()}")

## check later
```

```
Tick bars : 0.017765622179951635
Volume bars   : 0.01586229751891348
Dollar bars : 0.015393143038869895
```

(e) Apply the Jarque-Bera normality test on returns from the three bar types. What method achieves the lowest test statistic?

[33]:
```
from scipy.stats import jarque_bera
jb_value_tick, _ = jarque_bera(tick_log_returns)
jb_value_vol, _ = jarque_bera(vol_log_returns)
jb_value_dollar, _ = jarque_bera(doll_log_returns)
```

[34]:
```
print(f"Tick bars : {jarque_bera(tick_log_returns)[0]}")
print(f"Volume bars   : {jarque_bera(vol_log_returns)[0]}")
print(f"Dollar bars : {jarque_bera(doll_log_returns)[0]}")
```

```
Tick bars : 66.09189266712339
```

```
Volume bars  : 25.490368941053287
Dollar bars : 19.746611900225947
```

Inspiration : https://github.com/fernandodelacalle/adv-financial-ml-marcos-exercises