cltai9145 / research

Code | Pull requests | Actions | Projects | Security | Insights

master

research / Chapter2 / 2019-04-11_OP_Dollar-Imbalance-Bars.ipynb

Jackal08 fix latex f04bb38 · 5 years ago

370 lines (370 loc) · 179 KB

Preview | Code | Blame | Raw

- By: Proskurin Oleksandr
- Email: proskurinolexandr@gmail.com
- Reference: Advances in Financial Machine Learning, Marcos Lopez De Prado, pg 30, https://towardsdatascience.com/financial-machine-learning-part-0-bars-745897d4e4ba

In [10]:
```
from IPython.display import Image
```

Imbalance bars generation algorithm

Let's discuss imbalance bars generation on example of volume imbalance bars. As it is described in Advances in Financial Machine Learning book:

First let's define what is the tick rule:

For any given t, where p_t is the price associated with t and v_t is volume, the tick rule b_t is defined as:

$$b_t = \begin{cases} b_{t-1}, & \text{if } \Delta p_t = 0 \\ |\Delta p_t| / \Delta p_t, & \text{if } \Delta p_t \neq 0 \end{cases}$$

Tick rule is used as a proxy of trade direction, however, some data providers already provide customers with tick direction, in this case we don't need to calculate tick rule, just use the provided tick direction instead.

Cumulative volume imbalance from 1 to T is defined as:

$$\theta_t = \sum_{t=1}^{T} b_t * v_t$$

T is the time when the bar is sampled.

Next we need to define $E_0[T]$ as expected number of ticks, the book suggests to use EWMA of expected number of ticks from previously generated bars. Let's introduce the first hyperparameter for imbalance bars generation: **num_prev_bars** which corresponds to window used for EWMA calculation.

Here we face the problem of first bar generation, because we don't know expected number of ticks with no bars generated. To solve this we introduce the second hyperparameter: **expected_num_ticks_init** which corresponds to initial guess for expected number of ticks before the first imbalance bar is generated.

Bar is sampled when:

$$|\theta_t| >= E_0[T] * [2v^+ - E_0[v_t]]$$

To estimate $2v^+ - E_0[v_t]$ (expected imbalance) we simply calculate EWMA of volume imbalance from previous bars, that is why we need to store volume imbalances in *imbalance array*, the window for estimation is either **expected_num_ticks_init** before the first bar is sampled, or expected number of ticks($E_0[T]$) * **num_prev_bars** when the first bar is generated.

Note that when we have at least one imbalance bar generated we update $2v^+ - E_0[v_t]$ only when the next bar is sampled not on every trade observed

Algorithm logic

Now we have understood the logic of imbalance bar generation, let's understand how the process looks in details

```
num_prev_bars = 3
expected_num_ticks_init = 100000
expected_num_ticks = expected_num_ticks_init
cum_theta = 0
num_ticks = 0
imbalance_array = []
imbalance_bars = []
bar_length_array = []
for row in data.rows:
    #track high,low,close, volume info
    num_ticks += 1
    tick_rule = get_tick_rule(price, prev_price)
    volume_imbalance = tick_rule * row['volume']
    imbalance_array.append(volume_imbalance)
    cum_theta += volume_imbalance
    if len(imbalance_bars) == 0 and len(imbalance_array) >=
expected_num_ticks_init:
        expected_imbalance = ewma(imbalance_array,
window=expected_num_ticks_init)

    if abs(cum_theta) >= expected_num_ticks *
abs(expected_imbalance):
        bar = form_bar(open, high, low, close, volume)
        imbalance_bars.append(bar)
        bar_length_array.append(num_ticks)
        cum_theta, num_ticks = 0, 0
        expected_num_ticks = ewma(bar_lenght_array,
window=num_prev_bars)
        expected_imbalance = ewma(imbalance_array, window =
num_prev_bars * expected_num_ticks)
```

Note that in algorithm pseudo-code we reset θ_t when bar is formed, in our case the formula for θ_t is:

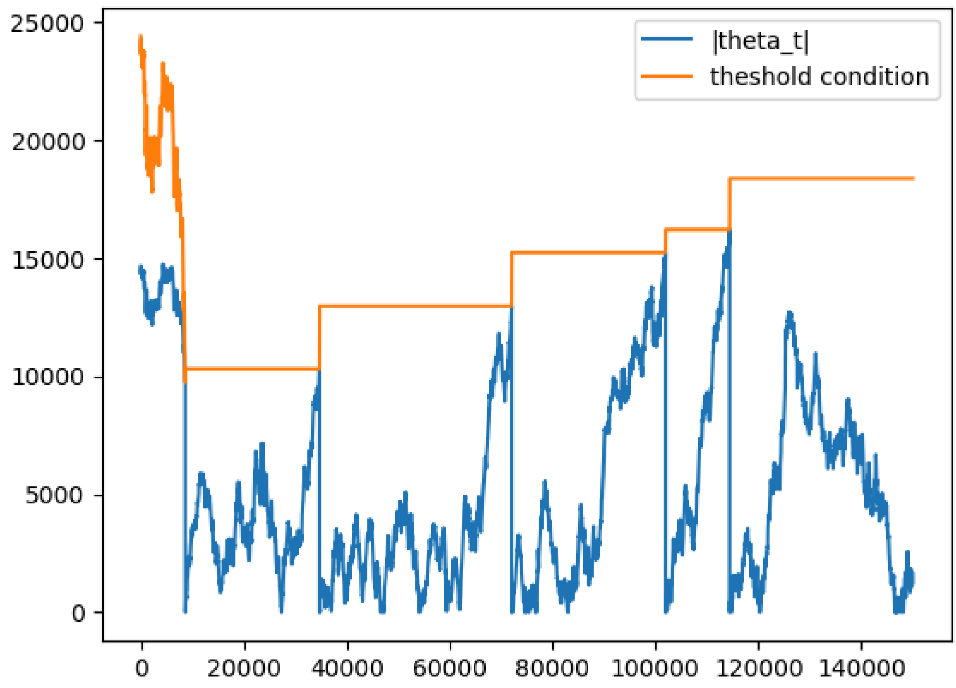
$$\theta_t = \sum_{t=t^*}^{T} b_t * v_t$$

t^* is time when previous imbalance bar was formed

Let's look at dynamics of $|\theta_t|$ and $E_0[T] * |2v^+ - E_0[v_t]|$ to understand why we decided to reset θ_t when bar is formed. The dynamics when theta value is reset:

In [11]:
```
Image('images/mlfinlab_implementation.png')
```

Out[11]:

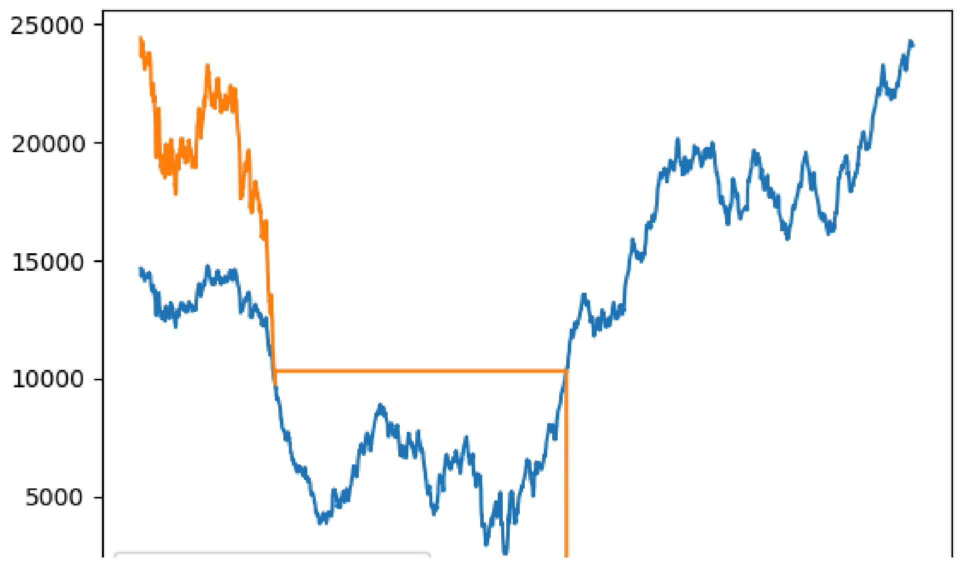


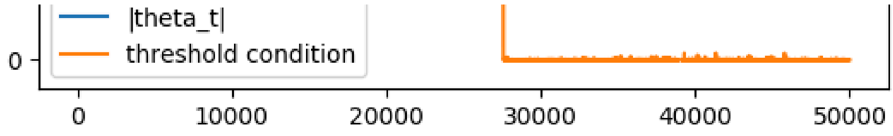
Note that on the first ticks, threshold condition is not stable. Remember, before the first bar is generated, expected imbalance is calculated on every tick with window = expected_num_ticks_init, that is why it changes with every tick. After the first bar was generated both expected number of ticks ($E_0[T]$) and exptected volume imbalance ($2v^+ - E_0[v_t]$) are updated only when the next bar is generated

When theta is not reset:

In [12]:
```
Image('images/book_implementation.png')
```

Out[12]:





The reason for that is due to the fact that theta is accumulated when several bars are generated theta value is not reset ⇒ condition is met on small number of ticks ⇒ length of the next bar converges to 1 ⇒ bar is sampled on the next consecutive tick.

The logic described above is implemented in **mlfinlab** package in *ImbalanceBars*

Statistical properties of imbalance bars. Exercise 2.2 from the book

In [13]:

```
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
import numpy as np
import pandas as pd
```

In [14]:

```
 # imbalance bars generated with num_prev_bars = 3, num_ticks_init =
imb_bars = pd.read_csv('../Sample-Data/imbalance_bars_3_100000.csv')
dollar_bars = pd.read_csv('../Sample-Data/dollar_bars_ex_2.2.csv')
```

In [15]:

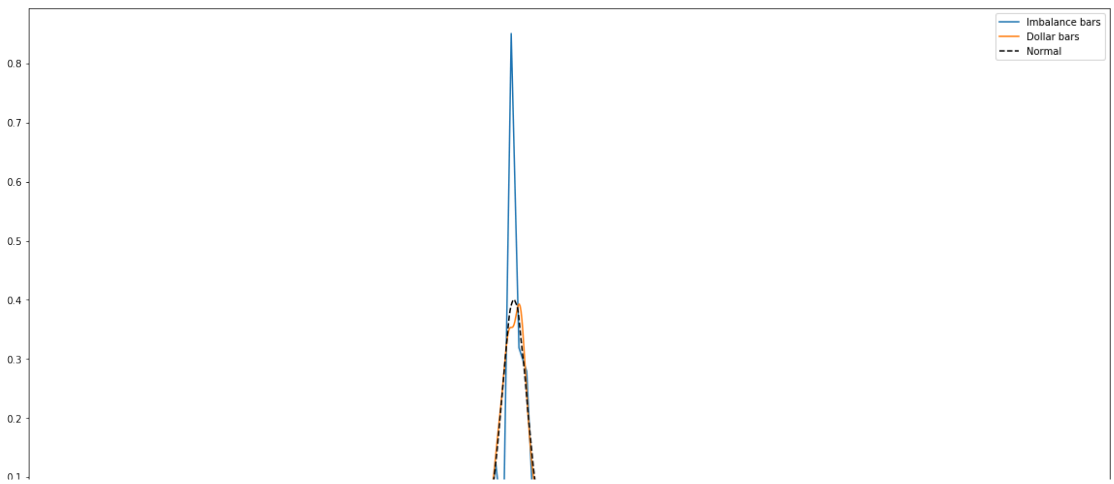
```
dollar_bars['log_ret'] = np.log(dollar_bars['close']).diff().fillna(0
imb_bars['log_ret'] = np.log(imb_bars['close']).diff().fillna(0)
```

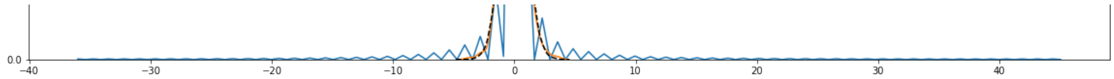
In [16]:

```
plt.figure(figsize=(20,10))
sns.kdeplot((imb_bars.log_ret - imb_bars.log_ret.mean()) / imb_bars.l
sns.kdeplot((dollar_bars.log_ret - dollar_bars.log_ret.mean()) / doll
sns.kdeplot(np.random.normal(size=len(imb_bars)), label="Normal", col
plt.title()
```

Out[16]:

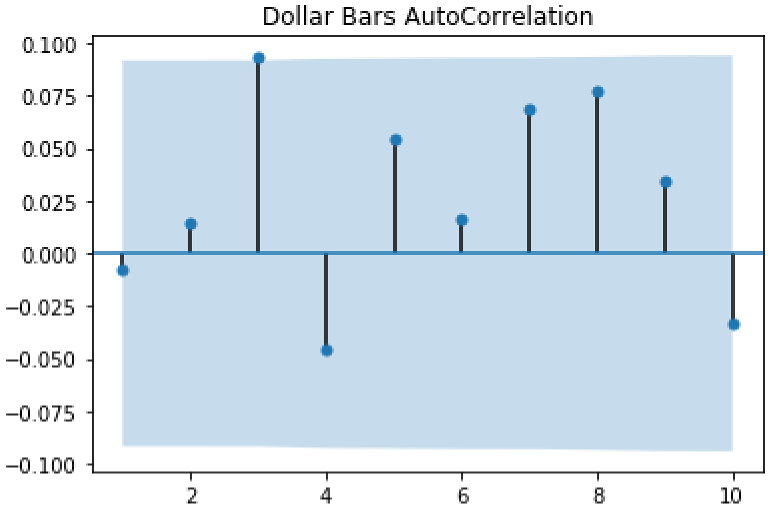
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fab536ccda0>
```





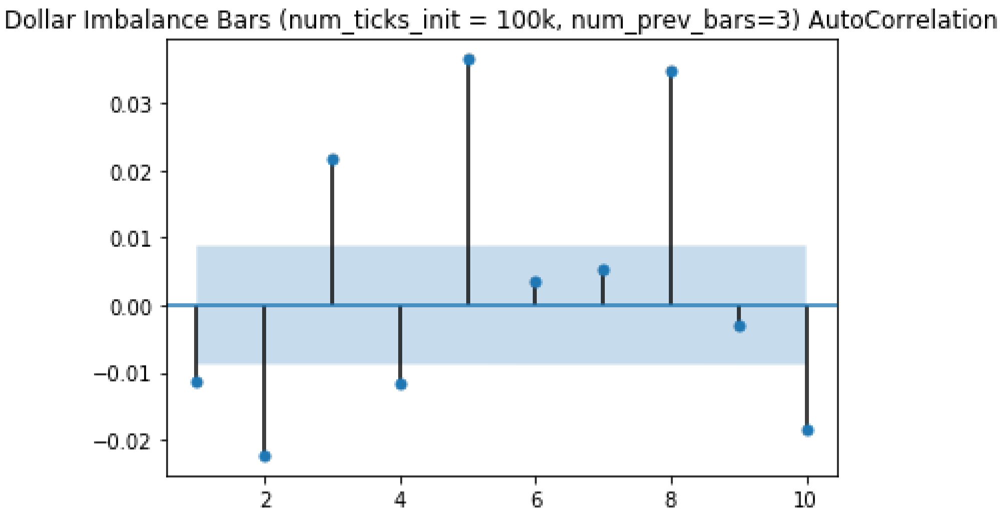
In [17]:

```
plot_acf(dollar_bars.log_ret, lags=10, zero=False)
plt.title('Dollar Bars AutoCorrelation')
plt.show()
```



In [18]:

```
plot_acf(imb_bars.log_ret, lags=10, zero=False)
plt.title('Dollar Imbalance Bars (num_ticks_init = 100k, num_prev_bar
plt.show()
```



In [20]:

```
imb_bars['date_time'] = pd.to_datetime(imb_bars.date_time)
dollar_bars['date_time'] = pd.to_datetime(dollar_bars.date_time)
```