Triple Barrier Method

This notebook will cover partial exercise answers:

- Exercise 3.1
- Exercise 3.2
- Exercise 3.3

As we go along, there will be some explanations.

More importantly, this method can be applied not just within mean-reversion strategy but also other strategies as well. Most of the functions below can be found under research/Labels.

Contact: boyboi86@gmail.com

In [1]:

```
import numpy as np
import pandas as pd
import research as rs
import matplotlib.pyplot as plt

%matplotlib inline

p = print

#pls take note of version
#numpy 1.17.3
#pandas 1.0.3
#sklearn 0.21.3

dollar = pd.read_csv('./research/Sample_data/dollar_bars.txt',
                 sep=',',
                 header=0,
                 parse_dates = True,
                 index_col=['date_time'])
```

```
Num of CPU core:  4
Machine info:  Windows-10-10.0.18362-SP0
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Numpy 1.17.3
Pandas 1.0.3
```

```
C:\Users\Wei_X\Anaconda3\lib\site-packages\pandas_datareader\compat\__init_
_.py:7: FutureWarning: pandas.util.testing is deprecated. Use the functions
in the public API at pandas.testing instead.
  from pandas.util.testing import assert_frame_equal
```

```
<Figure size 1500x800 with 1 Axes>
```

In [2]:

```
d_vol = rs.vol(dollar['close'], span0 = 50)
```

```
In [3]: events = rs.cs_filter(dollar['close'],
                              limit = d_vol.mean())

        events
```

```
Out[3]: DatetimeIndex(['2015-01-02 07:07:35.156000', '2015-01-02 09:35:57.204000',
                       '2015-01-02 12:59:42.176000', '2015-01-02 14:19:33.847000',
                       '2015-01-02 14:33:39.311000', '2015-01-02 14:42:28.315000',
                       '2015-01-02 14:51:59.300000', '2015-01-02 15:01:45.497000',
                       '2015-01-02 15:14:31.569000', '2015-01-02 15:22:54.187000',
                       ...
                       '2016-12-30 20:57:19.151000', '2016-12-30 20:58:34.724000',
                       '2016-12-30 20:59:16.663000', '2016-12-30 20:59:34.157000',
                       '2016-12-30 20:59:50.345000', '2016-12-30 20:59:58.848000',
                       '2016-12-30 21:00:00.352000', '2016-12-30 21:00:24.294000',
                       '2016-12-30 21:03:03.027000', '2016-12-30 21:13:31.990000'],
                      dtype='datetime64[ns]', length=22890, freq=None)
```

```
In [4]: vb = rs.vert_barrier(data = dollar['close'],
                             events = events,
                             period = 'days',
                             freq = 1)

        vb # Show some example output
```

```
Out[4]: 2015-01-02 07:07:35.156   2015-01-04 23:20:12.567
        2015-01-02 09:35:57.204   2015-01-04 23:20:12.567
        2015-01-02 12:59:42.176   2015-01-04 23:20:12.567
        2015-01-02 14:19:33.847   2015-01-04 23:20:12.567
        2015-01-02 14:33:39.311   2015-01-04 23:20:12.567
                                            ...
        2016-12-29 19:50:32.702   2016-12-30 19:55:31.030
        2016-12-29 20:43:20.886   2016-12-30 20:44:21.481
        2016-12-29 20:56:54.013   2016-12-30 20:57:19.151
        2016-12-29 21:00:00.349   2016-12-30 21:00:00.352
        2016-12-29 21:13:14.022   2016-12-30 21:13:31.990
        Name: date_time, Length: 22850, dtype: datetime64[ns]
```

```
In [5]: tb = rs.tri_barrier(data = dollar['close'],
                            events = events,
                            trgt = d_vol,
                            min_req = 0.002,
                            num_threads = 3,
                            ptSl = [1,1],
                            t1 = vb,
                            side = None)

        tb # Show some example

        # the pandas obj will break the data up process it then stich it back into 1
        # this will only happen when you use pandas obj multiprocess func using num_

        # if you scroll all the way to the bottom, that is your final dataframe outp
```

```
C:\Users\Wei_X\Desktop\Python\research\Labels\triple_barrier_method.py:75: U
serWarning: Data and events index shape must be same, reindex data to fit ev
ents
  warnings.warn('Data and events index shape must be same, reindex data to f
it events')
C:\Users\Wei_X\Desktop\Python\research\Labels\triple_barrier_method.py:112:
UserWarning: Not Recommended: No side prediction provided
  warnings.warn('Not Recommended: No side prediction provided')
```

```
[                                             t1                       sl  \
2015-01-05 14:54:26.286 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 14:57:13.616 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 15:01:57.494 2015-01-06 15:06:20.346 2015-01-05 16:21:16.062
2015-01-05 15:07:29.012 2015-01-06 15:13:19.811 2015-01-05 15:40:45.114
2015-01-05 15:13:09.655 2015-01-06 15:13:19.811 2015-01-05 16:10:05.172
...                                         ...                     ...
2015-09-16 19:10:49.674 2015-09-17 19:13:55.901                     NaT
2015-09-16 19:22:06.172 2015-09-17 19:22:59.160                     NaT
2015-09-16 19:32:47.172 2015-09-17 19:36:50.249                     NaT
2015-09-16 19:45:01.362 2015-09-17 19:47:12.228                     NaT
2015-09-16 19:54:03.737 2015-09-17 19:55:09.135                     NaT


                                             pt
2015-01-05 14:54:26.286                     NaT
2015-01-05 14:57:13.616                     NaT
2015-01-05 15:01:57.494                     NaT
2015-01-05 15:07:29.012                     NaT
2015-01-05 15:13:09.655                     NaT
...                                         ...
2015-09-16 19:10:49.674 2015-09-17 18:13:04.358
2015-09-16 19:22:06.172 2015-09-17 18:44:39.366
2015-09-16 19:32:47.172 2015-09-17 18:44:39.366
2015-09-16 19:45:01.362 2015-09-17 18:44:39.366
2015-09-16 19:54:03.737 2015-09-17 18:13:04.358

[7408 rows x 3 columns]] this out
[                                             t1                       sl  \
2015-01-05 14:54:26.286 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 14:57:13.616 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 15:01:57.494 2015-01-06 15:06:20.346 2015-01-05 16:21:16.062
2015-01-05 15:07:29.012 2015-01-06 15:13:19.811 2015-01-05 15:40:45.114
2015-01-05 15:13:09.655 2015-01-06 15:13:19.811 2015-01-05 16:10:05.172
...                                         ...                     ...
2015-09-16 19:10:49.674 2015-09-17 19:13:55.901                     NaT
2015-09-16 19:22:06.172 2015-09-17 19:22:59.160                     NaT
2015-09-16 19:32:47.172 2015-09-17 19:36:50.249                     NaT
2015-09-16 19:45:01.362 2015-09-17 19:47:12.228                     NaT
2015-09-16 19:54:03.737 2015-09-17 19:55:09.135                     NaT


                                             pt
2015-01-05 14:54:26.286                     NaT
2015-01-05 14:57:13.616                     NaT
2015-01-05 15:01:57.494                     NaT
2015-01-05 15:07:29.012                     NaT
2015-01-05 15:13:09.655                     NaT
...                                         ...
2015-09-16 19:10:49.674 2015-09-17 18:13:04.358
2015-09-16 19:22:06.172 2015-09-17 18:44:39.366
2015-09-16 19:32:47.172 2015-09-17 18:44:39.366
2015-09-16 19:45:01.362 2015-09-17 18:44:39.366
2015-09-16 19:54:03.737 2015-09-17 18:13:04.358

[7408 rows x 3 columns],                                              t1
sl  \
2016-04-28 08:11:31.935 2016-04-29 10:02:20.933 2016-04-28 19:53:44.370
2016-04-28 08:58:32.457 2016-04-29 10:02:20.933                     NaT
2016-04-28 10:52:03.623 2016-04-29 11:47:49.541 2016-04-28 19:31:00.850
2016-04-28 12:01:23.295 2016-04-29 12:37:34.150 2016-04-28 19:25:12.672
2016-04-28 13:01:28.025 2016-04-29 13:28:30.173 2016-04-28 19:25:12.672
...                                         ...                     ...
2016-12-30 20:59:58.848                     NaT                     NaT
2016-12-30 21:00:00.352                     NaT                     NaT
2016-12-30 21:00:24.294                     NaT                     NaT
2016-12-30 21:03:03.027                     NaT                     NaT
2016-12-30 21:13:31.990                     NaT                     NaT

                                             pt
2016-04-28 08:11:31.935 2016-04-28 13:30:00.579
2016-04-28 08:58:32.457 2016-04-28 13:01:28.025
2016-04-28 10:52:03.623 2016-04-28 13:39:44.393
2016-04-28 12:01:23.295 2016-04-28 13:44:48.201
2016-04-28 13:01:28.025 2016-04-28 13:44:48.201
...                                         ...
2016-12-30 20:59:58.848                     NaT
2016-12-30 21:00:00.352                     NaT
2016-12-30 21:00:24.294                     NaT
2016-12-30 21:03:03.027                     NaT
2016-12-30 21:13:31.990                     NaT

[7408 rows x 3 columns]] this out
```

```
2020-05-21 15:58:35.029251 33.33% _pt_sl_t1 done after 0.47 minutes. Remaini
ng 0.94 minutes.
2020-05-21 15:58:35.187171 66.67% _pt_sl_t1 done after 0.47 minutes. Remaini
ng 0.24 minutes.
```

```
[                                             t1                      sl  \
2015-01-05 14:54:26.286 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 14:57:13.616 2015-01-06 15:01:01.702 2015-01-05 15:40:45.114
2015-01-05 15:01:57.494 2015-01-06 15:06:20.346 2015-01-05 16:21:16.062
2015-01-05 15:07:29.012 2015-01-06 15:13:19.811 2015-01-05 15:40:45.114
2015-01-05 15:13:09.655 2015-01-06 15:13:19.811 2015-01-05 16:10:05.172
...                                         ...                     ...
2015-09-16 19:10:49.674 2015-09-17 19:13:55.901                     NaT
2015-09-16 19:22:06.172 2015-09-17 19:22:59.160                     NaT
2015-09-16 19:32:47.172 2015-09-17 19:36:50.249                     NaT
2015-09-16 19:45:01.362 2015-09-17 19:47:12.228                     NaT
2015-09-16 19:54:03.737 2015-09-17 19:55:09.135                     NaT
```

```
                                             pt
2015-01-05 14:54:26.286                     NaT
2015-01-05 14:57:13.616                     NaT
2015-01-05 15:01:57.494                     NaT
2015-01-05 15:07:29.012                     NaT
2015-01-05 15:13:09.655                     NaT
...                                         ...
2015-09-16 19:10:49.674 2015-09-17 18:13:04.358
2015-09-16 19:22:06.172 2015-09-17 18:44:39.366
2015-09-16 19:32:47.172 2015-09-17 18:44:39.366
2015-09-16 19:45:01.362 2015-09-17 18:44:39.366
2015-09-16 19:54:03.737 2015-09-17 18:13:04.358

[7408 rows x 3 columns],                                              t1
sl  \
2016-04-28 08:11:31.935 2016-04-29 10:02:20.933 2016-04-28 19:53:44.370
2016-04-28 08:58:32.457 2016-04-29 10:02:20.933                     NaT
2016-04-28 10:52:03.623 2016-04-29 11:47:49.541 2016-04-28 19:31:00.850
2016-04-28 12:01:23.295 2016-04-29 12:37:34.150 2016-04-28 19:25:12.672
2016-04-28 13:01:28.025 2016-04-29 13:28:30.173 2016-04-28 19:25:12.672
...                                         ...                     ...
2016-12-30 20:59:58.848                     NaT                     NaT
2016-12-30 21:00:00.352                     NaT                     NaT
2016-12-30 21:00:24.294                     NaT                     NaT
2016-12-30 21:03:03.027                     NaT                     NaT
2016-12-30 21:13:31.990                     NaT                     NaT

                                             pt
2016-04-28 08:11:31.935 2016-04-28 13:30:00.579
2016-04-28 08:58:32.457 2016-04-28 13:01:28.025
2016-04-28 10:52:03.623 2016-04-28 13:39:44.393
2016-04-28 12:01:23.295 2016-04-28 13:44:48.201
2016-04-28 13:01:28.025 2016-04-28 13:44:48.201
...                                         ...
2016-12-30 20:59:58.848                     NaT
2016-12-30 21:00:00.352                     NaT
2016-12-30 21:00:24.294                     NaT
2016-12-30 21:03:03.027                     NaT
2016-12-30 21:13:31.990                     NaT

[7408 rows x 3 columns],                                              t1
sl  \
2015-09-16 19:59:40.048 2015-09-17 19:59:49.542                     NaT
2015-09-16 20:00:13.782 2015-09-17 20:03:58.960                     NaT
2015-09-16 20:13:49.208 2015-09-17 20:49:02.616 2015-09-17 19:55:09.135
2015-09-17 02:31:29.158 2015-09-18 05:53:05.346 2015-09-17 20:03:58.960
2015-09-17 07:50:29.399 2015-09-18 08:36:51.929                     NaT
...                                         ...                     ...
2016-04-28 01:20:23.379 2016-04-29 04:43:02.149 2016-04-28 03:08:23.517
2016-04-28 03:08:23.517 2016-04-29 04:43:02.149 2016-04-28 08:11:31.935
2016-04-28 03:49:42.423 2016-04-29 04:43:02.149 2016-04-28 08:11:31.935
2016-04-28 06:14:15.071 2016-04-29 07:11:43.177 2016-04-28 08:11:31.935
2016-04-28 07:22:07.437 2016-04-29 08:08:00.258 2016-04-28 08:58:32.457
```

```
                                                 pt
2015-09-16 19:59:40.048 2015-09-17 18:13:04.358
2015-09-16 20:00:13.782 2015-09-17 18:13:04.358
2015-09-16 20:13:49.208 2015-09-17 18:44:39.366
2015-09-17 02:31:29.158 2015-09-17 18:13:04.358
2015-09-17 07:50:29.399 2015-09-17 17:23:18.817
...                                         ...
2016-04-28 01:20:23.379                     NaT
2016-04-28 03:08:23.517 2016-04-28 13:51:37.515
2016-04-28 03:49:42.423 2016-04-28 14:06:36.621
2016-04-28 06:14:15.071 2016-04-28 13:51:37.515
2016-04-28 07:22:07.437 2016-04-28 13:44:48.201

[7408 rows x 3 columns]] this out
```

```
2020-05-21 15:58:35.516536 100.0% _pt_sl_t1 done after 0.48 minutes. Remaini
ng 0.0 minutes.
```

Out[5]:

	t1	trgt
2015-01-05 14:54:26.286	2015-01-05 15:40:45.114	0.002244
2015-01-05 14:57:13.616	2015-01-05 15:40:45.114	0.002469
2015-01-05 15:01:57.494	2015-01-05 16:21:16.062	0.002787
2015-01-05 15:07:29.012	2015-01-05 15:40:45.114	0.002827
2015-01-05 15:13:09.655	2015-01-05 16:10:05.172	0.002882
...	...	...
2016-12-30 20:59:58.848	NaT	0.002397
2016-12-30 21:00:00.352	NaT	0.002350
2016-12-30 21:00:24.294	NaT	0.002304
2016-12-30 21:03:03.027	NaT	0.002261
2016-12-30 21:13:31.990	NaT	0.002228

22224 rows × 2 columns

In [6]:

```
m_label = rs.meta_label(data = dollar['close'],
                        events = tb,
                        drop = False)

m_label # Show some example

# previously when we run tri_bar func, NaT is present. However once func is
# There is an in-built drop func that will trigger the below drop_label func
# change drop = False to float value i.e. 0.05
```

Out[6]:

	ret	bin
2015-01-05 14:54:26.286	-0.003448	-1.0
2015-01-05 14:57:13.616	-0.002957	-1.0
2015-01-05 15:01:57.494	-0.003701	-1.0
2015-01-05 15:07:29.012	-0.002957	-1.0
2015-01-05 15:13:09.655	-0.003451	-1.0
...	...	...
2016-12-30 18:02:22.880	-0.003242	-1.0
2016-12-30 18:36:03.267	-0.002904	-1.0
2016-12-30 19:02:57.783	-0.002908	-1.0
2016-12-30 19:55:31.030	0.003028	1.0
2016-12-30 20:50:57.567	0.002915	1.0

22207 rows × 2 columns

> AFML page 54 section 3.9
>
> "Some ML classifiers do not perform well when data samples are too imbalanced. In those circumstances, it is preferably to drop those rare labels and focus on more common outcomes."

In [7]:

```
m_label['bin'].value_counts()

# Here is a quick look at our 'bin' values.
# Apparently we have a rare label, bin = 0
```

Out[7]:

```
 1.0    11343
-1.0    10784
 0.0       80
Name: bin, dtype: int64
```

In [8]:

```
m_label['bin'].value_counts(normalize = True)

# basically it's 0.003602 of all our metalabels. Max is 1
```

Out[8]:

```
 1.0    0.510785
-1.0    0.485613
 0.0    0.003602
Name: bin, dtype: float64
```

In [9]:

```
drop_meta_label = rs.drop_label(events = m_label,
                                min_pct = 0.05)
```

```
drop_meta_label # Show some example

# In the below case we dropped all bin = 0, while keeping only 1 & -1
```

Out[9]:

	ret	bin
2015-01-05 14:54:26.286	-0.003448	-1.0
2015-01-05 14:57:13.616	-0.002957	-1.0
2015-01-05 15:01:57.494	-0.003701	-1.0
2015-01-05 15:07:29.012	-0.002957	-1.0
2015-01-05 15:13:09.655	-0.003451	-1.0
...	...	...
2016-12-30 18:02:22.880	-0.003242	-1.0
2016-12-30 18:36:03.267	-0.002904	-1.0
2016-12-30 19:02:57.783	-0.002908	-1.0
2016-12-30 19:55:31.030	0.003028	1.0
2016-12-30 20:50:57.567	0.002915	1.0

22127 rows × 2 columns