ob-ipython Introduction

Elliott L Barcikowski

January 10th, 2018

Outline

Introduction

Analysis Bitcoin versus GBTC

Conclusions

Goals

1. Introduce ob-ipython
    - This means we must discuss org, babel
2. Do a quick live example
3. Comments from my experience
4. Generate this talk

Emacs org-mode

What is it?

- "Your life in plain text"
- sophistacated mark up language, designed for organization

Features

- outlining
- manages tasks, agendas, todos
- folding, navigation, links
- easily exported
- much more
  - really, org has an enormous amount of functionality

Babel

Dispatch source code and execute it inside org mode.

- Support for shell, R, perl, and more
- Can generate tables
- Can include figures inline
- "Literate Programming"

Example

```
#+BEGIN_SRC python
import numpy as np
x = np.arange(5)
return x
#+END_SRC
```

Evaluates to

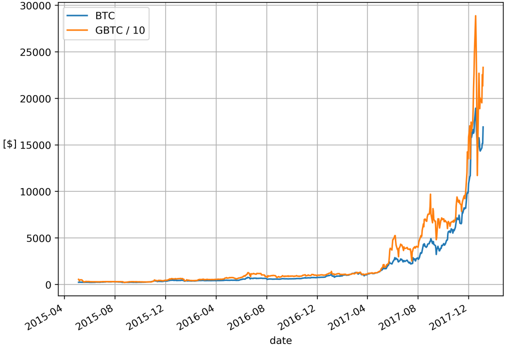
```
0   1   2   3   4
```

ob-ipython

- Allows Babel to interact with a Jupyter kernel
- Different kernel types
    - I havent tried this
- Remote kernels
- Inline plotting
- ipython features

Bitcoin - GBTC

- GBTC is a trust that holds BTC
  - Manges serves, security, etc.
- One GBTC share represents .1 BTC
- Generally more liquid than BTC
- Only trades Monday - Friday



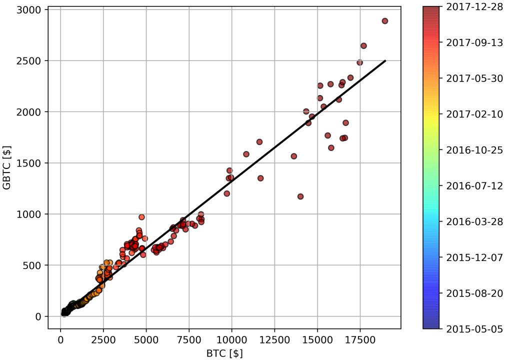
:END:

Thesis

When GBTC dips relative to BTC it could be an opportunity.

Fit with Least Squares

- Ok, it has a linear relationship
- But, does this model the aspects that we want to trade?



Kalman Filter Approach (1)

A kalman filter may be used to dynamically estimate the relationship between the two sets of data.

Allow

$$\vec{x}_t = \begin{bmatrix} m_t \\ b_t \end{bmatrix} \tag{1}$$

where m_t is the slope and b_t is the intercept, similar to our fit above.

Then, the kalman transition equation may be written, trivially as

$$\vec{x}_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \vec{x}_t + \vec{q} \tag{2}$$

and

$$z_t = \begin{bmatrix} BTC_t & 1 \end{bmatrix} \vec{x}_t + R \tag{3}$$

Kalman Filter Approach (2)

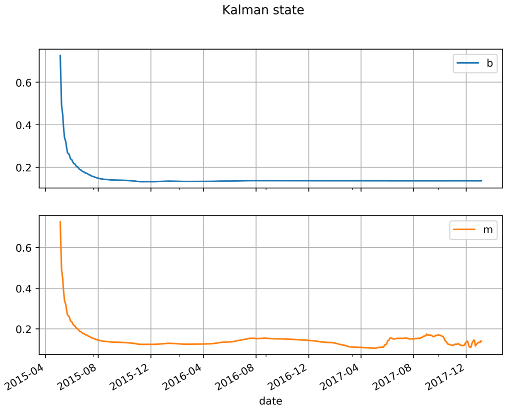
Our system has now been written as a Kalman filter with z_t representing the state t of the price of GBTC and h_t representing the price of BTC at time t. In this system, the Kalman state, x_t is our fit parameters m and b, the tradtional Kalman transition matrix,

$$A_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4}$$

and the observation matrix H_t holds the Bitcoin prices.

Apply Kalman Filter

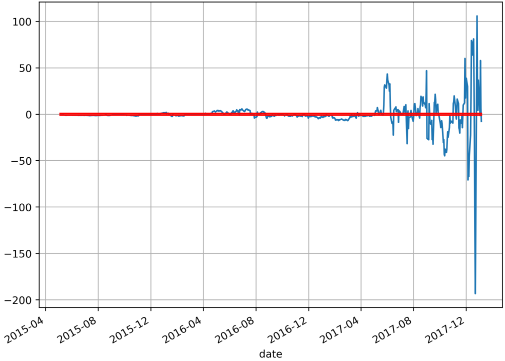
- Used pykalman package
- Slope, intercept (m, b) for each data point
- Covariances for each data point



Residuals (1)

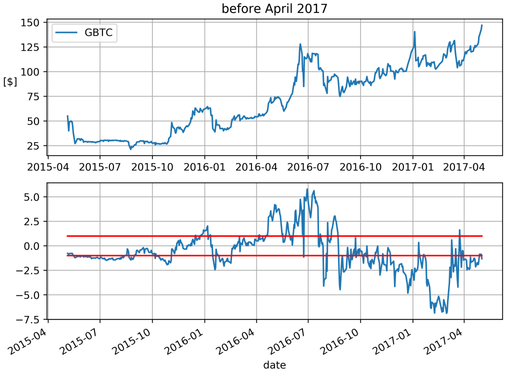
- Calculate errors from the returned state covariance
- Clearly this system behaves differently as bitcoin takes off

$$r_t = \frac{GBTC_t - m_t BTC_t - b_t}{\sigma_t}$$



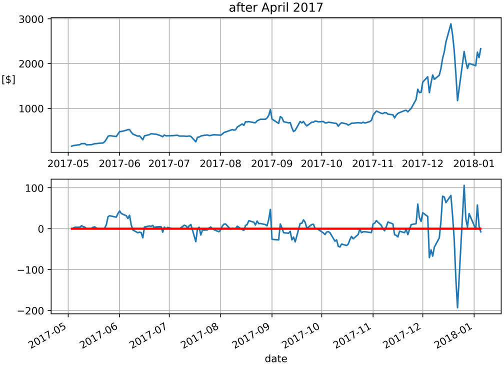
Residuals (2)

- Up to April 2017
- Thesis seems to play out



Residuals (3)

- After to April 2017
- Model performance has completely changed
    - Choices for R, q for the whole data set



Backtest

- When GBTC dips below σ buy
- Sell at above σ
- Should have just held BTC from 2009 ;-)

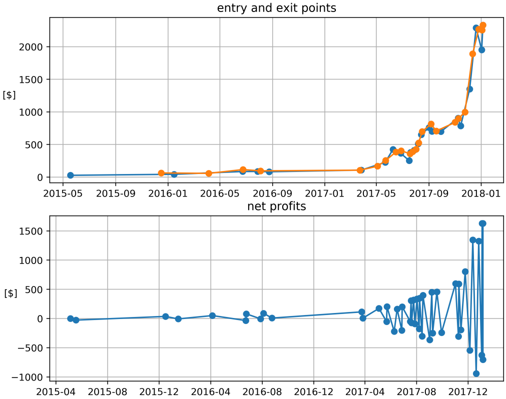
Simulation

```
thresh = 1.0
money = [(gbtc.index[0], 0.0)]
in_trade = False
buys = []
sells = []
for date, price, res in zip(gbtc.index, gbtc, residuals):
    if in_trade:
if res > thresh:
    money.append((date, money[-1][1] + price))
    in_trade = False
    sells.append((date, price))
    else:
if res < -1 * thresh:
    money.append((date, money[-1][1] - price))
    in_trade = True
    buys.append((date, price))
if in_trade:
    money.append((date, money[-1][1] + price))
    sells.append((date, price))

buys = np.array(buys)
sells = np.array(sells)
money = np.array(money)
```

Trades

- Not a particular exciting trade
- Should have just held BTC from 2009 ;-)



What are we left with?

1628.89

ob-ipython versus python notebooks (IMO)

For

- Integrates with org and emacs
- Works better with source control
- Code is just text files
- Better tools for documents

Against

- Needs emacs
- More cumbersome to excute lots of cells
- Lots of boilerplate
- No else on my team uses this
- I hate LaTeX

Resources

org-mode

- `https://orgmode.org`
- **Recommended**: `https://www.youtube.com/watch?v=oJTwQvgfgMM&t=512s`

Babel

- `https://orgmode.org/worg/org-contrib/babel/intro.html`

ob-ipython

- `https://github.com/gregsexton/ob-ipython`

Beamer

- `https://github.com/dfeich/org-babel-examples/blob/master/beamer/beamer-example.org`

Kalman Analysis

- `https://www.quantopian.com/posts/quantcon-2016-using-the-kalman-filter-in-algorithmic-trading`