Not Over Thinking

A Closer Look at Mean Reversion Trading Strategy

24.00% Annualised Return

72.23% Maximum Drawdown – need to improve on trend resistance

Easy to implement – 190 lines of code only

Haixiang
2022.03 | Vol 1.
hxyan.2015@gmail.com | github.com/hxyan2020

Strategy & Economic Rationale

Mean reversion trading in equities tries to capitalize on extreme changes in the pricing of a particular security, assuming that it will revert to its previous state.

Parameter Setup

- Trading Frequency: Daily
- Long/Short (function 'buy_signal'): Long only
- Moving Average (variable 'g.N'): 30-day close price
- Benchmark Index (function 'set_benchmark'): CSI 300
- Stock Holdings (variable 'g.num_stocks'): 10 stocks in CSI 300
- Back-testing period: 1 January 2006 – 31 May 2016
- Slippage and Transaction Fees (function 'set_slip_fee'): 0.0003 to 0.002 depending on exchange rates

Determine Buy Signal

1. Given a stock close price at a given trading day, compute the deviation from past 30-day mean.

```
def compute_difference_ratio(context, data, stocks):
    # create deviation list
    difference_ratio_table = []
    for stock in stocks:
        # get historical close price
        h = attribute_history(stock, g.N, '1d', ['close'])
        # moving average
        ma = sum(h['close']) / g.N
        # get current price
        current_price = data[stock].price
        # compute deviation
        difference_ratio = (ma - current_price) / ma
        difference_ratio_table.append({'stock': stock, 'difference_ratio' : difference_ratio})
    return(pd.DataFrame(difference_ratio_table))
```

2. Select the top 10 stocks from the results and labelled as 'buy_signal', then store the list in 'MR_should_buy'.

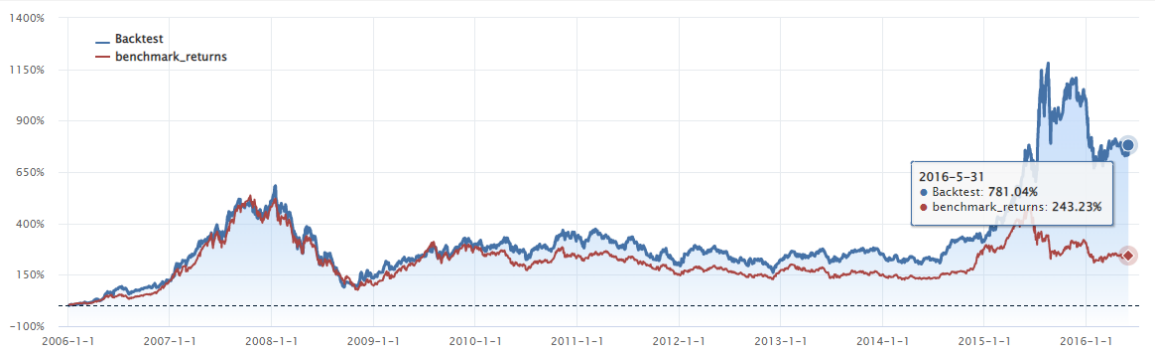
```
def buy_signal(context, data):
    # get deviation in list
    difference_ratio_table = compute_difference_ratio(context, data, g.feasible_stocks)
    # rank deviation from biggest to smallest
    try:
        sorted_table = difference_ratio_table.sort(columns = 'difference_ratio', ascending = False)
    except AttributeError:
        difference_ratio_table.sort_values(by = 'difference_ratio', ascending = False,inplace=True)
        sorted_table = difference_ratio_table
```

3. Execute strategy by buying the shortlisted stocks and selling the losers.

```
# calculate cash per stock
cash_per_stock = context.portfolio.portfolio_value / g.num_stocks
# for unwanted stocks, remove from holding
for stock in holding:
    if stock not in MR_should_buy:
        order_target_value(stock,0)
# got stocks that should buy, buy according to pre-determined portion
log.info(MR_should_buy)
for stock in MR_should_buy:
    order_target_value(stock,cash_per_stock)
```

Back-testing Performance

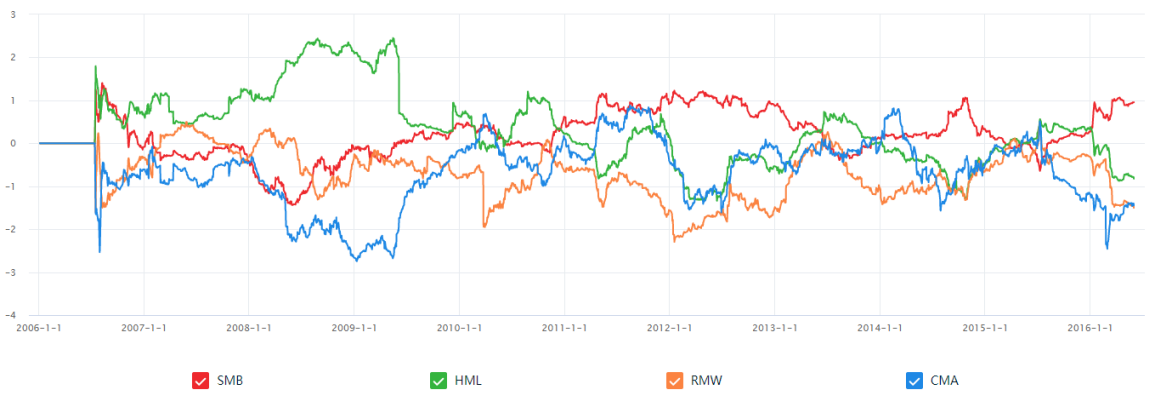
- Cumulative return = 781.04%, with annualised return = 24.00%



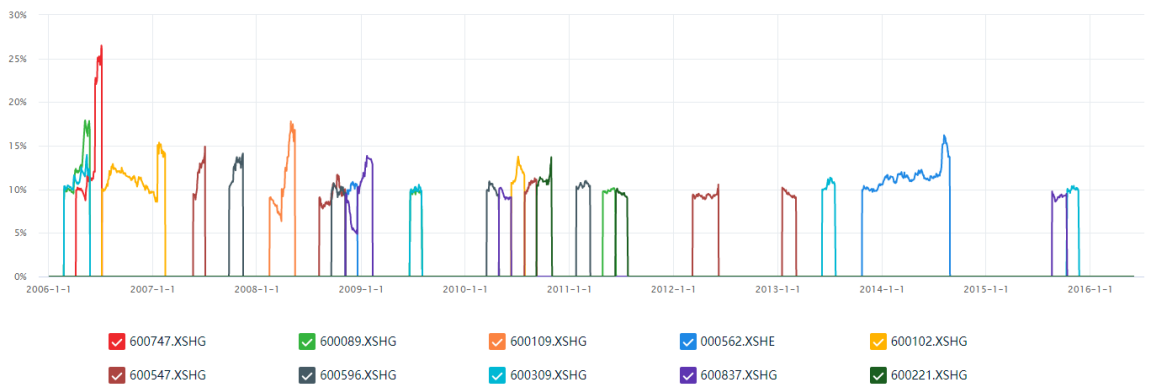
- Other Indicators

Alpha	Beta	Sharpe Ratio	Sortino Ratio	Win Ratio	Maximum Drawdown	Information Ratio	Volatility	Calmar Ratio	R^2
0.111	0.989	0.556	0.75	0.548	72.23%	0.557	0.36	0.33	0.36

- Factor Analysis



- Top Stock Holdings



Potential Improvements

- Parameter optimization - MA could be set at narrower interval (e.g. 10 days) for a more sensitive response to market shift.
- Poor resistance to macro market tanks – unable to outperform market when macroeconomic environment heads down, manifested as 72.23% drawdown.

For source codes, please visit my GitHub: github.com/hxyan2020. I will upload them once China wins the AFC Asia Cup, or earlier if they really take their sweet time about it.