

[2]:

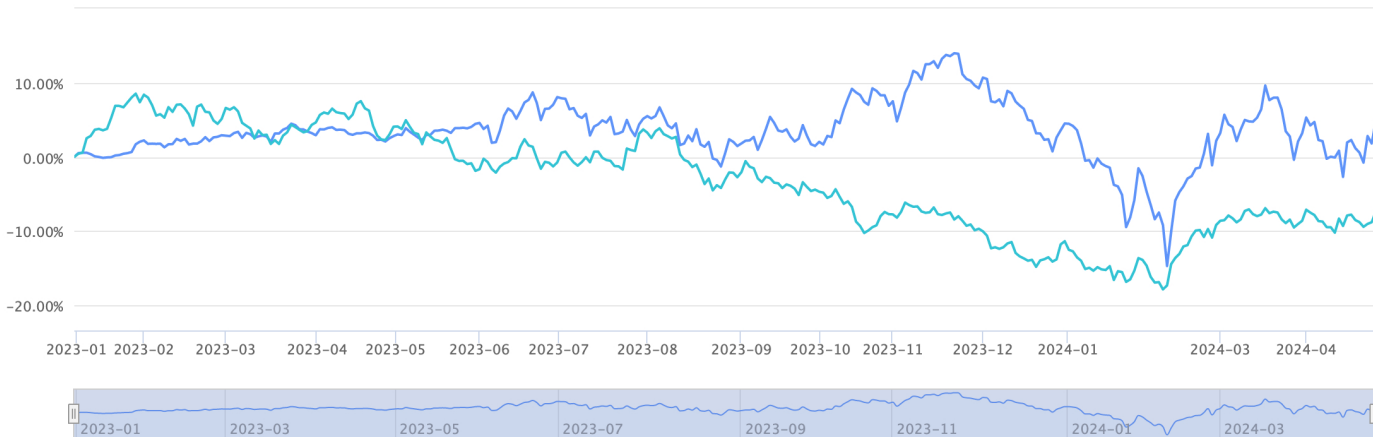
```
1 %%backtest
2 # from datetime import datetime
3 import talib
4
5 start = '2023-01-01'
6 end = '2024-04-28'
7 universe = StockUniverse('ZZ500')
8 benchmark = 'HS300'
9 freq = 'd'
10 refresh_rate = Monthly(1)
11 max_history_window = 60
12 fastperiod = 12 # 快线周期
13 slowerperiod = 26 # 慢线周期
14 signalperiod = 9 # Signal平滑周期
15 # nstocks = 150
16
17 accounts = {
18     'stock_account': AccountConfig(account_type='security', capital_base=1e5)
19 }
20
21
22 def initialize(context):
23     pass
24
25
26 def handle_data(context):
27     current_universe = context.get_universe(exclude_halt=True)
28     # today = context.current_date
29     # print(today)
30     # if today.strftime('%d') == '01':
31     #     prev_month_begin = today + pd.tseries.offsets.MonthBegin(-1)
32     #     else:
33     #         prev_month_begin = today + pd.tseries.offsets.MonthBegin(-2)
34     #     print('prev_month_begin:', prev_month_begin)
35     #     factor_exposure = get_data_cube(symbol=set_universe('A'),
36     #                                     field=['HSIGMA', 'VOL20'],
37     #                                     start=today, end=today,
38     #                                     style='ast').to_frame().reset_index()
39     factor_exposure = context.history(symbol=current_universe, time_range=20, attribute=['HSIGMA', 'VOL20'], freq='d',
40 style='tas', rtype='frame')
41     # factor_exposure = pd.DataFrame(factor_exposure)
42     # print(factor_exposure)
43     # print(list(factor_exposure.keys()))
44     factor_exposure = factor_exposure[list(factor_exposure.keys())[0]]
45     # factor_exposure['HSIGMA'] = factor_exposure['HSIGMA'].to_numpy().reshape(-1,1)
46     # factor_exposure['HSIGMA'] = pd.DataFrame(factor_exposure['HSIGMA'].tolist(), columns=['HSIGMA']).values
47     # factor_exposure['VOL20'] = pd.DataFrame(factor_exposure['VOL20'].tolist(), columns=['VOL20']).values
48     # display(factor_exposure)
49     ##### Intersection #####
50     # stocks_set = {}
51     # stocks_set['rev'] = set(factor_exposure.sort_values('rev', ascending=True)['sec_id'].iloc[0:nstocks].tolist())
52     # stocks_set['mom'] = set(factor_exposure.sort_values('mom', ascending=False)['sec_id'].iloc[0:nstocks].tolist())
53     # stocks_set['illiq'] = set(factor_exposure.sort_values('illiq', ascending=False)['sec_id'].iloc[0:nstocks].tolist())
54     # stocks_set['ivol'] = set(factor_exposure.sort_values('ivol', ascending=True)['sec_id'].iloc[0:nstocks].tolist())
55     # current_universe = set.intersection(
56     #     stocks_set['rev'],
57     #     stocks_set['mom'],
58     #     stocks_set['illiq'],
59     #     stocks_set['ivol']
60     # )
61     # current_universe = list(current_universe)
62     ##### Scoring #####
63     cols = ['HSIGMA', 'VOL20']
64     for col in cols:
65         factor_exposure[f'{col}_rank'] = factor_exposure[col].rank()
66     factor_exposure['HSIGMA_rank'] = factor_exposure['HSIGMA'].rank()
67     factor_exposure['VOL20_rank'] = factor_exposure['VOL20'].rank(ascending=False)
68     factor_exposure['rank_sum'] = factor_exposure['HSIGMA_rank'] + factor_exposure['VOL20_rank']
69     current_universe = list(factor_exposure.sort_values('rank_sum')['sec_id'].iloc[0:20].index)
70     # print(current_universe)
71     factor_exposure['rev_rank'] = factor_exposure['rev'].rank()
72     factor_exposure['mom_rank'] = factor_exposure['mom'].rank(ascending=False)
73     factor_exposure['illiq_rank'] = factor_exposure['illiq'].rank(
74         ascending=False)
75     factor_exposure['ivol_rank'] = factor_exposure['ivol'].rank()
76     factor_exposure['rank_sum'] = factor_exposure['mom_rank']
77     # + factor_exposure['rev_rank'] + factor_exposure['ivol_rank']
78     # + factor_exposure['illiq_rank'] +
79     # factor_exposure.sort_values('rank_sum', inplace=True)
80
81     # current_universe = list(factor_exposure['sec_id'].iloc[20:40])
82     history = context.history(current_universe, 'openPrice', 60, rtype='array') # 拿过去60个交易日的开盘价来估算MACD
83
84     account = context.get_account('stock_account')
85     current_positions = account.get_positions()
86     cash = account.cash
87     buylist = []
88
89     for stock in current_universe:
90         close = history[stock]['openPrice']
91         macd, signal, _ = talib.MACD(close, fastperiod=fastperiod, slowerperiod=slowerperiod, signalperiod=signalperiod)
92         if macd[-2] < signal[-2] and macd[-1] > signal[-1] and stock not in current_positions: # MACD上穿Signal, 且无持仓
93             buylist.append(stock)
94         elif macd[-2] > signal[-2] and macd[-1] < signal[-1] and stock in current_positions: # MACD下穿Signal, 且有持仓
95             account.order_to(stock, 0) # 全部卖出
96             cash += current_positions[stock].amount * context.current_price(stock) # 估计买入金额
97
98     d = min(len(buylist), int(cash // 20000)) # 可以买入的股票数量, 如果资金不够, 只买入部分
99     for stock in buylist[:d]:
100         account.order(stock, 20000 / context.current_price(stock))
```

回测完成, 耗时32.675 秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
3.8%	-5.9%	8.2%	0.84	0.04
收益波动率	信息比率	最大回撤	回撤恢复时间	年化换手率
20.8%	0.56	25.2%	--	57.89%

累计收益率

普通 相对收益



```
[3]: 1 # #查看调仓记录
      2 show_order(start,end)
```

请重新回测后查看调仓记录

```
[4]: 1 # #查看持仓记录
      2 show_position(start,end)
```

请重新回测后查看持仓记录

```
[8]: 1 %%backtest
      2 # from datetime import datetime
      3 import talib
      4
      5 start = '2023-01-01'
      6 end = '2024-03-31'
      7 universe = StockUniverse('ZZ1000')
      8 benchmark = 'HS300'
      9 freq = 'd'
     10 refresh_rate = 1
     11 max_history_window = 60
     12 fastperiod = 12 # 快线周期
     13 slowerperiod = 26 # 慢线周期
     14 signalperiod = 9 # Signal平滑周期
     15 nstocks = 150
     16
     17 accounts = {
     18     'stock_account': AccountConfig(account_type='security', capital_base=1e5)
     19 }
     20
     21
     22 def initialize(context):
     23     pass
     24
     25
     26 def handle_data(context):
     27     current_universe = context.get_universe(exclude_halt=True)
     28     today = context.current_date
     29     # print(today)
     30     if today.strftime('%d') == '01':
     31         prev_month_begin = today + pd.tseries.offsets.MonthBegin(-1)
     32     else:
     33         prev_month_begin = today + pd.tseries.offsets.MonthBegin(-2)
     34     # print('prev_month_begin:', prev_month_begin)
     35     factor_exposure = get_data_cube(symbol=current_universe,
     36                                     field=['d6dk9218qq.size', 'd6dk9218qq.rev',
     37                                             'd6dk9218qq.illiq', 'd6dk9218qq.ivol'],
     38                                     start=prev_month_begin, end=today,
     39                                     style='ast').to_frame().reset_index()
     40     # print(factor_exposure)
     41     factor_exposure.columns = ['date', 'sec_id', 'size', 'rev', 'illiq', 'ivol']
     42     # assert factor_exposure['date'].nunique() == 1
     43     if factor_exposure.shape[0] == 0:
     44         pass
     45     else:
     46         factor_exposure = factor_exposure[factor_exposure['date'] == factor_exposure['date'].unique()[-1]]
     47     ##### Intersection #####
     48     # stocks_set = {}
     49     # stocks_set['rev'] = set(factor_exposure.sort_values('rev', ascending=True)['sec_id'].iloc[0:nstocks].tolist())
     50     # stocks_set['mom'] = set(factor_exposure.sort_values('mom', ascending=False)['sec_id'].iloc[0:nstocks].tolist())
     51     # stocks_set['illiq'] = set(factor_exposure.sort_values('illiq', ascending=False)['sec_id'].iloc[0:nstocks].tolist())
     52     # stocks_set['ivol'] = set(factor_exposure.sort_values('ivol', ascending=True)['sec_id'].iloc[0:nstocks].tolist())
     53     # current_universe = set.intersection(
     54         # stocks_set['rev'],
```

```

55 # # stocks_set['mom'],
56 # # stocks_set['illiq'],
57 # stocks_set['ivol']
58 # )
59 # current_universe = list(current_universe)
60 ##### Scoring #####
61 factor_exposure['rev_rank'] = factor_exposure['rev'].rank()
62 factor_exposure['size_rank'] = factor_exposure['size'].rank(ascending=False)
63 factor_exposure['illiq_rank'] = factor_exposure['illiq'].rank(
64 ascending=False)
65 factor_exposure['ivol_rank'] = factor_exposure['ivol'].rank()
66 factor_exposure['rank_sum'] = factor_exposure['size_rank']
67 # factor_exposure['rev_rank'] + factor_exposure['ivol_rank']
68 # factor_exposure['illiq_rank'] +
69 factor_exposure.sort_values('rank_sum', inplace=True)
70
71 current_universe = list(factor_exposure['sec_id'].iloc[20:40])
72 history = context.history(current_universe, 'closePrice', 60, rtype='array') # 拿过去60个交易日的收盘价来估算MACD
73
74 account = context.get_account('stock_account')
75 current_positions = account.get_positions()
76 cash = account.cash
77 buylist = []
78
79 for stock in current_universe:
80 close = history[stock]['closePrice']
81 macd, signal, _ = talib.MACD(close, fastperiod=fastperiod, slowperiod=slowperiod, signalperiod=signalperiod)
82 if macd[-2] < signal[-2] and macd[-1] > signal[-1] and stock not in current_positions: # MACD上穿Signal, 且无持仓
83 buylist.append(stock)
84 elif macd[-2] > signal[-2] and macd[-1] < signal[-1] and stock in current_positions: # MACD下穿Signal, 且有持仓
85 account.order_to(stock, 0) # 全部卖出
86 cash += current_positions[stock].amount * context.current_price(stock) # 估计买入金额
87
88 d = min(len(buylist), int(cash) // 20000) # 可以买入的股票数量, 如果资金不够, 只买入部分
89 for stock in buylist[:d]:
90 account.order(stock, 20000 / context.current_price(stock))
91

```

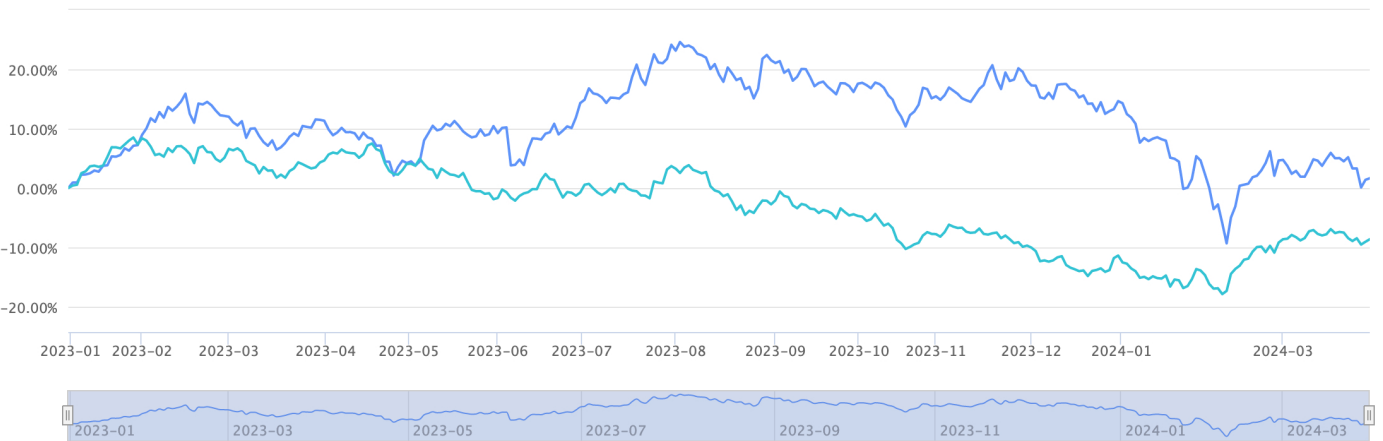
^

回测完成, 耗时288.023秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
1.4%	-7.2%	6.5%	0.79	-0.08
收益波动率	信息比率	最大回撤	回撤恢复时间	年化换手率
20.9%	0.48	27.2%	--	134.63%

累计收益率

普通 相对收益



[]:

1