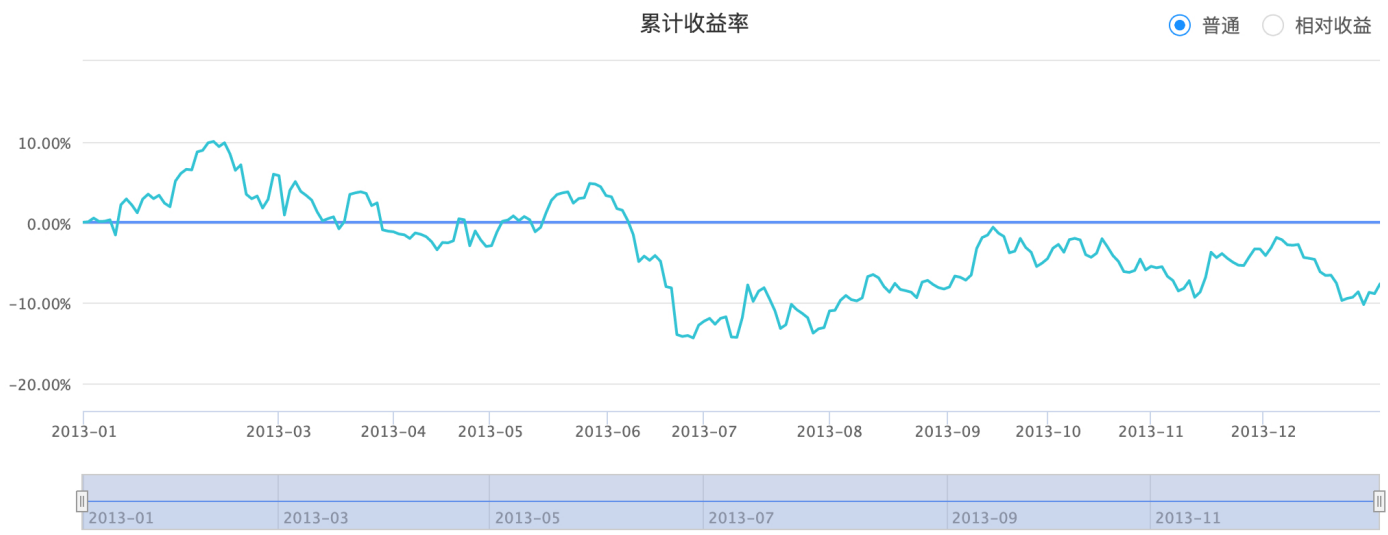
Check some variables

[1]:
```
%%backtest
pass
```

回测完成，耗时9.971 秒

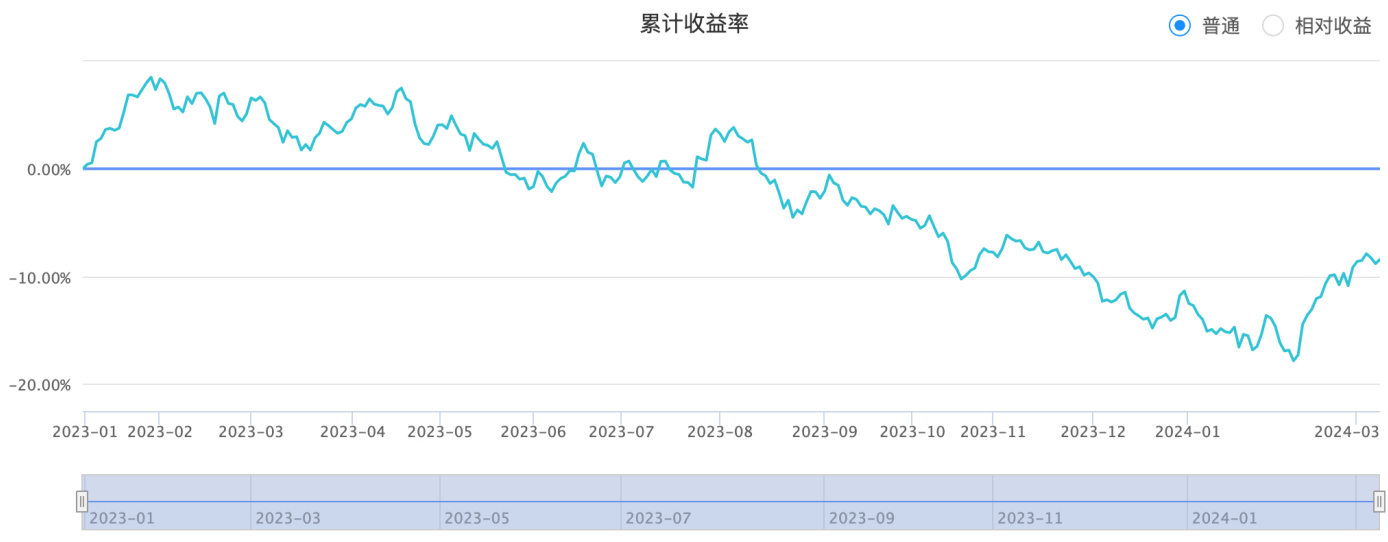
年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
0.0%	-8.0%	-3.0%	0.00	--
收益波动率	信息比率	最大回撤	回撤恢复时间	年化换手率
0.0%	0.36	0.0%	--	0.00%



[2]:
```
%%backtest
start = '2023-01-01'                    # 回测起始时间
end = '2024-03-09'                      # 回测结束时间
universe = StockUniverse('HS300')       # 证券池
benchmark = 'HS300'                     # 策略参考标准
freq = 'd'                              # 策略类型，'d'表示日间策略使用日线回测，'m'表示日内策略使用分钟线回测
refresh_rate = Weekly(2)
```

回测完成，耗时15.380 秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
0.0%	-7.4%	-3.0%	0.00	--
收益波动率	信息比率	最大回撤	回撤恢复时间	年化换手率
0.0%	0.52	0.0%	--	0.00%



[3]:
```
! python --version
```

```
Python 3.7.5
```

[4]:
```
#没有帮助
refresh_rate?
```

```
Type:           SpecialTradingDays
String form:    <quartz.utils.special_trading_days.SpecialTradingDays object at 0x7f50e68c2390>
File:           /usr/local/lib/python3.7/site-packages/mercuryq-quartz-4.200.0.post1635.egg/quartz/utils/special_trading_days.py
Docstring:      描述特殊交易日的类，用于处理特殊的freq，比如每周第3个交易日这种定义
Init docstring:
初始化，仅限于内部调用，不用检查输入

Args:
    freq (str): 频率，可以为'week'或者'month'
    indices (list of int): 序号，表示需要哪些序号的日期

Examples:
    >> special_trading_days = SpecialTradingDays('week', [2, 3])
    >> special_trading_days = SpecialTradingDays('month', [5, 6, 7, -7, -6, -5])
```

[5]:
```
?universe
```

```
Type:        StockUniverse
String form: <quartz.universe.stock_universe.StockUniverse object at 0x7f50e68c2150>
File:        /usr/local/lib/python3.7/site-packages/mercuryq-quartz-4.200.0.post1635.egg/quartz/universe/stock_universe.py
Docstring:   股票Universe
```

[6]:
```
?show_order
```

```
Signature: show_order(start='', end='')
Docstring:
查看回测的订单详情
:param start: [string] 订单的起始日期，默认是回测结束日期的向前30个自然日或者是end日期的向前30个自然日,格式: YYYY-MM-DD
:param end: [string] 订单的结束日期，默认是回测结束日期或者是start日期的向后30个自然日, 格式:  YYYY-MM-DD
:return: 返回start和end之间的订单列表
File:      /srv/data/notes_py3/<ipython-input-2-7c099d302ccb>
Type:      function
```

[7]:
```
?Weekly
```

```
Signature: Weekly(*args)
Docstring:
按序数取的每周交易日

Args:
    * (int): 所需序号

Returns:
    SpecialTradingDays: 包装好的SpecialTradingDays实例

Examples:
    >> refresh_rate = Weekly(1)
    >> refresh_rate = Weekly(2, -1)
File:      /srv/data/notes_py3/build/bdist.linux-x86_64/egg/quartz/utils/special_trading_days.py
Type:      function
```

[8]:
```
get_attribute_history?
```

```
Signature:
get_attribute_history(
    self,
    attribute,
    time_range,
    freq=None,
    rtype=None,
    f_adj=None,
    symbol=None,
    **kwargs,
)
Docstring: 获取单个数据变量的历史数据，日线和分钟线默认调用的方法不同，参数分别见相应方法
File:      /srv/data/notes_py3/build/bdist.linux-x86_64/egg/quartz/context/context.py
Type:      function
```

A Momentum Strategy

逻辑：

- 根据历史收益率表现对股票排名
- 过去表现好的，未来一段时间表现也好

细节：

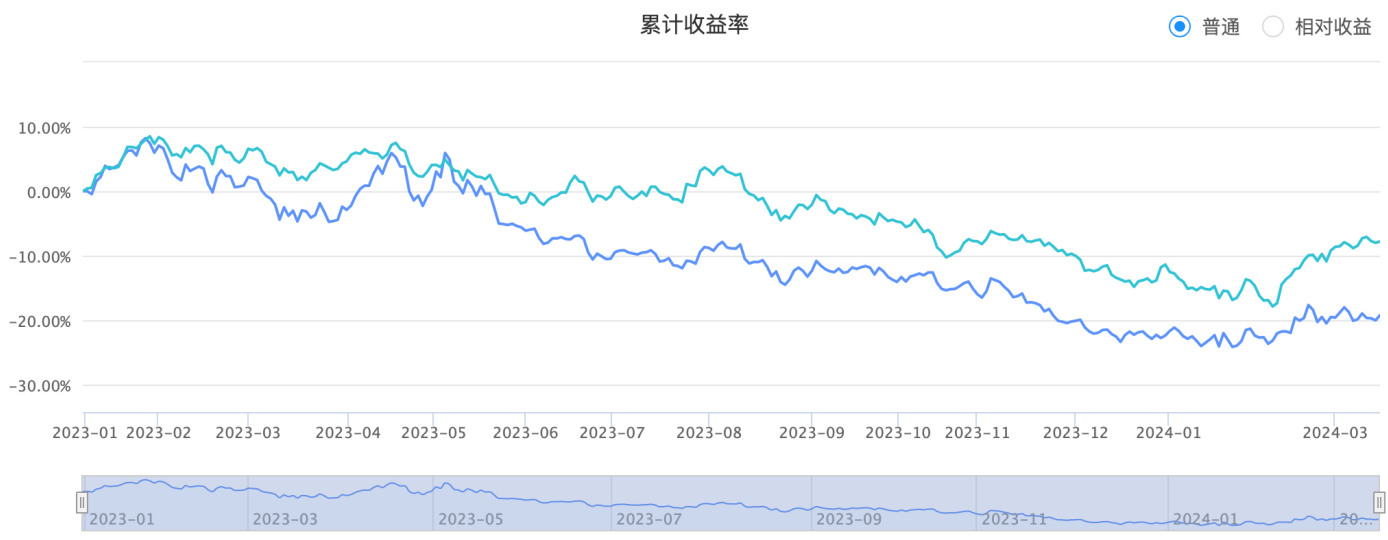
- 过去有多久？
- 未来有多久？
- 股票包括哪些？

[9]:
```
%%backtest
start = '2023-01-01'
end = '2024-03-17'
universe = StockUniverse('SH50')
benchmark = 'HS300'
freq = 'd' #以日数据回测，也即每天看一下收益率
refresh_rate = Weekly(2) #每周四调仓

accounts = {
    'stock_account': AccountConfig(account_type='security', capital_base=1e6)
}
```

```python
def initialize(context):
    """
    [帮助文档-->initialize策略初始化函数]
    策略初始化函数，用于配置策略运行环境context对象的属性或自定义各种变量。在策略运行周期中只执行一次。您可以通过给context添加新的属性，自定义各种变量。
    context在策略运行（回测或模拟交易）启动时被创建，持续整个策略的生命周期。策略运行时可以读取context的已有系统属性或者自定义属性。

    [帮助文档-->策略运行环境Context]
    context表示策略运行环境，包含运行时间、行情数据等内容，还可以用于存储策略中生成的临时数据。
    策略框架会在启动时创建context的对象示例，并以参数形式传递给initialize(context)和handle_data(context)，用于策略调度。
    回测时，context包含运行时间、回测参数、回测运行时数据等。模拟交易时，包含运行时间、模拟交易参数、实时运行数据等。
    """
#     context.amount = 300
    pass

def handle_data(context):
    """
    [帮助文档-->handle_data策略运行主函数]
    策略算法函数，策略运行时（回测或模拟交易），会根据初始化配置的策略算法运行频率调用该函数。策略算法可以通过context获取运行时的行情数据、K线图、因子数据、订单簿等数据，并根据分析的结果，通过交易账户进行订单委托。
    策略框架会根据实时的市场情况，进行订单撮合执行，在每日结束还会完成清算的操作。
    """
    current_universe = context.get_universe(exclude_halt=True)
#     print(current_universe)
#     print(len(current_universe))
#     pe_df = context.history(symbol='all', time_range=20, attribute=['PE'], freq='d', style='ast')
#     print(pe_df)
#     print(pe_df[list(pe_df.keys())[0]])
    price_dict = context.history(symbol='all', time_range=20, attribute=['closePrice'], freq='d', style='sat')
#     print(price_dict)
#     display(price_dict[list(price_dict.keys())[0]])
    momentum = {'symbol':[], 'cum_ret':[]}
    for stock in current_universe:
        if price_dict[stock].iloc[0,0]:
            momentum['symbol'].append(stock)
            momentum['cum_ret'].append(price_dict[stock].iloc[-1,0] / price_dict[stock].iloc[0,0])
#     print(momentum)
    momentum_df = pd.DataFrame(momentum).sort_values('cum_ret').reset_index()
#     display(momentum_df)
    buy_list = momentum_df.loc[np.floor(momentum_df.shape[0]*0.8):, 'symbol'].tolist() # momentum
#     print(buy_list)
#     buy_list = momentum_df.loc[:np.ceil(momentum_df.shape[0]*0.1), 'symbol'].tolist() # reversal

    stock_account = context.get_account('stock_account')
# #     print(dir(stock_account))
    current_positions = stock_account.get_positions(exclude_halt=True)
# #     print(current_positions)

    for stock in current_positions:
        if stock not in buy_list:
            stock_account.order_to(stock, 0)

    for stock in buy_list:
        stock_account.order_pct_to(stock, 0.10)
```

回测完成，耗时19.575 秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
-16.8%	-6.8%	-11.4%	0.86	-1.13
收益波动率	**信息比率**	**最大回撤**	**回撤恢复时间**	**年化换手率**
17.5%	-0.79	29.9%	--	4023.31%



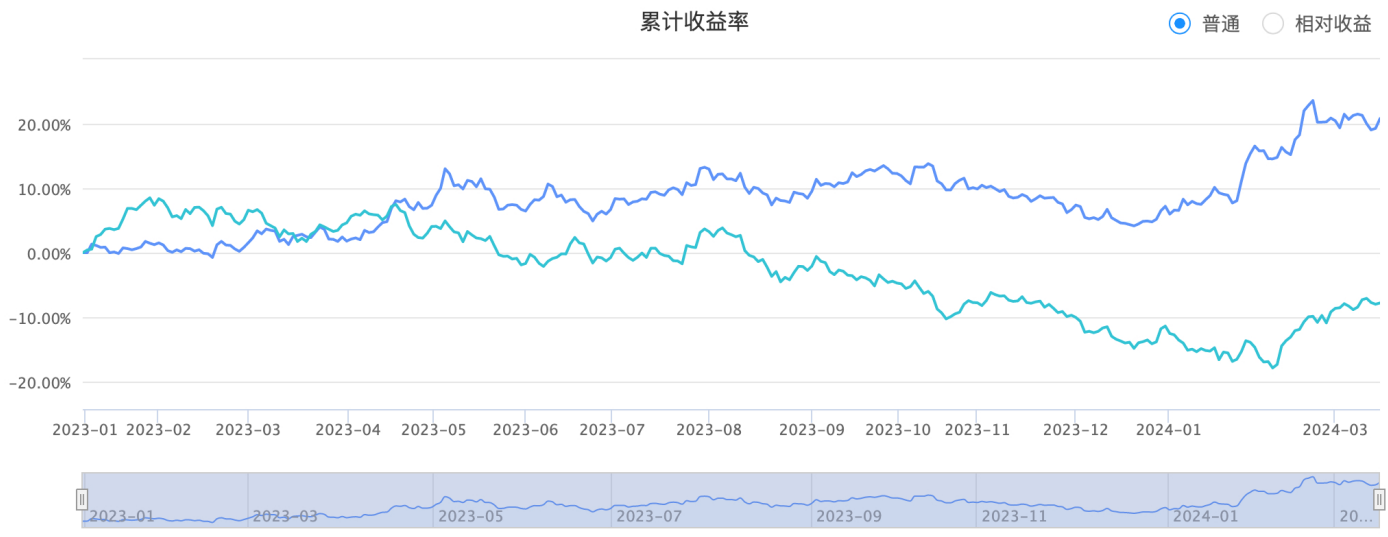
[10]:

```python
%%backtest
start = '2023-01-01'                    # 回测起始时间
```

```
start = '2023-01-01'                    # 回测起始时间
end = '2024-03-17'                      # 回测结束时间
universe = StockUniverse('HS300')       # 证券池
benchmark = 'HS300'                     # 策略参考标准
freq = 'd'                              # 策略类型，'d'表示日间策略使用日线回测，'m'表示日内策略使用分钟线回测
refresh_rate = Weekly(1)                        # 调仓频率，表示执行handle_data的时间间隔，若freq = 'd'时间间隔的单位为交易日，若freq = 'm'时间间隔为分钟

# 配置账户信息，支持多资产多账户
accounts = {
    'stock_account': AccountConfig(account_type='security', capital_base=10000000)
}

def initialize(context):
    pass

# 每个单位时间(如果按天回测,则每天调用一次,如果按分钟,则每分钟调用一次)调用一次
def handle_data(context):
    previous_date = context.previous_date.strftime('%Y-%m-%d')

    # 获取当天的可交易证券列表
    universe = context.get_universe(exclude_halt=True)

    # 获取因子PE的的历史数据，得到前一个交易日的横截面PE值
    history_data = context.history(symbol=universe, attribute='PE', time_range=1, style='tas')
    data = history_data.get(previous_date)

    # 将因子值从小到大排序，并取前10支股票作为目标持仓
    signal = data['PE'].sort_values(ascending=True)
    signal = signal[signal > 0]
    target_positions = signal[:10].index

    # 获取当前账户信息
    account = context.get_account('stock_account')
    current_position = account.get_positions(exclude_halt=True)
    for stock in set(current_position).difference(target_positions):
        account.order_to(stock, 0)

    for stock in target_positions:
        account.order_pct_to(stock,  1.0 / len(target_positions))
```

回测完成，耗时22.517 秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
17.7%	-6.8%	20.1%	0.56	1.12
收益波动率	**信息比率**	**最大回撤**	**回撤恢复时间**	**年化换手率**
13.2%	1.99	8.5%	25	813.14%



[11]:
```
# #查看调仓记录
show_order(start,end)
```

stock_account　查询日期: 2023-01-03 至 2024-03-11　导出EXCEL

资产代码	资产名称	业务类型	订单类型	委托价格	成交均价	委托数量	成交数量	成交金额	委托时间	成交时间	交易
600015	华夏银行	买入	市价单	市价	6.39	100	100	639.00	2024-03-11	2024-03-11	

601166	兴业银行	买入	市价单	市价	16.47	1,100	1,100	18,117.00	2024-03-11	2024-03-11
000001	平安银行	买入	市价单	市价	10.38	2,200	2,200	22,836.00	2024-03-11	2024-03-11
601818	光大银行	卖出	市价单	市价	3.26	300	300	978.00	2024-03-11	2024-03-11
601	上海									

每页20条，共 617 条 2 3 4 5 ••• 31 20 条/页 跳至 页

[12]:

```
# #查看持仓记录
show_position(start,end)
```

stock_account 查询日期: 2024-03-15 导出EXCEL

资产代码	资产名称	公允价格	持仓数量	可用数量	持仓市值	累计盈亏	持仓权重	摊薄成本
600015	华夏银行	6.29	190,100	190,100	1,195,729.00	288411.36	9.90%	4.77
601818	光大银行	3.29	373,800	373,800	1,229,802.00	152917.84	10.18%	2.88
601229	上海银行	6.61	183,500	183,500	1,212,935.00	192013.67	10.04%	5.56
601169	北京银行	5.46	222,900	222,900	1,217,034.00	338168.72	10.07%	3.94
601166	兴业银行	16.39	73,700	73,700	1,207,943.00	-24452.04	10.00%	16.72
601186	中国铁建	8.31	143,400	143,400	1,191,654.00	24085.89	9.86%	8.14
000001	平安银行	10.60	117,000	117,000	1,240,200.00	18910.71	10.27%	10.44
601838	成都银行	13.17	89,900	89,900	1,183,983.00	67182.54	9.80%	12.42
601668	中国建筑	5.21	229,200	229,200	1,194,132.00	106700.85	9.88%	4.74
600919	江苏银行	7.69	156,700	156,700	1,205,023.00	63915.42	9.97%	7.28
cash	现金	--	2,682.47	--	2,682.47	--	0.02%	--

每页20条，共 11 条 20 条/页

Type Markdown and LaTeX: α^2