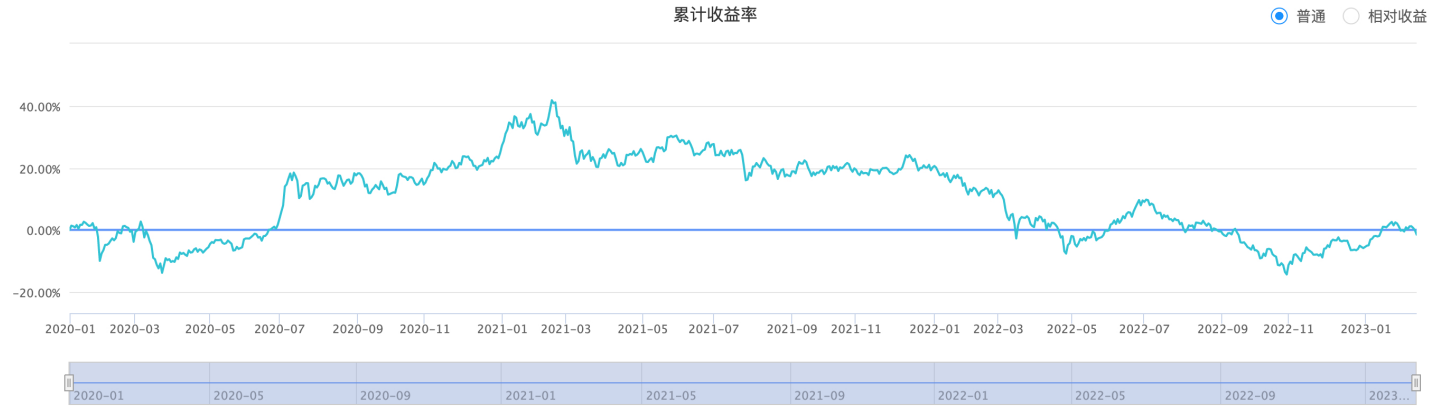
Check some variables

[2]:
```
%%backtest
start = '2020-01-01'                    # 回测起始时间
end = '2023-02-17'                      # 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)
```

回测完成，耗时39.591 秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
0.0%	-0.5%	-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%	0.02	0.0%	--	0.00%



[3]:
```
dir()
```

[3]:
```
['AccountConfig',
 'AlphaHorizon',
 'AssetType',
 'BacktestDetailObject',
 'BondUniverse',
 'Commission',
 'DataAPI',
 'DynamicUniverse',
 'FUNDDY',
 'FUNDDYV2',
 'Factor',
 'FundFactor',
 'FundUniverse',
 'IdxCN',
 'In',
 'IndSW',
 'IndZJH',
 'IndZZ',
 'Monthly',
 'Optimizer',
 'OptionsUniverse',
 'OrderObject',
 'OrderState',
 'OrderStatus',
 'Out',
 'PRESET_KEYARGS',
 'PortfolioObject',
 'PositionObject',
 'SecurityObject',
 'Signal',
 'SignalGenerator',
 'Slippage',
 'StockFactor',
 'StockScreener',
 'StockUniverse',
 'Weekly',
 '_',
 '__',
 '___',
 '__builtin__',
 '__builtins__',
 '__doc__',
 '__loader__',
 '__name__',
 '__orders__',
 '__package__',
 '__positions__',
 '__reduce_date_items__',
 '__spec__',
 '__start_time__',
 '_dh',
 '_i',
 '_i1',
 '_i2',
 '_i3',
 '_ih',
 '_ii',
 '_iii',
 '_oh',
 'accounts',
 'auth_url',
 'backtest',
 'benchmark',
 'bt',
 'bt_by_account',
 'bt_detail_define_code',
 'bt_detail_format_code',
 'cal_covariance',
 'cal_factor_return',
 'cal_srisk',
 'cancel_order',
 'capital_base',
```

```
'contest',
'create_factor_tear_sheet',
'datayes_risk_model',
'datetime',
'dp',
'e',
'end',
'evaluate_model',
'exit',
'f',
'format_multi_asset_bt',
'freq',
'gen_drawdown_table',
'get_account',
'get_asset',
'get_attribute_history',
'get_env',
'get_ipython',
'get_order',
'get_orders',
'get_symbol_history',
'get_universe',
'handle_data',
'headers',
'history',
'initialize',
'json',
'key',
'last_perf',
'log',
'logging',
'long_only',
'long_short',
'max_history_window',
'nav_analysis',
'neutralize',
'neutralize_pit',
'normalize_code',
'normalize_l1',
'np',
'observe',
'order',
'order_pct',
'order_pct_to',
'order_to',
'os',
'pd',
'perf',
'perf_chart',
'perf_parse',
'plot_drawdown_periods',
'plot_drawdown_underwater',
'plot_monthly_returns_dist',
'plot_monthly_returns_heatmap',
'plot_pure_alpha',
'pyuqer',
'quit',
're',
'refresh_rate',
'register_cell_magic',
'requests',
'response',
'result',
'security_base',
'security_cost',
'set_universe',
'show_order',
'show_position',
'simple_long_only',
'simulation_parameter',
'standardize',
'start',
'sys',
'test_package',
'threaded',
'time',
'universe',
'value',
'variables',
'warnings',
'winsorize',
'with_post_trading_day']
```

[10]:
```
#没有帮助
?refresh_rate
```

```
Type:        int
String form: 1
Docstring:
int([x]) -> integer
int(x, base=10) -> integer

Convert a number or string to an integer, or return 0 if no arguments
are given.  If x is a number, return x.__int__().  For floating point
numbers, this truncates towards zero.

If x is not a number or if base is given, then x must be a string,
bytes, or bytearray instance representing an integer literal in the
given base.  The literal can be preceded by '+' or '-' and be surrounded
by whitespace.  The base defaults to 10.  Valid bases are 0 and 2-36.
Base 0 means to interpret the base from the string as an integer literal.
>>> int('0b100', base=0)
4
```

[11]:
```
?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-f2b1e6424543>
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?

[18]:
```
history?
```

```
Signature:
history(
    self,
    symbol='all',
    attribute='closePrice',
    time_range=1,
    freq='d',
    style='sat',
    rtype='frame',
    f_adj=None,
    s_adj='pre_adj',
    **options,
)
Docstring: <no docstring>
File:      /srv/data/notes_py3/build/bdist.linux-x86_64/egg/quartz/context/context.py
Type:      function
```

[39]:
```
%%backtest
start = '2022-01-01'
end = '2023-02-18'
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(4) #每周四调仓

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

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17.496 秒

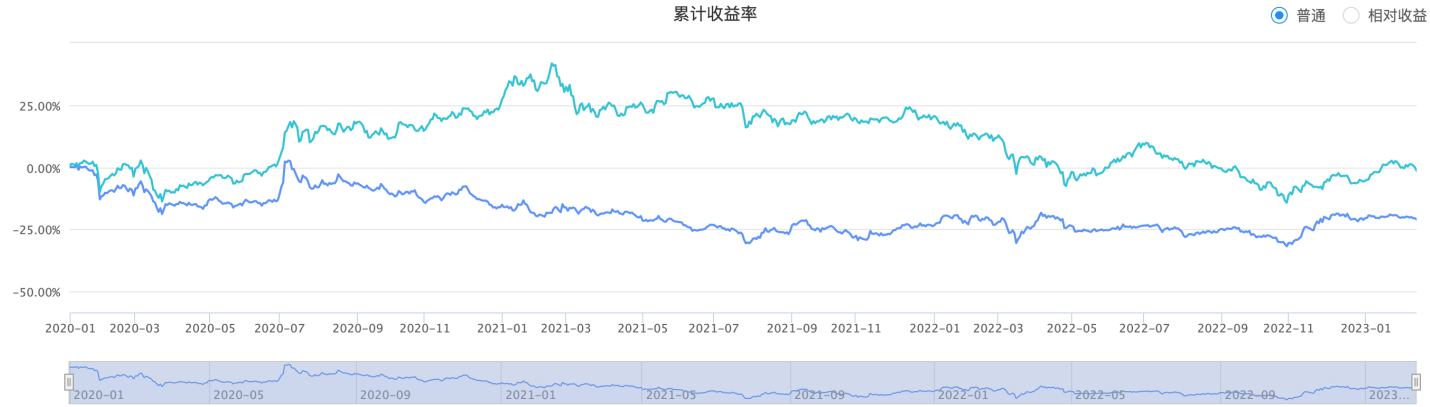
年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
-9.5%	-17.0%	4.1%	0.83	-0.56
收益波动率	**信息比率**	**最大回撤**	**回撤恢复时间**	**年化换手率**
22.5%	0.48	23.2%	--	3961.55%

[36]:

```
%%backtest
start = '2020-01-01'                        # 回测起始时间
end = '2023-02-18'                          # 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56.799秒

年化收益率	基准年化收益率	阿尔法	贝塔	夏普比率
-7.5%	-0.5%	-8.5%	0.58	-0.57
收益波动率	**信息比率**	**最大回撤**	**回撤恢复时间**	**年化换手率**
18.3%	-0.42	33.6%	--	870.33%



[37]:

```
# #查看调仓记录
show_order(start,end)
```

stock_account　查询日期: 2020-01-06 至 2023-02-13　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	交易费用	订单状态
601919	中远海控	卖出	市价单	市价	10.59	2,700	2,700	28,593.00	2023-02-06	2023-02-06	57.19	全部成交
601818	光大银行	买入	市价单	市价	3.02	2,300	2,300	6,946.00	2023-02-06	2023-02-06	6.95	全部成交
601229	上海银行	买入	市价单	市价	5.94	300	300	1,782.00	2023-02-06	2023-02-06	1.78	全部成交
601169	北京银行	卖出	市价单	市价	4.24	1,300	1,300	5,512.00	2023-02-06	2023-02-06	11.02	全部成交
601328	交通银行	卖出	市价单	市价	4.83	1,300	1,300	6,279.00	2023-02-06	2023-02-06	12.56	全部成交
601166	兴业银行	买入	市价单	市价	17.18	1,300	1,300	22,334.00	2023-02-06	2023-02-06	22.33	全部成交
601668	中国建筑	买入	市价单	市价	5.37	2,200	2,200	11,814.00	2023-02-06	2023-02-06	11.81	全部成交