# PyBarra 设计手册 - v0.1

2024 年 2 月 3 日

版本：V0.1，日期：2024.2.3

| 版本 | 日期 | 改动内容 |
| --- | --- | --- |
| v0.0 | 2024.1.2 | 初次编写 |
| v0.1 | 2024.2.3 | 对模块进行解耦，使类与类之间信息传递由格式化文件进行而非类的传递，文档中加入了一些图片辅助理解，加入了基于 tushare 的数据和因子库，包含国家，行业，规模，波动率，流动性，动量等数十个因子，加入了 alphalens 对因子进行分析，加入了 BackTrader 对投资组合回测，修改了模型之间配合的逻辑，使其基于 csv 表格配合，移除了代码结构展示 |

目录

# 1 项目概述

## 1.1 摘要

**PyBarra** 是一个旨在实现 **BARRA 风险模型**的 Python 库。它提供了一套工具，允许用户进行复杂的金融风险分析和投资组合管理。这个库利用了现代金融理论和数学模型来帮助投资者和风险管理者更好地理解市场风险和投资组合的性能。

## 1.2 应用场景

[1]:
```
from IPython.display import Image
Image(filename = '../../../420 Reports/Applications.png',embed=True, width=700)
```

[1]:


## 1.3 功能概述

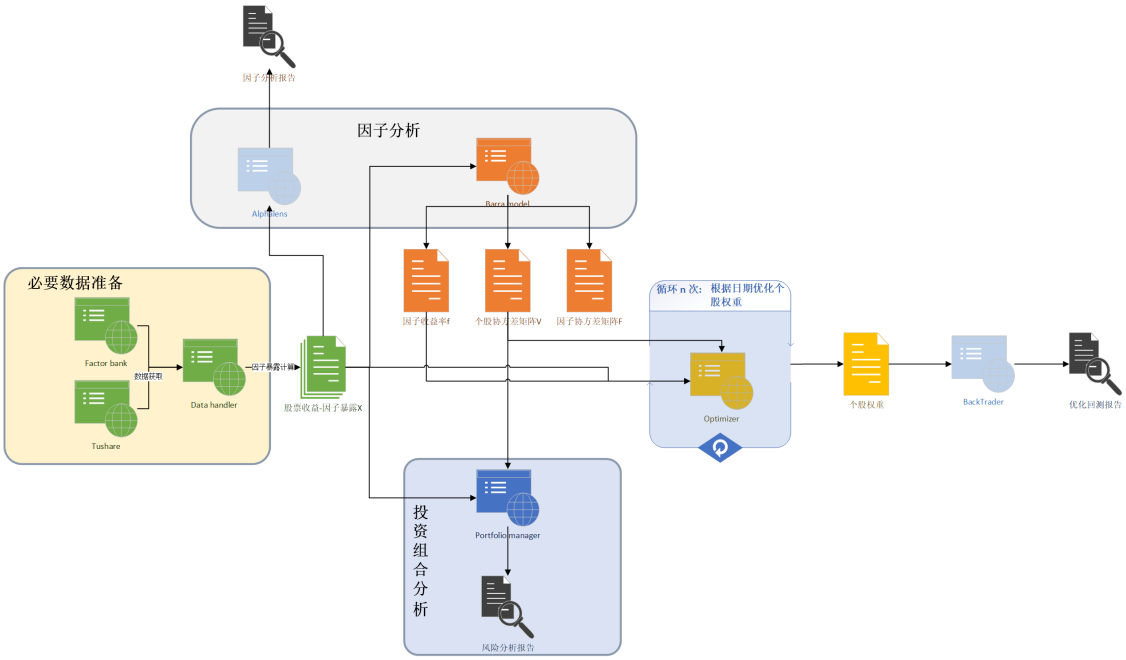
1. **数据处理**：PyBarra 能够处理和分析各种金融数据，包括股票价格、市场指数、财务指标等。
2. **风险因子建模**：库提供了一种方法来识别和计算不同的风险因子，如市场风险、行业风险、风格风险，并且能够通过历史数据进行截面回归求解模型等
3. **协方差矩阵**：通过估计不同资产之间的协并进行相应的矩阵优化调整方差，PyBarra 能够预测整体投资组合的风
4. **风险估计**：生成详细的风险分析报告，包括图表和关键指标，以帮助用户做出明智的投资决策。
5. **投资组合优化**：基于风险模型，PyBarra 提供了工具来优化投资组合的配置。

6. 策略回测：基于 BackTrader，PyBarra 提供接口供用户回测以某种策略优化后的投资组合。

### 1.4 模块总览

[2]:
```
from IPython.display import Image
Image(filename = '../../../420 Reports/模型架构.png',embed=True, width=800)
```

[2]:




### 1.5 使用方法

PyBarra 设计以用户为中心，通过高度集成化的代码帮助用户通过数行简单的代码实现复杂的 BARRA 模型风险分析。库的 API 文档详尽完整，包含示例代码和教程，使用户能够快速上手并开始自己的风险分析。

## 2 功能规划

### 2.1 数据处理

PyBarra 可以通过 tushare 导入和处理金融数据，并计算 BARRA 模型的因子暴露度。此外，还可以通过 Alphalens 进行因子分析

### 2.2 风险因子模型

根据得到的因子暴露度数据，进行截面回归，求解因子收益率和特质收益率。用户还可以调用 Alphalens 对因子进行分析。

### 2.3 投资组合风险估计

根据回归获得的因子收益率和特质收益率数据，计算表示因子风险的因子收益协方差矩阵和表示特质风险的特质风险对角矩阵。

通过 Newey-West 调整，特征值调整，波动率偏误调整对因子收益协方差矩阵进行调整和优化，似的其对风险的估计更加准确。

最终，通过因子风险和特质风险实现对任意的投资组合的准确的风险估计。

### 2.4 投资组合优化

使用优化算法优化投资组合里个股权重实现在不同限制条件下的风险控制。

比如，限制对行业风险暴露，或某风格因子风险暴露小于特定值的最优投资组合等。

### 2.5 策略回测

对于通过指定策略优化后的组合，模型可以生成时间序列上的投资组合成分股权重表格 trade_info.csv，并根据表格进行 BackTrader 回测。

## 3 应用实例

### 3.1 导入 pyBarra

[3]:
```
import os
path = 'G:\\My Drive\\400 Internship\\440 Programe projects\\442 Barra-Model'
os.chdir(path)

from pyBarra.data_handling import DataHandler
from pyBarra.portfolio_management import PortfolioManager
from pyBarra.barra_model import BarraModel
from pyBarra.optimizer import Optimizer
from pyBarra.utils import *
from pyBarra.alphalens_analysis import alphalens_analyze
```

```
from pyBarra.back_trader import *

import pandas as pd
pd.set_option("display.max_columns", None)
```

### 3.2 读取数据

[4]:
```
data_handler = DataHandler('data/')

data,meta_dict = data_handler.load_data(range_years=(2015,2022),␣
↪full_data_only=True, isupdate=False)
```

```
Factor exposures data is already up to date.
```

[5]:
```
data.head()
```

[5]:

| | trade_date | ts_code | capital | ret | country | Industry_ 专用机械 | Industry_ 中成药 | Industry_ 互联网 | Industry_ 保险 | Industry_ 元器件 | Industry_ 其他建材 | Industry_ 农用机械 | Industry_ 农药化肥 | Industry_ 化学制药 | Industry_ 化工原料 | Industry_ 区域地产 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2016-02-19 | 600560.SH | 3.227205e+05 | 0.0077 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2016-02-19 | 002294.SZ | 1.262745e+06 | 0.0285 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 2016-02-19 | 000680.SZ | 5.499825e+05 | 0.0122 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2016-02-19 | 600298.SH | 1.166139e+06 | 0.0313 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 2016-02-19 | 002095.SZ | 1.553655e+06 | 0.0259 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
4                0                0                0                0                0

   Industry_ 医疗保健  Industry_ 啤酒  Industry_ 塑料  Industry_ 家用电器 ␣
↪Industry_ 小金属  \
0                0                0                0                0                0
1                0                0                0                0                0
2                0                0                0                0                0
3                0                0                0                0                0
4                0                0                0                0                0

   Industry_ 工程机械  Industry_ 广告包装  Industry_ 建筑工程  Industry_ 影视音像␣
↪ Industry_ 摩托车  \
0                0                0                0                0                0
1                0                0                0                0                0
2                1                0                0                0                0
3                0                0                0                0                0
4                0                0                0                0                0

   Industry_ 机场  Industry_ 水力发电  Industry_ 水泥  Industry_ 汽车整车 ␣
↪Industry_ 汽车配件  \
0                0                0                0                0                0
1                0                0                0                0                0
2                0                0                0                0                0
3                0                0                0                0                0
4                0                0                0                0                0

   Industry_ 港口  Industry_ 火力发电  Industry_ 煤炭开采  Industry_ 特种钢 ␣
↪Industry_ 玻璃  \
0                0                0                0                0                0
1                0                0                0                0                0
2                0                0                0                0                0
3                0                0                0                0                0
4                0                0                0                0                0

   Industry_ 电器仪表  Industry_ 电气设备  Industry_ 白酒  Industry_ 百货 ␣
↪Industry_ 石油加工  \
```

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| | Industry_ 石油开采 | Industry_ 石油贸易 | Industry_ 红黄酒 | Industry_ 纺织 | Industry_ 纺织机械 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| | Industry_ 航空 | Industry_ 路桥 | Industry_ 软件服务 | Industry_ 运输设备 | Industry_ 通信设备 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| | Industry_ 银行 | Industry_ 食品 | LNCAP | MIDCAP | BETA | HSIGMA | DASTD |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | -0.857504 | -0.808212 | 0.438817 | 1.054376 | 0.934932 |
| 1 | 0 | 0 | 0.162090 | 0.832750 | -1.395009 | -1.162452 | -1.398277 |
| 2 | 0 | 0 | -0.459085 | -0.974880 | 1.128639 | 0.073733 | 0.402369 |
| 3 | 0 | 1 | 0.102607 | 0.055568 | -1.233278 | -0.184876 | -0.624077 |
| 4 | 0 | 0 | 0.317035 | 0.800669 | 0.518225 | 2.874360 | 2.331352 |

| | CMRA | STOM | STOQ | STOA | ATVR | STREV | SEASON |
|---|---|---|---|---|---|---|---|
| 0 | 1.271312 | 0.603752 | 0.616327 | 0.653506 | 0.711837 | -1.396835 | 1.082611 |
| 1 | -1.583901 | -0.474919 | -0.541687 | -0.584412 | -0.675309 | -0.318683 | -0.126288 |
| 2 | 0.308684 | 0.692760 | 0.701366 | 0.767042 | 0.836114 | -0.709894 | -0.456811 |
| 3 | -0.781988 | 0.677372 | 0.668748 | 0.655636 | -0.349826 | 0.863576 | 0.753122 |
| 4 | 2.787139 | 1.134939 | 1.152651 | 1.124464 | 1.059079 | 0.591313 | 2.660852 |

```
       RSTR    HALPHA
0 -1.101647  1.075918
1 -0.086608 -0.139754
2 -1.345985 -0.588674
3  0.896868  1.043775
4 -0.272874  3.415458
```

### 3.3 创建 Barra 模型实例

[6]:
```
barra_model = BarraModel(data, meta_dict)
```

```
截面回归中…: 100%|                                        |
1674/1674 [00:15<00:00, 107.27it/s]
计算因子协方差矩阵中…: 100%|                                |
1674/1674 [05:03<00:00,  5.52it/s]
计算股票协方差矩阵中…: 100%|                                |
1674/1674 [05:55<00:00,  4.70it/s]
```

### 3.4 创建投资组合实例

[7]:
```
portfolio_manager = PortfolioManager(data, barra_model.V_dict, barra_model.
 ↪meta_dict)


portfolio_manager.set_weights(method='capital')
```

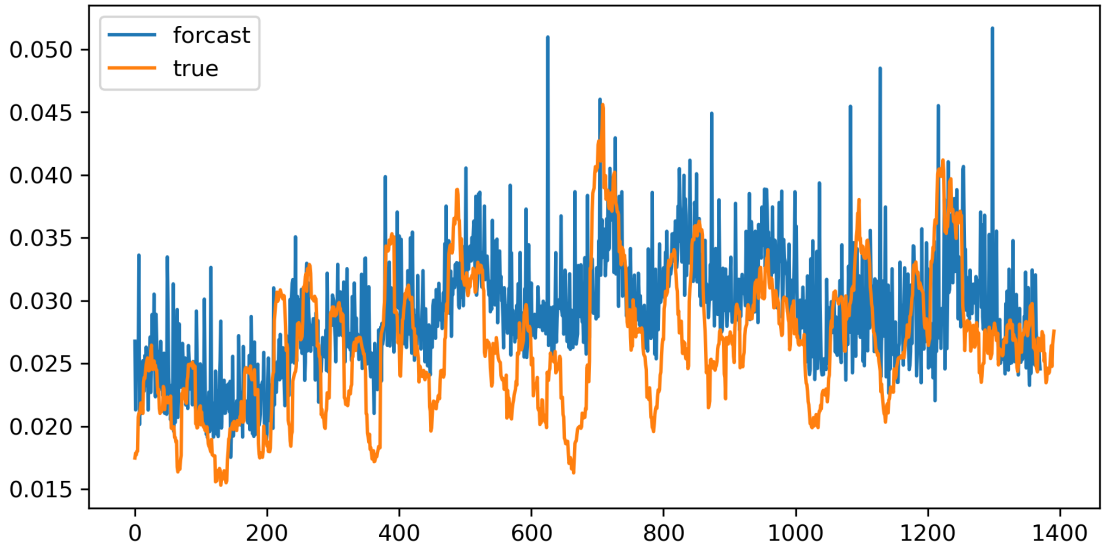
预测投资组合风险

[8]:
```
import matplotlib.pyplot as plt


forcast_risk = []
true_risk = []
for date in portfolio_manager.meta_dict['date_list'][252:-30]:
    forcast_risk.append(np.sqrt(portfolio_manager.
 ↪estimate_portfolio_risk(date)))
    true_risk.append(portfolio_manager.get_true_risk(date, 21))


plt.figure(figsize=(8,4), dpi=300)
```

```
plt.plot(forcast_risk[21:], label='forcast')
plt.plot(true_risk[:], label='true')
plt.legend()
plt.show()
```




### 3.5 对投资组合进行优化

在优化投资组合前，先查看一下未经优化的投资组合的收益和风险。

[9]:
```
date = portfolio_manager.meta_dict['date_list'][400]

print("未优化的投资组合的收益为: ")
print(portfolio_manager.get_portfolio_return(date))
```

未优化的投资组合的收益为:

0.005653813237681406

[10]:
```
print("未优化的投资组合的风险为: ")
print(portfolio_manager.get_true_risk(date))
```

未优化的投资组合的风险为:

0.02026025251870652

接下来，以投资组合风险最小化为目标进行优化，同时，限制投资组合：1. 权重之和为 1 2. 对每个成分股的权重限制在（0,0.5）之间 3. 成分股数量为 30 个

[11]:
```
X = data.loc[data['trade_date']==date, portfolio_manager.
 ↪meta_dict['common_factors']]
f = barra_model.factor_ret_df.loc[date]
V = barra_model.calculate_stock_covariance(date)

optimizer = Optimizer(portfolio_manager.meta_dict)

# 每个成分股权重限制在（0,0.5）之间
optimizer.set_boundary_limit(limit=(0,0.5))

# 权重之和限制为 1
optimizer.cons_sum_of_weights(1)

# 成分股数量为 30 个
optimizer.cons_num_of_stocks(30)

optimizer.target_min_risk()

w = optimizer.optimize(X,f,V)

portfolio_manager.W = w
```

优化后，我们再次验证投资组合的风险：

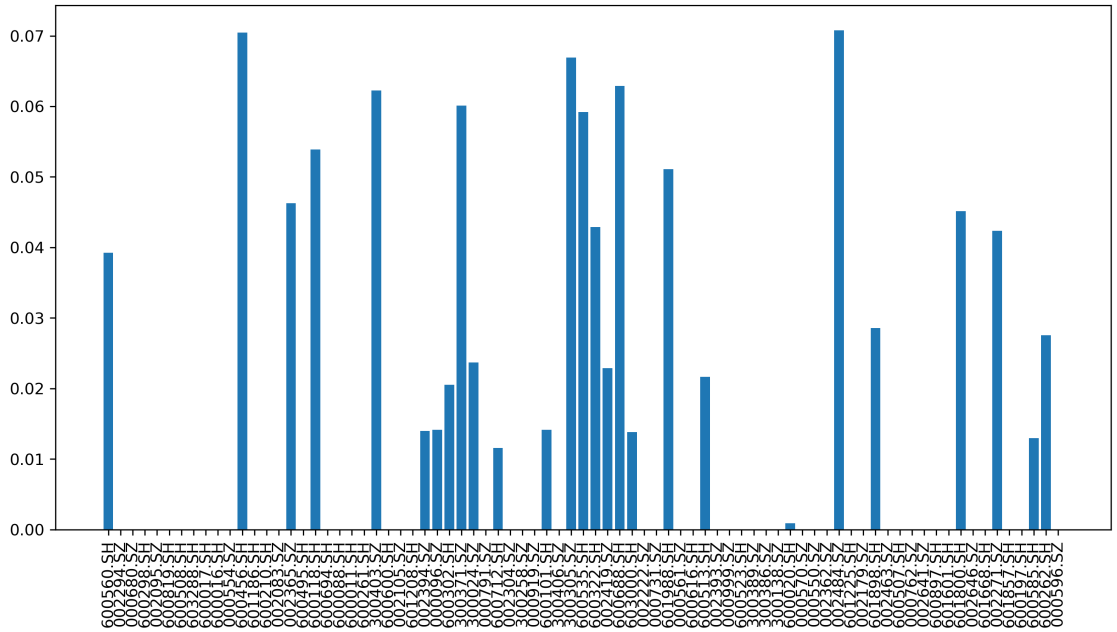
[12]:
```
print("优化后的投资组合的风险为：")
print(portfolio_manager.get_true_risk(date))
```

```
优化后的投资组合的风险为：
0.019956945583768503
```

可以看到，经过优化后，投资组合的风险降低了。

接下来，我们再来验证一下投资组合各成分股的权重，成分股数量。

[13]:
```
portfolio_manager.visualize_weights()
```




上图显示了投资组合里各成分股的权重，权重被限制在了 0~0.5 之间且仅有 30 支个股有非 0 的权重

接下来，以投资组合收益最大化为目标进行优化，同时，限制投资组合：

1. 权重之和为 1
2. 对 BETA 因子暴露为 0.5
3. 对 LNCAP 因子暴露为 0.5
4. 对医疗保健行业因子暴露为 0

[14]:
```
optimizer = Optimizer(portfolio_manager.meta_dict)

# 设置投资组合权重之和
optimizer.cons_sum_of_weights(1)

# 设置投资组合权重范围
optimizer.set_boundary_limit(limit=(0,0.5))

# 设置投资组合对 beta 因子暴露为 0.5
optimizer.cons_factor_exposure('BETA', 0.5)
```

```
# 设置投资组合对 size 因子暴露为 1
optimizer.cons_factor_exposure('LNCAP', 0.5)

# 设置投资组合对医疗行业暴露为 0
optimizer.cons_factor_exposure('Industry_ 医疗保健', 0)

# 设置投资组合优化目标为最大化收益
optimizer.target_max_ret()

# 开始优化
w = optimizer.optimize(X,f,V)

portfolio_manager.W = w
```

优化完成后查看优化后的投资组合收益

[15]:
```
print("优化后的投资组合的收益为: ")
print(portfolio_manager.get_portfolio_return(date))
```

```
优化后的投资组合的收益为:
0.05875872220463486
```

可以看到，优化后的投资组合收益有明显提高，接下来验证优化后的投资组合对各因子的暴露

[16]:
```
print("优化后的投资组合的对各各因子的暴露: ")
print(portfolio_manager.get_portfolio_exposure(date))
```

```
优化后的投资组合的对各各因子的暴露:
   country  Industry_ 专用机械  Industry_ 中成药  Industry_ 互联网  Industry_ 保险  \
0      1.0   1.466047e-16  5.506936e-17          0.0  3.577979e-17

   Industry_ 元器件  Industry_ 其他建材  Industry_ 农用机械  Industry_ 农药化肥  Industry_ 化学制药  \
0  4.391882e-17   3.757999e-17          0.0          0.0  1.374952e-16

   Industry_ 化工原料  Industry_ 区域地产  Industry_ 医疗保健  Industry_ 啤酒  Industry_ 塑料  \
0          0.0          0.0          0.0  1.276579e-18          0.0
```

```
   Industry_ 家用电器  Industry_ 小金属  Industry_ 工程机械  Industry_ 广告包装 ␣
↪Industry_ 建筑工程  \
0   7.966413e-17  4.707208e-20   3.080091e-17            0.0       0.377414

   Industry_ 影视音像  Industry_ 摩托车  Industry_ 机场  Industry_ 水力发电 ␣
↪Industry_ 水泥  \
0           0.5  1.229669e-16  5.802457e-17   3.238143e-17  1.576054e-17

   Industry_ 汽车整车  Industry_ 汽车配件  Industry_ 港口  Industry_ 火力发电 ␣
↪Industry_ 煤炭开采  \
0           0.0   1.044416e-17          0.0            0.0       0.075618

   Industry_ 特种钢  Industry_ 玻璃  Industry_ 电器仪表  Industry_ 电气设备 ␣
↪Industry_ 白酒  \
0          0.0  2.820477e-18            0.0   5.558607e-17          0.0

    Industry_ 百货  Industry_ 石油加工  Industry_ 石油开采  Industry_ 石油贸易 ␣
↪Industry_ 红黄酒  \
0  5.007566e-17            0.0   1.421375e-17            0.0  8.808148e-18

    Industry_ 纺织  Industry_ 纺织机械  Industry_ 航空  Industry_ 路桥  Industry_
软件服务  \
0  4.648685e-18       0.017531          0.0          0.0            0.0

   Industry_ 运输设备  Industry_ 通信设备  Industry_ 银行  Industry_ 食品  LNCAP ␣
↪  MIDCAP  \
0           0.0            0.0     0.029437          0.0     0.5 -0.440624

   BETA    HSIGMA     DASTD      CMRA     STOM      STOQ      STOA      ATVR  \
0   0.5  0.077347  0.151222 -0.201579 -0.5789 -0.577927 -0.590149 -0.646959

      STREV    SEASON     RSTR    HALPHA
0 -0.137486 -0.260677 -0.28484 -0.263822
```

可以看到，对因子的暴露满足我们的要求，对 BETA 的暴露为 0.5，对 LNCAP 的暴露为 0.5，对医疗保健行业的暴露为 0

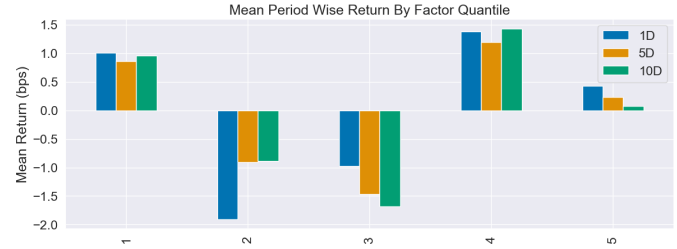
### 3.6 使用 **Alphalens** 进行因子分析

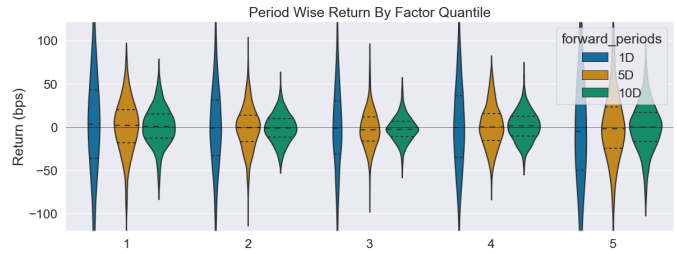
[17]:
```
data1 = data[:2000*len(portfolio_manager.meta_dict['stock_list'])]
alphalens_analyze(data1, 'LNCAP', data_handler.daily[data_handler.
↪daily['ts_code'].isin(portfolio_manager.meta_dict['stock_list'])])
```

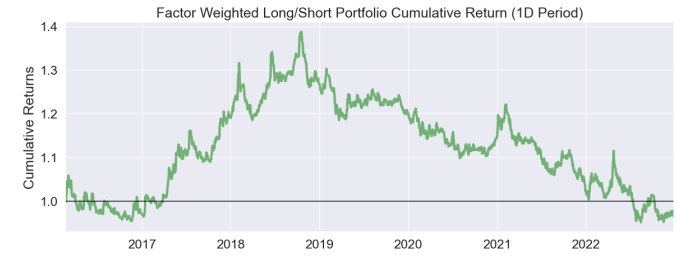
```
Dropped 0.0% entries from factor data: 0.0% in forward returns computation and
0.0% in binning phase (set max_loss=0 to see potentially suppressed Exceptions).
max_loss is 35.0%, not exceeded: OK!
Returns Analysis
```

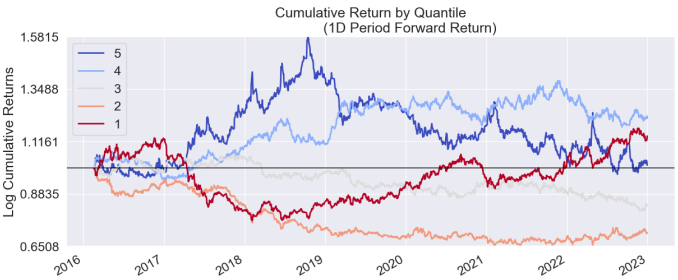
| | 1D | 5D | 10D |
|---|---|---|---|
| Ann. alpha | 0.012 | 0.012 | 0.012 |
| beta | -0.224 | -0.262 | -0.263 |
| Mean Period Wise Return Top Quantile (bps) | 0.425 | 0.235 | 0.075 |
| Mean Period Wise Return Bottom Quantile (bps) | 1.008 | 0.856 | 0.956 |
| Mean Period Wise Spread (bps) | -0.583 | -0.767 | -1.004 |

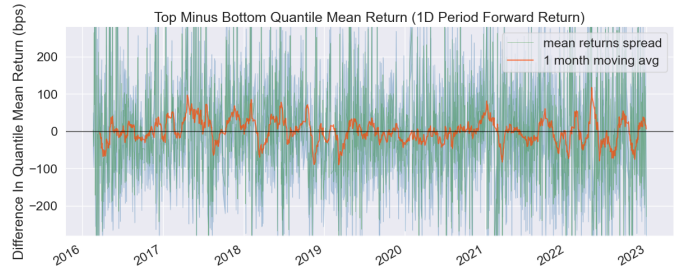
```
<Figure size 500x300 with 0 Axes>
```

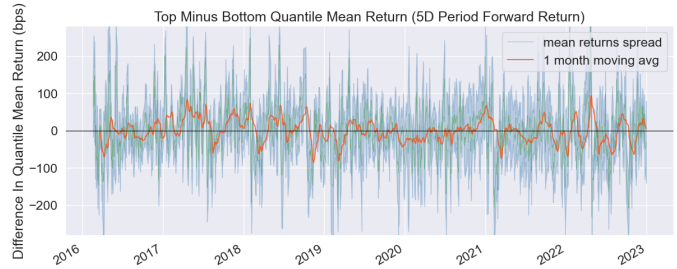

Mean Period Wise Return By Factor Quantile
Mean Return (bps)
1D
5D
10D
1
2
3
4
5

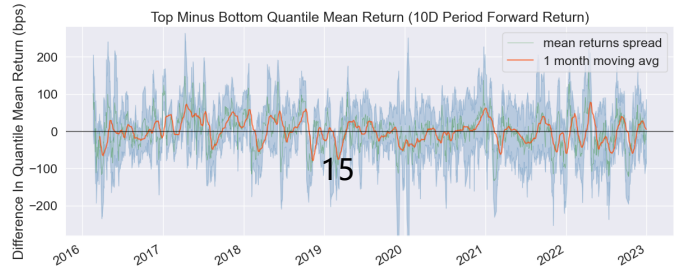

Period Wise Return By Factor Quantile
Return (bps)
forward_periods
1D
5D
10D
1
2
3
4
5


Factor Weighted Long/Short Portfolio Cumulative Return (1D Period)
Cumulative Returns
2017
2018
2019
2020
2021
2022


Cumulative Return by Quantile
(1D Period Forward Return)
Log Cumulative Returns
5
4
3
2
1


Top Minus Bottom Quantile Mean Return (1D Period Forward Return)
Difference In Quantile Mean Return (bps)
mean returns spread
1 month moving avg


Top Minus Bottom Quantile Mean Return (5D Period Forward Return)
Difference In Quantile Mean Return (bps)
mean returns spread
1 month moving avg


Top Minus Bottom Quantile Mean Return (10D Period Forward Return)
Difference In Quantile Mean Return (bps)
mean returns spread
1 month moving avg

### 3.7 使用 backtrader 进行回测

首先测试基准投资组合收益，以市值作为组合里成分股的权重。我们以 10000 元为起始。因为目前模型是以日频进行回归和优化的，如果设置佣金和滑点结果将非常可怜，所以暂时将佣金和滑点设置为 0，后续对模型进行完善可以实现月频回归和优化后可以再次尝试加入佣金和滑点。

[18]:
```
# 设置投资组合以成分股市值为权重
portfolio_manager.set_weights(method='capital')

# 生成交易信息
df_base = pd.DataFrame({
    'trade_date':portfolio_manager.meta_dict['date_list'][252],
    'ts_code':portfolio_manager.meta_dict['stock_list'],
    'weight': portfolio_manager.W.flatten()
    })

# 进行 backtrader 回测分析
back_trader(df_base, data_handler.daily[data_handler.daily['ts_code'].
 ↪isin(portfolio_manager.meta_dict['stock_list'])])
```

```
转换股票数据格式: 100%|                                          |
79/79 [00:00<00:00, 130.75it/s]

初始投资组合价值: 10000.00
最终投资组合价值: 9534.50
```

可以看到，经过 5 年时间，基准投资组合最终价值为 9534.5 元。

接下来测试模型优化后的投资组合

[19]:
```
dfs = []
for i,date in enumerate(portfolio_manager.meta_dict['date_list'][252:-30]):
    if i%1 ==0: # 每一个天调整一次频率
        X = data.loc[data['trade_date']==date, portfolio_manager.
 ↪meta_dict['common_factors']]
        f = barra_model.factor_ret_df.loc[date]
```

```
        V = barra_model.calculate_stock_covariance(date)

        w = optimizer.optimize(X,f,V)
        df_temp = pd.DataFrame({
            'trade_date': [date] * portfolio_manager.meta_dict['N'],
            'ts_code': portfolio_manager.meta_dict['stock_list'],
            'weight': w.flatten()
        })

        dfs.append(df_temp)

df = pd.concat(dfs, axis=0).reset_index(drop=True)
```

[20]:
```
back_trader(df, data_handler.daily[data_handler.daily['ts_code'].
↪isin(portfolio_manager.meta_dict['stock_list'])])
```

```
转换股票数据格式: 100%|                                                  |
79/79 [00:00<00:00, 128.15it/s]

初始投资组合价值: 10000.00
最终投资组合价值: 19964.69
```

最终投资组合价值为 19964 元。

# 4 技术架构

## 4.1 开发环境

- Python 3.11

## 4.2 主要依赖库

- Pandas 和 Numpy：数据处理和矩阵运算
- SciPy：优化算法
- Alphalens: 因子分析报告
- BackTrader: 回测分析
- Matplotlib：可视化
- Tushare：数据获取

## 4.3 模块设计

[2]:
```
# Folder/
#
#    pyBarra/
#        __init__.py
#        data_handling.py
#        factor_lib.py
#        barra_model.py
#        portfolio_manager.py
#        alphalens_analysis.py
#        back_trader.py
#        utils.py
#
#    data/
#        ts_data/
#               daily.csv
#               daily_basic.csv
#               index_daily.csv
#               stock_basic.csv
#               income.csv
#               balancesheet.csv
#               cashflow.csv
#
#        factor_exposures/
#                factor_exposures.csv
#
#    Reports/
#        BARRA 模型（1）-收益归因.ipynb
#        BARRA 模型（2）-风险估计.ipynb
#        PyBarra 设计文档.ipynb
#
#    pyBarra_test.py  # 用于测试的脚本
```

### 4.3.1 数据处理 data_handling.py

数据处理模块负责管理从 tushare 上读取的文件，确保他们包含最新的数据；数据处理模块还负责因子提取，从 factor_lib.py 里提取各类因子，并保存在 factor_exposures 里，并且实时更新。

### 4.3.2 Barra 模型 barra_model.py

Barra 模型类是对于 Barra 多因子模型建模得到的模型的类，它与投资组合/个股等没有关系，只依赖于 Estimation universe。此类涵盖了跨越一段时间，每一天上的模型参数，如因子收益率，特质收益率，因子协方差矩阵，特质风险矩阵等。这些模型参数与投资组合权重等无关，而仅与时间和 Estimation universe 的构成有关。

对于一个特定的时间节点，Barra 模型类可以获得一些模型参数，包括因子收益率 f，特质收益率，因子协方差矩阵 F，特质风险矩阵 Delta。

### 4.3.3 投资组合管理 portfolio_management.py

投资组合管理模块负责管理投资组合权重，成分股，投资组合收益估计，投资组合风险估计。

### 4.3.4 优化器 optimizer.py

优化器模块主要使用 scipy 库进行投资组合风险最小化或收益最大化优化，此外，优化器模块支持添加各类限制条件，如投资组合因子暴露度，成分股数量，成分股权重范围等。

### 4.3.5 因子分析 alphalens_analysis.py

### 4.3.6 回测分析 back_trader.py

### 4.3.7 辅助工具模块 utils