02_fama_macbeth

September 29, 2021

1 How to build a linear factor model

Algorithmic trading strategies use linear factor models to quantify the relationship between the return of an asset and the sources of risk that represent the main drivers of these returns. Each factor risk carries a premium, and the total asset return can be expected to correspond to a weighted average of these risk premia.

There are several practical applications of factor models across the portfolio management process from construction and asset selection to risk management and performance evaluation. The importance of factor models continues to grow as common risk factors are now tradeable:

- A summary of the returns of many assets by a much smaller number of factors reduces the amount of data required to estimate the covariance matrix when optimizing a portfolio
- An estimate of the exposure of an asset or a portfolio to these factors allows for the management of the resultant risk, for instance by entering suitable hedges when risk factors are themselves traded
- A factor model also permits the assessment of the incremental signal content of new alpha factors
- A factor model can also help assess whether a manager's performance relative to a benchmark is indeed due to skill in selecting assets and timing the market, or if instead, the performance can be explained by portfolio tilts towards known return drivers that can today be replicated as low-cost, passively managed funds without incurring active management fees

1.1 Imports & Settings

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import pandas as pd
import numpy as np

from statsmodels.api import OLS, add_constant
import pandas_datareader.data as web

from linearmodels.asset_pricing import LinearFactorModel

import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```python
sns.set_style('whitegrid')
```

1.2 Get Data

Fama and French make updated risk factor and research portfolio data available through their website, and you can use the `pandas_datareader` package to obtain the data.

1.2.1 Risk Factors

In particular, we will be using the five Fama—French factors that result from sorting stocks first into three size groups and then into two for each of the remaining three firm-specific factors.

Hence, the factors involve three sets of value-weighted portfolios formed as 3 x 2 sorts on size and book-to-market, size and operating profitability, and size and investment. The risk factor values computed as the average returns of the portfolios (PF) as outlined in the following table:

Label	Name	Description
SMB	Small Minus Big	Average return on the nine small stock portfolios minus the average return on the nine big stock portfolios
HML	High Minus Low	Average return on the two value portfolios minus the average return on the two growth portfolios
RMW	Robust minus Weak	Average return on the two robust operating profitability portfolios minus the average return on the two weak operating profitability portfolios
CMA	Conservative Minus Aggressive	Average return on the two conservative investment portfolios minus the average return on the two aggressive investment portfolios
Rm-Rf	Excess return on the market	Value-weight return of all firms incorporated in the US and listed on the NYSE, AMEX, or NASDAQ at the beginning of month t with 'good' data for t minus the one-month Treasury bill rate

The Fama-French 5 factors are based on the 6 value-weight portfolios formed on size and book-to-market, the 6 value-weight portfolios formed on size and operating profitability, and the 6 value-weight portfolios formed on size and investment.

We will use returns at a monthly frequency that we obtain for the period 2010 – 2017 as follows:

[4]:
```python
ff_factor = 'F-F_Research_Data_5_Factors_2x3'
ff_factor_data = web.DataReader(ff_factor, 'famafrench', start='2010',␣
↪end='2017-12')[0]
ff_factor_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 96 entries, 2010-01 to 2017-12
Freq: M
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Mkt-RF  96 non-null     float64
```

```
 1   SMB     96 non-null     float64
 2   HML     96 non-null     float64
 3   RMW     96 non-null     float64
 4   CMA     96 non-null     float64
 5   RF      96 non-null     float64
dtypes: float64(6)
memory usage: 5.2 KB
```

[5]:
```python
ff_factor_data.describe()
```

[5]:

	Mkt-RF	SMB	HML	RMW	CMA	RF
count	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000
mean	1.158750	0.054063	-0.051771	0.126042	0.052813	0.012604
std	3.580187	2.290739	2.191621	1.591052	1.409858	0.022583
min	-7.890000	-4.510000	-4.520000	-3.930000	-3.350000	0.000000
25%	-0.917500	-1.660000	-1.627500	-1.160000	-0.965000	0.000000
50%	1.235000	0.190000	-0.305000	0.135000	-0.015000	0.000000
75%	3.197500	1.517500	1.142500	1.140000	0.927500	0.010000
max	11.350000	6.800000	8.220000	3.530000	3.780000	0.090000

1.2.2 Portfolios

Fama and French also make available numerous portfolios that we can illustrate the estimation of the factor exposures, as well as the value of the risk premia available in the market for a given time period. We will use a panel of the 17 industry portfolios at a monthly frequency.

We will subtract the risk-free rate from the returns because the factor model works with excess returns:

[6]:
```python
ff_portfolio = '17_Industry_Portfolios'
ff_portfolio_data = web.DataReader(ff_portfolio, 'famafrench', start='2010',␣
↪end='2017-12')[0]
ff_portfolio_data = ff_portfolio_data.sub(ff_factor_data.RF, axis=0)
ff_portfolio_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 96 entries, 2010-01 to 2017-12
Freq: M
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Food    96 non-null     float64
 1   Mines   96 non-null     float64
 2   Oil     96 non-null     float64
 3   Clths   96 non-null     float64
 4   Durbl   96 non-null     float64
 5   Chems   96 non-null     float64
 6   Cnsum   96 non-null     float64
 7   Cnstr   96 non-null     float64
```

```
 8   Steel   96 non-null     float64
 9   FabPr   96 non-null     float64
 10  Machn   96 non-null     float64
 11  Cars    96 non-null     float64
 12  Trans   96 non-null     float64
 13  Utils   96 non-null     float64
 14  Rtail   96 non-null     float64
 15  Finan   96 non-null     float64
 16  Other   96 non-null     float64
dtypes: float64(17)
memory usage: 13.5 KB
```

[7]:

```
ff_portfolio_data.describe()
```

[7]:

	Food	Mines	Oil	Clths	Durbl	Chems
count	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000
mean	1.045625	0.197083	0.547917	1.396979	1.155208	1.303229
std	2.795857	7.902185	5.577552	5.025167	5.137482	5.594216
min	-5.170000	-24.380000	-12.010000	-10.000000	-13.210000	-17.390000
25%	-0.785000	-5.847500	-3.167500	-1.865000	-2.017500	-1.445000
50%	0.930000	-0.460000	1.040000	1.160000	1.205000	1.435000
75%	3.187500	5.715000	3.915000	3.857500	4.322500	4.442500
max	6.670000	21.920000	16.300000	17.200000	16.580000	18.370000

	Cnsum	Cnstr	Steel	FabPr	Machn	Cars
count	96.000000	96.000000	96.000000	96.000000	96.000000	96.000000
mean	1.136250	1.731354	0.555625	1.350521	1.227604	1.278854
std	3.174283	5.246518	7.389824	4.694408	4.811242	5.718887
min	-7.300000	-13.960000	-20.490000	-11.960000	-9.080000	-11.650000
25%	-0.920000	-2.462500	-4.410000	-1.447500	-2.047500	-1.245000
50%	1.470000	2.190000	0.660000	1.485000	1.545000	0.645000
75%	3.317500	5.390000	4.220000	3.837500	4.657500	4.802500
max	8.290000	15.550000	21.350000	17.660000	14.650000	20.860000

	Trans	Utils	Rtail	Finan	Other
count	96.000000	96.000000	96.000000	96.000000	96.000000
mean	1.465000	0.890313	1.234375	1.241562	1.282396
std	4.150833	3.235140	3.508655	4.809791	3.708972
min	-8.560000	-6.990000	-9.180000	-11.040000	-7.920000
25%	-0.880000	-0.745000	-0.962500	-1.467500	-1.075000
50%	1.505000	1.215000	0.880000	1.955000	1.575000
75%	4.235000	2.952500	3.355000	4.092500	3.517500
max	13.160000	7.900000	12.360000	13.480000	10.790000

1.2.3 Equity Data

[8]:
```python
with pd.HDFStore('../data/assets.h5') as store:
    prices = store['/quandl/wiki/prices'].adj_close.unstack().loc['2010':'2017']
    equities = store['/us_equities/stocks'].drop_duplicates()
```

[9]:
```python
sectors = equities.filter(prices.columns, axis=0).sector.to_dict()
prices = prices.filter(sectors.keys()).dropna(how='all', axis=1)
```

[10]:
```python
returns = prices.resample('M').last().pct_change().mul(100).to_period('M')
returns = returns.dropna(how='all').dropna(axis=1)
returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 95 entries, 2010-02 to 2017-12
Freq: M
Columns: 1986 entries, A to ZUMZ
dtypes: float64(1986)
memory usage: 1.4 MB
```

1.2.4 Align data

[11]:
```python
ff_factor_data = ff_factor_data.loc[returns.index]
ff_portfolio_data = ff_portfolio_data.loc[returns.index]
```

[12]:
```python
ff_factor_data.describe()
```

[12]:

	Mkt-RF	SMB	HML	RMW	CMA	RF
count	95.000000	95.000000	95.000000	95.000000	95.000000	95.000000
mean	1.206316	0.051053	-0.055789	0.139789	0.048842	0.012737
std	3.568555	2.302701	2.202892	1.593750	1.416798	0.022665
min	-7.890000	-4.510000	-4.520000	-3.930000	-3.350000	0.000000
25%	-0.565000	-1.670000	-1.655000	-0.965000	-0.990000	0.000000
50%	1.290000	0.150000	-0.360000	0.140000	-0.020000	0.000000
75%	3.265000	1.555000	1.165000	1.140000	0.935000	0.010000
max	11.350000	6.800000	8.220000	3.530000	3.780000	0.090000

1.2.5 Compute excess Returns

[13]:
```python
excess_returns = returns.sub(ff_factor_data.RF, axis=0)
excess_returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 95 entries, 2010-02 to 2017-12
Freq: M
Columns: 1986 entries, A to ZUMZ
dtypes: float64(1986)
memory usage: 1.4 MB
```

[14]:
```
excess_returns = excess_returns.clip(lower=np.percentile(excess_returns, 1),
                                     upper=np.percentile(excess_returns, 99))
```

1.3 Fama-Macbeth Regression

Given data on risk factors and portfolio returns, it is useful to estimate the portfolio's exposure, that is, how much the risk factors drive portfolio returns, as well as how much the exposure to a given factor is worth, that is, the what market's risk factor premium is. The risk premium then permits to estimate the return for any portfolio provided the factor exposure is known or can be assumed.

[15]:
```
ff_portfolio_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 95 entries, 2010-02 to 2017-12
Freq: M
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Food    95 non-null     float64
 1   Mines   95 non-null     float64
 2   Oil     95 non-null     float64
 3   Clths   95 non-null     float64
 4   Durbl   95 non-null     float64
 5   Chems   95 non-null     float64
 6   Cnsum   95 non-null     float64
 7   Cnstr   95 non-null     float64
 8   Steel   95 non-null     float64
 9   FabPr   95 non-null     float64
 10  Machn   95 non-null     float64
 11  Cars    95 non-null     float64
 12  Trans   95 non-null     float64
 13  Utils   95 non-null     float64
 14  Rtail   95 non-null     float64
 15  Finan   95 non-null     float64
 16  Other   95 non-null     float64
dtypes: float64(17)
memory usage: 13.4 KB
```

[16]:
```
ff_factor_data = ff_factor_data.drop('RF', axis=1)
ff_factor_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 95 entries, 2010-02 to 2017-12
Freq: M
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   Mkt-RF  95 non-null     float64
 1   SMB     95 non-null     float64
 2   HML     95 non-null     float64
 3   RMW     95 non-null     float64
 4   CMA     95 non-null     float64
dtypes: float64(5)
memory usage: 4.5 KB
```

To address the inference problem caused by the correlation of the residuals, Fama and MacBeth proposed a two-step methodology for a cross-sectional regression of returns on factors. The two-stage Fama—Macbeth regression is designed to estimate the premium rewarded for the exposure to a particular risk factor by the market. The two stages consist of:

- First stage: N time-series regression, one for each asset or portfolio, of its excess returns on the factors to estimate the factor loadings.
- Second stage: T cross-sectional regression, one for each time period, to estimate the risk premium.

See corresponding section in Chapter 7 of Machine Learning for Trading for details.

Now we can compute the factor risk premia as the time average and get t-statistic to assess their individual significance, using the assumption that the risk premia estimates are independent over time.

If we had a very large and representative data sample on traded risk factors we could use the sample mean as a risk premium estimate. However, we typically do not have a sufficiently long history to and the margin of error around the sample mean could be quite large.

The Fama—Macbeth methodology leverages the covariance of the factors with other assets to determine the factor premia. The second moment of asset returns is easier to estimate than the first moment, and obtaining more granular data improves estimation considerably, which is not true of mean estimation.

1.3.1 Step 1: Factor Exposures

We can implement the first stage to obtain the 17 factor loading estimates as follows:

[17]:
```
betas = []
for industry in ff_portfolio_data:
    step1 = OLS(endog=ff_portfolio_data.loc[ff_factor_data.index, industry],
                exog=add_constant(ff_factor_data)).fit()
    betas.append(step1.params.drop('const'))
```

[18]:
```
betas = pd.DataFrame(betas,
                     columns=ff_factor_data.columns,
                     index=ff_portfolio_data.columns)
betas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 17 entries, Food  to Other
Data columns (total 5 columns):
```

```
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Mkt-RF  17 non-null     float64
 1   SMB     17 non-null     float64
 2   HML     17 non-null     float64
 3   RMW     17 non-null     float64
 4   CMA     17 non-null     float64
dtypes: float64(5)
memory usage: 1.3+ KB
```

1.3.2 Step 2: Risk Premia

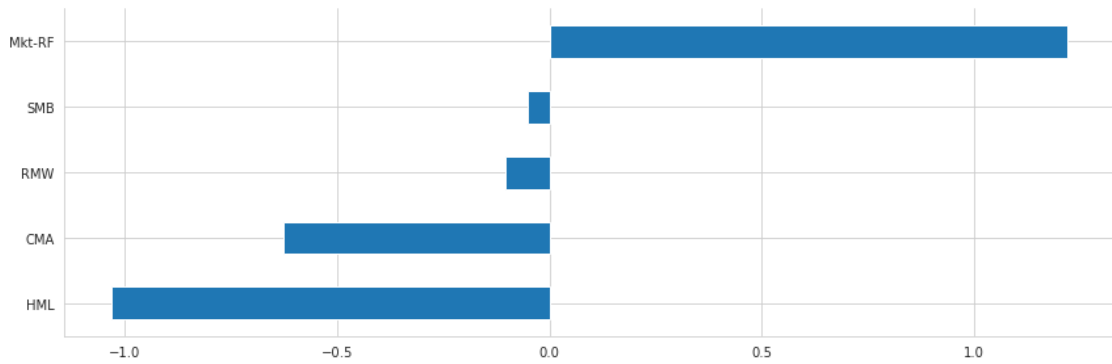
For the second stage, we run 96 regressions of the period returns for the cross section of portfolios on the factor loadings

[19]:
```
lambdas = []
for period in ff_portfolio_data.index:
    step2 = OLS(endog=ff_portfolio_data.loc[period, betas.index],
                exog=betas).fit()
    lambdas.append(step2.params)
```

[20]:
```
lambdas = pd.DataFrame(lambdas,
                       index=ff_portfolio_data.index,
                       columns=betas.columns.tolist())
lambdas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 95 entries, 2010-02 to 2017-12
Freq: M
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Mkt-RF  95 non-null     float64
 1   SMB     95 non-null     float64
 2   HML     95 non-null     float64
 3   RMW     95 non-null     float64
 4   CMA     95 non-null     float64
dtypes: float64(5)
memory usage: 9.3 KB
```

[21]:
```
lambdas.mean().sort_values().plot.barh(figsize=(12, 4))
sns.despine()
plt.tight_layout();
```

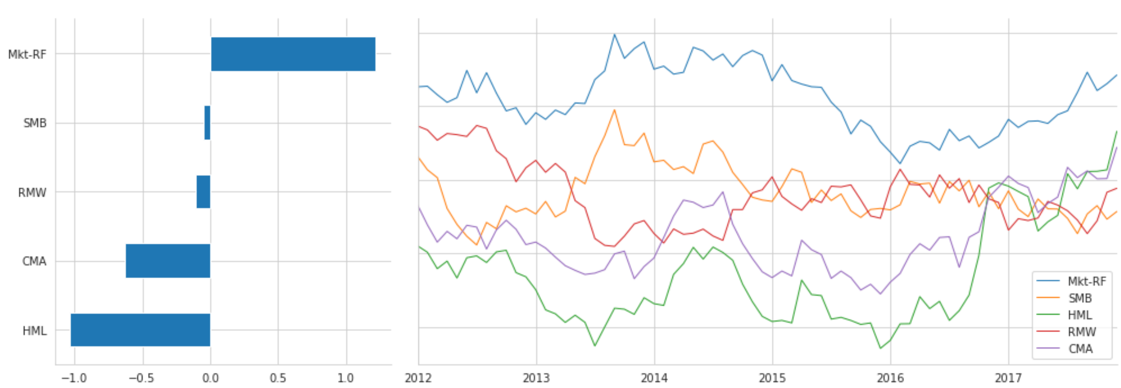


[22]:
```
t = lambdas.mean().div(lambdas.std())
t
```

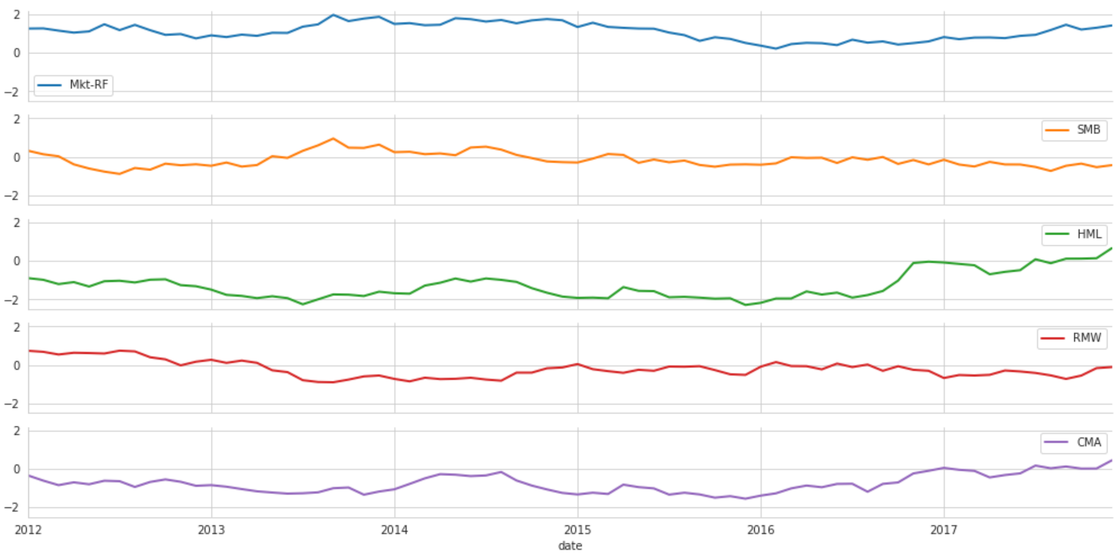
[22]:
```
Mkt-RF    0.340179
SMB      -0.013216
HML      -0.261636
RMW      -0.035186
CMA      -0.172461
dtype: float64
```

Results

[23]:
```
window = 24  # months
ax1 = plt.subplot2grid((1, 3), (0, 0))
ax2 = plt.subplot2grid((1, 3), (0, 1), colspan=2)
lambdas.mean().sort_values().plot.barh(ax=ax1)
lambdas.rolling(window).mean().dropna().plot(lw=1,
                                             figsize=(14, 5),
                                             sharey=True,
                                             ax=ax2)
sns.despine()
plt.tight_layout()
```



[24]:
```
window = 24  # months
lambdas.rolling(window).mean().dropna().plot(lw=2,
                                               figsize=(14, 7),
                                               subplots=True,
                                               sharey=True)
sns.despine()
plt.tight_layout()
```



1.4 Fama-Macbeth with the LinearModels library

The linear_models library extends statsmodels with various models for panel data and also implements the two-stage Fama—MacBeth procedure:

[25]:
```
mod = LinearFactorModel(portfolios=ff_portfolio_data,
                        factors=ff_factor_data)
res = mod.fit()
print(res)
```

```
                      LinearFactorModel Estimation Summary
================================================================================
No. Test Portfolios:                 17   R-squared:                      0.6885
No. Factors:                          5   J-statistic:                    17.038
No. Observations:                    95   P-value                         0.1482
Date:                  Thu, Apr 15 2021   Distribution:                 chi2(12)
Time:                          14:55:21
Cov. Estimator:                  robust
```

```
                             Risk Premia Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
Mkt-RF         1.2208     0.4076     2.9951     0.0027      0.4219      2.0197
SMB           -0.0523     0.7979    -0.0656     0.9477     -1.6161      1.5115
HML           -1.0316     0.6332    -1.6292     0.1033     -2.2726      0.2094
RMW           -0.1044     0.7738    -0.1350     0.8926     -1.6210      1.4121
CMA           -0.6252     0.5222    -1.1973     0.2312     -1.6488      0.3983
==============================================================================

Covariance estimator:
HeteroskedasticCovariance
See full_summary for complete results
```

[26]:

```python
print(res.full_summary)
```

```
                      LinearFactorModel Estimation Summary
================================================================================
No. Test Portfolios:                 17   R-squared:                      0.6885
No. Factors:                          5   J-statistic:                    17.038
No. Observations:                    95   P-value                         0.1482
Date:                  Thu, Apr 15 2021   Distribution:                 chi2(12)
Time:                          14:55:21
Cov. Estimator:                  robust

                             Risk Premia Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
Mkt-RF         1.2208     0.4076     2.9951     0.0027      0.4219      2.0197
SMB           -0.0523     0.7979    -0.0656     0.9477     -1.6161      1.5115
HML           -1.0316     0.6332    -1.6292     0.1033     -2.2726      0.2094
RMW           -0.1044     0.7738    -0.1350     0.8926     -1.6210      1.4121
CMA           -0.6252     0.5222    -1.1973     0.2312     -1.6488      0.3983


                               Food  Coefficients
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
alpha          0.1874     0.2393     0.7831     0.4336     -0.2816      0.6563
Mkt-RF         0.6866     0.0465     14.773     0.0000      0.5955      0.7777
SMB           -0.3100     0.1126    -2.7532     1.9941     -0.5307     -0.0893
HML           -0.3493     0.1420    -2.4595     1.9861     -0.6277     -0.0710
RMW            0.3075     0.1243     2.4747     0.0133      0.0640      0.5510
CMA            0.4666     0.1636     2.8517     0.0043      0.1459      0.7873
```

Mines Coefficients

alpha	-0.6490	0.5444	-1.1922	1.7668	-1.7159	0.4180
Mkt-RF	1.2987	0.1950	6.6584	0.0000	0.9164	1.6810
SMB	0.1805	0.3380	0.5341	0.5933	-0.4819	0.8429
HML	0.1891	0.3305	0.5721	0.5673	-0.4587	0.8368
RMW	0.1449	0.4295	0.3375	0.7358	-0.6968	0.9867
CMA	0.6112	0.5507	1.1098	0.2671	-0.4682	1.6905

Oil Coefficients

alpha	0.2044	0.3452	0.5922	0.5537	-0.4722	0.8811
Mkt-RF	1.0553	0.1027	10.273	0.0000	0.8540	1.2567
SMB	0.1554	0.1941	0.8005	0.4234	-0.2251	0.5359
HML	0.6685	0.2048	3.2642	0.0011	0.2671	1.0698
RMW	-0.0247	0.2326	-0.1064	1.0847	-0.4806	0.4311
CMA	0.3117	0.2831	1.1010	0.2709	-0.2432	0.8666

Clths Coefficients

alpha	0.1726	0.3388	0.5093	0.6105	-0.4915	0.8367
Mkt-RF	0.9685	0.1214	7.9776	0.0000	0.7306	1.2065
SMB	0.3430	0.1966	1.7450	0.0810	-0.0422	0.7282
HML	-0.1882	0.2098	-0.8969	1.6302	-0.5993	0.2230
RMW	0.5649	0.2720	2.0767	0.0378	0.0318	1.0980
CMA	0.0381	0.3163	0.1205	0.9040	-0.5818	0.6581

Durbl Coefficients

alpha	-0.1564	0.3204	-0.4883	1.3747	-0.7843	0.4715
Mkt-RF	1.1740	0.0834	14.072	0.0000	1.0105	1.3375
SMB	0.5378	0.1194	4.5035	0.0000	0.3037	0.7719
HML	0.0706	0.1480	0.4771	0.6333	-0.2195	0.3607
RMW	0.5117	0.1940	2.6380	0.0083	0.1315	0.8919
CMA	-0.1310	0.2655	-0.4936	1.3784	-0.6514	0.3893

Chems Coefficients

alpha	-0.2048	0.3111	-0.6584	1.4897	-0.8145	0.4049
Mkt-RF	1.3510	0.1063	12.704	0.0000	1.1426	1.5594
SMB	0.1660	0.1489	1.1154	0.2647	-0.1257	0.4578
HML	0.1952	0.1480	1.3189	0.1872	-0.0949	0.4852
RMW	0.1410	0.1912	0.7374	0.4609	-0.2338	0.5158

CMA	-0.2301	0.2633	-0.8738	1.6178	-0.7462	0.2860

Cnsum Coefficients

alpha	-0.0380	0.3566	-0.1065	1.0848	-0.7368	0.6609
Mkt-RF	0.7625	0.0591	12.897	0.0000	0.6466	0.8784
SMB	-0.3327	0.1006	-3.3088	1.9991	-0.5298	-0.1356
HML	-0.5773	0.1259	-4.5845	2.0000	-0.8241	-0.3305
RMW	-0.0606	0.1316	-0.4603	1.3547	-0.3186	0.1974
CMA	0.5748	0.2271	2.5306	0.0114	0.1296	1.0199

Cnstr Coefficients

alpha	0.6213	0.3917	1.5862	0.1127	-0.1464	1.3890
Mkt-RF	1.1161	0.0828	13.478	0.0000	0.9538	1.2784
SMB	0.4463	0.1337	3.3389	0.0008	0.1843	0.7083
HML	0.0920	0.1892	0.4861	0.6269	-0.2789	0.4629
RMW	-0.0107	0.2232	-0.0482	1.0384	-0.4482	0.4267
CMA	0.1409	0.2425	0.5811	0.5612	-0.3344	0.6162

Steel Coefficients

alpha	-0.3503	0.4030	-0.8692	1.6153	-1.1403	0.4396
Mkt-RF	1.4647	0.1381	10.604	0.0000	1.1940	1.7355
SMB	0.4104	0.2548	1.6103	0.1073	-0.0891	0.9098
HML	0.4000	0.2653	1.5076	0.1317	-0.1200	0.9200
RMW	0.1355	0.3342	0.4054	0.6852	-0.5196	0.7906
CMA	0.4840	0.4192	1.1547	0.2482	-0.3376	1.3056

FabPr Coefficients

alpha	0.2168	0.2831	0.7659	0.4437	-0.3381	0.7718
Mkt-RF	1.0695	0.0734	14.573	0.0000	0.9257	1.2133
SMB	0.4602	0.0979	4.7024	0.0000	0.2684	0.6520
HML	-0.0294	0.1111	-0.2646	1.2087	-0.2471	0.1883
RMW	0.1531	0.1456	1.0510	0.2933	-0.1324	0.4385
CMA	0.1865	0.1855	1.0055	0.3147	-0.1771	0.5502

Machn Coefficients

alpha	-0.3139	0.2688	-1.1677	1.7571	-0.8409	0.2130
Mkt-RF	1.1883	0.0582	20.424	0.0000	1.0742	1.3023
SMB	0.1817	0.1074	1.6922	0.0906	-0.0287	0.3921

HML	0.0384	0.1060	0.3621	0.7173	-0.1694	0.2462
RMW	0.0540	0.1581	0.3416	0.7327	-0.2559	0.3639
CMA	-0.3765	0.1786	-2.1079	1.9650	-0.7266	-0.0264

Cars Coefficients

alpha	-0.0952	0.3906	-0.2436	1.1925	-0.8607	0.6704
Mkt-RF	1.1895	0.0996	11.949	0.0000	0.9944	1.3846
SMB	0.5941	0.1290	4.6069	0.0000	0.3414	0.8469
HML	0.0213	0.1757	0.1214	0.9034	-0.3231	0.3658
RMW	0.0223	0.2201	0.1012	0.9194	-0.4091	0.4537
CMA	0.0123	0.2993	0.0412	0.9671	-0.5743	0.5990

Trans Coefficients

alpha	0.4814	0.3269	1.4725	0.1409	-0.1594	1.1221
Mkt-RF	1.0248	0.0508	20.161	0.0000	0.9251	1.1244
SMB	0.2537	0.1030	2.4641	0.0137	0.0519	0.4556
HML	0.0117	0.1222	0.0961	0.9235	-0.2278	0.2512
RMW	0.3778	0.1606	2.3525	0.0186	0.0630	0.6926
CMA	0.2634	0.2018	1.3056	0.1917	-0.1320	0.6589

Utils Coefficients

alpha	0.3695	0.3118	1.1849	0.2361	-0.2417	0.9807
Mkt-RF	0.5022	0.0911	5.5126	0.0000	0.3237	0.6808
SMB	-0.2454	0.1567	-1.5664	1.8827	-0.5525	0.0617
HML	-0.2932	0.1772	-1.6549	1.9021	-0.6405	0.0540
RMW	0.2424	0.1949	1.2435	0.2137	-0.1396	0.6244
CMA	0.5207	0.2955	1.7621	0.0781	-0.0585	1.0998

Rtail Coefficients

alpha	-0.0421	0.2681	-0.1570	1.1247	-0.5676	0.4834
Mkt-RF	0.9087	0.0689	13.192	0.0000	0.7737	1.0437
SMB	0.1316	0.0994	1.3235	0.1857	-0.0633	0.3265
HML	-0.3830	0.1296	-2.9560	1.9969	-0.6370	-0.1291
RMW	0.6884	0.1610	4.2748	0.0000	0.3728	1.0041
CMA	0.1961	0.1741	1.1259	0.2602	-0.1452	0.5373

Finan Coefficients

alpha	0.3752	0.3715	1.0097	0.3126	-0.3530	1.1034

```
Mkt-RF           1.0565      0.0426      24.782      0.0000       0.9730       1.1401
SMB              0.0756      0.0856      0.8825      0.3775      -0.0923       0.2434
HML              0.7333      0.0878      8.3550      0.0000       0.5613       0.9054
RMW             -0.4296      0.1061     -4.0490      1.9999      -0.6376      -0.2216
CMA             -0.5083      0.1124     -4.5216      2.0000      -0.7286      -0.2879


                              Other Coefficients
==============================================================================
alpha           -0.1368      0.2280     -0.5998      1.4513      -0.5837       0.3102
Mkt-RF           1.0416      0.0244      42.676      0.0000       0.9937       1.0894
SMB             -0.1150      0.0397     -2.8966      1.9962      -0.1927      -0.0372
HML             -0.2042      0.0379     -5.3820      2.0000      -0.2786      -0.1299
RMW             -0.0685      0.0626     -1.0934      1.7258      -0.1912       0.0543
CMA              0.0194      0.0657      0.2945      0.7684      -0.1095       0.1482
==============================================================================

Covariance estimator:
HeteroskedasticCovariance
See full_summary for complete results
```

This provides us with the same result:

[27]:
```
lambdas.mean()
```

[27]:
```
Mkt-RF    1.220797
SMB      -0.052301
HML      -1.031603
RMW      -0.104427
CMA      -0.625245
dtype: float64
```