q2

May 17, 2023

[ ]:
```python
import pandas as pd
import pandas_datareader as pdr
import numpy as np
from statsmodels.api import OLS
from statsmodels.tools import add_constant
```

[ ]:
```python
ff5_month = pdr.get_data_famafrench('F-F_Research_Data_5_Factors_2x3',␣
  ↪start='1963-07', end='2023-03')[0]
ff5_month = ff5_month
mom_month = pdr.get_data_famafrench('F-F_Momentum_Factor', start='1927-01',␣
  ↪end='2023-03')[0]
mom_month = mom_month

ff5_month = ff5_month.reset_index()
mom_month = mom_month.reset_index()
ff5mom_month = ff5_month.merge(mom_month, on='Date', how='left')
ff5mom_month = ff5mom_month.set_index('Date')
```

```
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/1568211615.py:1
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  ff5_month = pdr.get_data_famafrench('F-F_Research_Data_5_Factors_2x3',
start='1963-07', end='2023-03')[0]
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/1568211615.py:1
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  ff5_month = pdr.get_data_famafrench('F-F_Research_Data_5_Factors_2x3',
start='1963-07', end='2023-03')[0]
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/1568211615.py:3
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  mom_month = pdr.get_data_famafrench('F-F_Momentum_Factor', start='1927-01',
end='2023-03')[0]
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/1568211615.py:3
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
```

```
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  mom_month = pdr.get_data_famafrench('F-F_Momentum_Factor', start='1927-01',
end='2023-03')[0]
```

[ ]:
```
ff5_daily = pdr.get_data_famafrench('F-F_Research_Data_5_Factors_2x3_daily',␣
  ↪start='1963-07-01', end='2023-03-31')[0]
ff5_daily = ff5_daily
mom_daily = pdr.get_data_famafrench('F-F_Momentum_Factor_daily',␣
  ↪start='1926-11-03', end='2023-03-31')[0]
mom_daily = mom_daily

ff5_daily = ff5_daily.reset_index()
mom_daily = mom_daily.reset_index()
ff5mom_daily = ff5_daily.merge(mom_daily, on='Date', how='left')
ff5mom_daily = ff5mom_daily.set_index('Date')
```

```
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/3591460299.py:1
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  ff5_daily = pdr.get_data_famafrench('F-F_Research_Data_5_Factors_2x3_daily',
start='1963-07-01', end='2023-03-31')[0]
/var/folders/sg/4dp480wd1cjd288xvby34rpr0000gn/T/ipykernel_30017/3591460299.py:3
: FutureWarning: The argument 'date_parser' is deprecated and will be removed in
a future version. Please use 'date_format' instead, or read your data in as
'object' dtype and then call 'to_datetime'.
  mom_daily = pdr.get_data_famafrench('F-F_Momentum_Factor_daily',
start='1926-11-03', end='2023-03-31')[0]
```

[ ]:
```
ff5mom_month.dropna(inplace=True)
```

[ ]:
```
ff5mom_month
```

[ ]:

Date	Mkt-RF	SMB	HML	RMW	CMA	RF	Mom
1963-07	-0.39	-0.41	-0.97	0.68	-1.18	0.27	0.90
1963-08	5.07	-0.80	1.80	0.36	-0.35	0.25	1.01
1963-09	-1.57	-0.52	0.13	-0.71	0.29	0.27	0.19
1963-10	2.53	-1.39	-0.10	2.80	-2.01	0.29	3.12
1963-11	-0.85	-0.88	1.75	-0.51	2.24	0.27	-0.74
…	…	…	…	…	…	…	…
2022-11	4.60	-2.67	1.38	6.01	3.11	0.29	-2.01
2022-12	-6.41	-0.16	1.32	0.09	4.19	0.33	4.52
2023-01	6.65	4.43	-4.05	-2.62	-4.53	0.35	-15.98
2023-02	-2.58	0.69	-0.78	0.90	-1.41	0.34	0.20
2023-03	2.51	-7.01	-9.01	1.92	-2.29	0.36	-2.52

[717 rows x 7 columns]

[ ]:
```
# calculate rolling variance
rolling_variance_daily = ff5mom_daily.rolling(22, min_periods=22).var()
rolling_variance = rolling_variance_daily.resample('M').last().
 ↪drop(columns=['RF'])
```

[ ]:
```
rolling_variance = rolling_variance ** -1
```

[ ]:
```
rolling_variance = rolling_variance.reset_index()
```

[ ]:
```
rolling_variance
```

[ ]:
```
           Date     Mkt-RF        SMB        HML        RMW        CMA  \
0    1963-07-31   4.469762  47.760330  26.396983  45.127324  31.593416
1    1963-08-31  11.500977  29.715197  25.290679  75.218573  43.235757
2    1963-09-30   6.484129  19.019860  28.526971  34.794397  35.608309
3    1963-10-31   5.835389  12.909608   8.586850  16.207967  11.664752
4    1963-11-30   0.669356   4.814682  12.504567  10.765115  12.213660
..          …         …         …         …         …         …
712 2022-11-30   0.314242   4.052429   0.814639   1.621913   1.123394
713 2022-12-31   0.545313   5.844314   1.208067   2.663494   2.285861
714 2023-01-31   0.855394   4.766012   2.212627   3.374825   2.648352
715 2023-02-28   0.895622   2.863532   1.117726   3.761771   1.721952
716 2023-03-31   0.683783   2.484213   0.885675   3.738393   2.976488

           Mom
0    12.978512
1    20.963409
2    13.757862
3    10.592710
4     1.562705
..         …
712   0.196801
713   0.578830
714   0.673593
715   0.368082
716   1.439367

[717 rows x 7 columns]
```

[ ]:
```
row_sums = rolling_variance.iloc[:, 1:].sum(axis=1)

# Divide each entry by the sum of other entries in the row
weighted_df = rolling_variance.iloc[:, 1:].div(row_sums, axis=0)
```

```
# Concatenate the 'Date' column with the weighted DataFrame
weights = pd.concat([rolling_variance['Date'], weighted_df], axis=1)
weights.index = weights.Date
weights = weights.drop(columns='Date')
weights = weights.shift(1)
```

[ ]:
```
weights = weights.reset_index()
```

[ ]:
```
weights['Date'] = weights['Date'].dt.strftime('%Y-%m')
```

[ ]:
```
weights = weights.dropna()
```

[ ]:
```
weights.index = weights.Date
weights = weights.drop(columns=['Date'])
```

[ ]:
```
weights
```

[ ]:
```
           Mkt-RF       SMB       HML       RMW       CMA      Mom
Date
1963-08  0.026554  0.283737  0.156820  0.268094  0.187691  0.077103
1963-09  0.055850  0.144301  0.122815  0.365272  0.209959  0.101801
1963-10  0.046921  0.137634  0.206431  0.251784  0.257674  0.099556
1963-11  0.088687  0.196203  0.130505  0.246332  0.177283  0.160990
1963-12  0.015738  0.113206  0.294017  0.253118  0.287177  0.036744
…             …         …         …         …         …         …
2022-11  0.031721  0.378612  0.147422  0.167967  0.210137  0.064141
2022-12  0.038683  0.498858  0.100283  0.199659  0.138291  0.024226
2023-01  0.041545  0.445251  0.092037  0.202919  0.174149  0.044098
2023-02  0.058868  0.327994  0.152271  0.232253  0.182258  0.046356
2023-03  0.083479  0.266904  0.104181  0.350627  0.160500  0.034308

[716 rows x 6 columns]
```

[ ]:
```
factor_returns = ff5mom_month[1:]
```

[ ]:
```
weights.index = factor_returns.index
```

[ ]:
```
excess_returns = (weights * ff5mom_month[['Mkt-RF','SMB','HML','RMW','CMA','Mom␣
↪ ']]).sum(axis=1)
```

[ ]:
```
def estimate_models(excess_returns, ff5):
    # CAPM Model
    X = add_constant(ff5[['Mkt-RF']].loc[excess_returns.index])
    capm_model = OLS(excess_returns, X).fit()

    # FF3 Model
    X = add_constant(ff5[['Mkt-RF', 'SMB', 'HML']].loc[excess_returns.index])
```

```
    ff3_model = OLS(excess_returns, X).fit()

    # Carhart Model
    X = add_constant(ff5[['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA']].
 ↪loc[excess_returns.index])
    carhart_model = OLS(excess_returns, X).fit()

    print(ff5.columns)
    # FF5 Model
    X = add_constant(ff5[['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'Mom   ']].
 ↪loc[excess_returns.index,:])
    ff5_model = OLS(excess_returns, X).fit()

    return capm_model, ff3_model, carhart_model, ff5_model

factor_returns = ff5mom_month[['Mkt-RF','SMB','HML','RMW','CMA','Mom   ']]
capm_model, ff3_model, carhart_model, ff5_model =␣
 ↪estimate_models(excess_returns, factor_returns)
```

```
Index(['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'Mom   '], dtype='object')
```

[ ]:

```
print(capm_model.summary())
print(ff3_model.summary())
print(carhart_model.summary())
print(ff5_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     3.909
Date:                Tue, 16 May 2023   Prob (F-statistic):             0.0484
Time:                        20:49:56   Log-Likelihood:                -1038.9
No. Observations:                 717   AIC:                             2082.
Df Residuals:                     715   BIC:                             2091.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3390      0.039      8.730      0.000       0.263       0.415
Mkt-RF        -0.0170      0.009     -1.977      0.048      -0.034      -0.000
==============================================================================
Omnibus:                       45.404   Durbin-Watson:                   1.758
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              150.744
Skew:                           0.194   Prob(JB):                     1.85e-33
Kurtosis:                       5.213   Cond. No.                         4.57
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.381
Model:                            OLS   Adj. R-squared:                  0.378
Method:                 Least Squares   F-statistic:                     146.0
Date:                Tue, 16 May 2023   Prob (F-statistic):           9.25e-74
Time:                        20:49:56   Log-Likelihood:                -869.18
No. Observations:                 717   AIC:                             1746.
Df Residuals:                     713   BIC:                             1765.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2563      0.031      8.280      0.000       0.195       0.317
Mkt-RF        -0.0008      0.007     -0.114      0.909      -0.015       0.013
SMB            0.0665      0.010      6.345      0.000       0.046       0.087
HML            0.2034      0.010     19.493      0.000       0.183       0.224
==============================================================================
Omnibus:                       50.245   Durbin-Watson:                   1.826
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              207.771
Skew:                           0.105   Prob(JB):                     7.64e-46
Kurtosis:                       5.629   Cond. No.                         4.80
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.627
Model:                            OLS   Adj. R-squared:                  0.624
Method:                 Least Squares   F-statistic:                     238.7
Date:                Tue, 16 May 2023   Prob (F-statistic):          1.90e-149
Time:                        20:49:56   Log-Likelihood:                -687.65
No. Observations:                 717   AIC:                             1387.
Df Residuals:                     711   BIC:                             1415.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1278      0.025      5.158      0.000       0.079       0.176
```

```
Mkt-RF         0.0338      0.006      5.696      0.000       0.022       0.045
SMB            0.1211      0.009     14.013      0.000       0.104       0.138
HML            0.0810      0.011      7.312      0.000       0.059       0.103
RMW            0.2110      0.012     18.069      0.000       0.188       0.234
CMA            0.2556      0.017     15.129      0.000       0.222       0.289
==============================================================================
Omnibus:                       90.299   Durbin-Watson:                   1.754
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              416.154
Skew:                          -0.468   Prob(JB):                     4.30e-91
Kurtosis:                       6.613   Cond. No.                         5.19
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.705
Model:                            OLS   Adj. R-squared:                  0.703
Method:                 Least Squares   F-statistic:                     282.9
Date:                Tue, 16 May 2023   Prob (F-statistic):         1.79e-184
Time:                        20:49:56   Log-Likelihood:                -603.15
No. Observations:                 717   AIC:                             1220.
Df Residuals:                     710   BIC:                             1252.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0787      0.022      3.524      0.000       0.035       0.123
Mkt-RF         0.0453      0.005      8.479      0.000       0.035       0.056
SMB            0.1184      0.008     15.396      0.000       0.103       0.133
HML            0.1172      0.010     11.493      0.000       0.097       0.137
RMW            0.1974      0.010     18.921      0.000       0.177       0.218
CMA            0.2292      0.015     15.132      0.000       0.199       0.259
Mom            0.0723      0.005     13.737      0.000       0.062       0.083
==============================================================================
Omnibus:                       84.243   Durbin-Watson:                   1.731
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              458.715
Skew:                          -0.353   Prob(JB):                    2.46e-100
Kurtosis:                       6.854   Cond. No.                         5.35
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

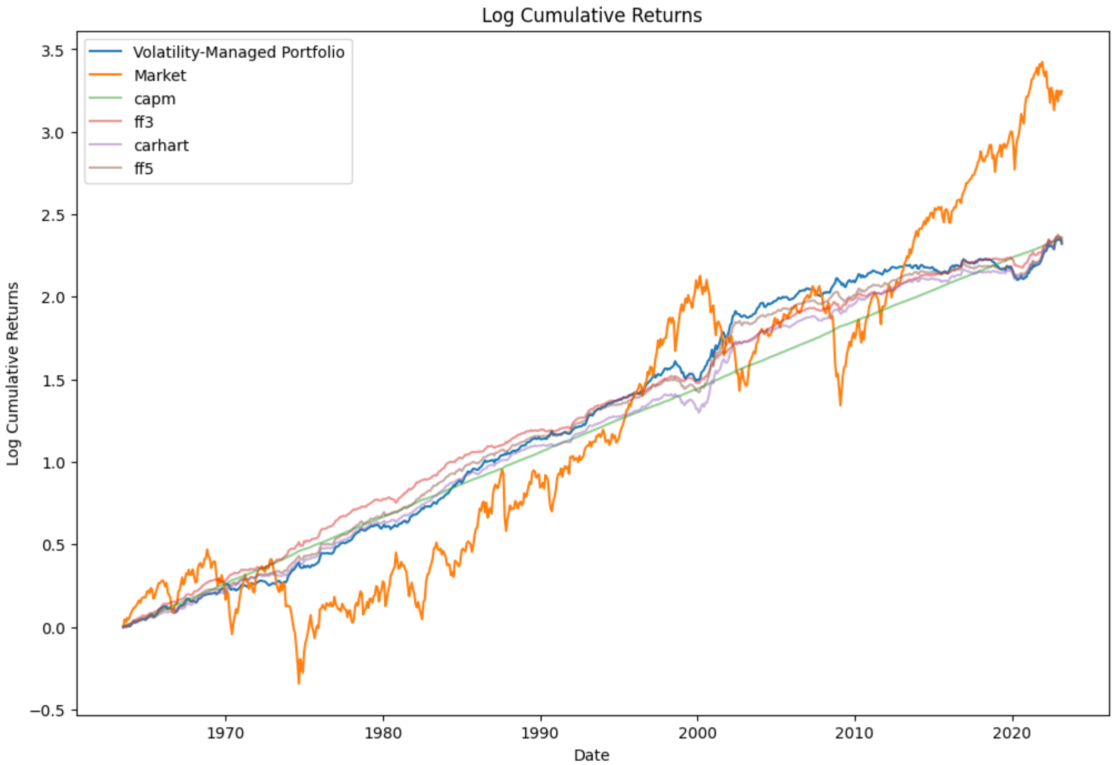
[ ]:

```python
import matplotlib.pyplot as plt
def get_rets(series):
    return np.log(((1 + series/100).cumprod()))
# calculate log cumulative returns
portfolio_cum_returns = get_rets(excess_returns)
market_cum_returns = get_rets(factor_returns['Mkt-RF'])
capm_cum_returns = get_rets(capm_model.predict())
ff3_cum_returns = get_rets(ff3_model.predict())
carhart_cum_returns = get_rets(carhart_model.predict())
ff5_cum_returns = get_rets(ff5_model.predict())

dt_index = excess_returns.index.to_timestamp()

# plot
plt.figure(figsize=(12, 8))
plt.plot(dt_index, list(portfolio_cum_returns), label='Volatility-Managed␣
↪Portfolio')
plt.plot(dt_index, list(market_cum_returns), label='Market')
plt.plot(dt_index, list(capm_cum_returns), label='capm', alpha=0.5)
plt.plot(dt_index, list(ff3_cum_returns), label='ff3', alpha=0.5)
plt.plot(dt_index, list(carhart_cum_returns), label='carhart', alpha=0.5)
plt.plot(dt_index, list(ff5_cum_returns), label='ff5', alpha=0.5)
plt.xlabel('Date')

plt.ylabel('Log Cumulative Returns')
plt.title('Log Cumulative Returns')
plt.legend()
plt.show()
```



(b) It seems that the aggregat portfolio outperforms each composite portfolio. Significant alphas.

(c) The volatility-managed portfolio produces alpha because it seeks to reduce risk by adjusting portfolio weights based on the volatility of the underlying assets. By doing this, the portfolio maintains a more stable performance during periods of high market volatility, which in turn leads to better risk-adjusted returns compared to the market.

However, the cumulative returns suggest that the volatility-managed portfolio does not beat the market since the turn of the century. This can be due to a few reasons: - Changing market conditions since the turn of the century, especially apparent with the effective flatline around 2008 - During periods of strong market performance, higher-risk assets tend to deliver better returns, resulting in this risk-averse strategy underperforming - During periods of low volatilty this portfolio is less effective