Empirical Asset Pricing A HW2

Xinyu Liu

January 23, 2021

1. Data Processing

I use Python to help analyze the data (see Appendix the full code). The Data range of this exercise is 03.1997-02.2017, a total span of 20 years. I get the monthly FF three-factor data, returns of 25 portfolios formed on size and book-to-market, and 10 portfolios formed on momentum from Kenneth French. Rather than relying on summary statistics, I provide graphic interpretation of the data, including both the factor mimicking portfolio and the testing portfolio.

Figure 1: Time series of FF-3-factor mimicking portfolio's monthly return

Mkt-RF is the excess market return, SMB and HML correspond to size and value portfolios respectively.

First and foremost, I notice that all factors fluctuate quite a lot over time. And good times seems to be less volatile compared with bad times. Therefore. they may have the potential to explain the risks in testing portfolios I will be looking at. Second, I calculate the average monthly return of these portfolios, which provides positive evidences for the existence of structural factor returns associated with them. I calculate the in-sample correlation coefficient of these factors to check that they are relatively independent sources of variance (the coefficients are relatively small in absolute sense).

Table 1: Correlation coefficients of FF-3-factor

	Mkt-RF	SMB	HML
Mkt-RF	1.00	0.24	-0.15
SMB	0.24	1.00	-0.28
HML	-0.15	-0.28	1.00

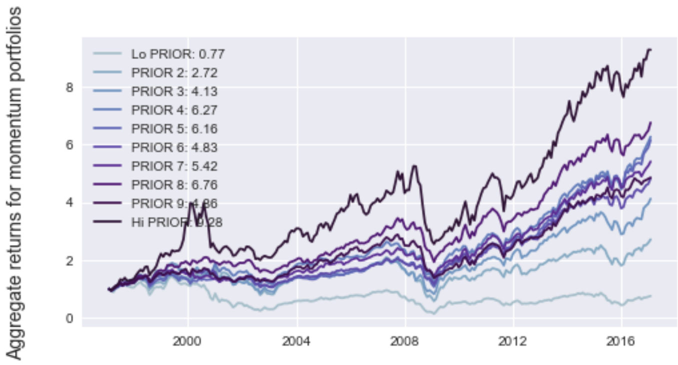
To give a more intuitive picture of the three factors, I plot the aggregate return of all factors on the same graph, which indicates a clear upward trend.

Figure 2: Time series of FF-3-factor mimicking portfolio's aggregate return

I add an additional row of 1s as the initial period's gross return (equivalently, can be regarded as starting off with 1 dollar), and compound the monthly returns. Although the aggregate gross return can get smaller than 1 sometimes, but it seems reasonable to believe that in a longer term these factors tend to deliver positive returns. In the legend I give the end gross return of the portfolio.

Next, I briefly show the aggregate returns of testing portfolios. I begin by first introducing momentum testing portfolios, mainly because it's of just one dimension thus can be easily presented.

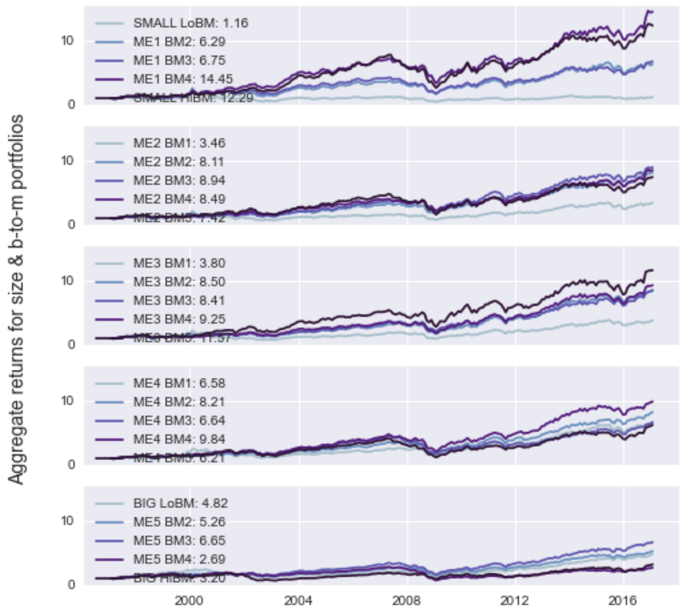
Figure 3: Time series of momentum testing portfolio's aggregate return



Following the construction of FF, stocks are divided into ten portfolios each month, based on their previous return from 1 year ago to the second last month. As is seen from the graph, winners continue to win. This heterogeneity in returns associated with momentum characteristic demands explanation from risk factor perspective. Another interesting observation is that portfolios of high momentum seem to do poorly during financial crisis.

As for the 5 $\times$ 5 size and book-to-market portfolios, I plot them on five subplots, based on the size category of the firm:

Figure 4: Time series of size and book-to-market testing portfolio's aggregate return



Following the construction of FF, stocks are two-way sorted into 25 portfolios in June each year, based on their previous size and book to market characteristics. With in each plot, it's the book-to-market variation that's associated with the returns. Expect for the largest firm category, most of the subgroups demonstrate a monotonic increase of returns as the book-to-market ratio increases. Second, if compared across subplots holding the book-to-market category fixed, small firms tend to deliver higher returns (See the average position of the end point of returns across subplots).

With the motivation from these descriptive graphs, I then move on to the time series regression.

2. Time series regressions

Before anything, note that the returns of all testing portfolios should be subtracted by the corresponding risk-free rate, in order to fit in the model of excess return:

$$R^e_{i,t} = a_i + \beta_i(Mkt - RF)_t + s_i SMB_t + h_i HML_t + \eta_{i,t}, \forall i, t$$

I use OLS to estimate the intercept a_i. The estimated intercepts, together with their t-stats and R^2(adjusted), are reported below. First for momentum testing portfolios:

Table 2: OLS estimated intercept of momentum testing portfolios

	Lo PRIOR	2	3	4	5	6	7	8	9	Hi PRIOR
intercept	-0.89	-0.35	-0.14	0.08	0.10	0.02	0.11	0.20	0.07	0.37
t-stats	-2.17	-1.43	-0.71	0.56	0.79	0.19	0.89	1.70	0.49	1.68
R^2_{adj}	0.62	0.72	0.73	0.80	0.83	0.82	0.80	0.82	0.77	0.72

From the table it can be seen that there is a clear increase in the intercept from loser portfolio to winner portfolio, and the intercept of the portfolio of prior worst performing stocks is statistically significant at 5% level. Which implies an opportunity to construct a zero cost portfolio that delivers significant positive alpha. Or to put equivalently, it's likely to find a higher ex-post sharp ratio that the that of the market, implying a rejection of the null hypothesis.

More officially, I implement GRS test, the finite sample version of the Wald test for the joint null that all a_i are zero. If correctly specified, the test statistic J_1 should follow a $F(N, T-N-1)$ distribution. I get the $J_1 = 1.77$, and its p-value is 0.068, meaning I can reject the null at 10% level.

Next, for size and book-to-market testing portfolios:

Table 3: OLS estimated intercept of momentum testing portfolios

Panel A: Intercept					
	BM1	BM2	BM3	BM4	BM5
ME1	-0.59	0.04	0.00	0.31	0.18
ME2	-0.17	0.10	0.11	0.05	-0.11
ME3	-0.07	0.13	0.11	0.10	0.14
ME4	0.16	0.15	0.03	0.19	-0.11
ME5	0.15	0.07	0.14	-0.35	-0.30
Panel B: T-stats					
	BM1	BM2	BM3	BM4	BM5
ME1	-3.08	0.27	0.02	3.16	1.82
ME2	-1.39	0.93	0.97	0.52	-1.28
ME3	-0.66	1.09	0.91	0.82	0.92
ME4	1.43	1.19	0.20	1.43	-0.69
ME5	2.85	0.75	1.21	-2.84	-1.41
Panel C: R^2-squared					
	BM1	BM2	BM3	BM4	BM5
ME1	0.89	0.91	0.94	0.93	0.94
ME2	0.94	0.93	0.90	0.94	0.96
ME3	0.94	0.89	0.87	0.88	0.86
ME4	0.93	0.86	0.84	0.85	0.85
ME5	0.97	0.89	0.85	0.88	0.76

Here, row index 'ME' represents the size dimension, measured using market value. The larger the suffix, the bigger the company. Column index 'BM' represents the book-to-market dimension, measured by the book-to-market ratio. The larger the suffix, the lower the market value relative to the book value of the company.

There are 4 out of 25 intercept coefficients significant at 5% level, which can be regarded as a fairly good explanation. Also note that the R^2 is around 0.9, i.e. the FF-3-factor explains most of the variations in the variation of these portfolios. However, I compute the GRS test statistic and get $J_1 = 3.20$, and p-value is 2×10^{-6}, i.e., I should confidently reject the null that all intercepts are zero.

3. Cross-sectional regressions

For this section, I first obtain the estimates of each testing portfolio's risk loading on the factor mimicking portfolios, which are then used as proxies of betas in the cross-sectional regression of return of testing portfolios. The period by period cross-sectional regression will produce a number of estimations of the factor risk premiums. The following model expresses the essence of the cross-sectional stage.

$$R^e_{i,t} = \alpha_t + (Mkt - RF)_t \hat{\beta}_i + SMB_t \hat{s}_i + HML_t \hat{h}_i + \varepsilon_{i,t}, \forall i, t$$

If we assume that these estimations are independent from each other, we can simple use their means as the best estimate of their true value, and use the sample standard error to get the t-statistics. The detailed procedure is documented in the code. I report the result below.

Table 4: Fama-Macbeth cross-sectional regression results

Panel A: Momentum testing portfolios				
	const	Mkt-RF	SMB	HML
FM coef	1.59	-1.05	1.49	0.75
t-stats	2.89	-1.64	1.75	0.89
Panel B: Size&B/M testing portfolios				
FM coef	1.70	-1.05	0.15	0.19
t-stats	4.09	-2.08	0.66	0.91

The test results seem to be against the model. In both cases the intercept is statistically significant positive. In other words, there is quite a portion of excess return that FF-3-factor model fails to capture. The market risk premium is negative in both cases which fails to satisfy the positive condition in the first place. Second, both value and size premiums are positive in both cases, where the size and book-to-market testing portfolios have closer estimates with the average returns of the mimicking portfolios (see Figure 1).

Appendix

```python
# # Empirical Asset Pricing A 2021
# ## Homework 2: on empirical tests for asset pricing models
# **Xinyu Liu, INSEAD**
#
# **20.01.2021**

# ## Overview
#
# The goal of this exercise is to get familiar with the common practice used to test classical FF 3-factor asset pricing model. Both time series and cross-sectional tests are implemented.

# ## Preparation: Import packages and access data
import pandas_datareader.data as web  # module for reading datasets directly from the web
#pip install pandas-datareader (in case you haven't install this package)
from pandas_datareader.famafrench import get_available_datasets
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
plt.style.use('seaborn')
from matplotlib.dates import DateFormatter
import matplotlib.dates as mdates
import statsmodels.api as sm
import scipy as sp
from dateutil.relativedelta import relativedelta
# print latex
# from IPython.display import display, Math

###########################
# Fama French Factor Grabber
###########################
#https://randlow.github.io/posts/finance-economics/pandas-datareader-KF/
#Please refer to this link if you have any further questions.

#You can extract all the available datasets from Ken French's website and find that there are 297 of them. We can opt to see all the datasets available.
datasets = get_available_datasets()
print('No. of datasets:{}'.format(len(datasets)))
#datasets # comment out if you want to see all the datasets

###########################
#Customize your data selection
###########################
#It is important to check the description of the dataset we access by using the following codes
sdate='1997-03-01'
edate='2017-02-27'


# #### For $M kt-Rf, SMB, HML$ Factors:

Datatoread='F-F_Research_Data_Factors'
ds_factors = web.DataReader(Datatoread,'famafrench',start=sdate,end=edate) # Taking [0] as extracting 1F-F-Research_Data_Factors_2x3')
print('\nKEYS\n{}'.format(ds_factors.keys()))
print('DATASET DESCRIPTION \n {}'.format(ds_factors['DESCR']))
#From the printed information we know that we need to select the "0" name in the dictionary
#copy the right dict for later examination
dfFactor = ds_factors[0].copy()
dfFactor.reset_index(inplace=True)

#Date format adjustment
# dfFactor['Date']=dfFactor['Date'].dt.strftime('%Y-%m')
dfFactor = dfFactor.set_index(['Date'])
# dfFactor['Date']=dfFactor['Date'].dt.to_timestamp(freq='M').dt.strftime('%Y-%m')
#Obtained object dtype
# dfFactor.index=pd.to_datetime(dfFactor.index)
#Obtained dt64, which is needed for the plotting
```

```python

RF = dfFactor['RF']
dfFactor=dfFactor.drop(columns = ['RF'])
# I check the scale of the data by printing out the head:
dfFactor.head()


# #### For 25 portfolios formed on size and book-to-market (5 x 5)

# I searched for the exact name for this portfolio set by methods mentioned above
#It is important to check the description of the dataset we access by using the following codes
Datatoread_PORT='25_Portfolios_5x5'
ds_PORT = web.DataReader(Datatoread_PORT,'famafrench',start=sdate,end=edate) # Taking [0] as extracting 1F-F-Research_Data_Factors_2x3')
print('\nKEYS\n{}'.format(ds_PORT.keys()))
print('DATASET DESCRIPTION \n {}'.format(ds_PORT['DESCR']))
#From the printed information we know that we need to select the "0" name in the dictionary
#copy the right dict for later examination
dfPORT = ds_PORT[0].copy()
dfPORT.reset_index(inplace=True)

dfPORT = dfPORT.set_index(['Date'])
# I check the scale of the data by printing out the head:
dfPORT.head()


# #### For 10 portfolios formed on momentum

Datatoread_MOM='10_Portfolios_Prior_12_2'
ds_MOM = web.DataReader(Datatoread_MOM,'famafrench',start=sdate,end=edate) # Taking [0] as extracting 1F-F-Research_Data_Factors_2x3')
print('\nKEYS\n{}'.format(ds_MOM.keys()))
print('DATASET DESCRIPTION \n {}'.format(ds_MOM['DESCR']))
dfMOM = ds_MOM[0].copy()
dfMOM.reset_index(inplace=True)

dfMOM = dfMOM.set_index(['Date'])
# I check the scale of the data by printing out the head:
dfMOM.head()


# ## Test functions
# #### Define the function for conducting time series test

def Time_Series_Test(factor_matrix, test_assets, RF):
    X = sm.add_constant(factor_matrix)
    const_value = list()
    t_value = list()
    rsquared_adj_value = list()
    risidual_matrix = pd.DataFrame()
    # Loop to perform regression
    # Note that we should deduct RF from the portfolio return to get the excess return
    for i in range(len(test_assets.columns)):
        y= test_assets.iloc[:,i]-RF
        model = sm.OLS(y, X)
        results = model.fit()
        const_value.append(results.params[0])
        t_value.append(results.tvalues[0])
        rsquared_adj_value.append(results.rsquared_adj)
        if i == 0:
            risidual_matrix = pd.DataFrame(results.resid,columns=[i])
        else:
            risidual_matrix=risidual_matrix.join(pd.DataFrame(results.resid,columns=[i]))
    # convert result into dataframe
    ts_result = {'intercept': const_value, 't-stats': t_value, 'R^2-adj': rsquared_adj_value, 'test_assets_name': test_assets.columns}
    ts_result = pd.DataFrame.from_dict(ts_result, orient='index')
    ts_result.columns = ts_result.loc['test_assets_name',:]
    ts_result = ts_result.drop(['test_assets_name'])
    del ts_result.columns.name
```

```

    # Compute GRS test statistics
    T = len(test_assets.index)
    N = len(test_assets.columns)
    mu_mkt = factor_matrix['Mkt-RF'].mean()
    sigma_mkt = factor_matrix['Mkt-RF'].std()
    alpha = ts_result.T['intercept']
    GRS_sigma = (risidual_matrix.T @ risidual_matrix)/T
    GRS_sigma = np.matrix(GRS_sigma)
    GRS_sigma = np.linalg.inv(GRS_sigma)
    GRS_sigma_inv = pd.DataFrame(GRS_sigma)
    ## Key formula to calculate the statistics
    J_1 = (T-N-1)/N*(1+(mu_mkt/sigma_mkt)**2)**(-1)*np.dot(np.dot(alpha.T,GRS_sigma_inv),alpha)

    # Test procedure
    df1 = N
    df2 = T-N-1
    p_value = 1 - sp.stats.f.cdf(J_1, df1, df2)
    print('The GRS test statistic J_1 is {:2.2f}, and its p-value is {:2.6f}'.format(J_1, p_value))
    ts_result=ts_result.astype(float).round(2)
    print(ts_result.to_latex())
    return ts_result, np.round(J_1, 3), np.round(p_value, 3)

ts_result, J_1, p_value= Time_Series_Test(dfFactor, dfMOM, RF)
ts_result, J_1, p_value= Time_Series_Test(dfFactor, dfPORT, RF)

# Make the output table more readable
for content in ts_result.index:
    print_report = pd.DataFrame(ts_result.loc[content,:].values.reshape(5,5),columns= ["BM" + str(i+1) for i in range(5)], index= ["ME" + str(i+1) for i in range(5)])
    print_report = pd.concat([print_report], axis=1, keys=[content])
    print(print_report.to_latex())


# ## Plot for building up intuitions

###########################
#Plot out the graphs
###########################
#See this link for detailed guidance on date ticks
# https://matplotlib.org/3.1.1/gallery/text_labels_and_annotations/date.html
# I am troubled by adjusting the format and making subplots for the whole evening and it turns out that things can be simplified in the following way:
years_fmt = mdates.DateFormatter('%Y')
#This will be used as input to adjust the axis label to be in the unit of year
n = len(dfFactor.columns)
fig, axes = plt.subplots(n,1,figsize=(8,8),sharex=True,sharey=True)
#Using sharex help making the plot simple and easy to read
# Create fig and axes class so I can then process with them in the for loop.
# fig.suptitle('Time series of relevant variables',fontsize=16)
for k,factortitle in enumerate(dfFactor.columns):
    ax = axes[k]
#      ax.set_xticks(dfFactor.index)
    ax.plot(dfFactor.index.to_timestamp(),dfFactor[factortitle])
    ax.axhline(y=dfFactor[factortitle].mean(),color='r', linestyle='--',label='Average monthly return is {:.2f}'.format(dfFactor[factortitle].mean()))
    ax.xaxis.set_major_formatter(years_fmt)
    ax.set_ylabel(factortitle,fontsize = 14)
    ax.legend(fontsize = 14,loc=2)
plt.savefig("Time series of momnthly factor returns")
plt.show()

print(dfFactor.corr().round(2).to_latex())


# #### Define a function to plot aggregate gross returns of factor mimicking portfolios and testing portfolios

def portfolio_plot(df, num_subplot, plot_name='testing' ,figsize=(8,8), cmap ='twilight'):
    n = num_subplot
```

```
    fig, axes = plt.subplots(n,1,figsize=figsize,sharex=True,sharey=True)

    # fig.suptitle('Time series of relevant variables',fontsize=16)
    # Add an origin point at the top of the dataframe
    dfcopy = df.copy()
    dfcopy.index = dfcopy.index.to_timestamp()
    origin = dfcopy.index[0]-relativedelta(months=1)
    dfcopy.loc[origin,:] = [1]*len(dfcopy.columns)
    dfcopy=dfcopy.sort_index()

    dfFactor_cum = (dfcopy/100+1).cumprod()
    for k,factortitle in enumerate(dfcopy.columns):
        if n==1:
            ax = axes
        else:
            ax = axes[k//n]
        ax.plot(dfFactor_cum.index,dfFactor_cum[factortitle], label='{}: {:.2f}'.format(
    factortitle, dfFactor_cum[factortitle][-1]))
        ax.xaxis.set_major_formatter(years_fmt)
        colormap = plt.cm.get_cmap(cmap)
        colors = [colormap(i) for i in np.linspace(0.1, 0.5,len(ax.lines))]
        for i,j in enumerate(ax.lines):
            j.set_color(colors[i])
        ax.legend(fontsize = 10,loc=2)
    fig.text(0.04, 0.5, 'Aggregate returns for ' +plot_name+' portfolios', va='center',
    ha='center',rotation='vertical',fontsize = 14)
    plt.savefig("Time series of "+plot_name)
    plt.show()


portfolio_plot(dfFactor, 1, plot_name='factor' ,figsize=(8,4), cmap ='twilight')

portfolio_plot(dfMOM, 1, plot_name='momentum' ,figsize=(8,4), cmap ='twilight')

portfolio_plot(dfPORT, 5, plot_name="size & b-to-m" ,figsize=(8,8), cmap ='twilight')


# #### Define the function for conducting cross-sectional test
def FamaMacbeth_Test(factor_matrix, test_assets, RF):
    # Step one, time series regression, obtain estimated beta for each portfolio
    X = sm.add_constant(factor_matrix)
    beta_matrix = pd.DataFrame()
    for i in range(len(test_assets.columns)):
        y= test_assets.iloc[:,i]-RF
        model = sm.OLS(y, X)
        results = model.fit()
        beta_i = pd.DataFrame(results.params[1:]).T
        beta_matrix= pd.concat([beta_matrix, beta_i])
    beta_matrix.index = test_assets.columns

    # Step two, cross sectional regression, obtain estimated intercept and factor risk
    premium period by period
    X = sm.add_constant(beta_matrix)
    premium_matrix = pd.DataFrame()
    for i in range(len(test_assets.index)):
        # Note to be consisitent we should still use the excess return
        y= test_assets.iloc[i,:]-RF[i]
        model = sm.OLS(y, X)
        results = model.fit()
        premium_i = pd.DataFrame(results.params).T
        premium_matrix= pd.concat([premium_matrix, premium_i])
    premium_matrix.index = factor_matrix.index

    ## Key formula to calculate the statistics
    point_estimate = premium_matrix.mean()
    N = len(test_assets.index)
    std = premium_matrix.std()/np.sqrt(N)
    df = N-1
    significant_level = 0.975
    critical_value = sp.stats.t.ppf(significant_level, df)
    CI = [point_estimate-std*critical_value, point_estimate+std*critical_value]
    reports = pd.DataFrame(point_estimate).T
    reports = reports.rename(index={0:'FM coef'})
```

```python
    reports.loc['t-stats',:]= reports.iloc[0,:]/std

    print(reports.round(2).to_latex())
    return premium_matrix, point_estimate, reports

premium_matrix, point_estimate, reports = FamaMacbeth_Test(dfFactor, dfPORT, RF)
```