Empirical Asset Pricing A HW3

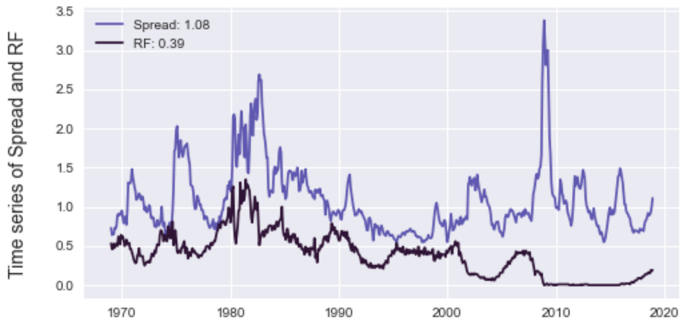
Xinyu Liu

February 2, 2021

1. Data Processing

I use Python to help analyze the data (see Appendix B the full code). The Data range of this exercise is 01.1969-12.2018, a total span of 50 years. I get the monthly FF three-factor data, and returns of 25 portfolios formed on size and book-to-market from Kenneth French; monthly labor compensation from BEA Table 2.6; and monthly time-series of the default spread ("Baa - Aaa") from FRED. I first discuss the test of conditional CAPM and then come to the return prediction practice. Note that I also go through the practice using unconditional CAPM as a benchmark, the result of which is given in the Appendix A.

Figure 1: Time series of monthly default spread and the short rate (%)

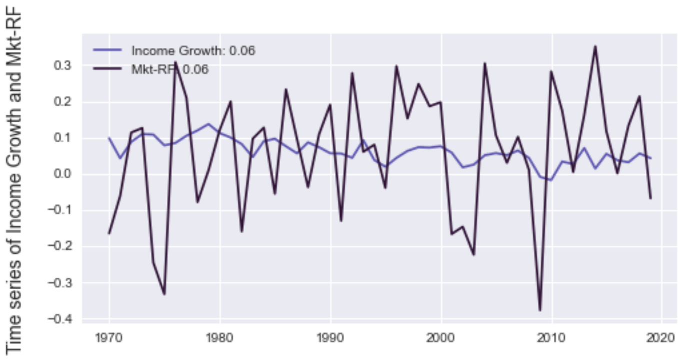


Spread is the difference of Moody's Seasoned Aaa Corporate Bond Yield [AAA] and Baa Corporate Bond Yield [BAA] (not seasonally adjusted), retrieved from FRED, Federal Reserve Bank of St. Louis. RF is the from Fama-French, proxied by the yield of 3-month US Treasury bill, not adjusted by inflation.

I plot this graph because I hope to get a sense of the moves of the bond yield spread, denoted as R^{prem}, which is used by Jagannathan and Wang (1996) as a proxy of market risk premium. In particular, they assume that the market risk premium, denoted as $E_t(R_{t+1}^{Mkt-RF})$, is a linear function of R_t^{prem}. And their argument says: " stock prices vary over the business cycle, and market risk premium will also vary over the business cycle. . . ," ". . . , interest-rate variables are likely to be most helpful in predicting future business conditions." As suggested by the literature, I do find a strong association between the pattern of Spread and the business cycle. Interestingly, another observation is that the risk-free rate has been pushing to the ground for a decade, now it is just 0.03%, due to the effect of QE.

The next graph gives the time series of labor income growth:

Figure 2: Time series of annual labor income growth and the market return



In terms of volatility, labor income growth is much more stable than the market return. The inclusion of labor income is to better capture the risk of the market portfolio. Note that I annualise the growth and return to get a better sense of the patterns.

2. Time Series Regression

In this part, I use the same technique to estimate the factor loading of 25 portfolios. I report their betas as follows:

Table 1: Factor loading estimates of 25 testing portfolios

Panel A: Mkt-RF	BM1	BM2	BM3	BM4	BM5
ME1	1.41	1.23	1.10	1.01	1.04
ME2	1.39	1.17	1.04	1.00	1.11
ME3	1.32	1.12	1.00	0.96	1.04
ME4	1.23	1.08	1.00	0.94	1.06
ME5	0.98	0.94	0.87	0.88	0.94
Panel B: Spread	BM1	BM2	BM3	BM4	BM5
ME1	0.60	0.57	0.77	0.46	0.43
ME2	0.45	0.69	0.58	0.49	0.20
ME3	0.42	0.65	0.46	0.43	0.40
ME4	0.18	0.37	0.23	0.23	0.06
ME5	-0.16	0.01	-0.27	-0.46	-0.18
Panel C: Labor	BM1	BM2	BM3	BM4	BM5
ME1	0.57	0.33	0.32	0.38	0.41
ME2	0.08	0.12	0.06	0.34	0.31
ME3	0.08	0.10	-0.05	-0.06	-0.13
ME4	-0.12	-0.16	-0.19	-0.05	0.27
ME5	-0.19	0.03	-0.02	0.10	0.20

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.

Overall I notice a strong tendency for small stocks to have high loadings on all three risk factors. Second, the loadings by large fail to explain the risks premium associated with high book-to-market characteristics, as it shows that high ratios correspond to low loadings. This raises a warning for the choice of testing portfolios, which I will get back in the next section.

3. Cross Sectional Regression

For the second stage, I use Fama-Macbeth to estimate the factor loadings, I add a constant in the regression and test whether it is significantly different from 0:

$$R_i^e = c + c_{vw}\hat{\beta}_i + c_{labor}\hat{\beta}_i^{labor} + c_{prem}\hat{\beta}_i^{prem} + \varepsilon_i, \forall i$$

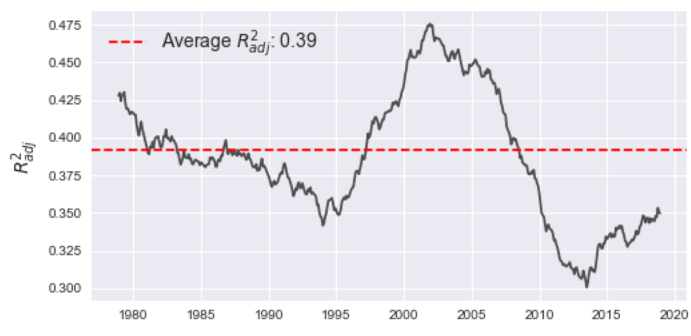
If we assume that these estimations are independent from each other, we can simple use their means as the best estimate of their true value, and use the sample standard error to get the t-statistics. The detailed procedure is documented in the code. I report the result below.

Table 2: Fama-Macbeth estimates of price of risk

Price of risk estimated from 25 testing portfolios				
	c	c_{vw}	c_{prem}	c_{labor}
FM coef	1.65	-1.04	0.34	-0.00
t-stats	4.39	-2.65	2.85	-0.02

Apparently this model is not well supported by the test. More specifically, Its constant is significantly positive, indicating a large portion of unexplained premiums. Second, the market risk premium c_{vw} is negative and significant, violating our assumption that it should deliver positive risk premium. Thirdly, c_{prem} is not significant at all, contradicting with the estimate of Jagannathan and Wang (1996)[1]. My guess is that this has to do with the test portfolio. In the paper they choose 100 portfolios sorted on size, which is less convincing than using the double-sorting portfolios in my view. Next, I report the average R^2 and plot it using a rolling window of 10 years.

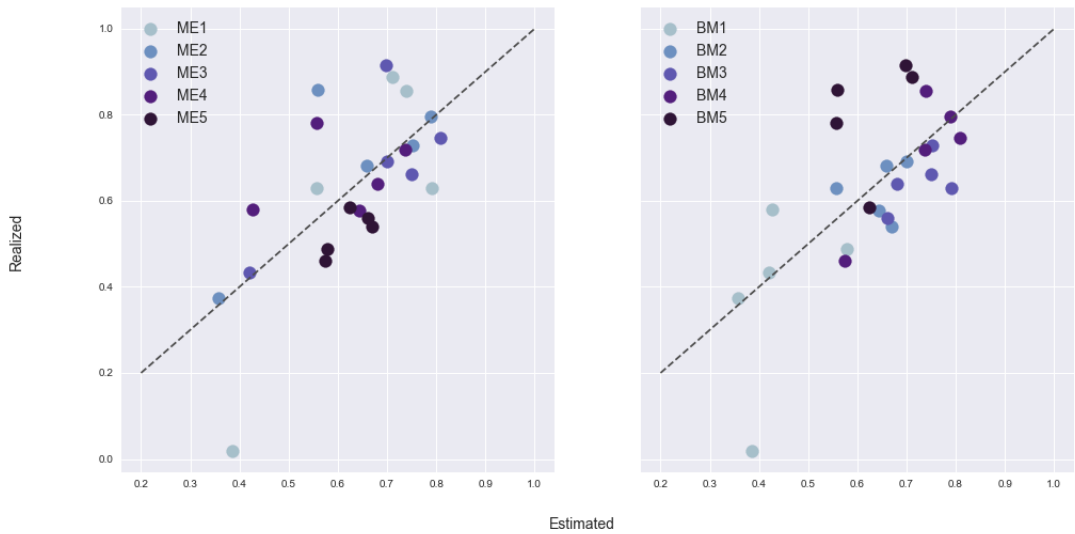
Figure 3: Moving average of R^2_{adj} from Fama-Macbeth



There is quite a bit variation of power of explanation across time. Again, this result is not as strong as the result from the original paper (0.55). Lastly, I show the scatter plots group by ME and BM.

[1] I tried to use the same time period but this does not help get any closer.

Figure 4: Scatter plots between realized and estimated excess return



From this graph we can evaluate the fitting condition. First of all, the overall fitting situation is captured by R^2, and there is a moderate linear association between the estimation and real excess returns. However, there is also sizable deviation from the diagonal, suggesting that the conditional CAPM model is still insufficient to capture the entire risk structure. Thirdly, compared with the left scatter plot, the right one classified by group of BM ratios shows less explaining power within each subgroups (the estimated returns do not move accordingly as size changes). Lastly, the least fitted point is the portfolio with smallest size and lowest book to market value, where the estimated value is much higher than realized excess return.

I use the 25 market cap and book to market ratio sorted portfolios to study the difference (see Appendix A) between the unconditional CAPM and the conditional CAPM (market portfolio adjusted) advocated by Jagannathan and Wang (1996), the estimated betas of market excess returns remain almost the same. The price of risk from Fama-Macbeth shows a decrease in market risk and an increase in the unexplained intercept, which seems to make things worse. By looking at the R^2 and scattered plot, the fitting is indeed better in conditional CAPM. Overall the attempt to use conditional CAPM delivers a mix change to the original model.

For the purpose of completeness, I also use the sorting originally used by Jagannathan and Wang (1996), but with a coarser grid[2]. The result aligns largely with their findings, and it has a better fit than the previous testing portfolio (average adjusted R^2 is 0.52, see fig. 8). Except that I don't get significant price of risk on labor in come growth (see table 6). The simple comparison raise questions on the credibility of the model used in Jagannathan and Wang (1996). The change of testing portfolios fundamentally impacts the power of explanation.

4. Return predictability

Below I run predictability regressions for the one-year ahead market excess return using (a) the default spread and (b) the short rate as predictor.

[2] 25 market cap and beta sorted portfolios

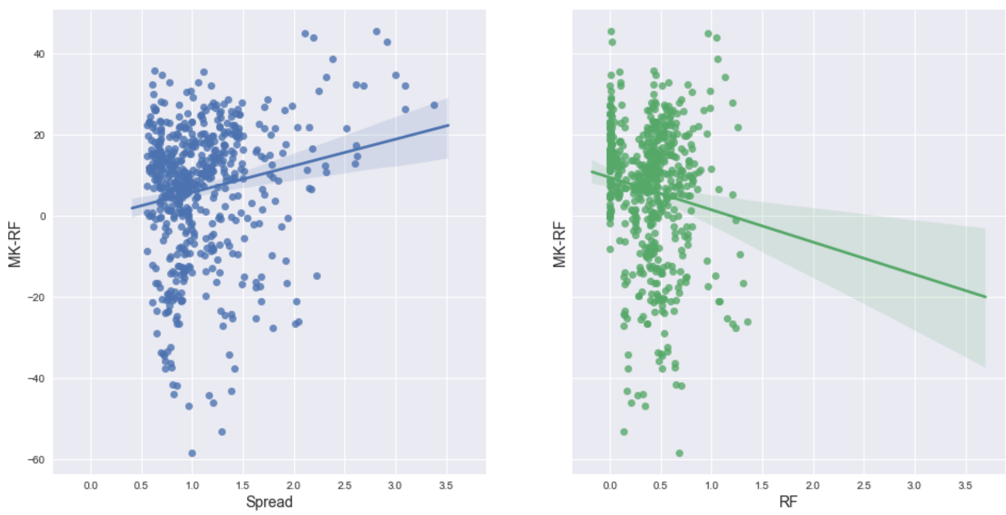
Table 3: Predictability regressions

Univariate regression coefficients and t-values		
	RF	Spread
coef	NaN	6.55
t	NaN	4.32
coef	-7.96	NaN
t	-3.27	NaN

I conduct two regression tests, both using 1 year ahead market excess return as the dependent variable, and short rate and yield spread as regressor respectively. Note that I calculate the 1 year ahead market excess return using rolling window sum.

Note that both regressions show significant prediction power, the sign of the coefficients aligns with intuition: worse economic environment is associated with low risk free rate and high yield spread, which is also the time when risk premium is high. However, I **cannot** find strong short term statistical correlation between this Spread and **next month** market excess return (very small correlation coefficient: -0.02). This also coincides with the classical observation that return predictability is mostly seen in the longer horizon. Lastly, I show the scatter plots using regressors and the 1 year ahead market excess return:

Figure 5: Scatter plot from predictive regressions



References

Jagannathan, R. and Wang, Z. (1996). The conditional capm and the cross-section of expected returns. The Journal of finance, 51(1):3–53.

Appendices

Appendix A

The following results are counterpart benchmarks for the second practice question.

Table 4: Factor loading estimates of 25 testing portfolios, unconditional CAPM

Panel A: Mkt-RF	BM1	BM2	BM3	BM4	BM5
ME1	1.42	1.23	1.10	1.02	1.04
ME2	1.39	1.18	1.05	1.00	1.11
ME3	1.32	1.12	1.00	0.96	1.04
ME4	1.23	1.09	1.00	0.95	1.07
ME5	0.98	0.94	0.86	0.88	0.94

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.

Table 5: Fama-Macbeth estimates of price of risk, unconditional CAPM

Price of risk estimated from 25 testing portfolios	c	c_{vw}
FM coef	1.30	-0.62
t-stats	3.26	-1.43

Figure 6: Moving average of R^2_{adj} from Fama-Macbeth, unconditional CAPM

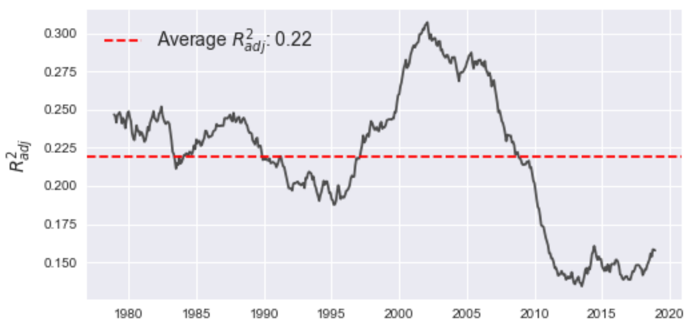


Figure 7: Scatter plots between realized and estimated excess return, unconditional CAPM

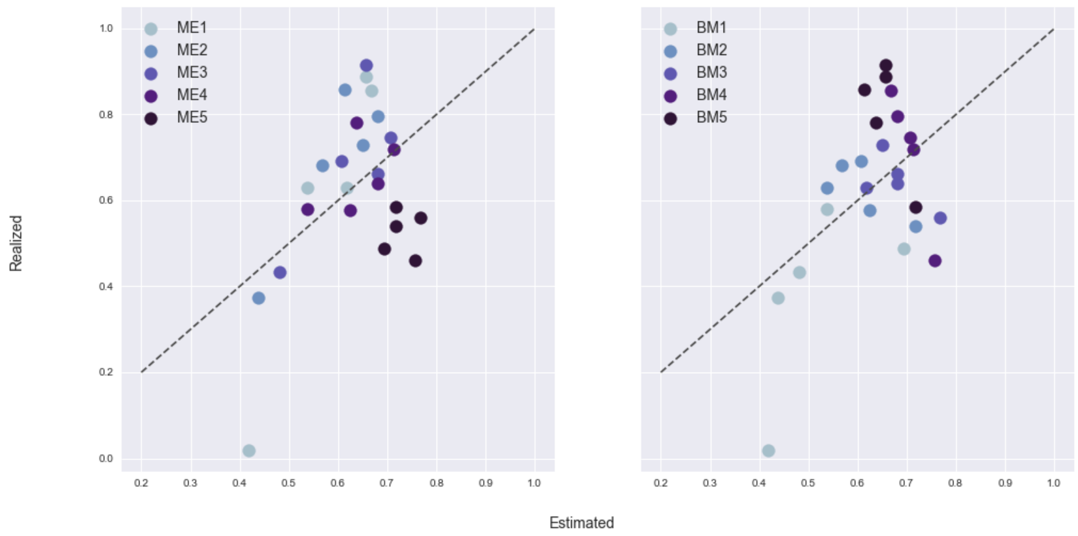
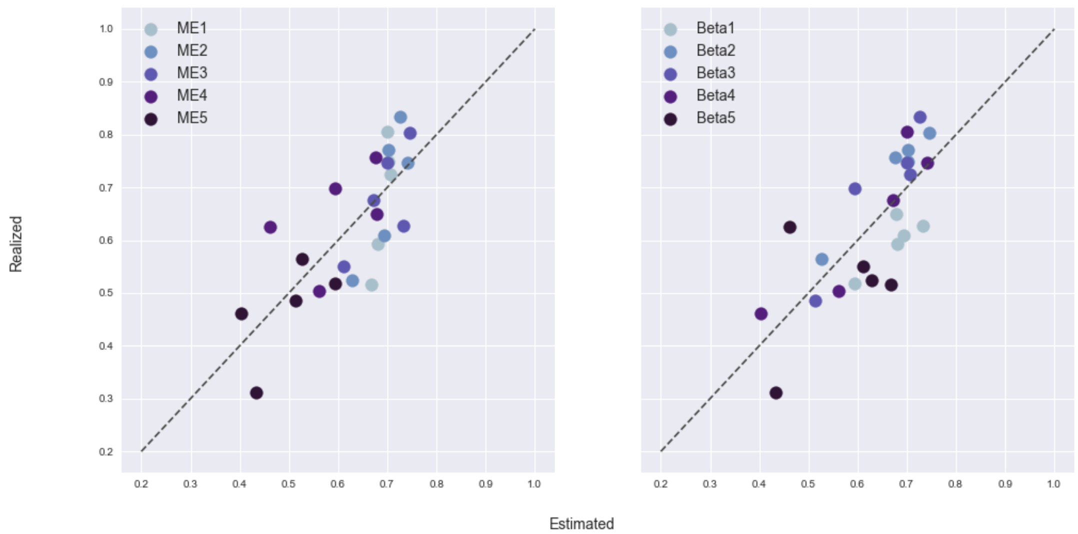


Table 6: Fama-Macbeth estimates of price of risk, 25 ME&Beta sorted portfolios

Price of risk estimated from 25 testing portfolios				
	c	c_{vw}	c_{prem}	c_{labor}
FM coef	0.77	-0.22	0.35	0.09
t-stats	4.53	-0.89	2.48	0.45

Figure 8: Scatter plots between realized and estimated excess return, 25 ME&Beta sorted portfolios



Appendix B

```python
#!/usr/bin/env python
# coding: utf-8

# # Empirical Asset Pricing A 2021
# ## Homework 3&4: on empirical tests for conditional CAPM, return predictability
# **Xinyu Liu, INSEAD**
#
# **02.02.2021**

# ## Overview
#
# The goal of this exercise is to get a sense of the testing procedures in conditional
    CAPM, and check predictability of the market return.

# ## Preparation: Import packages and access data
#

# In[1]:


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
import datapungibea as dpb
import os
# print latex
# from IPython.display import display, Math


# In[2]:


###########################
# 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


# In[7]:


###########################
#Customize your data selection
###########################
#It is important to check the description of the dataset we access by using the
    following codes
sdate='1969-01-01'
edate='2018-12-31'
dir = os.path.realpath('.')


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

```

```
# In[4]:


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()
# 0 for monthly data and 1 for yearly data
dfFactor.reset_index(inplace=True)

#Date format adjustment
# dfFactor['Date']=dfFactor['Date'].dt.strftime('%Y-%m')
dfFactor = dfFactor.set_index(['Date'])
dfFactor.index=dfFactor.index.to_timestamp()
# 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

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)

# In[16]:


# 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'
Datatoread_PORT='25_Portfolios_ME_BETA_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 monthly time-series of the default spread ( Baa - Aaa )

# In[53]:


# from fredapi import Fred
# fred = Fred(api_key='867c31a2baca3a69effa928b9b294289')
# Aaa = fred.get_series_latest_release('AAA')
# Baa = fred.get_series_latest_release('BAA')
####
# The API above is not stable so I make a local copy and access them below
####
filename = os.path.join(dir, 'Data','AAA.csv')
Aaa = pd.read_csv(filename,index_col='DATE',parse_dates=True)
filename = os.path.join(dir, 'Data','BAA.csv')
Baa = pd.read_csv(filename,index_col='DATE',parse_dates=True)

Bond_spread = pd.DataFrame({'Aaa':Aaa.iloc[:,0].values,'Baa':Baa.iloc[:,0].values},index
```

```
    = Aaa.index)
Bond_spread = Bond_spread[(Bond_spread.index<=pd.to_datetime(edate)) & (Bond_spread.index>=pd.to_datetime(sdate))]
Bond_spread['Spread'] = Bond_spread['Baa']- Bond_spread['Aaa']
dfFactor = dfFactor.merge(Bond_spread[['Spread']], how='inner', left_index=True, right_index=True)


# In[56]:


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)
    years_fmt = mdates.DateFormatter('%Y')
    # 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
    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].mean()))
        ax.xaxis.set_major_formatter(years_fmt)
        colormap = plt.cm.get_cmap(cmap)
        colors = [colormap(i) for i in np.linspace(0.3, 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, 'Time series of ' +plot_name, va='center', ha='center',rotation='vertical',fontsize = 14)
    plt.savefig("Time series of "+plot_name)
    plt.show()
portfolio_plot(dfFactor[['Spread', 'RF']], 1, plot_name='Spread and RF' ,figsize=(8,4), cmap ='twilight')


# #### For monthly time-series of labor income growth (BEA)

# In[57]:


BEA_data = dpb.data('FDA2D756-CC0A-4AAA-A1D5-980FA23F31BB') #or data = dpb.data("API Key")
NIPA_cons=BEA_data.NIPA('T20600',frequency='M')
#Download annual consumption data on nondurable goods from Table 2.6.
#on   Personal   Income and Its Disposition, Monthly
NIPA_cons.reset_index(inplace=True)
Compensation_data=NIPA_cons[NIPA_cons['LineDescription']=='-Compensation of employees']
Compensation_data = Compensation_data.T.iloc[4:,:]
Compensation_data.columns=['Compensation']
Compensation_data.index = pd.to_datetime(Compensation_data.index.values, format='%YM%m')
Compensation_data['Income Growth'] = (Compensation_data['Compensation']-Compensation_data['Compensation'].shift(1))/Compensation_data['Compensation'].shift(1)
# Convert strings to datetime
Compensation_data = Compensation_data[(Compensation_data.index<=pd.to_datetime(edate)) & (Compensation_data.index>=pd.to_datetime(sdate))]
Compensation_data['Mkt-RF'] = dfFactor['Mkt-RF']/100
Compensation_data['Income Growth'] = Compensation_data['Income Growth']
labor_market = (Compensation_data[['Income Growth','Mkt-RF']]+1).astype('f').resample('Y').prod()-1
portfolio_plot(labor_market, 1, plot_name='Income Growth and Mkt-RF (monthly)' ,figsize=(8,4), cmap ='twilight')
```

```python
dfFactor['Labor'] = Compensation_data['Income Growth'].astype('f')*100
# I don't know why but the api is not stable so I kept a copy of data
# Compensation_data.to_pickle('compensation')
#or [All just for saving the intermediary data]
# Compensation_data.to_csv(os.path.join(dir, 'Data','Compensation.csv'))
# Compensation_data = pd.read_pickle('compensation')
# dfFactor.to_csv(os.path.join(dir, 'Data','dfFactor.csv'))


# ## Test functions
# #### Define the function for conducting cross-sectional test, where the first stage is a time series regression

# In[10]:


# I can import directly the saved dfFactor
filename = os.path.join(dir, 'Data','dfFactor.csv')
dfFactor = pd.read_csv(filename,index_col='Date',parse_dates=True)


# In[17]:


def FamaMacbeth_Test(factor_matrix, test_assets, RF):
    try:
        test_assets.index = test_assets.index.to_timestamp()
    except Exception:
        pass
    # 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()
    rsquare_matrix = []
    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])

        rsquare_matrix.append(results.rsquared_adj)
    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'})
    reports.loc['t-stats',:]= reports.iloc[0,:]/std

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


# In[18]:
```

```python


beta_matrix, premium_matrix, point_estimate, rsquare_mean = FamaMacbeth_Test(dfFactor[['Mkt-RF', 'Spread','Labor']], dfPORT, RF)


# In[73]:


beta_matrix, premium_matrix, point_estimate, rsquare_mean = FamaMacbeth_Test(dfFactor[['Mkt-RF']], dfPORT, RF)


# In[286]:


# Sensitivity check for the parameters
cut = 240
beta_matrix, premium_matrix, point_estimate, rsquare_mean = FamaMacbeth_Test(dfFactor[['Mkt-RF', 'Spread','Labor']].iloc[:cut,:], dfPORT.iloc[:cut,:], RF[:cut])


# In[21]:


# Rolling average calcualtion for list data
numbers = rsquare_mean
window_size = 120
numbers_series = pd.Series(numbers)
windows = numbers_series.rolling(window_size)
moving_averages = windows.mean()
moving_averages_list = moving_averages.tolist()
without_nans = moving_averages_list[window_size - 1:]


# In[22]:


# plot time series of rolling average
fig, axes = plt.subplots(1,1,figsize=(8,4),sharex=True,sharey=True)
fig.text(0.04, 0.5, r'$R^2_{adj}$', va='center', ha='center',rotation='vertical', fontsize = 14)
colormap = plt.cm.get_cmap('twilight')
axes.plot(dfPORT.index[window_size - 1:],without_nans,c=".3")
axes.axhline(y=np.mean(rsquare_mean),color='r', linestyle='--',label='Average '+r'$R^2_{adj}$'+': {}'.format(np.round(np.mean(rsquare_mean),2)))
axes.legend(fontsize = 14)
plt.plot()
plt.savefig('Rsquared')
plt.show()


# In[23]:


# Make the output table more readable
beta_matrix = beta_matrix.round(2)
for content in beta_matrix.T.index:
    print_report = pd.DataFrame(beta_matrix.T.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())


# In[24]:


# Process result from regressions to plot scatter plot
X = sm.add_constant(beta_matrix)
Estimated = X @ point_estimate
Realized = (dfPORT.sub(RF,axis = 'index')).mean()


```

```
# In[26]:


# Make the scatter plot
fig, axes = plt.subplots(1,2,figsize=(16,8),sharex=True,sharey=True)
fig.text(0.04, 0.5, 'Realized', va='center', ha='center',rotation='vertical',fontsize =
    14)
fig.text(0.5,0.04, 'Estimated', va='center', ha='center',rotation='horizontal',fontsize
    = 14)
colormap = plt.cm.get_cmap('twilight')
colors = [colormap(i) for i in np.linspace(0.1, 0.5,5)]
axes[0].plot([0.2, 1], [0.2, 1], ls="--", c=".3")
for i in range(0,5):
    axes[0].scatter(Estimated[i*5:(i+1)*5],Realized[i*5:(i+1)*5],c=colors[i],label = 'ME
    '+str(i+1), s=140)
axes[0].legend(fontsize = 14)
axes[1].plot([0.2, 1], [0.2, 1], ls="--", c=".3")
for i in range(0,5):
    axes[1].scatter(Estimated[i::5],Realized[i::5],c=colors[i],label = 'BM'+str(i+1), s
    =140)
axes[1].legend(fontsize = 14)
plt.plot()
plt.savefig('Scatter_mebetaCAPM')
plt.show()


# ### Return predictability test
# 1. Default spread
# 2. Short rate

# In[144]:


to_predict= dfFactor[['Mkt-RF']].rolling(12).sum().shift(-12)


# In[145]:


# Make the scatter plot
import seaborn as sns
fig, axes = plt.subplots(1,2,figsize=(16,8),sharex=True,sharey=True)
colormap = plt.cm.get_cmap('twilight')
colors = [colormap(i) for i in np.linspace(0.3, 0.5,5)]
# axes[0].plot([0.2, 1], [0.2, 1], ls="--", c=".3")
for i, k in enumerate(dfFactor[['Spread','RF']].columns):
    print(i,k)
    sns.regplot(dfFactor[[k]],to_predict['Mkt-RF'],ax= axes[i])
    axes[i].set_xlabel(k, fontsize = 14)
    axes[i].set_ylabel('MK-RF', fontsize = 14)
# plt.plot()
plt.savefig('Return_predictability')
plt.show()


# In[150]:


# Output the regression test result in latex
beta_matrix = pd.DataFrame()
for i in range(len(dfFactor[['Spread','RF']].columns)):
    y = to_predict[:-12]
    X = sm.add_constant(dfFactor[['Spread','RF']].iloc[:-12,i])
    model = sm.OLS(y, X)
    results = model.fit()
    beta_i = pd.DataFrame(results.params[1:]).T
    beta_i= beta_i.rename(index= {0:'coef'})
    beta_matrix= pd.concat([beta_matrix, beta_i])
    t_i = pd.DataFrame(results.tvalues[1:]).T
    t_i= t_i.rename(index= {0:'t'})
    beta_matrix= pd.concat([beta_matrix, t_i])
print(beta_matrix.round(2).to_latex())
```