

Empirical Asset Pricing A HW1

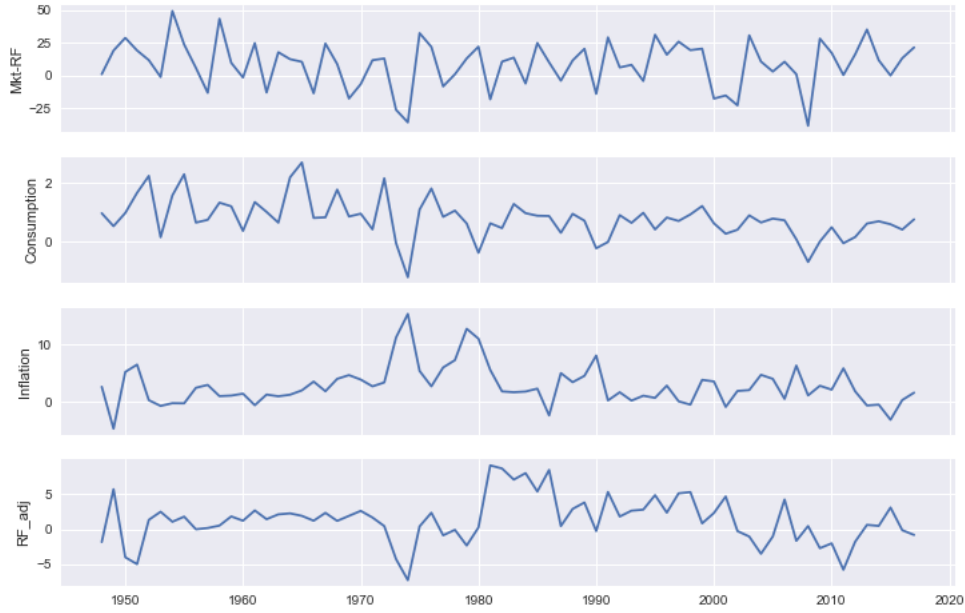
Xinyu Liu

January 13, 2021

1. Data Processing

I use Python to help analyze the data (see Appendix the full code). The Data range of this exercise is 1948-2017, a total span of 70 years. I get the annual excess market return and risk free rate from Kenneth French, quarterly consumption growth and inflation from BEA, which I annualized by taking the arithmetic average. I merged them together and make the time series plot below (in the unit of %).

Figure 1: Time series of relevant variables



Mkt-RF is the excess market return, RF_{adj} the inflation adjusted risk-free rate.

First and foremost, I find quite an apparent correlation between market return and consumption growth, which implies a positive risk premium for the market portfolio, i.e. $E[MKT - R_F] > 0$. However, consumption growth is much less volatile than market excess return, making it impossible to explain the large expected excess return observed. Another thing worth mentioning is that the risk-free rate is also quite a smooth process with little volatility.

2. Summary Statistics

I first adjust the unit of variables to 1 and then take log transformation to matching the setting of the model. I give the result of my calculation below which are quite comparable with the table presented in class:

Table 1: Summary Statistics

aer_e	r_f	$\sigma(aer_e)$	$\sigma(\Delta c)$	$Cov(er_e, \Delta c)$
0.067613	0.013193	0.176063	0.01191	0.000462

aer_e is the average annualized excess log return on the stock market over the inflation adjusted risk-free rate. $\sigma(aer_e), \sigma(c)$ denote the annualized standard deviation of this excess return and log consumption growth, respectively. $Cov(er_e, \Delta c)$ denotes the correlation between the log excess return and log consumption growth.

I check the characteristics of the first two moments of SDF, denoted as M . The data confirms that: 1. the conditional expectation of SDF is close to 1, and because the risk-free rate does not fluctuate much, the conditional mean does not move dramatically in the short run, but can vary in the long run; 2. the conditional standard deviation of the SDF should be greater to the largest conditional sharp ratio of all possible portfolio:

$$E(M) = \frac{1}{1 + r_f} \approx 0.987$$

$$\sigma(M) \geq \frac{aer_e + \frac{\sigma(aer_e)^2}{2}}{\sigma(aer)} \approx 0.431$$

Note that here I approximate the log return with return as it's very small, and the relations holds for unconditional means and variance.

3. Equity Premium Puzzle

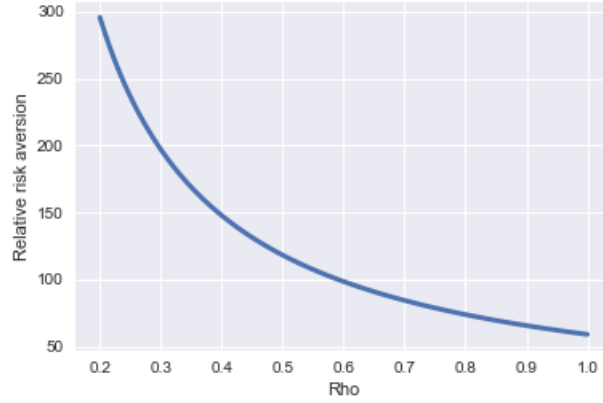
Suppose the assumptions discussed in class holds, I can infer the relative risk aversion rate (γ) using the following relationship:

$$aer_e - r_f + \sigma(aer_e)^2 = \gamma Cov(er_e, \Delta c) = \sigma(er_e) \sigma(\Delta c) \rho$$

which gives that $\gamma = 151$ (the corresponding correlation coefficient being 0.385). The correlation coefficient is twice as big as the number 0.193 from the Compbell table. This is likely because I first took simple average of the quarterly consumption growth as the annual growth ,and then calculated the coefficient from the annual data. Whereas if I use the higher frequency at quarter level of market return and consumption to calculate correlation first, then the correlation becomes $\rho = 0.182$ ($\gamma = 200$), which is very close to 0.193. I tend to believe that the original table used the second way of estimating the correlation. This is essentially a choice between more data points and potentially more noises. Different methods will result in different γ , but both are too big to be realistic. I will stick to the $\rho = 0.182$ estimates for the rest of my practice.

If I allow the correlation coefficient of the market return with the consumption growth to vary freely say from 0.2 to 1, I can calculate the lower bound of γ at the perfect correlation case, $\gamma_{min} = 60$. Also note that for $\rho = 0.2$. γ gets close to 200:

Figure 2: γ v.s. ρ



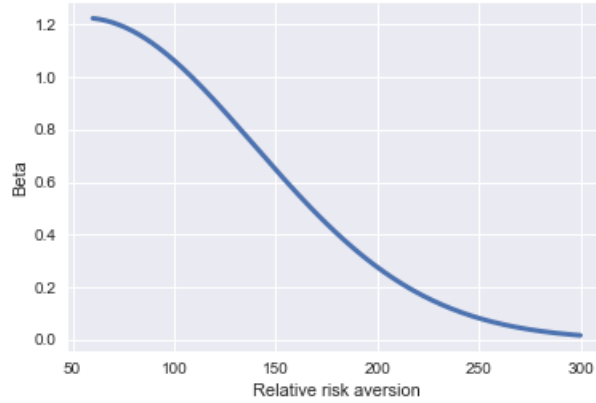
3. Risk-free Rate Puzzle

Suppose we accept that investors are as risk-averse as needed to match the equity premium puzzle, I show that the subject discount factor will be nowhere close to the acceptable range. The model discussed in class implies that:

$$r_f = -\ln \beta + \gamma E(\Delta c) - \frac{\gamma^2 \sigma(\Delta c)^2}{2}$$

When the relative risk aversion coefficient is as high as $\gamma = 200$, β must be very small to reconcile the average risk-free rate ¹. Suppose all else equal, relationship between γ and β is as follows:

Figure 3: β v.s. γ



In reality β should be slightly smaller than 1, but here it is not possible under the requirement of high γ . The trade of between intertemporal substitution and precautionary saving are so strong that β must adjust in large scale to accommodate it. Overall, my results corresponds to patterns summarized in the Campbell table. Through test, I found γ is extremely sensitive to ρ , the correlation coefficient between consumption growth and market return. And that β is sensitive to γ and $\sigma(\Delta c)$, the volatility of consumption. Given that consumption can only be measured at relatively low frequency, this constrains the testing of the model at high frequency.

¹I choose $r_f = 0.013$, $\Delta c = 0.008$, $\sigma(\Delta c) = 0.012$

Appendix

```
1 import pandas_datareader.data as web # module for reading datasets directly from the
   web
2 #pip install pandas-datareader (in case you haven't install this package)
3 from pandas_datareader.famafrench import get_available_datasets
4 import pandas as pd
5 import numpy as np
6 import datetime as dt
7 import matplotlib.pyplot as plt
8 plt.style.use('seaborn')
9 from matplotlib.dates import DateFormatter
10 import matplotlib.dates as mdates
11 import datapungibea as dpb
12
13 #####
14 # Fama French Data Grabber
15 #####
16 #https://randlow.github.io/posts/finance-economics/pandas-datareader-KF/
17 #Please refer to this link if you have any further questions.
18
19 #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.
20 datasets = get_available_datasets()
21 print('No. of datasets:{}'.format(len(datasets)))
22 #datasets # comment out if you want to see all the datasets
23
24 #####
25 #Customize your data selection
26 #####
27 #Note:If this is what you are intended to find: '6_Portfolios_ME_OP_2x3', but don't know
   exactly what it is named, do the following line
28 #df_me_op_factor = [dataset for dataset in datasets if 'ME' in dataset and 'OP' in
   dataset and '2x3' in dataset]
29 #print(df_me_op_factor)
30
31 #It is important to check the description of the dataset we access by using the
   following codes
32 Datatoread='F-F_Research_Data_Factors'
33 sdate='1948-01-01'
34 edate='2017-12-31'
35 ds_factors = web.DataReader(Datatoread,'famafrench',start=sdate,end=edate) # Taking [0]
   as extracting 1F-F-Research_Data_Factors_2x3')
36 print('\nKEYS\n{}'.format(ds_factors.keys()))
37 print('DATASET DESCRIPTION \n {}'.format(ds_factors['DESCR']))
38 #From the printed information we know that we need to select the "0" name in the
   dictionary
39 #copy the right dict for later examination
40 dfFactor = ds_factors[1].copy()
41 dfFactor.reset_index(inplace=True)
42 # I check the scale of the data by printing out the head:
43 dfFactor.head()
44
45 #Date format adjustment
46 dfFactor['Date']=dfFactor['Date'].dt.year
47 dfFactor = dfFactor.set_index(['Date'])
48 #####
49 # Consumption & Inflation Data Grabber
50 #####
51 # https://pypi.org/project/datapungibea/
52 # Connect to Bureau of Economic Analysis (BEA) API
53 BEA_data = dpb.data('FDA2D756-CC0A-4AAA-A1D5-980FA23F31BB') #or data = dpb.data("API Key
   ")
54 NIPA_cons=BEA_data.NIPA('T20302')
55 #Download annual consumption data on nondurable goods from Table 2.3.2.
56 #on Contributions to Percent Change in Real Personal Consumption Expenditures by
   Major Type of Product
57 NIPA_cons.reset_index(inplace=True)
58 Consumption_data=NIPA_cons[NIPA_cons['LineDescription']=='-Nondurable goods']
59 Consumption_data = Consumption_data.T.iloc[4,:,:]
60 Consumption_data.columns=['Consumption']
61
62 NIPA_inflation=BEA_data.NIPA('T10107')
```

```

63 NIPA_inflation.reset_index(inplace=True)
64 Inflation_data=NIPA_inflation[NIPA_inflation['LineDescription']=='--Nondurable goods']
65 Inflation_data = Inflation_data.T.iloc[4:,:]
66 Inflation_data.columns=['Inflation']
67
68 NIPA_data = pd.merge(Consumption_data,Inflation_data, how='inner', left_index= True,
69 right_index=True)
69 NIPA_data=NIPA_data.astype(float)
70 NIPA_data.index=pd.to_datetime(NIPA_data.index)
71 NIPA_data=NIPA_data.resample('Y').mean()
72 year = NIPA_data.reset_index()['index'].dt.year
73 NIPA_data.loc[:, 'Date']=year.values
74 NIPA_data = NIPA_data[(NIPA_data['Date']>=1948) & (NIPA_data['Date']<=2017)]
75 NIPA_data = NIPA_data.set_index('Date')
76
77 # Merge the data
78 merged_data = dfFactor.merge(NIPA_data,how='left', left_index=True,right_index=True)
79 merged_data = merged_data.reset_index()
80 merged_data['Date'] = pd.to_datetime(merged_data['Date'],format='%Y')
81 merged_data = merged_data.set_index('Date')
82 merged_data['Mkt'] = merged_data['Mkt-RF'] + merged_data['RF']
83 merged_data['RF_adj'] = merged_data['RF'] - merged_data['Inflation']
84 # Drop irrelevant columns
85 merged_data_plot = merged_data.drop(columns=['SMB','HML','RF','Mkt'])
86
87 #####
88 #Plot out the graphs
89 #####
90 #See this link for detailed guidance on date ticks
91 # https://matplotlib.org/3.1.1/gallery/text_labels_and_annotations/date.html
92 # I am troubled by adjusting the format and making subplots for the whole evening and it
93 # turns out that things can be simplified in the following way:
94 years_fmt = mdates.DateFormatter('%Y')
95 #This will be used as input to adjust the axis label to be in the unit of year
96 n = len(merged_data_plot.columns)
97 fig, axes = plt.subplots(n,1,figsize=(12,8),sharex=True)
98 #Using sharex help making the plot simple and easy to read
99 # Create fig and axes class so I can then process with them in the for loop.
100 # fig.suptitle('Time series of relevant variables',fontsize=16)
101 for k,factortitle in enumerate(merged_data_plot.columns):
102     ax = axes[k]
103     ax.plot(merged_data_plot.index,merged_data_plot[[factortitle]])
104     ax.xaxis.set_major_formatter(years_fmt)
105     ax.set_ylabel(factortitle)
106 plt.savefig("Time series")
107 plt.show()
108
109 #I take log transformation of the returns to fit the setting of theory
110 merged_data = np.log(merged_data/100+1)*100
111 #Calculate summary statistics
112 summary_stats = pd.DataFrame()
113 summary_stats.loc[0,'average market excess return'] = merged_data['Mkt-RF'].mean()*0.01
114 summary_stats.loc[0,'average risk-free rate'] = (merged_data['RF'].mean()-merged_data['
115 Inflation'].mean())*0.01
116 summary_stats.loc[0,'volatility of market return'] = merged_data['Mkt'].std()*0.01
117 summary_stats.loc[0,'volatility of market excess return'] = merged_data['Mkt-RF'].std()
118 *0.01
119 summary_stats.loc[0,'volatility of consumption growth'] = merged_data['Consumption'].std
120 ()*0.01
121 summary_stats.loc[0,'covariance of market and consumption'] = merged_data[['Consumption'
122 , 'Mkt']].cov().iloc[0,1]*0.0001
123
124 #Make latex table
125 print(summary_stats.to_latex(index=False))
126
127 # Define a function to calculate the risk aversion coefficient
128 def risk_aversion_etd(Er_Mkt, rf, vol_mkt, vol_cons, rho):
129     gamma = (Er_Mkt - rf + vol_mkt**2/2)/(vol_cons* vol_mkt* rho)
130     return gamma
131 merged_data[['Consumption','Mkt']].corr().iloc[0,1]
132 x_rho = np.linspace(0.20,1, 100)
133 y_rra = risk_aversion_etd(summary_stats.loc[0,'average market excess return'],
134 summary_stats.loc[0,'average risk-free rate'],summary_stats.loc[0,'volatility of

```

```

129     'market excess return'],summary_stats.loc[0,'volatility of consumption growth'],x_rho
130     )
131 # Make the plot of RRA(rho)
132 plt.plot(x_rho, y_rra,linewidth=3)
133 plt.xlabel('Rho')
134 plt.ylabel('Relative risk aversion')
135 plt.savefig("p3q1")
136
137 # Define a function to calculate the subjective discount factor
138 def beta_cal(rf, gamma, ave_c, vol_c):
139     beta = np.exp(-rf+gamma*ave_c-gamma**2*vol_c**2/2)
140     return beta
141 rf = merged_data['RF_adj'].mean()*0.01
142 ave_c = Consumption_data['Consumption'].mean()*0.01
143 vol_c = Consumption_data['Consumption'].std()*0.01
144 # Make the plot of gamma(RRA)
145 x_gamma= np.linspace(60,300, 300)
146 y_Beta = beta_cal(rf, x_gamma, ave_c, vol_c)
147 plt.plot(x_gamma, y_Beta,linewidth=3)
148 plt.xlabel('Relative risk aversion')
149 plt.ylabel('Beta')
150 plt.savefig("p3q2")

```