Empirical Asset Pricing A HW1

Xinyu Liu

January 11, 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.

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,Δc)
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

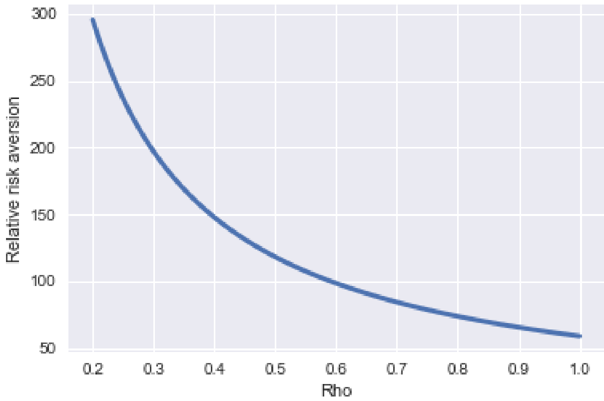
Suppose the assumptions discussed in class holds, I can infer the relative risk aversion rate (γ) using the following relationship:

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

which gives that $\gamma =$ (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.

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



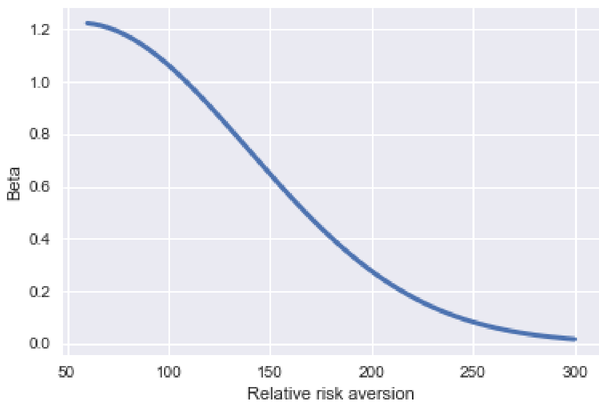
3. Risk-free Rate Puzzle

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 Compbell 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.

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

Appendix

```
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 datapungibea as dpb

###########################
# Fama French Data 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
###########################
#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
#df_me_op_factor = [dataset for dataset in datasets if 'ME' in dataset and 'OP' in dataset and '2x3' in dataset]
#print(df_me_op_factor)

#It is important to check the description of the dataset we access by using the following codes
Datatoread='F-F_Research_Data_Factors'
sdate='1948-01-01'
edate='2017-12-31'
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[1].copy()
dfFactor.reset_index(inplace=True)
# I check the scale of the data by printing out the head:
dfFactor.head()

#Date format adjustment
dfFactor['Date']=dfFactor['Date'].dt.year
dfFactor = dfFactor.set_index(['Date'])
###########################
# Consumption & Inflation Data Grabber
###########################
# https://pypi.org/project/datapungibea/
# Connect to Bureau of Economic Analysis (BEA) API
BEA_data = dpb.data('FDA2D756-CC0A-4AAA-A1D5-980FA23F31BB') #or data = dpb.data("API Key")
NIPA_cons=BEA_data.NIPA('T20302')
#Download annual consumption data on nondurable goods from Table 2.3.2.
#on   Contributions  to Percent Change in Real Personal Consumption Expenditures by Major Type of Product
NIPA_cons.reset_index(inplace=True)
Consumption_data=NIPA_cons[NIPA_cons['LineDescription']=='-Nondurable goods']
Consumption_data = Consumption_data.T.iloc[4:,:]
Consumption_data.columns=['Consumption']

NIPA_inflation=BEA_data.NIPA('T10107')
```

```
NIPA_inflation.reset_index(inplace=True)
Inflation_data=NIPA_inflation[NIPA_inflation['LineDescription']=='--Nondurable goods']
Inflation_data = Inflation_data.T.iloc[4:,:]
Inflation_data.columns=['Inflation']

NIPA_data = pd.merge(Consumption_data,Inflation_data, how='inner', left_index= True,
    right_index=True)
NIPA_data=NIPA_data.astype(float)
NIPA_data.index=pd.to_datetime(NIPA_data.index)
NIPA_data=NIPA_data.resample('Y').mean()
year = NIPA_data.reset_index()['index'].dt.year
NIPA_data.loc[:,'Date']=year.values
NIPA_data = NIPA_data[(NIPA_data['Date']>=1948) & (NIPA_data['Date']<=2017)]
NIPA_data = NIPA_data.set_index('Date')

# Merge the data
merged_data = dfFactor.merge(NIPA_data,how='left', left_index=True,right_index=True)
merged_data = merged_data.reset_index()
merged_data['Date'] = pd.to_datetime(merged_data['Date'],format='%Y')
merged_data = merged_data.set_index('Date')
merged_data['Mkt'] = merged_data['Mkt-RF'] + merged_data['RF']
merged_data['RF_adj'] = merged_data['RF'] - merged_data['Inflation']
# Drop irrelavent columns
merged_data_plot = merged_data.drop(columns=['SMB','HML','RF','Mkt'])

###########################
#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(merged_data_plot.columns)
fig, axes = plt.subplots(n,1,figsize=(12,8),sharex=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(merged_data_plot.columns):
    ax = axes[k]
    ax.plot(merged_data_plot.index,merged_data_plot[[factortitle]])
    ax.xaxis.set_major_formatter(years_fmt)
    ax.set_ylabel(factortitle)
plt.savefig("Time series")
plt.show()

#I take log transformation of the returns to fit the setting of theory
merged_data = np.log(merged_data/100+1)*100
#Calculate summary statistics
summary_stats = pd.DataFrame()
summary_stats.loc[0,'average market excess return'] = merged_data['Mkt-RF'].mean()*0.01
summary_stats.loc[0,'average risk-free rate'] = (merged_data['RF'].mean()-merged_data['
    Inflation'].mean())*0.01
summary_stats.loc[0,'volatility of market return'] = merged_data['Mkt'].std()*0.01
summary_stats.loc[0,'volatility of market excess return'] = merged_data['Mkt-RF'].std()
    *0.01
summary_stats.loc[0,'volatility of consumption growth'] = merged_data['Consumption'].std
    ()*0.01
summary_stats.loc[0,'covariance of market and consumption'] = merged_data[['Consumption'
    ,'Mkt']].cov().iloc[0,1]*0.0001

#Make latex table
print(summary_stats.to_latex(index=False))

# Define a funtion to calculate the risk aversion coefficient
def risk_aversion_etd(Er_Mkt, rf, vol_mkt, vol_cons, rho):
    gamma = (Er_Mkt - rf + vol_mkt**2/2)/(vol_cons* vol_mkt* rho)
    return gamma
merged_data[['Consumption','Mkt']].corr().iloc[0,1]
x_rho = np.linspace(0.20,1, 100)
y_rra = risk_aversion_etd(summary_stats.loc[0,'average market excess return'],
    summary_stats.loc[0,'average risk-free rate'],summary_stats.loc[0,'volatility of
```

```
    market excess return'],summary_stats.loc[0,'volatility of consumption growth'],x_rho
    )

# Make the plot of RRA(rho)
plt.plot(x_rho, y_rra,linewidth=3)
plt.xlabel('Rho')
plt.ylabel('Relative risk aversion')
plt.savefig("p3q1")

# Define a funtion to calculate the subjective discount factor
def beta_cal(rf, gamma, ave_c, vol_c):
    beta = np.exp(-rf+gamma*ave_c-gamma**2*vol_c**2/2)
    return beta
rf = merged_data['RF_adj'].mean()*0.01
ave_c = Consumption_data['Consumption'].mean()*0.01
vol_c = Consumption_data['Consumption'].std()*0.01
# Make the plot of gamma(RRA)
x_gamma= np.linspace(60,300, 300)
y_Beta = beta_cal(rf, x_gamma, ave_c, vol_c)
plt.plot(x_gamma, y_Beta,linewidth=3)
plt.xlabel('Relative risk aversion')
plt.ylabel('Beta')
plt.savefig("p3q2")
```