# QF600 Homework 6

## Behavior Finance

### Wang Hairuo | October 31, 2022

Consider a Barberis, Huang and Santos (2001) economy with the following parameter choices for the investor's utility function:

$$\delta = 0.99,\ \gamma = 1,\ \lambda = 2$$

Consumption growth has lognormal distribution:

$$\ln\tilde{g} = 0.02 + 0.02\tilde{\epsilon}$$

where $\epsilon$ is a standard normal random variable. Simulate the distribution for consumption growth with (at least) $10^4$ random draw for $\epsilon$.

The risk-free rate is constant at 1.0303 per year. Let x be one plus the dividend yield for the market portfolio:

$$x = (1 + \frac{P}{D})\frac{D}{P} = 1 + \frac{D}{P}$$

and define the error term:

$$e(x) = 0.99b_0E[\hat{\nu}(x\tilde{g})] + 0.99x - 1$$

where utility from recent financial gain or loss is given by:

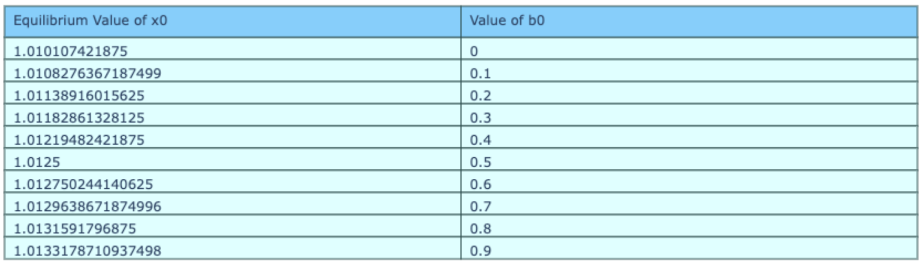
$$\hat{\nu}(R) = R - 1.0303 \quad R \geq 1.0303$$
$$\hat{\nu}(R) = 2(R - 1.0303) \quad R < 1.0303$$

Calculate the equilibrium values of x for $b_0$ in the range from 0 to 10, increments of 0.1 (or less), using bisection search:

1. Set $x_- = 1$ and $x_+ = 1.1$ and use the simulated distribution of consumption growth to confirm that $e(x_-) < 0$ and $e(x_+) > 0$ —> equilibrium value of x must lie between $x_-$ and $x_+$.
2. Set $x_0 = 0.5 * (x_- + x_+)$, and use the simulated distribution of consumption growth to calculate $e(x_0)$.
3. If $|e(x_0)| < 10^{-5}$, then $x_0$ is (close enough to) the equilibrium value of x.
4. Otherwise, if $e(x_0) < 0$, then the equilibrium value of x lies between $x_0$ and $x_+$, so repeat the procedure with $x_- = x_0$.
5. Otherwise, if $e(x_0) > 0$, then the equilibrium value of x lies between $x_-$ and $x_0$, so repeat the procedure $x_+ = x_0$.

As we perform the bisector search to compute the equilibrium value of $x_0$ (which is one plus dividend yield) and we computed 101 number of equilibrium value of $x_0$, where $x_0$ is one plus the price-dividend ratio for the market portfolio.

Table 1. The First Ten Equilibrium Value of x0 with Corresponding Value of b0



| Equilibrium Value of x0 | Value of b0 |
|---|---|
| 1.010107421875 | 0 |
| 1.0108276367187499 | 0.1 |
| 1.01138916015625 | 0.2 |
| 1.01182861328125 | 0.3 |
| 1.01219482421875 | 0.4 |
| 1.0125 | 0.5 |
| 1.012750244140625 | 0.6 |
| 1.0129638671874996 | 0.7 |
| 1.0131591796875 | 0.8 |
| 1.0133178710937498 | 0.9 |

**Table 1** shows the first ten equilibrium value of $x_0$ with corresponding value of $b_0$ which generate by bisector search.

As well, we also checked the assumption for the bisector search, whether the error term for lower bound of x is all negative and the error term for upper bound of x is all positive or not.

We use the code below for check if the lower bound $x_- = 1$ have all negative error term or not.

```
x_less = 1
e_xless = []
for i in b0:
    nu = np.where(x_less * g_list > r_f,
                 x_less * g_list - r_f,
                 lambda_ * (x_less * g_list - r_f))
    e_x = delta * i * nu.mean() + delta * x_less - 1
    e_xless.append(e_x)

all(i < 0 for i in e_xless)
```

and the output for the code is `True` and that indicate the error term for $x_- = 1$ with each value of $b_0$ is all negative, and that does not violate the assumption for bisector search.

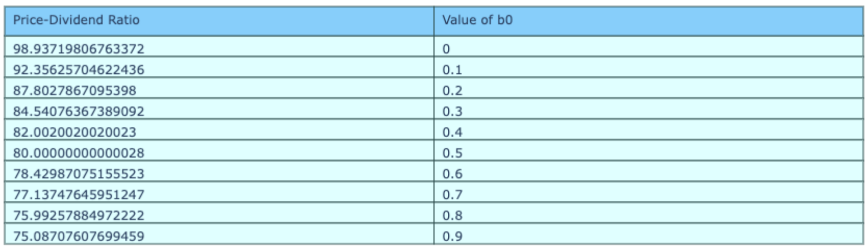
Furthermore, we use the code below to check if the upper bound $x_+ = 1.1$ have all positive error or not.

```
x_more = 1.1
e_xmore = []

for i in b0:
    nu = np.where(x_more * g_list > r_f,
                 x_more * g_list - r_f,
                 lambda_ * (x_more * g_list - r_f))
    e_x = delta * i * nu.mean() + delta * x_more - 1
    e_xmore.append(e_x)

all(i > 0 for i in e_xmore)
```

and the output for the code is `True` and that indicate the error term for $x_+ = 1.1$ with each value of $b_0$ is all positive, and that does not violate the assumption for bisector search as well. Then we could conclude that the equilibrium value of x lie between $x_- \text{ and } x_+$.

## Question 1: Use the equilibrium value of x to calculate the price-dividend ratio for the market portfolio: $\frac{P}{D} = \frac{1}{x-1}$ and plot the price-dividend ratio (on the vertical axis) vs $b_0$

Table 2. The First Ten Price-Dividend Ratio with Corresponding Value of b0



| Price-Dividend Ratio | Value of b0 |
|---|---|
| 98.93719806763372 | 0 |
| 92.35625704622436 | 0.1 |
| 87.8027867095398 | 0.2 |
| 84.54076367389092 | 0.3 |
| 82.0020020020023 | 0.4 |
| 80.00000000000028 | 0.5 |
| 78.42987075155523 | 0.6 |
| 77.13747645951247 | 0.7 |
| 75.99257884972222 | 0.8 |
| 75.08707607699459 | 0.9 |

**Table 2** shows the first ten price-dividend ratio with corresponding value of $b_0$.

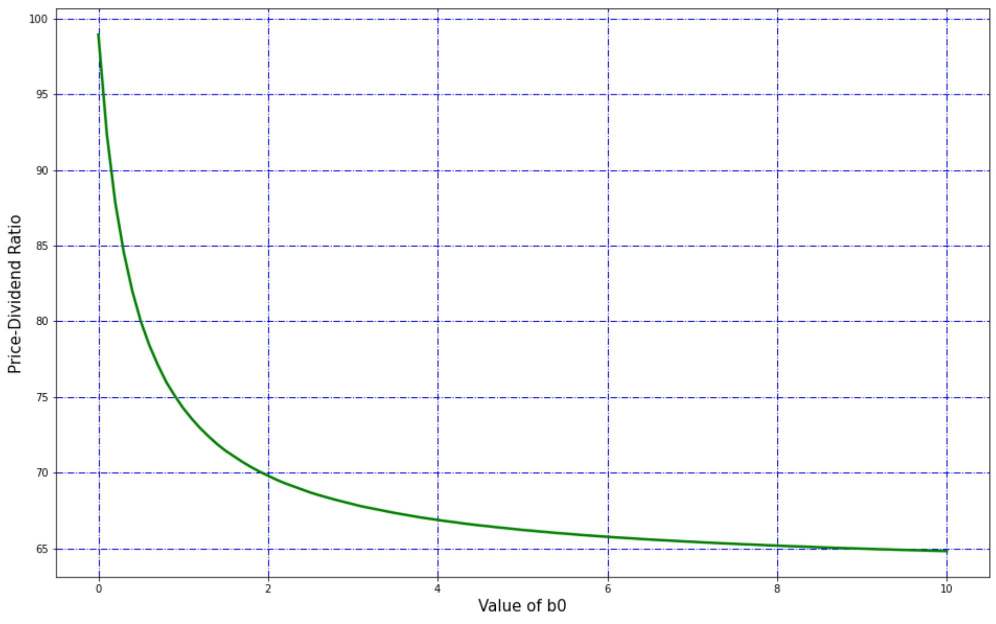
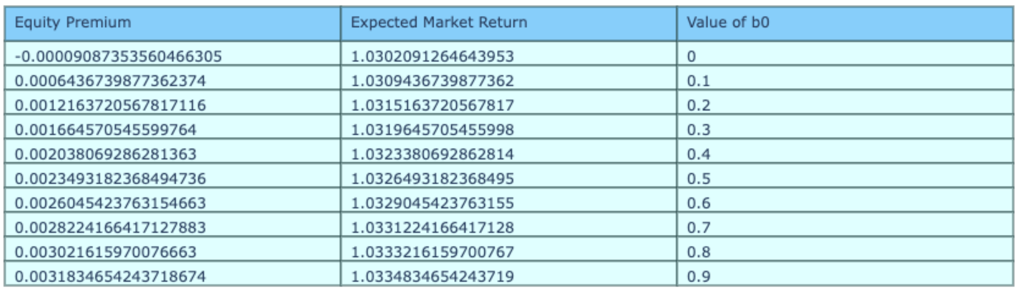


*Figure 1: The Plot for Price-Dividend Ratio with corresponding value of b0*

**Figure 1** shows the plot for price-dividend ratio with corresponding value of b0 and we could see there is a **downward trend** for the price-dividend ratio with increasing amount of emphasis investor puts on utility from financial gain or loss.

## Question 2: Use the equilibrium value of x to calculate the expected market return: $E[R_m] = E[x\tilde{g}]$ and plot the equity premium (on the vertical axis) vs $b_0$

Table 3. The First Ten Equity Premium and Expected Market Return with Corresponding Value of b0



| Equity Premium | Expected Market Return | Value of b0 |
|---|---|---|
| -0.00009087353560466305 | 1.0302091264643953 | 0 |
| 0.0006436739877362374 | 1.0309436739877362 | 0.1 |
| 0.0012163720567817116 | 1.0315163720567817 | 0.2 |
| 0.001664570545599764 | 1.0319645705455998 | 0.3 |
| 0.002038069286281363 | 1.0323380692862814 | 0.4 |
| 0.0023493182368494736 | 1.0326493182368495 | 0.5 |
| 0.0026045423763154663 | 1.0329045423763155 | 0.6 |
| 0.0028224166417127883 | 1.0331224166417128 | 0.7 |
| 0.003021615970076663 | 1.0333216159700767 | 0.8 |
| 0.0031834654243718674 | 1.0334834654243719 | 0.9 |

**Table 3** shows the first ten equity premium, expected market return with corresponding value of $b_0$.

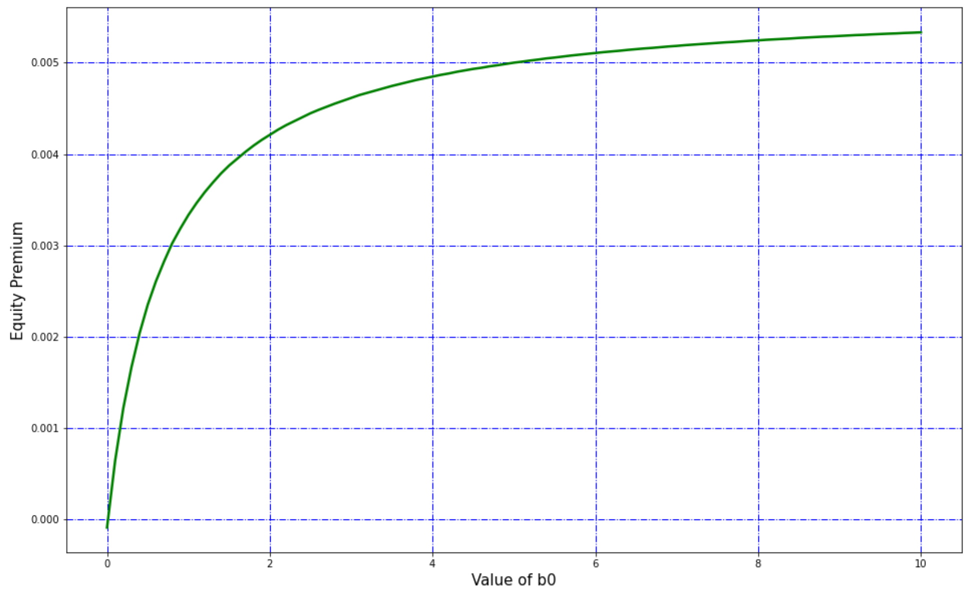


*Figure 2: The Plot for Equity Premium with corresponding value of b0*

**Figure 2** shows the plot for equity premium with the corresponding value of $b_0$, and we could see there is an upward trend for the equity premium with increasing amount of emphasis investor puts on utility from financial gain or loss. That indicates as the investor with higher level of loss aversion, then the investor would like to have higher equity premium for the investment.

## Question 3: Briefly explain the economic significance of the investor's utility function for the recent financial gain and loss [$\nu(R)$], as well as the economic significance of $b_0$ and $\lambda$

### Utility Function for Recent Financial Gain and Loss $[\nu(R)]$

$\nu(R)$ is the utility function for recent financial gain and loss and the formula is below:

$$\nu(R) = R - R_f \quad \text{if} \;\; R \geq R_f$$
$$\nu(R) = \lambda(R - R_f) \quad \text{if} \;\; R < R_f$$

utility from financial gain or loss is under **prospect theory**, which illustrate the investors value gains and losses differently and investors exhibit **loss aversion**. That indicates the investors would be **more sensitive to loss** than the same magnitude gain. As in the formula, if the return is greater than risk-free rate and there is financial gain in the investment. Then the utility from the financial gain would depends on the return and risk-free rate. However, if the return is less than the risk-free rate and there is the financial loss in the investment. Then the utility for the financial loss to the investors would be multiple times of the utility for the financial gain to the investor. The utility function for the recent financial gain and loss demonstrates that the **loss aversion** make investor more sensitive to outright financial loss or shortfall in financial gain. As well, loss aversion would also impact the magnitude of $\lambda$ in the utility function, we would discuss in the next section.

### $\lambda$

$\lambda$ is the **magnitude of the utility for the financial loss to the investors, relative to utility for the financial gain to investor**. As we discussed in the last section, the prospect theory illustrate the investors exhibit loss aversion and that indicate the investor would be more sensitive to loss than the same magnitude gain. Then the utility for the financial loss to investors would be more compare with the utility to the financial gain. Therefore, we could conclude that the $\lambda$ represents the **degree of loss aversion** for the investor, and the value of $\lambda$ **supposed to be greater than one**.

### $b_0$ - Amount of Emphasis Investor puts on Utility from Financial Gain or Loss

The value of $b_0$ determine the amount of emphasis investor puts on utility from financial gain or loss. As the value of $b_0$ increased, that indicates there would be more emotional impacts for the investors for their financial gain or loss. Thus, the value of $b_0$ must be a non-negative variable. As shown in the result for question 2, if the investors have higher amount of emphasis they puts on utility from financial gain or loss, then they would be more loss aversion and the investor is likely to have higher equity premium. Therefore, the value of $b_0$, which is known as amount of emphasis investor puts on utility from financial gain or loss, indicate the degree of investors care about the utility for the financial gain or loss.

## Appendix

### Set up Codes and Equilibrium Value of x Codes

```
# import the necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%pip install dataframe_image
import dataframe_image as dfi
import math
import plotly.graph_objects as go
```

```
# set up the parameter as given above
delta = 0.99
gamma = 1
lambda_ = 2
# risk-free rate is 1.0303
r_f = 1.0303
# set up the different amount of emphasis investors put on utility from financial gain or loss
b0 = np.linspace(0, 10, 101)
# number of simulation
n_sim = 10 ** 4
```

```
# set seed in order to get the same simulation everytime
np.random.seed(600)
```

```
# create a list to save the consumption growth rate
g_list = []
# compute the consumption growth rate with simulated error
for i in range(n_sim):
    epsilon = np.random.standard_normal()
    g = np.exp(0.02 + 0.02 * epsilon)
    g_list.append(g)
g_list = np.array(g_list)
```

```
# check if the error term for lower bound for x is all negative or not
x_less = 1
e_xless = []
for i in b0:
    nu = np.where(x_less * g_list > r_f,
                  x_less * g_list - r_f,
                  lambda_ * (x_less * g_list - r_f))
    e_x = delta * i * nu.mean() + delta * x_less - 1
    e_xless.append(e_x)

all(i < 0 for i in e_xless)
```

```
# check if the error term for upper bound for x is all positive or not
x_more = 1.1
e_xmore = []

for i in b0:
    nu = np.where(x_more * g_list > r_f,
                  x_more * g_list - r_f,
                  lambda_ * (x_more * g_list - r_f))
    e_x = delta * i * nu.mean() + delta * x_more - 1
    e_xmore.append(e_x)

all(i > 0 for i in e_xmore)
```

```
# bisector research
# create a new list to save the equilibrum value of x
x_list = []
# for each value in b0
for i in b0:
    # lower bound for generate x is 1
    x_less = 1
    # upper bound for generate x is 1.1
    x_more = 1.1
    # compute the initial value of x
    x0 = (x_less + x_more) / 2
    # compute the value of utility for financial gain or loss with different conditions
    nu = np.where(x0 * g_list > r_f,
                  x0 * g_list - r_f,
                  # investors more sensitive to the shortfall in financial gain or outright financial loss
                  lambda_ * (x0 * g_list - r_f))
    # compute the error term for various b0 and x0
    e_x0 = delta * i * np.mean(nu) + delta * x0 - 1
    # for the case error term is not extremly small
    while np.absolute(e_x0) > 10 ** -5:
        # if the error term is negative
        if e_x0 < 0:
            # then the lower bound x_ would be x0 and repeat the procedure
            x_less = x0
            x0 = (x_less + x_more) / 2
            nu = np.where(x0 * g_list > r_f,
                          x0 * g_list - r_f,
                          lambda_ * (x0 * g_list - r_f))
            e_x0 = delta * i * np.mean(nu) + delta * x0 - 1
        # if the error term is positive
        elif e_x0 > 0:
            # the the upper bound x+ would be x0 and repeat the procedure
            x_more = x0
            x0 = (x_less + x_more) / 2
            nu = np.where(x0 * g_list > r_f,
                          x0 * g_list - r_f,
                          lambda_ * (x0 * g_list - r_f))
            e_x0 = delta * i * np.mean(nu) + delta * x0 - 1
    # for the case of error term is extremely small, then euilibrium value of x would be x0
    x_list.append(x0)
```

```
# Create a Table to show the first ten equilibrium value of x0 with corresponding value of b0
fig = go.Figure(data = [go.Table(
    header = dict(values = ['Equilibrium Value of x0', 'Value of b0'],
                 line_color = 'darkslategrey',
                 fill_color = 'lightskyblue',
                 align='left'),
cells = dict(values = [[x_list[0], x_list[1], x_list[2], x_list[3], x_list[4], x_list[5], x_list[6], x_list[7], x_list[8], x_list[9]],
                       [b0[0], b0[1], b0[2], round(b0[3], 1), b0[4], b0[5], round(b0[6], 1), round(b0[7], 1), b0[8], b0[9]]],
            line_color = 'darkslategrey',
            fill_color = 'lightcyan',
            align='left'))])
# set up the title for the table
layout = go.Layout(title = 'Table 1. The First Ten Equilibrium Value of x0 with Corresponding Value of b0')

fig.update_layout(layout)
# show the table
fig.show()
```

## Question 1 - Codes

```
# create a new list to save the value of price-dividend ratio
ratio_list = []
# compute the price-dividend ratio with each equilibrium value of x0
for i in range(len(x_list)):
    ratio = 1 / (x_list[i] - 1)
    ratio_list.append(ratio)
```

```
# create a table to show the first ten price-dividend ratio with corresponding value of b0
fig = go.Figure(data = [go.Table(
    header = dict(values = ['Price-Dividend Ratio', 'Value of b0'],
                 line_color = 'darkslategrey',
                 fill_color = 'lightskyblue',
                 align='left'),
cells = dict(values = [[ratio_list[0], ratio_list[1], ratio_list[2], ratio_list[3], ratio_list[4], ratio_list[5], ratio_list[6], ratio
                       [b0[0], b0[1], b0[2], round(b0[3], 1), b0[4], b0[5], round(b0[6], 1), round(b0[7], 1), b0[8], b0[9]]],
            line_color = 'darkslategrey',
            fill_color = 'lightcyan',
            align='left'))])
# set up the title for the table
layout = go.Layout(title = 'Table 2. The First Ten Price-Dividend Ratio with Corresponding Value of b0')

fig.update_layout(layout)
# show the table
fig.show()
```

```
# set up the figure size
plt.figure(figsize = (16, 10))
# b0 on the horizontal axis
x = b0
# price-dividend ratio on the vertical axis
y = ratio_list
# plot the price-dividend ratio with corresponding value of b0
plt.plot(x, y, color = 'green', linewidth = 2.5)
# add grid on the plot
plt.grid(color='blue', linewidth = 1, linestyle = '-.')
# set up the title, the position of title and font style and font size for the title
plt.title(label = "Figure 1: The Plot for Price-Dividend Ratio with corresponding value of b0",
         y = -0.15,
          fontstyle = 'italic',
         fontsize = 20)
# set up the y label and font size
plt.ylabel('Price-Dividend Ratio', fontsize = 15)
# set up the x label and font size
plt.xlabel('Value of b0', fontsize = 15)
# export the figure
plt.savefig("Price-Dividend Ratio with value of b0.png")
# show the plot
plt.show()
```

## Question 2 - Codes

```
# create a new list for saving the value of market return
rm_list = []
# for each equilibrium value of x to calculate the market return
for x in x_list:
    rm = (x * g_list).mean()
    rm_list.append(rm)
rm_list = np.array(rm_list)
# calculate the equity premium for the market portfolio
equity_premium = rm_list - r_f
```

```
# create a table to show the equity premiun, market return with the corresponding value of b0
fig = go.Figure(data = [go.Table(
    header = dict(values = ['Equity Premium', 'Expected Market Return', 'Value of b0'],
                 line_color = 'darkslategrey',
                 fill_color = 'lightskyblue',
                 align='left'),
cells = dict(values = [[equity_premium[0], equity_premium[1], equity_premium[2], equity_premium[3], equity_premium[4], equity_premium[
                       [rm_list[0], rm_list[1], rm_list[2], rm_list[3], rm_list[4], rm_list[5], rm_list[6], rm_list[7], rm_list[8], rm
                       [b0[0], b0[1], b0[2], round(b0[3], 1), b0[4], b0[5], round(b0[6], 1), round(b0[7], 1), b0[8], b0[9]]],
            line_color = 'darkslategrey',
            fill_color = 'lightcyan',
            align='left'))])
# set up the title for the table
layout = go.Layout(title = 'Table 3. The First Ten Equity Premium and Expected Market Return with Corresponding Value of b0')

fig.update_layout(layout)
# show the table
fig.show()
```

```
# set up the figure size for the figure 2
plt.figure(figsize = (16, 10))
# value of b0 on horizontal axis
x = b0
# equity premiun on vertical axis
y = equity_premium
# plot the equity premium and the value of b0 with green line and adjust the line width
plt.plot(x, y, color = 'green', linewidth = 2.5)
# add grid on the plot
plt.grid(color='blue', linewidth = 1, linestyle = '-.')
# add title for the figure and adjust the position, font style and font size for the title
plt.title(label = "Figure 2: The Plot for Equity Premium with corresponding value of b0",
         y = -0.15,
          fontstyle = 'italic',
         fontsize = 20)
# add the y label for the figure
plt.ylabel('Equity Premium', fontsize = 15)
# add the x label for the figure
plt.xlabel('Value of b0', fontsize = 15)

# export the figure
plt.savefig("Equity Premium with value of b0.png")
```

```
# show the figure
plt.show()
```