Yifu-He-Homework5

April 19, 2020

1 Yifu He Homework 5 190003956

In [1]:
```
import pandas as pd
import numpy as np
import datetime
from sklearn.linear_model import LinearRegression
import scipy.stats
```

In [2]:
```
## Question 1


# the PERMNO code of 10 stocks: 84788 12060 22111 59328 14008 12490 19561 18411 73139
# the TICKER of 10 stocks: AMZN GE JNJ INTC AMGN IBM BA SO SYK ITW
#----------------------------------------
# Œ AMAZON COM INC 84788
# Œ GENERAL ELECTRIC CO 12060
# Œ JOHNSON & JOHNSON 22111
# Œ INTEL CORP 59328
# Œ AMGEN INC 14008
# Œ INTERNATIONAL BUSINESS MACHS CO 12490
# Œ BOEING CO 19561
# Œ SOUTHERN CO 18411
# Œ STRYKER CORP 73139
# Œ ILLINOIS TOOL WORKS INC 56573
#----------------------------------------

# add 10 stocks
df = pd.read_csv("./assets.csv")
colname = ["Date","AMZN","GE","JNJ","INTC","AMGN","IBM","BA","SO","SYK","ITW"]
Date= pd.to_datetime(df["date"].astype(str))[:240]
ret = df["RET"].values.reshape(10,240)
data = [Date]
for i in ret:
    data.append(i)
data = np.array(data).transpose()
new_df = pd.DataFrame(data,columns=colname)
```

```
# add sp500
sp500 = pd.read_csv("./sp500.csv")
new_df["sp500"] = sp500["vwretd"]

# add risk free return
rf = pd.read_csv("./rf.csv")
new_df["T90"] = rf["t90ret"]

# add market premium: Rm - Rf
new_df["Rm-Rf"] = new_df["sp500"] - new_df["T90"]

# add the Ri-Rf for each stocks
for i in range(1,11):
    ticker = new_df.columns[i]
    new_df[ticker+"-rf"] = new_df[ticker] - new_df["T90"]
```

In [3]:
```
print(f"columns: {new_df.columns}")
print(f"dates: {new_df.Date}")
new_df
```

```
columns: Index(['Date', 'AMZN', 'GE', 'JNJ', 'INTC', 'AMGN', 'IBM', 'BA', 'SO', 'SYK',
       'ITW', 'sp500', 'T90', 'Rm-Rf', 'AMZN-rf', 'GE-rf', 'JNJ-rf', 'INTC-rf',
       'AMGN-rf', 'IBM-rf', 'BA-rf', 'SO-rf', 'SYK-rf', 'ITW-rf'],
      dtype='object')
dates: 0     1999-01-29
1      1999-02-26
2      1999-03-31
3      1999-04-30
4      1999-05-28
          ...
235    2018-08-31
236    2018-09-28
237    2018-10-31
238    2018-11-30
239    2018-12-31
Name: Date, Length: 240, dtype: datetime64[ns]
```

Out[3]:

	Date	AMZN	GE	JNJ	INTC	AMGN	IBM	\
0	1999-01-29	0.028186	-0.006102	0.222355	-0.061591	0.063218	0.014903	
1	1999-02-26	-0.043504	-0.072469	-0.022983	-0.069606	0.031063	0.005874	
2	1999-03-31	0.106293	0.044183	0.199199	-0.069825	-0.045614	0.095168	
3	1999-04-30	-0.047458	0.180183	-0.179466	0.175228	0.194853	0.042781	
4	1999-05-28	-0.034994	0.1102	0.029502	0.048499	0.038831	-0.047128	
..	...	...	...	...	...	...	...	
235	2018-08-31	-0.050624	0.021528	0.023302	-0.086831	-0.033118	0.023166	
236	2018-09-28	-0.118238	0.032291	0.037436	-0.004111	0.084921	0.025837	
237	2018-10-31	-0.105403	-0.236625	-0.06995	0.032798	-0.045819	0.013172	

```
238 2018-11-30 -0.257426  0.090184  0.087038  0.064402 -0.018007   0.05579
239 2018-12-31  0.010667 -0.085298  -0.06521 -0.072047 -0.069962 -0.121511

           BA        SO       SYK  ...   AMZN-rf     GE-rf    JNJ-rf  \
0    0.039871  0.188719 -0.157775  ...  0.024614 -0.009674  0.218783
1    0.139896 -0.148718  0.016173  ... -0.046546 -0.075511 -0.026025
2   -0.097818 -0.008859  0.070292  ...  0.101962  0.039852  0.194868
3    0.244444  0.029443  0.213135  ... -0.051015  0.176626 -0.183023
4   -0.003247 -0.115955 -0.027579  ... -0.038498  0.106696  0.025998
..        ...       ...       ...  ...       ...       ...       ...
235 -0.031047  0.013098  0.037856  ... -0.052299  0.019853  0.021627
236  0.023329 -0.023539  0.051467  ... -0.119762  0.030767  0.035912
237 -0.096018  -0.00867  -0.08701  ... -0.107334 -0.238556 -0.071881
238   0.08999  0.058234  0.081618  ... -0.259326  0.088284  0.085138
239 -0.081697 -0.048266  -0.10367  ...  0.008732 -0.087233 -0.067145

      INTC-rf   AMGN-rf    IBM-rf     BA-rf     SO-rf    SYK-rf    ITW-rf
0   -0.065163  0.059646  0.011331  0.036299  0.185147 -0.161347  0.088451
1   -0.072648  0.028021  0.002832  0.136854  -0.15176  0.013131  0.092629
2   -0.074156 -0.049945  0.090837 -0.102149  -0.01319  0.065961  0.339571
3    0.171671  0.191296  0.039224  0.240887  0.025886  0.209578 -0.004283
4    0.044995  0.035327 -0.050632 -0.006751 -0.119459 -0.031083 -0.313348
..        ...       ...       ...       ...       ...       ...       ...
235 -0.088506 -0.034793  0.021491 -0.032722  0.011423  0.036181   0.13069
236 -0.005635  0.083397  0.024313  0.021805 -0.025063  0.049943 -0.006348
237  0.030867  -0.04775  0.011241 -0.097949 -0.010601 -0.088941 -0.204123
238  0.062502 -0.019907   0.05389   0.08809  0.056334  0.079718  0.055772
239 -0.073982 -0.071897 -0.123446 -0.083632 -0.050201 -0.105605 -0.113285

[240 rows x 24 columns]
```

In [21]:
```python
# Run the regression to get the betas
res1 = []
for j in range(10):
    beta = []
    for i in range(180):
        x = new_df.iloc[i:i+60,13].values.reshape(-1,1)
        y = new_df.iloc[i:i+60,j+14]
        reg = LinearRegression().fit(x, y)
        beta.append(reg.coef_[0])
    res1.append(beta)
res1 = np.array(res1).transpose()

colname_beta = ["beta_{}".format(i) for i in range(1,11)]
res1_df = pd.DataFrame(res1,columns = colname_beta)
print(f"res1_df.shape: {res1_df.shape}")
res1_df
```

```
res1_df.shape: (180, 10)
```

```
Out[21]:        beta_1    beta_2    beta_3    beta_4    beta_5    beta_6    beta_7  \
         0    1.083460  1.430386  0.578084 -0.231656  0.714447  0.245545  0.824172
         1    1.098231  1.456510  0.529167 -0.213752  0.703390  0.249829  0.812186
         2    1.090485  1.443052  0.519594 -0.234636  0.716243  0.251815  0.848716
         3    1.079166  1.450938  0.486086 -0.212411  0.742590  0.237526  0.883572
         4    1.102955  1.421619  0.544574 -0.249381  0.697160  0.223937  0.821127
         ..        ...       ...       ...       ...       ...       ...       ...
         175  1.004140  0.894505  1.356805  0.046438  1.458936  0.738456  1.254776
         176  0.969592  0.874112  1.393517 -0.054164  1.443147  0.693862  1.247506
         177  0.974529  0.868747  1.394955 -0.045464  1.395555  0.700352  1.228664
         178  0.954736  1.185049  1.370203 -0.082051  1.299232  0.582922  1.263484
         179  0.902362  1.206669  1.406452 -0.066525  1.296980  0.587033  1.288393

                beta_8    beta_9   beta_10
         0    2.074575  0.490468  2.232954
         1    2.034648  0.550433  2.224084
         2    2.004106  0.548687  2.237397
         3    2.033682  0.544462  2.170875
         4    2.051933  0.490715  2.191785
         ..        ...       ...       ...
         175  1.004582  0.628478  1.621038
         176  0.970707  0.594174  1.617104
         177  0.973568  0.594937  1.601531
         178  0.876696  0.647445  1.718806
         179  0.905558  0.667977  1.714043

         [180 rows x 10 columns]
```

```
In [22]: # add Ri - Rf as responsors
         for i in range(14,24):
             res1_df[new_df.columns[i]] = new_df.iloc[60:,14].reset_index().drop("index",axis=
         print(f"res1_df.columns:\n{res1_df.columns}")
         res1_df
```

```
res1_df.columns:
Index(['beta_1', 'beta_2', 'beta_3', 'beta_4', 'beta_5', 'beta_6', 'beta_7',
       'beta_8', 'beta_9', 'beta_10', 'AMZN-rf', 'GE-rf', 'JNJ-rf', 'INTC-rf',
       'AMGN-rf', 'IBM-rf', 'BA-rf', 'SO-rf', 'SYK-rf', 'ITW-rf'],
      dtype='object')
```

```
Out[22]:        beta_1    beta_2    beta_3    beta_4    beta_5    beta_6    beta_7  \
         0    1.083460  1.430386  0.578084 -0.231656  0.714447  0.245545  0.824172
         1    1.098231  1.456510  0.529167 -0.213752  0.703390  0.249829  0.812186
         2    1.090485  1.443052  0.519594 -0.234636  0.716243  0.251815  0.848716
         3    1.079166  1.450938  0.486086 -0.212411  0.742590  0.237526  0.883572
         4    1.102955  1.421619  0.544574 -0.249381  0.697160  0.223937  0.821127
```

```
..        ...       ...       ...       ...       ...       ...       ...
175  1.004140  0.894505  1.356805  0.046438  1.458936  0.738456  1.254776
176  0.969592  0.874112  1.393517 -0.054164  1.443147  0.693862  1.247506
177  0.974529  0.868747  1.394955 -0.045464  1.395555  0.700352  1.228664
178  0.954736  1.185049  1.370203 -0.082051  1.299232  0.582922  1.263484
179  0.902362  1.206669  1.406452 -0.066525  1.296980  0.587033  1.288393

       beta_8    beta_9   beta_10   AMZN-rf     GE-rf    JNJ-rf   INTC-rf  \
0    2.074575  0.490468  2.232954  0.084671  0.084671  0.084671  0.084671
1    2.034648  0.550433  2.224084 -0.027722 -0.027722 -0.027722 -0.027722
2    2.004106  0.548687  2.237397 -0.062419 -0.062419 -0.062419 -0.062419
3    2.033682  0.544462  2.170875 -0.019492 -0.019492 -0.019492 -0.019492
4    2.051933  0.490715  2.191785  0.038316  0.038316  0.038316  0.038316
..        ...       ...       ...       ...       ...       ...       ...
175  1.004582  0.628478  1.621038 -0.052299 -0.052299 -0.052299 -0.052299
176  0.970707  0.594174  1.617104 -0.119762 -0.119762 -0.119762 -0.119762
177  0.973568  0.594937  1.601531 -0.107334 -0.107334 -0.107334 -0.107334
178  0.876696  0.647445  1.718806 -0.259326 -0.259326 -0.259326 -0.259326
179  0.905558  0.667977  1.714043  0.008732  0.008732  0.008732  0.008732

      AMGN-rf    IBM-rf     BA-rf     SO-rf    SYK-rf    ITW-rf
0    0.084671  0.084671  0.084671  0.084671  0.084671  0.084671
1   -0.027722 -0.027722 -0.027722 -0.027722 -0.027722 -0.027722
2   -0.062419 -0.062419 -0.062419 -0.062419 -0.062419 -0.062419
3   -0.019492 -0.019492 -0.019492 -0.019492 -0.019492 -0.019492
4    0.038316  0.038316  0.038316  0.038316  0.038316  0.038316
..        ...       ...       ...       ...       ...       ...
175 -0.052299 -0.052299 -0.052299 -0.052299 -0.052299 -0.052299
176 -0.119762 -0.119762 -0.119762 -0.119762 -0.119762 -0.119762
177 -0.107334 -0.107334 -0.107334 -0.107334 -0.107334 -0.107334
178 -0.259326 -0.259326 -0.259326 -0.259326 -0.259326 -0.259326
179  0.008732  0.008732  0.008732  0.008732  0.008732  0.008732

[180 rows x 20 columns]
```

In [83]:
```python
gama_df = []
for i in range(180):
    x=res1_df.iloc[i,0:10].values.reshape(10,1)
    y=res1_df.iloc[i,10:]
    reg = LinearRegression().fit(x,y)
    gama_df.append([reg.intercept_, reg.coef_[0]])
gama_df = pd.DataFrame(gama_df,columns=["gama_0","gama_1"])
gama_df
```

Out[83]:
```
       gama_0        gama_1
0    0.084671 -6.592153e-35
1   -0.027722 -0.000000e+00
2   -0.062419 -0.000000e+00
```

```
3   -0.019492 -2.498988e-34
4    0.038316  1.080171e-33
..        ...           ...
175 -0.052299 -0.000000e+00
176 -0.119762 -1.201609e-32
177 -0.107334 -0.000000e+00
178 -0.259326  0.000000e+00
179  0.008732  0.000000e+00

[180 rows x 2 columns]
```

In [29]:
```
# Question 1 A. get the t-statistics for each paramters and determine whether it is s
degree_of_freedom = gama_df.shape[0]-1
print(f" Degree of Freedom is: {degree_of_freedom}")
gama_0_T_stat = (gama_df.gama_0.mean() - 0)/gama_df.gama_0.std()* np.sqrt(degree_of_fr
gama_1_T_stat = (gama_df.gama_1.mean() - 0)/gama_df.gama_1.std()* np.sqrt(degree_of_fr
print(f"t-statistic (179 degree of freedom) at 95% confidence interval: {scipy.stats.t
print(f"gama_0_T_stat: {gama_0_T_stat}")
print(f"gama_1_T_stat: {gama_1_T_stat}")
print("Our t-statistics of each stock are smaller than the criterion. Thus, both of th
```

```
 Degree of Freedom is: 179
t-statistic (179 degree of freedom) at 95% confidence interval: 1.653410800122353
gama_0_T_stat: -0.5406561239363877
gama_1_T_stat: 1.312268648423659
Our t-statistics of each stock are smaller than the criterion. Thus, both of the paramters are
```

In [30]:
```
# Question 1 B.
print(f"gamma_1_T_stat: {gama_1_T_stat}")
print("It is not statistical significant, thus We don't have a trade-off between marke
```

```
gamma_1_T_stat: 1.312268648423659
It is not statistical significant, thus We don't have a trade-off between market beta and expec
```

In [31]:
```
# Question 1 C.
empirical_risk_premium = new_df["Rm-Rf"].mean()
print(f"empirical_risk_premium: {empirical_risk_premium}")
gama_1_T_stat = (gama_df.gama_1.mean() - empirical_risk_premium)/gama_df.gama_1.std()*
print(f"gama_1_T_stat: {gama_1_T_stat}")
print("It is higher than - 1.65, thus, it is not significantly different empirical ris
```

```
empirical_risk_premium: 0.003959791666666666
gama_1_T_stat: -1.6475527617324288e+31
It is higher than - 1.65, thus, it is not significantly different empirical risk premium.
```

In [32]:
```
# Question 1 D.
print(f"gamma_0_T_stat: {gama_0_T_stat}")
print("It is not statistical significant, thus We don't have a mid-pricing.")
```

```
gamma_0_T_stat: -0.5406561239363877
It is not statistical significant, thus We don't have a mid-pricing.
```

In [55]:
```python
## Question 2

# Get annualy return: We assume that the previous return are log-return.
ret = new_df.iloc[:,1:13]
new_ret = []
for i in range(20):
    new_ret.append(ret.iloc[i*12:(i+1)*12-1,:].sum(axis=0))
colname =["AMZN","GE","JNJ","INTC","AMGN","IBM","BA","SO","SYK","ITW","sp500","T90"]
new_ret = pd.DataFrame(np.array(new_ret), columns =colname)

# Rm-Rf
new_ret["Rm-Rf"] = new_ret["sp500"] - new_ret["T90"]

# add the Ri-Rf for each stocks
for i in range(0,10):
    ticker = colname[i]
    new_ret[ticker+"-rf"] = new_ret[ticker] - new_ret["T90"]
new_ret = new_ret.iloc[:,12:]

Ri_Rf = new_ret.iloc[5:,1:].reset_index().drop("index",axis=1)
print(f"Ri-Rf.shape: {Ri_Rf.shape}")

print(new_ret.shape)

# get beta
res2 = []
for j in range(10):
    beta = []
    for i in range(15):
        x = new_ret.iloc[i:i+5,0].values.reshape(-1,1)
        y = new_ret.iloc[i:i+5,j+1]
        reg = LinearRegression().fit(x, y)
        beta.append(reg.coef_[0])
    res2.append(beta)
res2 = np.array(res2).transpose()

colname_beta = ["beta_{}".format(i) for i in range(1,11)]
res2_df = pd.DataFrame(res2,columns = colname_beta)
print(f"res2_df.shape: {res2_df.shape}")
res2_df
```

```
Ri-Rf.shape: (15, 10)
(20, 11)
res2_df.shape: (15, 10)
```

Out[55]:
```
      beta_1    beta_2    beta_3    beta_4    beta_5    beta_6    beta_7  \
0   1.322726  0.459057  1.136855 -0.580764  0.893527 -0.050227  0.658862
1   1.253075  0.374141  0.463619 -0.254742  0.946519 -0.140561  0.675401
2   1.381634  0.139995  0.651804 -0.060730  1.655714 -0.149125  0.418404
3   1.407462  1.015950  0.688784 -0.005918  0.850957  0.012436  0.455550
4   0.985542  1.007925  0.943262  0.150975  0.335045 -0.449358  0.794119
5   1.402770  0.522316 -0.502576  0.184358  1.645865  0.348485  0.908799
6   1.300668  0.823768 -0.381170  0.006505  1.470939  0.293074  1.040499
7   1.322651  0.831519 -0.420281  0.060241  1.463243  0.284430  1.021935
8   1.340259  0.885055 -0.381631  0.074307  1.447837  0.279619  1.034823
9   1.372969  0.788044 -0.179872  0.036117  1.354951  0.297987  1.088372
10  1.319070 -0.203441  0.654913 -1.025790  1.745656  0.845767  1.879149
11  1.449707 -1.223362  1.334118 -0.871737  2.021592  1.201527  1.728681
12  0.950201 -0.640231  1.232736 -0.343248  1.796997  1.193816  1.622492
13  0.568838 -0.104253  1.444478  0.054903  2.224122  1.459427  0.938468
14 -0.365586 -0.165127  1.344488  0.118736  2.712866  1.459955  1.031602

      beta_8    beta_9   beta_10
0   1.954596 -0.075261  2.506559
1   1.494529 -0.050869  1.810560
2   1.595577  0.164924  0.539667
3   2.195064  0.344185 -0.305714
4   3.811793  0.674113  3.350206
5   0.976428  1.343776  1.529174
6   1.243257  1.303810  2.185541
7   1.221589  1.284371  2.143952
8   1.352494  1.272262  2.346548
9   1.136712  1.230325  2.181774
10  0.380111  1.657227  2.750970
11 -0.172291  1.683146  1.521500
12  0.120113  1.516344  0.051039
13  0.685362  1.142271 -0.707793
14  0.890638  1.199635 -0.505526
```

In [46]:
```python
data2 = pd.read_csv("./book.csv")
colname =[i+"_BM" for i in ["AMZN","GE","JNJ","INTC","AMGN","IBM","BA","SO","SYK","ITW
# check the number of observations in each ticker
D = dict()
for i in data2["tic"]:
    if i not in D:
        D[i] = 1
    else:
        D[i] +=1
print(f"Show number of observations in each class:\n{D}")
# set Book value of Equity
BE = np.absolute(((data2["bkvlps"] * data2["csho"]).values).reshape(20,10))
BE = pd.DataFrame(BE, columns=colname)
BE = BE.iloc[5:,:].reset_index().drop("index",axis=1)
```

```
BE

Show number of observations in each class:
{'AMGN': 20, 'BA': 20, 'GE': 20, 'ITW': 20, 'INTC': 20, 'IBM': 20, 'JNJ': 20, 'SO': 20, 'SYK':
```

Out[46]:

```
          AMZN_BM          GE_BM         JNJ_BM        INTC_BM        AMGN_BM  \
0   117290.626078  118935.734242  116438.521618  123025.704375  130566.089266
1     4815.429985    5400.995405    6040.725841    6649.080129    7874.291995
2     8808.210592    9370.298498   10017.794638   10560.983332    9702.983437
3    32664.865000   37321.713000   35830.302000   35468.180000   37845.806700
4    41704.173000   49429.812300   45911.000000   51203.030400   58255.956200
5    20264.054798   20376.941060   23613.961298   22781.920546   27863.997643
6    22637.023720   23045.990230   20137.954642   18860.054472   22792.032173
7    16213.014656   18807.971927   24232.977288   22697.067717   26869.059569
8    50588.062943   56578.941169   57080.036267   64826.038255   74053.049575
9     9204.030484   10690.025536    7984.027283    8710.015516    9647.997480
10   14877.986450   16201.995078   17578.042313   18296.975057   19008.035265
11     671.496480     854.907600    1056.200320    1498.190680    2154.802940
12    6595.112920    7173.595310    7683.017400    8597.006000    9047.014200
13     266.287082     967.242262    1439.987009    1352.822175    1036.095420
14    5257.004400    6863.994500    7757.022000    8192.021400    9745.992900

           IBM_BM         BA_BM         SO_BM        SYK_BM        ITW_BM
0   128159.181864  98267.738305  75822.068438  64257.058770  30974.706784
1     7627.618365   7546.905538   9017.498625   9351.335158   7663.461622
2     6819.017939   5224.012806   4254.000010   4585.013150   3253.997370
3    38579.134100  36182.255100  36751.907400  42761.718200  39088.067400
4    55864.968000  61084.800000  66226.149000  69018.887200  74563.224400
5    29747.037466  33097.966236  28505.972623  28470.021785  13465.012009
6    11867.963306  14261.982445  18245.963603  17593.976487  16796.008541
7    31813.120079  37871.053896  39318.127731  43319.081196  42511.036067
8    69752.021537  71150.130037  70418.003198  60159.915165  59751.868694
9    10278.009900  10688.977440  11371.030800  12384.981120  13275.993744
10   19949.035019  20592.039334  23513.042993  24166.954914  24723.040020
11    2752.013250   3251.811040   4191.009340   5378.510400   5406.697800
12    8595.001800   8511.002100   9550.012500   9966.003840  11729.989440
13     227.225721    245.980800    431.015400   1196.998400   2672.004000
14   10740.988500  13383.983100  19285.014600  27709.000000  43548.999500
```

In [49]:

```python
data2 = pd.read_csv("./ME.csv")
colname =[i+"_ME" for i in ["AMZN","GE","JNJ","INTC","AMGN","IBM","BA","SO","SYK","ITW

# set Book value of Equity
ME = ((data2["PRC"] * data2["SHROUT"]).values).reshape(240,10)
ME = pd.DataFrame(ME, columns=colname)
ls = [i*12+11 for i in range(20)]
ME = ME.iloc[ls,:].reset_index().drop("index",axis=1)
```

```
ME = ME.iloc[5:,:].reset_index().drop("index",axis=1)
ME
```

Out[49]:

```
         AMZN_ME         GE_ME        JNJ_ME       INTC_ME       AMGN_ME  \
0   1.136079e+08  1.154545e+08  1.188556e+08  1.197989e+08  1.272213e+08
1   2.723139e+07  2.857864e+07  2.777307e+07  2.679104e+07  2.728569e+07
2   4.502438e+07  4.669871e+07  4.542196e+07  4.684835e+07  4.916498e+07
3   5.688063e+07  6.376830e+07  6.219776e+07  4.938549e+07  4.340307e+07
4   1.925390e+08  1.943257e+08  2.051617e+08  1.954618e+08  2.046972e+08
5   1.835678e+08  1.898499e+08  1.880861e+08  1.813225e+08  1.929616e+08
6   3.437803e+08  3.393271e+08  3.208388e+08  3.254521e+08  3.554385e+08
7   2.531289e+07  2.744373e+07  2.807339e+07  2.483737e+07  2.433389e+07
8   5.314158e+07  4.817546e+07  4.868024e+07  4.693222e+07  4.855490e+07
9   1.211920e+08  1.275053e+08  1.327750e+08  1.211472e+08  1.247522e+08
10  2.426928e+08  2.405492e+08  2.572320e+08  2.292128e+08  2.217891e+08
11  2.677217e+07  2.668163e+07  2.658130e+07  2.589361e+07  2.643306e+07
12  6.004134e+07  6.331412e+07  6.503319e+07  6.310484e+07  6.105387e+07
13  2.973210e+07  3.284218e+07  3.409104e+07  3.123858e+07  3.251481e+07
14  7.019599e+08  7.599286e+08  7.907356e+08  8.278026e+08  8.669303e+08

          IBM_ME         BA_ME         SO_ME        SYK_ME        ITW_ME
0   1.293314e+08  1.327692e+08  1.228494e+08  1.327009e+08  1.225642e+08
1   2.888971e+07  2.902835e+07  2.644822e+07  2.813275e+07  2.865946e+07
2   4.439887e+07  4.421633e+07  4.566654e+07  4.869732e+07  4.518880e+07
3   4.656366e+07  4.073255e+07  3.694740e+07  3.004707e+07  3.007527e+07
4   1.969356e+08  2.136595e+08  2.015197e+08  1.969198e+08  1.831429e+08
5   1.968164e+08  1.936027e+08  1.714144e+08  1.625342e+08  1.660024e+08
6   3.613404e+08  3.706764e+08  3.754501e+08  3.939825e+08  3.461093e+08
7   2.576585e+07  2.308591e+07  1.706897e+07  1.744088e+07  1.791626e+07
8   4.657382e+07  4.732502e+07  4.278098e+07  4.613693e+07  4.203587e+07
9   1.285751e+08  1.041763e+08  8.916442e+07  7.675560e+07  8.153892e+07
10  2.233107e+08  2.158316e+08  2.139603e+08  2.250508e+08  2.141885e+08
11  2.770324e+07  2.568704e+07  2.158180e+07  1.571337e+07  1.612921e+07
12  6.336512e+07  6.645054e+07  6.070062e+07  6.565485e+07  5.865381e+07
13  3.441868e+07  3.121404e+07  2.454634e+07  1.831113e+07  2.194784e+07
14  9.816812e+08  9.694520e+08  7.813774e+08  8.264408e+08  7.344168e+08
```

In [75]:

```
colname = ["lnME_"+str(i) for i in range(1,11)]
lnME = np.log(ME.values)
lnME = pd.DataFrame(lnME,columns=colname)
colname = ["lnBEME_"+str(i) for i in range(1,11)]
lnBEME = np.log(BE.values)-np.log(ME.values)
lnBEME = pd.DataFrame(lnBEME,columns=colname)
```

In [88]:

```
Ri_Rf.columns = ["Ri_Rf_"+str(i) for i in range(1,11)]
data2 = pd.concat([res2_df, lnME,lnBEME,Ri_Rf],axis=1)
```

In [93]:

```
print(f"data2.shape: {data2.shape}")
data2
```

```python
rename = []
for i in range(1,11):
    rename.append("beta"+str(i))
    rename.append("lnME_"+str(i))
    rename.append("lnBEME_"+str(i))
for i in range(1,11):
    rename.append("Ri_Rf_"+str(i))
data2.columns = rename
data2
```

data2.shape: (15, 40)

Out[93]:

```
       beta1    lnME_1  lnBEME_1     beta2    lnME_2  lnBEME_2     beta3  \
0   1.322726  0.459057  1.136855 -0.580764  0.893527 -0.050227  0.658862
1   1.253075  0.374141  0.463619 -0.254742  0.946519 -0.140561  0.675401
2   1.381634  0.139995  0.651804 -0.060730  1.655714 -0.149125  0.418404
3   1.407462  1.015950  0.688784 -0.005918  0.850957  0.012436  0.455550
4   0.985542  1.007925  0.943262  0.150975  0.335045 -0.449358  0.794119
5   1.402770  0.522316 -0.502576  0.184358  1.645865  0.348485  0.908799
6   1.300668  0.823768 -0.381170  0.006505  1.470939  0.293074  1.040499
7   1.322651  0.831519 -0.420281  0.060241  1.463243  0.284430  1.021935
8   1.340259  0.885055 -0.381631  0.074307  1.447837  0.279619  1.034823
9   1.372969  0.788044 -0.179872  0.036117  1.354951  0.297987  1.088372
10  1.319070 -0.203441  0.654913 -1.025790  1.745656  0.845767  1.879149
11  1.449707 -1.223362  1.334118 -0.871737  2.021592  1.201527  1.728681
12  0.950201 -0.640231  1.232736 -0.343248  1.796997  1.193816  1.622492
13  0.568838 -0.104253  1.444478  0.054903  2.224122  1.459427  0.938468
14 -0.365586 -0.165127  1.344488  0.118736  2.712866  1.459955  1.031602

      lnME_3  lnBEME_3     beta4  ...   Ri_Rf_1   Ri_Rf_2   Ri_Rf_3   Ri_Rf_4  \
0   1.954596 -0.075261  2.506559  ...  0.149042  0.021143 -0.029488  0.121815
1   1.494529 -0.050869  1.810560  ... -0.026359 -0.092758  0.256438  0.055165
2   1.595577  0.164924  0.539667  ... -0.008170  0.092057 -0.132418  0.054246
3   2.195064  0.344185 -0.305714  ...  0.011198  0.060877 -0.215201  0.023368
4   3.811793  0.674113  3.350206  ... -0.684775 -0.237239  0.215141 -0.026689
5   0.976428  1.343776  1.529174  ...  0.216464  0.440836  0.006029 -0.074466
6   1.243257  1.303810  2.185541  ...  0.111866  0.107077 -0.053672  0.182803
7   1.221589  1.284371  2.143952  ... -0.094812  0.274605  0.073359  0.187400
8   1.352494  1.272262  2.346548  ...  0.204846  0.058830  0.355150 -0.013868
9   1.136712  1.230325  2.181774  ...  0.275165 -0.033684  0.313697  0.001841
10  0.380111  1.657227  2.750970  ... -0.022727 -0.104999  0.414967  0.197208
11 -0.172291  1.683146  1.521500  ...  0.219063 -0.098230  0.067547 -0.041009
12  0.120113  1.516344  0.051039  ...  0.018043  0.221907 -0.067293  0.054221
13  0.685362  1.142271 -0.707793  ... -0.501549 -0.034918  0.225799  0.082850
14  0.890638  1.199635 -0.505526  ... -0.765248 -0.140112  0.206491  0.032288

     Ri_Rf_5   Ri_Rf_6   Ri_Rf_7   Ri_Rf_8   Ri_Rf_9  Ri_Rf_10
```

```
0    0.261167  0.171792  0.125362 -0.327919  0.053162 -0.187387
1    0.276973 -0.032351 -0.054991  0.144512 -0.094768  0.176965
2    0.220206  0.076930  0.054724 -0.149162  0.129629 -0.088696
3    0.026615  0.012555  0.158834  0.243511  0.245125  0.925640
4   -0.636682 -0.107721 -0.395275 -0.575955 -0.605274 -0.673875
5    0.332795  0.099118  0.395732  0.341370  0.278139  1.055799
6    0.232385 -0.001940  0.046412  0.097992  0.027686  0.327405
7    0.093379  0.088299 -0.107460  0.232353 -0.065589  0.081721
8    0.045455  0.101551  0.311256 -0.160100  0.113231  0.417433
9    0.624482  0.344658  0.295523  0.194956  0.336475  0.481861
10   0.020588  0.201019  0.149744  0.412249  0.236795 -0.137706
11   0.159134  0.004256  0.020519  0.013167  0.039610  0.830236
12   0.099430  0.111423  0.331695  0.056980  0.217297  0.134242
13   0.630926  0.215653  0.341499  0.256734  0.275521  0.465375
14   0.197992  0.069332 -0.162269  0.086556  0.130415  0.417722

[15 rows x 40 columns]
```

In [105]:
```python
gama_df = []
for i in range(15):
    x=data2.iloc[i,0:30].values.reshape(10,3)
    y=data2.iloc[i,30:]
    reg = LinearRegression().fit(x,y)
    gama_df.append([reg.intercept_, reg.coef_[0],reg.coef_[1],reg.coef_[2]])
gama_df = pd.DataFrame(gama_df,columns=["gama_0","gama_1","gama_2","gama_3"])
gama_df
```

Out[105]:

	gama_0	gama_1	gama_2	gama_3
0	-0.014790	0.007920	0.001163	0.004865
1	0.074351	0.003134	-0.007298	0.006111
2	0.009210	0.004804	-0.000771	0.003031
3	0.203046	-0.003617	-0.004198	-0.007603
4	-0.462436	-0.027520	0.034569	-0.000940
5	0.362140	0.010986	-0.012995	-0.010034
6	0.108061	-0.005584	0.003117	0.001300
7	0.083051	-0.008555	0.000729	0.007838
8	0.147166	0.008293	-0.002278	-0.008362
9	0.314193	0.027163	-0.026198	0.005230
10	0.137763	-0.001764	0.001255	-0.000560
11	0.161373	0.006518	-0.014272	0.000887
12	0.099075	-0.000897	0.009537	-0.010239
13	0.187425	0.007842	-0.000435	-0.004543
14	0.026439	-0.000453	-0.005185	0.004338

In [110]:
```python
# Question 2.a

degree_of_freedom = gama_df.shape[0]-1
print(f" Degree of Freedom is: {degree_of_freedom}")
```

```
gama_0_T_stat = (gama_df.gama_0.mean() - 0)/gama_df.gama_0.std()* np.sqrt(degree_of_
gama_1_T_stat = (gama_df.gama_1.mean() - 0)/gama_df.gama_1.std()* np.sqrt(degree_of_
gama_2_T_stat = (gama_df.gama_2.mean() - 0)/gama_df.gama_2.std()* np.sqrt(degree_of_
gama_3_T_stat = (gama_df.gama_3.mean() - 0)/gama_df.gama_3.std()* np.sqrt(degree_of_
print(f"t-statistic (14 degree of freedom) at 95% confidence interval: {scipy.stats.
print(f"gama_0_T_stat: {gama_0_T_stat}")
print(f"gama_1_T_stat: {gama_1_T_stat}")
print(f"gama_0_T_stat: {gama_2_T_stat}")
print(f"gama_1_T_stat: {gama_3_T_stat}")
print("\nOnly gama_0 is statistically significant.")
```

```
 Degree of Freedom is: 14
t-statistic (14 degree of freedom) at 95% confidence interval: 1.7613101357748562
gama_0_T_stat: 1.926231389470359
gama_1_T_stat: 0.597124302325168
gama_0_T_stat: -0.44080809623378886
gama_1_T_stat: -0.3518066741130468

Only gama_0 is statistically significant.
```

In [111]:
```
# Question 2 B.
print(f"gamma_1_T_stat: {gama_1_T_stat}")
print("It is not statistical significant, thus We don't have a trade-off between mar
```

```
gamma_1_T_stat: 0.597124302325168
It is not statistical significant, thus We don't have a trade-off between market beta and expe
```

In [115]:
```
# Question 2 C.
print(f"gamma_2_T_stat: {gama_2_T_stat}")
print(f"gamma_3_T_stat: {gama_3_T_stat}")
print("All of them are not statistically significant. Thus, they cannot explain the
```

```
gamma_2_T_stat: -0.44080809623378886
gamma_3_T_stat: -0.3518066741130468
All of them are not statistically significant. Thus, they cannot explain the cross-section sto
```