MFE 230E

Empirical Methods

Assignment 6

Nicolas Corthorn
Robert Sharow
Xian Li
Sherry Xiu

1. (a) The coefficients, standard errors and p-values using Newey-West of each α_i and β_i are shown in Tables 1 and 2. We can see that the p-values are all 0, meaning that the beta's are all significant.

i	coef	s.e.	p-value
1.0	-0.68	0.19	0.0003
2.0	0.10	0.14	0.4410
3.0	0.39	0.13	0.0022
4.0	0.51	0.13	0.0001
5.0	0.68	0.17	0.0001
6.0	-0.60	0.16	0.0003
7.0	0.09	0.12	0.4405
8.0	0.27	0.10	0.0074
9.0	0.42	0.10	0.0000
10.0	0.51	0.14	0.0002
11.0	-0.46	0.16	0.0044
12.0	0.03	0.10	0.7916
13.0	0.17	0.09	0.0562
14.0	0.22	0.09	0.0101
15.0	0.50	0.12	0.0001
16.0	-0.49	0.16	0.0031
17.0	0.01	0.10	0.9442
18.0	0.13	0.08	0.0960
19.0	0.26	0.07	0.0002
20.0	0.42	0.11	0.0001
21.0	-0.48	0.16	0.0027
22.0	-0.01	0.10	0.9386
23.0	-0.04	0.07	0.5346
24.0	0.09	0.06	0.1785
25.0	0.25	0.10	0.0139

Table 1: Results for each α_i

i	coef	s.e.	p-value
1.0	1.38	0.06	0.0
2.0	1.06	0.05	0.0
3.0	0.99	0.04	0.0
4.0	1.00	0.04	0.0
5.0	1.21	0.05	0.0
6.0	1.46	0.06	0.0
7.0	1.13	0.04	0.0
8.0	1.03	0.04	0.0
9.0	1.05	0.04	0.0
10.0	1.28	0.04	0.0
11.0	1.37	0.06	0.0
12.0	1.10	0.03	0.0
13.0	1.02	0.03	0.0
14.0	1.01	0.03	0.0
15.0	1.22	0.04	0.0
16.0	1.33	0.06	0.0
17.0	1.11	0.03	0.0
18.0	1.01	0.03	0.0
19.0	1.00	0.02	0.0
20.0	1.15	0.03	0.0
21.0	1.23	0.06	0.0
22.0	0.94	0.04	0.0
23.0	0.90	0.02	0.0
24.0	0.89	0.02	0.0
25.0	1.02	0.03	0.0

Table 2: Results for each β_i

(b) At the significance level of 5%, we can see from the table that 16 out of 25 portfolios have statistically significant α's. Economically, when trading in large dollar amount, 0.01 in α can still bring a lot of profit or loss to the firm. Since the absolute value of the α's are at least 0.01, I would say that they are all economically significant.

(c) The in-sample performance of the CAPM model can be seen in Figure 1. This plot was constructed by assuming $\alpha_i =' 0, \forall i$ as the model assumes. The model does a poor job in accomplishing similar fitted and actual returns, even in-sample.

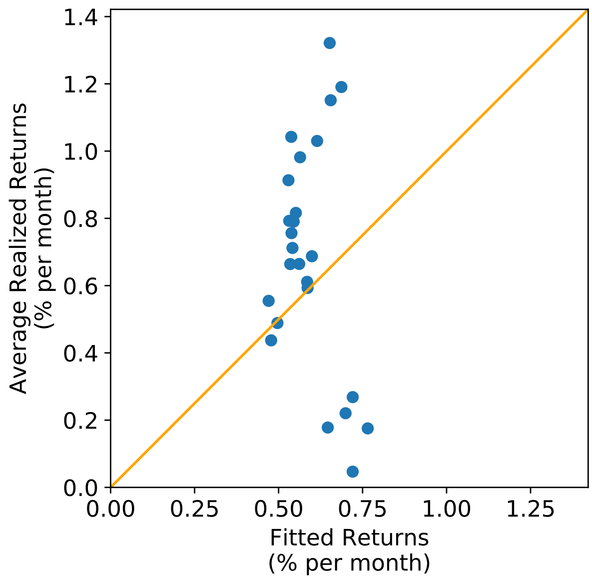


Figure 1: Average excess returns vs fitted returns for CAPM model

(d) The t-test results for each α individually can be seen in Table 1. As mentioned in part b, 16 out of 25 portfolios have statistically significant α's. This should be enough to reject that all α_i are equal to zero.

(e) In the notebook attached we implemented the GRS joint test $\alpha_i = 0, \forall i$. Consistent with lectures, we got a huge χ^2 statistic of 135.77 with a p-value so small that the not even the machine can distinguish it from zero. Therefore, the null hypothesis is rejected and consequently the CAPM as well is rejected from the data.

(f) Figure 2 shows the efficient frontier, the tangency portfolio, as well as the capital market line. We can see that clearly, CML has a lower Sharpe ratio than the tangency portfolio formed by the 25 sample portfolios. Do notice that the tangency portfolio is not quite "tangent", we suspect this comes from the fact that we assume the volatility of risk free return is 0, while there is always some volatility in the return that can change the slope of the line. But this volatility should be insignificant and neglecting it won't affect our result heavily. Overall, the CAPM model doesn't work well.

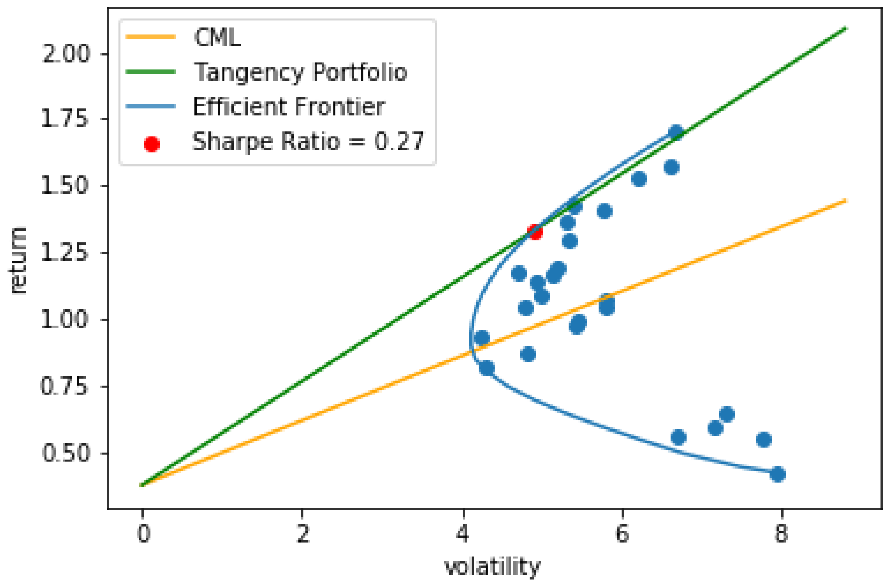


Figure 2: Efficient Frontier of the 25 portfolios

(g) Overall, I wouldn't say CAPM explain the 25 test portfolios well and there are a number of clear indication throughout the analysis supporting this result. first of all, more than half of the α's are significant, meaning that the CAPM model doesn't work well for those portfolios. In addition, the tangency portfolio has a much higher slope than the CML, meaning that we can produce a portfolio that has higher Sharpe ratio than the market. And is obviously incorrect in the CAPM world: CML should have the highest Sharpe ratio, higher than any portfolio composed in the market. Therefore, we have reached to the conclusion that CAPM doesn't work well for the 25 test portfolios.

2. (a) The coefficients, standard errors and p-values using Newey-West of each α_i and β_i are shown in Tables 3, 4, 5, and 6. We can see that the p-values for the market excess return are all 0, meaning that β_1's are all significant. For the beta of stock size, we can see that the bottom "corner" portfolios, the big market sized firms with low and high momentum have statistically insignificant betas. With a 5% significance level, we can see that the beta for value stocks are insignificant for 2 portfolios: the small low momentum portfolio, and the 4th highest momentum for large stocks. Overall, the model works fairly well for the portfolios in the middle. For the "corner" portfolios with significant momentum or market equity size, the the model is like to produce insignificant betas.

i	coef	s.e.	p-value
1.0	-0.96	0.13	0.0000
2.0	-0.18	0.07	0.0107
3.0	0.12	0.06	0.0344
4.0	0.29	0.06	0.0000
5.0	0.54	0.09	0.0000
6.0	-0.81	0.12	0.0000
7.0	-0.14	0.08	0.0643
8.0	0.06	0.06	0.3099
9.0	0.23	0.05	0.0000
10.0	0.45	0.08	0.0000
11.0	-0.62	0.15	0.0000
12.0	-0.16	0.08	0.0513
13.0	-0.02	0.07	0.7448
14.0	0.05	0.06	0.4181
15.0	0.48	0.09	0.0000
16.0	-0.64	0.16	0.0001
17.0	-0.15	0.09	0.1008
18.0	-0.01	0.07	0.8387
19.0	0.17	0.06	0.0078
20.0	0.44	0.09	0.0000
21.0	-0.55	0.16	0.0004
22.0	-0.09	0.10	0.3665
23.0	-0.08	0.06	0.1967
24.0	0.09	0.06	0.1437
25.0	0.34	0.10	0.0004

Table 3: Results for each α_i

i	coef	s.e.	p-value
1.0	1.19	0.05	0.0
2.0	0.93	0.03	0.0
3.0	0.88	0.02	0.0
4.0	0.86	0.02	0.0
5.0	0.98	0.03	0.0
6.0	1.31	0.06	0.0
7.0	1.03	0.03	0.0
8.0	0.95	0.02	0.0
9.0	0.94	0.02	0.0
10.0	1.07	0.03	0.0
11.0	1.29	0.06	0.0
12.0	1.06	0.03	0.0
13.0	0.98	0.02	0.0
14.0	0.97	0.02	0.0
15.0	1.05	0.03	0.0
16.0	1.32	0.06	0.0
17.0	1.13	0.03	0.0
18.0	1.02	0.02	0.0
19.0	0.99	0.02	0.0
20.0	1.03	0.03	0.0
21.0	1.30	0.06	0.0
22.0	1.02	0.03	0.0
23.0	0.97	0.02	0.0
24.0	0.94	0.02	0.0
25.0	0.99	0.03	0.0

Table 4: Results for $\beta_{i,1}$

i	coef	s.e.	p-value
1.0	1.23	0.09	0.0000
2.0	0.98	0.06	0.0000
3.0	0.90	0.05	0.0000
4.0	0.92	0.03	0.0000
5.0	1.16	0.05	0.0000
6.0	0.95	0.07	0.0000
7.0	0.78	0.06	0.0000
8.0	0.69	0.04	0.0000
9.0	0.77	0.03	0.0000
10.0	0.96	0.05	0.0000
11.0	0.61	0.09	0.0000
12.0	0.48	0.06	0.0000
13.0	0.47	0.05	0.0000
14.0	0.45	0.04	0.0000
15.0	0.72	0.04	0.0000
16.0	0.31	0.08	0.0002
17.0	0.18	0.06	0.0042
18.0	0.18	0.06	0.0012
19.0	0.18	0.05	0.0005
20.0	0.45	0.06	0.0000
21.0	-0.13	0.07	0.0758
22.0	-0.19	0.05	0.0000
23.0	-0.20	0.03	0.0000
24.0	-0.22	0.03	0.0000
25.0	-0.02	0.04	0.5524

Table 5: Results for $\beta_{i,2}$

i	coef	s.e.	p-value
1.0	0.25	0.11	0.0200
2.0	0.38	0.05	0.0000
3.0	0.35	0.04	0.0000
4.0	0.23	0.04	0.0000
5.0	-0.06	0.05	0.2334
6.0	0.20	0.10	0.0433
7.0	0.31	0.06	0.0000
8.0	0.29	0.04	0.0000
9.0	0.23	0.03	0.0000
10.0	-0.18	0.05	0.0006
11.0	0.20	0.10	0.0496
12.0	0.31	0.06	0.0000
13.0	0.32	0.05	0.0000
14.0	0.27	0.04	0.0000
15.0	-0.20	0.05	0.0001
16.0	0.28	0.10	0.0041
17.0	0.32	0.06	0.0000
18.0	0.30	0.05	0.0000
19.0	0.18	0.05	0.0002
20.0	-0.20	0.06	0.0017
21.0	0.23	0.10	0.0148
22.0	0.27	0.06	0.0000
23.0	0.17	0.04	0.0000
24.0	0.07	0.04	0.0640
25.0	-0.22	0.05	0.0001

Table 6: Results for $\beta_{i,3}$

(b) At the significance level of 5%, we can see from the table that 15 out of 25 portfolios have statistically significant α's. Economically, when trading in large dollar amount, 0.01 in α can still bring a lot of profit or loss to the firm. Since the absolute value of the α's are at least 0.01, I would say that they are all economically significant.

(c) The in-sample performance of the Fama-French model can be seen in Figure 3. This plot was constructed by assuming $\alpha_i =' 0, \forall i$ as the model assumes. The model still has a poor performance.

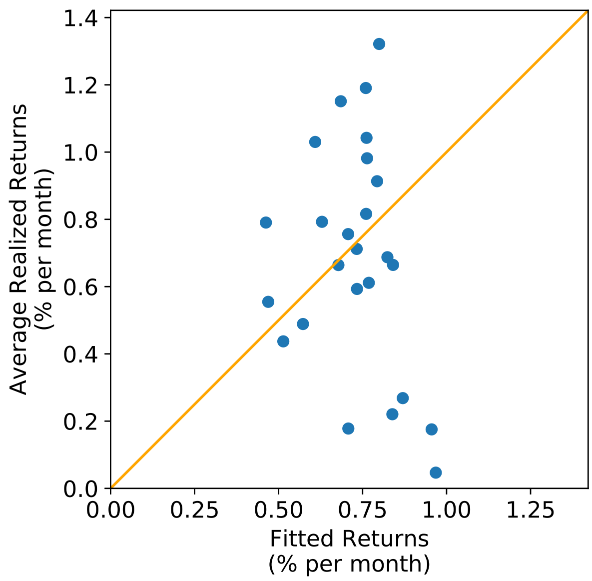


Figure 3: Average excess returns vs fitted returns for Fama-French model

(d) The t-test results for each α individually can be seen in Table 3. As mentioned in part b, 15 out of 25 portfolios have statistically significant α's.

(e) We got a high χ^2 statistic of 136.02 with again p-value of zero essentially. The null hypothesis and the model are again rejected.

(f) Please refer to Figure 2 in question 1. Since the same sample is used, CML and efficient frontier should be the same.

(g) Overall, the model explains the 25 test portfolios better than CAPM, but there is still space of improvement. We can see that with an additional factor, we have reduced the number of significant α's from 16 to 15, but that is still more than half of the α's being statistically significant. There is still a lot variation in the excess return that is not explained by the model.

3. (a) The coefficients, standard errors and p-values using Newey-West of each α_i and β_i are shown in Tables 7, 8, 9, 10, and 11. We can see that the p-values for the market excess return are all 0, meaning that β_1's are all significant. For the beta of stock size, again, the p-values indicate all the β_2's are significant, except for the "BIG HiPRIOR" portfolio. At a 5% significance level, we can see that the betas of value stocks generally work well for the portfolios in the middle, but are more likely to be insignificant for small and big stocks. The momentum betas are all statistically significant for all 25 portfolios. Overall, the model works fairly well. Comparing the significance of the β's, we would say this model works better than the previous 2 models.

i	coef	s.e.	p-value
1.0	-0.36	0.12	0.0033
2.0	0.03	0.06	0.6117
3.0	0.19	0.06	0.0012
4.0	0.22	0.07	0.0011
5.0	0.29	0.08	0.0005
6.0	-0.19	0.08	0.0258
7.0	0.13	0.06	0.0345
8.0	0.11	0.06	0.0722
9.0	0.15	0.06	0.0069
10.0	0.14	0.07	0.0281
11.0	0.03	0.09	0.7025
12.0	0.12	0.06	0.0759
13.0	0.09	0.06	0.1469
14.0	-0.04	0.07	0.5618
15.0	0.13	0.06	0.0323
16.0	0.05	0.10	0.6160
17.0	0.18	0.07	0.0121
18.0	0.10	0.07	0.1274
19.0	0.10	0.07	0.1430
20.0	0.06	0.08	0.4349
21.0	0.10	0.11	0.3449
22.0	0.28	0.07	0.0001
23.0	0.00	0.07	0.9529
24.0	-0.06	0.05	0.2744
25.0	-0.06	0.07	0.3482

Table 7: Results for each α_i

i	coef	s.e.	p-value
1.0	1.06	0.03	0.0
2.0	0.89	0.02	0.0
3.0	0.86	0.02	0.0
4.0	0.88	0.02	0.0
5.0	1.04	0.02	0.0
6.0	1.18	0.02	0.0
7.0	0.97	0.02	0.0
8.0	0.94	0.02	0.0
9.0	0.96	0.02	0.0
10.0	1.14	0.02	0.0
11.0	1.15	0.03	0.0
12.0	1.00	0.02	0.0
13.0	0.96	0.02	0.0
14.0	0.99	0.02	0.0
15.0	1.13	0.02	0.0
16.0	1.17	0.03	0.0
17.0	1.05	0.02	0.0
18.0	1.00	0.02	0.0
19.0	1.01	0.02	0.0
20.0	1.11	0.02	0.0
21.0	1.15	0.03	0.0
22.0	0.94	0.02	0.0
23.0	0.95	0.02	0.0
24.0	0.97	0.02	0.0
25.0	1.08	0.02	0.0

Table 8: Results for β_i 1

i	coef	s.e.	p-value
1.0	1.24	0.04	0.0000
2.0	0.98	0.04	0.0000
3.0	0.90	0.05	0.0000
4.0	0.92	0.03	0.0000
5.0	1.15	0.05	0.0000
6.0	0.96	0.03	0.0000
7.0	0.78	0.04	0.0000
8.0	0.70	0.04	0.0000
9.0	0.77	0.03	0.0000
10.0	0.95	0.03	0.0000
11.0	0.62	0.04	0.0000
12.0	0.48	0.04	0.0000
13.0	0.48	0.04	0.0000
14.0	0.45	0.05	0.0000
15.0	0.72	0.03	0.0000
16.0	0.32	0.04	0.0000
17.0	0.18	0.04	0.0000
18.0	0.18	0.05	0.0001
19.0	0.17	0.05	0.0013
20.0	0.44	0.03	0.0000
21.0	-0.12	0.04	0.0005
22.0	-0.19	0.03	0.0000
23.0	-0.20	0.03	0.0000
24.0	-0.22	0.03	0.0000
25.0	-0.03	0.03	0.3020

Table 9: Results for β_i 2

i	coef	s.e.	p-value
1.0	0.01	0.07	0.9080
2.0	0.29	0.04	0.0000
3.0	0.32	0.03	0.0000
4.0	0.26	0.04	0.0000
5.0	0.04	0.05	0.4014
6.0	-0.06	0.03	0.0880
7.0	0.20	0.03	0.0000
8.0	0.27	0.03	0.0000
9.0	0.26	0.03	0.0000
10.0	-0.05	0.03	0.1142
11.0	-0.07	0.05	0.1459
12.0	0.20	0.04	0.0000
13.0	0.28	0.04	0.0000
14.0	0.30	0.04	0.0000
15.0	-0.06	0.03	0.0189
16.0	-0.00	0.05	0.9592
17.0	0.19	0.04	0.0000
18.0	0.25	0.05	0.0000
19.0	0.21	0.04	0.0000
20.0	-0.04	0.03	0.2520
21.0	-0.04	0.05	0.4550
22.0	0.12	0.04	0.0013
23.0	0.13	0.03	0.0001
24.0	0.13	0.03	0.0001
25.0	-0.06	0.03	0.0687

Table 10: Results for β_i 3

i	coef	s.e.	p-value
1.0	-0.70	0.06	0.0000
2.0	-0.25	0.02	0.0000
3.0	-0.07	0.03	0.0035
4.0	0.08	0.03	0.0054
5.0	0.29	0.04	0.0000
6.0	-0.72	0.05	0.0000
7.0	-0.31	0.03	0.0000
8.0	-0.06	0.02	0.0117
9.0	0.09	0.02	0.0001
10.0	0.35	0.02	0.0000
11.0	-0.77	0.03	0.0000
12.0	-0.32	0.02	0.0000
13.0	-0.13	0.02	0.0000
14.0	0.10	0.03	0.0013
15.0	0.41	0.02	0.0000
16.0	-0.81	0.04	0.0000
17.0	-0.38	0.03	0.0000
18.0	-0.14	0.03	0.0000
19.0	0.09	0.03	0.0016
20.0	0.44	0.02	0.0000
21.0	-0.76	0.04	0.0000
22.0	-0.43	0.03	0.0000
23.0	-0.10	0.03	0.0020
24.0	0.17	0.03	0.0000
25.0	0.47	0.03	0.0000

Table 11: Results for β_i 4

(b) At the significance level of 5%, we can see from the table that 11 out of 25 portfolios have statistically significant α's. Economically, when trading in large dollar amount, 0.01 in α can still bring a lot of profit or loss to the firm. Since the absolute value of the α's are at least 0.01, I would say that they are all economically significant.

(c) The in-sample performance of the Fama-French-Carhart model can be seen in Figure 4. This model does a good job fitting returns to realized returns in sample for the given portfolios, that not coincidentally are based in part in momentum.

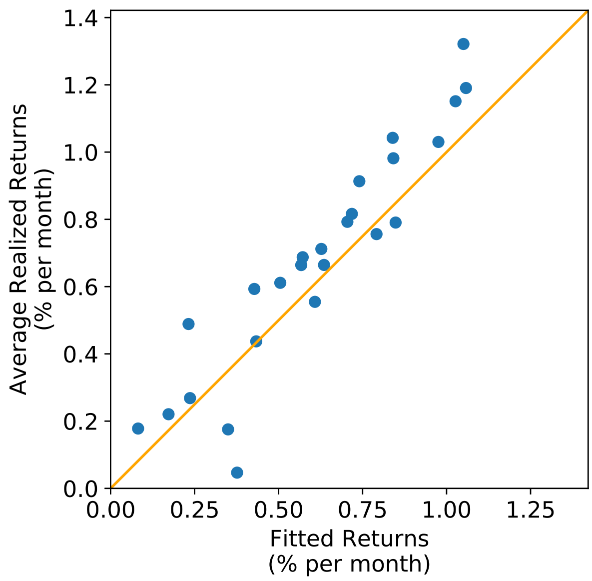


Figure 4: Average excess returns vs fitted returns for Fama-French-Carhart model

(d) The t-test results for each α individually can be seen in Table 7. As mentioned in part b, 11 out of 25 portfolios have statistically significant α's.

(e) We got a still a high χ^2 statistic of 102.77 with a p-value of $2 \cdot 10^{-11}$. The null hypothesis and the model are still (dramatically) rejected, but nevertheless we see an important improvement in the χ^2 statistic.

(f) Please refer to Figure 2 in question 1. Since the same sample is used, CML and efficient frontier should be the same.

(g) Out of the 3 models, this one explains the 25 test portfolios the best. It has the least number of statistically significant α's and most of the β's are significant. Although there a a few α's that are not close to 0, but it seems like the 4 factors have explained a lot of variations in the excess return and overall, the model works fairly well for all the portfolios. In contrast, the previous 2 models may incur more problem while explaining small or big stocks.

4. From our cross-sectional regression:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.856
Model:                            OLS   Adj. R-squared:                  0.827
Method:                 Least Squares   F-statistic:                     29.77
Date:                Wed, 08 May 2019   Prob (F-statistic):           3.62e-08
Time:                        22:23:39   Log-Likelihood:                 16.334
No. Observations:                  25   AIC:                            -22.67
Df Residuals:                      20   BIC:                            -16.57
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1131      0.708      0.160      0.875      -1.364       1.590
x1             0.4866      0.649      0.750      0.462      -0.866       1.840
x2             0.2118      0.065      3.261      0.004       0.076       0.347
x3             0.5539      0.430      1.289      0.212      -0.342       1.450
x4             0.7403      0.075      9.834      0.000       0.583       0.897
==============================================================================
Omnibus:                        2.620   Durbin-Watson:                   1.325
Prob(Omnibus):                  0.270   Jarque-Bera (JB):                1.538
Skew:                          -0.599   Prob(JB):                        0.463
Kurtosis:                       3.207   Cond. No.                         56.5
==============================================================================
```

	Estimated $\alpha_i's$ (Residuals)
0	-0.330152
1	-0.064008
2	0.065128
3	0.100153
4	0.222379
5	-0.146268
6	0.055824
7	-0.006174
8	0.031867
9	0.087408
10	0.073104
11	0.040083
12	-0.021674
13	-0.178996
14	0.069332
15	0.070439
16	0.105063
17	-0.009512
18	-0.027528
19	-0.023076
20	0.113714
21	0.210012
22	-0.096820
23	-0.181696
24	-0.158605

Table 12

From our results, we can see that the market risk premium is 0.4866, the SMB risk premium is 0.2118, the HML risk premium is 0.5539, and the momentum risk premium is 0.7403. The standard errors for the risk premia are 0.649, 0.065, 0.430, and 0.075 respectively. The α of our regression is 0.1131 (using the α as the constant of the cross-sectional regression as done in class), with a standard error of 0.708 and a P-value of 0.875, which is statistically insignificant. This last result is consistent with the theory of the model.

5. The graphs below are the betas for the "corner" portfolios from rolling regressions. We can see that the beta's are actually varying quite a bit over time.

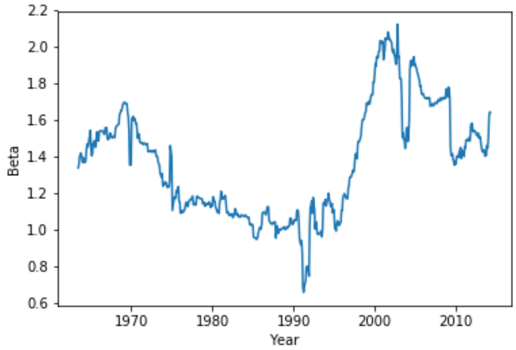


Figure 5: Betas for upper left corner: SMALL LoPRIOR

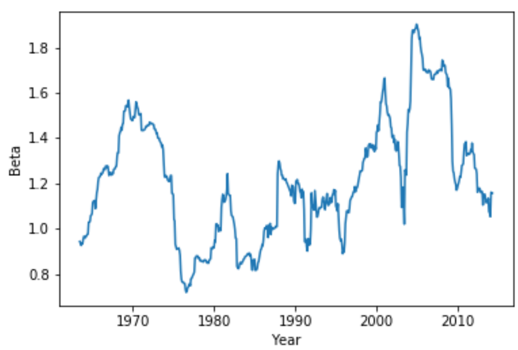


Figure 7: Betas for upper left corner: BIG LoPRIOR

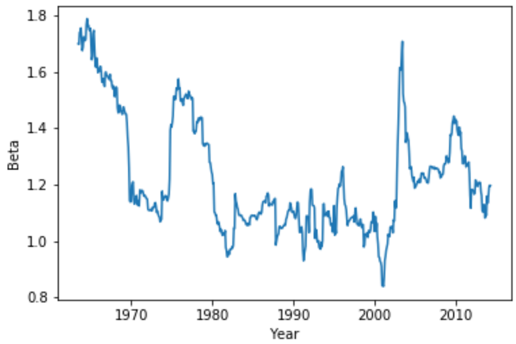


Figure 6: Betas for upper right corner: SMALL HiPRIOR

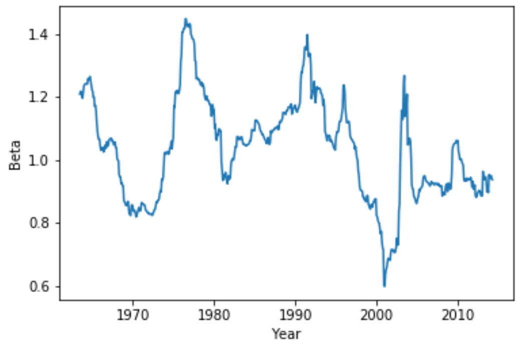


Figure 8: Betas for upper right corner: BIG HiPRIOR

6. Using the Fama-Macbeth estimation method, our lambdas for the 4 factor model are:

$$\begin{aligned}
\hat{\lambda}_0 &= 0.96628831 \\
se(\hat{\lambda}_0) &= 0.23753546 \\
\hat{\lambda}_1 &= -0.35323413 \\
se(\hat{\lambda}_1) &= 0.27353665 \\
\hat{\lambda}_2 &= 0.13772876 \\
se(\hat{\lambda}_2) &= 0.12487135 \\
\hat{\lambda}_3 &= -0.08895711 \\
se(\hat{\lambda}_3) &= 0.15668635 \\
\hat{\lambda}_4 &= 0.60973755 \\
se(\hat{\lambda}_4) &= 0.17426541
\end{aligned}$$

The covariance matrix of our lambda estimates is:

$$cov(\hat{\lambda}) = \begin{bmatrix} 0.056 & -0.049 & 0.004 & -0.008 & 0.001 \\ -0.049 & 0.075 & 0.002 & -0.001 & -0.006 \\ 0.004 & 0.002 & 0.016 & -0.003 & -0.002 \\ -0.008 & -0.001 & -0.003 & 0.024 & -0.002 \\ 0.001 & -0.006 & -0.002 & -0.002 & 0.030 \end{bmatrix}$$

The α_i's estimated using Fama-Macbeth model are showed in Table 13. We can see from Table 14 there are significant absolute differences between the α_i's using F-M model against regular cross-sectional model for some portfolios.

Comparing with our results in problem 4, we can see that our estimated factor premia are very different. This tells us that the betas for each factor change over time, which agrees with our observations from question 5. From the magnitude of the standard errors of our lambda estimates, we can see that the betas fluctuate and that our lambda estimates are not very stable. Comparing the differences in our alpha estimates, we can see that the magnitude of the suggested pricing errors for our 25 portfolios vary quite a bit, reinforcing our conclusion that the Fama-Macbeth results are really different from our regular cross sectional regression on problem 4.

i	coefficients
0	-0.476330
1	-0.094612
2	0.062863
3	0.059872
4	0.287965
5	-0.208161
6	0.014264
7	0.046741
8	0.086373
9	0.131586
10	-0.067525
11	0.037431
12	0.017274
13	-0.085734
14	0.072046
15	0.056594
16	0.151075
17	0.084502
18	0.024592
19	0.016560
20	0.107214
21	0.150633
22	-0.104543
23	-0.156942
24	-0.213739

Table 13: Estimated $\alpha_i's$ for F-M

i	absolute difference
0	0.146177
1	0.030604
2	0.002265
3	0.040281
4	0.065587
5	0.061893
6	0.041560
7	0.052915
8	0.054506
9	0.044177
10	0.140629
11	0.002652
12	0.038948
13	0.093262
14	0.002714
15	0.013846
16	0.046012
17	0.094014
18	0.052120
19	0.039636
20	0.006500
21	0.059379
22	0.007723
23	0.024753
24	0.055134

Table 14: Absolute difference of α_i's

MFE 230E Assignment 6

May 9, 2019

In [1]:
```
import numpy as np
import statsmodels.api as sm
import pandas as pd
from datetime import datetime
from statsmodels.stats.sandwich_covariance import cov_hac as cov
import matplotlib.pyplot as plt
from scipy.stats import chi2
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
from pypfopt import risk_models
from pypfopt import expected_returns
from pypfopt.efficient_frontier import EfficientFrontier
```

In [2]:
```
# Import the data sets
factors = pd.read_csv('F-F_Research_Data_5_Factors_2x3.CSV',skiprows=3)
mom = pd.read_csv('F-F_Momentum_Factor.CSV',skiprows=13)
data = pd.read_csv('25_Portfolios_ME_Prior_12_2.CSV',skiprows=11)

# Clean up
factors = factors.drop(range(669,len(factors)),axis=0)
factors.columns = ['Date', 'Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']
factors['Date'] = factors['Date'].apply(lambda x: datetime.strptime(str(x), '%Y%m'))
for column in factors.columns[1:]:
    factors[column]= factors[column].astype('float')

mom = mom.drop(range(1107,len(mom)),axis=0)
mom.columns = ['Date', 'MOM']
mom['Date'] = mom['Date'].apply(lambda x: datetime.strptime(str(x), '%Y%m'))
for column in mom.columns[1:]:
    mom[column]= mom[column].astype('float')

data = data.drop(range(1107,len(data)),axis=0)
data = data.drop(range(0,438),axis=0)
data = data.reset_index(drop=True)
data.rename(columns={'Unnamed: 0':'Date'}, inplace=True)
data['Date'] = data['Date'].apply(lambda x: datetime.strptime(str(x), '%Y%m'))
for column in data.columns[1:]:
```

```
            data[column]= data[column].astype('float')

    # Final data output
    data = pd.merge(data, factors.loc[:,['Date','RF']], on='Date', how='left')
    factors = pd.merge(factors, mom, on='Date', how='left')
```

In [3]:
```
### Wrapped OLS statsmodel routine
def OLS(x, y, addcon=True, cov_type=None, sig_level=.05, summary=0, cov_kwds = None):
    """Wrapper for statsmodels OLS regression
    """
    if addcon:
        X = sm.add_constant(x)
    else:
        X = x
    if cov_type==None:
        ols_results = sm.OLS(y,X).fit(cov_type='nonrobust')
    else:
        ols_results = sm.OLS(y,X).fit(cov_type=cov_type, cov_kwds=cov_kwds)

    ### print out the OLS estimation results
    if summary==1:
        print(ols_results.summary())

    ols_cov_mat  = cov(ols_results)
    ols_beta_hat = ols_results.params # beta_hat
    ols_resids   = ols_results.resid  # resids
    ols_se       = ols_results.bse
    ols_pvalues  = ols_results.pvalues

    return ols_beta_hat, ols_resids, ols_se, ols_cov_mat, ols_pvalues
```

In [4]:
```
def latex_table(df, caption="", label="", index=False):
    return "\\begin{table}[H]\n\centering\n"+df.to_latex(index=index)+"\caption{"+captio
```

In [5]:
```
def GRS(alpha_hat, x, residuals, N, print_result=1):
    """
        alpha_hat is a numpy array with shape (T,1)
        x is a numpy array that contains de factors as columns [x_t,1 | ... | x_t,N ]
        residuals is a numpy array that contains de residuals as rows [r_t,1 | ... | r_t
    """
    T = len(x)
    k = x.shape[1]
    sigma = np.cov(residuals)
    sigma_inv = np.linalg.inv(sigma)
    mu_F = np.mean(x,axis=0)
    if x.shape[1]==1:
        omega_F = np.array([[np.var(x)]])
    else:
```

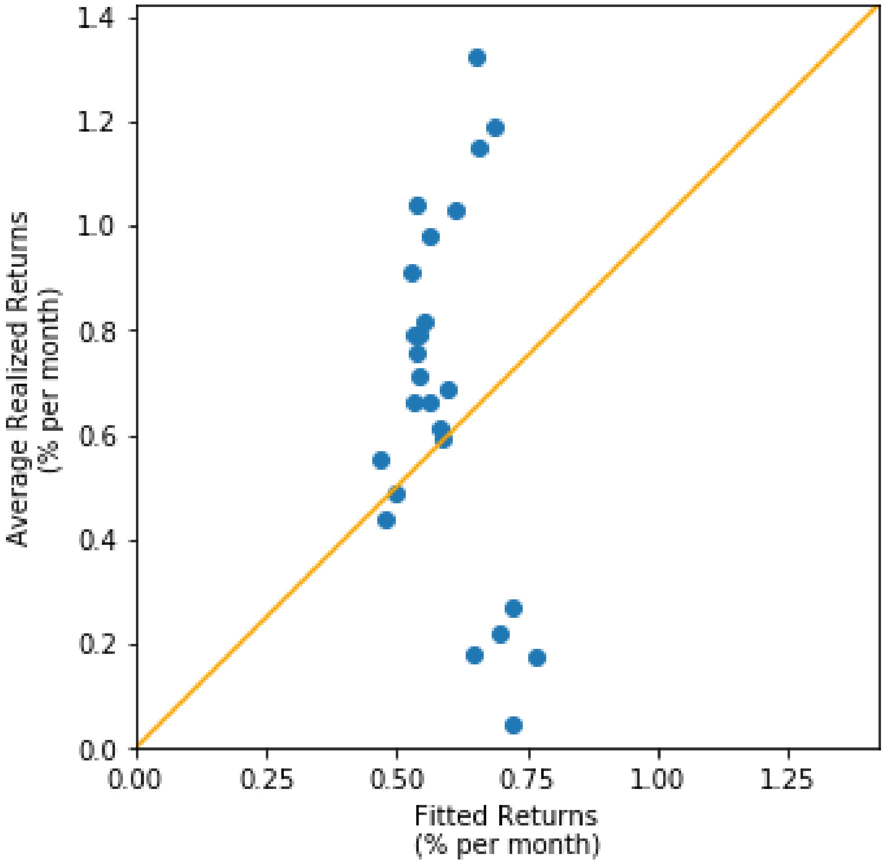
```
            omega_F = np.cov(x.T)
        omega_F_inv = np.linalg.inv(omega_F)
        num = np.matmul(alpha_hat.T,np.matmul(sigma_inv,alpha_hat))
        den = 1+np.matmul(mu_F.T,np.matmul(omega_F_inv,mu_F))
        chi2_statistic = (T*num/den)[0][0]
        if print_result:
            print('Chi-square test statistic:', np.round(chi2_statistic,2))
            print('P-value:',1-chi2.cdf(chi2_statistic, N))
```

In [6]:
```
## Problem 1-3 wrapper
def HW_question(covariables, question, print_OLS, print_latex_table):
    """
    Question 1: covariables=['Mkt-RF']
    Question 2: covariables=['Mkt-RF','SMB','HML']
    Question 3: covariables=['Mkt-RF','SMB','HML','MOM']
    question is a number between 1 and 3 for printing purposes
    """
    k = len(covariables)
    x = np.array(factors.loc[:,covariables])
    alpha = np.zeros((25,4))
    betas = np.zeros((k,25,4))
    mean_returns = np.zeros((25,1))
    excess_returns = np.zeros((25,len(x)))
    fitted_returns = np.zeros((25,len(x)))
    residuals = np.zeros((25,len(x)))

    # Regressions (parts a, b and d)
    for i in range(25):
        y = np.array(data.iloc[:,i+1]-data.iloc[:,26]).reshape(-1,1) # Second term is RF
        mean_returns[i] = np.mean(y)
        # White
        #xw_beta, xw_resids, xw_se, xw_cov, xw_pvalues = OLS(x, y, addcon=True, sig_leve
        # Newey-West
        xn_beta, xn_resids, xn_se, xn_cov, xn_pvalues  = OLS(x, y, addcon=True, cov_type
        # Alpha
        alpha_i = xn_beta[0]
        alpha_i_se = xn_se[0]
        alpha_i_pvalue = xn_pvalues[0]
        alpha[i] = np.array([i+1, alpha_i, alpha_i_se, alpha_i_pvalue])
        # Beta and residuals
        fit = alpha_i
        for kk in range(k):
            betas_i_kk = xn_beta[kk+1]
            betas_i_kk_se = xn_se[kk+1]
            betas_i_kk_pvalue = xn_pvalues[kk+1]
            betas[kk][i] = np.array([i+1, betas_i_kk, betas_i_kk_se, betas_i_kk_pvalue])
            fit += betas_i_kk*x[:,kk]
        excess_returns[i] = y.reshape(1,-1)
```

```
    fitted_returns[i] = fit.reshape(1,-1)
    residuals[i] = y.reshape(1,-1)-fit.reshape(1,-1)

# Output
alpha_table = pd.DataFrame(alpha, columns = ['i', 'coef', 's.e.', 'p-value'])
decimals = pd.Series([0, 2, 2, 4], index=['i', 'coef', 's.e.', 'p-value'])
if print_latex_table==1:
    print('\n alpha\n\n'+latex_table(alpha_table.round(decimals),caption="Results fo
betas_table = {}
for kk in range(k):
    betas_table[kk+1] = pd.DataFrame(betas[kk], columns = ['i', 'coef', 's.e.', 'p-v
    if print_latex_table==1:
        print('\n beta '+str(kk+1)+'\n\n'+latex_table(betas_table[kk+1].round(decima

# Fitted returns vs mean returns. Off sample prediction (part c)
betas_model = np.zeros((k,25,1))
fitted_returns = np.zeros((25,1))
for i in range(25):
    y = np.array(data.iloc[:,i+1]-data.iloc[:,26]).reshape(-1,1)
    xn_beta_is, xn_resids_is, xn_se_is, xn_cov_is, xn_pvalues_is  = OLS(x, y, addcon
    for kk in range(k):
        betas_model[kk][i] = xn_beta_is[kk]
for kk in range(k):
    risk_premium = np.mean(x[:,kk])
    beta_kk_coef = betas_model[kk]
    fitted_returns += beta_kk_coef*risk_premium
return_t = np.array(np.array(data.iloc[:,1:26])-np.array(data.iloc[:,26]).reshape(-1
mean_returns = np.mean(return_t,axis=0)
max_return = max([np.max(fitted_returns), np.max(mean_returns)])+0.1
line = np.linspace(0,max_return,100)
plt.figure(figsize=(5,5))
plt.scatter(fitted_returns, mean_returns)
plt.plot(line, line, 'orange')
plt.xlim([0,max_return])
plt.ylim([0,max_return])
plt.gca().set_aspect('equal', adjustable='box')
plt.rcParams.update({'font.size': 13})
plt.ylabel("Average Realized Returns\n(% per month)")
plt.xlabel("Fitted Returns\n(% per month)")
plt.tight_layout()
plt.savefig('Q'+str(question)+'c.eps', format='eps', dpi=1000)
plt.show()

# GRS (part e)
alpha_hat = np.array(alpha_table.loc[:,'coef']).reshape(-1,1)
GRS(alpha_hat, x, residuals, N=25, print_result=1)

return alpha_table, betas_table
```

0.1 Question 1

```
In [7]: alpha, betas = HW_question(covariables=['Mkt-RF'], question=1, print_OLS=0, print_latex_
```




```
Chi-square test statistic: 135.77
P-value: 0.0
```

```
In [8]: #Graph for part f
        market_y = np.mean(factors['Mkt-RF'])
        market_x = np.std(factors['Mkt-RF'])
        Rf_y = np.mean(factors['RF'])
        Rf_x = 0
        S = np.cov(data[data.columns[1:-1]].T)
        CML = lambda x : (market_y/market_x) * (x - Rf_x) + Rf_y
        portfolios = np.zeros((2,25))
        for counter in range(25):
            variable = data[data.columns[counter+1]]
            portfolios[0][counter] = np.mean(variable)
            portfolios[1][counter] = np.std(variable)
```

```
In [9]: #Efficient Frontier
        ef = EfficientFrontier(portfolios[0], S)
        return_range = np.arange(0.35,1.8,0.05)
        frontier_return = []
        frontier_vol = []
        for r in return_range:
            ef.efficient_return(target_return=r)
            frontier_return.append(ef.portfolio_performance()[0])
            frontier_vol.append(ef.portfolio_performance()[1])
        raw_weights = ef.max_sharpe()
        #cleaned_weights = ef.clean_weights()
        #print(cleaned_weights)
        ef.portfolio_performance(verbose=True)
        tan_y = ef.portfolio_performance()[0]
        tan_x = ef.portfolio_performance()[1]
        tangent_line = lambda x : ((tan_y-Rf_y)/(tan_x-Rf_x)) * (x - Rf_x) + Rf_y
```

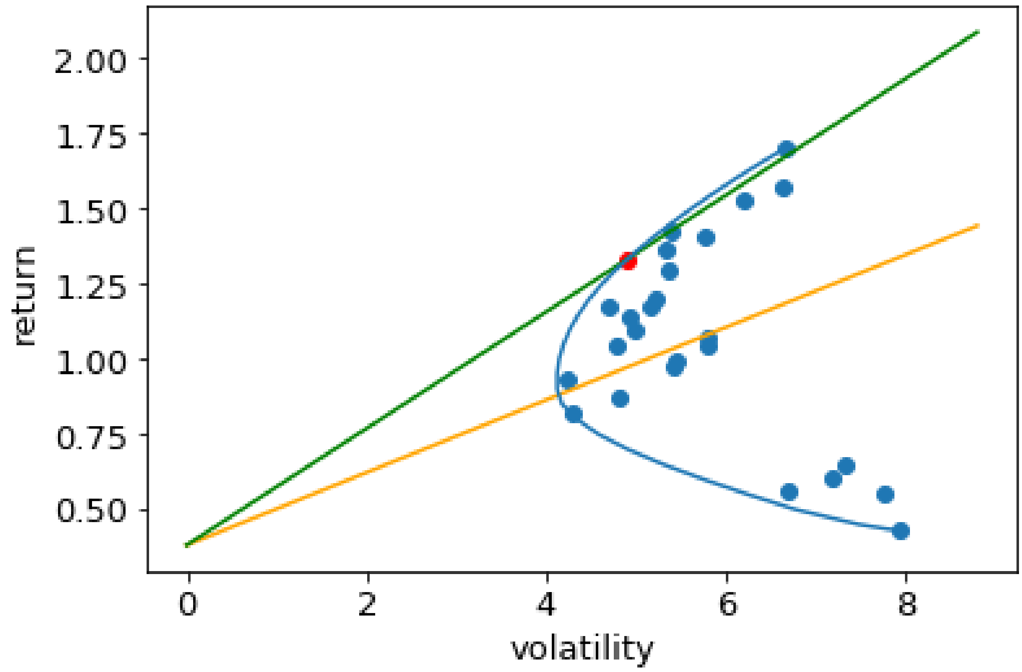
```
Expected annual return: 133.2%
Annual volatility: 490.5%
Sharpe Ratio: 0.27
```

```
In [10]: market_x
```

```
Out[10]: 4.388794451764749
```

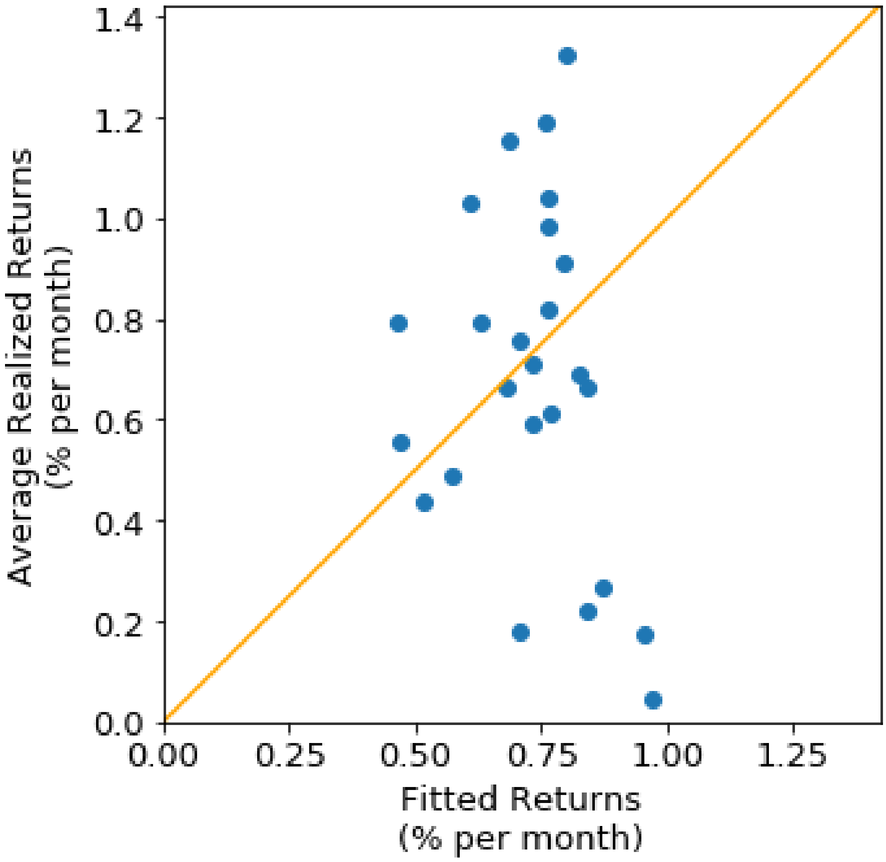
```
In [11]: plt.scatter(portfolios[1],portfolios[0])
         plt.scatter(tan_x,tan_y,color='red')
         xrange = np.arange(0,9,0.2)
         plt.plot(xrange, [CML(x) for x in xrange], color='orange')
         plt.plot(xrange, [tangent_line(x) for x in xrange], color='green')
         plt.plot(frontier_vol,frontier_return)
         plt.xlabel('volatility')
         plt.ylabel('return')
```

```
Out[11]: Text(0, 0.5, 'return')
```



0.2 Question 2

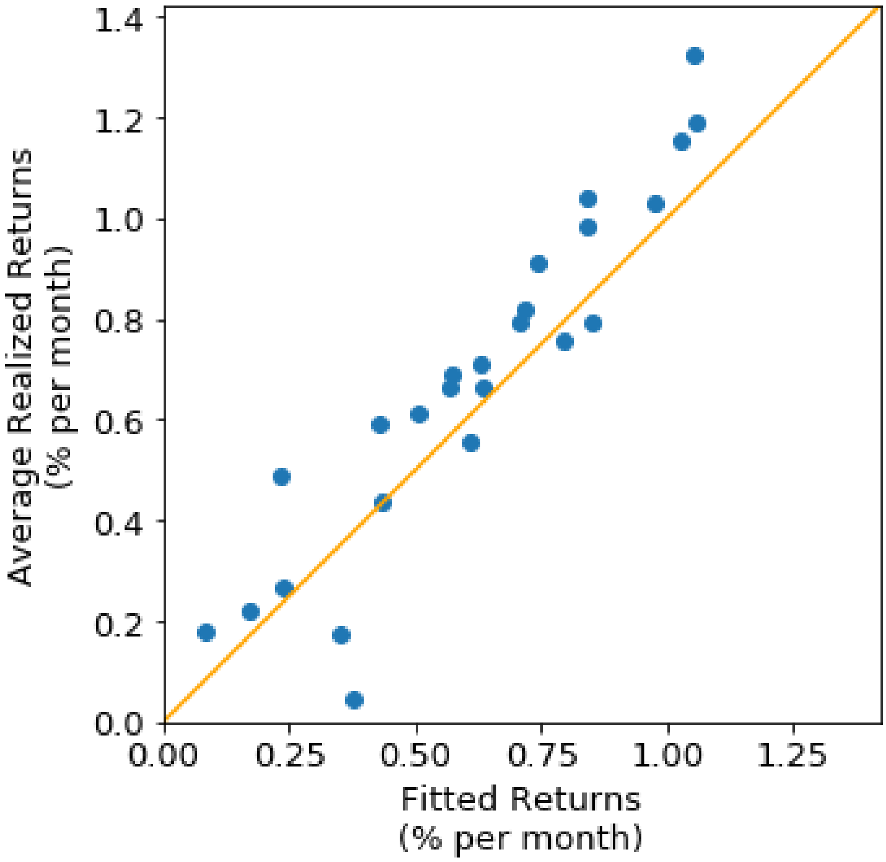
```
In [12]: alpha, betas = HW_question(covariables=['Mkt-RF','SMB','HML'], question=2, print_OLS=0,
```




```
Chi-square test statistic: 136.02
P-value: 0.0
```

0.3 Question 3

```
In [13]: alpha, betas = HW_question(covariables=['Mkt-RF','SMB','HML','MOM'], question=3, print_
```




```
Chi-square test statistic: 102.77
P-value: 2.1340151867832446e-11
```

0.4 Question 4

```
In [14]: covariables=['Mkt-RF','SMB','HML','MOM']
         FF4_Factors = np.array(factors.loc[:,covariables])
         alpha_coefs = alpha['coef']
         beta_1 = betas[1]['coef']
         beta_2 = betas[2]['coef']
         beta_3 = betas[3]['coef']
         beta_4 = betas[4]['coef']
         X = np.c_[beta_1, beta_2, beta_3, beta_4]
```

```
In [15]: avg_returns = []
         for i in range(25):
             returns = data.iloc[:,i+1]-data.iloc[:,26]
             avg_returns.append(np.mean(returns))
```

```
avg_returns = np.array(avg_returns)
xn_beta, xn_resids, xn_se, xn_cov, xn_pvalues  = OLS(X, avg_returns, addcon=True, summa
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.856
Model:                            OLS   Adj. R-squared:                  0.827
Method:                 Least Squares   F-statistic:                     29.77
Date:                Wed, 08 May 2019   Prob (F-statistic):           3.62e-08
Time:                        23:15:18   Log-Likelihood:                 16.334
No. Observations:                  25   AIC:                            -22.67
Df Residuals:                      20   BIC:                            -16.57
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1131      0.708      0.160      0.875      -1.364       1.590
x1             0.4866      0.649      0.750      0.462      -0.866       1.840
x2             0.2118      0.065      3.261      0.004       0.076       0.347
x3             0.5539      0.430      1.289      0.212      -0.342       1.450
x4             0.7403      0.075      9.834      0.000       0.583       0.897
==============================================================================
Omnibus:                        2.620   Durbin-Watson:                   1.325
Prob(Omnibus):                  0.270   Jarque-Bera (JB):                1.538
Skew:                          -0.599   Prob(JB):                        0.463
Kurtosis:                       3.207   Cond. No.                         56.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [16]:
```
P4_alphas = pd.DataFrame({'Estimated Alphas':xn_resids})
#print(latex_table(P4_alphas, caption="", label="Estimated Alphas", index=True))
P4_alphas
```

Out[16]:

	Estimated Alphas
0	-0.330152
1	-0.064008
2	0.065128
3	0.100153
4	0.222379
5	-0.146268
6	0.055824
7	-0.006174
8	0.031867
9	0.087408

```
10        0.073104
11        0.040083
12       -0.021674
13       -0.178996
14        0.069332
15        0.070439
16        0.105063
17       -0.009512
18       -0.027528
19       -0.023076
20        0.113714
21        0.210012
22       -0.096820
23       -0.181696
24       -0.158605
```

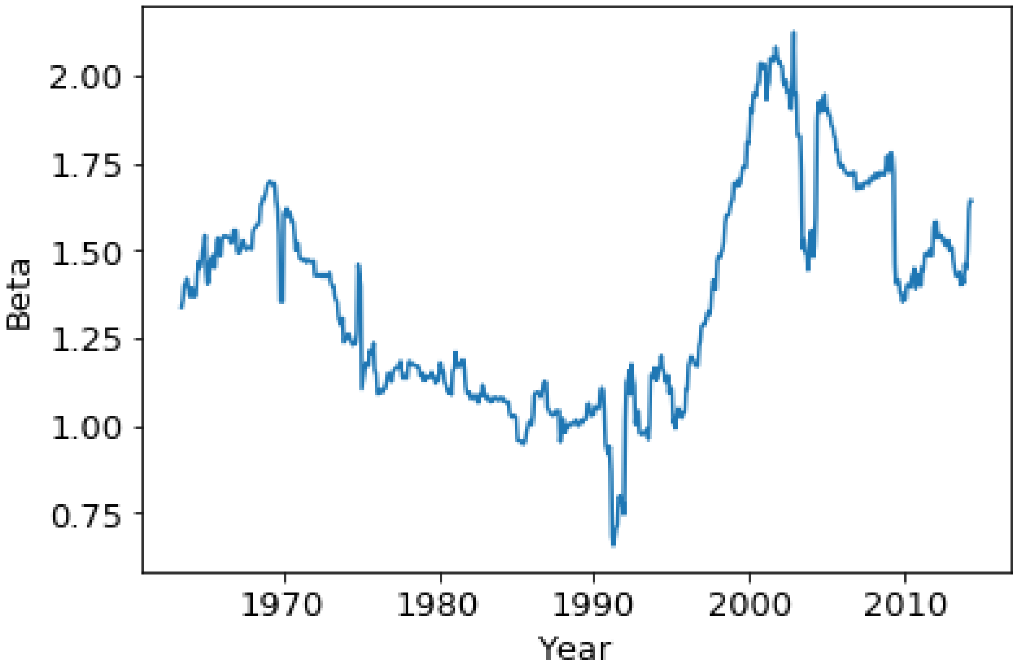
0.5 Problem 5

In [17]:
```
corner_upper_left = data['SMALL LoPRIOR']
corner_upper_right = data['SMALL HiPRIOR']
corner_lower_left = data['BIG LoPRIOR']
corner_lower_right = data['BIG HiPRIOR']
x = sm.add_constant(factors['Mkt-RF'])
```

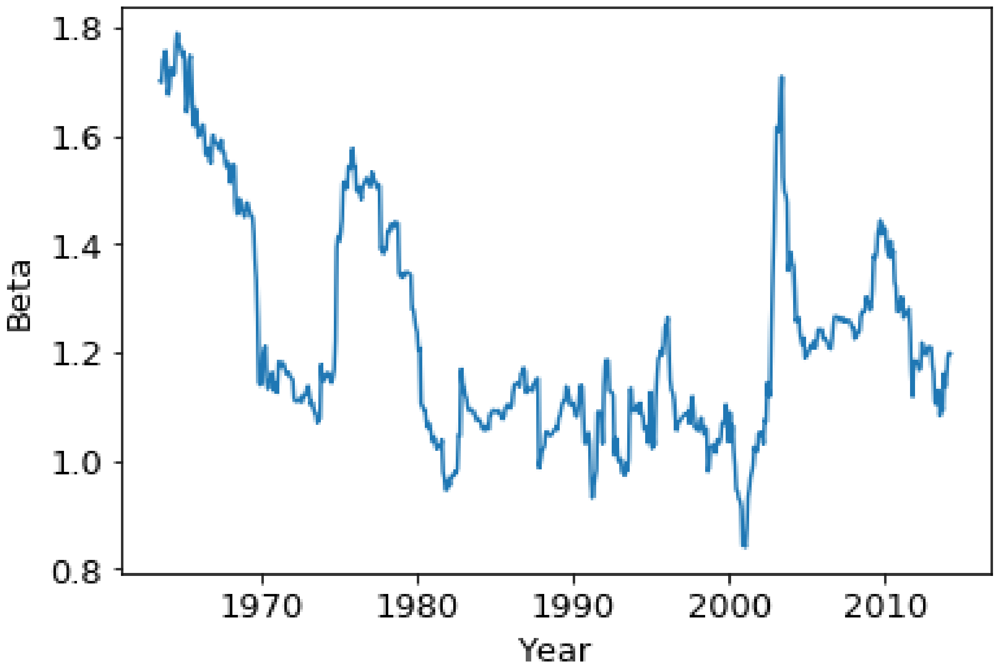
```
C:\Users\conan\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method
  return ptp(axis=axis, out=out, **kwargs)
```

In [18]:
```
#Run the rolling regressions
beta_upper_left = np.zeros(len(x)-60+1)
beta_upper_right = np.zeros(len(x)-60+1)
beta_lower_left = np.zeros(len(x)-60+1)
beta_lower_right = np.zeros(len(x)-60+1)
for counter in range(len(x)-60+1):
    reg_ul = sm.OLS(corner_upper_left[counter:60+counter],x[counter:60+counter]).fit()
    reg_ur = sm.OLS(corner_upper_right[counter:60+counter],x[counter:60+counter]).fit()
    reg_ll = sm.OLS(corner_lower_left[counter:60+counter],x[counter:60+counter]).fit()
    reg_lr = sm.OLS(corner_lower_right[counter:60+counter],x[counter:60+counter]).fit()
    beta_upper_left[counter] = reg_ul.params[1]
    beta_upper_right[counter] = reg_ur.params[1]
    beta_lower_left[counter] = reg_ll.params[1]
    beta_lower_right[counter] = reg_lr.params[1]
```

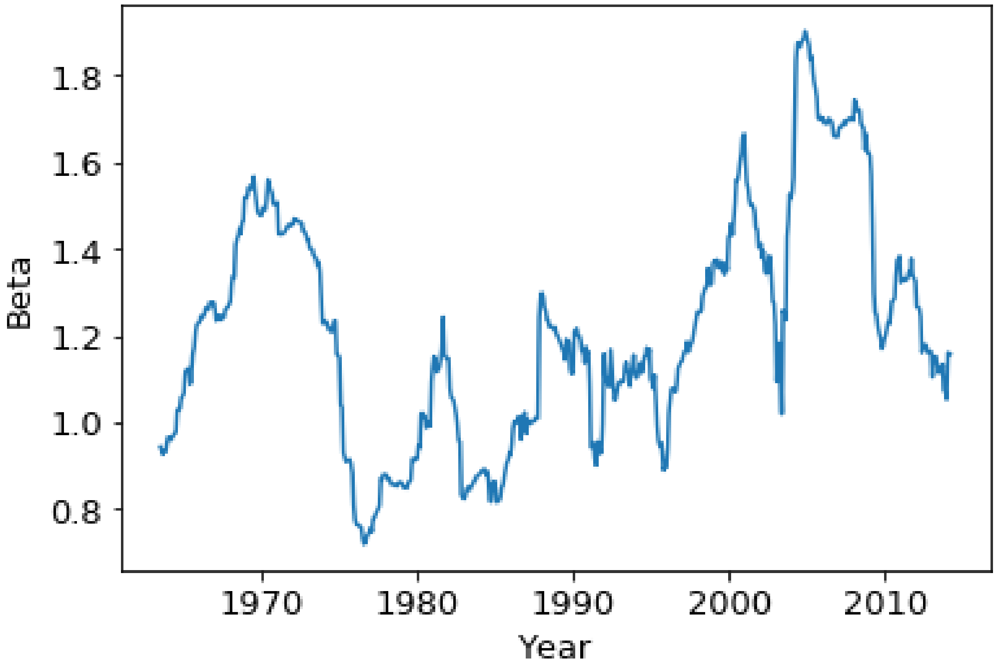
In [19]:
```
#Plot the betas
plt.plot(data['Date'][0:len(data)-60+1],beta_upper_left)
plt.xlabel('Year')
plt.ylabel('Beta')
plt.savefig('beta_SMALL LoPRIOR.png')
```



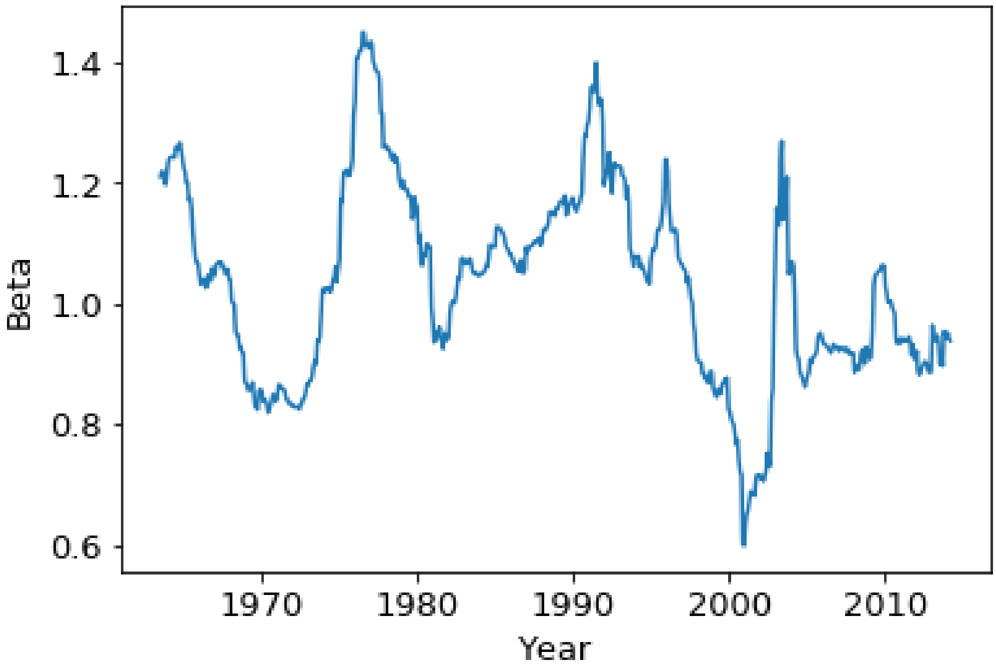

```
In [20]: plt.plot(data['Date'][0:len(data)-60+1],beta_upper_right)
         plt.xlabel('Year')
         plt.ylabel('Beta')
         plt.savefig('beta_SMALL HiPRIOR.png')
```



```
In [21]: plt.plot(data['Date'][0:len(data)-60+1],beta_lower_left)
         plt.xlabel('Year')
         plt.ylabel('Beta')
         plt.savefig('beta_BIG LoPRIOR.png')
```




```
In [22]: plt.plot(data['Date'][0:len(data)-60+1],beta_lower_right)
         plt.xlabel('Year')
         plt.ylabel('Beta')
         plt.savefig('beta_BIG HiPRIOR.png')
```



0.6 Problem 6

```
In [23]: covariables=['Mkt-RF','SMB','HML','MOM']
         FF4_Factors = np.array(factors.loc[:,covariables])
         all_returns = []
         for i in range(25):
             returns = data.iloc[:,i+1]-data.iloc[:,26]
             all_returns.append(returns)
         all_returns = np.array(all_returns)
```

```
In [24]: T = len(all_returns[0])
         FM_betas = []
         FM_lambdas = []
         FM_alphas = []
         for j in range(T-60):
             #60 beta estimates
             betas = []
             data60 = all_returns[0:len(all_returns), j:(60+j)]
             factors60 = FF4_Factors[j:(60+j), 0:len(FF4_Factors[0])]
             for i in range(25):
                 ols_beta_hat, ols_resids, ols_se, ols_cov_mat, ols_pvalues = OLS(factors60, dat
                 betas.append(ols_beta_hat)
             betas = np.array(betas)
             FM_betas.append(betas)
```

```
        #Lambda estimates
        R_61 = all_returns[0:len(all_returns), (60+j)]
        alphas = betas[:, 0]
        betas = betas[:, 1:len(betas[0])]
        ols_beta_hat, ols_resids, ols_se, ols_cov_mat, ols_pvalues = OLS(betas, R_61, addco
        FM_lambdas.append(ols_beta_hat)
        FM_alphas.append(ols_resids)
    FM_lambdas = np.array(FM_lambdas)
    FM_betas = np.array(FM_betas)
    FM_alphas = np.array(FM_alphas)
```

Fama-MacBeth Lamda Estimates

```
In [25]: FM_lambda_est = (1 / (T - 59)) * np.sum(FM_lambdas, axis = 0)
         FM_lambda_est

Out[25]: array([ 0.96628831, -0.35323413,  0.13772876, -0.08895711,  0.60973755])
```

Fama-MacBeth Lamda Covariance Matrix

```
In [26]: sum_err = np.zeros((5, 5))
         for i in range(len(FM_lambdas)):
             err = FM_lambdas[i] - FM_lambda_est
             sum_err = sum_err + np.matmul(err.reshape(len(err), 1), err.reshape(1, len(err)))
         FM_cov = (1 / (T - 59)**2) * sum_err
         FM_cov

Out[26]: array([[ 0.05642309, -0.0486933 ,  0.00369129, -0.00832966,  0.00102219],
                [-0.0486933 ,  0.0748223 ,  0.00215264, -0.00050352, -0.00590454],
                [ 0.00369129,  0.00215264,  0.01559285, -0.00258516, -0.0022236 ],
                [-0.00832966, -0.00050352, -0.00258516,  0.02455061, -0.0019045 ],
                [ 0.00102219, -0.00590454, -0.0022236 , -0.0019045 ,  0.03036843]])

In [27]: np.sqrt(np.diagonal(FM_cov))

Out[27]: array([0.23753546, 0.27353665, 0.12487135, 0.15668635, 0.17426541])
```

Fama-Macbeth Alpha Estimate

```
In [28]: FM_alpha_est = (1 / (T - 59)) * np.sum(FM_alphas, axis = 0)
         FM_alphas = pd.DataFrame({'Estimated Alphas':FM_alpha_est})
         #print(latex_table(FM_alphas, caption="", label="Estimated Alphas", index=True))
         FM_alphas

Out[28]:     Estimated Alphas
         0          -0.476330
         1          -0.094612
         2           0.062863
         3           0.059872
```

```
4           0.287965
5          -0.208161
6           0.014264
7           0.046741
8           0.086373
9           0.131586
10         -0.067525
11          0.037431
12          0.017274
13         -0.085734
14          0.072046
15          0.056594
16          0.151075
17          0.084502
18          0.024592
19          0.016560
20          0.107214
21          0.150633
22         -0.104543
23         -0.156942
24         -0.213739
```

In [29]:
```
#print(latex_table(abs(P4_alphas - FM_alphas), caption="", label="Alpha Differences", i
abs(P4_alphas - FM_alphas)
```

Out[29]:
```
    Estimated Alphas
0           0.146177
1           0.030604
2           0.002265
3           0.040281
4           0.065587
5           0.061893
6           0.041560
7           0.052915
8           0.054506
9           0.044177
10          0.140629
11          0.002652
12          0.038948
13          0.093262
14          0.002714
15          0.013846
16          0.046012
17          0.094014
18          0.052120
19          0.039636
20          0.006500
21          0.059379
```

22	0.007723
23	0.024753
24	0.055134