Graded Problem Set: Investment and Portfolio Management

Imperial College London - Business School

Rodolphe Lajugie

2025-10-31

Table of contents

List of Figures

List of Tables

In this report, all the computed values are monthly unless otherwise specified.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
```

```
df = pd.read_excel('../data/data_coursework1_Q1.xlsx')
df = df[['year_', 'month_', 'date_', '1-month Tbill', 'SP500', 'IBM']]

# Calculate returns
df['SP500_ret'] = df['SP500'].pct_change()
df['IBM_ret'] = df['IBM'].pct_change()

# Adjust T-bill to monthly rate and scale (if needed)
df['rft'] = df['1-month Tbill'] / 100

df = df.dropna(subset=['SP500_ret', 'IBM_ret', 'rft'])

# Excess returns
df['excess_stock'] = df['IBM_ret'] - df['rft']
df['excess_market'] = df['SP500_ret'] - df['rft']
df['date'] = pd.to_datetime(df['year_'].astype(
    str) + '-' + df['month_'].astype(str), format='%Y-%m')
df.drop(columns=['year_', 'month_', 'date_'], inplace=True)
```

Model 1: Simple CAPM Regression

munl computation first then with built-in functions to compare results.

```
y = df['excess_stock'].values
X = np.column_stack((np.ones(len(df)), df['excess_market'].values))
```

```
beta_hat = np.linalg.inv(X.T @ X) @ (X.T @ y)
alpha, beta = beta_hat
print('alpha:', alpha, 'beta:', beta)
```

```
alpha: 0.0027020468479849896 beta: 0.8856018941150132
```

builtin methods to validate results

```python
X1 = sm.add_constant(df['excess_market'])
model1 = sm.OLS(df['excess_stock'], X1).fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           excess_stock   R-squared:                       0.286
Model:                            OLS   Adj. R-squared:                  0.284
Method:                 Least Squares   F-statistic:                     138.4
Date:                Mon, 17 Nov 2025   Prob (F-statistic):           4.24e-
27
Time:                        18:04:49   Log-Likelihood:                 543.44
No. Observations:                 347   AIC:                            -
1083.
Df Residuals:                     345   BIC:                            -
1075.
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.0027      0.003      0.993      0.321      -0.003       0.008
excess_market     0.8856      0.075     11.766      0.000       0.738       1.034
==============================================================================
Omnibus:                        0.111   Durbin-Watson:                   2.120
Prob(Omnibus):                  0.946   Jarque-Bera (JB):                0.038
Skew:                           0.022   Prob(JB):                        0.981
Kurtosis:                       3.025   Cond. No.                         27.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
```

Part 2: non linear regression

Manual computation

```python
Dt = (df['excess_market'] > 0).astype(int).values
D_excess = Dt * df['excess_market'].values
nonD_excess = (1 - Dt) * df['excess_market'].values
excess_market_sq = df['excess_market'].values ** 2
```

```
X2 = np.column_stack(
    (np.ones(len(df)), D_excess, nonD_excess, excess_market_sq))
beta_hat2 = np.linalg.inv(X2.T @ X2) @ (X2.T @ y)
print('Model 2 coefficients:\nalpha: ', beta_hat2[0], ' beta1: ', beta_hat2[1],
      '\nbeta2: ', beta_hat2[2], ' beta3: ', beta_hat2[3])
```

```
Model 2 coefficients:
alpha:  -0.010031847996995746  beta1:  1.6230560811213248
beta2:  0.11105859440112376  beta3:  -6.095601716965916
```

With builtin functions

```
df['Dt'] = (df['excess_market'] > 0).astype(int)
df['D_excess'] = df['Dt'] * df['excess_market']
df['nonD_excess'] = (1 - df['Dt']) * df['excess_market']
df['excess_market_sq'] = df['excess_market']**2

df.head()
```

	1-month Tbill	SP500	IBM	SP500_ret	IBM_ret	rft	excess_stock	excess_market	dat
25	0.20	70.22	2.66	0.016650	-0.011152	0.0020	-0.013152	0.014650	196
26	0.20	70.29	2.64	0.000997	-0.007519	0.0020	-0.009519	-0.001003	196
27	0.22	68.05	2.25	-0.031868	-0.147727	0.0022	-0.149927	-0.034068	196
28	0.24	62.99	1.95	-0.074357	-0.133333	0.0024	-0.135733	-0.076757	196
29	0.20	55.63	1.68	-0.116844	-0.138462	0.0020	-0.140462	-0.118844	196

```
X2 = sm.add_constant(df[['D_excess', 'nonD_excess', 'excess_market_sq']])
model2 = sm.OLS(df['excess_stock'], X2).fit()
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           excess_stock   R-squared:                       0.299
Model:                            OLS   Adj. R-squared:                  0.293
Method:                 Least Squares   F-statistic:                     48.87
Date:                Mon, 17 Nov 2025   Prob (F-statistic):           2.53e-
26
Time:                        18:04:49   Log-Likelihood:                 546.66
No. Observations:                 347   AIC:                            -
1085.
```

```
Df Residuals:                     343   BIC:                            -
1070.
Df Model:                           3
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -0.0100      0.006     -1.757      0.080      -0.021       0.001
D_excess              1.6231      0.309      5.253      0.000       1.015       2.231
nonD_excess           0.1111      0.340      0.327      0.744      -0.558       0.780
excess_market_sq     -6.0956      3.162     -1.928      0.055     -12.315       0.124
==============================================================================
Omnibus:                        0.252   Durbin-Watson:                   2.115
Prob(Omnibus):                  0.882   Jarque-Bera (JB):                0.133
Skew:                           0.039   Prob(JB):                        0.936
Kurtosis:                       3.056   Cond. No.                     1.18e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
[2] The condition number is large, 1.18e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

F Test

Manual computation

```python
y_hat_unres = X2 @ beta_hat2
RSS_unres = np.sum((y - y_hat_unres) ** 2)
RSS_unres
```

```
np.float64(0.8699570012080915)
```

```python
D_total_excess = df['excess_market'].values
X2_res = np.column_stack((np.ones(len(df)), D_total_excess, excess_market_sq))
beta_hat2_res = np.linalg.inv(X2_res.T @ X2_res) @ (X2_res.T @ y)
y_hat_res = X2_res @ beta_hat2_res
RSS_res = np.sum((y - y_hat_res) ** 2)

RSS_res
```

```
np.float64(0.8845833266825458)
```

```python
q = 1
n = len(y)
k_unres = X2.shape[1]
F_stat = ((RSS_res - RSS_unres)/q) / (RSS_unres/(n - k_unres))
print('F statistic:', F_stat)
```

```
F statistic: 5.766755863532402
```

With builtin functions

```python
f_test_result = model2.f_test('D_excess = nonD_excess')
print(f_test_result)
```

```
<F test: F=5.766755863532396, p=0.01686377114880381, df_denom=343, df_num=1>
```

T-test

```python
y_hat1 = X @ beta_hat
residuals1 = y - y_hat1
sigma2 = np.sum(residuals1 ** 2) / (len(y) - X.shape[1])
cov_beta_hat = sigma2 * np.linalg.inv(X.T @ X)
se_alpha = np.sqrt(cov_beta_hat[0, 0])
t_stat_alpha = alpha / se_alpha
print('t statistic for alpha:', t_stat_alpha)
```

```
t statistic for alpha: 0.9930805477699152
```

```python
t_test_alpha = model1.t_test('const = 0')
print(t_test_alpha)
```

```
                             Test for Constraints
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0             0.0027      0.003      0.993      0.321      -0.003       0.008
==============================================================================
```

```python
t_test_alpha2 = model2.t_test('const = 0')
print(t_test_alpha2)
```

```
                             Test for Constraints
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0            -0.0100      0.006     -1.757      0.080      -0.021       0.001
==============================================================================
```