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

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import itertools
```

```
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)
df
```

	1-month Tbill	SP500	IBM	SP500_ret	IBM_ret	rft	excess_stock	excess_market	d
25	0.20	70.22	2.66	0.016650	-0.011152	0.0020	-0.013152	0.014650	19
26	0.20	70.29	2.64	0.000997	-0.007519	0.0020	-0.009519	-0.001003	19
27	0.22	68.05	2.25	-0.031868	-0.147727	0.0022	-0.149927	-0.034068	19
28	0.24	62.99	1.95	-0.074357	-0.133333	0.0024	-0.135733	-0.076757	19
29	0.20	55.63	1.68	-0.116844	-0.138462	0.0020	-0.140462	-0.118844	19
...	...	...	...	...	...	...	...	...	..
367	0.66	330.70	19.89	-0.081389	-0.076173	0.0066	-0.082773	-0.087989	19
368	0.60	315.40	20.77	-0.046265	0.044243	0.0060	0.038243	-0.052265	19
369	0.68	307.10	20.60	-0.026316	-0.008185	0.0068	-0.014985	-0.033116	19
370	0.57	315.20	22.43	0.026376	0.088835	0.0057	0.083135	0.020676	19
371	0.60	328.70	22.31	0.042830	-0.005350	0.0060	-0.011350	0.036830	19

Model 1: Simple CAPM Regression

Part 1: linear regression

Manual computation

We know that $\hat{\beta} = (X'X)^{-1}X'y$ Where X is the design matrix and y the response variable.

```
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
```

With builtin functions

```
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:                Tue, 18 Nov 2025   Prob (F-statistic):           4.24e-
27
Time:                        17:05:20   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
```

Using the OLS function from the statsmodels library or by computing it manually, we obtain the following results for the CAPM regression of IBM's excess returns against the market's excess returns: - Intercept (α): 0.0027 - Slope (β): 0.8856

α is close to zero which means IBM doesn't outperform or underperform the market on average. The β of 0.89 indicates that IBM is less volatile than the market.

Part 2: 2nd linear regression

We want to find α, β_1, β_2 and β_3 such that:

$$r_{it} - r_{ft} = \alpha + \beta_1[D_t(r_{mt} - r_{ft})] + \beta_2[(1 - D_t)(r_{mt} - r_{ft})] + \beta_3(r_{mt} - r_{ft})^2 + u_t$$

Where \$D_t = 1 \$ if the market excess return is positive and 0 otherwise.

Manual computation

```
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
```

Again using the formula $\hat{\beta} = (X'X)^{-1}X'y$ we obtain:

```
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

```
X3 = sm.add_constant(df[['D_excess', 'nonD_excess', 'excess_market_sq']])
model2 = sm.OLS(df['excess_stock'], X3).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:                Tue, 18 Nov 2025   Prob (F-statistic):           2.53e-
26
Time:                        17:05:20   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.
```

The second model allows different market exposures depending on whether excess market returns are positive or negative. The high value for beta1 relative to beta2 suggests IBM's response to positive market movements is much larger than to negative movements. The negative beta3 indicates a possible concavity in the relationship, showing diminishing marginal response for larger market returns. R^2 has improved slightly from the first model, indicating a better fit.

F Test

Here, we want to test the null hypothesis $H_0 : \beta_1 = \beta_2$

Manual computation

$\hat{y} = X\hat{\beta}$

$((((RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2))))$

```
R= np.array([[0, 1, -1,0]])
r = 0

var1 = np.linalg.inv(X2.T @ X2)
var_R = np.linalg.inv(R @ var1 @ R.T)

e = df['excess_stock'].values - X2 @ beta_hat2
T = X2.shape[0]
e_eprime = (e.T @ e)/(T-4)
```

```
F_stats = ((R @ beta_hat2 - r)*var_R * (R @ beta_hat2 - r).T) / 1 / e_eprime
print('F statistic:', F_stats)
```

```
F statistic: [[5.76675586]]
```

```
from scipy import stats

stats.f.sf(F_stats, 1, T - 4)
```

```
array([[0.01686377]])
```

With builtin functions

```
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

```
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
```

```
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
==============================================================================
```

```
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
==============================================================================
```

Question 2

```
df = pd.read_excel('../data/data_coursework1_Q2.xlsx', sheet_name="final_m")
```

Let's use EMU stock total excess return as our Rt as defined in the question. We can then compute Y as follows:

$$Y_t = \begin{cases} 1 & \text{if } R_t > 0 \\ 0 & \text{otherwise} \end{cases}$$

```
R = df['ESR']
df['Y'] = (R > 0).astype(int)

predictor_candidates = ['USR', 'EIF', 'ERECB',
                        'EFX', 'EDY', 'ESP', 'UIF', 'URR', 'UDY', 'UOIL']

df['Y_next'] = df['Y'].shift(-1)
data = df.dropna().copy() #because shift
```

The regression model is then defined as:

$$Y_{t+1} = \beta' X_t + \epsilon_t$$

```
df['Y'] = (df['ESR'] > 0).astype(int)
df['Y_next'] = df['Y'].shift(-1)

predictors = ['USR', 'EIF', 'ERECB', 'EFX',
              'EDY', 'ESP', 'UIF', 'URR', 'UDY', 'UOIL']

# lagged
for col in predictors:
```

```
    df[col + '_lag'] = df[col].shift(1)

# cleaned dataset
data = df.dropna().copy()

X = data[[p + '_lag' for p in predictors]]
X = sm.add_constant(X)
y = data['Y_next']

# OLS estimation
model_full = sm.OLS(y, X).fit()
print(model_full.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Y_next   R-squared:                       0.102
Model:                            OLS   Adj. R-squared:                  0.070
Method:                 Least Squares   F-statistic:                     3.152
Date:                Tue, 18 Nov 2025   Prob (F-statistic):           0.000771
Time:                        17:07:37   Log-Likelihood:                -
190.36
No. Observations:                 289   AIC:                             402.7
Df Residuals:                     278   BIC:                             443.1
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3795      0.313      1.212      0.227      -0.237       0.996
USR_lag        0.5042      0.639      0.789      0.431      -0.754       1.762
EIF_lag        7.6867      3.966      1.938      0.054      -0.122      15.495
ERECB_lag     -0.6023      0.153     -3.936      0.000      -0.904      -
0.301
EFX_lag        0.1618      0.899      0.180      0.857      -1.608       1.932
EDY_lag        0.6529      0.297      2.197      0.029       0.068       1.238
ESP_lag       -0.0573      0.040     -1.450      0.148      -0.135       0.020
UIF_lag      -12.1568      3.824     -3.179      0.002     -19.684      -
4.629
URR_lag        0.4050      0.096      4.235      0.000       0.217       0.593
UDY_lag        0.0902      0.164      0.551      0.582      -0.232       0.413
UOIL_lag      -0.4010      0.291     -1.378      0.169      -0.974       0.172
==============================================================================
Omnibus:                     3917.697   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               31.160
```

```
Skew:                          -0.299   Prob(JB):                     1.71e-
07
Kurtosis:                       1.506   Cond. No.                       479.
==============================================================================

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

About the Z score computation, we define a function compute_Z such that:

$$Z_t(\alpha) = \begin{cases} 1 & \text{if } \hat{P}(R_t > 0 \mid X_{t-1}) > \alpha \text{ and } R_t > 0 \\ 1 & \text{if } \hat{P}(R_t > 0 \mid X_{t-1}) \leq \alpha \text{ and } R_t \leq 0 \\ 0 & \text{Otherwise} \end{cases}$$

For a fixed $\alpha \in \ ]0;1[$

$$Z(\alpha) = \frac{\sum_{t=2}^{n} Z_t(\alpha)}{n-1}$$

Where TPR is the true positive rate and FPR the false positive rate.

```python
def compute_Z(y_true, p_hat, alpha):
    Z = np.where((p_hat > alpha) & (y_true == 1), 1,
                 np.where((p_hat <= alpha) & (y_true == 0), 1, 0))
    return Z.mean()
```

```python
best_Z = -1 #initialize so that any Z_val is better but can't be worse
best_predictors = None
results = []

for k in range(1, 11): #all combos possible of size 1 to 10
    for subset in itertools.combinations(predictor_candidates, k):

        X = data[list(subset)]
        y = data['Y_next']

        model = sm.OLS(y, X).fit()
        p_hat = model.predict(X)

        Z_val = compute_Z(y, p_hat, alpha=0.5) #first set (question specify it)
        results.append((subset, Z_val))

        if Z_val > best_Z:
```

```
            best_Z = Z_val
            best_predictors = subset
```

```
df_results = pd.DataFrame([
    {
        "Subset": ", ".join(subset),
        "Subset_size": len(subset),
        "Z_score": z
    }
    for subset, z in results
])

# Tri du plus grand au plus petit Z
df_results = df_results.sort_values(
    "Z_score", ascending=False).reset_index(drop=True)

# Affichage
df_results.head(10)
```

Table 3: Top 10 combinations of predictors by Z score

	Subset	Subset_size	Z_score
0	USR, ERECB, EFX, EDY, ESP, UIF, URR, UDY	8	0.667820
1	USR, EIF, ERECB, EDY, UIF, URR, UOIL	7	0.664360
2	USR, EIF, ERECB, EDY, ESP, UIF, URR, UOIL	8	0.664360
3	USR, ERECB, EFX, EDY, UIF, URR	6	0.664360
4	USR, EIF, ERECB, EFX, EDY, UIF, URR, UDY	8	0.664360
5	USR, EIF, ERECB, EFX, EDY, ESP, UIF, URR, UOIL	9	0.660900
6	USR, EIF, EFX, EDY, ESP, UIF, URR, UOIL	8	0.660900
7	USR, EIF, ERECB, EFX, EDY, UIF, URR	7	0.660900
8	USR, ERECB, EFX, EDY, UIF, URR, UOIL	7	0.660900
9	USR, ERECB, EDY, ESP, UIF, URR, UDY, UOIL	8	0.657439

```
print("Best predictor combination (alpha = 0.5):\n", best_predictors)
print("Achieved Z(alpha):", best_Z)
```

```
Best predictor combination (alpha = 0.5):
 ('USR', 'ERECB', 'EFX', 'EDY', 'ESP', 'UIF', 'URR', 'UDY')
Achieved Z(alpha): 0.6678200692041523
```

```
X_best = data[list(best_predictors)]
y = data['Y_next']

model = sm.OLS(y, X_best).fit()
p_hat = model.predict(X_best)

alphas = np.linspace(0.00001, 0.99999, 99999) #increase for better precision but takes t
Z_list = [compute_Z(y, p_hat, a) for a in alphas]

alpha_best = alphas[np.argmax(Z_list)]
Z_best_alpha = max(Z_list)

print("Optimal alpha:", alpha_best)
print("Z(alpha*) =", Z_best_alpha)
```

```
Optimal alpha: 0.50102
Z(alpha*) = 0.6747404844290658
```