Homework2

April 6, 2024

1 Homework 2 by Wei Minn

1.1 Question 1 Findings

Vanilla OLS coefficients, OLS coefficients with White SE, OLS coefficients Clustered by Firm and Year, and Fama-Macbeth coefficients produce roughly the same Standard Errors ($\simeq 0.028$ for the constant and $\simeq 0.28$ for the independent variable) and T-statistics ($\simeq 1.05$ for the constant and $\simeq 36.2$ for the independent variable).

On the other hand, OLS coefficients Clustered by Firm and OLS coefficients Clustered by Year produce roughly the same Standard Errors ($\simeq 0.023$ for the constant and $\simeq 0.033$ for the independent variable) and T-statistics ($\simeq 1.27$ for the constant and $\simeq 30.99$ for the independent variable).

The code for processing the data and producing the coefficients and standard errors are shown in the below sections.

1.1.1 Import Libraries

[ ]:
```
import numpy as np
import pandas as pd
import re
import statsmodels.api as sm
```

1.1.2 Import Data

[ ]:
```
data = []
with open("test_data.txt", "r") as file:
  for line in file:
    # Process each line here
    _line = line.strip()
    __line = re.sub(r"\s+", ',', line)
    splitted = __line.split(',')
    data.append(splitted[1:-1])

df = pd.DataFrame(np.array(data, dtype=float))
df
```

[ ]:

	0	1	2	3
0	1.0	1.0	-1.113973	2.251535

```
1       1.0   2.0 -0.080854  1.242346
2       1.0   3.0 -0.237607 -1.426376
3       1.0   4.0 -0.152486 -1.109394
4       1.0   5.0 -0.001426  0.914686
…       …     …       …       …
4995  500.0   6.0 -0.077057  3.720502
4996  500.0   7.0  0.218847  0.559121
4997  500.0   8.0 -0.155530 -3.766785
4998  500.0   9.0 -0.040172  0.903354
4999  500.0  10.0 -0.001172 -0.529761

[5000 rows x 4 columns]
```

Save Data

[ ]:
```
df.to_csv('./hw2.csv')
```

1.1.3 OLS Coefficents and Standard Errors

[ ]:
```
X = df[2].to_numpy()
X = sm.add_constant(X)
y = df[3].to_numpy()
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.208
Model:                            OLS   Adj. R-squared:                  0.208
Method:                 Least Squares   F-statistic:                     1311.
Date:                Fri, 05 Apr 2024   Prob (F-statistic):          4.25e-255
Time:                        22:55:34   Log-Likelihood:                -10573.
No. Observations:                5000   AIC:                         2.115e+04
Df Residuals:                    4998   BIC:                         2.116e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0297      0.028      1.047      0.295      -0.026       0.085
x1             1.0348      0.029     36.204      0.000       0.979       1.091
==============================================================================
Omnibus:                        4.912   Durbin-Watson:                   1.096
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                4.862
Skew:                           0.070   Prob(JB):                       0.0880
Kurtosis:                       3.063   Cond. No.                         1.01
==============================================================================
```

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

1.1.4 OLS Coefficents and White Standard Errors

[ ]:
```
X = df[2].to_numpy()
X = sm.add_constant(X)
y = df[3].to_numpy()
model = sm.OLS(y, X)
results = model.fit(cov_type='HC0')
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.208
Model:                            OLS   Adj. R-squared:                  0.208
Method:                 Least Squares   F-statistic:                     1329.
Date:                Fri, 05 Apr 2024   Prob (F-statistic):          3.48e-258
Time:                        22:56:06   Log-Likelihood:                -10573.
No. Observations:                5000   AIC:                         2.115e+04
Df Residuals:                    4998   BIC:                         2.116e+04
Df Model:                           1
Covariance Type:                  HC0
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0297      0.028      1.047      0.295      -0.026       0.085
x1             1.0348      0.028     36.451      0.000       0.979       1.090
==============================================================================
Omnibus:                        4.912   Durbin-Watson:                   1.096
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                4.862
Skew:                           0.070   Prob(JB):                       0.0880
Kurtosis:                       3.063   Cond. No.                         1.01
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC0)
```

1.1.5 OLS Coefficents and Standard Errors Clustered by Year

[ ]:
```
ye_X = sm.add_constant(df[2])
year_dummies = pd.get_dummies(df, columns=[1], drop_first=True)  # Avoid␣
 ↪collinearity
ye_X = pd.concat([ye_X, year_dummies], axis=1)
ye_X = ye_X.replace({True: 1, False: 0})
ye_X
```

```
/tmp/ipykernel_72982/1769855915.py:4: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  ye_X = ye_X.replace({True: 1, False: 0})
```

[ ]:
```
      const         2      0         2         3  1_2.0  1_3.0  1_4.0  1_5.0  \
0       1.0 -1.113973    1.0 -1.113973  2.251535      0      0      0      0
1       1.0 -0.080854    1.0 -0.080854  1.242346      1      0      0      0
2       1.0 -0.237607    1.0 -0.237607 -1.426376      0      1      0      0
3       1.0 -0.152486    1.0 -0.152486 -1.109394      0      0      1      0
4       1.0 -0.001426    1.0 -0.001426  0.914686      0      0      0      1
…     …        …    …        …       …    …    …    …    …
4995    1.0 -0.077057  500.0 -0.077057  3.720502      0      0      0      0
4996    1.0  0.218847  500.0  0.218847  0.559121      0      0      0      0
4997    1.0 -0.155530  500.0 -0.155530 -3.766785      0      0      0      0
4998    1.0 -0.040172  500.0 -0.040172  0.903354      0      0      0      0
4999    1.0 -0.001172  500.0 -0.001172 -0.529761      0      0      0      0

      1_6.0  1_7.0  1_8.0  1_9.0  1_10.0
0         0      0      0      0       0
1         0      0      0      0       0
2         0      0      0      0       0
3         0      0      0      0       0
4         0      0      0      0       0
…     …     …    …    …      …
4995      1      0      0      0       0
4996      0      1      0      0       0
4997      0      0      1      0       0
4998      0      0      0      1       0
4999      0      0      0      0       1

[5000 rows x 14 columns]
```

[ ]:
```
year_clusters = df[1]
ye_model = sm.OLS(df[3], ye_X)
ye_results = model.fit(cov_type='cluster', cov_kwds={'groups': year_clusters})

print("Parameters: [Constant IndependentVariable]")
print("Coefficients:", ye_results.params)
print("Standard Error:", ye_results.bse)
print("T-statistic:", ye_results.tvalues)
print("R-Squared:", ye_results.rsquared)
```

```
Parameters: [Constant IndependentVariable]
Coefficients: [0.02967972 1.03483344]
Standard Error: [0.02338672 0.03338891]
T-statistic: [ 1.26908428 30.99332495]
```

R-Squared: 0.20776573087275374

1.1.6 OLS Coefficents and Standard Errors Clustered by Firm

[ ]:
```
fe_X = sm.add_constant(df[2])
firm_dummies = pd.get_dummies(df, columns=[0], drop_first=True)  # Avoid␣
 ↪collinearity
fe_X = pd.concat([fe_X, firm_dummies], axis=1)
fe_X = fe_X.replace({True: 1, False: 0})
fe_X
```

```
/tmp/ipykernel_72982/3006770163.py:4: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  fe_X = fe_X.replace({True: 1, False: 0})
```

[ ]:

	const	2	1	2	3	0_2.0	0_3.0	0_4.0	0_5.0	\
0	1.0	-1.113973	1.0	-1.113973	2.251535	0	0	0	0	
1	1.0	-0.080854	2.0	-0.080854	1.242346	0	0	0	0	
2	1.0	-0.237607	3.0	-0.237607	-1.426376	0	0	0	0	
3	1.0	-0.152486	4.0	-0.152486	-1.109394	0	0	0	0	
4	1.0	-0.001426	5.0	-0.001426	0.914686	0	0	0	0	
…	…	…	…	…	…	…	…	…	…	
4995	1.0	-0.077057	6.0	-0.077057	3.720502	0	0	0	0	
4996	1.0	0.218847	7.0	0.218847	0.559121	0	0	0	0	
4997	1.0	-0.155530	8.0	-0.155530	-3.766785	0	0	0	0	
4998	1.0	-0.040172	9.0	-0.040172	0.903354	0	0	0	0	
4999	1.0	-0.001172	10.0	-0.001172	-0.529761	0	0	0	0	

	0_6.0	…	0_491.0	0_492.0	0_493.0	0_494.0	0_495.0	0_496.0	\
0	0	…	0	0	0	0	0	0	
1	0	…	0	0	0	0	0	0	
2	0	…	0	0	0	0	0	0	
3	0	…	0	0	0	0	0	0	
4	0	…	0	0	0	0	0	0	
…	…	…	…	…	…	…	…	…	
4995	0	…	0	0	0	0	0	0	
4996	0	…	0	0	0	0	0	0	
4997	0	…	0	0	0	0	0	0	
4998	0	…	0	0	0	0	0	0	
4999	0	…	0	0	0	0	0	0	

	0_497.0	0_498.0	0_499.0	0_500.0
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

```
4            0        0        0        0
…        …       …        …        …
4995         0        0        0        1
4996         0        0        0        1
4997         0        0        0        1
4998         0        0        0        1
4999         0        0        0        1

[5000 rows x 504 columns]
```

[ ]:
```
fe_clusters = df[1]
fe_model = sm.OLS(df[3], fe_X)
fe_results = model.fit(cov_type='cluster', cov_kwds={'groups': fe_clusters})

print("Parameters: [Constant IndependentVariable]")
print("Coefficients:", fe_results.params)
print("Standard Error:", fe_results.bse)
print("T-statistic:", fe_results.tvalues)
print("R-Squared:", fe_results.rsquared)
```

```
Parameters: [Constant IndependentVariable]
Coefficients: [0.02967972 1.03483344]
Standard Error: [0.02338672 0.03338891]
T-statistic: [ 1.26908428 30.99332495]
R-Squared: 0.20776573087275374
```

1.1.7 OLS Coefficents and Standard Errors Clustered by Firm-Year

[ ]:
```
fy_clusters = pd.DataFrame({'cluster': df[[0, 1]].apply(tuple, 1)})
fy_X = df[2].to_numpy()
fy_X = sm.add_constant(fy_X)
fy_y = df[3].to_numpy()
fy_model = sm.OLS(fy_y, fy_X, clusters=fy_clusters.cluster)
fy_results = fy_model.fit()
print(fy_results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.208
Model:                            OLS   Adj. R-squared:                  0.208
Method:                 Least Squares   F-statistic:                     1311.
Date:                Fri, 05 Apr 2024   Prob (F-statistic):          4.25e-255
Time:                        23:00:14   Log-Likelihood:                -10573.
No. Observations:                5000   AIC:                         2.115e+04
Df Residuals:                    4998   BIC:                         2.116e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
```

```
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0297      0.028      1.047      0.295      -0.026       0.085
x1             1.0348      0.029     36.204      0.000       0.979       1.091
==============================================================================
Omnibus:                        4.912   Durbin-Watson:                   1.096
Prob(Omnibus):                  0.086   Jarque-Bera (JB):                4.862
Skew:                           0.070   Prob(JB):                       0.0880
Kurtosis:                       3.063   Cond. No.                         1.01
==============================================================================

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

```
/home/weiminn/Documents/FDA/venv/lib/python3.11/site-
packages/statsmodels/base/model.py:130: ValueWarning: unknown kwargs
['clusters']
  warnings.warn(msg, ValueWarning)
```

1.1.8 Fama-Macbeth Coefficents and Standard Errors

[ ]:
```
unique_t = df[1].unique()
time_groups = {}
fb_regresults = []

for t in unique_t:
    time_groups[t] = df.loc[df[1]==t]

    fb_X = df[2].to_numpy()
    fb_X = sm.add_constant(fb_X)
    fb_y = df[3].to_numpy()
    fb_model = sm.OLS(fb_y, fb_X)
    fb_results = fb_model.fit()
    fb_regresults.append(fb_results)
```

[ ]:
```
coeffs = np.array([fb.params for fb in fb_regresults])
tstats = np.array([fb.tvalues for fb in fb_regresults])
stderr = np.array([fb.bse for fb in fb_regresults])

avg_coeffs = np.mean(coeffs, axis=0)
avg_tstats = np.mean(tstats, axis=0)
avg_stderr = np.mean(stderr, axis=0)
```

[ ]:
```
print("Parameters: [Constant IndependentVariable]")
print("Average Coefficients:", avg_coeffs)
print("Average T-Statistic:", avg_tstats)
print("Average Standard Error:", avg_stderr)
```

```
print("R-Squared:", fb_results.rsquared)
```

```
Parameters: [Constant IndependentVariable]
Average Coefficients: [0.02967972 1.03483344]
Average T-Statistic: [ 1.04655977 36.20414301]
Average Standard Error: [0.02835932 0.02858329]
R-Squared: 0.20776573087275374
```