In [8]:

```
import os

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import matplotlib
from statsmodels.iolib.summary2 import summary_col
```

In [9]:

```
niodata = pd.read_csv("NIO.csv")
PGdata = pd.read_csv("PG.csv")
EFFR = pd.read_csv("EFFRX.csv")
Sdata = pd.read_csv("SP500X.csv")
```

In [10]:

```
niodata["Date"] = pd.to_datetime(niodata["Date"], dayfirst = True)
PGdata["Date"] = pd.to_datetime(PGdata["Date"], dayfirst = True)
EFFR["DATE"] = pd.to_datetime(EFFR["DATE"], dayfirst = True)
Sdata["DATE"] = pd.to_datetime(Sdata["DATE"], dayfirst = True)
```

In [11]:

```
EFFR.set_index('DATE',inplace=True)
Sdata.set_index('DATE',inplace=True)
```

In [12]:

```
niodata.head()
```

Out[12]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-09-03	2.82	2.83	2.58	2.60	2.60	26854400
1	2019-09-04	2.70	2.75	2.64	2.75	2.75	15177900
2	2019-09-05	2.81	3.00	2.80	2.95	2.95	23244700
3	2019-09-06	2.99	3.00	2.87	2.98	2.98	14509300
4	2019-09-09	3.00	3.14	2.93	3.14	3.14	17386300

In [13]:

```
niodata.reset_index(drop=True, inplace=True)
PGdata.reset_index(drop=True, inplace=True)
EFFR.reset_index(drop=True, inplace=True)
Sdata.reset_index(drop=True, inplace=True)
```

In [14]:

```
niodata.rename(columns={'Close': 'Nio'}, inplace=True)

PGdata.rename(columns={'Close': 'PG'}, inplace=True)
```

In [15]:

```
niodata = niodata.loc[:,['Nio']]
```

In [16]:

```
MAdata = pd.concat([niodata, PGdata, EFFR, Sdata], axis = 1)
```

In [17]:

```
MAdata.head()
```

Out[17]:

	Nio	Date	Open	High	Low	PG	Adj Close	Volume	EFFR
0	2.60	2019-09-03	119.790001	121.540001	119.629997	121.360001	117.683128	4959300.0	2.13
1	2.75	2019-09-04	121.849998	123.279999	121.459999	123.209999	119.477066	5586900.0	2.13
2	2.95	2019-09-05	123.239998	123.430000	122.180000	122.760002	119.040710	7491900.0	2.13
3	2.98	2019-09-06	123.000000	123.379997	122.540001	122.870003	119.147377	5306900.0	2.12
4	3.14	2019-09-09	122.709999	122.779999	121.779999	122.169998	118.468582	7662400.0	2.13

In [18]:

```
MAdata["Date"] = pd.to_datetime(MAdata["Date"], dayfirst = True)
```

In [19]:

```
MAdata.set_index('Date',inplace=True)
```

In [20]:

```
MAdata = MAdata.loc[:,['Nio', 'PG', 'SP500', 'EFFR']]
```

In [21]:

```
MAdata.head()
```

Out[21]:

Date	Nio	PG	SP500	EFFR
2019-09-03	2.60	121.360001	2906.27	2.13
2019-09-04	2.75	123.209999	2937.78	2.13
2019-09-05	2.95	122.760002	2976.00	2.13
2019-09-06	2.98	122.870003	2978.71	2.12
2019-09-09	3.14	122.169998	2978.43	2.13

In [22]:

```
MAdata.drop(MAdata.tail(4).index,inplace=True) # drop last n rows
```

In [23]:

```
MAdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 251 entries, 2019-09-03 to 2020-08-28
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Nio     251 non-null    float64
 1   PG      251 non-null    float64
 2   SP500   251 non-null    float64
 3   EFFR    251 non-null    float64
dtypes: float64(4)
memory usage: 9.8 KB
```

In [24]:

```
# Compute log returns for SP500, Nio and PG

MAdata['log_retNio'] = 100*np.log(MAdata['Nio']).diff()

MAdata['log_retPG'] = 100*np.log(MAdata['PG']).diff()

MAdata['log_retSP'] = 100*np.log(MAdata['SP500']).diff()
```

In [25]:

```
MAdata.dropna(inplace=True)
MAdata.shape
```

Out[25]:

```
(250, 7)
```

In [26]:

```
# Calculating the excess return for SP500, Nio and P&G
MAdata['ex_retSP'] = MAdata['log_retSP'] - MAdata['EFFR']
MAdata['ex_retNio'] = MAdata['log_retNio'] - MAdata['EFFR']
MAdata['ex_retPG'] = MAdata['log_retPG'] - MAdata['EFFR']
MAdata.head()
```

Out[26]:

Date	Nio	PG	SP500	EFFR	log_retNio	log_retPG	log_retSP	ex_retSP	ex_retNio
2019-09-04	2.75	123.209999	2937.78	2.13	5.608947	1.512887	1.078372	-1.051628	3.478947
2019-09-05	2.95	122.760002	2976.00	2.13	7.020426	-0.365896	1.292592	-0.837408	4.890426
2019-09-06	2.98	122.870003	2978.71	2.12	1.011813	0.089566	0.091020	-2.028980	-1.108187
2019-09-09	3.14	122.169998	2978.43	2.13	5.229950	-0.571341	-0.009400	-2.139400	3.099950
2019-09-10	3.32	119.879997	2979.39	2.13	5.574198	-1.892228	0.032227	-2.097773	3.444198

In [27]:

```
# Descriptive stats of excess return series
MAdata_summary = MAdata[['ex_retSP','ex_retNio','ex_retPG']].describe()
MAdata_summary
```

Out[27]:

	ex_retSP	ex_retNio	ex_retPG
count	250.000000	250.000000	250.000000
mean	-0.882402	-0.159056	-0.890337
std	2.292258	7.526478	2.254624
min	-13.015214	-24.490471	-10.232849
25%	-1.839556	-4.778383	-1.857885
50%	-1.112795	-0.818166	-0.883179
75%	0.192118	3.345140	0.166503
max	8.848316	41.445613	10.250938

In [28]:

```
# Estimate OLS regression (Nio on SP500): with intercept, nonrobust
Nio_SP_model = smf.ols(formula='ex_retNio ~ ex_retSP',data=MAdata)

# Fit the model
Nio_SP_result = Nio_SP_model.fit()

# Print out the regression result summary
print(Nio_SP_result.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:              ex_retNio   R-squared:                       0.0
21
Model:                            OLS   Adj. R-squared:                  0.0
17
Method:                 Least Squares   F-statistic:                     5.3
19
Date:                Sun, 08 Nov 2020   Prob (F-statistic):             0.02
19
Time:                        16:54:55   Log-Likelihood:                -856.
19
No. Observations:                 250   AIC:                             171
6.
Df Residuals:                     248   BIC:                             172
3.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept      0.2608      0.506      0.516      0.607      -0.736       1.2
57
ex_retSP       0.4758      0.206      2.306      0.022       0.069       0.8
82
==============================================================================
==
Omnibus:                       65.149   Durbin-Watson:                   1.7
17
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              258.6
16
Skew:                           1.009   Prob(JB):                     6.95e-
57
Kurtosis:                       7.556   Cond. No.                         2.
69
==============================================================================
==

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

In [29]:

```
# Estimate OLS regression (P&G on SP500): with intercept, nonrobust
PG_SP_model = smf.ols(formula='ex_retPG ~ ex_retSP',data=MAdata)

# Fit the model
PG_SP_result = PG_SP_model.fit()

# Print out the regression result summary
print(PG_SP_result.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                ex_retPG   R-squared:                       0.0
71
Model:                            OLS   Adj. R-squared:                  0.0
67
Method:                 Least Squares   F-statistic:                     19.
02
Date:                Sun, 08 Nov 2020   Prob (F-statistic):           1.90e-
05
Time:                        16:54:55   Log-Likelihood:                -548.
24
No. Observations:                 250   AIC:                             110
0.
Df Residuals:                     248   BIC:                             110
8.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept     -0.6587      0.148     -4.463      0.000      -0.949      -0.3
68
ex_retSP       0.2625      0.060      4.361      0.000       0.144       0.3
81
==============================================================================
==
Omnibus:                       46.772   Durbin-Watson:                   2.0
32
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              559.2
90
Skew:                          -0.097   Prob(JB):                    3.56e-1
22
Kurtosis:                      10.325   Cond. No.                         2.
69
==============================================================================
==

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

In [30]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [Nio_SP_result, PG_SP_result]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['CAPM Nio','CAPM P&G']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared),}

# Calling the summary_col function and specify the parameters
result_table_c = summary_col(results=result_list, float_format='%0.3f',\
                             stars=True, model_names=name_list, info_dict = info_dict)

result_table_c
```

Out[30]:

	CAPM Nio	CAPM P&G
Intercept	0.261	-0.659***
	(0.506)	(0.148)
ex_retSP	0.476**	0.263***
	(0.206)	(0.060)
R-squared	0.017	0.067
	0.021	0.071
No. observations	250	250
R-squared	0.021	0.071

In [31]:

```
# Define the OLS model
Nio_SP_result = smf.ols(formula='ex_retNio ~ ex_retSP',data=MAdata).fit()

# Fit the model with HAC & small error correction to get robust results
robust_Nio_SP_result = Nio_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correcti
print(robust_Nio_SP_result.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:              ex_retNio   R-squared:                       0.0
21
Model:                            OLS   Adj. R-squared:                  0.0
17
Method:                 Least Squares   F-statistic:                     7.0
88
Date:                Sun, 08 Nov 2020   Prob (F-statistic):            0.008
27
Time:                        16:54:56   Log-Likelihood:                -856.
19
No. Observations:                 250   AIC:                             171
6.
Df Residuals:                     248   BIC:                             172
3.
Df Model:                           1
Covariance Type:                  HAC
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept      0.2608      0.540      0.483      0.629      -0.798       1.3
19
ex_retSP       0.4758      0.179      2.662      0.008       0.126       0.8
26
==============================================================================
==
Omnibus:                       65.149   Durbin-Watson:                   1.7
17
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              258.6
16
Skew:                           1.009   Prob(JB):                     6.95e-
57
Kurtosis:                       7.556   Cond. No.                         2.
69
==============================================================================
==

Warnings:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC)
using 12 lags and with small sample correction
```

In [32]:

```
# Define the OLS model
PG_SP_result = smf.ols(formula='ex_retPG ~ ex_retSP',data=MAdata).fit()

# Fit the model with HAC & small error correction to get robust results
robust_PG_SP_result = PG_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correction
print(robust_PG_SP_result.summary())
```

```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                ex_retPG   R-squared:
0.071
Model:                            OLS   Adj. R-squared:
0.067
Method:                 Least Squares   F-statistic:                     1
7.01
Date:                Sun, 08 Nov 2020   Prob (F-statistic):           5.07
e-05
Time:                        16:54:56   Log-Likelihood:                -54
8.24
No. Observations:                 250   AIC:                             1
100.
Df Residuals:                     248   BIC:                             1
108.
Df Model:                           1
Covariance Type:                  HAC
```

In [33]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [robust_Nio_SP_result, robust_PG_SP_result]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['Robust_CAPM Nio','Robust_CAPM P&G']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared),}

# Calling the summary_col function and specify the parameters
result_table_c = summary_col(results=result_list, float_format='%0.3f',\
                             stars=True, model_names=name_list, info_dict = info_dict)

result_table_c
```

Out[33]:

	Robust_CAPM Nio	Robust_CAPM P&G
Intercept	0.261	-0.659***
	(0.540)	(0.179)
ex_retSP	0.476***	0.263***
	(0.179)	(0.064)
R-squared	0.017	0.067
	0.021	0.071
No. observations	250	250
R-squared	0.021	0.071

In [34]:

```
# Run the robust model
robust_Nio_SP_result = Nio_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correcti

# Testing the alpha = 0
hypotheses = 'Intercept = 0'
t_stat = robust_Nio_SP_result.t_test(hypotheses)
print(t_stat)
```

```
                             Test for Constraints
==============================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
------------------------------------------------------------------------------
----
c0             0.2608      0.540      0.483      0.629      -0.798
1.319
==============================================================================
====
```

In [35]:

```
# Run the robust model
robust_PG_SP_result = PG_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correction

# Testing the alpha = 0
hypotheses = 'Intercept = 0'
t_stat = robust_PG_SP_result.t_test(hypotheses)
print(t_stat)
```

```
                             Test for Constraints
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
c0            -0.6587      0.179     -3.679      0.000      -1.010      -0.3
08
==============================================================================
==
```

In [36]:

```
# Run the robust model

robust_Nio_SP_result = Nio_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correcti

# Testing the hypothesis
hypotheses_r = 'ex_retSP = 1'
t_stat_r = robust_Nio_SP_result.t_test(hypotheses_r)
print(t_stat_r)
```

```
                             Test for Constraints
==============================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
------------------------------------------------------------------------------
----
c0             0.4758      0.179     -2.934      0.003       0.126
0.826
==============================================================================
====
```

In [37]:

```
# Run the robust model

robust_PG_SP_result = PG_SP_model.fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correction

# Testing the hypothesis
hypotheses_r = 'ex_retSP = 1'
t_stat_r = robust_PG_SP_result.t_test(hypotheses_r)
print(t_stat_r)
```

```
                            Test for Constraints
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
c0             0.2625      0.064    -11.588      0.000       0.138       0.3
87
==============================================================================
==
```

In [38]:

```
robust_PG_SP_result.conf_int(alpha=0.05, cols=None)
```

Out[38]:

	0	1
Intercept	-1.009656	-0.307736
ex_retSP	0.137771	0.387253

In [39]:

```
# Slicing the data into two parts: use the Date index
subset1 = MAdata[:'2020-01-27']
subset2 = MAdata['2020-01-28':'2020-06-18']

subset1 = subset1.loc[:,['EFFR','SP500','Nio','PG']]
subset2 = subset2.loc[:,['EFFR','SP500','Nio','PG']]

subset1.info
```

Out[39]:

```
<bound method DataFrame.info of             EFFR    SP500   Nio          PG
Date
2019-09-04  2.13  2937.78  2.75  123.209999
2019-09-05  2.13  2976.00  2.95  122.760002
2019-09-06  2.12  2978.71  2.98  122.870003
2019-09-09  2.13  2978.43  3.14  122.169998
2019-09-10  2.13  2979.39  3.32  119.879997
...          ...      ...   ...         ...
2020-01-21  1.54  3289.29  5.17  126.089996
2020-01-22  1.54  3316.81  4.79  126.309998
2020-01-23  1.55  3329.62  4.92  124.989998
2020-01-24  1.55  3329.62  4.66  125.139999
2020-01-27  1.55  3320.79  4.01  125.690002

[100 rows x 4 columns]>
```

In [40]:

```
subset2.info
```

Out[40]:

```
<bound method DataFrame.info of             EFFR    SP500   Nio          PG
Date
2020-01-28  1.55  3321.75  4.21  126.029999
2020-01-29  1.55  3325.54  4.27  125.059998
2020-01-30  1.55  3295.47  4.08  125.949997
2020-01-31  1.55  3243.63  3.78  124.620003
2020-02-03  1.55  3276.24  4.06  125.110001
...          ...      ...   ...         ...
2020-06-12  0.06  3122.87  6.10  115.620003
2020-06-15  0.06  3112.35  6.83  116.690002
2020-06-16  0.07  3193.93  6.99  118.129997
2020-06-17  0.07  3232.39  6.84  117.930000
2020-06-18  0.07  3207.18  7.18  119.279999

[100 rows x 4 columns]>
```

In [41]:

```
# Log returns for subset1
subset1['log_retSP1'] = 100*np.log(subset1['SP500']).diff()
subset1['log_retNio1'] = 100*np.log(subset1['Nio']).diff()
subset1['log_retPG1'] = 100*np.log(subset1['PG']).diff()
# Log returns for subset2
subset2['log_retSP2'] = 100*np.log(subset2['SP500']).diff()
subset2['log_retNio2'] = 100*np.log(subset2['Nio']).diff()
subset2['log_retPG2'] = 100*np.log(subset2['PG']).diff()
```

In [42]:

```
# Drop the NA
subset1.dropna(inplace=True)
subset2.dropna(inplace=True)
```

In [43]:

```
# Excess returns for subset1
subset1['ex_retSP1'] = subset1['log_retSP1'] - subset1['EFFR']
subset1['ex_retNio1'] = subset1['log_retNio1'] - subset1['EFFR']
subset1['ex_retPG1'] = subset1['log_retPG1'] - subset1['EFFR']
# Excess returns for subset2
subset2['ex_retSP2'] = subset2['log_retSP2'] - subset2['EFFR']
subset2['ex_retNio2'] = subset2['log_retNio2'] - subset2['EFFR']
subset2['ex_retPG2'] = subset2['log_retPG2'] - subset2['EFFR']
```

In [44]:

```
# CAPM with subset1&2
sub1_Nio_SP_result = smf.ols(formula='ex_retNio1 ~ ex_retSP1',data=subset1).fit()
sub2_Nio_SP_result = smf.ols(formula='ex_retNio2 ~ ex_retSP2',data=subset2).fit()
# Robust CAPM with subset1&2
robust_sub1_Nio_SP_result = smf.ols(formula='ex_retNio1 ~ ex_retSP1',data=subset1\
                                    ).fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correc
robust_sub2_Nio_SP_result = smf.ols(formula='ex_retNio2 ~ ex_retSP2',data=subset2\
                                    ).fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correc
```

In [45]:

```
# CAPM with subset1&2
sub1_PG_SP_result = smf.ols(formula='ex_retPG1 ~ ex_retSP1',data=subset1).fit()
sub2_PG_SP_result = smf.ols(formula='ex_retPG2 ~ ex_retSP2',data=subset2).fit()
# Robust CAPM with subset1&2
robust_sub1_PG_SP_result = smf.ols(formula='ex_retPG1 ~ ex_retSP1',data=subset1\
                                    ).fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correc
robust_sub2_PG_SP_result = smf.ols(formula='ex_retPG2 ~ ex_retSP2',data=subset2\
                                    ).fit(cov_type='HAC',cov_kwds={'maxlags':12,'use_correc
```

In [46]:

```
print(sub1_Nio_SP_result.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:              ex_retNio1   R-squared:                       0.0
14
Model:                             OLS   Adj. R-squared:                  0.0
04
Method:                  Least Squares   F-statistic:                     1.4
02
Date:                 Sun, 08 Nov 2020   Prob (F-statistic):              0.2
39
Time:                         16:54:58   Log-Likelihood:                -351.
47
No. Observations:                  99   AIC:                             70
6.9
Df Residuals:                      97   BIC:                             71
2.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
Intercept      1.3892      2.446      0.568      0.571      -3.466       6.2
44
ex_retSP1      1.7159      1.449      1.184      0.239      -1.161       4.5
93
==============================================================================
==
Omnibus:                       49.681   Durbin-Watson:                   1.8
26
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              224.6
00
Skew:                           1.565   Prob(JB):                     1.69e-
49
Kurtosis:                       9.682   Cond. No.                         6.
36
==============================================================================
==

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

In [47]:

```
print(sub1_PG_SP_result.summary())
print(sub1_PG_SP_result.summary())
```

```
----
Omnibus:                       22.494   Durbin-Watson:
1.893
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               3
9.987
Skew:                          -0.928   Prob(JB):                     2.07
e-09
Kurtosis:                       5.500   Cond. No.
6.36
==============================================================================
====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
                            OLS Regression Results
==============================================================================
====
Dep. Variable:              ex_retPG1   R-squared:
0.035
```

In [48]:

```
from statsmodels.iolib.summary2 import summary_col
```

In [49]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [Nio_SP_result,robust_Nio_SP_result,\
               sub1_Nio_SP_result,robust_sub1_Nio_SP_result,sub2_Nio_SP_result,robust_sub2_N

# 2. Define a list of names that correspond to the regression results variables
name_list = ['CAPM','Robust_CAPM','Sub1','Robust_Sub1','Sub2','Robust_Sub2']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared)}

# Calling the summary_col function and specify the parameters
result_table_Nio = summary_col(results=result_list, float_format='%0.3f',\
                               stars=True, model_names=name_list, info_dict = info_dict)

result_table_Nio
```

Out[49]:

	CAPM	Robust_CAPM	Sub1	Robust_Sub1	Sub2	Robust_Sub2
Intercept	0.261	0.261	1.389	1.389	0.167	0.167
	(0.506)	(0.540)	(2.446)	(2.575)	(0.646)	(0.767)
R-squared	0.017	0.017	0.004	0.004	0.024	0.024
	0.021	0.021	0.014	0.014	0.034	0.034
ex_retSP	0.476**	0.476***				
	(0.206)	(0.179)				
ex_retSP1			1.716	1.716		
			(1.449)	(1.563)		
ex_retSP2					0.354*	0.354**
					(0.191)	(0.155)
No. observations	250	250	99	99	99	99
R-squared	0.021	0.021	0.014	0.014	0.034	0.034

In [50]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [PG_SP_result,robust_PG_SP_result,\
               sub1_PG_SP_result,robust_sub1_PG_SP_result,sub2_PG_SP_result,robust_sub2_PG_S

# 2. Define a list of names that correspond to the regression results variables
name_list = ['CAPM','Robust_CAPM','Sub1','Robust_Sub1','Sub2','Robust_Sub2']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared)}

# Calling the summary_col function and specify the parameters
result_table_PG = summary_col(results=result_list, float_format='%0.3f',\
                              stars=True, model_names=name_list, info_dict = info_dict)

result_table_PG
```

Out[50]:

	CAPM	Robust_CAPM	Sub1	Robust_Sub1	Sub2	Robust_Sub2
Intercept	-0.659***	-0.659***	-1.197***	-1.197***	-0.528	-0.528*
	(0.148)	(0.179)	(0.280)	(0.316)	(0.323)	(0.301)
R-squared	0.067	0.067	0.025	0.025	0.032	0.032
	0.071	0.071	0.035	0.035	0.041	0.041
ex_retSP	0.263***	0.263***				
	(0.060)	(0.064)				
ex_retSP1			0.309*	0.309		
			(0.166)	(0.221)		
ex_retSP2					0.195**	0.195***
					(0.095)	(0.058)
No. observations	250	250	99	99	99	99
R-squared	0.071	0.071	0.035	0.035	0.041	0.041

In [51]:

```
Nio_SP_model_new = smf.ols(formula='ex_retNio ~ ex_retSP -1',data=MAdata).fit()

print(Nio_SP_model_new.summary())
```

```
                                 OLS Regression Results
================================================================================
===========
Dep. Variable:              ex_retNio   R-squared (uncentered):
0.020
Model:                            OLS   Adj. R-squared (uncentered):
0.016
Method:                 Least Squares   F-statistic:
5.182
Date:                Sun, 08 Nov 2020   Prob (F-statistic):
0.0237
Time:                        16:54:59   Log-Likelihood:
-856.32
No. Observations:                 250   AIC:
1715.
Df Residuals:                     249   BIC:
1718.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
ex_retSP       0.4375      0.192      2.276      0.024       0.059       0.8
16
================================================================================
==
Omnibus:                       64.971   Durbin-Watson:                   1.7
09
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              258.2
91
Skew:                           1.005   Prob(JB):                     8.18e-
57
Kurtosis:                       7.556   Cond. No.                         1.
00
================================================================================
==

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

In [52]:

```
PG_SP_model_new = smf.ols(formula='ex_retPG ~ ex_retSP -1',data=MAdata).fit()

print(PG_SP_model_new.summary())
```

```
                                 OLS Regression Results
================================================================================
===========
Dep. Variable:                ex_retPG   R-squared (uncentered):
0.132
Model:                            OLS   Adj. R-squared (uncentered):
0.129
Method:                 Least Squares   F-statistic:
38.02
Date:                Sun, 08 Nov 2020   Prob (F-statistic):
2.81e-09
Time:                        16:54:59   Log-Likelihood:
-557.90
No. Observations:                 250   AIC:
1118.
Df Residuals:                     249   BIC:
1121.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
==
                 coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
--
ex_retSP       0.3592      0.058      6.166      0.000       0.244       0.4
74
================================================================================
==
Omnibus:                       52.747   Durbin-Watson:                   1.9
37
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              724.0
81
Skew:                          -0.239   Prob(JB):                    5.86e-1
58
Kurtosis:                      11.324   Cond. No.                         1.
00
================================================================================
==

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

In [53]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [PG_SP_result, PG_SP_model_new]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['CAPM PG','No Intercept']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared),}

# Calling the summary_col function and specify the parameters
result_table_n = summary_col(results=result_list, float_format='%0.3f',\
                             stars=True, model_names=name_list, info_dict = info_dict)

result_table_n
```

Out[53]:

	CAPM PG	No Intercept
Intercept	-0.659***	
	(0.148)	
R-squared	0.067	0.129
	0.071	0.132
ex_retSP	0.263***	0.359***
	(0.060)	(0.058)
No. observations	250	250
R-squared	0.071	0.132

In [54]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [Nio_SP_result, Nio_SP_model_new]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['CAPM Nio','No Intercept']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared),}

# Calling the summary_col function and specify the parameters
result_table_p = summary_col(results=result_list, float_format='%0.3f',\
                             stars=True, model_names=name_list, info_dict = info_dict)

result_table_p
```

Out[54]:

	CAPM Nio	No Intercept
Intercept	0.261	
	(0.506)	
R-squared	0.017	0.016
	0.021	0.020
ex_retSP	0.476**	0.437**
	(0.206)	(0.192)
No. observations	250	250
R-squared	0.021	0.020

In [55]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [robust_Nio_SP_result,\
            robust_sub1_Nio_SP_result,robust_sub2_Nio_SP_result]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['Robust_CAPM Nio','Robust_Sub1','Robust_Sub2']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared)}

# Calling the summary_col function and specify the parameters
result_table_Nio = summary_col(results=result_list, float_format='%0.3f',\
                           stars=True, model_names=name_list, info_dict = info_dict)

result_table_Nio
```

Out[55]:

	Robust_CAPM Nio	Robust_Sub1	Robust_Sub2
Intercept	0.261	1.389	0.167
	(0.540)	(2.575)	(0.767)
R-squared	0.017	0.004	0.024
	0.021	0.014	0.034
ex_retSP	0.476***		
	(0.179)		
ex_retSP1		1.716	
		(1.563)	
ex_retSP2			0.354**
			(0.155)
No. observations	250	99	99
R-squared	0.021	0.014	0.034

In [56]:

```
# Define the parameters
# 1. Put all regression results' variables into a list
result_list = [robust_PG_SP_result,\
            robust_sub1_PG_SP_result,robust_sub2_PG_SP_result]

# 2. Define a list of names that correspond to the regression results variables
name_list = ['Robust_CAPM PG','Robust_Sub1','Robust_Sub2']

# 3. Define the 'info_dict', dict of lambda functions to be applied to results instances to
info_dict={'No. observations':lambda x: "{0:d}".format(int(x.nobs)),\
           'R-squared':lambda x: "{:.3f}".format(x.rsquared)}

# Calling the summary_col function and specify the parameters
result_table_PG = summary_col(results=result_list, float_format='%0.3f',\
                              stars=True, model_names=name_list, info_dict = info_dict)

result_table_PG
```

Out[56]:

	Robust_CAPM PG	Robust_Sub1	Robust_Sub2
Intercept	-0.659***	-1.197***	-0.528*
	(0.179)	(0.316)	(0.301)
R-squared	0.067	0.025	0.032
	0.071	0.035	0.041
ex_retSP	0.263***		
	(0.064)		
ex_retSP1		0.309	
		(0.221)	
ex_retSP2			0.195***
			(0.058)
No. observations	250	99	99
R-squared	0.071	0.035	0.041