Regression in Python with Statsmodels

Fan Wang

2021-01-05

Contents

1 Regressions with Statsmodel

Go to the **RMD**, **PDF**, or **HTML** version of this file. Go back to Python Code Examples Repository (bookdown site) or the pyfan Package (API).

```
import numpy as np
import statsmodels.api as sm
```

1.1 Test Regression with Statsmodel

Testing default example from statsmodel.

```
import numpy as np
import statsmodels.api as sm
spector_data = sm.datasets.spector.load(as_pandas=False)
spector_data.exog = sm.add_constant(spector_data.exog, prepend=False)
mod = sm.OLS(spector_data.endog, spector_data.exog)
res = mod.fit()
print(res.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.416
## Model:                            OLS   Adj. R-squared:                  0.353
## Method:                 Least Squares   F-statistic:                     6.646
## Date:                Tue, 05 Jan 2021   Prob (F-statistic):            0.00157
## Time:                        16:35:53   Log-Likelihood:                -12.978
## No. Observations:                  32   AIC:                             33.96
## Df Residuals:                      28   BIC:                             39.82
## Df Model:                           3
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## x1             0.4639      0.162      2.864      0.008       0.132       0.796
## x2             0.0105      0.019      0.539      0.594      -0.029       0.050
## x3             0.3786      0.139      2.720      0.011       0.093       0.664
```

```
## const         -1.4980      0.524     -2.859      0.008      -2.571      -0.425
## ==============================================================================
## Omnibus:                        0.176   Durbin-Watson:                   2.346
## Prob(Omnibus):                  0.916   Jarque-Bera (JB):                0.167
## Skew:                           0.141   Prob(JB):                        0.920
## Kurtosis:                       2.786   Cond. No.                         176.
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

1.2 Estimation with More Coefficients than Observations

Testing the *sm_OLS* function where there are more observations and coefficients, as well as when there are less. Tests similar to the test on this page.

First, more observations than coefficients:

```
# Number of observations
it_sample = 1000000
# Vaues of the x variable
ar_x = np.linspace(0, 10, it_sample)
# generate matrix of inputs with polynomial expansion
mt_x = np.column_stack((ar_x, ar_x**2, ar_x**3, ar_x**4, ar_x**5, ar_x**6, ar_x**7, ar_x**8))
# model coefficients
ar_beta = np.array([100, 10, 1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6])
# Generate the error term
ar_e = np.random.normal(size=it_sample)
# add constant
mt_x = sm.add_constant(mt_x)
# generate the outocome variable
ar_y = np.dot(mt_x, ar_beta) + ar_e
# regression result
ob_model = sm.OLS(ar_y, mt_x)
# Show results
ob_results = ob_model.fit()
print(ob_results.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       1.000
## Model:                            OLS   Adj. R-squared:                  1.000
## Method:                 Least Squares   F-statistic:                 4.984e+09
## Date:                Tue, 05 Jan 2021   Prob (F-statistic):               0.00
## Time:                        16:35:54   Log-Likelihood:            -1.4189e+06
## No. Observations:             1000000   AIC:                         2.838e+06
## Df Residuals:                  999991   BIC:                         2.838e+06
## Df Model:                           8
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const        100.0073      0.009   1.11e+04      0.000      99.990     100.025
## x1             9.9569      0.042    239.526      0.000       9.875      10.038
## x2             1.0701      0.062     17.265      0.000       0.949       1.192
```

```
## x3             0.0536      0.042      1.279      0.201      -0.029       0.136
## x4             0.0257      0.015      1.712      0.087      -0.004       0.055
## x5            -0.0020      0.003     -0.651      0.515      -0.008       0.004
## x6             0.0004      0.000      1.201      0.230      -0.000       0.001
## x7         -8.192e-06   2.13e-05     -0.385      0.700    -4.99e-05    3.35e-05
## x8          1.424e-06   5.31e-07      2.684      0.007     3.84e-07    2.46e-06
## ==============================================================================
## Omnibus:                        1.159   Durbin-Watson:                   2.000
## Prob(Omnibus):                  0.560   Jarque-Bera (JB):                1.157
## Skew:                          -0.001   Prob(JB):                        0.561
## Kurtosis:                       3.005   Cond. No.                     2.10e+09
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## [2] The condition number is large, 2.1e+09. This might indicate that there are
## strong multicollinearity or other numerical problems.
```

second, less observations than coefficients. Note that there are nine coefficients to estimates, so if we have only 5 observations, that is less than coefficients. Unlike Stata and many other packages, estimates are provided even when full rank is not possible. See here for more information. This is actually very useful for testing purposes. For models in very large parameter space, can test solution and estimation structure even when the number of observations are limited. See also Moore–Penrose inverse.

```python
# Number of observations
it_sample = 5
# Vaues of the x variable
ar_x = np.linspace(0, 10, it_sample)
# generate matrix of inputs with polynomial expansion
mt_x = np.column_stack((ar_x, ar_x**2, ar_x**3, ar_x**4, ar_x**5, ar_x**6, ar_x**7, ar_x**8))
# model coefficients
ar_beta = np.array([100, 10, 1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6])
# Generate the error term
ar_e = np.random.normal(size=it_sample)
# add constant
mt_x = sm.add_constant(mt_x)
# generate the outocome variable
ar_y = np.dot(mt_x, ar_beta) + ar_e
# regression result
ob_model = sm.OLS(ar_y, mt_x)
# Show results
ob_results = ob_model.fit()
print(ob_results.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       1.000
## Model:                            OLS   Adj. R-squared:                    nan
## Method:                 Least Squares   F-statistic:                       nan
## Date:                Tue, 05 Jan 2021   Prob (F-statistic):                nan
## Time:                        16:35:55   Log-Likelihood:                 94.858
## No. Observations:                   5   AIC:                            -179.7
## Df Residuals:                       0   BIC:                            -181.7
## Df Model:                           4
## Covariance Type:            nonrobust
```

```
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const         98.9493        inf          0        nan         nan         nan
## x1             0.0688        inf          0        nan         nan         nan
## x2             0.1573        inf          0        nan         nan         nan
## x3             0.3297        inf          0        nan         nan         nan
## x4             0.5732        inf          0        nan         nan         nan
## x5             0.5979        inf          0        nan         nan         nan
## x6            -0.3161        inf         -0        nan         nan         nan
## x7             0.0457        inf          0        nan         nan         nan
## x8            -0.0021        inf         -0        nan         nan         nan
## ==============================================================================
## Omnibus:                          nan   Durbin-Watson:                   1.078
## Prob(Omnibus):                    nan   Jarque-Bera (JB):                1.684
## Skew:                           1.420   Prob(JB):                        0.431
## Kurtosis:                       3.144   Cond. No.                     1.01e+08
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## [2] The input rank is higher than the number of observations.
## [3] The condition number is large, 1.01e+08. This might indicate that there are
## strong multicollinearity or other numerical problems.
##
## C:\Users\fan\AppData\Roaming\Python\Python38\site-packages\statsmodels\stats\stattools.py:70: ValueWa
##   warn("omni_normtest is not valid with less than 8 observations; %i "
## C:\Users\fan\AppData\Roaming\Python\Python38\site-packages\statsmodels\regression\linear_model.py:169
##   return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
## C:\Users\fan\AppData\Roaming\Python\Python38\site-packages\statsmodels\regression\linear_model.py:169
##   return 1 - (np.divide(self.nobs - self.k_constant, self.df_resid)
## C:\Users\fan\AppData\Roaming\Python\Python38\site-packages\statsmodels\regression\linear_model.py:162
##   return np.dot(wresid, wresid) / self.df_resid
```