panel-ols

April 8, 2022

AUTHOR: SPARSHSAH

[1]:
```
from numpy.random import default_rng as rng
import pandas as pd
from statsmodels.api import add_constant, OLS
from linearmodels.panel import PanelOLS
```

1 ground-truth underlying model

there are 3 racecars, each of which has a different BHP-to-weight ratio ("slow" / "medium" / "fast"), each of whose drivers get a jolt of adrenaline in the homestretch ("initial" / "middle" / "final" lap), and each of which consumes fuel at a different rate. we want to understand the relationship between these variables and the racecar's lap speed.

- 3 entities (`slow`, `med`, `fast`)
- 3 timesteps (`init`, `mid`, `fin`)

in truth, we know that $E[\texttt{speed}_{i,t} \mid \texttt{is\_fast}_i, \texttt{is\_fin}_t, \texttt{fuel}_{i,t}] = \texttt{is\_fast}_i + \texttt{is\_fin}_t + \texttt{fuel}_{i,t}$.

2 construct dataset

[2]:
```
rng = rng(seed=1337)
```

[3]:
```
slow_x = pd.DataFrame(
    {"is_slow": {"init": 1, "mid": 1, "fin": 1},
     "is_med": {"init": 0, "mid": 0, "fin": 0},
     "is_fast": {"init": 0, "mid": 0, "fin": 0},
     "is_init": {"init": 1, "mid": 0, "fin": 0},
     "is_mid": {"init": 0, "mid": 1, "fin": 0},
     "is_fin": {"init": 0, "mid": 0, "fin": 1},
     "fuel": {"init": 1, "mid": 0.75, "fin": 0.5},
     "speed": {"init": 0 + 0 + 1 + rng.normal(scale=0.01),
               "mid": 0 + 0 + 0.75 + rng.normal(scale=0.01),
               "fin": 0 + 1 + 0.5 + rng.normal(scale=0.01)}
    # rename because PanelOLS requires time variable to be numeric
    }).rename(index={"init": 0, "mid": 1, "fin": 2})
slow_x
```

[3]:

	is_slow	is_med	is_fast	is_init	is_mid	is_fin	fuel	speed
0	1	0	0	1	0	0	1.00	1.000383
1	1	0	0	0	1	0	0.75	0.754739
2	1	0	0	0	0	1	0.50	1.498623

[4]:
```
med_x = pd.DataFrame(
    {"is_slow": {"init": 0, "mid": 0, "fin": 0},
     "is_med": {"init": 1, "mid": 1, "fin": 1},
     "is_fast": {"init": 0, "mid": 0, "fin": 0},
     "is_init": {"init": 1, "mid": 0, "fin": 0},
     "is_mid": {"init": 0, "mid": 1, "fin": 0},
     "is_fin": {"init": 0, "mid": 0, "fin": 1},
     "fuel": {"init": 1, "mid": 0.66, "fin": 0.33},
     "speed": {"init": 0 + 0 + 1 + rng.normal(scale=0.01),
               "mid": 0 + 0 + 0.66 + rng.normal(scale=0.01),
               "fin": 0 + 1 + 0.33 + rng.normal(scale=0.01)}
    # rename because PanelOLS requires time variable to be numeric
    }).rename(index={"init": 0, "mid": 1, "fin": 2})
med_x
```

[4]:

	is_slow	is_med	is_fast	is_init	is_mid	is_fin	fuel	speed
0	0	1	0	1	0	0	1.00	0.986107
1	0	1	0	0	1	0	0.66	0.685201
2	0	1	0	0	0	1	0.33	1.319936

[5]:
```
fast_x = pd.DataFrame(
    {"is_slow": {"init": 0, "mid": 0, "fin": 0},
     "is_med": {"init": 0, "mid": 0, "fin": 0},
     "is_fast": {"init": 1, "mid": 1, "fin": 1},
     "is_init": {"init": 1, "mid": 0, "fin": 0},
     "is_mid": {"init": 0, "mid": 1, "fin": 0},
     "is_fin": {"init": 0, "mid": 0, "fin": 1},
     "fuel": {"init": 1, "mid": 0.5, "fin": 0},
     "speed": {"init": 1 + 0 + 1 + rng.normal(scale=0.01),
               "mid": 1 + 0 + 0.5 + rng.normal(scale=0.01),
               "fin": 1 + 1 + 0 + rng.normal(scale=0.01)}
    # rename because PanelOLS requires time variable to be numeric
    }).rename(index={"init": 0, "mid": 1, "fin": 2})
fast_x
```

[5]:

	is_slow	is_med	is_fast	is_init	is_mid	is_fin	fuel	speed
0	0	0	1	1	0	0	1.0	2.018568
1	0	0	1	0	1	0	0.5	1.474976
2	0	0	1	0	0	1	0.0	2.001483

[6]:
```
x = pd.concat([slow_x, med_x, fast_x], keys=["slow", "med", "fast"])
x
```

[6]:

		is_slow	is_med	is_fast	is_init	is_mid	is_fin	fuel	speed
slow	0	1	0	0	1	0	0	1.00	1.000383
	1	1	0	0	0	1	0	0.75	0.754739
	2	1	0	0	0	0	1	0.50	1.498623
med	0	0	1	0	1	0	0	1.00	0.986107
	1	0	1	0	0	1	0	0.66	0.685201
	2	0	1	0	0	0	1	0.33	1.319936
fast	0	0	0	1	1	0	0	1.00	2.018568
	1	0	0	1	0	1	0	0.50	1.474976
	2	0	0	1	0	0	1	0.00	2.001483

3 regress

[7]:
```
# some people call this "y"
lhs = x["speed"]
```

3.1 "stacked" OLS estimator suffers from omitted variable bias..

Gives *negative* `fuel` slope coefficient point estimate and insignificant t-stat! And, to be fair, it doesn't know about 2/3 relevant variables. It simply sees that the cars speed up as fuel runs out, which is actually just picking up on the effect of the drivers' "adrenaline jolt" during the final lap.

[8]:
```
rhs = add_constant(x["fuel"])

OLS(endog=lhs, exog=rhs,
    hasconst=True).fit().summary()
```

[8]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  speed   R-squared:                       0.160
Model:                            OLS   Adj. R-squared:                  0.040
Method:                 Least Squares   F-statistic:                     1.331
Date:                Fri, 29 Jan 2021   Prob (F-statistic):              0.287
Time:                        20:31:02   Log-Likelihood:                -5.0872
No. Observations:                   9   AIC:                             14.17
Df Residuals:                       7   BIC:                             14.57
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.6697      0.355      4.701      0.002       0.830       2.509
fuel          -0.5726      0.496     -1.154      0.287      -1.746       0.601
==============================================================================
Omnibus:                        2.528   Durbin-Watson:                   1.792
```

```
Prob(Omnibus):                  0.282   Jarque-Bera (JB):                0.714
Skew:                           0.687   Prob(JB):                        0.700
Kurtosis:                       3.127   Cond. No.                         4.44
================================================================================

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

3.2 .. panel OLS estimator fixes that..

Gives accurate `fuel` slope coefficient point estimate with significant t-stat.

[9]:
```
rhs = x["fuel"]

PanelOLS(dependent=lhs, exog=rhs,
         entity_effects=True, time_effects=True).fit().summary
```

[9]:
```
<class 'linearmodels.compat.statsmodels.Summary'>
"""
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                   speed   R-squared:                        0.9747
Estimator:                    PanelOLS   R-squared (Between):              0.6521
No. Observations:                    9   R-squared (Within):              -2.0217
Date:                Fri, Jan 29 2021   R-squared (Overall):              0.5468
Time:                        20:31:03   Log-likelihood                   25.576
Cov. Estimator:            Unadjusted
                                        F-statistic:                      115.53
Entities:                            3   P-value                          0.0017
Avg Obs:                        3.0000   Distribution:                    F(1,3)
Min Obs:                        3.0000
Max Obs:                        3.0000   F-statistic (robust):             115.53
                                         P-value                          0.0017
Time periods:                        3   Distribution:                    F(1,3)
Avg Obs:                        3.0000
Min Obs:                        3.0000
Max Obs:                        3.0000

                             Parameter Estimates
================================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
--------------------------------------------------------------------------------
fuel           1.0333     0.0961     10.749     0.0017      0.7274      1.3393
================================================================================
```

```
F-test for Poolability: 682.19
P-value: 0.0001
Distribution: F(4,3)

Included effects: Entity, Time
"""
```

3.3 .. and by manually adding structure to the "stacked" OLS, we can replicate panel OLS!

Note: We're able to replicate panel OLS's t-stats only because we didn't specify a "sandwich" SE estimator for the panel OLS. In practice, college classes teach you how to use entity- and time-clustered SE's properly, and then everybody definitely remembers how to use them forever.

[10]:
```
"""
w/ both entity + time FE's, even w/o intercept,
including all dummies (binary indicators) would cause perfect multicollinearity.
 ↪.
must drop one of each type of FE then add a global intercept to compensate
"""
rhs = add_constant(x.loc[:, :"fuel"].drop(labels=["is_slow", "is_init"],␣
 ↪axis="columns"))

OLS(endog=lhs, exog=rhs,
    hasconst=True).fit().summary()
```

[10]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  speed   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                     649.6
Date:                Fri, 29 Jan 2021   Prob (F-statistic):           9.51e-05
Time:                        20:31:03   Log-Likelihood:                 25.576
No. Observations:                   9   AIC:                            -39.15
Df Residuals:                       3   BIC:                            -37.97
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0341      0.108     -0.315      0.774      -0.379       0.311
is_med         0.0021      0.022      0.095      0.930      -0.067       0.071
is_fast        1.0054      0.031     32.184      0.000       0.906       1.105
is_mid         0.0121      0.040      0.300      0.784      -0.116       0.140
is_fin         1.0191      0.072     14.087      0.001       0.789       1.249
```

```
fuel           1.0333      0.096     10.749      0.002       0.727       1.339
==============================================================================
Omnibus:                        0.052   Durbin-Watson:                   3.311
Prob(Omnibus):                  0.975   Jarque-Bera (JB):                0.248
Skew:                          -0.123   Prob(JB):                        0.883
Kurtosis:                       2.224   Cond. No.                         28.0
==============================================================================

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