Problem Set #3

Creator: Junjie Lei

CreationDate: Feb_10

Title: Problem Sets #3 experiments and regressions

Notes: $Colab$

In-class notebook exercises

1.1 Replicate and simulate a real study

1.1.1

```
print(r_col3.summary())
print(r_col4.summary())
```

1.1.2

- $Column\ 4$ has more control variables
- to calculate the $R\ Square$

  ```
  print(r_col3.rsquared) ## --> 0.0671339542940943
  print(r_col4.rsquared) ## --> 0.10819457890249662
  ```

  $Column\ 4$ has a higher R-squared, because it contains more control variables hence the model has more explanatory power;

- to calculate the $Std.ev$

  ```
  print(r_col3.bse.rem_any) ## --> 0.009153114318622113
  print(r_col4.bse.rem_any) ## --> 0.008949743622819572
  ```

  $Column\ 4$ has a smaller a standard error on the estimated ATE;

1.1.3

```
=====================================
                  Model 1   Model 2
-------------------------------------
d                 0.0317*** 0.0324***
                  (0.0082)  (0.0082)
grad_high_school            0.1043***
                            (0.0080)
R-squared         0.0011    0.0135
No. coefficients 2          3
=====================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

- for the estimation of the effect of `rem _any` in the experiments;

  the estimated coefficients and the standard error are listed as follow

  ```
  print(r_col3.params.rem_any)  ## --> 0.03185505049068642
  print(r_col4.params.rem_any)  ## --> 0.03186131518614749
  print(r_sim_biv.params.d)     ## --> 0.031709816056386286
  print(r_sim_control.params.d) ## --> 0.03240621068111807
  ```

  - the estimated coefficients and the standard error from the simulation are pretty close to real data;

1.1.4

- set the `beta_hs = 0.35` for the simulation
- set `B = 1000`

similar to the `compare_lpm_prop_test` function, we define a new function

```
def new_simulation_finction(
  N = 10000,
  beta_hs = 0.35,
  ATE = 0.032
):
  grad_high_school = np.random.binomial(n=1, p=0.5, size=N)
  D = np.random.binomial(n=1, p=0.61, size=N)
  baseline_probability = 0.25 + beta_hs * grad_high_school
  Y0 = np.random.binomial(n=1, p=baseline_probability)
  Y1 = np.random.binomial(n=1, p=baseline_probability + ATE * D)
  dff = pd.DataFrame({
    'grad_high_school': grad_high_school,
    'd': D,
    'y0': Y0,
    'y1': Y1
  })
  dff['y'] = dff.eval("y1 * d + y0 * (1 - d)")
```

```
    regr = sfa.ols("y ~ d", dff).fit()
    a = regr.params['d']
    return a
```

and we loop it;

```
B = 1000
res = pd.DataFrame([new_simulation_function() for i in np.arange(0, B)])
```

and we see the results;

	0	1	2	3	4	5	6	7
index	count	mean	std	min	25%	50%	75%	max
0	1000	0.0319545	0.0099911	-0.000497493	0.0254244	0.0324918	0.0388504	0.07187

1.1.5

for this question, we replicate what we did from the previous simulation;

and we plot the `params['d']` to show the estimated treatment effects;