Problem Set #3

Creator: Junjie Lei

CreationDate: Feb_10

Title: Problem Sets #3 experiments and regressions

Notes: $Colab$

In-class notebook exercises

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)  ## --> 0.031855050490686 42
  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)])
```

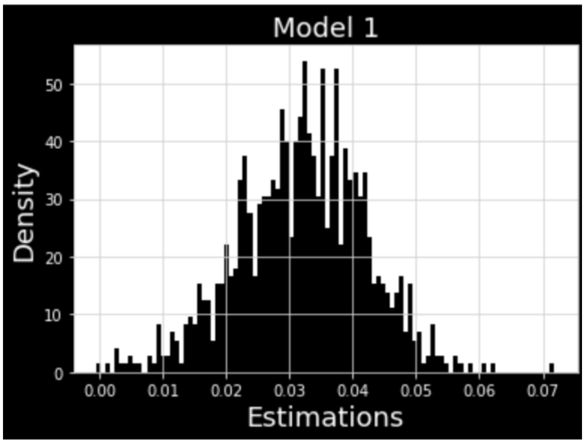
and we see the results;

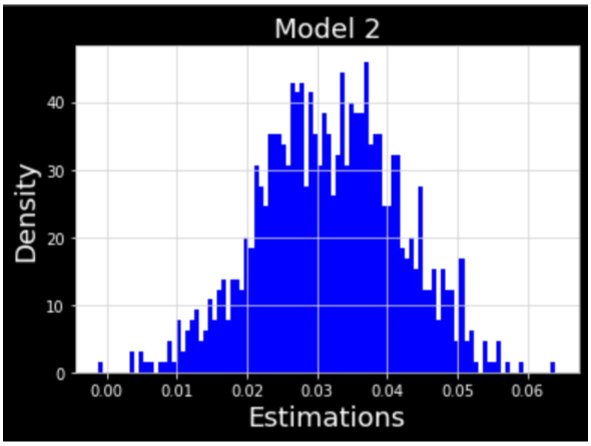
	0	1	2	3	4	5	6	7
index	count	mean	std	min	25%	50%	75%	max
0	1000	0.0319545	0.0099911	-0.000497493	0.0254244	0.0324918	0.0388504	0.07187

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;





1.1.6

from the model1 the $Std.ev \rightarrow 0.009991$

from the model2 the $Std.ev \rightarrow 0.009918$

1.1.7

refer to the `Confidence Interval machine`

```
Left board of confidence lever for model 1 =  `0.03159373629542879`
Right board of confidence lever for model 1 =  `0.03284826370457121`
Left board of confidence lever for model 2 = `0.03114525827711666`
Right board of confidence lever for model 2 =  `0.032286741722883344`
```

and we count when the confidence intervals include the `true ATE` in each model

```
k = 0
for i in np.arange(0, B):
  if ch[0][i] >= lboard1 and ch[0][i] <= rboard1:
    k = k+1
  else:
    k = k + 0
```

the output for `k` is 58

1.2 Shoe technology experiment

1.2.1

- set the baseline `inter-person variability` $\rightarrow 5$
- use the `std()` function to find the standard deviation of the estimated ATE;

$\textit{for the block distrivution} \rightarrow 0.134456$

$\textit{for the non-block distrivution} \rightarrow\ 0.236282$

1.2.2

- If we increase the `inter-person variabilty` $\rightarrow$ 10, and rep

$\textit{for the block distrivution} \rightarrow 0.445792$

$\textit{for the non-block distrivution} \rightarrow\ 0.14316$

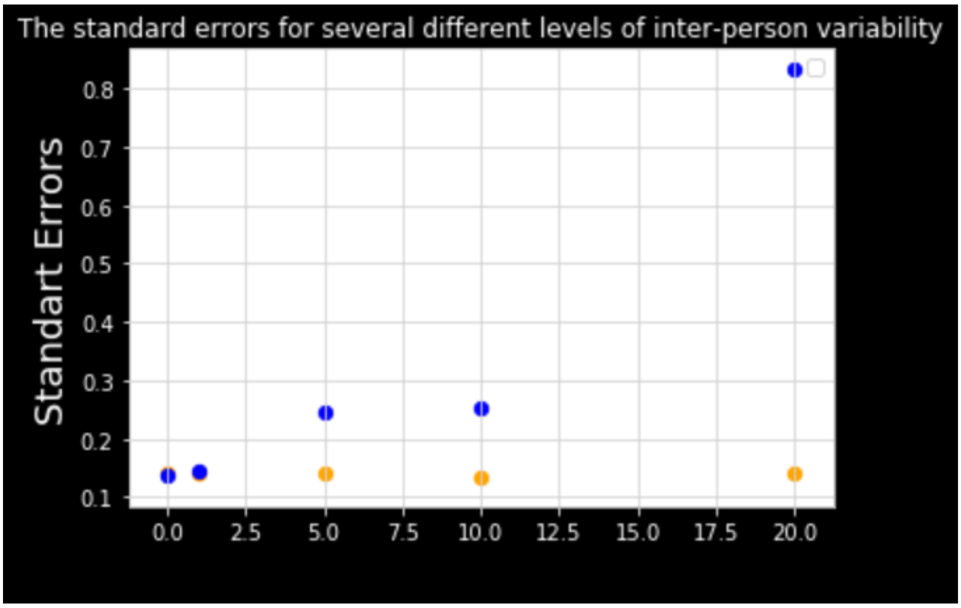
> if we increase the `inter-person variability` , then the $Std.ev$ of estimated ATE will also increases;

1.2.3

- in this exercise, we simulate different levels of `inter-person variability` as follows

	inter_person_variability	Block_std	Noblock_std
0	0	0.141987	0.138251
1	1	0.141083	0.143926
2	5	0.141181	0.246841
3	10	0.134415	0.254024
4	20	0.141145	0.831896

- and then we plot the values of $Std.ev$ under different levels of inter-person variability



1.2.4

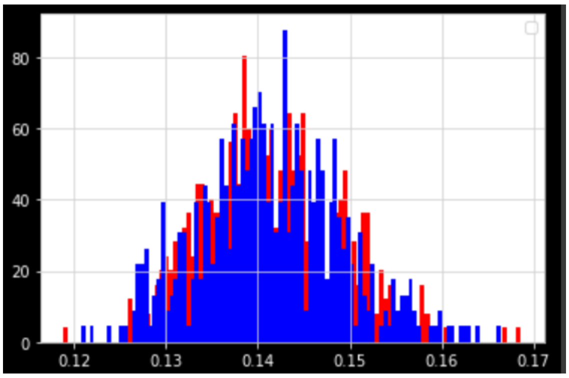
`inter-person variability` indicates different person's changeability in choices.

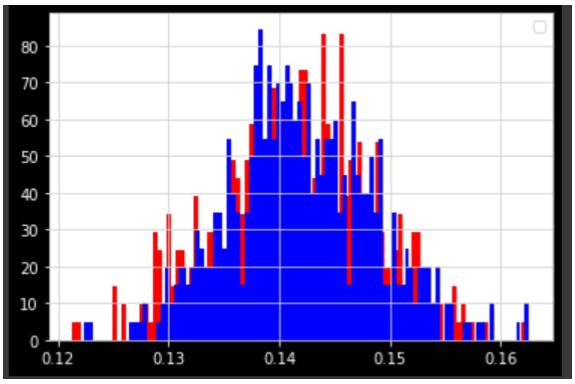
if the inter-person variability = 0, then it indicates the homogeneity of person

however, if it = 20, then it means the heterogeneity between individuals

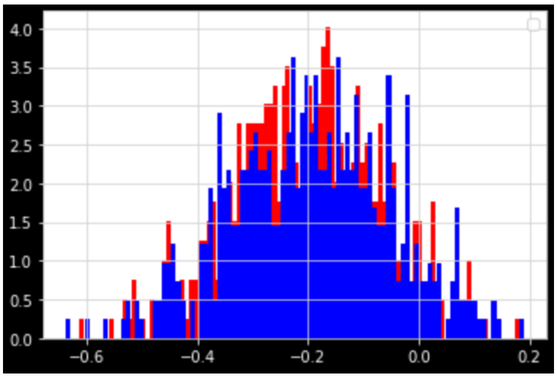
1.2.5

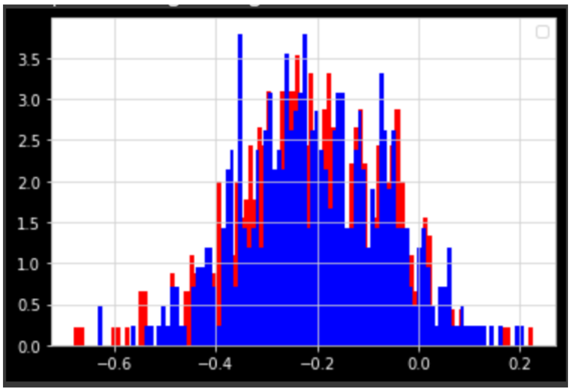
- the red bins indicates the regression analysis under the `block design`
- the blue bins indicates the regression analysis under the `no block design`





- above is the standard errors under different `inter person variability`
  - 1st one is the `block design` with inter person variability of `5`
  - 2nd plot is `no-block design` with inter person variability of `20`





above is the standard errors under different `inter person variability`

- 1st one is the `no-block design` with inter person variability of `5`
- 2nd plot is `block design` with inter person variability of `20`

1.2.6 ***Bonus Question***

change score approach; use the lagged historical wear

1.2.7

if we cut the sample tp `N = 20` and use the inter person variability to be 20

with `no-block design`

	estimated_ate	estimated_ate_se
count	500.000000	500.000000
mean	-0.201915	0.448095
std	0.462198	0.094789
min	-1.495914	0.220221
25%	-0.526280	0.382364
50%	-0.182086	0.441377
75%	0.088735	0.503351
max	1.463598	0.794321

and without `block design`

	estimated_ate	estimated_ate_se
count	500.000000	500.000000
mean	-0.211386	0.312513
std	0.305815	0.052608
min	-1.111531	0.181971
25%	-0.408049	0.277723
50%	-0.224399	0.314055
75%	0.005594	0.346586
max	0.826867	0.464395

2 Regression

2.1

ATE -->

grad_high_school --> dummy variable

2.2

```
B=1000
pvalues1 = []

for i in progressbar(np.arange(0, B)):
  sample = newfunction2(N = 13560, beta_hs = .35, ATE = 0.018)
  model1 = sfa.ols(formula='y ~ d', data=sample).fit()
  pvalues1.append(model1.pvalues.d)

pvalues1 = pd.Series(pvalues1)
stpower = (pvalues1 < .05).mean()
print("statistical power Model 1= ", stpower)
```

---

output: statistical power Model = 0.523

with the same code if we add the `control variable -- grad_high_school` in, then we can observe the new statistical power increases to 0.588

2.3

if we change the parameter `Beta_hs` to 0.1 then we can get

```
#Model 2
pvalues2= []

for i in progressbar(np.arange(0, B)):
  sample = newfunction2(N = 13560, beta_hs = .1, ATE = .018)
  model2 = sfa.ols(formula='y ~ d + grad_high_school', data=sample).fit()
  pvalues2.append(model2.pvalues.d)

pvalues2 = pd.Series(pvalues2)
stpower2 = (pvalues2 < .05).mean()
print(" new statistical power Model 2= ", stpower2)
```

---

output

new statistical power Model 2= 0.619

2.4

2.5

3 Shoe tech experiment redux

```
B=1000
list_noblock = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=False, person_variability=20)
  noblock2 = sfa.ols(formula='y ~ d', data=sample).fit()
  list_noblock.append(noblock2.pvalues.d)

list_noblock = pd.Series(list_noblock)
stpower_noblock = (list_noblock < .05).mean()
print("the statistical power of the non-blocking design = ", stpower_noblock)
```

---

output:

the statistical power of the non-blocking design = 0.048

with the same `block & no-block design`, if we modify the formula to `formula = 'y ~ d + r_yo'` then we can detect that

```
B=1000
list_block = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=False, person_variability=20)
  block2 = sfa.ols(formula='y ~ d + r_y0', data=sample).fit()
  list_block.append(block2.pvalues.d)

list_block = pd.Series(list_block)
stpower_block = (list_block < .05).mean()
print("the statistical power of the blocking design = ", stpower_block)
```

---

output:

the statistical power of the blocking design = 0.271

3.2

3.3

3.4

3.5

Bonus

```
B=5000

list_noblock = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=False, person_variability=20)
  noblock3 = sfa.ols(formula='y ~ d', data=sample).fit()
  list_noblock.append(noblock3.pvalues.d)

list_noblock = pd.Series(list_noblock)
stpower_noblock = (list_noblock < .05).mean()
print("the statistical power of the non-blocking design = ", stpower_noblock)
```

---

output:

the statistical power of the non-blocking design = 0.0574

```python
B = 5000
list_block = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=True, person_variability=20)
  block3 = sfa.ols(formula='y ~ d + r_y0', data=sample).fit()
  list_block.append(block3.pvalues.d)

list_block = pd.Series(list_block)
stpower_block = (list_block < .05).mean()
print("the statistical power of the blocking design = ", stpower_block)
```

---

output:

the statistical power of the non-blocking design = 0.2828
