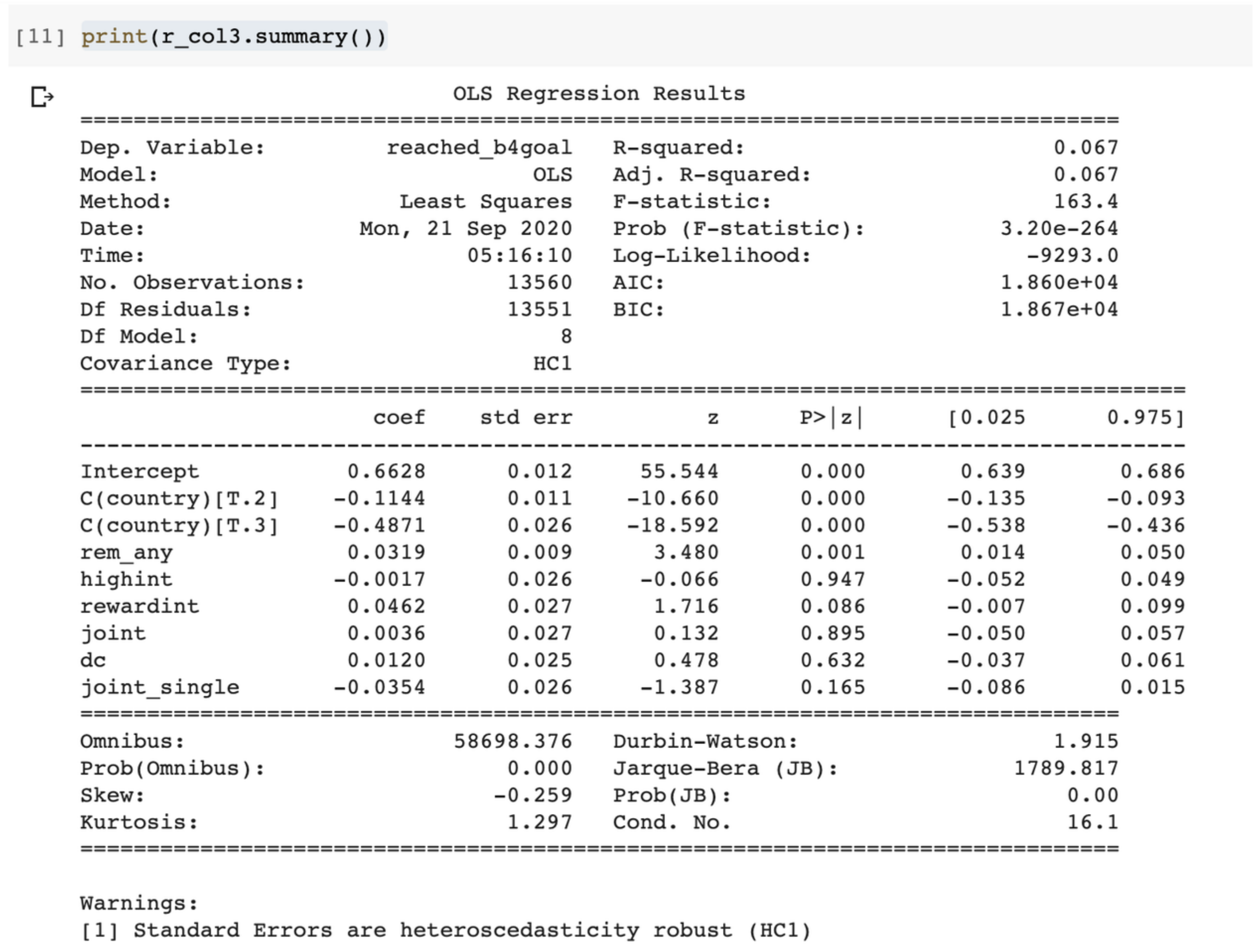
ECON626_Lab 3

Qingwen Wang

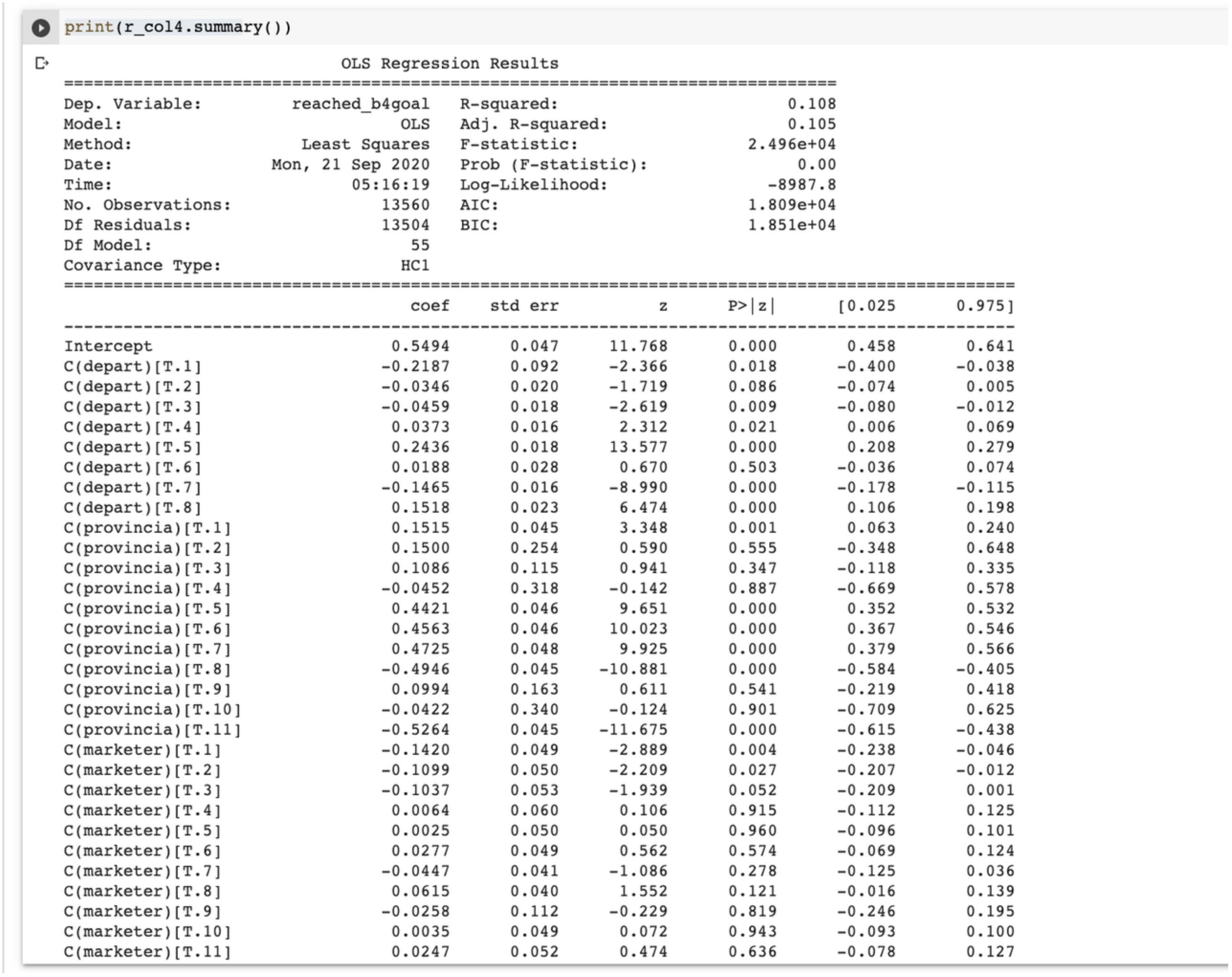
1.1 Replicate and simulate a real study

1.

```
print(r_col3.summary())

print(r_col4.summary())
```



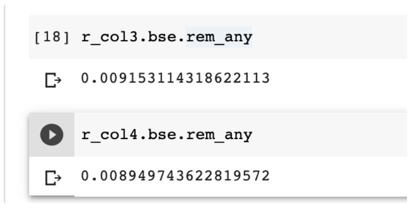
```
[11] print(r_col3.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:          reached_b4goal   R-squared:                       0.067
Model:                             OLS   Adj. R-squared:                  0.067
Method:                  Least Squares   F-statistic:                     163.4
Date:                 Mon, 21 Sep 2020   Prob (F-statistic):          3.20e-264
Time:                         05:16:10   Log-Likelihood:                -9293.0
No. Observations:                13560   AIC:                         1.860e+04
Df Residuals:                    13551   BIC:                         1.867e+04
Df Model:                            8
Covariance Type:                   HC1
=====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            0.6628      0.012     55.544      0.000       0.639       0.686
C(country)[T.2]     -0.1144      0.011    -10.660      0.000      -0.135      -0.093
C(country)[T.3]     -0.4871      0.026    -18.592      0.000      -0.538      -0.436
rem_any              0.0319      0.009      3.480      0.001       0.014       0.050
highint             -0.0017      0.026     -0.066      0.947      -0.052       0.049
rewardint            0.0462      0.027      1.716      0.086      -0.007       0.099
joint                0.0036      0.027      0.132      0.895      -0.050       0.057
dc                   0.0120      0.025      0.478      0.632      -0.037       0.061
joint_single        -0.0354      0.026     -1.387      0.165      -0.086       0.015
==============================================================================
Omnibus:                    58698.376   Durbin-Watson:                   1.915
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1789.817
Skew:                          -0.259   Prob(JB):                         0.00
Kurtosis:                       1.297   Cond. No.                         16.1
==============================================================================

Warnings:
[1] Standard Errors are heteroscedasticity robust (HC1)
```



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

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          reached_b4goal   R-squared:                       0.108
Model:                             OLS   Adj. R-squared:                  0.105
Method:                  Least Squares   F-statistic:                 2.496e+04
Date:                 Mon, 21 Sep 2020   Prob (F-statistic):               0.00
Time:                         05:16:19   Log-Likelihood:                -8987.8
No. Observations:                13560   AIC:                         1.809e+04
Df Residuals:                    13504   BIC:                         1.851e+04
Df Model:                           55
Covariance Type:                   HC1
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
Intercept                  0.5494      0.047     11.768      0.000       0.458       0.641
C(depart)[T.1]            -0.2187      0.092     -2.366      0.018      -0.400      -0.038
C(depart)[T.2]            -0.0346      0.020     -1.719      0.086      -0.074       0.005
C(depart)[T.3]            -0.0459      0.018     -2.619      0.009      -0.080      -0.012
C(depart)[T.4]             0.0373      0.016      2.312      0.021       0.006       0.069
C(depart)[T.5]             0.2436      0.018     13.577      0.000       0.208       0.279
C(depart)[T.6]             0.0188      0.028      0.670      0.503      -0.036       0.074
C(depart)[T.7]            -0.1465      0.016     -8.990      0.000      -0.178      -0.115
C(depart)[T.8]             0.1518      0.023      6.474      0.000       0.106       0.198
C(provincia)[T.1]          0.1515      0.045      3.348      0.001       0.063       0.240
C(provincia)[T.2]          0.1500      0.254      0.590      0.555      -0.348       0.648
C(provincia)[T.3]          0.1086      0.115      0.941      0.347      -0.118       0.335
C(provincia)[T.4]         -0.0452      0.318     -0.142      0.887      -0.669       0.578
C(provincia)[T.5]          0.4421      0.046      9.651      0.000       0.352       0.532
C(provincia)[T.6]          0.4563      0.046     10.023      0.000       0.367       0.546
C(provincia)[T.7]          0.4725      0.048      9.925      0.000       0.379       0.566
C(provincia)[T.8]         -0.4946      0.045    -10.881      0.000      -0.584      -0.405
C(provincia)[T.9]          0.0994      0.163      0.611      0.541      -0.219       0.418
C(provincia)[T.10]        -0.0422      0.340     -0.124      0.901      -0.709       0.625
C(provincia)[T.11]        -0.5264      0.045    -11.675      0.000      -0.615      -0.438
C(marketer)[T.1]          -0.1420      0.049     -2.889      0.004      -0.238      -0.046
C(marketer)[T.2]          -0.1099      0.050     -2.209      0.027      -0.207      -0.012
C(marketer)[T.3]          -0.1037      0.053     -1.939      0.052      -0.209       0.001
C(marketer)[T.4]           0.0064      0.060      0.106      0.915      -0.112       0.125
C(marketer)[T.5]           0.0025      0.050      0.050      0.960      -0.096       0.101
C(marketer)[T.6]           0.0277      0.049      0.562      0.574      -0.069       0.124
C(marketer)[T.7]          -0.0447      0.041     -1.086      0.278      -0.125       0.036
C(marketer)[T.8]           0.0615      0.040      1.552      0.121      -0.016       0.139
C(marketer)[T.9]          -0.0258      0.112     -0.229      0.819      -0.246       0.195
C(marketer)[T.10]          0.0035      0.049      0.072      0.943      -0.093       0.100
C(marketer)[T.11]          0.0247      0.052      0.474      0.636      -0.078       0.127
```

2. from above pictures, we can know that col4 has more control variables, and col3 has a higher r-squared value. And, col4 has a smaller standard error on ATE.

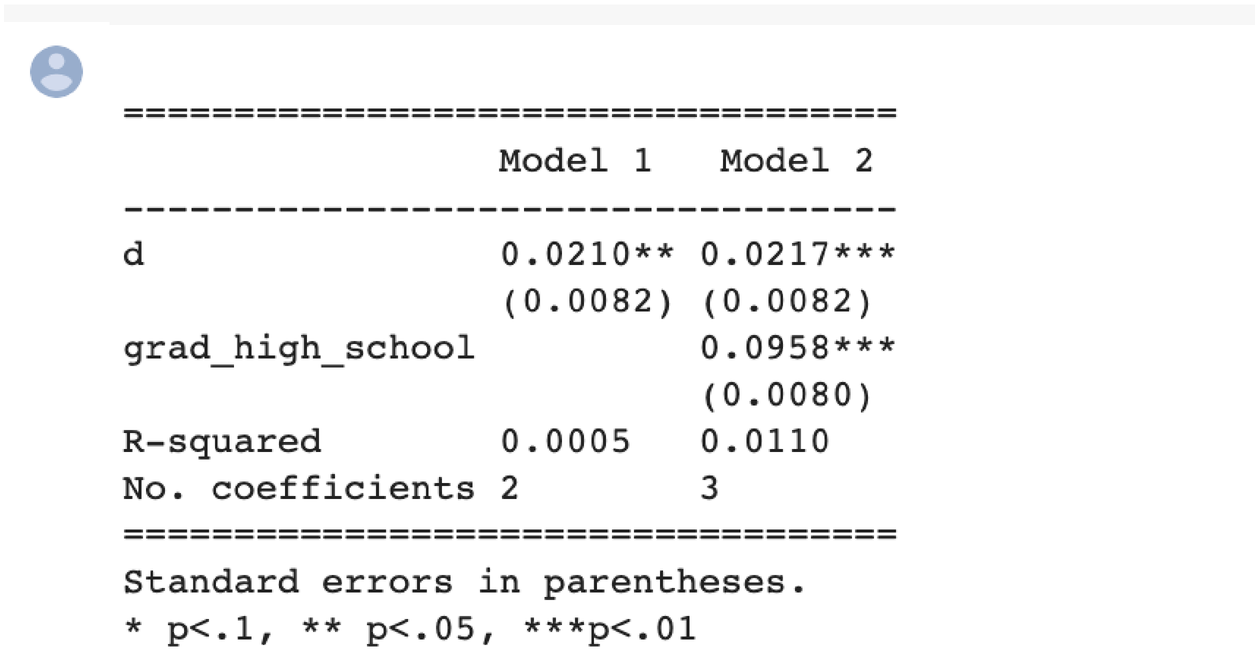


```
[18] r_col3.bse.rem_any

0.009153114318622113

r_col4.bse.rem_any

0.008949743622819572
```

3.
real experiment results:

```
Reproducing table 4 in Karlan et al. 2016

=================================================================================
                    Column 3     Column 4   Only country controls Simplest possible
                  (in paper)  (in paper)     (not in paper)        (not in paper)
---------------------------------------------------------------------------------
rem_any            0.031855*** 0.031861*** 0.031593***            0.024317***
                  (0.009153)  (0.008950)  (0.009125)             (0.008767)
R-squared          0.0671      0.1082      0.0668                 0.0006
No. observations 13560        13560       13560                  13560
No. coefficients 9            61          4                      2
=================================================================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

modeling the simulated data results:



```
=====================================
                   Model 1   Model 2
-------------------------------------
d                  0.0210** 0.0217***
                   (0.0082) (0.0082)
grad_high_school            0.0958***
                            (0.0080)
R-squared          0.0005   0.0110
No. coefficients 2          3
=====================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

in the simulated data model, model2 has one more control variable " grad_high_school" than model 1. R^2 in model 2 is 0.011, higher than R^2 in model 1 (which is 0.005); so model 2 explains more variations in Y than model 1
in real experiments, coefficients is around 0.032, standard error is around 0.009; while in the simulated model the coefficient is around 0.021 , standard error is around 0.008.
for standard error part, simulated one do a good job, coefficients is a bit lower than the original one. However, if we take a look at the R square, the simulated models are

too low, it only explains a little variations in independent variable. So, these simulated models need more improvements.

4.
packaging gdp and modeling into one function; and set ‘beta_hs‘ = 0.35, B = 1000

```
def new_simulation_function(
  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)")
  regr1 = sfa.ols("y ~ d", dff).fit()
  regr2 = sfa.ols("y ~ d + grad_high_school", dff).fit()
  m1 = regr1.params['d']
  m2 = regr2.params['d']
  return m1, m2
```

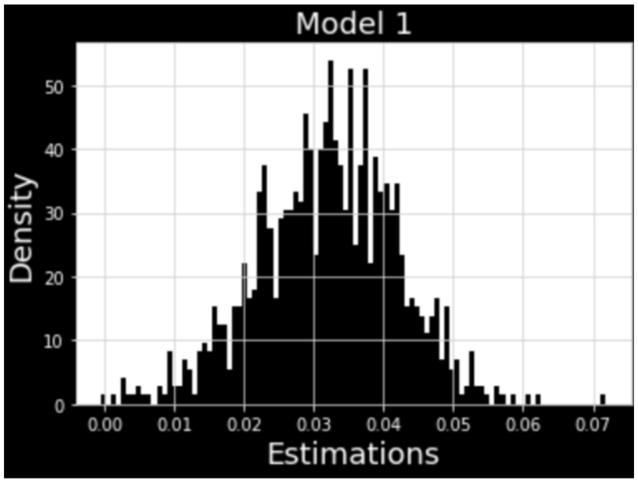
run new function

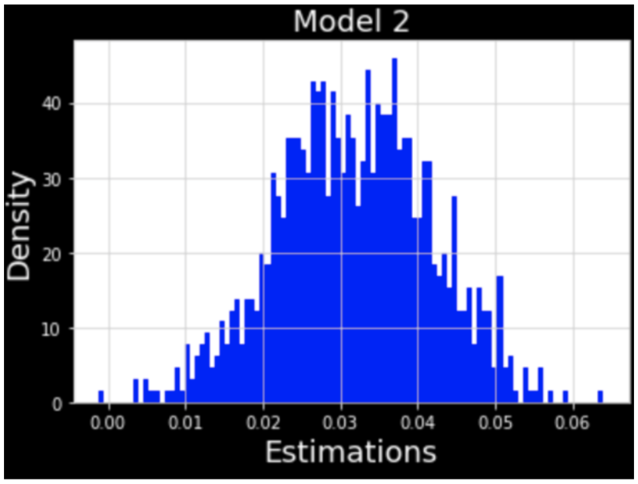
```
B = 1000
res = pd.DataFrame([new_simulation_function() for i in np.aran
ge(0, B)])
```

```
res.describe().T
```

	count	mean	std	min	25%	50%	75%	max
0	1000.0	0.031981	0.009801	0.007126	0.025377	0.032147	0.038830	0.063152
1	1000.0	0.032001	0.009180	0.010804	0.025837	0.031735	0.038173	0.064081

5.
for this question, we replicate what we did from the previous simulation;
stand deviation is 0.0098 for model1 ; 0.00918 for model2.
and we plot the `params['d']` to show the estimated treatment effects(remark : `params['d']` is already including in the newly created function).





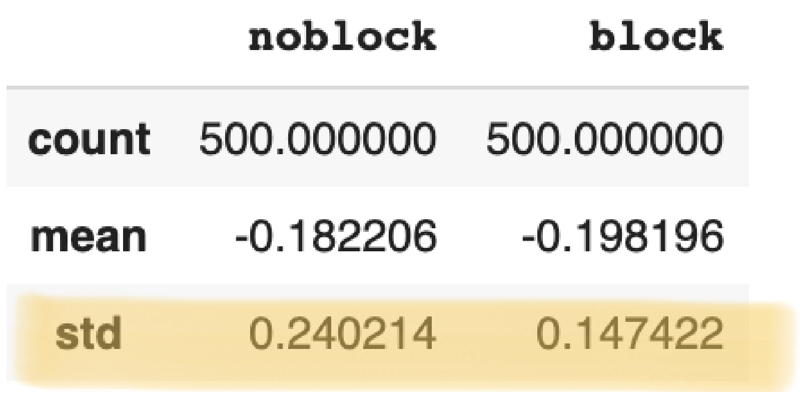
6.
The standard errors in question 4 are still very small, only a little bit bigger than that in question3.

7.
Here I set 95% confidence level.
Thus, confidence interval = Beta-hat +/- 1.96*standard error.

```
confidence interval for model 1: [0.012771972578369074,0.05119020944376145]
confidence interval for model 2: [0.01400846557729125,0.04999382943754023]
```

1.2 Shoe technology experiment

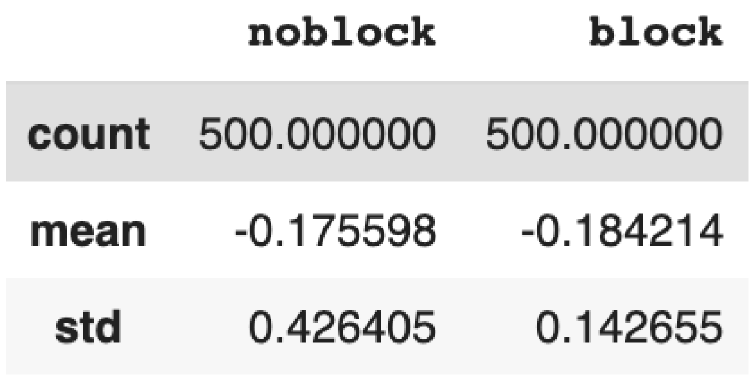
1.



	noblock	block
count	500.000000	500.000000
mean	-0.182206	-0.198196
std	0.240214	0.147422

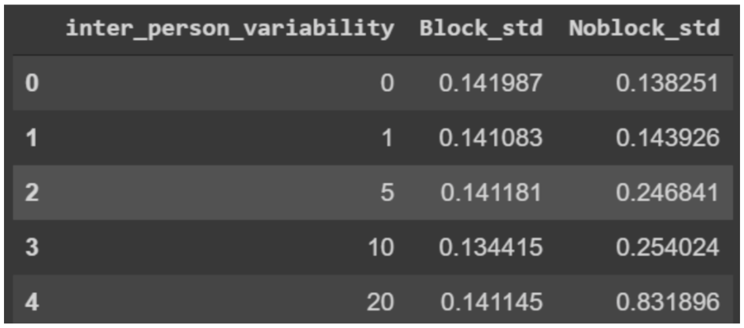
2. Blocked design can help reduces the extraneous variability. So if we increase "inter-person variability", we are expected to see the std will have a increase in the noblock experiment; while will do not have a big difference in the blocked experiment.

Test: Set "inter-person variability" = 10.
std for "noblock" has a significant increase; while std for "block" do not have big difference.
Test result is coincident with my guess.

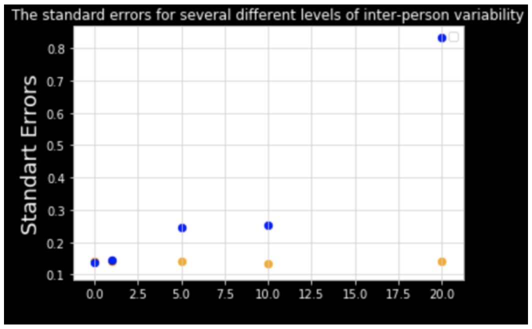


	noblock	block
count	500.000000	500.000000
mean	-0.175598	-0.184214
std	0.426405	0.142655

3. plot: yellow for block std; blue for noblock std.

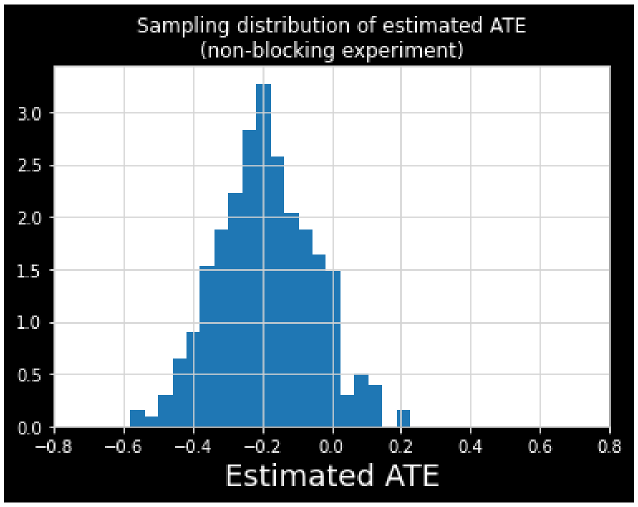


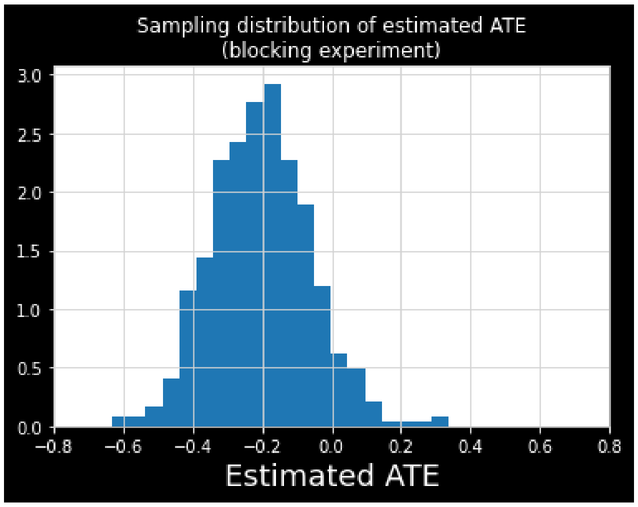
	inter_person_variability	Block_std	Noblock_std
0	0	0.141987	0.138251
1	1	0.141083	0.143926
2	5	0.141181	0.246841
3	10	0.134415	0.254024
4	20	0.141145	0.831896



4. Each person has a personal wear rate that is a function of their weight, strength, and steps per day. This rate depends on the inter-person variability. If inter-person variability = 0, it means all people share the same wear rate; if inter-person variability = 20, it means people's wear rate varies from 0 to 20.
5. set inter_person_variability = 20; the two standard errors have similar values.

```
m_noblock = pd.DataFrame([regression_ate(d) for d in progressb
ar(noblock)])
m_block = pd.DataFrame([regression_ate(d) for d in progressbar
(block)])
```

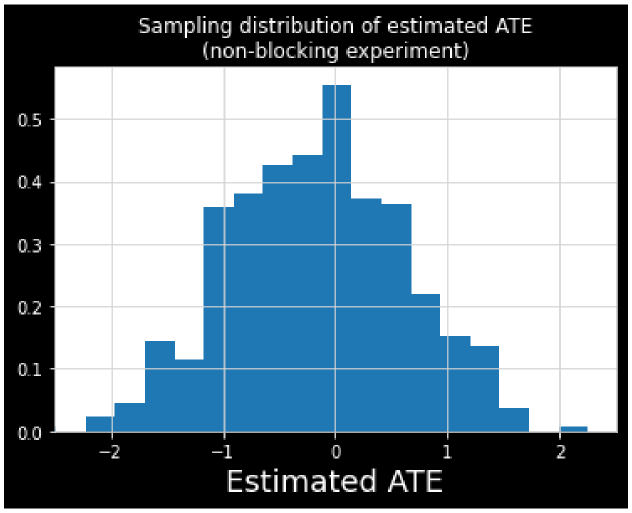


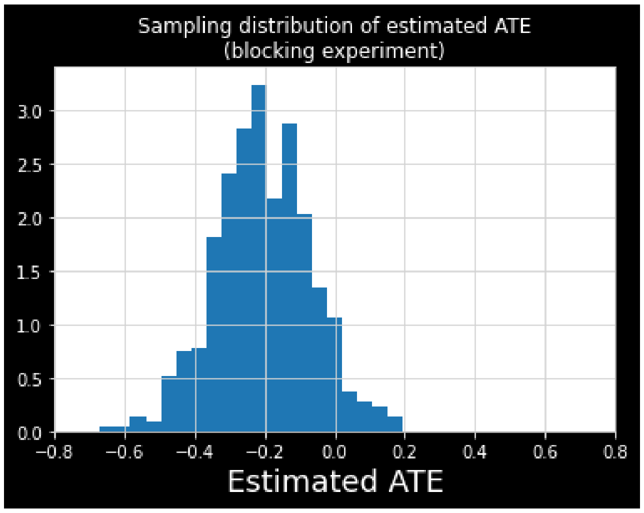


6. keep inter_person_variability = 20; calculate the change score

```
def estimate_ate(df):
  means = df.groupby('d')['y'].mean()
  return {'estimated_ate': means[1] - means[0]}
```

the non-blocking experiment has a large standard error; while the standard error for the blocking experiment is still small. the standard error becomes much larger for the non-blocking, compared to the previous approach.
The distribution of estimate is much widely spread for the non-blocking experiment. As we can see from below picture1 xlim(-2.5 to 2.5) compared to picture2 xlim(-0.8 to 0.8).





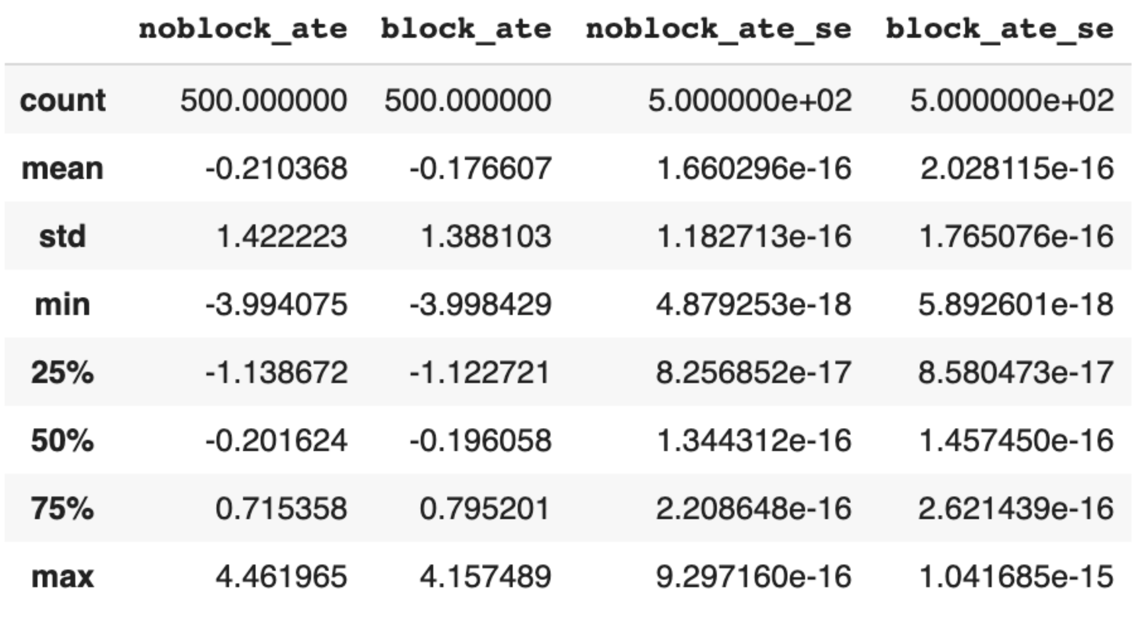
7. N = 20,
   5 pairS of shoes for each person; remove 'person_variability', formula="y ~ d+C(i)"

```
def gen_shoe_data_new(N, block=False):
  people = np.arange(0, N)
  df_left = pd.DataFrame({'i': people})
  df_left['left_foot'] = 1
  df_right = pd.DataFrame({'i': people})
  df_right['left_foot'] = 0
  df = pd.concat([df_left, df_right], axis=0)
  df = pd.concat([df, df], axis=0)
  df = pd.concat([df, df], axis=0)
  df = pd.concat([df, df], axis=0)
  df = pd.concat([df, df], axis=0)
  df['shoe'] = np.arange(0, df.shape[0])
  ATE = -0.20
  Y0 = np.random.normal(loc= 0)
  Y1 = np.random.normal(loc= ATE)
  df['y0'] = Y0
  df['y1'] = Y1

  P_treatment = 0.5
  if block == False:
    treatment_shoes = np.random.choice(df['shoe'], int(len(df
['shoe']) * P_treatment), replace=False)
    df['d'] = df['shoe'].isin(treatment_shoes).astype('float')
  else:
    df['d'] = df['left_foot']

  df['y'] = df.eval("y1 * d + y0 * (1 - d)")
  del df['y0']
  del df['y1']
  return df
```

```
def regression_ate(df, formula="y ~ d+C(i)"):
  r = sfa.ols(formula, df).fit()
  return {
      'estimated_ate': r.params['d'],
      'estimated_ate_se': r.bse['d']
  }
```

```
B = 500
noblock = [gen_shoe_data_new(20, False) for _ in progressbar(n
p.arange(0, B))]
block = [gen_shoe_data_new(20, True) for _ in progressbar(np.a
range(0, B))]
```

```
# Estimate the ATE of each data set
m_noblock = pd.DataFrame([regression_ate(d) for d in progressbar(noblock)])
m_block = pd.DataFrame([regression_ate(d) for d in progressbar(block)])
# Show results
results = pd.concat([
    m_noblock['estimated_ate'].describe(),
    m_block['estimated_ate'].describe(),
    m_noblock['estimated_ate_se'].describe(),
    m_block['estimated_ate_se'].describe()
], axis=1)
results.columns = ['noblock_ate', 'block_ate', 'noblock_ate_se', 'block_ate_se']
results
```
results:



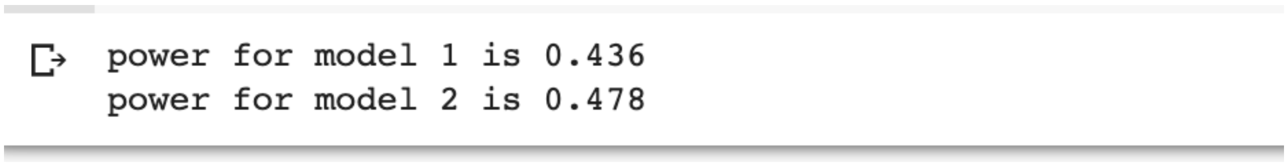
	noblock_ate	block_ate	noblock_ate_se	block_ate_se
count	500.000000	500.000000	5.000000e+02	5.000000e+02
mean	-0.210368	-0.176607	1.660296e-16	2.028115e-16
std	1.422223	1.388103	1.182713e-16	1.765076e-16
min	-3.994075	-3.998429	4.879253e-18	5.892601e-18
25%	-1.138672	-1.122721	8.256852e-17	8.580473e-17
50%	-0.201624	-0.196058	1.344312e-16	1.457450e-16
75%	0.715358	0.795201	2.208648e-16	2.621439e-16
max	4.461965	4.157489	9.297160e-16	1.041685e-15

In this approach, we do not require lots of people to participate in this experiment; the sample size goes down; and we do not need to know every person's wear rate. but the std is higher.

2 Regression

1. ATE in this paper indicates the average treatment effect of how the `reminders`(treatment) increase savings to different groups of individuals; the coefficient means the percentage of people who complete the high school
2. by adding the control variables, the experiment's statistical power increases from 0.436 to 0.478

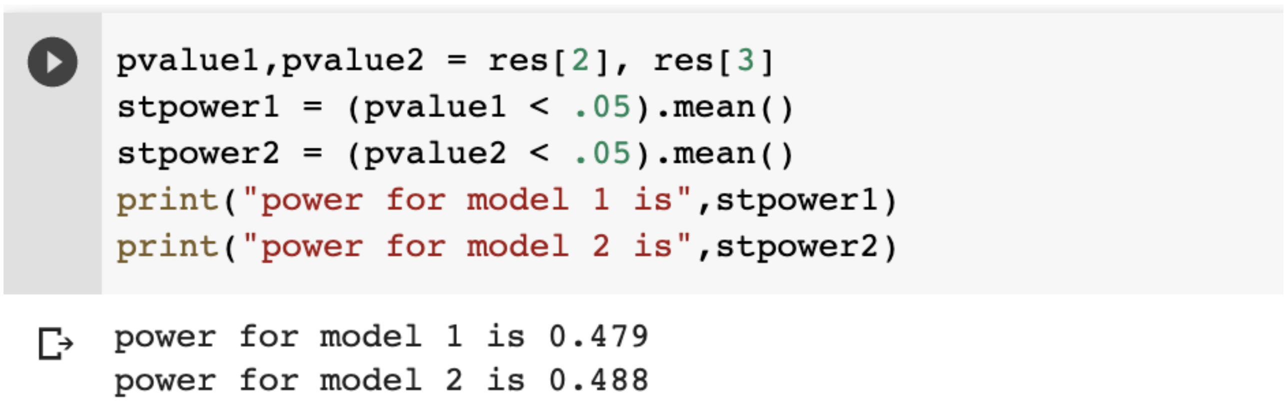
```
B = 1000
res = pd.DataFrame([new_simulation_function(N = 10000,beta_hs = 0.35, ATE = 0.018) for i in np.arange(0, B)])
pvalue1,pvalue2 = res[2], res[3]
stpower1 = (pvalue1 < .05).mean()
stpower2 = (pvalue2 < .05).mean()
print("power for model 1 is",stpower1)
print("power for model 2 is",stpower2)
```



```
power for model 1 is 0.436
power for model 2 is 0.478
```

3. set beta_hs = 0.10;

```
B = 1000
res = pd.DataFrame([new_simulation_function(N = 10000,beta_hs = 0.10, ATE = 0.018) for i in np.arange(0, B)])
```



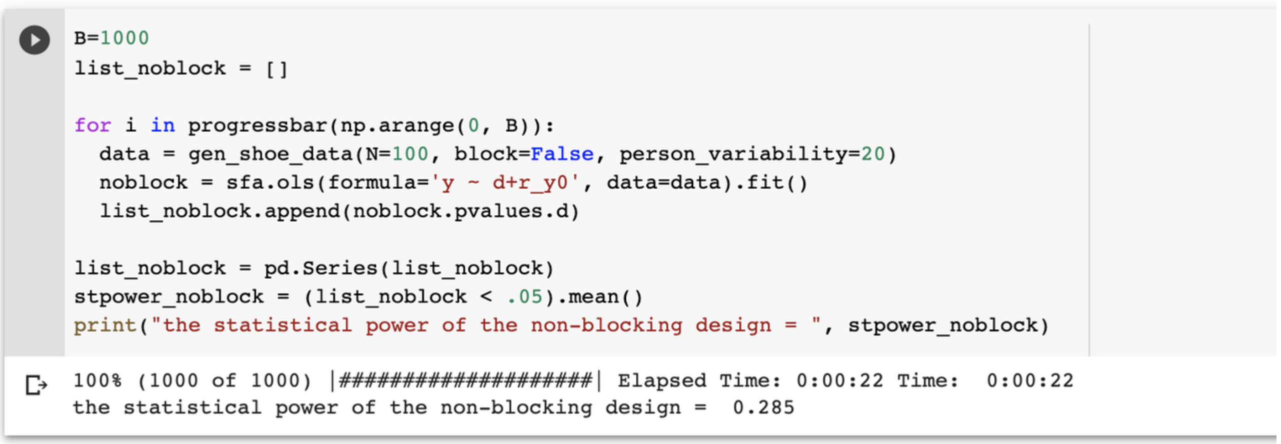
```
pvalue1,pvalue2 = res[2], res[3]
stpower1 = (pvalue1 < .05).mean()
stpower2 = (pvalue2 < .05).mean()
print("power for model 1 is",stpower1)
print("power for model 2 is",stpower2)
```

```
power for model 1 is 0.479
power for model 2 is 0.488
```

4. from the previous simulation, we can observe that when we lower the `beta_hs`, we got a higher experiment's statistical power; when less portion of people is high_school graduated, 'grad_high_school` explains less while 'd' explains more.
5. Adding the additional covariates increased the R-squared; it helps better explain the variation in Y. In the data collecting process, we expect the control variables to be less correlated with the independent variable.

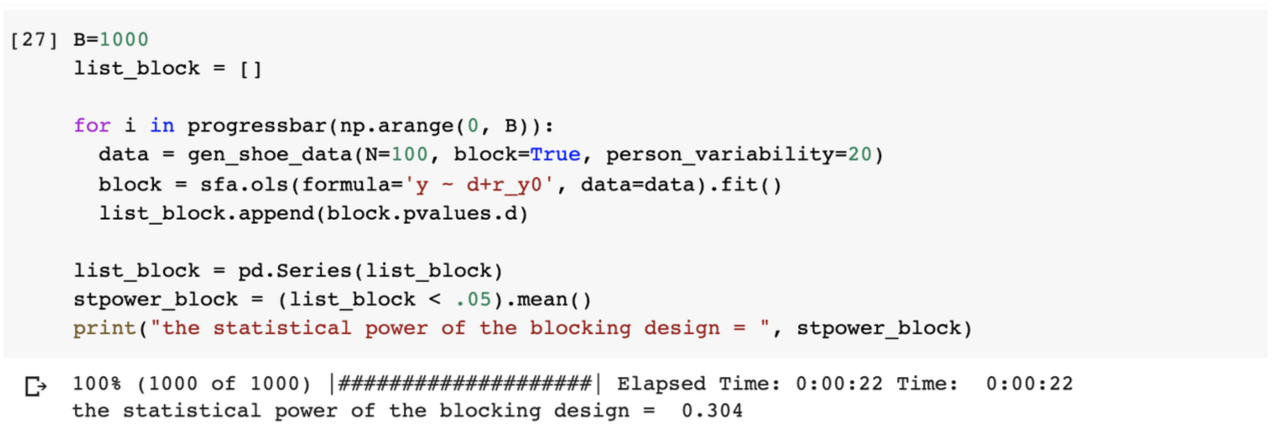
3 Shoe tech experiment rudex

1. for non-blocking design, statistical power = 0. 285; for blocking design, statistical power = 0.304



```
B=1000
list_noblock = []

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

```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:22 Time:  0:00:22
the statistical power of the non-blocking design =  0.285
```



```
[27] B=1000
     list_block = []

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

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

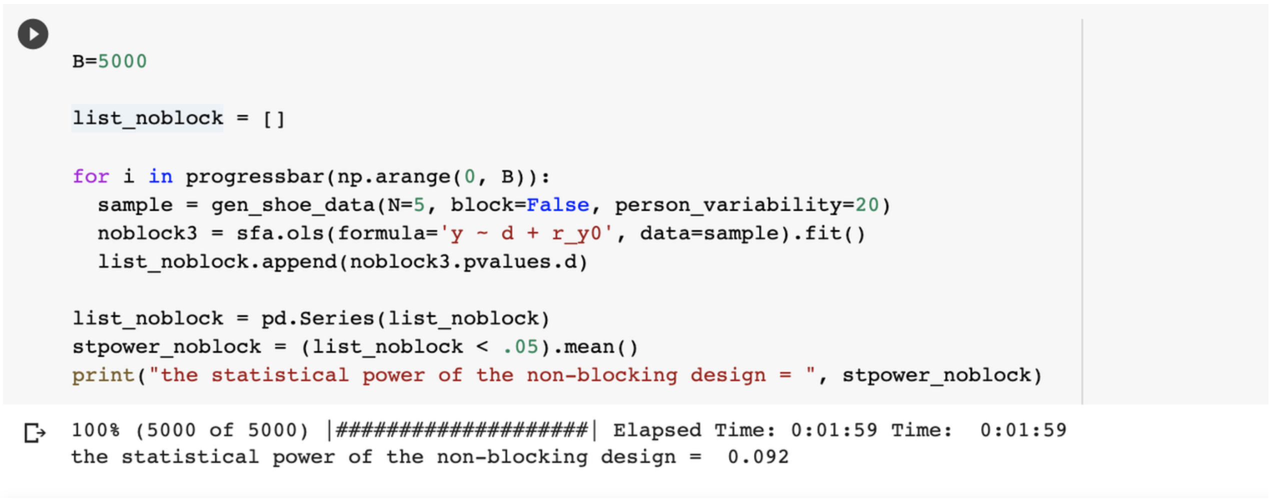
```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:22 Time:  0:00:22
the statistical power of the blocking design =  0.304
```

2. blocking can increase power: because by blocking similar units into the a block and splitting each block into treatment and control group can balance the characteristics across control and treatment groups and ultimately minimize the extraneous variation in our estimated ATE.

3. regression with covariates can increase power: because adding covariates and control variables can soak up the variations in the outcome and make the estimations more precise ; and adding the additional covariates increased the R-squared; it helps better explain the variation in Y.
4. "Change scores" analysis the before and after differences can increase power; because it can increase effect size that will increase power. The larger the effect size, the more likely it is that an experiment would find a significant effect.
5. For blocking, each person effectively serves as their own control group. It is a completely randomized design. But this technique is effective only when there is a high heterogeneity between the people. like what we implemented before the `inter person variability` parameter.
   For the estimation of changing scores techniques, each observation serves as their own control. we can add pre-treatment covariates that predict the outcome, but with this approach, we can not rollback.
   For the use of covariates in the regression, it is a very robust technique. Also, it is very convenient for researchers to operate. But choosing the right covariates, unbiased control variables can be challenging.

Bonus

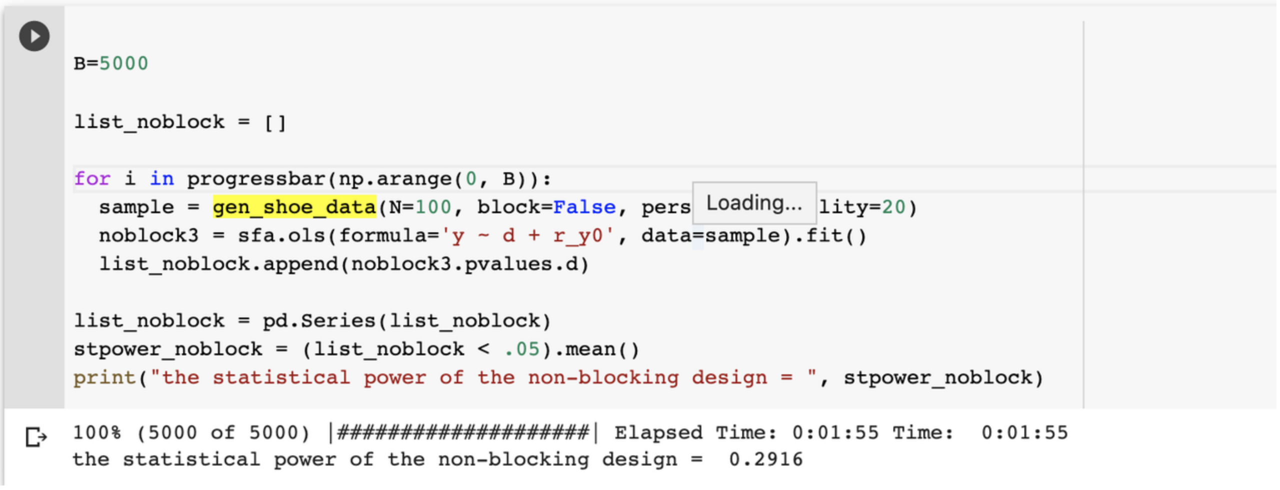
N = 5 ; The statistical power of the non-blocking design = 0.092



```
B=5000

list_noblock = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=5, block=False, person_variability=20)
  noblock3 = sfa.ols(formula='y ~ d + r_y0', 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)
```

```
100% (5000 of 5000) |####################| Elapsed Time: 0:01:59 Time:  0:01:59
the statistical power of the non-blocking design =  0.092
```

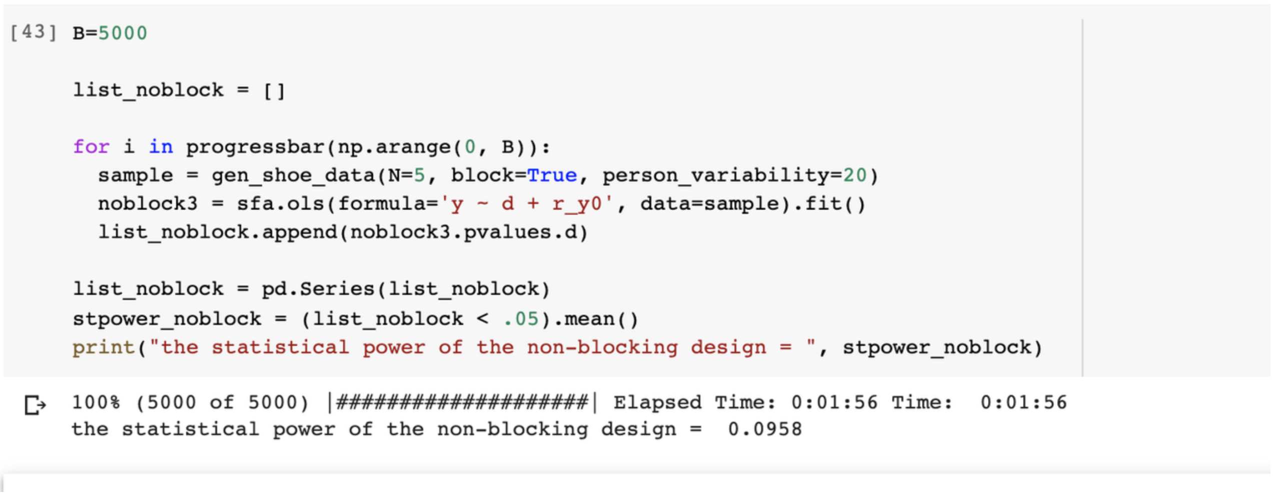
N = 100 ; The statistical power of the non-blocking design = 0.2916



```
B=5000

list_noblock = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=False, pers Loading... lity=20)
  noblock3 = sfa.ols(formula='y ~ d + r_y0', 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)
```

```
100% (5000 of 5000) |####################| Elapsed Time: 0:01:55 Time:  0:01:55
the statistical power of the non-blocking design =  0.2916
```

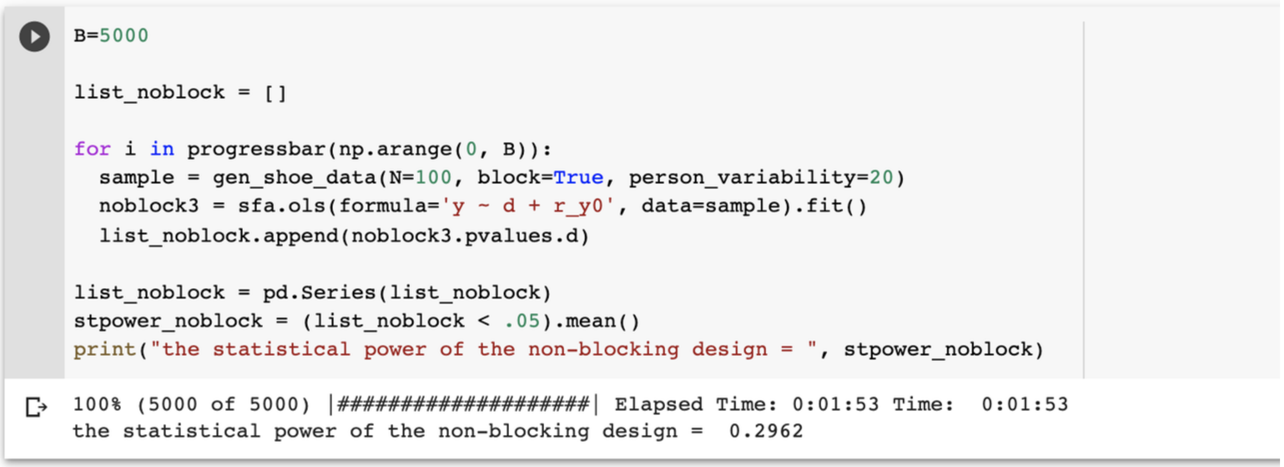
N = 5 ; The statistical power of the non-blocking design = 0.0958



```
[43] B=5000

     list_noblock = []

     for i in progressbar(np.arange(0, B)):
       sample = gen_shoe_data(N=5, block=True, person_variability=20)
       noblock3 = sfa.ols(formula='y ~ d + r_y0', 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)
```

```
100% (5000 of 5000) |####################| Elapsed Time: 0:01:56 Time:  0:01:56
the statistical power of the non-blocking design =  0.0958
```

N = 100 ; The statistical power of the blocking design = 0.2962



```
B=5000

list_noblock = []

for i in progressbar(np.arange(0, B)):
  sample = gen_shoe_data(N=100, block=True, person_variability=20)
  noblock3 = sfa.ols(formula='y ~ d + r_y0', 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)
```

```
100% (5000 of 5000) |####################| Elapsed Time: 0:01:53 Time:  0:01:53
the statistical power of the non-blocking design =  0.2962
```

So, we can observe that one N increase, power increase; and the power of blocking one is higher than that of the unblocking one.