ECON626_Lab4

Qingwen Wang

1 Adjusting randomization proportions during experiment

1.1 In class foundation

1. $ATE = E[Y1] - E[Y0] = 0.029 \approx 0.03$

   $NATE = E[Y^1 | D = 1] - E[Y^0 | D = 0] = -0.01185999212058414$

   $selectionbias = E[Y^0 | D = 1] - E[Y^0 | D = 0] = -0.04120887943836349$

$Differentialeffectbias = NATE - ATE - selectionbias = 0.0002347834200250$

We have selection bias = -0.04; differential effect bias = 0.0002; However, differential effect bias is very trivial in this case. Hence, we can say that selection bias is the main bias here.

```
[17] print("E[Y1|D=1] - E[Y0|D=0] = {}".format(df.query("d == 1")['y1'].mean() - df.query("d == 0")['y0'].mean()))
     print("E[Y0|D=1] - E[Y0|D=0] = {}".format(df.query("d == 1")['y0'].mean() - df.query("d == 0")['y0'].mean()))
     print("E[Y0|D=1] - E[Y0|D=0] = {}".format(df['y1'].mean() - df['y0'].mean()))

E[Y1|D=1] - E[Y0|D=0] = -0.01185999212058414
E[Y0|D=1] - E[Y0|D=0] = -0.04120887943836349
E[Y0|D=1] - E[Y0|D=0] = 0.029114103897754345
```

```
[29] nate = df.query("d == 1")['y1'].mean() - df.query("d == 0")['y0'].mean()
     sb = df.query("d == 1")['y0'].mean() - df.query("d == 0")['y0'].mean()
     ate = df['y1'].mean() - df['y0'].mean()
     nate - sb -ate

0.0002347834200250032
```

2. Independence assumption holds when $E[Y^0 | D] = E[Y^0], E[Y^1 | D] = E[Y^1]$, while these two conditions violates, so Independence assumption does not hold in this dataset.

```
print("E[Y0|D=0] - E[Y0] = {}".format(df.query("d == 0")['y0'].mean() - df['y0'].mean()))
print("E[Y0|D=1] - E[Y0] = {}".format(df.query("d == 1")['y0'].mean() - df['y0'].mean()))
print("E[Y1|D=0] - E[Y1] = {}".format(df.query("d == 0")['y1'].mean() - df['y1'].mean()))
print("E[Y1|D=1] - E[Y1] = {}".format(df.query("d == 1")['y1'].mean() - df['y1'].mean()))

E[Y0|D=0] - E[Y0] = 0.013982621736781747
E[Y0|D=1] - E[Y0] = -0.02722625770158174
E[Y1|D=0] - E[Y1] = 0.013862043734903573
E[Y1|D=1] - E[Y1] = -0.026991474281556738
```

3. ate = 0.0291, matches what we set (ate = 0.03)in the data-generating process.

```
ate = df['y1'].mean() - df['y0'].mean()
ate
```

```
0.029114103897754345
```

4. modify ‘STRATA_VARIABLE’ to ‘day_of_week’. We can get NATE = 0.02938, ATE = 0.02938; differential effect bias = 0; and selection bias = 0; so selection bias is removed here. We can get the unbiased estimate because we stratify the data into several subgroups (strata); and we put each person in the experiment into exactly the right group such that all selection bias is accounted for. In this case, the randomization proportions is affected by the day of the week. That is, the conditional independence assumption holds within each stratum.

nate = 0.02938

```
nate_df = df.groupby([STRATA_VARIABLE, 'd'])['y0','y1'].mean().unstack(-1).iloc[:,[0, 3]]
nate_counts_by_strata = df.groupby([STRATA_VARIABLE])['y'].count()
nate_df.columns = ['control', 'treatment']
nate_df['counts'] = nate_counts_by_strata
nate_df['difference'] = nate_df.eval("treatment - control")
print("nate = {}".format(nate_df.eval("difference * counts").sum() / nate_df['counts'].sum()))
```

```
nate = 0.029383135157494574
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tu
  """Entry point for launching an IPython kernel.
```

ate = 0.02938:

```
# Calculate an estimate of the ATE for each day of the week.
STRATA_VARIABLE = 'day_of_week'
conversion_rates_by_strata = df.groupby([STRATA_VARIABLE, 'd'])['y'].mean().unstack(-1)
conversion_rates_by_strata.columns = ['control', 'treatment']
conversion_rates_by_strata['difference'] = conversion_rates_by_strata.eval("treatment - control")

counts_by_strata = df.groupby([STRATA_VARIABLE])['y'].count()

conversion_rates_by_strata['counts'] = counts_by_strata

display(conversion_rates_by_strata)

print("\nEstimate of ATE (perfect stratification) = {:.5f}".format(
      (conversion_rates_by_strata.eval("difference * counts").sum() / conversion_rates_by_strata['counts'].sum()))
)
```

day_of_week	control	treatment	difference	counts
0	0.301443	0.327008	0.025566	200130
1	0.301645	0.329106	0.027462	200206
2	0.299382	0.330466	0.031084	200326
3	0.300361	0.329135	0.028774	200562
4	0.299397	0.333436	0.034039	199548
5	0.101000	0.130139	0.029140	199992
6	0.099654	0.129285	0.029631	199863

```
Estimate of ATE (perfect stratification) = 0.02938
```

selection bias = 0:

```
sb_df = df.groupby([STRATA_VARIABLE, 'd'])['y0'].mean().unstack(-1)
sb_counts_by_strata = df.groupby([STRATA_VARIABLE])['y0'].count()
sb_df.columns = ['control', 'treatment']
sb_df['counts'] = sb_counts_by_strata
sb_df['difference'] = sb_df.eval("treatment - control")

print("selection bias = {}".format(
      (sb_df.eval("difference * counts").sum() / sb_df['counts'].sum()))
)
```

```
selection bias = -7.562299626165767e-05
```

differential effect bias = 0:

```
nate_df.eval("difference * counts").sum() / nate_df['counts'].sum()
- sb_df.eval("difference * counts").sum() / sb_df['counts'].sum()
- conversion_rates_by_strata.eval("difference * counts").sum() / conversion_rates_by_strata['counts'].sum()
```

```
7.562299626165739e-05
```

5. naive estimate of ate: = -0.011

```
r_d = sfa.ols("y ~ d ", df).fit(cov_type='HC1')
print(r_d.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     209.5
Date:                Mon, 28 Sep 2020   Prob (F-statistic):           1.82e-47
Time:                        03:10:30   Log-Likelihood:            -8.2045e+05
No. Observations:             1400799   AIC:                         1.641e+06
Df Residuals:                 1400797   BIC:                         1.641e+06
Df Model:                           1
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.2567      0.000    565.398      0.000       0.256       0.258
d             -0.0112      0.001    -14.473      0.000      -0.013      -0.010
==============================================================================
Omnibus:                   276603.374   Durbin-Watson:                   1.919
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           330842.098
Skew:                           1.137   Prob(JB):                         0.00
Kurtosis:                       2.293   Cond. No.                         2.41
==============================================================================

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

add ‘day_of_week‘ as a control variable, estimate ate = 0.03, unbiased.

```
r_day = sfa.ols("y ~ d + C(day_of_week) ", df).fit(cov_type='HC1')
print(r_day.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.041
Model:                            OLS   Adj. R-squared:                  0.041
Method:                 Least Squares   F-statistic:                 1.177e+04
Date:                Mon, 28 Sep 2020   Prob (F-statistic):               0.00
Time:                        03:13:51   Log-Likelihood:            -7.9090e+05
No. Observations:             1400799   AIC:                         1.582e+06
Df Residuals:                 1400791   BIC:                         1.582e+06
Df Model:                           7
Covariance Type:                  HC1
=========================================================================================
                            coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Intercept                 0.3000      0.001    285.397      0.000       0.298       0.302
C(day_of_week)[T.1]       0.0006      0.001      0.440      0.660      -0.002       0.004
C(day_of_week)[T.2]   -1.133e-05      0.001     -0.008      0.994      -0.003       0.003
C(day_of_week)[T.3]      -0.0008      0.001     -0.520      0.603      -0.004       0.002
C(day_of_week)[T.4]      -0.0005      0.001     -0.330      0.741      -0.003       0.002
C(day_of_week)[T.5]      -0.2004      0.001   -158.681      0.000      -0.203      -0.198
C(day_of_week)[T.6]      -0.2006      0.001   -158.779      0.000      -0.203      -0.198
d                         0.0302      0.001     39.032      0.000       0.029       0.032
==============================================================================
Omnibus:                   256869.668   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           284566.395
Skew:                           1.044   Prob(JB):                         0.00
Kurtosis:                       2.284   Cond. No.                         8.35
==============================================================================
```

1.2 homework extension

1. run a perfect stratification analysis by the hour of the day, we get estimated ate = -0.011, which is a biased estimate. Hour of the day is not a pattern like day of the week - there's hardly any trend of conversion rate based on hour. Stratifying the data into several strata by hour can not ensure that each person in the experiment are put into exactly the right group such that all selection bias is accounted for. The experiment last for 14 days and data is generated based on days. While hour is only randomly added to units. If our data is generated based on hours during day, then the stratification may work.

```
[13] STRATA_VARIABLE = 'hour_of_day'
     conversion_rates_by_strata = df.groupby([STRATA_VARIABLE, 'd'])['y'].mean().unstack(-1)
     conversion_rates_by_strata.columns = ['control', 'treatment']
     conversion_rates_by_strata['difference'] = conversion_rates_by_strata.eval("treatment - control")
     counts_by_strata = df.groupby([STRATA_VARIABLE])['y'].count()
     conversion_rates_by_strata['counts'] = counts_by_strata
     print("\nEstimate of ATE (perfect stratification) = {:.5f}".format(
         (conversion_rates_by_strata.eval("difference * counts").sum() / conversion_rates_by_strata['counts'].sum()))
     )
```

```
Estimate of ATE (perfect stratification) = -0.01117
```

2. modify the data-generating process:

```
def daily_ate(day_of_week):
  if day_of_week in [5, 6]:
    return 0.005
  else:
```

```
    return 0.10
p_y0 = df['day_of_week'].apply(daily_conversion_rates)
p_ate = df['day_of_week'].apply(daily_ate)
df['y0'] = np.random.binomial(n=1, p=p_y0).astype('int8')
df['y1'] = np.random.binomial(n=1, p=p_y0 + p_ate).astype('int
8')
```

true ATE = 0.0724

```
df['y1'].mean()-df['y0'].mean()

0.07235135654773092
```

estimated ate with regression = 0.0683

```
r_day_new = sfa.ols("y ~ d + C(day_of_week) ", df).fit(cov_type='HC1')
print(r_day_new.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.059
Model:                            OLS   Adj. R-squared:                  0.059
Method:                 Least Squares   F-statistic:                 1.715e+04
Date:                Mon, 28 Sep 2020   Prob (F-statistic):               0.00
Time:                        05:25:00   Log-Likelihood:            -7.9587e+05
No. Observations:             1399103   AIC:                         1.592e+06
Df Residuals:                 1399095   BIC:                         1.592e+06
Df Model:                           7
Covariance Type:                  HC1
===========================================================================================
                              coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------------
Intercept                   0.3097      0.001    291.459      0.000       0.308       0.312
C(day_of_week)[T.1]        -0.0002      0.001     -0.145      0.884      -0.003       0.003
C(day_of_week)[T.2]        -0.0011      0.001     -0.731      0.465      -0.004       0.002
C(day_of_week)[T.3]        -0.0009      0.001     -0.584      0.559      -0.004       0.002
C(day_of_week)[T.4]         0.0019      0.001      1.274      0.203      -0.001       0.005
C(day_of_week)[T.5]        -0.2430      0.001   -192.600      0.000      -0.245      -0.240
C(day_of_week)[T.6]        -0.2413      0.001   -190.833      0.000      -0.244      -0.239
d                           0.0683      0.001     86.708      0.000       0.067       0.070
==============================================================================
Omnibus:                   297921.935   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           250217.765
Skew:                           0.946   Prob(JB):                         0.00
Kurtosis:                       2.158   Cond. No.                         8.35
==============================================================================
```

estimate ate with perfect stratification = 0.0728

```python
STRATA_VARIABLE = 'day_of_week'
conversion_rates_by_strata = df.groupby([STRATA_VARIABLE, 'd'])['y'].mean().unstack(-1)
conversion_rates_by_strata.columns = ['control', 'treatment']
conversion_rates_by_strata['difference'] = conversion_rates_by_strata.eval("treatment - control")
counts_by_strata = df.groupby([STRATA_VARIABLE])['y'].count()
conversion_rates_by_strata['counts'] = counts_by_strata
display(conversion_rates_by_strata)
print("\nEstimate of ATE (perfect stratification) = {:.5f}".format(
    (conversion_rates_by_strata.eval("difference * counts").sum() / conversion_rates_by_strata['counts'].sum()))
)
```

	control	treatment	difference	counts
day_of_week				
0	0.300219	0.402952	0.102733	199992
1	0.299788	0.403181	0.103393	199807
2	0.301157	0.396411	0.095254	199869
3	0.300775	0.398302	0.097527	200096
4	0.303007	0.402388	0.099381	199756
5	0.097405	0.104363	0.006958	199725
6	0.100190	0.104884	0.004694	199858

```
Estimate of ATE (perfect stratification) = 0.07286
```

The perfect stratification gives the unbiased ate, while the estimated ate with the regression is still biased. Regression can be a convenient tool to implement a stratification, but it still can fail to give unbiased estimate when ATE are not same within each strata. (in which we have 0.005 and 0.01 in weekend and weekdays); while perfect stratification still works.

3. By specifying the regression this way, we have implicitly assumed that the ATE is the same within each strata. This assumption is a substantive one. It happens to hold in this example, but it is not always true. The regression approach can give a biased estimate in those cases.
   For the perfect stratification, it put each person into the group that the selection bias is accounted for. The ATE is the average of these subgroups, which balance the bias out. So, the perfect stratification is more robust.

2 One-sided non-compliance in a web experiment

2.1 In class foundation

1. the assignment variable Z refers to 'coin_flip'; the actual treatment variable D refers to 'saw_treatment_page'.
2. 'saw_treatment_page == coin_flip' indicates people always do as they are assigned, which means $D_i^1 = 1$ and $D_i^0 = 0$. So, it selects compliers.

'viewed_page == 1' indicates those anyone who view the treatment page, they can be either compliers in treatment group($D_i^1 = 1$) or defiers in control group($D_i^0 = 1$). So, it selects always takers.
The main difference is that the 'saw_treatment_page == coin_flip' set contains $D_i^0 = 0$ (people who are assigned to control group and also actually do not take the treatment); while 'viewed_page == 1' set contains $D_i^0 = 1$(people who are assigned to control group while actually take the treatment).

3. as-treated estimate is 0.281, it is a biased one. The as-treated approach compares groups in terms of their actual treatment, and ignores non-compliance in the analysis. However, when there is non-compliance, our treatment and control groups are no longer correctly randomized. Each person's actual treatment may have correlation with other variables(i.e. charitability). Non-compliance can invalidate the independence assumption. Randomization is broken.

Estimation methods

	Dependent variable: P(Donation)			
	As-treated (1)	Per-protocol (2)	ITT (3)	CACE (4)
Intercept	0.264*** (0.001)	0.284*** (0.001)	0.284*** (0.001)	0.344*** (0.001)
coin_flip			0.1*** (0.001)	0.201*** (0.001)
saw_treatment_page	0.281*** (0.001)	0.261*** (0.001)		
Observations	1000000.0	749736.0	1000000.0	499848.0

4. It returns us the ATE(0.10/0.5 = 0.2) with dilution(0.5). In an intention-to-treat analysis, we consider the assignment itself to be the treatment. We calculate the donation rate among the control and compare it to the donation rate among the treatment, regardless of whether they actually take the web page treatment.
   The 'charitability' and 'viewed_page' (a function of 'charitability') only determines how people will actually take treatment or not, which is considered in the ITT analysis. And, these two variables have no relationship with 'coin_flip'.

```python
r_itt_1 = sfa.ols("y ~ coin_flip + charitability", df).fit(cov_type='HC1')
print(r_itt_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.130
Model:                            OLS   Adj. R-squared:                  0.130
Method:                 Least Squares   F-statistic:                 8.714e+04
Date:                Mon, 28 Sep 2020   Prob (F-statistic):               0.00
Time:                        19:32:41   Log-Likelihood:            -5.9789e+05
No. Observations:             1000000   AIC:                         1.196e+06
Df Residuals:                  999997   BIC:                         1.196e+06
Df Model:                           2
Covariance Type:                  HC1
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept         0.2845      0.001    469.280      0.000       0.283       0.286
coin_flip         0.1003      0.001    113.998      0.000       0.099       0.102
charitability     0.1626      0.000    395.274      0.000       0.162       0.163
==============================================================================
Omnibus:                   692538.085   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           105025.206
Skew:                           0.540   Prob(JB):                         0.00
Kurtosis:                       1.837   Cond. No.                         2.62
==============================================================================
```

```python
r_itt_2 = sfa.ols("y ~ coin_flip + viewed_page", df).fit(cov_type='HC1')
print(r_itt_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.066
Model:                            OLS   Adj. R-squared:                  0.066
Method:                 Least Squares   F-statistic:                 3.467e+04
Date:                Mon, 28 Sep 2020   Prob (F-statistic):               0.00
Time:                        19:32:45   Log-Likelihood:            -6.3341e+05
No. Observations:             1000000   AIC:                         1.267e+06
Df Residuals:                  999997   BIC:                         1.267e+06
Df Model:                           2
Covariance Type:                  HC1
================================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       0.1738      0.001    232.049      0.000       0.172       0.175
coin_flip       0.0998      0.001    109.512      0.000       0.098       0.102
viewed_page     0.2210      0.001    242.416      0.000       0.219       0.223
==============================================================================
Omnibus:                  2686257.762   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           137297.089
Skew:                           0.642   Prob(JB):                         0.00
Kurtosis:                       1.717   Cond. No.                         3.19
==============================================================================
```

5. cace = itt/ complier_ration = 0.2. ITT analyze data based on treatment assignment rather than treatment received, so we can easily get complier

average causal effects by dividing by complier ratio.

```
complier_ratio = df.query('viewed_page == coin_flip').unit.count()/df.unit.count()
itt = r_itt.params['coin_flip']
cace = itt/complier_ratio
round(cace,3)
```

```
0.2
```

2.2 Heterogeneous treatment effects

1.

```
df.y1.mean() - df.y0.mean()
```

```
0.14884599999999998
```

2. In the last problem2.1, we successfully get our unbiased ate. And the true ate we have calculated is 0.149. Comparing to the below table, all four methods fails to identify the true treatment effect. When the advertisement has a stronger behavioral effect on more charitable visitors, our methods used before does not work again.

Estimation methods

	Dependent variable: P(Donation)y			
	As-treated	Per-protocol	ITT	CACE
	(1)	(2)	(3)	(4)
Intercept	0.264***	0.284***	0.284***	0.344***
	(0.001)	(0.001)	(0.001)	(0.001)
coin_flip			0.087***	0.175***
			(0.001)	(0.001)
saw_treatment_page	0.256***	0.235***		
	(0.001)	(0.001)		
Observations	1,000,000	749,736	1,000,000	499,848

3. The key difference is that we replaced `ATE` with `ITE = 0.3*scipy.special.expit(charitability)`. CACE approach does not work any more in this case3. P(view) is correlated with P(donate) and ITE, too many correlation for us to identify ATE.
4. Since `viewed_page` is no longer reliable, so we throw away this variable in the dataset; instead we can roughly use `charitability` to indicates the possibility the people actually take the treatment; so here I assume anyone who has over average charitability will be actually take treatment. The result shows below. We

may also get more precise estimate by stratifying people into different subgroups based on their charitability.

```
[39] df.y1.mean() - df.y0.mean()

0.14884599999999998
```

```
ch_mean = df['charitability'].mean()
df['diff'] = df['charitability'] - ch_mean
r_cace = sfa.ols("y ~ coin_flip ",df.query("diff > 0")).fit(cov_type='HC1')
```

```
[41] print(r_cace.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.019
Model:                            OLS   Adj. R-squared:                  0.019
Method:                 Least Squares   F-statistic:                     9909.
Date:                Tue, 29 Sep 2020   Prob (F-statistic):               0.00
Time:                        00:20:55   Log-Likelihood:            -3.5780e+05
No. Observations:              500202   AIC:                         7.156e+05
Df Residuals:                  500200   BIC:                         7.156e+05
Df Model:                           1
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4121      0.001    418.936      0.000       0.410       0.414
coin_flip      0.1393      0.001     99.544      0.000       0.137       0.142
==============================================================================
Omnibus:                 1792695.102   Durbin-Watson:                   2.000
Prob(Omnibus):                 0.000   Jarque-Bera (JB):            76900.896
Skew:                          0.071   Prob(JB):                         0.00
Kurtosis:                      1.084   Cond. No.                         2.62
==============================================================================
```