Tarea3_Munoz

December 10, 2022

1 TAREA 3: Laboratorio de métodos aplicados avanzados

1.0.1 Autor: Sebastián Muñoz

1.0.2 Fecha: 28/11/22

2 Parte 1: Experimentos

1. A nivel de estudiante, el tratamiento que se sugiere es la constante motivación por parte del docente hacia los alumnos acerca de la asistencia a clases y los efectos positivos que conllevan en el rendimiento academico, además de consideraciones especiales hacia los alumnos que se encuentran al borde de reprobar pero que han asistido constantemente a clase; por lo que cada sesión de clase será registrada la asistencia de los alumnos, sin embargo, al grupo de control, solo se les registrará la asistencia y el docente hará su clase con normalidad, en ningún momento la importancia de la asistencia será tema de conversación por parte del profesor.

Para hacer efectivo el tratamiento se requiere de una aplicación movil linkeada con la cuenta UdeC del alumno, la cual le permite al alumno registrar su asistencia por cada sesión de clase mediante el escaneo de un codigo QR en momentos aleatorios durante la clase, el cual será impreso en una o dos hojas por el docente y para agilizar el procedimiento será distribuido por todo el salón para que las personas puedan escanearlo; por medidas preventivas este codigo será dinámico por cada clase para evitar actitudes deshonestas entre alumnos, al final de la clase las hojas serán devueltas al profesor.

2. Se escoge una asignatura con una cantidad par de estudiantes inscritos en ella, y que por lo menos tenga 80 estudiantes inscritos para poder dividir el curso en dos secciones con igual cantidad de estudiantes, considerando 40 alumnos por sección un número significativo para el experimento dadas las condiciones que se debe aplicar a los estudiantes de un curso en particular; la distribución de los alumnos a las respectivas secciones será de forma aleatoria en base a la lista de estudiantes que se hayan inscrito. Por lo tanto la sección 1 tendrá el rol de grupo de control y la sección 2 será el grupo de tratamiento.

3. Para estimar el efecto promedio se escogerá post test, al finalizar el semestre se les aplicará a todos los estudiantes un cuestionario acerca de lo importante que es la asistencia a clases en la educación superior en forma de escala likert(ejemplo: de acuerdo, en desacuerdo, etc). No se aplicará pre test debido a que se pretende reducir la probabilidad de que pueda condicionar el comportamiento de alguna manera de los alumnos o de darles algún indicio a los que pertenecen al grupo de control de que pertenecen a dicho grupo. No sería posible implementar Salomon 4-group primero por lo costoso que resulta ser y por que sería muy dificil generar 4 grupos equivalentes de alumnos que pertenecen a una clase en común sin que se den cuenta

del proposito del experimento.

4. Como el experimento ahora es a nivel de clase, se escogen cursos o asignaturas que se impartan en el mismo semestre y por distintas facultades a modo de reducir la probabilidad de exista contaminación entre grupos; la cantidad total de cursos a escoger debe ser par y de un minimo de cuatro para poder destinar de forma aleatoria la mitad como grupo de control y la otra mitad como grupo de tratamiento.

Ahora bien, el tratamiento consiste en que al grupo tratado se les evaluará la asistencia, correspondiendole una ponderación de 15% en la nota final del alumno, controlando para ello la asistencia por sesión de clase y además se les informará de la existencia de este item evaluativo durante la primera semana en la que empiezen las clases del curso, mientras que por otro lado a los grupos de control solo se les registra la asistencia pero sin que esta sea motivo de evaluación. El registro de asistencia será de la misma forma que para el experimento a nivel estudiante.

Para estimar el efecto promedio se escogerá pre-post test y el mismo instrumento que para el experimento en nivel estudiante, pero el motivo para incluir el pre test es para comparar las percepciones de la importancia de asistencia antes del experimento y posterior a este.

5. El programa se implementa de forma gradual por cada una de las facultades que conforman la universidad. El docente de una asignatura perteneciente a una facultad adherida al programa, tiene la elección de tomar o no el tratamiento que consiste en la misma metodologia del experimento a nivel cluster, o sea, si toma el tratamiento entonces su clase implementará la asistencia evaluada, de otro modo, la clase no evaluará la asistencia pero de todas maneras debe procurar registrar la asistencia de los alumnos.

3 Parte 2 estimación de efectos

[1]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
from scipy.stats import norm
```

3.0.1 Pregunta 1

[2]:
```
# experiment parameters
np.random.seed(33) #seed
nsize = 4000 #2 periodos*50 alumnos*40 grupos

#error normal
Xc=norm.rvs(size=(1, nsize))
Xc = Xc.transpose()
Xc = pd.DataFrame(Xc, columns=['X'])

#periodos de tiempo y asignación del tratamiento
```

```
Xc['p'] = 1
Xc.loc[0:1999,'p'] = 0

T = np.array([0] * 1000 + [1] * 1000)
np.random.shuffle(T)
Xc.loc[0:1999,'T'] = T

#Se les hace corresponder el grupo (control o tratamiento) correcto a cada una␣
 ↪de las personas en el segundo periodo (p=1)
for i in range(0,2000):
    Xc.loc[i+2000,'T'] = Xc.loc[i,'T']

#Realizando los 40 grupos con 50 personas cada uno, tanto para el primero como␣
 ↪el segundo periodo
Xc['cl']=1
i = 50
j= 99
for k in range(1,40,1):
    if i + 2000 == 2000:
        Xc.loc[i+2000:j+2000,'cl'] = 1
    else:
        Xc.loc[i:j,'cl']=k+1
        Xc.loc[i+2000:j+2000,'cl']=k+1
    i+=50
    j+=50

Xc.describe()
```

[2]:

	X	p	T	cl
count	4000.000000	4000.000000	4000.000000	4000.00000
mean	0.010042	0.500000	0.500000	20.50000
std	0.999169	0.500063	0.500063	11.54484
min	-3.716564	0.000000	0.000000	1.00000
25%	-0.679875	0.000000	0.000000	10.75000
50%	0.033261	0.500000	0.500000	20.50000
75%	0.692908	1.000000	1.000000	30.25000
max	3.736254	1.000000	1.000000	40.00000

3.0.2 Pregunta 6

[3]:
```
#NIVEL CLASE
#Se asigna a cada grupo la condición de si es tratamiento o es control de forma␣
 ↪equitativa y aleatoria
np.random.seed(33)
T = np.array([0] * 20 + [1] * 20)
np.random.shuffle(T)
```

```
Xc['Tcl']=1
for i in range(len(Xc)):
    Xc.loc[i,'Tcl'] = T[(Xc.loc[i,'cl'])-1]


#pre test
alpha = 0.881  #Con este alfa se logra la proporcion de 0.8 de asistencia en␣
 ↪los dos grupos
beta = 0.000

#outcome asistencia
Xc['ycl'] = alpha + beta*Xc['Tcl'] + Xc['X']

Xc['yBincl']= 1
for i in range (4000):
    Xc.loc[i,'yBincl'] = norm.cdf(Xc.loc[i,'ycl'])
    if Xc.loc[i,'yBincl'] >= 0.5:
        Xc.loc[i,'yBincl'] = 1
    else:
        Xc.loc[i,'yBincl'] = 0

print('Proporción de alumnos pertenecientes a los grupos de control que␣
 ↪asistieron en el primer periodo:',Xc[(Xc['p']==0) &␣
 ↪(Xc['Tcl']==0)]['yBincl'].mean())
print('Proporción de alumnos pertenecientes a los grupos de tratamiento que␣
 ↪asistieron en el primer periodo:',Xc[(Xc['p']==0) &␣
 ↪(Xc['Tcl']==1)]['yBincl'].mean())
Xc[['cl','Tcl','yBincl']].describe()
```

```
Proporción de alumnos pertenecientes a los grupos de control que asistieron en
el primer periodo: 0.8
Proporción de alumnos pertenecientes a los grupos de tratamiento que asistieron
en el primer periodo: 0.821
```

[3]:

	cl	Tcl	yBincl
count	4000.00000	4000.000000	4000.000000
mean	20.50000	0.500000	0.809000
std	11.54484	0.500063	0.393138
min	1.00000	0.000000	0.000000
25%	10.75000	0.000000	1.000000
50%	20.50000	0.500000	1.000000
75%	30.25000	1.000000	1.000000
max	40.00000	1.000000	1.000000

3.0.3 Pregunta 7

[4]:
```
#NIVEL ESTUDIANTE
#pre test
alpha = 0.856 #Con este alfa se logra la proporcion de 0.8 de asistencia en los␣
 ↪dos grupos
beta = 0.000

#outcome asistencia
Xc['y'] = alpha + beta*Xc['T'] + Xc['X']

Xc['yBin']= 1
for i in range (4000):
    Xc.loc[i,'yBin'] = norm.cdf(Xc.loc[i,'y'])
    if Xc.loc[i,'yBin'] >= 0.5:
        Xc.loc[i,'yBin'] = 1
    else:
        Xc.loc[i,'yBin'] = 0

print('Proporción de alumnos del grupo de control que asistieron en el primer␣
 ↪periodo:',Xc[(Xc['p']==0) & (Xc['T']==0)]['yBin'].mean())
print('Proporción de alumnos del grupo de tratamiento que asistieron en el␣
 ↪primer periodo:',Xc[(Xc['p']==0) & (Xc['T']==1)]['yBin'].mean())
Xc[['cl','T','yBin']].describe()
```

```
Proporción de alumnos del grupo de control que asistieron en el primer periodo:
0.8
Proporción de alumnos del grupo de tratamiento que asistieron en el primer
periodo: 0.809
```

[4]:

	cl	T	yBin
count	4000.00000	4000.000000	4000.000000
mean	20.50000	0.500000	0.801000
std	11.54484	0.500063	0.399298
min	1.00000	0.000000	0.000000
25%	10.75000	0.000000	1.000000
50%	20.50000	0.500000	1.000000
75%	30.25000	1.000000	1.000000
max	40.00000	1.000000	1.000000

3.0.4 Pregunta 8

[5]:
```
#post test del grupo de control
alpha = 0.856
beta = 0.000
delta = -0.184
#outcome variable
```

```
Xc['y'] = alpha + beta*Xc['T'] + delta*Xc['p'] + Xc['X']

Xc['yBin']= 1
for i in range (4000):
    Xc.loc[i,'yBin'] = norm.cdf(Xc.loc[i,'y'])
    if Xc.loc[i,'yBin'] >= 0.5:
        Xc.loc[i,'yBin'] = 1
    else:
        Xc.loc[i,'yBin'] = 0

print('Proporción de alumnos del grupo de control que asistieron en el segundo␣
 ↪periodo:',Xc[(Xc['p']==1) & (Xc['T']==0)]['yBin'].mean())
Xc[['cl','T','yBin']].describe()
```

```
Proporción de alumnos del grupo de control que asistieron en el segundo periodo:
0.751
```

[5]:

	cl	T	yBin
count	4000.00000	4000.000000	4000.000000
mean	20.50000	0.500000	0.774750
std	11.54484	0.500063	0.417799
min	1.00000	0.000000	0.000000
25%	10.75000	0.000000	1.000000
50%	20.50000	0.500000	1.000000
75%	30.25000	1.000000	1.000000
max	40.00000	1.000000	1.000000

[6]:
```
#post test del grupo de tratamiento
alpha = 0.856
beta = 0.000
delta = -0.184
gamma = 0.565
#outcome variable
Xc['y'] = alpha + beta*Xc['T'] + delta*Xc['p'] + gamma*Xc['p']*Xc['T'] + Xc['X']

Xc['yBin']= 1
for i in range (4000):
    Xc.loc[i,'yBin'] = norm.cdf(Xc.loc[i,'y'])
    if Xc.loc[i,'yBin'] >= 0.5:
        Xc.loc[i,'yBin'] = 1
    else:
        Xc.loc[i,'yBin'] = 0

print('Proporción de alumnos del grupo de tratamiento que asistieron en el␣
 ↪segundo periodo:',Xc[(Xc['p']==1) & (Xc['T']==1)]['yBin'].mean())
Xc[['cl','T','yBin']].describe()
```

Proporción de alumnos del grupo de tratamiento que asistieron en el segundo periodo: 0.9

[6]:

	cl	T	yBin
count	4000.00000	4000.000000	4000.000000
mean	20.50000	0.500000	0.815000
std	11.54484	0.500063	0.388346
min	1.00000	0.000000	0.000000
25%	10.75000	0.000000	1.000000
50%	20.50000	0.500000	1.000000
75%	30.25000	1.000000	1.000000
max	40.00000	1.000000	1.000000

[7]:
```
#post-test

y = Xc.loc[2000:3999,'yBin']
X = Xc.loc[2000:3999,'T']
X = sm.add_constant(X)
model = sm.Logit(y, X)
results = model.fit()
print(results.summary())
mfx = results.get_margeff()
print(mfx.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.443158
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                   yBin   No. Observations:                 2000
Model:                          Logit   Df Residuals:                     1998
Method:                           MLE   Df Model:                            1
Date:                Mon, 28 Nov 2022   Pseudo R-squ.:                 0.04275
Time:                        20:56:22   Log-Likelihood:                -886.32
converged:                       True   LL-Null:                       -925.90
Covariance Type:            nonrobust   LLR p-value:                 5.706e-19
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.1040      0.073     15.096      0.000       0.961       1.247
T              1.0933      0.128      8.522      0.000       0.842       1.345
==============================================================================
        Logit Marginal Effects
=====================================
Dep. Variable:                   yBin
Method:                          dydx
At:                           overall
==============================================================================
```

```
                dy/dx    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
T              0.1514      0.017      8.715      0.000       0.117       0.185
==============================================================================
```

El efecto del tratamiento usando solo post test es de 0.1514, es decir, pasar de ser control a tratamiento la asistencia promedio aumenta 15.14 puntos porcentuales

3.0.5 Pregunta 9

[8]:
```
#pre-post test
y=Xc['yBin']
Xc['dd']= Xc['p']*Xc['T']
X=Xc[['p','T','dd',]]
X = sm.add_constant(X)
model = sm.Logit(y, X)
results2 = model.fit()
print(results2.summary())
mfx2 = results2.get_margeff()
print(mfx2.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.468597
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                   yBin   No. Observations:                 4000
Model:                          Logit   Df Residuals:                     3996
Method:                           MLE   Df Model:                            3
Date:                Mon, 28 Nov 2022   Pseudo R-squ.:                 0.02150
Time:                        20:56:22   Log-Likelihood:                -1874.4
converged:                       True   LL-Null:                       -1915.6
Covariance Type:            nonrobust   LLR p-value:                 9.608e-18
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.3863      0.079     17.535      0.000       1.231       1.541
p             -0.2823      0.108     -2.622      0.009      -0.493      -0.071
T              0.0572      0.113      0.507      0.612      -0.164       0.278
dd             1.0360      0.171      6.065      0.000       0.701       1.371
==============================================================================
        Logit Marginal Effects
=====================================
Dep. Variable:                   yBin
Method:                          dydx
At:                           overall
==============================================================================
                dy/dx    std err          z      P>|z|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
p             -0.0418      0.016     -2.627      0.009      -0.073      -0.011
T              0.0085      0.017      0.507      0.612      -0.024       0.041
dd             0.1532      0.025      6.097      0.000       0.104       0.202
==============================================================================
```

El efecto del tratamiento utilizando pre-post test es de 0.1532, es decir, 15.32 puntos porcentuales, muy similar al valor estimado usando solo post test

3.0.6 Pregunta 10

[9]:
```
#clustered standard errors
results3 = model.fit(cov_type="cluster", cov_kwds={'groups': Xc['cl']})
print(results3.summary())
mfx3 = results3.get_margeff()
print(mfx3.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.468597
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                    yBin   No. Observations:                 4000
Model:                          Logit   Df Residuals:                     3996
Method:                           MLE   Df Model:                            3
Date:                Mon, 28 Nov 2022   Pseudo R-squ.:                 0.02150
Time:                        20:56:22   Log-Likelihood:                -1874.4
converged:                       True   LL-Null:                       -1915.6
Covariance Type:              cluster   LLR p-value:                 9.608e-18
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.3863      0.103     13.483      0.000       1.185       1.588
p             -0.2823      0.124     -2.277      0.023      -0.525      -0.039
T              0.0572      0.133      0.431      0.666      -0.203       0.317
dd             1.0360      0.168      6.153      0.000       0.706       1.366
==============================================================================
        Logit Marginal Effects
=====================================
Dep. Variable:                    yBin
Method:                           dydx
At:                           overall
==============================================================================
                dy/dx    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
p             -0.0418      0.018     -2.292      0.022      -0.077      -0.006
T              0.0085      0.020      0.431      0.666      -0.030       0.047
dd             0.1532      0.025      6.105      0.000       0.104       0.202
```

==========

El efecto del tratamiento ajustando los errores estandar por cluster resultó ser de 0.1532, la diferencia entre ambas estimaciones es cero, debido a que como se diseñó la data, no existe correlacion intra grupos.

4 Parte 3

4.0.1 Pregunta 11

[10]:
```
charls = pd.read_csv('../data/charls.csv')
np.random.seed(33)
charls['drinkly']=charls['drinkly'].replace('.m',np.nan)
charls['drinkly']=charls['drinkly'].replace('.d',np.nan)
charls.dropna(inplace=True)
charls.reset_index(drop=True, inplace=True)
charls['drinkly'] = charls['drinkly'].astype(int)
charls['sdrinkly']=charls['drinkly']

print('Proporción original de las personas con 3 hijos o más que han bebido␣
 ↪alcohol en el tercer periodo: ',charls[(charls['child']>=3) &␣
 ↪(charls['wave']==3)]['drinkly'].mean())

total = charls.loc[(charls['child']>=3) & (charls['wave']==3)]['drinkly'].
 ↪count() #3236 (total de personas con 3 hijos o más en el periodo 3)
weight= charls.loc[(charls['child']>=3) & (charls['drinkly']==1) &␣
 ↪(charls['wave']==3)]['drinkly'].count() #1062 (total de personas con 3 hijos␣
 ↪o más que han bebido alcohol en el periodo 3)
weight1 = int(weight/2) #531 (total de personas con 3 hijos o más que han␣
 ↪bebido alcohol en el periodo 3 necesarias para reducir a la mitad la␣
 ↪proporción)
drink = np.array([0] * (total-weight1) + [1] * weight1)
np.random.shuffle(drink)


subset=charls.loc[(charls['child']>=3) & (charls['wave']==3)]
c=0
for i in subset.index:
    charls.loc[i,'sdrinkly']=drink[c]
    c+=1

print('Proporción de las mismas personas reducida a la mitad en el tercer␣
 ↪periodo: ',charls[(charls['child']>=3) & (charls['wave']==3)]['sdrinkly'].
 ↪mean())
charls.loc[(charls['child']>=3) &␣
 ↪(charls['wave']==3)][['wave','drinkly','sdrinkly']].describe() #describe de␣
 ↪la data de personas con 3 hijos o mas en el tercer periodo
```

```
Proporción original de las personas con 3 hijos o más que han bebido alcohol en
el tercer periodo:  0.32818294190358466
Proporción de las mismas personas reducida a la mitad en el tercer periodo:
0.16409147095179233
```

[10]:

	wave	drinkly	sdrinkly
count	3236.0	3236.000000	3236.000000
mean	3.0	0.328183	0.164091
std	0.0	0.469624	0.370416
min	3.0	0.000000	0.000000
25%	3.0	0.000000	0.000000
50%	3.0	0.000000	0.000000
75%	3.0	1.000000	0.000000
max	3.0	1.000000	1.000000

[11]:
```
charls.loc[(charls['child']>=3) &␣
 ↪(charls['wave']==3)][['wave','drinkly','sdrinkly']].head() #data de personas␣
 ↪con 3 hijos o mas en el tercer periodo
```

[11]:

	wave	drinkly	sdrinkly
20	3	0	0
23	3	1	0
74	3	1	1
77	3	0	0
87	3	1	0

4.0.2 Pregunta 12

[12]:
```
np.random.seed(33)

charls['tratamiento']=0
for i in range(len(charls)):
    if charls.loc[i,'child'] < 2:   #Las personas con menos de 2 hijos no son␣
 ↪parte de la intervención y por lo tanto no tienen tratamiento
        charls.loc[i,'tratamiento'] == 0

    charls.loc[i,'tratamiento']=np.random.binomial(1,0.5)   #asignacion␣
 ↪aleatoria del tratamiento
    try:
        if (charls.loc[i,'inid']==charls.loc[i+1,'inid']) and charls.
 ↪loc[i,'tratamiento']==1: #Si la persona ya ha tomado el tratamiento en el␣
 ↪periodo 2 entonces
            charls.loc[i+1,'tratamiento']=1                                  ␣
 ↪              #en el periodo 3 tambien debe tener registrado que lo tomó
    except KeyError:
        pass
```

```
charls2=charls.loc[(charls['wave']>=2) &␣
 ↪(charls['child']>=2)][['wave','drinkly','tratamiento','sdrinkly','married']]

#pre post test (diferencia en diferencias)

y=charls2['sdrinkly']
charls2['dd']= charls2['wave']*charls2['tratamiento']
X=charls2[['wave','tratamiento','dd']]
X = sm.add_constant(X)

model = sm.Logit(y, X)
results = model.fit(cov_type="HC1")
print(results.summary())
mfx = results.get_margeff()
print(mfx.summary())
charls2.head()
```

```
Optimization terminated successfully.
         Current function value: 0.600927
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                sdrinkly   No. Observations:                11576
Model:                           Logit   Df Residuals:                    11572
Method:                            MLE   Df Model:                            3
Date:                 Mon, 28 Nov 2022   Pseudo R-squ.:                0.006834
Time:                         20:56:25   Log-Likelihood:                -6956.3
converged:                       True   LL-Null:                       -7004.2
Covariance Type:                  HC1   LLR p-value:                 1.286e-20
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1556      0.146      1.068      0.285      -0.130       0.441
wave          -0.4261      0.059     -7.271      0.000      -0.541      -0.311
tratamiento   -0.0970      0.206     -0.471      0.637      -0.500       0.306
dd             0.0510      0.083      0.618      0.537      -0.111       0.213
==============================================================================
        Logit Marginal Effects
=====================================
Dep. Variable:                sdrinkly
Method:                           dydx
At:                           overall
==============================================================================
                 dy/dx    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
wave          -0.0876      0.012     -7.323      0.000      -0.111      -0.064
tratamiento   -0.0199      0.042     -0.471      0.637      -0.103       0.063
```

```
dd              0.0105       0.017       0.618       0.537       -0.023        0.044
==============================================================================
```

[12]:

	wave	drinkly	tratamiento	sdrinkly	married	dd
1	2	0	0	0	1	0
2	3	0	0	0	1	0
4	2	1	1	1	1	2
5	3	1	0	1	1	0
7	3	0	1	0	1	3

El efecto del tratamiento comparando los periodos 2 y 3 fue de 0.0105, es decir, 1.05 puntos porcentuales; sin embargo este efecto no es estadisticamente significativo con p > 0.1

4.0.3 Pregunta 13

[13]:
```
charls3=charls.loc[(charls['child']>=2) & (charls['child']<=3) &␣
 ↪(charls['wave']>=2)][['age','female','cesd','wave','child','drinkly','married','tratamiento
charls3.reset_index(drop=True, inplace=True)

charls3['cl']=1     #Se asignan los tratamientos como clusters, donde las␣
 ↪personas con 2 hijos son control cl=0 y los que tienen 3 hijos son␣
 ↪tratamiento cl=1
for i in range(len(charls3)):
    if charls3.loc[i,'child'] == 3:
        charls3.loc[i,'cl'] = 1
    else:
        charls3.loc[i,'cl'] = 0

charls3['dd']= charls3['wave']*charls3['cl']
Xa=charls3[['wave','cl']]
ya=charls3['sdrinkly']
Xa = sm.add_constant(Xa)
model = sm.Logit(ya, Xa)
results2 = model.fit(cov_type="HC1")
print(results2.summary())
mfx2 = results2.get_margeff()
print(mfx.summary())
charls3.head()
```

```
Optimization terminated successfully.
         Current function value: 0.614553
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:                sdrinkly   No. Observations:                 8224
Model:                           Logit   Df Residuals:                     8221
Method:                            MLE   Df Model:                            2
Date:                Mon, 28 Nov 2022   Pseudo R-squ.:                 0.01636
```

```
Time:                        20:56:25   Log-Likelihood:                -5054.1
converged:                       True   LL-Null:                       -5138.1
Covariance Type:                  HC1   LLR p-value:                 3.151e-37
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2286      0.121      1.883      0.060      -0.009       0.467
wave          -0.3113      0.048     -6.511      0.000      -0.405      -0.218
cl            -0.5542      0.049    -11.277      0.000      -0.651      -0.458
==============================================================================
        Logit Marginal Effects
=====================================
Dep. Variable:                sdrinkly
Method:                           dydx
At:                           overall
==============================================================================
                 dy/dx    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
wave          -0.0876      0.012     -7.323      0.000      -0.111      -0.064
tratamiento   -0.0199      0.042     -0.471      0.637      -0.103       0.063
dd             0.0105      0.017      0.618      0.537      -0.023       0.044
==============================================================================
```

[13]:

	age	female	cesd	wave	child	drinkly	married	tratamiento	sdrinkly	\
0	48	1	7.0	2	2	0	1	0	0	
1	50	1	5.0	3	2	0	1	0	0	
2	50	0	5.0	2	2	1	1	1	1	
3	52	0	6.0	3	2	1	1	0	1	
4	60	1	6.0	3	2	0	1	1	0	

	cl	dd
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0

El efecto del tratamiento comparando los periodos 2 y 3 fue de 0.0105

4.0.4 Pregunta 14

[14]:
```
Xf=charls3[['wave','married']] #married es el instrumento
yf=charls3['cl']
Xf = sm.add_constant(Xf)
model = sm.OLS(yf, Xf)
first = model.fit(cov_type="HC1")
charls3['ptratamiento']=first.predict(Xf)
```

```
print(first.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                     cl   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     7.461
Date:                Mon, 28 Nov 2022   Prob (F-statistic):           0.000579
Time:                        20:56:25   Log-Likelihood:                -5872.2
No. Observations:                8224   AIC:                         1.175e+04
Df Residuals:                    8221   BIC:                         1.177e+04
Df Model:                           2
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.5374      0.034     15.753      0.000       0.471       0.604
wave          -0.0074      0.011     -0.660      0.509      -0.029       0.015
married       -0.0966      0.027     -3.576      0.000      -0.150      -0.044
==============================================================================
Omnibus:                    30169.783   Durbin-Watson:                   0.817
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1363.377
Skew:                           0.296   Prob(JB):                     8.84e-297
Kurtosis:                       1.095   Cond. No.                         20.7
==============================================================================

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

[15]:
```
Xa=charls3[['wave','ptratamiento']]
ya=charls3['sdrinkly']
Xa = sm.add_constant(Xa)
model = sm.OLS(ya, Xa)
second = model.fit(cov_type="HC1")

print(second.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                sdrinkly   R-squared:                       0.006
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     24.80
Date:                Mon, 28 Nov 2022   Prob (F-statistic):           1.82e-11
Time:                        20:56:25   Log-Likelihood:                -5354.2
No. Observations:                8224   AIC:                         1.071e+04
Df Residuals:                    8221   BIC:                         1.074e+04
Df Model:                           2
Covariance Type:                  HC1
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8728      0.121      7.191      0.000       0.635       1.111
wave          -0.0779      0.011     -7.011      0.000      -0.100      -0.056
ptratamiento  -0.8463      0.252     -3.360      0.001      -1.340      -0.353
==============================================================================
Omnibus:                   117628.520   Durbin-Watson:                   1.773
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1469.784
Skew:                           0.778   Prob(JB):                         0.00
Kurtosis:                       1.634   Cond. No.                         155.
==============================================================================

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

4.0.5 Pregunta 15

[16]:
```
second.params['ptratamiento']/first.params['married'] #Efecto del estimador = Efecto de la prediccion del tratamiento / efecto del instrumento
```

[16]:
```
8.758385128943804
```

[17]:
```
np.random.seed(33)
print('Proporción original de las personas en el tercer periodo:',charls[charls['wave']==3]['drinkly'].mean())


total = charls.loc[charls['wave']==3]['drinkly'].count() #
weight= charls.loc[(charls['drinkly']==1) & (charls['wave']==3)]['drinkly'].count() #
weight1 = int(weight/2) #531
drink = np.array([0] * (total-weight1) + [1] * weight1)
np.random.shuffle(drink)
subset=charls.loc[(charls['wave']==3)]
charls['tdrinkly']=charls['drinkly']
c=0
charls['T']=0
for i in subset.index:
    charls.loc[i,'tdrinkly']=drink[c]
    charls.loc[i,'T']=1
    c+=1

print('Proporción de las personas reducida a la mitad en el tercer periodo:',charls[charls['wave']==3]['tdrinkly'].mean())

charls['pdrinkly']=0
```

```
charls['cdrinkly']=charls['drinkly']
charls2=charls.
 ↪loc[charls['wave']<=2][['married','female','age','drinkly','child','pdrinkly']]

Xf=charls2[['married','female','age','child']]
yf=charls2['drinkly']
model = sm.OLS(yf, Xf)
first = model.fit(cov_type="HC1")
print(first.summary())
charls['pdrinkly']=first.predict(charls[['married','female','age','child']])

for i in range(len(charls)):
    if charls.loc[i,'pdrinkly'] < 0.5:
        charls.loc[i,'pdrinkly'] = 0
    else:
        charls.loc[i,'pdrinkly'] = 1


charls['Tc']= charls['wave']*charls['T']
```

```
Proporción original de las personas en el tercer periodo:  0.34807972775887214
Proporción de las personas reducida a la mitad en el tercer periodo:
0.17403986387943607
                                OLS Regression Results
==============================================================================
=======
Dep. Variable:                drinkly   R-squared (uncentered):
0.448
Model:                           OLS   Adj. R-squared (uncentered):
0.447
Method:                Least Squares   F-statistic:
2331.
Date:               Mon, 28 Nov 2022   Prob (F-statistic):
0.00
Time:                       20:56:26   Log-Likelihood:
-8473.5
No. Observations:               14867   AIC:
1.696e+04
Df Residuals:                   14863   BIC:
1.699e+04
Df Model:                           4
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
married        0.1710      0.009     19.744      0.000       0.154       0.188
female        -0.3837      0.007    -55.110      0.000      -0.397      -0.370
```

```
age             0.0070      0.000     33.215      0.000       0.007       0.007
child          -0.0138      0.003     -4.794      0.000      -0.019      -0.008
==============================================================================
Omnibus:                     1715.990   Durbin-Watson:                   1.488
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              851.913
Skew:                           0.425   Prob(JB):                   1.02e-185
Kurtosis:                       2.193   Cond. No.                         151.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors are heteroscedasticity robust (HC1)
```

[18]:
```
ye = charls['tdrinkly']
Xe = charls[['Tc','wave']]
Xe = sm.add_constant(Xe)
model = sm.OLS(ye, Xe)
results = model.fit(cov_type="HC1")
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                tdrinkly   R-squared:                       0.025
Model:                            OLS   Adj. R-squared:                  0.025
Method:                 Least Squares   F-statistic:                     325.1
Date:                Mon, 28 Nov 2022   Prob (F-statistic):          9.04e-140
Time:                        20:56:27   Log-Likelihood:                -12855.
No. Observations:               21038   AIC:                         2.572e+04
Df Residuals:                   21035   BIC:                         2.574e+04
Df Model:                           2
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3141      0.012     26.384      0.000       0.291       0.337
Tc            -0.0586      0.005    -13.009      0.000      -0.067      -0.050
wave           0.0119      0.008      1.529      0.126      -0.003       0.027
==============================================================================
Omnibus:                     9040.090   Durbin-Watson:                   1.594
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3869.422
Skew:                           0.906   Prob(JB):                         0.00
Kurtosis:                       1.937   Cond. No.                         11.8
==============================================================================

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

[19]:
```
ye = charls['cdrinkly']
Xe = charls[['Tc','wave']]
Xe = sm.add_constant(Xe)
model = sm.OLS(ye, Xe)
results = model.fit(cov_type="HC1")
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                cdrinkly   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     3.876
Date:                Mon, 28 Nov 2022   Prob (F-statistic):             0.0207
Time:                        20:56:27   Log-Likelihood:                -14073.
No. Observations:               21038   AIC:                         2.815e+04
Df Residuals:                   21035   BIC:                         2.818e+04
Df Model:                           2
Covariance Type:                  HC1
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3141      0.012     26.384      0.000       0.291       0.337
Tc            -0.0005      0.005     -0.117      0.907      -0.010       0.009
wave           0.0119      0.008      1.529      0.126      -0.003       0.027
==============================================================================
Omnibus:                   122790.887   Durbin-Watson:                   1.314
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3703.672
Skew:                           0.693   Prob(JB):                         0.00
Kurtosis:                       1.481   Cond. No.                         11.8
==============================================================================

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