Tarea2_Jara_Retamal

October 14, 2022

1 Tarea 2

1.1 Laboratorio de métodos aplicados avanzados

1.2 Integrantes:

-

1.2.1 Francisca Carolina Jara Yévenes

-

1.2.2 Luis Fernando Retamal Fuentes

1.3 Fecha: 5 de octubre de 2022

2 Bibliotecas requeridas

[150]:
```
%pip install linearmodels
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
import linearmodels.panel as lmp
from numpy import empty
%matplotlib inline
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: linearmodels in /usr/local/lib/python3.7/dist-
packages (4.25)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (1.3.5)
Requirement already satisfied: patsy in /usr/local/lib/python3.7/dist-packages
(from linearmodels) (0.5.2)
Requirement already satisfied: scipy>=1.2 in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (1.7.3)
```

```
Requirement already satisfied: property-cached>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from linearmodels) (1.6.4)
Requirement already satisfied: mypy-extensions>=0.4 in
/usr/local/lib/python3.7/dist-packages (from linearmodels) (0.4.3)
Requirement already satisfied: formulaic in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (0.5.2)
Requirement already satisfied: statsmodels>=0.11 in
/usr/local/lib/python3.7/dist-packages (from linearmodels) (0.12.2)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (1.21.6)
Requirement already satisfied: Cython>=0.29.21 in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (0.29.32)
Requirement already satisfied: pyhdfe>=0.1 in /usr/local/lib/python3.7/dist-
packages (from linearmodels) (0.1.0)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.24->linearmodels) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
packages (from pandas>=0.24->linearmodels) (2022.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.7.3->pandas>=0.24->linearmodels) (1.15.0)
Requirement already satisfied: astor>=0.8 in /usr/local/lib/python3.7/dist-
packages (from formulaic->linearmodels) (0.8.1)
Requirement already satisfied: cached-property>=1.3.0 in
/usr/local/lib/python3.7/dist-packages (from formulaic->linearmodels) (1.5.2)
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.7/dist-
packages (from formulaic->linearmodels) (1.14.1)
Requirement already satisfied: graphlib-backport>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from formulaic->linearmodels) (1.0.3)
Requirement already satisfied: interface-meta>=1.2.0 in
/usr/local/lib/python3.7/dist-packages (from formulaic->linearmodels) (1.3.0)
Requirement already satisfied: typing-extensions>=4.2.0 in
/usr/local/lib/python3.7/dist-packages (from formulaic->linearmodels) (4.3.0)
```

3 1. Lectura de archivo y eliminación de datos faltantes

[151]:
```
charls = pd.read_csv('/charls.csv')
charls.dropna(inplace=True)
charls.reset_index(drop=True, inplace=True)
list=[]
for j in range(len(charls)):
  list.append(0)
charls.head(5)
```

[151]:

	cesd	child	drinkly	female	hrsusu	hsize	inid	intmonth	\
0	6	2	0.None	1	0.0	4	1.010410e+10	7	
1	7	2	0.None	1	49.0	4	1.010410e+10	7	
2	5	2	0.None	1	56.0	7	1.010410e+10	8	

```
3     0      2   1.Yes       0    63.0      4  1.010410e+10         7
4     5      2   1.Yes       0    49.0      4  1.010410e+10         7

   married  retired  schadj  urban  wave   wealth  age
0        1        0       0      0     1  -5800.0   46
1        1        0       0      0     2    100.0   46
2        1        0       0      0     3 -59970.0   46
3        1        0       4      0     1  -5800.0   48
4        1        0       4      0     2    100.0   48
```

[152]:
```
charls.describe()
```

[152]:
```
                cesd         child        female        hrsusu         hsize  \
count  34371.000000  34371.000000  34371.000000  34371.000000  34371.000000
mean       8.173606      2.708184      0.537837     27.966476      3.441011
std        6.183766      1.399999      0.498574     28.270626      1.702308
min        0.000000      0.000000      0.000000      0.000000      1.000000
25%        3.000000      2.000000      0.000000      0.000000      2.000000
50%        7.000000      2.000000      1.000000     21.000000      3.000000
75%       12.000000      3.000000      1.000000     54.000000      4.000000
max       30.000000     11.000000      1.000000    168.000000     16.000000

                inid      intmonth       married       retired       schadj  \
count  3.437100e+04  34371.000000  34371.000000  34371.000000  34371.00000
mean   1.768873e+11      7.524745      0.874545      0.283960      4.67749
std    1.063153e+11      0.916180      0.331238      0.450924      3.85762
min    1.010410e+10      1.000000      0.000000      0.000000      0.00000
25%    7.400430e+10      7.000000      1.000000      0.000000      0.00000
50%    1.624310e+11      7.000000      1.000000      0.000000      4.00000
75%    2.811160e+11      8.000000      1.000000      1.000000      8.00000
max    3.477630e+11     12.000000      1.000000      1.000000     18.00000

              urban          wave        wealth           age
count  34371.000000  34371.000000  3.437100e+04  34371.000000
mean       0.356289      2.000000  9.892309e+03     58.223531
std        0.478909      0.816508  9.738829e+04      9.154338
min        0.000000      1.000000 -2.110050e+06     16.000000
25%        0.000000      1.000000  0.000000e+00     51.000000
50%        0.000000      2.000000  4.000000e+02     58.000000
75%        1.000000      3.000000  5.000000e+03     64.000000
max        1.000000      3.000000  9.902500e+06     97.000000
```

3.1 Variables

- **INID**: identificador único
- **wave**: periodo de la encuesta (1-3)
- **cesd**: puntaje en la escala de salud mental (0-30)

- **child**: numero de hijos
- **drinkly**: bebió alcohol en el último mes (binario)
- **hrsusu**: horas promedio trabajo semanal
- **hsize**: tamaño del hogar
- **intmonth**: mes en que fue encuestado/a (1-12)
- **married**: si está casado/a (binario)
- **retired**: si está pensionado/a (binario)
- **schadj**: años de escolaridad
- **urban**: zona urbana (binario)
- **wealth**: riqueza neta (miles RMB)
- **age**: edad al entrar a la encuesta (no varía entre periodos)

3.2 Datatype

Para detectar el tipo de dato presente en el archivo se utiliza un atributo de pandas.

[153]:
```
charls.dtypes
```

[153]:
```
cesd          int64
child         int64
drinkly      object
female        int64
hrsusu      float64
hsize         int64
inid        float64
intmonth      int64
married       int64
retired       int64
schadj        int64
urban         int64
wave          int64
wealth      float64
age           int64
dtype: object
```

[154]:
```
charls['inid'] = charls['inid'].astype(np.int64) #Para que no se muestre en␣
 ↪notación científica el identiificador único
```

Dado que drinkly es una variable binaria para un mejor análisis se procede a convertir sus datos a datos enteros utilizando una lista de python.

[155]:
```
for i in range(0,len(charls)):
  if (charls.iloc[i,2]=='0.None'):
    list[i]=int(0)
  else: list[i]=int(1)

charls=charls.drop(['drinkly'], axis=1)
charls.insert(2, "drinkly", list, True)
```

```
charls.head(5)
```

[155]:

	cesd	child	drinkly	female	hrsusu	hsize	inid	intmonth	\
0	6	2	0	1	0.0	4	10104101001	7	
1	7	2	0	1	49.0	4	10104101001	7	
2	5	2	0	1	56.0	7	10104101001	8	
3	0	2	1	0	63.0	4	10104101002	7	
4	5	2	1	0	49.0	4	10104101002	7	

	married	retired	schadj	urban	wave	wealth	age
0	1	0	0	0	1	-5800.0	46
1	1	0	0	0	2	100.0	46
2	1	0	0	0	3	-59970.0	46
3	1	0	4	0	1	-5800.0	48
4	1	0	4	0	2	100.0	48

Se analiza el tipo de dato presente en el dataframe

[156]:
```
charls.dtypes #muestra el tipo de dato
```

[156]:
```
cesd          int64
child         int64
drinkly       int64
female        int64
hrsusu      float64
hsize         int64
inid          int64
intmonth      int64
married       int64
retired       int64
schadj        int64
urban         int64
wave          int64
wealth      float64
age           int64
dtype: object
```

[157]:
```
ch= charls ##copia del dataframe
```

En el **anexo 1** se muestran los histogramas de las diversas variables incluidas en el dataframe, para observar las distribuciones de cada una de ellas.

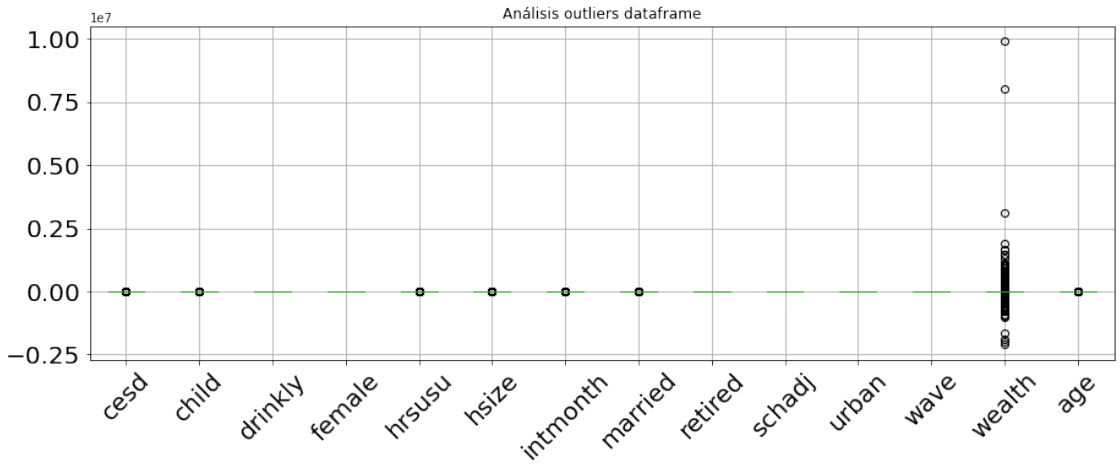
3.3 Outliers

Para analizar los outliers se procede a mostrar un gráfico de cajas con las variables del Dataframe.

[158]:
```

```

```
c=charls.drop(['inid'], axis=1) #se omite la variable inid en la representación
 ↪gráfica
boxplot = c.boxplot(grid=True,rot=45, fontsize=20,figsize =(15, 5))
plt.title('Análisis outliers dataframe')
```

[158]:
```
Text(0.5, 1.0, 'Análisis outliers dataframe')
```



Se analizan las variables con más puntos atípicos. En este caso, evidentemente, la variable **wealth**. En el **anexo 2** se presentan gráficos de caja para cada una de las variables. Posteriormente se eliminan aquellos puntos atípicos para producir un mejor modelo.

[159]:
```
charls = charls.drop(charls[charls['wealth']>1.112e+06].index)
charls = charls.drop(charls[charls['wealth']<-1e+06].index)
```

[160]:
```
charls = charls.drop(charls[charls['hrsusu']>133].index)
```

[161]:
```
charls = charls.drop(charls[charls['age']>89].index)
charls = charls.drop(charls[charls['age']<29].index)

charls.describe()
```

[161]:

	cesd	child	drinkly	female	hrsusu
count	34298.000000	34298.000000	34298.000000	34298.000000	34298.000000
mean	8.175316	2.708030	0.331973	0.537786	27.856807
std	6.184136	1.400063	0.470928	0.498577	28.020395
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.000000	2.000000	0.000000	0.000000	0.000000
50%	7.000000	2.000000	0.000000	1.000000	21.000000
75%	12.000000	3.000000	1.000000	1.000000	54.000000
max	30.000000	11.000000	1.000000	1.000000	133.000000

```
              hsize          inid      intmonth       married       retired  \
count  34298.000000  3.429800e+04  34298.000000  34298.000000  34298.000000
mean       3.441396  1.768912e+11      7.525133      0.874628      0.284040
std        1.702677  1.063349e+11      0.916219      0.331145      0.450962
min        1.000000  1.010410e+10      1.000000      0.000000      0.000000
25%        2.000000  7.400430e+10      7.000000      1.000000      0.000000
50%        3.000000  1.624310e+11      7.000000      1.000000      0.000000
75%        4.000000  2.811160e+11      8.000000      1.000000      1.000000
max       16.000000  3.477630e+11     12.000000      1.000000      1.000000

             schadj         urban          wave        wealth           age
count  34298.000000  34298.000000  34298.000000  3.429800e+04  34298.000000
mean       4.676949      0.356056      1.999242  9.225632e+03     58.251939
std        3.857684      0.478839      0.816449  6.025820e+04      9.095483
min        0.000000      0.000000      1.000000 -1.000000e+06     29.000000
25%        0.000000      0.000000      1.000000  0.000000e+00     51.000000
50%        4.000000      0.000000      2.000000  4.000000e+02     58.000000
75%        8.000000      1.000000      3.000000  5.000000e+03     64.000000
max       18.000000      1.000000      3.000000  1.112000e+06     89.000000
```

Para la columna 'inid', las observaciones se mezclan cuando el número es muy grande, ya que este se lee como notación cientifica desde el archivo original. Es por eso que se procede a eliminar las tuplas que se mezclan entre ellas. Se dejan solo las observaciones que tiene un 'inid' que se repite hasta 3 veces.

[162]:
```
#Hay observaciones mezcladas, dado el gran tamaño de 'inid'. Se eliminan las␣
 ↪tuplas que se repiten más de 3 veces
t = charls['inid'].value_counts().to_frame(name='contar')

indexC = t[ (t['contar'] <= 3)].index.astype(np.int64)
t.drop(indexC , inplace=True)

for i in t.index.astype(int):
  charls.drop(charls[charls['inid']==i].index, inplace=True)

#variable construction
X =␣
 ↪charls[['cesd','child','drinkly','hrsusu','hsize','intmonth','married','retired','schadj','
Xm = (X.groupby(charls['inid']).transform('mean'))
Xid =␣
 ↪charls[['inid','wave','cesd','child','drinkly','hrsusu','hsize','intmonth','married','retir
Xc=pd.DataFrame(np.c_[Xid, Xm],␣
 ↪columns=['inid','wave','cesd','child','drinkly','hrsusu','hsize','intmonth','married','reti
Xc['inid'] = Xc['inid'].astype(np.int64)

#set panel structure
Xc = Xc.set_index(["inid","wave"])
```

```
Xc
```

[162]:

```
                    cesd  child  drinkly  hrsusu  hsize  intmonth  married  \
inid        wave
10104101001 1.0      6.0    2.0      0.0     0.0    4.0       7.0      1.0
            2.0      7.0    2.0      0.0    49.0    4.0       7.0      1.0
            3.0      5.0    2.0      0.0    56.0    7.0       8.0      1.0
10104101002 1.0      0.0    2.0      1.0    63.0    4.0       7.0      1.0
            2.0      5.0    2.0      1.0    49.0    4.0       7.0      1.0
…                    …      …        …      …       …         …        …
94004303002 1.0      0.0    1.0      0.0     0.0    2.0       8.0      1.0
            2.0      5.0    1.0      0.0     0.0    5.0       7.0      1.0
94004308001 1.0      4.0    2.0      0.0     0.0    2.0       8.0      0.0
            2.0      5.0    2.0      0.0     0.0    2.0       7.0      0.0
            3.0      5.0    2.0      0.0     0.0    4.0       8.0      0.0

                    retired  schadj  urban  …  mdrinkly  mhrsusu    mhsize  \
inid        wave                             …
10104101001 1.0         0.0     0.0    0.0  …       0.0     35.0  5.000000
            2.0         0.0     0.0    0.0  …       0.0     35.0  5.000000
            3.0         0.0     0.0    0.0  …       0.0     35.0  5.000000
10104101002 1.0         0.0     4.0    0.0  …       1.0     56.0  5.000000
            2.0         0.0     4.0    0.0  …       1.0     56.0  5.000000
…                       …       …      …    …       …       …        …
94004303002 1.0         1.0     8.0    1.0  …       0.0      0.0  3.500000
            2.0         1.0     8.0    1.0  …       0.0      0.0  3.500000
94004308001 1.0         1.0     8.0    1.0  …       0.0      0.0  2.666667
            2.0         1.0     8.0    1.0  …       0.0      0.0  2.666667
            3.0         1.0     8.0    1.0  …       0.0      0.0  2.666667

                    mintmonth  mmarried  mretired  mschadj  murban  \
inid        wave
10104101001 1.0      7.333333       1.0       0.0      0.0     0.0
            2.0      7.333333       1.0       0.0      0.0     0.0
            3.0      7.333333       1.0       0.0      0.0     0.0
10104101002 1.0      7.333333       1.0       0.0      4.0     0.0
            2.0      7.333333       1.0       0.0      4.0     0.0
…                       …          …         …        …       …
94004303002 1.0      7.500000       1.0       1.0      8.0     1.0
            2.0      7.500000       1.0       1.0      8.0     1.0
94004308001 1.0      7.666667       0.0       1.0      8.0     1.0
            2.0      7.666667       0.0       1.0      8.0     1.0
            3.0      7.666667       0.0       1.0      8.0     1.0

                         mwealth  mage
inid        wave
10104101001 1.0   -21890.000000  46.0
```

```
             2.0   -21890.000000  46.0
             3.0   -21890.000000  46.0
10104101002 1.0    -21890.000000  48.0
             2.0   -21890.000000  48.0
…                          …    …
94004303002 1.0    255015.000000  69.0
             2.0   255015.000000  69.0
94004308001 1.0     11833.333333  61.0
             2.0     11833.333333  61.0
             3.0     11833.333333  61.0

[10033 rows x 24 columns]
```

4 2. Pooled OLS

[163]:
```
y=Xc['cesd']
X=Xc[['child','drinkly','hrsusu','hsize','intmonth','married','retired','schadj','urban','weal
X=sm.add_constant(X)

model = sm.OLS(y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cesd   R-squared:                       0.066
Model:                           OLS   Adj. R-squared:                  0.065
Method:                Least Squares   F-statistic:                     64.86
Date:               Wed, 05 Oct 2022   Prob (F-statistic):          9.78e-141
Time:                       23:57:07   Log-Likelihood:                -32341.
No. Observations:              10033   AIC:                         6.471e+04
Df Residuals:                  10021   BIC:                         6.479e+04
Df Model:                         11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         12.8629      0.695     18.501      0.000      11.500      14.226
child          0.1122      0.049      2.289      0.022       0.016       0.208
drinkly       -0.6015      0.134     -4.496      0.000      -0.864      -0.339
hrsusu         0.0047      0.003      1.627      0.104      -0.001       0.010
hsize         -0.0959      0.035     -2.709      0.007      -0.165      -0.027
intmonth       0.1224      0.056      2.191      0.028       0.013       0.232
married       -1.5135      0.187     -8.111      0.000      -1.879      -1.148
retired        0.6715      0.183      3.660      0.000       0.312       1.031
schadj        -0.2469      0.018    -13.575      0.000      -0.283      -0.211
```

```
urban          -1.7353      0.139    -12.513      0.000      -2.007      -1.463
wealth     -8.115e-06    1.1e-06     -7.355      0.000   -1.03e-05   -5.95e-06
age            -0.0355      0.008     -4.296      0.000      -0.052      -0.019
==============================================================================
Omnibus:                      671.821   Durbin-Watson:                   1.337
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              815.648
Skew:                           0.698   Prob(JB):                    7.66e-178
Kurtosis:                       3.058   Cond. No.                     6.51e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.51e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning:

In a future version of pandas all arguments of concat except for the argument
'objs' will be keyword-only
```

[164]:
```
model=lmp.PooledOLS(y, X.astype(float))
OLS=model.fit(cov_type="robust")
print(OLS)
```

```
                          PooledOLS Estimation Summary
==============================================================================
Dep. Variable:                   cesd   R-squared:                        0.0665
Estimator:                  PooledOLS   R-squared (Between):              0.1031
No. Observations:               10033   R-squared (Within):              -0.0058
Date:                Wed, Oct 05 2022   R-squared (Overall):              0.0665
Time:                        23:57:07   Log-likelihood                -3.234e+04
Cov. Estimator:              Robust
                                        F-statistic:                      64.855
Entities:                        3348   P-value                           0.0000
Avg Obs:                       2.9967   Distribution:                F(11,10021)
Min Obs:                       2.0000
Max Obs:                       3.0000   F-statistic (robust):             66.643
                                        P-value                           0.0000
Time periods:                       3   Distribution:                F(11,10021)
Avg Obs:                       3344.3
Min Obs:                       3340.0
Max Obs:                       3347.0

                            Parameter Estimates
==============================================================================
```

```
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const          12.863     0.6962     18.475     0.0000      11.498      14.228
child          0.1122     0.0500     2.2445     0.0248      0.0142      0.2101
drinkly       -0.6015     0.1312    -4.5831     0.0000     -0.8588     -0.3443
hrsusu         0.0047     0.0029     1.6519     0.0986     -0.0009      0.0103
hsize         -0.0959     0.0347    -2.7610     0.0058     -0.1640     -0.0278
intmonth       0.1224     0.0546     2.2415     0.0250      0.0154      0.2294
married       -1.5135     0.2004    -7.5513     0.0000     -1.9064     -1.1206
retired        0.6715     0.1853     3.6234     0.0003      0.3082      1.0347
schadj        -0.2469     0.0175    -14.130     0.0000     -0.2811     -0.2126
urban         -1.7353     0.1328    -13.070     0.0000     -1.9956     -1.4751
wealth     -8.115e-06  9.978e-07    -8.1329     0.0000  -1.007e-05  -6.159e-06
age           -0.0355     0.0084    -4.2380     0.0000     -0.0519     -0.0191
==============================================================================
```

Dados los resultados del modelo, y los respectivos valores-p se deberían incluir en el modelo final las variables child, drinkly, hsize, intmonth, married, retired, schadj, urban, wealth y age. La variable hrsusu no se debe agregar al modelo puesto que su valor-p es mayor al 5%.

5 3. Efectos fijos

[165]:
```
X=Xc[['child','drinkly','hrsusu','hsize','intmonth','married','retired','schadj','urban','weal
X=sm.add_constant(X)
model=lmp.PanelOLS(y,X, entity_effects=True, drop_absorbed=True)
fe=model.fit(cov_type="robust")
print(fe)
```

```
                          PanelOLS Estimation Summary
================================================================================
Dep. Variable:                    cesd   R-squared:                        0.0041
Estimator:                   PanelOLS   R-squared (Between):              0.0146
No. Observations:               10033   R-squared (Within):               0.0041
Date:                Wed, Oct 05 2022   R-squared (Overall):              0.0110
Time:                        23:57:08   Log-likelihood                -2.718e+04
Cov. Estimator:                Robust
                                        F-statistic:                      3.4178
Entities:                        3348   P-value                           0.0006
Avg Obs:                       2.9967   Distribution:                  F(8,6677)
Min Obs:                       2.0000
Max Obs:                       3.0000   F-statistic (robust):             3.0219
                                        P-value                           0.0022
Time periods:                       3   Distribution:                  F(8,6677)
Avg Obs:                       3344.3
Min Obs:                       3340.0
Max Obs:                       3347.0
```

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	9.9094	0.6705	14.779	0.0000	8.5951	11.224
child	0.1548	0.0970	1.5957	0.1106	-0.0354	0.3449
drinkly	0.2015	0.1882	1.0711	0.2842	-0.1673	0.5704
hrsusu	-0.0004	0.0029	-0.1327	0.8944	-0.0061	0.0054
hsize	-0.1205	0.0444	-2.7142	0.0067	-0.2075	-0.0335
intmonth	-0.0189	0.0507	-0.3726	0.7094	-0.1184	0.0806
married	-1.1916	0.5044	-2.3626	0.0182	-2.1804	-0.2029
retired	0.3584	0.2021	1.7736	0.0762	-0.0377	0.7544
wealth	-1.294e-06	9.439e-07	-1.3705	0.1706	-3.144e-06	5.567e-07

F-test for Poolability: 3.8126
P-value: 0.0000
Distribution: F(3347,6677)

Included effects: Entity

/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning:

In a future version of pandas all arguments of concat except for the argument
'objs' will be keyword-only

/usr/local/lib/python3.7/dist-packages/linearmodels/panel/model.py:1833:
AbsorbingEffectWarning:

Variables have been fully absorbed and have removed from the regression:

schadj, urban, age

Para el de modelo efectos fijos se incluye el parámetro “drop_absorbed”, esto dado que por defecto el modelo de efectos fijos presenta problemas para variables que se mantienen fijas en el tiempo como por ejemplo la variable “age”. Y este parámetro elimina todas aquellas variables que generan dependencia lineal. En este caso se eliminan las variables edad, años de escolaridad y zona urbana.

En la tabla resumen del modelo se aprecian los valores-p de la prueba de significancia para cada una de las variables, de lo cual se aprecia que las únicas variables significativas al 95% son el tamaño del hogar y la variable binaria que expresa el estado civil de la persona.

6 4. Efectos aleatorios

[166]:
```
model=lmp.RandomEffects(y,X)
re=model.fit(cov_type="robust")
print(re)
```

```
                          RandomEffects Estimation Summary
================================================================================
Dep. Variable:                    cesd   R-squared:                        0.0354
Estimator:                RandomEffects   R-squared (Between):              0.0962
No. Observations:                 10033   R-squared (Within):               0.0018
Date:                  Wed, Oct 05 2022   R-squared (Overall):              0.0644
Time:                          23:57:08   Log-likelihood                -2.925e+04
Cov. Estimator:                Robust
                                          F-statistic:                      33.416
Entities:                        3348     P-value                           0.0000
Avg Obs:                       2.9967     Distribution:                 F(11,10021)
Min Obs:                       2.0000
Max Obs:                       3.0000     F-statistic (robust):             35.906
                                          P-value                           0.0000
Time periods:                       3     Distribution:                 F(11,10021)
Avg Obs:                       3344.3
Min Obs:                       3340.0
Max Obs:                       3347.0

                             Parameter Estimates
==============================================================================
             Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const           13.692     0.8001     17.114     0.0000      12.124      15.260
child           0.1401     0.0593     2.3644     0.0181      0.0240      0.2563
drinkly        -0.2781     0.1421    -1.9573     0.0503     -0.5566      0.0004
hrsusu          0.0016     0.0026     0.5997     0.5487     -0.0035      0.0067
hsize          -0.1080     0.0351    -3.0770     0.0021     -0.1769     -0.0392
intmonth        0.0319     0.0465     0.6869     0.4921     -0.0592      0.1230
married        -1.4800     0.2492    -5.9395     0.0000     -1.9684     -0.9916
retired         0.4992     0.1745     2.8605     0.0042      0.1571      0.8413
schadj         -0.2613     0.0238    -10.965     0.0000     -0.3080     -0.2146
urban          -1.6888     0.1816    -9.2996     0.0000     -2.0448     -1.3329
wealth     -4.319e-06  8.526e-07    -5.0654     0.0000   -5.99e-06  -2.648e-06
age            -0.0383     0.0108    -3.5356     0.0004     -0.0596     -0.0171
==============================================================================
```

[167]:
```
re.variance_decomposition
```

[167]:
```
Effects                 16.909895
Residual                19.858395
```

```
Percent due to Effects     0.459904
Name: Variance Decomposition, dtype: float64
```

EL modelo de efectos aleatorios tiene en cuenta la estructura del modelo y, por tanto, es más eficiente que el modelo Pooled OLS. En la tabla resumen del Modelo de Efectos Aleatorios, las variables significativas al 95% son: child, drinkly, hsize, married, retired, schadj, urban, wealth y age.

[168]:
```
re.theta.head()
```

[168]:
```
                theta
inid
10104101001  0.469596
10104101002  0.469596
10104102001  0.469596
10104102002  0.469596
10104103001  0.469596
```

El coeficiente theta determina el grado de degradación que se produce. Cuando los paneles no están equilibrados, el valor de theta varia entre las entidades, pero en este panel equilibrado todos los valores son iguales.

7 5. Comparación y Análisis

[169]:
```
print(lmp.compare({"FE": fe, "RE": re, "Pooled": OLS}))
```

```
                            Model Comparison
=============================================================================
                                    FE                RE         Pooled
-----------------------------------------------------------------------------
Dep. Variable                     cesd              cesd           cesd
Estimator                     PanelOLS     RandomEffects      PooledOLS
No. Observations                 10033             10033          10033
Cov. Est.                       Robust            Robust         Robust
R-squared                       0.0041            0.0354         0.0665
R-Squared (Within)              0.0041            0.0018        -0.0058
R-Squared (Between)             0.0146            0.0962         0.1031
R-Squared (Overall)             0.0110            0.0644         0.0665
F-statistic                     3.4178            33.416         64.855
P-value (F-stat)                0.0006            0.0000         0.0000
====================     ============   ===============   ============
const                           9.9094            13.692         12.863
                              (14.779)          (17.114)       (18.475)
child                           0.1548            0.1401         0.1122
                              (1.5957)          (2.3644)       (2.2445)
drinkly                         0.2015           -0.2781        -0.6015
                              (1.0711)         (-1.9573)      (-4.5831)
```

```
hrsusu                        -0.0004           0.0016          0.0047
                             (-0.1327)         (0.5997)        (1.6519)
hsize                         -0.1205           -0.1080         -0.0959
                             (-2.7142)         (-3.0770)       (-2.7610)
intmonth                      -0.0189           0.0319          0.1224
                             (-0.3726)         (0.6869)        (2.2415)
married                       -1.1916           -1.4800         -1.5135
                             (-2.3626)         (-5.9395)       (-7.5513)
retired                       0.3584            0.4992          0.6715
                              (1.7736)          (2.8605)        (3.6234)
wealth                     -1.294e-06        -4.319e-06      -8.115e-06
                             (-1.3705)         (-5.0654)       (-8.1329)
schadj                                         -0.2613         -0.2469
                                              (-10.965)       (-14.130)
urban                                          -1.6888         -1.7353
                                              (-9.2996)       (-13.070)
age                                            -0.0383         -0.0355
                                              (-3.5356)       (-4.2380)
====================== ============= ================ ==============
Effects                        Entity
-------------------------------------------------------------------------
```

T-stats reported in parentheses

Entre los modelos Pooled OLS y RE, se consideran las mismas variables como significativas, a excepción de la variable intmonth que es significativa solo en el modelo Pooled OLS.

[170]:
```
import numpy.linalg as la
from scipy import stats

def hausman(fe, re):
  diff = fe.params-re.params
  psi = fe.cov - re.cov
  dof = diff.size -1
  W = diff.dot(la.inv(psi)).dot(diff)
  pval = stats.chi2.sf(W, dof)
  return W, dof, pval
```

[171]:
```
htest = hausman(fe, re)
print("Hausman Test: chi-2 = {0}, df = {1}, p-value = {2}".format(htest[0],␣
 ↪htest[1], htest[2]))
```

```
Hausman Test: chi-2 = nan, df = 11, p-value = nan
```

El test de Hausman es un contraste clásico de robustez frente a la eficiencia en los estimadores.

Si el valor p del test Chi cuadrado es mayor al 5% de significancia, el modelo de efectos aleatorios es consistente y eficiente y el modelo de efectos fijos es consistente e ineficiente.

8 6. Efectos aleatorios correlacionados (CRE)

[172]:
```
X=Xc[['child','drinkly','hrsusu','hsize','intmonth','married','retired','schadj','urban','weal
X=sm.add_constant(X)
model=lmp.RandomEffects(y,X)
cre=model.fit(cov_type="robust")
print(cre)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:142:
FutureWarning:

In a future version of pandas all arguments of concat except for the argument
'objs' will be keyword-only
```

RandomEffects Estimation Summary

Dep. Variable:	cesd	R-squared:	0.0354
Estimator:	RandomEffects	R-squared (Between):	0.0962
No. Observations:	10033	R-squared (Within):	0.0018
Date:	Wed, Oct 05 2022	R-squared (Overall):	0.0644
Time:	23:57:08	Log-likelihood	-2.925e+04
Cov. Estimator:	Robust		
		F-statistic:	33.416
Entities:	3348	P-value	0.0000
Avg Obs:	2.9967	Distribution:	F(11,10021)
Min Obs:	2.0000		
Max Obs:	3.0000	F-statistic (robust):	35.906
		P-value	0.0000
Time periods:	3	Distribution:	F(11,10021)
Avg Obs:	3344.3		
Min Obs:	3340.0		
Max Obs:	3347.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	13.692	0.8001	17.114	0.0000	12.124	15.260
child	0.1401	0.0593	2.3644	0.0181	0.0240	0.2563
drinkly	-0.2781	0.1421	-1.9573	0.0503	-0.5566	0.0004
hrsusu	0.0016	0.0026	0.5997	0.5487	-0.0035	0.0067
hsize	-0.1080	0.0351	-3.0770	0.0021	-0.1769	-0.0392
intmonth	0.0319	0.0465	0.6869	0.4921	-0.0592	0.1230
married	-1.4800	0.2492	-5.9395	0.0000	-1.9684	-0.9916
retired	0.4992	0.1745	2.8605	0.0042	0.1571	0.8413
schadj	-0.2613	0.0238	-10.965	0.0000	-0.3080	-0.2146
urban	-1.6888	0.1816	-9.2996	0.0000	-2.0448	-1.3329

```
wealth     -4.319e-06  8.526e-07    -5.0654     0.0000   -5.99e-06  -2.648e-06
age           -0.0383     0.0108    -3.5356     0.0004     -0.0596     -0.0171
==============================================================================
```

Según el Modelo Aleatorios Correlacionados (CRE), son significativas las variables child, drinkly, hsize married, retired, schadj, urban, wealth y age, ya que presentan un valor p menor al 5%.

En el caso de la varible drinkly, aunque tenga un valor p mayor al 5%, se considera significativa porque es levemente menor a 0,05.

9 7. Predicción del componente no observado

10 8. Comparación y Análisis

[173]:
```
print(lmp.compare({"FE": fe, "RE": re, "CRE": cre}))
```

```
                            Model Comparison
================================================================================
                                    FE                RE                CRE
--------------------------------------------------------------------------------
Dep. Variable                     cesd              cesd              cesd
Estimator                     PanelOLS     RandomEffects     RandomEffects
No. Observations                 10033             10033             10033
Cov. Est.                       Robust            Robust            Robust
R-squared                       0.0041            0.0354            0.0354
R-Squared (Within)              0.0041            0.0018            0.0018
R-Squared (Between)             0.0146            0.0962            0.0962
R-Squared (Overall)             0.0110            0.0644            0.0644
F-statistic                     3.4178            33.416            33.416
P-value (F-stat)                0.0006            0.0000            0.0000
====================     ============   ===============   ===============
const                           9.9094            13.692            13.692
                               (14.779)          (17.114)          (17.114)
child                           0.1548            0.1401            0.1401
                               (1.5957)          (2.3644)          (2.3644)
drinkly                         0.2015           -0.2781           -0.2781
                               (1.0711)          (-1.9573)         (-1.9573)
hrsusu                         -0.0004            0.0016            0.0016
                              (-0.1327)          (0.5997)          (0.5997)
hsize                          -0.1205           -0.1080           -0.1080
                              (-2.7142)          (-3.0770)         (-3.0770)
intmonth                       -0.0189            0.0319            0.0319
                              (-0.3726)          (0.6869)          (0.6869)
married                        -1.1916           -1.4800           -1.4800
                              (-2.3626)          (-5.9395)         (-5.9395)
retired                         0.3584            0.4992            0.4992
                               (1.7736)          (2.8605)          (2.8605)
wealth                       -1.294e-06        -4.319e-06        -4.319e-06
```

```
                              (-1.3705)         (-5.0654)         (-5.0654)
schadj                                           -0.2613           -0.2613
                                                (-10.965)         (-10.965)
urban                                            -1.6888           -1.6888
                                                (-9.2996)         (-9.2996)
age                                              -0.0383           -0.0383
                                                (-3.5356)         (-3.5356)
======================= ============== ================= =================
Effects                            Entity
-------------------------------------------------------------------------------

T-stats reported in parentheses
```

Los estadísticos F utilizados resultaron ser altamente significativos en los tres modelos. Esto confirma que el set de variables empleadas en cada modelo están correctamente especificadas.

En los tres modelos se puede apreciar un aumento en el puntaje CESD ante un aumento en la cantidad de hijos y la posibilidad de estar pensionado/a. Además el aumento en el tamaño del hogar, en la riqueza neta, años de escolaridad o si vive en una zona urbana traerá una disminuciòn en el puntaje CESD.

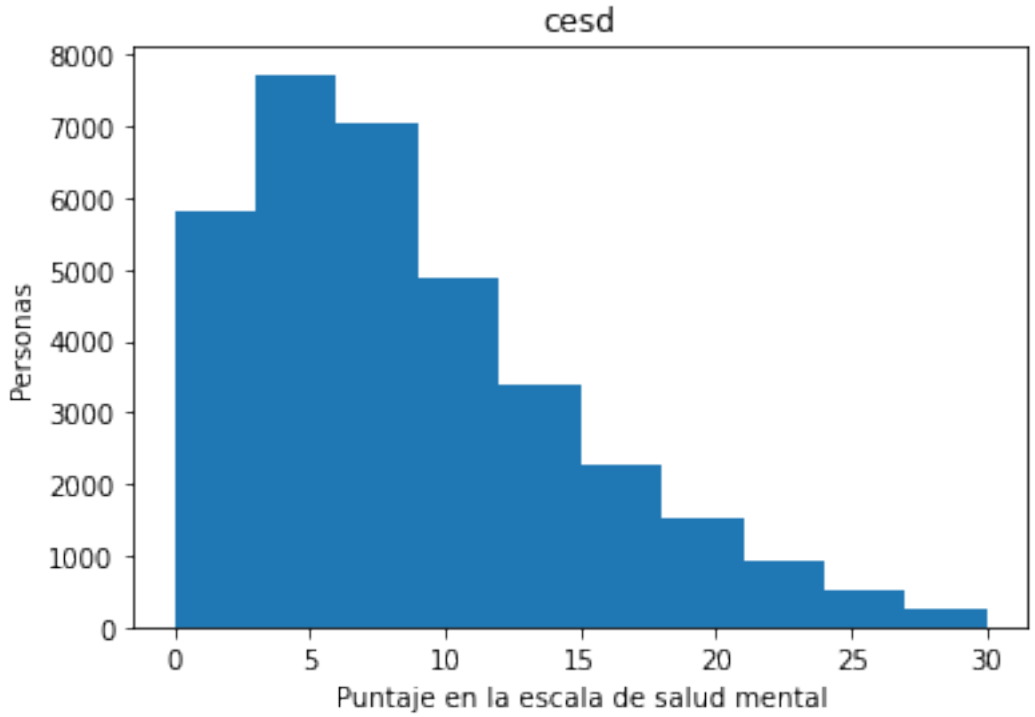
11 Anexos

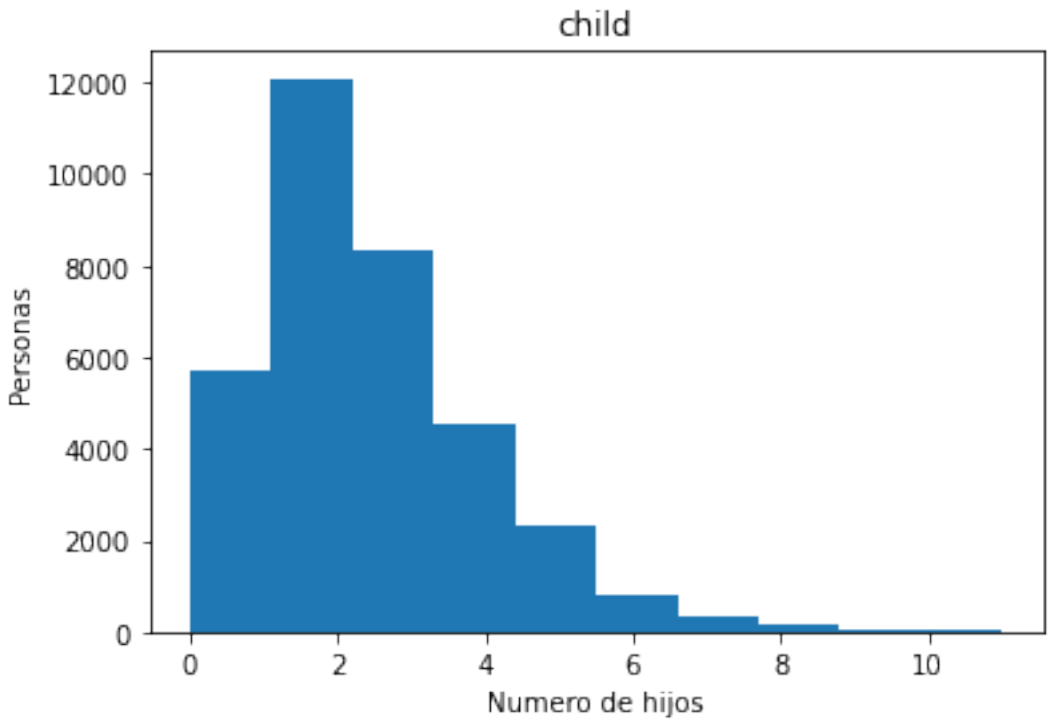
11.1 Anexo 1: Distribuciones de variables

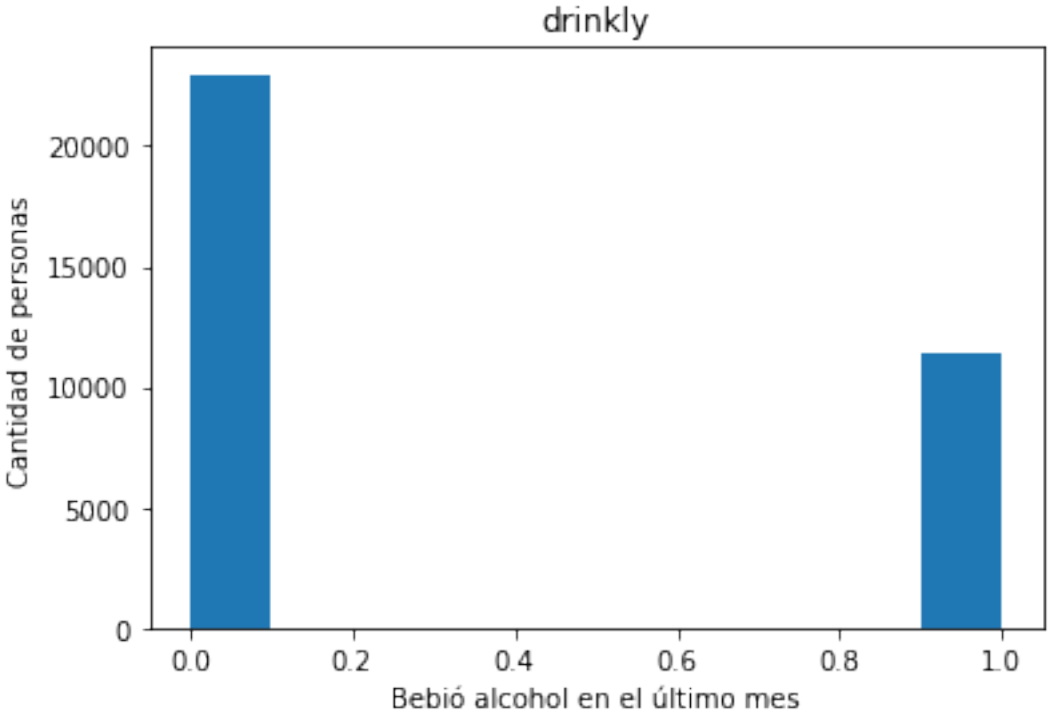
[174]:
```
plt.hist(ch.cesd)
plt.title(ch.columns.values[0])
plt.xlabel('Puntaje en la escala de salud mental')
plt.ylabel('Personas')
plt.show()

plt.hist(ch.child)
plt.title(ch.columns.values[1])
plt.xlabel('Numero de hijos')
plt.ylabel('Personas')
plt.show()

plt.hist(ch.drinkly)
plt.title(ch.columns.values[2])
plt.xlabel('Bebió alcohol en el último mes')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.female)
plt.title(ch.columns.values[3])
plt.xlabel('Persona entrevistada es mujer')
plt.ylabel('Cantidad de personas')
plt.show()
```

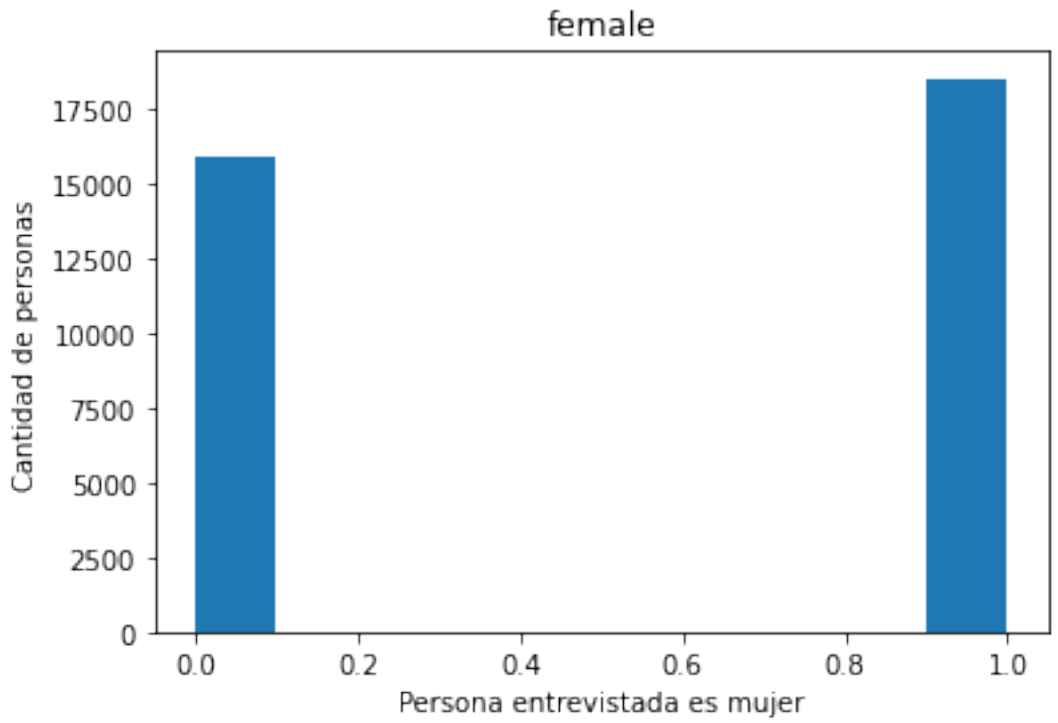
```
plt.hist(ch.hrsusu)
plt.title(ch.columns.values[4])
plt.xlabel('Horas promedio trabajo semanal')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.hsize)
plt.title(ch.columns.values[5])
plt.xlabel('Tamaño del hogar')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.intmonth        )
plt.title(ch.columns.values[7])
plt.xlabel('Mes en que fue encuestado/a')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.married)
plt.title(ch.columns.values[8])
plt.xlabel('Si está casado/a')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.retired        )
plt.title(ch.columns.values[9])
plt.xlabel('Si está pensionado/a')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.schadj)
plt.title(ch.columns.values[10])
plt.xlabel('Años de escolaridad')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.urban)
plt.title(ch.columns.values[11])
plt.xlabel('Zona urbana')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.wave)
plt.title(ch.columns.values[12])
plt.xlabel('Periodo de la encuesta')
plt.ylabel('Cantidad de personas')
```

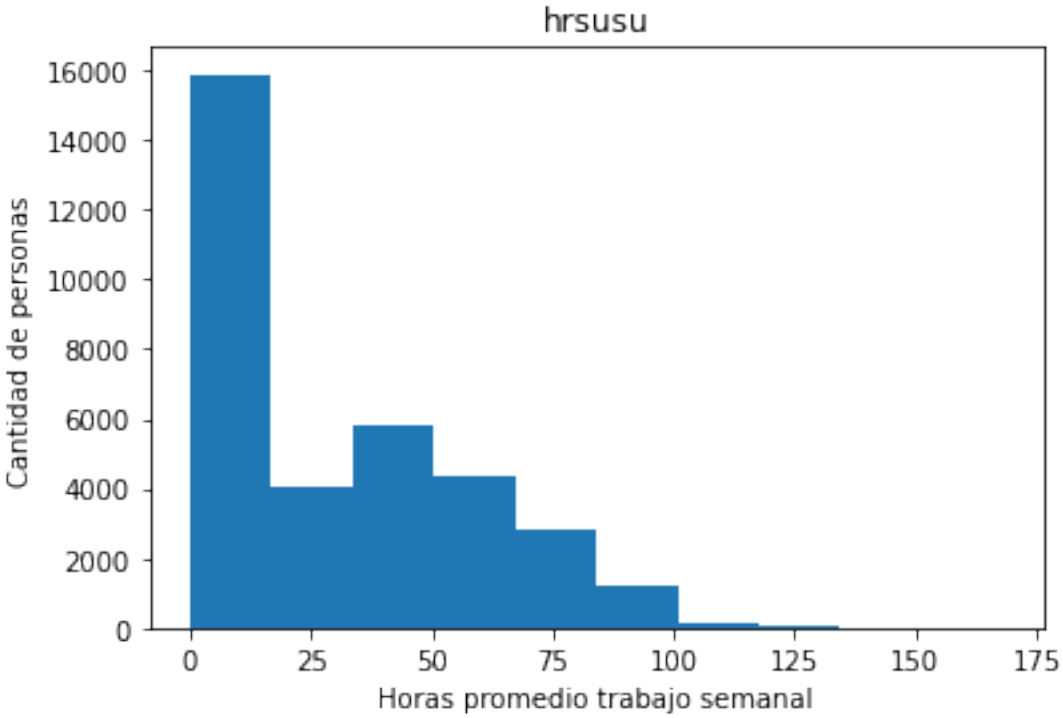
```
plt.show()

plt.hist(ch.wealth)
plt.title(ch.columns.values[13])
plt.xlabel('Riqueza neta (miles RMB)')
plt.ylabel('Cantidad de personas')
plt.show()

plt.hist(ch.age )
plt.title(ch.columns.values[14])
plt.xlabel('Edad al entrar a la encuesta')
plt.ylabel('Cantidad de personas')
plt.show()
```

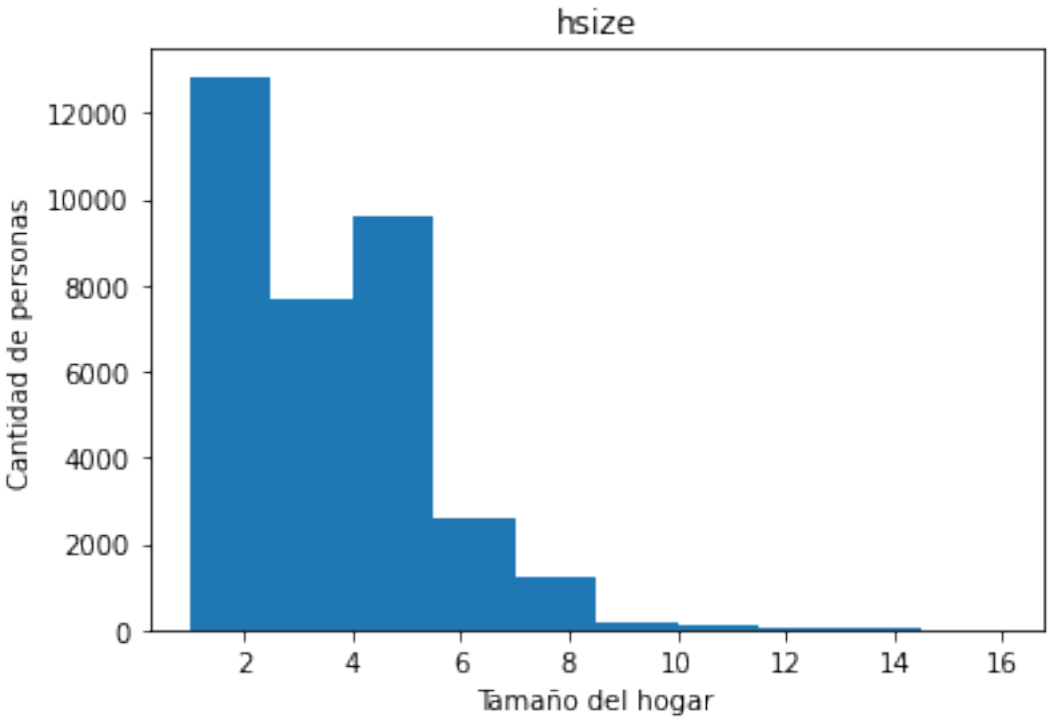


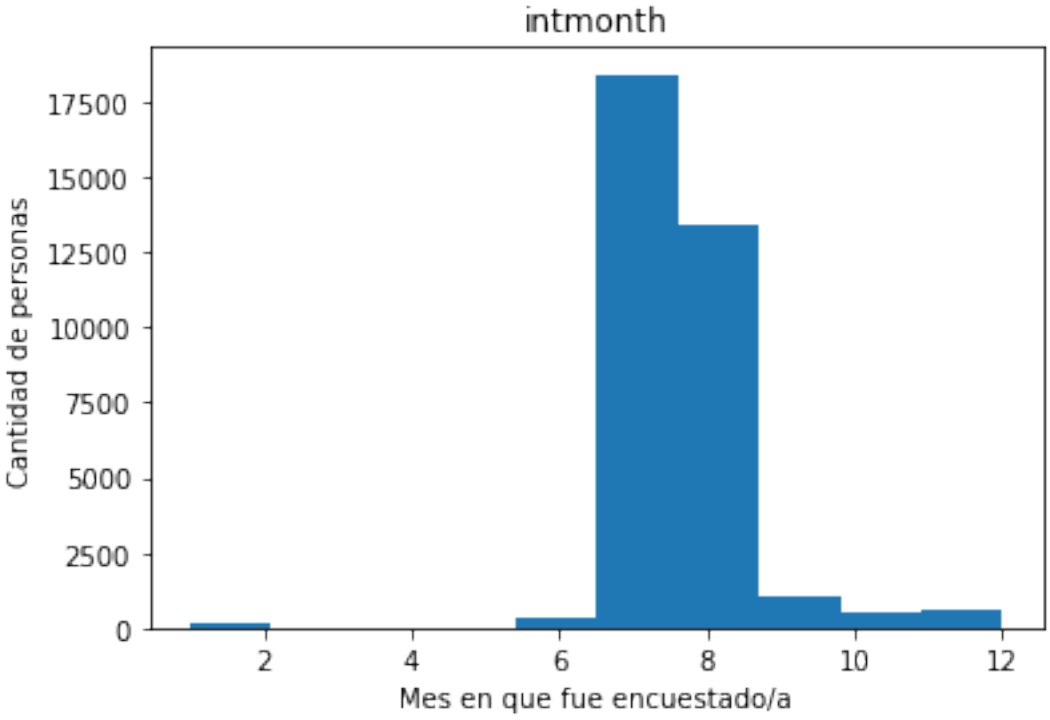

child
Personas
Numero de hijos
0
2000
4000
6000
8000
10000
12000
0
2
4
6
8
10

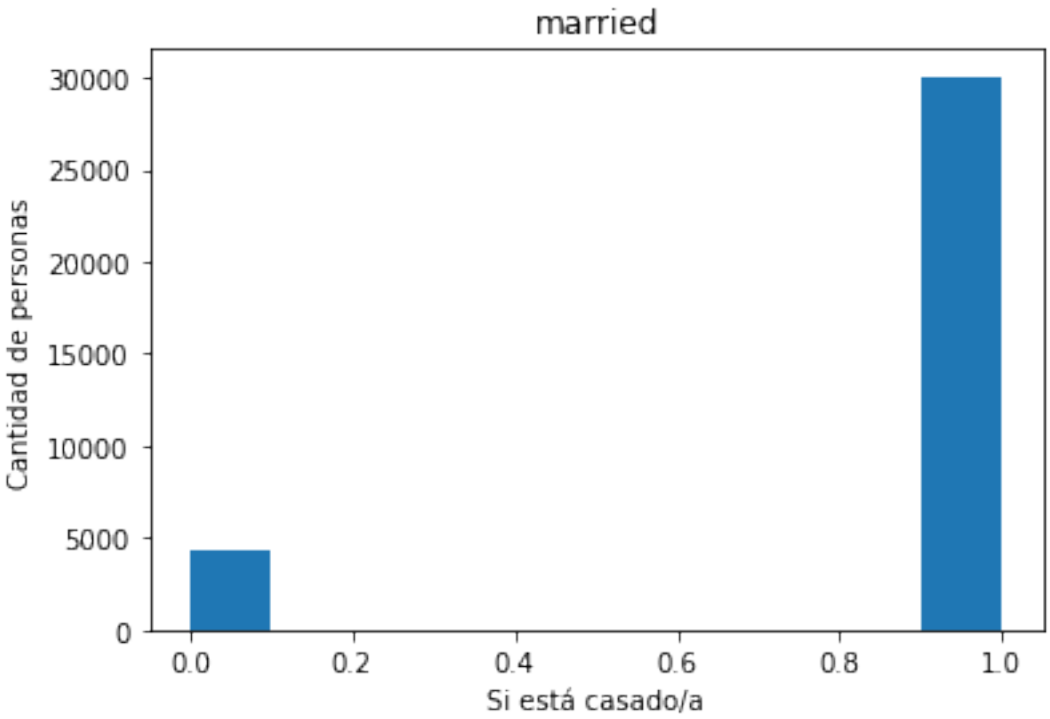

drinkly
Cantidad de personas
Bebió alcohol en el último mes
0
5000
10000
15000
20000
0.0
0.2
0.4
0.6
0.8
1.0

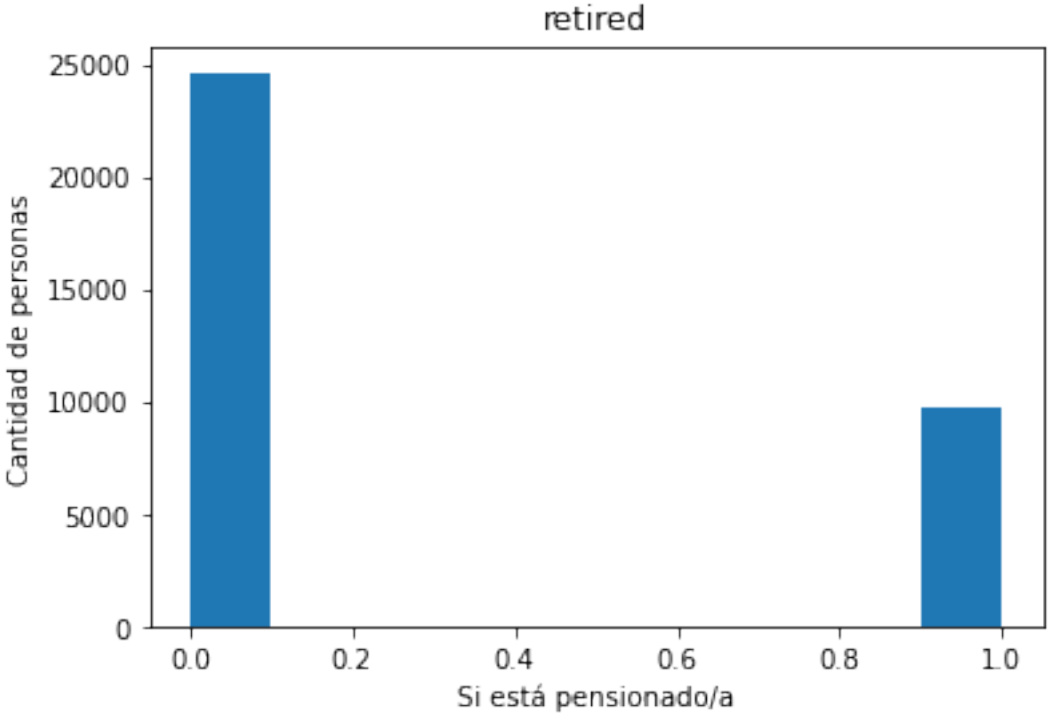

female
Cantidad de personas
17500
15000
12500
10000
7500
5000
2500
0
0.0
0.2
0.4
0.6
0.8
1.0
Persona entrevistada es mujer

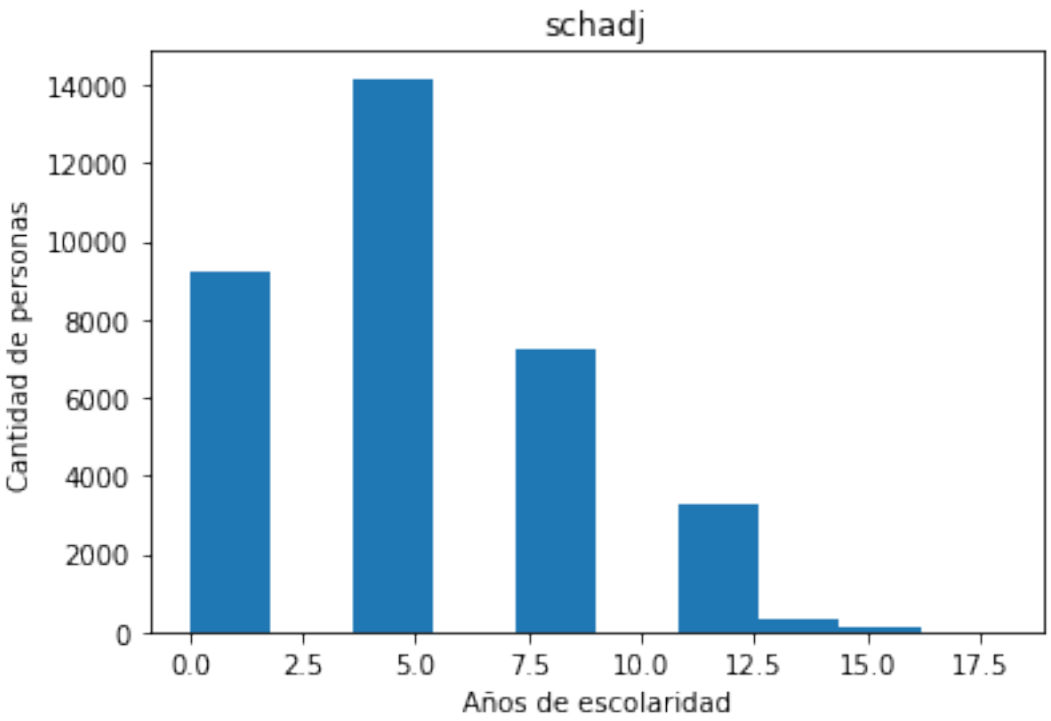

hrsusu
Cantidad de personas
16000
14000
12000
10000
8000
6000
4000
2000
0
0
25
50
75
100
125
150
175
Horas promedio trabajo semanal

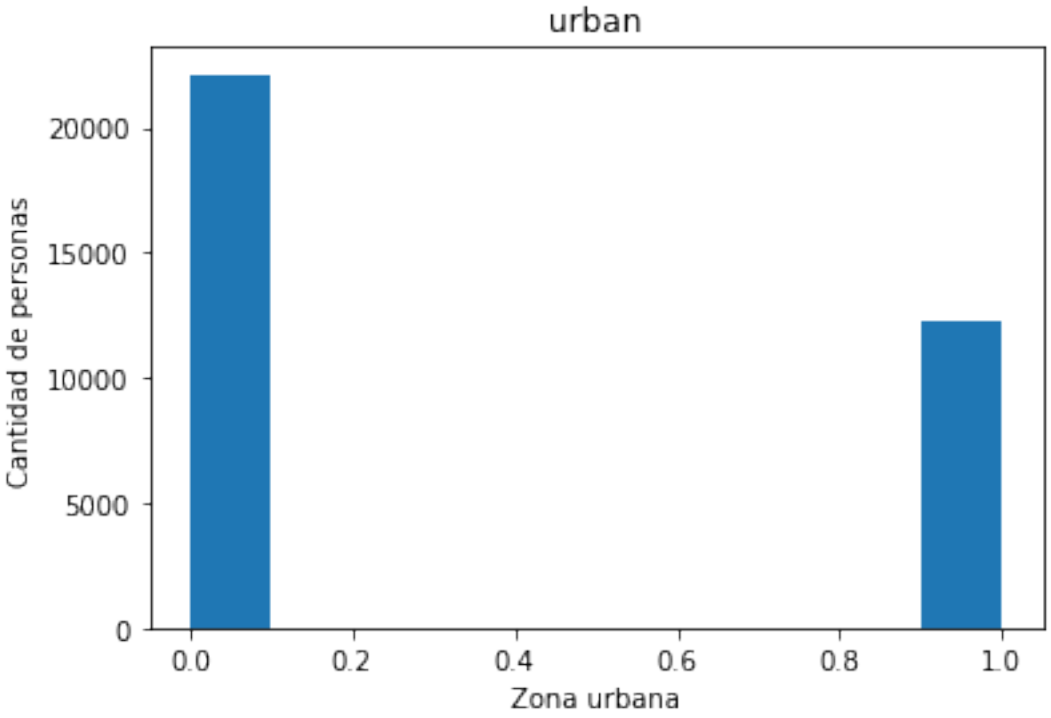

hsize
Cantidad de personas
0
2000
4000
6000
8000
10000
12000
2
4
6
8
10
12
14
16
Tamaño del hogar

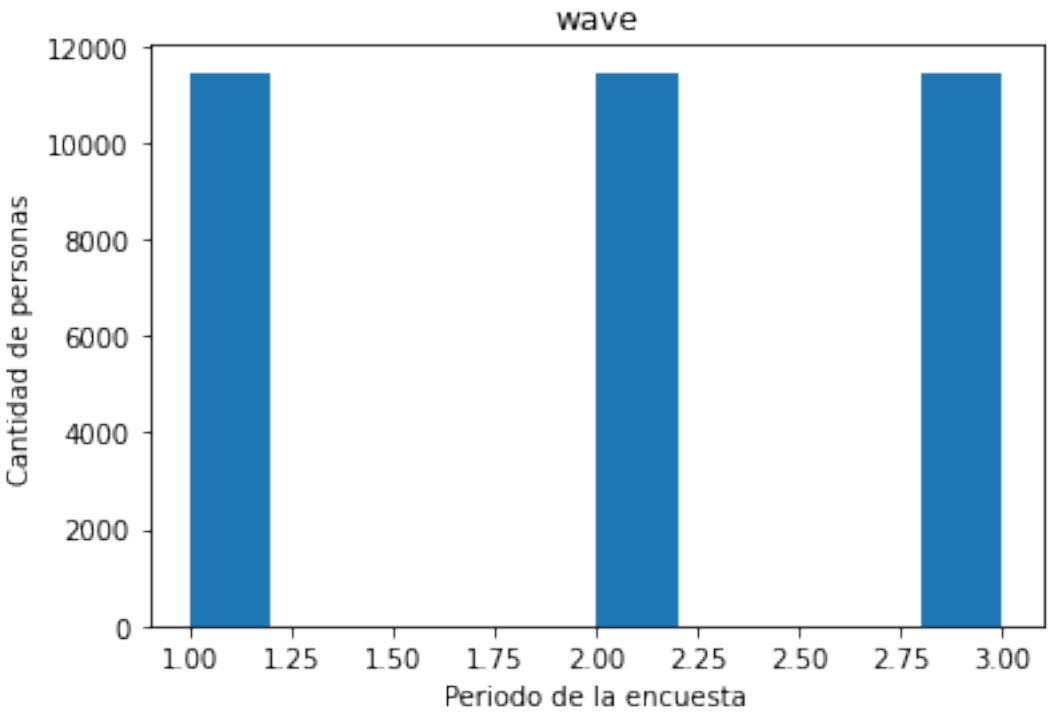

intmonth
Cantidad de personas
0
2500
5000
7500
10000
12500
15000
17500
2
4
6
8
10
12
Mes en que fue encuestado/a

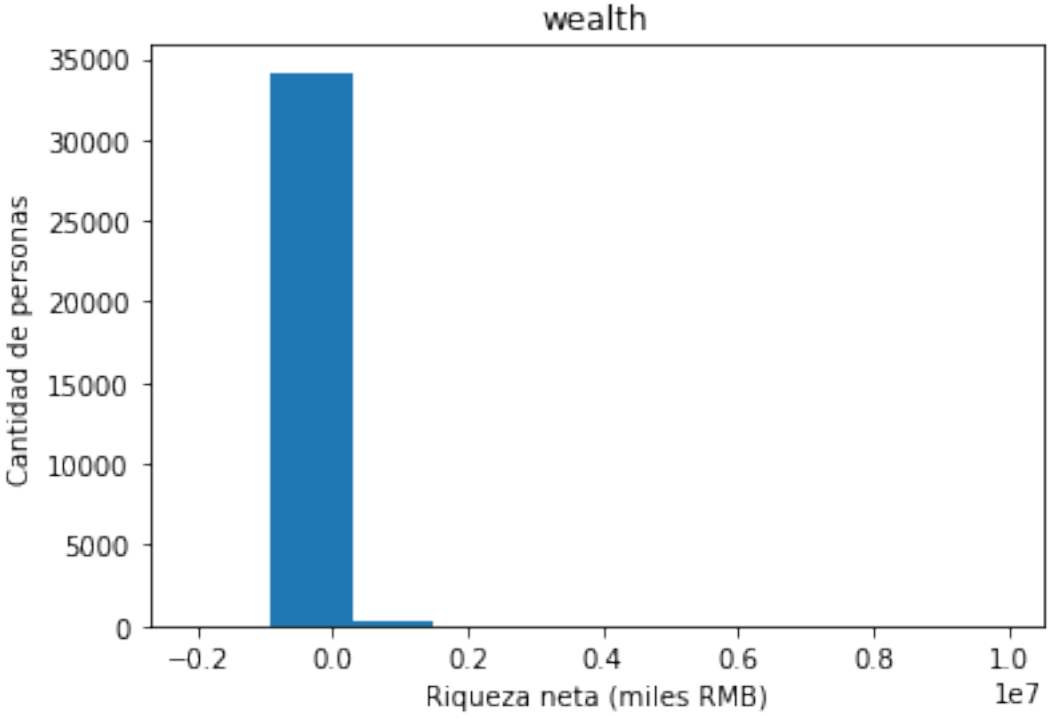

married
30000
25000
20000
15000
10000
5000
0
Cantidad de personas
0.0
0.2
0.4
0.6
0.8
1.0
Si está casado/a

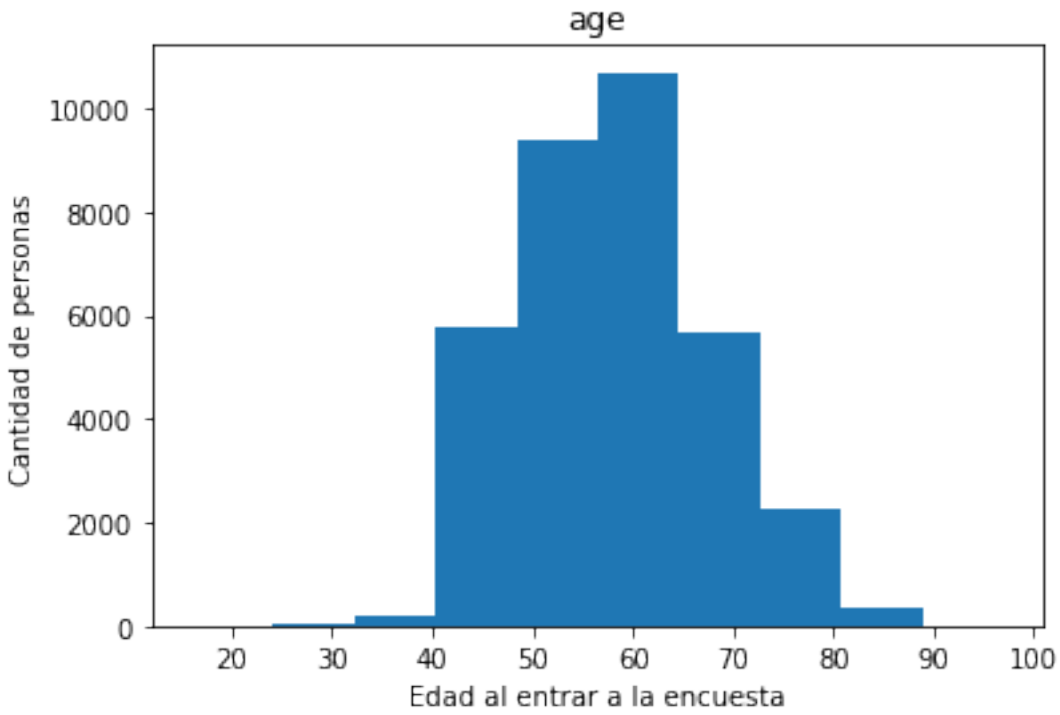

retired
25000
20000
15000
10000
5000
0
Cantidad de personas
0.0
0.2
0.4
0.6
0.8
1.0
Si está pensionado/a


schadj
Cantidad de personas
14000
12000
10000
8000
6000
4000
2000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Años de escolaridad


urban
Cantidad de personas
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
Zona urbana


wave
12000
10000
8000
6000
4000
2000
0
Cantidad de personas
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Periodo de la encuesta


wealth
35000
30000
25000
20000
15000
10000
5000
0
Cantidad de personas
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e7
Riqueza neta (miles RMB)



11.2 Anexo 2: Gráficos para detección de outliers

[175]:
```
import plotly.express as px
fig1 = px.histogram(c, x="age",␣
 ↪marginal="box",color_discrete_sequence=['green'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="cesd",␣
 ↪marginal="box",color_discrete_sequence=['brown'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="hrsusu",␣
 ↪marginal="box",color_discrete_sequence=['magenta'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="intmonth",␣
 ↪marginal="box",color_discrete_sequence=['blue'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="schadj",␣
 ↪marginal="box",color_discrete_sequence=['red'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="wealth",␣
 ↪marginal="box",color_discrete_sequence=['black'],height=500,width=700)
fig1.show()
fig1 = px.histogram(c, x="child",␣
 ↪marginal="box",color_discrete_sequence=['gray'],height=500,width=700)
```

```
fig1.show()
fig1 = px.histogram(c, x="hsize",␣
  ↪marginal="box",color_discrete_sequence=['orange'],height=500,width=700)
fig1.show()
```