Comercio

May 25, 2021

[1]:

```
import numpy as np
import pandas as pd
from scipy.optimize import fsolve
from math import pi as pi
import requests
import plotly.express as px
import statsmodels.api as sm
import json

# página para crear tu goejson: https://geojson-maps.ash.ms/
```

[2]:

```
with open(r'C:\Users\joaco\Documents\UP\8° Semestre\Econometría␣
 ↪espacial\Comercio\mapa.json') as f:
    mapa = json.load(f)

Z = pd.read_excel('Corroboración nombres.xlsx', sheet_name = 'Ej_v')
```

1 Normalizando

[3]:

```
W = pd.read_excel('Matriz de pesos.xlsx', sheet_name = 'Hoja1')
W = W.set_index('Unnamed: 0')
W = W.fillna(0)
W = W.values # convertir el dataframe en array
W = W + W.T
W = np.array(W, dtype = np.float) # todo es flotantes
r,c = W.shape # renglones, columnas
print('renglones:',r,'columnas:',c)
```

```
renglones: 34 columnas: 34
```

[4]:

```
ParaNormalizar = np.sum(W,axis=1).tolist() # sumar elementos de cada renglon
ParaNormalizar = np.tile(ParaNormalizar,(r,1)) # r copias de ParaNormalizar
ParaNormalizar = np.transpose(ParaNormalizar) # transponer
ParaNormalizar = np.reciprocal(ParaNormalizar) # transponer
z = np.where(ParaNormalizar == np.inf, 0, ParaNormalizar)
W = W * z
W = np.array(W, dtype = np.float)
```

```
W = pd.DataFrame(data = W)
```

1.1 Eigenstuff

[5]:

```
val, vec = np.linalg.eig(W)
x = pd.DataFrame(data = val)
x.head()
```

[5]:

```
          0
0  1.000000
1  0.828956
2  0.571643
3 -0.360419
4  0.200611
```

2 MCO del profe sin constante

[6]:

```
z = Z.values # convertir el dataframe en array
n,c = z.shape # renglones, columnas
print('renglones:',n,'columnas:',c)
regiones = np.reshape(z[:,0],(n,1))
y = np.array(np.reshape(z[:,1],(n,1)),dtype = np.float)
x = np.array(np.reshape(z[:,2],(n,1)),dtype = np.float)
```

```
renglones: 34 columnas: 3
```

[7]:

```
beta = np.dot(np.linalg.inv(np.dot(np.transpose(x),x)),np.dot(np.transpose(x),y))
u = y - np.dot(x,beta)
k = c - 2 # # regresores
sigma2 = np.dot(np.transpose(u),u)[0][0]/(n - k) # varianza del residual
SIGMA = sigma2*np.linalg.inv(np.dot(np.transpose(x),x)) # matriz de covarianzas de las estimaciones
ErrEst = np.reshape((np.diagonal(SIGMA))**0.5,(k,1)) # errores estandar de las estimaciones
beta_t = beta / ErrEst # estadistica t
beta = (np.reshape(beta,(1,k))[0]).tolist()
print(beta)
print(beta_t)
print(k)
```

```
[4481074.30034737]
[[0.5023849]]
1
```

3 MCO mío con y sin constante

[8]:
```
X = Z['x1']
#X = sm.add_constant(X)
Y = Z['y0']

model = sm.OLS(Y, X).fit()
model.summary()
```

[8]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                     y0   R-squared (uncentered):
0.008
Model:                            OLS   Adj. R-squared (uncentered):
-0.022
Method:                 Least Squares   F-statistic:
0.2524
Date:                Tue, 25 May 2021   Prob (F-statistic):
0.619
Time:                        00:13:45   Log-Likelihood:
-706.50
No. Observations:                  34   AIC:
1415.
Df Residuals:                      33   BIC:
1417.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1          4.481e+06   8.92e+06      0.502      0.619   -1.37e+07    2.26e+07
==============================================================================
Omnibus:                       62.330   Durbin-Watson:                   1.890
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              520.512
Skew:                           4.097   Prob(JB):                    9.38e-114
Kurtosis:                      20.328   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[9]:
```
X = Z['x1']
X = sm.add_constant(X)
Y = Z['y0']

model = sm.OLS(Y, X).fit()
model.summary()
```

```
C:\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2580: FutureWarning:
Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp
instead.
  return ptp(axis=axis, out=out, **kwargs)
```

[9]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                     y0   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                 -0.029
Method:                 Least Squares   F-statistic:                   0.07086
Date:                Tue, 25 May 2021   Prob (F-statistic):              0.792
Time:                        00:13:46   Log-Likelihood:                -704.72
No. Observations:                  34   AIC:                             1413.
Df Residuals:                      32   BIC:                             1416.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       8.143e+07   4.33e+07      1.880      0.069   -6.81e+06     1.7e+08
x1          2.309e+06   8.67e+06      0.266      0.792   -1.54e+07       2e+07
==============================================================================
Omnibus:                       62.921   Durbin-Watson:                   2.094
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              536.953
Skew:                           4.143   Prob(JB):                    2.52e-117
Kurtosis:                      20.617   Cond. No.                         5.04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

4 Mapa de residuales del MCO del profe

[10]:

```
residuales = np.concatenate((regiones, u), axis = 1) # juntamos el nombre de la␣
 ↪region y su residual en una matriz
errores = pd.DataFrame(residuales,columns = ['País','residual']) # convertimos␣
 ↪la matriz en dataframe
Z = pd.merge(Z,errores,on = 'País')
print(Z.dtypes) # tipo de datos de los campos del dataframe
Z['residual'] = Z['residual'].astype(float)
```

```
País          object
y0           float64
x1           float64
residual      object
dtype: object
```

[11]:

```
fig = px.choropleth(data_frame = Z,
                   geojson = mapa,
                   locations = 'País',
                   featureidkey = 'properties.name',
                   color = 'residual',
                   color_continuous_scale = 'mint' # paleta de colores
                   )

fig.update_geos(showcountries = True,showcoastlines = True,showland =␣
 ↪True,fitbounds = 'locations')
```

5 I de Moran

[12]:

```
I = (np.dot(np.dot(np.transpose(u),W),u) / np.dot(np.transpose(u),u)) [0][0]
Mx = np.eye(n) - np.dot(np.dot(x,np.linalg.inv(np.dot(np.transpose(x),x))),np.
 ↪transpose(x))
S0 = np.sum(W)
S1 = np.sum(S0)
EI = (n*np.matrix.trace(np.dot(Mx,W))) / (S1*(n - k))
varI = np.matrix.trace(np.dot(np.dot(np.dot(Mx,W),Mx),np.transpose(W)))
varI = varI + np.matrix.trace(np.dot(np.dot(np.dot(Mx,W),Mx),W))
varI = varI + (np.matrix.trace(np.dot(Mx,W)))**2
varI = (n/S1)**2*varI/((n - k)*(n - k + 2))
varI = varI - EI**2
z = (I - EI)/varI**0.5
print('I de Moran:',I)
print('E[I] =',EI)
print('var(I) =',varI)
print('z =',z)
```

```
I de Moran: 0.14720865621208307
```

```
E[I] = -0.00044187681156997605
var(I) = 0.004347751760917539
z = 2.2392503423941106
```

6 Mapitas descriptivos

[13]:

```
fig = px.choropleth(data_frame = Z,
                   geojson = mapa,
                   locations = 'País',
                   featureidkey = 'properties.name',
                   color = 'y0',
                   color_continuous_scale = 'mint' # paleta de colores
                   )

fig.update_geos(showcountries = True,showcoastlines = True,showland =␣
 ↪True,fitbounds = 'locations')
```

[14]:

```
fig = px.choropleth(data_frame = Z,
                   geojson = mapa,
                   locations = 'País',
                   featureidkey = 'properties.name',
                   color = 'x1',
                   color_continuous_scale = 'mint' # paleta de colores
                   )

fig.update_geos(showcountries = True,showcoastlines = True,showland =␣
 ↪True,fitbounds = 'locations')
```