

BOOK_PAPER

Jane Doe

2/9/23

Table of contents

Preface	3
1 Introduction	4
2 Data Acquisition and Preprocessing	5
3 Preprocessing	6
3.1 Concentración de desventajas	6
3.1.1 Análisis de Componentes Principales (PCA)	18
3.1.2 Nivel Colonias	19
3.1.3 Nivel Alcaldías	22
4 Desorden Social y Físico	27
4.1 El jardín de las bifurcaciones	29
4.1.1 Corrección columna clave de manzana	29
4.1.2 Caso por Mananzas y Manzanas	31
4.1.3 Desorden Social y Fisico a nivel Colonia	33
4.1.4 Desorden Social y Fisico a nivel Alcaldia	35
4.1.5 Caso por Mananzas y Geometria Punto	40
5 Methods	41
6 Summary	42
References	43

Preface

This is a Quarto book.

To learn more about Quarto books visit <https://quarto.org/docs/books>.

1 Introduction

Aquí va la introducción y algunos antecedentes Tener muy claro el planteamiento del problema
Definir subsecciones

2 Data Acquisition and Preprocessing

Censo de poblacion 2020

Descarga masiva Manzanas

Llamadas del 911

3 Preprocessing

3.1 Concentración de desventajas

Construido a partir de cuatro dimensiones (con base a censo 2010) y reducida a una componente principal por PCA. Las dimensiones que se exponen en el artículo son: - Porcentaje de masculinos de 15 a 29 - Porcentaje de población sin servicios a salud - Promedio de habitantes que ocupan un hogar privado - Porcentaje de personas que hablan una lengua indígena

Con base al Censo de población de 2020, las dimensiones se resumen de la forma:

Clave	Descripción
P_15A17_M	Población masculina de 15 a 17 años
P_18A24_M	Población masculina de 18 a 24 años
PSINDER	Población sin afiliación a servicios de salud
PROM_OCUP	Promedio de ocupantes en viviendas particulares habitadas
P3YM_HLI	Población de 3 años y más que habla alguna lengua indígena

La primer parte consta de cargar la base de datos de censo, seleccionar las dimensiones, limpiar la información y preparala para poder hacerla unión de estas en la geometry de la unidd geografica manzanas. Se cargan las librerías necesarias.

```
## Librerias

import numpy as np
import pandas as pd
import geopandas as gpd
import contextily as ctx
import matplotlib.pyplot as plt
from IPython.display import Markdown
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

### Warnings
import warnings
```

```
warnings.filterwarnings('ignore')
```

Se carga la información del Censo por unidad manzanas conservando únicamente los campos de interés: **ENTIDAD**, **NOM_ENT**, **MUN**, **NOM_MUN LOC**, **NOM_LOC**, **AGEB**, **MZA**, **POBTOT**, **P_15A17_M**, **P_18A24_M**, **PSINDER**, **PROM_OCUP** y **P3YM_HLI**

```
CENSO_2020 = pd.read_csv(Entradas + 'MANZANAS.csv', encoding = 'Latin-1')
CENSO_2020 = CENSO_2020 [['ENTIDAD','NOM_ENT','MUN',
                           'NOM_MUN','LOC','NOM_LOC',
                           'AGEB','MZA','POBTOT',
                           'P_15A17_M','P_18A24_M','PSINDER',
                           'PROM_OCUP', 'P3YM_HLI']]
```

```
CENSO_2020.head(2)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC
0	9	Ciudad de México	0	Total de la entidad Ciudad de México	0	Total de la entidad Ciudad de México
1	9	Ciudad de México	2	Azcapotzalco	0	Total del municipio Azcapotzalco

De la base de datos filtramos eliminando las filas que contienen los totales

```
Values = ['Total de la entidad','Total del municipio',
          'Total de la localidad urbana', 'Total AGEB urbana']
CENSO_2020_USE = CENSO_2020.query("NOM_LOC != @Values")
```

```
CENSO_2020_USE.head(2)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	AGEB	MZA	POBTOT
4	9	Ciudad de México	2	Azcapotzalco	1	Azcapotzalco	10	1	15
5	9	Ciudad de México	2	Azcapotzalco	1	Azcapotzalco	10	2	14

Dentro de la base de datos existe la presencia de filas donde los valores son simbolos o caracteres especiales(*, N/D, 999999, etc), por lo que es necesario remplazarlos por valores NAN.

```
CENSO_2020_USE = CENSO_2020_USE.replace({'999999999': np.nan,
                                             '99999999': np.nan,
                                             '*': np.nan,
                                             'N/D': np.nan})
```

```

DIM_NUM = CENSO_2020_USE.iloc[:, -6: ].columns.tolist()
DIM_TEXT = CENSO_2020_USE.iloc[:, :8].columns.tolist()

CENSO_2020_USE[DIM_NUM] = CENSO_2020_USE[DIM_NUM].astype('float')
CENSO_2020_USE[DIM_TEXT] = CENSO_2020_USE[DIM_TEXT].astype(str)

```

De igual manera la información referida a a las columnas de “Entidad, Municipio, Localidad, AGEB y Manzana”, no presetan el formato necesario para crear la columna CVEGEO, por lo que debemos completar la información de la forma correcta.

Creada la columna CVEGEO, calculamos la dimensión población masculina de 15 a 24 años como la suma de “P_15A17_M y P_18A24_M”. Una vez que hemos creado la dimensión se hacen poco necesarias “P_15A17_M y P_18A24_M, por lo que se eliminan.

```

## Corrección de información
CENSO_2020_USE['ENTIDAD'] = CENSO_2020_USE['ENTIDAD'].str.zfill(2)
CENSO_2020_USE['MUN'] = CENSO_2020_USE['MUN'].str.zfill(3)
CENSO_2020_USE['LOC'] = CENSO_2020_USE['LOC'].str.zfill(4)
CENSO_2020_USE['AGEB'] = CENSO_2020_USE['AGEB'].str.zfill(4)
CENSO_2020_USE['MZA'] = CENSO_2020_USE['MZA'].str.zfill(3)

CENSO_2020_USE['CVEGEO'] = CENSO_2020_USE[['ENTIDAD', 'MUN',
                                             'LOC', 'AGEB', 'MZA']].agg(''.join, axis=1)

## Cálculo de Población masculina de 15 a 24 años
CENSO_2020_USE['P_15A24_M'] = CENSO_2020_USE[["P_15A17_M",
                                                "P_18A24_M"]].sum(axis=1,
                                                                     min_count=1)

## Eliminación de dimensiones
CENSO_2020_USE = CENSO_2020_USE.drop(['P_15A17_M', 'P_18A24_M'], axis = 1)

CENSO_2020_USE.head(2)

```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	AGEB	MZA	POBTO
4	09	Ciudad de México	002	Azcapotzalco	0001	Azcapotzalco	0010	001	159
5	09	Ciudad de México	002	Azcapotzalco	0001	Azcapotzalco	0010	002	145

Se carga la base de datos geoespacial que corresponde a la unidad geográfica de

“manzanas”. Para este caso, el archivo se encuentra en formato “json”. Del archivo, únicamente consideramos las columnas “CVEGEO y geometry”

```
### Se carga el archivo espacial de Manzanas

MANZA_CDMX = gpd.read_file(Entradas +'MANZA_CDMX.json')
MANZA_CDMX = MANZA_CDMX[['CVEGEO','geometry']]
MANZA_CDMX.head(2)
```

	CVEGEO	geometry
0	0901000010898031	POLYGON ((-99.22982 19.35939, -99.23006 19.359...
1	0901000012269024	POLYGON ((-99.21840 19.36539, -99.21848 19.365...

Union (merge) de las unidades geoespaciales con la información del Censo de Población y vivienda 2020. En este punto a cada unidad geográfica le asignamos la información del censo de población.

```
### Union a la izquierda con campo llave primaria "CVEGEO"
MERGE = MANZA_CDMX.merge( CENSO_2020_USE,
                          left_on = 'CVEGEO',
                          right_on = 'CVEGEO',
                          how = 'inner')

MERGE.head(2)
```

	CVEGEO	geometry	ENTIDAD	NOM_ENT
0	0901000010898031	POLYGON ((-99.22982 19.35939, -99.23006 19.359...	09	Ciudad de México
1	0901000012269024	POLYGON ((-99.21840 19.36539, -99.21848 19.365...	09	Ciudad de México

Siempre es importante saber en qué sistema de proyección se encuentran nuestros datos, para eso usamos “crs”

```
MERGE.crs
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
```

Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich

Hacemos un mapa por que nos gustan los mapitas.

```
fig, ax = plt.subplots(figsize=(8, 8))
ax = MERGE.plot(ax = ax, column='P_15A24_M',
                  legend=False,
                  alpha=0.8,
                  scheme='NaturalBreaks',
                  cmap='copper',
                  classification_kwds={'k':6})
ax.set(title='Población Masculina 15 a 24 años, Ciudad de México')

ax.set_axis_off()

plt.show()
```

Población Masculina 15 a 24 años, Ciudad de México



Hasta el punto anterior tenemos la información contenida dentro de las manzanas, el paso que sigue es llevar las manzanas a colonias. Para esto es necesario entender que ambos elementos son poligonales y que los centroides de manzanas no necesariamente refieren a la solución contenida dentro de un polígono mayor.

Por eso es necesario usar el criterio de máxima área de la sobreposición de polígonos. Al hablar de área el sistema de proyección debe estar en metros, por lo que si no

lo esta se debe cambiar. Para este caso se cambio a [EPSG:6362](#)

Se cargan las colonias y se valida que ambos crs se encuentren en metros “6362 o 6362” , en caso contrario es necesario llevar a cabo una reproyección.

```
COLONIAS_CDMX = gpd.read_file(Entradas +'COLONIAS.json')
print("Colonias CRS", COLONIAS_CDMX.crs)
print("Manzanas CRS", MERGE.crs)
```

```
Colonias CRS epsg:4326
Manzanas CRS epsg:4326
```

Los archivos estan en coordenadas geograficas, por lo que se reprojeta

```
MANZANA_METROS = MERGE.to_crs(6362)
COLONIAS_METROS = COLONIAS_CDMX.to_crs(6362)

print("Crs Manzanas", MANZANA_METROS.crs )
print("Crs Colonias", COLONIAS_METROS.crs )
```

```
Crs Manzanas epsg:6362
Crs Colonias epsg:6362
```

Buscamos en este punto identificar la intersección entre colonias y manzanas para asignar a cada manzana (base al criterio de área maxima) la clave de la colonia a la que pertenece.

```
INTERSECCION = gpd.overlay(COLONIAS_METROS,
                            MANZANA_METROS,
                            how = 'intersection')
```

Se calcula el valor de área para cada polígono intersectado

```
## Se calcula el area
INTERSECCION['area'] = INTERSECCION.geometry.area
```

Para el overlay se reordena la información del área de manera descendente y se eliminan los duplicados con base a la “CVEGEO” manteniendo únicamente el primer valor

```

INTERSECCION = (INTERSECCION.sort_values('area', ascending = False).
                 drop_duplicates(subset="CVEGEO", keep = 'first').
                 drop(['geometry','area'], axis = 1))

### Se eliminan columnas no necesarias
INTERSECCION_USE = INTERSECCION.drop(['ENT', 'CVEDT', 'NOMDT', 'DTTOLOC'], axis = 1)

```

En la base de colonias se identificaron caracteres especiales, por lo que se procede a remplazarlos, por su valor correspondiente.

```

Dic_Ca = {'À': 'Ñ'}
INTERSECCION_USE.replace(Dic_Ca, inplace=True, regex=True)
INTERSECCION_USE.columns = INTERSECCION_USE.columns.to_series().replace(Dic_Ca, regex=True)

INTERSECCION_USE.shape

```

(66353, 17)

Se une la información de overly de las Manzanas ya alineadas con colonias en la geometry de las manzanas para tener la base final. En la base final podemos observar la información a nivel: manzana, colonia y alcaldía, donde esta ultima se extraer en razon directa de la informacion contenida en la base de manzanas.

```

DATA_FINAL_USE = MANZA_CDMX.merge(INTERSECCION_USE,
                                   left_on = 'CVEGEO',
                                   right_on = 'CVEGEO',
                                   how = 'inner').rename({"CVEUT": "CVE_COL", "NOMUT": "NOM_COL", "NOMDT": "COL_NOMDT"})

DATA_FINAL_USE.shape

```

(66353, 18)

Se valida que cada manzana este asociada a cada una de las colonias
 Recorndado que la relación es una colonia a muchas manzanzanas
 “ ” Por lo que no deben existir manzanas repetididas”

$$M_{n-1} = f(C)$$

$$C \leftarrow M_{n-1}$$

Se valida que no existan manzanas repetidas

```
DATA_FINAL_USE.CVEGEO.value_counts()
```

La relacion de muchas manzanas a una colonia, se valida para cada clave de colonias
se repite tantas veces existan manzanas

```
DATA_FINAL_USE.CVE_COL.value_counts()
```

CVE_COL	
07-320	471
07-263	370
17-078	297
17-076	213
11-057	213
07-289	209
13-067	184
09-012	180
07-145	178
11-004	178
07-286	173
13-043	173
07-157	169
11-052	168
03-030	168
08-045	159
07-270	158
07-285	149
11-058	148
05-002	148
09-002	147
07-273	146
07-196	145
07-225	144
09-011	142
07-049	140
07-021	140
13-054	139
17-022	135
09-008	134
03-037	133
07-290	132
05-032	132
03-012	132
07-269	132
15-012	131
06-059	130
07-162	129
07-274	129
07-051	129
07-292	127
11-032	126
06-060	125
11-038	125
02-025	125
12-147	124
13-060	124
05-170	124
06-047	121
07-046	120
16-045	120
16-019	119

Aquí validamos como las claves de las manzanas son diferentes para una misma colonia y se entiende la relación, muchas manzanas a una colonia

```
DATA_FINAL_USE.query('CVE_COL == "07-320"').head(3)
```

	CVEGEO	geometry	CVE_COL	NOM_COL
51265	0900700014594012	POLYGON ((-99.05630 19.37395, -99.05642 19.373..., -99.05630 19.37395))	07-320	CHINAMP...
51303	0900700014645010	POLYGON ((-99.05146 19.37286, -99.05154 19.372..., -99.05146 19.37286))	07-320	CHINAMP...
51304	0900700014611041	POLYGON ((-99.05389 19.37262, -99.05333 19.374..., -99.05389 19.37262))	07-320	CHINAMP...

Reordenamos la información de la forma “Regional a local” es decir:

Alcaldía → Colonia → Manzana

1. **Alcaldía:** ENTIDAD, NOM_ENT, MUN, NOM_MUN, LOC, NOM_LOC,
2. **Colonia:** ID_COL, CVE_COL, NOM_COL,
3. **Manzana:** AGEB, MZA, CVEGEO,
4. **Dimensiones:** POBTOT, PSINDER, PROM_OCUP, P3YM_HLI, P_15A24_M
5. **Geometry**

Reordenamiento de la información

```
DATA_FIN_USE = DATA_FINAL_USE[['ENTIDAD', 'NOM_ENT', 'MUN', 'NOM_MUN', 'LOC', 'NOM_LOC',
                               'ID_COL', 'CVE_COL', 'NOM_COL', 'AGEB', 'MZA', 'CVEGEO',
                               'POBTOT', 'PSINDER', 'PROM_OCUP', 'P3YM_HLI', 'P_15A24_M',
                               'geometry']]
```

```
DATA_FIN_USE.head(3)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	ID_COL	CVE_COL
0	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
1	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1716	10-143
2	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1652	10-082

Hacemos un mapita porque nos gustan los mapitas

```
fig, ax = plt.subplots(figsize=(8, 8))
ax = DATA_FIN_USE.plot(ax = ax, column='P_15A24_M',
                       legend= False,
                       alpha=0.8,
```

```
    scheme='NaturalBreaks',
    cmap='copper',
    classification_kwds={'k':6})
ax.set(title='Población Masculina 15 a 24 años, Ciudad de México')

ax.set_axis_off()

plt.show()
```

Población Masculina 15 a 24 años, Ciudad de México



3.1.1 Análisis de Componentes Principales (PCA)

En esta sección se calcula el índice de *Concentración de desventajas* mediante la reducción de las dimensiones por componentes principales (PCA). Esto se hace a nivel **Alcaldías y Delegaciones**. La información a nivel alcaldía y delegación es un proceso de reagrupación y nuevos cálculos de los valores.

3.1.2 Nivel Colonias

Para este punto agrupamos los datos por nivel colonia para extraer el valor del índice por “PCA”

```
COLONIA_PCA = pd.DataFrame(DATA_FIN_USE.groupby(['CVE_COL']).agg({'POBTOT': 'sum',
                                                               'PSINDER': 'sum',
                                                               'PROM_OCUP': 'mean',
                                                               'P3YM_HLI': 'sum',
                                                               'P_15A24_M': 'sum'})).reset_index()
COLONIA_PCA.head(2)
```

	CVE_COL	POBTOT	PSINDER	PROM_OCUP	P3YM_HLI	P_15A24_M
0	02-001	1957.0	496.0	2.907000	7.0	138.0
1	02-002	3670.0	813.0	3.282381	43.0	219.0

Importante

Para que componentes principales tenga un alto rendimiento la información sdebe estar normalizada por **Z-SCORE (StandardScaler)**

$$Z = \frac{x - \mu}{\sigma}$$

Para calcular la componente principal, separamos nuestra base de datos con el fin de tener las dimensiones que contruyen el índice.

```
### Hacemos un copia por si necesitamos un proceso con la base original
PCA_COLONIAS = COLONIA_PCA.copy()

### Seleccion de las dimensiones con las que se calcula el indice de desventajas "PSINDER";

PCA_X = PCA_COLONIAS.drop(['CVE_COL','POBTOT'], axis = 1)
PCA_y = PCA_COLONIAS[['CVE_COL']]
```

Se normaliza la informacion por Z-Score, determinamos el número de componentes y aplicamos la función para calcular

```
### Se normaliza la informacion por Z-Score

S_TRANSF = StandardScaler()
PCA_X_SCALER = pd.DataFrame(S_TRANSF.fit_transform(PCA_X),
```

```
columns = PCA_X.columns)
```

```
### Se determina el número de componentes
```

```
PCA_N = PCA(n_components = 1)
```

```
PCA_COMPONENTE = PCA_N.fit_transform(PCA_X_SCALER)
```

Se calcula la varianza total por respecto al numero de componentes

```
### Se calcula la varianza total por respecto al numero de componentes
```

```
VARIANZA_TOTA = PCA_N.explained_variance_ratio_.sum() * 100
```

```
print("\n Total de la variancia explicada \n", round(VARIANZA_TOTA,3), "%")
```

Total de la variancia explicada

66.816 %

Con una componente (PC1) se explica 66.82 % de la varianza total, lo cual implica que mas de la mitad de la información e los datos puede encapsularse en ese componente principal.

finalmente se indexan los resultados a la base de datos como una nueva columna con clave DIS_COL = concentrated disadvantage component

```
### Pegamos los valores de PCA en la base de datos
```

```
PCA_COLONIAS['DIS_COL'] = PCA_COMPONENTE
```

```
PCA_COLONIAS.head(5)
```

	CVE_COL	POBTOT	PSINDER	PROM_OCUP	P3YM_HLI	P_15A24_M	DIS_COL
0	02-001	1957.0	496.0	2.907000	7.0	138.0	-1.121714
1	02-002	3670.0	813.0	3.282381	43.0	219.0	-0.533961
2	02-003	1830.0	318.0	2.853333	15.0	122.0	-1.193403
3	02-005	2330.0	497.0	2.734615	7.0	132.0	-1.175400
4	02-006	4306.0	855.0	3.022222	37.0	310.0	-0.459954

Importante

Lo anterior lo transformamos a una funcion para optimizar el proceso de trabajo. Funcion que podemos llamar despues

```

def CONC_DIS (TABLA, DIM_CLAVE, DIM_POBLA):

    PCA_TABLA = TABLA.copy()

    ### Seleccion de las dimensiones con las que se calcula el indice de desventajas "PSIN"

    PCA_X = PCA_TABLA.drop([DIM_CLAVE, DIM_POBLA], axis = 1)
    PCA_y = PCA_TABLA[[DIM_CLAVE]]

    ### Se normaliza la informacion por Z-Score

    S_TRANSF = StandardScaler()
    PCA_X_SCALER = pd.DataFrame( S_TRANSF.fit_transform(PCA_X),
                                  columns = PCA_X.columns)

    ### Se determina el número de componentes

    PCA_N = PCA(n_components = 1)
    PCA_COMPONENTE = PCA_N.fit_transform(PCA_X_SCALER)

    ### Se calcula la varianza total por respecto al numero de componentes

    VARIANZA_TOTA = PCA_N.explained_variance_ratio_.sum() * 100

    print("\n Total de la variancia explicada \n", round(VARIANZA_TOTA,3), "%")
    ### Se indexan los resultados a la base de datos como una nueva columna con clave DIS_

    PCA_TABLA['DISAD'] = PCA_COMPONENTE

    PCA_TABLA = PCA_TABLA[[DIM_CLAVE, 'DISAD']]

    return (PCA_TABLA)

```

Podemos usar la función creada y aplicarla en los datos para revalidar los resultados.

```

### Revalidación de información

DESVE_COL = CONC_DIS (COLONIA_PCA, 'CVE_COL','POBTOT')
DESVE_COL.head(2)

```

Total de la variancia explicada
66.816 %

	CVE_COL	DISAD
0	02-001	-1.121714
1	02-002	-0.533961

3.1.3 Nivel Alcaldías

El Agrupamiento de datos por nivel Alcaldia para “PCA”. Recordando que la clave de Alcaldia == Municipio la podemos observar de la forma:
DATA_FIN_USE.NOM_MUN.value_counts()

```
ALCALDIA_PCA = pd.DataFrame(DATA_FIN_USE.groupby(['MUN']).agg({'POBTOT': 'sum',
                                                               'PSINDER': 'sum',
                                                               'PROM_OCUP': 'mean',
                                                               'P3YM_HLI': 'sum',
                                                               'P_15A24_M': 'sum'})).reset_index()
ALCALDIA_PCA.head(2)
```

	MUN	POBTOT	PSINDER	PROM_OCUP	P3YM_HLI	P_15A24_M
0	002	432205.0	90063.0	3.005215	2139.0	27478.0
1	003	614447.0	160637.0	2.866638	6103.0	39714.0

Aplicamos la funcion creada con anterioridad

```
### Aplicando la función creada arriba
```

```
DESVE_ALCA = CONC_DIS (ALCALDIA_PCA, 'MUN', 'POBTOT')
DESVE_ALCA.head(2)
```

Total de la variancia explicada
74.241 %

	MUN	DISAD
0	002	-0.996294
1	003	-0.069455

En este punto, podemos unir toda la información a la tabla original y renombramos las columnas de desventajas en cada nivel

```
MERGE_DESVENTAJAS = DATA_FIN_USE.merge(DESVE_COL,
                                         left_on = 'CVE_COL',
                                         right_on = 'CVE_COL',
                                         how = 'inner').merge(DESVE_ALCA,
                                         left_on = 'MUN',
                                         right_on = 'MUN',
                                         how = 'inner').rename({"DISAD_"
                                         MERGE_DESVENTAJAS.head(2)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	ID_COL	CVE_COL
0	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
1	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085

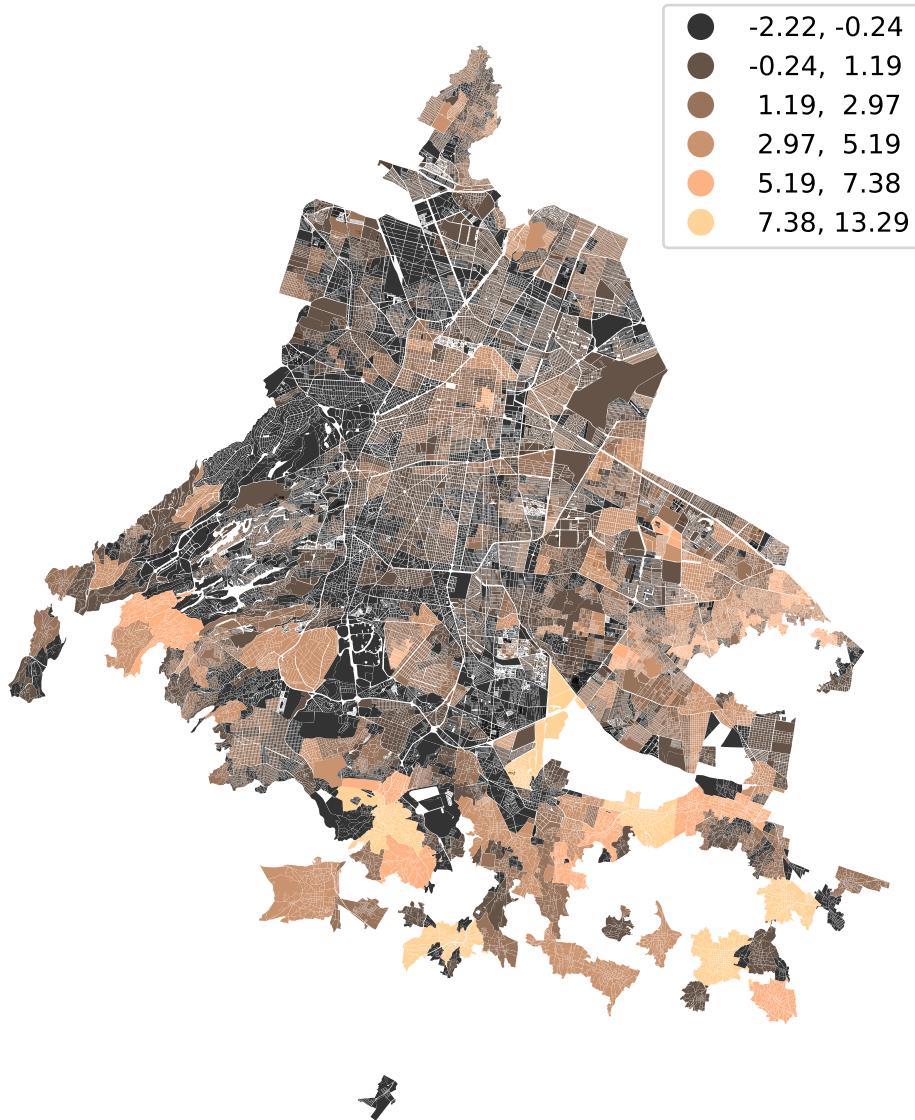
Hacemos un mapita nuevamente

```
fig, ax = plt.subplots(figsize=(8, 8))
ax = MERGE_DESVENTAJAS.plot(ax = ax, column='DIS_COL',
                             legend= True,
                             alpha=0.8,
                             scheme='NaturalBreaks',
                             cmap='copper',
                             classification_kwds={'k':6})
ax.set(title='Desventajas a nivel Colonia, Ciudad de México')

ax.set_axis_off()

plt.show()
```

Desventajas a nivel Colonia, Ciudad de México



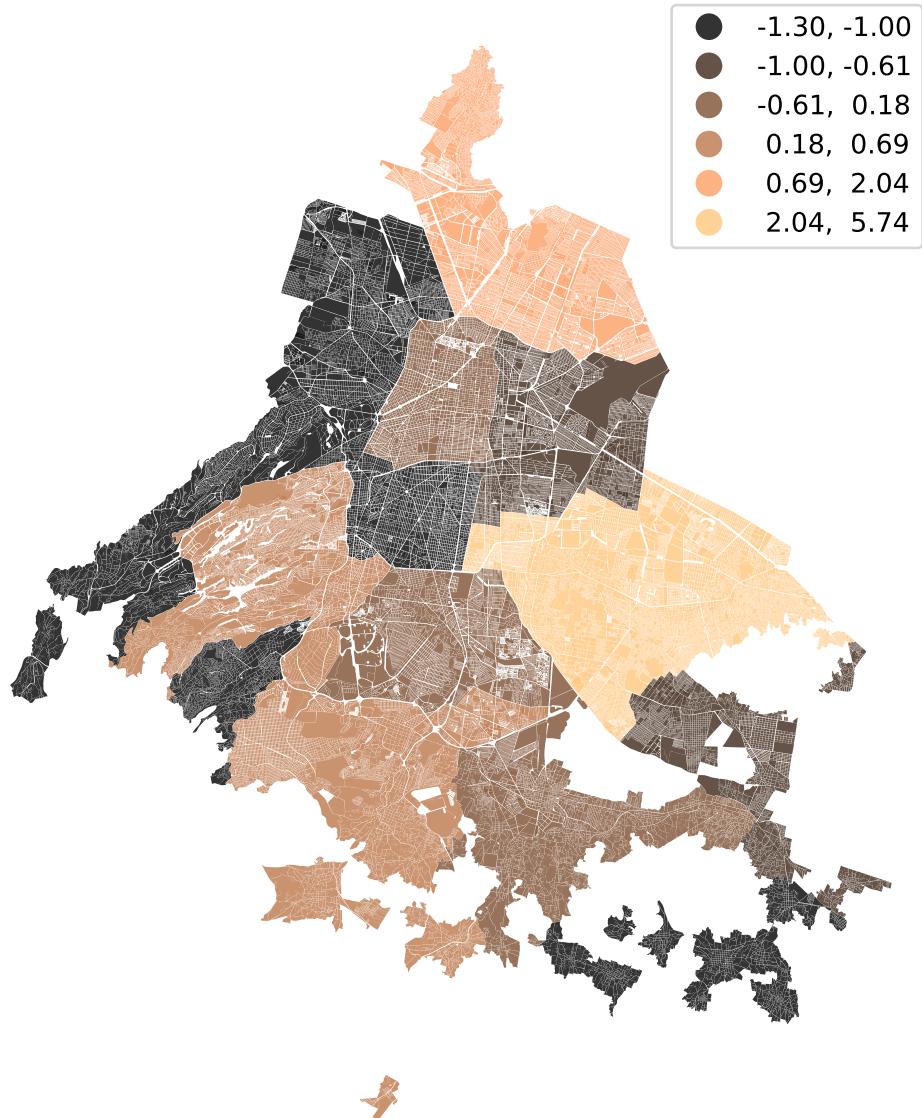
```
fig, ax = plt.subplots(figsize=(8, 8))
ax = MERGE_DESVENTAJAS.plot(ax = ax, column='DIS_MUN',
                            legend= True,
                            alpha=0.8,
                            scheme='NaturalBreaks',
```

```
cmap='copper',
classification_kwds={'k':6})
ax.set(title='Desventajas a nivel Alcaldías, Ciudad de México')

ax.set_axis_off()

plt.show()
```

Desventajas a nivel Alcaldías, Ciudad de México



4 Desorden Social y Físico

En esta sección se usarán las llamadas del 911 para crear los índices de desorden social y desorden físico comprendidas en dos semestres, para este caso se han usado las llamadas del primer y segundo semestre del 2021.

Las dimensiones que se han de usar son las siguientes

Table 4.1: Desorden Social

Clave	Descripción
Intoxicación pública con drogas	Administrativas-Drogados
Intoxicación pública con alcohol	Administrativas-Ebrios
Incidencia pública	NA
Orinar en público	NA
Denuncia de persona en riesgo	Denuncia-Persona en Riesgo
Disturbios públicos escándalo callejero	Disturbio-Escándalo
Disturbios públicos fiestas ruidosas	Disturbio-Escándalo
Tirar basura	Administrativas-Tirar Basura en Vía Pública

Table 4.2: Desorden Físico

Clave	Descripción
Vehículo abandonado con placas	Abandono-Vehículo
Graffiti	Administrativas-Grafitis
Fuga de agua potable	Derrame o Fuga-Agua Potable
Derrame de aguas residuales	Derrame o Fuga-Aguas Negras
Líneas eléctricas caídas	Servicios-Corto Circuito instalación o subestación eléctrica
Fugas de gas	Derrame o Fuga-Gas Natural

Se cargan las bases de datos correspondientes a los semestres del año 2021 y se concatenan a un solo dataframe para tener todos los meses

```
### Llamadas del 911
LL_911_s1 = pd.read_csv(Entradas + 'llamadas_911_2021_s1.csv', encoding = 'Latin-1')
```

```

LL_911_s2 = pd.read_csv(Entradas + 'llamadas_911_2021_s2.csv', encoding = 'Latin-1')

### Concatenaciones de semestres
Frame_C = [LL_911_s1, LL_911_s2]
All_Data = pd.concat(Frame_C)

```

Validamos que se encuentren los doce meses en la columna “mes_cierre”. En este punto surgen dudas, ¿se eliminan los folios repetidos o se mantienen?

```

### Validación de meses
All_Data.mes_cierre.value_counts()

```

	mes_cierre
Marzo	117554
Septiembre	113749
Abril	112352
Enero	111484
Mayo	110801
Octubre	109590
Agosto	104439
Diciembre	104276
Julio	103319
Febrero	102057
Junio	96030
Noviembre	95324

Con la validación de la información, seleccionamos únicamente de la dimensión “incidente_c4” las clases con las que trabajaremos. De igual manera, se simplifica la descripción de las clases, se resetea el index para comenzar desde cero, seleccionamos únicamente las dimensiones que usaremos para los demás procesos (folio, categoria_incidente_c4, incidente_c4, manzana, latitud, longitud) y renombramos los campos necesarios.

```

### Selección los valores de la dimensión como una lista
Lista = ["Drogados", "Ebrios" , "Persona en riesgo", "Escandalo", "Fiestas" , "Tirar basura",
         "Vehiculo", "Grafitis", "Agua potable", "Aguas negras", "Corto circuito instalacion"]

All_Data_filtered = All_Data[All_Data['incidente_c4'].isin(Lista)]

All_Data_filtered = All_Data_filtered.replace({'Fiestas':'FIESTA','Escandalo':'ESCANDALO',
                                                'Agua potable':'AGUA_P','Drogados':'DROGADO'})

```

```

'Grafitis':'GRAFITI', 'Persona en riesgo':'PE
'Aguas negras':'AGUA_N', 'Tirar basura en vi
'Corto circuito instalacion o subestacion el
'Fuga de gas natural':'FUGAS_N'}}).reset_index

All_Data_filtered = All_Data_filtered[["folio","categoria_incidente_c4","incidente_c4",
                                         "manzana","latitud","longitud"]].rename({"categoria
                                         "incidente"

```

4.1 El jardín de las bifurcaciones

En esta sección usaremos para calcular los valores de desorden social y físico, usaremos dos formas: las claves de las manzanas y las coordenadas geográficas a fin de comparar resultados.

4.1.1 Corrección columna clave de manzana

Inicialmente notamos de nuestra base de datos las columnas manzana, longitud y latitud. Para la primera bifurcación usaremos las claves de manzanas que se encuentran asignadas. Al hacer esto nos damos cuenta que necesitamos una limpieza de datos. Lo anterior por:

1. El tipo de dato es string esta en su forma científica,
2. Tenemos que transformar los string a números aplicando la función “to_numeric”
3. Se extiende el número de su forma científica a su forma natural
4. Los elementos con caracteres alfanuméricos se asignan como “NAN”.

En este punto podemos notar que ya tenemos el string científico en “float”, lo que tenemos que hacer es extraer el número sin notación científica, para eso aplicamos una función “lambda” que convierta el float en string sin notación científica y los “nan_text” sean remplazados por “NAN” verdaderos.

Importante

Recordar que las claves comienzan con un cero y que el total de caracteres para manzanas son 16.

CVEGEO = “0900700000000000”

```

## Transformación de string to float
Manza_Num = pd.DataFrame(pd.to_numeric(All_Data_filtered['manzana'], errors = 'coerce'))

```

```

## Se extiende el numero a su forma natural
Manza_Num = pd.DataFrame(Manza_Num['manzana']).apply(lambda x: '%.0f' %x).replace('nan', np.nan)

### Recordando que las claves comienzan con un cero y que para manzanas es un total de 16
### Asignamos el cero

Manza_Num['manzana'] = Manza_Num['manzana'].str.zfill(16)

Manza_Num.shape

```

(301784, 1)

Ya tenemos un dataframe con las claves en string y los indices como campo llave primaria con NAN verdaderos

Ahora vamos a hacer una union entre estos resultados y los el dataframe filtrado (All_Data_filtered) con base a su indice. Al ser un merge por indices, es recomendable que este este reiniciado (por lo que se entiende la necesidad de su reinicio en procesos anteriores).

De la union tendremos dos columnas con claves manzanas, las cuales las usaremos en una función para asignación de clave corregida

```

### Union entre tablas
USO_MANZAS = All_Data_filtered.merge(Manza_Num,
                                      left_index = True,
                                      right_index = True)
USO_MANZAS.head(3)

```

	folio	Categoría	Incidente	manzana_x	latitud	longitud	manzana_y
0	C5/210101/00770	Disturbio	FIESTA	9.007E+14	19.368088	-99.132832	09007000000000000000
1	C5/210101/01333	Disturbio	FIESTA	9.007E+14	19.354583	-99.061529	09007000000000000000
2	C5/210101/01973	Disturbio	FIESTA	9.003E+14	19.332512	-99.175247	09003000000000000000

La **asignacion de clave correcta:** se lleva a cabo considerando las combinaciones de resultados posibles entre las dos columnas de nombre manzanas (x,y). De lo anterior se obtienen las siguientes reglas de negocios:

1. Si manzana_x == manzana_y & manzana_y == manzana_y, entonces el valor sera manzana_y
2. Si manzana_x == manzana_y & manzana_y != manzana_y, entonces el valor sera manzana_x

3. Si ninguno de los dos anteriores se cumple se asigna NAN

Con base a las reglas se configura la función:

```
### Creamos una función
def manzan_a(C):
    if ((C.manzana_x == C.manzana_x) & (C.manzana_y == C.manzana_y)):
        return C.manzana_y
    elif ((C.manzana_x == C.manzana_x) & (C.manzana_y != C.manzana_y)):
        return C.manzana_x
    else:
        return np.nan
```

Se aplica la función y se eliminan las dimensiones no necesarias

```
### Aplicamos la función
USO_MANZAS['CVEGEO'] = USO_MANZAS.apply(manzan_a, axis = 1)

## Eliminamos las dimensiones no necesarias y tenemos la tabla de inicio

USO_MANZAS = USO_MANZAS.drop(['manzana_x', 'manzana_y'], axis = 1)

USO_MANZAS.head(3)
```

	folio	Categoría	Incidente	latitud	longitud	CVEGEO
0	C5/210101/00770	Disturbio	FIESTA	19.368088	-99.132832	0900700000000000
1	C5/210101/01333	Disturbio	FIESTA	19.354583	-99.061529	0900700000000000
2	C5/210101/01973	Disturbio	FIESTA	19.332512	-99.175247	0900300000000000

4.1.2 Caso por Mananzas y Manzanas

Para este caso, vamos a contar cuantos eventos de una determinada clase en el tipo de incidentes están inmersos en las manzanas. Para la forma con columna y claves manzanas, seleccionamos únicamente las columnas de trabajo, en este caso se eliminan longitud y latitud.

Para contabilizar las clases en la columna haremos uso de un dataframe dummy

```
### Se eliminan columnas no necesarias

USO_MANZAS_TRUE = USO_MANZAS.drop(['latitud','longitud'], axis = 1)
```

```
### Para contabilizar, notamos que la descripcion del incidente esta en fila,
### por lo que tenemos que crear un dataframe dummy
```

```
USO_MANZAS_TRUE_DMY = pd.get_dummies(USO_MANZAS_TRUE,
                                      columns=["Incidente"],
                                      prefix=["DM"])
```

```
USO_MANZAS_TRUE_DMY.head(3)
```

	folio	Categoría	CVEGEO	DM_AGUA_N	DM_AGUA_P	DM_BASURA_P
0	C5/210101/00770	Disturbio	0900700000000000	0	0	
1	C5/210101/01333	Disturbio	0900700000000000	0	0	
2	C5/210101/01973	Disturbio	0900300000000000	0	0	

Ahora que tenemos nuestro dataframe dummy podemos contar, para lograr eso, tenemos que agrupar y sumar los eventos

```
### Agrupación de dataframe Dummy
```

```
MANZANA_DUMMMY = pd.DataFrame(USO_MANZAS_TRUE_DMY.groupby(['CVEGEO']).agg({'DM_AGUA_N': 'sum',
                                                                           'DM_AGUA_P': 'sum',
                                                                           'DM_BASURA_P': 'sum',
                                                                           'DM_CORTO_ELE': 'sum',
                                                                           'DM_DROGADO': 'sum',
                                                                           'DM_EBRIOS': 'sum',
                                                                           'DM_ESCANDALO': 'sum',
                                                                           'DM_FIESTA': 'sum',
                                                                           'DM_FUGAS_N': 'sum',
                                                                           'DM_GRAFITI': 'sum',
                                                                           'DM_PERSONA_R': 'sum',
                                                                           'DM_VEHICULO': 'sum'})).reset_index()
```

```
MANZANA_DUMMMY.head(3)
```

	CVEGEO	DM_AGUA_N	DM_AGUA_P	DM_BASURA_P	DM_CORTO_ELE	DM
0	0900000000000000	202.0	1684.0	190.0		4
1	0900200000000000	4.0	102.0	8.0		0
2	090020001003A003	0.0	0.0	0.0		0

Ahora haremos otra union y seleccionaremos aquellos elementos para crear el Desorden Social y fisico. Se unen la tabla DATA_FIN_USE y MAN-

ZANA_DUMMMY. Esto se hace en esa tabla porque debemos recordar que las manzanas tienen su asignacion de clave de colonias y alcaldias con base al criterio de maxima area.

```
### Union de tablas
SOCIAL_FISICO = DATA_FIN_USE.merge(MANZANA_DUMMMY,
                                     left_on = 'CVEGEO',
                                     right_on = 'CVEGEO',
                                     how = 'left')
SOCIAL_FISICO.head(3)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	ID_COL	CVE_COL
0	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
1	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1716	10-143
2	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1652	10-082

4.1.3 Desorden Social y Fisico a nivel Colonia

De la misma forma que se ejecuto codigo pasado, lo que necesitamos es el agrupamiento de datos por nivel colonia para calcular la componente de Desorden Social por “PCA”

```
### Agrupación de datos
SOCIAL_PCA = pd.DataFrame(SOCIAL_FISICO.groupby(['CVE_COL']).agg({'POBTOT':'sum',
                                                               'DM_DROGADO': 'sum',
                                                               'DM_EBRIOS': 'sum',
                                                               'DM_ESCANDALOS': 'sum',
                                                               'DM_FIESTAS': 'sum',
                                                               'DM_BASURA_P': 'sum',
                                                               'DM_PERSONAS_R': 'sum'})).reindex()
SOCIAL_PCA.head(2)
```

	CVE_COL	POBTOT	DM_DROGADO	DM_EBRIOS	DM_ESCANDALOS	DM_FIESTAS	DM_BASURA_P
0	02-001	1957.0	0.0	0.0	0.0	0.0	0.0
1	02-002	3670.0	0.0	0.0	0.0	0.0	0.0

PCA DESORDEN SOCIAL

De igual manera aplicamos la funcion creada en pasos anteriores

```
DESORDEN_SOCIAL = CONC_DIS(SOCIAL_PCA, 'CVE_COL', 'POBTOT')
DESORDEN_SOCIAL.head(2)
```

Total de la variancia explicada
61.258 %

	CVE_COL	DISAD
0	02-001	-0.636744
1	02-002	-0.636744

Ahora, con las dimensioes con las que se calcula el desorden fisico, agrupamos los datos por nivel colonia para calcular la componente por “PCA”

```
### Agrupamiento de datos por nivel colonia para "PCA" DESORDEN FISICO
```

```
FISICO_PCA = pd.DataFrame(SOCIAL_FISICO.groupby(['CVE_COL']).agg({'POBTOT': 'sum',
'DM_VEHICULO': 'sum',
'DM_GRAFITI': 'sum',
'DM_AGUA_P': 'sum',
'DM_AGUA_N': 'sum',
'DM_CORTO_ELE': 'sum',
'DM_FUGAS_N': 'sum'})).re
FISICO_PCA.head(2)
```

	CVE_COL	POBTOT	DM_VEHICULO	DM_GRAFITI	DM_AGUA_P	DM_AGUA_N	DM
0	02-001	1957.0		0.0	0.0	0.0	0.0
1	02-002	3670.0		0.0	0.0	0.0	0.0

Nuevamente, aplicamos la funcion creada en pasos anteriores

```
#### Aplicamos la funcion creada en pasos anteriores
```

```
DESORDEN_FISICO = CONC_DIS(FISICO_PCA, 'CVE_COL', 'POBTOT')
DESORDEN_FISICO.head(2)
```

Total de la variancia explicada
35.943 %

CVE_COL	DISAD
0 02-001	-0.443488
1 02-002	-0.443488

4.1.4 Desorden Social y Fisico a nivel Alcaldia

El agrupamiento se hace ahora por alcaldia o Municipio para desorden social

```
#### Agrupamiento de datos por nivel Alcaldia para "PCA" DESORDEN SOCIAL
```

```
SOCIAL_ALPCA = pd.DataFrame(SOCIAL_FISICO.groupby(['MUN']).agg({'POBTOT': 'sum',
                                                               'DM_DROGADO': 'sum',
                                                               'DM_EBRIOS': 'sum',
                                                               'DM_ESCANDALOS': 'sum',
                                                               'DM_FIESTAS': 'sum',
                                                               'DM_BASURA_P': 'sum',
                                                               'DM_PERSONAS_R': 'sum'}).reset_index()

SOCIAL_ALCAL = CONC_DIS(SOCIAL_ALPCA, 'MUN', 'POBTOT')
SOCIAL_ALCAL.head(2)
```

Total de la variancia explicada
76.681 %

MUN	DISAD
0 002	-0.108228
1 003	1.059589

Aplicado el agrupamiento a municipio para desorden fisico

```
#### Agrupamiento de datos por nivel Alcaldia para "PCA" DESORDEN FISICO
```

```
FISICO_ALPCA = pd.DataFrame(SOCIAL_FISICO.groupby(['MUN']).agg({'POBTOT': 'sum',
                                                               'DM_VEHICULO': 'sum',
                                                               'DM_GRAFITI': 'sum',
                                                               'DM_AGUA_P': 'sum',
                                                               'DM_AGUA_N': 'sum',
                                                               'DM_CORTO_ELE': 'sum',
                                                               'DM_FUGAS_N': 'sum'}).reset_index()

FISICO_ALCAL = CONC_DIS(FISICO_ALPCA, 'MUN', 'POBTOT')
```

```
FISICO_ALCAL.head(2)
```

```
Total de la variancia explicada  
60.443 %
```

	MUN	DISAD
0	002	-1.863468
1	003	0.573604

Ahora crearemos la base de datos finales, para eso haremos una serie de uniones con toda la información a la tabla original y renombramos las columnas de lo necesario en cada nivel.

```
MERGE_SOCIAL = MERGE_DESVENTAJAS.merge(DESORDEN_SOCIAL,  
                                         left_on = 'CVE_COL',  
                                         right_on = 'CVE_COL',  
                                         how = 'inner').merge(SOCIAL_ALCAL,  
                                         left_on ='MUN',  
                                         right_on = 'MUN',  
                                         how = 'inner').rename({"DISAD_x": "DISAD_y": "DISAD")  
  
MERGE_FISICO = MERGE_SOCIAL.merge(DESORDEN_FISICO,  
                                         left_on = 'CVE_COL',  
                                         right_on = 'CVE_COL',  
                                         how = 'inner').merge(FISICO_ALCAL,  
                                         left_on ='MUN',  
                                         right_on = 'MUN',  
                                         how = 'inner').rename({"DISAD_x": "DISAD_y": "DISAD")
```

```
MERGE_FISICO.head(5)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	ID_COL	CVE_COL
0	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
1	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
2	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
3	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
4	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085

A este punto tendremos la base como la necesitamos, solo queda reordenar para un uso mas facil

```
DATA_SET_FINAL = MERGE_FISICO[['ENTIDAD', 'NOM_ENT', 'MUN', 'NOM_MUN', 'LOC', 'NOM_LOC',  
                               'CVE_COL', 'NOM_COL', 'AGEB', 'MZA', 'CVEGEO', 'POBTOT',  
                               'DIS_COL', 'DIS_MUN', 'SOCIAL_COL', 'SOCIAL_MUN', 'FISICO_C',  
                               'FISICO_MUN', 'geometry']]
```

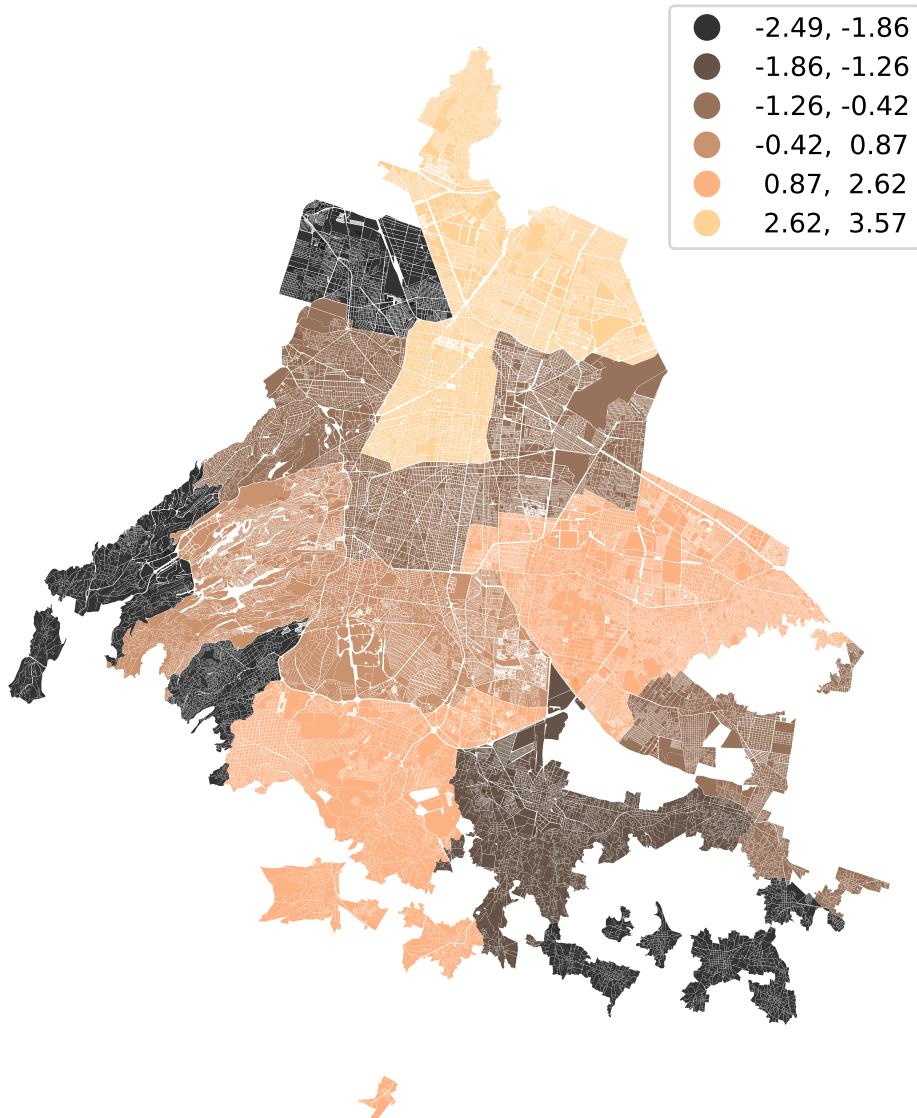
```
DATA_SET_FINAL.head(3)
```

	ENTIDAD	NOM_ENT	MUN	NOM_MUN	LOC	NOM_LOC	ID_COL	CVE_C
0	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
1	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085
2	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1656	10-085

Hacemos otro mapita, porque ya sabemos que nos gustan

```
fig, ax = plt.subplots(figsize=(8, 8))  
ax = DATA_SET_FINAL.plot(ax = ax, column='FISICO_MUN',  
                        legend= True,  
                        alpha=0.8,  
                        scheme='NaturalBreaks',  
                        cmap='copper',  
                        classification_kwds={'k':6})  
ax.set(title='Desorden Fisico a nivel Alcaldia, Ciudad de México')  
  
ax.set_axis_off()  
  
plt.show()
```

Desorden Fisico a nivel Alcaldia, Ciudad de México



Un ultimo paso es la validación de los resultados de esta forma con los obtenidos antes de las uniones a las manzanas, para eso agrupamos y calculamos la media de cada dimension

```
### validamos la informacion de las dimensiones creadas
```

```

DATA_SET_FINAL.groupby(['MUN']).agg({'DIS_COL': 'mean',
                                     'DIS_MUN': 'mean',
                                     'SOCIAL_COL': 'mean',
                                     'SOCIAL_MUN': 'mean',
                                     'FISICO_COL': 'mean',
                                     'FISICO_MUN': 'mean'
                                    }).reset_index()

```

	MUN	DIS_COL	DIS_MUN	SOCIAL_COL	SOCIAL_MUN	FISICO_COL	FISICO_MUN
0	002	-0.312325	-0.996294	0.196748	-0.108228	-0.156930	-1.863468
1	003	-0.006193	-0.069455	0.311090	1.059589	0.293600	0.573604
2	004	1.106869	-1.304359	0.098220	-2.826147	-0.103746	-2.381901
3	005	0.565336	2.040898	0.184904	2.975016	0.329208	3.574120
4	006	1.201381	-0.650310	0.903878	0.204031	0.327934	-0.687370
5	007	1.719481	5.736332	0.224290	4.864047	0.138066	2.623189
6	008	0.868157	-1.099762	0.329092	-2.475364	0.275849	-1.962040
7	009	4.909275	-1.163220	-0.315863	-3.290602	-0.179439	-2.487223
8	010	0.126637	0.533438	0.098015	1.033927	0.190218	0.873328
9	011	1.633560	-0.609511	0.438813	-1.867119	0.587161	-0.548826
10	012	0.829067	0.688221	0.058954	-0.280469	0.486348	2.281060
11	013	2.961643	0.183677	0.006548	-2.229841	0.007994	-1.260814
12	014	-0.000506	-1.302296	0.735603	-0.326258	0.350952	-0.580746
13	015	1.070981	-0.151071	1.250985	1.979941	0.810097	3.063466
14	016	-0.491578	-1.227027	0.644503	1.155803	0.207842	-0.424593
15	017	0.548961	-0.609259	0.788632	0.131673	0.413103	-0.791786

FISICO_ALCAL

	MUN	DISAD
0	002	-1.863468
1	003	0.573604
2	004	-2.381901
3	005	3.574120
4	006	-0.687370
5	007	2.623189
6	008	-1.962040
7	009	-2.487223
8	010	0.873328
9	011	-0.548826
10	012	2.281060
11	013	-1.260814
12	014	-0.580746
13	015	3.063466
14	016	-0.424593
15	017	-0.791786

4.1.5 Caso por Mananzas y Geometria Punto

5 Methods

6 Summary

In summary, this book has no content whatsoever.

References