

# **BOOK\_PAPER**

Jane Doe

2/9/23

# Table of contents

<b>Preface</b>	<b>3</b>
<b>1 Introduction</b>	<b>4</b>
<b>2 Data Acquisition and Preprocessing</b>	<b>5</b>
<b>3 Preprocessing</b>	<b>6</b>
3.1 Concentración de desventajas . . . . .	6
3.2 Análisis de Componentes Principales (PCA) . . . . .	16
3.2.1 Nivel Colonias . . . . .	17
3.2.2 Nivel Alcaldías . . . . .	18
3.3 Índice del Desorden Social y Físico . . . . .	22
3.3.1 Desorden a nivel Alcaldía . . . . .	29
3.3.2 Desorden a nivel Colonia . . . . .	31
3.4 Resultados Finales . . . . .	38
<b>4 Methods</b>	<b>39</b>
<b>5 Summary</b>	<b>40</b>
<b>References</b>	<b>41</b>

# 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**

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

```
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']]

Values = ['Total de la entidad','Total del municipio',
          'Total de la localidad urbana', 'Total AGEB urbana']

CENSO_2020 = CENSO_2020.query("NOM_LOC != @Values")

CENSO_2020.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	14
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 símbolos o caracteres especiales(\*, N/D, 999999, etc), por lo que es necesario reemplazarlos por valores NAN.

```
CENSO_2020 = CENSO_2020.replace({'999999999': np.nan,
                                    '99999999': np.nan,
                                    '*': np.nan,
                                    'N/D': np.nan})

DIM_NUM = CENSO_2020.iloc[:, -6:].columns.tolist()
DIM_TEXT = CENSO_2020.iloc[:, :8].columns.tolist()

CENSO_2020[DIM_NUM] = CENSO_2020[DIM_NUM].astype('float')
CENSO_2020[DIM_TEXT] = CENSO_2020[DIM_TEXT].astype(str)
```

De igual manera la información referida a las columnas de “Entidad, Municipio, Localidad, AGEB y Manzana”, no presentan 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['ENTIDAD'] = CENSO_2020['ENTIDAD'].str.zfill(2)
CENSO_2020['MUN'] = CENSO_2020['MUN'].str.zfill(3)
CENSO_2020['LOC'] = CENSO_2020['LOC'].str.zfill(4)
CENSO_2020['AGEB'] = CENSO_2020['AGEB'].str.zfill(4)
CENSO_2020['MZA'] = CENSO_2020['MZA'].str.zfill(3)

CENSO_2020['CVEGEO'] = CENSO_2020[['ENTIDAD', 'MUN',
                                         'LOC', 'AGEB', 'MZA']].agg(''.join, axis=1)

## Cálculo de Población masculina de 15 a 24 años
CENSO_2020['P_15A24_M'] = CENSO_2020[['P_15A17_M',
                                         "P_18A24_M"]].sum(axis = 1,
                                                               min_count = 1)

## Eliminación de dimensiones
CENSO_2020 = CENSO_2020.drop(['P_15A17_M', 'P_18A24_M'], axis = 1)

CENSO_2020.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..., -99.22982 19.35939, -99.23006 19.359..., -99.22982 19.35939)
1	0901000012269024	POLYGON ((-99.21840 19.36539, -99.21848 19.365..., -99.21840 19.36539, -99.21848 19.365..., -99.21840 19.36539))

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" #### Mantener Merge con cualquier campo
MERGE = MANZA_CDMX.merge( CENSO_2020,
                          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..., -99.22982 19.35939, -99.23006 19.359..., -99.22982 19.35939))	09	Ciudad de México
1	0901000012269024	POLYGON ((-99.21840 19.36539, -99.21848 19.365..., -99.21840 19.36539, -99.21848 19.365..., -99.21840 19.36539))	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 de P\_15A24\_M para ver su distribución espacial.

```
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

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

print("Crs Manzanas", MERGE.crs )
print("Crs Colonias", COLONIAS_CDMX.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_CDMX,
                           MERGE,
                           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 = INTERSECCION.drop(['ENT', 'CVEDT', 'NOMDT', 'DTTOLOC'], axis = 1) ##### Se m
```

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.replace(Dic_Ca, inplace=True, regex=True)
INTERSECCION.columns = INTERSECCION.columns.to_series().replace(Dic_Ca, regex=True)

INTERSECCION.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.

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

$$\text{Alcaldia} \rightarrow \text{Colonia} \rightarrow \text{Manzana}$$

1. **Alcaldia:** 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**

```

DATA_FINAL_USE = MANZA_CDMX.merge(INTERSECCION,
                                    left_on = 'CVEGEO',
                                    right_on = 'CVEGEO',
                                    how = 'inner').rename({"CVEUT": "CVE_COL", "NOMUT": "NOM_MUN",
                                                          "NOMUT": "MUN", "NOMUT": "LOC", "NOMUT": "NOM_LOC",
                                                          "NOMUT": "ENTIDAD", "NOMUT": "NOM_ENT"}, axis = 1)

DATA_FIN_USE = DATA_FINAL_USE[['ENTIDAD', 'NOM_ENT', 'MUN', 'NOM_MUN', 'LOC', 'NOM_LOC',
                               'CVEGEO', 'AGEB', 'MZA']]

```

```

        'ID_COL', 'CVE_COL', 'NOM_COL', 'AGEB', 'MZA', 'CVEGEO',
        'PSINDER', 'PROM_OCUP', 'P3YM_HLI', 'P_15A24_M',
        'geometry']]]

print("Forma de los datos: ", DATA_FINAL_USE.shape)

DATA_FIN_USE.head(3)

```

Forma de los datos: (66353, 18)

	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	1716	10-143
2	09	Ciudad de México	010	Álvaro Obregón	0001	Álvaro Obregón	1652	10-082

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

Aqui validamos como las claves de las manzanas son diferentes para una misma colonia y se entiende la relacion, 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...)	07-320	CHINAMP...
51303	0900700014645010	POLYGON ((-99.05146 19.37286, -99.05154 19.372...)	07-320	CHINAMP...
51304	0900700014611041	POLYGON ((-99.05389 19.37262, -99.05333 19.374...)	07-320	CHINAMP...

## 3.2 Análisis de Componentes Principales (PCA)

En esta sección se cálcula el indice 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 alcaldia y delegacion es un proceso de reagrupacion y nuevos calculos de los valores.

Importante

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

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

Para calcular la componente principal se ha creado una funcion que encapsula los siguientes procesos: 1. Seleccion de las dimensiones con las que se calcula el indice 2. Se normaliza la informacion por Z-Score 3. Se determina el número de componentes a reducir la informacion 4. Se calcula la varianza total por respecto al numero de componentes 5. Se indexan los resultados a la base de datos como una nueva columna con un nombre de clave

```
def CONC_DIS (TABLA, DIM_CLAVE, PCA_NAME):

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

    PCA_X = TABLA.drop([DIM_CLAVE], axis = 1)
    PCA_y = 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 por la componente \n", round(VARIANZA_TOTA,3))

### Se indexan los resultados a la base de datos como una nueva columna con clave DIS_

TABLA[PCA_NAME] = PCA_COMPONENTE

TABLA = TABLA[[DIM_CLAVE, PCA_NAME]]

return (TABLA)

```

### 3.2.1 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({'PSINDER': 'sum',
                                                               'PROM_OCUP': 'mean',
                                                               'P3YM_HLI': 'sum',
                                                               'P_15A24_M': 'sum'})).reset_index()

COLONIA_PCA.head(2)

```

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

Usamos la función creada en los datos para obtener los resultados.

```

#### Revalidación de información
DESVE_COL = CONC_DIS (COLONIA_PCA, 'CVE_COL', 'DIS_COL')
DESVE_COL.head(2)

```

Total de la variancia explicada por la componente  
66.816 %

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

### 3.2.2 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({'PSINDER': 'sum',
                                                               'PROM_OCUP': 'mean',
                                                               'P3YM_HLI': 'sum',
                                                               'P_15A24_M': 'sum'}).reset_
ALCALDIA_PCA.head(2)

```

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

Aplicamos la funcion creada para obtener los resultados a nivel alcaldia

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

```

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

```

Total de la variancia explicada por la componente  
74.241 %

	MUN	DIS_MUN
0	002	-0.996294
1	003	-0.069455

En este punto, podemos unir toda la información a la tabla original y hacer unos mapas para ver como se vería la distribución a nivel colonia y alcaldía

```
DATA_FIN_USE = DATA_FIN_USE.merge(DESVE_COL, ##### NO NECESITO MAS COPIAS :(
    left_on = 'CVE_COL',
    right_on = 'CVE_COL',
    how = 'inner').merge(DESVE_ALCA,
        left_on ='MUN' ,
        right_on = 'MUN' ,
        how = 'inner')

DATA_FIN_USE.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

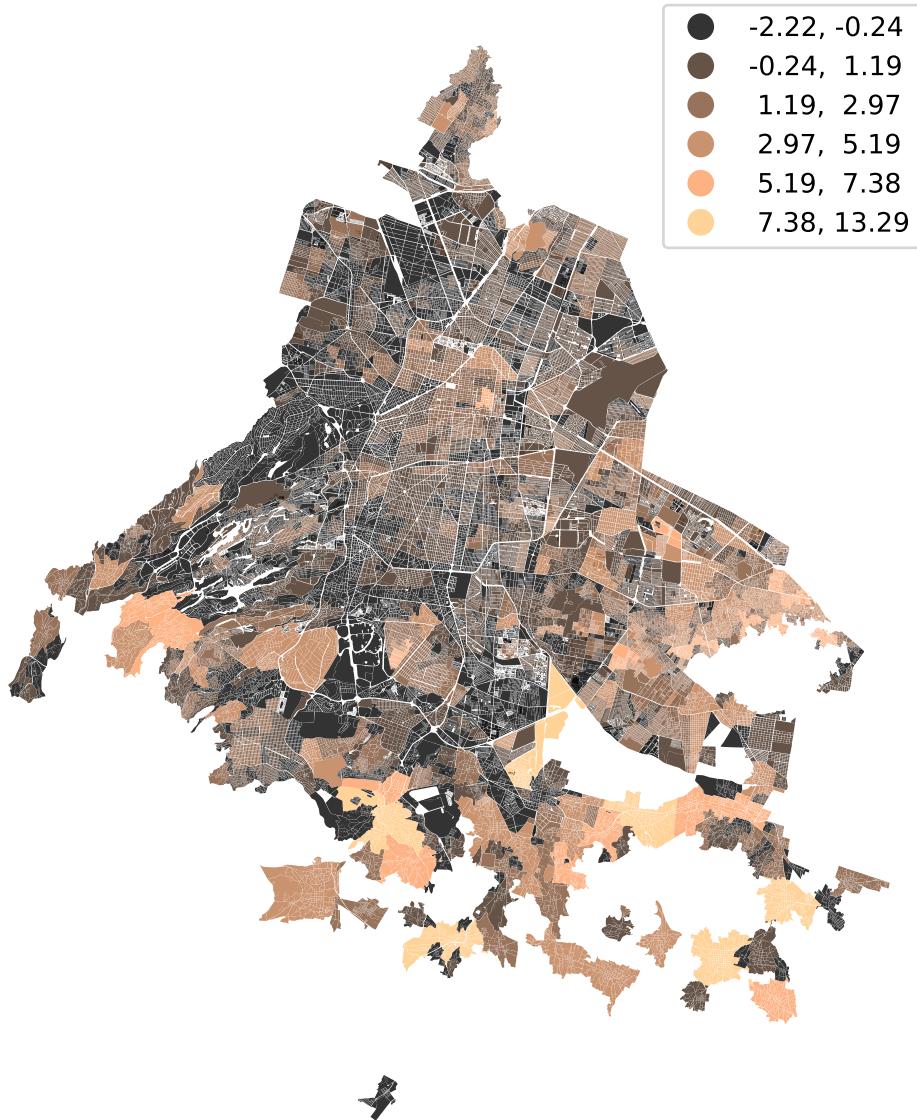
Hacemos unos mapas de Concentración de desventajas a nivel Colonia y Alcaldía

```
fig, ax = plt.subplots(figsize=(8, 8))
ax = DATA_FIN_USE.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



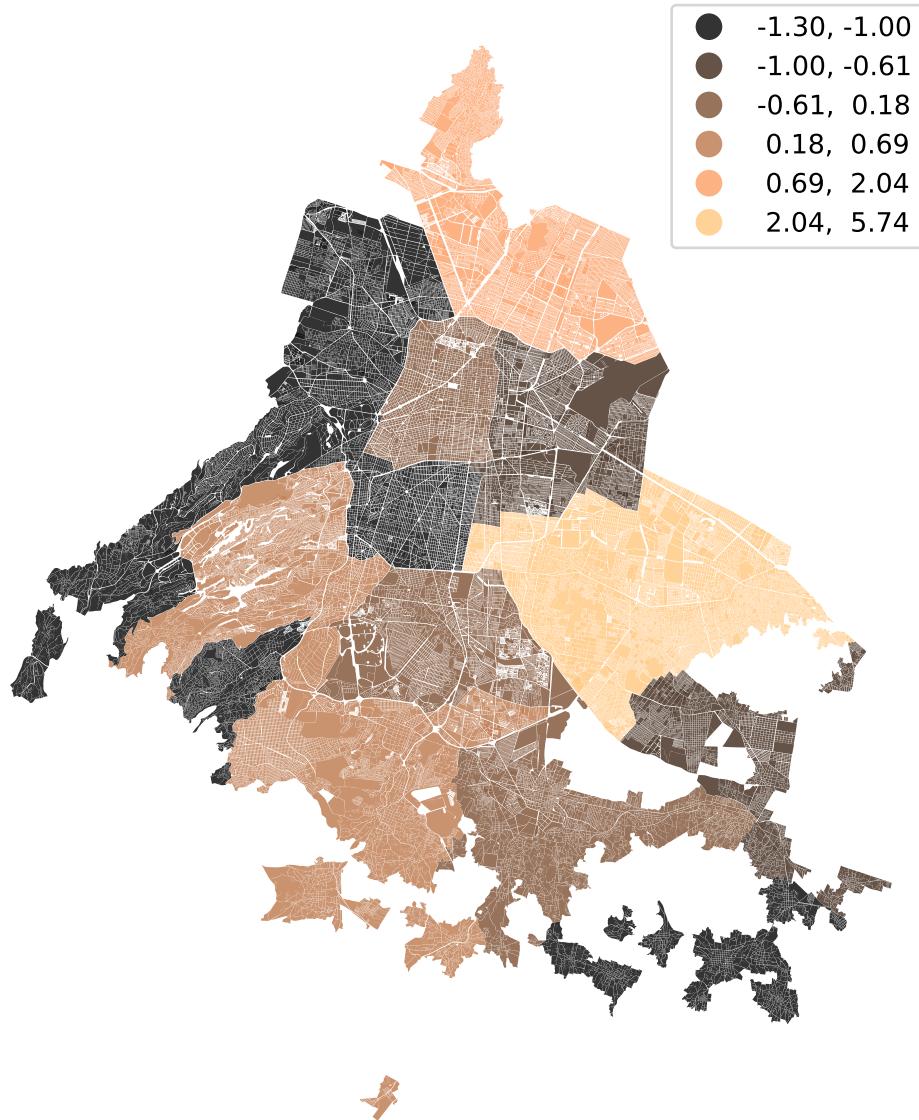
```
fig, ax = plt.subplots(figsize=(8, 8))
ax = DATA_FIN_USE.plot(ax = ax, column='DIS_MUN',
                      legend= True,
                      alpha=0.8,
                      scheme='NaturalBreaks',
```

```
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



### 3.3 Índice del 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 3.2: 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 3.3: 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 en vialidad", "Vehiculo", "Grafitis", "Agua potable", "Aguas negras", "Corto circuito instalacion electrica", "Incendio"]
All_Data = All_Data[All_Data['incidente_c4'].isin(Lista)]

All_Data = All_Data.replace({'Fiestas':'FIESTA','Escandalo':'ESCANDALO','Ebrios':'EBRIO',
                            'Agua potable':'AGUA_P','Drogados':'DROGADO',
                            'Grafitis':'GRAFITI','Persona en riesgo':'PE_RIESGO',
                            'Aguas negras':'AGUA_N', 'Tirar basura en vialidad':'TIRAR_BASURA',
                            'Corto circuito instalacion o subestacion electrica':'CORTO_CIRCUITO',
                            'Fuga de gas natural':'FUGAS_N'}).reset_index()

All_Data = All_Data[["folio","categoria_incidente_c4","incidente_c4",
                     "manzana","latitud","longitud"]].rename({"categoria_incidente_c4": "Categoría_incidente_c4",
                                                               "incidente_c4": "Incidente_c4"})


All_Data.head(2)
```

	folio	Categoría	Incidente	manzana	latitud	longitud
0	C5/210101/00770	Disturbio	FIESTA	9.007E+14	19.368088	-99.132832
1	C5/210101/01333	Disturbio	FIESTA	9.007E+14	19.354583	-99.061529

En esta sección para calcular los valores de desorden social y físico, nos centraremos en usar la información de la columna “manzana” de las llamadas para unir con la base donde de desventajas y tener la información a nivel Alcaldía.

### Corrección de la columna clave de manzana

Las claves de manzanas que se encuentran asignadas necesitan un procedimiento de corrección y limpieza.

1. Inicialmente 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['manzana'], errors = 'coerce'))

## Se extiende el número 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.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	0900700000000000
1	C5/210101/01333	Disturbio	FIESTA	9.007E+14	19.354583	-99.061529	0900700000000000
2	C5/210101/01973	Disturbio	FIESTA	9.003E+14	19.332512	-99.175247	0900300000000000

**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\_x & manzana\_y == manzana\_y, entonces el valor sera manzana\_y
2. Si manzana\_x == manzana\_x & 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	09007000000000000000
1	C5/210101/01333	Disturbio	FIESTA	19.354583	-99.061529	09007000000000000000
2	C5/210101/01973	Disturbio	FIESTA	19.332512	-99.175247	09003000000000000000

Ahora vamos a contar los eventos de una determinada clase en el tipo de incidentes estan 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
### Para contabilizar, notamos que la descripción del incidente esta en fila,
### por lo que tenemos que crear un dataframe dummy
```

```
USO_MANZAS_DMY = pd.get_dummies(USO_MANZAS.drop(['latitud','longitud'], axis = 1),
                                 columns=["Incidente"],
                                 prefix=["DM"])
```

```
USO_MANZAS_DMY.head(3)
```

	folio	Categoría	CVEGEO	DM_AGUA_N	DM_AGUA_P	DM_BASURA
0	C5/210101/00770	Disturbio	09007000000000000000	0	0	
1	C5/210101/01333	Disturbio	09007000000000000000	0	0	
2	C5/210101/01973	Disturbio	09003000000000000000	0	0	

Ahora que tenemos nuestro dataframe dummy podemos contar los elementos de cada clase en las manzanas, para eso agrupamos y sumamos

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

```
MANZANA_DUMMMY = pd.DataFrame(USO_MANZAS_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_ESCANDALOS': 'sum',
'DM_FIESTAS': 'sum',
'DM_FUGAS_N': 'sum',
'DM_GRAFITIS': 'sum',
'DM_PERSONAS_R': 'sum',
'DM_VEHICULOS': 'sum'
}).reset_index())
```

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

	CVEGEO	DM_AGUA_N	DM_AGUA_P	DM_BASURA_P	DM_CORTO_ELE	DM
0	09000000000000000000	202.0	1684.0	190.0		4
1	09002000000000000000	4.0	102.0	8.0		0
2	090020001003A003	0.0	0.0	0.0		0

Se une la tabla y seleccionamos aquellos elementos para crear el Desorden Social y fisico. Se unen la tabla DATA\_FIN\_USE y MANZANA\_DUMMMY. Esto se hace en esa tabla porque debemos recordar que las manzanas tienen su asignacion de clave alcaldias.

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

SOCIAL_FISICO.head(2)
```

	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

### 3.3.1 Desorden a nivel Alcaldia

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

#### Desorden Social

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

```
SOCIAL_ALPCA = pd.DataFrame(SOCIAL_FISICO.groupby(['MUN']).agg({'DM_DROGADO': 'sum',
                                                               'DM_EBRIOS': 'sum',
                                                               'DM_ESCANDALO': 'sum',
                                                               'DM_FIESTA': 'sum',
                                                               'DM_BASURA_P': 'sum',
                                                               'DM_PERSONA_R': 'sum'})).reset_index()

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

Total de la variancia explicada por la componente  
76.681 %

MUN	SOCIAL_MUN
0	002 -0.108228
1	003 1.059589

#### Desorden Físico

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

```
FISICO_ALPCA = pd.DataFrame(SOCIAL_FISICO.groupby(['MUN']).agg({'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', 'FISICO_MUN')
FISICO_ALCAL.head(2)
```

Total de la variancia explicada por la componente  
60.443 %

	MUN	FISICO_MUN
0	002	-1.863468
1	003	0.573604

Ahora se unen los resultados de la concentración de desventajas con los resultados del desorden social y físico

```
#### Union Desventajas + Desorden Social y Fisico
```

```
DESVE_ALCA = DESVE_ALCA.merge(SOCIAL_ALCAL,
                                left_on = 'MUN',
                                right_on = 'MUN',
                                how = 'inner').merge(FISICO_ALCAL,
                                left_on = 'MUN',
                                right_on = 'MUN',
                                how = 'inner')

DESVE_ALCA
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN
0	002	-0.996294	-0.108228	-1.863468
1	003	-0.069455	1.059589	0.573604
2	004	-1.304359	-2.826147	-2.381901
3	005	2.040898	2.975016	3.574120
4	006	-0.650310	0.204031	-0.687370
5	007	5.736332	4.864047	2.623189
6	008	-1.099762	-2.475364	-1.962040
7	009	-1.163220	-3.290602	-2.487223
8	010	0.533438	1.033927	0.873328
9	011	-0.609511	-1.867119	-0.548826
10	012	0.688221	-0.280469	2.281060
11	013	0.183677	-2.229841	-1.260814
12	014	-1.302296	-0.326258	-0.580746
13	015	-0.151071	1.979941	3.063466
14	016	-1.227027	1.155803	-0.424593
15	017	-0.609259	0.131673	-0.791786

### 3.3.2 Desorden a nivel Colonia

Para este caso y al contar con la geometria de las colonias previamente usada, contaremos el total de eventos de cada una de las clases en las colonias. Para eso necesitamos inicialmente que nuestra base de datos sea espacializada, es decir que cuente con una columna de geometria espacial

Comenzamos usando la tabla que habiamos creado con anterioridad, aquella donde se hicieron las correcciones a la clave manzanada (USO\_MANZAS). De la tabla se crean las geometrias las cuales necesitan tener definido un sistema de proyección.

Como notamos en los datos, el sistema de coordendas se encuentra en longitud / latitud, por lo que se asigna una proyeccion geografica wgs84 con epsg 4326

```
USO_MANZAS = gpd.GeoDataFrame(USO_MANZAS,
                                geometry = gpd.points_from_xy(USO_MANZAS.longitud,
                                                               USO_MANZAS.latitud), crs=4326).
USO_MANZAS.head(2)
```

	folio	Categoría	Incidente	CVEGEO	geometry
0	C5/210101/00770	Disturbio	FIESTA	0900700000000000	POINT (2800344.195 821939.006)
1	C5/210101/01333	Disturbio	FIESTA	0900700000000000	POINT (2807842.900 820599.774)

Validamos que ambos se encuentren en el mismo sistema de coordenadas (por si algo no tenemos bien, mejor validar)

```
print (USO_MANZAS.crs)
print (COLONIAS_CDMX.crs)
```

```
epsg:6362
epsg:6362
```

Ahora se van a contabilizar el número de clases de geometria punto dentro de las colonias de geometria poligonal, para que eso suceda, debemos hacer un join geoespacial.

```
### Hacemos un join geoespacial de los puntos que se encuentran contenidos dentro del polígonos
MANZANA_911 = gpd.sjoin(COLONIAS_CDMX,
                         USO_MANZAS[['folio', 'Categoría',
                                      'Incidente',
                                      'geometry']],
```

```

        how = "left",
        op = 'contains').rename({'CVEUT':'CVE_COL'}, axis = 1)
MANZANA_911.head(2)

```

	ENT	CVEDT	NOMDT	DTTOLOC	CVE_COL	NOMUT	ID	geometry
0	9	2	AZCAPOTZALCO	05	02-001	AGUILERA	1	POLYGON ((279
0	9	2	AZCAPOTZALCO	05	02-001	AGUILERA	1	POLYGON ((279

Retomamos parte de los procesos anteriores y aplicamos un dataframe dummies para contabilizar en este caso a nivel manzana

```

## Creación del dataframe Dummy
MANZAS_911_DMY = pd.get_dummies(MANZANA_911,
                                  columns = ["Incidente"],
                                  prefix = ["DM"])

### Ahora que tenemos nuestro dummy podemos contar, para esto agrupamos

MANZANA_911_DUMMMY = pd.DataFrame(MANZAS_911_DMY.groupby(['CVE_COL']).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_911_DUMMMY.head(5)

```

	CVE_COL	DM_AGUA_N	DM_AGUA_P	DM_BASURA_P	DM_CORTO_ELE	DM_DROGO
0	02-001	0	4	1	0	
1	02-002	4	7	0	0	
2	02-003	0	3	0	0	
3	02-005	0	0	0	0	
4	02-006	1	1	0	0	

Una vez realizado lo anterior, podemos validar que las manzanas no se repiten

```
MANZANA_911_DUMMMY.CVE_COL.value_counts()
```

---

CVE\_COL

---

02-001 1

10-250 1

11-018 1

11-017 1

11-016 1

11-015 1

11-014 1

11-013 1

11-012 1

11-011 1

11-010 1

11-009 1

11-008 1

11-007 1

11-006 1

11-005 1

11-004 1

11-003 1

11-001 1

10-259 1

10-258 1

10-257 1

10-256 1

10-255 1

10-254 1

10-253 1

10-252 1

11-019 1

11-020 1

11-021 1

11-036 1

11-048 1

11-047 1

11-045 1

11-044 1

11-043 1

11-042 1

11-041 1

11-040 1

11-039 1

11-038 1

11-037 1

11-035 1

11-022 1

11-034 1

11-033 1

11-032 1

11-031 1

11-030 1

11-029 1

11-028 1

11-027 1

## Desorden Social

Para construirlo repetimos los procesos anteriores para agrupacion y aplicacion de la función.

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

SOC_POINT_COL = pd.DataFrame(MANZANA_911_DUMMMY.groupby(['CVE_COL']).agg({'DM_DROGADO': 'sum',
                                                                           'DM_EBRIOS': 'sum',
                                                                           'DM_ESCANDALOS': 'sum',
                                                                           'DM_FIESTAS': 'sum',
                                                                           'DM_BASURA_P': 'sum',
                                                                           'DM_PERSONAS_R': 'sum'}))

SOCIAL_PNT_COL = CONC_DIS(SOC_POINT_COL, 'CVE_COL', 'SOCIAL_COL')

SOCIAL_PNT_COL.head(2)
```

Total de la variancia explicada por la componente  
56.94 %

CVE_COL	SOCIAL_COL
0 02-001	-0.850950
1 02-002	0.679359

## Desorden fisico

Para construirlo repetimos los procesos anteriores para agrupacion y aplicacion de la función.

```
### Desorden Fisico a nivel colonia

FISI_POINT_COL = pd.DataFrame(MANZANA_911_DUMMMY.groupby(['CVE_COL']).agg({'DM_VEHICULO': 'sum',
                                                                           'DM_GRAFITI': 'sum',
                                                                           'DM_AGUA_P': 'sum',
                                                                           'DM_AGUA_N': 'sum',
                                                                           'DM_CORTO_ELE': 'sum',
                                                                           'DM_FUGAS_N': 'sum'}))

FISICO_PNT_COL = CONC_DIS(FISI_POINT_COL, 'CVE_COL', 'FISICO_COL')

FISICO_PNT_COL.head(2)
```

Total de la variancia explicada por la componente  
31.445 %

	CVE_COL	FISICO_COL
0	02-001	-0.978118
1	02-002	-0.159875

Ahora se unen los resultados de la concentración de desventajas con los resultados del desorden social y físico

```
#### Union Desventajas + Desorden Social y Fisico

DESVE_COL = DESVE_COL.merge(SOCIAL_PNT_COL,
                            left_on = 'CVE_COL',
                            right_on = 'CVE_COL',
                            how = 'inner').merge(FISICO_PNT_COL,
                            left_on = 'CVE_COL',
                            right_on = 'CVE_COL',
                            how = 'inner')

DESVE_COL
```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL
0	02-001	-1.121714	-0.850950	-0.978118
1	02-002	-0.533961	0.679359	-0.159875
2	02-003	-1.193403	-1.380046	-1.018299
3	02-005	-1.175400	-0.935409	-0.580366
4	02-006	-0.459954	1.067722	-0.455032
5	02-007	-0.989835	0.485621	-0.566413
6	02-008	0.067596	2.743257	0.284400
7	02-009	-0.507203	0.281993	-0.486051
8	02-010	-0.868412	-0.855373	-0.968222
9	02-011	-1.386359	-1.154174	-0.368234
10	02-012	-1.388461	-1.818890	-1.224151
11	02-013	-1.321711	-1.269423	-0.063019
12	02-014	-1.175255	-1.067415	-0.580366
13	02-015	0.306681	0.495794	-0.630599
14	02-016	-0.524159	-0.983154	0.579116
15	02-017	-1.543894	-1.483580	-1.183970
16	02-018	0.263918	1.852063	0.274505
17	02-019	-1.143008	-1.443838	-1.183970
18	02-020	-1.568669	-1.818890	-1.224151
19	02-021	-1.070925	0.099622	-0.325327
20	02-023	-0.328730	-0.244951	-0.194888
21	02-024	0.381971	-0.147322	1.704287
22	02-025	0.665947	2.241317	1.200888
23	02-026	-0.541830	-0.144073	-0.505108
24	02-027	-1.170419	-1.515097	-1.098661
25	02-028	-0.900503	-1.542505	0.109093
26	02-029	-0.855128	-1.548089	-1.098661
27	02-030	-0.956972	-0.562259	0.372634
28	02-031	-1.509839	-1.182509	-1.224151
29	02-032	-1.073714	-0.491167	-0.158765
30	02-033	-1.322663	-1.756815	-1.224151
31	02-034	-1.863336	-1.669088	-1.143790
32	02-035	-1.230615	0.515848	-1.018299
33	02-036	-1.949873	-1.033171	0.990150
34	02-037	-1.420773	-1.792725	-1.224151
35	02-038	-1.163039	-0.311667	-0.892808
36	02-040	-0.773887	0.109825	0.010367
37	02-041	-1.098953	0.128560	-0.042948
38	02-042	-1.373636	-1.206642	-0.847679
39	02-043	-0.980659	-0.122293	-0.928041
40	02-044	-1.355790	-1.205382	-0.717241
41	02-045	-0.701175	-0.431780	-0.448440
42	02-046	-1.065667	-1.750844	-1.224151
43	02-047	-1.110419	-1.144263	-1.143790
44	02-048	-1.131071	-1.581845	-1.092381
45	02-049	-1.142764	-0.595693	-0.456363
46	02-050	-1.805286	-1.750115	-1.224151
47	02-051	-1.117869	-1.219623	-0.966890
48	02-052	0.190627	2.769786	1.168674
49	02-053	-0.753652	-1.236778	-0.921762
50	02-054	-0.453821	2.400281	0.015912
51	02-055	-1.524232	1.513112	0.978118

### 3.4 Resultados Finales

En este punto tenemos listas nuestras bases de datos a nivel colonia y alcaldia con las variables de concentración de desventajas, Desorden Social y Desorden Físico.

A modo de resguardo, exportaremos esta información.

```
#### Información a nivel Colonia : Concentración de desventajas + Desorden Social + Desorden Físico  
DESVE_COL.head(5)  
  
DESVE_COL.to_csv(Entradas + "Variables_Colonias.csv", encoding = 'Latin-1', index = False)  
  
#### Información a nivel Alcaldia : Concentración de desventajas + Desorden Social + Desorden Físico  
DESVE_ALCA.head(5)  
  
DESVE_ALCA.to_csv(Entradas + "Variables_Alcaldia.csv", encoding = 'Latin-1', index = False)
```

## **4 Methods**

## **5 Summary**

In summary, this book has no content whatsoever.

## **References**