

BOOK_PAPER

Jane Doe

2/9/23

Table of contents

Preface	3
1 Introduction	4
2 Data Acquisition and Preprocessing	5
3 Índice de Concentración de desventajas	6
4 Análisis de Componentes Principales (PCA)	16
4.0.1 Nivel Colonia	17
4.0.2 Nivel Alcaldías	18
5 Índice del Desorden Social y Físico	19
6 Variables de Control	27
6.1 METRO	27
6.2 METROBUS	29
6.3 DENUE: Tiendas, Bares y Restaurantes	32
6.4 Densidad de Población	34
7 Methods	37
8 Summary	38
References	39

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 Índice de 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	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 símbolos o caracteres especiales(*, N/D, 999999, etc), por lo que es necesario remplazarlos 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 consideraremos 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
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').rename({ "CVEUT": "CVE_COL"}, axis=1)
COLONIAS_CDMX['CVEDT'] = COLONIAS_CDMX['CVEDT'].str.zfill(3)

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 reproyecta

```
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 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”

$$\begin{aligned} M_{n-1} &= f(C) \\ C &\leftarrow M_{n-1} \end{aligned}$$

Se valida que no existan manzanas repetidas

```
INTERSECCION.CVEGEO.value_counts()
```

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

```
INTERSECCION.CVE_COL.value_counts()
```

CVE_COL	
07-320	471
07-263	370
17-078	297
11-057	213
17-076	213
07-289	209
13-067	184
09-012	180
07-145	178
11-004	178
13-043	173
07-286	173
07-157	169
11-052	168
03-030	168
08-045	159
07-270	158
07-285	149
05-002	148
11-058	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
03-012	132
05-032	132
07-269	132
07-290	132
15-012	131
06-059	130
07-274	129
07-051	129
07-162	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
16-045	120
07-046	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

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

	CVE_COL	NOMUT	ID	CVEGEO	ENTIDAD	NOM_ENT
32781	07-320	CHINAMPAC DE JUAREZ III	793	0900700014645052	09	Ciudad de M
31918	07-320	CHINAMPAC DE JUAREZ III	793	0900700014594040	09	Ciudad de M
32314	07-320	CHINAMPAC DE JUAREZ III	793	0900700014611041	09	Ciudad de M

4 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.

Importante

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

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

Para calcular la componente principal se ha creado una función que encapsula los siguientes procesos: 1. Selección de las dimensiones con las que se calcula el índice 2. Se normaliza la información por Z-Score 3. Se determina el número de componentes a reducir la información 4. Se calcula la varianza total por respecto al número 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):  
  
    ### Selección de las dimensiones con las que se calcula el índice de desventajas "PSIN"  
  
    PCA_X = TABLA.drop([DIM_CLAVE], axis = 1)  
    PCA_y = TABLA[[DIM_CLAVE]]  
  
    ### Se normaliza la información 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)

```

4.0.1 Nivel Colonias

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

```

COLONIA_PCA = pd.DataFrame(INTERSECCION.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

4.0.2 Nivel Alcaldías

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

```
ALCALDIA_PCA = pd.DataFrame(INTERSECCION.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 función creada para obtener los resultados a nivel alcaldía

```
### 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

5 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 5.1: 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 5.2: 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 = 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
'Aguas negras':'AGUA_N', 'Tirar basura en vi
'Corto circuito instalacion o subestacion el
'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": "C
                     "incidente_c4": "Incidente"}]

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

```

All_Data = gpd.GeoDataFrame(All_Data,
                            geometry = gpd.points_from_xy(All_Data.longitud,
                            All_Data.latitud), crs=4326).d
All_Data.head(2)

```

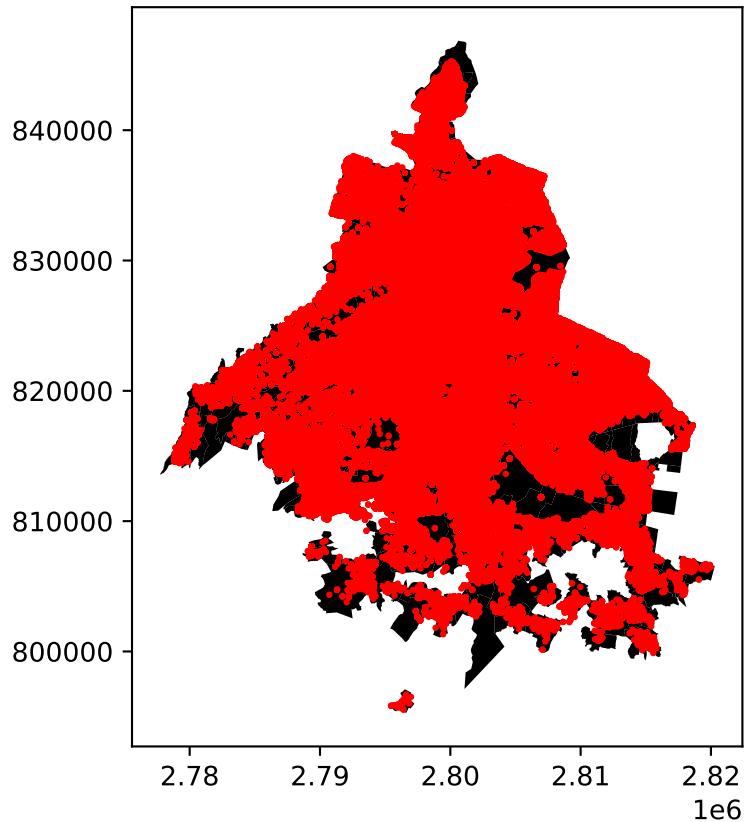
	folio	Categoría	Incidente	manzana	geometry
0	C5/210101/00770	Disturbio	FIESTA	9.007E+14	POINT (2800344.195 821939.006)
1	C5/210101/01333	Disturbio	FIESTA	9.007E+14	POINT (2807842.900 820599.774)

```

base = COLONIAS_CDMX.plot(color ='black',figsize=(5,5))
All_Data.plot(ax = base, color = 'red', markersize = 2, figsize=(5,5))

```

<AxesSubplot: >



```
CATEGORIAS = gpd.sjoin(COLONIAS_CDMX.drop(['ENT','NOMDT','NOMUT','DTTOLOC','ID'], axis = 1),
                       All_Data[['Categoria','Incidente','geometry']],
                       how = "inner",
                       op = 'contains').drop(['geometry','index_right'], axis = 1)

CATEGORIAS.head(2)
```

	CVEDT	CVE_COL	Categoría	Incidente
0	002	02-001	Disturbio	ESCANDALO
0	002	02-001	Disturbio	Fiesta

```
CATEGORIAS_DMY = pd.get_dummies(CATEGORIAS,
                                 columns=["Incidente"],
                                 prefix=["DM"]).rename({'CVEDT': 'MUN'}, axis = 1)
```

```
CATEGORIAS_DMY.head(3)
```

MUN	CVE_COL	Categoría	DM_AGUA_N	DM_AGUA_P	DM_BASURA_P	DM_CORTO
0	002	02-001	Disturbio	0	0	0
0	002	02-001	Disturbio	0	0	0
0	002	02-001	Disturbio	0	0	0

Ahora calculamos los indices

```
### Social a nivel colonia
```

```
SOCIAL_COL = pd.DataFrame(CATEGORIAS_DMY.groupby(['CVE_COL']).agg({'DM_DROGADO': 'sum',
                                                               'DM_EBRIOS': 'sum',
                                                               'DM_ESCANDALOS': 'sum',
                                                               'DM_FIESTAS': 'sum',
                                                               'DM_BASURA_P': 'sum',
                                                               'DM_PERSONAS_R': 'sum'})).reset_index()

COLONIA_SOCIAL = CONC_DIS(SOCIAL_COL, 'CVE_COL', 'SOCIAL_COL')
COLONIA_SOCIAL.head(2)
```

Total de la variancia explicada por la componente
56.539 %

CVE_COL	SOCIAL_COL
0 02-001	-0.899873
1 02-002	0.628847

```
### Fisico a nivel colonia
```

```
FISICO_COL = pd.DataFrame(CATEGORIAS_DMY.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'})).reset_index()

COLONIA_FISICO = CONC_DIS(FISICO_COL, 'CVE_COL', 'FISICO_COL')
COLONIA_FISICO.head(2)
```

Total de la variancia explicada por la componente
31.102 %

CVE_COL	FISICO_COL
0	02-001
1	02-002

```
### Social a nivel Municipal
```

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

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

Total de la variancia explicada por la componente
87.597 %

MUN	SOCIAL_MUN
0	002
1	003

```
### Fisico a nivel colonia
```

```
FISICO_MUN = pd.DataFrame(CATEGORIAS_DMY.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()

MUNI_FISICO = CONC_DIS(FISICO_MUN, 'MUN', 'FISICO_MUN')
MUNI_FISICO.head(2)
```

Total de la variancia explicada por la componente
66.908 %

MUN	FISICO_MUN
0 002	-0.647192
1 003	0.744289

UNION DE RESULTADOS

```
### Resultados Concentración de desventajas + Desorden Social + Desorden Físico
### NIVEL COLONIA
```

```
COLONIA_RESULTADOS = DESVE_COL.merge(COLONIA_SOCIAL,
                                       left_on = 'CVE_COL',
                                       right_on = 'CVE_COL',
                                       how = 'left').merge(COLONIA_FISICO,
                                       left_on = 'CVE_COL',
                                       right_on = 'CVE_COL',
                                       how = 'left')
```

```
COLONIA_RESULTADOS.head(5)
```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL
0	02-001	-1.121714	-0.899873	-1.004272
1	02-002	-0.533961	0.628847	-0.193206
2	02-003	-1.193403	-1.425013	-1.044140
3	02-005	-1.175400	-0.988030	-0.609931
4	02-006	-0.459954	1.012260	-0.483310

```
### Resultados Concentración de desventajas + Desorden Social + Desorden Físico
### NIVEL ALCALDIA
```

```
ALCALDIA_RESULTADOS = DESVE_ALCA.merge(MUNI_SOCIAL,
                                         left_on = 'MUN',
                                         right_on = 'MUN',
                                         how = 'left').merge(MUNI_FISICO,
                                         left_on = 'MUN',
                                         right_on = 'MUN',
                                         how = 'left')
```

```
ALCALDIA_RESULTADOS.head(5)
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN
0	002	-0.996294	-0.839195	-0.647192
1	003	-0.069455	0.905507	0.744289
2	004	-1.304359	-2.897155	-2.414544
3	005	2.040898	3.474421	3.265199
4	006	-0.650310	-0.448365	-0.774346

6 Variables de Control

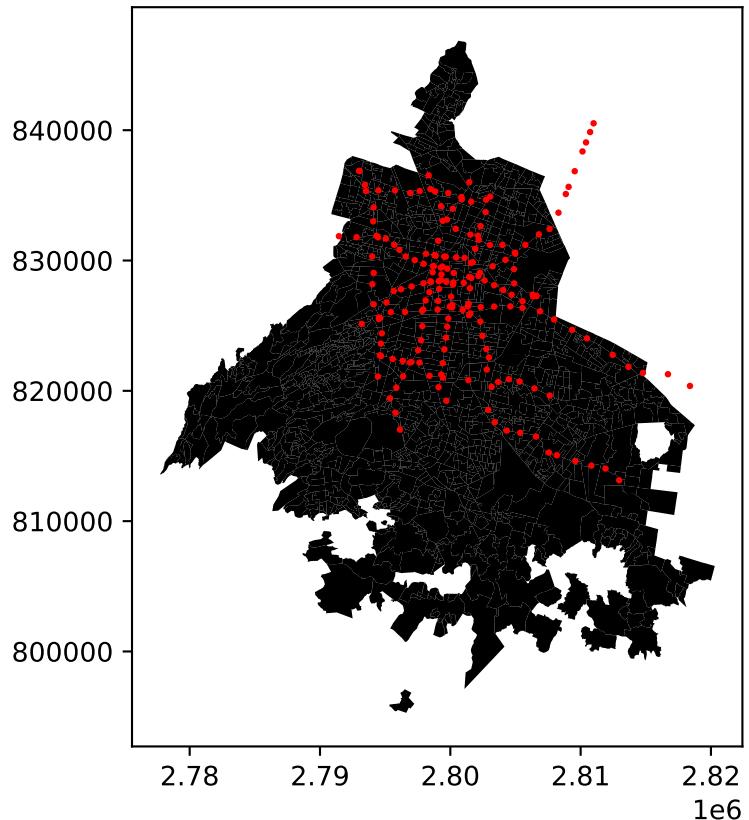
6.1 METRO

```
METRO_CDMX = gpd.read_file(Entradas +'STC_Metro.json').to_crs(6362)
METRO_CDMX = METRO_CDMX[['CVE_EST','geometry']]
METRO_CDMX.head(2)
```

	CVE_EST	geometry
0	STC0101	POINT Z (2806320.502 827386.150 0.000)
1	STC0102	POINT Z (2805539.596 826882.979 0.000)

```
base = COLONIAS_CDMX.plot(color ='black',figsize=(5,5))
METRO_CDMX.plot(ax = base, color = 'red', markersize = 2, figsize=(5,5))
```

```
<AxesSubplot: >
```



```

METRO_CDMX = gpd.sjoin(COLONIAS_CDMX,
                        METRO_CDMX[['CVE_EST','geometry']],
                        how = "inner",
                        op = 'contains').rename({'CVEDT': 'MUN',
                                                 'CVE_EST': 'METRO'}, axis = 1)
METRO_CDMX.head(2)

```

	ENT	MUN	NOMDT	DTTOLOC	CVE_COL	NOMUT	ID	geom
2	9	002	AZCAPOTZALCO	05	02-005	ANGEL ZIMBRON	3	POL
21	9	002	AZCAPOTZALCO	03	02-025	EL ROSARIO C (U HAB)	22	POL

```

METRO_COL = pd.DataFrame(METRO_CDMX.groupby(['CVE_COL']).agg({'METRO':'count'}).reset_index())
METRO_MUN = pd.DataFrame(METRO_CDMX.groupby(['MUN']).agg({'METRO':'count'}).reset_index())

```

```
### Resultados Concentración de desventajas + Desorden Social + Desorden Físico + METRO  
### NIVEL COLONIA
```

```
COLONIA_RESULTADOS = COLONIA_RESULTADOS.merge(METRO_COL,  
                                              left_on = 'CVE_COL',  
                                              right_on = 'CVE_COL',  
                                              how = 'left')  
COLONIA_RESULTADOS.head(5)
```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL	METRO
0	02-001	-1.121714	-0.899873	-1.004272	NaN
1	02-002	-0.533961	0.628847	-0.193206	NaN
2	02-003	-1.193403	-1.425013	-1.044140	NaN
3	02-005	-1.175400	-0.988030	-0.609931	1.0
4	02-006	-0.459954	1.012260	-0.483310	NaN

```
### Resultados Concentración de desventajas + Desorden Social + Desorden Físico + METRO  
### NIVEL ALCALDIA
```

```
ALCALDIA_RESULTADOS = ALCALDIA_RESULTADOS.merge(METRO_MUN,  
                                                left_on = 'MUN',  
                                                right_on = 'MUN',  
                                                how = 'left')  
ALCALDIA_RESULTADOS.head(5)
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN	METRO
0	002	-0.996294	-0.839195	-0.647192	10.0
1	003	-0.069455	0.905507	0.744289	6.0
2	004	-1.304359	-2.897155	-2.414544	NaN
3	005	2.040898	3.474421	3.265199	20.0
4	006	-0.650310	-0.448365	-0.774346	10.0

6.2 METROBUS

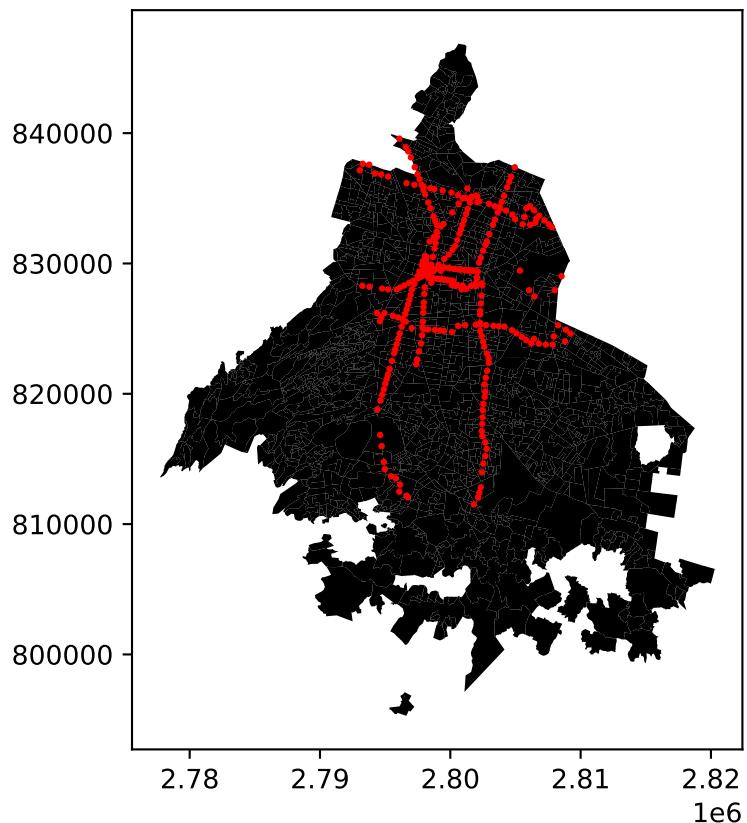
```
METROBUS_CDMX = gpd.read_file(Entradas +'STC_MetroBus.json').drop(['id', 'description',  
                      'styleUrl', 'stroke-opac  
                      'icon-opacity', 'icon-c  
                      'icon'], axis = 1).to_c
```

```
METROBUS_CDMX.head(2)
```

	SISTEMA	NOMBRE	LINEA	EST	CVE_EST	CVE_EOD17	TIPO	ALCALDI
0	Metrobús	Terminal 1 - Aeropuerto	04	41	MB0441			Venustiano
1	Metrobús	Terminal 2 - Aeropuerto	04	42	MB0442			Venustiano

```
base = COLONIAS_CDMX.plot(color ='black',figsize=(5,5))
METROBUS_CDMX.plot(ax = base, color = 'red', markersize = 2, figsize=(5,5))
```

```
<AxesSubplot: >
```



```
METROBUS_CDMX = gpd.sjoin(COLONIAS_CDMX,
                           METROBUS_CDMX[['CVE_EST','geometry']],
                           how = "inner",
```

```

op = 'contains').rename({'CVEDT': 'MUN',
                       'CVE_EST': 'METROBUS'}, axis = 1)
METROBUS_CDMX.head(2)

```

	ENT	MUN	NOMDT	DTTOLOC	CVE_COL	NOMUT	ID	geom
20	9	002	AZCAPOTZALCO	03	02-024	EL ROSARIO B (U HAB)	21	POL
21	9	002	AZCAPOTZALCO	03	02-025	EL ROSARIO C (U HAB)	22	POL

```

METROBUS_COL = pd.DataFrame(METROBUS_CDMX.groupby(['CVE_COL']).agg({'METROBUS': 'count'})).reset_index()
METROBUS_MUN = pd.DataFrame(METROBUS_CDMX.groupby(['MUN']).agg({'METROBUS': 'count'})).reset_index()

```

```

### Resultados Concentración de desventajas + Desorden Social + Desorden Físico + METRO +
### NIVEL COLONIA

```

```

COLONIA_RESULTADOS = COLONIA_RESULTADOS.merge(METROBUS_COL,
                                               left_on = 'CVE_COL',
                                               right_on = 'CVE_COL',
                                               how = 'left')

```

```
COLONIA_RESULTADOS.head(2)
```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL	METRO	METROBUS
0	02-001	-1.121714	-0.899873	-1.004272	NaN	NaN
1	02-002	-0.533961	0.628847	-0.193206	NaN	NaN

```

### Resultados Concentración de desventajas + Desorden Social + Desorden Físico + METRO +
### NIVEL ALCALDIA

```

```

ALCALDIA_RESULTADOS = ALCALDIA_RESULTADOS.merge(METROBUS_MUN,
                                                 left_on = 'MUN',
                                                 right_on = 'MUN',
                                                 how = 'left')

```

```
ALCALDIA_RESULTADOS.head(5)
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN	METRO	METROBUS
0	002	-0.996294	-0.839195	-0.647192	10.0	8.0
1	003	-0.069455	0.905507	0.744289	6.0	11.0
2	004	-1.304359	-2.897155	-2.414544	NaN	NaN
3	005	2.040898	3.474421	3.265199	20.0	72.0
4	006	-0.650310	-0.448365	-0.774346	10.0	17.0

6.3 DENUE: Tiendas, Bares y Restaurantes

```

DENUE = pd.read_csv(Entradas + 'DENUE.csv', encoding = 'Latin-1').drop(['entidad_cvegeo'],
DENUE = gpd.GeoDataFrame(DENUE,
                         geometry = gpd.points_from_xy(DENUE.denue_longitud,
                                                        DENUE.denue_latitud),
                         crs=4326).drop(['denue_latitud',
                                          'denue_longitud'], axis = 1).to_crs(6362)
DENUE.head(2)

```

	actividad_nombre	actividad_cve	geometry
0	Comercio al por menor de vinos y licores	461211	POINT (2803952.280 831440.241)
1	Comercio al por menor de vinos y licores	461211	POINT (2805964.648 834595.217)

```

def CLASE (codigo):
    Tienda = ['461110', '461211', '461212', '461213', '461220', '462111', '462112',
              '462210', '464111', '464112']
    Bares = ['722411', '722412']
    Restaurante = ['722511', '722512', '722513', '722514', '722515', '722516', '722517', '722518']
    usos = {'Tienda': Tienda, 'Bares':Bares, 'Restaurante': Restaurante}
    for actividad, claves in usos.items():
        for c in claves:
            if str(codigo).startswith(c):
                return actividad
    clases = DENUE['actividad_cve'].apply(CLASE)

DENUE['Clase'] = clases
DENUE = DENUE.drop(['actividad_nombre',
                     'actividad_cve'], axis = 1)
DENUE.head(5)

```

	geometry	Clase
0	POINT (2803952.280 831440.241)	Tienda
1	POINT (2805964.648 834595.217)	Tienda
2	POINT (2799011.530 839362.925)	Tienda
3	POINT (2799739.565 841534.399)	Tienda
4	POINT (2803412.265 837674.031)	Tienda

```

CLASE_DENUE = gpd.sjoin(COLONIAS_CDMX.drop(['ENT', 'NOMDT', 'NOMUT', 'DTTOLOC', 'ID'], axis = 1),
                         DENUE,
                         how = "inner",
                         op = 'contains').drop(['geometry', 'index_right'], axis = 1)
CLASE_DENUE.head(2)

```

	CVEDT	CVE_COL	Clase
0	002	02-001	Restaurante
0	002	02-001	Restaurante

```

DENUE_DMY = pd.get_dummies(CLASE_DENUE,
                           columns=["Clase"],
                           prefix=["DM"]).rename({'CVEDT': 'MUN'}, axis = 1)

```

```
DENUE_DMY.head(3)
```

	MUN	CVE_COL	DM_Bares	DM_Restaurante	DM_Tienda
0	002	02-001	0	1	0
0	002	02-001	0	1	0
0	002	02-001	0	0	1

```

DENUE_COL = pd.DataFrame(DENUE_DMY.groupby(['CVE_COL']).agg({'DM_Bares': 'count',
                                                               'DM_Restaurante': 'count',
                                                               'DM_Tienda': 'count'}).reset_

```

```

DENUE_MUN = pd.DataFrame(DENUE_DMY.groupby(['MUN']).agg({'DM_Bares': 'count',
                                                               'DM_Restaurante': 'count',
                                                               'DM_Tienda': 'count'}).reset_index()

```

```

COLONIA_RESULTADOS = COLONIA_RESULTADOS.merge(DENUE_COL,
                                               left_on = 'CVE_COL',
                                               right_on = 'CVE_COL',

```

```

    how = 'left')
COLONIA_RESULTADOS.head(5)

```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL	METRO	METROBUS	BAR	RESTAU
0	02-001	-1.121714	-0.899873	-1.004272	NaN	NaN	19.0	
1	02-002	-0.533961	0.628847	-0.193206	NaN	NaN	28.0	
2	02-003	-1.193403	-1.425013	-1.044140	NaN	NaN	17.0	
3	02-005	-1.175400	-0.988030	-0.609931	1.0	NaN	44.0	
4	02-006	-0.459954	1.012260	-0.483310	NaN	NaN	66.0	

```

ALCALDIA_RESULTADOS = ALCALDIA_RESULTADOS.merge(DENUE_MUN,
                                                left_on = 'MUN',
                                                right_on = 'MUN',
                                                how = 'left')

```

```
ALCALDIA_RESULTADOS.head(5)
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN	METRO	METROBUS	BAR	RESTAURA
0	002	-0.996294	-0.839195	-0.647192	10.0	8.0	4818	
1	003	-0.069455	0.905507	0.744289	6.0	11.0	6182	
2	004	-1.304359	-2.897155	-2.414544	NaN	NaN	2189	
3	005	2.040898	3.474421	3.265199	20.0	72.0	13156	
4	006	-0.650310	-0.448365	-0.774346	10.0	17.0	4405	

6.4 Densidad de Población

```
DEN_POB = pd.DataFrame(INTERSECCION.groupby(['CVE_COL']).agg({'POBTOT': 'sum'}).reset_index()
```

```

DEN_POB = COLONIAS_CDMX.merge(DEN_POB,
                                left_on = 'CVE_COL',
                                right_on = 'CVE_COL',
                                how = 'left').drop(['ENT', 'NOMDT',
                                                    'NOMUT', 'DTTOLOC',
                                                    'ID'],
                                                    axis = 1).rename({'CVEDT': 'MUN'}, axis =

```

```
DEN_POB['DEN_POB'] = DEN_POB['POBTOT']/DEN_POB.geometry.area
```

```

DENSIDAD_COLONIA = pd.DataFrame(DEN_POB.groupby(['CVE_COL']).agg({'DEN_POB':'sum'}).reset_index())

DENSIDAD_MUN = pd.DataFrame(DEN_POB.groupby(['MUN']).agg({'DEN_POB':'sum'}).reset_index())

COLONIA_RESULTADOS = COLONIA_RESULTADOS.merge(DENSIDAD_COLONIA,
                                              left_on = 'CVE_COL',
                                              right_on = 'CVE_COL',
                                              how = 'left')

COLONIA_RESULTADOS.head(5)

```

	CVE_COL	DIS_COL	SOCIAL_COL	FISICO_COL	METRO	METROBUS	BAR	RESTAUR
0	02-001	-1.121714	-0.899873	-1.004272	NaN	NaN	19.0	
1	02-002	-0.533961	0.628847	-0.193206	NaN	NaN	28.0	
2	02-003	-1.193403	-1.425013	-1.044140	NaN	NaN	17.0	
3	02-005	-1.175400	-0.988030	-0.609931	1.0	NaN	44.0	
4	02-006	-0.459954	1.012260	-0.483310	NaN	NaN	66.0	

```

ALCALDIA_RESULTADOS = ALCALDIA_RESULTADOS.merge(DENSIDAD_MUN,
                                                 left_on = 'MUN',
                                                 right_on = 'MUN',
                                                 how = 'left')

```

```
ALCALDIA_RESULTADOS.head(5)
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN	METRO	METROBUS	BAR	RESTAURA
0	002	-0.996294	-0.839195	-0.647192	10.0	8.0	4818	
1	003	-0.069455	0.905507	0.744289	6.0	11.0	6182	
2	004	-1.304359	-2.897155	-2.414544	NaN	NaN	2189	
3	005	2.040898	3.474421	3.265199	20.0	72.0	13156	
4	006	-0.650310	-0.448365	-0.774346	10.0	17.0	4405	

```
#### Se exportan los resultados
```

```
COLONIA_RESULTADOS
```

```
ALCALDIA_RESULTADOS
```

	MUN	DIS_MUN	SOCIAL_MUN	FISICO_MUN	METRO	METROBUS	BAR	RESTAURA
0	002	-0.996294	-0.839195	-0.647192	10.0	8.0	4818	
1	003	-0.069455	0.905507	0.744289	6.0	11.0	6182	
2	004	-1.304359	-2.897155	-2.414544	NaN	NaN	2189	
3	005	2.040898	3.474421	3.265199	20.0	72.0	13156	
4	006	-0.650310	-0.448365	-0.774346	10.0	17.0	4405	
5	007	5.736332	5.704887	4.365424	21.0	21.0	21223	
6	008	-1.099762	-2.221917	-1.689137	NaN	NaN	2197	
7	009	-1.163220	-3.380190	-2.896337	NaN	NaN	2033	
8	010	0.533438	0.970945	1.509073	2.0	6.0	6854	
9	011	-0.609511	-2.192476	-1.621580	5.0	NaN	4773	
10	012	0.688221	0.177062	0.829076	NaN	8.0	7239	
11	013	0.183677	-1.641095	-1.434769	NaN	4.0	5442	
12	014	-1.302296	-0.115571	0.024759	21.0	23.0	5449	
13	015	-0.151071	1.786959	2.497975	44.0	97.0	11005	
14	016	-1.227027	-0.369035	-1.343037	15.0	8.0	5347	
15	017	-0.609259	1.085217	-0.414850	30.0	28.0	7080	

7 Methods

8 Summary

In summary, this book has no content whatsoever.

References