Reto: Parte 1 Limpieza, análisis, visualización y kmeans

Ciencia y analítica de datos

Profesor: María de la Paz Rico Fernández

Juan Sebastián Ortega Briones A01794327

Equipo 13

16 de Noviembre del 2022

In [1]:
```
!pip3 install geopandas;
!pip3 install qeds;
```

...

In [2]:
```
import pandas as pd
import numpy as np
import geopandas as gpd
import fiona
import matplotlib.pyplot as plt
import matplotlib.colors as color
from tqdm import tqdm
from shapely.geometry import Point
import qeds
import seaborn as sns
qeds.themes.mpl_style()
%matplotlib inline
```

In [3]:
```
qeds.themes.mpl_style()
pd.set_option("display.max_columns", 57)
pd.set_option("display.max_rows", 100)
```

Uso de Base de datos de Aguas Subterraneas

In [4]:
```
df=pd.read_csv("https://raw.githubusercontent.com/PosgradoMNA/activida
```

In [5]:
```
df.head()
```

Out[5]:

	CLAVE	SITIO	ORGANISMO_DE_CUENCA	ESTADO	MUNICIPIO	
0	DLAGU6	POZO SAN GIL	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	ASIENTOS	
1	DLAGU6516	POZO R013 CAÑADA HONDA	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	AGUASCALIENTES	
2	DLAGU7	POZO COSIO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	COSIO	A
3	DLAGU9	POZO EL SALITRILLO	LERMA SANTIAGO PACIFICO	AGUASCALIENTES	RINCON DE ROMOS	A
4	DLBAJ107	RANCHO EL TECOLOTE	PENINSULA DE BAJA CALIFORNIA	BAJA CALIFORNIA SUR	LA PAZ	

In [6]:
```
df.shape
```

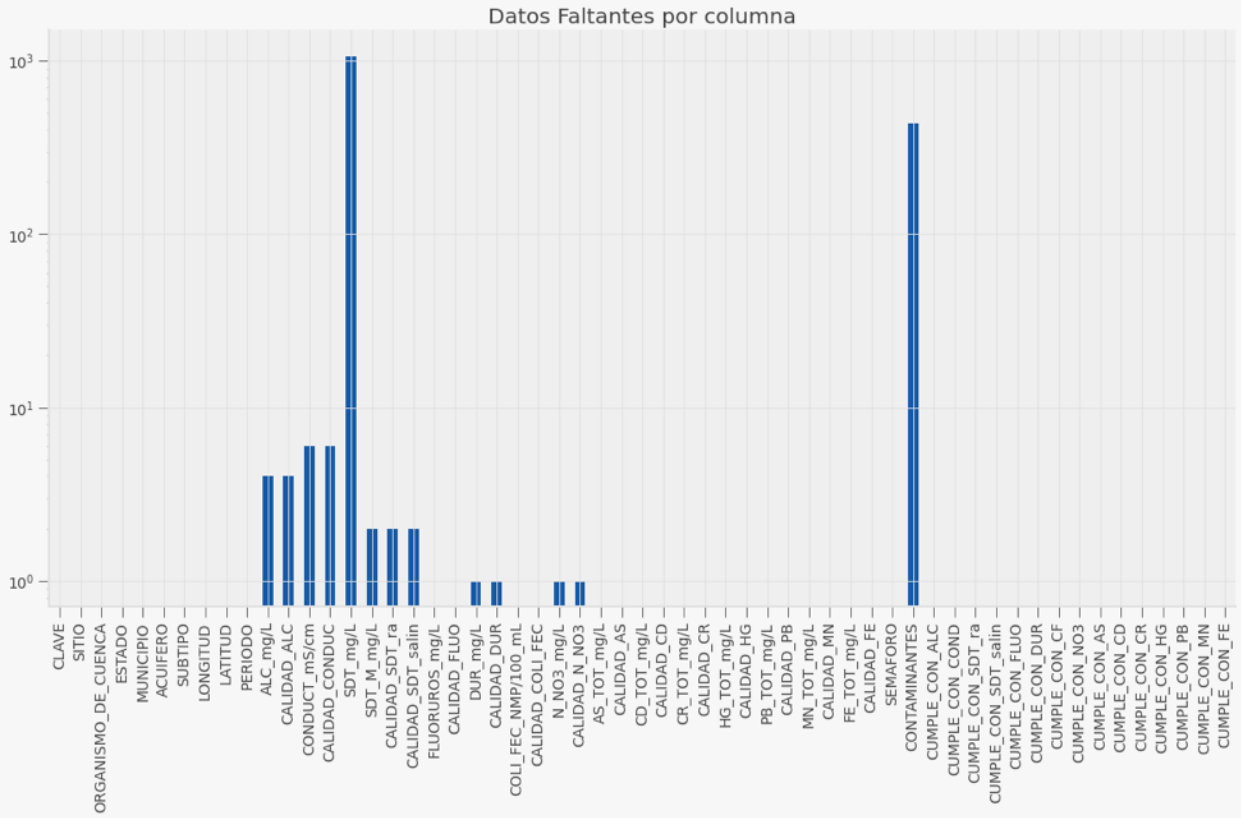
Out[6]:
```
(1068, 57)
```

Limpieza de datos

In [7]:
```
#Array de solo las columnas numericas
datos_numericos=['LATITUD', 'LONGITUD','ALC_mg/L', 'CONDUCT_mS/cm', 'S
```

In [8]:
```
#Remplaza datos numericos que incluyen el simbolo < por 0
df.replace(to_replace=r'[<]\w+', value=0, regex=True, inplace=True)
```

In [9]:
```
#Cuantas filas contienen NaN por columna
plt.figure(figsize=(20, 10))
ax=df.isna().sum().plot.bar(logy=True).set_title('Datos Faltantes por
plt.show()
```



In [10]:
```
#La columna SDT_mg/L no contiene datos será eliminada
df.drop(['SDT_mg/L'], axis=1, inplace=True)
```

In [11]:
```
df.shape
```

Out[11]:
```
(1068, 56)
```

Exporación de datos

In [12]:
```
#Convierte columnas que contienen datos números de tipo objeto a tip
df[datos_numericos]=df[datos_numericos].astype('float')
```

Identificación de Tenedencias y descripción de los datos

In [13]:
```
df[datos_numericos].describe()
```

Out[13]:

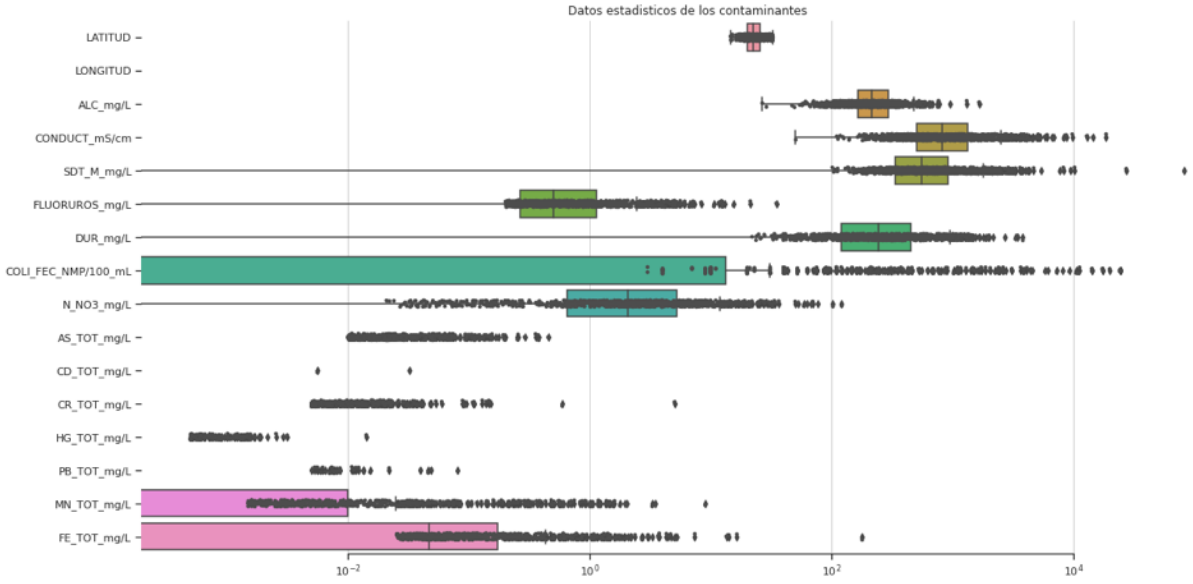
	LATITUD	LONGITUD	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	FLUORUROS_
count	1068.000000	1068.000000	1064.000000	1062.000000	1066.000000	1068.00
mean	23.163618	-101.891007	235.633759	1138.953013	896.078115	1.04
std	3.887670	6.703263	116.874291	1245.563674	2751.538128	1.93
min	14.561150	-116.664250	26.640000	50.400000	0.000000	0.00
25%	20.212055	-105.388865	164.000000	501.750000	337.500000	0.26
50%	22.617190	-102.174180	215.527500	815.000000	550.400000	0.50
75%	25.510285	-98.974716	292.710000	1322.750000	916.100000	1.13
max	32.677713	-86.864120	1650.000000	18577.000000	82170.000000	34.80

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1068 entries, 0 to 1067
Data columns (total 56 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CLAVE                 1068 non-null   object
 1   SITIO                 1068 non-null   object
 2   ORGANISMO_DE_CUENCA   1068 non-null   object
 3   ESTADO                1068 non-null   object
 4   MUNICIPIO             1068 non-null   object
 5   ACUIFERO              1068 non-null   object
 6   SUBTIPO               1068 non-null   object
 7   LONGITUD              1068 non-null   float64
 8   LATITUD               1068 non-null   float64
 9   PERIODO               1068 non-null   int64
 10  ALC_mg/L              1064 non-null   float64
 11  CALIDAD_ALC           1064 non-null   object
 12  CONDUCT_mS/cm         1062 non-null   float64
 13  CALIDAD_CONDUC        1062 non-null   object
 14  SDT_M_mg/L            1066 non-null   float64
 15  CALIDAD_SDT_ra        1066 non-null   object
 16  CALIDAD_SDT_salin     1066 non-null   object
 17  FLUORUROS_mg/L        1068 non-null   float64
 18  CALIDAD_FLUO          1068 non-null   object
 19  DUR_mg/L              1067 non-null   float64
 20  CALIDAD_DUR           1067 non-null   object
 21  COLI_FEC_NMP/100_mL   1068 non-null   float64
 22  CALIDAD_COLI_FEC      1068 non-null   object
 23  N_NO3_mg/L            1067 non-null   float64
 24  CALIDAD_N_NO3         1067 non-null   object
 25  AS_TOT_mg/L           1068 non-null   float64
 26  CALIDAD_AS            1068 non-null   object
 27  CD_TOT_mg/L           1068 non-null   float64
```

```
 28  CALIDAD_CD            1068 non-null   object
 29  CR_TOT_mg/L           1068 non-null   float64
 30  CALIDAD_CR            1068 non-null   object
 31  HG_TOT_mg/L           1068 non-null   float64
 32  CALIDAD_HG            1068 non-null   object
 33  PB_TOT_mg/L           1068 non-null   float64
 34  CALIDAD_PB            1068 non-null   object
 35  MN_TOT_mg/L           1068 non-null   float64
 36  CALIDAD_MN            1068 non-null   object
 37  FE_TOT_mg/L           1068 non-null   float64
 38  CALIDAD_FE            1068 non-null   object
 39  SEMAFORO              1068 non-null   object
 40  CONTAMINANTES         634 non-null    object
 41  CUMPLE_CON_ALC        1068 non-null   object
 42  CUMPLE_CON_COND       1068 non-null   object
 43  CUMPLE_CON_SDT_ra     1068 non-null   object
 44  CUMPLE_CON_SDT_salin  1068 non-null   object
 45  CUMPLE_CON_FLUO       1068 non-null   object
 46  CUMPLE_CON_DUR        1068 non-null   object
 47  CUMPLE_CON_CF         1068 non-null   object
 48  CUMPLE_CON_NO3        1068 non-null   object
 49  CUMPLE_CON_AS         1068 non-null   object
 50  CUMPLE_CON_CD         1068 non-null   object
 51  CUMPLE_CON_CR         1068 non-null   object
 52  CUMPLE_CON_HG         1068 non-null   object
 53  CUMPLE_CON_PB         1068 non-null   object
 54  CUMPLE_CON_MN         1068 non-null   object
 55  CUMPLE_CON_FE         1068 non-null   object
dtypes: float64(16), int64(1), object(39)
memory usage: 467.4+ KB
```

In [15]:
```
sns.set_theme(style="ticks")
f, ax = plt.subplots(figsize=(20, 10))
ax.set_xscale("log")

sns.boxplot(data=df[datos_numericos],orient='h').set(title='Datos esta
sns.stripplot(data=df[datos_numericos],size=4, color=".3", linewidth=0
ax.xaxis.grid(True)
ax.set(ylabel="")
sns.despine(trim=True, left=True)
plt.show()
```



Correlación entre contaminantes de Aguas Subterraneas

In [16]:
```
corr=df[datos_numericos].corr()
corr
```

Out[16]:

	LATITUD	LONGITUD	ALC_mg/L	CONDUCT_mS/cm	SDT_M_mg/L	F
LATITUD	1.000000	-0.760204	-0.080026	0.053786	0.059881	
LONGITUD	-0.760204	1.000000	0.167234	0.061118	-0.013923	
ALC_mg/L	-0.080026	0.167234	1.000000	0.232003	0.079350	
CONDUCT_mS/cm	0.053786	0.061118	0.232003	1.000000	0.286562	
SDT_M_mg/L	0.059881	-0.013923	0.079350	0.286562	1.000000	
FLUORUROS_mg/L	0.140579	-0.133322	0.068982	-0.023772	-0.012557	
DUR_mg/L	0.084626	0.088529	0.242484	0.692690	0.346973	
COLI_FEC_NMP/100_mL	0.063152	-0.084477	-0.016441	0.017829	-0.001110	
N_NO3_mg/L	0.148279	-0.107561	-0.000442	0.219482	0.101710	
AS_TOT_mg/L	0.111556	-0.097533	0.072592	-0.008690	-0.013096	
CD_TOT_mg/L	-0.052289	0.043521	0.030589	0.027178	0.010109	
CR_TOT_mg/L	-0.060092	0.058767	-0.014136	0.004159	0.000194	
HG_TOT_mg/L	-0.141567	0.049549	0.076929	0.048255	0.037751	
PB_TOT_mg/L	-0.073510	0.013652	0.023182	0.026432	0.030445	
MN_TOT_mg/L	-0.036325	-0.036189	0.130074	0.096223	0.019085	
FE_TOT_mg/L	-0.042002	0.022395	0.043638	0.083540	0.020271	

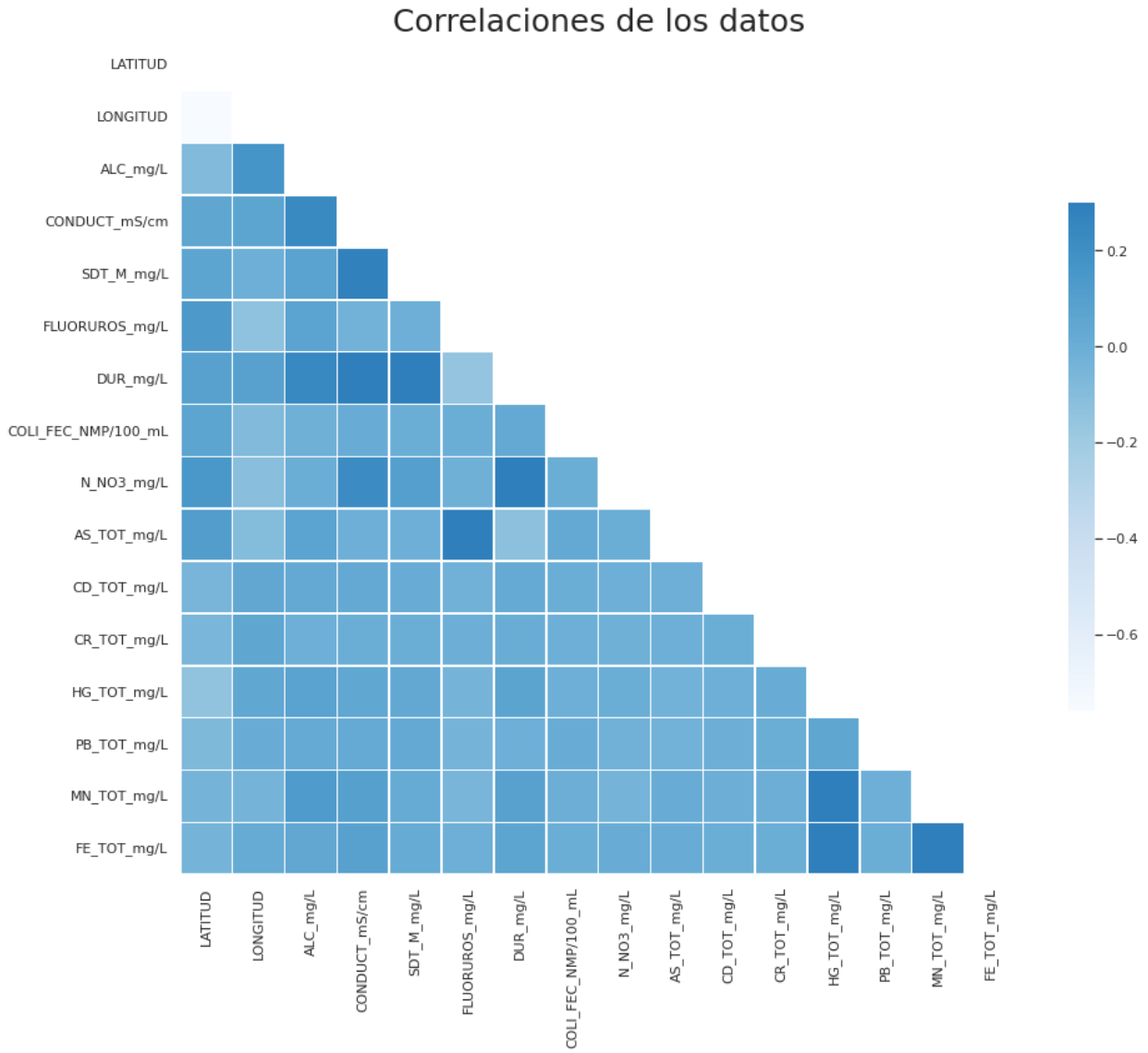
Correlaciones más grandes

In [17]:
```
df[datos_numericos].corr().unstack().sort_values().drop_duplicates().h
```

Out[17]:
```
LATITUD              LONGITUD      -0.760204
FLUORUROS_mg/L       DUR_mg/L      -0.152799
HG_TOT_mg/L          LATITUD       -0.141567
FLUORUROS_mg/L       LONGITUD      -0.133322
AS_TOT_mg/L          DUR_mg/L      -0.121827
N_NO3_mg/L           LONGITUD      -0.107561
LONGITUD             AS_TOT_mg/L   -0.097533
COLI_FEC_NMP/100_mL  LONGITUD      -0.084477
ALC_mg/L             LATITUD       -0.080026
PB_TOT_mg/L          LATITUD       -0.073510
dtype: float64
```

In [18]:
```
sns.set_theme(style="white")
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(15, 15))

# Generate a custom diverging colormap
#cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,  cmap='Blues',  vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Correlaciones de los datos', fontsize = 25)
```

Out[18]: Text(0.5, 1.0, 'Correlaciones de los datos')



Kmeans

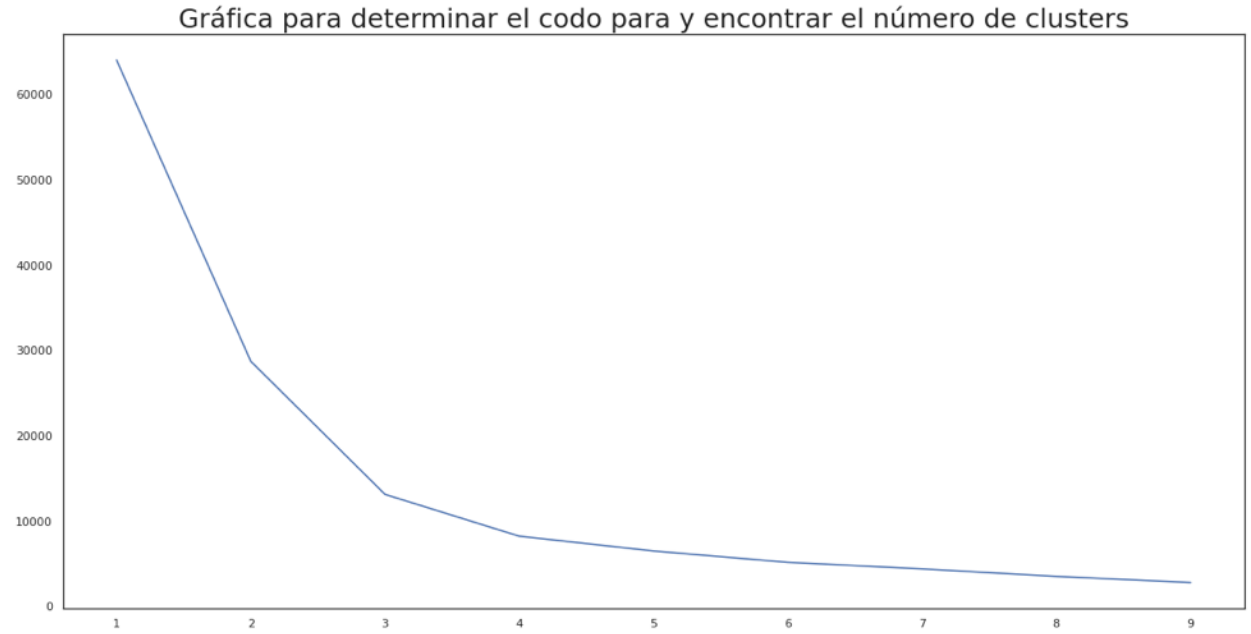
In [19]:
```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
```

In [20]:
```
clusters=10
wcss=[]

for i in range(1,clusters):
  kmeans=KMeans(n_clusters=i, max_iter=3000)
  kmeans.fit(df[['LATITUD', 'LONGITUD']])
  wcss.append(kmeans.inertia_)
```

In [21]:
```
len(wcss)
```

Out[21]: 9

In [22]:
```
from matplotlib.pyplot import figure
figure(figsize=(20, 10))
plt.plot(range(1,clusters), wcss)
plt.title('Gráfica para determinar el codo para y encontrar el número
plt.show()
```



In [23]:
```
df["COORDENADAS"] = list(zip(df.LONGITUD, df.LATITUD))   # Se crea la
df["COORDENADAS"] = df["COORDENADAS"].apply(Point)        # Se convier
```

In [24]:
```
gdf = gpd.GeoDataFrame(df, geometry="COORDENADAS")
gdf[['LATITUD','LONGITUD', 'COORDENADAS']].head()
```

Out[24]:

	LATITUD	LONGITUD	COORDENADAS
0	22.20887	-102.02210	POINT (-102.02210 22.20887)
1	21.99958	-102.20075	POINT (-102.20075 21.99958)
2	22.36685	-102.28801	POINT (-102.28801 22.36685)
3	22.18435	-102.29449	POINT (-102.29449 22.18435)
4	23.45138	-110.24480	POINT (-110.24480 23.45138)

In [25]:
```
k = 30     # Para tratar de agrupar en zonas pequeñas que traten de as
kmeans = KMeans(n_clusters=k, max_iter=300)
y_pred = kmeans.fit(gdf[['LATITUD', 'LONGITUD']])
y_pred.labels_
```

Out[25]:
```
array([ 0,  0,  0, ..., 16, 16, 16], dtype=int32)
```

In [26]:
```
df.shape, y_pred.labels_.shape
```

Out[26]:
```
((1068, 57), (1068,))
```

In [27]:
```
gdf['y']=y_pred.labels_
gdf[['CLAVE', 'COORDENADAS', 'SEMAFORO', 'y']].head(5)
```

Out[27]:

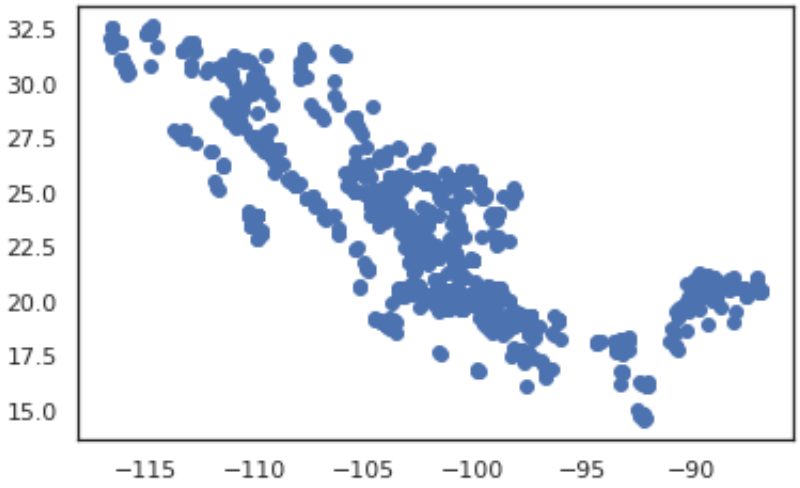
	CLAVE	COORDENADAS	SEMAFORO	y
0	DLAGU6	POINT (-102.02210 22.20887)	Verde	0
1	DLAGU6516	POINT (-102.20075 21.99958)	Verde	0
2	DLAGU7	POINT (-102.28801 22.36685)	Rojo	0
3	DLAGU9	POINT (-102.29449 22.18435)	Verde	0
4	DLBAJ107	POINT (-110.24480 23.45138)	Rojo	8

In [28]:
```
gdf.to_excel("salida.xls")
```

...

In [29]:
```
gdf.plot()
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f43ea94f510>
```



In [30]:
```
cmap=color.ListedColormap(["darkred", "gold", "lawngreen"])
```

In [31]:
```
categorias=gdf['SEMAFORO'].unique()
categorias
```

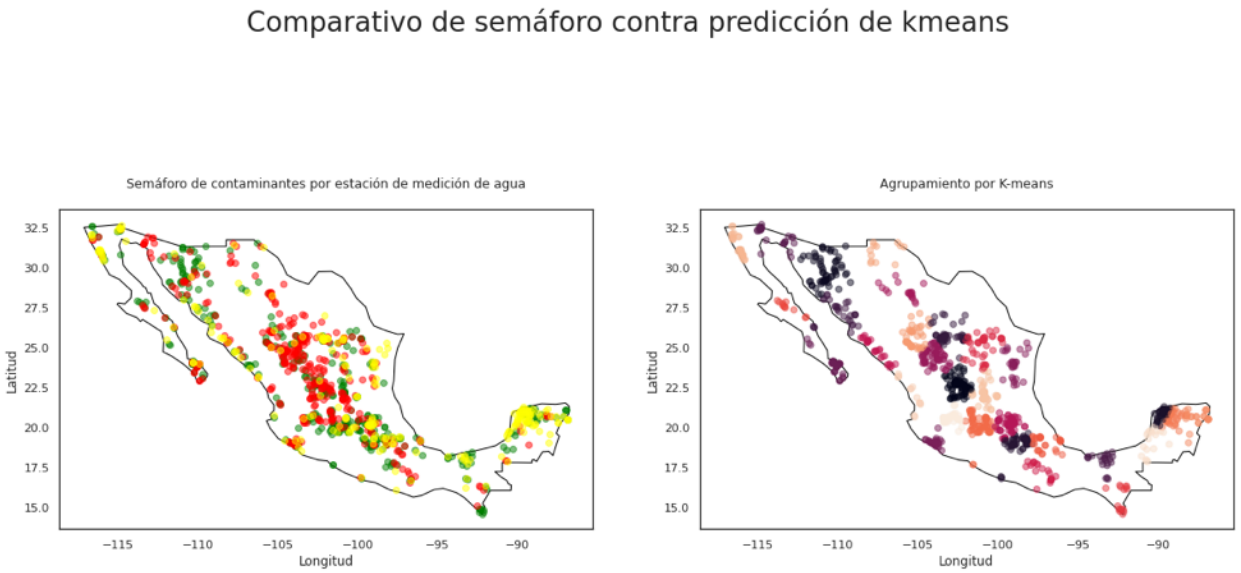
Out[31]:
```
array(['Verde', 'Rojo', 'Amarillo'], dtype=object)
```

In [32]:
```
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world = world.set_index("iso_a3")

#world.head()
```

Type *Markdown* and LaTeX: α^2

In [33]:

```
fig, gax = plt.subplots(1,2,figsize=(20,10))

world.query("name == 'Mexico'").plot(ax = gax[0], edgecolor='black', c
world.query("name == 'Mexico'").plot(ax = gax[1], edgecolor='black', c

gdf[gdf['SEMAFORO']==categorias[0]].plot(ax=gax[0],color='green' ,alph
gdf[gdf['SEMAFORO']==categorias[1]].plot(ax=gax[0],color='red' ,alpha
gdf[gdf['SEMAFORO']==categorias[2]].plot(ax=gax[0],color='yellow' ,alp

gax[0].set_xlabel('Longitud')
gax[0].set_ylabel('Latitud')
gax[0].set_title('Semáforo de contaminantes por estación de medición d

gdf.plot(ax=gax[1],column='y' ,alpha = 0.5)

gax[1].set_xlabel('Longitud')
gax[1].set_ylabel('Latitud')
gax[1].set_title('Agrupamiento por K-means\n')

fig.suptitle('Comparativo de semáforo contra predicción de kmeans', fo

plt.show()
```



En caso de usar 3 clusters para kmeans, podemos comparar directamente contra el semáforo

In [34]:

```
k = 3
kmeans = KMeans(n_clusters=k, max_iter=100000)
y_pred = kmeans.fit(gdf[['LATITUD', 'LONGITUD']])
y_pred.labels_
```

Out[34]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int32)
```

In [35]:
```
gdf['y']=y_pred.labels_
gdf[['CLAVE', 'COORDENADAS', 'SEMAFORO', 'y']].head(5)
```

Out[35]:

	CLAVE	COORDENADAS	SEMAFORO	y
0	DLAGU6	POINT (-102.02210 22.20887)	Verde	1
1	DLAGU6516	POINT (-102.20075 21.99958)	Verde	1
2	DLAGU7	POINT (-102.28801 22.36685)	Rojo	1
3	DLAGU9	POINT (-102.29449 22.18435)	Verde	1
4	DLBAJ107	POINT (-110.24480 23.45138)	Rojo	2

In [36]:
```
categorias
```

Out[36]:
```
array(['Verde', 'Rojo', 'Amarillo'], dtype=object)
```

In [37]:
```
#Cambia la columna del semaforo de un string a un numero para poder co
gdf['SEMAFORO_CAT']=gdf['SEMAFORO'].apply(lambda x: categorias.tolist(
```

In [38]:
```
gdf[['SEMAFORO_CAT', 'y']].head()
```

Out[38]:

	SEMAFORO_CAT	y
0	0	1
1	0	1
2	1	1
3	0	1
4	1	2

In [39]:
```
iguales=0
diferentes=0

for n in range(gdf.shape[0]):
 if gdf[['SEMAFORO_CAT']].iloc[n].values.tolist()[0]== gdf[['y']].iloc
    iguales+=1
print("Son iguales: ", iguales)
print("Porcentaje: ", iguales/gdf.shape[0]*100,'%')
```

```
Son iguales:  407
Porcentaje:  38.10861423220974 %
```