Visualización de Información Geoespacial.

En esta sección vamos a ver cómo representar información en un mapa. Muchas veces es importante marcar puntos importantes en un mapa para la ayuda en la toma de desiciones. Contar con la información geoespacial adecuada nos ayuda a observar patrones, carencias u oportunidades. La mayoría de las personas se siente cómoda interpretando un mapa.

Lo primero que necesitamos es el mapa donde vamos a colocar la información. Los mapas se pueden encontrar en archivos con formato **shape** o **shapefiles**. Estos se pueden leer usando la librería `geopandas`. En este ejemplo usaremos el mapa de México que se encuentra en el archivo `dest2029gw.shp`.

Tenemos archivos que componen la capa geoespacial en formato vectorial, los metadatos y las imágenes de referencia, todos con el mismo nombre pero diferente extensión, que se deben almacenar en el mismo directorio.

- La capa vectorial en formato shapefile que se conforma de cuatro archivos que son indispensables para abrir el mapa, estos son:
  - shp: archivo principal que almacena la geometría de la entidad.
  - shx: archivo que almacena el índice de la geometría de la entidad.
  - dbf: tabla dBASE que almacena la información de atributos de las entidades.
  - prj: archivo con la información del sistema de coordenadas de la capa.

Veamos un ejemplo de cómo se usa.

```
import geopandas as gpd
import matplotlib.pyplot as plt
```

```
map_data = gpd.read_file("dest2019gw.shp")
map_data.shape
```

```
(32, 9)
```

```
map_data.columns
```

```
Index(['CVE_ENT', 'NOM_ENT', 'CVE_CAP', 'NOM_CAP', 'AREA', 'PERIMETER', 'COV_',
       'COV_ID', 'geometry'],
      dtype='object')
```

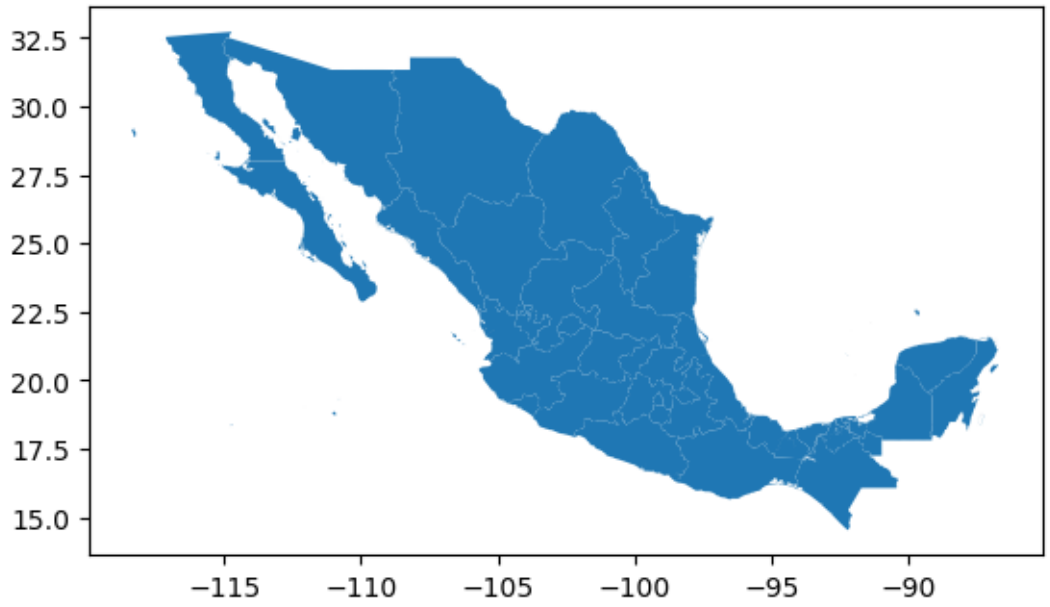
Tiene 32 entradas, una por cada estado de la república. La geografía de cada estado se encuentra en la columna geometry.

```
map_data[['NOM_ENT', 'geometry']].head(2)
```

	NOM_ENT	geometry
0	Aguascalientes	POLYGON ((-101.86167 22.02888, -101.86167 22.0...
1	Baja California	MULTIPOLYGON (((-114.12889 28.01224, -114.1284...

Podemos ver que el estado de Aguascalientes tiene un polígono, formado por los pares de las coordenadas de cada vértice. El estado de Baja California se compone de varios polígonos ya que su territorio incluye algunas islas. Además de polígonos, la información geoespacial pueden ser puntos o líneas. Para mostrar el mapa simplemente se usa el método `plot`.

```
map_data.plot()
```



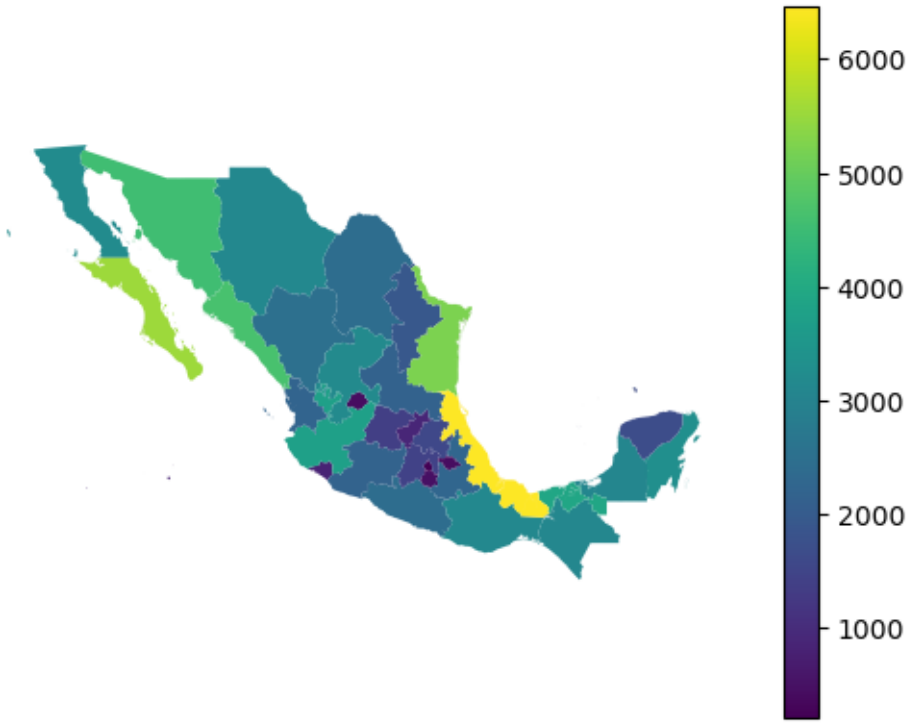
Este método tiene varias opciones y podemos mostrar sólo los bordes. También quitamos los ejes ya que no son relevantes en el caso de los mapas.

```
ax = map_data.boundary.plot(edgecolor="black")
ax.set_axis_off()
```

El archivo shape contiene información adicional como la longitud del perímetro de cada estado. Podemos colorear el mapa deacuerdo a alguna de sus columnas. En el siguiente mapa podemos ver que el estado con mayor perímetro es Veracruz.

> ***Actividad:*** Mostrar el mapa de los estados de la república coloreados de acuerdo a la columna AREA.

```
ax = map_data.plot(column='PERIMETER', legend=True)
ax.set_axis_off();
```



Los siguientes mapas se sacaron de esta página. Hay muchos más que pueden usar para practicar. En el ejemplo se usó el mapa Régimen térmico donde muestra zonas con diferentes climas. Contiene varias columnas con información adicional.

Actividad: Descargar otros datos y mostrarlos en un mapa.

```
temp = gpd.read_file("GeoJSON - Régimen térmico.json")
temp.columns
```

```
Index(['id', 'gid', 'zona_clima', 'simbol_cli', 'reg_temp', 'precipita',
       'subt_clima', 'limite_ter', 'limite_pre', 'perio_lluv', 'area_km2',
       'geometry'],
      dtype='object')
```

Vamos a usar la columna `zona_clima` para mostrar los diferentes climas que hay en México. Primero mostramos los nombres de las zonas y usamos la columna `area_km2` para calcular el total de superficie de cada zona.

```
temp['zona_clima'].value_counts()
```

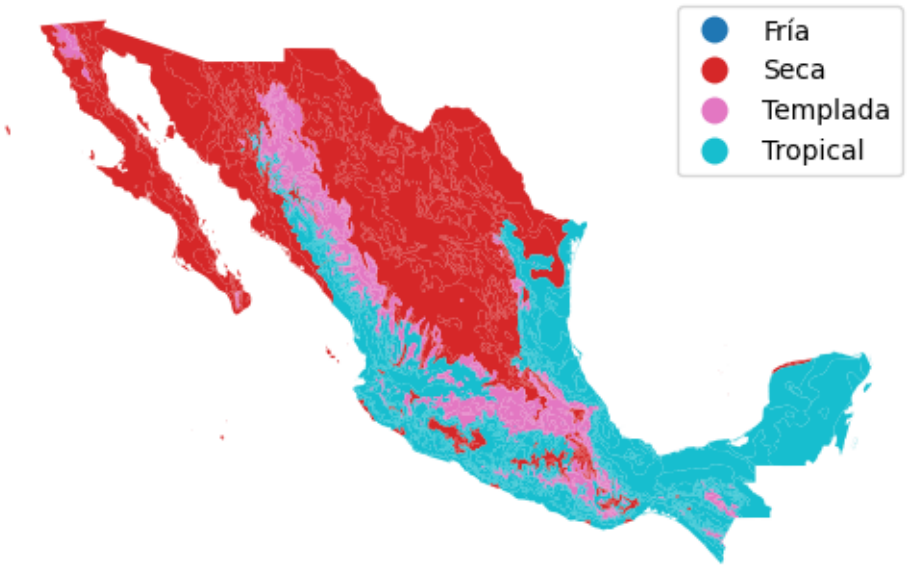
```
zona_clima
Tropical    476
Seca        387
Templada    311
Fría          8
Name: count, dtype: int64
```

```
zonas = ['Tropical', 'Seca', 'Templada', 'Fría']
for z in zonas:
    area = temp[temp['zona_clima'] == z]['area_km2'].sum()
    print(f"Area de la zona {z} es", area)
```

```
Area de la zona Tropical es 695874.5643182335
Area de la zona Seca es 1004438.2674344108
Area de la zona Templada es 242276.1553813477
Area de la zona Fría es 82.639321188954
```

Ahora mostramos el mapa.

```
ax = temp.plot(column='zona_clima', legend=True)
ax.set_axis_off()
```

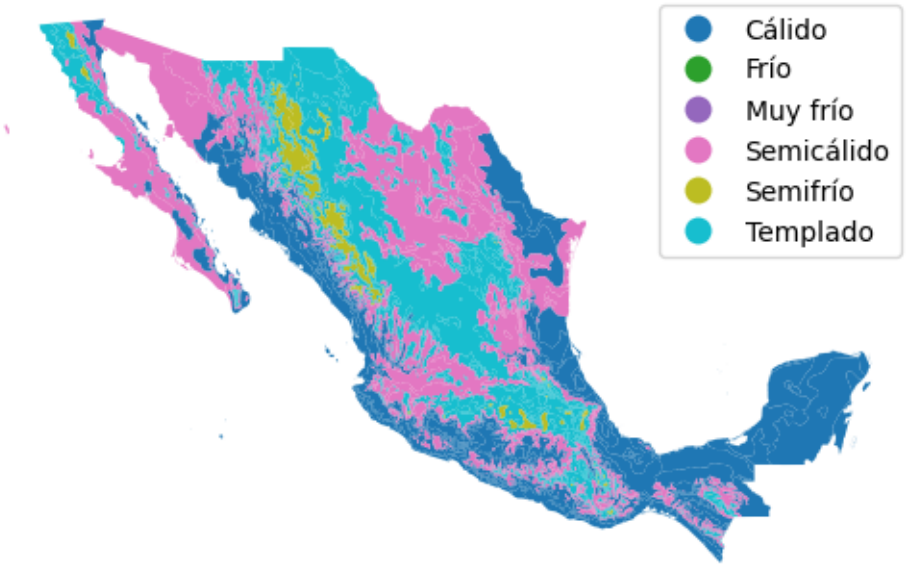


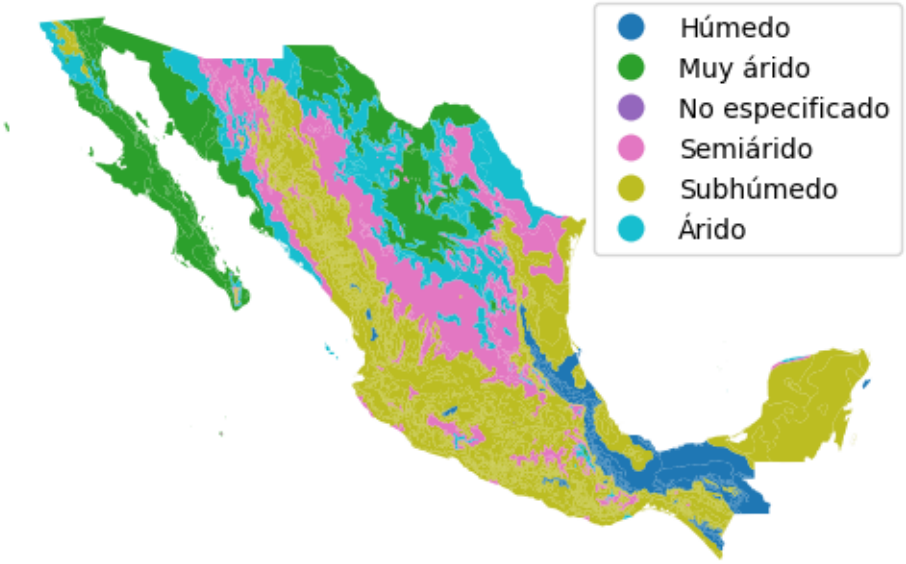
Podemos notar que la zona `Tropical` le tocó un color azul, que normalmente se usa para indicar climas frios. Vamos a cambiar los colores por unos más aprociados.

```
def zone2color(zone):
    d = {'Seca':'tan', 'Templada':'g', 'Tropical':'y', 'Fría':'b'}
    return d[zone]

temp['color'] = temp['zona_clima'].map(zone2color)
ax = temp.plot(color=temp['color'])
ax.set_axis_off()
```

Actividad: Mostrar los mapas usando las columnas `reg_temp` y `precipita`.





Combinar dos mapas en uno

Podemos poner la información de dos mapas en uno usando objetos de `plotly`. Primero, leemos el archivo que contiene las coordenadas de las estaciones de ferrocaril. Luego graficamos el mapa de México y sobre él ponemos las marcas de las terminales.

```
trenes = gpd.read_file("GeoJSON - Terminales de ferrocarril.json")
ax = map_data.plot(figsize=(6,6), color='none', edgecolor='gainsboro', zorder=3)
trenes.plot(markersize=1, ax=ax)
ax.set_axis_off()
```

Sistemas de coordenadas

Los datos geográficos tiene un sistema de coordenadas que es el que se usa para poner los puntos en el mapa. Cuando se van a combinar varios datos de diferentes fuentes, es importante verificar que todas usen el mismo sistema de coordenadas.

```
map_data.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 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

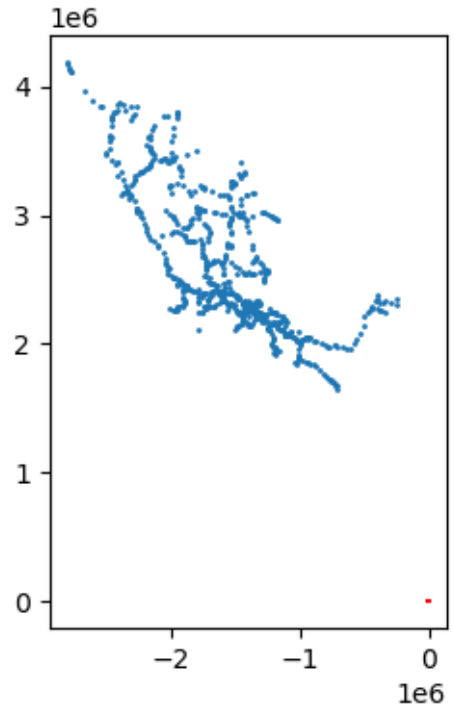
```
trenes.crs
```

```
<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 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Podemos ver que ambos mapas usan sistema EPSG 4326 por lo que no hay problema. Este sistema preserva ángulos, lo que lo hace útil para navegación, pero modifica un poco las áreas. Existen diferentes sistemas de coordenadas con diferentes características, todos son una proyeccción de la esfera a un plano en 2 dimensiones. Todas las proyecciones tienen un error o distorción de los datos. Algunos preservan distancias, áreas o ángulos y tienen diferentes usos.

Se puede cambiar de sistema de cooredenadas con `to_crs`. En el ejemplo se ve que si cambiamos el sistema de coordenadas del mapa de estaciones de trenes, ya no empalma al mapa de México (es el punto rojo en la esquina inferior derecha).

```
ax = map_data.plot(figsize=(4,4), color='none', edgecolor='red', zorder=3)
trenes.to_crs(epsg=26717).plot(markersize=1, ax=ax)
```



Los polígonos que se encuentran en la columna `geometry` tienen propiedades que podemos usar, como la de sacar su centroide. Al sacar el centroide se usan las distancias y como el sistema de coordenadas modifica un poco las distancias, obtenemos un *warning* si queremos obtenerlos.

```
map_data['geometry'].centroid.head()
```

```
C:\Users\msubr\AppData\Local\Temp\ipykernel_5740\1010000702.py:1: UserWarning: Geometry is in

  map_data['geometry'].centroid.head()

0    POINT (-102.36194 22.00644)
1    POINT (-115.09757 30.55386)
2    POINT (-112.04864 25.92118)
3     POINT (-90.31589 18.84845)
4    POINT (-102.04404 27.29544)
dtype: geometry
```

Lo que podemos hacer es proyectar los polígonos usando un sistema de coordenas que preserve distancias y luego devolver los centroides al sistema original.

```
map_with_names = map_data[['NOM_ENT', 'geometry']].copy()
map_with_names['centroids'] = map_data['geometry'].to_crs("+proj=cea").centroid.to_crs(epsg='
map_with_names.head(2)
```

	NOM_ENT	geometry	centroids
0	Aguascalientes	POLYGON ((-101.86167 22.02888, -101.86167 22.0...	POINT (-102.36198 22.00611)
1	Baja California	MULTIPOLYGON (((-114.12889 28.01224, -114.1284...	POINT (-115.08685 30.5255)

Luego podemos usar los centroides para colocar los nombres de los estados.

```
ax = map_with_names.plot(figsize=(6, 6), color='lightblue', edgecolor='gray')
ax.set_axis_off()
for idx, row in map_with_names.iterrows():
    plt.annotate(
        text=row['NOM_ENT'],
        xy=(row['centroids'].x, row['centroids'].y),
        horizontalalignment='center',
        fontsize=6,
        color='darkblue'
    )
```

La operación Join

La operación **join** nos permite combinar dataframes de pandas. En geopandas existe el **sjoin** que nos permite hacer intersecciones. Usaremos un nuevo mapa de Red ferroviaria y los que ya teníamos para obtener las estaciones de trenes y redes de vías que se encuentran en Michoacán (el estado número 15 en nuestro dataframe).

Actividad: Mostrar los mapas de otros estados de la república.

```
mich_map = gpd.GeoDataFrame(map_data, index=[15])
mich_trenes = gpd.sjoin(trenes, mich_map)
vias = gpd.read_file("GeoJSON - Red ferroviaria.json")
mich_vias = gpd.sjoin(vias, mich_map)
ax = mich_map.plot(figsize=(6,6), color='none', edgecolor='gray', zorder=3)
mich_vias.plot(ax = ax, zorder=1)
mich_trenes.plot(markersize=20, color='red', ax = ax, zorder=2)
ax.set_axis_off()
```

Proximidad

Se puede visualizar la proximidad de las regiones usando `buffer`s. Con esta función obtenemos una región alrededor de nuestras zonas en `geometry`. En el siguiente ejemplo usamos uun buffer

para marcar las zonas a 10 km de cada estación del tren. Primero pasamos al sistema 6362 que está en metros y luego lo devolvemos al 4326.

```
ax = mich_map.plot(figsize=(6,6), color='none', edgecolor='gray', zorder=3)
buff = mich_trenes['geometry'].to_crs(epsg=6362).buffer(10000)
buff = buff.to_crs(epsg=4326)
buff.plot(color='none', edgecolor='green', ax=ax, zorder=2)
mich_vias.plot(ax = ax, zorder=1)
ax.set_axis_off()
```