



Centro de Investigación
en Computación

IPN – CIC

PROGRAMACIÓN PYTHON CON APLICACIONES EN EL ÁMBITO CIENTÍFICO

Profesor: Alan Badillo

Estrada Hernández Vanessa

Contacto: v.estrada.hdz@gmail.com

Julio, 2023

Introducción

— Planteamiento —

Generar un DataFrame que tenga los siguientes campos:

latitud, longitud, persona_id, partido_preferido, partido_rechazado

Generar al menos 100 datos con ubicaciones reales de México y usar siglas de partidos políticos inexistentes.

A continuación se adjunta un ejemplo de DataFrame para este ejercicio

latitud	longitud	persona_id	partido_preferido	partido_rechazado
19.3204968	-99.1526134	1	MOT	RAP
18.8928205	-98.6017383	2	RAP	MOT
20.643614	-103.423788	3	MAP	MOT
19.914386	-101.5954695	4	PRO	RAP

Figura 1: DataFrame ejemplo

— Motvación —

Se decide trabajar en este problema porque me interesa desarrollarme en *visualización de datos*, por lo que me parece una excelente oportunidad para explorar sobre ello.

Además me parece interesante el que el mapa pueda ser interactivo y su relación con la política, ya que lo que me parece más complicado del Análisis de Datos es el relacionar lo aprendido en las sesiones con problemas o aplicaciones de la vida cotidiana.

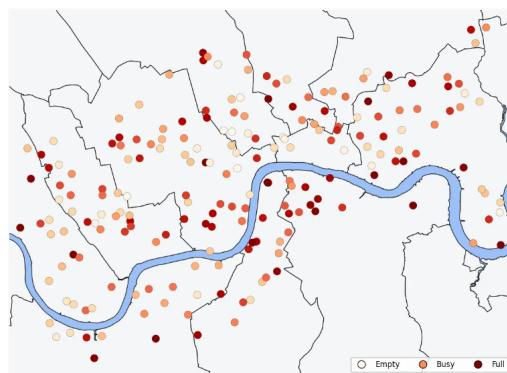


Figura 2: Ejemplo de mapa con puntos Python

Desarrollo

— Preeliminares —

Conociendo librerías

Primeramente se exploran las posibles librerías que se podrían utilizar para el desarrollo de la práctica tales como:

- **geoplot**: Esta librería es ideal para hacer mapas estáticos.

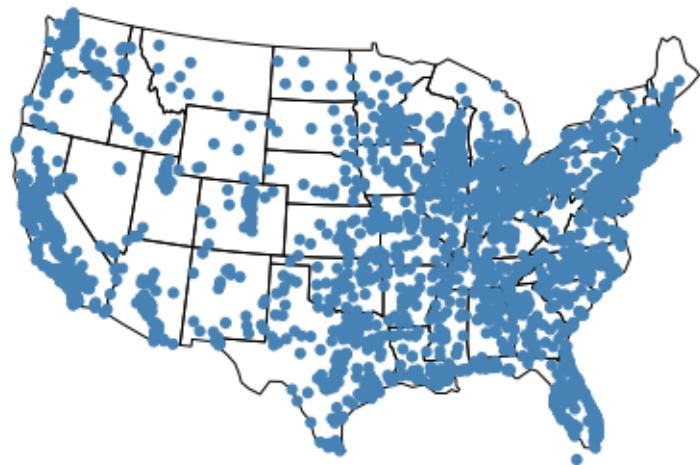


Figura 3: Ejemplo de mapa estático librería geoplot

- **plotly**: Utilizada para crear mapas dinámicos o interactivos.

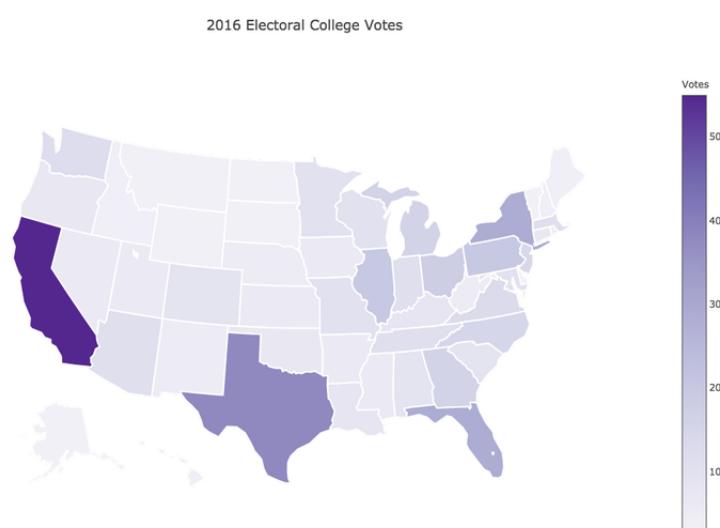


Figura 4: Ejemplo de mapa dinámico con plotly

- **folium:** Para visualizar mapas al estilo google maps.

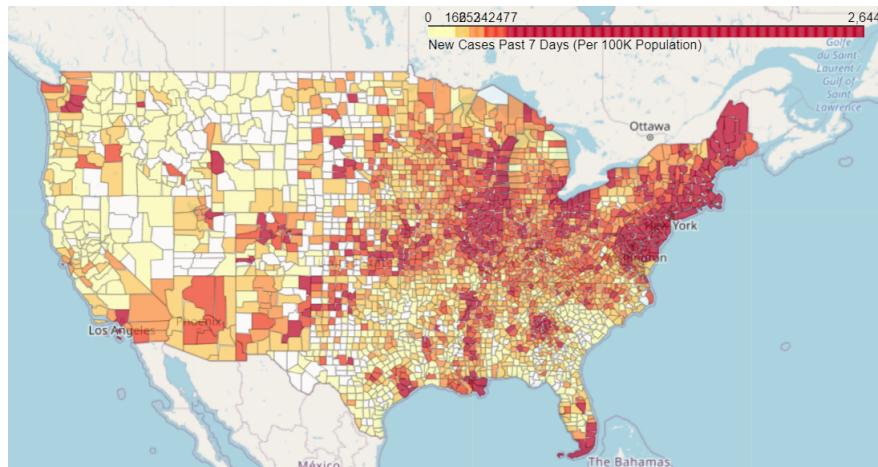


Figura 5: Ejemplo de mapa estilo google maps con folium

Se decide continuar con **folium**, pues como el objetivo es la visualización de datos se opta por la opción con mejor estética.

Recopilar coordenadas reales

Como siguiente paso, se busca algún documento que contenga coordenadas reales de México para poder usarlas y graficarlas en el mapa.

En el portal ”Datos Abiertos México” se localiza un archivo CSV asociado a las ubicaciones, a nivel nacional, de los centros de trabajo de la Secretaría de Educación Pública.

ENTIDAD	MUNICIPIO	COLONIA	LATITUD	LONGITUD
AGUASCALIENTES	AGUASCALIENTES	3 ALTAVISTA	0.911243	102.297222
AGUASCALIENTES	AGUASCALIENTES	268 NINGUNO	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	268 NINGUNO	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	427 VILLA TERESA	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	268 NINGUNO	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	268 NINGUNO	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	135 OJOCALIENTE I	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	58 FERRONALES	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	58 FERRONALES	21.521139	-102.175
AGUASCALIENTES	AGUASCALIENTES	160 PRIMO VERDAD	21.521139	-102.175

Figura 6: Captura de pantalla documento cct

Se filtraron los datos por Estado y se seleccionaron al azar puntos de 6 a 10 puntos de cada uno, con el fin de lograr representación de las 32 entidades federativas que conforman nuestro país.

Seguido de ello, se construyó una nueva tabla en Excel con las columnas especificados en el planteamiento del problema, se decide utilizar 212 registros y 4 partidos políticos.

Con el fin de realizar pruebas, se generó aparte un archivo de Excel con las mismas columnas pero solamente 4 puntos para graficar y 3 partidos políticos.

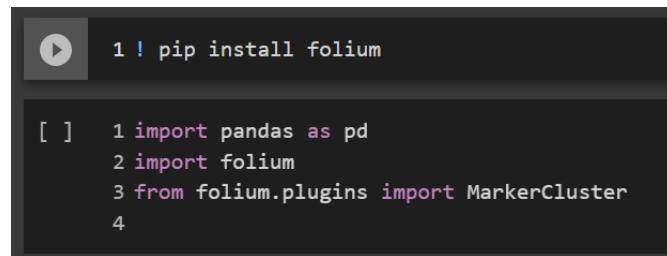
latitud	longitud	persona_id	partido_preferido	partido_rechazado
22.115556	-102.25306	1	MAT	PER
0.916759	102.243333	2	REC	MAT
21.874782	-102.31095	3	MAT	REC
21.896738	-102.27641	4	PER	REC

Figura 7: Captura de pantalla documento de prueba

— Programación —

Instalación e importación de librerías

Con la búsqueda de librerías y documento de prueba listo, comenzamos a programar, instalando e importando las librerías correspondientes.



```
1 ! pip install folium
[ ] 1 import pandas as pd
2 import folium
3 from folium.plugins import MarkerCluster
4
```

Figura 8: Instalación e importación de librerías

Comenzamos con graficar el mapa de México con la nueva librería, para ello necesitamos las coordenadas correspondientes que son:

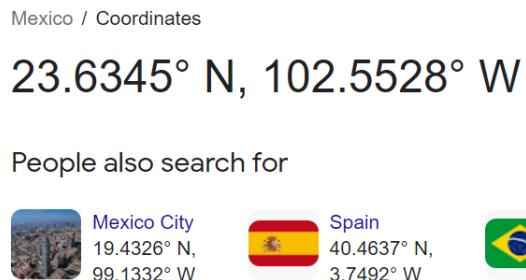


Figura 9: Resultado de búsqueda 'Coordendadas México'

Visualizar mapa de México con folium

Con ayuda del atributo **folium.Map()** y las coordenadas obtenida en Google graficamos el mapa de México, ajustando el zoom de inicio a 5.3.

Si no se ajusta el zoom de inicio, folium nos muestra una localidad en México ubicada en Zacatecas, seg\xfcre las coordenadas especificadas, como se muestra en la figura siguiente.



Figura 10: Resultado al no especificar el zoom de inicio

Con el atributo **zoom start** se visualiza por completo el mapa de la Rep\xfablica Mexicana.

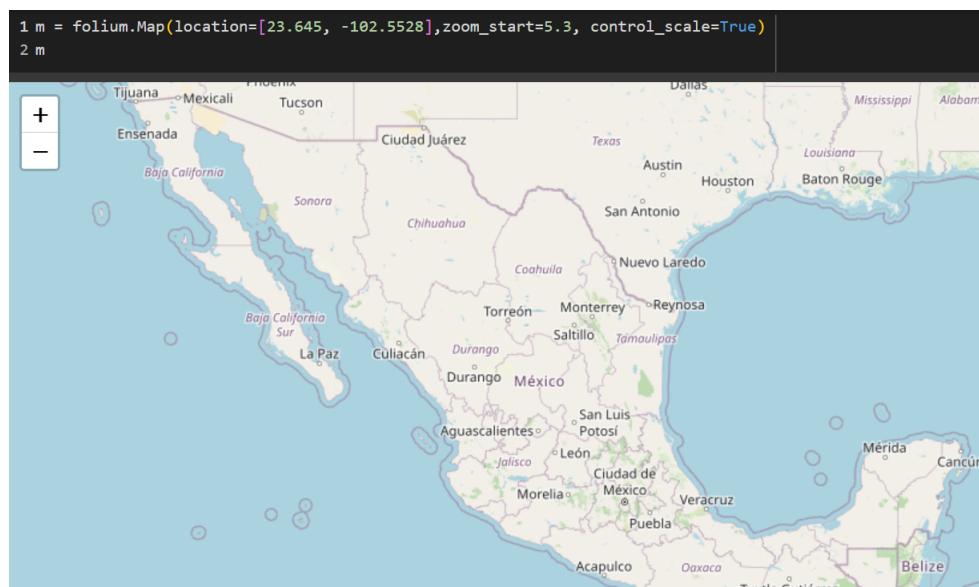


Figura 11: Resultado al especificar zoom de inicio

Graficar puntos con folium

Ahora que ya se visualiza el mapa a la escala que queremos, resta graficar los datos que obtendremos del archivo CSV generado con anterioridad.

Subimos nuestro archivo de prueba a la carpeta para poder hacer uso de él.



Figura 12: Archivo de prueba guardado en la carpeta sample data

Creamos un DataFrame llamado **data** para guardar los datos del archivo CSV llamado **Pruebaaa** e imprimimos el DF para corroborar que se haya guardado correctamente.

```

1 data=pd.read_csv("/content/sample_data/Pruebaaa.csv")
2 data

      latitud  longitud persona_id partido_preferido partido_rechazado
0  22.115556 -102.253056          1           MAT            PER
1   0.916759   102.243333          2           REC            MAT
2  21.874782 -102.310947          3           MAT            REC
3  21.777253 -102.276410          4           REC            PER

```

Figura 13: Archivo CSV guardado en DataFrame correctamente

Convertimos las columnas del DataFrame en listas para poder manipular la información con facilidad.

```

1 latitudes=data['latitud'].values.tolist()
2 longitudes=data['longitud'].values.tolist()
3 preferidos= data['partido_preferido'].values.tolist()
4 rechazados=data["partido_rechazado"].values.tolist()

```

Figura 14: Listas de cada columna del df

Para cada partido se utilizó el color *claro* para denotar si el partido era **preferido** y el color *oscuro* si era **rechazado**

- **MAT:** Azul claro y marino.
- **REC:** Morado y violeta.
- **PER:** Verde claro y fuerte.
- **PVU:** Naranja y beige.

Creamos un **feature group** en folium para agrupar las etiquetas de los partidos políticos y poder agregarlas al mapa.

```
1 feature_group=folium.FeatureGroup(name='PartidosPolíticos')
```

Figura 15: Creación de Feature Group

Para el siguiente paso creamos un ciclo **for** que recorre las listas **latitudes**, **longitudes**,**preferidos** y **rechazados** declaradas anteriormente.

```
for lat,lon,tagp,tagr in zip(latitudes,longitudes,preferidos,rechazados):
```

Figura 16: Ciclo **for** para agregar los puntos

Dentro del ciclo, se declaran condicionales para filtrar el nombre de partido político y si este corresponde a **rechazados** o **preferidos** para poder seleccionar el color del círculo a graficar.

Se especifican los siguientes parámetros de la función **folium.Circle()**:

- **location:** Se ingresan las coordenadas del punto.
- **color:** Para seleccionar el color de la circunferencia.
- **fill_color:** Selecciona el color de relleno del círculo.
- **fill_opacity:** Para la opacidad del relleno.
- **radius:** Especificar el radio del círculo, para nuestro caso todos son iguales.

Para evitar que se empalmen los círculos graficados, movemos las coordenadas en **+0.001** cuando se grafica el partido rechazado.

Por último, se utiliza el atributo **.add_to()** para agregar cada punto al **featureGroup** que creamos anteriormente.

```
1 for lat,lon,tagp,tagr in zip(latitudes,longitudes,preferidos,rechazados):
2     fLat=float(lat)
3     fLon=float(lon)
4     if tagp=="MAT":
5         folium.Circle(location=[fLat,fLon],color='lightblue',fill_color='lightblue' , fill_opacity=1,weight=4,radius=400).add_to(feature_group)
6     elif tagp=="REC":
7         folium.Circle(location=[fLat,fLon],color='violet', fill_color='violet', fill_opacity=1,weight=4,radius=400).add_to(feature_group)
8     elif tagp=="PER":
9         folium.Circle(location=[fLat,fLon],color='lightgreen',fill_color='lightgreen',fill_opacity=1,weight=4,radius=400).add_to(feature_group)
10    elif tagp=="PVI":
11        folium.Circle(location=[fLat,fLon],color='lightred',fill_color='lightred',fill_opacity=1,weight=4,radius=400).add_to(feature_group)
12    if tagr=="MAT":
13        folium.Circle(location=[fLat+0.001,fLon+0.001],color='navy',fill_color='navy',weight=4,fill_opacity=1,radius=400).add_to(feature_group)
14    elif tagr=="REC":
15        folium.Circle(location=[fLat+0.001,fLon+0.001],color='purple',fill_color='purple',weight=4,fill_opacity=1,radius=400).add_to(feature_group)
16    elif tagr=="PER":
17        folium.Circle(location=[fLat+0.001,fLon+0.001],color='darkgreen' , fill_color='darkgreen' , fill_opacity=1,weight=4,radius=400).add_to(
18    elif tagr=="PVI":
19        folium.Circle(location=[fLat,fLon],color='orange',fill_color='orange',fill_opacity=1,weight=4,radius=400).add_to(feature_group)
```

Figura 17: Bloque de código para agregar cada punto

Para mostrar el mapa colocamos en una celda nueva la función `m.add_child`

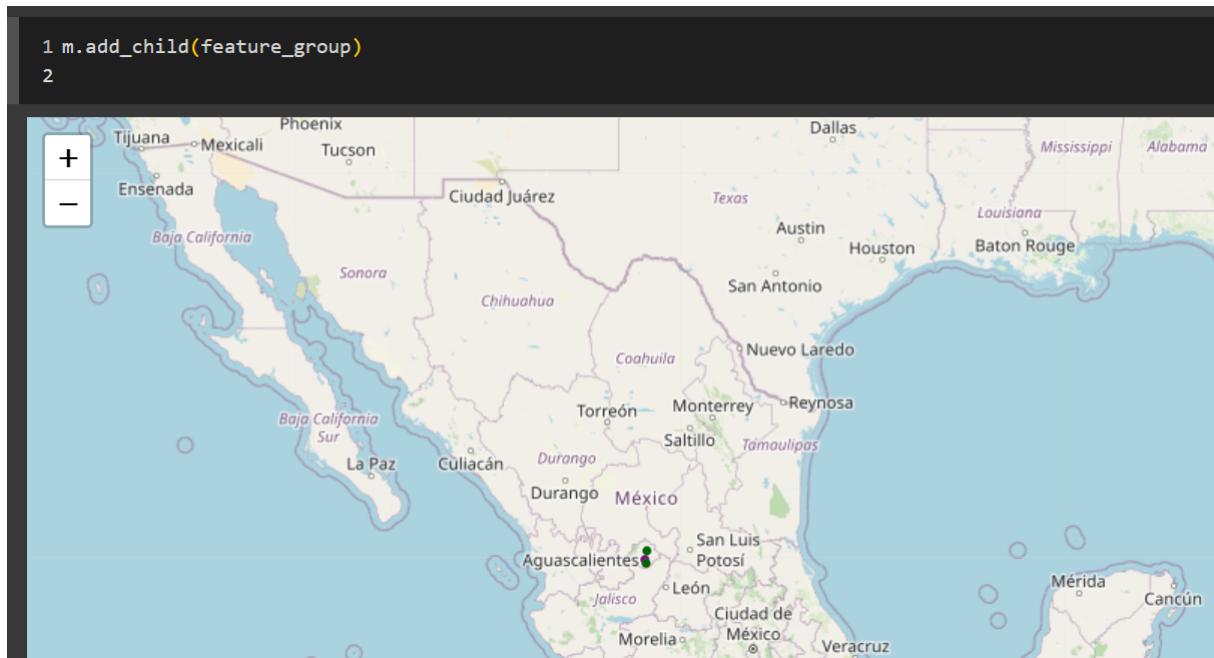


Figura 18: Mapa de México con puntos graficados

Se observan los puntos graficados cerca de Aguascalientes

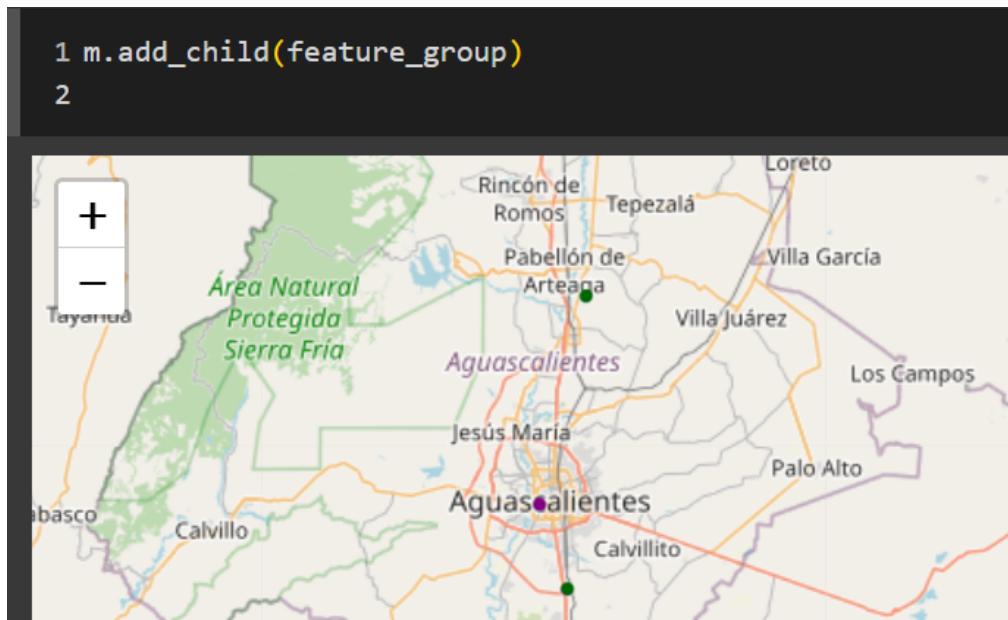


Figura 19: Mapa en zoom con los puntos de prueba

Resultados

Al quedar lista nuestra prueba, podemos proceder a subir el archivo CSV con los 212 registros que se pretende graficar, añadiendo además un partido político.

Se imprimen 10 registros en aleatorio para comprobar que el archivo se haya subido y guardado correctamente.

```
1 data=pd.read_csv("/content/sample_data/Prueba.csv")
2 data.sample(5)
```

	latitud	longitud	persona_id	partido_preferido	partido_rechazado
60	21.267924	-99.719622	61	PER	MAT
200	21.004550	-89.597674	201	MAT	PER
148	21.262700	-98.791500	149	REC	PER
181	22.276389	-97.831389	182	MAT	PER
165	27.080833	-109.445278	166	REC	PER

Figura 20: Impresión de 10 registros en aleatorio

Graficando los registros, el mapa resulta



Figura 21: Mapa de México con ubicaciones de los 212 registros

Al hacer zoom en uno de los puntos graficados, se observa la separación de ambos colores

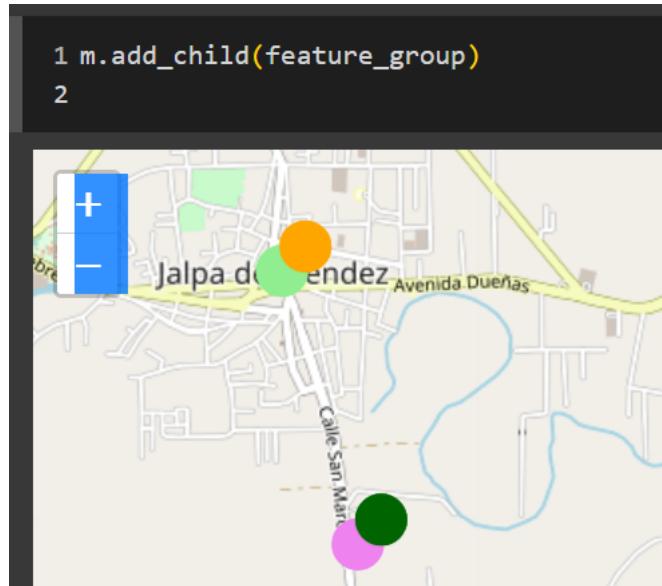


Figura 22: Visualización en zoom de uno de los registros

Lo anterior puede interpretarse como sigue:

Persona en la parte superior (naranja/verde claro)

- **Partido Preferido:** PER
- **Partido Rechazado:** PVU

Persona en la parte inferior (verde fuerte/ violeta)

- **Partido Preferido:** REC
- **Partido Rechazado:** PER

— Extra —

Además de esta actividad me llamó la atención la relacionada a hacer una **Gráfica de Red** y su interpretación, por lo que quisiera realizar la conjunción de ambas.

Como me gustaría complementar la actividad realizada, uniré el DataFrame relacionado a sus preferencias políticas con una tabla nueva que contiene las columnas *persona_id*, *edad*, *trabaja*, *casado*, *fuma*, *bebe*, *deporte*.

Unir dos DataFrame

Se crea la tabla de **rasgos psicológicos**

persona_id	edad	trabaja	casado	fuma	bebe	deporte
1	53	NO	SI	SI	NO	SI
2	18	SI	NO	SI	SI	SI
3	23	NO	NO	SI	SI	NO
4	62	NO	NO	NO	SI	NO
5	74	NO	SI	NO	NO	SI
6	45	SI	NO	SI	NO	NO

Figura 23: Captura de algunos registros del archivo rasgos psicológicos”

Trabajaremos sobre un nuevo **notebook** por lo que subimos ambas tablas y creamos sus respectivos DataFrame.

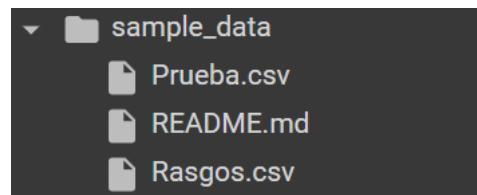


Figura 24: Ubicación de archivos CSV a emplear

Imprimimos 5 elementos aleatorios del DF sobre rasgos psicológicos y preferencias políticas para comprobar que se subió y guardó correctamente.

```

1 data1=pd.read_csv("/content/sample_data/Rasgos.csv")
2 data1.sample(5)

  persona_id  edad  trabaja  casado  fuma  bebe  deporte
0         94    95      SI     NO    SI     SI      SI
1        114   115      SI     NO    SI     SI      SI
2         75    76      NO     SI    SI     NO      SI
3         22    23      NO     NO    SI     SI      SI
4         53    54      SI     SI    SI     SI     NO

```

Figura 25: Registros al azar del DataFrame sobre rasgos psicológicos

```
1 data2=pd.read_csv("/content/sample_data/Prueba.csv")
2 data2.sample(5)
```

	latitud	longitud	persona_id	partido_preferido	partido_rechazado
55	25.176101	-104.559276	56	MAT	REC
29	28.650321	-106.104367	30	PER	MAT
119	17.806800	-97.776200	120	PER	MAT
44	18.910500	-103.874000	45	PER	MAT
56	25.049513	-105.421265	57	PVU	REC

Figura 26: Registros al azar del DataFrame sobre preferencias políticas

Unimos ambas tablas mediante la llave **persona_id** con la función **pandas.merge()** e imprimimos 5 elementos en aleatorio para observar la unión.

```
1 data=pd.merge(data1,data2,on="persona_id")
2 data.sample(4)
```

	persona_id	edad	trabaja	casado	fuma	bebe	deporte	latitud	longitud	partido_preferido	partido_rechazado
6	7	65	SI	NO	SI	SI	NO	32.6320	-115.470	MAT	PER
130	131	30	NO	NO	SI	NO	NO	20.5934	-100.390	MAT	PVU
43	44	69	NO	SI	NO	NO	NO	19.0532	-104.316	PVU	REC
46	47	69	NO	NO	SI	NO	NO	19.3288	-103.602	PER	MAT

Figura 27: Resultado de la unión de ambos DataFrame

Gráfica de Red

Antes de comenzar a programar, para familiarizarnos con la terminología utilizada en gráficas de red realizamos una pequeña lectura en internet y se visualizan ejemplos de este tipo de gráficas y su uso.

Terminología:

- **node:** Se traduce como *nodo*, es el elemento principal de estas gráficas, también se conocen como vértices.
- **edges:** Traducido como *aristas* y son las conexiones entre dos nodos.
- **undirected graph:** Gráfica sin direcciones, es decir, las aristas son bidireccionales

De igual forma como lo hicimos con la actividad pasada, comenzemos con pocos registros para hacer pruebas, por lo que tomamos 5 registros aleatorios.

```
1 data_prueba=data.sample(5)
```

	persona_id	edad	trabaja	casado	fuma	bebé	deporte	latitud	longitud	partido_preferido	partido_rechazado
169	170	47	SI	NO	NO	SI	NO	17.591800	-92.826300	REC	PER
198	199	43	NO	NO	SI	NO	NO	20.995789	-89.589889	MAT	REC
21	22	57	NO	SI	NO	NO	SI	17.514102	-93.118376	MAT	REC
119	120	28	SI	NO	NO	SI	SI	17.806800	-97.776200	PER	MAT
117	118	50	NO	SI	SI	NO	NO	18.081200	-96.118500	PVU	MAT

Figura 28: Registros al azar del DataFrame unido

Para realizar la gráfica de red se emplea la librería **NetworkX**, así que se importa.

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3 from matplotlib.pyplot import figure
```

Figura 29: Librerías que fueron importadas

Cada columna del DataFrame la convertimos en una lista para poder manipularla con mayor facilidad

```
1 idpersona=data_prueba["persona_id"].values.tolist()
2 fuma=data_prueba["fuma"].values.tolist()
3 bebe=data_prueba["bebé"].values.tolist()
4 deporte=data_prueba['deporte'].values.tolist()
5 trabaja=data_prueba['trabaja'].values.tolist()
```

Figura 30: Listas de cada columna del df

Creamos un objeto del tipo **Graph**, que nombramos **G**.

Para identificar las relaciones entre fumadores o no fumadores, creamos dos listas vacías para separarlos por categoría:

- **s[]**: Fumadores (sí)
- **n[]**: No fumadores

Después, se utiliza un ciclo **for** que recorre los nodos y los categoriza en fumadores o no.

```

1 G=nx.Graph()
2 #Crear listas vacías para saber si fuma o no fuma
3 s=[]
4 n=[]
5 for nodo,fum in zip(idpersona,fuma):
6     if fum=='SI':
7         s.append(nodo)
8     elif fum=='NO':
9         n.append(nodo)
10 print("Sí:",s,"No:",n)

Sí: [128, 158, 7] No: [179, 16]

```

Figura 31: Código del ciclo **for** que categoriza

Podemos observar en la parte inferior de la imagen anterior que se muestran los resultados para ambas listas, lo que significa que los nodos

- 128,158 y 7 **fuman**
- 179 y 16 **no fuman**

Recorremos la lista **s[]** que contiene los **id** de las personas que fuman, se agregan aristas al objeto **G** utilizando la función **G.add_edge()**. No es necesario crear los nodos porque cuando agregamos las uniones entre ellos se guardan por defecto.

```

1 for i in s:
2     for j in s:
3         if i!=j:
4             G.add_edge(i,j)

```

Figura 32: Ciclo **for** para crear las aristas de *fumadores*

Mismo procedimiento para los no fumadores

```

for i in n:
    for j in n:
        if i!=j:
            G.add_edge(i,j)

```

Figura 33: Ciclo **for** para crear las aristas de *no fumadores*

Por último, se imprime la gráfica y activamos la opción de agregar etiquetas a los nodos para poder identificarlos.

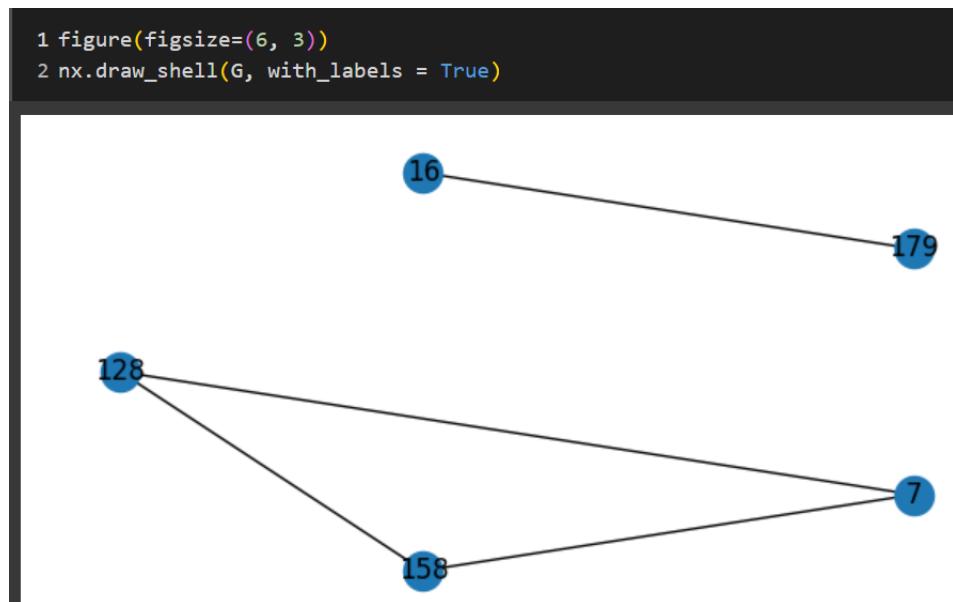


Figura 34: Gráfica de red para la columna **fuma**

Ahora utilicemos 50 registros, como lo solicita el problema.

	persona_id	edad	trabaja	casado	fuma	bebé	deporte	latitud	longitud	partido_preferido	partido_rechazado
72	73	33	SI	NO	NO	SI	SI	20.080250	-98.777660	MAT	REC
177	178	18	NO	NO	NO	SI	NO	23.722757	-99.162526	MAT	PVU
185	186	35	SI	NO	NO	NO	NO	19.299889	-98.238897	MAT	REC
50	51	40	NO	SI	NO	NO	SI	23.979211	-104.675435	MAT	REC
61	62	26	NO	NO	NO	NO	SI	20.540442	-100.841856	MAT	PVU
38	39	65	SI	NO	SI	NO	SI	25.923700	-103.135000	PER	MAT
134	135	70	SI	NO	NO	NO	NO	20.386400	-99.999600	MAT	PER
104	105	67	NO	NO	SI	NO	SI	21.808300	-105.206000	PVU	REC
39	40	74	SI	NO	SI	SI	SI	25.724000	-103.155000	MAT	REC
172	173	28	NO	SI	SI	SI	SI	17.758675	-92.598295	MAT	REC
91	92	59	SI	NO	SI	NO	SI	19.533600	-100.263400	REC	MAT

Figura 35: Impresión del DataFrame con 50 registros aleatorios

Gráficamente, los nodos se representan como sigue

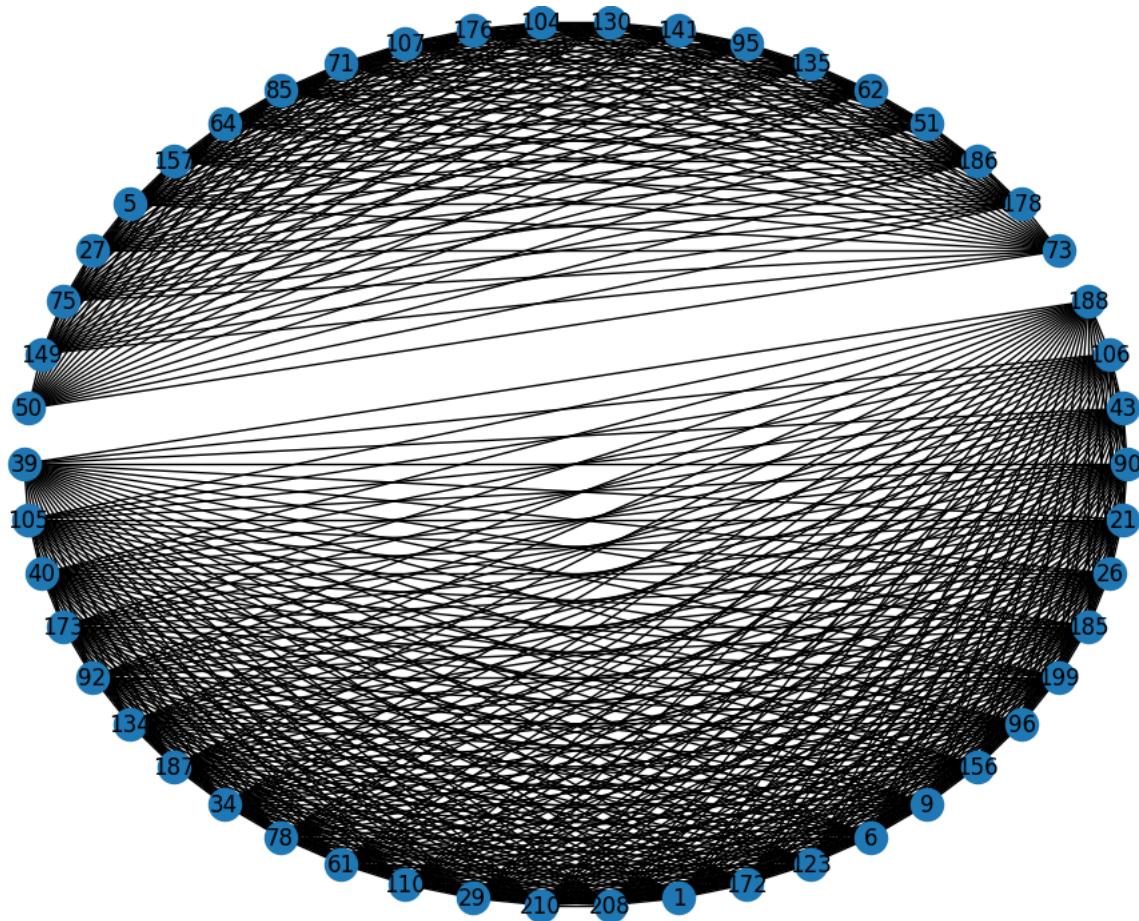


Figura 36: Gráfica de red que representa a los fumadores y no fumadores

Se visualiza claramente la separación entre los no fumadores (parte superior de la figura) y las personas que fuman (parte inferior).

Por otro lado, para la categoría **"bebé"**, realizamos el mismo procedimiento, por lo que el código se ve como sigue.

```
1 G=nx.Graph()
2 #Crear listas vacías para saber si fuma o no fuma
3 s=[]
4 n=[]
5 for nodo,beb in zip(idpersona,bebé):
6     if beb=='SI':
7         s.append(nodo)
8     elif beb=='NO':
9         n.append(nodo)
```

Figura 37: Ciclo **for** para categorizar si alguien bebe o no

Gráficamente, resulta:

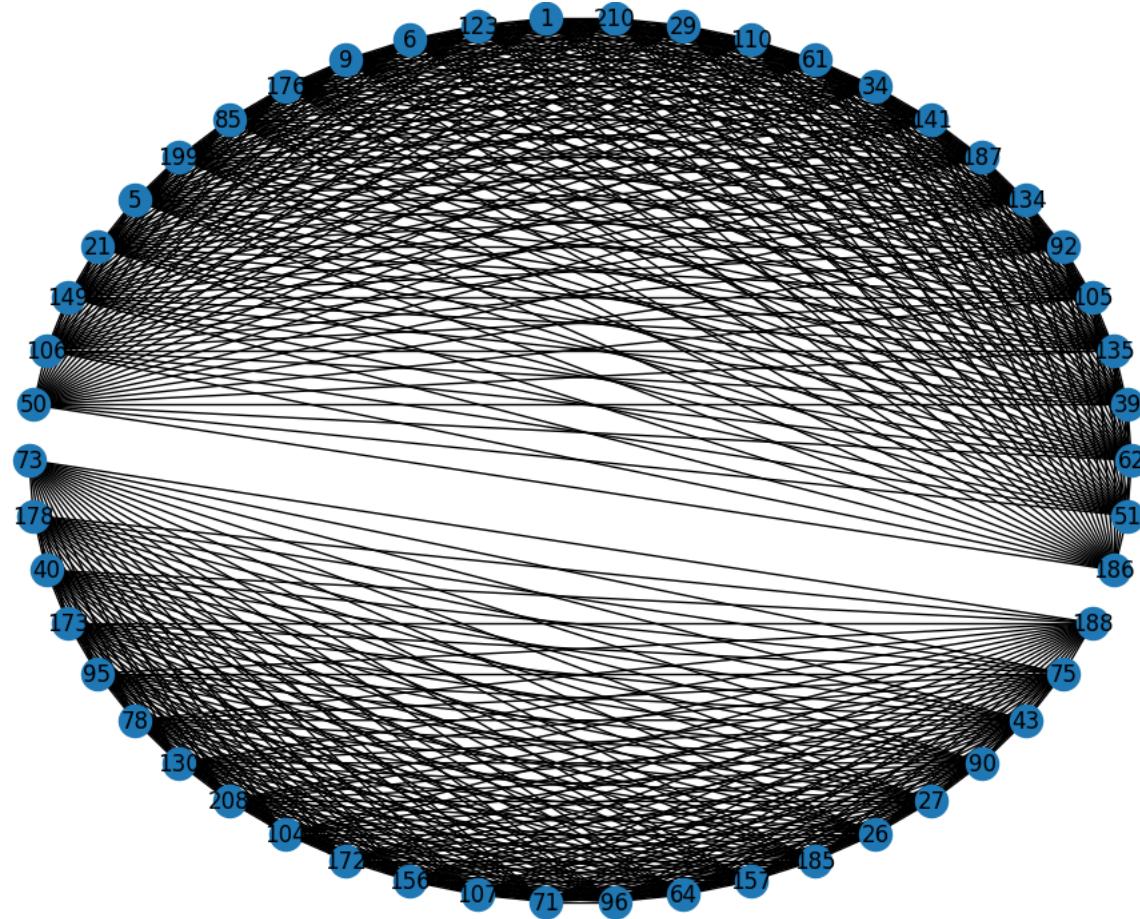


Figura 38: Gráfica de red que representa a las personas que beben y a las que no

De igual forma podemos ver la separación entre las personas que beben (parte inferior) y las que no lo hacen (parte superior).

Partido Político de preferencia

Ahora queremos visualizar la información referente al **partido político de preferencia** para la muestra de los 50 registros aleatorios.

Comenzamos definiendo listas vacías para guardar los **id** de las personas que tienen como preferido a cada partido político

```

1 graph=nx.Graph()
2 PVU=[]
3 MAT=[]
4 REC=[]
5 PER=[]

```

Figura 39: Listas vacías de cada partido político

Mediante un ciclo **for** recorremos los nodos con su respectiva preferencia política y se clasifican en las listas declaradas en el paso anterior.

```

6 for nodo,pref in zip(idpersona,prefiere):
7     if pref=='PVU':
8         PVU.append(nodo)
9     elif pref=='MAT':
10        MAT.append(nodo)
11    elif pref=='REC':
12        REC.append(nodo)
13    elif pref=='PER':
14        PER.append(nodo)

```

Figura 40: Ciclo **for** para categorizar las preferencias políticas

Por último, realizamos un ciclo **for** para cada partido, como lo hicimos con las listas que guardaban los nodos que fumaban o bebían respectivamente.

La figura que resulta al graficar los nodos de preferencia de partidos políticos es:

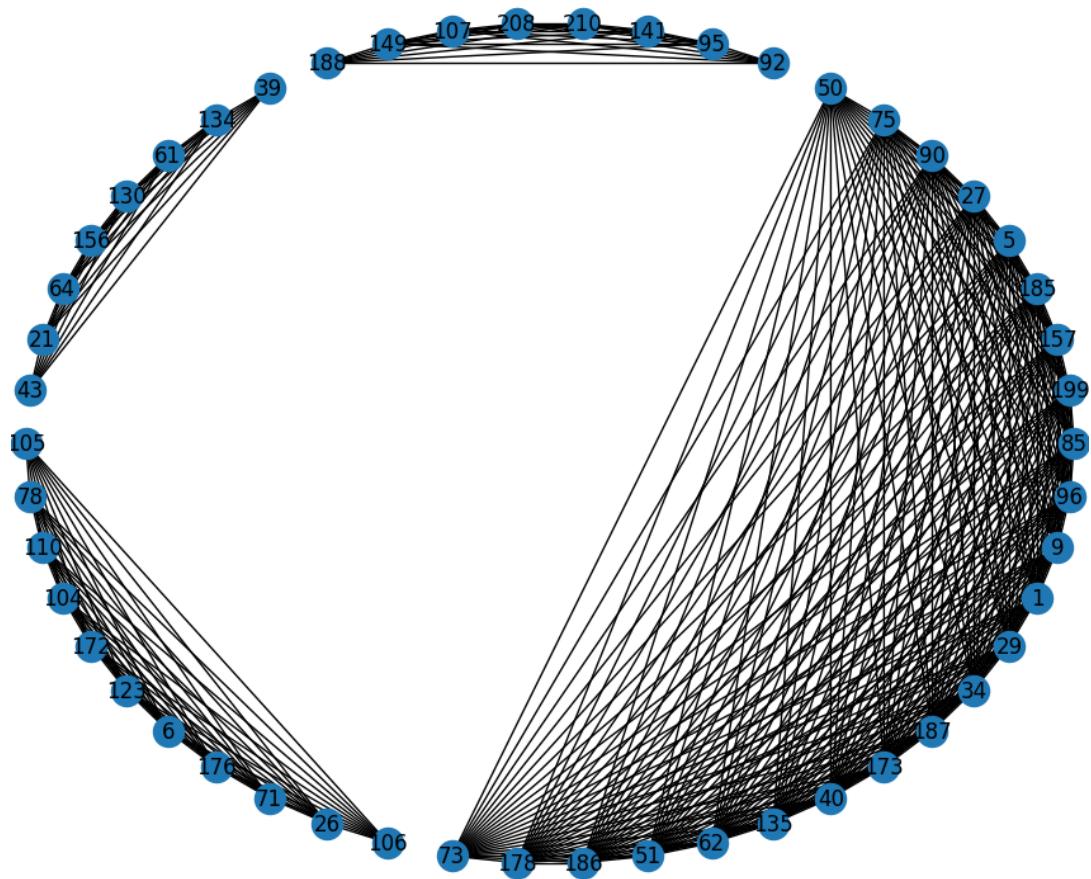


Figura 41: Gráfica de red que representa las preferencias de partido

— Conclusiones —

Como mejora para el **gráfico del mapa**, me gustaría crear un **FeatureGroup** para cada una de las clasificaciones (rechazado o preferido) y además poder seleccionar si se quiere visualizar alguna categoría o ambas con *checkbox* como se muestra en el ejemplo:

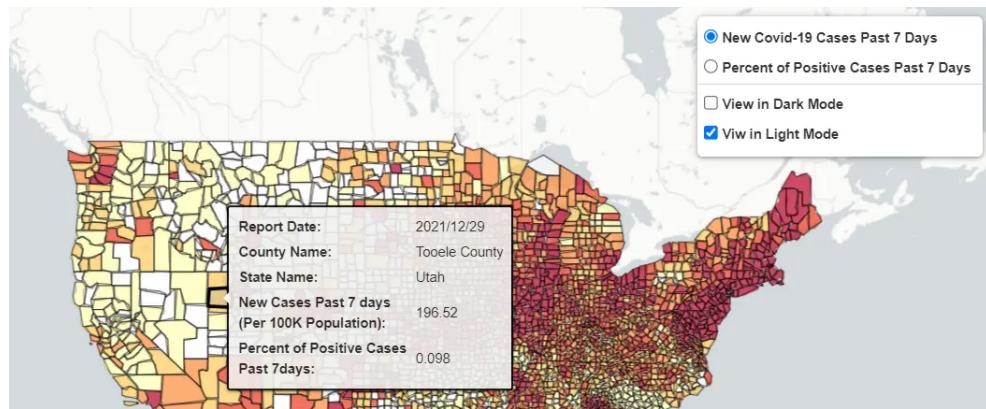


Figura 42: Ejemplo de mapa con checkbox

Para la **gráfica de red** me hubiera gustado hacer análisis en con dos ejes, por ejemplo *"partido político de preferencia y si la persona es casada* o hacer la gráfica que conllevaba hacer análisis de niveles.

Estéticamente hablando, me hubiera gustado poder cambiar el color y acomodo de los nodos, o realizar una conjunción entre **"heatmap"** y la gráfica de red.

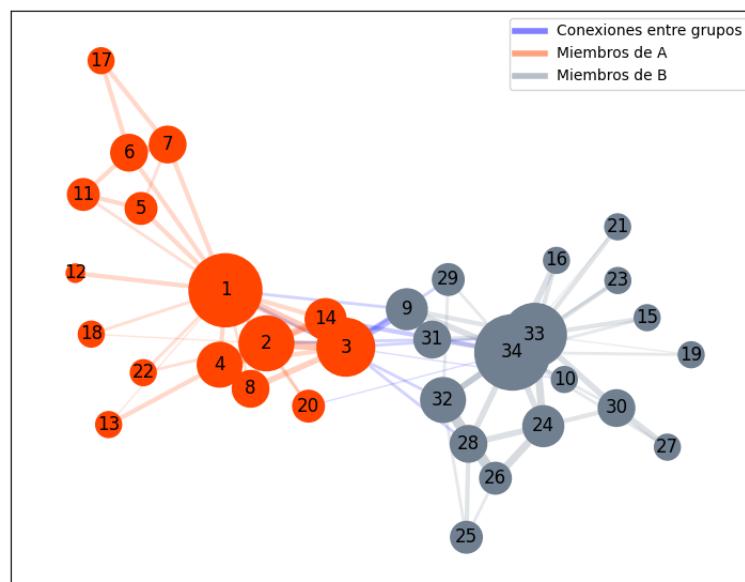


Figura 43: Ejemplo de **network** con colores