Segunda Entrega Proyecto COVID 19 - COLOMBIA

Daniel Julian Siachoque Peralta
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda
Bogotá, Colombia
daniel.siachoque01@correo.usa.edu.co

Juan Guillermo Torres Delgado
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda
Bogotá, Colombia
juan.torres01@.correo.usa.edu.co

Carlos Antonio Plaza Amado
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda
Bogotá, Colombia
carlos.plaza01@correo.usa.edu.co

Resumen

Durante el proyecto se habló sobre la actual pandemia "Coronavirus" o "covid19", enfocándonos en los datos actualizados para Colombia y especificamente Bogota. Para esto fue necesario una pagina que nos brindó en tiempo real los datos de la pandemia, así mismo, se utilizó el Python como lenguaje de programación, sql como método de extracción de información a través de una base de datos y Látex como método de escritura para presentar los datos. El objetivo principal del proyecto se basa en plasmar mediante gráficas de barras, tortas, 2 dimensiones y mapas de calor los diferentes datos que se presentan por localidades, haciendo enfasis en la discriminación por edades, estado y genero.

Palabras cláve:

Bases de datos sql, Coronavirus, Mapa de calor, Python.

1. Marco teórico

1.1. Covid 19

Es una enfermedad que se descubre por primera vez en Wuhan-China, la cual se ha esparcido por todas las áreas del mundo, llegando así a identificarse casos en cada uno de los continentes. El primer caso confirmado en Colombia se dio a conocer el 6 de marzo del presente 2020.

Esta enfermedad produce una infección Respiratoria Aguda o lo que conocemos como una gripe, pero variando desde ser grave, moderada o simplemente leve. Algunos de sus síntomas parten de la fiebre, tos seca y cansancio, sin dejar de lado dolores o molestias generales. Cabe recalcar que un 80 % de las personas infectadas por el virus se logran recuperar sin un tratamiento hospitalario, teniendo una mortalidad baja respecto a otras enfermedades.

Algunas recomendaciones generales para evitar el contagio de esta enfermedad parte del lavado constante de las manos y el uso constante de la mascarilla junto con gel antibacterial.

En el momento de este proyecto, se tiene conocimiento de 33 Millones de casos a nivel mundial, de los cuales Estados Unidos, India, Brasil, Rusia y Colombia se encuentran en el top 5 de países con más casos registrados (7 Millones, 6 Millones, 4 Millones, 1 Millón y 800 Mil) respectivamente.

En Colombia, como ya se comentó anteriormente, se tiene conocimiento de más de 800 Mil casos confirmados, un número de muertes aproximadas a 25 mil y un aproximado de 5 Mil casos diarios. Datos con los cuales se trabajó durante este proyecto para graficar posteriormente.

Dicho virus tiene al mundo actual limitado en sus numerosos campos de acción, desde el campo social hasta el económico, pausando las economías a nivel mundial y produciendo nuevas alternativas a las acciones de la vida cotidiana. [1].

1.2. Bases de datos

Se entiende como el conjunto de datos los cuales se almacenan en un mismo lugar con los cuales se trabajara a corto o largo plazo. Compuestas de columnas y filas que forman múltiples registros.

Algunas ventajas notables a la hora de usar dichas bases, son la posibilidad de guardar cantidades inmensas de información, encontrarla y utilizarla de una manera práctica, ordenada y eficaz.

Con el paso del tiempo, como la mayoría de cosas en nuestra actualidad, se busca una optimización de nuestros recursos con el uso de estas bases, dando seguridad, compatibilidad máxima, la menor cantidad de redundancia y respaldo a nuestros datos. [2].

Se encuentran diferentes tipos de bases de datos como lo son:

1. SQL: También llamada como bases de datos relacionales. Su lenguaje se basa en consultas estructuradas de los datos. Dichas bases son escalables verticalmente a través de mejoras de hardware en su servidor y en cuanto a su estructura, se encuentra que están basadas en tablas.
2. NOSQL: Son bases conocidas por tener un esquema dinámico para datos no estructurados. Tiene una escalabilidad horizontal, lo cual se entiende como un mayor tráfico de fragmentación de datos. Usualmente su estructura está basada en pares clave-valor, gráficas y columnas.

1.3. Python

Es un lenguaje de programación de código abierto, orientado a objetos y con múltiples bibliotecas de herramientas al uso de todos.Python permite ejecutar el código sin necesidad de compilar nuestro código, dando agilidad durante su uso. Python posee la característica principal de multiplataforma, así mismo, posee una notación identidad, la cual trabaja con tabulación en el código de sus respectivas funciones o bucles, permitiendo una mayor organización y optimización del proyecto. Por otro lado encontramos las librerías, que son un código prediseñado que nos ofrece un sin número de funciones a nuestros proyectos. Para el desarrollo de este proyecto fue necesario el uso de librerías como: , Requests, Json.

1. Matplotlib: Se encarga del trazado 2D y 3D, produciendo diferentes gráficos, histogramas, espectros de potencia, gráficos de barras, gráficos de error, gráficos de dispersión, etc., con solo unas pocas líneas de código. [3].
2. Requests: Permite enviar solicitudes HTTP via python, entendiéndose como “Web Scraping” o simplemente como la recoleccion de informacion web. [4].
3. Json: Se encarga de parsear el JSON de archivos o strings, convirtiendo los datos y listas de python en cadenas JSON. [5].
4. Seaborn: Esta libreria se encarga de personalizar el estilo de las graficas que realizamos en nuestro documento, tiene una paleta de visualizacion diferente a la que viene por defecto, asi mismo, permite una gran libertad de manejo de graficos.
5. NumPy: Dicha libreria pretende dar soporte hacia la creacion de vectores y matrices multidimencionales, dando una cantidad de funciones matematicas de alto nivel para sus operaciones.

Para el uso de algunas de estas librerías, es necesario realizar su instalación correspondiente por medio de las siguientes líneas de comando. [6].

```
pip3 install requests
pip3 install json
pip3 install matplotlib
pip3 install seaborn
```

Figura 1: Librerias Python

1.4. Mapas de Calor

Nacen en 1991 con fin de visualizacion 2D de los valores en una matriz. Se entienden como graficas que nos permiten representar y entender los datos a través de una termografia, representada principalmente por dos polos, entendiendo que, un polo indica poca cantidad y el otro una mayor cantidad de datos. [7].

Algunos tipos de mapa de calor son:

1. Biologicos: Son aquellos encargados de representar el nivel de expresión de muchos genes en una serie de muestras comparables a través de microarrays de ADN.
2. Mapa de Árbol: Es una particion jerarquica de datos 2D.
3. Mosaico: Se entiende como el mapa que representa la tabla de datos bidireccionales o superior.

2. Resultados

Recordamos que el lenguaje que se utilizó durante este proyecto fue python y que se deben instalar las librerías previamentes mencionadas en la sección 1.3 Inicialmente se procede a extraer la información de la página web, por medio de sentencias sql para su debido almacenamiento.

```
import requests
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sb

ListaValor=[]
ListaLocalidad=[]
Localidad=[]
Casos=[]
Tiempo=[]

urlDatos = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
urlDatosSQL = 'sql=SELECT "Localidad de residencia", count(*) as Cantidad from ...
    "b64ba3c4-9e41-41b8-b3fd-2da21d627558" GROUP BY "Localidad de residencia" ORDER BY ...
    "Localidad de residencia"'

req = requests.get(url=urlDatos+urlDatosSQL)
reqJson = req.json() #Organizar datos
Datos = reqJson['result']['records']
i=0
```

Listing 1: Código de obtención datos COVID-19

La primera gráfica que se realizó, fue la de barras de forma horizontal, teniendo el número de casos en el eje x y la localidad en el eje y.

```
#Grafica de barras

  for fila in Datos:
  ListaValor.append(int(fila["cantidad"]))
  ListaLocalidad.append(fila["Localidad de residencia"])
  i = i + 1

  plt.rcdefaults()
  fig, ax = plt.subplots(figsize=(11, 5))

  y_pos = np.arange(len(ListaLocalidad))

  ax.barh(y_pos, ListaValor, align='center')
  ax.set_yticks(y_pos)
  ax.set_yticklabels(ListaLocalidad)
  ax.invert_yaxis()
  ax.set_xlabel('N mero de contagiados')
  ax.set_ylabel('Localidades')
  ax.set_title('Contagios Covid-19 Bogot ')

  plt.grid()
```

Listing 2: Gráfica de barras

A continuación se procedió con realizar dos gráficas de torta y mostrar los códigos correspondientes.

```
#Graficas tortas

labels = ListaLocalidad
sizes = ListaValor

fig1, ax1 = plt.subplots(figsize=(10,7))
ax1.pie(sizes, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
ax.set_title("Contagios Covid-19 Bogot ")

fig, ax = plt.subplots(figsize=(15, 10), subplot_kw=dict(aspect="equal"))
recipe = ListaLocalidad
data = ListaValor
wedges, texts = ax.pie(data, wedgeprops=dict(width=0.5), startangle=-40)

bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
```

```
kw = dict(arrowprops=dict(arrowstyle="-"),
          bbox=bbox_props, zorder=0, va="center")

for i, p in enumerate(wedges):
    ang = (p.theta2 - p.theta1)/2. + p.theta1
    y = np.sin(np.deg2rad(ang))
    x = np.cos(np.deg2rad(ang))
    horizontalalignment = {-1: "right", 1: "left"}[int(np.sign(x))]
    connectionstyle = "angle,angleA=0,angleB={}".format(ang)
    kw["arrowprops"].update({"connectionstyle": connectionstyle})
    ax.annotate(recipe[i], xy=(x, y), xytext=(1.35*np.sign(x), 1.4*y),
                horizontalalignment=horizontalalignment, **kw)

ax.set_title("Contagios Covid-19 Bogot ")
```

Listing 3: Gráficas de torta

Seguida a esta, se graficó en dos dimensiones.

```

#Grafica 2 dimensiones

  t = np.arange(0, 22)
  s = np.array(ListaValor)

  fig, ax = plt.subplots()
  ax.plot(t, s)

  ax.grid(True, linestyle='dashed')
  ax.tick_params(labelcolor='r', labelsize='medium', width=3)
  ax.set_title("Contagios Covid-19 Bogot ")
  ax.set_xlabel('Numero de Localidades')
  ax.set_ylabel('Contagiados')

  plt.show()
```

Listing 4: Código Gráfica en dos dimensiones

A continuación se exponen los códigos correspondientes a la segunda entrega del laboratorio:
Partimos de la gráfica de discriminación por Estados:

```

#EVOLUCION COVID 19

  Fallecidos=[]
  Fallecidos2=[]
  Graves=[]
  Leves=[]
  Moderados=[]
  Recuperados=[]

  urlDatos2 = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
  urlDatosSQL2 = 'sql=SELECT "ESTADO" ,"LOCALIDAD_ASIS", count(*) as Cantidad  from ...
      "b64ba3c4-9e41-41b8-b3fd-2da21d627558"  GROUP BY "ESTADO","LOCALIDAD_ASIS" ORDER BY ...
      "ESTADO","LOCALIDAD_ASIS"'

  req2 = requests.get(url=urlDatos2+urlDatosSQL2)
  reqJson2 = req2.json() #Organizar datos
  Datos2 = reqJson2['result']['records']


  #Grafica de barras
  auxDict2=[]
  for x in range(int(len(Datos2)/6)):

  auxDict2.append(Datos2[x]["LOCALIDAD_ASIS"])


  for I in auxDict2:
  for J in range(len(Datos2)):
```

```
        if(Datos2[J]["LOCALIDAD_ASIS"]==I):
            if(Datos2[J]["ESTADO"]=='Fallecido'):
                Fallecidos.append(int(Datos2[J]["cantidad"]))

            elif(Datos2[J]["ESTADO"]=='Fallecido No aplica No causa Directa'):
                Fallecidos2.append(int(Datos2[J]["cantidad"]))
            elif (Datos2[J]["ESTADO"] == 'Grave'):
                Graves.append(int(Datos2[J]["cantidad"]))
            elif (Datos2[J]["ESTADO"] == 'Leve'):
                Leves.append(int(Datos2[J]["cantidad"]))
            elif (Datos2[J]["ESTADO"] == 'Moderado'):
                Moderados.append(int(Datos2[J]["cantidad"]))
            elif (Datos2[J]["ESTADO"] == 'Recuperado'):
                Recuperados.append(int(Datos2[J]["cantidad"]))

    fig2= plt.figure("Evoluci n Casos Covid-19 Bogot ", figsize=(17.5, 8.5))
    fig2.suptitle("Evoluci n Casos Covid-19 Bogot ")

    FallecidosG = fig2.add_subplot(231)
    FallecidosNCDG = fig2.add_subplot(232)
    GravesG = fig2.add_subplot(233)
    ModeradosG = fig2.add_subplot(234)
    RecuperadosG = fig2.add_subplot(235)

    Localidad1 = [15 , 12, 7 , 2, 19, 10, 9, 20, 8 , 17, 14, 16, 18, 4, 3, 21, 11, 13, 6, 1 , 5]
    FallecidosGG = np.array(Fallecidos)

    Localidad2 = [15 , 12, 7 , 2, 19, 10, 9, 20, 8 , 17, 14, 16, 18, 4, 3, 21, 11, 13, 6, 1 , 5]
    FallecidosNCDGG = np.array(Fallecidos2)

    Localidad3 = [15 , 12, 7 , 2, 19, 10, 9, 20, 8 , 14, 16, 18, 4, 3, 21, 11, 13, 6, 1 , 5]
    GravesGG = np.array(Graves)

    Localidad4 = [15 , 12, 7 , 2, 19, 10, 9, 20, 8 , 17, 14, 16, 18, 4, 3, 21, 11, 13, 6, 1 , 5]
    ModeradosGG = np.array(Moderados)

    Localidad5 = [15 , 12, 7 , 2, 19, 10, 9, 20, 8 , 17, 14, 16, 18, 4, 3, 21, 11, 13, 6, 1 , 5]
    RecuperadosGG = np.array(Recuperados)


    FallecidosG.bar(Localidad1, FallecidosGG, align="center")
    FallecidosG.set_xticks(Localidad1)
    FallecidosG.set_xticklabels(Localidad1)
    FallecidosG.set_ylabel("Fallecidos")
    FallecidosG.set_xlabel("N mero de Localidad")

    FallecidosNCDG.bar(Localidad2, FallecidosNCDGG, align="center")
    FallecidosNCDG.set_xticks(Localidad2)
    FallecidosNCDG.set_xticklabels(Localidad2)
    FallecidosNCDG.set_ylabel("Fallecidos No Causa Directa")
    FallecidosNCDG.set_xlabel("N mero de Localidad")

    GravesG.bar(Localidad3, GravesGG, align="center")
    GravesG.set_xticks(Localidad3)
    GravesG.set_xticklabels(Localidad3)
    GravesG.set_ylabel("Graves")
    GravesG.set_xlabel("N mero de Localidad")

    ModeradosG.bar(Localidad4, ModeradosGG, align="center")
    ModeradosG.set_xticks(Localidad4)
    ModeradosG.set_xticklabels(Localidad4)
    ModeradosG.set_ylabel("Moderados")
    ModeradosG.set_xlabel("N mero de Localidad")

    RecuperadosG.bar(Localidad5, RecuperadosGG, align="center")
    RecuperadosG.set_xticks(Localidad4)
    RecuperadosG.set_xticklabels(Localidad4)
    RecuperadosG.set_ylabel("Recuperados")
    RecuperadosG.set_xlabel("N mero de Localidad")

    plt.show()
```

Listing 5: Código Discriminación por estados

De segunda gráfica encontramos la discriminacion por genero en las localidades.

```
1
2   #Genero
3
4   ListaValor=[]
5   ListaLocalidad=[]
6   Localidad=[]
7   Casos=[]
8   Tiempo=[]
9   Edades=[]
10  SexoH=[]
11  SexoM=[]
12
13  #LOCALIDAD
14  urlDatos = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
15  urlDatosSQL = 'sql=SELECT "LOCALIDAD_ASIS" , count(*) as Cantidad from ...
        "b64ba3c4-9e41-41b8-b3fd-2da21d627558" GROUP BY "LOCALIDAD_ASIS" ORDER BY "LOCALIDAD_ASIS"'
16
17  req = requests.get(url=urlDatos+urlDatosSQL)
18  reqJson = req.json() #Organizar datos
19  #print(req.json())
20  Datos = reqJson['result']['records']
21  i=0
22
23  urlDatos2 = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
24  urlDatosSQL2 = 'sql=SELECT "SEXO" ,"LOCALIDAD_ASIS", count(*) as Cantidad  from ...
        "b64ba3c4-9e41-41b8-b3fd-2da21d627558"  GROUP BY "SEXO","LOCALIDAD_ASIS" ORDER BY ...
        "SEXO","LOCALIDAD_ASIS"'
25
26  req2 = requests.get(url=urlDatos2+urlDatosSQL2)
27  reqJson2 = req2.json() #Organizar datos
28  #print(req2.json())
29  Datos2 = reqJson2['result']['records']
30
31  #Grafica de barras
32  auxDict=[]
33  #print(int(len(Datos2)/2))
34  for x in range(int(len(Datos2)/2)):
35
36  auxDict.append(Datos2[x]["LOCALIDAD_ASIS"])
37
38  #print(auxDict)
39
40  for I in auxDict:
41  for J in range(len(Datos2)):
42      if(Datos2[J]["LOCALIDAD_ASIS"]==I):
43          if(Datos2[J]["SEXO"]=='M'):
44              SexoH.append(int(Datos2[J]["cantidad"]))
45          else:
46              SexoM.append(int(Datos2[J]["cantidad"]))
47
48  #print(SexoH)
49  #print(SexoM)
50  plt.rcdefaults()
51
52  X=np.arange(len(auxDict))
53  width=0.25
54
55  fig, ax = plt.subplots(figsize=(18, 7))
56  rects1= ax.bar(X - (width/2), SexoM, width, label='Hombres')
57  rects2= ax.bar(X + (width/2), SexoH, width, label='Mujeres')
58
59  ax.set_ylabel('Contagios')
60
61  ax.set_title('Contagios Covid-19 ')
62  ax.set_xticks(X)
```

```
ax.set_xticklabels(auxDict)
ax.legend()

def autolabel(rects):
"""Attach a text label above each bar in *rects*, displaying its height."""
for rect in rects:
    height = rect.get_height()
    ax.annotate('{}'.format(height),
                xy=(rect.get_x() + rect.get_width() / 2, height),
                xytext=(0, 3),  # 3 points vertical offset
                textcoords="offset points",
                ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```

Listing 6: Código Discriminación Género

Aquí encontraremos el codigo realizado para la discriminacion por edades, recalcando que los datos van desde 1 año de edad hasta los 99 años.

```

#EDADES


ListaValor=[]
ListaLocalidad=[]
Localidad=[]
Casos=[]
Tiempo=[]
Edades=[]


urlDatos6 = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
urlDatosSQL6 = 'sql=SELECT "EDAD","LOCALIDAD_ASIS", count(*) as Cantidad from ...
    "b64ba3c4-9e41-41b8-b3fd-2da21d627558"  GROUP BY "EDAD","LOCALIDAD_ASIS" ORDER BY      ...
    "EDAD","LOCALIDAD_ASIS"'

req6 = requests.get(url=urlDatos6+urlDatosSQL6)
reqJson6 = req6.json() #Organizar datos
Datos6 = reqJson6['result']['records']
auxDict6=[]

for x in range(int(len(Datos6)/96)):

auxDict6.append(Datos6[x]["LOCALIDAD_ASIS"])


Edad1=[]
Edad2=[]
Edad3=[]
Edad4=[]
Edad5=[]
Edad6=[]
Edad7=[]
Edad8=[]
Edad9=[]
Edad10=[]
Edad11=[]
Edad12=[]
Edad13=[]
Edad14=[]
Edad15=[]
Edad16=[]
Edad17=[]
Edad18=[]
Edad19=[]
```

```
    Edad20=[]
    Edad21=[]
    Edad22=[]
    Edad23=[]
    Edad24=[]
    Edad25=[]
    Edad26=[]
    Edad27=[]
    Edad28=[]
    Edad29=[]
    Edad30=[]
    Edad31=[]
    Edad32=[]
    Edad33=[]
    Edad34=[]
    Edad35=[]
    Edad36=[]
    Edad37=[]
    Edad38=[]
    Edad39=[]
    Edad40=[]
    Edad41=[]
    Edad42=[]
    Edad43=[]
    Edad44=[]
    Edad45=[]
    Edad46=[]
    Edad47=[]
    Edad48=[]
    Edad49=[]
    Edad50=[]
    Edad51=[]
    Edad52=[]
    Edad53=[]
    Edad54=[]
    Edad55=[]
    Edad56=[]
    Edad57=[]
    Edad58=[]
    Edad59=[]
    Edad60=[]
    Edad61=[]
    Edad62=[]
    Edad63=[]
    Edad64=[]
    EdadV=[]

    for I in auxDict6:
    for J in range(len(Datos6)):
        if(Datos6[J]["LOCALIDAD_ASIS"]==I):
            if(Datos6[J]["EDAD"]) == "1":
                Edad1.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "2":
                Edad2.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "3":
                Edad3.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "4":
                Edad4.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "5":
                Edad5.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "6":
                Edad6.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "7":
                Edad7.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "8":
                Edad8.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "9":
                Edad9.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "10":
                Edad10.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "11":
```

```
            Edad11.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "12":
            Edad12.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "13":
            Edad13.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "14":
            Edad14.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "15":
            Edad15.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "16":
            Edad16.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "17":
            Edad17.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "18":
            Edad18.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "19":
            Edad19.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "20":
            Edad20.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "21":
            Edad21.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "22":
            Edad22.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "23":
            Edad23.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "24":
            Edad24.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "25":
            Edad25.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "26":
            Edad26.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "27":
            Edad27.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "28":
            Edad28.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "29":
            Edad29.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "30":
            Edad30.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "31":
            Edad31.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "32":
            Edad32.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "33":
            Edad33.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "34":
            Edad34.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "35":
            Edad35.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "36":
            Edad36.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "37":
            Edad37.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "38":
            Edad38.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "39":
            Edad39.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "40":
            Edad40.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "41":
            Edad41.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "42":
            Edad42.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "43":
            Edad43.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "44":
            Edad44.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "45":
            Edad45.append(int(Datos6[J]["cantidad"]))
        elif(Datos6[J]["EDAD"]) == "46":
            Edad46.append(int(Datos6[J]["cantidad"]))
```

```
            elif(Datos6[J]["EDAD"]) == "47":
                Edad47.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "48":
                Edad48.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "49":
                Edad49.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "50":
                Edad50.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "51":
                Edad51.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "52":
                Edad52.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "53":
                Edad53.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "54":
                Edad54.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "55":
                Edad55.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "56":
                Edad56.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "57":
                Edad57.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "58":
                Edad58.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "59":
                Edad59.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "60":
                Edad60.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "61":
                Edad61.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "62":
                Edad62.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "63":
                Edad63.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) == "64":
                Edad64.append(int(Datos6[J]["cantidad"]))
            elif(Datos6[J]["EDAD"]) > "64":
                EdadV.append(int(Datos6[J]["cantidad"]))

    nios = (sum(Edad1)+sum(Edad2)+sum(Edad3)+sum(Edad4)+sum(Edad5)+sum(Edad6)+sum(Edad7)+sum(Edad8)
    +sum(Edad9)+sum(Edad10)+sum(Edad11)+sum(Edad12)+sum(Edad13))
    Adolecentes = (sum(Edad14)+sum(Edad15)+sum(Edad16)+sum(Edad17))
    AdultosJ = (sum(Edad19)+sum(Edad20)+sum(Edad21)+sum(Edad22)+sum(Edad23)+sum(Edad24)+sum(Edad25)
    +sum(Edad26)+sum(Edad27)+sum(Edad28)+sum(Edad29)+sum(Edad30)+sum(Edad31)+sum(Edad32)+sum(Edad33)
    +sum(Edad34)+sum(Edad35))
    Adultos = (sum(Edad36)+sum(Edad37)+sum(Edad38)+sum(Edad39)+sum(Edad40)+sum(Edad41)+sum(Edad42)
    +sum(Edad43)+sum(Edad44)+sum(Edad45)+sum(Edad46)+sum(Edad47)+sum(Edad48)+sum(Edad49)+sum(Edad50)
    +sum(Edad51)+sum(Edad52)+sum(Edad53)+sum(Edad54)+sum(Edad55)+sum(Edad56)+sum(Edad57)+sum(Edad58)
    +sum(Edad59)+sum(Edad60)+sum(Edad61)+sum(Edad62)+sum(Edad63)+sum(Edad64))
    TerceraEdad = (sum(EdadV))

    labels = 'Nios (1-13) ', 'Adolecentes (14-17)', 'Adultos (36-64)', 'Adultos     jovenes ...
        (18-35)', 'Tercera Edad (65 en adelante)'
    sizes = [nios, Adolecentes, Adultos, AdultosJ, TerceraEdad]
    explode = (0.1, 0.1, 0.1, 0.1, 0.1)


    fig6, ax6 = plt.subplots(figsize=(15, 7))
    ax6.pie(sizes,explode=explode, autopct='%1.1f %%', shadow=True, startangle=90)
    ax6.set_title("Discriminacion por edades Covid-19 Bogot ")
    ax6.legend(labels,title="EDADES",loc="lower left",bbox_to_anchor=(1, 0, 0.5, 1))

    plt.show()
```

Listing 7: Código Discriminación Edad

Por ultimo, encontramos el codigo realizado para graficar un mapa de calor con el numero de contagiados por localidades en Bogotá.

```

    #MAPA DE CALOR
```

```
3
4   ListaValor=[]
5   ListaLocalidad=[]
6   Localidad=[]
7   Casos=[]
8   Tiempo=[]
9
10
11  urlDatos5 = 'https://datosabiertos.bogota.gov.co/api/3/action/datastore_search_sql?'
12  urlDatosSQL5 = 'sql=SELECT "LOCALIDAD_ASIS", count(*) as Cantidad from ...
        "b64ba3c4-9e41-41b8-b3fd-2da21d627558" GROUP BY "LOCALIDAD_ASIS" ORDER BY "LOCALIDAD_ASIS"'
13
14  req5 = requests.get(url=urlDatos+urlDatosSQL)
15  reqJson5 = req5.json() #Organizar datos
16  Datos5 = reqJson5['result']['records']
17  i=0
18
19  for fila in Datos5:
20  ListaValor.append(int(fila["cantidad"]))
21  ListaLocalidad.append(fila["LOCALIDAD_ASIS"])
22  i = i + 1
23
24  DatosMapa = np.asarray(ListaValor).reshape(22,1)
25  text = np.asarray(ListaLocalidad)
26
27  labels = (np.asarray(["{0}\n{1:.0f}".format(text,DatosMapa) for text, DatosMapa in ...
        zip(text.flatten(),      DatosMapa.flatten())])).reshape(22,1)
28
29  fig5, ax5 = plt.subplots(figsize=(15.5,9))
30  heat_map = sb.heatmap(DatosMapa, annot=labels, xticklabels=False, fmt='', cbar_kws={'label': ...
        'Numero de      contagios', 'orientation': 'vertical'}, )
31  heat_map.set_title("MAPA DE CALOR COVID-19 POR LOCALIDADES")
```

Listing 8: Mapa de calor

Para encontrar este documento, junto con el código y video correspondiente por favor visite nuestro repositorio https://github.com/CarlosPA90-666/ProyectoSeniales.git

3. Conclusiones

- Se evidencia que el 2020 ha sido un año complicado para la humanidad, trayendo con él diferentes problemas a nivel mundial, en este caso el virus "Covid19, del cual es necesario tener una información actualizada de sus síntomas, tratamientos, prevención y expansión. Para esto, encontramos diferentes páginas web que nos brindan dicha información. Evidenciando la necesidad de extraer la información con recursos que se tienen a la mano durante nuestra formación académica como python y sql. Este proyecto nos servirá para presentar de forma gráfica el dia a dia de esta enfermedad para así tomar decisiones, controlar y actuar a tiempo.
- Es necesario conocer los datos actualizados de nuestra ciudad, separada por localidades y diferentes discriminaciones, todo esto con el fin de saber como cuidarnos de una mejor forma, de saber que edades son las mas afectadas y poder cuidar así a nuestros familiares, por otro lado conocer como actua realmente el virus en nuestros cuerpos, saber como evoluciona y como es el proceso de recuperacion del mismo.

Referencias

[1] M. Colombia, "Coronavirus (covid-19)." Colombia, 2020.

[2] A. Oppel, "Fundamentos de bases de datos." México D.F: McGraw-Hill, 2011.

[3] J. Hunter, "Python plotting — matplotlib 3.3.2 documentation." Matplotlib, 2012.

[4] Unypython, "Solicitudes http en python con requests." Unypython, 2018.

[5] A. Ahmed, "Cómo trabajar con datos json utilizando python." Ahmed, 2016.

[6] T. Donaldson, "Python." Berkeley, Calif.: Peachpit Press, 2014.

[7] M. Colombia, "Mapas." Colombia, 2015.

4. Anexos

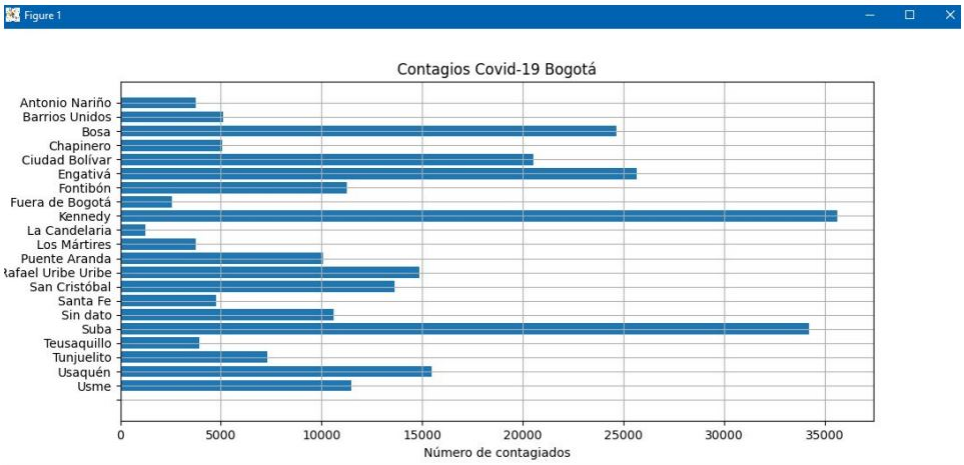

Figura 2: Gráfica de Barras Contagio Por Localidades

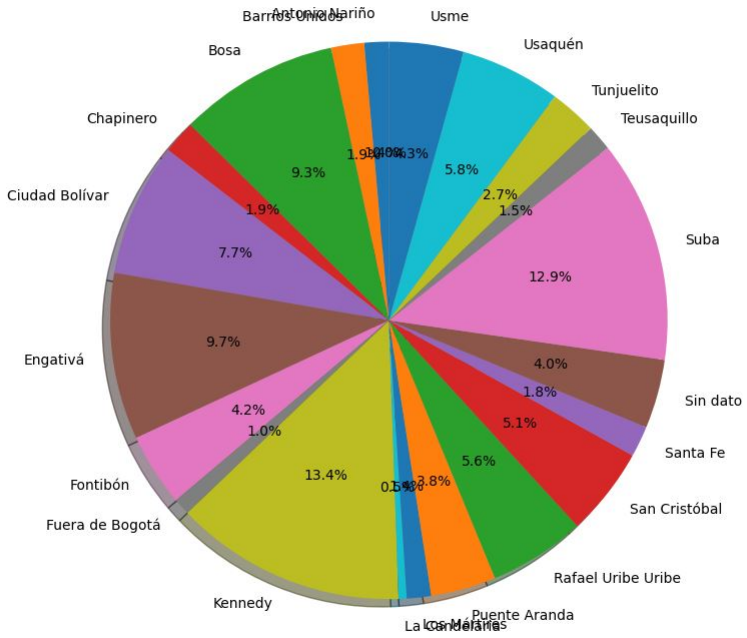

Figura 3: Gráfico de Torta por Porcentajes

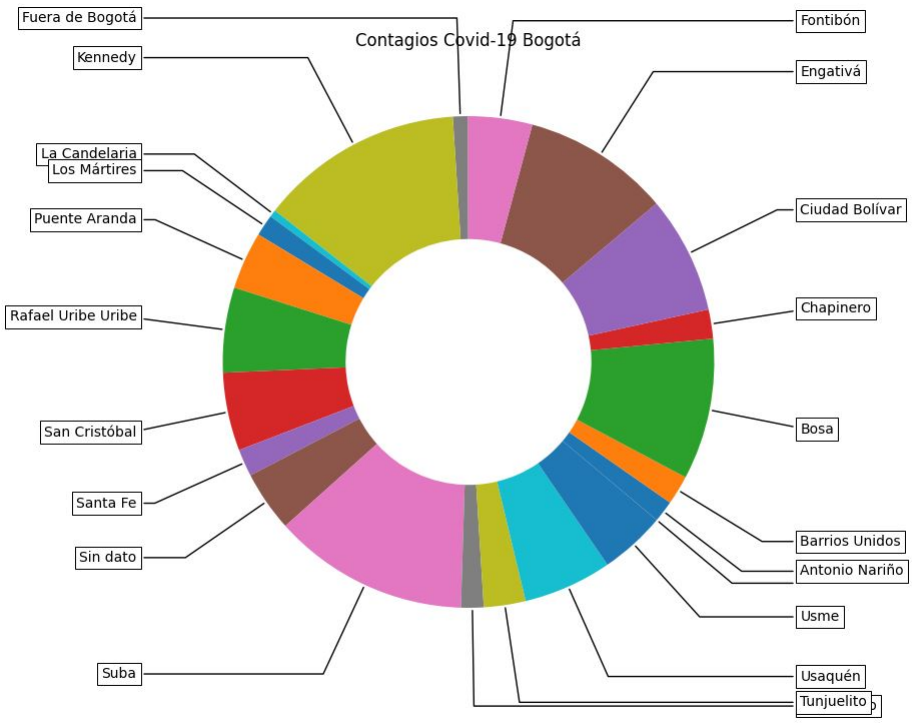

Figura 4: Gráfico de Torta por Nombres

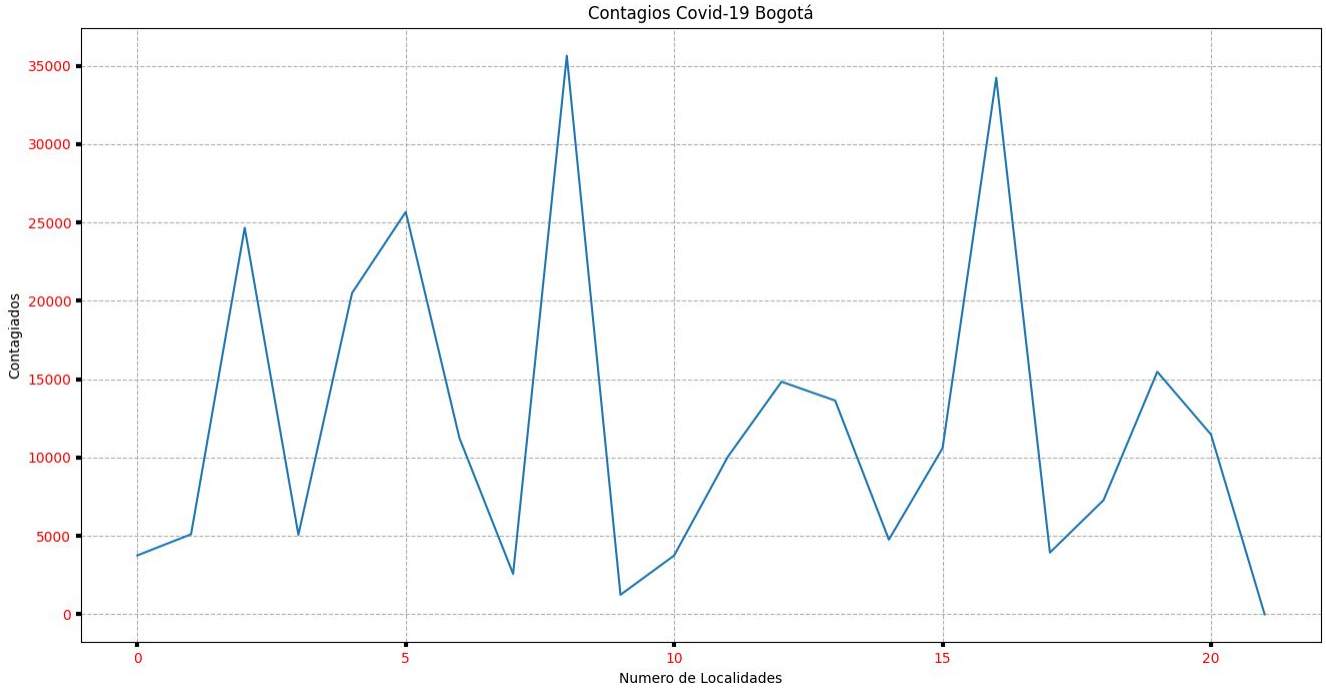

Figura 5: Gráfico de Dos Dimensiones

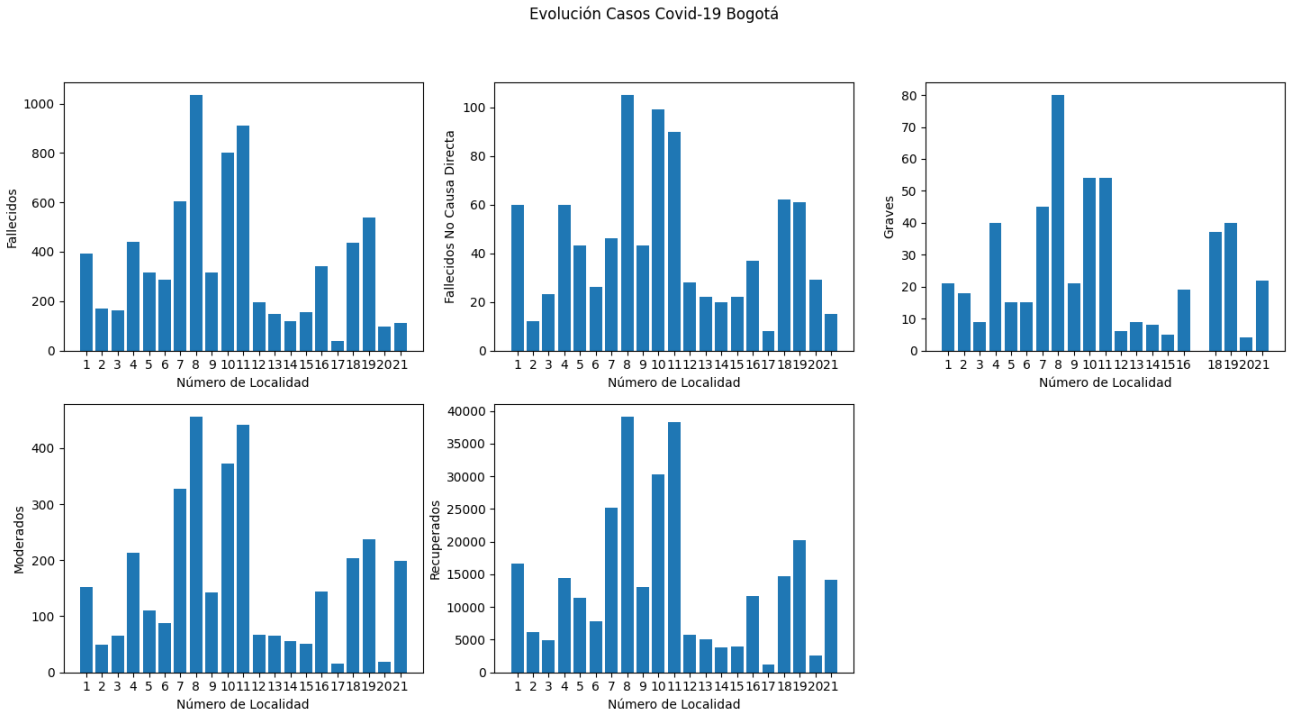

Figura 6: Gráfico discriminacion por edades

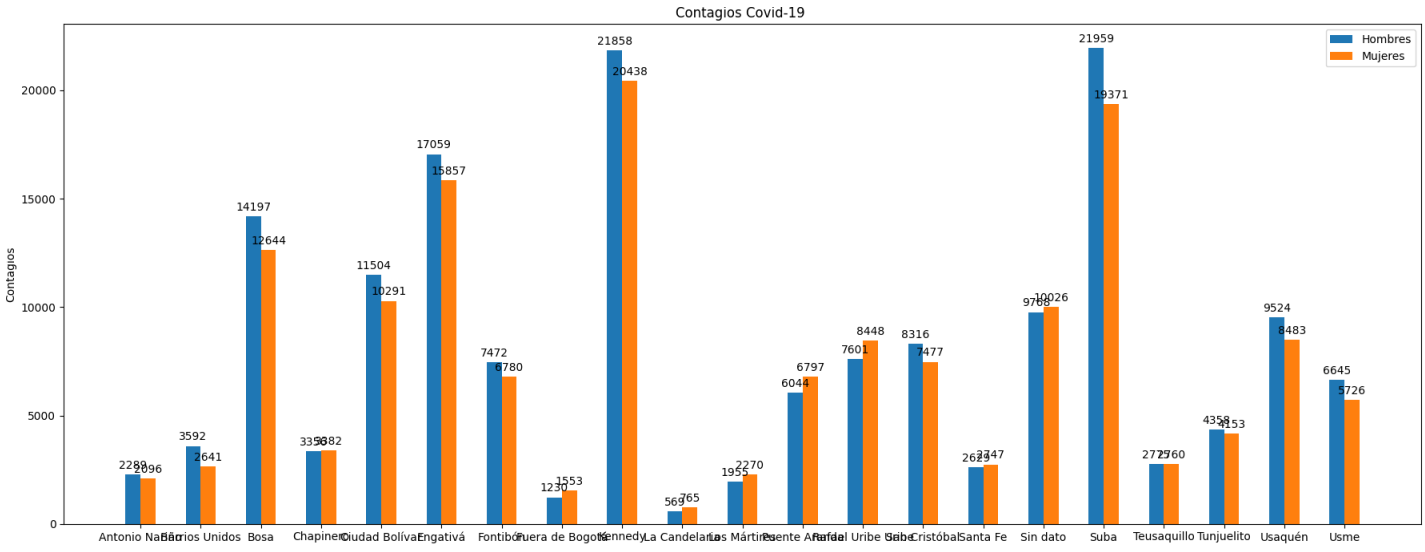

Figura 7: Gráfico discriminacion por genero

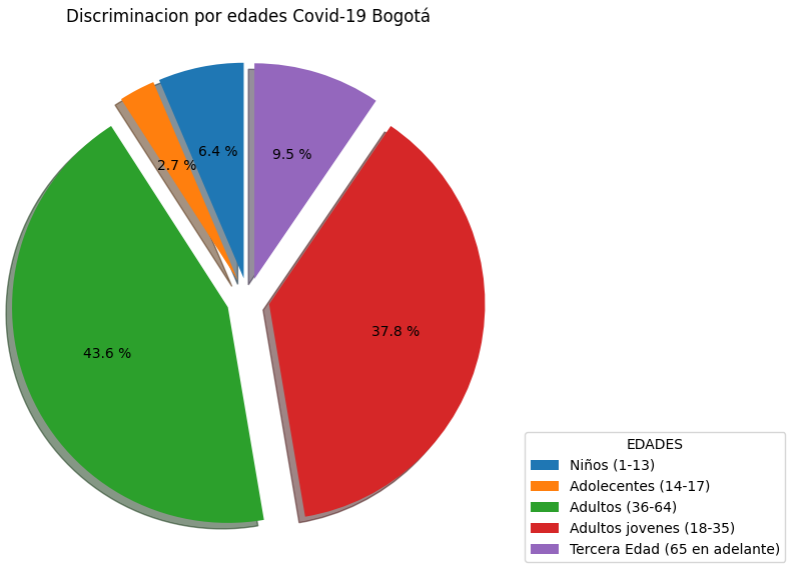

Figura 8: Gráfico discriminacion por edad

Figura 9: Mapa de calor de Bogotá