Ejercicio Python Parte 2

José Julián Barbosa Ayala
Michael Hernández Vera
Camilo Vega Ramírez

Contenido

Limpieza de datos.

Estas son las bibliotecas que se utilizarán en este estudio.

```
import pandas as pd # Librería para manipular y analizar datos en formato de tabla
import numpy as np  # Librería para realizar operaciones matemáticas en arrays
import datetime # Librería para trabajar con fechas y horas
import seaborn as sns # Librería para realizar visualizaciones estadísticas
import matplotlib.pyplot as plt # Librería para crear gráficos y visualizaciones
import matplotlib.ticker as ticker # Librería para personalizar los ticks en los ejes
import matplotlib.cm as cm # Librería para definir colores
import squarify # Librería para crear gráficas de tipo treemap
```

Para iniciar, cargamos los datos y aplicamos el mismo proceso de limpieza y transformación que se llevó a cabo durante la Fase 1.

```
# Cargamos los datos desde un archivo CSV
datosP = pd.read_csv("time_series_covid19_confirmed_global.csv")

# Creamos data frame con los datos limpios
datosP_clean1 = datosP.assign(
    **{"Country/Region": lambda x: np.where(
        # Verificamos que la columna "Province/State" no sea nula
        (~x["Province/State"].isna()) &
        # Verificamos si el país está en esta lista
        x["Country/Region"].isin(["Denmark",
                                  "France",
                                  "Netherlands",
                                  "New Zealand",
                                  "United Kingdom"]),
        # Si el país está en la lista, reemplazamos el nombre del país con el
        # nombre de la provincia/estado
        x["Province/State"],
        # Si el país no está en la lista, mantenemos el nombre del país
        x["Country/Region"]
    )}
    # Quitamos columnas que no vamso a utilizar
).drop(["Province/State", "Lat", "Long"], axis=1)


datosP_clean1 = datosP_clean1.assign(
    **{"Country/Region": np.select(
        # Verificamos si el nombre del país es "Summer Olympics 2020"
        [datosP_clean1["Country/Region"] == "Summer Olympics 2020",
        # Verificamos si el nombre del país es "Winter Olympics 2022"
         datosP_clean1["Country/Region"] == "Winter Olympics 2022"],
        # Si el nombre del país es "Summer Olympics 2020", reemplazamos el nombre
        # del país con "Japan"
        ["Japan",
        # Si el nombre del país es "Winter Olympics 2022", reemplazamos el nombre
        # del país con "China"
         "China"],
        default=datosP_clean1["Country/Region"])}
```

```
)

# Eliminamos las filas donde el nombre del país es 'Diamond Princess' o 'MS Zaandam'
datosP_clean1 = datosP_clean1[
    ~datosP_clean1["Country/Region"
    ].isin(['Diamond Princess', 'MS Zaandam'])]

# Definimos las columnas que queremos mantener en nuestro conjunto de datos
cols_mantener = ['Country/Region']

# Obtenemos las columnas que queremos transformar, es decir, aquellas que contienen
# información sobre casos de COVID-19
cols_transformar = datosP_clean1.columns.difference(cols_mantener)

# Utilizamos la función melt() de pandas para transformar nuestro conjunto de datos.
# Esta función desagrega las columnas que contienen información sobre casos de COVID-19
# y los convierte en filas, con una nueva columna llamada "contagios"
datosP_tab = datosP_clean1.melt(
    # columnas que queremos mantener como identificadores
    id_vars = cols_mantener,
    # columnas que queremos desagregar
    value_vars = cols_transformar,
    # nombre de la columna que contendrá las fechas
    var_name = 'fecha',
    # nombre de la columna que contendrá el número de contagios
    value_name = 'contagios')

# Agrupamos los datos por país/ubicación, latitud, longitud y fecha, y sumamos
# el número de contagios para cada grupo
datosP_tab = datosP_tab.groupby(
    ['Country/Region', 'fecha']
    ).agg(
        {'contagios':'sum'}
    ).reset_index()

# Convertir la columna fecha en un objeto de fecha y hora de Pandas
datosP_tab['fecha'] = pd.to_datetime(datosP_tab['fecha'], format='%m/%d/%y')

datosP_tab_non0 = datosP_tab[datosP_tab['contagios'] != 0]

# Ordenar el conjunto de datos primero por Country/Region y luego por fecha,
# y restablecer el índice
datosP_tab_non0 = datosP_tab_non0.sort_values(
    ['Country/Region', 'fecha']
    ).reset_index(drop=True)
```

Preguntas de interes Caso COVID-19 Fase 2.

Una vez transformados nuestros datos a formato tabular, procederemos a responder las preguntas de la fase 2. En primer lugar, identificaremos los cinco países con mayor cantidad de contagios totales. Utilizaremos los tres primeros países de esta lista para responder la primera pregunta, y los cinco países para responder las preguntas 2 y 3.

```
datosP_tab_non0.loc[
  # Seleccionar las filas con la fecha más reciente
    datosP_tab_non0['fecha'] == datosP_tab_non0['fecha'].max(),
  # Ordenar por número de contagios de manera descendente y
  # seleccionar las primeras 5 filas
    ['Country/Region', 'contagios']
    ].sort_values(['contagios'], ascending=False).head(5).reset_index(drop=True)
```

```
##   Country/Region  contagios
## 0             US  103802702
## 1          India   44690738
## 2         France   38618509
## 3        Germany   38249060
## 4         Brazil   37076053
```

Pregunta 1.

> *Construir una gráfica que muestre la variación de contagios, mes a mes, de los tres países con más casos reportados.*

Para responder a la primera pregunta, es necesario realizar una ingeniería de características que nos permita obtener el recuento de contagios diarios exclusivamente para los tres países con mayor cantidad de contagios. Además, debemos crear una columna que indique el año y mes de cada observación. Finalmente, debemos agregar los datos por año-mes y país.

```
# Calcular el número de contagios por día para cada país
datosP_dia = datosP_tab_non0.loc[
  datosP_tab_non0['Country/Region'].isin(['US', 'India', 'France'])
  ].groupby(
    'Country/Region', as_index=False
    ).apply(
        lambda x: x.assign(contagios_dia = x['contagios'] - x['contagios'].shift(1))
        )

# Crear una columna "año_mes" con el formato YYYY-MM
datosP_dia['año_mes'] = pd.to_datetime(datosP_dia['fecha']).dt.strftime('%Y-%m')

# Calcular el número de contagios por mes por pais
datosP_mes = datosP_dia.groupby(
    ['Country/Region', 'año_mes'], as_index=False
    ).agg(contagios_mes=('contagios_dia', 'sum'))
```

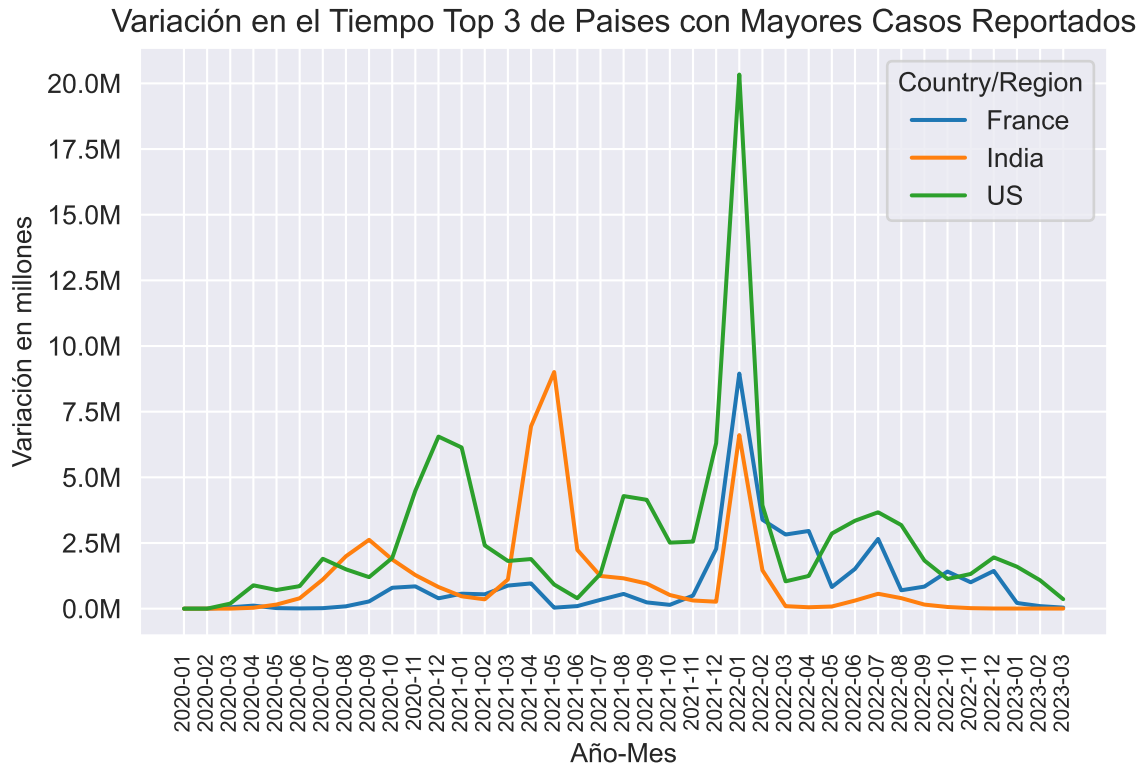
Una vez nuestros datos listos, procederemos a configurar la gráfica.

```
# Definir una función que convierte un número en formato de millones
def formato_millones(x, pos):
    return '{:.1f}M'.format(x/1000000)

# Establecer el estilo de la gráfica
sns.set_style('darkgrid')
```

```
# Crear una gráfica de línea y asignarla a la variable 'grafica'
grafica = sns.lineplot(x='año_mes',
                       y='contagios_mes',
                       hue='Country/Region',
                       data=datosP_mes, ci=None)

# Rotar las etiquetas del eje x en 90 grados y establecer el tamaño de fuente
grafica.set_xticklabels(grafica.get_xticklabels(), rotation=90, fontsize=8)

# Formatear el eje y para que los números aparezcan en millones
grafica.yaxis.set_major_formatter(ticker.FuncFormatter(formato_millones))

# Establecer las etiquetas de los ejes x e y y el título de la gráfica
grafica.set_xlabel('Año-Mes')
grafica.set_ylabel('Variación en millones')
grafica.set_title('Variación en el Tiempo Top 3 de Paises con Mayores Casos Reportados')

# Ajustar la posición de los subplots para evitar que los títulos se superpongan
plt.subplots_adjust(bottom=0.2)
```

A continuación, se presenta la gráfica que muestra la variación mensual de contagios para los tres países con mayor cantidad de casos reportados.

```
plt.show()
```



```
plt.clf()
```

Pregunta 2.

Construir una gráfica que muestre una comparación del total de contagios de cinco países.

Para responder a esta pregunta, crearemos un conjunto de datos que contenga el total de contagios para cada uno de los países dentro del top 5 de países con mayor cantidad de casos reportados.

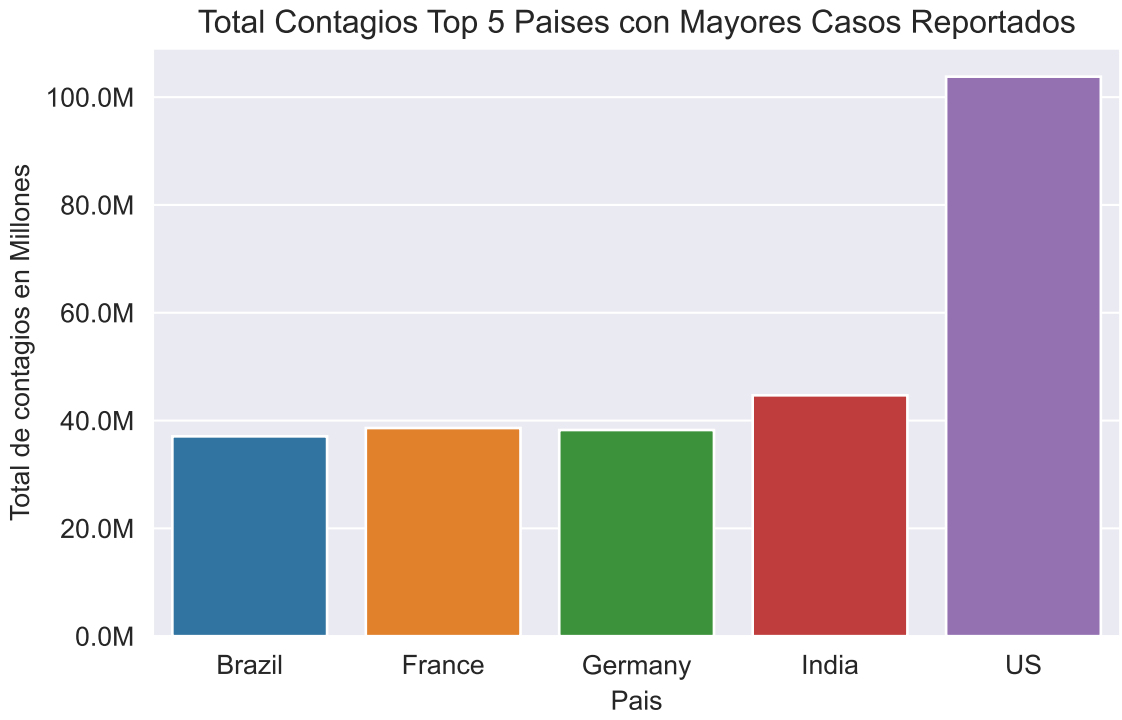
```
# Seleccionar las filas con la fecha más reciente y para los países top 5
datos_top_5 = datosP_tab_non0.loc[
    (datosP_tab_non0['fecha'] == datosP_tab_non0['fecha'].max()) &
    (datosP_tab_non0['Country/Region'].isin(
      ['US', 'India', 'France', 'Germany', 'Brazil']
      )),
    ['Country/Region', 'contagios']
    ].reset_index(drop=True)
```

Con nuestros datos listos, procederemos a configurar nuestra gráfica.

```
# Establecer el estilo de la gráfica
sns.set_style('darkgrid')

# Crear una gráfica de barras y asignarla a la variable 'grafica'
grafica = sns.barplot(x='Country/Region', y='contagios', data=datos_top_5)

# Formatear el eje y para que los números aparezcan en millones
grafica.yaxis.set_major_formatter(ticker.FuncFormatter(formato_millones))

# Establecer las etiquetas de los ejes x e y y el título de la gráfica
grafica.set_xlabel('Pais')
grafica.set_ylabel('Total de contagios en Millones')
grafica.set_title('Total Contagios Top 5 Paises con Mayores Casos Reportados')
```

A continuación, se presenta la gráfica que compara el total de contagios entre los cinco países con mayor cantidad de casos reportados.

```
plt.show()
```




```
plt.clf()
```

Pregunta 3.

> *Usando los países elegidos en el punto anterior, construir una gráfica que muestre el porcentaje de contagios de cada país con respecto al total de contagios en el mundo.*

Para responder a esta pregunta, llevaremos a cabo una ingeniería de características. En primer lugar, seleccionaremos únicamente los datos correspondientes al último día, los cuales contienen los contagios totales más recientes. Luego, asignaremos el nombre `Otros` a aquellos países que no se encuentren en el top 5. Después, agruparemos los datos para obtener la suma de contagios y crearemos una columna que indique el porcentaje del total de contagios. Finalmente, crearemos otra columna en formato string que contenga el nombre del país y su respectivo porcentaje, la cual utilizaremos como etiqueta para la gráfica.

```
# Seleccionar los datos correspondientes al último día registrado
total_contagios_pais = datosP_tab_non0.loc[
    datosP_tab_non0['fecha'] == datosP_tab_non0['fecha'].max(),
    ['Country/Region', 'contagios']
    ].sort_values(['contagios'], ascending=False).reset_index(drop=True)

# Crear una copia de los datos de los 5 países con más casos, y
# renombrar el resto como "Otros"
total_top_5_others = total_contagios_pais.copy()
total_top_5_others['Country/Region'] = total_top_5_others[
```

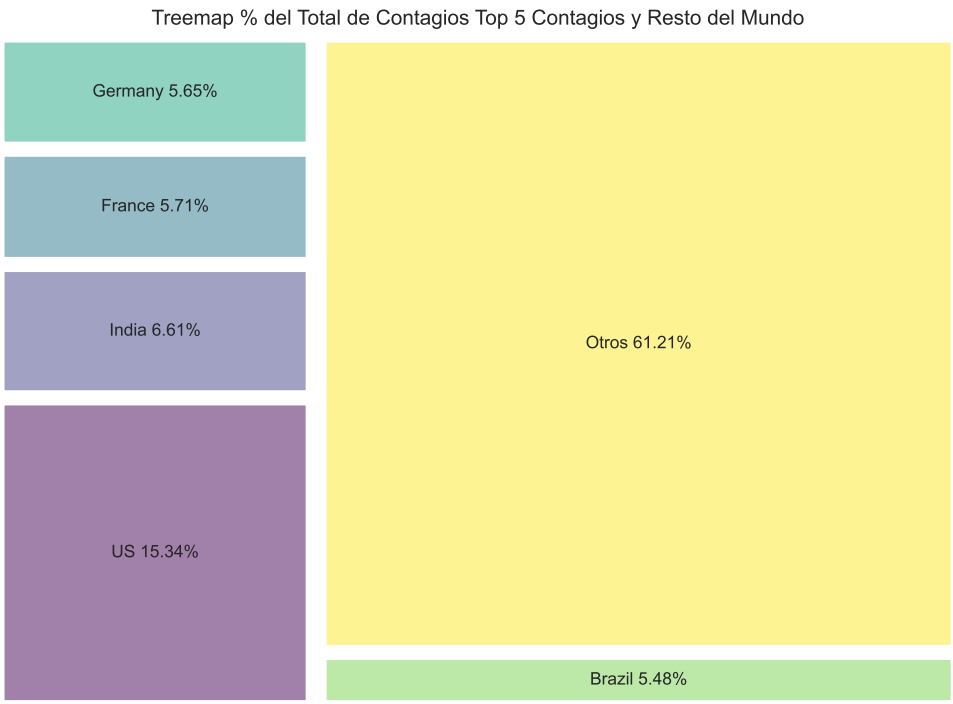
```
  'Country/Region'
  ].apply(lambda x: x if x in [
    'US', 'India', 'France', 'Germany', 'Brazil'
    ] else 'Otros')

# Agrupar los datos por país y sumar los casos, y calcular el porcentaje
# que representa cada país del total
total_top_5_others = total_top_5_others.groupby(
    ['Country/Region'], as_index=False, sort=False
    ).agg(contagios=('contagios', 'sum'))
total_top_5_others['percent'] = total_top_5_others[
  'contagios'
  ]/total_top_5_others['contagios'].sum()

# Agregar una columna que combine el nombre del país con su porcentaje de casos
total_top_5_others['Country/Region percent'] = total_top_5_others[
  'Country/Region'
  ] + ' ' + (total_top_5_others['percent']*100).round(2).astype(str) + '%'
```

Una vez que tengamos nuestros datos listos, procederemos a definir la gráfica. Es importante mencionar que preferimos utilizar gráficas de tipo `treemap` en lugar de `pie chart`, ya que el cerebro humano procesa mejor la información presentada en forma de rectángulos en lugar de ángulos. Además, las gráficas de tipo `treemap` son una mejor opción cuando se trata de visualizar la proporción de tres o más categorías.

```
# Creamos una lista con los valores percent
values = total_top_5_others['percent']

# Asignamos los valores de percent a los labels y los guardamos en una lista
labels = total_top_5_others['Country/Region percent']

# Asignamos los colores para cada observación
colores = ["#440154FF",
           "#414487FF",
           "#2A788EFF",
           "#22A884FF",
           "#7AD151FF",
           "#FDE725FF"]

# Creamos un layout para el treemap
fig, ax = plt.subplots(figsize=(10,7))
squarify.plot(sizes = values,
              label = labels,
              alpha = 0.5,
              pad = 1,
              color = colores)

# Agregamos un título y desactivamos los ejes
plt.title("Treemap % del Total de Contagios Top 5 Contagios y Resto del Mundo")
plt.axis('off')
```

A continuación, se presenta la gráfica que muestra el porcentaje de contagios de cada país dentro del top 5 de países con mayor cantidad de casos reportados en relación al total de contagios a nivel mundial.

```
plt.show()
```




```
plt.clf()
```

Reflexion.

Importancia de utilizar una herramienta de visualización de datos para realizar análisis de los datos en el contexto de la Ciencia de Datos.

La visualización de datos es una parte crucial en el proceso de análisis de datos y puede ayudar a descubrir patrones y tendencias ocultas en grandes conjuntos de información. Las herramientas de visualización, como matplotlib, seaborn y otras, permiten crear gráficos y visualizaciones efectivas que facilitan la interpretación y la comunicación de los resultados del análisis.

Una de las ventajas de las herramientas de visualización es que permiten personalizar los gráficos y las visualizaciones para adaptarlos a las necesidades del análisis y la audiencia. Esto puede aumentar la claridad y la eficacia de la comunicación de los resultados del análisis de datos.

La visualización de datos no solo ayuda a presentar los resultados de manera clara y efectiva, sino que también puede ayudar a descubrir patrones y relaciones entre variables. Al presentar visualmente los datos, es más fácil identificar tendencias, detectar valores atípicos y comunicar información de manera clara y efectiva a un público diverso. Esto puede ayudar a los científicos de datos y otros profesionales a tomar decisiones informadas en función de los datos.