Ejercicio Python

José Julián Barbosa Ayala
Michael Hernández Vera
Camilo Vega Ramírez

Contenido

Limpieza de datos.

Exploración 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

pd.options.display.float_format = '{:.0f}'.format # Elimina los decimales en la salida
```

Comenzamos cargando los datos y observando la estructura de nuestro conjunto de datos.

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

# Mostramos la forma (número de filas y columnas) de los datos cargados
datosP.shape
```

```
## (289, 1147)
```

Es recomendable analizar las columnas que integran el conjunto de datos y examinar tanto las primeras como las últimas observaciones para tener una idea clara del contenido del mismo.

```
# Mostramos los nombres de las columnas de los datos
datosP.columns
```

```
## Index(['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20', '1/23/20',
##        '1/24/20', '1/25/20', '1/26/20', '1/27/20',
##        ...
##        '2/28/23', '3/1/23', '3/2/23', '3/3/23', '3/4/23', '3/5/23', '3/6/23',
##        '3/7/23', '3/8/23', '3/9/23'],
##       dtype='object', length=1147)
```

```
# Mostramos las primeras 5 filas de los datos
datosP.head(5)
```

```
##   Province/State Country/Region  Lat  Long  ...  3/6/23  3/7/23  3/8/23  3/9/23
## 0            NaN    Afghanistan   34    68  ...  209406  209436  209451  209451
## 1            NaN        Albania   41    20  ...  334427  334427  334443  334457
## 2            NaN        Algeria   28     2  ...  271477  271490  271494  271496
## 3            NaN        Andorra   43     2  ...   47875   47875   47890   47890
## 4            NaN         Angola  -11    18  ...  105277  105277  105288  105288
##
## [5 rows x 1147 columns]
```

```
# Mostramos las últimas 5 filas de los datos
datosP.tail(5)
```

```
##     Province/State        Country/Region  Lat  ...  3/7/23  3/8/23  3/9/23
## 284            NaN    West Bank and Gaza   32  ...  703228  703228  703228
## 285            NaN  Winter Olympics 2022   40  ...     535     535     535
## 286            NaN                 Yemen   16  ...   11945   11945   11945
## 287            NaN                Zambia  -13  ...  343135  343135  343135
## 288            NaN              Zimbabwe  -19  ...  264127  264276  264276
##
## [5 rows x 1147 columns]
```

El conjunto de datos consta de 289 filas y 1147 columnas. Observamos que no está en formato tabular y que las primeras cuatro columnas corresponden a información sobre el estado/provincia, país/región, latitud y longitud. Las columnas restantes contienen datos de fechas desde el 22 de enero de 2020 hasta el 9 de marzo de 2023. En este momento, nuestro enfoque está en limpiar las primeras cuatro columnas.

Province/State.

Observemos la columna `Province/State`

```
# Mostramos los valores únicos en la columna "Province/State"
datosP['Province/State'].unique()
```

```
## array([nan, 'Australian Capital Territory', 'New South Wales',
##        'Northern Territory', 'Queensland', 'South Australia', 'Tasmania',
##        'Victoria', 'Western Australia', 'Alberta', 'British Columbia',
##        'Diamond Princess', 'Grand Princess', 'Manitoba', 'New Brunswick',
##        'Newfoundland and Labrador', 'Northwest Territories',
##        'Nova Scotia', 'Nunavut', 'Ontario', 'Prince Edward Island',
##        'Quebec', 'Repatriated Travellers', 'Saskatchewan', 'Yukon',
##        'Anhui', 'Beijing', 'Chongqing', 'Fujian', 'Gansu', 'Guangdong',
##        'Guangxi', 'Guizhou', 'Hainan', 'Hebei', 'Heilongjiang', 'Henan',
##        'Hong Kong', 'Hubei', 'Hunan', 'Inner Mongolia', 'Jiangsu',
##        'Jiangxi', 'Jilin', 'Liaoning', 'Macau', 'Ningxia', 'Qinghai',
##        'Shaanxi', 'Shandong', 'Shanghai', 'Shanxi', 'Sichuan', 'Tianjin',
##        'Tibet', 'Unknown', 'Xinjiang', 'Yunnan', 'Zhejiang',
##        'Faroe Islands', 'Greenland', 'French Guiana', 'French Polynesia',
##        'Guadeloupe', 'Martinique', 'Mayotte', 'New Caledonia', 'Reunion',
##        'Saint Barthelemy', 'Saint Pierre and Miquelon', 'St Martin',
##        'Wallis and Futuna', 'Aruba', 'Bonaire, Sint Eustatius and Saba',
##        'Curacao', 'Sint Maarten', 'Cook Islands', 'Niue', 'Anguilla',
##        'Bermuda', 'British Virgin Islands', 'Cayman Islands',
##        'Channel Islands', 'Falkland Islands (Malvinas)', 'Gibraltar',
##        'Guernsey', 'Isle of Man', 'Jersey', 'Montserrat',
##        'Pitcairn Islands', 'Saint Helena, Ascension and Tristan da Cunha',
##        'Turks and Caicos Islands'], dtype=object)
```

```
# Mostramos el número de valores únicos en la columna "Province/State"
datosP['Province/State'].unique().shape[0]
```

```
## 92
```

```
# Mostramos el número de valores faltantes (NaN) en la columna "Province/State"
datosP['Province/State'].isna().sum()
```

```
## 198
```

Al revisar la columna `Province/State`, observamos que existen 92 valores distintos, de los cuales 198 observaciones tienen valores vacíos (`nan`), es decir, más de dos tercios del número total de observaciones. El tipo de dato es un objeto, lo cual es adecuado para campos de texto.

Al observar las provincias, podemos identificar dos grupos distintos. En el caso de Australia, Canadá y China, se listan los estados dentro del país (equivalentes a los departamentos en Colombia). En el caso de Dinamarca, Francia, Países Bajos, Nueva Zelanda y Reino Unido, se listan los territorios que, en su mayoría, cuentan con gobiernos autónomos.

Además, encontramos las observaciones `Repatriated Travellers`, correspondientes a ciudadanos repatriados por Canadá, y `Unknown`, que se refiere a los contagios de China que no tienen asociada ninguna provincia o estado.

Country/Region.

Observemos ahora la columna `Country/Region`

```
# Mostramos los valores únicos en la columna "Country/Region"
datosP['Country/Region'].unique()
```

```
## array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
##        'Antarctica', 'Antigua and Barbuda', 'Argentina', 'Armenia',
##        'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
##        'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin',
##        'Bhutan', 'Bolivia', 'Bosnia and Herzegovina', 'Botswana',
##        'Brazil', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
##        'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
##        'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia',
##        'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica',
##        "Cote d'Ivoire", 'Croatia', 'Cuba', 'Cyprus', 'Czechia', 'Denmark',
##        'Diamond Princess', 'Djibouti', 'Dominica', 'Dominican Republic',
##        'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
##        'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France',
##        'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
##        'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana',
##        'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
##        'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
##        'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
##        'Korea, North', 'Korea, South', 'Kosovo', 'Kuwait', 'Kyrgyzstan',
##        'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia', 'Libya',
##        'Liechtenstein', 'Lithuania', 'Luxembourg', 'MS Zaandam',
##        'Madagascar', 'Malawi', 'Malaysia', 'Maldives', 'Mali', 'Malta',
##        'Marshall Islands', 'Mauritania', 'Mauritius', 'Mexico',
##        'Micronesia', 'Moldova', 'Monaco', 'Mongolia', 'Montenegro',
##        'Morocco', 'Mozambique', 'Namibia', 'Nauru', 'Nepal',
##        'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria',
##        'North Macedonia', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Panama',
##        'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland',
```

```
##        'Portugal', 'Qatar', 'Romania', 'Russia', 'Rwanda',
##        'Saint Kitts and Nevis', 'Saint Lucia',
##        'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
##        'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
##        'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
##        'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
##        'Spain', 'Sri Lanka', 'Sudan', 'Summer Olympics 2020', 'Suriname',
##        'Sweden', 'Switzerland', 'Syria', 'Taiwan*', 'Tajikistan',
##        'Tanzania', 'Thailand', 'Timor-Leste', 'Togo', 'Tonga',
##        'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Tuvalu', 'US',
##        'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom',
##        'Uruguay', 'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam',
##        'West Bank and Gaza', 'Winter Olympics 2022', 'Yemen', 'Zambia',
##        'Zimbabwe'], dtype=object)
```

```
# Mostramos el número de valores únicos en la columna "Country/Region"
datosP['Country/Region'].unique().shape[0]
```

```
## 201
```

```
# Mostramos el número de valores faltantes (NaN) en la columna "Country/Region"
datosP['Country/Region'].isna().sum()
```

```
## 0
```

Al examinar los datos, identificamos 201 valores distintos para países o regiones sin valores nulos. El tipo de dato es objeto, lo cual es adecuado para campos de texto.

Es importante destacar que aparecen los valores `Summer Olympics 2020` y `Winter Olympics 2022`, los cuales corresponden a los casos de COVID-19 detectados durante los Juegos Olímpicos de Tokio 2020 y los Juegos Olímpicos de Invierno de Beijing 2022. Estos valores se midieron fuera de las estadísticas de los respectivos países anfitriones.

Asimismo, encontramos 2 cruceros: el `Diamond Princess` y el `MS Zaandam`. Es difícil determinar los posibles países de origen sin realizar un estudio exhaustivo de sus itinerarios en el transcurso del tiempo de los datos. Sin embargo, consideramos que esta tarea es desgastante y no agrega valor al estudio.

Lat Long.

Se miraran la latitud y longitud en conjunto.

```
# Mostramos estadísticas descriptivas de las columnas "Lat" y "Long"
datosP[['Lat', 'Long']].describe()
```

```
##         Lat  Long
## count  287   287
## mean    20    22
## std     26    78
## min    -72  -178
## 25%      4   -33
## 50%     22    21
## 75%     40    89
## max     72   178
```

```
# Mostramos el número de valores faltantes (NaN) en las columnas "Lat" y "Long"
datosP[['Lat', 'Long']].isna().sum()
```

```
## Lat     2
## Long    2
## dtype: int64
```

La latitud y la longitud tienen el tipo de dato int64, lo cual es apropiado ya que representan números con decimales. Se observa que hay dos observaciones sin valores de latitud y longitud, correspondientes a los casos en los que el valor de `Province/State` es igual a `Repatriated Travellers` y `Unknown`. Todas las demás observaciones tienen valores válidos dentro del rango de $\pm 90^{\circ}$ de latitud y $\pm 180^{\circ}$ de longitud.

Transformación Inicial.

Teniendo en cuenta la información obtenida de las primeras cuatro columnas del conjunto de datos, realizaremos las siguientes transformaciones iniciales. En primer lugar, listaremos como países todos aquellos territorios de Dinamarca, Francia, Países Bajos, Nueva Zelanda y Reino Unido, teniendo en cuenta que la mayoría son autónomos y/o islas en otros continentes. Esto se debe a que su dinámica de contagios no representa necesariamente al país del cual son miembros. En segundo lugar, eliminaremos la columna `Province/State`, ya que solo quedan los valores de las provincias de Australia, Canadá y China. Debido a la falta de datos de provincias para el resto de los países, no se pueden comparar con los demás. Con esta transformación, las observaciones de `Repatriated Travellers` y `Unknown` quedarán dentro de Canadá y China, respectivamente.

```
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"]
    )}
).drop("Province/State", axis=1)
```

Se realizará una transformación en los valores `Summer Olympics 2020` y `Winter Olympics 2022` para asignarlos a los países anfitriones Japón y China, respectivamente. Esta decisión se tomó debido a la imposibilidad de detectar la nacionalidad de los contagiados durante los juegos, por lo que se optó por registrarlos en los países donde se detectaron los casos. Además, se procederá a eliminar los registros de los cruceros `Diamond Princess` y `MS Zaandam` debido a la imposibilidad de homologar los orígenes de los contagios sin realizar un estudio detallado de sus itinerarios, lo cual se considera una tarea que no aporta valor al estudio.

```
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'])]
```

Por último, se realizará una reasignación de las latitudes y longitudes de Australia, Canadá, China y Japón debido al registro individual de casos en sus provincias y a la asignación de casos de los Juegos Olímpicos. Se asignarán los valores de latitud y longitud de Australian Capital Territory, Ontario y Beijing a todas las observaciones de Australia, Canadá y China, respectivamente, ya que estos territorios son donde se encuentran sus ciudades capitales. En el caso de Japón, se asignará la ubicación geográfica de Japón a ambas observaciones de los Juegos Olímpicos para asegurar que reflejen la misma ubicación.

```
# Se ajusta el valor de "Lat" y "Long" para los países "Australia", "Canada",
# "China" y "Japan".
# El valor de "Lat" y "Long" se asigna según la capital del país.
datosP_clean1 = datosP_clean1.assign(
    **{"Lat": np.select(
        [datosP_clean1["Country/Region"] == "Australia",
         datosP_clean1["Country/Region"] == "Canada",
         datosP_clean1["Country/Region"] == "China",
         datosP_clean1["Country/Region"] == "Japan"],
        [-35.473500,
          51.253800,
          40.182400,
          36.20482],
        default=datosP_clean1["Lat"])}
).assign(
    **{"Long": np.select(
        [datosP_clean1["Country/Region"] == "Australia",
         datosP_clean1["Country/Region"] == "Canada",
         datosP_clean1["Country/Region"] == "China",
         datosP_clean1["Country/Region"] == "Japan"],
        [149.012400,
         -85.323200,
         116.414200,
         138.2529],
        default=datosP_clean1["Long"])}
)
```

Transformación a forma tabular.

Como se mencionó previamente, el conjunto de datos no está en un formato tabular, lo que dificulta su análisis o uso en modelos de machine learning. Por esta razón, el siguiente paso es transformar el conjunto de datos actual de formato ancho a formato largo (tabular).

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

# 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')
```

La cantidad de días entre el 22 de enero de 2020 y el 9 de marzo de 2023 es de 1143, por lo que se esperaría que cada país tenga esa misma cantidad de observaciones en el conjunto de datos. Sin embargo, al examinar la tabla, se puede observar que los valores para Japón, Australia, Canadá y China son mayores a esa cantidad.

```
# Agrupamos los datos por país/ubicación y contamos el número de filas para cada grupo
datosP_tab.groupby(['Country/Region']).agg(
    n=('Country/Region', 'count')
    ).sort_values("n")
```

```
##                    n
## Country/Region
## Afghanistan     1143
## Nauru           1143
## Nepal           1143
## Netherlands     1143
## New Caledonia   1143
## ...              ...
## Zimbabwe        1143
## Japan           2286
## Australia       9144
## Canada         18288
## China          40005
##
## [230 rows x 1 columns]
```

La razón de esto se debe a que en la base de datos original, cada uno de estos países aparece varias veces debido a sus diferentes provincias o estados, lo que resulta en un número de observaciones mayor que el

número de días transcurridos desde el inicio del registro. Para corregir esto, es necesario agregar un paso adicional para agrupar los valores de contagios por 'País/Región', 'Latitud', 'Longitud' y 'Fecha', sumando así los casos confirmados, recuperados y muertes para cada combinación única de esas variables. De esta manera, se obtendrá un conjunto de datos tabular con una observación por país y fecha, lo que facilitará el análisis y el uso de modelos de machine learning.

```
# 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', 'Lat', 'Long', 'fecha']
    ).agg(
        {'contagios':'sum'}
    ).reset_index()
```

Con esta transformación, se ha logrado que el valor de observaciones para Japón, Australia, Canadá y China sea de 1143.

```
# Seleccionamos las filas correspondientes a los países especificados y contamos
# el número de fechas para cada país
datosP_tab[datosP_tab['Country/Region'].isin(
    ['Japan', 'Australia', 'Canada', 'China',]
    )].groupby(
        ['Country/Region']
        ).agg(n=('Country/Region', 'count'))
```

```
##                    n
## Country/Region
## Australia       1143
## Canada          1143
## China           1143
## Japan           1143
```

Exploración de datos formato tabular.

La siguiente tarea es trabajar en las columnas restantes, que son 'fecha' y 'contagios'.

fecha.

Estas serían las características de la columna `fecha`

```
# Obtener una lista de valores únicos en la columna fecha
datosP_tab['fecha'].unique()
```

```
## array(['1/1/21', '1/1/22', '1/1/23', ..., '9/9/20', '9/9/21', '9/9/22'],
##       dtype=object)
```

```
# Obtener el número total de fechas únicas en el conjunto de datos
datosP_tab['fecha'].unique().shape[0]
```

```
## 1143
```

```
# Contar el número de valores faltantes (NaN) en la columna fecha
datosP_tab['fecha'].isna().sum()
```

```
## 0
```

Podemos observar que la columna tiene 1143 valores únicos, correspondientes a los 1143 días en los que se registraron los contagios. Además, no se presentan valores nulos en esta columna. Sin embargo, se puede notar que su tipo de datos es objeto, lo cual no es apropiado ya que debería ser una variable de fecha. También se puede observar que las fechas están en formato mes, día y año. Por lo tanto, es importante tener esto en cuenta al convertir los datos a formato de fecha.

contagios.

Estas son las características de la columna`contagios`

```
# Obtener estadísticas descriptivas de la columna contagios
datosP_tab['contagios'].describe()
```

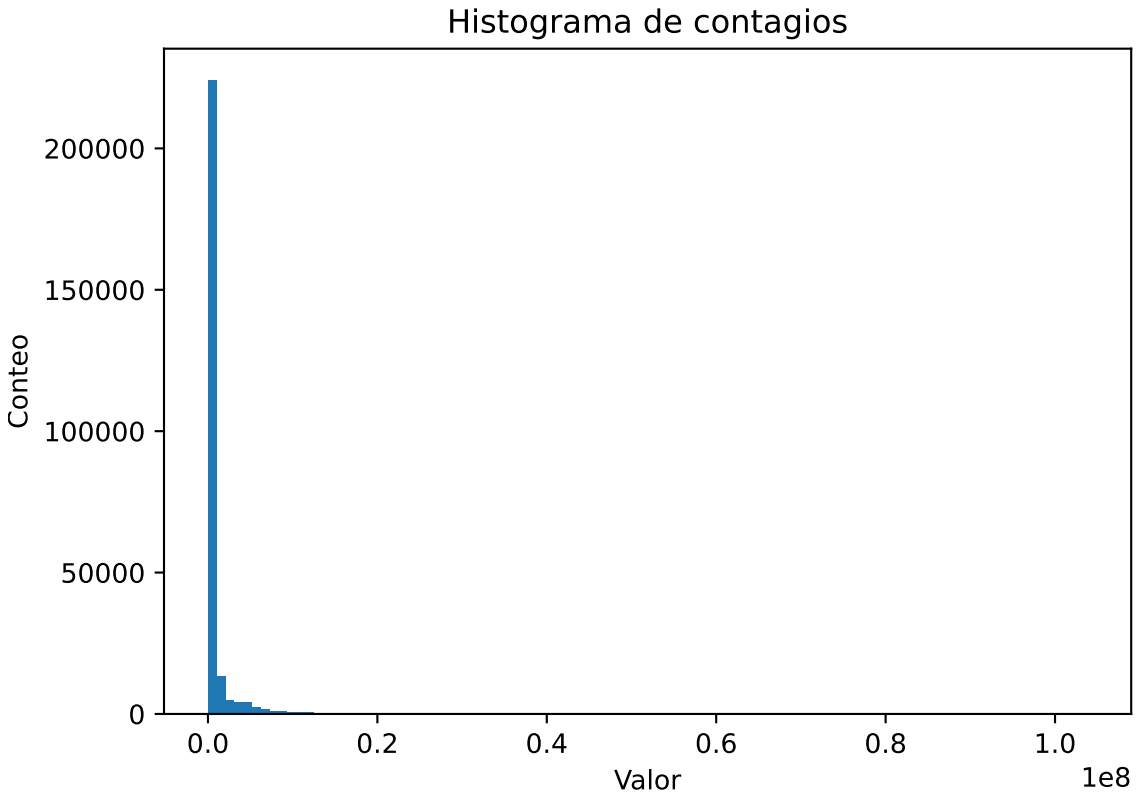
```
## count      262890
## mean      1205483
## std       5438787
## min             0
## 25%          2101
## 50%         32899
## 75%        355962
## max     103802702
## Name: contagios, dtype: float64
```

```
# Contar el número de valores faltantes (NaN) en la columna contagios
datosP_tab['contagios'].isna().sum()
```

```
## 0
```

La columna de contagios tiene un tipo de dato adecuado, float64, lo cual es apropiado para datos de conteo. Los valores medios y percentiles no brindan una gran cantidad de información en este caso, por lo que es necesario explorar la distribución de los datos de una manera más visual. Para ello, se realizará una gráfica con el fin de entender mejor la distribución de los contagios a lo largo del tiempo.

```
# Crear un histograma de la columna contagios
hist_data = plt.hist(datosP_tab['contagios'], bins = 100)
xlabel = plt.xlabel('Valor')
ylabel = plt.ylabel('Conteo')
tittle = plt.title('Histograma de contagios')
plt.show();
```




```
plt.clf()
```

De la gráfica, podemos observar que la columna “contagios” tiene una gran cantidad de valores cercanos a cero, lo cual es normal para datos de conteo. Para obtener una mejor idea del pico en este sector, realizaremos un conteo de los valores de contagio.

```
# Contar el número de ocurrencias de cada valor único en la columna contagios
datosP_tab['contagios'].value_counts()
```

```
## contagios
## 0         19611
## 1          2149
## 4          1751
## 3          1167
## 2           890
##            ...
## 20949         1
## 20083         1
## 16540         1
## 19670         1
## 256859        1
## Name: count, Length: 114172, dtype: int64
```

Podemos observar que hay una gran cantidad de observaciones con un valor de 0, lo cual es normal en datos de conteo. En este caso, alrededor del 7.4% del total de observaciones tienen un valor de 0.

Transformación fecha y contagios.

Podemos proceder a transformar los valores de la columna `fecha` a un formato de fecha utilizando la función to_datetime de pandas. Esto nos permitirá trabajar con los datos de tiempo de una manera más sencilla y efectiva.

```
# 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')
```

Tras la transformación, las fechas ahora se presentan en formato de fecha (año-mes-día), en lugar del formato de cadena de texto original

```
# Obtener un arreglo de todas las fechas únicas en la columna fecha
datosP_tab['fecha'].unique()
```

```
## <DatetimeArray>
## ['2021-01-01 00:00:00', '2022-01-01 00:00:00', '2023-01-01 00:00:00',
##  '2021-01-10 00:00:00', '2022-01-10 00:00:00', '2023-01-10 00:00:00',
##  '2021-01-11 00:00:00', '2022-01-11 00:00:00', '2023-01-11 00:00:00',
##  '2021-01-12 00:00:00',
##  ...
##  '2022-09-06 00:00:00', '2020-09-07 00:00:00', '2021-09-07 00:00:00',
##  '2022-09-07 00:00:00', '2020-09-08 00:00:00', '2021-09-08 00:00:00',
##  '2022-09-08 00:00:00', '2020-09-09 00:00:00', '2021-09-09 00:00:00',
##  '2022-09-09 00:00:00']
## Length: 1143, dtype: datetime64[ns]
```

En esta etapa inicial, se eliminarán los contagios que tengan un valor de 0, ya que este valor indica la ausencia de contagios y no aporta información relevante al análisis solicitado. Es importante destacar que, en caso de utilizar estos datos para modelar, podría resultar útil conservar los valores con valor 0, dependiendo del tipo de modelo que se vaya a implementar. Sin embargo, dado que en esta fase no se sabe si serán necesarios estos valores en el futuro, se asignarán a otro dataframe los datos sin ceros.

```
# Crear un nuevo conjunto de datos que contenga solo las filas donde contagios
# es distinto de cero
datosP_tab_non0 = datosP_tab[datosP_tab['contagios'] != 0]
```

Por último, reordenaremos el dataframe por país y fecha.

```
# 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)
```

La siguiente es la descripción del marco de datos final que utilizaremos para responder las preguntas de la FASE 1.

```
# Obtener estadísticas descriptivas del conjunto de datos
datosP_tab_non0.describe()
```

```
##          Lat   Long                          fecha  contagios
## count 243279 243279                         243279     243279
## mean      19     12  2021-09-14 11:00:34.848877568    1302659
## min      -72   -178            2020-01-22 00:00:00          1
## 25%        4    -52            2020-12-16 00:00:00       5031
## 50%       18     17            2021-09-19 00:00:00      45390
## 75%       40     45            2022-06-17 00:00:00     436630
## max       72    178            2023-03-09 00:00:00  103802702
## std       25     67                            NaN    5642547
```

```
# Obtener información sobre el conjunto de datos
datosP_tab_non0.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 243279 entries, 0 to 243278
## Data columns (total 5 columns):
##  #   Column          Non-Null Count   Dtype
## ---  ------          --------------   -----
##  0   Country/Region  243279 non-null  object
##  1   Lat             243279 non-null  float64
##  2   Long            243279 non-null  float64
##  3   fecha           243279 non-null  datetime64[ns]
##  4   contagios       243279 non-null  int64
## dtypes: datetime64[ns](1), float64(2), int64(1), object(1)
## memory usage: 9.3+ MB
```

```
# Mostrar las primeras 10 filas del conjunto de datos
datosP_tab_non0.head(10)
```

```
##   Country/Region  Lat  Long      fecha  contagios
## 0    Afghanistan   34    68 2020-02-24          5
## 1    Afghanistan   34    68 2020-02-25          5
## 2    Afghanistan   34    68 2020-02-26          5
## 3    Afghanistan   34    68 2020-02-27          5
## 4    Afghanistan   34    68 2020-02-28          5
## 5    Afghanistan   34    68 2020-02-29          5
## 6    Afghanistan   34    68 2020-03-01          5
## 7    Afghanistan   34    68 2020-03-02          5
## 8    Afghanistan   34    68 2020-03-03          5
## 9    Afghanistan   34    68 2020-03-04          5
```

```
# Obtener la forma del conjunto de datos (número de filas y columnas)
datosP_tab_non0.shape
```

```
## (243279, 5)
```

```
# Obtener los nombres de las columnas del conjunto de datos
datosP_tab_non0.columns
```

```
## Index(['Country/Region', 'Lat', 'Long', 'fecha', 'contagios'], dtype='object')
```

Preguntas de interes Caso COVID-19 Fase 1.

Ya con nuestros datos en forma tabular pasaremos a contestar las interrogantes de la fase 1.

Pregunta 1.

> *¿En cuál mes se presentó el mayor número de contagios?*

Para responder a esta pregunta, es necesario llevar a cabo ingeniería de características para obtener el recuento de contagios diarios, así como para crear una columna que indique el año y mes de cada observación. Finalmente, agregaremos los datos por año y mes.

```
# Calcular el número de contagios por día para cada país
datosP_dia = datosP_tab_non0.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 a nivel global
datosP_mes = datosP_dia.groupby(
    ['año_mes'], as_index=False
    ).agg(contagios_mes=('contagios_dia', 'sum'))
```

Con respecto al conjunto de datos `datosP_mes`, podemos determinar cuál fue el mes que registró el mayor número de contagios.

```
# Selecciona la fila correspondiente al mes con el máximo número de contagios en
# el DataFrame "datosP_mes"
datosP_mes[datosP_mes['contagios_mes'] == datosP_mes['contagios_mes'].max()]
```

```
##     año_mes  contagios_mes
## 24  2022-01       90483564
```

El mes que registró el mayor número de contagios fue enero de 2022, con un total de 90'483.564 contagios reportados durante el mes.

Pregunta 2.

> *¿En ese mismo mes, cuál fue el país que reportó más contagios?*

Para responder a esta pregunta, utilizaremos el conjunto de datos `datosP_dia` que creamos en el punto anterior. A continuación, filtraremos únicamente los datos correspondientes al año y mes `2022-01` y los agruparemos por país para obtener la suma total de contagios registrados durante ese mes.

```python
# Seleccionar las filas del DataFrame "datosP_dia" correspondientes al mes de enero
# de 2022, y agrupar los datos por país para obtener el total de contagios en
# enero de 2022 para cada país.
datosP_202201 = datosP_dia.loc[datosP_dia['año_mes'] == '2022-01'].groupby(
    ['Country/Region'], as_index=False
    ).agg(contagios_202201=('contagios_dia', 'sum'))
```

Utilizando el conjunto de datos `datosP_202201`, podemos determinar cuál fue el país que registró el mayor número de contagios durante enero de 2022.

```python
# Seleccionar la fila en "datosP_202201" donde el valor de la columna "contagios_202201"
# es igual al valor máximo de la columna "contagios_202201".
datosP_202201[datosP_202201['contagios_202201'] == datosP_202201['contagios_202201'].max()]
```

```
##     Country/Region  contagios_202201
## 209             US          20336435
```

Durante enero de 2022, Estados Unidos registró el mayor número de contagios con un total de 20'336.435 casos reportados.

Pregunta 3.

¿Cuál es el país con el menor número de casos reportados hasta la fecha?

Para responder a esta pregunta, utilizaremos nuestro conjunto de datos tabular `datosP_tab_non0`. En primer lugar, filtraremos los datos correspondientes a la fecha más reciente disponible y los ordenaremos de menor a mayor según el número total de contagios reportados. Posteriormente, seleccionaremos los 25 países con el menor número de contagios totales registrados hasta la fecha seleccionada.

```python
# Seleccionar las filas correspondientes a la fecha más reciente, y las columnas
# "Country/Region" y "contagios".
# Luego, ordenar los datos por "contagios" de forma ascendente,
# seleccionar las primeras 25 filas.
datosP_tab_non0.loc[
    datosP_tab_non0['fecha'] == datosP_tab_non0['fecha'].max(),
    ['Country/Region', 'contagios']
    ].sort_values(['contagios']).head(25).reset_index(drop=True)
```

```
##                                 Country/Region  contagios
## 0                                   Korea, North          1
## 1                                Pitcairn Islands          4
## 2                                     Antarctica         11
## 3                                       Holy See         29
## 4                                           Niue        792
## 5                                     Montserrat       1403
## 6                     Falkland Islands (Malvinas)       1930
## 7   Saint Helena, Ascension and Tristan da Cunha       2166
## 8                                         Tuvalu       2805
## 9                               Wallis and Futuna       3427
## 10                    Saint Pierre and Miquelon       3452
## 11                                      Anguilla       3904
```

```
## 12                                  Kiribati       5014
## 13                                     Nauru       5247
## 14                           Saint Barthelemy      5441
## 15                                     Palau       5991
## 16                      Sao Tome and Principe      6281
## 17                   Turks and Caicos Islands      6561
## 18                      Saint Kitts and Nevis      6597
## 19                               Cook Islands      7031
## 20                    British Virgin Islands       7305
## 21                                      Chad       7679
## 22                              Sierra Leone       7760
## 23                                   Liberia       8090
## 24                             Guinea-Bissau       8960
```

En la lista de países con menos casos de contagio, Corea del Norte ocupa el primer lugar al reportar únicamente un caso. Sin embargo, dada la falta de transparencia que históricamente ha caracterizado a este país en cuanto a la divulgación de cifras, resulta difícil verificar la veracidad de esta información. Del puesto 2 al 21 se encuentran países o territorios con poblaciones muy reducidas, inferiores a los 500,000 habitantes, lo cual dificulta su comparación con otros países de mayor tamaño poblacional. Finalmente, en el puesto 22 se encuentra Chad, el primer país o territorio con una población superior a los 500,000 habitantes.

De acuerdo con los datos que hemos analizado, Chad es el país con la menor cantidad de contagios de COVID-19 entre aquellos con una población de más de 500,000 habitantes. Sin embargo, debemos tener en cuenta que los países africanos en general han tenido un acceso limitado a las pruebas de COVID-19, como se indica en este artículo de la BBC: https://www.bbc.com/mundo/noticias-internacional-52575102. Esto ha contribuido a que las cifras de contagios reportadas en el continente sean muy bajas y no reflejen la verdadera magnitud de la pandemia en la región. Por lo tanto, aunque Chad tiene una baja cantidad de contagios registrados, es importante tener en cuenta este contexto más amplio al interpretar los datos.

Reflexion.

Importancia del uso de Pandas como herramienta de limpieza, preparación y gestion de datos en el contexto de la Ciencia de Datos.

El análisis de datos es esencial en muchas áreas, especialmente en la Ciencia de Datos. Sin embargo, antes de comenzar cualquier análisis, es crucial que los datos se limpien y preparen adecuadamente. Esta es precisamente la función de las herramientas de limpieza y preparación de datos, como la biblioteca de Python llamada Pandas.

Pandas es una herramienta muy poderosa para manipular y analizar conjuntos de datos grandes y complejos de manera eficiente y organizada. Con Pandas, los datos pueden transformarse, filtrarse y manipular de muchas maneras, lo que facilita la tarea de la limpieza y preparación de datos. Por ejemplo, Pandas permite limpiar datos faltantes o erróneos, seleccionar y filtrar datos, agrupar y agregar datos y crear nuevas columnas y variables. Estas funciones y métodos de Pandas ayudan a los científicos de datos a realizar tareas de limpieza y preparación de datos de manera más efectiva y eficiente, lo que se traduce en un análisis más rápido y preciso.

Pandas también es altamente flexible y escalable, lo que lo hace ideal para trabajar con grandes conjuntos de datos en tiempo real. La biblioteca está diseñada para manejar grandes cantidades de datos de manera eficiente, lo que permite a los usuarios analizar grandes conjuntos de datos sin problemas de rendimiento o velocidad. Además, Pandas se integra con otras bibliotecas populares de Python, como NumPy, Matplotlib y Scikit-learn, para proporcionar una solución completa para el análisis y la visualización de datos. Esta integración facilita el análisis y la visualización de datos para los científicos de datos, lo que les permite realizar análisis más complejos y sofisticados.

Pandas como herramiente de ayuda para mejorar el almacenamiento en un Data Lake.

Un Data Lake es una solución de almacenamiento de datos que permite a las empresas almacenar grandes cantidades de datos estructurados y no estructurados en un solo lugar. A medida que las empresas recopilan cada vez más datos, es importante contar con herramientas de manipulación y preparación de datos que permitan una mayor eficiencia en la organización y preparación de los datos antes de ser almacenados en un Data Lake.

En este contexto, pandas es una herramienta muy útil que puede ayudar a mejorar el almacenamiento en un Data Lake. Con su capacidad para manipular y analizar grandes cantidades de datos de una manera eficiente y organizada, pandas puede mejorar la eficiencia en la preparación de los datos en bruto antes de ser almacenados. La limpieza y el filtrado de los datos antes de ser almacenados en un Data Lake puede reducir el tamaño total de los datos y mejorar su calidad, lo que a su vez mejora la capacidad del Data Lake para proporcionar información valiosa y útil para la toma de decisiones.

Además de la limpieza y el filtrado de los datos, pandas también puede ser utilizado para transformar y enriquecer los datos antes de ser almacenados en un Data Lake. Al realizar estas transformaciones y enriquecimientos de los datos antes de ser almacenados, se puede mejorar la calidad de los datos y aumentar su valor para la empresa.