Documentación: modelado de datos

Fuente de datos: https://data.cdc.gov/Case-Surveillance/COVID-19-Case-Surveillance-Public-Use-Data/vbim-akqf/about_data

Github con código: https://github.com/WendyGalvis/Modelado_Covid19.git

1. Información general

Este proyecto tiene como objetivo analizar los datos públicos de vigilancia de casos de COVID-19 en Estados Unidos, proporcionados por el CDC. El propósito es identificar tendencias clave en la propagación del virus, así como analizar las diferencias en la afectación según edad, género, raza, comorbilidades y su impacto en la gravedad de los casos. A través de visualizaciones interactivas, buscamos facilitar la interpretación de estos datos para el público en general, investigadores y tomadores de decisiones en salud pública.

La fuente original contiene un total de 106 millones de registros y 12 variables, que se describen a continuación:

Nombre de la Columna	Descripción	Tipo de Dato
cdc_case_earliest_dt	La fecha más temprana entre la Fecha Clínica (fecha relacionada con la enfermedad o la recolección de la muestra) o la Fecha de Recepción por el CDC. Fecha calculada: usa la mejor fecha disponible relacionada con la enfermedad/muestra o la fecha relacionada con el reporte del caso. Si no hay fecha disponible, se deja en blanco.	Floating Timestamp
cdc_report_dt	Fecha en la que el caso fue reportado por primera vez al CDC. Fecha calculada: Descontinuada; se recomienda usar cdc_case_earliest_dt para series temporales y otros análisis. Se llenó usando la fecha en que un registro de caso fue enviado por primera vez a la base de datos. Si falta, se usó la fecha en la que el caso apareció por primera vez en la base de datos.	Floating Timestamp
pos_spec_dt	Fecha de la primera recolección de muestra positiva (Formulario de Reporte de Caso)	Floating Timestamp

onset_dt	Fecha de inicio de los síntomas, si es sintomático (Formulario de Reporte de Caso)	Floating Timestamp
current_status	Estado del caso (Formulario de Reporte de Caso: ¿Cuál es el estado actual de esta persona?) -- Valores: Caso confirmado en laboratorio; Caso probable; Consulte la última definición de caso CSTE para más información.	Texto
sex	Sexo (Formulario de Reporte de Caso): Masculino; Femenino; Desconocido; Otro; Faltante; NA	Texto
age_group	Grupo de edad: 0 - 9 años; 10 - 19 años; 20 - 39 años; 40 - 49 años; 50 - 59 años; 60 - 69 años; 70 - 79 años; 80+ años; Faltante; NA. Las categorías de grupo de edad fueron completadas usando el valor de edad reportado en el formulario de reporte de caso. Se usó la fecha de nacimiento para completar valores de edad faltantes/desconocidos.	Texto
race_ethnicity_combined	Raza y etnia (combinados): Americano Nativo/Alaska Nativo, No Hispano; Asiático, No Hispano; Negro, No Hispano; Múltiple/Otro, No Hispano; Nativo Hawaiano/Otro Isleño del Pacífico, No Hispano; Blanco, No Hispano; Hispano/Latino; Desconocido; Faltante; NA. Si se reportaron más de una raza, se clasificó como múltiples/otras razas.	Texto
hosp_yn	Estado de hospitalización (Formulario de Reporte de Caso: ¿El paciente fue hospitalizado?) -- Valores: Sí; No; Desconocido; Faltante;	Texto
icu_yn	Estado de admisión en la UCI (Formulario de Reporte de Caso: ¿El paciente fue admitido en una unidad de cuidados intensivos (UCI)?) -- Valores: Sí; No; Desconocido; Faltante;	Texto
death_yn	Estado de muerte (Formulario de Reporte de Caso: ¿El paciente murió como resultado de esta enfermedad?) -- Valores: Sí; No; Desconocido; Faltante;	Texto

medcond_yn	Presencia de comorbilidad o enfermedad preexistente (Formulario de Reporte de Caso: ¿Condiciones médicas preexistentes?) -- Valores: Sí; No; Desconocido; Faltante;	Texto

Los datos se recolectaron entre enero de 2020 y julio de 2024.

2. Extracción y procesamiento inicial

Para extraer la información tenemos dos opciones:

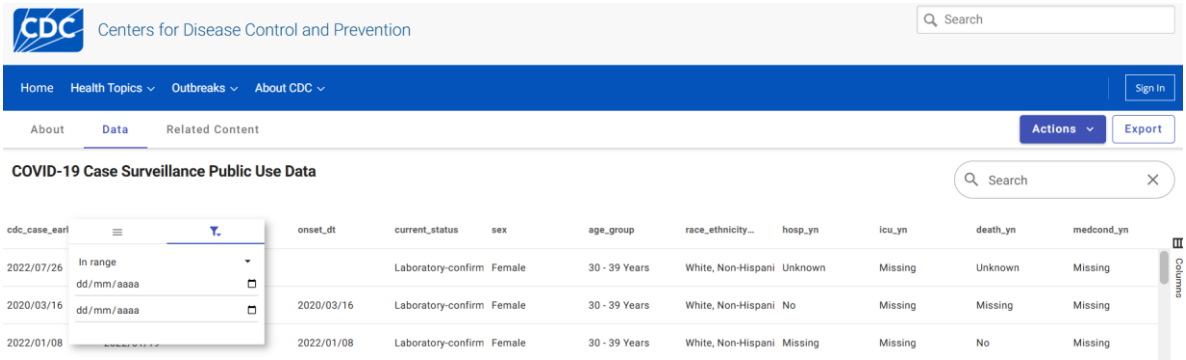
- Extraer usando una API
- Descargar los datos en CSV

Dado que la base de datos dejó de actualizarse en 2024 y no es dinámica, la utilización de la API no ofrece ventajas significativas. Además, el tiempo necesario para la extracción a través de la API sería considerablemente alto debido a la gran cantidad de datos disponibles. Por lo tanto, se optó por descargar los datos en formato CSV y limitar el análisis al año 2020.

Debido al tamaño de la base de datos, fue necesario descargar los datos en varios periodos y luego combinarlos. El proceso fue el siguiente:

- En la opción Data, se filtró la variable cdc_case_earliest_dt en los siguientes periodos:
  Enero a marzo
  Abril a junio
  Julio a agosto
  Septiembre a octubre
  Noviembre

Diciembre



Este proceso se hizo para cada uno de los periodos descritos y en cada uno se seleccionó la opción “Export”, ubicada en la esquina superior derecha.

Posteriormente, utilizando la herramienta **Google Colab**, se inició con el procesamiento de la información, el código completo se puede visualizar en: https://github.com/WendyGalvis/Modelado_Covid19/blob/main/Preprocesamiento_COVID19.ipynb

En primer lugar, se cargaron todos los archivos descargados (Archivos>Subir al almacenamiento de infomación>Seleccionar archivos). Asimismo, se importó la librería pandas, que fue fundamental para el procesamiento y análisis de la información:

Importar librerías y datos

```
import pandas as pd
```

```
df_ene_mar = pd.read_csv("Ene-mar.csv")
df_abr_jun = pd.read_csv("Abr-jun.csv")
df_jul_ago = pd.read_csv("Jul-Ago.csv")
df_sep_oct = pd.read_csv("Sep-oct.csv")
df_nov = pd.read_csv("nov.csv")
df_dic = pd.read_csv("dic.csv")
```

A continuación, se procedió a revisar la información contenida en cada uno de los dataframes.

```
df_ene_mar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238500 entries, 0 to 238499
Data columns (total 12 columns):
 #   Column                   Non-Null Count   Dtype 
---  ------                   --------------   ----- 
 0   cdc_case_earliest_dt     238500 non-null  object
 1   cdc_report_dt            236553 non-null  object
 2   pos_spec_dt              136376 non-null  object
 3   onset_dt                 166301 non-null  object
 4   current_status           238500 non-null  object
 5   sex                      238500 non-null  object
 6   age_group                238500 non-null  object
 7   race_ethnicity_combined  238500 non-null  object
 8   hosp_yn                  238500 non-null  object
 9   icu_yn                   238500 non-null  object
 10  death_yn                 238500 non-null  object
 11  medcond_yn               238500 non-null  object
```

La suma de los registros de todos los dataframes tiene más de 18 millones de registros.

Luego de esto, se revisaron los valores nulos. Se encontró que muchas de las columnas de fecha contienen valores nulos:

Valores nulos

```
In [ ]:
```

```
print(df_ene_mar.isnull().sum())
```

```
cdc_case_earliest_dt         0
cdc_report_dt             1947
pos_spec_dt             102124
onset_dt                 72199
current_status               0
sex                          0
age_group                    0
race_ethnicity_combined      0
hosp_yn                      0
icu_yn                       0
death_yn                     0
medcond_yn                   0
dtype: int64
```

En este punto, se decidió realizar una eliminación preliminar de las variables 'cdc_report_dt', 'pos_spec_dt', 'onset_dt' y 'icu_yn', ya que no están relacionadas con el propósito narrativo de nuestro tablero o contienen demasiados valores nulos. Esta acción nos permitió centrarnos en la exploración y limpieza de las variables relevantes para el análisis que deseamos mostrar en el tablero.

```
df_1 = df_ene_mar.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
df_2 = df_abr_jun.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
df_3 = df_jul_ago.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
df_4 = df_sep_oct.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
df_5 = df_nov.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
df_6 = df_dic.drop(columns=['cdc_report_dt', 'pos_spec_dt', 'onset_dt', 'icu_yn'])
```

A continuación, se llevó a cabo una validación de los valores en las variables categóricas, ya que, en ocasiones, no se presentan valores nulos, pero los registros incluyen valores inválidos o aquellos que representan información ausente. A continuación, se muestran los resultados:

```
Valores únicos en current_status: ['Probable Case' 'Laboratory-confirmed case']
Valores únicos en sex: ['Male' 'Female' 'Unknown' 'Missing' 'Other']
Valores únicos en age_group: ['10 - 19 Years' '20 - 29 Years' '40 - 49 Years' '60 - 69 Year
s'
 '50 - 59 Years' '70 - 79 Years' '80+ Years' 'Missing' '30 - 39 Years'
 '0 - 9 Years']
Valores únicos en race_ethnicity_combined: ['Black, Non-Hispanic' 'Hispanic/Latino' 'White,
Non-Hispanic'
 'Asian, Non-Hispanic' 'American Indian/Alaska Native, Non-Hispanic'
 'Unknown' 'Multiple/Other, Non-Hispanic' 'Missing'
 'Native Hawaiian/Other Pacific Islander, Non-Hispanic']
Valores únicos en hosp_yn: ['Missing' 'No' 'Unknown' 'Yes']
Valores únicos en death_yn: ['No' 'Yes' 'Missing' 'Unknown']
Valores únicos en medcond_yn: ['Missing' 'Unknown' 'Yes' 'No']
```

La imagen anterior revela que algunas variables contienen valores vacíos, etiquetados como "Missing", o valores desconocidos, etiquetados como "Unknown".

Para la limpieza inicial de los datos y considerando que nuestro principal interés radica en analizar los casos graves (muertes y hospitalizaciones), se decidió eliminar todos los registros con valores desconocidos o vacíos en las variables 'hosp_yn' y 'death_yn' en cada uno de los dataframes.

```
# Eliminar filas donde 'hosp_yn' o 'death_yn' contienen 'Missing' o 'Unknown'
df_1_depurada = df_1[~(df_1['hosp_yn'].isin(['Missing', 'Unknown']) | df_1['death_yn'].isin(['Missing', 'Unknown']))]
df_2_depurada = df_2[~(df_2['hosp_yn'].isin(['Missing', 'Unknown']) | df_2['death_yn'].isin(['Missing', 'Unknown']))]
df_3_depurada = df_3[~(df_3['hosp_yn'].isin(['Missing', 'Unknown']) | df_3['death_yn'].isin(['Missing', 'Unknown']))]
df_4_depurada = df_4[~(df_4['hosp_yn'].isin(['Missing', 'Unknown']) | df_4['death_yn'].isin(['Missing', 'Unknown']))]
df_5_depurada = df_5[~(df_5['hosp_yn'].isin(['Missing', 'Unknown']) | df_5['death_yn'].isin(['Missing', 'Unknown']))]
df_6_depurada = df_6[~(df_6['hosp_yn'].isin(['Missing', 'Unknown']) | df_6['death_yn'].isin(['Missing', 'Unknown']))]
```

Como resultado de esta limpieza, la base de datos se redujo considerablemente, quedando con 5.783.932 registros. Finalmente, se unieron todos los dataframes en uno solo y se exportó a un archivo CSV denominado "datos".

```
datos = pd.concat([df_1_depurada, df_2_depurada, df_3_depurada, df_4_depurada, df_5_depurada, df_6_depurada], ignore_index=
```

```
# Exportar el DataFrame a un archivo CSV
datos.to_csv('datos.csv', index=False)
```

3. Exploración y limpieza de datos

Con los datos obtenidos en el paso anterior, creamos un nuevo notebook (https://github.com/WendyGalvis/Modelado_Covid19/blob/main/ETL_COVID19.ipynb) para trabajar con los datos definitivos que utilizaremos en nuestro tablero. En primer lugar, importamos las librerías necesarias y cargamos los datos:

```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

```
In [ ]:
df = pd.read_csv("datos.csv")
```

A continuación, se realizó una exploración general de los datos, destacándose lo siguiente:

- Todas las variables son categóricas, excepto 'cdc_case_earliest_dt', que contiene fechas.
- La mayoría de los registros corresponden a casos confirmados, en mujeres de entre 10 y 29 años, de raza blanca, y que no fueron casos graves (hospitalización o muerte).

	cdc_case_earliest_dt	current_status	sex	age_group	race_ethnicity_combined	hosp_yn	death_yn	medcond_yn
count	5783932	5783932	5783932	5783932	5783932	5783932	5783932	5783932
unique	360	2	5	10	9	2	2	4
top	2020/11/30	Laboratory-confirmed case	Female	20 - 29 Years	White, Non-Hispanic	No	No	Missing
freq	59783	5270869	3065510	1012120	2646505	5161470	5492275	3986504

Para facilitar el manejo de los datos, se simplificaron los nombres de las columnas, cambiándolos por versiones más sencillas y en español:

```
df = df.rename(columns={'cdc_case_earliest_dt ': 'Fecha'})
df = df.rename(columns={'current_status': 'Estado'})
df = df.rename(columns={'sex': 'Sexo'})
df = df.rename(columns={'age_group': 'Edad'})
df = df.rename(columns={'race_ethnicity_combined': 'Raza'})
df = df.rename(columns={'hosp_yn': 'Hospitalización'})
df = df.rename(columns={'death_yn': 'Muerte'})
df = df.rename(columns={'medcond_yn': 'Comorbilidad'})
```

De igual forma, las categorías de las variables se tradujeron al español y se unificaron las categorías 'Unknown' y 'Missing' bajo la etiqueta "Desconocido", ya que, para los efectos de nuestro análisis, ambas categorías representan información desconocida. Para esto, se creó un diccionario que luego se aplicó a cada una de las columnas, como se muestra a continuación:

```
traduccion = {
    'Estado': {
        'Laboratory-confirmed case': 'Confirmado',
        'Probable Case': 'probable'
    },
    'Sexo': {
        'Male': 'Masculino',
        'Female': 'Femenino',
        'Unknown': 'Desconocido',
        'Missing': 'Desconocido',
        'Other': 'Otro'
    },
    'Edad': {
        '20 - 29 Years': '20 - 29 Años',
        '40 - 49 Years': '40 - 49 Años',
        '60 - 69 Years': '60 - 69 Años',
        '50 - 59 Years': '50 - 59 Años',
        '30 - 39 Years': '30 - 39 Años',
        'Missing': 'Desconocido',
        '10 - 19 Years': '10 - 19 Años',
        '70 - 79 Years': '70 - 79 Años',
        '80+ Years': '80+ Años',
        '0 - 9 Years': '0 - 9 Años'
    },
    'Raza': {
        'Hispanic/Latino': 'Hispano/Latino',
        'White, Non-Hispanic': 'Blanco, No Hispano',
        'Black, Non-Hispanic': 'Negro, No Hispano',
        'Asian, Non-Hispanic': 'Asiático, No Hispano',
        'Unknown': 'Desconocido',
        'Multiple/Other, Non-Hispanic': 'Múltiple/Otro, No Hispano',
        'Missing': 'Desconocido',
        'American Indian/Alaska Native, Non-Hispanic': 'Indígena Americano/Nativo de Alaska, No Hispano',
        'Native Hawaiian/Other Pacific Islander, Non-Hispanic': 'Nativo Hawaiano/Otro Isleño del Pacífico, No Hispano'
    },

    'Hospitalización': {
        'No': 'No',
        'Yes': 'Sí'
    },
    'Muerte': {
        'No': 'No',
        'Yes': 'Sí'
    },
    'Comorbilidad': {
        'Missing': 'Desconocido',
        'Yes': 'Sí',
        'No': 'No',
        'Unknown': 'Desconocido'
    }
}

# Aplicar las traducciones a df
for column, trad in traduccion.items():
    df[column] = df[column].replace(trad)
```

Finalmente, se ajustó el formato de la columna fecha:

```
df['Fecha'] = pd.to_datetime(df['Fecha'], format='%Y/%m/%d')
print(df.dtypes)
print(df.head())
```

```
Fecha              datetime64[ns]
Estado                     object
Sexo                       object
Edad                       object
Raza                       object
Hospitalización            object
Muerte                     object
Comorbilidad               object
```

4. Identificar características y relaciones entre variables

Antes de proceder con el diseño de nuestras visualizaciones, quisimos explorar de manera general el contenido de cada una de las variables y las relaciones que puedan existir entre ellas.

El detalle de las visualizaciones se puede observar en:
https://github.com/WendyGalvis/Modelado_Covid19/blob/main/ETL_COVID19.ipynb

Inicialmente, se generó un gráfico para cada variable, utilizando las librerías seaborn y matplotlib. Para ello, se creó una función llamada estadísticas_descriptivas que toma como parámetro un dataframe de pandas (df). La función itera sobre cada una de las columnas, calcula la frecuencia de los valores (counts = df[x].value_counts()) y, si el número de valores únicos es mayor a 6, genera un histograma; de lo contrario, se genera un gráfico de barras.

```
def estadisticas_descriptivas(df):
    for x in df.columns:
        print(f"-------Estadísticas descriptivas de {x}-----------")
        counts = df[x].value_counts()
        if len(counts) > 6:
            sb.histplot(df[x])
            plt.title(f"Distribución de: {x}")
            plt.show()
            print(f"Moda: {df[x].mode().values}")
        else:
            plt.figure(figsize=(8, 6))
            sb.barplot(x=counts.index, y=counts.values)
            plt.title(f"Distribución de: {x}")
            plt.show()
            print(f"Moda: {df[x].mode().values}")
estadisticas_descriptivas(df)
```

Como resultado de este ejercicio, pudimos sacar algunas conclusiones generales de la información:

- Los casos reportados aumentaron con el paso del tiempo, alcanzando su pico máximo en noviembre de 2020.
- La mayoría de los registros corresponde a casos confirmados.
- Hay más casos de hombres que de mujeres.
- El rango de edad con más casos reportados es entre 20 y 29 años.
- La mayoría de los casos no llegaron a ser graves (muertes u hospitalizaciones).
- No hay mucha información de comorbilidad.

Dado que nuestro principal interés es analizar los casos graves y su relación con las variables disponibles, decidimos crear visualizaciones que muestren la relación entre cada una de las variables y la cantidad de muertes y hospitalizaciones. Para ello, se crearon dos nuevos dataframes: uno para las muertes y otro para las hospitalizaciones. Además, se

desarrollaron dos funciones: una para generar gráficos de barras y otra para crear gráficos de líneas, que luego se utilizarán según las necesidades de nuestro análisis.

```
# Filtramos el DataFrame para quedarnos solo con muertes y hospitalizaciones
df_muertes = df[df['Muerte'] == 'Sí']
df_hospitalizaciones = df[df['Hospitalización'] == 'Sí']


# Función para crear un gráfico de barras
def crear_grafico_barras(df, columna, titulo, xlabel, ylabel):
    plt.figure(figsize=(10, 6))
    sb.countplot(data=df, x=columna, palette='Set2')
    plt.title(titulo)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

# Función para crear un gráfico de línea  por fecha
def crear_grafico_linea(df, columna_fecha, titulo, xlabel, ylabel):
    casos_por_fecha = df.groupby(columna_fecha).size()
    plt.figure(figsize=(10, 6))
    plt.plot(casos_por_fecha.index, casos_por_fecha.values, marker='o', color='b', linestyle='-', markersize=5)
    plt.title(titulo)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# 1. Crear el gráfico de línea de muertes y hospitalizaciones por fecha
crear_grafico_linea(df_muertes, 'Fecha', 'Cantidad de muertes por fecha', 'Fecha', 'Número de muertes')
crear_grafico_linea(df_hospitalizaciones, 'Fecha', 'Cantidad de hospitalizaciones por fecha', 'Fecha', 'Número de hospi

# 2. Crear gráficos de barras para Sexo, Edad y Raza
crear_grafico_barras(df_muertes, 'Sexo', 'Muertes por sexo', 'Sexo', 'Número de muertes')
crear_grafico_barras(df_muertes, 'Edad', 'Muertes por edad', 'Edad', 'Número de muertes')
crear_grafico_barras(df_muertes, 'Raza', 'Muertes por raza', 'Raza', 'Número de muertes')
```

A partir de este análisis, se encontraron algunos hallazgos importantes:

- Aunque el número total de casos aumentó de enero a diciembre, las muertes y hospitalizaciones alcanzaron sus picos en marzo, julio y diciembre.
- Aunque la mayoría de los casos fueron reportados en mujeres, los hombres tuvieron una mayor cantidad de muertes y hospitalizaciones.
- La mayor cantidad de muertes y hospitalizaciones ocurrió en personas de edad avanzada.

5. Cargue de datos

Al intentar cargar los datos, se descubrió que su tamaño superaba los 500 MB. Sin embargo, en Looker Studio existe un límite de 100 MB, por lo que, para poder trabajar de manera adecuada con la información, se seleccionó aleatoriamente el 15% de los datos, lo que nos permitió continuar con el análisis.

```
data = pd.read_csv("data.csv")
covid = data.sample(frac=0.15, random_state=42)
```

Los datos finales se encuentran en: Datos

6. Modelado en Looker Studio

Por facilidad, en looker Studio se configuraron las variables que no representaban una dificultad mayor.

Particularmente, se creo una variable para medir la cantidad de muertes y otra para la cantidad de hospitalizaciones. Para esto, se agrego un nuevo campo y se configuró de la siguiente forma: