Misión 2: Herramientas y Software para Análisis de Datos

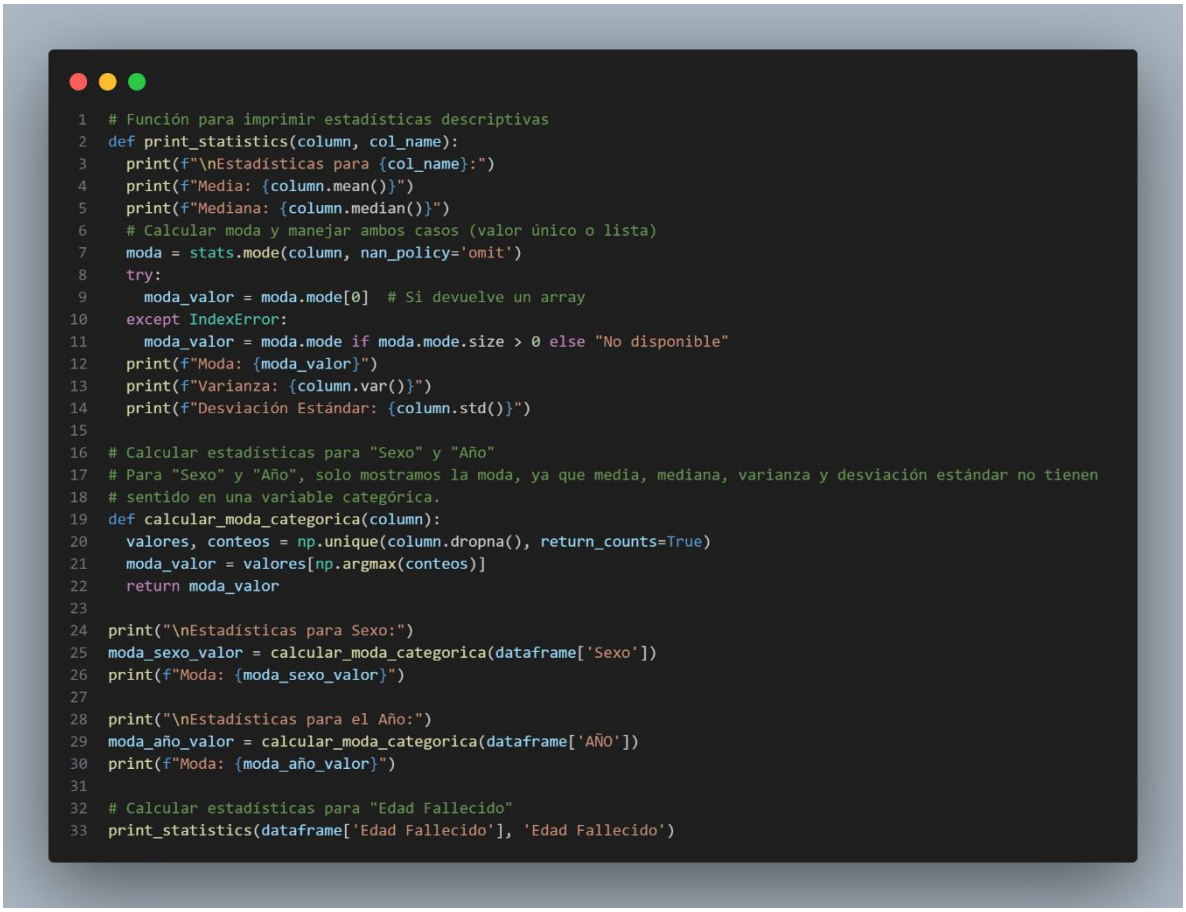
Adriana Lucia González Ardila
Anthoni Lexandre Hernández Diaz
Carlos Alberto Robayo Melendez
José Arley León Méndez

Sector de Análisis y Justificación:
Para el desarrollo de la Misión 2: Herramientas y Software para Análisis de Datos se analizan los datos con información sobre la cantidad de defunciones ocurridas durante los años 2018 a 2023 en una clínica del Área Metropolitana de Bucaramanga, la cantidad de defunciones por sexo y la edad de las personas en el momento de su defunción. Esta información es muy importante para el país dado que a partir de ella se establecen estadísticas respecto a tasa de mortalidad como un indicador de salud pública que permite a las instituciones del estado conocer los problemas de salud que afectan a la población, gestionar los diversos servicios de salud y establecer políticas que mejoren la calidad de vida de los habitantes.

Programa en Python para el cargue de los datos

```python
# Importar las librerías necesarias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
from scipy import stats

# 1. Cargar los datos desde un archivo Excel
df = pd.read_excel(r'c:\Carlos A Robayo M\Analitica de Datos Bootcamp\MORTALIDAD_2018a2022.xlsx', sheet_name='MORTALIDAD')

# Convertir las columnas "Edad Fallecido" y "AÑO" a enteros
df['Edad Fallecido'] = df['Edad Fallecido'].astype('Int64')
df['AÑO'] = df['AÑO'].astype('Int64')

# Selección de las columnas que se desea analizar
dataframe = df[['Edad Fallecido', 'AÑO', 'Sexo']]

# Eliminar filas donde el año es 2017, 2018 o 2014
dataframe = dataframe[~dataframe['AÑO'].isin([2016, 2017, 2024])]
```

Calcular las siguientes variables estadísticas de tendencia central:



```python
# Función para imprimir estadísticas descriptivas
def print_statistics(column, col_name):
  print(f"\nEstadísticas para {col_name}:")
  print(f"Media: {column.mean()}")
  print(f"Mediana: {column.median()}")
  # Calcular moda y manejar ambos casos (valor único o lista)
  moda = stats.mode(column, nan_policy='omit')
  try:
    moda_valor = moda.mode[0]  # Si devuelve un array
  except IndexError:
    moda_valor = moda.mode if moda.mode.size > 0 else "No disponible"
  print(f"Moda: {moda_valor}")
  print(f"Varianza: {column.var()}")
  print(f"Desviación Estándar: {column.std()}")

# Calcular estadísticas para "Sexo" y "Año"
# Para "Sexo" y "Año", solo mostramos la moda, ya que media, mediana, varianza y desviación estándar no tienen
# sentido en una variable categórica.
def calcular_moda_categorica(column):
  valores, conteos = np.unique(column.dropna(), return_counts=True)
  moda_valor = valores[np.argmax(conteos)]
  return moda_valor

print("\nEstadísticas para Sexo:")
moda_sexo_valor = calcular_moda_categorica(dataframe['Sexo'])
print(f"Moda: {moda_sexo_valor}")

print("\nEstadísticas para el Año:")
moda_año_valor = calcular_moda_categorica(dataframe['AÑO'])
print(f"Moda: {moda_año_valor}")

# Calcular estadísticas para "Edad Fallecido"
print_statistics(dataframe['Edad Fallecido'], 'Edad Fallecido')
```

La edad media de una persona en el momento de su defunción es de 72,31 años; la edad de mayor frecuencia de las personas en el momento de su defunción es de 79 años y la edad que ocupa el lugar central de todos los datos cuando éstos están ordenados de menor a mayor es de 75 años.

Estadísticas para Edad Fallecido:
Media: 72.31820848816741
Mediana: 75.0
Moda: 79

La moda es que el sexo de la persona fallecida sea MASCULINO.
Estadísticas para Sexo:
Moda: MASCULINO

La moda para el mayor numero de fallecidos es el año 2021.
Estadísticas para el Año:
Moda: 2021

Calcular las siguientes variables estadísticas de dispersión:

Varianza: La medida de dispersión que representa la variabilidad de la serie de datos de la edad de las personas en el momento de su defunción con respecto a su media es de 224.87.

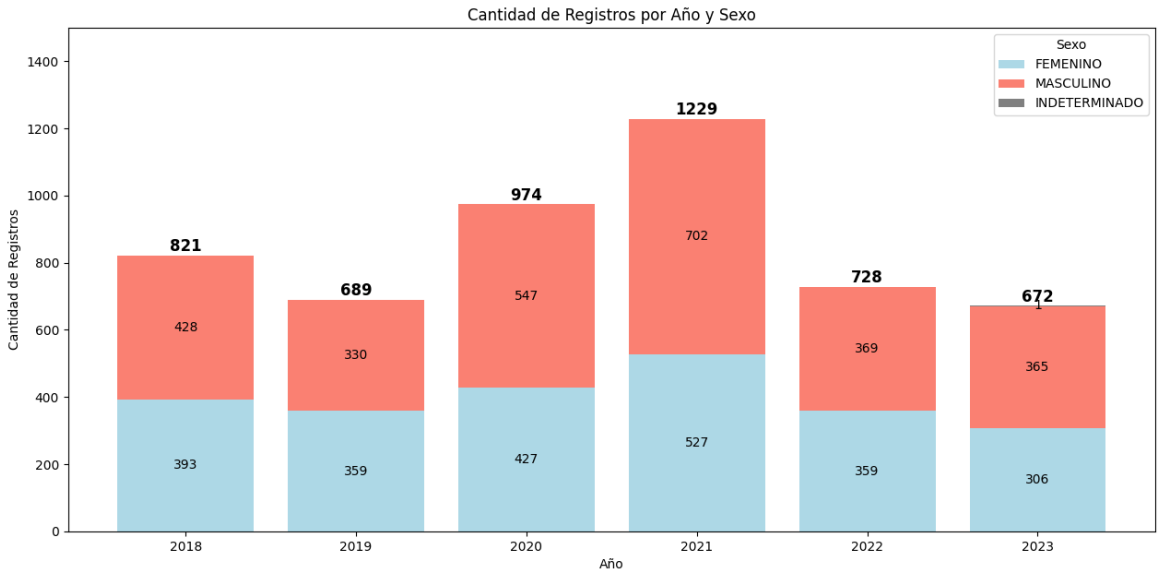
Desviación Estándar: La desviación estándar de la serie de datos de la edad de las personas en el momento de su defunción es de 14.99. Esta medida nos permite medir la dispersión de los valores en un conjunto de datos.

Estadísticas para Edad Fallecido:
Varianza: 224.8730975723301
Desviación Estándar: 14.995769322456587

Realizar las siguientes graficas en Python

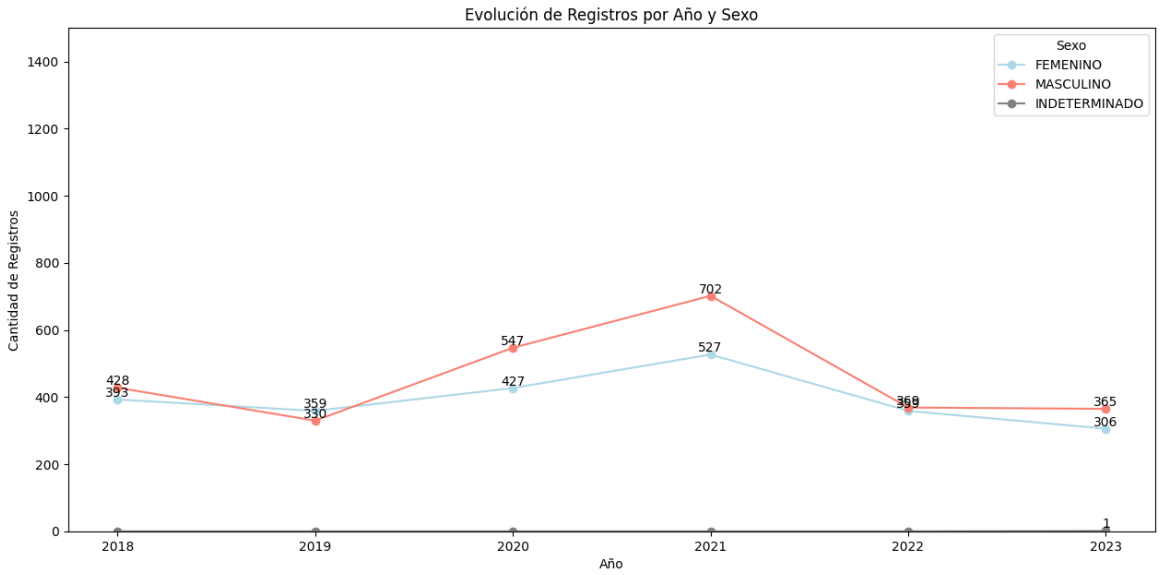
- **Gráficos de barras**

```python
# Agrupar por "AÑO" y "Sexo", y contar las ocurrencias
barras = dataframe.groupby(['AÑO', 'Sexo']).size().unstack(fill_value=0)

# Agregar una columna "Total" con la suma de los registros por año
barras['Total'] = barras.sum(axis=1)

#Data de la lista a usar con año sexo y total
Data = barras.reset_index()

# Configuración del gráfico de barras
fig, ax = plt.subplots(figsize=(12, 8))

# Crear las barras apiladas
ax.bar(Data['AÑO'], Data['FEMENINO'], label='FEMENINO', color='lightblue')
ax.bar(Data['AÑO'], Data['MASCULINO'], bottom=Data['FEMENINO'], label='MASCULINO', color='salmon')

# Agregar la categoria 'INDETERMINADO' solo si existe
if 'INDETERMINADO' in Data.columns:
  ax.bar(Data['AÑO'], Data['INDETERMINADO'],
  bottom=Data['FEMENINO'] + Data['MASCULINO'],
  label='INDETERMINADO', color='gray')

# Añadir etiquetas de valor para FEMENINO, MASCULINO e INDETERMINADO
for i, year in enumerate(Data['AÑO']):
  fem_value = Data.loc[i, 'FEMENINO']
  masc_value = Data.loc[i, 'MASCULINO']
  total_value = Data.loc[i, 'Total']

  # Etiqueta para FEMENINO
  ax.text(year, fem_value / 2, str(fem_value), ha='center', va='center', color='black', fontsize=10)

  # Etiqueta para MASCULINO, apilada sobre FEMENINO
  ax.text(year, fem_value + (masc_value / 2), str(masc_value), ha='center', va='center', color='black', fontsize=10)

  # Etiqueta para INDETERMINADO si existe
  if 'INDETERMINADO' in Data.columns:
    indet_value = Data.loc[i, 'INDETERMINADO']
    if indet_value > 0:
      ax.text(year, fem_value + masc_value + (indet_value / 2), str(indet_value), ha='center', va='center', color='black', fontsize=10)

  # Etiqueta del total encima de la barra
  ax.text(year, total_value + 2, str(total_value), ha='center', va='bottom', color='black', fontsize=12, fontweight='bold')

# Etiquetas y titulo
ax.set_xlabel('Año')
ax.set_ylabel('Cantidad de Registros')
ax.set_title('Cantidad de Registros por Año y Sexo')
ax.legend(title='Sexo')

# Configurar límites del eje y
ax.set_ylim(0, 1500)

# Mostrar el gráfico de barras
plt.tight_layout()
plt.show()
```



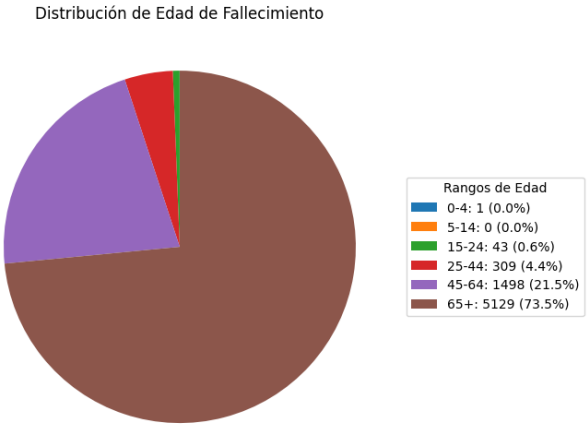
- **Gráficos de líneas**

```python
# Gráfico de líneas
fig, ax = plt.subplots(figsize=(12, 8))

# Crear las líneas para cada categoría de sexo
ax.plot(Data['AÑO'], Data['FEMENINO'], marker='o', color='lightblue', label='FEMENINO')
ax.plot(Data['AÑO'], Data['MASCULINO'], marker='o', color='salmon', label='MASCULINO')

# Agregar la categoría 'INDETERMINADO' solo si existe
if 'INDETERMINADO' in Data.columns:
  ax.plot(Data['AÑO'], Data['INDETERMINADO'], marker='o', color='gray', label='INDETERMINADO')

# Etiquetas de valor en cada punto
for i, year in enumerate(Data['AÑO']):
  ax.text(year, Data.loc[i, 'FEMENINO'], str(Data.loc[i, 'FEMENINO']), ha='center', va='bottom', color='black')
  ax.text(year, Data.loc[i, 'MASCULINO'], str(Data.loc[i, 'MASCULINO']), ha='center', va='bottom', color='black')

  if 'INDETERMINADO' in Data.columns and Data.loc[i, 'INDETERMINADO'] > 0:
    ax.text(year, Data.loc[i, 'INDETERMINADO'], str(Data.loc[i, 'INDETERMINADO']), ha='center', va='bottom', color='black')

# Configurar límites del eje y y etiquetas
ax.set_ylim(0, 1500)
ax.set_xlabel('Año')
ax.set_ylabel('Cantidad de Registros')
ax.set_title('Evolución de Registros por Año y Sexo')
ax.legend(title='Sexo')

# Mostrar el gráfico de lineas
plt.tight_layout()
plt.show()
```



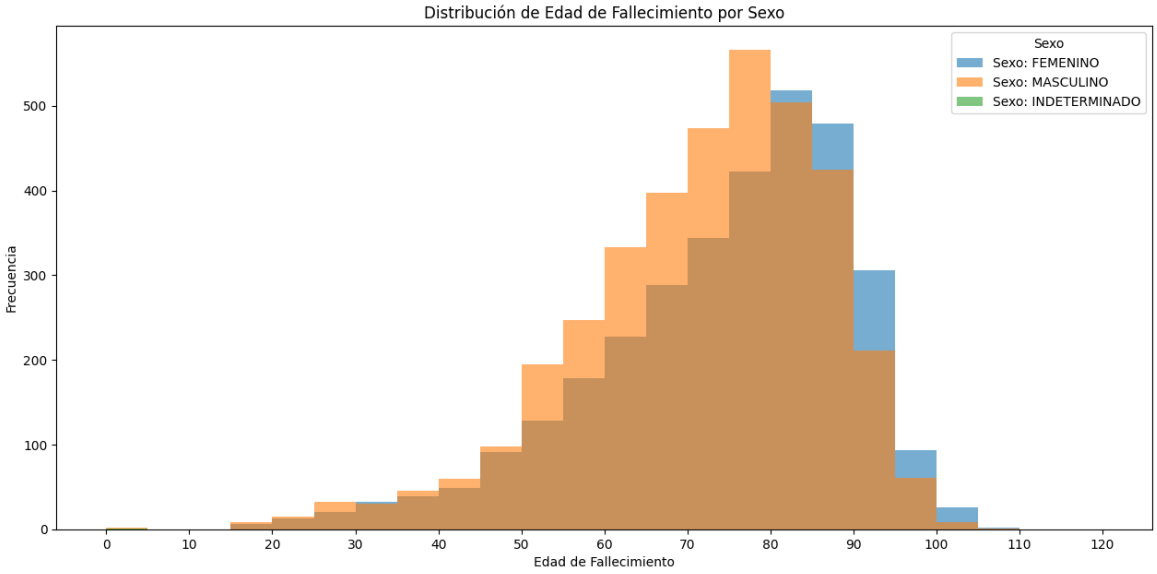
- **Gráficos de pastel**

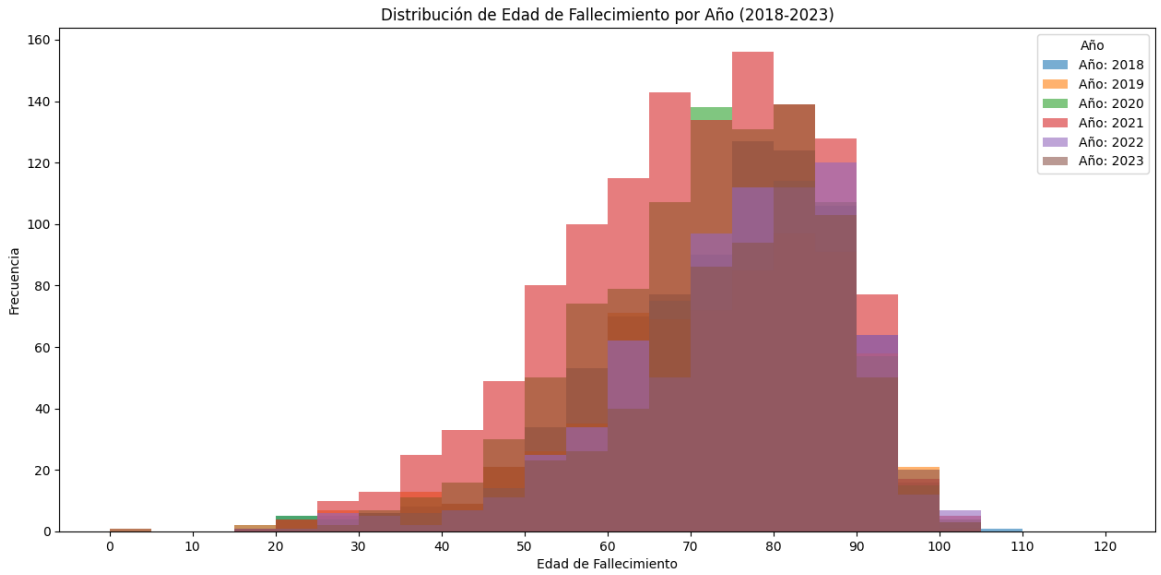
```python
# Grafico de Torta
# 1. Reemplazar valores en blanco en 'Edad Fallecido' con 0
dataframe['Edad Fallecido'] = dataframe['Edad Fallecido'].fillna(0)

# Definir los rangos de edad con un máximo de 120
bins = [0, 4, 14, 24, 44, 64, 120]
labels = ['0-4', '5-14', '15-24', '25-44', '45-64', '65+']
df['Rango de Edad'] = pd.cut(df['Edad Fallecido'], bins=bins, labels=labels, right=True)

# Contar la cantidad de ocurrencias en cada rango de edad
age_group_counts = df['Rango de Edad'].value_counts().sort_index()

# Crear el gráfico de torta sin etiquetas directamente en la torta
fig, ax = plt.subplots(figsize=(12, 8))
wedges, _ = ax.pie(age_group_counts, labels=['']*len(age_group_counts), startangle=90)

# Añadir el porcentaje en la leyenda
legend_labels = [f"{label}: {count} ({count / age_group_counts.sum():.1%})" for label, count in zip(age_group_counts.index, age_group_counts)]
plt.legend(wedges, legend_labels, title="Rangos de Edad", bbox_to_anchor=(1, 0.5), loc="center left")

# Titulo y ajuste de layout
plt.title('Distribución de Edad de Fallecimiento')
# plt.tight_layout()
plt.show()
```


Distribución de Edad de Fallecimiento
Rangos de Edad
0-4: 1 (0.0%)
5-14: 0 (0.0%)
15-24: 43 (0.6%)
25-44: 309 (4.4%)
45-64: 1498 (21.5%)
65+: 5129 (73.5%)

- **Histograma**

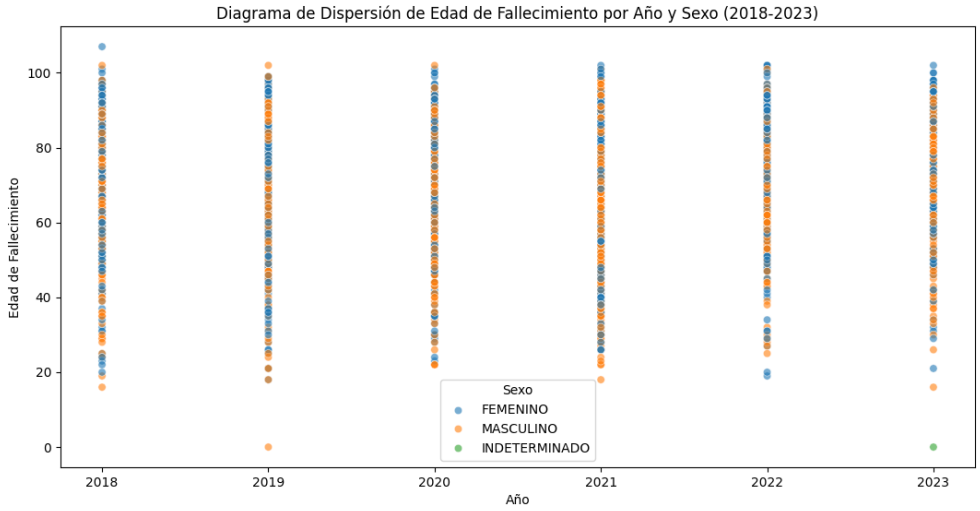
```python
# Histograma
# Asegurar que 'Edad Fallecido' es numérico y eliminar valores nulos
df['Edad Fallecido'] = pd.to_numeric(df['Edad Fallecido'], errors='coerce').fillna(0).astype(int)

# Crear el histograma de edad, separado por sexo
plt.figure(figsize=(12, 8))
for sexo in df['Sexo'].unique():
  subset = df[df['Sexo'] == sexo]
  plt.hist(subset['Edad Fallecido'], bins=range(0, 121, 5), alpha=0.6, label=f'Sexo: {sexo}')

# Configuración del gráfico
plt.xlabel('Edad de Fallecimiento')
plt.ylabel('Frecuencia')
plt.title('Distribución de Edad de Fallecimiento por Sexo')
plt.legend(title='Sexo')
plt.xticks(range(0, 121, 10))
plt.tight_layout()
plt.show()

# Histograma de edad y año
# Filtrar los datos para solo incluir los años 2018 a 2023
df_filtered = df[(df['AÑO'] >= 2018) & (df['AÑO'] <= 2023)]

# Crear el histograma de edad, separado por año (solo de 2018 a 2023)
plt.figure(figsize=(12, 8))
for year in sorted(df_filtered['AÑO'].unique()):
  subset = df_filtered[df_filtered['AÑO'] == year]
  plt.hist(subset['Edad Fallecido'], bins=range(0, 121, 5), alpha=0.6, label=f'Año: {year}')

# Configuración del gráfico
plt.xlabel('Edad de Fallecimiento')
plt.ylabel('Frecuencia')
plt.title('Distribución de Edad de Fallecimiento por Año (2018-2023)')
plt.legend(title='Año')
plt.xticks(range(0, 121, 10))
plt.tight_layout()
plt.show()
```





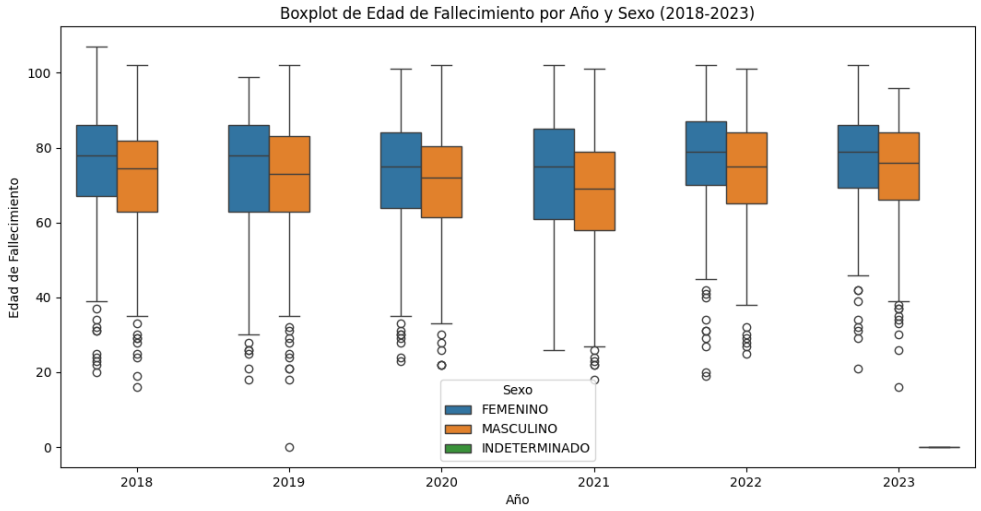
- **Diagramas de dispersión**

```python
# Grafico de dispersion
# Filtrar los datos para los años 2018 a 2023
df_filtered = df[(df['AÑO'] >= 2018) & (df['AÑO'] <= 2023)]

# Crear el diagrama de dispersión
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df_filtered, x='AÑO', y='Edad Fallecido', hue='Sexo', alpha=0.6)
plt.xlabel('Año')
plt.ylabel('Edad de Fallecimiento')
plt.title('Diagrama de Dispersión de Edad de Fallecimiento por Año y Sexo (2018-2023)')
plt.legend(title='Sexo')
plt.show()
```



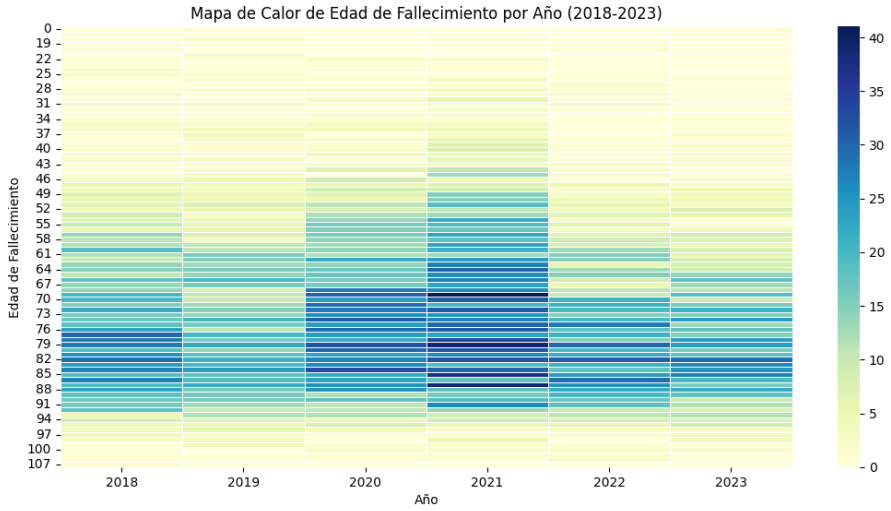
- **Boxplots**

```python
# Grafico de boxpot
plt.figure(figsize=(12, 8))
sns.boxplot(data=df_filtered, x='AÑO', y='Edad Fallecido', hue='Sexo')
plt.xlabel('Año')
plt.ylabel('Edad de Fallecimiento')
plt.title('Boxplot de Edad de Fallecimiento por Año y Sexo (2018-2023)')
plt.legend(title='Sexo')
plt.show()
```



- **Mapas de calor**

```python
# Grafico de calor
# Crear una tabla de frecuencias entre edad y año
heatmap_data = pd.crosstab(df_filtered['Edad Fallecido'], df_filtered['AÑO'])

# Crear el mapa de calor
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, cmap='YlGnBu', linewidths=0.5)
plt.xlabel('Año')
plt.ylabel('Edad de Fallecimiento')
plt.title('Mapa de Calor de Edad de Fallecimiento por Año (2018-2023)')
plt.show()
```



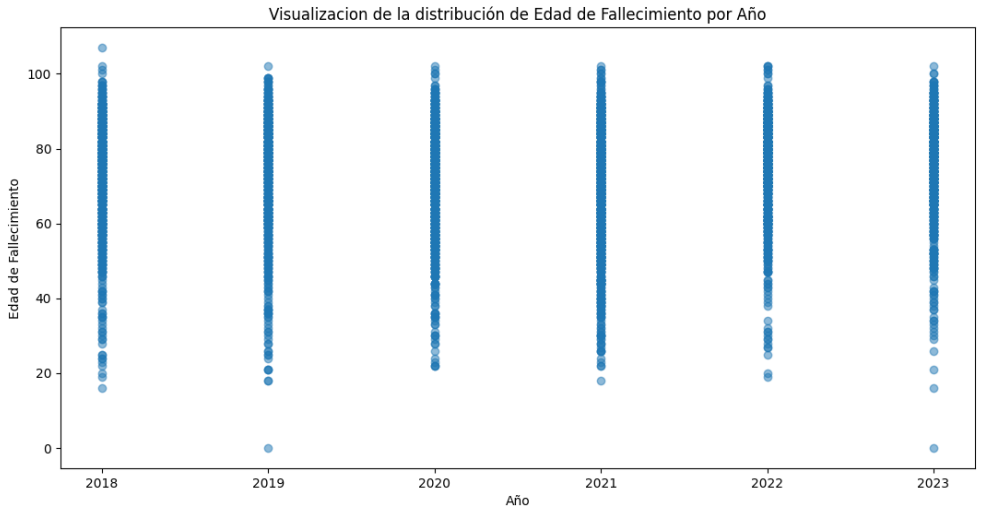
8. Realizar un modelo de regresión lineal

- **Cargar lo datos en un dataframe**

```python
# Cargar los datos en un DataFrame
# Filtrar los datos para los años 2018 a 2023
df_filtered = df[(df['AÑO'] >= 2018) & (df['AÑO'] <= 2023)]

# Seleccionar las columnas para el modelo
df_model = df_filtered[['AÑO', 'Edad Fallecido']]
```

- **Visualizar los datos**

```python
# Visualizar los datos
plt.figure(figsize=(12, 8))
plt.scatter(df_model['AÑO'], df_model['Edad Fallecido'], alpha=0.5)
plt.xlabel('Año')
plt.ylabel('Edad de Fallecimiento')
plt.title('Visualizacion de la distribución de Edad de Fallecimiento por Año')
plt.show()
```

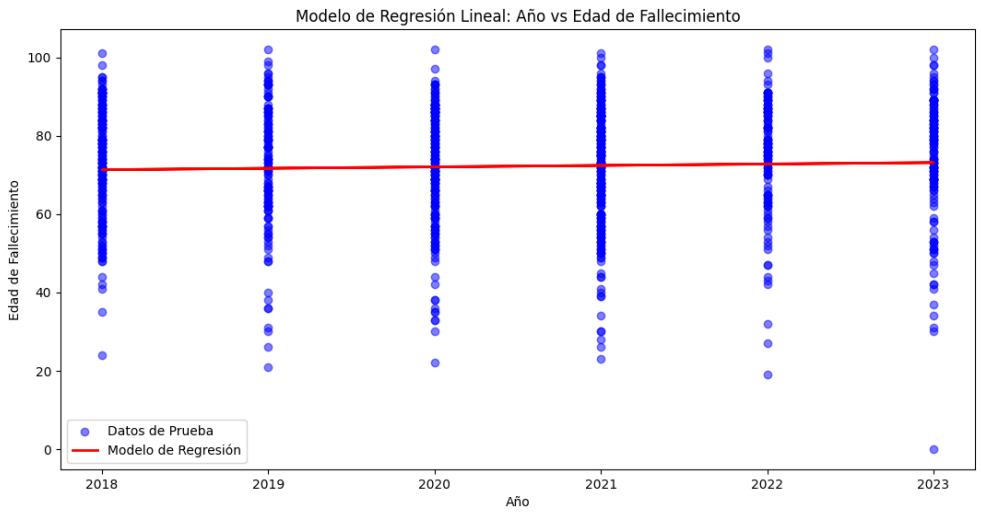


- **Dividir los datos en conjuntos de entrenamiento y prueba**

```python
# Dividir los datos en conjuntos de entrenamiento y prueba
from sklearn.model_selection import train_test_split

X = df_model[['AÑO']]
y = df_model['Edad Fallecido']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **Crear y entrenar el modelo de regresión lineal**

```python
# Crear y entrenar el modelo de regresión lineal
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```



- **Hacer predicciones**

```python
# Hacer predicciones
y_pred = model.predict(X_test)

# Hacer la predicción para el año 2024 usando un DataFrame para conservar el nombre de la característica
edad_2024 = model.predict(pd.DataFrame([[2024]], columns=['AÑO']))
print(f"Predicción de la edad de fallecimiento para el año 2024: {edad_2024[0]:.2f}")

# Hacer la predicción para el año 2025 usando un DataFrame para conservar el nombre de la característica
edad_2025 = model.predict(pd.DataFrame([[2025]], columns=['AÑO']))
print(f"Predicción de la edad de fallecimiento para el año 2025: {edad_2025[0]:.2f}")
```

La edad de una persona en el momento de su defunción para el año 2024 será de 73,51 años y para el año 2025 será de 73,87 años.

Predicción de la edad de fallecimiento para el año 2024: 73.51
Predicción de la edad de fallecimiento para el año 2025: 73.87

- **Evaluar el modelo**

```python
# Evaluar el modelo
# Calcular el Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calcular el Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calcular el R2 Score
r2 = r2_score(y_test, y_pred)

print(f"\n\nMean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R2 Score: {r2}\n\n")

# Visualizar el modelo y las predicciones
plt.figure(figsize=(12, 8))
plt.scatter(X_test, y_test, color='blue', alpha=0.5, label='Datos de Prueba')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Modelo de Regresión')
plt.xlabel('Año')
plt.ylabel('Edad de Fallecimiento')
plt.title('Modelo de Regresión Lineal: Año vs Edad de Fallecimiento')
plt.legend()
plt.show()
```

- **Calcular el Mean Squared Error**

Cuanto menor sea el MSE, mejor será la precisión predictiva del modelo.
Mean Squared Error (MSE): 229.12684675665454

- **Calcular el Root Mean Squared Error**

Un valor RMSE más bajo indica que el modelo es más preciso y se ajusta bien a los datos.
Root Mean Squared Error (RMSE): 15.136936505008356

- **Calcular el R2 Score**

Es una medida estadística utilizada para determinar la proporción de varianza en una variable dependiente que puede predecirse o explicarse mediante una variable independiente. Un valor R-cuadrado de 0 significa que el modelo explica o predice el 0% de la relación entre las variables dependientes e independientes.
R2 Score: -0.0005196781800487216