Compartir

Autor: German Preciat; Universidad de Guadalajara

Carrera: Ing. Biomédica

Materia: Analisis de datos clínicos

Contacto: german.preciat@academicos.udg.mx

Para el campo del análisis de datos, compartir nuestros hallazgos es una parte fundamental del proceso. Comunicar de manera efectiva los resultados de nuestros analisis no solo nos permite validar nuestras conclusiones, sino que también contribuye al avance del conocimiento en la comunidad científica o a la toma de decisiones basadas en datos.

Importancia de compartir descubrimientos

- **Validación de resultados**: Compartir nuestros hallazgos nos permite someter nuestras conclusiones a la revisión de expertos y colegas, lo que ayuda a validar la calidad y la robustez de nuestros análisis.
- **Replicabilidad y transparencia**: Al compartir nuestros datos y metodologías, otros investigadores tienen la oportunidad de replicar nuestros análisis, lo que aumenta la transparencia y la confiabilidad de la investigación.
- **Colaboración y retroalimentación**: Compartir nuestros hallazgos fomenta la colaboración entre investigadores y permite recibir retroalimentación constructiva, lo que puede enriquecer nuestro trabajo y llevar a nuevas ideas y descubrimientos.
- **Impacto social y científico**: La divulgación de resultados puede tener un impacto significativo en la sociedad al informar decisiones políticas, médicas o sociales basadas en evidencia científica.

Experimento sobre Cirrosis Hepática

En el contexto específico de un experimento sobre cirrosis hepática, compartir nuestros descubrimientos es crucial por varias razones:

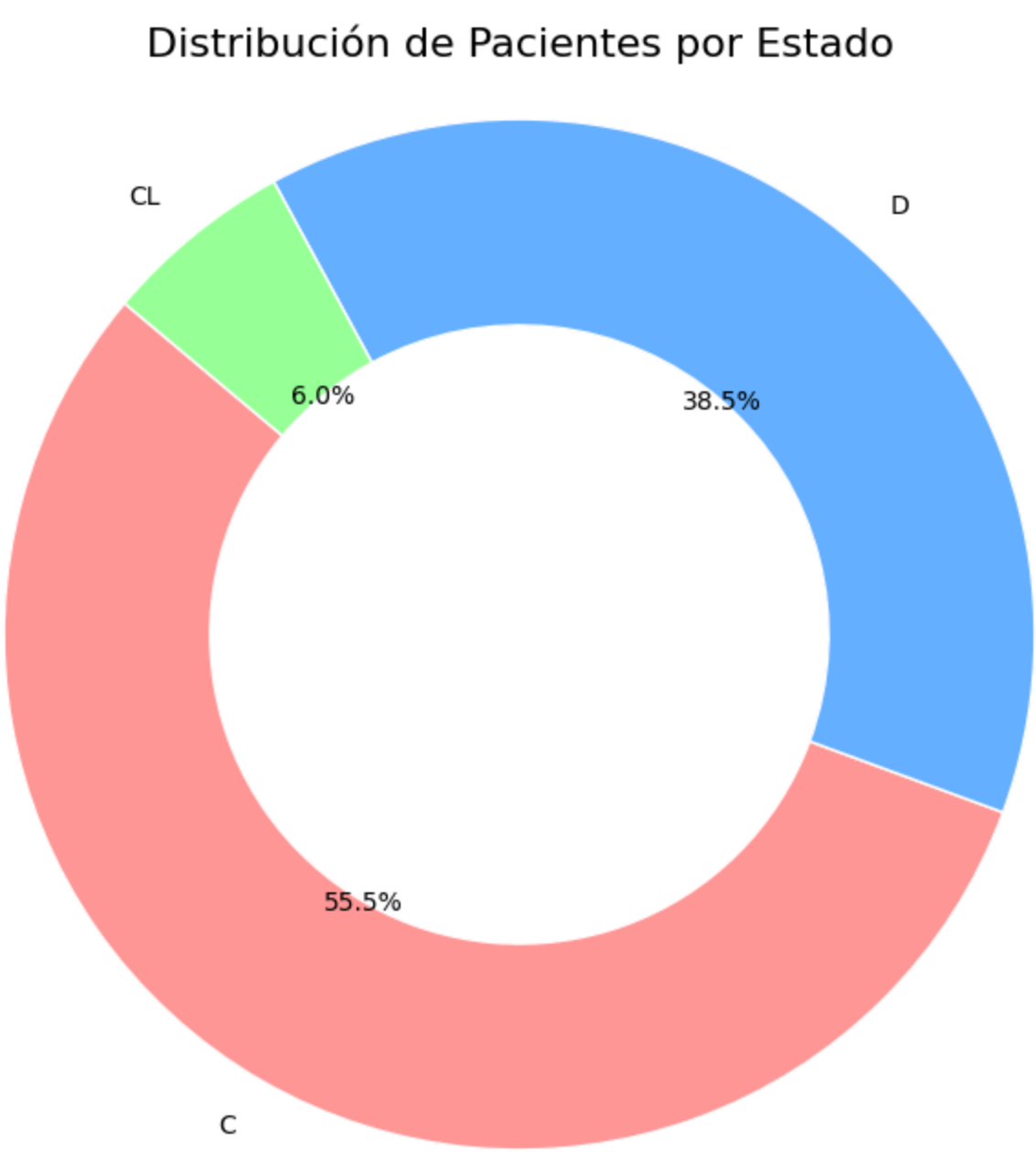
- **Mejor Comprensión de la Enfermedad**: Compartir datos y análisis relacionados con la cirrosis hepática puede ayudar a mejorar nuestra comprensión de la enfermedad, sus factores de riesgo y su progresión.
- **Desarrollo de Tratamientos y Políticas de Salud**: Los hallazgos obtenidos pueden proporcionar información valiosa para el desarrollo de tratamientos más efectivos, así como para la implementación de políticas de salud pública destinadas a prevenir y controlar la cirrosis hepática.

- **Concienciación y Prevención**: Compartir información sobre los factores asociados con la cirrosis hepática puede contribuir a la concienciación pública sobre la importancia de hábitos de vida saludables y la detección temprana de la enfermedad.
- **Avance Científico**: Al compartir nuestros datos y análisis con la comunidad científica, podemos contribuir al avance del conocimiento en el campo de la hepatología y la medicina en general.

La gráfica de pie, también conocida como gráfica circular o gráfica de pastel, es una forma de visualización que muestra proporciones o porcentajes relativos utilizando sectores circulares. Cada sector representa una categoría o grupo de datos y el tamaño del sector es proporcional a la cantidad de datos en ese grupo. Son útiles para mostrar la distribución de una variable categórica en un conjunto de datos y facilitan la comparación visual de las proporciones entre categorías.

Compararemos las personas en cada Status de la siguiente manera:

```
import matplotlib.pyplot as plt

# Contar la cantidad de pacientes en cada estado
status_counts = df['Status'].value_counts()

# Colores personalizados para cada estado
colors = ['#FF9999', '#66B2FF', '#99FF99']  # Puedes cambiar los colores según tus p

# Etiquetas para cada estado
status_labels = ['C', 'D', 'CL']

# Crear la gráfica de pastel
plt.figure(figsize=(8, 8))
plt.pie(status_counts, labels=status_labels, colors=colors, autopct='%1.1f%%', start

# Añadir título
plt.title('Distribución de Pacientes por Estado', fontsize=16)

# Mostrar la gráfica
plt.axis('equal')  # Para asegurar que el gráfico sea circular
plt.show()
```



- **Importar bibliotecas:** Importamos las bibliotecas necesarias para graficar: `seaborn` para el estilo y `matplotlib.pyplot` para la creación de la gráfica.
- **Configurar el estilo de Seaborn:** Establecemos el estilo de la gráfica utilizando `sns.set_style("whitegrid")` para un fondo blanco con una cuadrícula.
- **Contar la cantidad de personas por estado:** Usamos `value_counts()` para contar cuántas personas pertenecen a cada estado en la columna 'Status' y almacenamos los resultados en `status_counts`.
- **Definir colores personalizados:** Creamos una paleta de colores utilizando `sns.color_palette('pastel')`. La longitud de la paleta se ajusta para coincidir con la cantidad de estados. Esto garantiza que cada estado tenga un color único.

- **Crear la gráfica de pastel:** Utilizamos `plt.pie()` para generar la gráfica de pastel. Pasamos los conteos de estado, las etiquetas personalizadas y los colores correspondientes como argumentos. `autopct='%1.1f%%'` agrega etiquetas de porcentaje a cada sector y `startangle=140` gira la gráfica para que comience en un ángulo específico.
- **Añadir título:** Agregamos un título a la gráfica utilizando `plt.title()`.

1. **Relación entre el tiempo transcurrido desde el registro y el estado del paciente:**

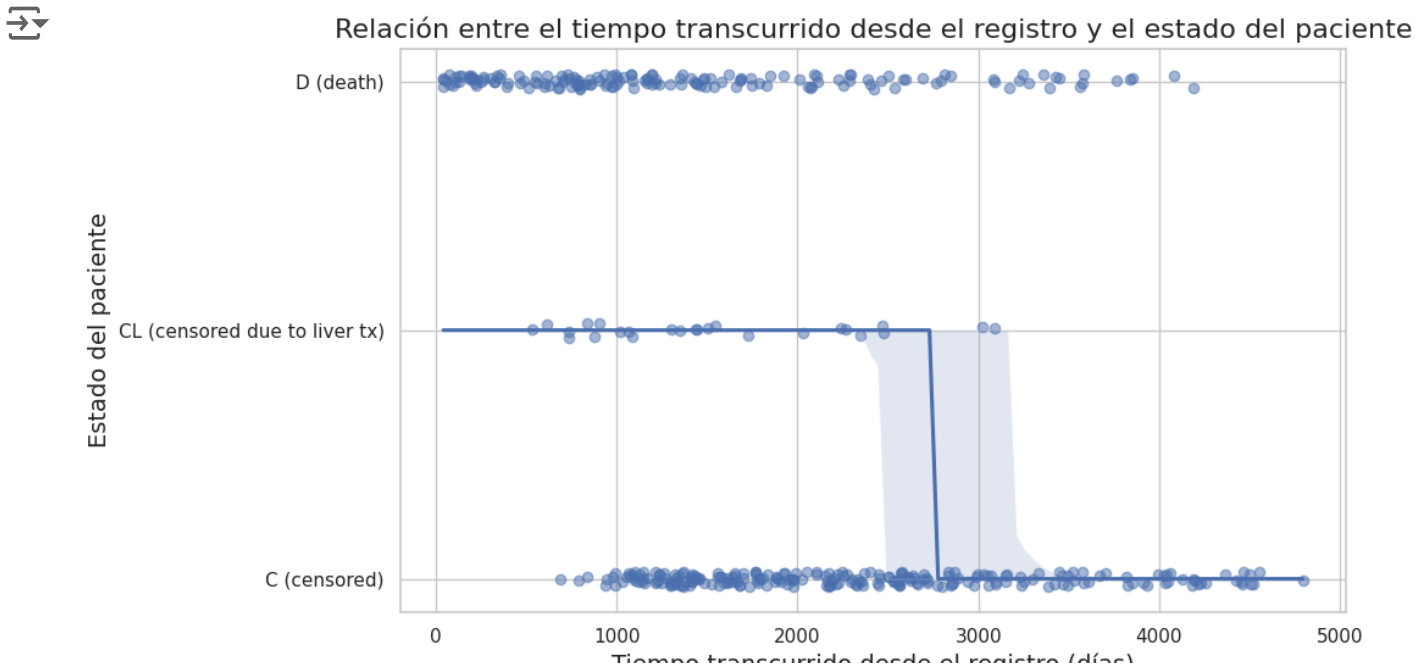
La gráfica de dispersión es una representación visual que muestra la relación entre dos variables numéricas. Cada punto en la gráfica representa una observación en el conjunto de datos y está ubicado en un plano cartesiano, donde uno de los ejes corresponde a una variable y el otro eje corresponde a la otra variable. La posición de cada punto en el gráfico indica los valores de las dos variables para esa observación en particular.

En el contexto de este análisis, la gráfica de dispersión se utilizará para visualizar cómo se distribuyen los pacientes en función del tiempo transcurrido desde el registro y su estado. En el eje X, representaremos el tiempo transcurrido desde el registro, mientras que en el eje Y representaremos el estado del paciente. Cada punto en la gráfica corresponderá a un paciente individual y su posición mostrará cuánto tiempo ha pasado desde el registro y cuál es su estado.

Al visualizar esta relación en una gráfica de dispersión, podremos identificar patrones o tendencias, como si ciertos estados están asociados con tiempos más largos o más cortos desde el registro. Esto nos ayudará a comprender mejor cómo el tiempo transcurrido está relacionado con el estado del paciente y nos proporcionará información importante para nuestro análisis.

```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

# Ignorar temporalmente las advertencias de desbordamiento
warnings.filterwarnings("ignore", category=RuntimeWarning)

# Configuración de estilo de Seaborn
sns.set(style="whitegrid")

# Convertir la columna 'Status' a valores numéricos
status_mapping = {'C': 0, 'CL': 1, 'D': 2}
df['Status'] = df['Status'].map(status_mapping)

# Crear un gráfico de dispersión con una línea de regresión
plt.figure(figsize=(10, 6))
sns.regplot(x='N_Days', y='Status', data=df, logistic=True, y_jitter=0.03, scatter_k

# Personalizar el gráfico
```

```
plt.title('Relación entre el tiempo transcurrido desde el registro y el estado del p
plt.xlabel('Tiempo transcurrido desde el registro (días)', fontsize=14)
plt.ylabel('Estado del paciente', fontsize=14)
plt.yticks([0, 1, 2], ['C (censored)', 'CL (censored due to liver tx)', 'D (death)']

warnings.filterwarnings("default", category=RuntimeWarning)
```



Aquí está la explicación línea por línea:

- `import seaborn as sns` : Importa la librería Seaborn, que se utiliza para crear gráficos estadísticos más atractivos y informativos.
- `import matplotlib.pyplot as plt` : Importa la sublibrería Pyplot de Matplotlib, que se utiliza para crear y personalizar gráficos.
- `import warnings` : Importa el módulo de advertencias de Python, que permite controlar el comportamiento de las advertencias en el código.
- `warnings.filterwarnings("ignore", category=RuntimeWarning)` : Ignora temporalmente las advertencias de tipo `RuntimeWarning` . Esto es útil si hay advertencias que no son críticas para el análisis y se desean ocultar temporalmente.
- `sns.set(style="whitegrid")` : Establece el estilo de Seaborn en "whitegrid", que agrega líneas de cuadrícula blancas al fondo del gráfico para una mejor visualización de los datos.
- `status_mapping = {'C': 0, 'CL': 1, 'D': 2}` : Crea un diccionario que asigna cada estado de paciente ('C', 'CL', 'D') a un valor numérico (0, 1, 2).
- `df['Status'] = df['Status'].map(status_mapping)` : Mapea los valores de la columna 'Status' del DataFrame `df` según el diccionario `status_mapping` , convirtiendo así los estados del paciente en valores numéricos.

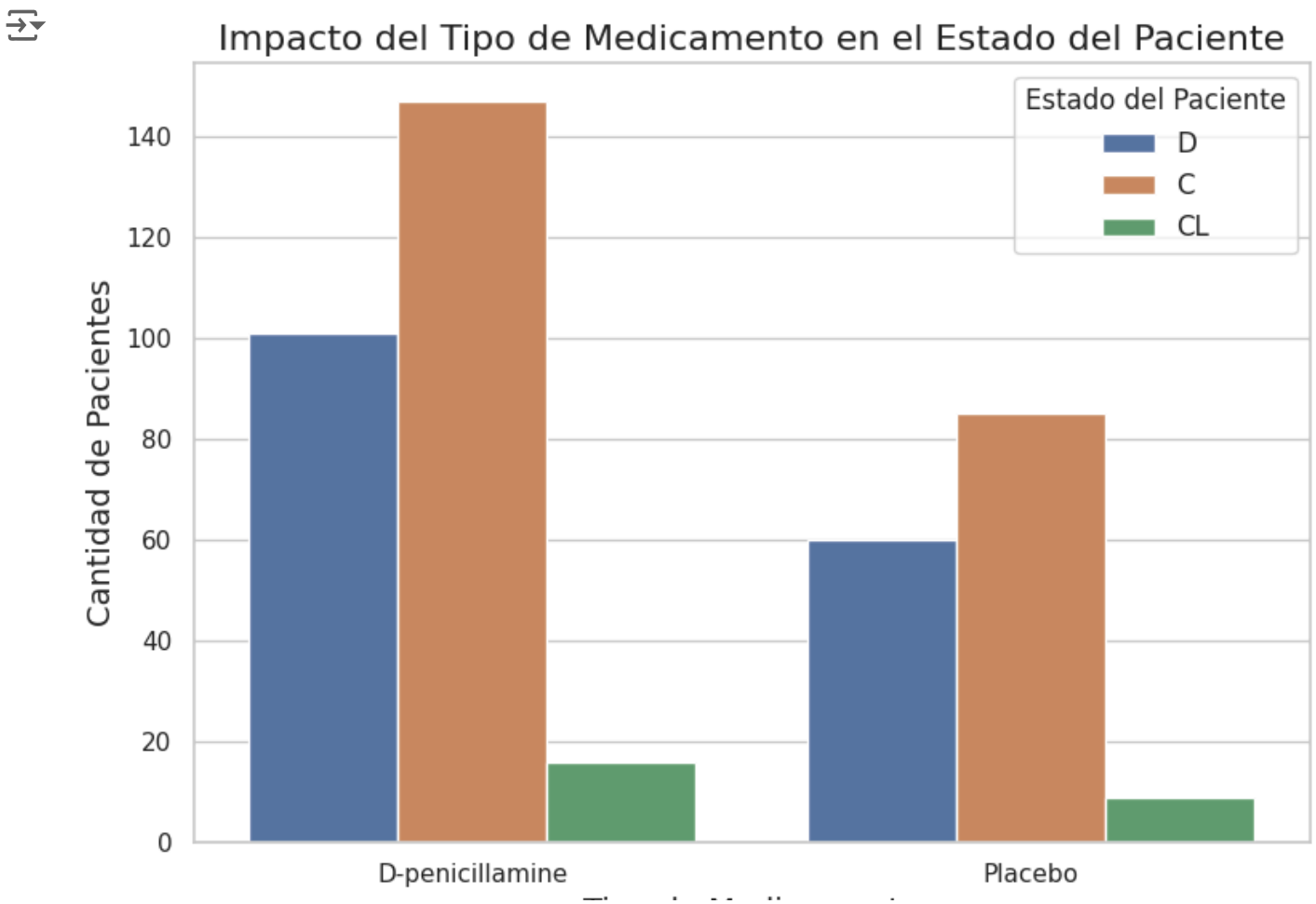
- `plt.figure(figsize=(10, 6))` : Crea una nueva figura de tamaño 10x6 pulgadas para el gráfico.
- `sns.regplot(x='N_Days', y='Status', data=df, logistic=True, y_jitter=0.03, scatter_kws={'alpha':0.5})` : Crea un gráfico de dispersión con una línea de regresión logística, donde el eje x representa el tiempo transcurrido desde el registro y el eje y representa el estado del paciente. El parámetro `y_jitter` agrega un ligero desplazamiento aleatorio a los puntos de dispersión en el eje y para evitar la superposición total de los puntos con la misma coordenada y. El parámetro `scatter_kws` se utiliza para personalizar la apariencia de los puntos de dispersión, en este caso, reduciendo su transparencia.
- `plt.title('Relación entre el tiempo transcurrido desde el registro y el estado del paciente', fontsize=16)` : Establece el título del gráfico con un tamaño de fuente de 16 puntos.
- `plt.xlabel('Tiempo transcurrido desde el registro (días)', fontsize=14)` : Establece la etiqueta del eje x con un tamaño de fuente de 14 puntos.
- `plt.ylabel('Estado del paciente', fontsize=14)` : Establece la etiqueta del eje y con un tamaño de fuente de 14 puntos.
- `plt.yticks([0, 1, 2], ['C (censored)', 'CL (censored due to liver tx)', 'D (death)'])` : Cambia las etiquetas de los ticks del eje y para representar los estados del paciente con su significado, utilizando las correspondientes conversiones del diccionario `status_mapping`.
- `warnings.filterwarnings("default", category=RuntimeWarning)` : Restaura el comportamiento predeterminado de las advertencias `RuntimeWarning` después de ejecutar el código problemático. Esto asegura que las advertencias vuelvan a aparecer normalmente en el entorno de ejecución.

2. **Impacto del tipo de medicamento administrado en el estado del paciente:**

La gráfica de barras es una herramienta visual comúnmente utilizada para representar la distribución de una variable categórica. Consiste en barras rectangulares que representan la frecuencia o proporción de observaciones en cada categoría. Es especialmente útil para comparar diferentes categorías entre sí.

En este contexto, la gráfica de barras se utilizó para visualizar el impacto del tipo de medicamento administrado en el estado del paciente. La variable categórica `Drug` representa los dos tipos de medicamentos administrados: D-penicilamina y placebo. La variable categórica `Status` indica el estado del paciente, que puede ser C (censurado), CL (censurado debido a trasplante de hígado) o D (fallecido).

```
import seaborn as sns
import matplotlib.pyplot as plt

# Mapear los valores numéricos a las etiquetas correspondientes
status_mapping = {0: 'C', 1: 'CL', 2: 'D'}
df['Status'] = df['Status'].map(status_mapping)

# Crear un gráfico de barras utilizando Seaborn
plt.figure(figsize=(8, 6))
sns.countplot(x='Drug', hue='Status', data=df)

# Personalizar el gráfico
plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)
plt.xlabel('Tipo de Medicamento', fontsize=14)
plt.ylabel('Cantidad de Pacientes', fontsize=14)
plt.legend(title='Estado del Paciente', fontsize=12)

# Mostrar el gráfico
plt.tight_layout()
```



- `import seaborn as sns`: Importa la biblioteca Seaborn, que se utiliza para crear visualizaciones estadísticas atractivas y informativas.

- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para personalizar y mostrar las visualizaciones creadas con Seaborn.
- `plt.figure(figsize=(8, 6))`: Crea una nueva figura de Matplotlib con un tamaño de 8x6 pulgadas.
- `sns.countplot(x='Drug', hue='Status', data=df)`: Crea un gráfico de barras utilizando Seaborn. La variable `Drug` se muestra en el eje X y se divide por color según la variable `Status`. Los datos se toman del DataFrame `df`.
- `plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)`: Establece el título del gráfico.
- `plt.xlabel('Tipo de Medicamento', fontsize=14)`: Etiqueta el eje X con "Tipo de Medicamento".
- `plt.ylabel('Cantidad de Pacientes', fontsize=14)`: Etiqueta el eje Y con "Cantidad de Pacientes".
- `plt.legend(title='Estado del Paciente', fontsize=12)`: Muestra una leyenda que indica los diferentes estados del paciente.
- `plt.tight_layout()`: Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.

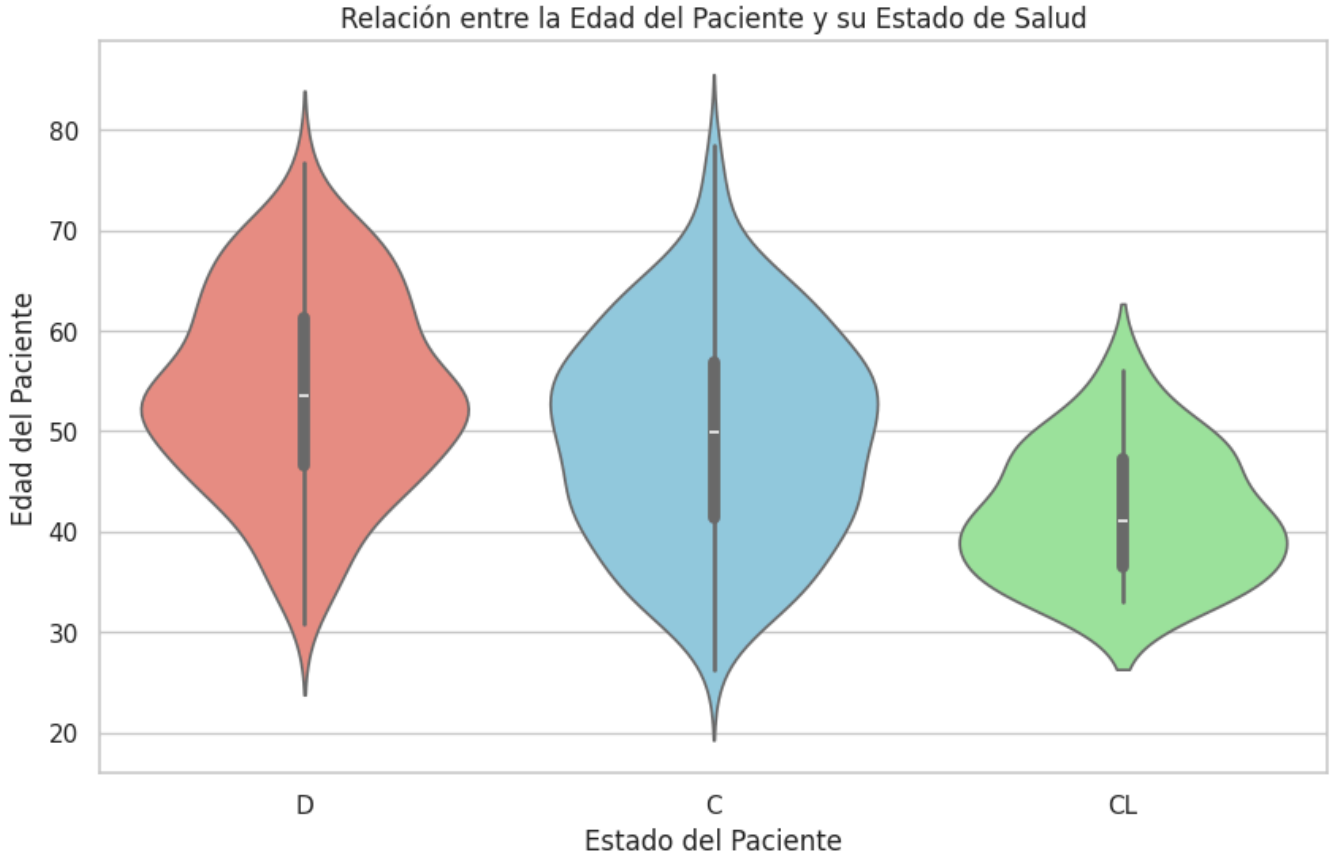
3. **Correlación entre la edad del paciente y su estado de salud:**

Las gráficas de violín son una representación visual de la distribución de datos y se utilizan comúnmente en análisis estadísticos y exploratorios. Cada violín muestra la distribución de los datos para una variable categórica, como el estado de salud en este caso, y permite comparar visualmente las distribuciones entre diferentes categorías.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Define los colores personalizados para cada estado de salud
colors = {'C': 'skyblue', 'CL': 'lightgreen', 'D': 'salmon'}

# Crea un gráfico de violín con colores personalizados
plt.figure(figsize=(10, 6))
sns.violinplot(x='Status', y='Age', data=df, palette=colors)

# Etiquetas y título
plt.xlabel('Estado del Paciente')
plt.ylabel('Edad del Paciente')
plt.title('Relación entre la Edad del Paciente y su Estado de Salud')
```

```
# Muestra el gráfico
plt.show()
```

```
<ipython-input-35-6da23b45d44d>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v

  sns.violinplot(x='Status', y='Age', data=df, palette=colors)
```



- `import seaborn as sns`: Importa la biblioteca Seaborn, que se utiliza para crear visualizaciones estadísticas atractivas e informativas.
- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para personalizar y mostrar las visualizaciones creadas con Seaborn.
- `plt.figure(figsize=(8, 6))`: Crea una nueva figura de Matplotlib con un tamaño de 8x6 pulgadas.
- `sns.countplot(x='Drug', hue='Status', data=df)`: Crea un gráfico de barras utilizando Seaborn. La variable `Drug` se muestra en el eje X y se divide por color según la variable `Status`. Los datos se toman del DataFrame `df`.
- `plt.title('Impacto del Tipo de Medicamento en el Estado del Paciente', fontsize=16)`: Establece el título del gráfico.
- `plt.xlabel('Tipo de Medicamento', fontsize=14)`: Etiqueta el eje X con "Tipo de Medicamento".

- `plt.ylabel('Cantidad de Pacientes', fontsize=14)` : Etiqueta el eje Y con "Cantidad de Pacientes".
- `plt.legend(title='Estado del Paciente', fontsize=12)` : Muestra una leyenda que indica los diferentes estados del paciente.
- `plt.tight_layout()` : Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.
- `plt.show()` : Muestra el gráfico.

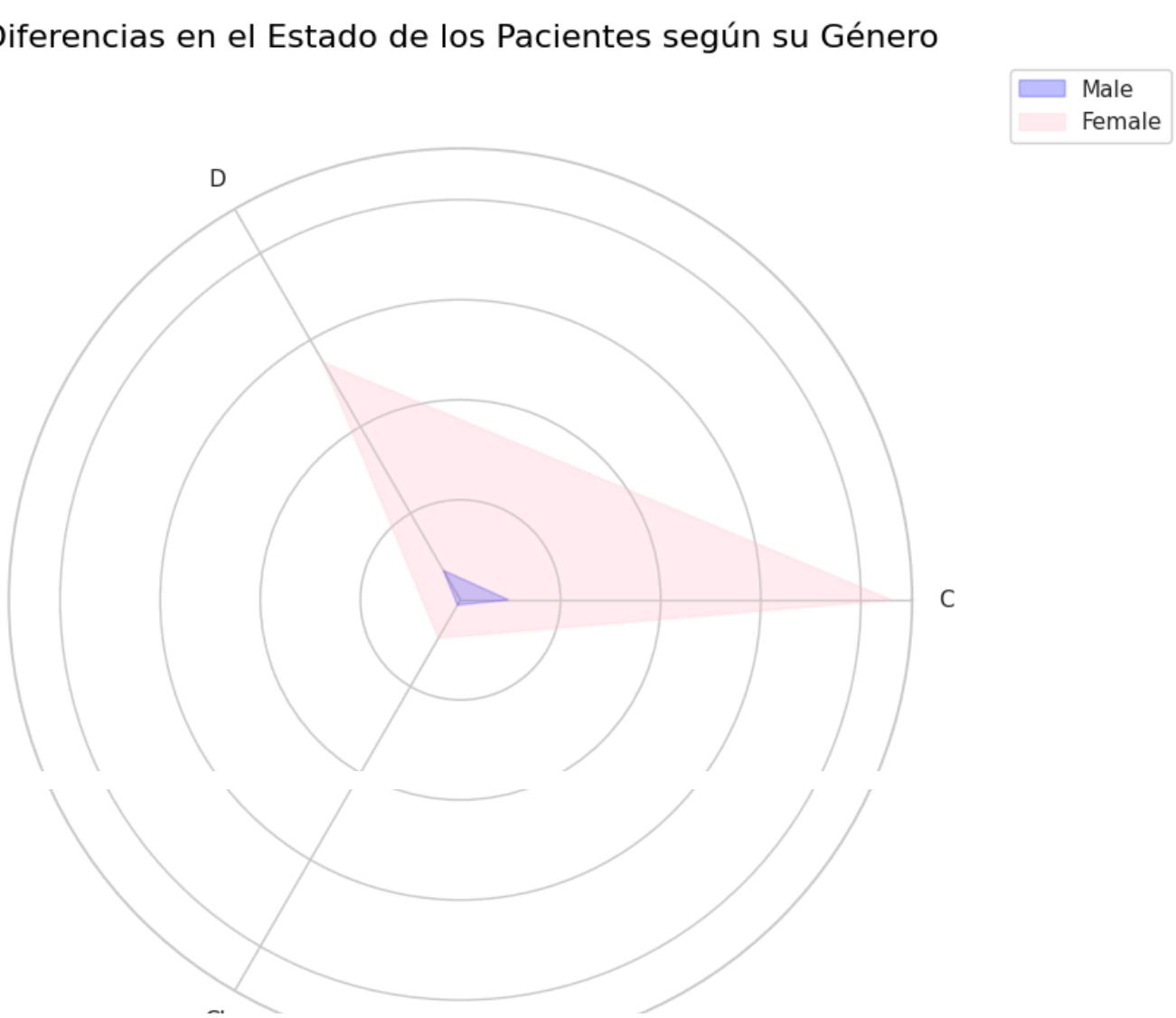
4. **Diferencias en el estado de los pacientes según su género**

El gráfico de radar es una herramienta visual que se utiliza para representar múltiples variables categóricas de forma simultánea en un solo gráfico circular. Cada variable se representa por un eje que parte del centro del círculo y se extiende hacia afuera, formando un ángulo igual. Las categorías de cada variable se colocan a lo largo de estos ejes, y el valor de cada categoría se representa por la distancia desde el centro del círculo.

En este ejemplo, el gráfico de radar se utiliza para comparar las diferencias en el estado de los pacientes según su género. Cada categoría en el gráfico representa un estado del paciente (C, CL, D), y los ejes radiales muestran la cantidad de pacientes en cada estado. Se utilizan dos áreas coloreadas para representar los datos de hombres y mujeres, lo que permite una fácil comparación entre ambos grupos.

```
import matplotlib.pyplot as plt
import numpy as np

# Variables para cada género
male_data = df[df['Sex'] == 'M']['Status'].value_counts().tolist()
female_data = df[df['Sex'] == 'F']['Status'].value_counts().tolist()

# Número de variables (estados de los pacientes)
labels = ['C', 'D', 'CL']

# Número de variables
num_vars = len(labels)

# Ángulos para cada variable
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# Asegurar que el gráfico sea cerrado
male_data += male_data[:1]
female_data += female_data[:1]
angles += angles[:1]

# Crear el gráfico
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))
```

```
ax.fill(angles, male_data, color='blue', alpha=0.25)
ax.fill(angles, female_data, color='pink', alpha=0.25)

# Establecer las etiquetas de las variables
ax.set_yticklabels([])

# Añadir las etiquetas de las variables
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

# Añadir título
plt.title('Diferencias en el Estado de los Pacientes según su Género', size=16, colo

# Añadir leyenda
plt.legend(['Male', 'Female'], loc='upper right', bbox_to_anchor=(1.3, 1.1))

# Mostrar el gráfico
plt.show()
```



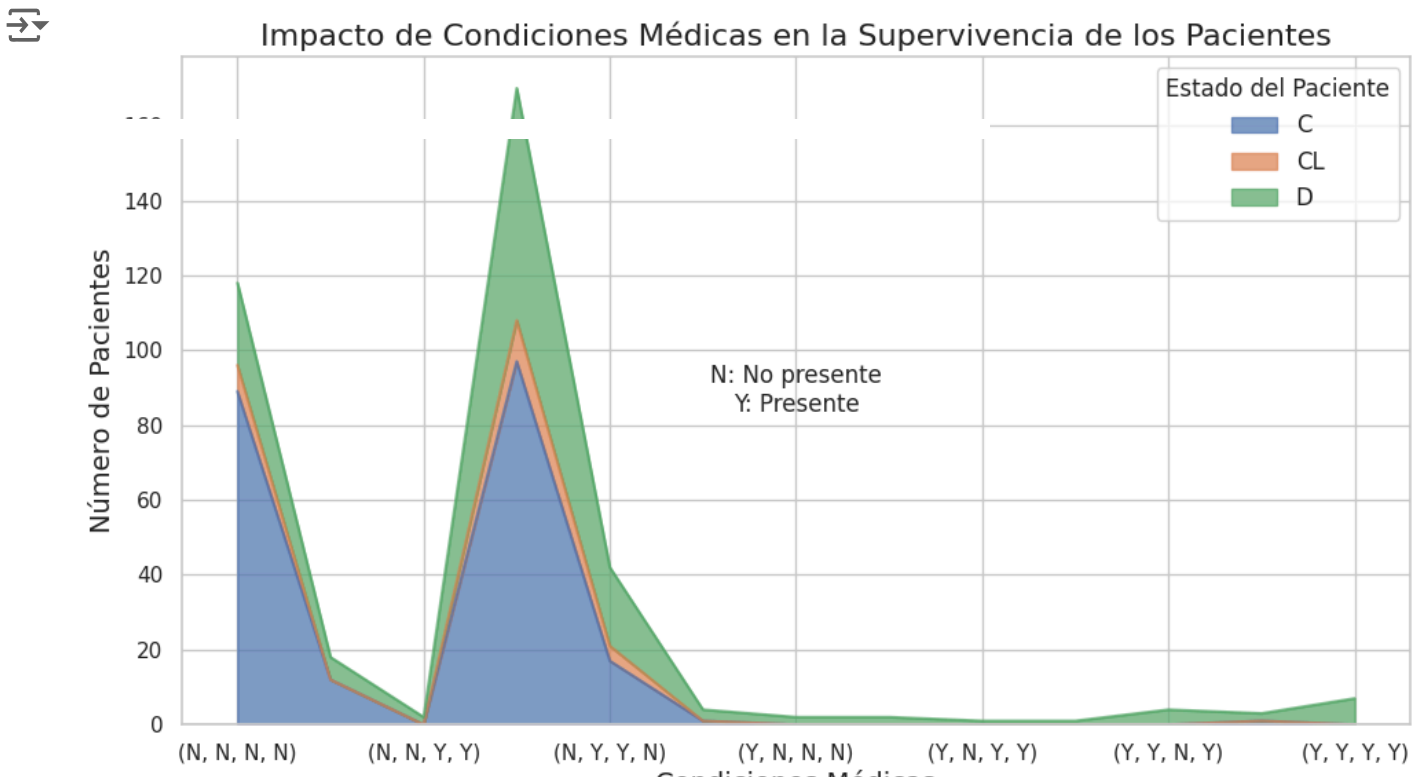
- `import matplotlib.pyplot as plt`: Importa la librería Matplotlib, que se utiliza para crear visualizaciones gráficas.
- `import numpy as np`: Importa la librería NumPy, que proporciona funciones matemáticas y numéricas para trabajar con arreglos.
- Se extraen los datos del DataFrame para hombres y mujeres, y se cuentan los estados de los pacientes.
- Se definen las etiquetas y los ángulos para cada variable (estado del paciente). Las etiquetas representan los estados del paciente (C, CL, D), y los ángulos se distribuyen uniformemente alrededor del círculo.
- Se asegura que el gráfico sea cerrado, agregando el primer valor al final de los datos y ángulos. Esto es necesario para cerrar el polígono que representa el gráfico.
- Se crea el gráfico polar utilizando `plt.subplots()` con el parámetro `subplot_kw=dict(polar=True)`, lo que indica que se trata de un gráfico polar.
- Se rellenan las áreas correspondientes a los datos de hombres y mujeres utilizando `ax.fill()`. Cada área está coloreada de manera diferente (azul para hombres y rosa para mujeres) y se define con respecto a los ángulos y los datos de estado de los pacientes.
- Se establecen las etiquetas de las variables en el eje X utilizando `ax.set_xticks()` y `ax.set_xticklabels()`. Las etiquetas se colocan a lo largo de los ángulos definidos anteriormente.
- Se añade un título al gráfico utilizando `plt.title()`.
- Se añade una leyenda que indica qué color representa a cada género utilizando `plt.legend()`.
- Finalmente, se muestra el gráfico utilizando `plt.show()`.

5. **Impacto de ciertas condiciones médicas en la supervivencia de los pacientes:**

El gráfico de áreas apiladas es una visualización que muestra la distribución de una variable a lo largo del tiempo o de diferentes categorías. En este tipo de gráfico, las áreas se superponen unas sobre otras, y la altura de cada área en un punto dado representa la cantidad total en ese punto. Se utiliza para comparar la contribución relativa de diferentes partes a un todo y para observar cómo cambian estas contribuciones con el tiempo o entre diferentes categorías.

```
import matplotlib.pyplot as plt

# Filtrar el DataFrame para incluir solo las condiciones médicas específicas
```

```
filtered_df = df[(df['Ascites'].isin(['N', 'Y'])) &
                 (df['Hepatomegaly'].isin(['N', 'Y'])) &
                 (df['Spiders'].isin(['N', 'Y'])) &
                 (df['Edema'].isin(['N', 'Y']))]

# Crear un DataFrame que contenga la frecuencia de cada combinación de condiciones m
medical_conditions = ['Ascites', 'Hepatomegaly', 'Spiders', 'Edema']
condition_counts = filtered_df.groupby(medical_conditions + ['Status']).size().unsta

# Crear un gráfico de áreas apiladas
condition_counts.plot(kind='area', stacked=True, figsize=(10, 6), alpha=0.7)

# Añadir una nota sobre el significado de 'N' y 'Y' en la figura
plt.text(0.5, 0.5, 'N: No presente\nY: Presente', fontsize=12, ha='center', va='cent

# Añadir etiquetas y título
plt.xlabel('Condiciones Médicas', fontsize=14)
plt.ylabel('Número de Pacientes', fontsize=14)
plt.title('Impacto de Condiciones Médicas en la Supervivencia de los Pacientes', fon

# Mostrar el gráfico
plt.legend(title='Estado del Paciente', fontsize=12)
plt.tight_layout()
plt.show()
```



- `import matplotlib.pyplot as plt`: Importa la biblioteca Matplotlib, que se utiliza para crear visualizaciones en Python.

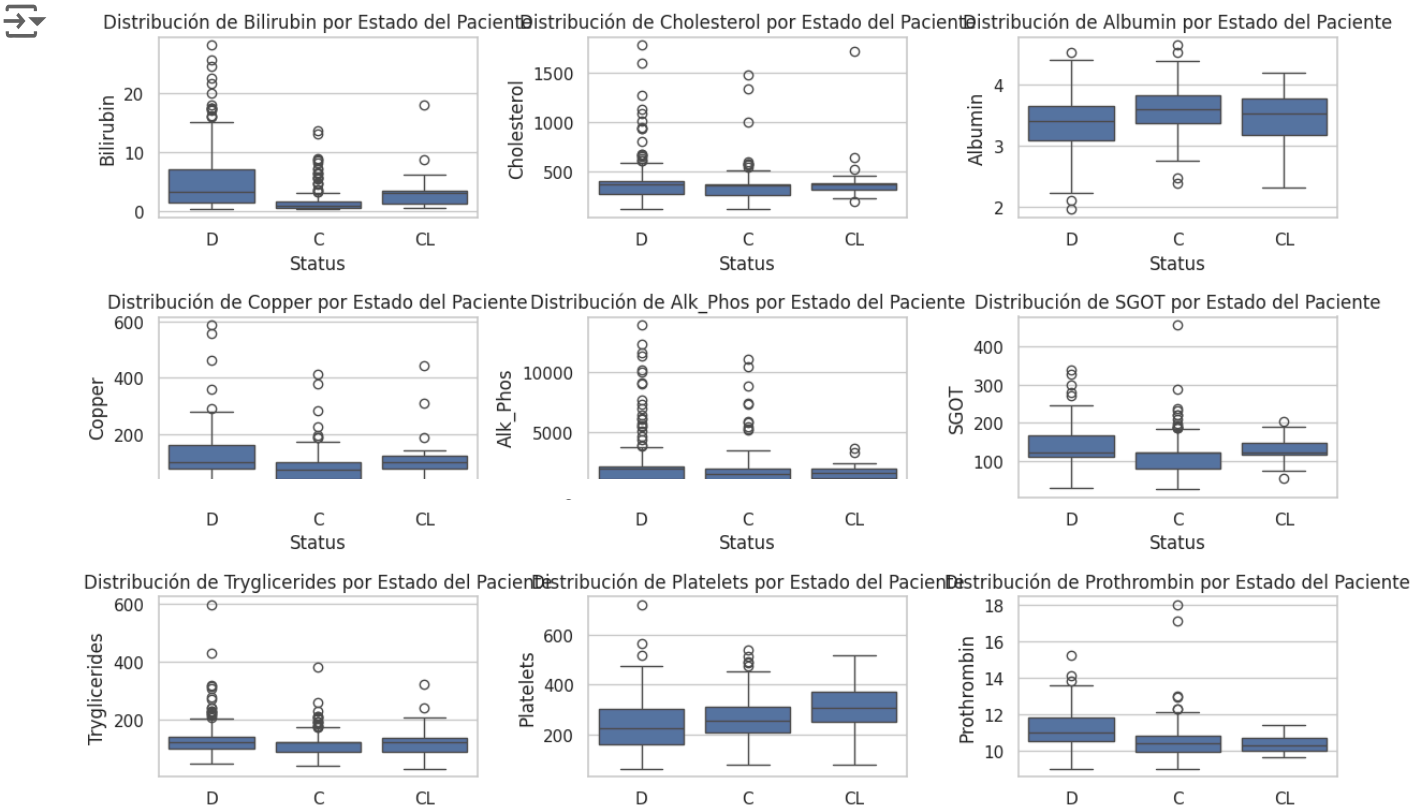
- Filtrado del DataFrame: Se filtra el DataFrame original para incluir solo las filas que contienen información sobre las condiciones médicas específicas ('Ascites', 'Hepatomegaly', 'Spiders', 'Edema').
- Creación del DataFrame de frecuencia: Se agrupa el DataFrame filtrado por las condiciones médicas y el estado del paciente y se calcula la frecuencia de cada combinación. Luego, se crea un nuevo DataFrame donde las filas representan las combinaciones de condiciones médicas y las columnas representan los estados del paciente.
- Creación del gráfico de áreas apiladas: Se utiliza el método `plot` con `kind='area'` para crear un gráfico de áreas apiladas. El parámetro `stacked=True` indica que las áreas deben apilarse una encima de la otra. El parámetro `alpha` establece la transparencia de las áreas para mejorar la legibilidad.
- Añadir una nota: Se utiliza `plt.text` para añadir una nota explicativa sobre el significado de 'N' y 'Y' en la figura. La posición de la nota se establece en el centro de la figura con coordenadas relativas.
- Añadir etiquetas y título: Se añaden etiquetas a los ejes x e y y se establece un título para el gráfico.
- Mostrar el gráfico: Se utiliza `plt.legend` para mostrar la leyenda que indica los diferentes estados del paciente y `plt.show()` para mostrar el gráfico completo.

6. **Relación entre los niveles de biomarcadores y el estado de los pacientes:**

El gráfico de caja y bigotes, también conocido como diagrama de caja, es una herramienta visual que nos permite representar la distribución de un conjunto de datos de una manera descriptiva y concisa. Este tipo de gráfico proporciona información sobre la mediana, los cuartiles, los valores atípicos y la dispersión de los datos. En el contexto de este análisis, se utiliza para visualizar la distribución de los niveles de biomarcadores para diferentes estados de los pacientes.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Lista de biomarcadores a considerar
biomarkers = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', '

# Crear un gráfico de caja y bigotes para cada biomarcador
plt.figure(figsize=(12, 8))
for i, biomarker in enumerate(biomarkers, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x='Status', y=biomarker, data=df)
    plt.title(f'Distribución de {biomarker} por Estado del Paciente')
```

```
# Ajustar espacios entre subgráficos
plt.tight_layout()
plt.show()
```



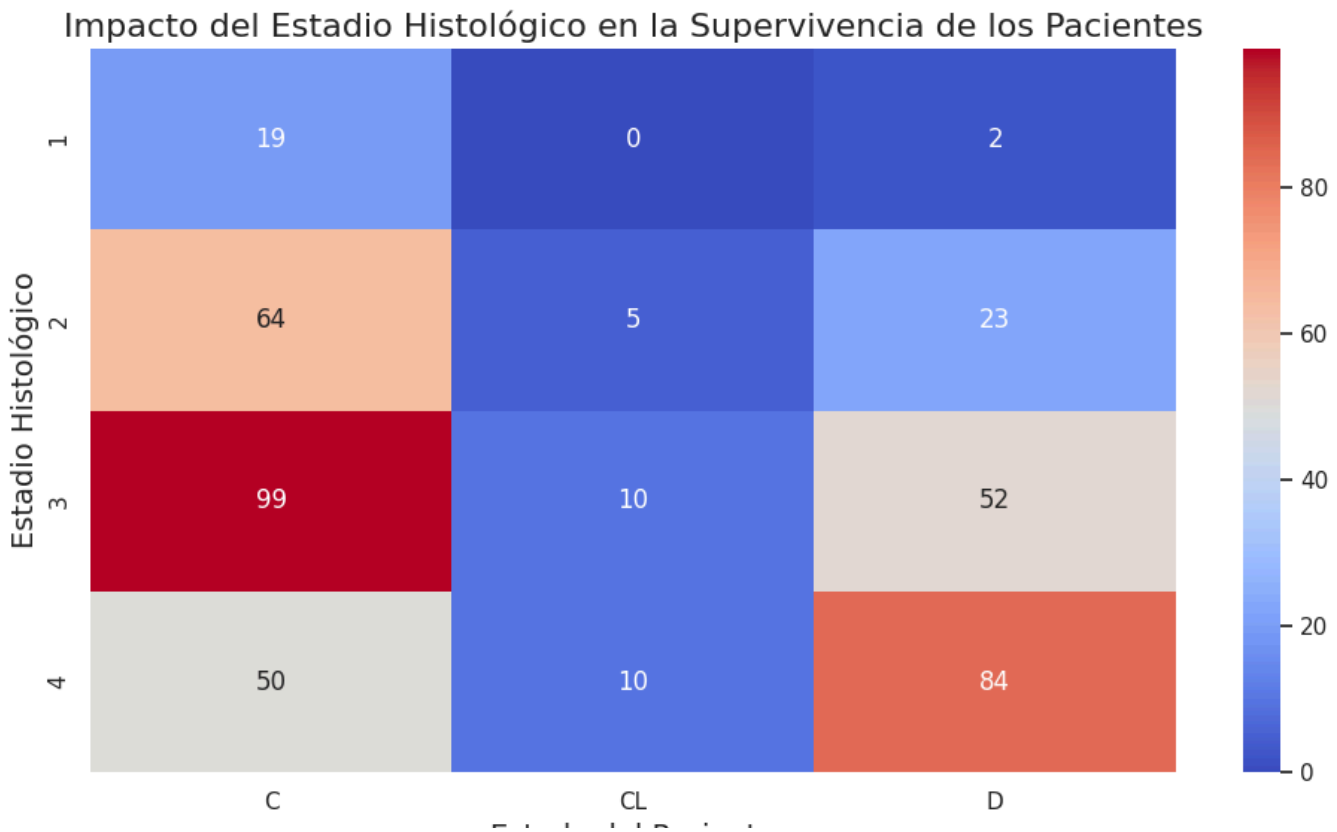
- `import seaborn as sns` : Importa la biblioteca Seaborn, que nos permite crear gráficos estadísticos atractivos y informativos.
- `import matplotlib.pyplot as plt` : Importa la biblioteca Matplotlib, que utilizamos para personalizar y mostrar nuestras visualizaciones.
- `biomarkers = ['Bilirubin', 'Cholesterol', 'Albumin', 'Copper', 'Alk_Phos', 'SGOT', 'Tryglicerides', 'Platelets', 'Prothrombin']` : Crea una lista de los biomarcadores que se analizarán en este gráfico.
- `plt.figure(figsize=(12, 8))` : Crea una nueva figura de Matplotlib con un tamaño de 12x8 pulgadas.
- `for i, biomarker in enumerate(biomarkers, 1):` : Itera sobre cada biomarcador en la lista `biomarkers` .
- `plt.subplot(3, 3, i)` : Crea un subgráfico en la posición i dentro de una cuadrícula de 3x3.
- `sns.boxplot(x='Status', y=biomarker, data=df)` : Crea un gráfico de caja y bigotes utilizando Seaborn, donde el eje X representa el estado del paciente y el eje Y representa los niveles del biomarcador actual.
- `plt.title(f'Distribución de {biomarker} por Estado del Paciente')` : Establece el título del subgráfico con el nombre del biomarcador actual.

- `plt.tight_layout()` : Ajusta automáticamente los márgenes del gráfico para que quepan todos los elementos.
- `plt.show()` : Muestra el gráfico generado.

7. **Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes:**

El tipo de gráfica que utilizamos es un mapa de calor (heatmap), el cual es una representación visual de los datos donde los valores de una matriz se representan como colores. En este contexto, utilizamos el mapa de calor para mostrar la relación entre el estadio histológico de la enfermedad y el estado de los pacientes.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Convertir la columna 'Stage' a tipo entero
df['Stage'] = df['Stage'].astype(int)

# Filtrar el DataFrame para incluir solo las columnas relevantes
relevant_columns = ['Stage', 'Status']
filtered_df = df[relevant_columns]

# Calcular la matriz de contingencia entre el estadio histológico y el estado del pa
contingency_matrix = pd.crosstab(filtered_df['Stage'], filtered_df['Status'])

# Crear un mapa de calor con Seaborn
plt.figure(figsize=(10, 6))  # Establecer el tamaño de la figura
sns.heatmap(contingency_matrix, annot=True, cmap='coolwarm', fmt='d')  # fmt='d' par

# Agregar etiquetas y título
plt.xlabel('Estado del Paciente', fontsize=14)
plt.ylabel('Estadio Histológico', fontsize=14)
plt.title('Impacto del Estadio Histológico en la Supervivencia de los Pacientes', fo

# Mostrar el mapa de calor
plt.tight_layout()
plt.show()
```



- **Importar bibliotecas**: Se importan las bibliotecas necesarias, `seaborn` para crear gráficos estadísticos y `matplotlib.pyplot` para personalizar y mostrar las visualizaciones.
- **Filtrar el DataFrame**: Se seleccionan las columnas relevantes del DataFrame original, que son 'Stage' (estadio histológico) y 'Status' (estado del paciente), y se almacenan en `filtered_df`.
- **Calcular la matriz de contingencia**: Se utiliza la función `pd.crosstab()` de Pandas para calcular la matriz de contingencia entre el estadio histológico y el estado del paciente, donde las filas representan los diferentes estadios histológicos y las columnas representan los diferentes estados de los pacientes. La matriz de contingencia se almacena en `contingency_matrix`.
- **Crear un mapa de calor**: Se crea un mapa de calor utilizando la función `sns.heatmap()` de Seaborn. Se pasa la matriz de contingencia como datos y se establece `annot=True` para mostrar los valores de las celdas en el mapa de calor. El argumento `cmap='coolwarm'` se utiliza para elegir un mapa de colores divergente que va desde azul (valores bajos) a rojo (valores altos). El argumento `fmt='d'` se utiliza para formatear los valores como enteros.
- **Personalizar etiquetas y título**: Se añaden etiquetas a los ejes X e Y y un título al gráfico para hacerlo más comprensible y legible.
- **Mostrar el mapa de calor**: Se llama a `plt.show()` para mostrar el mapa de calor en la salida.

Actuar

La toma de decisiones basadas en datos es fundamental para mejorar la eficacia y la eficiencia en el tratamiento y la gestión de la cirrosis. En esta sección, se proponen acciones específicas que se derivan del análisis de datos realizado previamente. Estas acciones están respaldadas por evidencia estadística y se basan en los resultados obtenidos de la exploración y el análisis de diversos conjuntos de datos relacionados con la cirrosis.

1. Personalizar el tratamiento

Datos utilizados:

- Estadísticas descriptivas de N_Days agrupadas por el estado del paciente.
- Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes.

Propuesta de acción: Dado que se han observado diferencias significativas en la supervivencia de los pacientes según el estadio histológico de la enfermedad, se recomienda personalizar el tratamiento de la cirrosis de acuerdo con el estadio de la enfermedad. Los pacientes en etapas avanzadas pueden requerir un enfoque más agresivo y una atención médica más intensiva, que puede incluir opciones terapéuticas como trasplante hepático, terapia farmacológica específica o intervenciones quirúrgicas. Por otro lado, los pacientes en etapas iniciales pueden beneficiarse de medidas de manejo conservador y estrategias de estilo de vida, como cambios en la dieta y el ejercicio, junto con el monitoreo regular de la progresión de la enfermedad. La personalización del tratamiento garantiza que cada paciente reciba la atención más adecuada y efectiva de acuerdo con su estado de salud y necesidades individuales.

2. Monitorización continua

Datos utilizados:

- Resultados del análisis de los niveles de biomarcadores en relación con el estado de los pacientes.

Propuesta de acción: Implementar un sistema de monitorización continua para seguir de cerca la progresión de la enfermedad en los pacientes. Se pueden utilizar pruebas regulares de biomarcadores, como bilirrubina, colesterol, albúmina, entre otros, para detectar cambios en el estado de salud y ajustar el tratamiento en consecuencia. Esto permitirá una intervención temprana y una gestión más efectiva de la cirrosis, optimizando así los resultados clínicos.

3. Promoción de la detección temprana

Datos utilizados:

- Impacto de ciertas condiciones médicas (Ascites, Hepatomegaly, Spiders, Edema) en la supervivencia de los pacientes.

Propuesta de acción: Promover la detección temprana de la cirrosis y sus complicaciones mediante campañas de concienciación y programas de detección regulares. Se debe prestar especial atención a identificar a los pacientes en etapas tempranas de la enfermedad, ya que el tratamiento oportuno puede mejorar significativamente los resultados y reducir la carga de la enfermedad.

4. Educación del paciente

Datos utilizados:

- Diferencias en el estado de los pacientes según su género.

Propuesta de acción: Brindar educación integral a los pacientes sobre los factores de riesgo de la cirrosis y la importancia de adherirse al tratamiento y llevar un estilo de vida saludable. Esto incluiría información sobre la dieta adecuada, la reducción del consumo de alcohol y la importancia de seguir las indicaciones médicas para controlar los síntomas y prevenir complicaciones. Una mayor conciencia y comprensión por parte de los pacientes pueden mejorar la gestión de la enfermedad y mejorar los resultados a largo plazo.

5. Mejora de los servicios de atención médica

Datos utilizados:

- Estadísticas descriptivas de N_Days agrupadas por el estado del paciente.

Propuesta de acción: Mejorar el acceso a los servicios de atención médica, especialmente en áreas donde la atención es limitada. Esto podría incluir la expansión de los servicios de atención primaria, la capacitación de profesionales de la salud en el manejo de la cirrosis y la mejora de la infraestructura de salud en general. Una atención médica más accesible y eficiente garantizará que los pacientes reciban el tratamiento y la atención adecuados en todas las etapas de la enfermedad.

6. Investigación continua

Datos utilizados:

- Impacto del estadio histológico de la enfermedad en la supervivencia de los pacientes.
- Resultados del ANOVA para cada biomarcador en relación con el estado de los pacientes.

Propuesta de acción: Continuar la investigación en el campo de la cirrosis para desarrollar nuevas terapias y estrategias de tratamiento. Esto incluiría estudios clínicos para evaluar la eficacia de nuevos medicamentos y tratamientos, así como investigaciones epidemiológicas para comprender mejor los factores de riesgo y las causas subyacentes de la cirrosis. La investigación continua es fundamental para mejorar los resultados a largo plazo y avanzar en la gestión de esta enfermedad compleja.

**Temas de consulta **

Aquí tienes algunos temas que podrías explorar en la búsqueda de artículos científicos para continuar la investigación en el campo de la cirrosis:

- **Terapias emergentes**: Investiga sobre nuevas terapias farmacológicas y tratamientos innovadores en desarrollo para el tratamiento de la cirrosis, como terapias génicas, terapias celulares o enfoques basados en la modulación del microbioma intestinal.
- **Estrategias de manejo de la enfermedad**: Examina enfoques integrales para el manejo de la cirrosis, incluyendo estrategias de tratamiento combinado, manejo de síntomas, y abordajes multidisciplinarios que involucren a diferentes especialidades médicas y profesionales de la salud.
- **Factores de riesgo y prevención**: Investiga los factores de riesgo ambientales, genéticos y comportamentales asociados con el desarrollo de la cirrosis, así como estrategias efectivas de prevención primaria y secundaria para reducir la incidencia y la progresión de la enfermedad.
- **Evaluación de biomarcadores**: Explora la investigación en curso sobre biomarcadores serológicos, genéticos, proteómicos y metabólicos para el diagnóstico precoz, la estratificación del riesgo y el seguimiento de la progresión de la cirrosis, así como su utilidad en la predicción de resultados clínicos y la respuesta al tratamiento.
- **Abordajes quirúrgicos y trasplante hepático**: Examina los avances en técnicas quirúrgicas, criterios de selección de pacientes y resultados a largo plazo del trasplante hepático en pacientes con cirrosis, así como estrategias para optimizar la disponibilidad de órganos y mejorar los resultados del trasplante.
- **Aspectos psicosociales y calidad de vida**: Investiga el impacto psicosocial de la cirrosis en la calidad de vida de los pacientes y sus familias, así como intervenciones psicosociales y de apoyo que puedan mejorar el bienestar emocional y la adherencia al tratamiento.
- **Epidemiología y carga global de la enfermedad**: Analiza estudios epidemiológicos sobre la prevalencia, incidencia y carga global de la cirrosis a nivel mundial, así como las tendencias