Seaborn

Luisa Gomez
luisa.gomez@pucp.edu.pe
https://github.com/4591526

PUCP

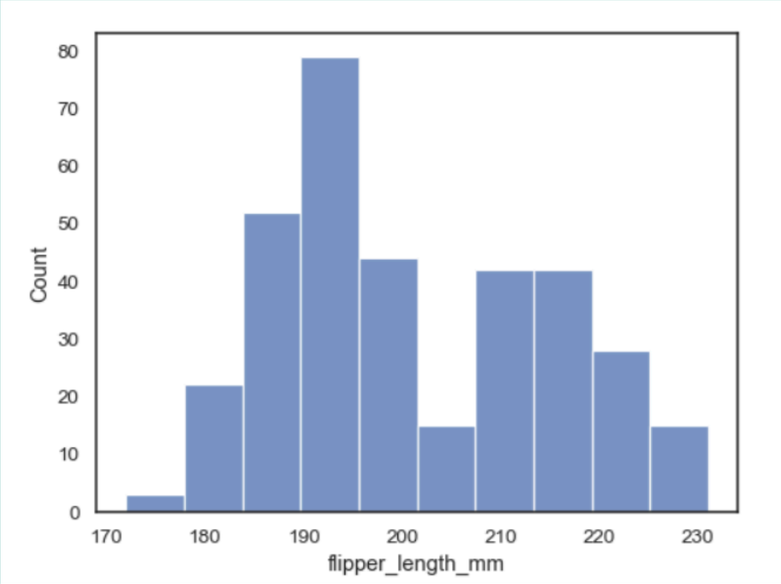
¿Qué es Seaborn?

- Seaborn es una librería de visualización de datos en Python basada en matplotlib.
- Ofrece una interfaz de alto nivel para crear gráficos estadísticos atractivos e informativos.

¿Cómo se importa la librería Matplotlib?

- **import seaborn**: Trae al entorno de trabajo todas las funciones y herramientas de la librería Seaborn
- **as sns**: Le da un alias más corto (en este caso, sns) para que puedas usarlo de manera más rápida al escribir códigos

Histograma
sns.histplot(X)

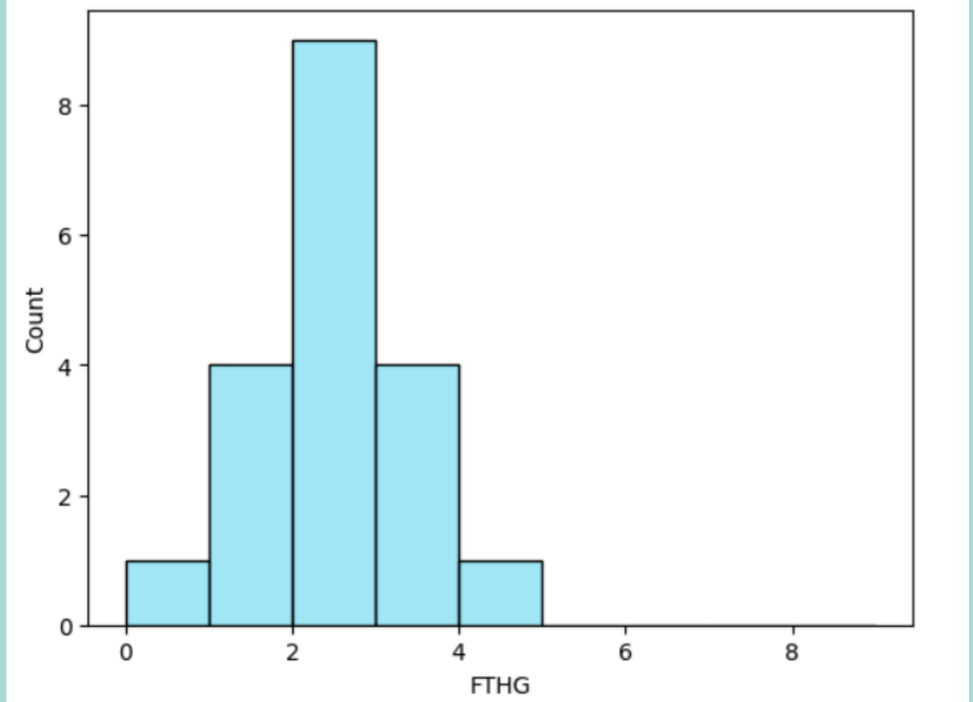


- Un histograma es una herramienta clásica de visualización que representa la distribución de una o más variables contando el número de observaciones que caen dentro de intervalos discretos (bins).
- Esta función puede normalizar la estadística calculada en cada intervalo para estimar la frecuencia, la densidad o la masa de probabilidad, y también puede añadir una curva suavizada de estimación de densidad por kernel.

Estructura de un histograma

```
sns.histplot(goles_anotados_local, bins=range(0, 10), kde=False,
             color='#80e2f6', edgecolor='black', alpha=0.75)
```

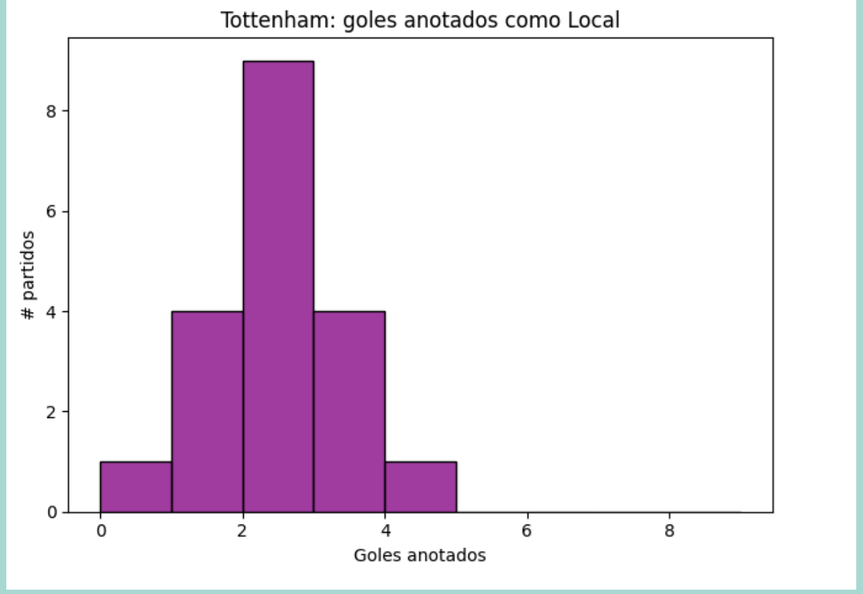
```
<Axes: xlabel='FTHG', ylabel='Count'>
```



- **sns.histplot(...)** → Es la función de Seaborn para crear histogramas, que muestran la frecuencia de los datos agrupados en intervalos
- goles_anotados_local → Es la Serie de pandas (o lista de datos) que contiene los goles anotados por el equipo local en diferentes partidos
- bins=range(0, 10) → Define los intervalos (bins) del histograma
  Por ejemplo:
  - Bin 0: goles entre 0 y 1
  - Bin 1: goles entre 1 y 2 y así sucesivamente hasta 9.
- **kde=False** → Indica que no se debe agregar la curva de densidad suavizada (Kernel Density Estimate), que normalmente se usa para ver la tendencia general
- color='#80e2f6' → Define el color de relleno de las barras
- edgecolor='black' → Colorea el borde de las barras en negro, lo que ayuda a diferenciarlas visualmente.
- alpha=0.75 → Ajusta la transparencia de las barras. 1.0 es completamente opaco, 0.0 es invisible

Cambio de nombre de las etiquetas con matplotlib

```python
import matplotlib.pyplot as plt

sns.histplot(goles_anotados_local, bins=range(0, 10), kde=False,
             color='purple', edgecolor='black', alpha=0.75)

plt.title('Tottenham: goles anotados como Local')
plt.xlabel('Goles anotados')
plt.ylabel('# partidos')

plt.tight_layout()
plt.show()
```



- **plt.title(**'Tottenham: goles anotados como Local'**)**
- **plt.xlabel(**'Goles anotados'**)**
- **plt.ylabel(**'# partidos'**)**

Estas funciones añaden información al gráfico:

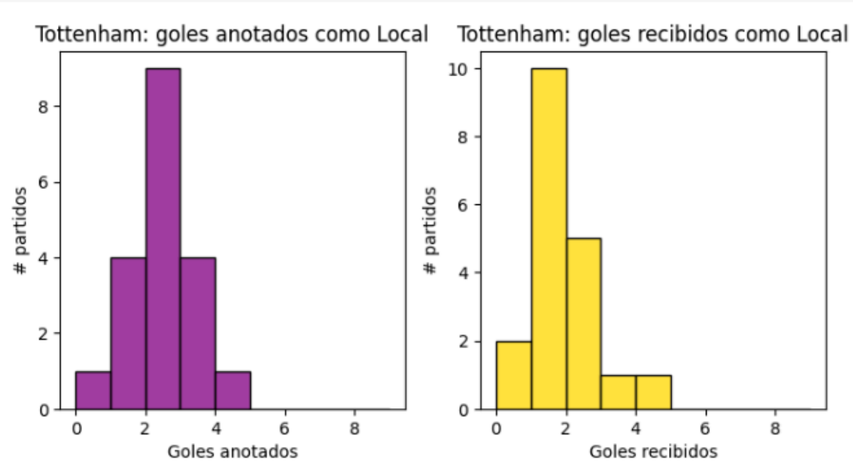
- title: título del gráfico
- xlabel: etiqueta del eje X
- ylabel: etiqueta del eje Y

- **plt.tight_layout()**

Ajusta automáticamente los márgenes y espacios entre los subgráficos para que no se sobrepongan las etiquetas ni títulos.

Cambio de tamaño y dividir en subgráficos con matplotlib

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 7))

plt.subplot(2, 2, 1)
sns.histplot(goles_anotados_local, bins=range(0, 10), kde=False,
             color='purple', edgecolor='black', alpha=0.75)
plt.title('Tottenham: goles anotados como Local')
plt.xlabel('Goles anotados')
plt.ylabel('# partidos')

plt.subplot(2, 2, 2)
sns.histplot(goles_recibidos_local, bins=range(0, 10), kde=False,
             color='gold', edgecolor='black', alpha=0.75)
plt.title('Tottenham: goles recibidos como Local')
plt.xlabel('Goles recibidos')
plt.ylabel('# partidos')

plt.tight_layout()
plt.show()
```

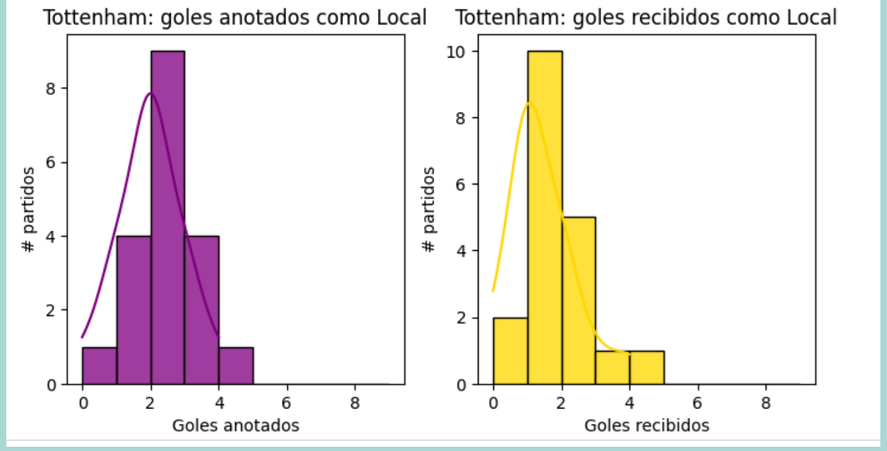


- **plt.figure(figsize=(7, 7))** → Crea una nueva figura (un lienzo en blanco)
- **figsize=(7, 7)** define el tamaño de la figura en pulgadas: 7 de ancho × 7 de alto

- Primer gráfico → **plt.subplot(2, 2, 1)**
  - Prepara el primer espacio en una cuadrícula de 2 filas × 2 columnas
  - Este será el primer gráfico, ubicado en la esquina superior izquierda
- Segundo gráfico → **plt.subplot(2, 2, 2)**
  - Prepara el segundo espacio en la cuadrícula (posición 2)
  - Este será el gráfico de arriba a la derecha

¿Cómo añadir una curva de densidad?

```
plt.figure(figsize=(7, 7))

plt.subplot(2, 2, 1)
sns.histplot(goles_anotados_local, bins=range(0, 10), kde=True,
             color='purple', edgecolor='black', alpha=0.75)
plt.title('Tottenham: goles anotados como Local')
plt.xlabel('Goles anotados')
plt.ylabel('# partidos')

plt.subplot(2, 2, 2)
sns.histplot(goles_recibidos_local, bins=range(0, 10), kde=True,
             color='gold', edgecolor='black', alpha=0.75)
plt.title('Tottenham: goles recibidos como Local')
plt.xlabel('Goles recibidos')
plt.ylabel('# partidos')

plt.tight_layout()
plt.show()
```



- **sns.histplot(datos, kde=True)**
  Esto dibuja dos cosas:
  - El histograma: barras que cuentan cuántos valores hay en cada intervalo
  - La curva KDE: una línea suavizada que muestra la tendencia general de los datos

¿Cuándo usar kde=True?

- Cuando quieres ver la forma general de la distribución.
- Cuando los datos son continuos (por ejemplo, notas, edades, temperaturas).
- No es muy útil si los datos son categóricos o enteros con pocos valores posibles, ya que la curva puede ser poco informativa o incluso confusa

¿Cómo guardar mis gráficos?

- **plt.savefig(...)** → Guarda la figura actual en un archivo de imagen
- 'goles_Tottenham.png' → Nombre del archivo de salida.
- El formato lo determina la extensión:
  - .png → imagen tipo PNG (sin pérdida, fondo transparente si se quiere)
  - .jpg o .jpeg → formato con compresión
- **dpi=300** → Significa dots per inch (puntos por pulgada)
  - Define la resolución de la imagen
  - 300 dpi es ideal para impresión de alta calidad
  - Para uso en web o presentaciones, 100–150 dpi suele ser suficiente

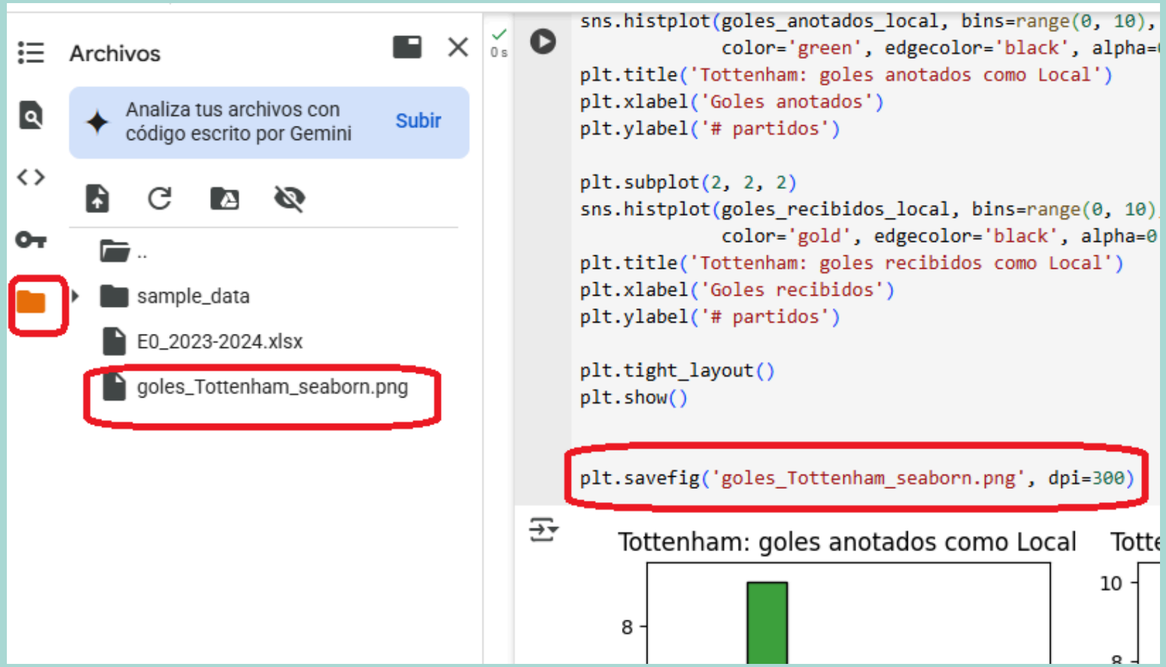
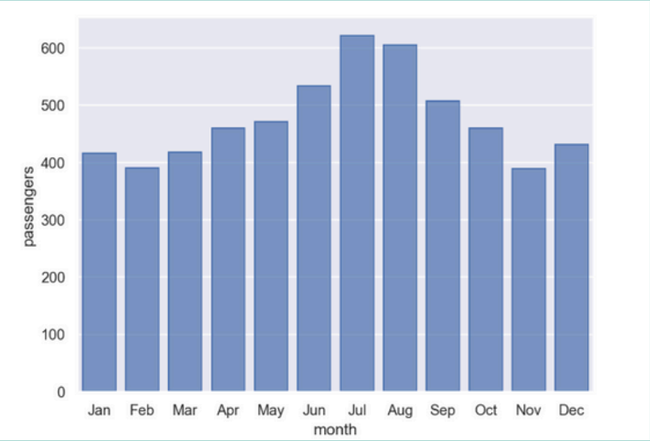
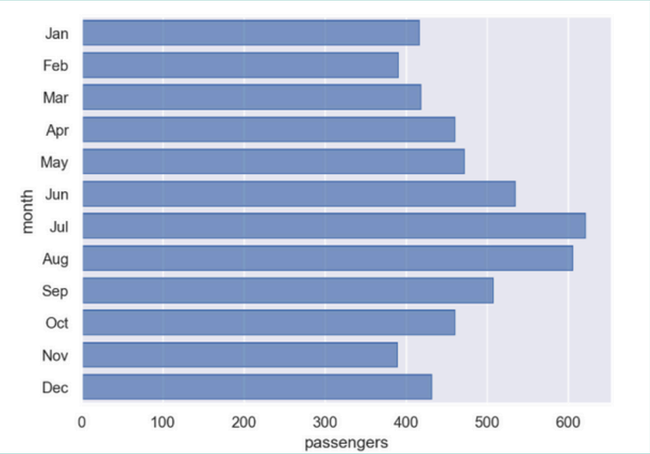


Gráfico de barras



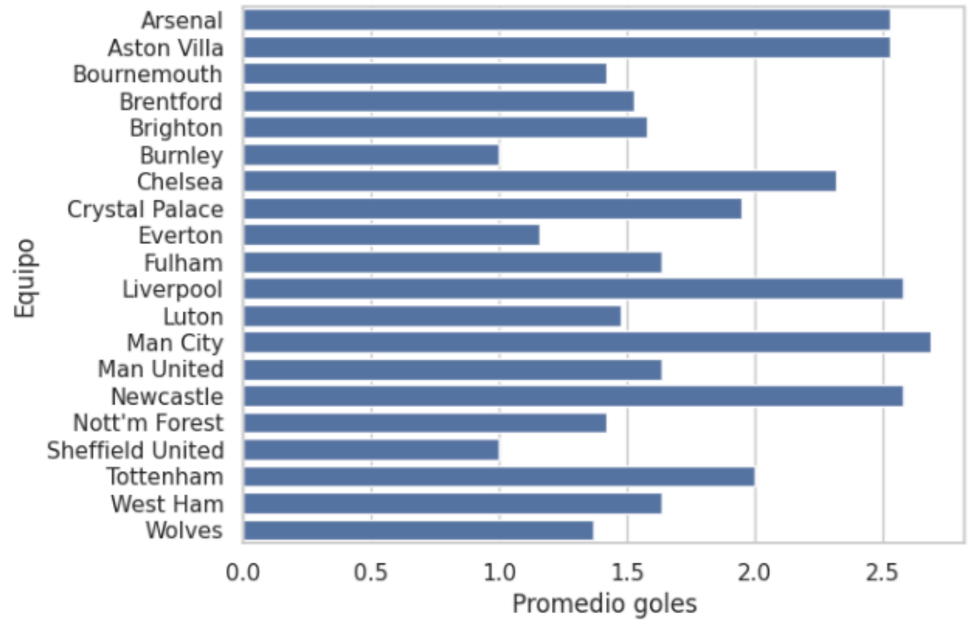


- Un bar plot (gráfico de barras) representa valores categóricos con barras cuya altura (o longitud) indica la cantidad, frecuencia o valor asociado a cada categoría.
- Se usa cuando los datos están agrupados por categorías (por ejemplo: equipos, géneros, regiones, etc.).

Estructura del gráfico de barras horizontales

```
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles'
)
```

```
<Axes: xlabel='Promedio goles', ylabel='Equipo'>
```



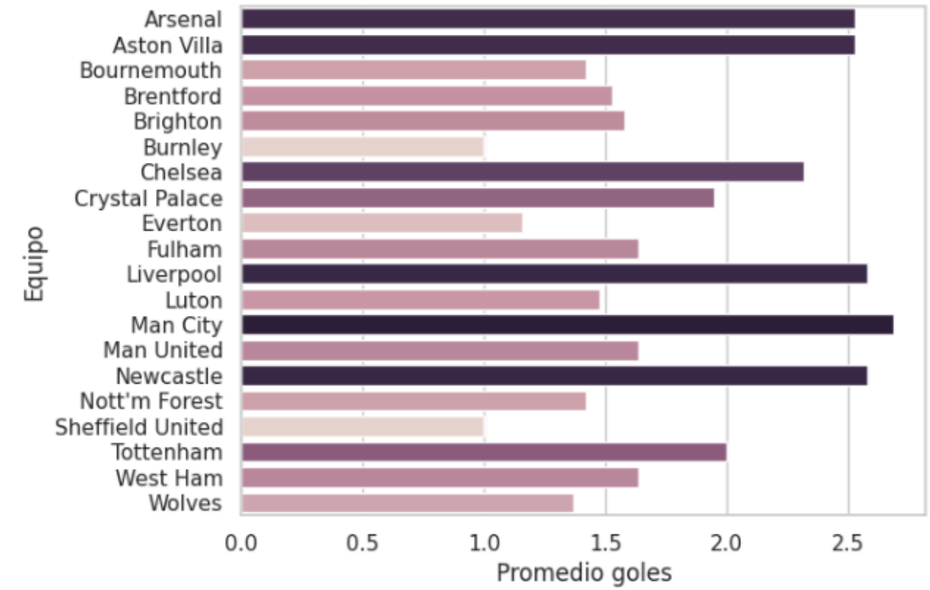
sns.barplot(
data=promedio_goles_por_equipo,
y='Equipo',
x='Promedio goles'
)

- data=promedio_goles_por_equipo → Le dice a Seaborn cuál es el DataFrame que contiene los datos.
- y='Equipo' → El eje vertical (eje Y) muestra las categorías
- x='Promedio goles' → El eje horizontal (eje X) representa los valores numéricos

¿Cómo asignarle colores distintos a cada barra tomando en cuenta el dato de los ejes?

```
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles',
    hue='Promedio goles',
    legend=False
)
```

```
<Axes: xlabel='Promedio goles', ylabel='Equipo'>
```



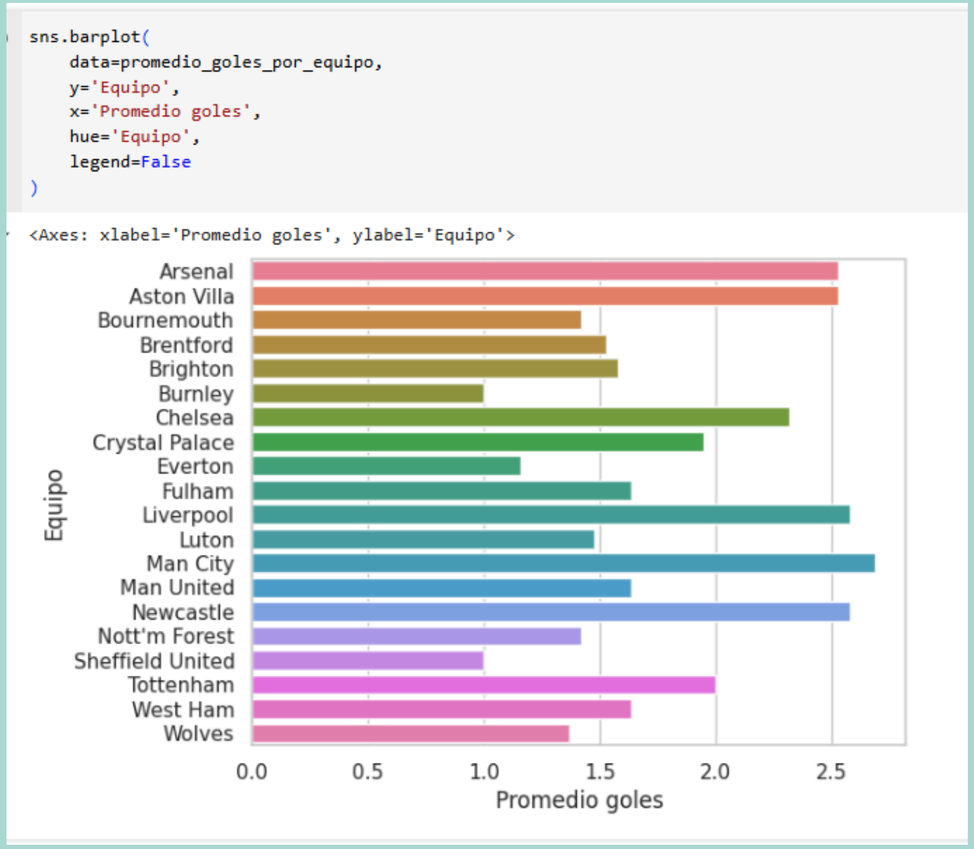
- **hue=**'Promedio goles' → Este parámetro generalmente se usa para distinguir grupos por color, según una categoría.
  - Sin embargo, en este caso se está usando con una variable numérica lo que genera una barra por cada equipo, pero asignándole un color único según su valor de promedio.
  - Es como decir: “Pinta cada barra con un color distinto, según cuántos goles promedia ese equipo.”

- **legend=False** → Se usa para ocultar la leyenda que Seaborn normalmente muestra al usar hue.
  - En este caso, como hue='Promedio goles' genera una leyenda redundante (cada barra ya muestra ese valor), tiene sentido ocultarla para que el gráfico quede más limpio.

¿Cómo asignarle colores distintos a cada barra tomando en cuenta el dato de los ejes?

```
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles',
    hue='Equipo',
    legend=False
)
```

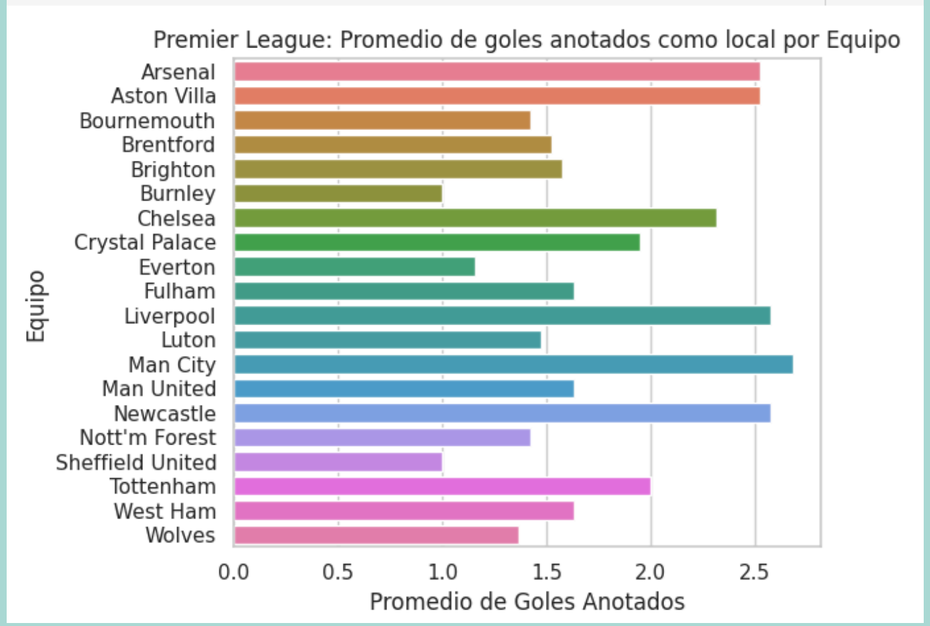
```
<Axes: xlabel='Promedio goles', ylabel='Equipo'>
```



- **hue**='Equipo' → Esta opción normalmente se usa para distinguir grupos por color, según una categoría.
  - En este caso, se usa la misma columna 'Equipo', por lo que cada barra será de un color distinto (uno por equipo).
  -
- **legend=False** → Se desactiva la leyenda del gráfico, probablemente porque ya se identifica a cada equipo en el eje Y.

¿Cómo cambiar las etiquetas con matplotlib?

```python
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles',
    hue="Equipo"
)

plt.title('Premier League: Promedio de goles anotados como local por Equipo')
plt.ylabel('Equipo')
plt.xlabel('Promedio de Goles Anotados')

plt.tight_layout()
plt.show()
```

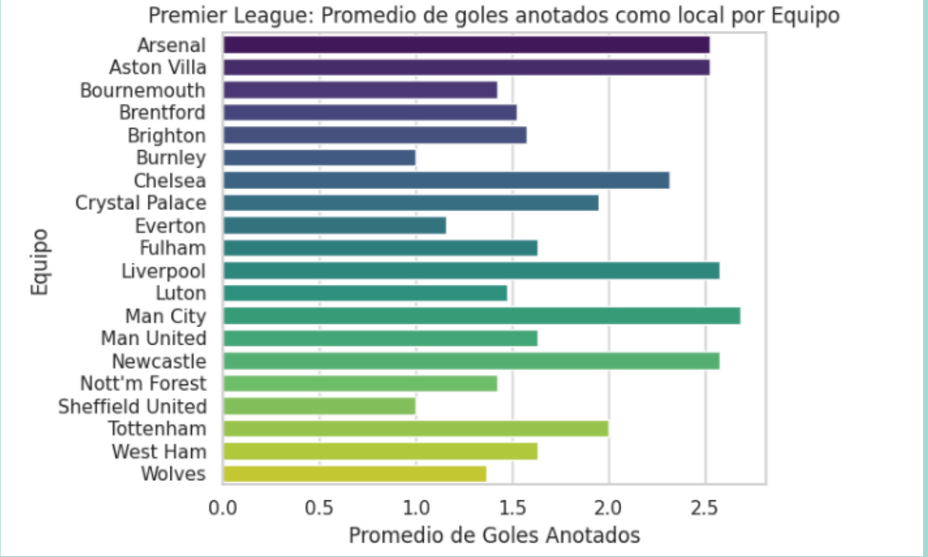


- **plt.title(...)** → Agrega un título al gráfico, que aparece en la parte superior
- **plt.xlabel(...)** → Agrega una etiqueta al eje X (horizontal)
- **plt.ylabel(...)** → Agrega una etiqueta al eje Y (vertical)
- **plt.tight_layout()** → Ajusta automáticamente los márgenes y espaciado del gráfico para que nada quede cortado y todo el contenido (título, ejes, etiquetas) se vea correctamente
- **plt.show()** → Muestra la figura completa con los subplots activos hasta ese momento

¿Cómo usar paletas de colores?

```
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles',
    palette=sns.color_palette("viridis", n_colors=len(promedio_goles_por_equipo))
)

plt.title('Premier League: Promedio de goles anotados como local por Equipo')
plt.ylabel('Equipo')
plt.xlabel('Promedio de Goles Anotados')

plt.tight_layout()
plt.show()
```

```
<ipython-input-37-464ce2a7eb50>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign th

  sns.barplot(
```



- **sns.color_palette(...)** → Es una función de Seaborn que genera una lista de colores para usar en gráficos.
  - Acepta nombres de paletas predefinidas, como: "deep", "muted", "pastel", "dark", "colorblind"
  - O paletas perceptualmente uniformes como "viridis", "plasma", "inferno", "magma", "cividis"
- **"viridis"** → Es una paleta de colores secuencial de Matplotlib, ideal para representar valores de manera clara y perceptible. Va de un azul oscuro a un amarillo verdoso brillante.
- **n_colors=len(**promedio_goles_por_equipo**)** → se indica cuántos colores quieres generar.
  - Se usa len(promedio_goles_por_equipo) para asegurarse de que haya un color diferente por cada equipo.

Definir paleta de colores en un gráfico

Seaborn facilita el uso de colores que se adaptan bien a las características de tus datos y a los objetivos de tu visualización. También puedes definir la paleta con la lista de colormaps de matplotlib.

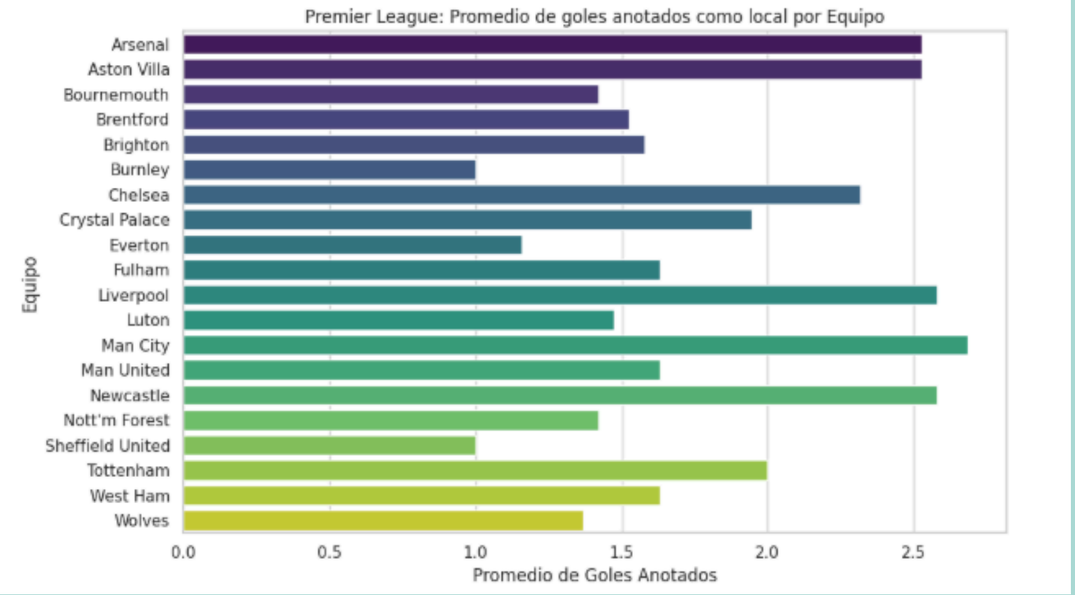
https://seaborn.pydata.org/tutorial/color_palettes.html

https://matplotlib.org/stable/users/explain/colors/colormaps.html

¿Cómo cambiar el tamaño del gráfico?

```
plt.figure(figsize=(10, 6))
sns.barplot(
    data=promedio_goles_por_equipo,
    y='Equipo',
    x='Promedio goles',
    palette=sns.color_palette("viridis", n_colors=len(promedio_goles_por_equipo))
)

plt.title('Premier League: Promedio de goles anotados como local por Equipo')
plt.ylabel('Equipo')
plt.xlabel('Promedio de Goles Anotados')

plt.tight_layout()
plt.show()
```

```
<ipython-input-39-33b5c907324a>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `leger

  sns.barplot(
```



- **plt.figure(figsize=(10, 6))** → pertenece a la librería Matplotlib y se usa para crear una nueva figura (o lienzo) donde luego se dibujarán gráficos.

- **plt.figure()** → Crea una figura nueva en la que se pueden colocar uno o más gráficos.
- **figsize=(10, 6)** → Establece el tamaño de la figura en pulgadas:
  - 10 de ancho
  - 6 de alto

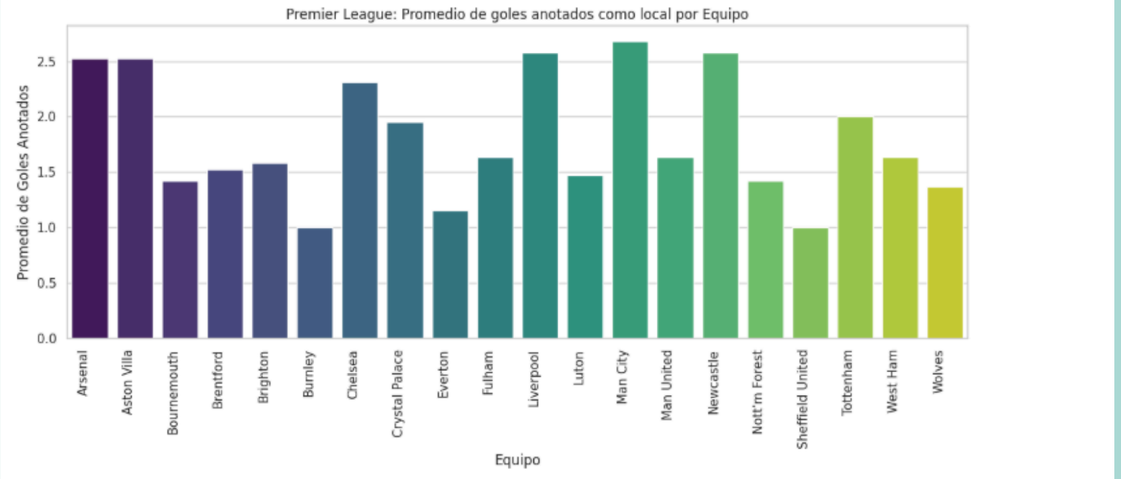
¿Para qué se usa?

- Para asegurarte de que el gráfico tenga un tamaño adecuado para:
  - Evitar que los elementos se vean apretados o superpuestos
  - Mejorar la legibilidad de etiquetas, leyendas y ejes
  - Generar figuras de mejor calidad para exportar

¿Cómo cambiar la orientación de las barras?

```python
plt.figure(figsize=(12, 6))
sns.barplot(
    data=promedio_goles_por_equipo,
    x='Equipo',
    y='Promedio goles',
    palette=sns.color_palette("viridis", n_colors=len(promedio_goles_por_equipo))
)

plt.title('Premier League: Promedio de goles anotados como local por Equipo')
plt.xlabel('Equipo')
plt.ylabel('Promedio de Goles Anotados')

plt.xticks(rotation=90, ha='right')

plt.tight_layout()

plt.show()
```

```
<ipython-input-43-b690bf2184cb>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(
```



- **x**='Equipo' → La columna 'Equipo' se usará en el eje X. Es decir, cada equipo aparecerá como una categoría a lo largo del eje horizontal.
- **y**='Promedio goles' → La columna 'Promedio goles' se usará en el eje Y, indicando cuántos goles promedio tiene cada equipo.

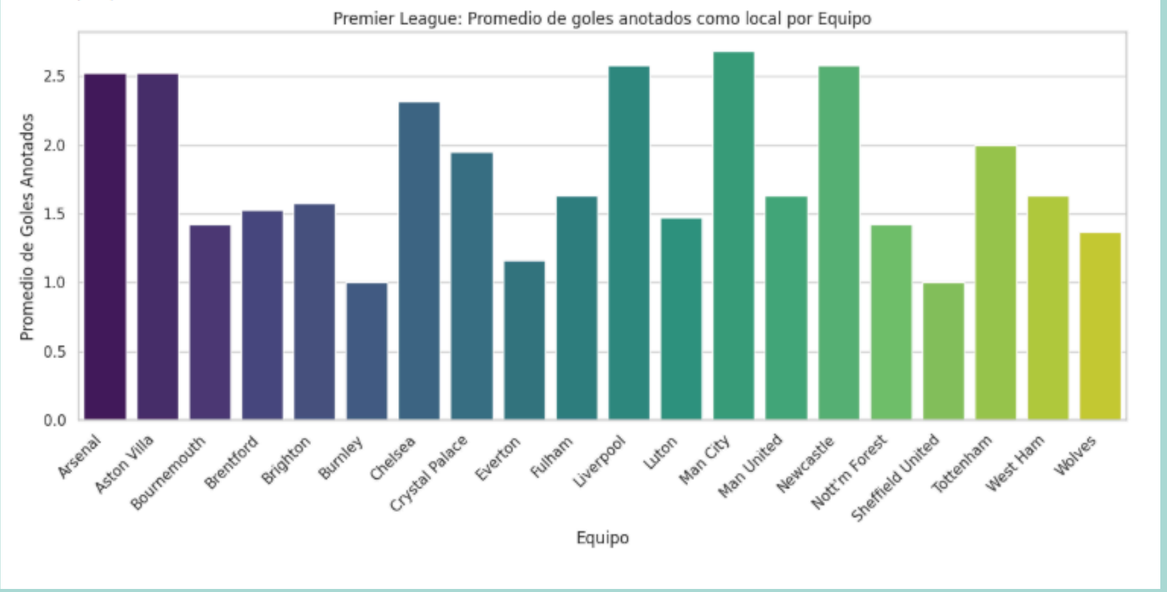
- Este tipo de configuración genera un gráfico de barras vertical, donde:
  - Cada barra representa a un equipo.
  - La altura de la barra muestra su promedio de goles.

- **plt.xticks(rotation=90, ha='right')** → se utiliza para ajustar la orientación y alineación de las etiquetas del eje X

¿Cómo cambiar la orientación de las etiquetas?

```
plt.figure(figsize=(12, 6))
sns.barplot(
    data=promedio_goles_por_equipo,
    x='Equipo',
    y='Promedio goles',
    palette=sns.color_palette("viridis", n_colors=len(promedio_goles_por_equipo))
)

plt.title('Premier League: Promedio de goles anotados como local por Equipo')
plt.xlabel('Equipo')
plt.ylabel('Promedio de Goles Anotados')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```

```
<ipython-input-42-d97dd240955a>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same

  sns.barplot(
```



- **plt.xticks(**rotation=45, ha='right'**)** → sirve para modificar la apariencia de las etiquetas del eje X en un gráfico
  - Ajusta la orientación y alineación horizontal de las etiquetas en el eje X

- **rotation**=45 → Rota las etiquetas del eje X a 45 grados en sentido antihorario
  - Esto es útil cuando las etiquetas son largas o están muy juntas, porque evita que se sobrepongan.
- **ha**='right'
  - **ha** significa alineamiento horizontal
  - 'right' alinea el extremo derecho del texto con su posición en el eje X
  - Esto hace que las etiquetas se vean más ordenadas después de rotarlas.

Revisa el siguiente link si deseas ver otros gráficos que puedes realizar con seaborn

https://seaborn.pydata.org/examples/index.html

Revisa el siguiente link si deseas ver el Colab con los ejemplos presentados en esta presentación