Visualización de datos

Semana Tec

Profesores: Luz Eunice Angeles /José Ignacio Treviño

Librerías de visualización de datos

- **Matplotlib es la librería más utilizada para crear gráficos en Python.**
  - **Está basada en la graficación de Matlab.**
  - **Permite agregar elementos sobre el gráfico, modificar títulos y leyendas, cambiar colores, entre muchas otras funciones.**
  - **Como desventaja, para cada detalle que queramos agregar es probable que requiramos agregar una línea de código.**
- **Seaborn es una librería que está basada en Matplotlib.**
  - **Tiene la ventaja de que los gráficos son más bonitos por defecto y que utiliza menos líneas de código.**
  - **Como desventaja, para detalles muy específicos requiere que agreguemos código de matplotlib.**
- **Plotly es una librería para crear gráficos tipo dashboard.**
  - **Las visualizaciones creadas con esta librería son interactivas.**
  - **Como principal desventaja, el código para generar este tipo de gráficos es más complejo que los casos anteriores.**

Matplotlib

- **La utilizamos cuando trabajamos con datos simples.**
- **Nos sirve para configurar cada elemento del gráfico:**
  - **Tamaño de la imagen.**
  - **Tipo de gráfico.**
  - **Colores y marcas**
  - **Leyendas y nombres de los ejes.**
  - **Título.**
  - **Cuadrícula, ejes primarios y secundarios.**
- **Para utilizarla, necesitamos imporatarla:**

```
import matplotlib.pyplot as plt
```

- **Normalmente usamos el siguiente formato:**

```
# Para configurar el tamaño de la imagen
fig = plt.figure(figsize=(<width, height>))
# Con la siguiente instruccion dibujamos un gráfico de línea
plt.plot(<x>, <y>, color = <"color">, label = <"etiqueta">)
# Las siguientes líneas son para agregar elementos del gráfico.
plt.title(<"título">) # Para agregar el título
plt.xlabel(<"título del eje x">) # Etiqueta del eje X
plt.ylabel(<"título del eje y">) # Etiqueta del eje Y
plt.legend(loc='best') # Para mostrar las etiquetas de datos.
plt.grid(<Booleano>) # Para agregar cuadrícula
```

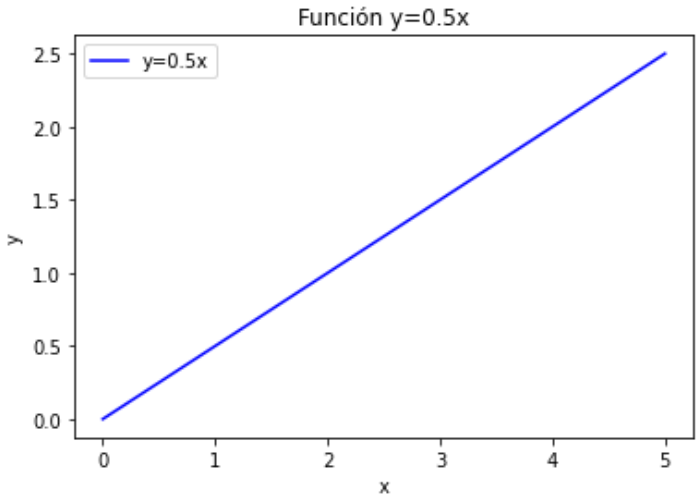
- **Vamos a graficar la función** $y_1 = 0.5 \cdot x$

In [1]:

```
# Primero hay que crear los datos
import numpy as np

x = np.linspace(0,5,num=50)
y1 = 0.5*x

# Importamos matplotlib para graficar
import matplotlib.pyplot as plt
# Configuramos el tamaño de imagen
fig = plt.figure(figsize=(6,4))
# Agregamos la gráfica
plt.plot(x, y1, color='b', label='y=0.5x')
# Agregamos títulos a los ejes y al gráfico
plt.xlabel('x')
plt.ylabel('y')
plt.title('Función y=0.5x')
# Cuando tenemos una sola línea, legend no hace mucho sentido.
plt.legend(loc='best')
```

Out[1]:

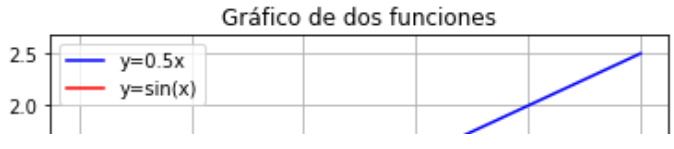
```
<matplotlib.legend.Legend at 0x7f64f9a94e50>
```

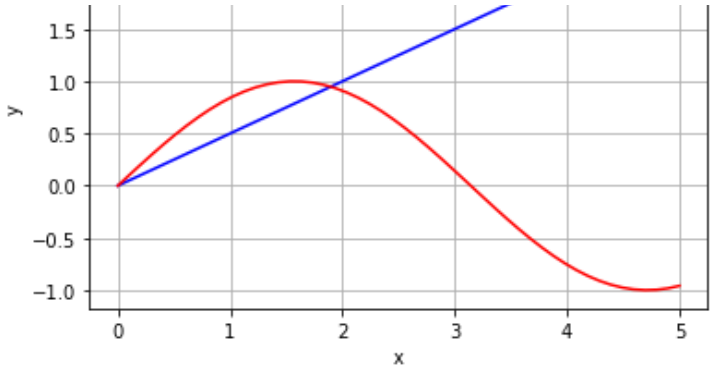


- **Podemos agregar más de una línea en un mismo gráfico.**
- **Vamos a graficar la función $y_2 = \sin(x)$ sobre el mismo gráfico anterior.**

In [2]:

```
# Hay que crear el vector de la segunda función
y2 = np.sin(x)

# Usamos el mismo código del gráfico anterior, pero agregamos un segundo plot
# Configuramos el tamaño de imagen
fig = plt.figure(figsize=(6,4))
# Agregamos la 1ra gráfica. En color, la 'b' significa blue.
plt.plot(x, y1, color='b', label='y=0.5x')
# Agregamos la 2da gráfica. Aquí cambiamos el color a rojo.
plt.plot(x, y2, color='r', label='y=sin(x)')

# Agregamos títulos a los ejes y al gráfico
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráfico de dos funciones')
# Aquí la leyenda hace mucho más sentido
plt.legend(loc='best')
# Agregamos la cuadrícula para que se vea mejor
plt.grid(True)
```

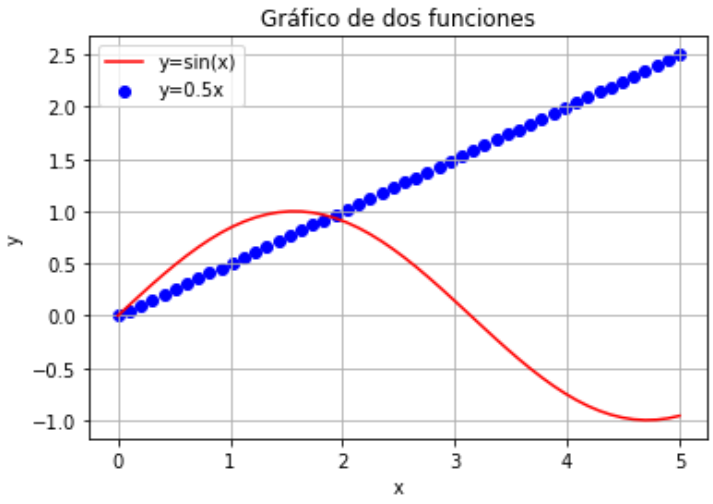




- `plot` **grafica *líneas continuas*.**
- **Si queremos graficar *puntos*, usamos la función** `scatter`**.**
- **Podemos mezclar los tipos de gráfico en una misma imagen.**

In [3]:

```
# Configuramos el tamaño de imagen
fig = plt.figure(figsize=(6,4))

# Agregamos la 2da gráfica primero. El orden no importa.
plt.plot(x, y2, color='r', label='y=sin(x)')
# Agregamos la 1ra gráfica. Cambiamos a un scatter
plt.scatter(x, y1, color='b', label='y=0.5x')

# Agregamos títulos a los ejes y al gráfico
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gráfico de dos funciones')
# Aquí la leyenda hace mucho más sentido
plt.legend(loc='best')
# Agregamos la cuadrícula para que se vea mejor
plt.grid(True)
```



- **Cuando trabajamos con datos *tabulares*, en matplotlib tenemos que configurar cada elemento del gráfico y puede resultar en muchas líneas de código.**
- **Seaborn nos ayuda mucho cuando queremos graficar DataFrames de manera rápida y en menos líneas de código.**

Seaborn

- **Para graficar con Seaborn, la tabla debe tener formato largo.**
- **Cada columna debe representar una variable y cada renglón una observación.**
- **Cuando en las columnas tenemos valores de una variable (por ejemplo, distintos años), la tabla no tiene formato largo.**
- **La forma general de graficar con Seaborn es:**

```
# Importamos la librería
import seaborn as sns
# Seaborn hace más bonitas las gráficas.
# Usamos set para definir el estilo de Seaborn para todas las gráficas.
sns.set()
# La gráfica en sí:
sns.<tipo_de_grafico>(data = <df>, x=<"columna X">,
                      y = <"columna y"> , hue = <"columna color">,
                      size = <"columna tamaño">, style = <'columna estilo'>,
                      palette = <'paleta de colores'>, cmap,...)
```

- **Vamos a cargar el dataset *Iris* y hacer algunas gráficas.**

In [4]:

```
# Si trabajamos en Google Colaboratory corremos las siguientes lineas de código
from google.colab import drive
drive.mount('/gdrive')
```

```
Mounted at /gdrive
```

In [8]:

```
# Nos cambiamos a la carpeta donde tengamos el repositorio
%cd '/gdrive/MyDrive/semanaTec/arte-analitica'
```

```
/gdrive/MyDrive/semanaTec/arte-analitica
```

In [9]:

```
# Cargamos el dataset
import pandas as pd
import numpy as np

iris_df = pd.read_csv('data/iris.csv')
iris_df.head()
```

Out[9]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [10]:

```
# Importamos seaborn
import seaborn as sns
sns.set()
```

Histogramas.

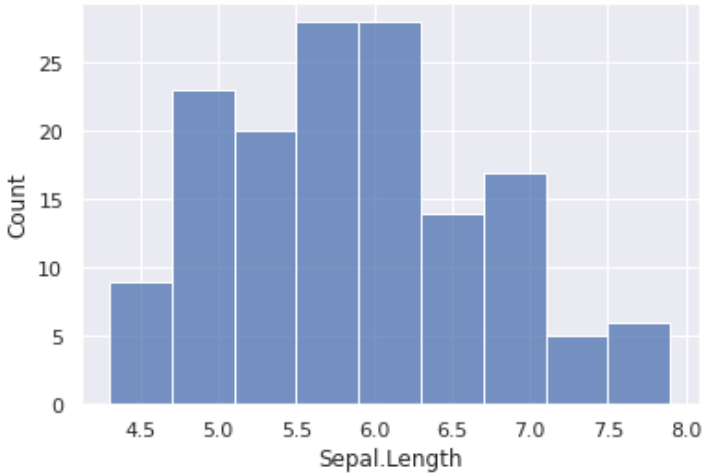
- **Los histogramas nos ayudan a visualizar la distribución de una variable numérica.**
- **El eje x está dividido en *secciones* o *cajas*. Este parámetro se llama *bins* en la función de Seaborn.**
- **El eje y muestra cuántos datos cayeron en cada *caja* en las que se dividió la variable que estamos graficando.**

In [11]:

```
# Graficamos un histograma con histplot.
# Sólo hay que definir el DF donde tomamos los datos y cuáles columnas queremos graficar.
sns.histplot(data=iris_df, x='Sepal.Length')
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f3280750>
```



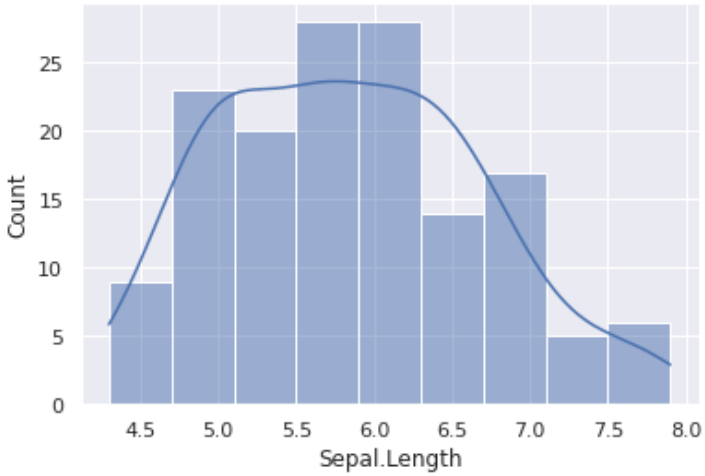
- **Con el parámetro *bins* controlamos el número de cajas. Por defecto, *bins= 10*.**
- **Entre más cajas grafiquemos, hay mayor detalle.**
- **Si el número de cajas es muy grande, no obtendremos una interpretación correcta del gráfico.**
- **Si el número de cajas es muy chico para el mismo problema.**
- **El parámetro *kde* nos sirve para obtener un estimado de la función de distribución.**

In [12]:

```
sns.histplot(data=iris_df, x='Sepal.Length', bins=9, kde=True)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64e3cbb750>
```



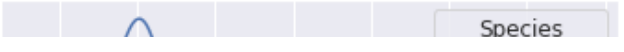
- **El parámetro *hue* separa los datos y pone un color diferente dependiendo de la variable que utilicemos.**
- **Los parámetros como *hue*, y *style* solamente son útiles cuando tenemos variables categóricas**

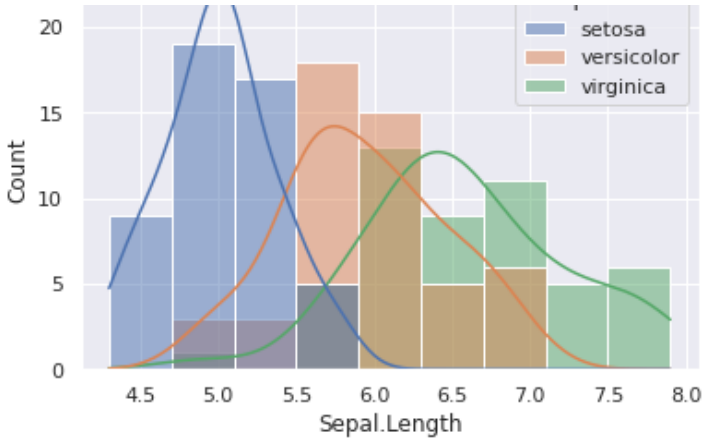
In [13]:

```
# Usamos la variables Species para separar los datos.
# En histogramas, al hacer esta separación debemos tener pocas categorías.
sns.histplot(data=iris_df, x='Sepal.Length', hue='Species', kde=True)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64f337e9d0>
```





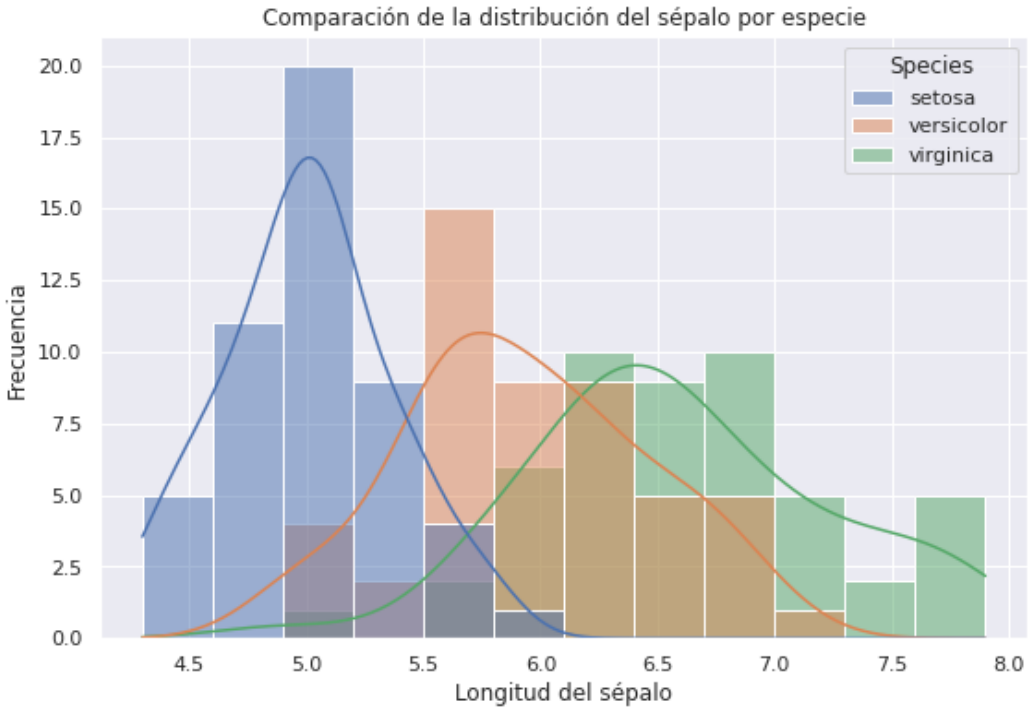
- **Por defecto, Seaborn usa el nombre de las columnas para definir las etiquetas de los ejes.**
- **Podemos usar las funciones de matplotlib para modificar los elementos del gráfico y el tamaño.**

In [14]:

```
# Tamaño de la imagen
fig = plt.figure(figsize=(9,6))
# Gráfico
sns.histplot(data=iris_df, x='Sepal.Length', hue='Species', bins=12,kde=True)
# Ejes y título
plt.xlabel('Longitud del sépalo')
plt.ylabel('Frecuencia')
plt.title('Comparación de la distribución del sépalo por especie')
```

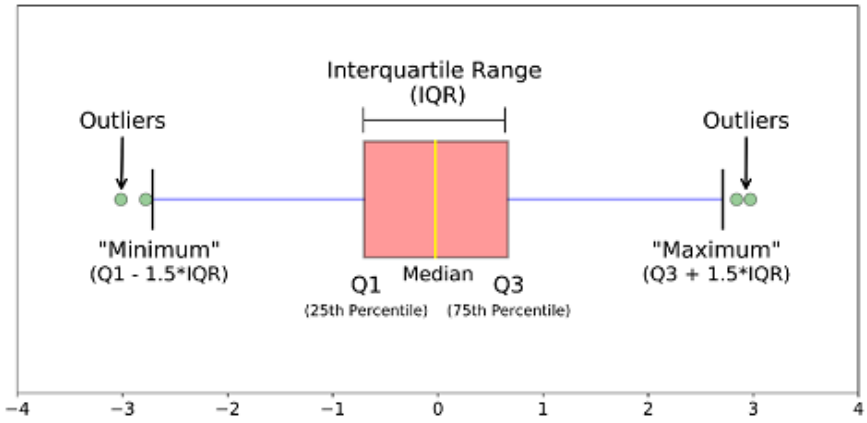
Out[14]:

```
Text(0.5, 1.0, 'Comparación de la distribución del sépalo por especie')
```



Gráficos de caja (Boxplot)

- **Un diagrama de caja (y bigotes) es un método estandarizado para representar gráficamente una serie de datos numéricos a través de sus cuartiles.**
- **Este gráfico es muy útil porque nos permite visualizar de manera rápida estos estadísticos:**
  - **1er cuartil**
  - **3er cuartil**
  - **Mediana**
  - **Máximo y mínimo**
  - **Valores atípicos.**



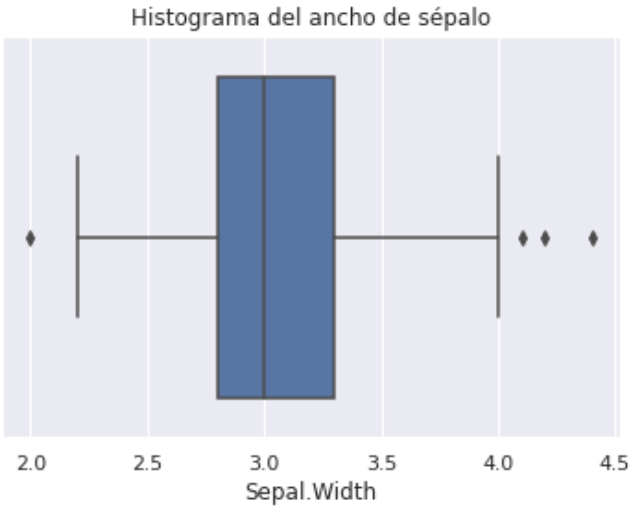
- **La usamos cuando queremos comparar una variable numérica que podemos separar en varios grupos o categorías.**
- **Es muy similar al último histograma que realizamos, pero la comparación es más clara.**
- **La ventaja es que podemos agregar más de dos categorías sin perder visibilidad de los datos.**

In [15]:

```
# Tamaño de la imagen
fig = plt.figure(figsize=(6,4))
# Gráfico boxplot
sns.boxplot(data=iris_df, x='Sepal.Width')
# Ejes y título
plt.title('Histograma del ancho de sépalo')
```

Out[15]:

```
Text(0.5, 1.0, 'Histograma del ancho de sépalo')
```



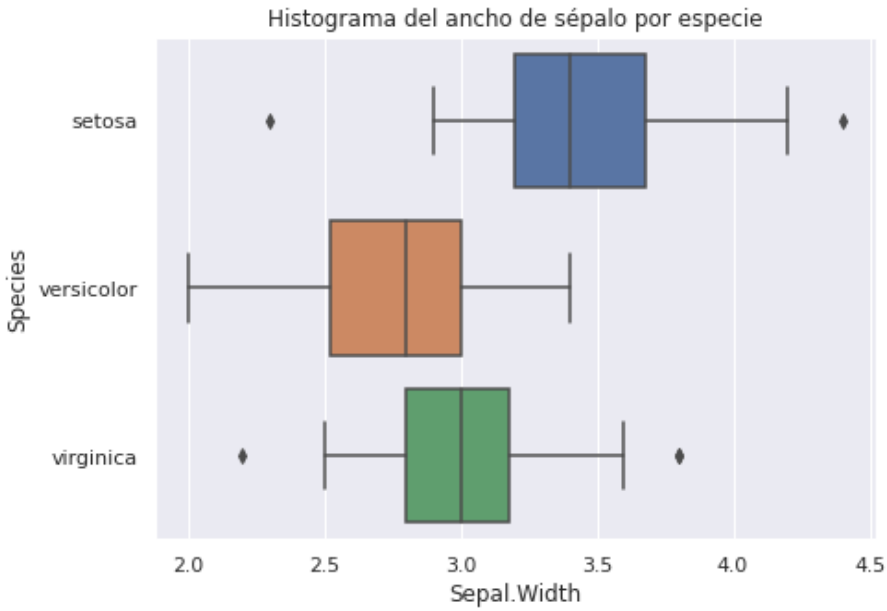
- **Los puntos en el histograma representan datos atípicos (outliers).**
- **Estos datos presentan valores extremos, es decir, que están muy por encima comparados con el resto de las observaciones.**
- **Ahora, veamos cómo se distribuye el ancho de sépalo por especie. Esta separación la hacemos usando la variable *Species* en el eje y.**

In [16]:

```
# Tamaño de la imagen
fig = plt.figure(figsize=(7,5))
# Gráfico boxplot
sns.boxplot(data=iris_df, x='Sepal.Width', y='Species')
# Ejes y título
plt.title('Histograma del ancho de sépalo por especie')
```

Out[16]:

```
Text(0.5, 1.0, 'Histograma del ancho de sépalo por especie')
```



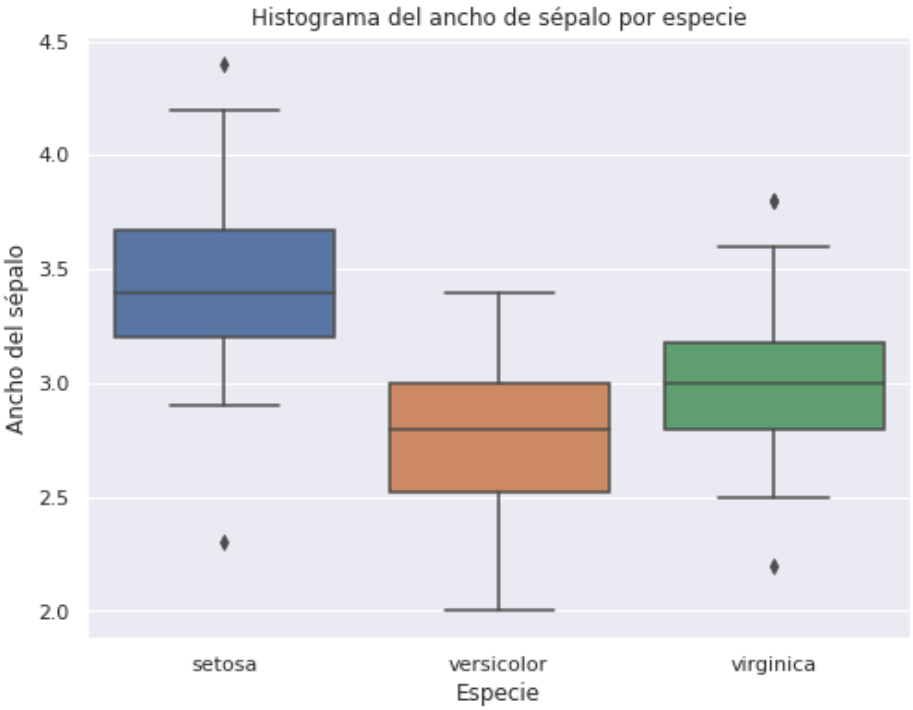
- **La ventaja de los diagramas de caja es que podemos invertir los ejes dependiendo de cómo se presenten mejor los datos.**
- **Si la variable categorica que usamos para separar los datos tiene muchos niveles, lo recomendable es usarla en el eje *y*.**

In [17]:

```
# Tamaño de la imagen
fig = plt.figure(figsize=(8,6))
# Gráfico boxplot. Invertimos los ejes.
sns.boxplot(data=iris_df, y='Sepal.Width', x='Species')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Histograma del ancho de sépalo por especie')
plt.xlabel('Especie')
plt.ylabel('Ancho del sépalo')
```

Out[17]:

```
Text(0, 0.5, 'Ancho del sépalo')
```



Mapa de calor (Heatmap)

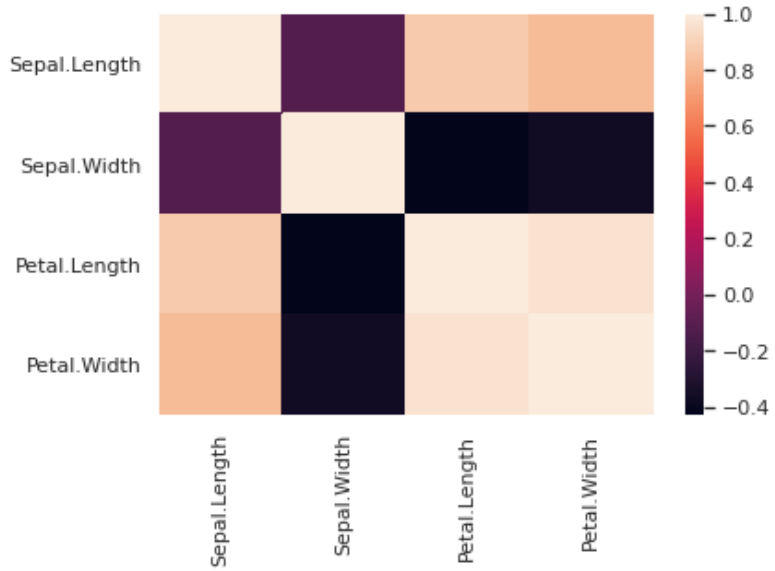
- **Esta técnica de visualización de datos nos muestra la magnitud de una variable que se distribuye en dos dimensiones.**
- **En esencia, es colorear una tabla donde la *intensidad* del color corresponde al *valor* de cada celda.**
- **Es muy útil cuando tenemos tablas en formato ancho, donde tenemos valores de variables en las columnas y en las filas.**
- **Normalmente, lo utilizamos para visualizar una matriz de correlación.**

In [18]:

```
# Vamos a graficar la matriz de correlación del dataset Iris
iris_corr = iris_df.corr()

# Para graficar el mapa de calor usamos heatmap. No necesitamos especificar x ni y
sns.heatmap(data=iris_corr)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64e19421d0>
```



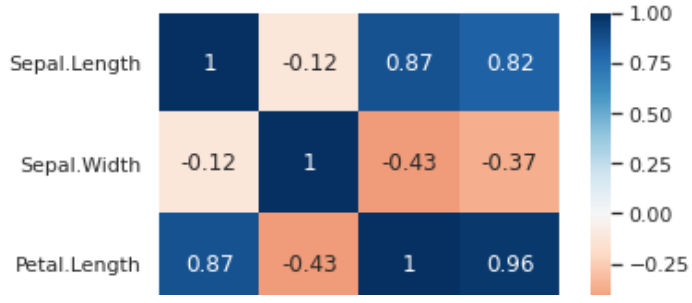
- **La barra lateral es el mapa de color, que nos muestra la correspondencia del valor de la tabla a un color en específico.**
- **Existen diferentes mapas de colores que podemos utilizar, la lista está en la siguiente liga:**
  - **Mapas de Color. Matplotlib**
- **Debemos escoger el mapa de color y la escala correcta dependiendo de los datos.**
- **Sabemos que la correlación va de -1 a 1. Esta es nuestra escala.**
- **Nos interesa saber si la correlacióne es positiva o negativa, nos conviene un mapa divergente.**

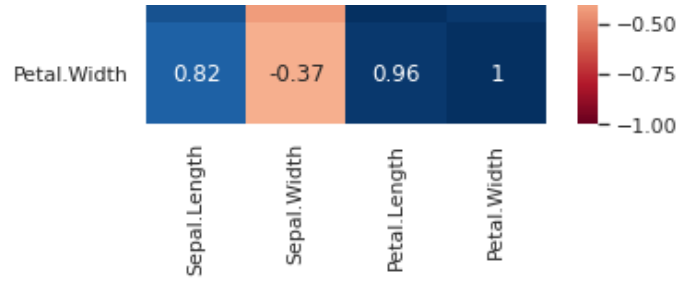
In [19]:

```
# Gráfico heatmap. Seleccionamos los valores extremos con vmin y vmax.
# El mapa de color que usaremos es de un extremo azul y del otro rojo.
# Con annot podemos desplegar el valor de cada celda
# Con square hacemos que el gráfico sea simétrico en tamaño de ejes
sns.heatmap(data=iris_corr, vmin=-1, vmax=1, cmap = 'RdBu', annot=True, square = True)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64e18b7f90>
```





Gráficos de barra

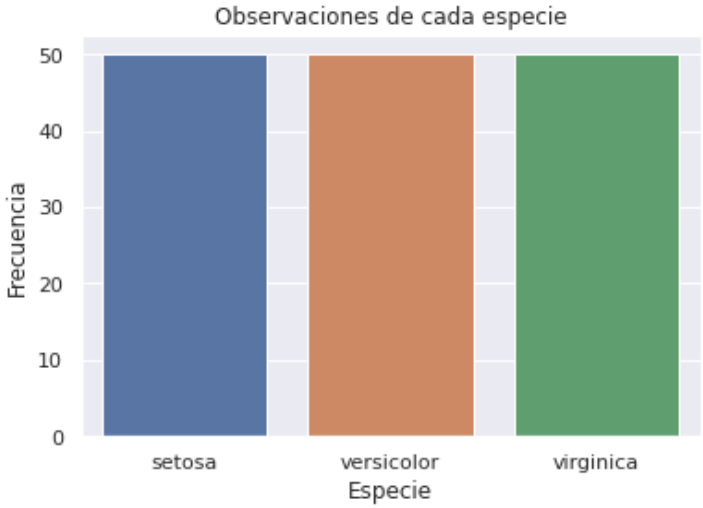
- **Esta visualización la usamos cuando queremos graficar los valores de una variable categórica.**
- **Es el equivalente al *conteo* de las apariciones de cada clase de la variable categórica.**
- **Con Seaborn, python hace el conteo automáticamente.**

In [20]:

```
# Vamos a graficar el conteo de la variable categórica (Species)
# Tamaño de la imagen
fig = plt.figure(figsize=(6,4))

# Gráfico countplot para hacer barras con el número de apariciones de cada especie.
sns.countplot(data=iris_df, x = 'Species')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Observaciones de cada especie')
plt.xlabel('Especie')
plt.ylabel('Frecuencia')
```

Out[20]:

```
Text(0, 0.5, 'Frecuencia')
```



- **Al igual que el boxplot, podemos invertir los ejes si nos conviene.**
- **Si hay muchas categorías, lo mejor es tenerlas en el eje Y.**

In [21]:

```
# Vamos a graficar el conteo de la variable categórica (Species)
# Tamaño de la imagen
fig = plt.figure(figsize=(6,4))

# Gráfico countplot para hacer barras con el número de apariciones de cada especie.
sns.countplot(data=iris_df, y = 'Species')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Observaciones de cada especie')
plt.xlabel('Frecuencia')
plt.ylabel('Especie')
```

Out[21]:

```
Text(0, 0.5, 'Especie')
```

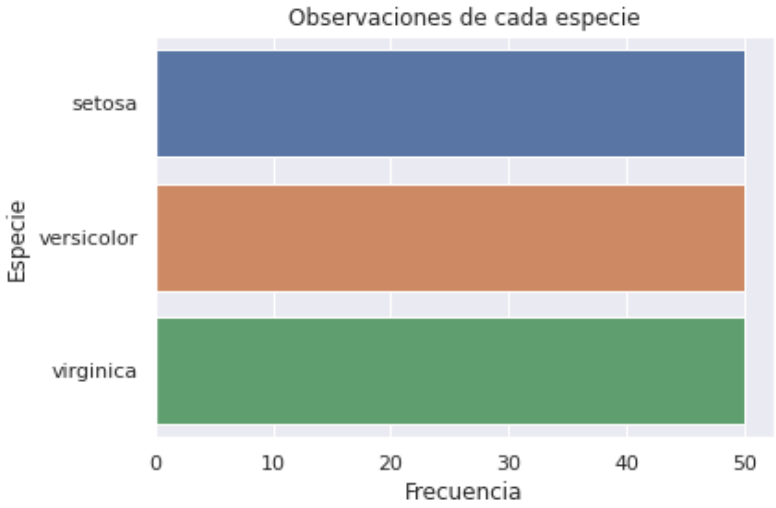


Gráfico de dispersión (scatterplot).

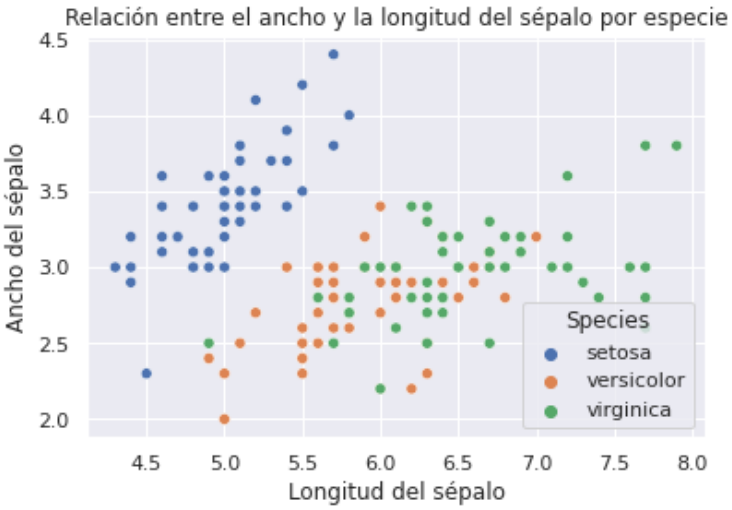
- **Los usamos cuando queremos ver la relación entre dos variables numéricas.**
- **Podemos usar variables categóricas para modificar el estilo o el color de cada punto.**
- **Es más rápido usar esta función que con matplotlib.**

In [22]:

```
# Vamos a graficar la relación entre el ancho y la longitud del sépalo.

# Tamaño de la imagen
fig = plt.figure(figsize=(6,4))

# Gráfico scatterplot.
sns.scatterplot(data=iris_df, x = 'Sepal.Length', y='Sepal.Width', hue='Species')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Relación entre el ancho y la longitud del sépalo por especie')
plt.xlabel('Longitud del sépalo')
plt.ylabel('Ancho del sépalo')
```

Out[22]:

```
Text(0, 0.5, 'Ancho del sépalo')
```



- **Si queremos visualizar la relación uno a uno entre todas las variables numéricas, podemos usar la función *pairplot*.**
- **La estructura es similar a la matriz de correlación.**

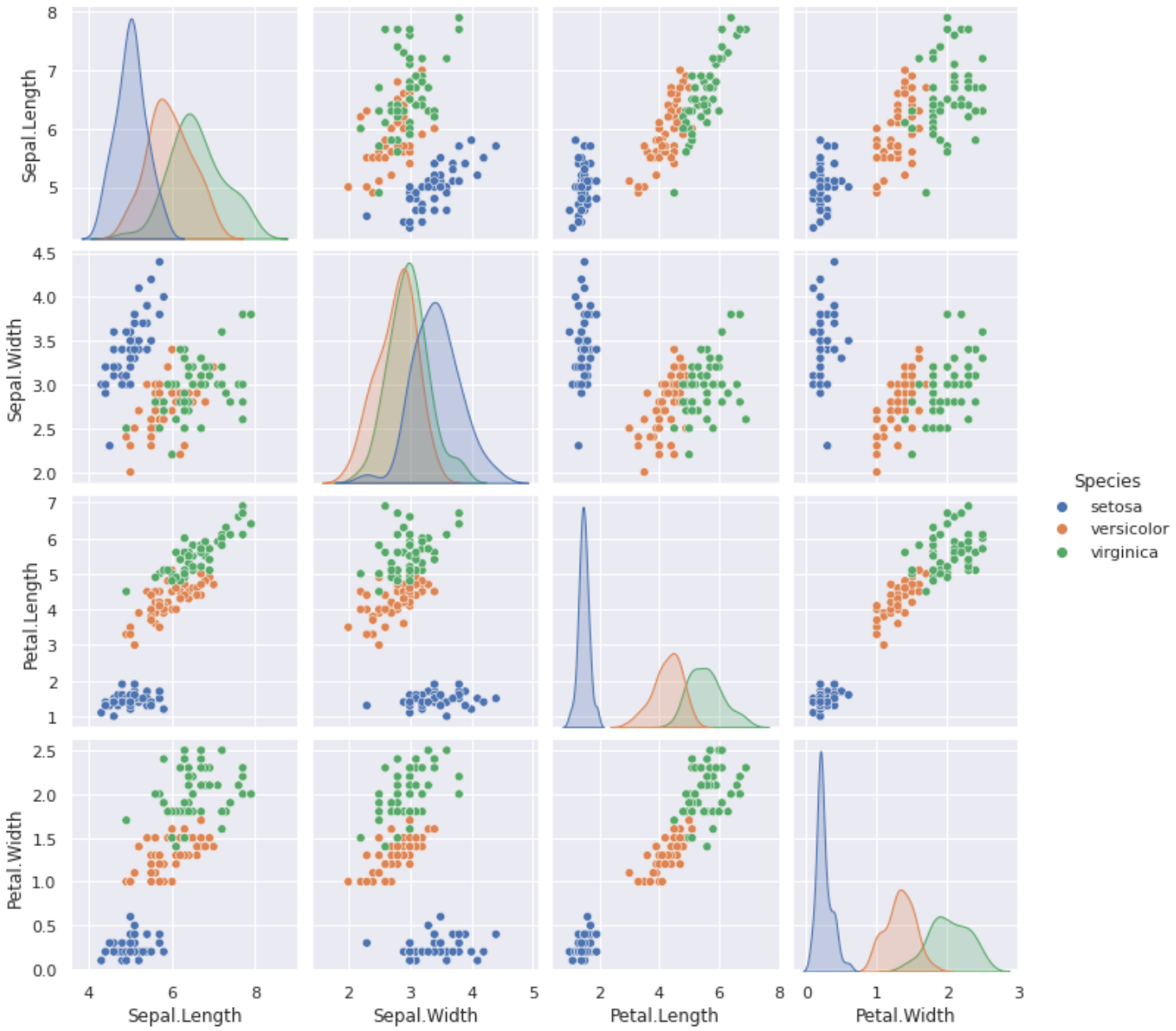
In [23]:

```
# Pairplot es una versión más compleja y grande del gráfico de dispersión.
```

```
sns.pairplot(data=iris_df, hue='Species')
```

Out[23]:

```
<seaborn.axisgrid.PairGrid at 0x7f64d8feec90>
```



Actividad

- **Nombre:** Daira Adriana Chavarría Rodríguez
- **Matrícula:** A01274745

Entregar

Archivo PDF de la actividad y la liga de la actividad en su repostitorio.

Nota:

Todas las tareas entregadas fuera de la fecha limite se califican sobre 50 de los 100 puntos posibles.

Highway MPG dataset

Este dataset contiene variables medidas por la agencia de protección ambiental de 38 modelos de vehículo diferentes de 1999 a 2008. Las variables que se registraron fueron:

- **manufacturer:** nombre del fabricante.
- **model:** nombre del modelo

- **model**: nombre del modelo.
- **displ**: desplacamiento del motor, en litros.
- **year**: año de fabricación.
- **cyl**: número de cilindros.
- **trans**: tipo de transmisión.
- **drv**: tipo de tracción, f-delantera, r-trasera, 4- 4 llantas
- **cty**: rendimiento del motor en ciudad, en millas por galón.
- **hwy**: rendimiento del motor en carretera, en millas por galón.
- **fl**: tipo de combustible.
- **class**: tipo de vehículo.

Los datos están en el archivo `auto-mpg.csv` dentro de la carpeta `data`.

In [28]:

```
# Carga las librerías y datos
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

auto_df = pd.read_csv('data/auto-mpg.csv')
sns.set()

auto_df
```

Out[28]:

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
...	...	...	...	...	...	...	...	...	...	...	...
229	volkswagen	passat	2.0	2008	4	auto(s6)	f	19	28	p	midsize
230	volkswagen	passat	2.0	2008	4	manual(m6)	f	21	29	p	midsize
231	volkswagen	passat	2.8	1999	6	auto(l5)	f	16	26	p	midsize
232	volkswagen	passat	2.8	1999	6	manual(m5)	f	18	26	p	midsize
233	volkswagen	passat	3.6	2008	6	auto(s6)	f	17	26	p	midsize

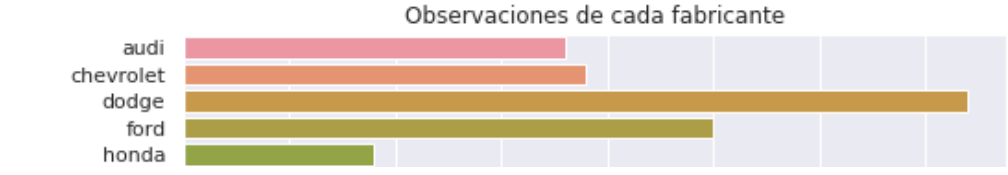
234 rows × 11 columns

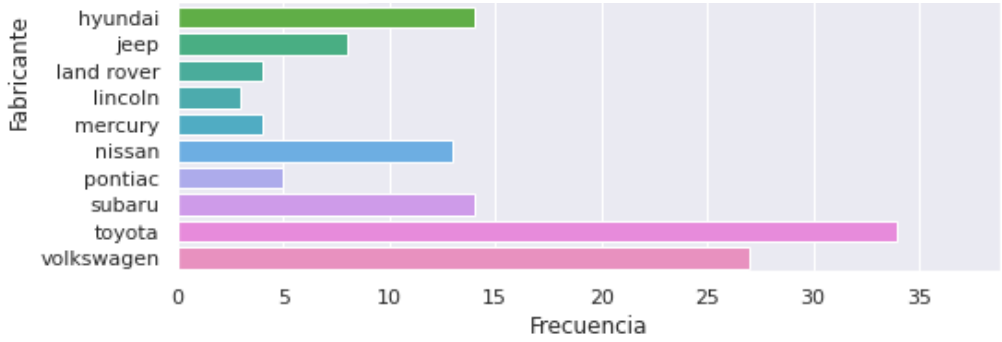
In [49]:

```
# ¿Cuántas datos hay de cada fabricante? Muéstralo en un gráfico.
# Tamaño de la imagen
fig = plt.figure(figsize=(8,4))

sns.countplot(data=auto_df, y = 'manufacturer')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Observaciones de cada fabricante')
plt.ylabel('Fabricante')
plt.xlabel('Frecuencia')
```

Out[49]:

```
Text(0.5, 0, 'Frecuencia')
```



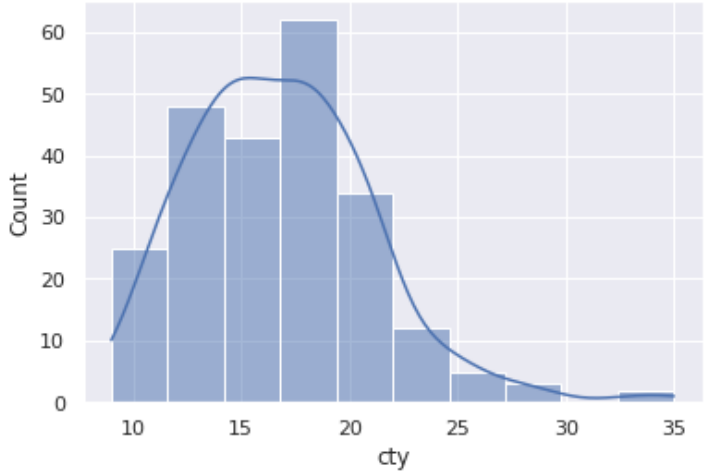


In [44]:

```
# Haz un histograma de las variables numéricas cty, hwy (uno para cada variable, en celdas distintas)
sns.histplot(data=auto_df, x='cty', bins=10, kde = True)
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64d1d1e0d0>
```

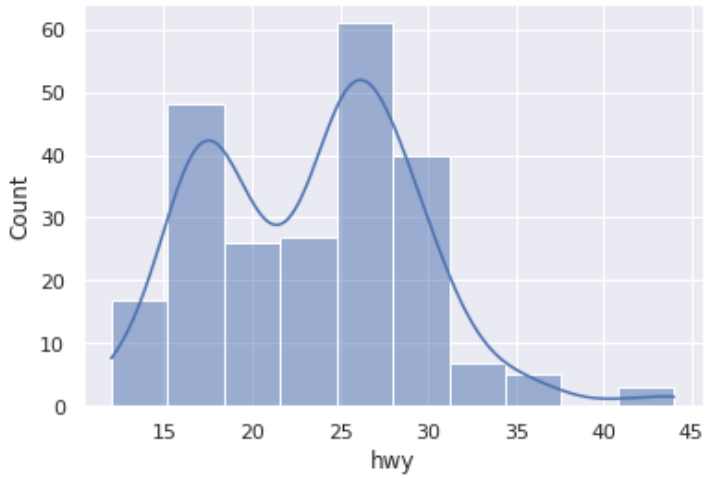


In [45]:

```
sns.histplot(data=auto_df, x='hwy', bins=10, kde = True)
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f64d1cbacd0>
```



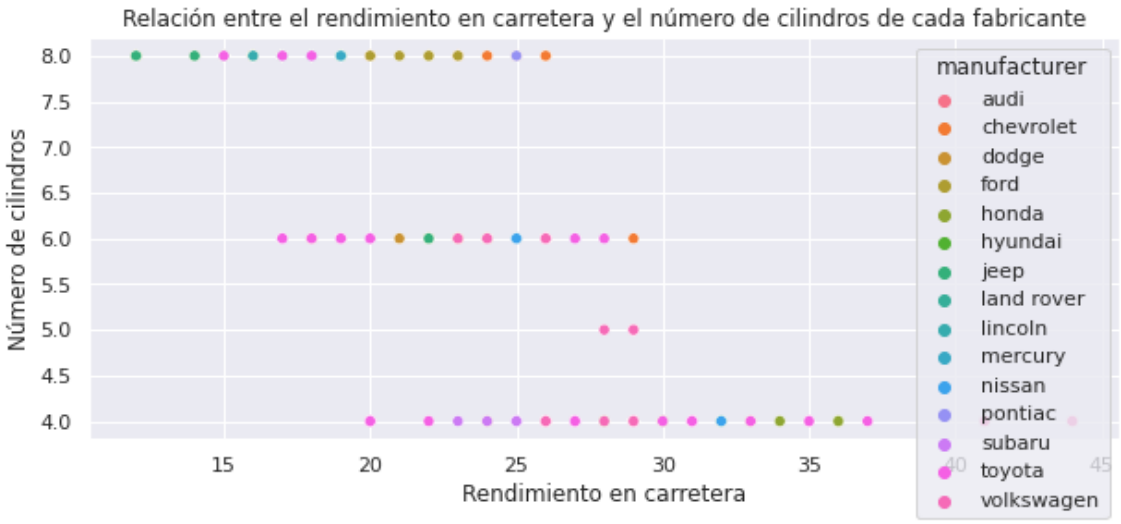
In [40]:

```
# ¿Cómo se comparar el rendimiento en carretera (hwy) contra el número de cilindros?
# Haz un gráfico que represente esta relación.
# Tamaño de la imagen
fig = plt.figure(figsize=(10,4))

# Gráfico scatterplot.
```

```
sns.scatterplot(data=auto_df, x = 'hwy', y='cyl', hue='manufacturer')
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Relación entre el rendimiento en carretera y el número de cilindros de cada fa
bricante')
plt.xlabel('Rendimiento en carretera')
plt.ylabel('Número de cilindros')
```

Out[40]:

```
Text(0, 0.5, 'Número de cilindros')
```

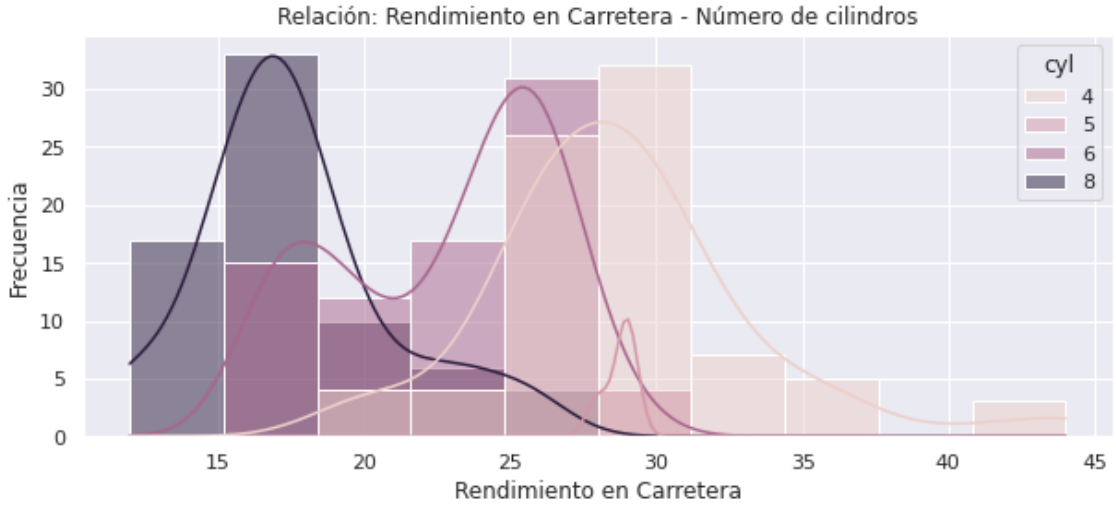


In [55]:

```
# cyl es una variable numérica, pero como tiene pocos valores podemos tomarla como si fue
ra categórica.
# ¿De qué otra forma se puede representar la relación de la pregunta anterior?
# (¿Qué otra gráfica podemos usar?)
fig = plt.figure(figsize=(10,4))
sns.histplot(data=auto_df, x="hwy", hue = "cyl", bins=10, kde=True)
plt.title("Relación: Rendimiento en Carretera - Número de cilindros")
plt.xlabel("Rendimiento en Carretera")
plt.ylabel("Frecuencia")
```

Out[55]:

```
Text(0, 0.5, 'Frecuencia')
```



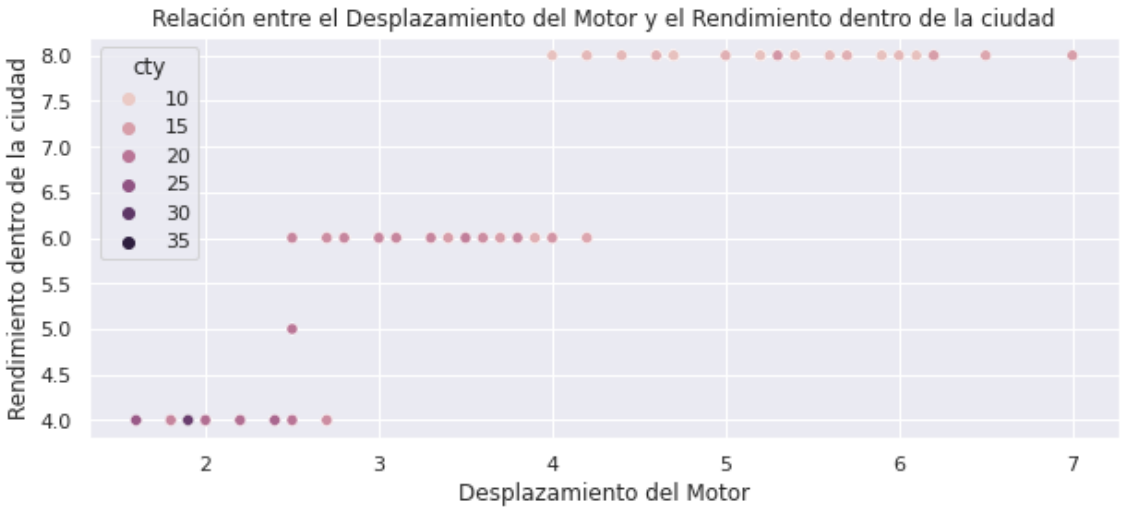
In [56]:

```
# Cuál es la relación entre el desplazamiento del motor y el rendimiento dentro de la ciu
dad?
# Tamaño de la imagen
fig = plt.figure(figsize=(10,4))

# Gráfico scatterplot.
sns.scatterplot(data=auto_df, x = 'displ', y='cyl', hue='cty')
```

```
# Ejes y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Relación entre el Desplazamiento del Motor y el Rendimiento dentro de la ciuda
d')
plt.xlabel('Desplazamiento del Motor')
plt.ylabel('Rendimiento dentro de la ciudad')
```

Out[56]:

```
Text(0, 0.5, 'Rendimiento dentro de la ciudad')
```



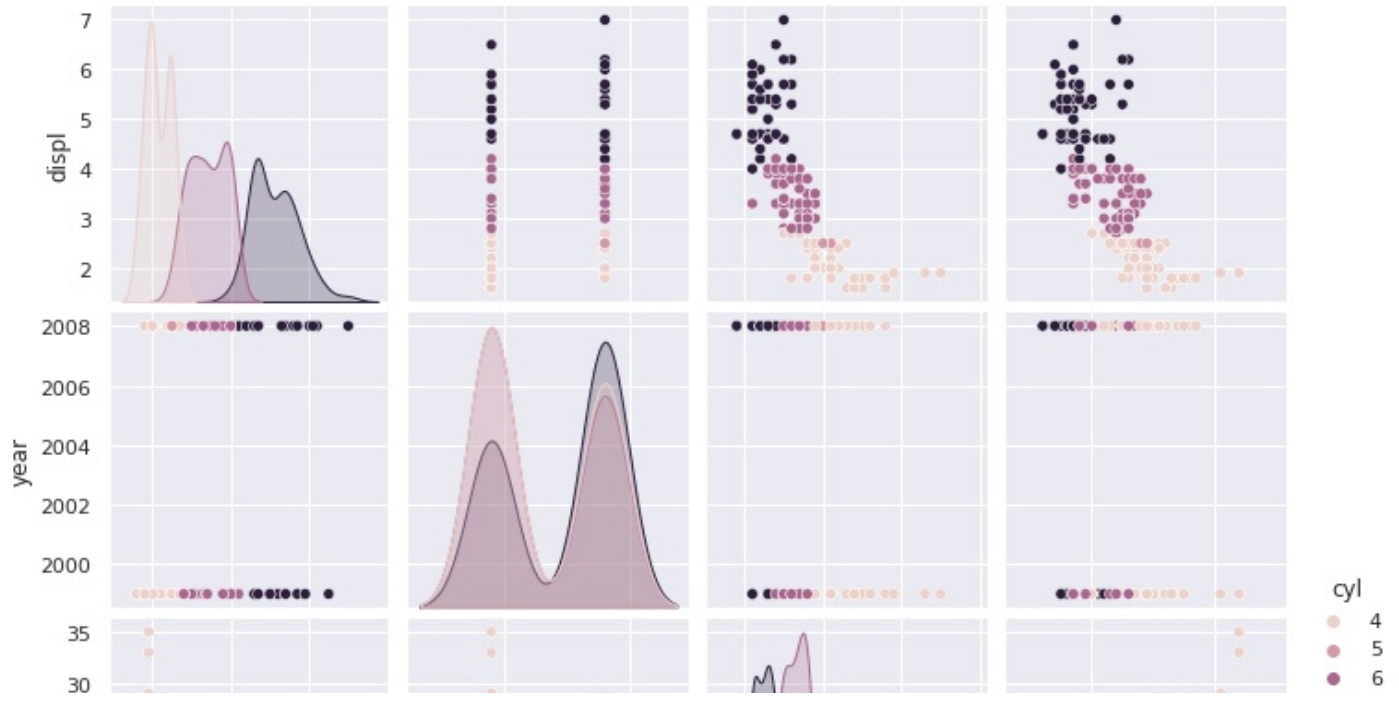
In [73]:

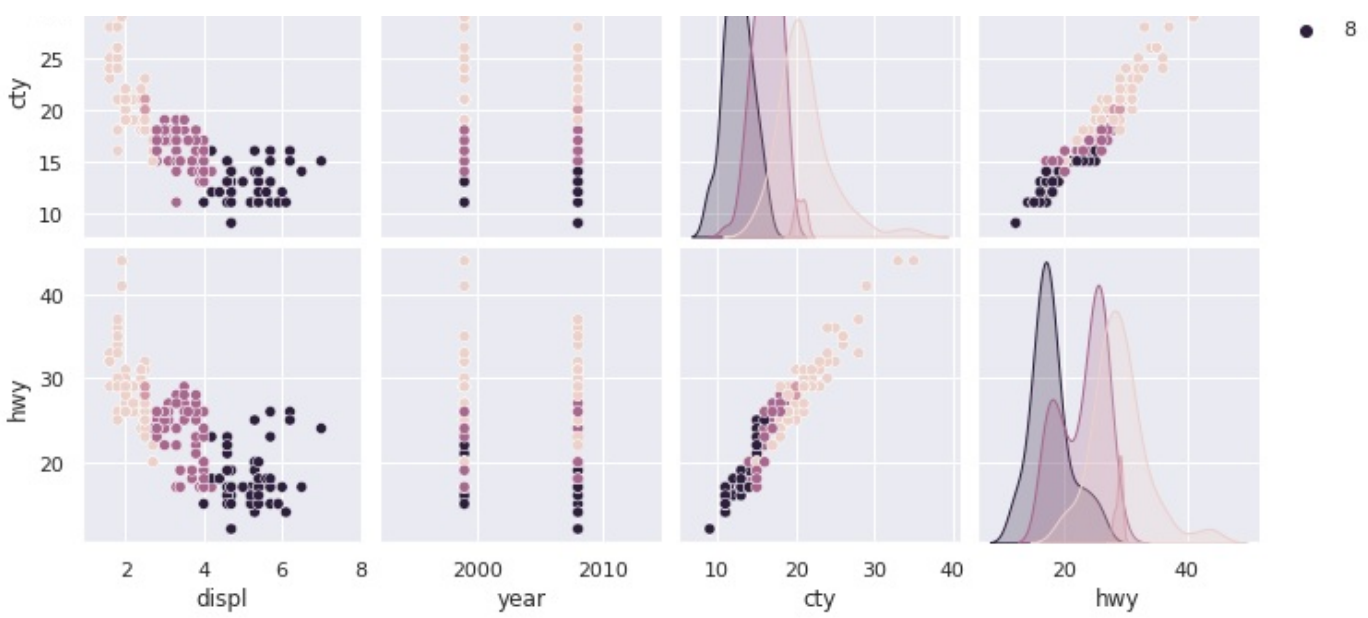
```
# ¿Cómo se correlacionan las variables numéricas? Muestra esta correlación en un gráfico.

# (No tomar en cuenta el año como variable numérica)
# Puedes quitar variables con el método df.drop(['col1', 'col2', ...], axis=1)
newData = auto_df.drop(['model', 'year', 'trans', 'drv', 'fl', 'class'], axis = 1)
newData
sns.set_palette("vlag")
sns.pairplot(data=auto_df, hue="cyl")
# Sugerencia: usa la paleta de colores 'vlag' o una paleta DIVERGENTE
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dataset
has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dataset
has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

Out[73]:

```
<seaborn.axisgrid.PairGrid at 0x7f64d0a5c250>
```





¿Cuales variables tienen una fuerte relación positiva entre sí y cuáles tienen una fuerte relación negativa? (Esta pregunta no es de código)