Programación con

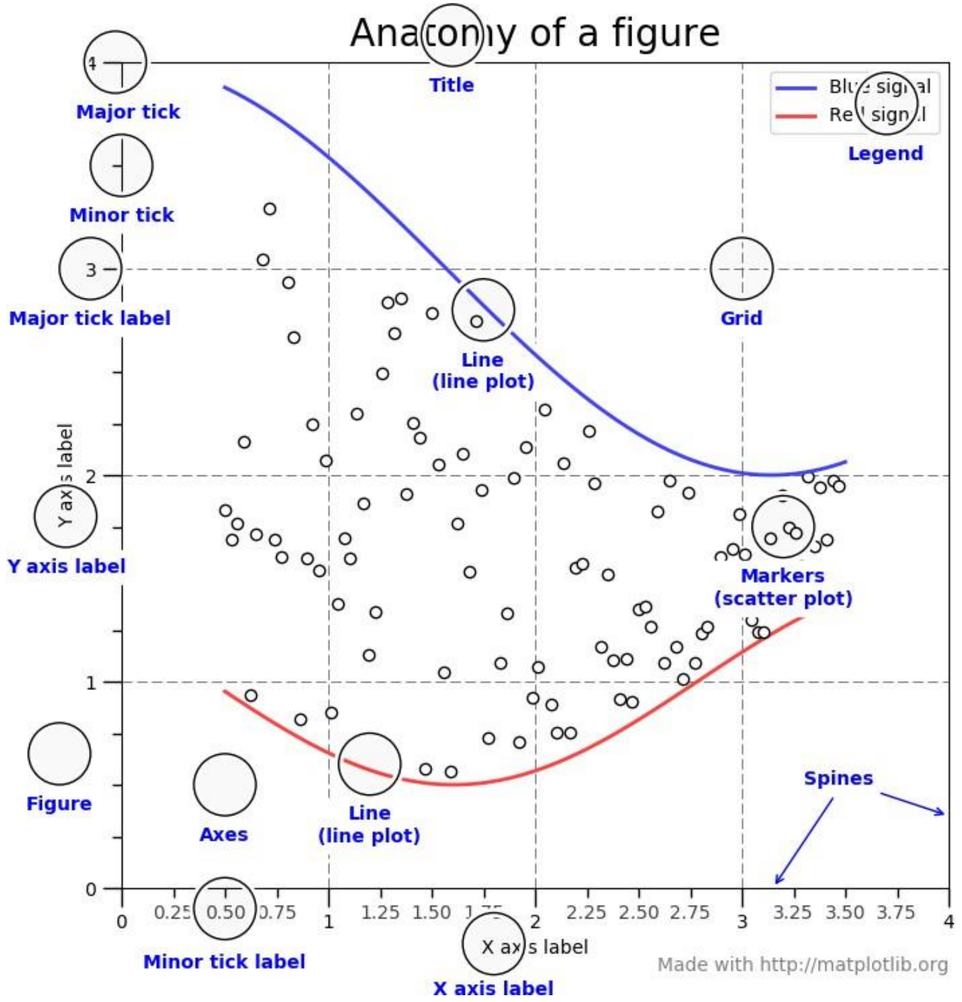
Gráficas en Python

Matplotlib

- Módulo de visualización por excelencia en Python. Estándar de facto para visualizar gráficas sencillas en 2D o 3D (no modelos 3D).
- Otros, más específicos y de más alto nivel, lo utilizan como base: Seaborn, vtkplotlib…
- Tres capas:
  - *Scripting* (`pyplot`). Trabaja a nivel de figuras, ejes, etiquetas, etc.
  - *Artist* (`artist`). Hay un objeto Artist por cada elemento visual. Determina qué hay que dibujar e invoca al backend para ello.
  - *Backend*. Interacciona con las librerías gráficas / interfaces de usuario:
    - `FigureCanvas` => Dónde se dibuja.
    - `Renderer` => Quién dibuja.
    - `Event` => Gestión de la interacción con el usuario (ratón, teclado…).

Matplotlib: partes de una figura

- 3 formas de usar Matplotlib:
  - Con pyplot:
    - La figura y los ejes (incluyendo su creación) son gestionados transparentemente por la librería.
    - Se usa en interactivo en IPython.
  - Estilo “orientado a objetos”:
    - Figura y ejes se crean y utilizan explícitamente como los objetos que son.
    - Se usa en programas más complejos.
  - Embebiendo las figuras en el GUI de una aplicación:
    - Más complejo, implica acceder al backend.



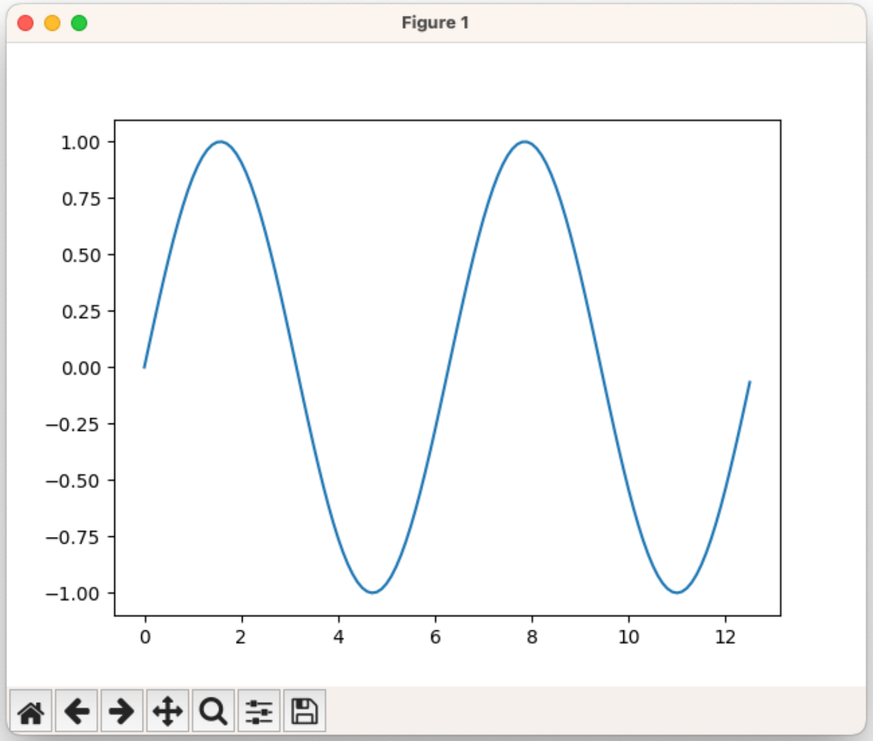
Matplotlib: ejemplos

- Importación del módulo:

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

- Gráfica 2D simple:

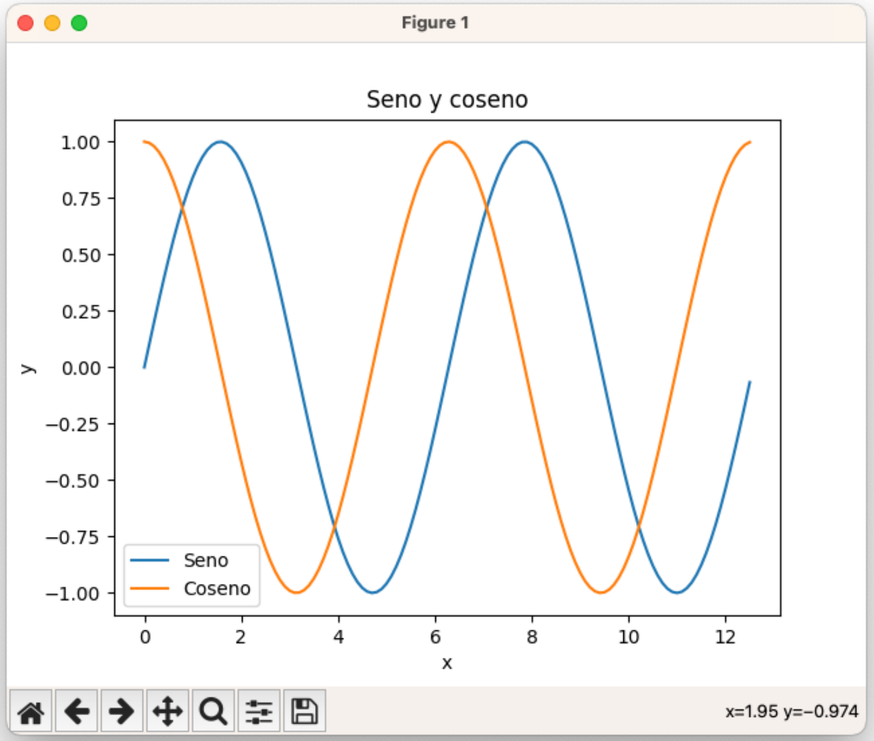
```
x = np.arange(0, 4 * np.pi, 0.1)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```



Matplotlib: ejemplos

- Dos funciones y con título, leyenda y etiquetas en los ejes:

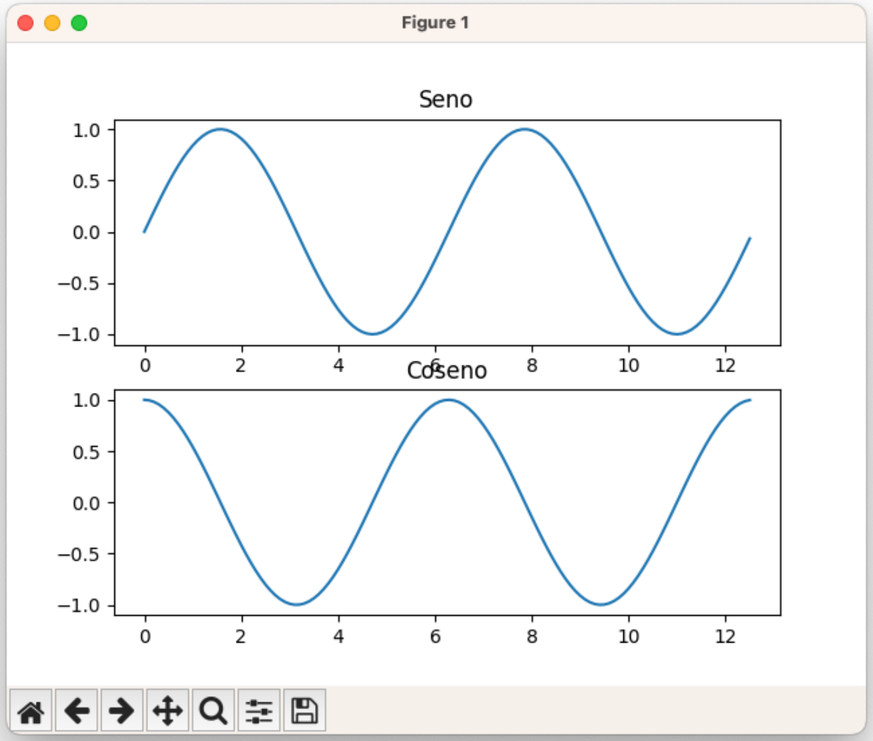
```
x = np.arange(0, 4 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Seno y coseno")
plt.legend(["Seno", "Coseno"])
plt.show()
```



Matplotlib: ejemplos

- Con subplots:

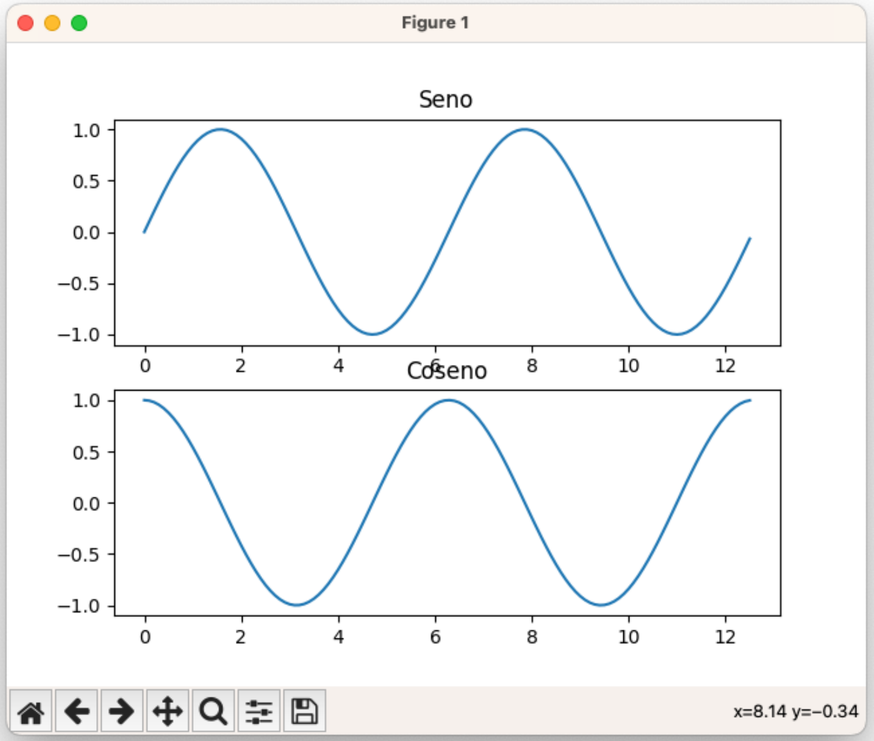
```
# Calculamos las funciones deseadas.
x = np.arange(0, 4 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
# Dividimos el área de nuestra figura en un grid 2x1
# y especificamos que vamos a pintar en el 1°.
plt.subplot(2, 1, 1)
# Primer plot con su título.
plt.plot(x, y_sin)
plt.title("Seno")
# Dividimos el área de nuestra figura en un grid 2x1
# y especificamos que vamos a pintar en el 2°.
plt.subplot(2, 1, 2)
# Segundo plot con su título.
plt.plot(x, y_cos)
plt.title("Coseno")
# Mostramos la figura.
plt.show()
```



Matplotlib: ejemplos

- Con subplots usando la interfaz orientada a objetos y grabando en un fichero a mayores:

```
# Calculamos las funciones deseadas.
x = np.arange(0, 4 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
# Crea una figura y sus ejes asociados.
fig, ax = plt.subplots(2, 1)
# Primer plot con su título.
ax[0].plot(x, y_sin)
ax[0].set_title("Seno")
# Segundo plot con su título.
ax[1].plot(x, y_cos)
ax[1].set_title("Coseno")
# Guardamos en un fichero.
plt.savefig("figura.png")
# Mostramos la figura.
plt.show()
```



Matplotlib: especificar backend

- Con una variable de entorno:
  - Útil para cambiar el backend sin tocar el programa.
  - Ej. en Linux / macOS (bash):

```
export MPLBACKEND=qt5agg
```

  - Ej. en Windows:

```
set MPLBACKEND=qt5agg
```

- En el propio programa. Ej.:

```
import matplotlib

# Consulta los backends disponibles
# (no todos tienen por qué ser válidos)
matplotlib.rcsetup.all_backends
# Consulta el backend en uso.
matplotlib.get_backend()
# Cambia el backend.
matplotlib.use("qt5agg")
```

Seaborn

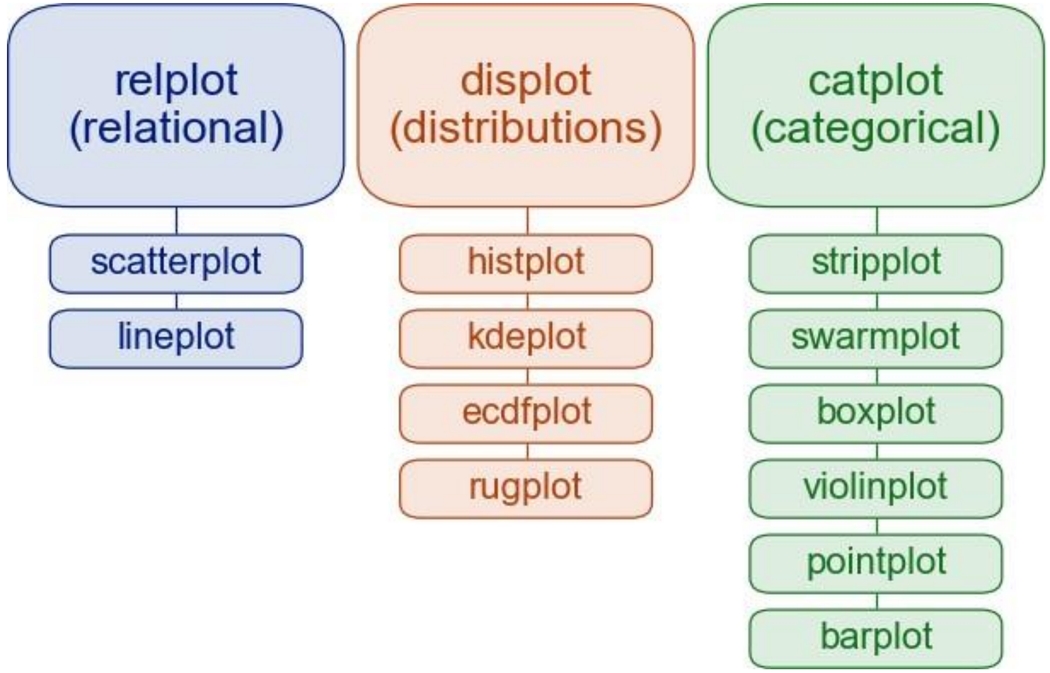
- Módulo de visualización basado en Matplotlib pero de más alto nivel:
  - Más sencillo de usar.
  - Figuras más atractivas por defecto.
  - Orientado a estadística.
  - Integración con Pandas (las funciones admiten como datos de entrada secuencias, diccionarios, arrays de NumPy o Dataframes de pandas).

Seaborn: características

- Espacio de nombres “plano” (todas las funciones están disponibles en el nivel superior).
- La mayoría de las funciones caen en estas categorías:
  - *Relational*. Gráficas para mostrar relaciones entre variables.
  - *Distributional*. Gráficas para mostrar distribuciones de probabilidad.
  - *Categorical*. Gráficas para mostrar la distribución de los datos en categorías.

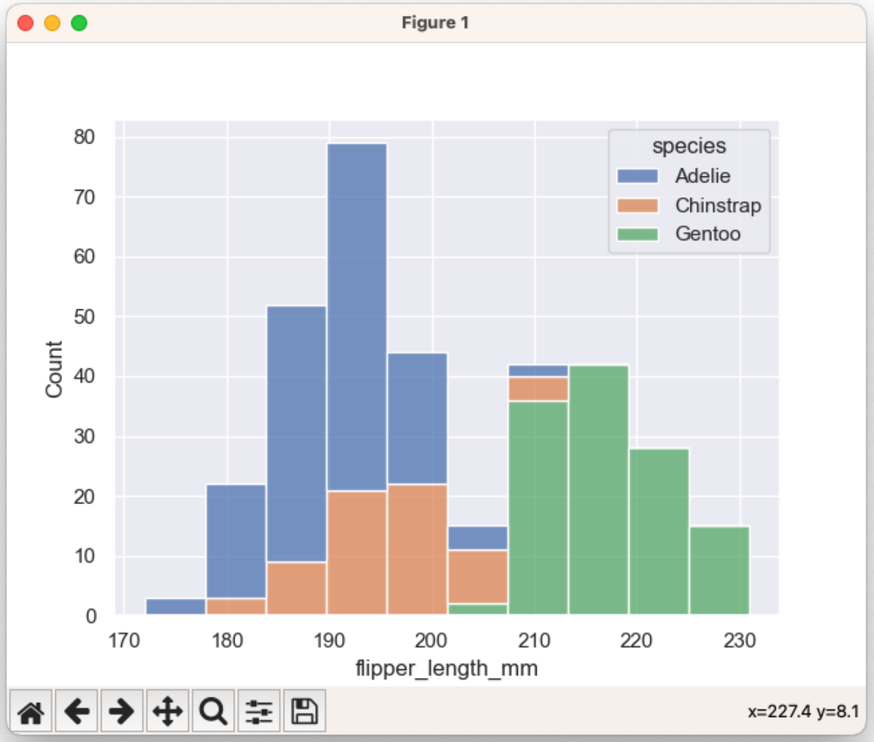
Seaborn: características

- Mismas funcionalidades disponibles en 2 niveles distintos:
  - Las funciones trabajan a nivel de ejes:
    - Sobre los ejes de la figura activa => similar Matplotlib interactivo.
    - Especificando los ejes => similar Matplotlib OO.
  - Las funciones trabajan a nivel de figura:
    - Se crea un objeto de Seaborn (ej: FacetGrid) que gestiona la figura.
    - Facilita algunas cosas como crear figuras con múltiples subplots del mismo tipo.
    - No vale si se quiere combinar gráficas de distinto tipo en una única figura.
    - La leyenda está fuera del área de los ejes.



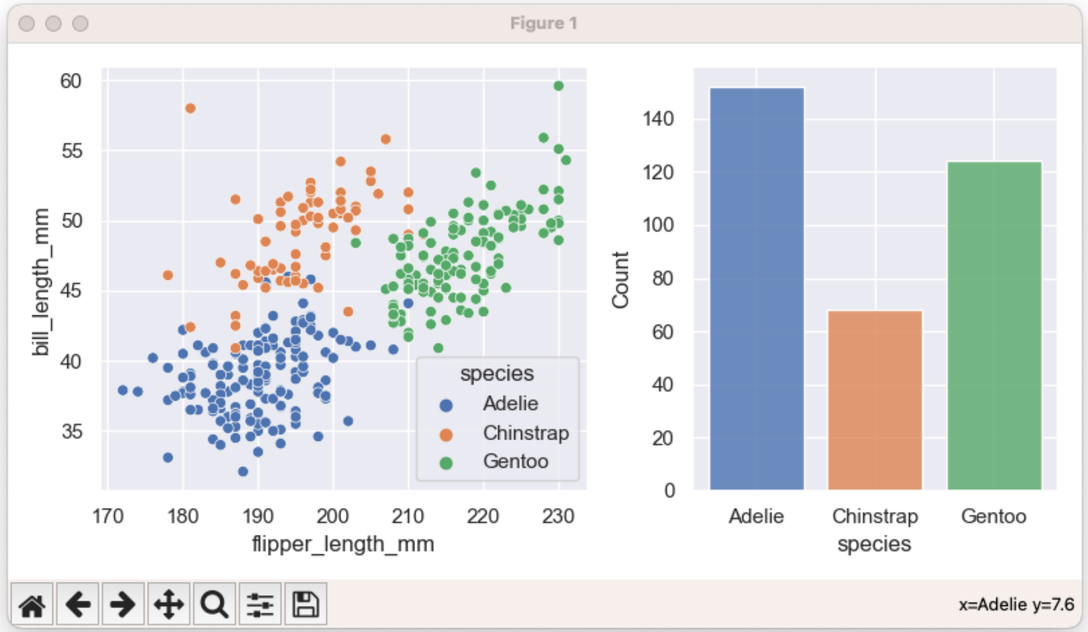
Seaborn: ejemplos a nivel de ejes

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

# Establecemos el tema para la visualización (valor por
# defecto de Seaborn).
sns.set_theme()

# Cargamos un dataset de ejemplo.
penguins = sns.load_dataset("penguins")

# Histograma.
sns.histplot(data=penguins, x="flipper_length_mm",
    hue="species", multiple="stack")
plt.show()
```



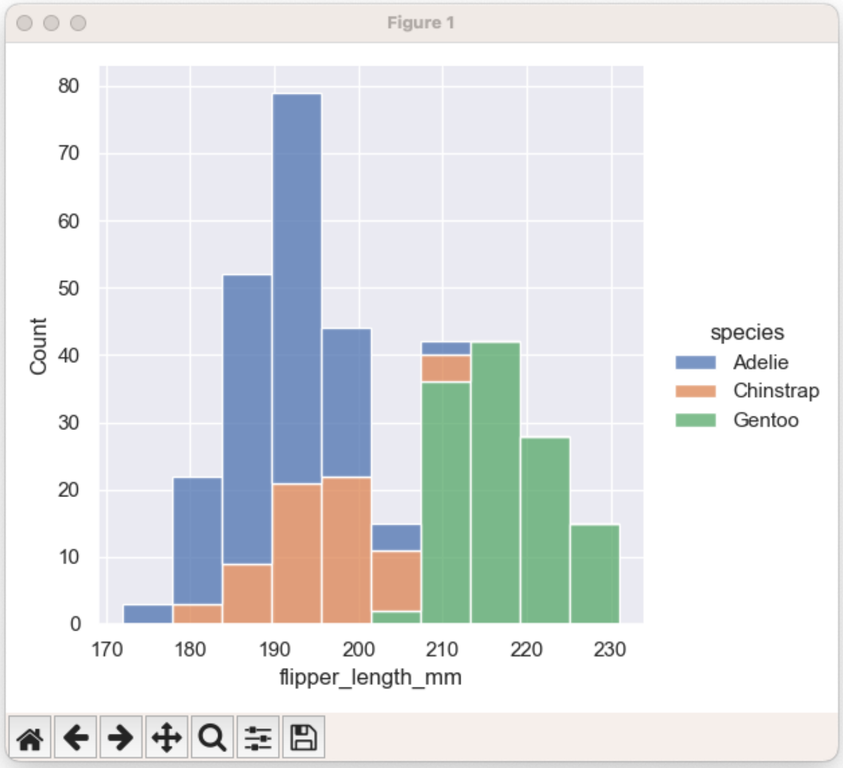
Seaborn: ejemplos a nivel de ejes

```
# Histograma y gráfica de dispersión en una única figura
usando la API OO de Matplotlib.
f, axs = plt.subplots(1, 2, figsize=(8, 4),
    gridspec_kw=dict(width_ratios=[4, 3]))
sns.scatterplot(
    data=penguins,
    x="flipper_length_mm",
    y="bill_length_mm",
    hue="species",
    ax=axs[0],
)
sns.histplot(
    data=penguins,
    x="species",
    hue="species",
    shrink=0.8,
    alpha=0.8,
    legend=False,
    ax=axs[1],
)
f.tight_layout()
plt.show()
```



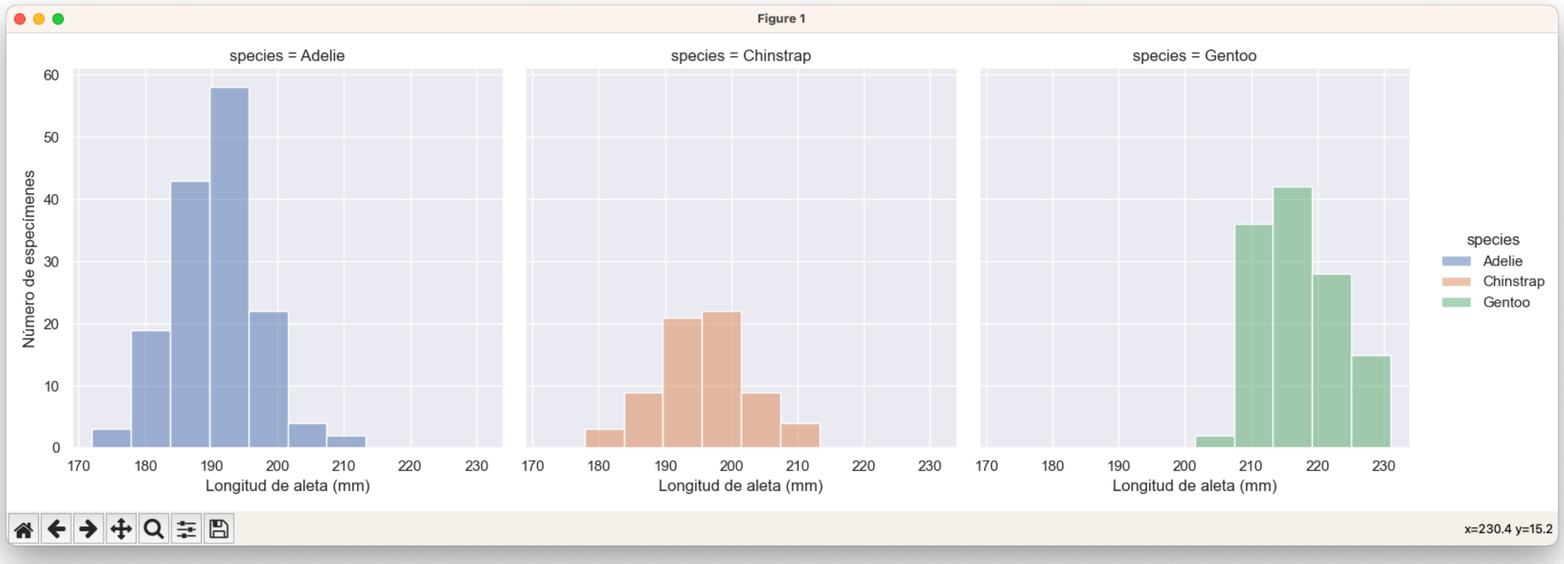
Seaborn: ejemplos a nivel de figura

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

# Establecemos el tema para la visualización (valor por
# defecto de Seaborn).
sns.set_theme()

# Cargamos un dataset de ejemplo.
penguins = sns.load_dataset("penguins")

# Histograma.
sns.displot(
    data=penguins,
    x="flipper_length_mm",
    hue="species",
    kind="hist",
    multiple="stack",
)
plt.show()
```



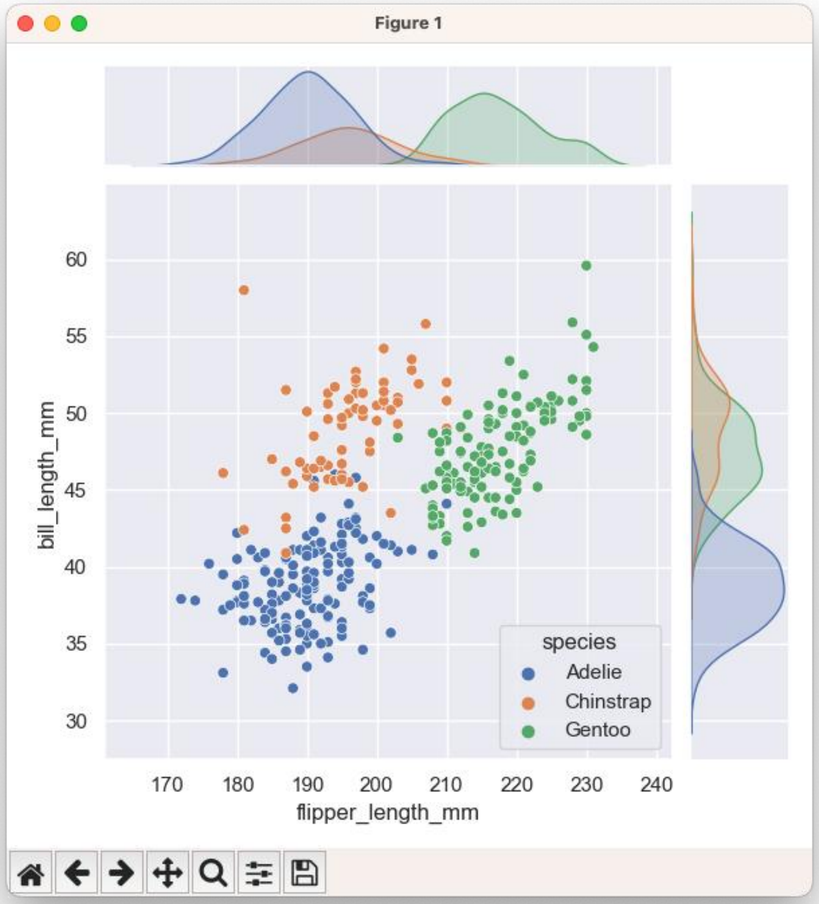
Seaborn: ejemplos a nivel de figura

```
# Histograma con una gráfica distinta para cada categoría
# y cambiando las etiquetas de los ejes.
grid = sns.displot(
    data=penguins,
    x="flipper_length_mm",
    hue="species",
    kind="hist",
    col="species",
)
grid.set_axis_labels(
    "Longitud de aleta (mm)",
    "Número de especímenes",
)
plt.show()
```



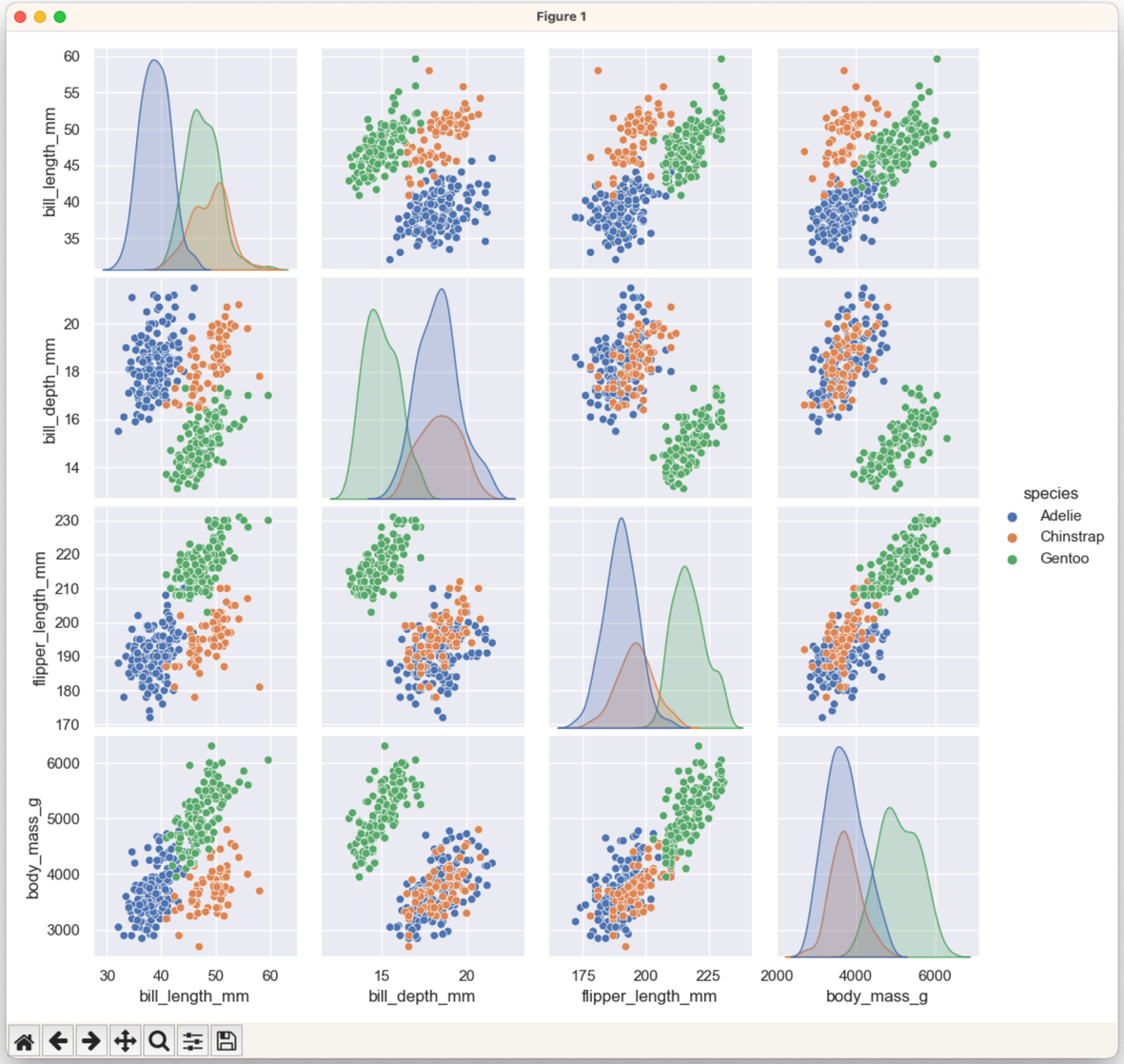
Seaborn: ejemplos a nivel de figura

```
# Distribución de dos variables, conjunta e independiente
para cada una de ellas.
sns.jointplot(
    data=penguins,
    x="flipper_length_mm",
    y="bill_length_mm",
    hue="species",
)
plt.show()
```



Seaborn: ejemplos a nivel de figura

```
# Distribución para cada combinación de 2 variables.
sns.pairplot(
    data=penguins,
    hue="species",
)
plt.show()
```



Referencias

- Matplotlib:
  - Documentación => https://matplotlib.org/stable/contents.html
  - Ejemplos pequeños sobre cómo hacer un tipo de gráfica concreto => https://matplotlib.org/stable/gallery/index.html
  - Tutoriales más complejos con explicaciones detalladas => https://matplotlib.org/stable/tutorials/index.html
  - Para empezar: https://matplotlib.org/stable/users/explain/quick_start.html
- Seaborn => https://seaborn.pydata.org/tutorial.html