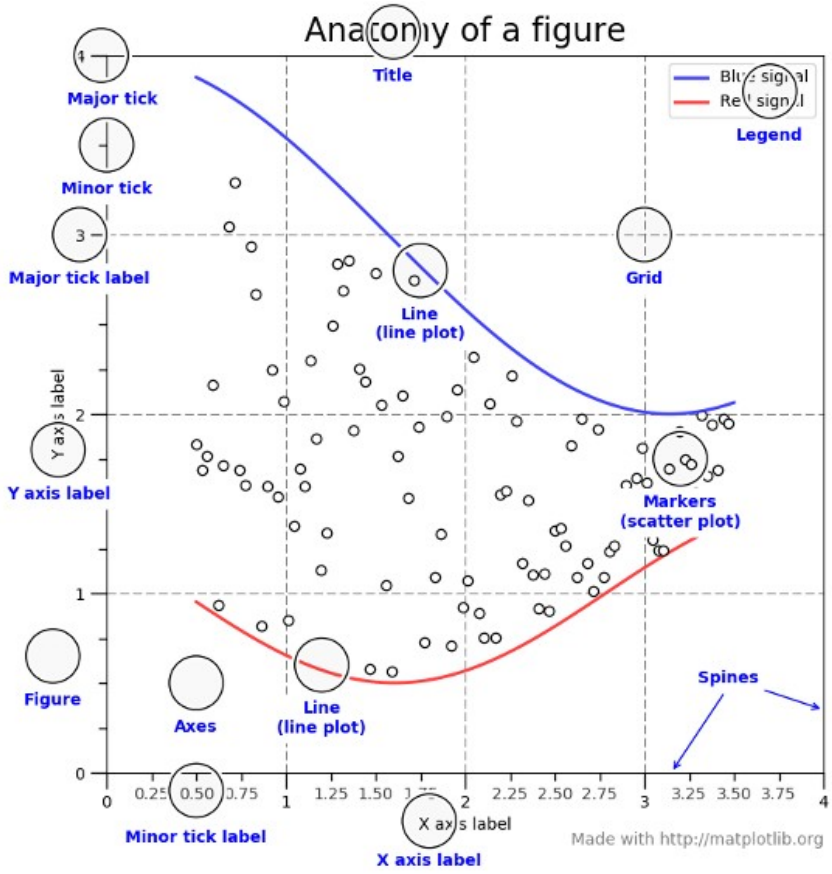
GRÁFICOS EN PYTHON

Para hacer gráficas hay varias bibliotecas:

- **Matplotlib**: Es la biblioteca más antigua pero la más utilizada.
- **Seaborn**: Es una capa por encima de Matplotlib y hace que sea más sencillo hacer gráficas
- **Yellowbrick**: También es una capa encima de Matplotlib pero es una librería especializa en gráficas para Machine Learning
- **Plotnine**: Esta librería sigue una filosofía totalmente distinta a las anteriores. La forma de especificar una gráfica se basa en los descrito en el libro The Grammar of Graphics que es una forma mas moderna que la usada por Matplotlib. Esta forma de especificar una gráfica se usa principalmente en el lenguaje R con su librería ggplot2. Por lo que Plotnine imita a ggplot2

MATPLOTLIB

Es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Es la base de los gráficos en Python. Conceptos (en inglés):



Instalación:

```
conda install matplotlib
```

Importación:

```
#Importing matplotlib with an Commonly used alias plt
import matplotlib as plt
```

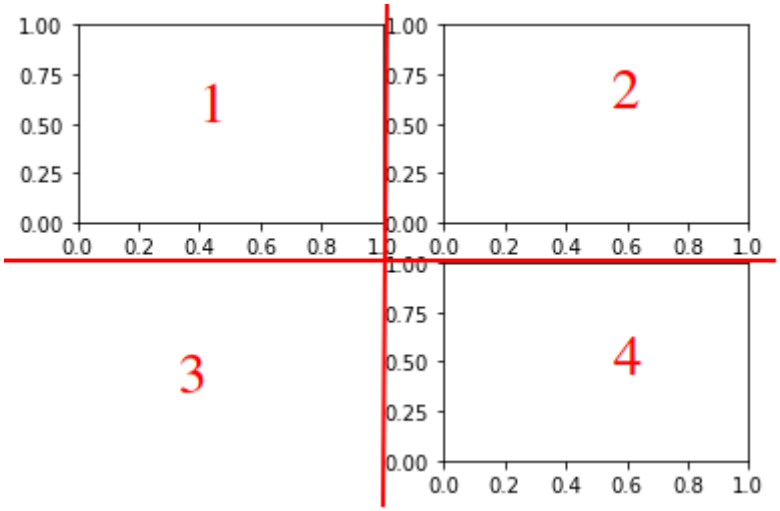
- ***figure***: Es como el "lugar" donde se van a colocar cada una de las gráficas. Siempre va a haber una figure. Un problema con figure es que en muchos ejemplos no se crea específicamente.
- ***axes***: Es como cada una de las **gráficas** que vamos a crea dentro de una figure.
- ***axis***: Son cada uno de los **ejes** de una gráfica.

OJO, NO CONFUNDIR **AXES** CON **AXIS**.

```
#IMPORTAMOS SÓLO LAS BIBLIOTECAS DE PYPLOT
import matplotlib.pyplot as plt

figure=plt.figure()
axes = figure.add_subplot()
```

El método ***add_subplot*** permite que le pasemos tres parámetros numéricos que indica: el número de filas, de columnas y la posición del gráfico. Coloca cada gráfica dentro de la figura.

```
import matplotlib.pyplot as plt

figure=plt.figure()
axes = figure.add_subplot(2,2,1)
axes2 = figure.add_subplot(2,2,2)
axes3 = figure.add_subplot(2,2,4)
```



Podemos mezclar organizaciones que no sean iguales: (rejilla)

```
import matplotlib.pyplot as plt

figure=plt.figure()
axes = figure.add_subplot(2,2,1)
axes = figure.add_subplot(2,2,3)
axes = figure.add_subplot(1,2,2)
```

Para hacer la figura mas grande solo hay que indicar el tamaño con el argumento `figsize`

El tamaño es el ancho y el alto en pulgadas.

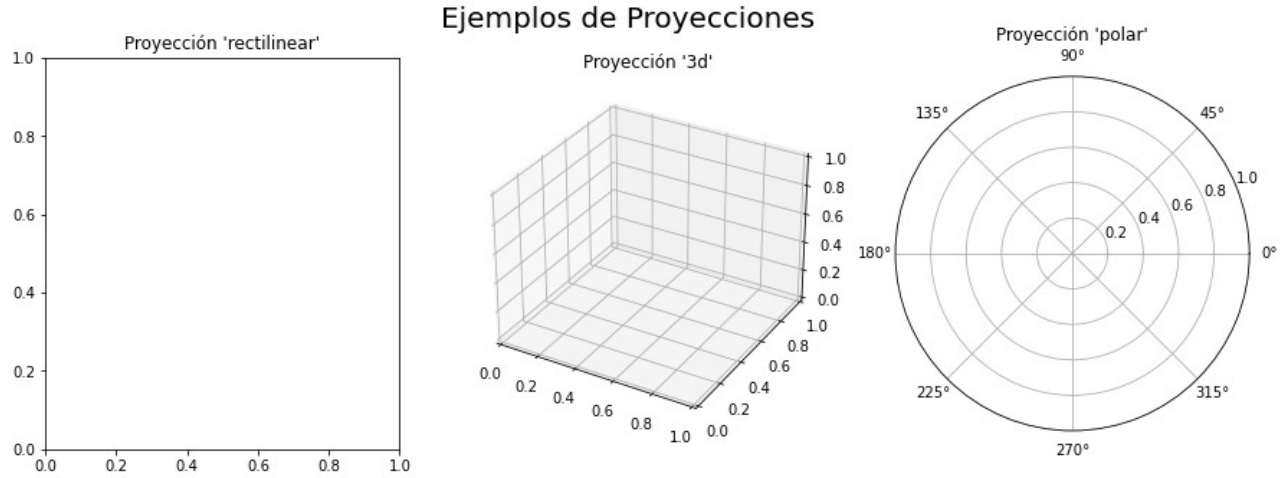
Para indicar el título , el color y el tamaño de letra se usa el método `suptitle`

Para grabar la figura entera con `savefig`

```
import matplotlib.pyplot as plt

figure=plt.figure(figsize=(15, 5))
axes1 = figure.add_subplot(1,2,1)
axes2 = figure.add_subplot(1,2,2)
figure.suptitle("Título de Figure", fontsize=14, color='red')
figure.savefig("fichero.png",facecolor="#FFFFFF", bbox_inches='tight')
# bbox_inches='tight' se usa para que no deje espacio alrededor de la imagen al guardarla.
```

La proyección de los ejes.

```
import matplotlib.pyplot as plt

figure=plt.figure(figsize=(15, 5))
figure.suptitle("Ejemplos de Proyecciones", fontsize=20)



axes1 = figure.add_subplot(1,3,1,projection='rectilinear')
axes1.set_title("Proyección 'rectilinear'")

axes2 = figure.add_subplot(1,3,2,projection='3d')
axes2.set_title("Proyección '3d'")

axes3 = figure.add_subplot(1,3,3,projection='polar')
axes3.set_title("Proyección 'polar'")
```



Dibujando en 2D
Métodos para dibujar en un axes o gráfica en 2 dimensiones:

- Para dibujar una serie de puntos se usa el método `scatter`
- Para dibujar una línea siguiendo una serie de puntos se usa el método `plot`
- Para dibujar una diagrama de barras en base a una serie de puntos se usa el método `bar`
- En un mismo axes se pueden dibujar varias cosas a la vez

Dibujando en 3D
Serie de puntos, `scatter`, la diferencia es que se pasa la `z` y la proyección es `3d`.
Una línea, `plot`, se pasa la `z` y la proyección es `3d`.

Si queremos interaccionar con el gráfico en 3D para poder rotarlo y así verlo mejor , debemos hacer lo siguiente:

Añadir lo siguiente antes de los imports

```
%matplotlib widget
```
o
```
%matplotlib ipympl
```

Debemos tener instalado **ipympl**

```
conda install -c conda-forge ipympl
```

Dibujando en 4D
Dibujar en 4D no es posible pero si lo que queremos es representar 2 variables en función de otras 2 variables , si que es posible mediante las siguientes técnicas:

- Variar el tamaño del punto en función de una variable
- Variar el color del punto en función de otra variable

Las 4 variables se mostrarían como:

- Eje X
- Eje Y
- Color
- Tamaño

Dibujando en 5D
La técnica anterior se puede aplicar a una gráfica en 3D con lo que conseguimos representar hasta 5 variables distintas.
Las 5 variables se mostrarían como:

- Eje X
- Eje Y
- Eje Z
- Color
- Tamaño

HISTOGRAMAS
Los histogramas consisten el mostrar la frecuencia con la que aparecen los valores en una secuencia unidimensional de datos. Podríamos pensar que son como diagramas de barras pero la información que muestran es de naturaleza distinta. En un diagrama de barras el origen de los datos es pares de números (x , y) , mientras que en un histograma sólo existe la x.

Para hacer histogramas no vamos a usar la librería `matplotlib` sino una mas avanzada que está sobre ella llamada `seaborn`. **`Seaborn`** permite hacer cosas como matplotlib pero de una forma más sencilla.

Es necesaior importar la librería de seaborn con:

```
import seaborn as sns
```

El método `histplot` para dibujar el histograma ya no se aplica sobre el axes sino sobre `sns` pero hay que pasarle el `axes` con `ax=axes`

```
sns.histplot(x=x,ax=axes)
```

Para dibujar el *Kernel density estimation* o KDE se usa el método `kdeplot` sobre el objeto `sns`

Si nos fijamos la escala del eje Y ya no es la frecuencia absoluta sino relativa
Si queremos mostrar tanto el histograma como el KDE se usa el método `histplot` pero sobre el objeto `sns` pero con el parámetro `kde=True`

Vemos que las curvas KDE de `histplot` y `kdeplot` no son iguales. Si dibujamos ambos en una misma `axes` e indicamos `stat="density"` en histplot vemos que son iguales. El motivo es que `histplot` puede mostrar los datos de distintas formas según el parámetro `stat` cuyas valores son `count`, `frequency`, `probability` o `density`.
seaborn.histplot

DIBUJANDO IMÁGENES
Se pueden dibujar imágenes, simplemente indicando los colores de cada pixel como un tensor de colores.

Para cargar una imagen se usa el método `imread`

Métodos y parámetros para PERSONALIZAR los `axes` o gráficos:

DATOS
Leyendas de los datos. En los métodos de `plot` o `scatter` hay que indicar el `label` y luego indicar que se muestre la leyenda con `legend`.

Colores de los datos. Sólo hay que indicar el argumento de `color`.

Al usar `plot` podemos hacer que junto a la línea también salgan los puntos o incluso solo los puntos. Para ello se añade un tercer parámetro con el estilo del *marker*. matplotlib.markers

Títulos
Establecer el título de cada eje `axis` y del propio gráfico `axes`.

No confundir el título del `axes` con el del `figure`.

Color del FONDO
Establecer el color del fondo con `set_facecolor`

EJES
Datos a mostrar en los ejes junto con el color , tamaño de fuente y el min/max
Guías interiores (`grid`)
Configurar las líneas de los ejes.

SUPERFICIES
Al dibujar una superficie en 3D, podemos cambiar los colores en función del valor de `z`. Para ello usamo el argumento `cmap`. Choosing Colormaps in Matplotlib
Podemos añadir una barra con los colores con el método `colorbar`. Notar que el método `colorbar` se usa sobre la figura y hay que pasarle el `axes`. Por último el parámetro `shrink` es para que la barra no salga tan alta.
Se pueden añadir sombreado a la imagen para que quede mas realista.