Tutorial de Matplotlib

José Luis Higuera Caraveo

Marzo 20 2022

1 Tutorial de Matplotlib

¿Alguna vez has pensado como serían los reportes sin visualizaciones para representar y entender los datos con los que se trabajan?

Leer artículos sería tedioso y poco amigable con los lectores. Las visualizaciones son muy importantes, en el mundo del Machine Learning, donde son pocas las personas que entienden como se están procesando los datos, es importante comunicarlos de una forma amigable, donde, cualquier persona, incluso no técnicos pueda entender que es lo que está pasando.

Las visualizaciones juegan un papel muy importante, con ellas podemos observar tendencias, incluso antes de aplicar cualquier algoritmo.

En este tutorial vamos aprender cómo realizar la siguiente visualización paso a paso, entendiendo cada línea de código.

Usaremos la librería de Matplotlib junto a su módulo `pyplot`.

De acuerdo a la documentación. matplotlib es una librería para crear visualizaciones estáticas y animadas con Python. Matplotlib hace que las cosas sean fáciles incluso si parece que estas son difíciles. Esta librería nos ayuda a:

- Crear visualizaciones de calidad.
- Hacer gráficas y figuras interactivas en las que puedes hacer zoom, desplazarte y actualizarlas.
- Visualizaciones personalizadas.
- Exportarlas a diferentes formatos.

Y mucho más, todo disponible en la Documentación.

1.1 Importando las librerías y los datos de ejemplo

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Para este tutorial no necesitaremos importar datasets, con la ayuda de `Numpy` generaremos los datos necesarios para el gráfico.

[3]:
```
# Creamos 300 números que van desde el rango -pi a pi
x = np.linspace(-np.pi, np.pi, 300, endpoint=True)
# Usaremos la función seno y coseno
seno, coseno = np.sin(x), np.cos(x)
```

1.2 Iniciando con el tutorial

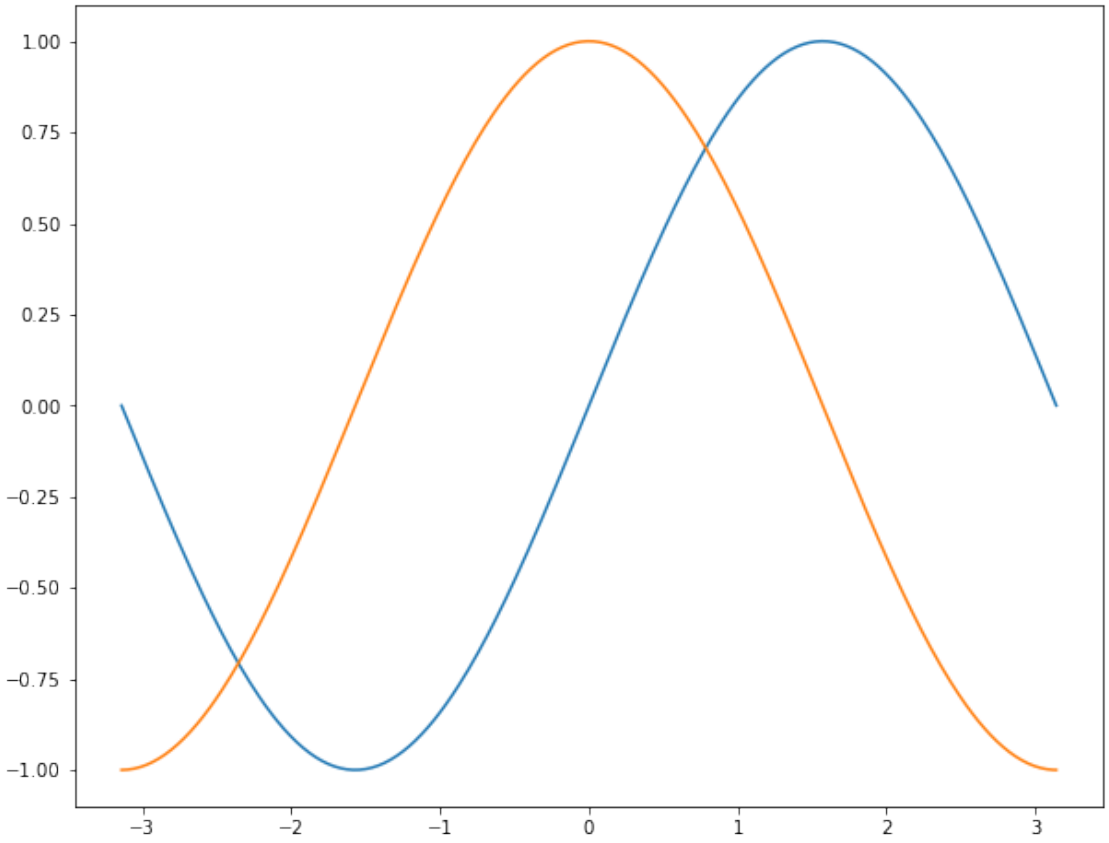
1.2.1 Entendiendo el método figure y plot

En matplotlib se trabaja con figuras, para iniciar con el tutorial, crearemos la que se va a necesitar para el proyecto. Si quieres leer más al respecto, te invito a que visites la documentación. matplotlib.pyplot.figure.

Así mismo, crearemos la gráfica de seno y coseno con los datos que se generaron usando el método `plot`. Para más información puede visitar matplotlib.pyplot.plot

> **Nota:** Usaremos el método `show` para evitar que matplotlib imprima una serie de información relacionada con el gráfico guardado en memoria.

[4]:
```
# Creando la figura
plt.figure(figsize=(10,8))
plt.plot(x, seno)
plt.plot(x, coseno)
plt.show()
```

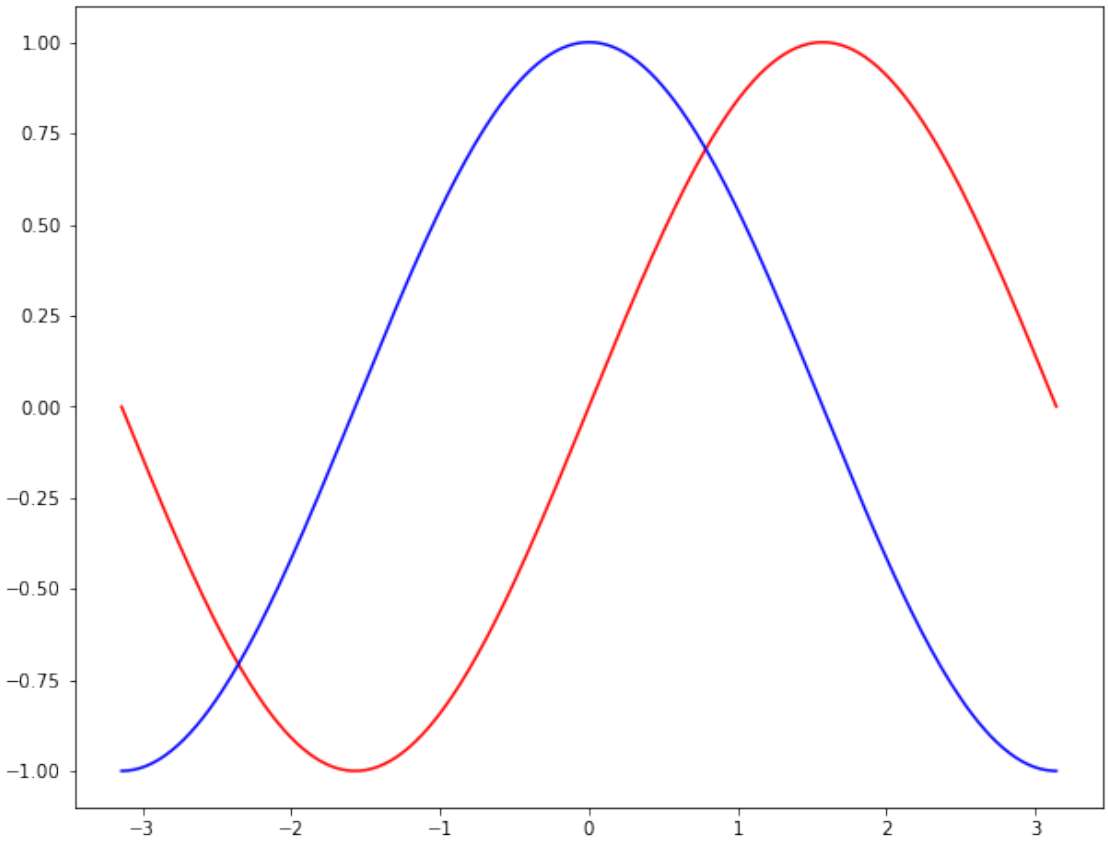


Iniciamos con unas gráficas que parecen simples, pero aquí es donde tu imaginación no tiene límites, podemos dar formato a los plot, para ello matplotlib nos ofrece una serie de parámetros que podemos configurar a nuestro gusto como por ejemplo:

- **Color:** Elegir el color de la gráfica.
- **Linewidth:** Elegir el grosor de la línea.
- **Linestyle:** Podemos escoger si usar línea continua, punteada, en guiones, entre otros disponibles Aquí
- **Label:** Asignarle un nombre o descripción a la línea.
- Para más parámetros, visitar la documentación.

Si observamos nuestro ejemplo, la línea seno es Roja, y la del coseno es Azul, además las líneas son más gruesas y tiene cada gráfico asignado una etiqueta. Agreguemos esto.

[5]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
plt.show()
```



1.2.2 Trabajando con los ejes de la figura.

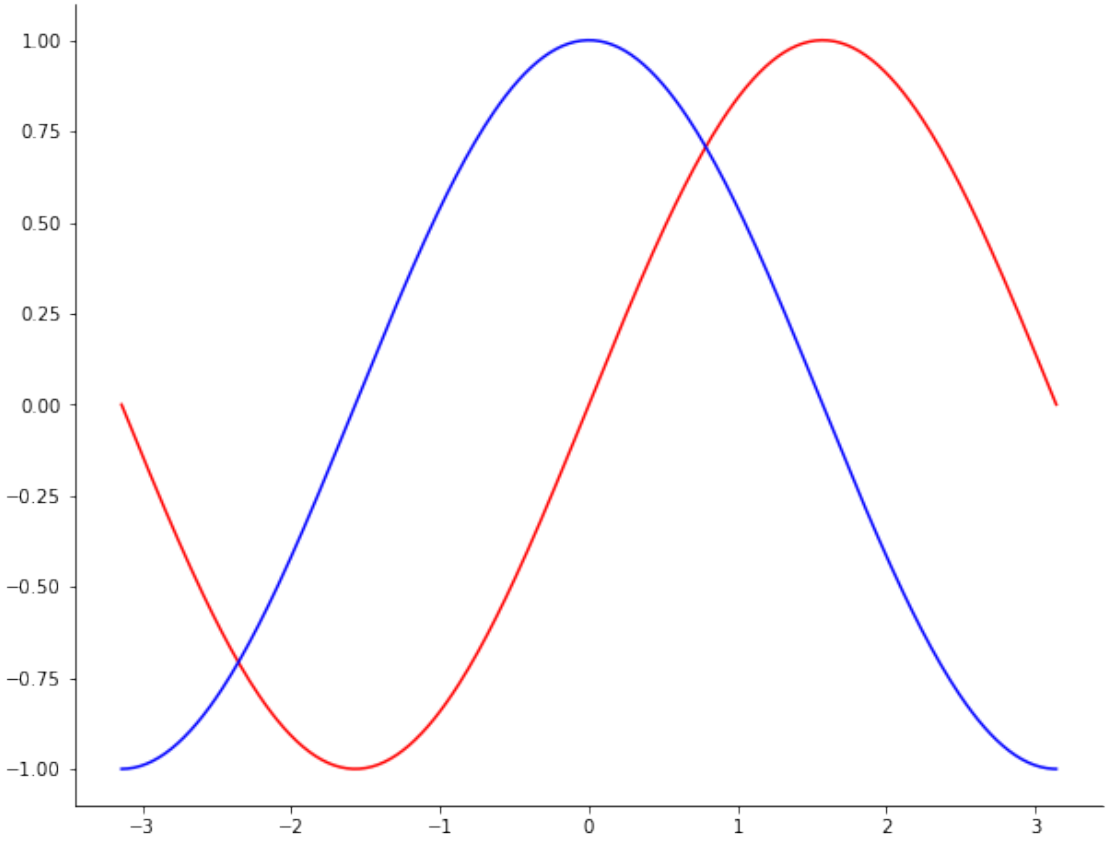
Observemos que el ejemplo no tiene ejes, tenemos que eliminarlos de nuestra gráfica. Para ello, podemos acceder a ellos con el método `gca` de `pyplot`.

Una figura está constituida por 4 ejes. Arriba, Abajo, Izquierdo y Derecho. En el ejemplo, el eje derecho y el de arriba no se utilizan, además el izquierdo y el de abajo los encontramos en una

posición diferente.

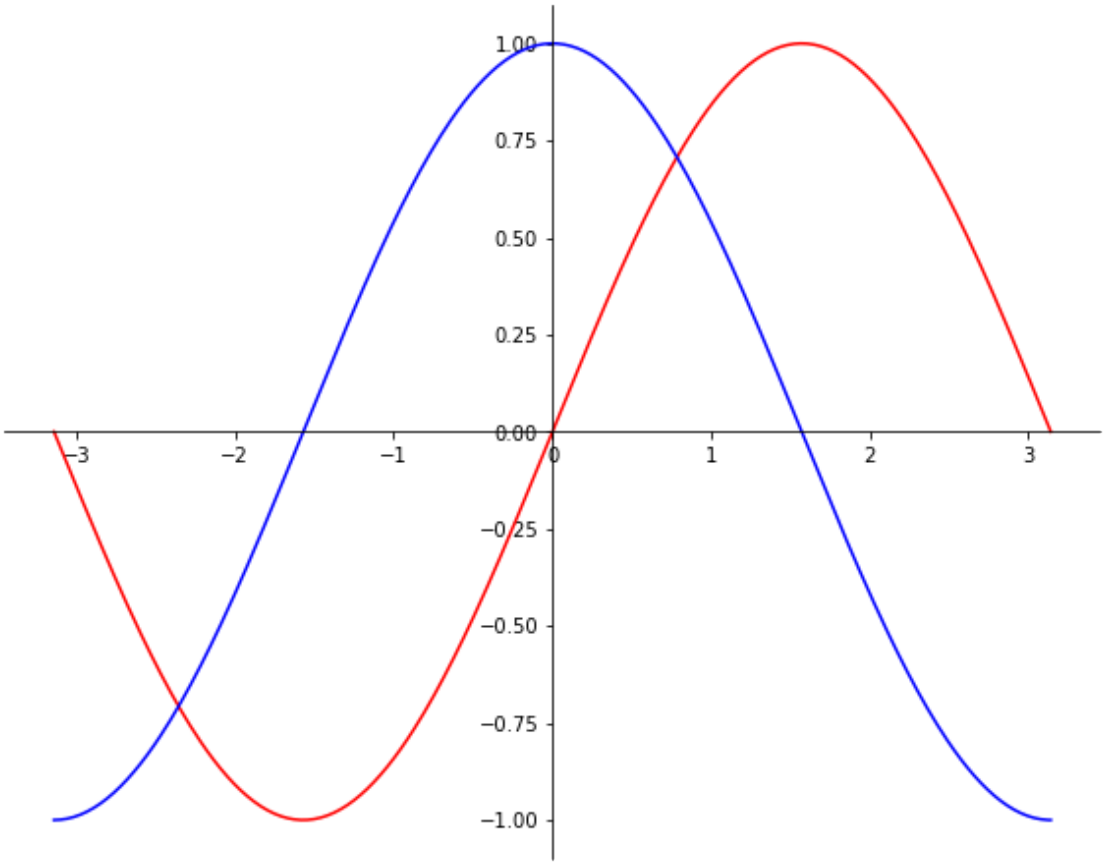
Procederemos a aplicar estos cambios con el método `spines` accediendo a los ejes de la figura con el método `gca` de `pyplot`. `Spines` nos proporciona una serie de parámetros que podemos agregar usando la función `set`, y la subfunción `set_color`, nos apoyaremos en esto para asignarles a los ejes que vamos a eliminar un `set_color` none.

[6]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')

axes = plt.gca()
# Eliminamos el eje de la derecha
axes.spines['right'].set_color('none')
# Eliminamos el eje de arriba
axes.spines['top'].set_color('none')
plt.show()
```



Hemos eliminado dos ejes, los otros dos si los vamos a necesitar, pero en diferente posición, para ello, con el mismo método `spines` y con el parámetro `set_position` lograremos este resultado. Encuentra más información Aquí

[7]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
axes = plt.gca()
axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')

# Posicionamos el eje de acuerdo a los datos y centrados en el cero
axes.spines['bottom'].set_position(('data', 0))
axes.spines['left'].set_position(('data', 0))
plt.show()
```

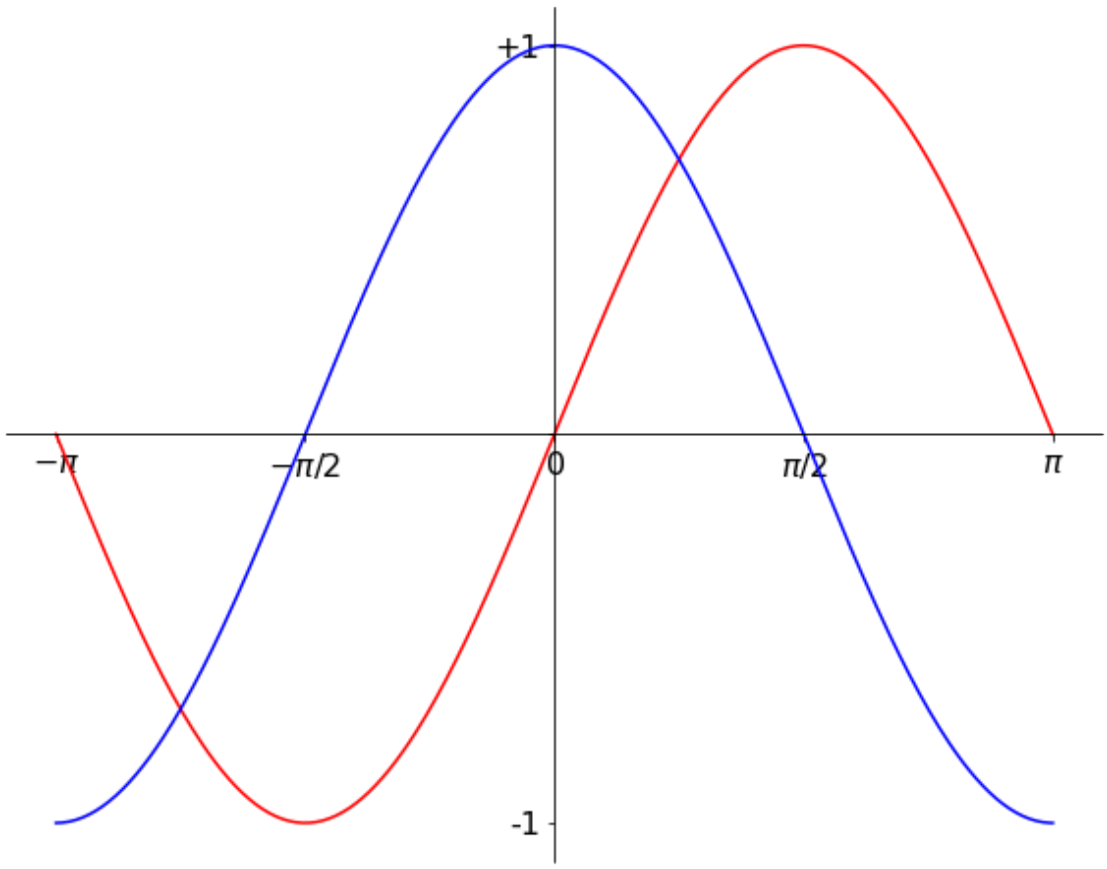


1.2.3 Cambiando los datos de los ejes x y y

Los datos de los ejes son diferentes a los que se muestran en el ejemplo, tenemos que mostrarlos igual. El método de `xticks` y `yticks` nos ayudará.

Además, gracias a un ciclo `for` podemos recorrer esos datos y darle el formato que queramos.

[8]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
axes = plt.gca()
axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')
axes.spines['bottom'].set_position(('data', 0))
axes.spines['left'].set_position(('data', 0))

# Agregamos la información que queremos que aparezca en los ejes
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$\pi/2$', r'$\pi$'])
plt.yticks(np.linspace(-1,1,3, endpoint=True),
           ['-1', '', '+1'])

for label in axes.get_xticklabels() + axes.get_yticklabels():
    label.set_fontsize(16)

plt.show()
```

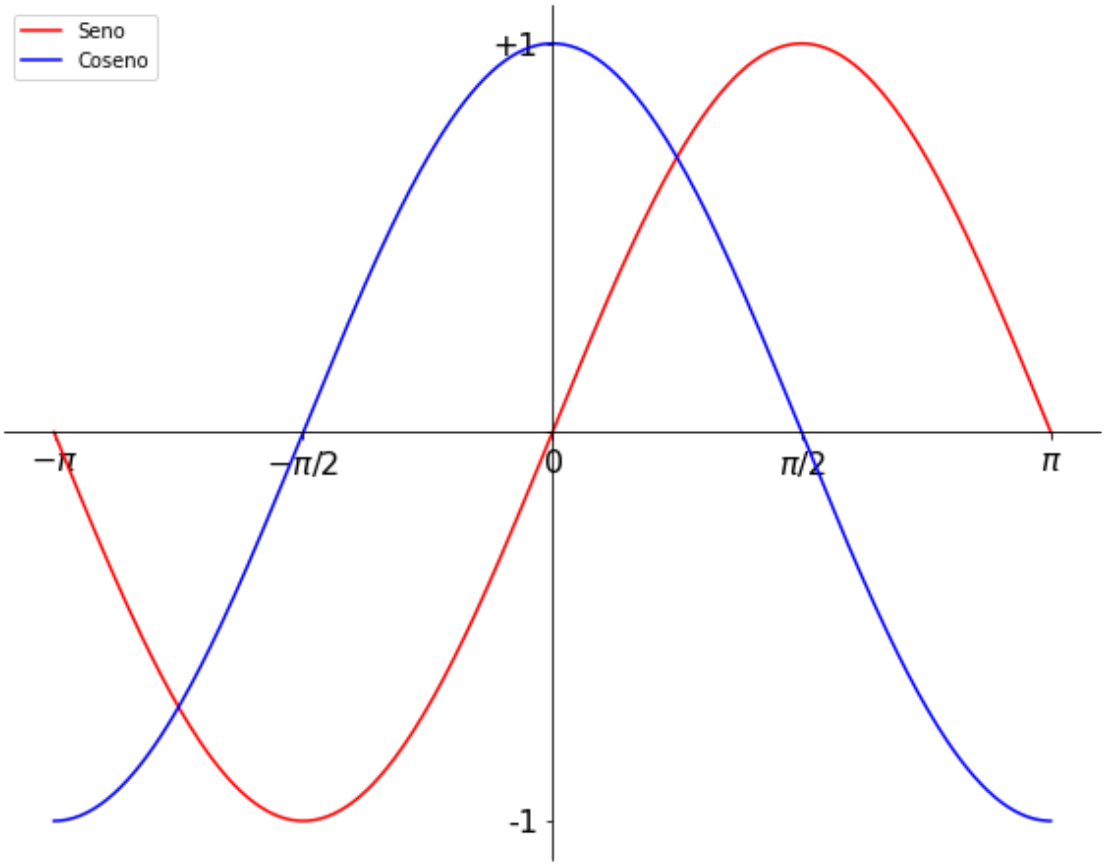


1.2.4 Ayuda visual, el método legend

Vemos que el gráfico de ejemplo nos proporciona una ayuda visual para poder identificar qué línea corresponde a que función.

Para agregar esta ayuda podemos usar el método `legend`, dándole una localización especificada por nosotros o simplemente dejando que matplotlib decida cuál es la mejor para el ejemplo.

[9]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
axes = plt.gca()
axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')
axes.spines['bottom'].set_position(('data', 0))
axes.spines['left'].set_position(('data', 0))
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$\pi/2$', r'$\pi$'])
plt.yticks(np.linspace(-1,1,3, endpoint=True),
           ['-1', '', '+1'])
for label in axes.get_xticklabels() + axes.get_yticklabels():
    label.set_fontsize(16)

# Agregando la ayuda visual
plt.legend(loc='upper left')

plt.show()
```



1.2.5 Multigráficos, Scatter y Plot juntos en una figura.

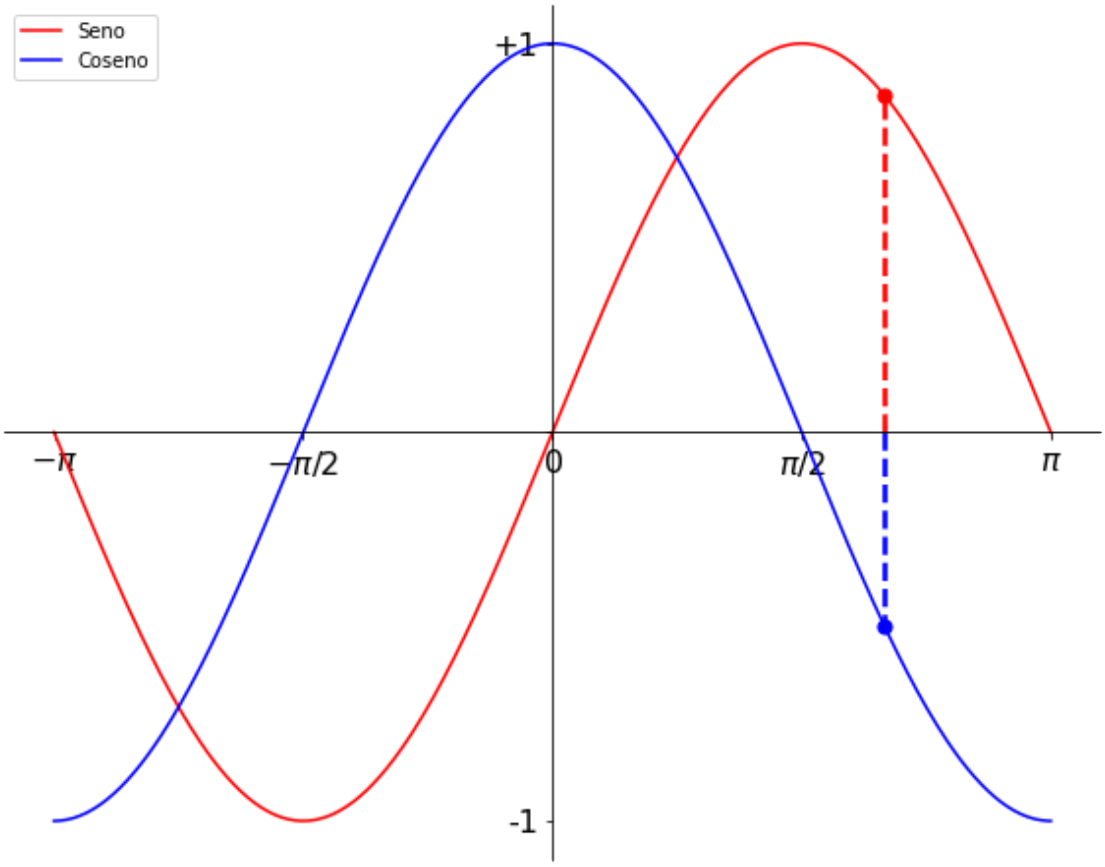
Agreguemos ahora el punto azul y rojo que aparece en la gráfica. Usaremos un tipo de gráfico llamado `scatter`. Los puntos están en $\frac{2}{3}$ de π en X, y en y el correspondiente valor de seno y coseno para ese $\frac{2}{3}$ de π.

Las líneas punteadas podemos agregarlas con el método `plot` usado anteriormente para pintar el gráfico de la función seno y coseno. Las coordenadas que usaremos son las mismas que se usarán para los puntos agregados con `scatter`. Tenemos que indicarle dos coordenadas, punto donde inicia la línea, punto donde termina. En nuestro ejemplo:

- Inicia en $(x = \frac{2\pi}{3}, y = 0)$.
- Termina en $(x = \frac{2\pi}{3}, y = sen/cos(x))$.

[11]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
axes = plt.gca()
axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')
axes.spines['bottom'].set_position(('data', 0))
```

```
axes.spines['left'].set_position(('data', 0))
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
          [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$\pi/2$', r'$\pi$'])
plt.yticks(np.linspace(-1,1,3, endpoint=True),
          ['-1', '', '+1'])
for label in axes.get_xticklabels() + axes.get_yticklabels():
    label.set_fontsize(16)
plt.legend(loc='upper left')

# Calculamos el valor de 2/3 de pi
x_0 = 2*np.pi/3
# Pintamos la línea punteada roja
plt.plot([x_0,x_0], [0, np.sin(x_0)], color='red', linewidth=2.5,␣
 ↪linestyle='--')
# Pintamos la línea punteada azul
plt.plot([x_0,x_0], [0, np.cos(x_0)], color='blue', linewidth=2.5,␣
 ↪linestyle='--')
# Pintamos el punto rojo de tamaño 50
plt.scatter(x_0, np.sin(x_0), 50, color='red')
# Pintamos el punto azul de tamaño 50
plt.scatter(x_0, np.cos(x_0), 50, color='blue')

plt.show()
```



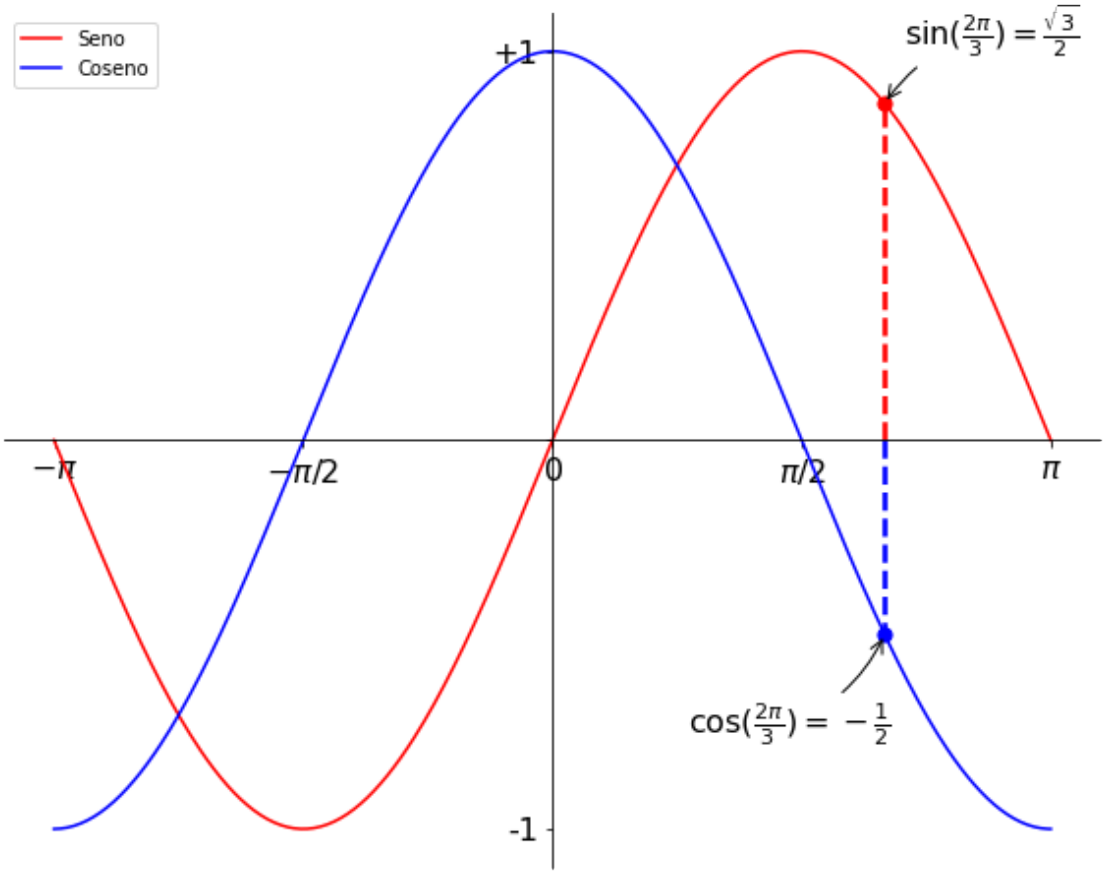
1.2.6 Agrega valor a tus visualizaciones colocando anotaciones.

Solo nos queda agregar unas anotaciones, para ello usamos el método `annotate`, con el cual, con la ayuda de coordenadas podemos indicar con una flecha un determinado punto.

- Primer parámetro: Anotación.
- Segundo parámetro: Coordenadas
- Resto de parámetros son configurables y de acuerdo al estilo que cada usuario quiera darle.

[15]:
```
plt.figure(figsize=(10,8))
plt.plot(x, seno, color='red', linewidth=1.5, linestyle='-', label='Seno')
plt.plot(x, coseno, color='blue', linewidth=1.5, linestyle='-', label='Coseno')
axes = plt.gca()
axes.spines['right'].set_color('none')
axes.spines['top'].set_color('none')
axes.spines['bottom'].set_position(('data', 0))
axes.spines['left'].set_position(('data', 0))
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           [r'$-\pi$', r'$-\pi/2$', r'$0$', r'$\pi/2$', r'$\pi$'])
plt.yticks(np.linspace(-1,1,3, endpoint=True),
```

```
          ['-1', '', '+1'])
for label in axes.get_xticklabels() + axes.get_yticklabels():
    label.set_fontsize(16)
plt.legend(loc='upper left')
x_0 = 2*np.pi/3
plt.plot([x_0,x_0], [0, np.sin(x_0)], color='red', linewidth=2.5,␣
 ↪linestyle='--')
plt.plot([x_0,x_0], [0, np.cos(x_0)], color='blue', linewidth=2.5,␣
 ↪linestyle='--')
plt.scatter(x_0, np.sin(x_0), 50, color='red')
plt.scatter(x_0, np.cos(x_0), 50, color='blue')

# Anotación para el punto de seno
plt.annotate(r'$\sin(\frac{2\pi}{3}) = \frac{\sqrt{3}}{2}$',
            xy = (x_0, np.sin(x_0)),
            xycoords = 'data',
            xytext = (+10,+30),
            textcoords= 'offset points',
            fontsize = 16,
            arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=.
 ↪2'))

# Anotación para el punto de coseno
plt.annotate(r'$\cos(\frac{2\pi}{3}) = -\frac{1}{2}$',
            xy = (x_0, np.cos(x_0)),
            xycoords = 'data',
            xytext = (-100,-50),
            textcoords= 'offset points',
            fontsize = 16,
            arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=.
 ↪2'))

plt.show()
```



Con esto hemos terminado nuestro tutorial, espero y hayas aprendido mucho acerca de cómo se manejan las visualizaciones con `matplotlib`.

Por último, un tip que te puede ser de gran utilidad, guardar tus visualizaciones, esto lo puedes hacer con una simple línea de código.

```
plt.savefig("Nombre_Archivo.extensión", bbox_inches='tight')
```

1.3 Sigue aprendiendo

Te invito y sigas investigando e indagando dentro de la documentación, `Matplotlib` es una herramienta muy poderosa y puedes sacarle mucho provecho para visualizar tus datos y darles sentido a tus investigaciones relacionadas a Ciencia de Datos.

Recuerda que, una visualización dice más que mil palabras y es el mejor medio por el cual puedes comunicar tus investigaciones y proyectos, esto para facilitar el entendimiento para técnicos y no técnicos.