prog_datasci_4_matplotlib

September 11, 2022

1 Fundamentos de Programación

1.1 Unidad 4: Librerías científicas en Python - Matplotlib

1.1.1 Instrucciones de uso

Este documento es un *notebook* interactivo que intercala explicaciones más bien teóricas de conceptos de programación con fragmentos de código ejecutables. Para aprovechar las ventajas que aporta este formato, se recomienda, en primer lugar, leer las explicaciones y el código que os proporcionamos. De esta manera tendréis un primer contacto con los conceptos que exponemos. Ahora bien, **¡la lectura es sólo el principio!** Una vez hayáis leído el contenido, no olvidéis ejecutar el código proporcionado y modificarlo para crear variantes que os permitan comprobar que habéis entendido su funcionalidad y explorar los detalles de implementación. Por último, se recomienda también consultar la documentación enlazada para explorar con más profundidad las funcionalidades de los módulos presentados.

Para guardar posibles modificaciones que hagáis sobre este notebook, os aconsejamos que montéis la unidad de Drive en Google Colaboratory (colab). Tenéis que ejecutar las instrucciones siguientes:

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
%cd /content/drive/MyDrive/Colab_Notebooks/prog_datasci_4
```

1.1.2 Introducción

A continuación se presentarán explicaciones y ejemplos de uso de la librería matplotlib. Recordad que podéis ir ejecutando los ejemplos para obtener los resultados.

A continuación se incluye la tabla de contenidos, que podéis utilizar para navegar por el documento:

2 1. Introducción

Matplotlib es una librería de representación de datos en 2D muy utilizada por la comunidad dada la gran calidad de las figuras generadas y por su capacidad de representación de datos de distinta índole. Un vistazo rápido a la galería de ejemplos nos convencerá de su gran potencia.

Esta librería puede ser utilizada mediante la interfaz Pyplot, que se integra muy bien con IPython y Notebook y tiene una sintaxis muy similar a MATLAB, o bien con los objetos que proporciona la librería para utilizar toda su capacidad de personalización de dibujo.

Esa similitud con MATLAB ha ayudado a su diseminación, debido a que muchos usuarios científicos buscaban una alternativa de código abierto frente a MATLAB y que pudiera ser integrada con otras librerías en Python.

El código de Matplotlib está dividido en tres partes: *pylab*, *matplotlib API* y *backends*. La primera parte, *pylab*, es la interfaz que permite crear gráficos con un código y funcionamiento muy similar a como se haría en Matlab. *Matplotlib API* es la parte esencial que el resto de código utiliza y, por último, *backends* es la parte encargada de la representación dependiente de la plataforma (tipos de ficheros de imagen, dispositivos de visualización, etc.). En este módulo solo cubriremos ejemplos y ejercicios utilizando *pylab*.

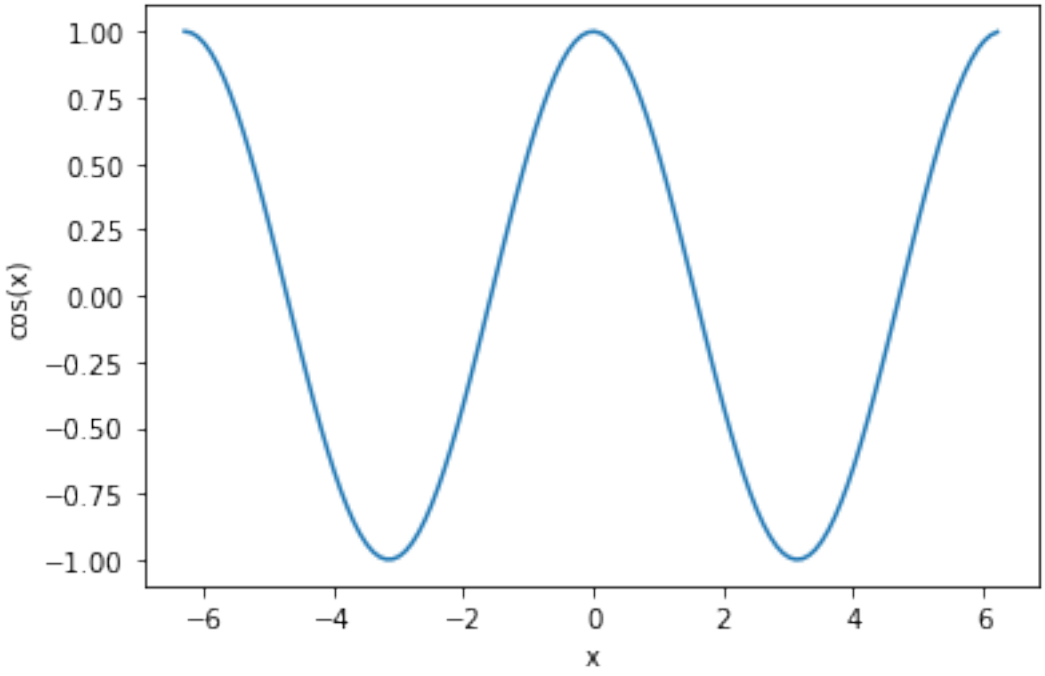
Podéis consultar muchos ejemplos en la ayuda de la librería.

3 2. Recopilación de ejemplos

3.1 2.1. Ejemplo 1: representar la función coseno

Vamos con el primer ejemplo en el que representaremos dos *arrays*, uno frente a otro, en los ejes x e y respectivamente. **Notad que para que los gráficos se muestren en el mismo Notebook debemos añadir la directiva especial:** *%matplotlib inline*.

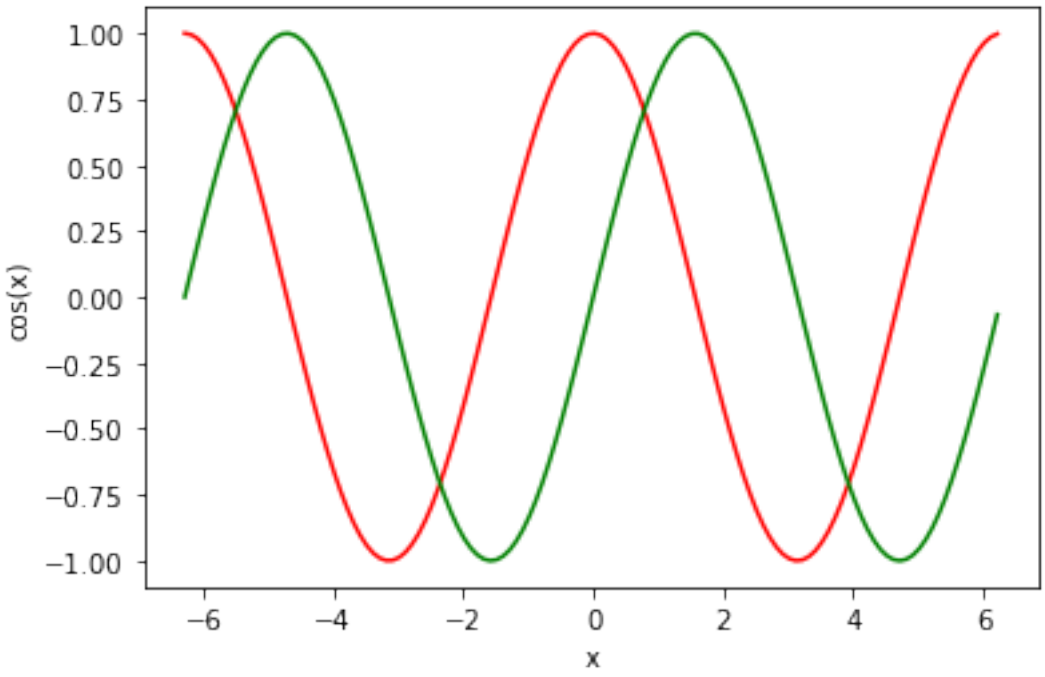
[1]:
```
%matplotlib inline

# Este primer import es necesario para inicializar el entorno de Matplotlib.
import matplotlib
import numpy as np
# Importamos la librería utilizando el alias 'plt'.
import matplotlib.pyplot as plt

# Calculamos un array de -2*PI a 2*PI con un paso de 0.1.
x = np.arange(-2*np.pi, 2*np.pi, 0.1)

# Representamos el array x frente al valor de cos(x).
plt.plot(x, np.cos(x))

# Añadimos los nombres a los ejes x e y respectivamente:
plt.xlabel('x')
plt.ylabel('cos(x)')

# Finalmente mostramos el gráfico.
plt.show()
```



3.2 2.2. Ejemplo 2: Representar las funciones coseno y seno a la vez

En este ejemplo, calcularemos los valores de las funciones seno y coseno para el mismo rango de valores y las representaremos en el mismo gráfico.

[2]:
```
%matplotlib inline

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

# Calculamos un array de -2*PI a 2*PI con un paso de 0.1.
x = np.arange(-2*np.pi, 2*np.pi, 0.1)

# Podemos encadenar la representación de múltiples funciones en el mismo gráfico.
# En este orden: array x, cos(x), 'r' hará utilizar el color rojo (red), array␣
 ↪x, sin(x) y verde (green).
plt.plot(x, np.cos(x), 'r', x, np.sin(x), 'g')

# De forma equivalente, podríamos llamar en dos ocasiones a la función plot:
# plt.plot(x, np.cos(x), 'r')
# plt.plot(x, np.sin(x), 'g')

# Añadimos los nombres a los ejes x e y respectivamente:
plt.xlabel('x')
plt.ylabel('cos(x)')

# Finalmente mostramos el gráfico.
plt.show()
```

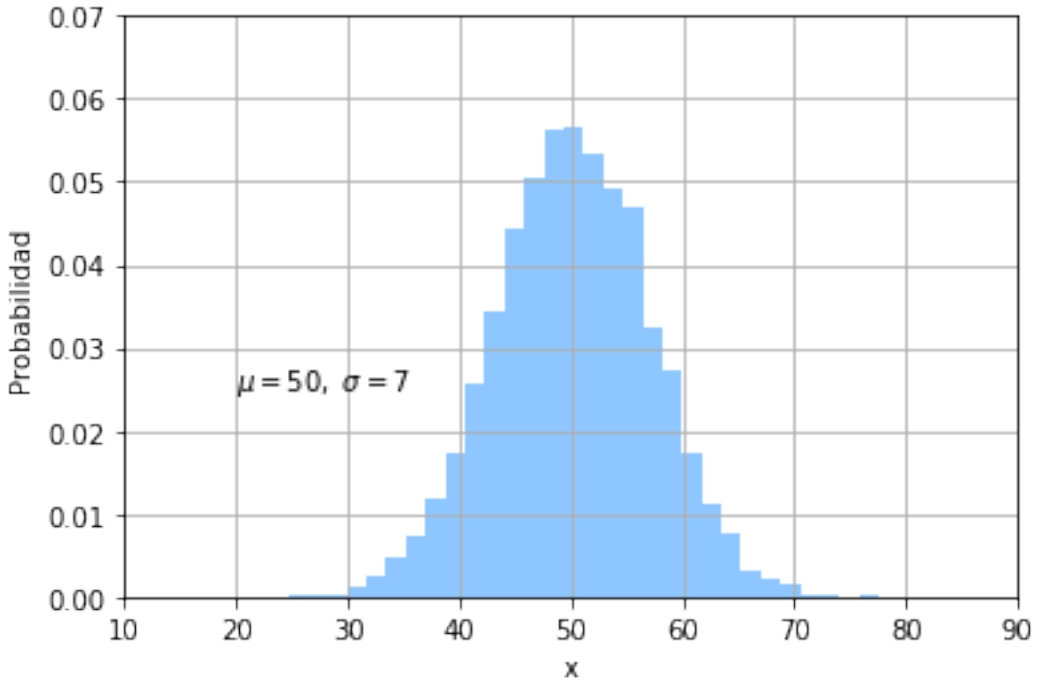


3.3 2.3. Ejemplo 3: Histogramas

Matplotlib dispone de muchos tipos de gráficos implementados, entre ellos los histogramas. En este ejemplo representamos una función gaussiana.

[1]:
```
%matplotlib inline

import matplotlib
import numpy as np
import matplotlib.pyplot as plt
#import matplotlib.mlab as mlab

# Parámetros de la función gaussiana
mu, sigma = 50, 7

# Generamos un array utilizando esos parámetros y números aleatorios.
x = mu + sigma * np.random.randn(10000)

# La función 'hist' nos calcula la frecuencia y el número de barras. El␣
 ↪argumento normed=1 normaliza los valores de
# probabilidad ([0,1]), facecolor controla el color del gráfico y alpha el valor␣
 ↪de la transparencia de las barras.
n, bins, patches = plt.hist(x, 30, density=1, facecolor='dodgerblue', alpha=0.5)
```

```
plt.xlabel('x')
plt.ylabel('Probabilidad')

# Situamos el texto con los valores de mu y sigma en el gráfico.
plt.text(20, .025, r'$\mu=50,\ \sigma=7$')

# Controlamos manualmente el tamaño de los ejes. Los dos primeros valores se␣
 ↪corresponden con xmin y xmax y los
# siguientes con ymin e ymax:
plt.axis([10, 90, 0, 0.07])

# Mostramos una rejilla.
plt.grid(True)

plt.show()
```



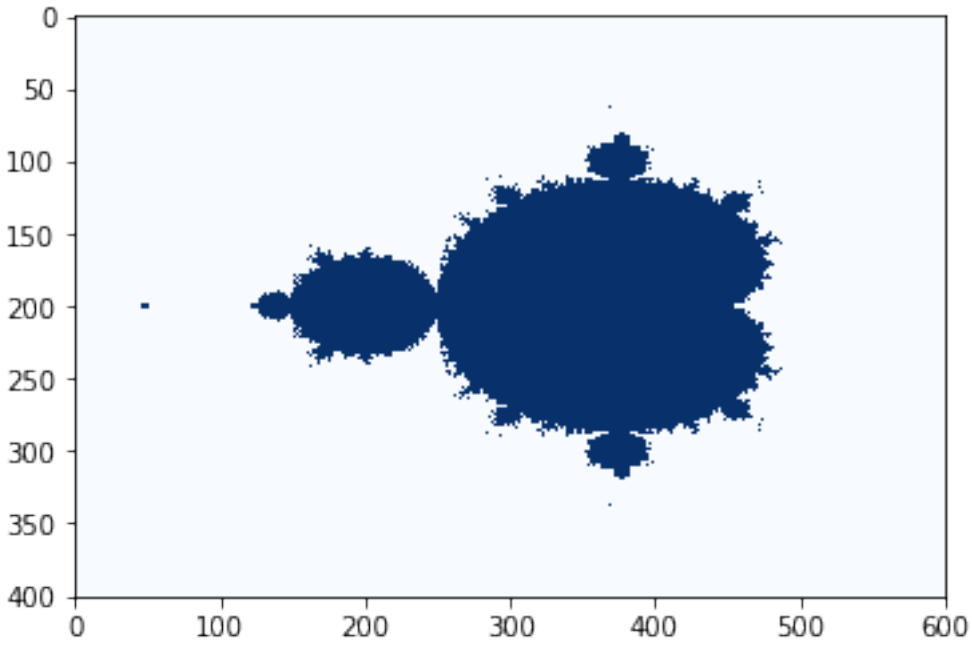
3.4 2.4. Ejemplo 4: Representación del conjunto de Mandelbrot

El conjunto de Mandelbrot es uno de los conjuntos fractales más estudiados y conocidos. Podéis encontrar más información en línea sobre el conjunto y los fractales en general.

El siguiente ejemplo es una adaptación de este código original.

[9]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from numpy import newaxis

# La función que calculará el conjunto de Mandelbrot.
def mandelbrot(N_max, threshold, nx, ny):
    # Creamos un array con nx elementos entre los valores -2 y 1.
    x = np.linspace(-2, 1, nx)
    # Lo mismo, pero en este caso entre -1.5 y 1.5, de ny elementos.
    y = np.linspace(-1.5, 1.5, ny)

    # Creamos el plano de números complejos necesario para calcular el conjunto.
    c = x[:,newaxis] + 1j*y[newaxis,:]

    # Iteración para calcular el valor de un elemento en la sucesión.
    z = c
    for j in range(N_max):
        z = z**2 + c

    # Finalmente, calculamos si un elemento pertenece o no al conjunto poniendo␣
 ↪un límite 'threshold'.
    conjunto = (abs(z) < threshold)

    return conjunto

conjunto_mandelbrot = mandelbrot(50, 50., 601, 401)

# Transponemos los ejes del conjunto de Mandelbrot calculado utilizando la␣
 ↪función de numpy 'T'
# Utilizamos la función imshow para representar una matriz como una imagen. El␣
 ↪argumento cmap significa
# 'color map' y es la escala de colores en la que representaremos nuestra imagen.
 ↪ Podéis encontrar muchos
# otros mapas en la documentación oficial: http://matplotlib.org/examples/color/
 ↪colormaps_reference.html
plt.imshow(conjunto_mandelbrot.T, cmap='Blues')

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: RuntimeWarning:
overflow encountered in square
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: RuntimeWarning:
invalid value encountered in square
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:23: RuntimeWarning:
invalid value encountered in less
```



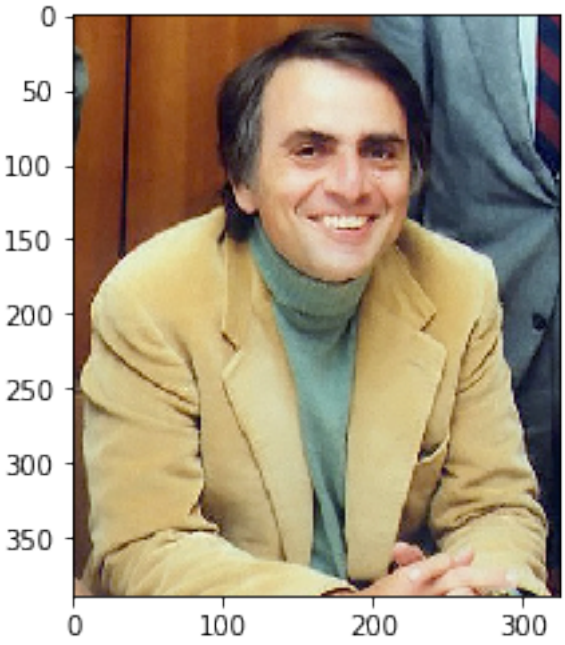
3.5 2.5. Ejemplo 5: Manipulación de imágenes

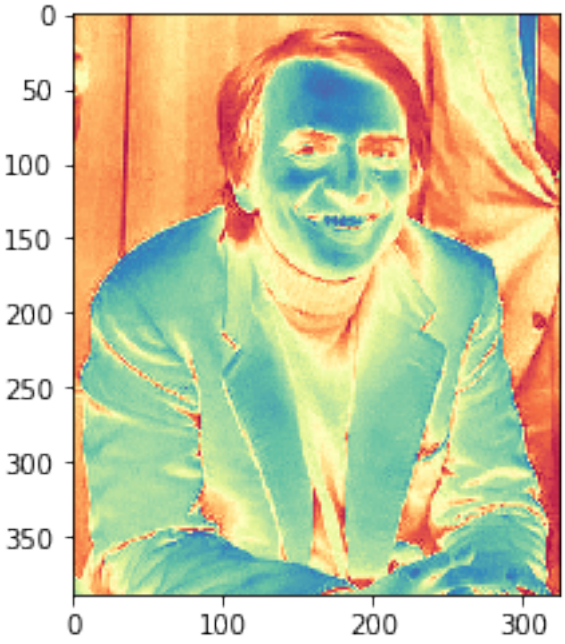
Una imagen puede asimilarse a una matriz multidimensional donde para valores de píxeles (x,y) tenemos calores de color. Matplotlib nos permite leer imágenes, manipularlas y aplicarles distintos mapas de colores a la hora de representarlas. En el siguiente ejemplo, cargaremos una fotografía en formato PNG de Carl Sagan.

Créditos de la foto: NASA JPL

[5]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Leemos la imagen mediante la función imread
carl = mpimg.imread('media/sagan.png')

# Podemos mostrar la imagen
plt.imshow(carl)
plt.show()

# Y podemos modificar los valores numéricos de color leídos por la función imread
# Obtenemos los valores de la escala de grises y mostramos los valores usando el␣
 ↪mapa de
```

```
# colores Spectral
grises = np.mean(carl, 2)
plt.imshow(grises, cmap='Spectral')
plt.show()
```





4 3. Ejercicios y preguntas teóricas

La parte evaluable de esta unidad consiste en la entrega de un fichero Notebook con extensión «.ipynb» que contendrá los diferentes ejercicios y las preguntas teóricas que hay que contestar. Encontraréis el archivo (`prog_datasci_4_scilib_entrega.ipynb`) con las actividades en la misma carpeta que este notebook que estáis leyendo. **Hay un solo archivo de actividades para toda la unidad, que cubre todas las librerías que se trabajan.**

4.1 3.1. Instrucciones importantes

Es muy importante que a la hora de entregar el fichero Notebook con vuestras actividades os aseguréis de que:

1. Vuestras soluciones sean originales. Esperamos no detectar copia directa entre estudiantes.
2. Todo el código esté correctamente documentado. El código sin documentar equivaldrá a un 0.
3. El fichero comprimido que entregáis es correcto (contiene las actividades de la PEC que tenéis que entregar).

Para hacer la entrega, tenéis que ir a la carpeta del drive `Colab Notebooks`, clicando con el botón derecho en la PEC en cuestión y haciendo `Download`. De este modo, os bajaréis la carpeta de la PEC comprimida en `zip`. Este es el archivo que tenéis que subir al campus virtual de la asignatura.
“‘

5 Autores

- Autor original Brian Jiménez Garcia, 2016.
- Actualizado por Cristina Pérez Solà, 2017 y 2019.

```
<div style="width:0%;"> </div>
    <img src="https://mirrors.creativecommons.org/presskit/buttons/88x31/png/by-sa.png">
```