David Gopar Morales.

Tarea No. 14

Entregar el 21 de junio de 2022

1. Hacer un resumen del uso de matplotlib que se usa en Python para construir representaciones gráficas. Pueden usar esta liga como referencia: https://realpython.com/python-matplotlib-guide/

2. En el programa del índice de la felicidad escribir un método para dibujar la serie de tiempo del índice utilizando matplotlib. El programa debe dibujar toda la serie que hayan podido descargar. En el eje horizontal deben aparecer fechas como etiquetas y en el vertical el valor del índice. Incluya textos explicativos de lo que se presenta en cada eje y un título.

1.

Tenemos que matplotlib sirve para crear gráficas a partir de cierta información que se tenga.
Para importar esta biblioteca basta con agregar lo siguiente a nuestro archivo de python:

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

Ahora, para ‘inicializar’ una figura se teclea algo como:

```
>>> fig, name = plt.subplots(figsize=(2, 3))
```

Donde ‘name’, es el nombre que se le asigna al dato de tipo (type) figura, y figsize=(,) es la escala que tendrá esta figura.
Para describir cómo editar la figura utilizaremos ejemplos que vienen en la página de referencia y describiremos que hace cada cosa.

Ejemplo código 1:

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

>>> rng = np.arange(50)
>>> rnd = np.random.randint(0, 10, size=(3, rng.size))
>>> yrs = 1950 + rng

>>> fig, ax = plt.subplots(figsize=(5, 3))
>>> ax.stackplot(yrs, rng + rnd, labels=['Eastasia', 'Eurasia', 'Oceania'])
>>> ax.set_title('Combined debt growth over time')
>>> ax.legend(loc='upper left')
>>> ax.set_ylabel('Total debt')
>>> ax.set_xlim(xmin=yrs[0], xmax=yrs[-1])
>>> fig.tight_layout()
>>> plt.show()
```

Como vemos primero se importa la biblioteca para crear la gráfica, luego se importar numpy simplemente como herramienta para crear la información a meter en la figura.
El siguiente párrafo se crea la información, que es como si tuviésemos tres tablas diferentes y queremos ver como se relacionan entre si con la figura.

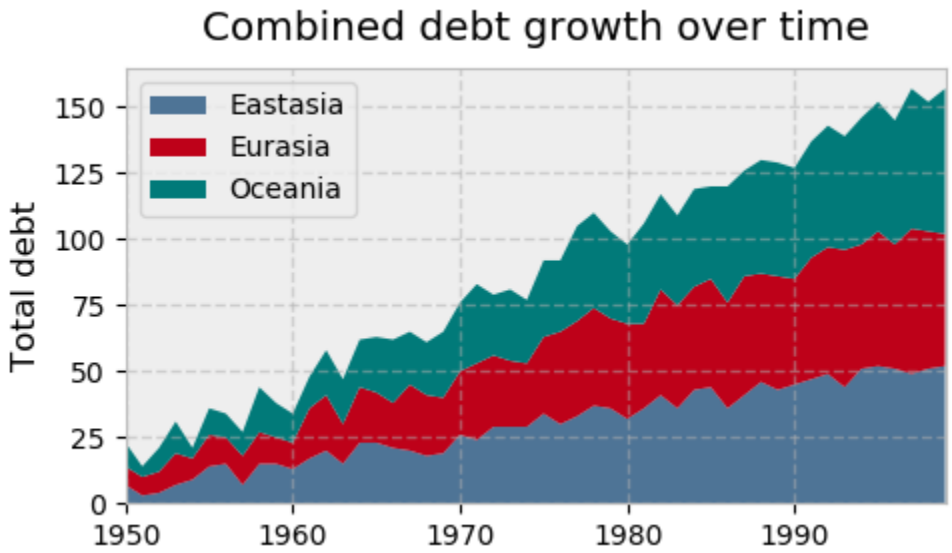
Pasamos al tercer párrafo. Notemos que primero le asignamos el nombre ax a la figura y posteriormente las dimensiones de esta.
En el siguiente renglón se indica qué tipo de gráfica crearemos, en este caso es ‘stackplot’, cada tipo de gráfica tiene sus atributos en este caso es claro que son primero los del eje x, luego y, y luego el nombre de cada relación que vamos a tener:

```
ax.stackplot(yrs, rng + rnd, labels=['Eastasia', 'Eurasia', 'Oceania'])
```

Lo siguiente es sencillo:

- ax.set_title('Combined debt growth over time'), indica el título de la figura.
- ax.legend(loc='upper left'), indica dónde se pondrán los labels que se pusieron en ax.stackplot(…).
- ax.set_ylabel('Total debt'), leyenda del eje y
- ax.set_xlim(xmin=yrs[0], xmax=yrs[-1]), de dónde a dónde queremos mostrar los datos, que año a qué año, en el eje x
- fig.tight_layout(), se aplica al objeto Figure como un todo para limpiar el relleno de espacios en blanco.

Y así obtenemos la gráfica siguiente:

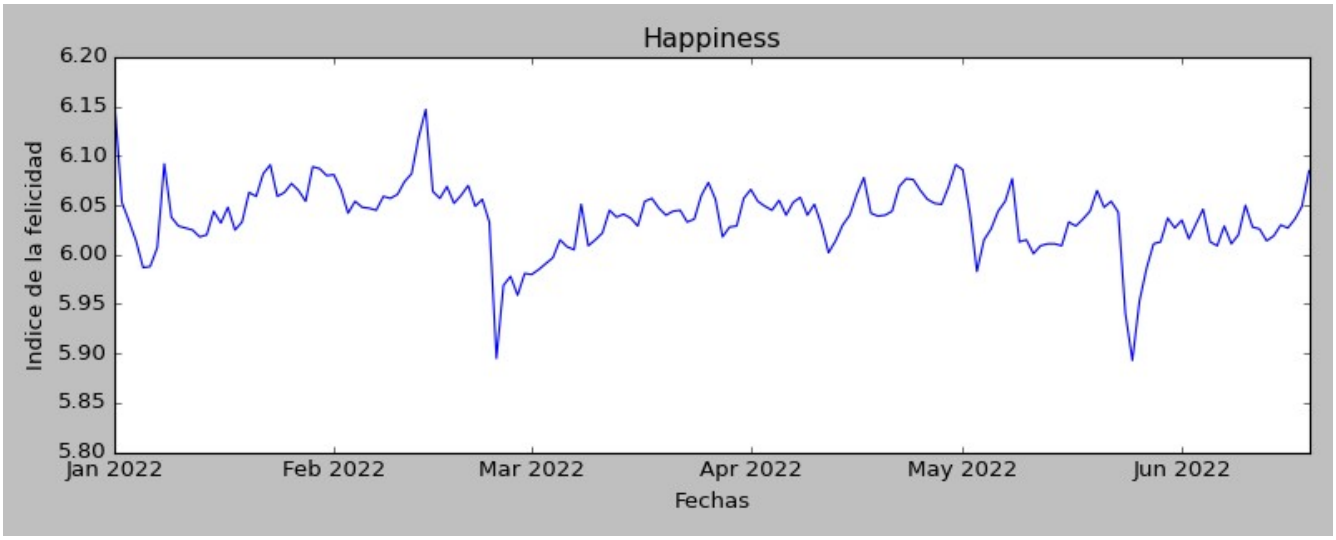


Dónde podemos observas que ax.stackplot(…) crea una gráfica, que colorea el área que se encuentre por debajo de los datos, como ejemplo en el caso del programa para la tarea 14, asumiendo el código donde se importan las bibliotecas y dónde se descarga la información, el método para crear la gráfica sería el siguiente:

```
def crea_fig(info):
    fechas = []
    felicidad = []
    for dato in info:
        fechas.append(dato[0])
        felicidad.append(dato[2])

    plt.style.use('classic')
```

```
fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(fechas, felicidad)
ax.set_title('Happiness')
ax.set_ylabel('Indice de la felicidad')
ax.set_xlabel('Fechas')
ax.set_xlim(xmin=fechas[0], xmax=fechas[-1])
ax.set_ylim(ymin=5.8, ymax=6.2)
fig.tight_layout()
plt.show()
```

Donde podemos notar que nuevamente llamamos a la figura ax, pero en este caso usamos ax.plot(fechas, felicidad), donde es como si pusiéramos solo los datos del eje x, y los del eje y, y con esto nos crea una gráfica que bajo los datos no tiene relleno, es decir, tenemos el resultado siguiente:



Otro cambio que podemos notar es que se usó:

- plt.style.use('classic'), que es el que le da un stylo a nuestro gráfico, teniendo como estilos los siguientes:
  >>> plt.style.available
  ['seaborn-dark', 'seaborn-darkgrid', 'seaborn-ticks', 'fivethirtyeight',
  'seaborn-whitegrid', 'classic', '_classic_test', 'fast', 'seaborn-talk',
  'seaborn-dark-palette', 'seaborn-bright', 'seaborn-pastel', 'grayscale',
  'seaborn-notebook', 'ggplot', 'seaborn-colorblind', 'seaborn-muted',
  'seaborn', 'Solarize_Light2', 'seaborn-paper', 'bmh', 'seaborn-white',
  'dark_background', 'seaborn-poster', 'seaborn-deep']

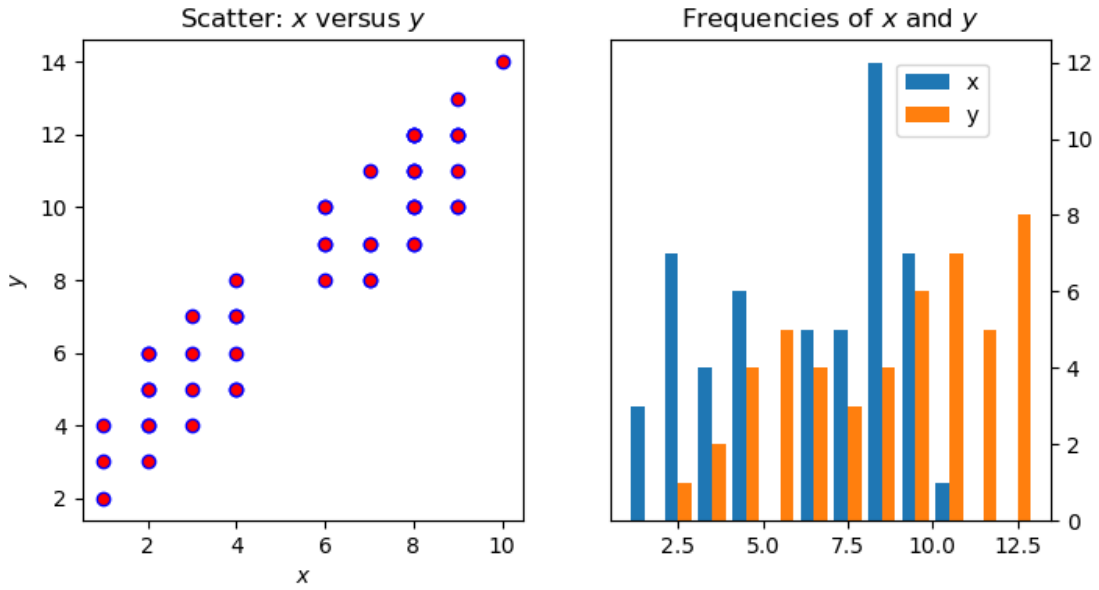
Los ejemplos que se nos muestran después, son cómo ir graficando de acuerdo a otro tipo de datos que tengamos, otra forma en la que los queramos comparar.

Ejemplo 2.

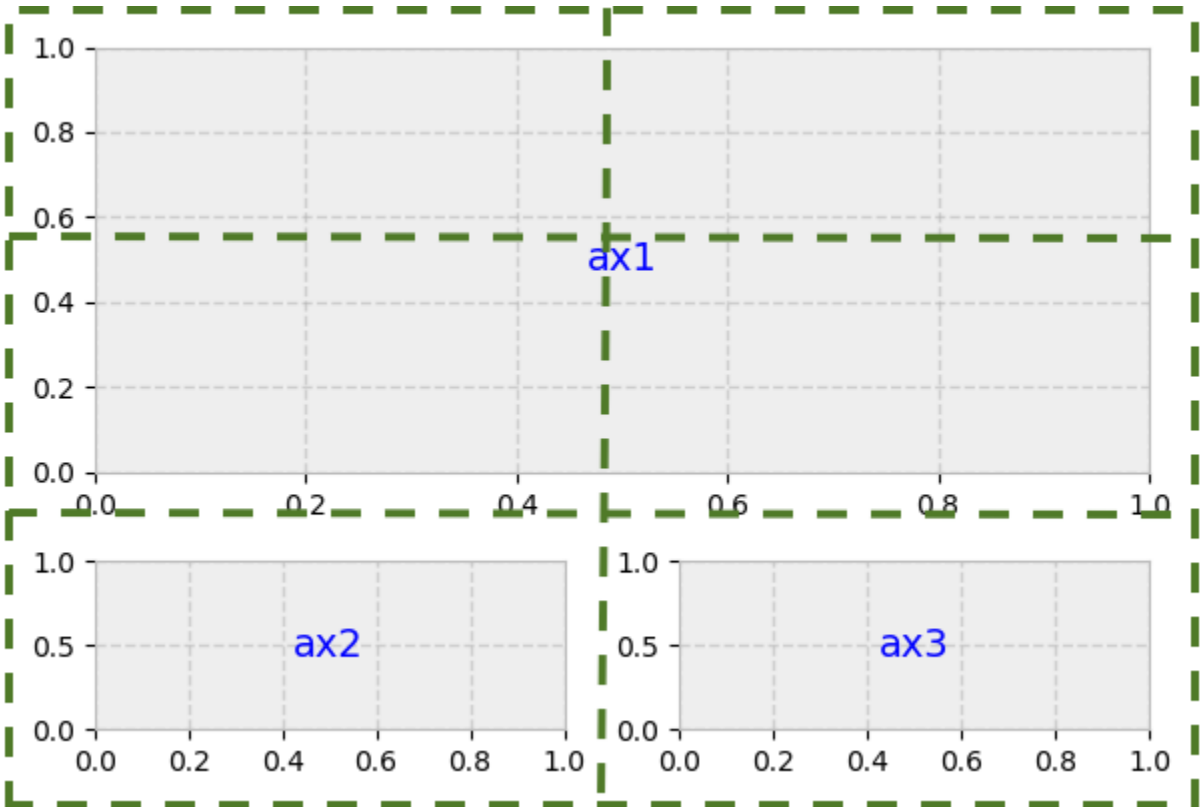
En este ejemplo, nos muestran como trabajar con 2 arreglos que estén correlacionados comparar su distribución discreta.

```
>>> x = np.random.randint(low=1, high=11, size=50)
>>> y = x + np.random.randint(1, 5, size=x.size)
>>> data = np.column_stack((x, y))

>>> fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2,
...                                figsize=(8, 4))

>>> ax1.scatter(x=x, y=y, marker='o', c='r', edgecolor='b')
>>> ax1.set_title('Scatter: $x$ versus $y$')
>>> ax1.set_xlabel('$x$')
>>> ax1.set_ylabel('$y$')

>>> ax2.hist(data, bins=np.arange(data.min(), data.max()),
...          label=('x', 'y'))
>>> ax2.legend(loc=(0.65, 0.8))
>>> ax2.set_title('Frequencies of $x$ and $y$')
>>> ax2.yaxis.tick_right()
```

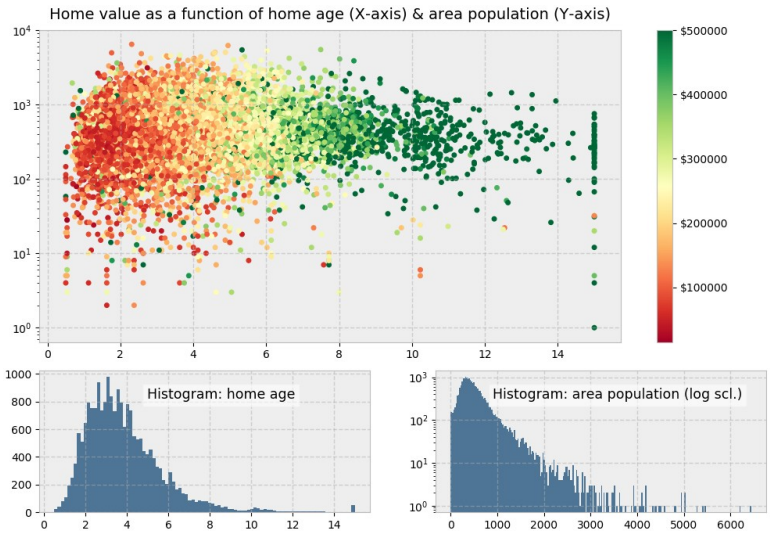
En el primer párrafo simplemente se crea la información que es como si se tuviesen dos columnas que se están comparando, una es y y la otra es x.
Luego se crean las figuras con las que se va a trabajar, que son ax1, ax2, y se indica que se va a trabajar con una fila y dos columnas cada gráfica.
En el siguiente párrafo se nos indica la forma en la que se va a dibujar a ax1, que es con scatter, donde los parámetros son los que se nos indican ahí.
En el siguiente párrafo se nos indica como se va a trabajar con ax2 y los parámetros que va a tener, en este caso se trabaja con data, en conjunto se compara la columna x y la y.
Por lo anterior es importante qué tipo de gráfico vamos a trabajar, pues en este caso el resultado es el mostrado a continuación:



Al final es importante recalcar que el objeto es una sóla figura, en la que se trabaja con dos ‘subfiguras’, es decir, como un espacio en el que se decide qué poner en cada subespacio. Y en esto se puede adentrar más para poder manipular estos espacios, algo así como lo que muestra la figura a contiuación:



En la página se muestra ejemplo para esto. Si se quiere adentrar en esto podemos ver el siguiente ejemplo:



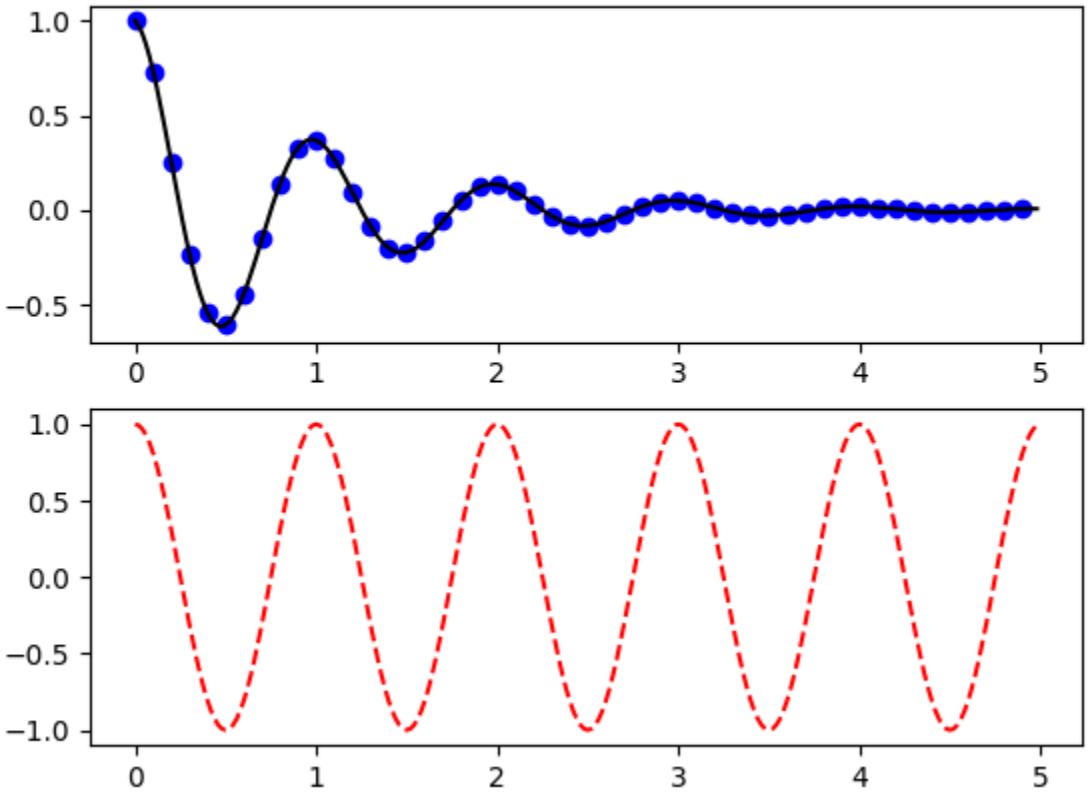
El cual se realiza con el código mostrado a continuación:

```
>>> from io import BytesIO
>>> import tarfile
>>> from urllib.request import urlopen

>>> url = 'http://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.tgz'
>>> b = BytesIO(urlopen(url).read())
>>> fpath = 'CaliforniaHousing/cal_housing.data'

>>> with tarfile.open(mode='r', fileobj=b) as archive:
...     housing = np.loadtxt(archive.extractfile(fpath), delimiter=',')

>>> y = housing[:, -1]
>>> pop, age = housing[:, [4, 7]].T

>>> def add_titlebox(ax, text):
...     ax.text(.55, .8, text,
...         horizontalalignment='center',
...         transform=ax.transAxes,
...         bbox=dict(facecolor='white', alpha=0.6),
...         fontsize=12.5)
...     return ax

>>> gridsize = (3, 2)
>>> fig = plt.figure(figsize=(12, 8))
>>> ax1 = plt.subplot2grid(gridsize, (0, 0), colspan=2, rowspan=2)
>>> ax2 = plt.subplot2grid(gridsize, (2, 0))
>>> ax3 = plt.subplot2grid(gridsize, (2, 1))

>>> ax1.set_title('Home value as a function of home age & area population',
...               fontsize=14)
>>> sctr = ax1.scatter(x=age, y=pop, c=y, cmap='RdYlGn')
>>> plt.colorbar(sctr, ax=ax1, format='$%d')
>>> ax1.set_yscale('log')
>>> ax2.hist(age, bins='auto')
>>> ax3.hist(pop, bins='auto', log=True)

>>> add_titlebox(ax2, 'Histogram: home age')
>>> add_titlebox(ax3, 'Histogram: area population (log scl.)')
```

El cuál nos muestra la manera en la que se puede ir manipulando esto, y cómo obtener los resultados que deseamos.

En el siguiente enlace:
https://matplotlib.org/stable/tutorials/introductory/pyplot.html

Se nos muestra más a detalle como es posible modificar las figuras, ciertos parámetros.
Por ejemplo para obtener la figura siguiente:



Basta tener el siguiente código:

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure()                                    (1)
plt.subplot(211)                                (1)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')       (1)

plt.subplot(212)                                (2)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')         (2)
plt.show()
```

Dónde (1) indica lo necesario para obtener la primera gráfica, dónde se grafica t1 con f(t1) utilizando ‘bo’ que nos da como resultado una línea continua, y ahí mismo grafica la relación t2 con f(t2), utilizando ‘k’, con lo que obtenemos los puntos.
Y como se observa en (2) se obtiene ese resultado con el parámetro ‘r--’.
En esta página tenemos la tabla de posibles valores para modificar nuestras figuras y un poco más de detalles.

Referencia:
https://realpython.com/python-matplotlib-guide/#understanding-pltsubplots-notation
https://matplotlib.org/stable/tutorials/introductory/pyplot.html