Visualizaciones de datos con Python

Las visualizaciones son una herramienta fundamental para entender y compartir ideas sobre los datos. La visualización correcta puede ayudar a expresar una idea central, o abrir un espacio para una más profunda investigación; con ella se puede conseguir que todo el mundo hable sobre un conjunto de datos , o compartir una visión sobre lo que los datos nos quieren decir.Una buena visualización puede dar a quien la observa un sentido rico y amplio de un conjunto de datos .

Puede comunicar los datos de manera precisa a la vez que expone los lugares en dónde se necesita más información o dónde una hipótesis no se sostiene. Por otra parte, la visualización nos proporciona un lienzo para aplicar nuestras propias ideas, experiencias y conocimientos cuando observamos y analizamos datos, permitiendo realizar múltiples interpretaciones. Si como dice el dicho *"una imagen vale más que mil palabras"* , un gráfico interactivo bien elegido entonces podría valer cientos de pruebas estadísticas.

Librerías para visualizar datos en Python

Como bien sabemos, la comunidad de Python es muy grande, por lo tanto vamos a poder encontrar un gran número de librerías para visualizar datos.

Al tener tanta variedad de opciones, a veces se hace realmente difícil determinar cuando utilizar cada una de ellas. En este artículo yo voy a presentar solo cuatro que creo que cubren un gran abanico de casos:

● **Matplotlib** : Que es la más antigua y se convirtió en la librería por defecto para visualizaciones de datos; muchas otras están basadas en ella. Es extremadamente potente, pero con ese poder viene aparejada la complejidad. Se puede hacer prácticamente de todo con Matplotlib pero no siempre es tan fácil de averiguar como hacerlo. Los que siguen el blog me habrán visto utilizarla en varios artículos.

● **Bokeh** : Una de las más jóvenes librerías de visualizaciones, pero no por ello menos potente. Bokeh es una librería para visualizaciones interactivas diseñada para funcionar en los navegadores web modernos. Su objetivo es proporcionar una construcción elegante y concisa de gráficos modernos al estilo de D3.js , y para ampliar esta capacidad con la interactividad y buen rendimiento sobre grandes volúmenes de datos. Bokeh puede ayudar a cualquier persona a crear en forma rápida y sencilla gráficos interactivos, *dashboards* y aplicaciones de datos. Puede crear tanto gráficos estáticos como gráficos interactivos en el servidor de Bokeh . **Bokeh** permite interaccionar con los gráficos e incluso poner los datos en un servidor e interaccionar con ese servidor filtrando datos en tiempo real mientras se pinta la gráfica.

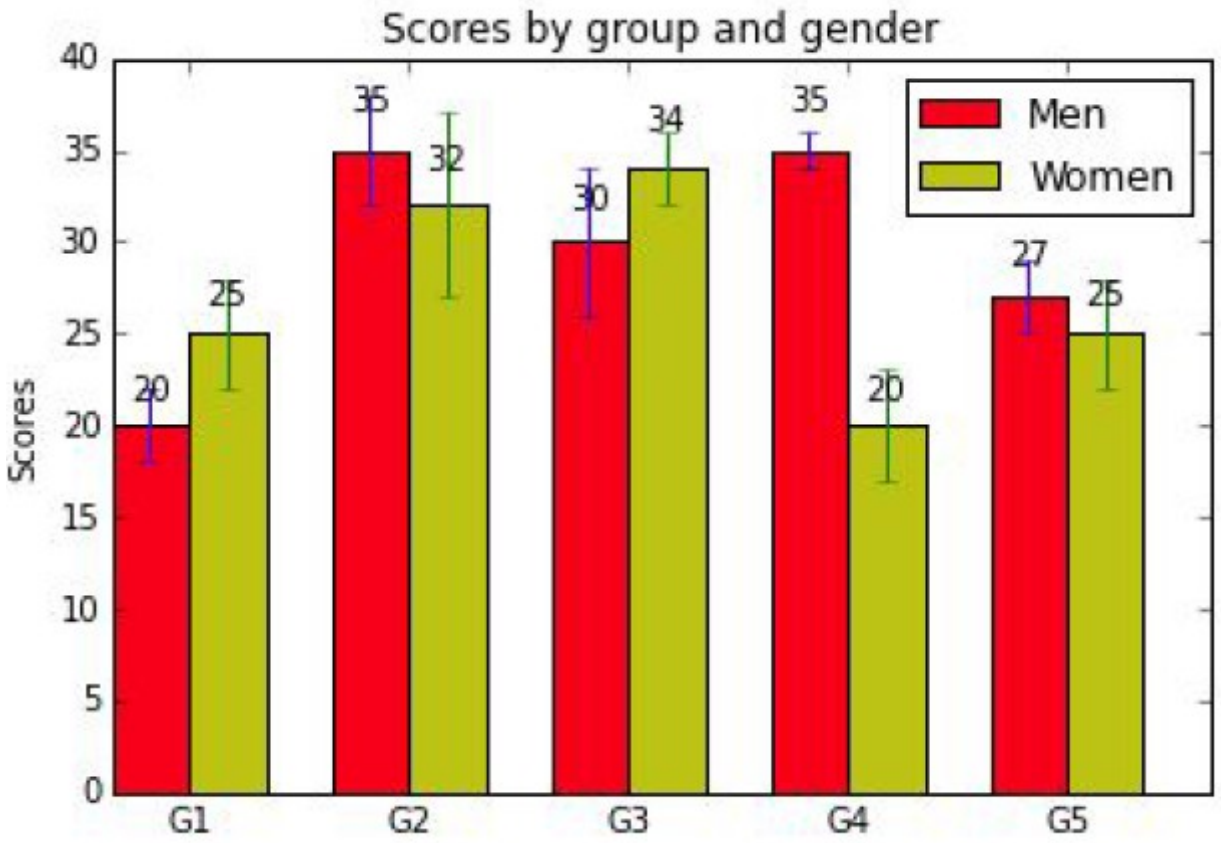
● **Seaborn** : Si de gráficos estadísticos se trata, Seaborn es la librería que deberíamos utilizar, con ella podemos crear gráficos estadísticos informativos y atractivos de forma muy sencilla. Es una de las tantas librerías que se basan en Matplotlib pero nos ofrece varias características interesantes tales como temas, paletas de colores, funciones y herramientas para visualizar distribuciones de una o varias variables aleatorias , regresiones lineales , series de tiempo , entre muchas otras. Con ella podemos construir visualizaciones complejas en forma sencilla.

● **Folium** : Si lo que necesitamos es visualizar datos de geolocalización en mapas interactivos, entonces Folium es una muy buena opción. Esta librería de Python es una herramienta sumamente poderosa para realizar mapas al estilo leaflet.js . El hecho de que los resultados de Folium son interactivos hace que esta librería sea útil para la construcción de *dashboards* .

Gráficas con matplotlib

Para pintar gráficas con matplotlib lo mejor es activar el modo inline , que las incrusta en el mismo notebook

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
N = 5
ind = np.arange(N) # the x locations for the groups
width = 0.35 # the width of the bars
# render men data bar charts with std candle
menMeans = (20, 35, 30, 35, 27)
menStd = (2, 3, 4, 1, 2)
rects1 = plt.bar(ind, menMeans, width, color='r', yerr=menStd)
# render women data bar charts with std candle
womenMeans = (25, 32, 34, 20, 25)
womenStd = (3, 5, 2, 3, 3)
rects2 = plt.bar(ind+width, womenMeans, width, color='y', yerr=womenStd)
# add legend
plt.legend( (rects1[0], rects2[0]), ('Men', 'Women') )
# label bars
def autolabel(rects):
# attach some text labels
for rect in rects:
height = rect.get_height()
plt.text(rect.get_x()+rect.get_width()/2., 1.05*height,
'%d'%int(height),
ha='center', va='bottom')
autolabel(rects1)
autolabel(rects2)
# add some text for labels, title and axes ticks
ax = plt.gca()
ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.set_xticks(ind+width)
ax.set_xticklabels( ('G1', 'G2', 'G3', 'G4', 'G5') )
plt.show() # show the plot
```


Scores by group and gender
Men
Women
Scores
40
35
30
25
20
15
10
5
0
20
25
35
32
30
34
35
20
27
25
G1
G2
G3
G4
G5