Python para Análisis de Datos

Módulo 06 - Matplotlib 4

Matplotlib

Otros tipos de gráficos

Vimos que el módulo pyplot tiene numerosas funciones para activar distintos elementos dentro de un gráfico. También tiene varias funciones para realizar distintos tipos de gráficos.

Hasta ahora solo vimos la función `plot`, pero encontramos también las funciones `scatter`, `hist`, `bar`, `barh`, `pie`, `violinplot`, `boxplot`, etc. Con ellas vamos a poder representar información de diversas formas.

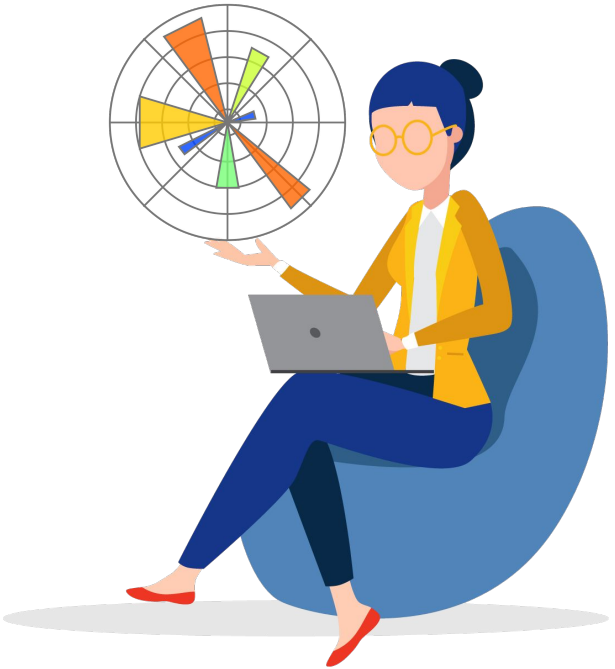


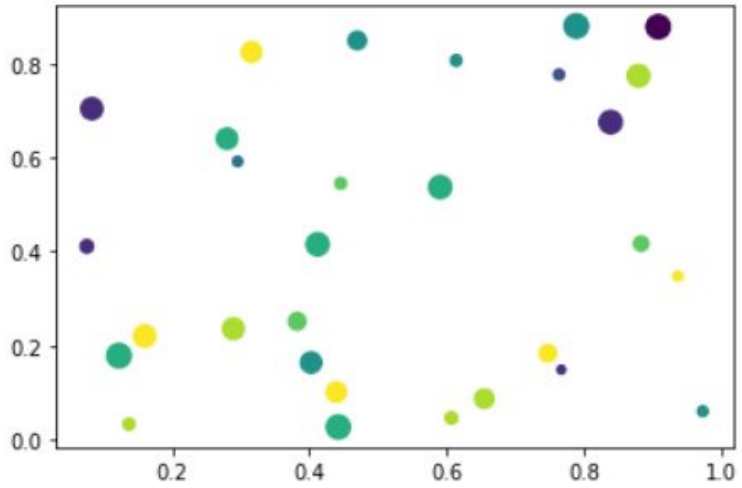
Gráfico de puntos

La función `scatter` función permite graficar puntos en el plano donde además podemos controlar el color y el tamaño de cada punto para representar más información aparte de las coordenadas de los puntos, como categorías o magnitudes continuas asociadas a cada punto.

```
x,y = np.random.random((2,30))
c = np.random.randint(0,9,30)
s = np.random.randint(10,150,30)

plt.scatter(x,y, c=c, s=s)
```

```
<matplotlib.collections.PathCollection at 0x7f0ef568ccd0>
```



Histogramas

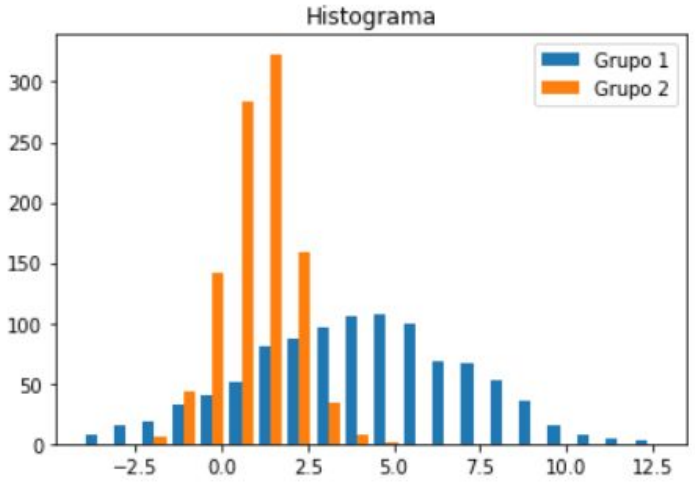
Se pueden crear histogramas con la función `hist`. Este tipo de gráficos es útil para apreciar la distribución de números. Se conforma de barras cuyo alto es la cantidad de datos que están entre los valores determinados por el ancho de la barra.

Se puede controlar la cantidad de barras con el parámetro `bins`. A su vez se puede hacer varios histogramas en una misma figura y controlar cómo se muestran juntas con el parámetro `histtype`.

Ejemplos

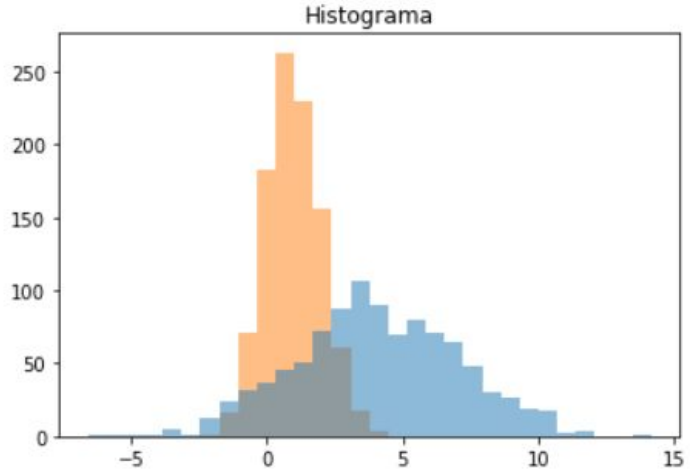
```
x1 = np.random.normal(4,3,1000)
x2 = np.random.normal(1,1,1000)
plt.hist((x1,x2), bins=20, label=["Grupo 1", "Grupo 2"])
plt.title("Histograma")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f0ef59c70d0>
```



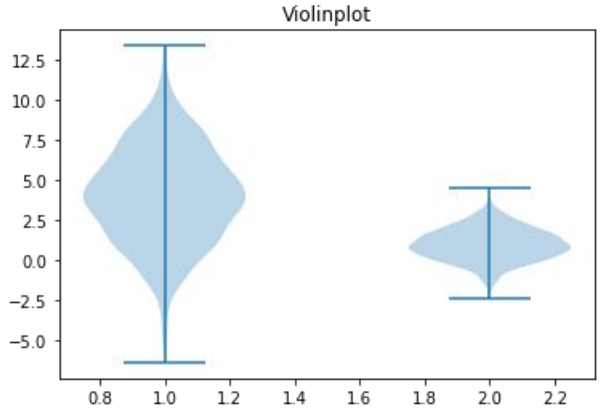

```
x1 = np.random.normal(4,3,1000)
x2 = np.random.normal(1,1,1000)
plt.hist((x1,x2), bins=30, histtype="stepfilled", alpha=0.5)
plt.title("Histograma")
```

```
Text(0.5, 1.0, 'Histograma')
```



Violinplot y Boxplot

Otros gráficos útiles para ver la distribución de números son `violinplot` y `boxplot`. El primero muestra una distribución suavizada de los datos y el segundo marca la mediana y los cuartiles de la distribución de números.



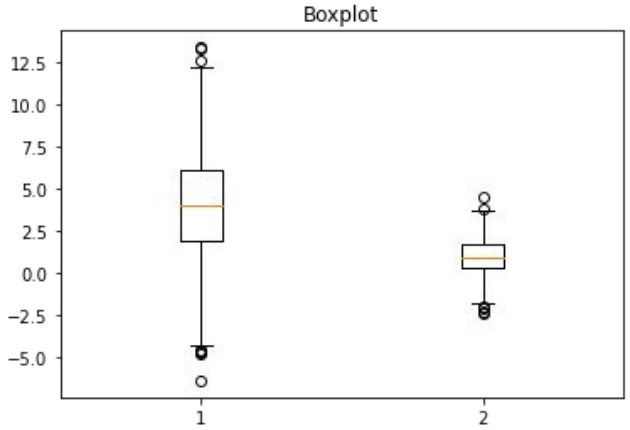


Gráfico de barras

Este tipo de gráfico permite visualizar cantidades mediante barras en donde la longitud de cada barra es proporcional a la magnitud que se quiere representar. Se suele utilizar para representar información que pertenece a categorías distintas.

La función `bar` necesita dos parámetros *x* e *y* que representan las posiciones de los centros de cada barra en el eje *x* y la altura de cada barra en el eje *y* respectivamente. Así, podemos poner barras en cualquier posición del eje *x y* con cualquier altura. El ancho de las barras las podemos controlar con el parámetro `width`. Las etiquetas del eje x se controlan con `tick_label.`

La función `barh` permite hacer barras horizontales.

Ejemplos

```
plt.bar([1,2,4], [18,12,21], tick_label=["A","B","C"])
```

```
<BarContainer object of 3 artists>
```

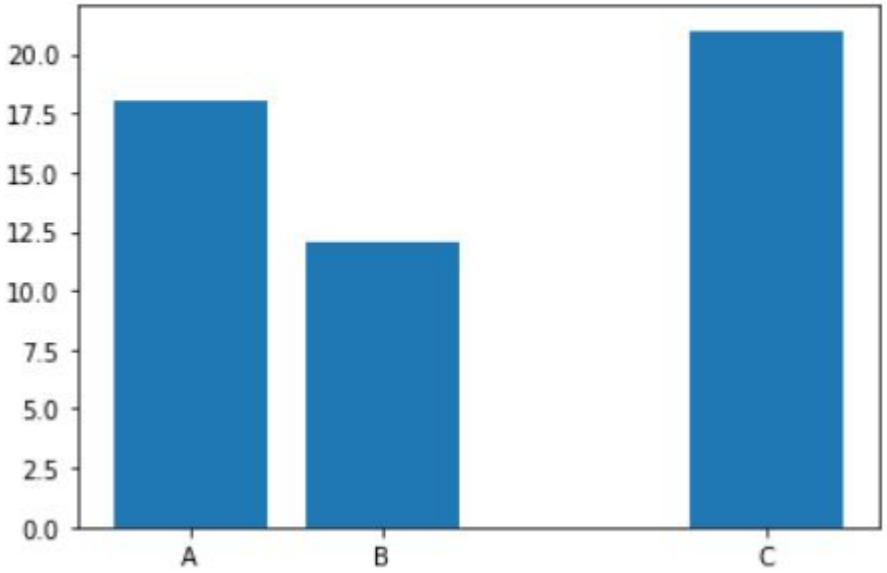


Gráfico de tortas

Este tipo de gráfico es muy útil para mostrar las proporciones relativas entre distintas cantidades visualizándolas como cuñas de un mismo círculo. Se realizan con la función `pie` que toma las magnitudes a representar.

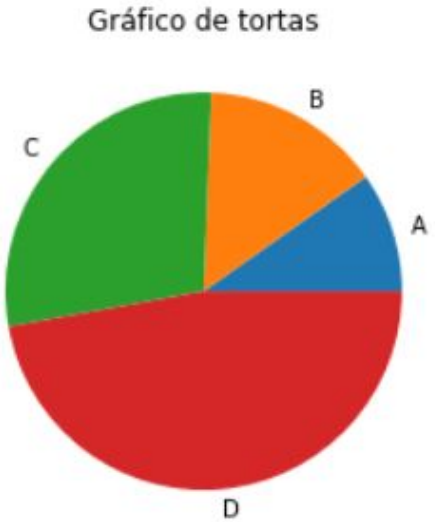
Se le puede especificar el parámetro `labels` para asignar textos a cada cuña. También está el parámetro `explode,` que permite separar las cuñas del centro del gráfico para destacar alguna de ellas.

Ejemplos

```
plt.pie([12,18,35,58], labels=list("ABCD"))

plt.title("Gráfico de tortas")
```

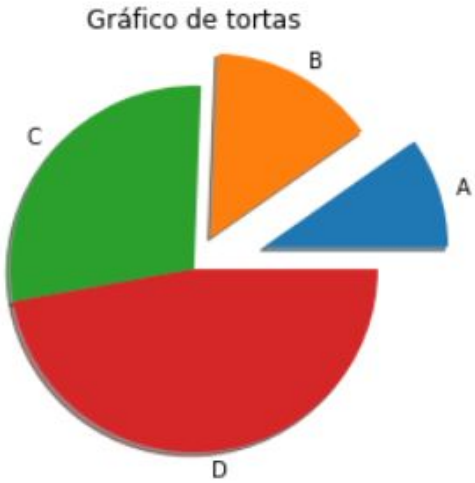
```
Text(0.5, 1.0, 'Gráfico de tortas')
```




```
plt.pie([12,18,35,58], labels=list("ABCD"),
        explode = [0.4,0.2,0,0], shadow=True)

plt.title("Gráfico de tortas")
```

```
Text(0.5, 1.0, 'Gráfico de tortas')
```



¡Muchas gracias!

¡Sigamos trabajando!