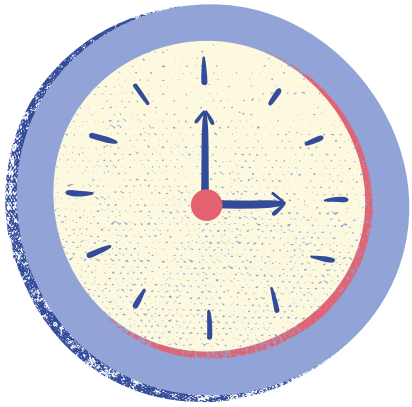
TIC

Lección 3:

Gráficos con Pandas

Planteamiento de la sesión
Tiempo de ejecución: 4 horas



Materiales:

- La librería Matplotlib | Aprende con Alf
- M2_integrador_Pandas.ipynb
- M2_integrador_Pandas.ipynb
- List of named colors – Matplotlib 3.8.3 documentation
- matplotlib.markers – Matplotlib 3.8.3 documentation
- Linestyles – Matplotlib 3.8.3 documentation

TIC

Utilizando la librería Matplotlib

Una de las herramientas más interesantes para el análisis de datos, son los gráficos ya que, una de sus características es, permitir encontrar relaciones entre variables o series o columnas de un DataFrame.

Las últimas versiones de pandas, ya incluyen la librería y no es necesario realizar instalaciones, sin embargo, si solo se quiere trabajar con la librería, puede realizar su instalación con el comando

pip install matplotlib

Para utilizarlo solo es necesario entonces hacer el llamado de la librería, comúnmente acompañada de la librería NumPy

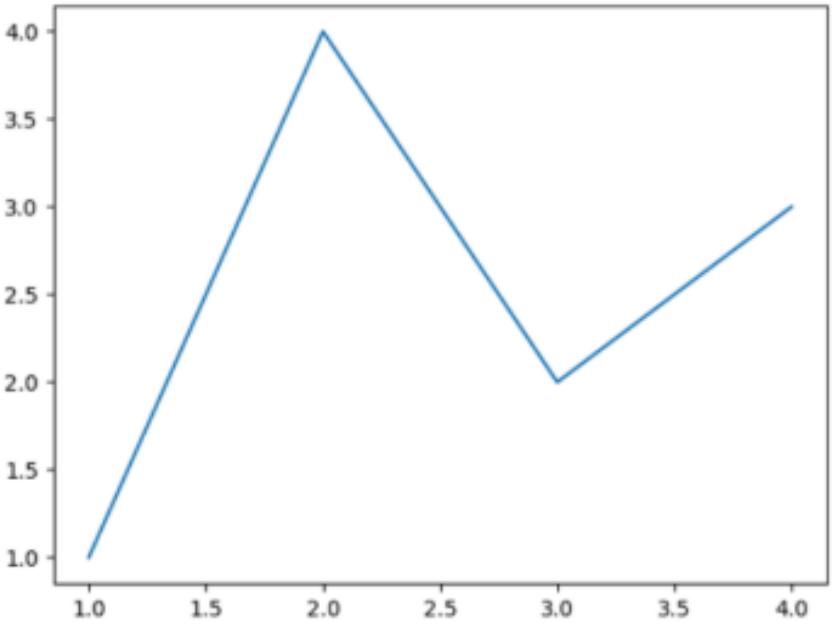
```
import matplotlib.pyplot as plt
import numpy as np

import matplotlib as mpl
```

La funcionalidad base de la librería es convertir los datos que proporcione el usuario en una figura que puede contener uno o más ejes y un área donde se ubican los puntos en una serie de coordenadas normalmente en términos de los ejes X, Y y Z.

Una simple forma que indica la librería para crear una figura, es utilizando pyplot.subplots y el método Axes.plot para dibujar un poco de información.

```
fig, ax = plt.subplots() # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3]) # Plot some data on the axes.
```

- plt.show()



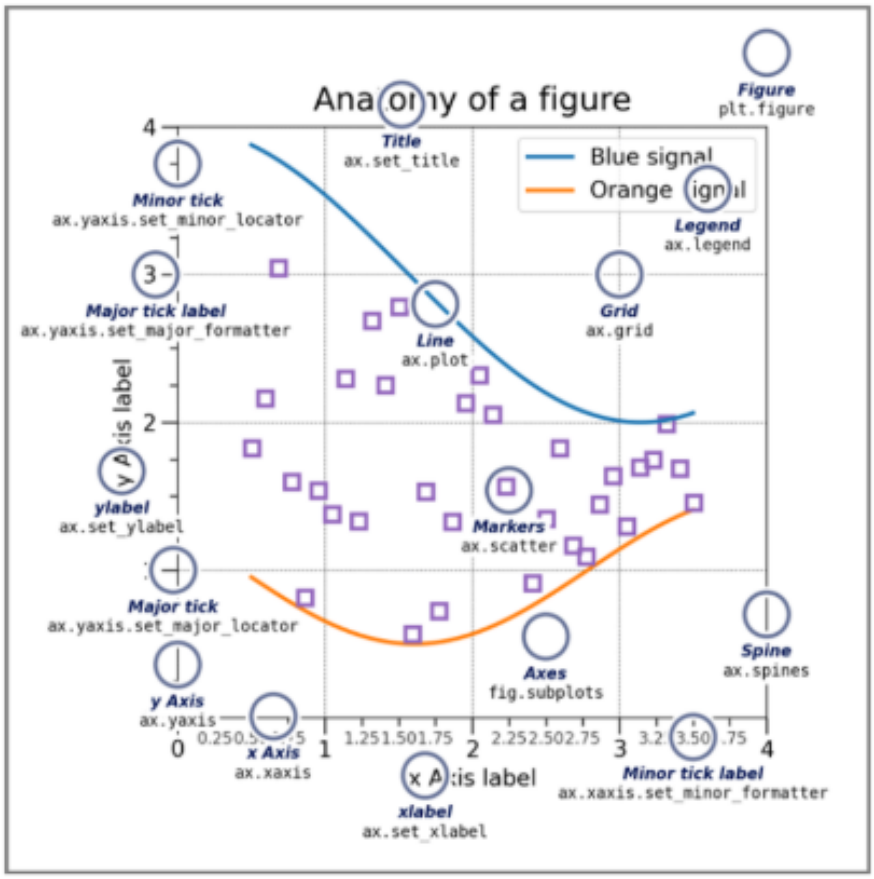
A simple Example, matplotlib, consultado en febrero de 2024, disponible en:
https://matplotlib.org/stable/users/explain/quick_start.html

Como se puede observar, con 3 líneas de código fue suficiente para crear un gráfico, si la data viene de una fuente externa o incluso de otra fuente dentro del proyecto, solo hacen falta un par de líneas más que importen la data y ya estaría todo listo.

Partes de una figura

Una figura consta de diferentes componentes que describen el gráfico y contextualizan a los usuarios sobre la escala, las variables, los rangos y demás marcadores que permiten una lectura e interpretación más acertada de la información presentada.

Parts of a Matplotlib figure, Matplotlib, consultado en febrero de 2024, disponible en: Quick start guide – Matplotlib 3.8.3 documentation



Figura

La figura es el elemento más general que contiene todos los demas objetos, entre los objetos principales se tiene a los ejes, X y Y en 2D y a X, Y y Z en 3D, las etiquetas de eje, las marcas principales, las marcas secundarias, la leyenda, la rejilla, el título, en general, contiene los gráficos.

Ejes

Los ejes son los elementos donde se aplican las marcas guías, y la escala de la data, poseen título y etiqueta, se conocen como XY o XYZ según las dimensiones.

Eje

El eje es cada uno de los objetos como tal que se describen en los ejes, es decir que es propiamente el eje X o Y o Z, y cuentan con etiquetas y formatos.

Artista

Se habla de Artista como todos lo representado en la figura, incluyendo la figura misma, cuando una figura es dibujada, cada uno de los artistas es pintado en el lienzo. La mayoría de los Artistas están atados a los ejes, es por esto que no pueden ser compartidos por varios ejes o movido de uno a otro.

Tipos de gráficos

Cada dataset o conjunto de datos tiene sus particularidades y cada análisis requiere de apoyos gráficos diferentes, es por este motivo que se cuentan con diferentes tipos de gráficos, entendiendo también que cada uno de ellos tiene una definición particular y por ende una aplicación que cobija una serie de casos de uso y que dependerá de lo que el usuario busca para utilizar entre uno y otro.

Algunos de los gráficos más utilizados son:

- Diagramas de barras
- Histograma
- Diagramas de sectores
- Diagramas de caja y bigotes
- Diagramas de violín
- Diagramas de dispersión o puntos

- Diagramas de líneas
- Diagramas de áreas
- Diagramas de contorno
- Mapas de color

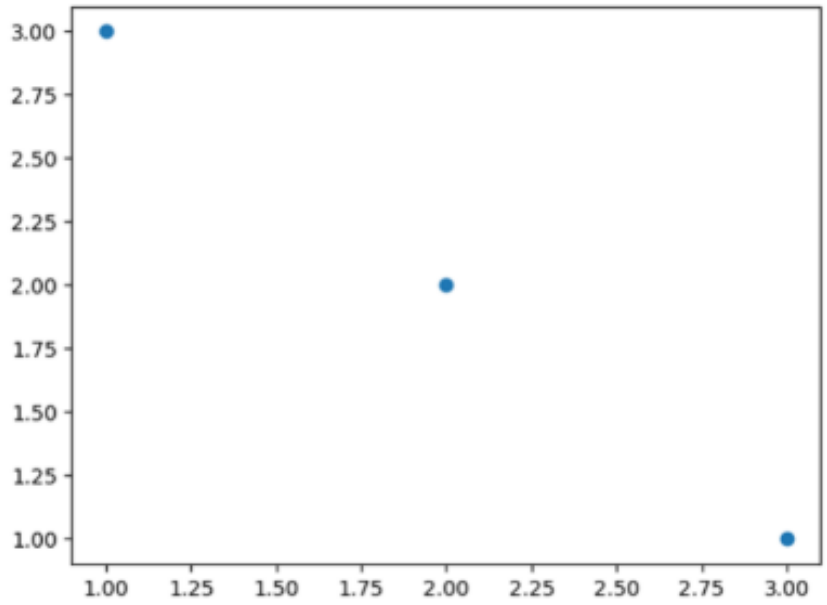
y combinaciones de todos ellos.

Tipos de gráficos

Como se menciona al inicio de la lección, la creación de gráficos requiere de unos pocos pasos y con ello, pocas líneas de código, recordando entonces lo que se requiere es posible listar:

- Importar el módulo pyplot.
- Definir la figura que contendrá el gráfico, que es la region (ventana o página) donde se dibujará y los ejes sobre los que se dibujarán los datos. Para ello se utiliza la función subplots().
- Dibujar los datos sobre los ejes. Para ello se utilizan distintas funciones dependiendo del tipo de gráfico que se quiera.
- Personalizar el gráfico. Para ello existen multitud de funciones que permiten añadir un título, una leyenda, una rejilla, cambiar colores o personalizar los ejes.
- Guardar el gráfico. Para ello se utiliza la función savefig().
- Mostrar el gráfico. Para ello se utiliza la función show().

```
# Importar el módulo pyplot con el alias plt
import matplotlib.pyplot as plt
# Crear la figura y los ejes
fig, ax = plt.subplots()
# Dibujar puntos
ax.scatter(x = [1, 2, 3], y = [3, 2, 1])
# Guardar el gráfico en formato png
plt.savefig('diagrama-dispersion.png')
# Mostrar el gráfico
plt.show()
```



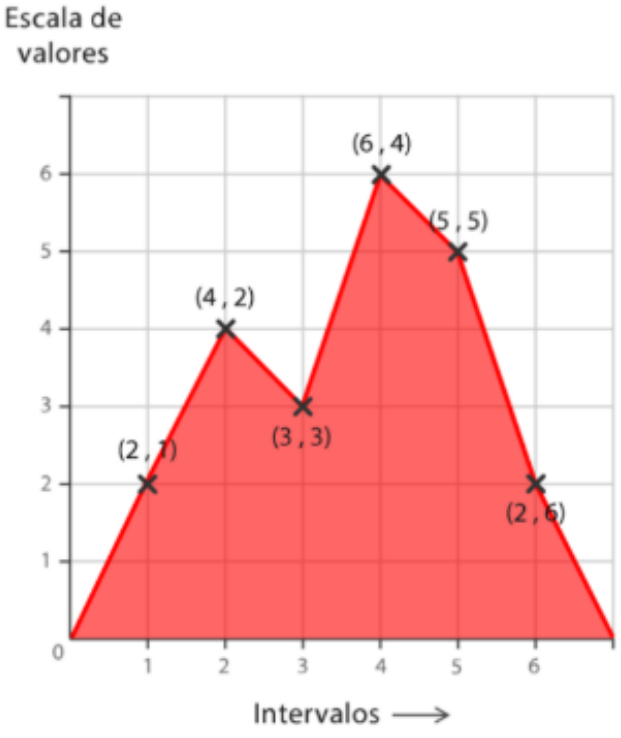
A continuación, se presentan algunas breves descripciones de cada uno de los tipos de gráficos y un ejemplo corto de su implementación.

Gráficos de líneas

Permiten visualizar los cambios a lo largo de un rango continuo, como el tiempo o la distancia, un ejemplo muy utilizado es la población a lo largo de los años, en donde uno de los ejes, normalmente el Y, cuenta con una escala en cientos, miles o millones de habitantes y el otro eje, normalmente el eje X, llevaría los años, similar al gráfico presentado.

```
#plot(x, y)

import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 0, 0.5, 3, 2.2, 1.5, 2.5])
ax.plot([2,4,6,8,10,12,14,16], [1, 2, 0, 0.5, 3, 2.2, 1.5, 2.5])
plt.show()
```

Gráficos de áreas

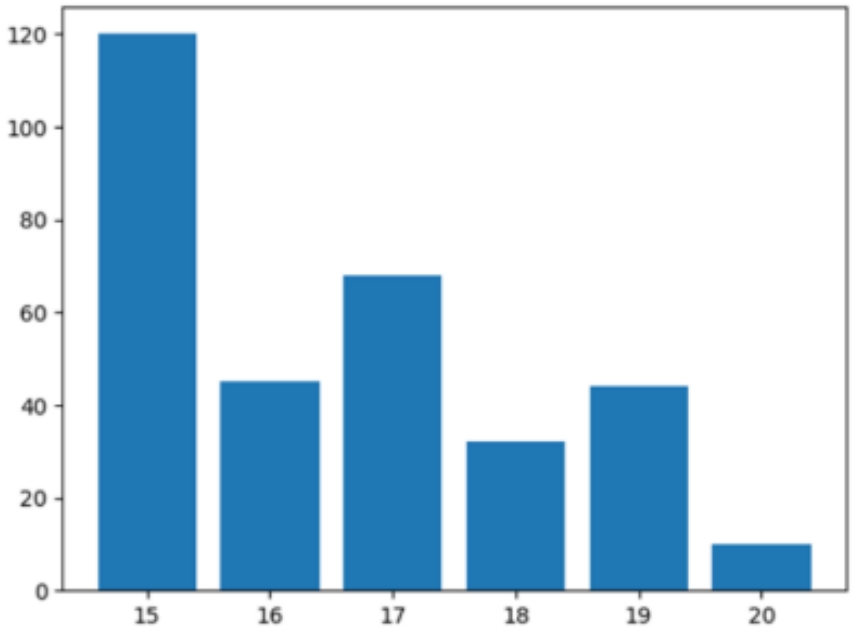


Fuente: Catálogo de visualización de Datos, Gráfica de Área, consultado en febrero de 2024, disponible en: Gráfica de Área

Se utilizan para mostrar el desarrollo de valores cuantitativos a lo largo de un intervalo o período de tiempo. Son parecidos a los gráficos de líneas, pero se diferencian en que el espacio situado debajo de las líneas está sombreado para que se aprecie mejor la magnitud de las tendencias.

```
#fill_between(x, y)

import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4, 5, 6, 7, 8], [1, 2, 0, 0.5, 3, 2.2, 1.5, 2.5])
ax.plot([2,4,6,8,10,12,14,16], [1, 2, 0, 0.5, 3, 2.2, 1.5, 2.5])
plt.show()
```

Gráfico de barras verticales

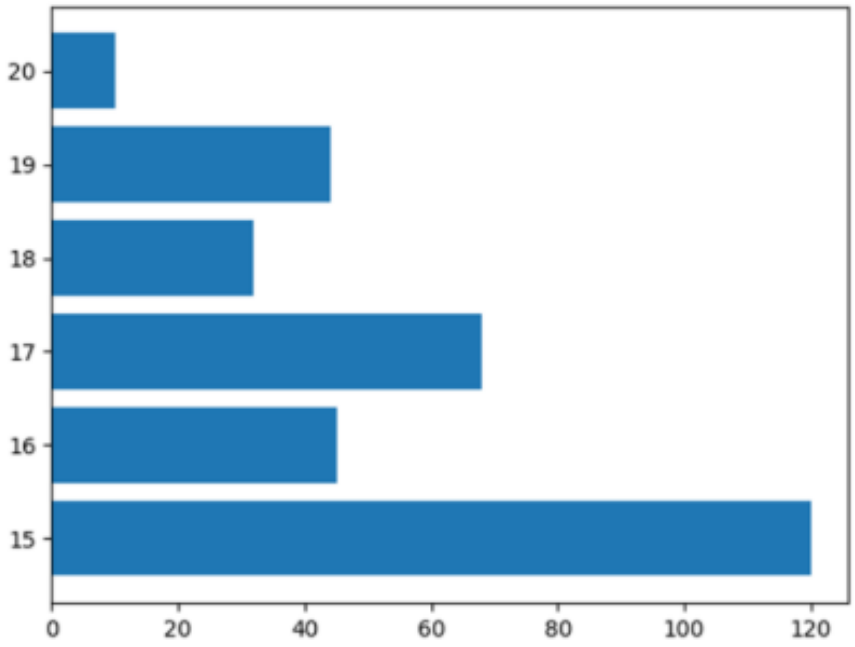


Fuente: Propia

Una forma de representar gráficamente datos numéricos mediante rectángulos verticales (u horizontales), conocidos como barras. El tamaño de cada barra se ajusta proporcionalmente al valor que representa. Este tipo de gráficos proporcionan una comparación visual de cantidades o frecuencias, lo que facilita la interpretación de los datos.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.bar([15,16,17,18,19,20], [120,45,68,32,44, 10])
plt.show()
```

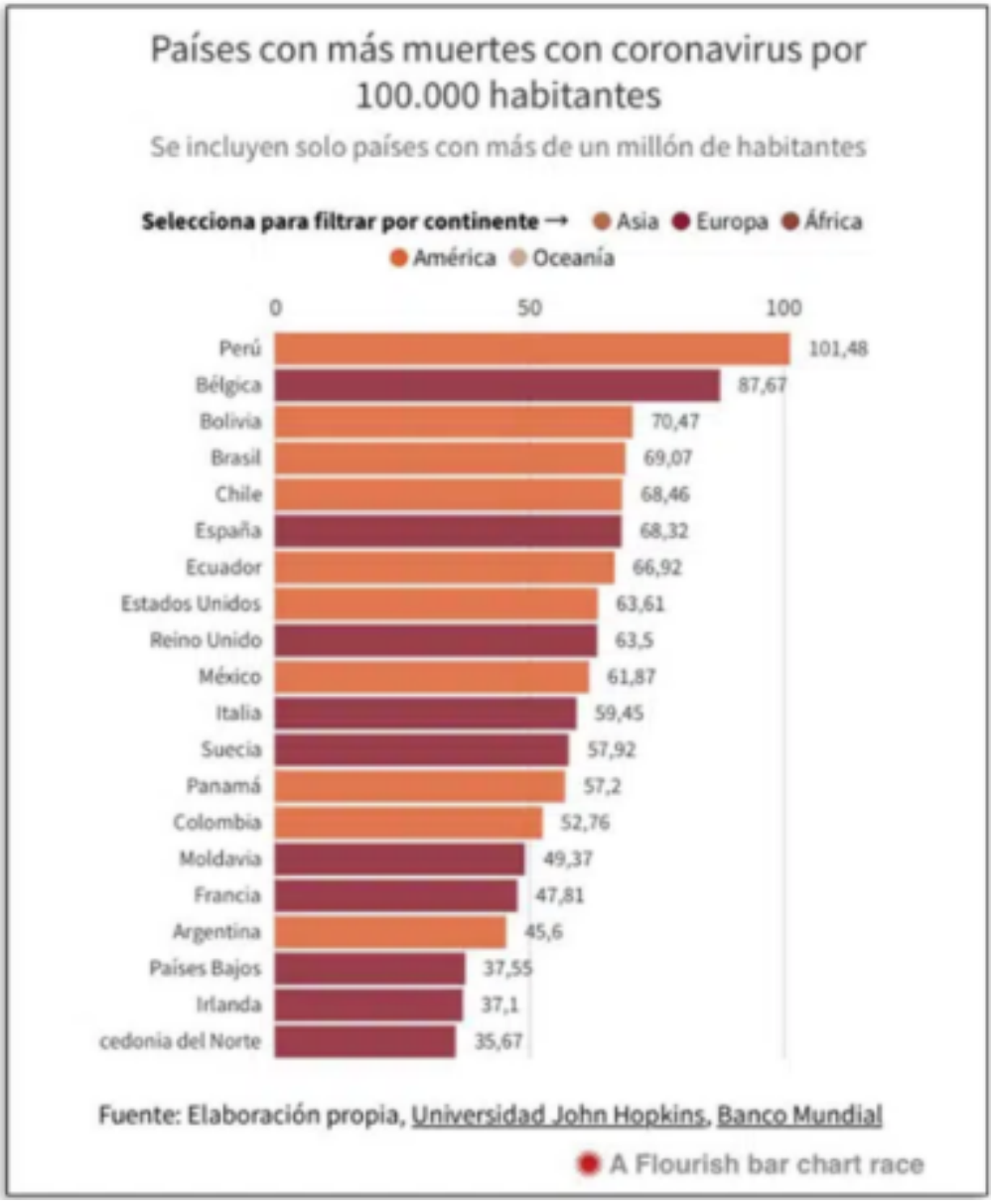
Gráfico de barras horizontales



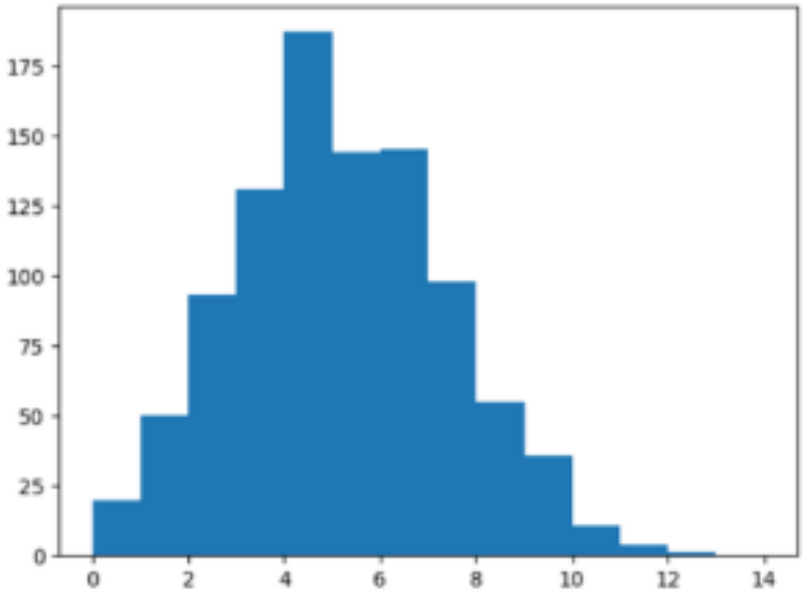
Fuente: Propia

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.barh([15,16,17,18,19,20], [120,45,68,32,44, 10])
plt.show()
```

Los gráficos de líneas son útiles para mostrar cambios más pequeños en una tendencia a lo largo del tiempo. Los gráficos de barras son mejores para comparar cambios más grandes o diferencias en los datos entre grupos. Las barras horizontales son utilizadas en casos , por ejemplo, donde las etiquetas sobre el eje horizontal no se puedan visualizar correctamente, como el siguiente gráfico.



Histogramas



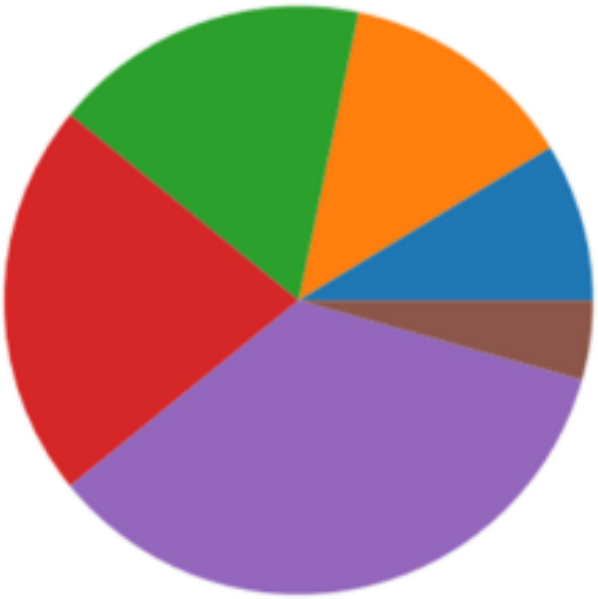
Los histogramas son gráficos que indican la frecuencia de un hecho mediante una distribución de los datos. Los histogramas no se pueden elaborar con atributos, sino con variables medibles tales como peso, temperatura, tiempo, etc.

En definitiva, un histograma es una representación gráfica de una variable en forma de barras, donde la superficie de cada barra es proporcional a la frecuencia de los valores representados. En el eje vertical se representan las frecuencias, y en el eje horizontal los valores de las variables, normalmente señalando las marcas de clase, es decir, la mitad del intervalo en el que están agrupados los dato.

ofrecen una buena forma de evaluar los datos. Se pueden usar para comprobar valores extremos o atípicos y ayudar a comprender la distribución de sus datos.

```
import numpy as np
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.random.normal(5, 2.5, size=1000)
ax.hist(x, np.arange(0, 15))
plt.show()

print(x);
```

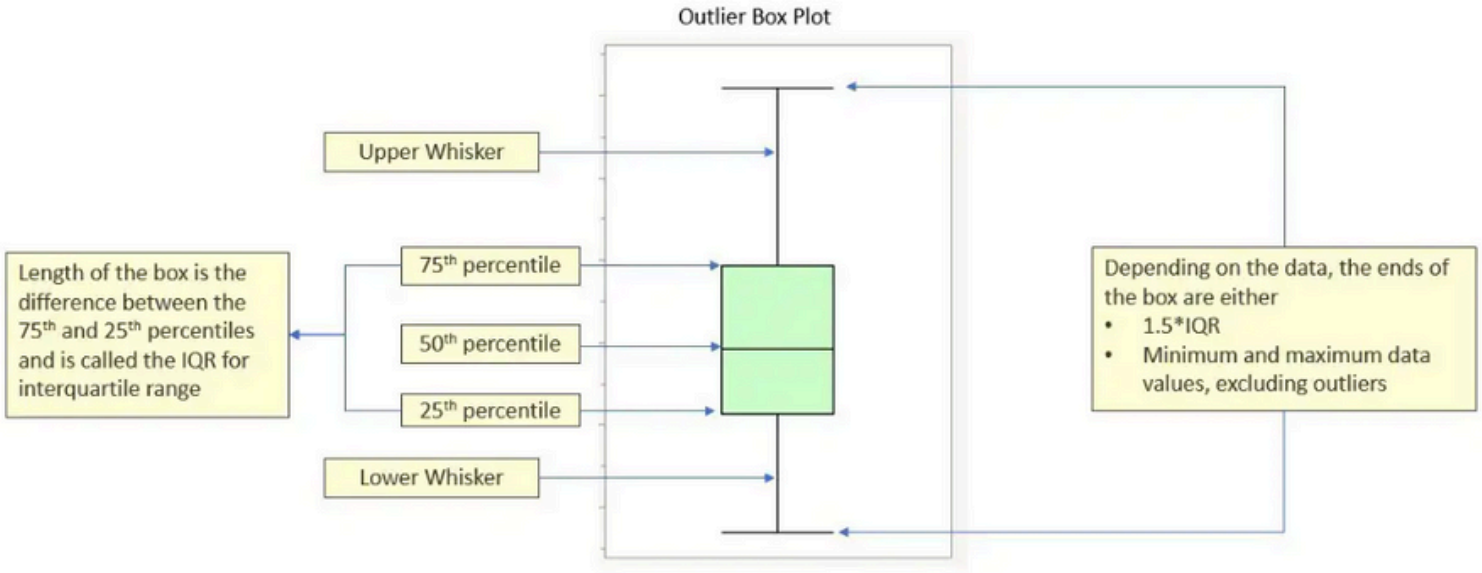
Diagramas de sectores



El Diagrama de Sectores también se conoce como Gráfico de Torta o Gráfico Circular. Representa los datos en un círculo, de modo que la frecuencia de cada valor viene dada por un trozo de área del círculo. Así, el círculo queda dividido en sectores cuya amplitud es proporcional a las frecuencias de los valores. Con este tipo de gráfica estadística se puede representar cualquier variable.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.pie([100, 150, 200, 250, 400, 50])

plt.show()
```

Diagramas de caja y bigotes



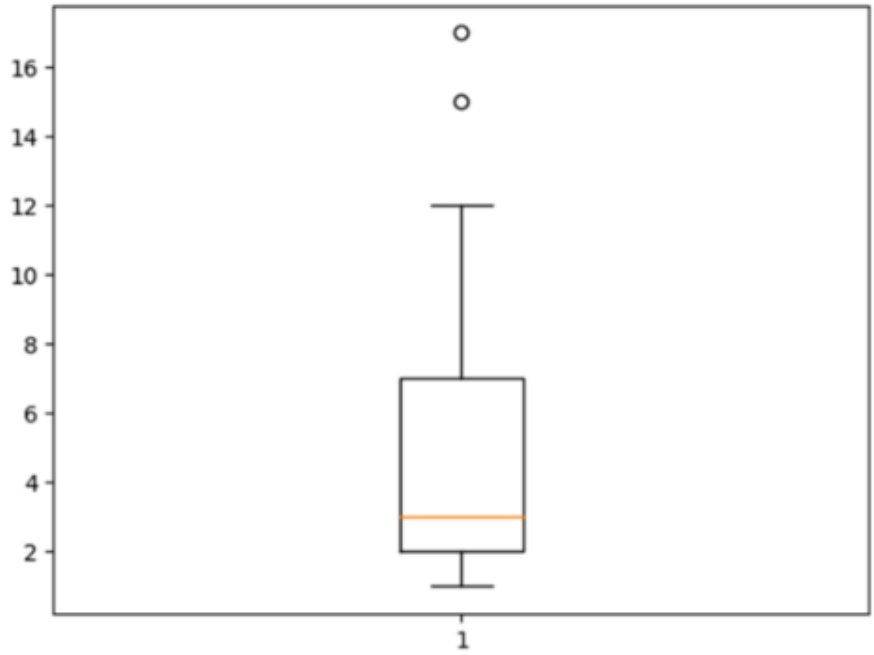
Fuente: Diagrama de caja con etiquetas percentiles, consultado en febrero de 2024, disponible en: https://www.jmp.com/es_co/statistics-knowledge-portal/exploratory-data-analysis/box-plot/_jcr_content/par/styledcontainer_2069/par/image_1343387437.img.jpg/1613495956050.jpg

El término «diagrama de caja» se refiere a los diagramas de caja de valores atípicos. También se los conoce como diagramas de caja y bigotes o diagramas de caja de Tukey.

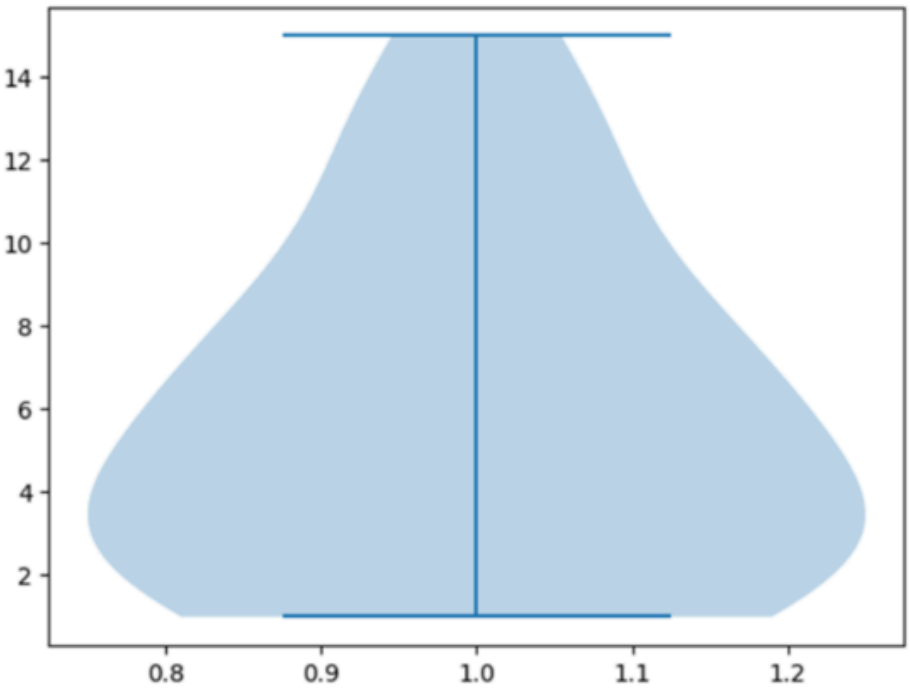
Estas son las partes básicas de un diagrama de caja:

- La línea central de la caja indica la mediana de los datos. Una mitad de los datos está por debajo de este valor, y la otra por encima. Si los datos son simétricos, la mediana estará en el centro de la caja. Si los datos están sesgados, la mediana estará más cerca de la parte superior o inferior de la caja.
- Los extremos de arriba y abajo de la caja indican los cuartiles, o percentiles, 25 y 75. La longitud de la caja es la diferencia entre estos dos percentiles y se conoce como rango intercuartílico (IQR).
- Las líneas que se extienden desde la caja se llaman bigotes. Los bigotes representan la varianza esperada de los datos. Estos bigotes se extienden 1,5 veces el IQR desde los extremos superior e inferior de la caja. Si los datos no llegan hasta el final de los bigotes, estos se ajustan a los valores mínimo y máximo de los datos. Si hay datos que queden por encima o por debajo de los extremos de los bigotes, se los representa con puntos. Estos puntos se conocen como valores atípicos. Un valor atípico es el que supera la varianza esperada. Merece la pena revisar estos puntos de datos para aclarar si son atípicos o erróneos. Los bigotes no incluyen dichos valores.

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.boxplot([1, 2, 1, 2, 3, 4, 3, 3, 5, 7, 12, 17, 15])

plt.show()
```



Diagramas de violín

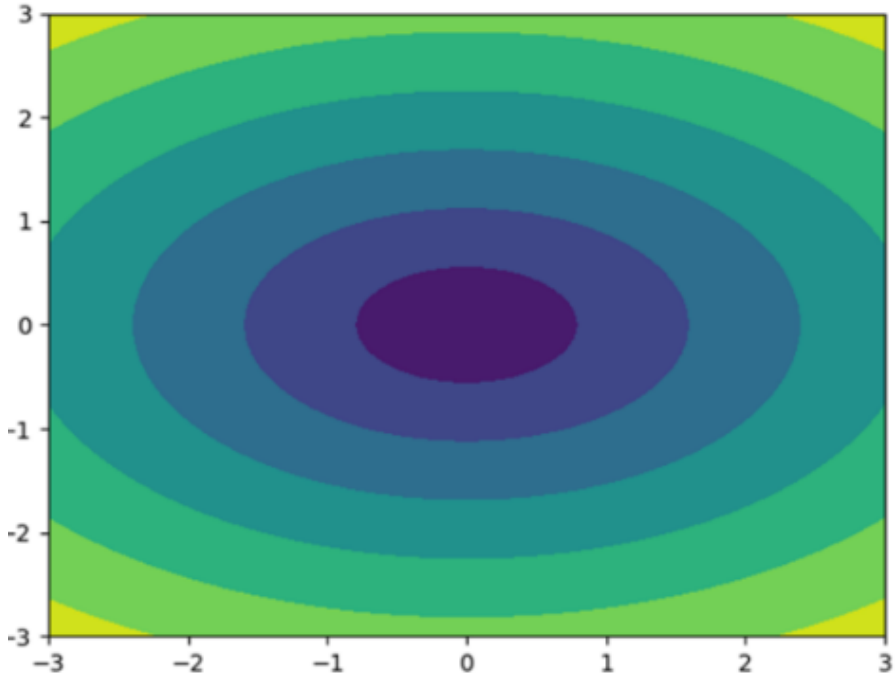


Con la ayuda de un diagrama de violín se puede mostrar gráficamente la distribución de los datos métricos. Es una forma de visualizar la distribución de un conjunto de datos, mostrando no sólo la media y la varianza, sino también todo el rango de los datos. El diagrama de violín es similar al diagrama de caja, pero además muestra la distribución de probabilidad de la variable en cuestión. Por lo tanto, un diagrama de violín es una combinación de un diagrama de caja y un diagrama de densidad del núcleo.

Los diagramas de violín son especialmente útiles cuando el conjunto de datos tiene muchos valores atípicos o cuando la distribución está muy sesgada.

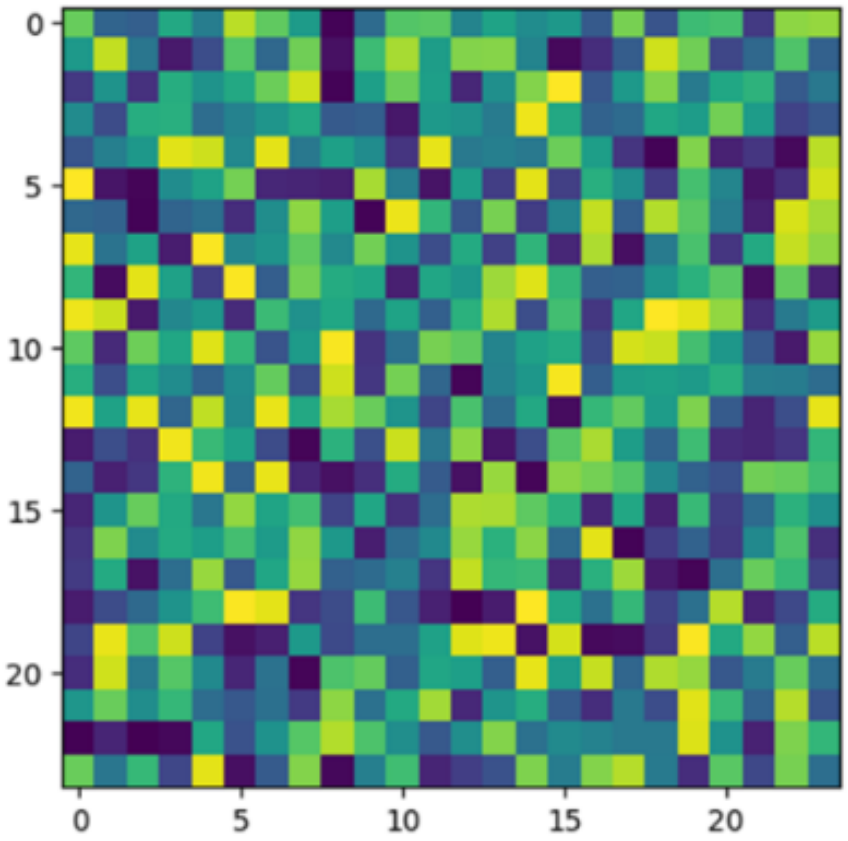
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.violinplot([1, 2, 1, 2, 3, 4, 3, 3, 5, 7, 9, 12, 6, 7, 8, 12 , 15])

plt.show()
```

Diagramas de contorno



```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.linspace(-3.0, 3.0, 100)
y = np.linspace(-3.0, 3.0, 100)
x, y = np.meshgrid(x, y)
z = np.sqrt(x**2 + 2*y**2)
ax.contourf(x, y, z)

plt.show()
```

Mapas de color




```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
x = np.random.random((24, 24))
ax.imshow(x)

plt.show()
```

Cambiar el aspecto de los gráficos

Los gráficos utilizados durante la lección, mantiene las etiquetas, colores, títulos, marcadores y estilos por defecto de la librería, sin embargo es posible realizar algunos ajustes para adaptarlos a la presentación que se necesite.

Colores

El cambio de color se puede realizar con el parámetro "color = nombre_color" , y el nombre_color, puede ser consultado en la documentación oficial List of named colors – Matplotlib 3.8.3 documentation

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], color = 'tab:purple')
ax.plot(dias, temperaturas['Barcelona'], color = 'tab:green')

plt.show()
```

Marcadores

Los marcadores son los puntos con los que se representan la intersección de los dos ejes con cada dato, para hacerlos más visibles es posible cambiar la figura que se imprime en pantalla utilizando al parámetro "marker = nombre_marcador", los cuales pueden ser escogidos del listado que entrega la documentación oficial matplotlib.markers – Matplotlib 3.8.3 documentation

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'], marker = 's')
ax.plot(dias, temperaturas['Barcelona'], marker = 'o')

plt.show()
```

Líneas

Los gráficos que se han presentado, que utilizan líneas, utilizan por defecto el estilo de líneas continuas, pero dispone de una variedad de estilos que pueden ser modificados con el parámetro "linestyle = nombre_estilo" , los cuales pueden ser consultados en el listado disponible en la documentación oficial Linestyles – Matplotlib 3.8.3 documentation

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25], 'Valencia': [20, 21, 22, 23,20, 25, 24.5]}
ax.plot(dias, temperaturas['Madrid'], linestyle = 'dashed')
ax.plot(dias, temperaturas['Barcelona'], linestyle = 'dotted')
ax.plot(dias, temperaturas['Valencia'], linestyle = 'dashdot')

plt.show()
```

Títulos

Los títulos de los gráficos se pueden agregar y alinear a la figura, el atributo que se agrega es ax.set_title(titulo, loc=alineacion, fontdict=fuente), el primer parámetro define el texto del título, el segundo la ubicación y el tercero es un diccionario con el que se puede modificar la fuente como tamaño, grosor y color con fontsize, fontweight y color respectivamente.

```
fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25], 'Valencia': [20, 21, 22, 23,20, 25, 24.5]}
ax.plot(dias, temperaturas['Madrid'], linestyle = 'dashed')
ax.plot(dias, temperaturas['Barcelona'], linestyle = 'dotted')
ax.plot(dias, temperaturas['Valencia'], linestyle = 'dashdot')
ax.set_title("Evolución de la temperaatura diaria",
loc = "center",
fontdict = {
 'fontsize':14,
 'fontweight':'bold',
 'color':'tab:green'
}
)

plt.show()
```

Leyenda

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25], 'Valencia': [20, 21, 22, 23,20, 25, 24.5]}
ax.plot(dias, temperaturas['Madrid'], label='Madrid', linestyle = 'dashed')
ax.plot(dias, temperaturas['Barcelona'], label='Barcelona', linestyle = 'dotted')
ax.plot(dias, temperaturas['Valencia'], label='Valencia', linestyle = 'dashdot')

ax.set_title("Evolución de la temperaatura diaria",
loc = "center",
fontdict = {
  'fontsize':14,
```

```
    'fontweight':'bold',
    'color':'tab:green'
}
)

ax.legend(loc = 'upper right')

plt.show()
```

Ejes

- ax.set_xlabel(titulo) : Añade un título con el contenido de la cadena titulo al eje x de ax. Se puede personalizar la alineación y la fuente con los mismos parámetros que para el título principal.
- ax.set_ylabel(titulo) : Añade un título con el contenido de la cadena titulo al eje y de ax. Se puede personalizar la alineación y la fuente con los mismos parámetros que para el título principal.
- ax.set_xlim([limite-inferior, limite-superior]) : Establece los límites que se muestran en el eje x de ax.
- ax.set_ylim([limite-inferior, limite-superior]) : Establece los límites que se muestran en el eje y de ax.
- ax.set_xticks(marcas) : Dibuja marcas en el eje x de ax en las posiciones indicadas en la lista marcas.
- ax.set_yticks(marcas) : Dibuja marcas en el eje y de ax en las posiciones indicadas en la lista marcas.
- ax.set_xscale(escala) : Establece la escala del eje x de ax, donde el parámetro escala puede ser 'linear' (lineal) o 'log' (logarítmica).
- ax.set_yscale(escala) : Establece la escala del eje y de ax, donde el parámetro escala puede ser 'linear' (lineal) o 'log' (logarítmica).

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
dias = ['L', 'M', 'X', 'J', 'V', 'S', 'D']
temperaturas = {'Madrid':[28.5, 30.5, 31, 30, 28, 27.5, 30.5], 'Barcelona':[24.5,
25.5, 26.5, 25, 26.5, 24.5, 25]}
ax.plot(dias, temperaturas['Madrid'])
ax.plot(dias, temperaturas['Barcelona'])
ax.set_xlabel("Días", fontdict = {'fontsize':14, 'fontweight':'bold',
'color':'tab:blue'})
ax.set_ylabel("Temperatura ºC")
ax.set_ylim([20,35])
ax.set_yticks(range(20, 35))

plt.show()
```