
Universidad
Tecmilenio®

Universidad Tecmilenio

Visualización y programación en Python

Tema 13.
Graficación en Python

Introducción

Universidad Tecmilenio

Cuando analizamos bastante información y tenemos muchos datos, la mayoría de las veces, es para obtener solamente un número representativo o bien, nuevos datos que representen un nuevo reto de esta información. **Ahí es donde los gráficos pueden apoyarnos.**

En el lenguaje Python se tienen cada vez más librerías que apoyan esta tarea y que cada vez son más potentes y al mismo tiempo sencillas de usar para realizar estos gráficos de manera **práctica y provechosa.**

Universidad Tecmilenio

Graficando con Seaborn

Para presentaciones y publicaciones profesionales se usa la librería **Seaborn**, que está basada en **Matplotlib.**

Tabla 1. Ventajas y desventajas de la librería Seaborn

Ventajas	Desventajas
Fácil de usar.	Desarrollada sobre Matplotlib.
Sintaxis breve.	Poca personalización.
Gráficos para *dataframes* y matrices.	
Gráficos prediseñados (*out of the box*).	

Fuente: Mulla, R. (2022). ALL Python Data Visualization Libraries in 2022.*Kaggle.* Recuperado de: https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Universidad Tecmilenio

Seaborn cuenta con varios conjuntos de datos para practicar los gráficos, para saber cuáles son se usa este comando.

Tabla 2. Conjuntos de datos de la librería Seaborn

```
import seaborn as sb
print (sb.get_dataset_names())
```

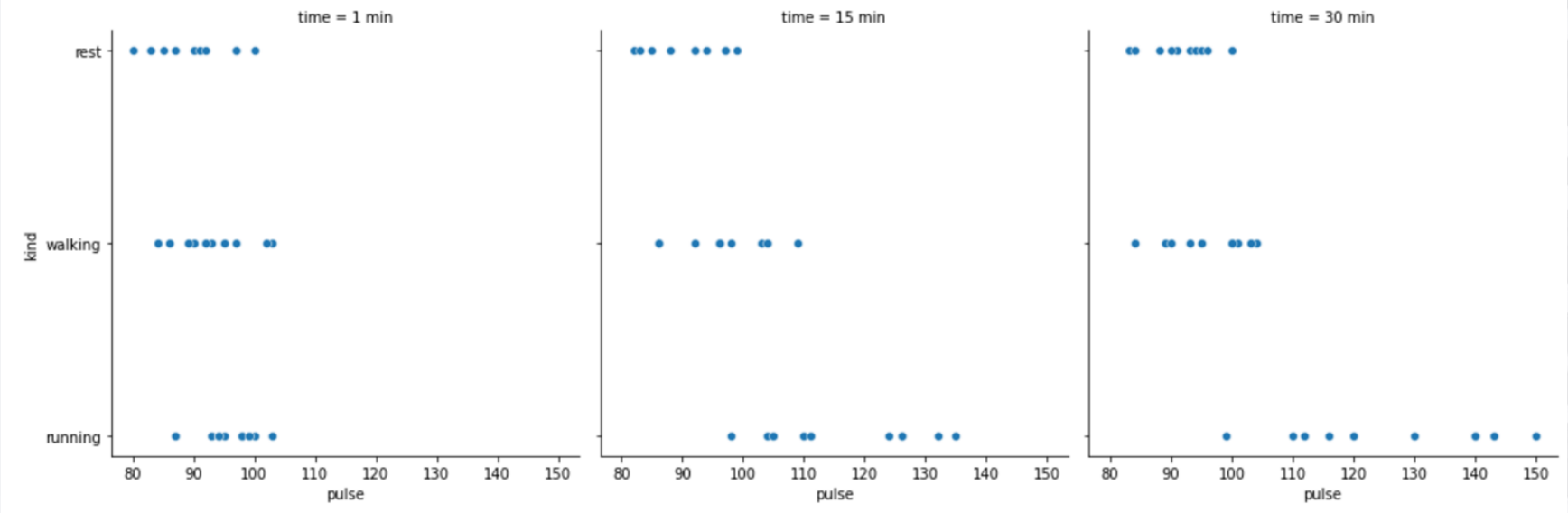
Fuente: Khatri, V. (2022). *7 Best Python Data Visualization Libraries*. *TechGeekBuzz.* Recuperado de https://www.techgeekbuzz.com/blog/best-python-data-visualization-libraries/

Tabla 3. Gráfico básico con la librería Seaborn

```
#Paso 0: Importar las librerías necesarias.
import seaborn as sb
import pandas as df
#Paso 1: Se cargan los datos en un dataframe.
df = sb.load_dataset('exercise')
#Paso 2: Primera exploración de datos revisando columnas y renglones.
print (df.head())
#Paso 3: Se define el dato que estará en eje x y y.
sb.relplot(
  data=df,
  x="pulse", y="kind", col="time"
)
```

Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium*. Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829

Figura 1. Gráfica de dispersión realizada con Seaborn



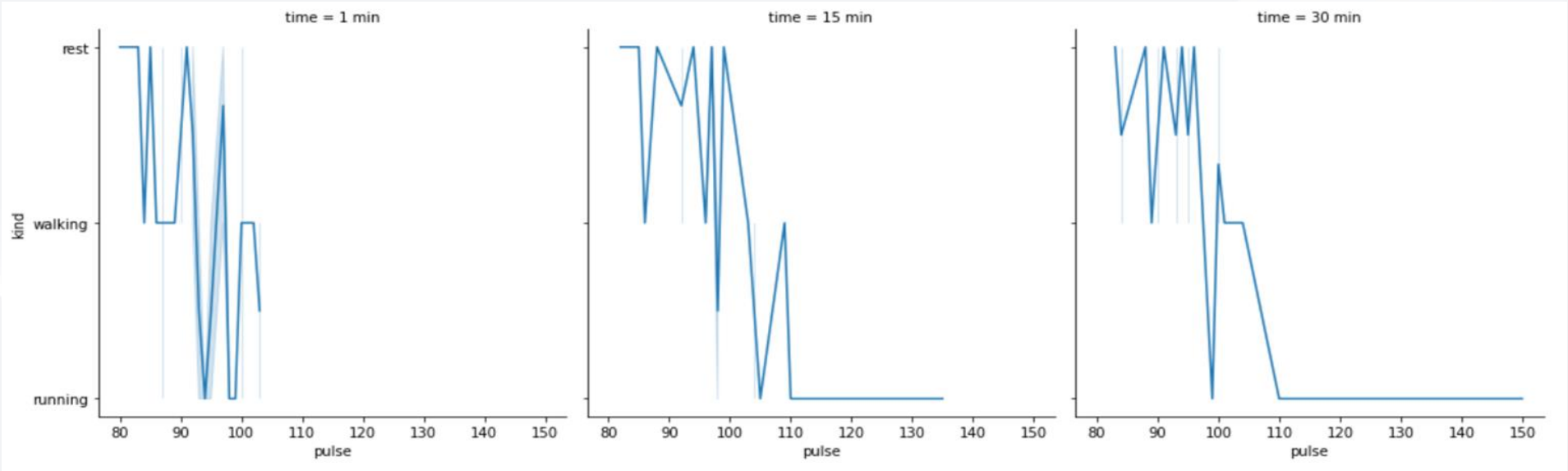
Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium*. Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829.

Este es el tipo de grafico básico en **Seaborn**, dado que no estamos eligiendo el tipo, se puede ver que separa por columnas de acuerdo al tiempo y las cuales nos hacen notar en este caso cómo varía el pulso de acuerdo con: si la persona corre, camina o está en reposo.

Tabla 4. Gráfico de líneas con la librería Seaborn

```
#Paso 3: Graficando con líneas.
sb.relplot(
  data=df,
  kind="line", x="pulse", y="kind", col="time"
)
```

Figura 2. Gráfica de líneas realizada con Seaborn.



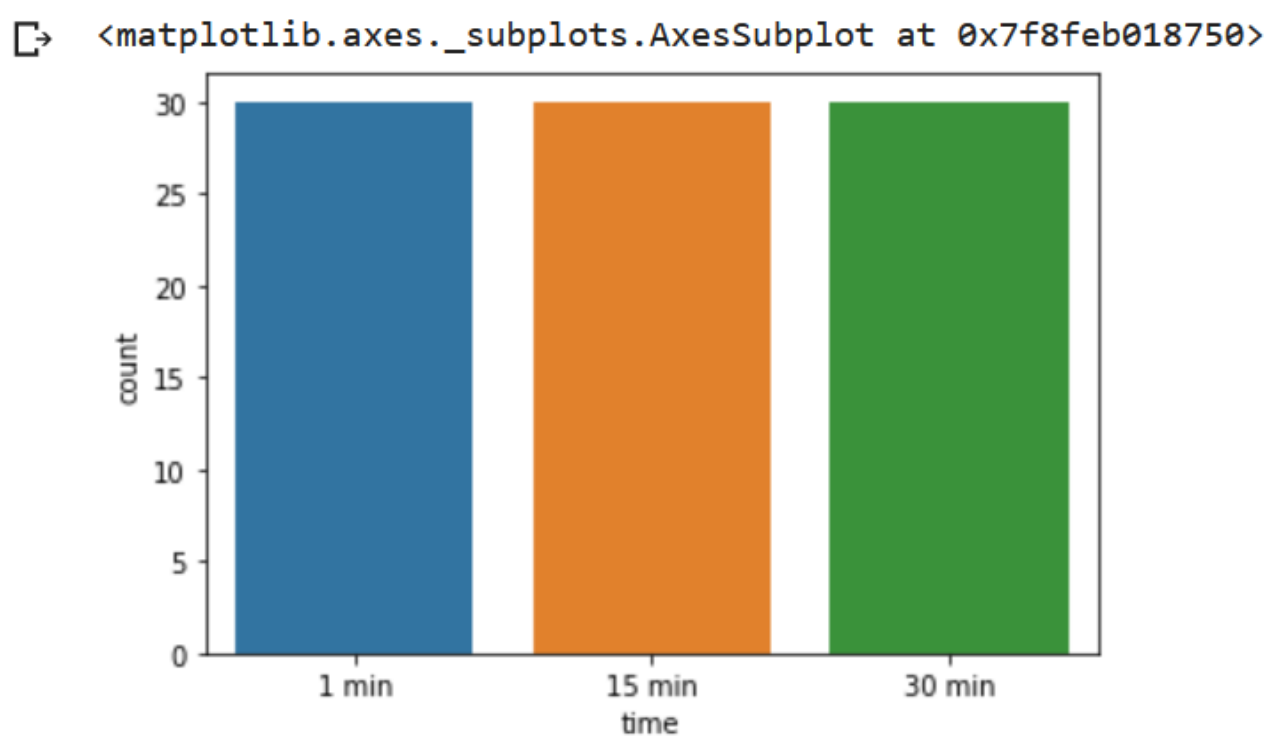
Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium.* Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829

Universidad Tecmilenio

Tabla 5. Gráfico de barras con la librería Seaborn

```
#Grafica de barras
sb.countplot(data=df, x='time')
```

Figura 3. Gráfica de barras realizada con Seaborn



Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium.* Recuperado de: https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829

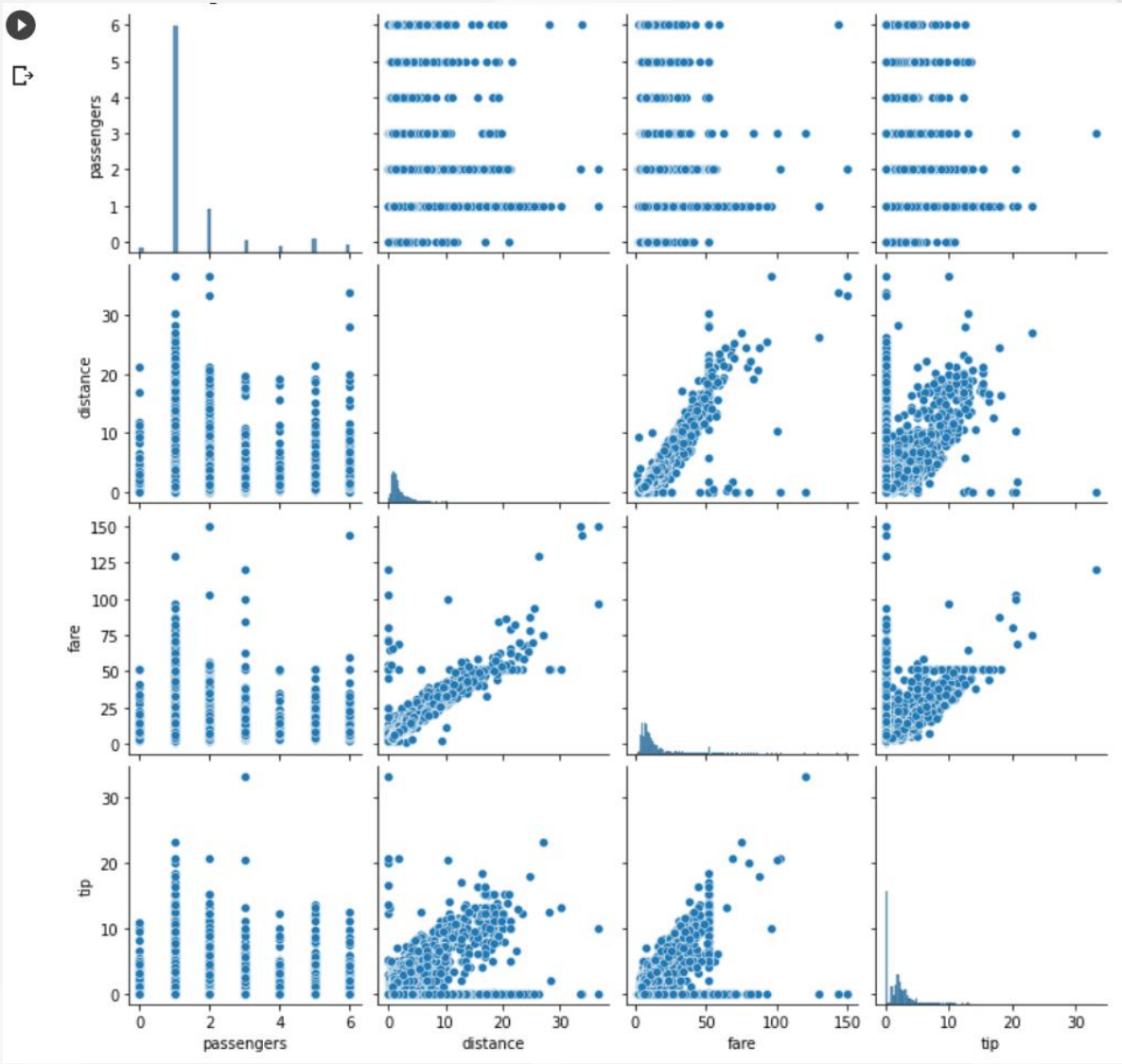
También hay gráficos que apoyan para **encontrar correlaciones entre datos**, es decir, datos con los que se encuentra relación de uno con el otro en el conjunto de datos dado y que puede darnos una pauta para plantear una o varias hipótesis.

Tabla 6. Gráfico de correlación con la librería Seaborn.

```
#Paso 1: Se cargan los datos en un dataframe.
df_a = sb.load_dataset('taxis')
#Paso 2: Primera exploración de datos revisando columnas y renglones
.
print (df_a.head())
sb.pairplot(df_a[["passengers", "distance", "fare", "tip"]])
```

Fuente: Pavan, N. (2022*).* Getting started with seaborn using Google colab notebook (Data viz). *Medium*. Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829.

Figura 4. Gráfica de correlación realizada con Seaborn.



Fuente: Pavan, N. (2022*).* Getting started with seaborn using Google colab notebook (Data viz). *Medium*. Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829.

Universidad Tecmilenio

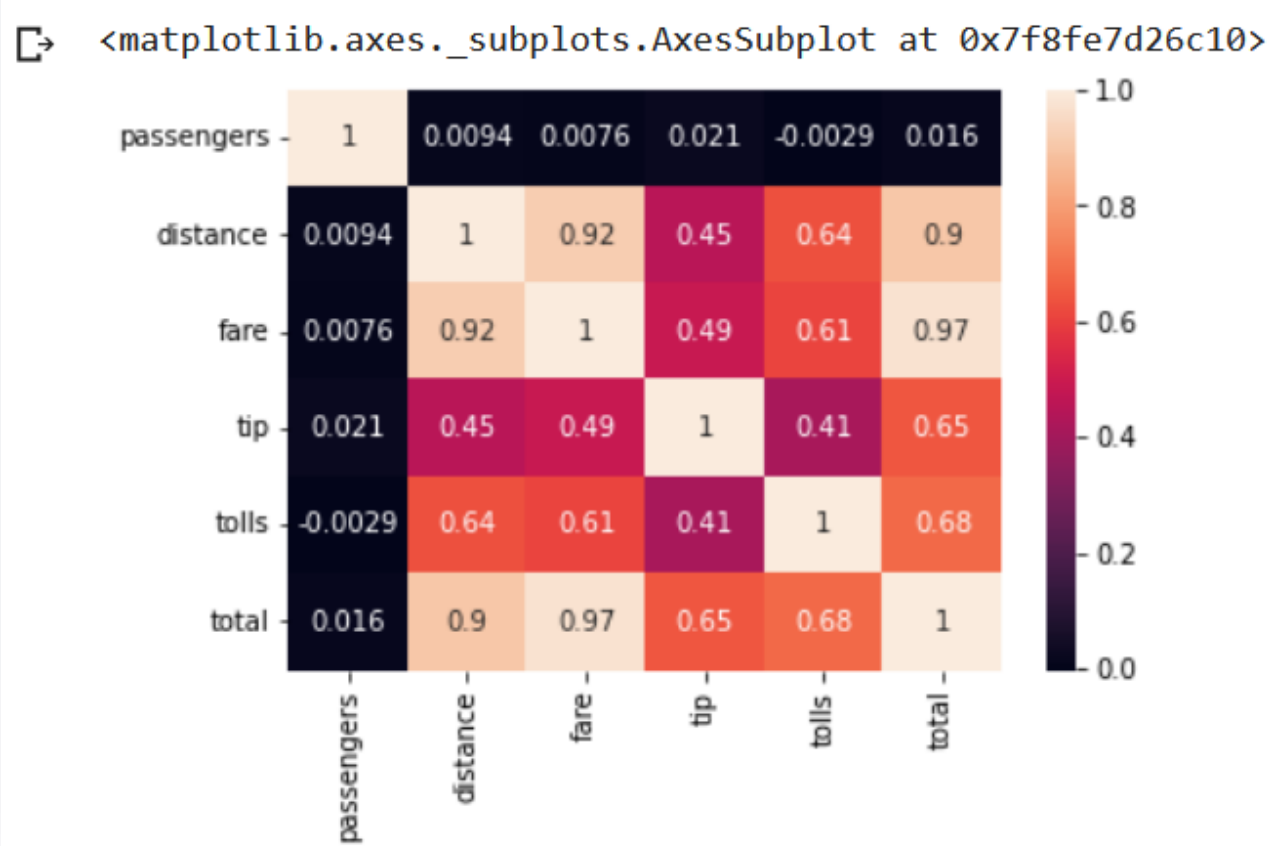
Existe un gráfico que tiene el valor que muestra a qué variable corresponde cada correlación para tener un dato numérico y además en colores que nos permite ver las que son más directas:

Tabla 7. Gráfico de coeficiente de correlación con la librería Seaborn

```
sb.heatmap(df_a.corr(), annot=True)
```

Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium.* Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829.

Figura 5. Gráfica de coeficiente de correlación realizada con Seaborn



Fuente: Pavan, N. (2022). Getting started with seaborn using Google colab notebook (Data viz). *Medium.* Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829.

Universidad Tecmilenio

Graficando con Plotnine

Plotnine es también una librería para visualización de datos, que le es familiar a quienes programan en el lenguaje R, muy usado por estadistas en análisis de información, está basado en ggplot. Una ventaja que tiene es que genera gráficos agradables visualmente.

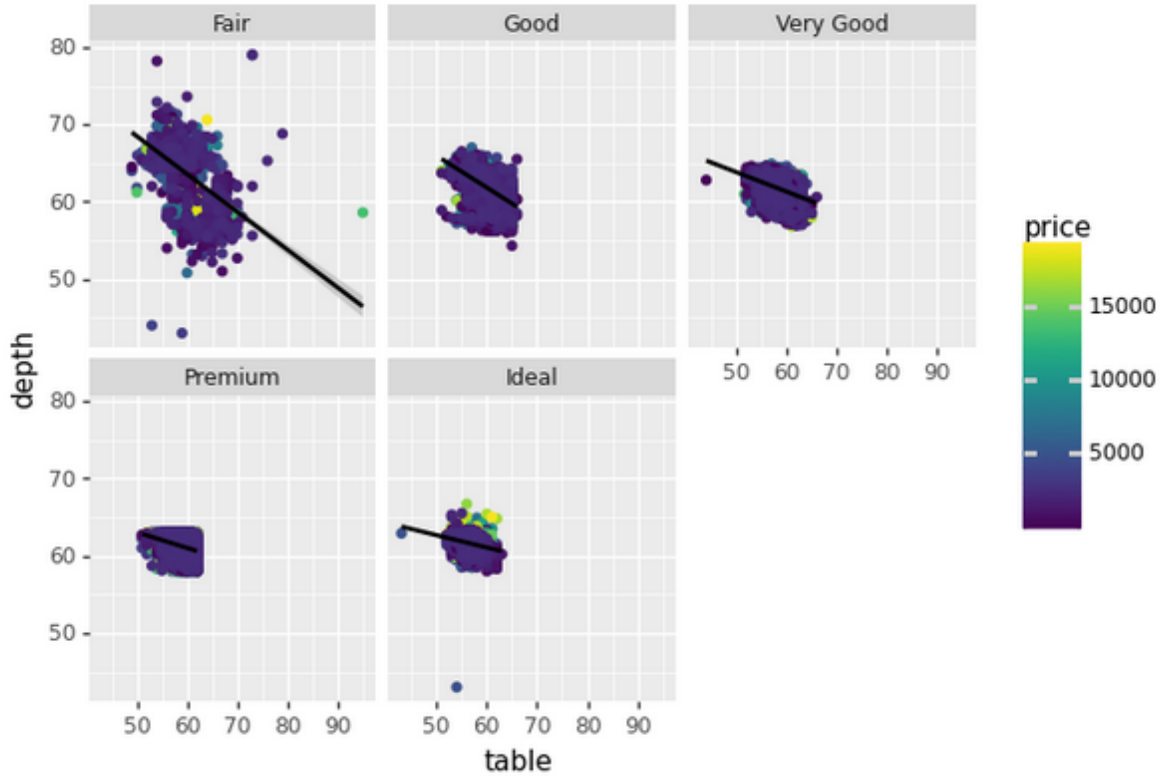
Tabla 8. Gráfico combinado con la librería Plotnine

```
#Paso 0: Importar a la librería.
from plotnine import ggplot, geom_point, aes, stat_smooth, facet_wrap
#Paso 1: Cargar los datos de un set de datos de prueba.
from plotnine.data import diamonds
#Paso 2: Explorar los datos para graficar.
print (diamonds.head())
#Paso 3: Se genera el gráfico, indicando campos.
(
    ggplot(diamonds, aes("table", "depth", color="price"))
    + geom_point()
    + stat_smooth(method="lm")
    + facet_wrap("~cut")
)
```

Fuente: Mulla, R. (2022). ALL Python Data Visualization Libraries in 2022. *Kaggle.* Recuperado de https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Figura 6. Gráfica combinada realizada con Plotnine

```
    carat      cut color clarity  depth  table  price     x     y     z
0    0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1    0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2    0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3    0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4    0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```




```
<ggplot: (8789762202397)>
```

Fuente: Mulla, R. (2022). ALL Python Data Visualization Libraries in 2022. *Kaggle.* Recuperado de https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Universidad Tecmilenio

Lo importante de las gráficas de **Plotnine** es esa idea visual que puede darnos la información y también tener un manejo ágil de la librería. Tiene bastantes gráficos útiles para los análisis que puedes explorar para ver si te son de utilidad.

En este notebook de Google colab puedes ver todo el código desarrollado en el tema:

https://colab.research.google.com/drive/1ZGOsWROybhCQpVLgWJA-a-TEN05ZO5gJ?usp=sharing

Explicación

Universidad Tecmilenio

Cuadro Comparativo

Objetivo: Realizar un cuadro comparativo donde se visualicen las diferencias de las librerías: Seaborn y Plotnine enfocadas a la creación de gráficas para la presentación de datos.

Instrucciones:

- Para realizar esta actividad es importante que consultes el contenido del tema.
- Con base en lo aprendido elabora un cuadro comparativo que incluya los conceptos más relevantes y significativos de la graficación interactiva.
- Incluye una reflexión sobre el impacto del uso de graficas en movimiento y ejemplifica brevemente con un caso de éxito en tu área laboral.
- Utiliza colores, elementos gráficos y tipografías a través de una plataforma digital que permita la mayor comprensión de lo expuesto.
- Máximo 2 cuartillas.

Universidad Tecmilenio

Seaborn y Plotnine son **dos librerías que permiten graficar la información de manera más atractiva con menos líneas de código.**

¿Qué grafico te parece más atractivo para qué tipos de hipótesis? ¿Cuántas librerías hay en Python que faciliten el trabajo de gráficos? ¿Cuándo valor aportan estas gráficas a un informe de resultados? ¿Qué otro gráfico o funcionalidad de ellos agregarías?

Referencias

Mulla, R. (2022). ALL Python Data Visualization Libraries in 2022. *Kaggle*. Recuperado de https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Pavan, N. (2022). *Getting started with seaborn using Google colab notebook (Data viz)*. Recuperado de https://medium.com/@pavansaish/getting-started-with-seaborn-library-using-google-colaboratory-82e62d826829

Referencias

Universidad Tecmilenio

Visualización y programación en Python

Tema 14.
Graficación en Python II.

Introducción

En la actualidad es común ver en las comunicaciones de redes sociales, no solo imágenes o iconos fijos sino que están tomando auge los que son animados o bien tienen movimiento que permite visualizar lo que se quiere expresar, haciéndolos más atractivos.

La visualización abre las posibilidades de análisis para poder explorarlos desde varios ángulos y así plantear hipótesis estratégicas e interesantes para la empresa.

Universidad Tecmilenio

Con pocas líneas de código **Bokeh** (Desenfoque) se permite crear visualizaciones en Python y que además son interactivas basadas en JavaScript por lo que se pueden mostrar en un navegador web.

Esto consta de dos pasos:

- Seleccionar los bloques de Bokeh para generar la visualización.
- Personalizar estos bloques de construcción para que satisfagan las necesidades.

La librería Bokeh en automático generará el código necesario en JavaScript y HTML para mostrarlos en un navegador web.

Universidad Tecmilenio

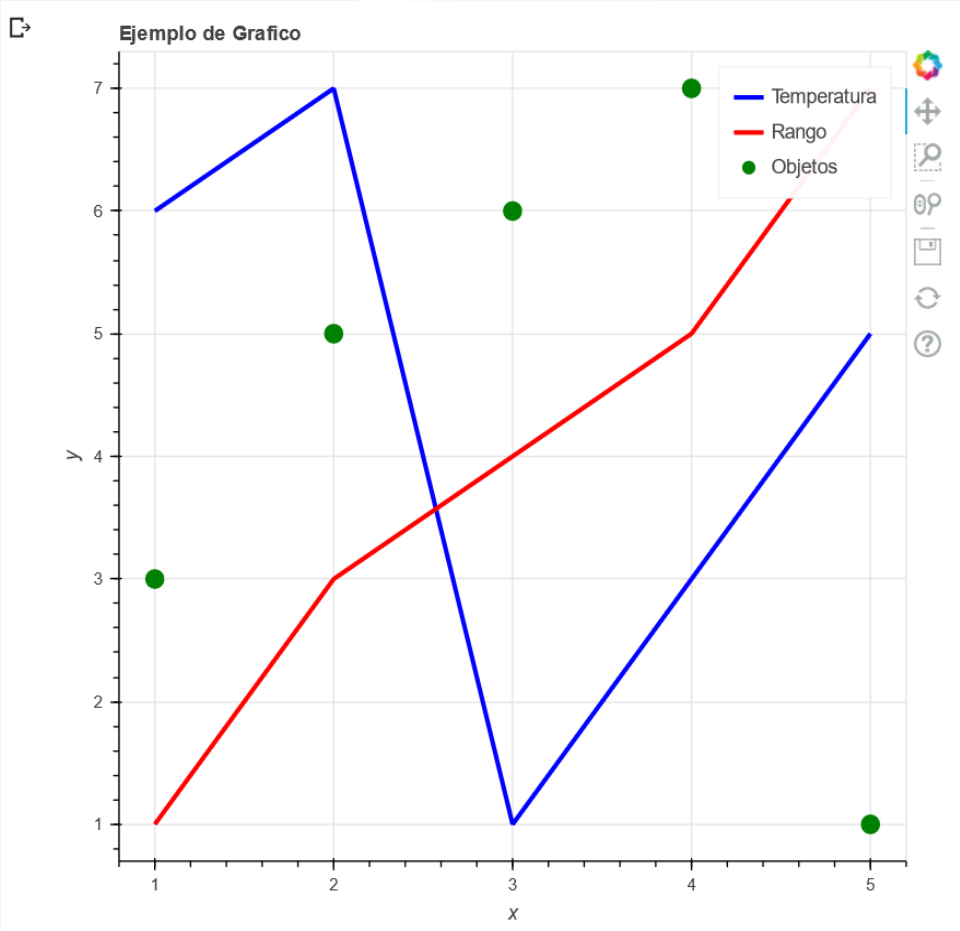
Tabla 1. Gráfico combinado e interactivo con la librería Bokeh.

```
#Paso 0: Se cargan las librerías
import bokeh.plotting
import bokeh.io
#Paso 1: Se establece la salida en el notebook
bokeh.io.output_notebook()

#Paso 2: Se tienen los datos
x = [1, 2, 3, 4, 5]
y1 = [6, 7, 1, 3, 5]
y2 = [1, 3, 4, 5, 7]
y3 = [3, 5, 6, 7, 1]

#Paso 3: Crear el gráfico con título y etiqueta en ejes
p = bokeh.plotting.figure(title="Ejemplo de Grafico", x_axis_label="x", y_axis_label="y")

#Paso 4: Añadir las representaciones a graficar
p.line(x, y1, legend_label="Temperatura", color="blue", line_width=3)
p.line(x, y2, legend_label="Rango", color="red", line_width=3)
p.circle(x, y3, legend_label="Objetos", color="green", size=12)

#Paso 5: Mostrar el gráfico
bokeh.plotting.show(p)
```

Universidad Tecmilenio

Es importante realizar esta práctica en el notebook de colab de Google para poder ver su funcionalidad e interacción, esta es una imagen alejando el gráfico, se puede acercar o alejar según se desee para observar los datos.

Figura 1. Gráfica original combinada realizada con Bookeh.



Universidad Tecmilenio

Bokeh también permite hacer múltiples gráficos que interactúan con los mismos datos:

Tabla 2. Gráfico doble e interactivo con la librería Bokeh.

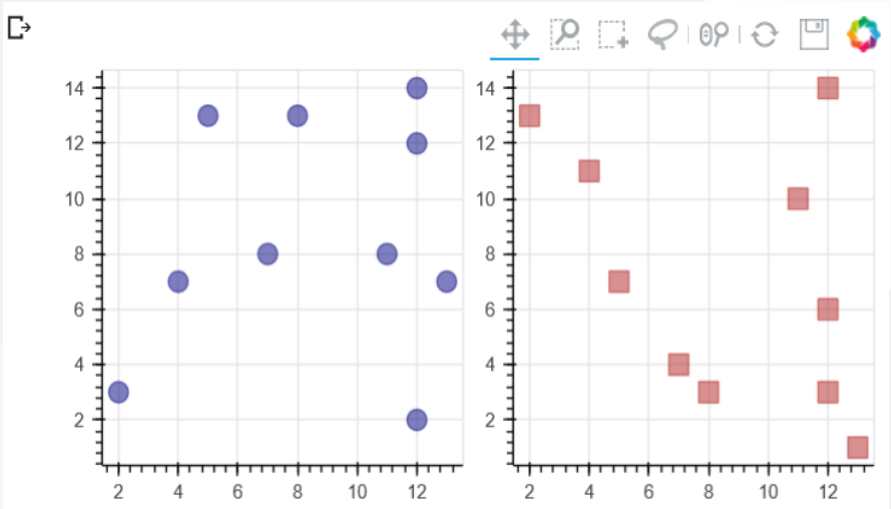
```
#Paso 0: Se cargan las librerías
import bokeh.layouts
import bokeh.plotting
import numpy as np

#Paso 1: Se establece la salida en el notebook
bokeh.io.output_notebook()

#Paso 2: Se generan los datos con numpy
x = np.random.randint(low=1, high=15, size=10)
y = np.random.randint(low=1, high=15, size=10)
z = np.random.randint(low=1, high=15, size=10)

#Paso 3: Define herramientas en el gráfico
H = "pan,wheel_zoom,box_zoom,reset,save,box_select,lasso_select"

#Paso 4: Se crea el gráfico y su representación
g1 = bokeh.plotting.figure(tools=H, width=250, plot_height=250, title=None)
g1.circle(x, y, size=12, color="navy", alpha=0.5)

#Paso 5: Se crea un segundo gráfico y su representación
g2 = bokeh.plotting.figure(tools=H, width=250, height=250, x_range=g1.x_range,
 y_range=g1.y_range, title=None)
g2.square(x, z, size=12, color="firebrick", alpha=0.5)

#Paso 6: Se agregan ambos gráficos en la plantilla
p = bokeh.layouts.gridplot([[g1, g2]])

#Paso 7: Mostrar el gráfico
bokeh.plotting.show(p)
```

La diferencia que se puede observar de este gráfico doble, es que repite la instrucción **figure** y se puede seguir agregando figuras tanto como sea necesario para llevar a cabo el análisis.

En el segundo gráfico se usaron estos parámetros: **x_range y y_range** en este caso es para que los rangos de los ejes de ambos gráficos sean iguales.

Otra funcionalidad que es importante notar es: **bokeh.layouts** aquí se usa el **gridplot**; sin embargo se pudieran elegir otros tipos como: column y row.

Figura 3. Gráfica doble realizada con Bookeh.



En la siguiente gráfica se puede observar un grado más alto de **visualización y personalización** de la información

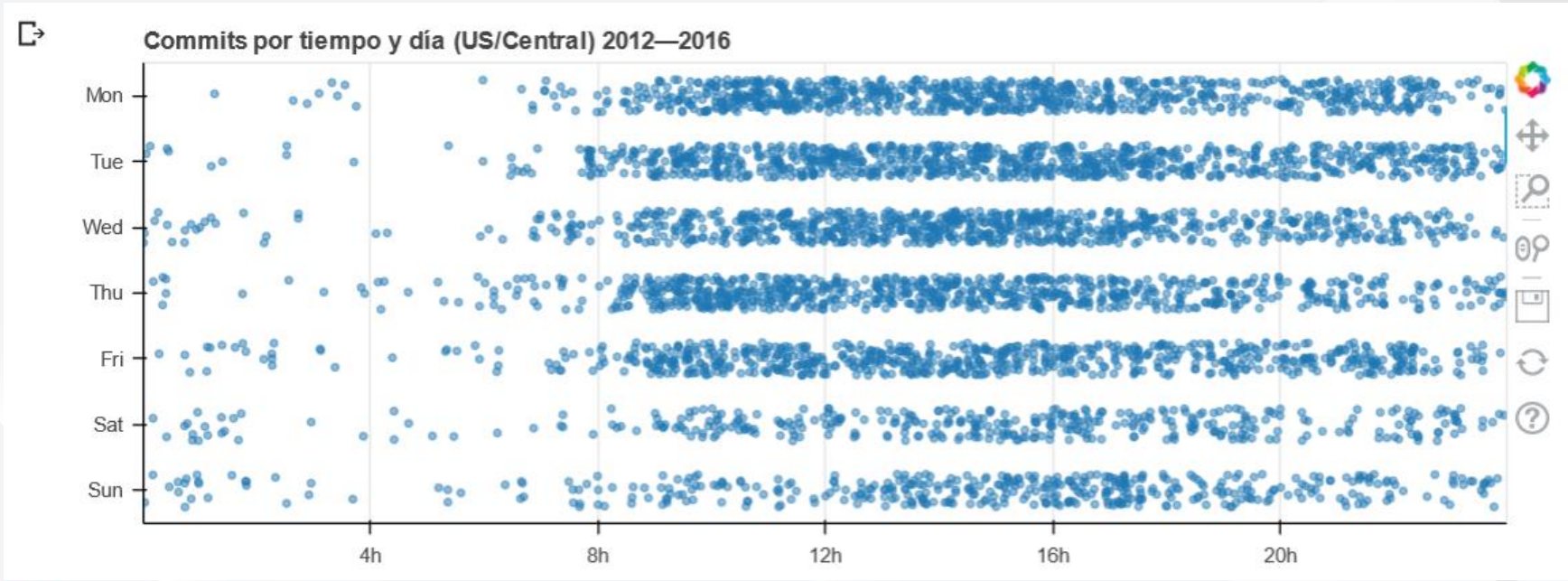
Tabla 3. Gráfico con personalización y detalle realizado con la librería Bokeh.

```
#Paso 0: Se importan las librerías
from bokeh.transform import jitter
from bokeh.sampledata.commits import datos

#Paso 1: Se establece la salida en el notebook
bokeh.io.output_notebook()
#Tambien una salida a un archivo html
bokeh.io.output_file("barras.html")

#Paso 2: Define la división de los datos y fuente
dias = ['Sun', 'Sat', 'Fri', 'Thu', 'Wed', 'Tue', 'Mon']
fuente = bokeh.plotting.ColumnDataSource(datos)

#Paso 3: Define el gráfico que se usará
grafico = bokeh.plotting.figure(plot_width=900, plot_height=300, y_range=di
as, x_axis_type='datetime',
           title="Commits por tiempo y día (US/Central) 2012—2016")

#Paso 4: Define personalización para el gráfico
grafico.circle(x='time', y=jitter('day', width=0.5, range=p.y_range),  sour
ce=fuente, alpha=0.5)
grafico.xaxis[0].formatter.days = ['%Hh']
grafico.x_range.range_padding = 0
grafico.ygrid.grid_line_color = None

#Paso 5: Se muestra el gráfico
bokeh.plotting.show(grafico)
```

Fuente: Anand, S. (2020). *Data Visualization with Bokeh*. Recuperado de https://www.kaggle.com/code/saurav9786/data-visualization-with-bokeh

En este ejemplo se puede ver la diferencia en el paso 4, donde se personalizan más detalles con cuatro líneas de código indicando sobre todo **circle y xaxis** para definir cómo queremos que muestre los datos y también los formatos que se tendrán en el eje x.

Figura 4. Gráfica personalizada realizada con Bookeh.



Universidad Tecmilenio

Tabla 4. Código para generar gráfico animado en archivo gif.

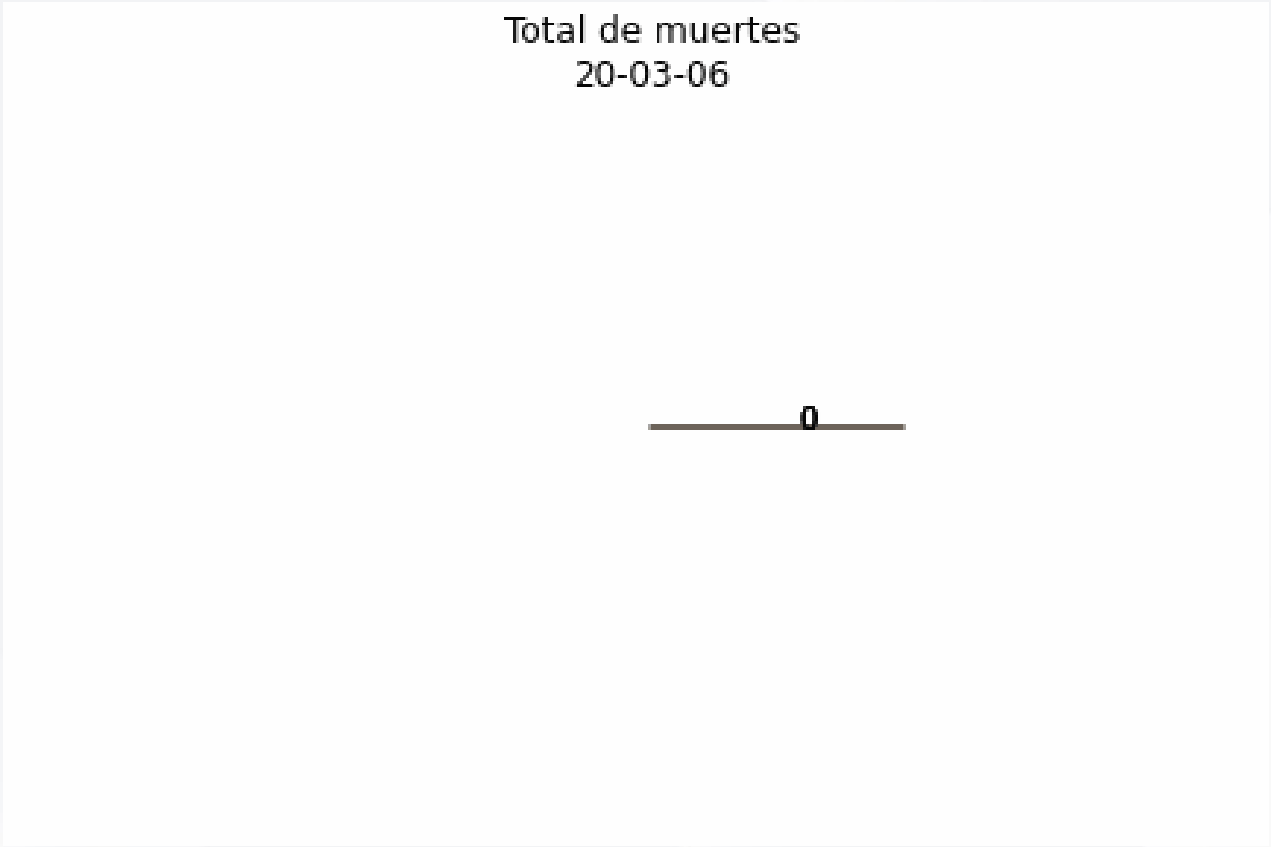
```
#Paso 0: Cargar las librerias

from matplotlib import animation, rc

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd


#Paso 1: Se toman los datos de un sitio en un dataframe

url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-

19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_

deaths_global.csv'

df = pd.read_csv(url, delimiter=',', header='infer')

#print (df['Country/Region'].unique)


#Paso 2: Elegimos la información a graficar

print('Datos originales:',df.head)

df_interest = df.loc[df['Country/Region'].isin(['Mexico', 'Brazil', 'Argent

ina', 'Peru']) & df['Province/State'].isna()]

df_interest.rename(index=lambda x: df_interest.at[x, 'Country/Region'], inp

lace=True)

print('Datos que interesan:',df interest.head)
```

```
df1 = df_interest.transpose()

df1 = df1.drop(['Province/State', 'Country/Region', 'Lat', 'Long'])

df1 = df1.loc[(df1 != 0).any(1)]

df1.index = pd.to_datetime(df1.index)

print('Datos preparados:',df1.head)


#Paso 3: Se define la figura y los titulos

fig,ax = plt.subplots()

explode=[0.01,0.01,0.01,0.01] #Destaca cada rebanada del pastel

def getmepie(i):

  def absolute_value(val): #Convierte porcentaje a numero

    a  = np.round(val/100.*df1.head(i).max().sum(), 0)

    return int(a)

  ax.clear()

  plot = df1.head(i).max().plot.pie(y=df1.columns,autopct=absolute_value, l

abel='',explode = explode, shadow = True)

  plot.set_title('Total de muertes\n' + str(df1.index[min( i, len(df1.index

)-1 )].strftime('%y-%m-%d')), fontsize=12)
```

```
#Paso 4: Se define la animación

anim = animation.FuncAnimation(fig,getmepie,interval=200)



#Paso 5: Se obtiene la gráfica en un archivo gif

from google.colab import drive

drive.mount('/content/gdrive/', force_remount=True)

writergif = animation.PillowWriter(fps=5)

anim.save(r'/content/gdrive/My Drive/GraficoAnimado.gif',writergif)



#Paso 6: Se muestra en el notebook la animación

rc('animation', html='jshtml')

anim
```

Fuente: Costas, A. (2020). *Learn How to Create Animated Graphs in Python*. Recuperado de: https://towardsdatascience.com/learn-how-to-create-animated-graphs-in-python-fce780421afe

Archivo gif de gráfico animado.



Universidad Tecmilenio

Existen otras maneras de generar una animación como pueden ser ir tomando fotos (snaphots) de cada gráfica y uniéndolas para generar el movimiento. Sin embargo, hoy sabemos que es una manera más rudimentaria pero funcional.

En este notebook de Google colab puedes ver todo el código desarrollado en el tema:

https://colab.research.google.com/drive/189wpAeqUCXLkn4AYxBZ4xD-KcOZg0UqO?usp=sharing

Cuadro Sinóptico

Objetivo: Realizar un cuadro sinóptico donde se apliquen los conceptos aprendidos.

Instrucciones:

- Para realizar esta actividad es importante que consultes el contenido del tema.
- Con base en lo aprendido elabora un código donde el programa solicite el ingreso de dos números diferentes y los compare.
- Incluye los operadores lógicos vistos en las lecciones anteriores y los ciclos para el correcto funcionamiento del programa y realiza un diagrama de flujo que explique la lógica que seguirá tu código.
- Agrega comentarios donde se explique el proceso que va siguiendo tu código.
- Agrega capturas del proceso llevado a cabo y el funcionamiento final.

La versatilidad al mostrar los hallazgos del análisis de la información es clave para saber qué datos importantes son los que apoyan o no a la hipótesis que tenemos.

Mostrar la información mediante animaciones permite ver los cambios en la información a trevés del tiempo y así darle propósito al querer explicarla.

Los gráficos son un gran apoyo para el análisis de datos. ¿Qué información te gustaría explorar con estas librerías?¿qué otras ventajas identificas al presentar con interacción o animación la información?¿has observado algún análisis o artículo haciendo uso de la animación para presentar los datos?

Referencias

Anand, S. (2020). *Data Visualization with Bokeh*. Recuperado de https://www.kaggle.com/code/saurav9786/data-visualization-with-bokeh

Costas, A. (2020). *Learn How to Create Animated Graphs in Python*. Recuperado de: https://towardsdatascience.com/learn-how-to-create-animated-graphs-in-python-fce780421afe