cuaderno_graficos

March 2, 2023

[ ]:

[133]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Matplotlib es una biblioteca de visualización de datos muy potente y flexible. Proporciona una amplia variedad de gráficos, desde gráficos de líneas y de barras hasta diagramas de dispersión, histogramas, gráficos de contorno y mucho más. Matplotlib es ampliamente utilizado en la comunidad científica y académica, ya que permite a los usuarios personalizar cada aspecto de un gráfico, desde la colocación de las etiquetas del eje hasta la apariencia de las líneas y la fuente utilizada en el texto.

Las principales ventajas de Matplotlib son su flexibilidad y potencia, lo que significa que es capaz de crear una amplia variedad de gráficos. Además, como es una biblioteca ampliamente utilizada, hay una gran cantidad de documentación y ejemplos disponibles en línea. Otra ventaja es que es fácil de integrar con otras bibliotecas de análisis de datos en Python, como Pandas y NumPy.

Sin embargo, la principal desventaja de Matplotlib es que puede ser bastante complejo y requiere una curva de aprendizaje bastante empinada para utilizarlo de forma efectiva. Además, aunque Matplotlib es muy flexible, esto significa que puede ser difícil obtener gráficos con un aspecto estético agradable de forma rápida y sencilla. También puede requerir una gran cantidad de código para crear gráficos complejos.

Seaborn, por otro lado, es una biblioteca de visualización de datos de nivel superior que utiliza Matplotlib en segundo plano. Seaborn proporciona una interfaz de usuario más amigable y fácil de usar que Matplotlib, y permite crear gráficos complejos con muy poco código. Seaborn proporciona una amplia variedad de gráficos, desde gráficos de barras y de líneas hasta diagramas de violín, mapas de calor y diagramas de dispersión.

Las principales ventajas de Seaborn son su facilidad de uso y su capacidad para crear gráficos complejos con muy poco código. Seaborn también está diseñado para trabajar con DataFrames de Pandas, lo que significa que es fácil de integrar con otras bibliotecas de análisis de datos en Python. También proporciona muchas opciones para personalizar la apariencia de los gráficos.

La principal desventaja de Seaborn es que no es tan flexible como Matplotlib. Esto significa que, aunque Seaborn proporciona una amplia variedad de gráficos, es posible que no puedas crear exactamente el gráfico que deseas. Además, Seaborn no proporciona la misma cantidad de documentación

y ejemplos disponibles en línea que Matplotlib, lo que puede dificultar la resolución de problemas.

En resumen, si estás buscando una biblioteca de visualización de datos muy flexible y potente, Matplotlib es una buena opción. Si buscas una biblioteca más fácil de usar y que te permita crear gráficos complejos con muy poco código, Seaborn es una buena opción. En general, la elección entre Matplotlib y Seaborn dependerá de tus necesidades específicas y de tu nivel de experiencia en programación.

Bokeh y Plotly son dos librerías de visualización interactiva que permiten crear gráficos interactivos, como por ejemplo gráficos de líneas, de barras, de dispersión, mapas de calor, entre otros.

Algunas de las ventajas de Bokeh son:

Permite crear gráficos interactivos que pueden ser explorados y manipulados por los usuarios. Es fácil de usar y se integra bien con otras librerías de Python como Pandas y NumPy. Ofrece una amplia variedad de herramientas interactivas para zoom, panning y selección de datos. Puede ser utilizado tanto en notebooks de Jupyter como en aplicaciones web. Algunas de las desventajas de Bokeh son:

Puede ser lento al crear gráficos muy complejos o con grandes conjuntos de datos. La documentación puede ser limitada y los ejemplos pueden ser difíciles de entender para usuarios principiantes. Por otro lado, algunas de las ventajas de Plotly son:

Permite crear gráficos interactivos que pueden ser explorados y manipulados por los usuarios. Ofrece una amplia variedad de tipos de gráficos y opciones de personalización. Ofrece una gran cantidad de ejemplos y tutoriales que pueden ayudar a los usuarios a aprender a utilizar la librería. Algunas de las desventajas de Plotly son:

Requiere una cuenta en línea para algunas funciones, lo que puede ser un obstáculo para algunos usuarios. La documentación puede ser limitada y los ejemplos pueden ser difíciles de entender para usuarios principiantes. En general, tanto Bokeh como Plotly son librerías potentes y útiles para visualización de datos interactiva, y la elección de una u otra dependerá de las necesidades específicas del usuario y de su nivel de experiencia con Python.

1 1. Matplotlib

[2]:
```
data = {'name': ['Alice', 'Bob', 'Charlie', 'Dave', 'Eve', 'adrian', 'david',␣
↪'carol'],
        'age': [25, 35, 42, 18, 29, 24, 24, 20],
        'city': ['New York', 'London', 'Paris', 'Tokyo', 'Sydney', 'New York',␣
↪'London', 'New York']}

df = pd.DataFrame(data)
df
```

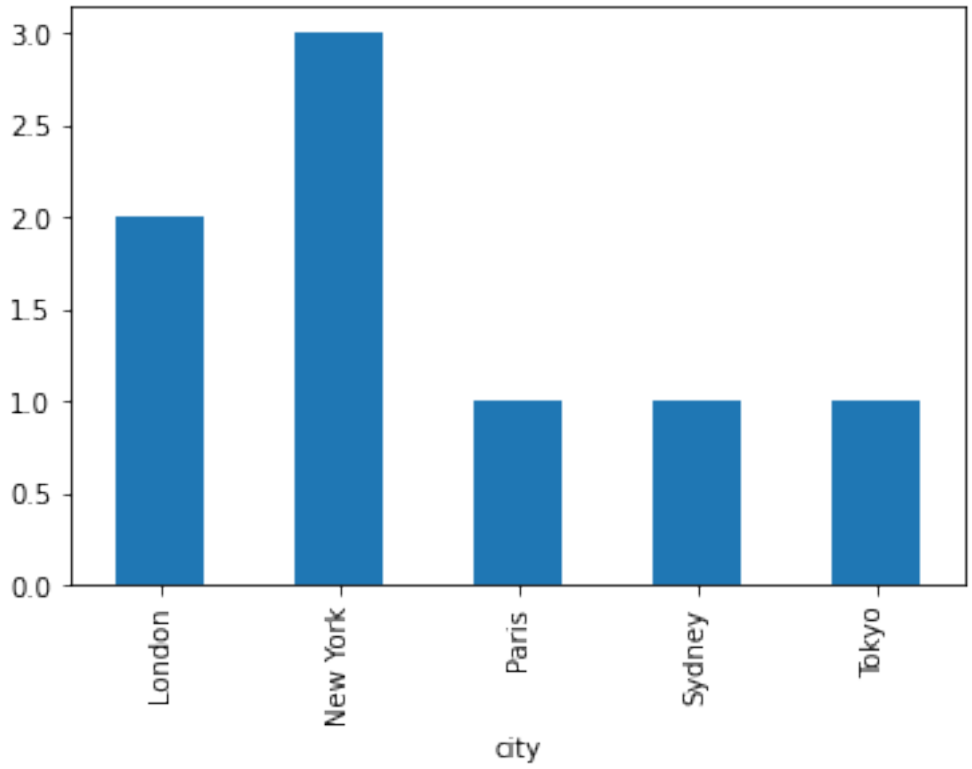
[2]:

	name	age	city
0	Alice	25	New York
1	Bob	35	London
2	Charlie	42	Paris
3	Dave	18	Tokyo

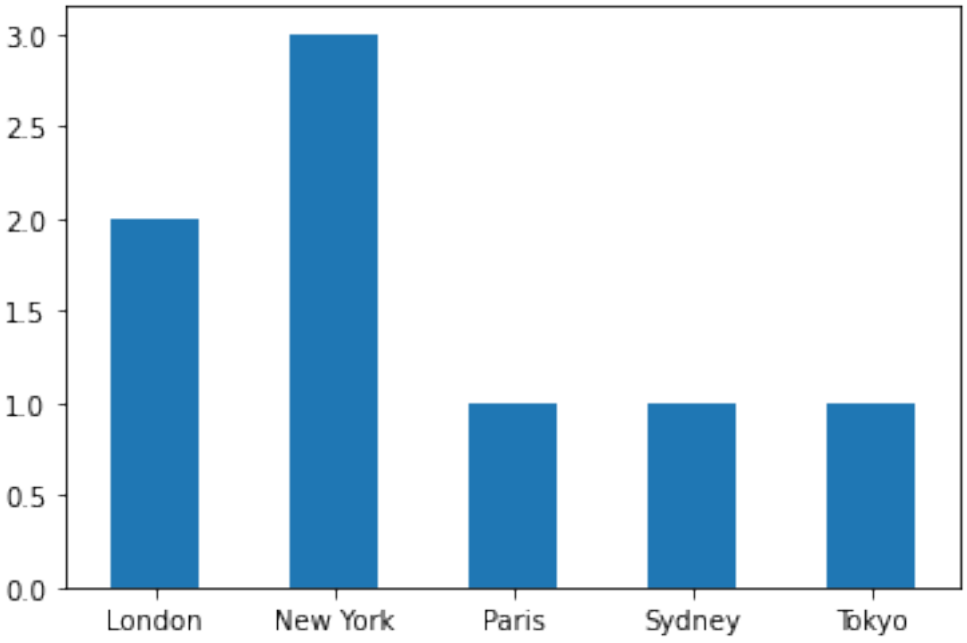
```
4      Eve   29    Sydney
5   adrian   24  New York
6    david   24    London
7    carol   20  New York
```

1.1 Grafico de barras

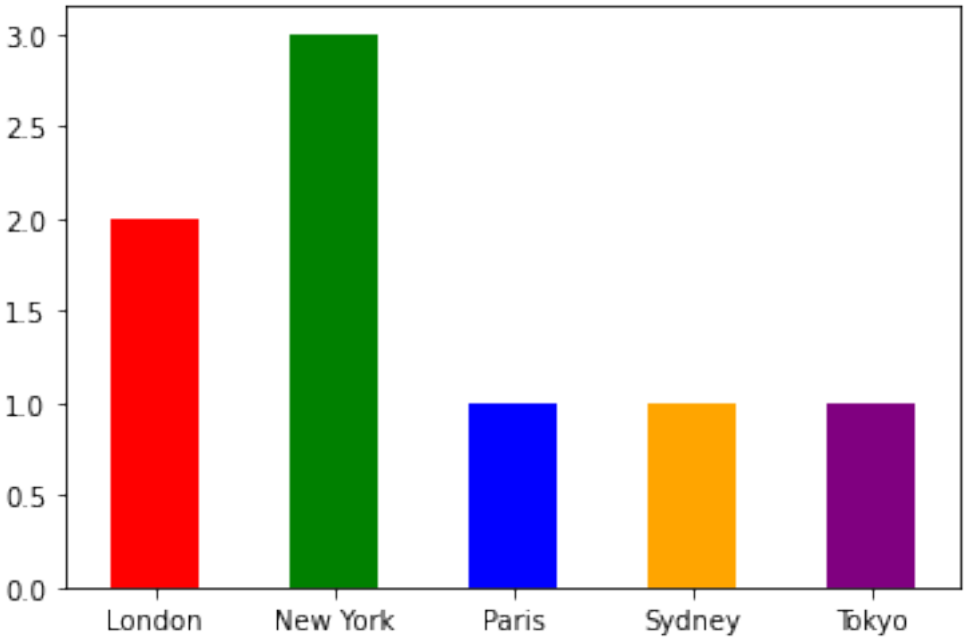
[3]:
```
#Representa cuantas veces aparece cada ciudad
df.groupby('city').size().plot(kind='bar')
plt.show()
```



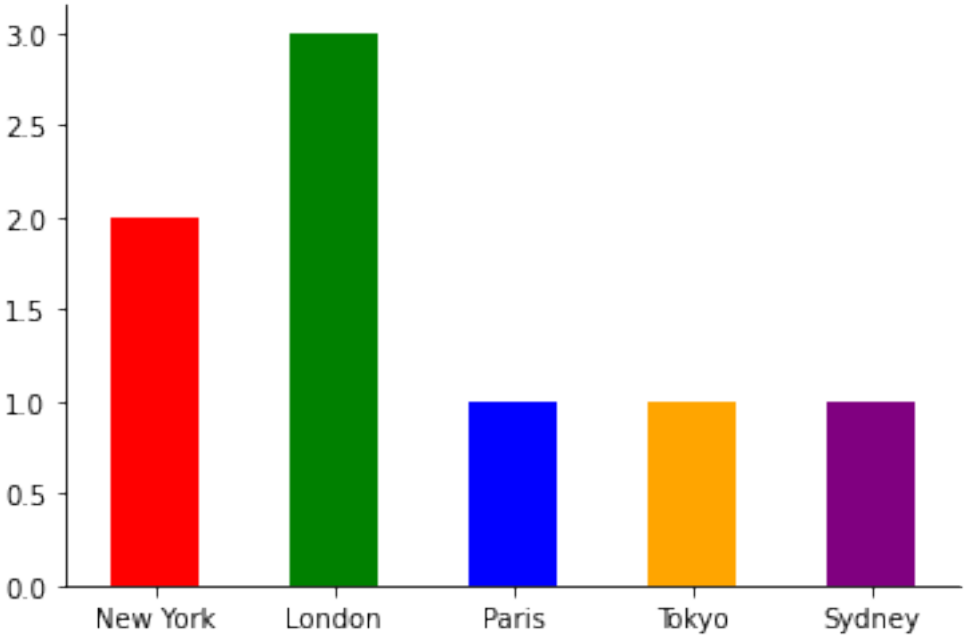
[4]:
```
# quitamos el nombre del eje x y cambiamos horientacion a eje x
df.groupby('city').size().plot(kind='bar')
plt.xticks(rotation=0)
plt.xlabel('')
plt.show()
```



[65]:
```
colors = ['red', 'green', 'blue', 'orange', 'purple']
df.groupby('city').size().plot(kind='bar', color=colors) # poniendo 'tab10'␣
 ↪debería ser una selccion predefinidad
plt.xticks(rotation=0)
plt.xlabel('')
plt.show()
```

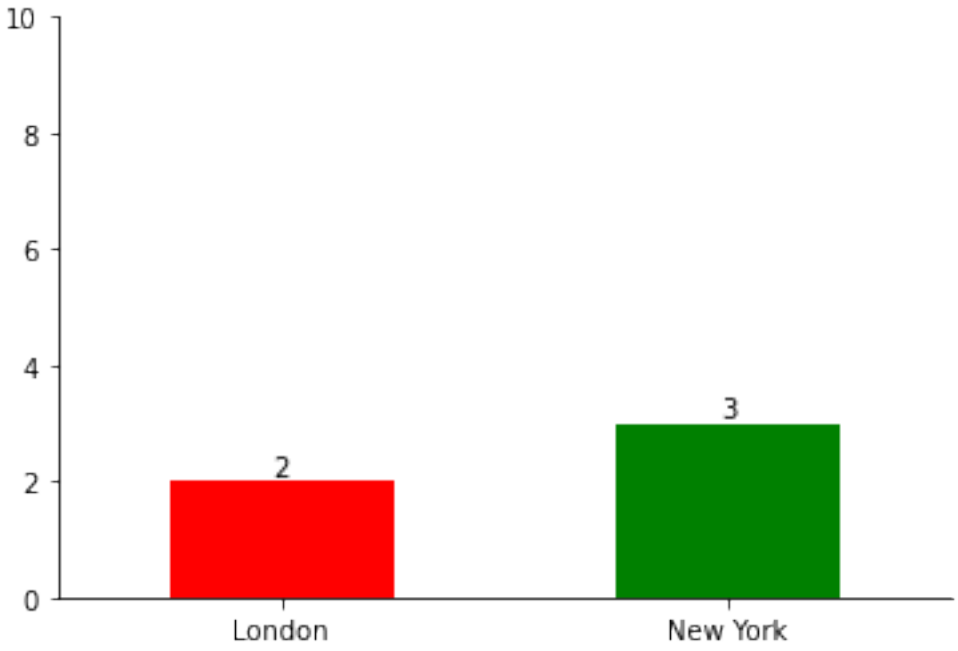


[67]:
```
# Lo hacemos con el formato correcto

fig, ax = plt.subplots()
df.groupby('city').size().plot(kind='bar', color=colors, ax=ax)
ax.set_xticklabels(df['city'], rotation=0)
ax.set_xlabel('')
ax.spines['right'].set_visible(False) # quitar marco derecheo
ax.spines['top'].set_visible(False) # quitar marco de arriba
plt.show()
```



[18]:
```
# añadamimos labels, quitamos los recuadros y ampliamos escala. Ademas solo␣
↪valores mayor que 1

grouped_data = df.groupby('city').size().loc[lambda x: x > 1]
ax = grouped_data.plot(kind='bar', color=colors)
ax.spines['right'].set_visible(False) # quitar marco derecheo
ax.spines['top'].set_visible(False) # quitar marco de arriba
plt.xticks(rotation=0) # rotación eje x
plt.xlabel('') # nombre del eje x
plt.ylim(0, 10) # escala eje y

# Agregar etiquetas de valor encima de cada barra
for i, v in enumerate(grouped_data.values):
```

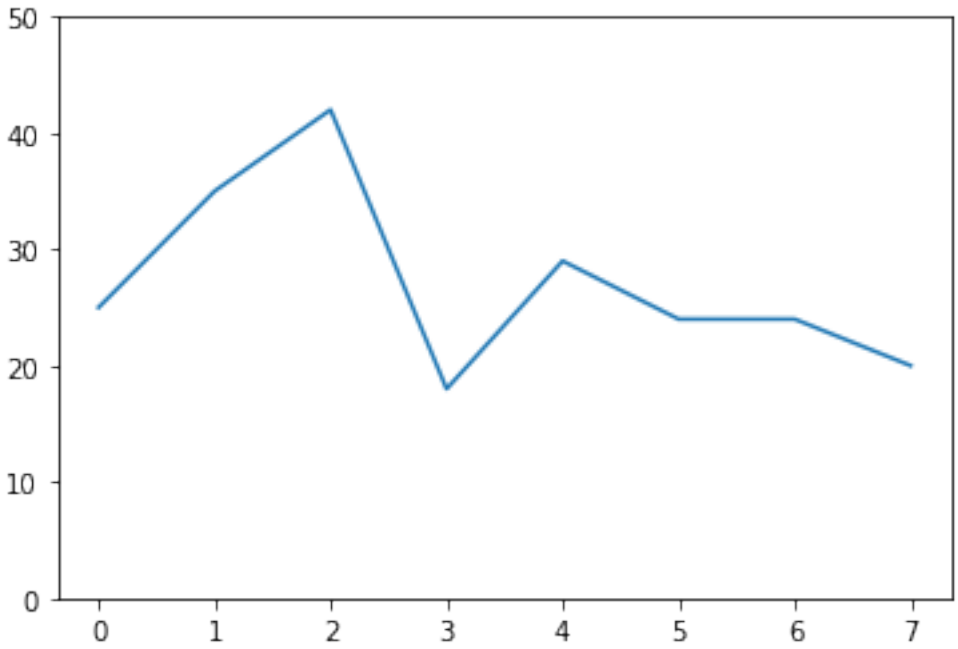
```
    plt.text(i, v + 0.1, str(v), color='black', ha='center')

plt.show()
```



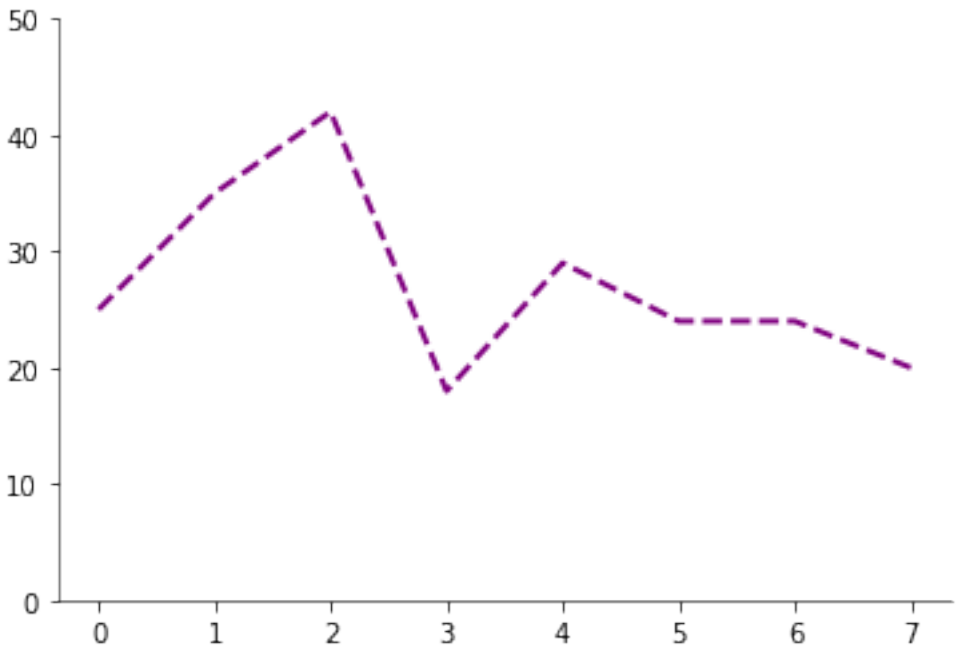
[ ]:

1.2 Grafico de lineas

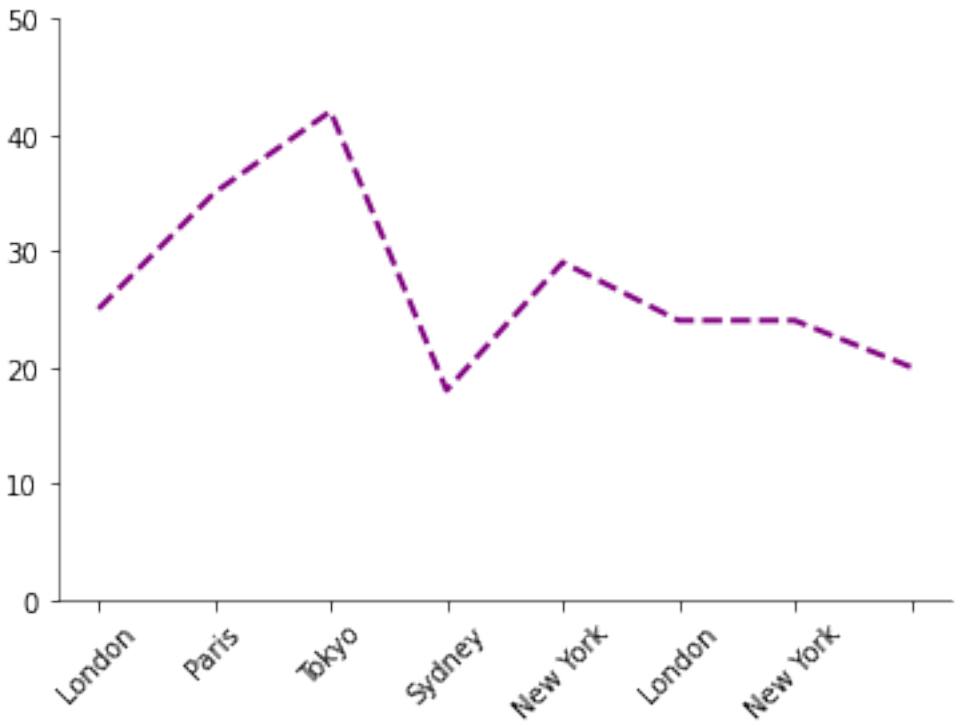
[21]:
```
# la evolución de las edades
plt.plot(df['age'])
ax.spines['right'].set_visible(False) # quitar marco derecheo
ax.spines['top'].set_visible(False) # quitar marco de arriba
plt.ylim(0, 50) # escala eje y
plt.show()
```



[28]:
```
# quito cuadro, color grosor, discontinua

fig, ax = plt.subplots()
ax.plot(df['age'], color='purple', linewidth=2, linestyle='--')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_linewidth(0.5) # esto disminuye el grosor de la lines y
ax.spines['bottom'].set_linewidth(0.5) # esto disminuye el grosor de la lines x
plt.ylim(0, 50)
plt.show()
```



[30]:
```
# en vez del indice del df ponga el nombre ciudad. No tiene sentido pq algunas␣
↪ciudades se repiten

fig, ax = plt.subplots()
ax.plot(df['age'], color='purple', linewidth=2, linestyle='--')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.spines['left'].set_linewidth(0.5) # esto disminuye el grosor de la lines y
ax.spines['bottom'].set_linewidth(0.5) # esto disminuye el grosor de la lines x
plt.ylim(0, 50)
ax.set_xticklabels(df['city'], rotation=45)
plt.show()
```



[190]:
```
# JUgamos con un data frame diferente
# creamos una lista con los países
paises = ['España', 'Francia', 'Alemania', 'Italia', 'Portugal']

# generamos un DataFrame con 11 columnas (años) y 5 filas (países)
data = np.random.randint(1000, 5000, (5, 11)) # matriz de 5 filas y 11 columnas
pib = pd.DataFrame(data, columns=[2010, 2011, 2012, 2013, 2014, 2015, 2016,␣
↪2017, 2018, 2019, 2020], index=paises)

# imprimimos el DataFrame
pib
```

[190]:

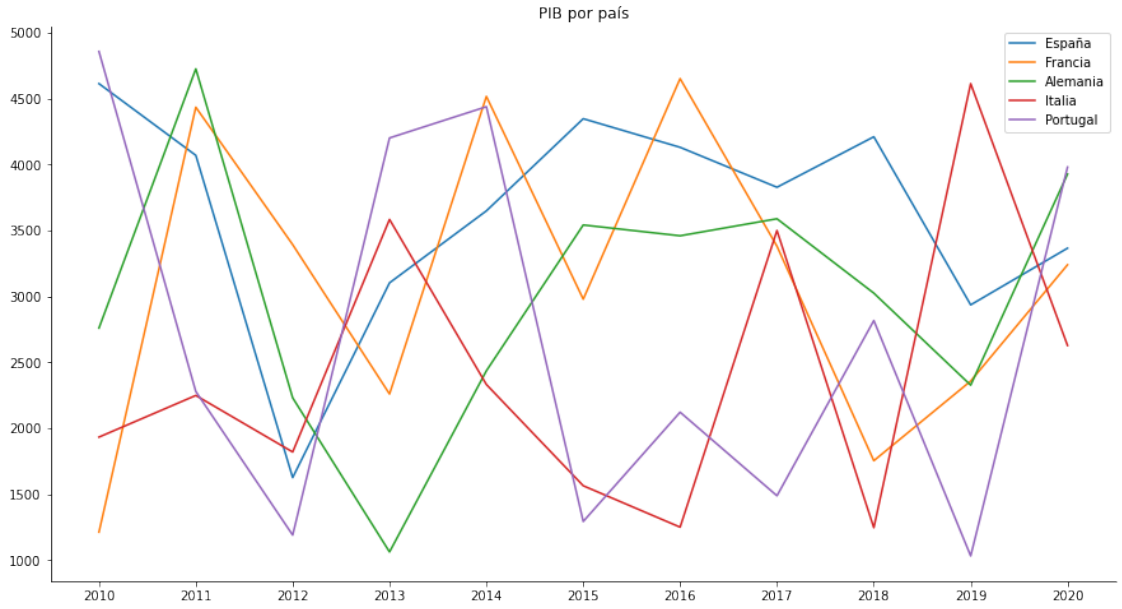
	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
España	3727	1529	1923	2373	2196	2353	4894	2871	2711	3725	3663
Francia	2060	2142	3663	2654	2694	3409	4987	3366	1800	3366	4236
Alemania	3842	3451	1766	3857	3012	4543	1087	4692	3458	1396	1907
Italia	1510	2716	2895	4797	4333	2025	2245	3076	4792	2398	3345
Portugal	2450	2864	3818	3398	2292	2397	4510	2203	2365	3405	3634

[191]:
```
# lo ponemos de forma correcta
pib = pib.T
pib
```

[191]:

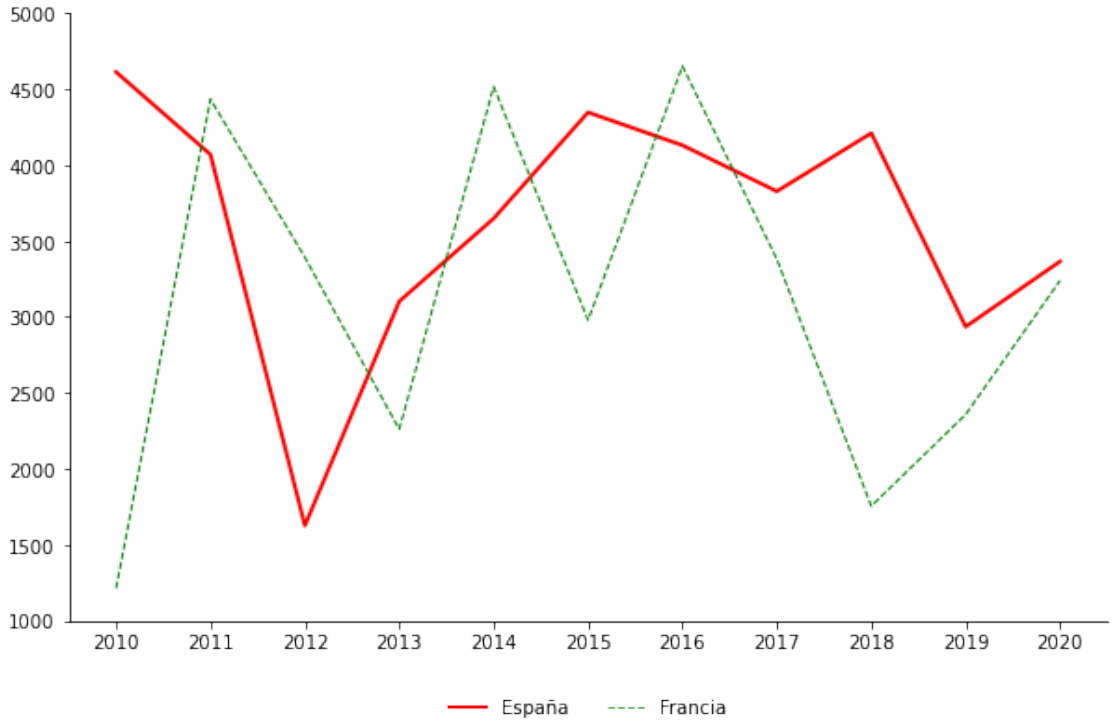
	España	Francia	Alemania	Italia	Portugal
2010	3727	2060	3842	1510	2450
2011	1529	2142	3451	2716	2864
2012	1923	3663	1766	2895	3818
2013	2373	2654	3857	4797	3398
2014	2196	2694	3012	4333	2292
2015	2353	3409	4543	2025	2397
2016	4894	4987	1087	2245	4510
2017	2871	3366	4692	3076	2203
2018	2711	1800	3458	4792	2365
2019	3725	3366	1396	2398	3405
2020	3663	4236	1907	3345	3634

[99]:
```
# de forma general todo los paises

fig, ax = plt.subplots(figsize=(15, 8)) # definimos el tamaño del grafico
for pais in pib.columns:
    ax.plot(pib.index, pib[pais], label=pais)

ticks = np.arange(2010, 2021, 1) # perosnalizar eje x
ax.set_xticks(ticks) # para que salgan todos los años
ax.set_xticklabels(pib.index, rotation=0)  # por si queremos poner otra rotación
ax.legend()


ax.set_title('PIB por país')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

plt.show()
```



[126]:
```
# especificando cuales y con que formato. OTRA FORMA ES HACER UN DATAFRAME CON␣
 ↪LOS PAISES QUE QUEREMOS
# vamos a guardar también la imagen
print(os.getcwd()) # ruta actual
# os.chdir("/Users/adrian_gr/Desktop") por si queremos poner otra


fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(pib.index, pib.España, color='red', linewidth=2, label='España')
ax.plot(pib.index, pib.Francia, color='green', linewidth=1, linestyle='--',␣
 ↪label='Francia')
ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(1000, 5500, 500))
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.legend(loc='upper center', frameon=False, ncol=2, bbox_to_anchor=(0.5, -0.
 ↪1)) # el -0.1 cuanto de cerca de eje x
ax.text(0.15, -0.2, 'Fuente: datos de ejemplo', ha='center', transform=ax.
 ↪transAxes, fontsize=12)
plt.savefig('pib_paises.png', bbox_inches='tight')
plt.show()
```

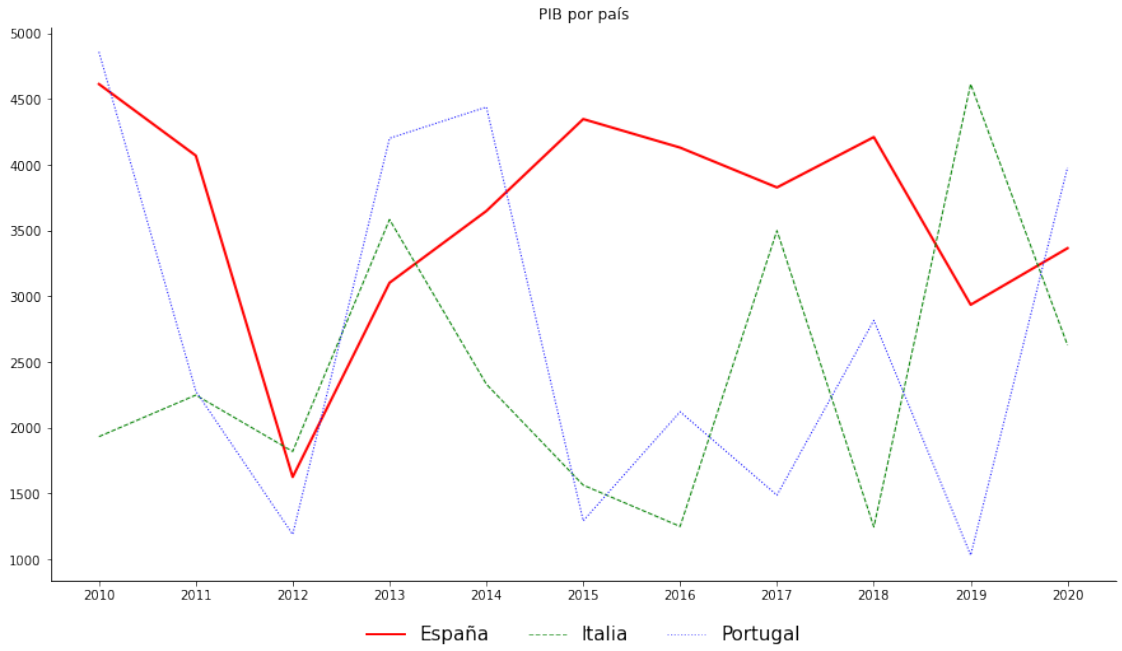
```
/Users/adrian_gr/Desktop/4.cnmv/04.practica/02. visualizacion
```



Fuente: datos de ejemplo

[132]:
```
# Vamos a sacar un grafico parecido pero combinando bucle y condicionales
# se ha conseguido pero casi que tiene mas sentido porner un ax.plot() por pais
paises = ['España', 'Italia', 'Portugal']


fig, ax = plt.subplots(figsize=(15, 8)) # definimos el tamaño del grafico
for pais in paises:
    if pais == 'España':
        colp = 'red'
        lwp = 2
        ltp = 'solid'
    elif pais == 'Italia':
        colp = 'green'
        lwp = 1
        ltp = 'dashed'
    else:
        colp = 'blue'
        lwp = 1
        ltp = 'dotted'
    ax.plot(pib.index, pib[pais], color=colp, linewidth=lwp, linestyle=ltp,␣
 ↪label=pais)

ticks = np.arange(2010, 2021, 1) # perosnalizar eje x
```

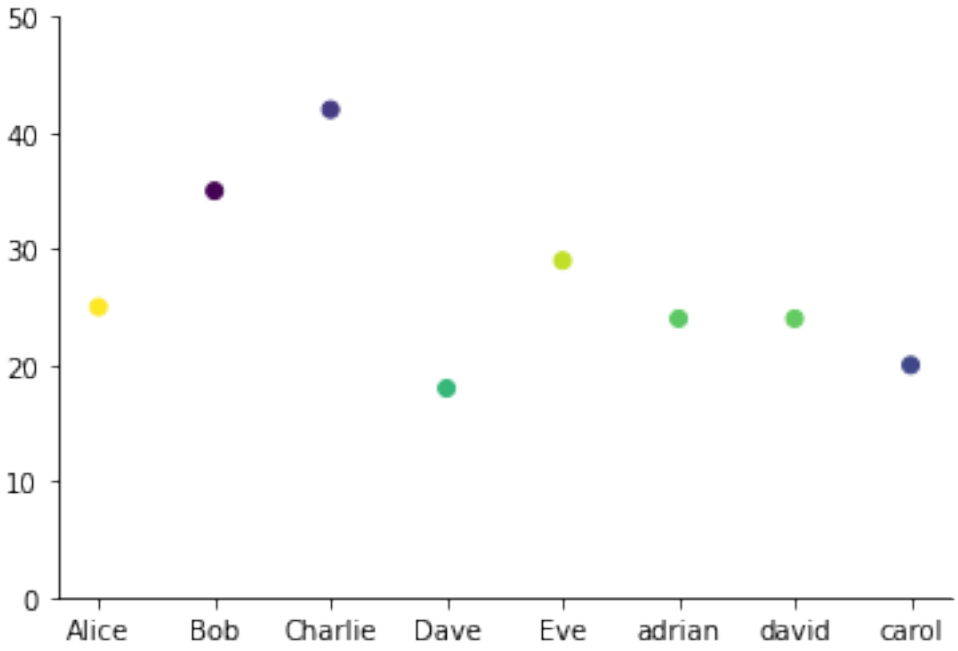
```
ax.set_xticks(ticks) # para que salgan todos los años
ax.set_xticklabels(pib.index, rotation=0)  # por si queremos poner otra rotación
ax.legend()


ax.set_title('PIB por país')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.legend(loc='upper center', frameon=False, ncol=3, bbox_to_anchor=(0.5, -0.
 ↪05), fontsize=15)

plt.show()
```



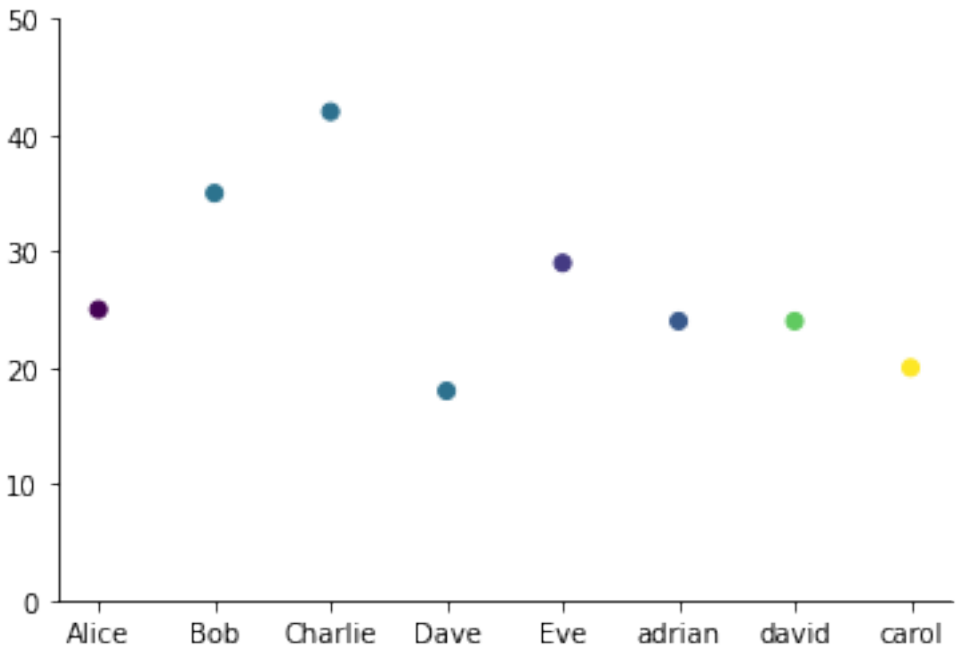
1.3 Scatter

[46]:
```
#grafico de dispersión
# la relación de cada nombre con su edad
# en scater podemos generar numeros aleatorios que asigna como colores, hay que␣
 ↪poner tantas como opciones haya
diferentes = df['name'].nunique() # contamos cuantos nombres diferentes hay
colors = np.random.rand(diferentes)
plt.scatter(df['name'], df['age'], c=colors)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.ylim(0, 50)
```

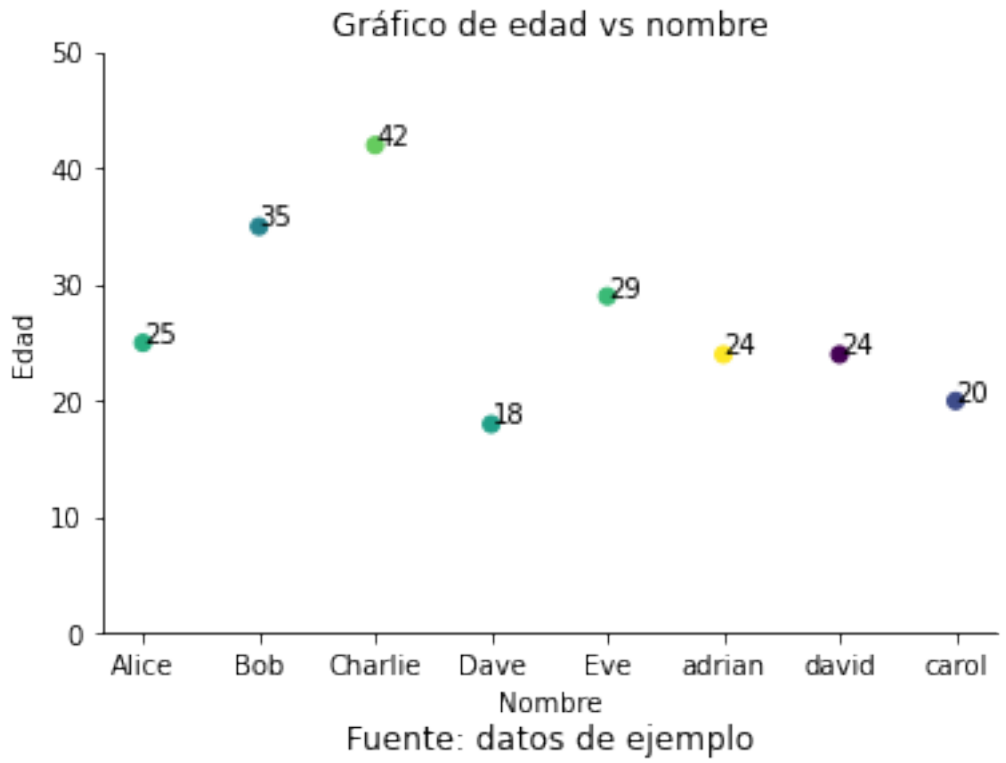
```
plt.show()
```



[54]:
```
#hacemos el grafico con el formato correcto, definiendo una figura

diferentes = df['name'].nunique() # contamos cuantos nombres diferentes hay
colors = np.random.rand(diferentes)

fig, ax = plt.subplots()
ax.scatter(df['name'], df['age'], c=colors)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.set_ylim(0, 50)
plt.show()
```



[62]:
```
# Añadimos los valores en cada punto, y nombre de los ejes, titulo y fuente

diferentes = df['name'].nunique() # contamos cuantos nombres diferentes hay
colors = np.random.rand(diferentes)

fig, ax = plt.subplots()
ax.scatter(df['name'], df['age'], c=colors)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.set_ylim(0, 50)
ax.set_xlabel('Nombre')
ax.set_ylabel('Edad')
ax.set_title('Gráfico de edad vs nombre')
ax.text(0.5, -0.2, 'Fuente: datos de ejemplo', ha='center', transform=ax.
 ↪transAxes, fontsize=12)

for x, y, val in zip(df['name'], df['age'], df['age']):
    ax.annotate(str(val), xy=(x, y), textcoords='data')
plt.show()
```



Fuente: datos de ejemplo

[ ]:

[ ]:

2 1. Seaborn

[ ]:

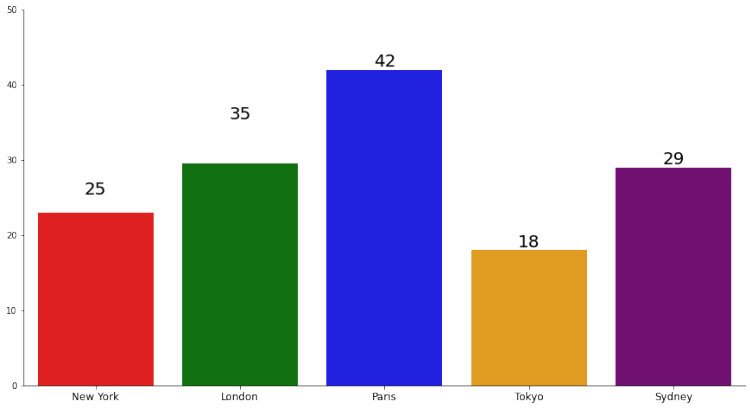
2.0.1 Barras

[135]:
```
df
```

[135]:

	name	age	city
0	Alice	25	New York
1	Bob	35	London
2	Charlie	42	Paris
3	Dave	18	Tokyo
4	Eve	29	Sydney
5	adrian	24	New York
6	david	24	London
7	carol	20	New York

[152]:
```
fig, ax = plt.subplots(figsize=(15,8))
sns.barplot(data=df, x='city', y='age', palette=colors, ci=None, ax=ax) #␣
 ↪importante el ci para que no salgan intervalos de confianza
ax.set_xticklabels(df['city'], rotation=0, fontsize = 12)
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_ylim(0, 50)
sns.despine(top=True, right=True)  # quitar marcos de arriba y derecha

for i, v in enumerate(df['age']):
    ax.annotate(str(v), xy=(i, v), ha='center', va='bottom', fontsize = 20)

plt.show()
# de cada ciudad solo coge el primer dato
```



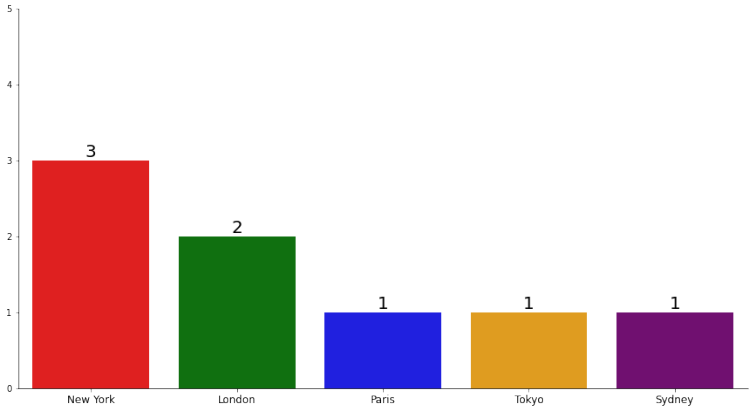
[149]:
```
## ESTO ES VITAL

df['count'] = df['city'].map(df['city'].value_counts())
df
```

[149]:

	name	age	city	count
0	Alice	25	New York	3
1	Bob	35	London	2
2	Charlie	42	Paris	1
3	Dave	18	Tokyo	1
4	Eve	29	Sydney	1
5	adrian	24	New York	3
6	david	24	London	2
7	carol	20	New York	3

[155]:
```
fig, ax = plt.subplots(figsize=(15,8))
sns.barplot(data=df, x='city', y='count', palette=colors, ci=None, ax=ax) #␣
 ↪importante el ci para que no salgan intervalos de confianza
```

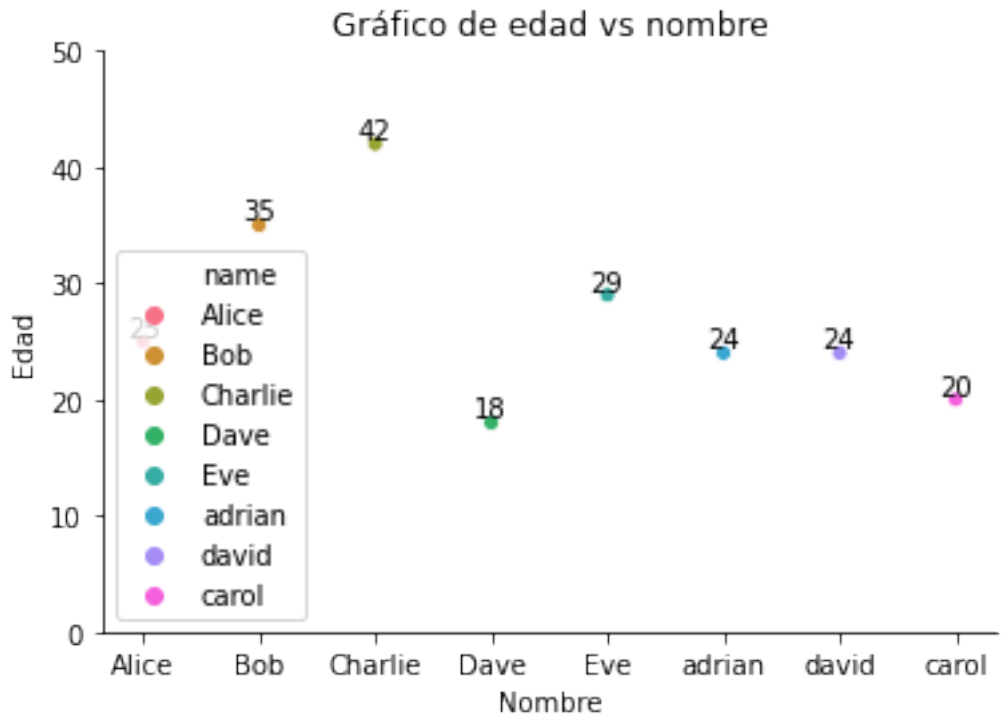
```
ax.set_xticklabels(df['city'], rotation=0, fontsize = 12)
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_ylim(0, 5)
sns.despine(top=True, right=True)  # quitar marcos de arriba y derecha

for i, v in enumerate(df['count']):
    ax.annotate(str(v), xy=(i, v), ha='center', va='bottom', fontsize = 20)

plt.show()
```



[ ]:

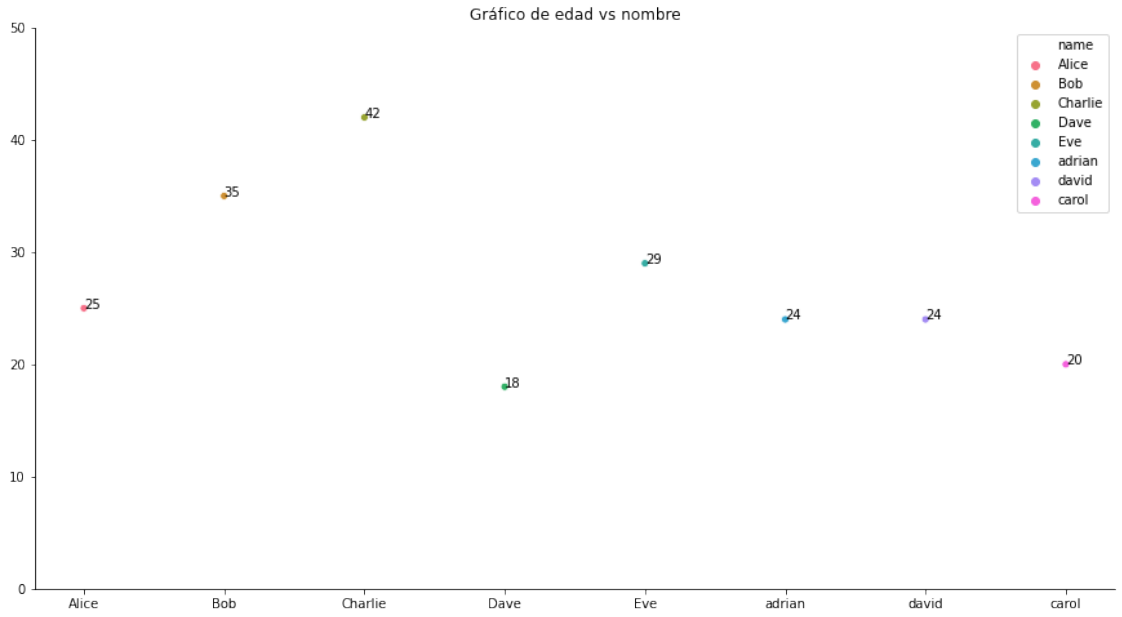
2.0.2 Scatter

[156]:
```
diferentes = df['name'].nunique()
colors = sns.color_palette('husl', diferentes)

sns.scatterplot(data=df, x='name', y='age', hue='name', palette=colors)
sns.despine(top=True, right=True)
plt.title('Gráfico de edad vs nombre')
plt.xlabel('Nombre')
plt.ylabel('Edad')
plt.ylim(0, 50)

for x, y, val in zip(df['name'], df['age'], df['age']):
    plt.text(x, y, str(val), ha='center', va='bottom')

plt.show()
```



[169]:
```
diferentes = df['name'].nunique()


# Creamos la figura y el objeto de ejes
fig, ax = plt.subplots(figsize=(15,8))

# Creamos el scatterplot
sns.scatterplot(data=df, x='name', y='age', hue='name', palette=colors, ax=ax)

# Personalizamos el eje y
ax.set_ylim(0, 50)

# Quitamos los marcos de arriba y derecha
sns.despine(top=True, right=True)

# Añadimos el título, las etiquetas de los ejes y la fuente
ax.set_title('Gráfico de edad vs nombre')
ax.set_xlabel('')
ax.set_ylabel('')


# Añadimos los valores de edad encima de cada punto
for x, y, val in zip(df['name'], df['age'], df['age']):
```

```
    ax.annotate(str(val), xy=(x, y), textcoords='data')
#ax.legend(loc='upper center', frameon=False, ncol=4, bbox_to_anchor=(0.5, -0.
 ↪1), fontsize=15)
plt.show()
```

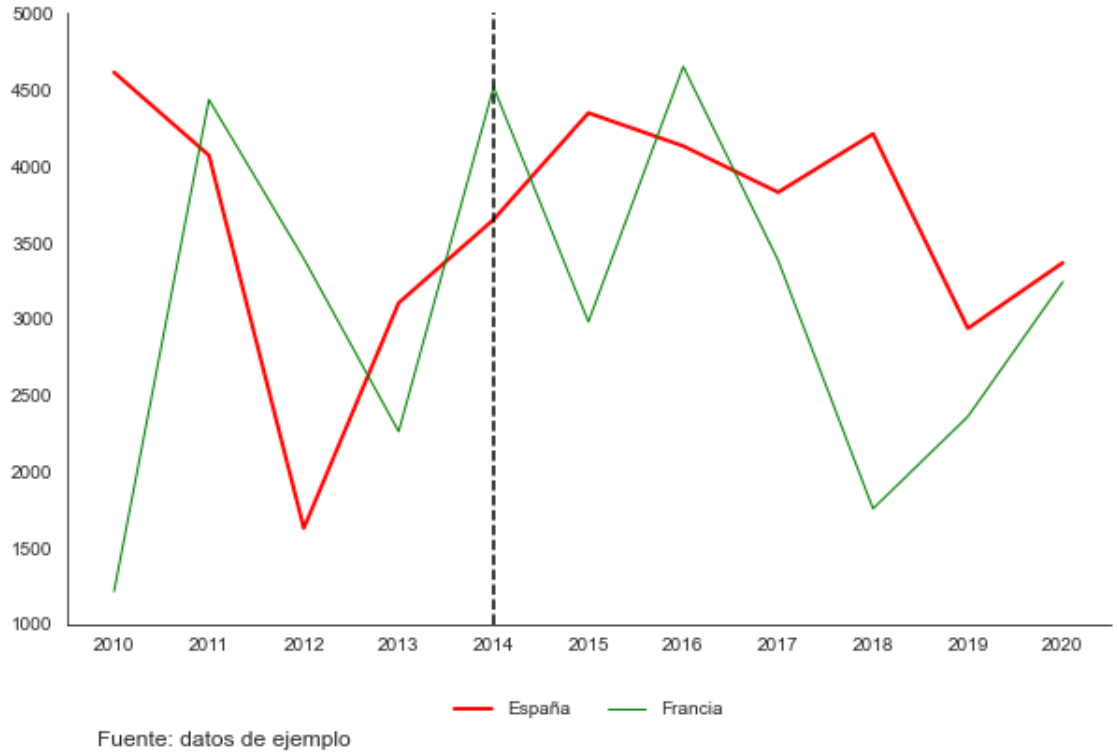


[ ]:

2.0.3 Lineas

[181]:
```
sns.set_style("white") # qu no salga el grid

fig, ax = plt.subplots(figsize=(10, 6))

sns.lineplot(data=pib, x=pib.index, y="España", color="red", linewidth=2,␣
 ↪label="España")
sns.lineplot(data=pib, x=pib.index, y="Francia", color="green", linewidth=1,␣
 ↪linestyle="--", label="Francia")


ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(1000, 5500, 500))

ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
ax.set_ylabel("")
```

```
ax.legend(loc="upper center", frameon=False, ncol=2, bbox_to_anchor=(0.5, -0.1))
ax.text(0.15, -0.2, "Fuente: datos de ejemplo", ha="center", transform=ax.
 ↪transAxes, fontsize=12)
ax.axvline(x=2014, color='black', linestyle='--')

plt.savefig("pib_paises_seaborn.png", bbox_inches="tight")
plt.show()
```



[ ]:
```
### Para terminar vamos a hacer unas modificaciones al df pib
```

[192]:
```
pib
```

[192]:

	España	Francia	Alemania	Italia	Portugal
2010	3727	2060	3842	1510	2450
2011	1529	2142	3451	2716	2864
2012	1923	3663	1766	2895	3818
2013	2373	2654	3857	4797	3398
2014	2196	2694	3012	4333	2292
2015	2353	3409	4543	2025	2397
2016	4894	4987	1087	2245	4510
2017	2871	3366	4692	3076	2203
2018	2711	1800	3458	4792	2365

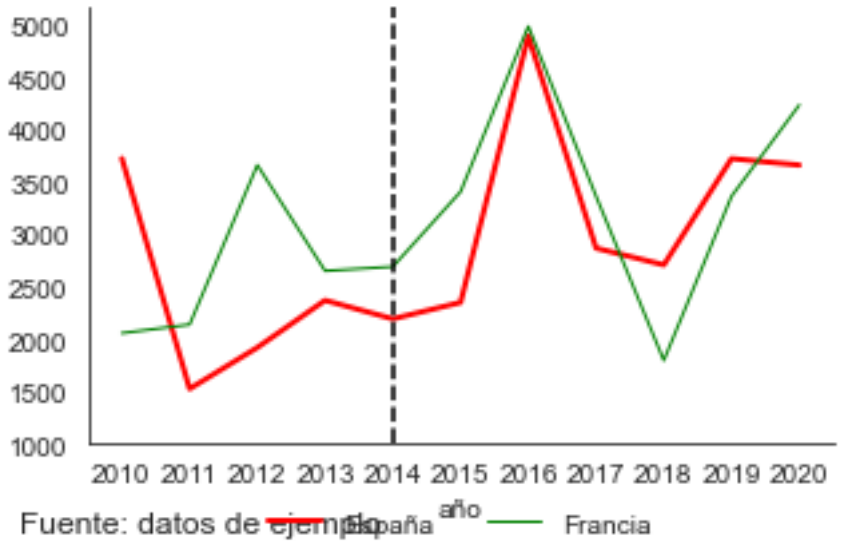
```
2019    3725      3366       1396     2398       3405
2020    3663      4236       1907     3345       3634
```

[195]:
```
# pasamos el indice a columna
pib = pib.reset_index().rename(columns={'index': 'año'})
pib
```

[195]:

	año	España	Francia	Alemania	Italia	Portugal
0	2010	3727	2060	3842	1510	2450
1	2011	1529	2142	3451	2716	2864
2	2012	1923	3663	1766	2895	3818
3	2013	2373	2654	3857	4797	3398
4	2014	2196	2694	3012	4333	2292
5	2015	2353	3409	4543	2025	2397
6	2016	4894	4987	1087	2245	4510
7	2017	2871	3366	4692	3076	2203
8	2018	2711	1800	3458	4792	2365
9	2019	3725	3366	1396	2398	3405
10	2020	3663	4236	1907	3345	3634

[198]:
```
# con esta modificacion ya sale el grafico

sns.set_style("white") # qu no salga el grid

fig, ax = plt.subplots(figsize=(5, 3))

sns.lineplot(data=pib, x=pib.año, y="España", color="red", linewidth=2,␣
 ↪label="España")
sns.lineplot(data=pib, x=pib.año, y="Francia", color="green", linewidth=1,␣
 ↪linestyle="--", label="Francia")



ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(1000, 5500, 500))

ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
ax.set_ylabel("")

ax.legend(loc="upper center", frameon=False, ncol=2, bbox_to_anchor=(0.5, -0.1))
ax.text(0.15, -0.2, "Fuente: datos de ejemplo", ha="center", transform=ax.
 ↪transAxes, fontsize=12)
ax.axvline(x=2014, color='black', linestyle='--')

plt.savefig("pib_paises_seaborn.png", bbox_inches="tight")
plt.show()
```



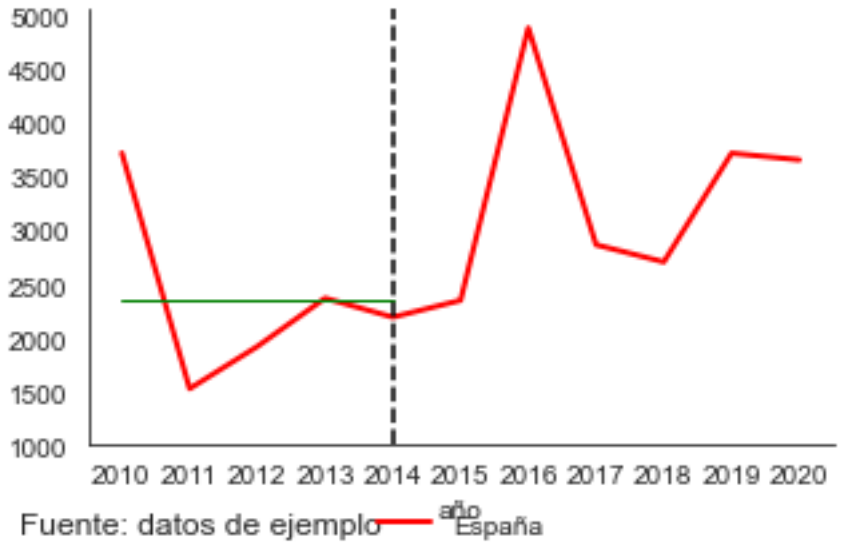
[206]:
```
# generamos la media hasta el 214

pib.loc[pib['año'] < 2015, 'media_espana'] = pib.loc[pib['año'] < 2015,␣
↪'España'].mean()
pib
```

[206]:

	año	España	Francia	Alemania	Italia	Portugal	media_espana
0	2010	3727	2060	3842	1510	2450	2349.6
1	2011	1529	2142	3451	2716	2864	2349.6
2	2012	1923	3663	1766	2895	3818	2349.6
3	2013	2373	2654	3857	4797	3398	2349.6
4	2014	2196	2694	3012	4333	2292	2349.6
5	2015	2353	3409	4543	2025	2397	NaN
6	2016	4894	4987	1087	2245	4510	NaN
7	2017	2871	3366	4692	3076	2203	NaN
8	2018	2711	1800	3458	4792	2365	NaN
9	2019	3725	3366	1396	2398	3405	NaN
10	2020	3663	4236	1907	3345	3634	NaN

[208]:
```
# lo pintamos

# con esta modificacion ya sale el grafico

sns.set_style("white") # qu no salga el grid

fig, ax = plt.subplots(figsize=(5, 3))
```

```
sns.lineplot(data=pib, x=pib.año, y="España", color="red", linewidth=2,␣
 ↪label="España")
sns.lineplot(data=pib, x=pib.año, y="media_espana", color="green", linewidth=1,␣
 ↪linestyle="--")


ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(1000, 5500, 500))

ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
ax.set_ylabel("")

ax.legend(loc="upper center", frameon=False, ncol=2, bbox_to_anchor=(0.5, -0.1))
ax.text(0.15, -0.2, "Fuente: datos de ejemplo", ha="center", transform=ax.
 ↪transAxes, fontsize=12)
ax.axvline(x=2014, color='black', linestyle='--')

plt.savefig("pib_paises_seaborn.png", bbox_inches="tight")
plt.show()
```



[209]:
```
# borramos la vriable creada
pib.drop("media_espana", axis=1, inplace=True)
pib
```

[209]:

	año	España	Francia	Alemania	Italia	Portugal
0	2010	3727	2060	3842	1510	2450
1	2011	1529	2142	3451	2716	2864

2	2012	1923	3663	1766	2895	3818
3	2013	2373	2654	3857	4797	3398
4	2014	2196	2694	3012	4333	2292
5	2015	2353	3409	4543	2025	2397
6	2016	4894	4987	1087	2245	4510
7	2017	2871	3366	4692	3076	2203
8	2018	2711	1800	3458	4792	2365
9	2019	3725	3366	1396	2398	3405
10	2020	3663	4236	1907	3345	3634

[210]:
```
pib = pib.set_index('año')
pib
```

[210]:

año	España	Francia	Alemania	Italia	Portugal
2010	3727	2060	3842	1510	2450
2011	1529	2142	3451	2716	2864
2012	1923	3663	1766	2895	3818
2013	2373	2654	3857	4797	3398
2014	2196	2694	3012	4333	2292
2015	2353	3409	4543	2025	2397
2016	4894	4987	1087	2245	4510
2017	2871	3366	4692	3076	2203
2018	2711	1800	3458	4792	2365
2019	3725	3366	1396	2398	3405
2020	3663	4236	1907	3345	3634

[211]:
```
sns.set_style("white") # qu no salga el grid

fig, ax = plt.subplots(figsize=(6, 6))

sns.lineplot(data=pib, x=pib.index, y="España", color="red", linewidth=2,␣
 ↪label="España")
sns.lineplot(data=pib, x=pib.index, y="Francia", color="green", linewidth=1,␣
 ↪linestyle="--", label="Francia")


ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(1000, 5500, 500))

ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
ax.set_ylabel("")

ax.legend(loc="upper center", frameon=False, ncol=2, bbox_to_anchor=(0.5, -0.1))
ax.text(0.15, -0.2, "Fuente: datos de ejemplo", ha="center", transform=ax.
 ↪transAxes, fontsize=12)
```

```
ax.axvline(x=2014, color='black', linestyle='--')

plt.savefig("pib_paises_seaborn.png", bbox_inches="tight")
plt.show()
```

Fuente: datos de ejemplo

[213]:
```
# si queremos calcular la media con año en el indice
pib.loc['2010':'2014', 'media_espana'] = pib.loc['2010':'2014', 'España'].mean()
pib
```

[213]:

año	España	Francia	Alemania	Italia	Portugal	media_espana
2010	3727	2060	3842	1510	2450	2349.6
2011	1529	2142	3451	2716	2864	2349.6
2012	1923	3663	1766	2895	3818	2349.6

2013	2373	2654	3857	4797	3398	2349.6
2014	2196	2694	3012	4333	2292	2349.6
2015	2353	3409	4543	2025	2397	NaN
2016	4894	4987	1087	2245	4510	NaN
2017	2871	3366	4692	3076	2203	NaN
2018	2711	1800	3458	4792	2365	NaN
2019	3725	3366	1396	2398	3405	NaN
2020	3663	4236	1907	3345	3634	NaN

[220]:
```
# para terminar, nos quedamos con el pib colapsado
total = pib.sum()
df_total = pd.DataFrame(data=total).T
df_total
```

[220]:

	España	Francia	Alemania	Italia	Portugal	media_espana
0	31965.0	34377.0	33011.0	34132.0	33336.0	11748.0

[ ]:

[ ]:

import pandas as pd from bokeh.plotting import figure, show from bokeh.models import ColumnDataSource

3 Crear el DataFrame

df = pd.DataFrame({ ‘name’: [‘Alice’, ‘Bob’, ‘Charlie’, ‘Dave’, ‘Eve’, ‘adrian’, ‘david’, ‘carol’], ‘age’: [25, 35, 42, 18, 29, 24, 24, 20], ‘city’: [‘New York’, ‘London’, ‘Paris’, ‘Tokyo’, ‘Sydney’, ‘New York’, ‘London’, ‘New York’] })

4 Crear la fuente de datos

source = ColumnDataSource(df)

5 Crear el gráfico de barras de la edad

p = figure(x_range=df[‘name’], plot_height=250, title=“Edad”) p.vbar(x=‘name’, top=‘age’, width=0.9, source=source, line_color=‘white’)

6 Crear el gráfico de tarta de la ciudad

city_counts = df.groupby(‘city’).size().reset_index(name=‘counts’) p2 = figure(plot_height=350, title=“Ciudad”, toolbar_location=None, tools=“hover”, tooltips=“@city: @counts”) p2.wedge(x=0, y=1, radius=0.4, start_angle=‘cumsum’, end_angle=‘cumsum’, line_color=“white”, fill_color=‘colors’, legend_field=‘city’, source=city_counts)

7 Mostrar los gráficos

show(p) show(p2)

[ ]: