Python_Matplotlib

January 23, 2026

0.1 MIAR - Representación Gráfica

Gabriel Salguero

Representacion grafica

- matplotlib basico (plot, legend, decoradores y anotaciones, grabar)
- dibujo con pandas (tipos de graficos)
- intro a seaborn https://seaborn.pydata.org/tutorial.html

Referencias

- https://plotdb.com/
- https://plotnine.org/
- https://d3js.org/
- https://plot.ly/python/
- https://bokeh.org/
- https://shiny.rstudio.com/gallery/
- https://www.tableau.com/es-es

0.2 matplotlib

- Librería estándar de representación gráfica
- Low level: fácil de usar, difícil de dominar
- Control
- Gráficos y diagramas

0.2.1 Elementos de visualización

- Gráfico. Tipo de representación dentro de una subfigura o figura. Existen gran cantidad de tipos de gráficos: barras, líneas, tarta…
- Subfigura. Representación dentro de una figura.
- Figura. Marco donde se representan de manera conjunta una visualización. Se caracterizan por tener un título general y suele tener una descripción así como una fuente o referencia

[2]:
```
# importar matplotlib
import matplotlib.pyplot as plt
```

[3]:
```
import numpy as np
import pandas as pd
```

- 'plot(data)' para crear un gráfico con 'data'
- 'show()' para mostrarlo

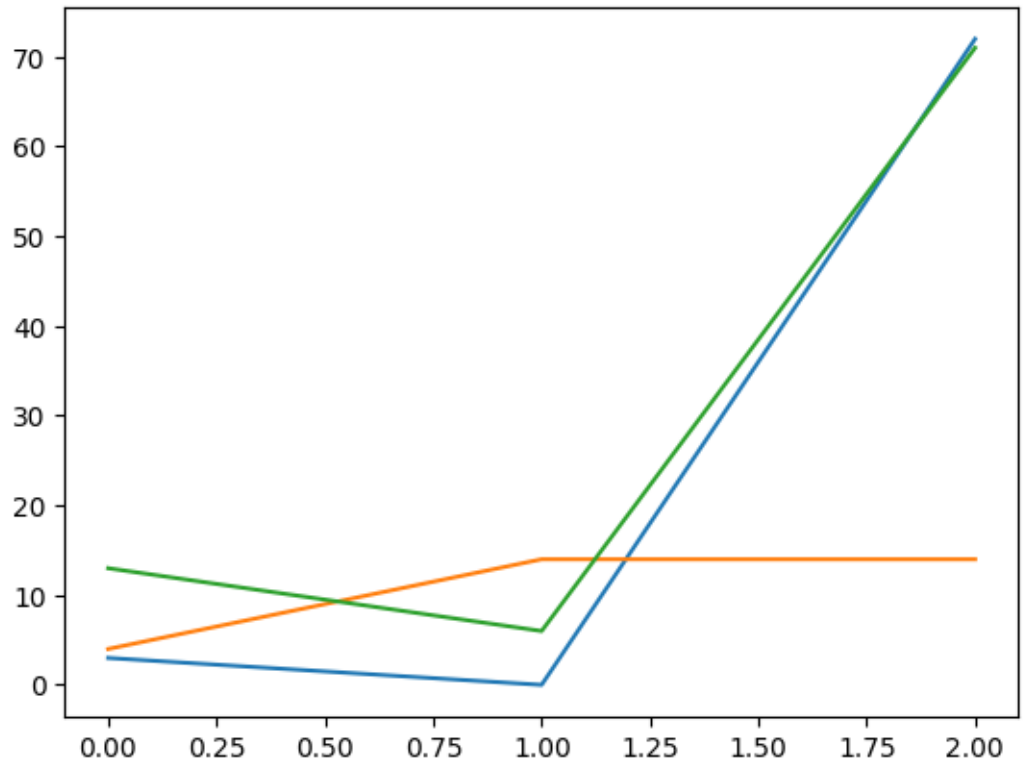
[4]:
```
%matplotlib --list
```

```
Available matplotlib backends: ['agg', 'auto', 'cairo', 'gtk3', 'gtk3agg',
'gtk3cairo', 'gtk4', 'gtk4agg', 'gtk4cairo', 'inline', 'macosx', 'nbagg',
'notebook', 'osx', 'pdf', 'pgf', 'ps', 'qt', 'qt5', 'qt5agg', 'qt5cairo', 'qt6',
'qtagg', 'qtcairo', 'svg', 'template', 'tk', 'tkagg', 'tkcairo', 'webagg', 'wx',
'wx', 'wxagg', 'wxcairo']
```

[5]:
```
# magic command para visualizar gráficos en jupyter en modo no interactivo
%matplotlib inline
```

[6]:
```
data = np.random.randint(100,size=(3,3))
display(data)
plt.plot(data) # creación del gráfico
plt.show() # comando que muestra los gráficos creados hasta el momento
```

```
array([[ 3,  4, 13],
       [ 0, 14,  6],
       [72, 14, 71]], dtype=int32)
```

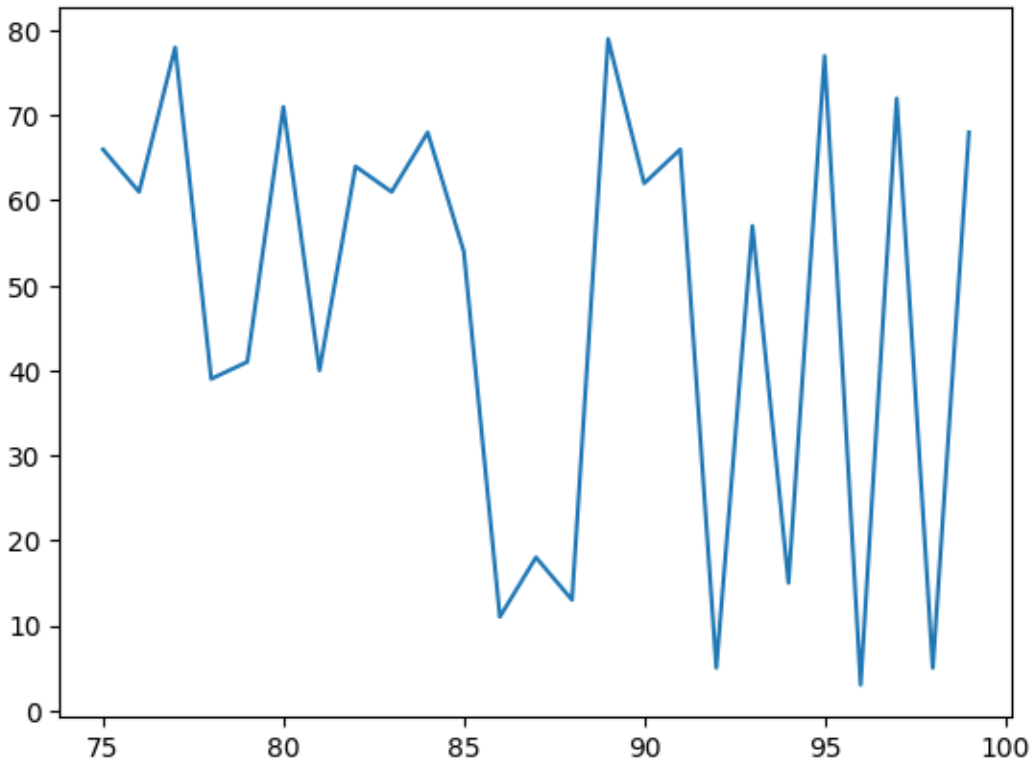


[7]:
```
# activar modo interactivo
%matplotlib notebook
plt.ioff() # evitar que se sobreescriban los gráficos
%matplotlib inline
```

[8]:
```
# Se pueden especificar x e y por separado
x_data = np.arange(75, 100) # dimensiones!
y_data = np.random.randint(80,size=(25))

print(x_data,y_data)
if x_data.size == y_data.size:
    plt.plot(x_data, y_data)
    plt.show()
```

```
else:
    print(f"{x_data.size} no tienen las mismas dimensiones {y_data.size}")
```

```
[75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98
 99] [66 61 78 39 41 71 40 64 61 68 54 11 18 13 79 62 66  5 57 15 77  3 72  5
 68]
```

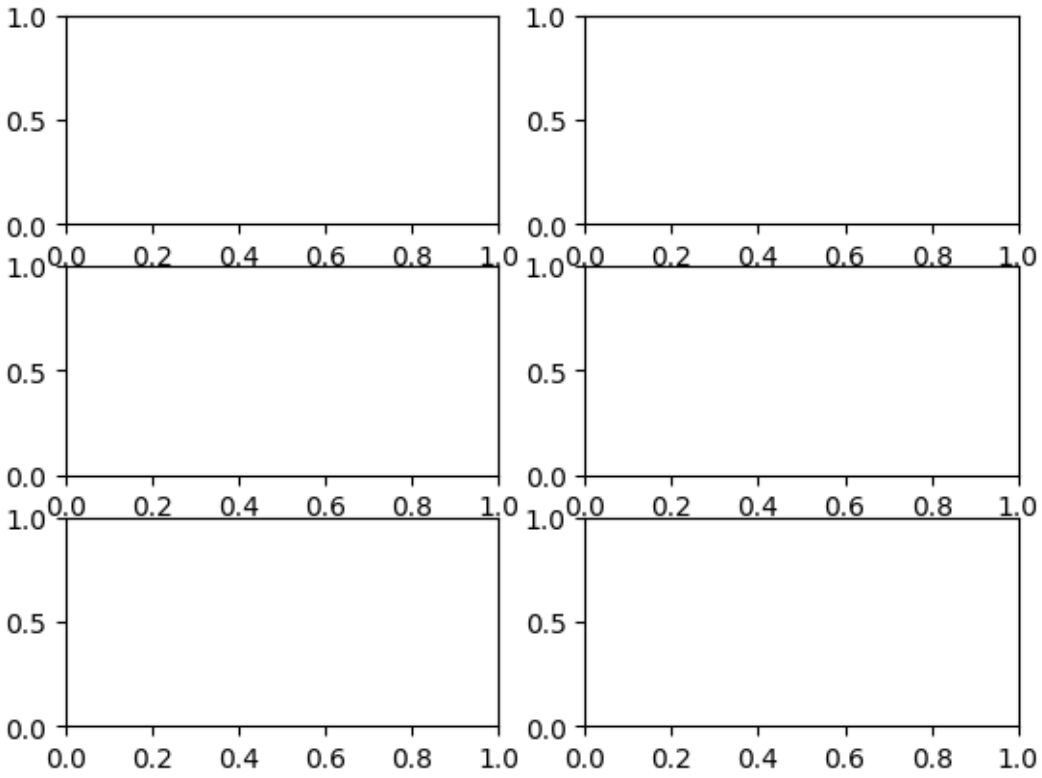


0.2.2 figure, axes

- Figure: objeto en el que residen todos los gráficos (uno o múltiples)
- Axes: cada uno de los subplots (o gráficos) dentro de Figure

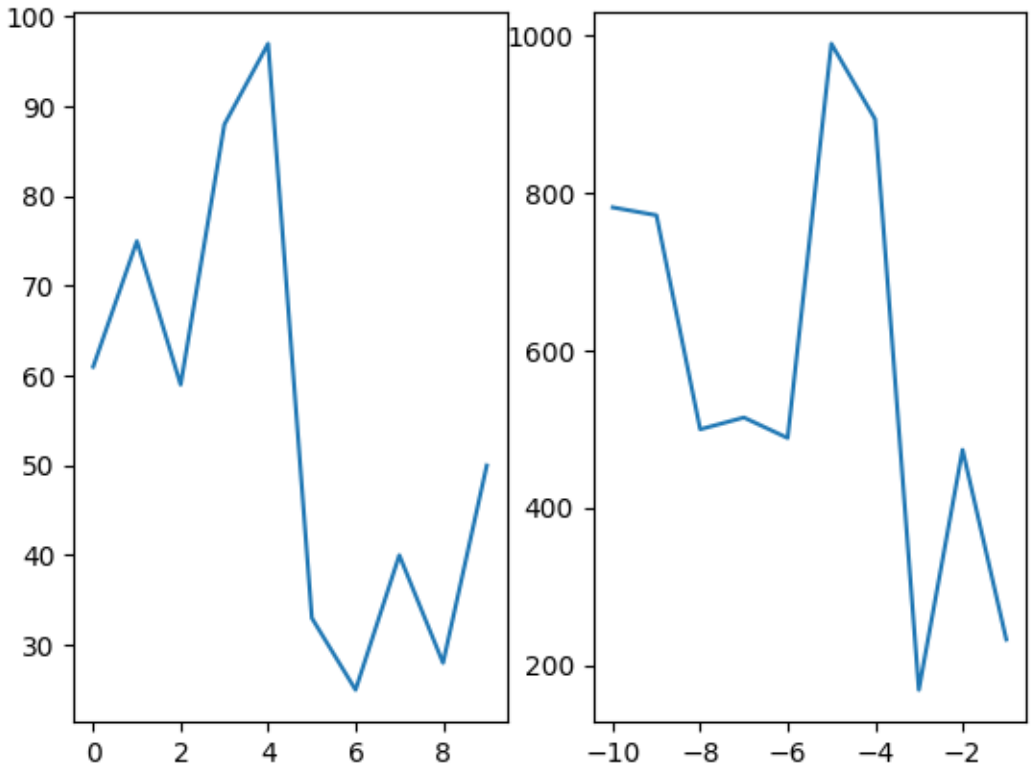
[9]:
```
fig, axes = plt.subplots(3,2) # crea una ndarray de 1x2 axes (o subplots)
print(type(axes))
print(axes.shape)
```

```
<class 'numpy.ndarray'>
(3, 2)
```

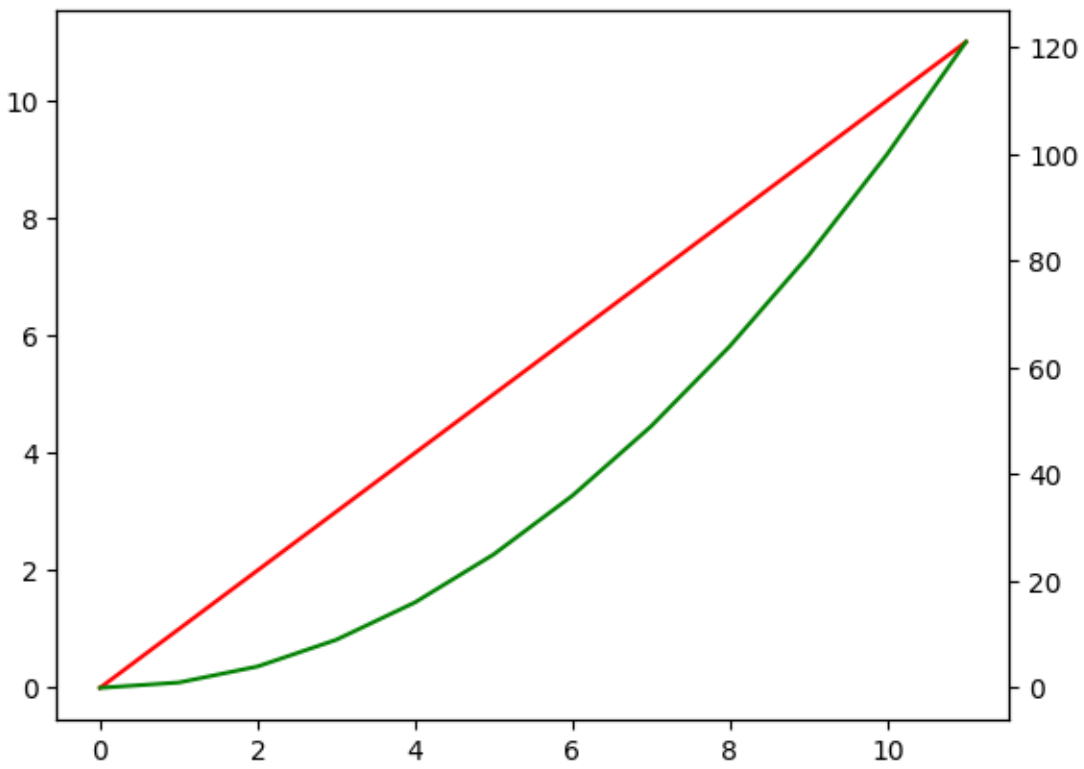


[10]:
```
# Generar los datos
x_data1 = np.arange(0,10)
x_data2 = np.arange(-10,0)
y_data1 = np.random.randint(100,size=(10))
y_data2 = np.random.randint(1000,size=(10))
```

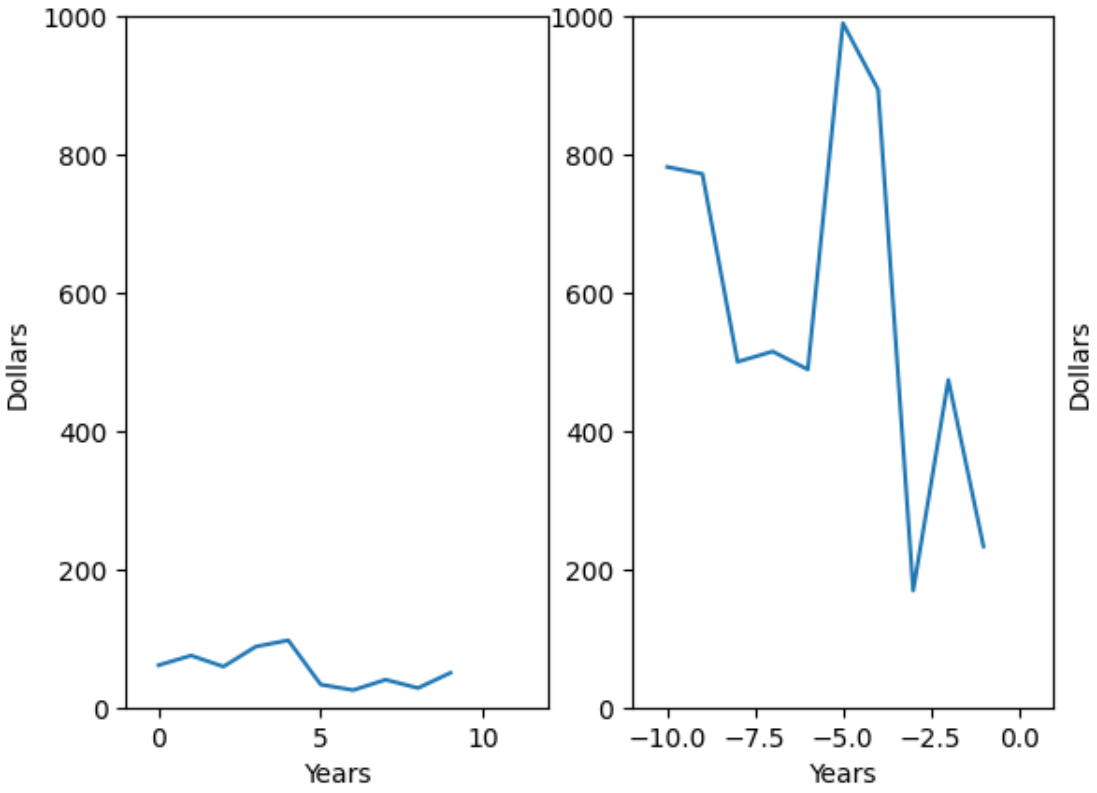
[11]:
```
# Crear la figura y los subplots
fig, axes = plt.subplots(1,2)
# preparar cada subplot
axes[0].plot(x_data1,y_data1)
axes[1].plot(x_data2,y_data2)
#mostrar ambos axes
plt.show() # Recomendable usar plt y no fig.show()
```



[12]:
```
# crear un nuevo eje con la x compartida
fig, ax = plt.subplots(1, 1)
x = np.arange(12)
ax.plot(x, "r")

ax2 = ax.twinx()
ax2.plot(x ** 2, "g")

plt.show()
```



[13]:
```
# Crear la figura y los subplots
fig, axes = plt.subplots(1,2)
# preparar cada subplot
axes[0].plot(x_data1,y_data1)
axes[1].plot(x_data2,y_data2)

# tanto axes como fig pueden configurarse a través de funciones
axes[0].set_ylim([0,1000])
axes[0].set_xlim([-1,12])

axes[0].set_xlabel("Years")
axes[0].set_ylabel("Dollars")

axes[1].set_ylim([0,1000])
axes[1].set_xlim([-11,1])

axes[1].set_xlabel("Years")
axes[1].set_ylabel("Dollars", loc="center", labelpad=-210)

plt.show()
```



0.2.3 Más métodos

- Documentación general de la API

0.3 .plot()

Múltiples parámetros

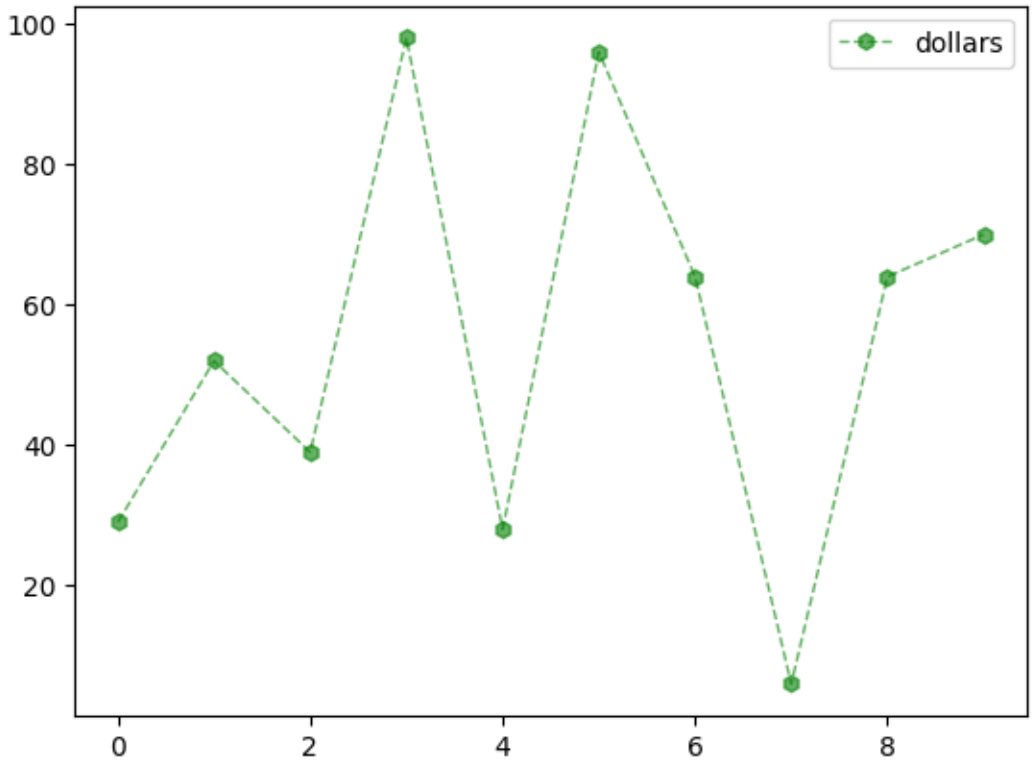
`plt.plot(x, y, style, label, linewidth=1, alpha=1)`

- 'style': cadena de texto con formato (color, tipo...) [color][marker][line]
- 'label': referencia para leyenda
- 'linewidth': brocha
- 'alpha': transparencia

Otros: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html

[14]:
```
y_data = np.random.randint(100,size=(10))
x_data = np.arange(10)

fig,axis = plt.subplots(1,1)
axis.plot(x_data,y_data,'gh--',label='dollars',linewidth=1,alpha=0.6) #␣
 ↪creación del gráfico
```

```
axis.legend(loc='best') # representar la leyenda
plt.show()
```



Color	Descripción
`b`	blue
`g`	green
`r`	red
`c`	cyan
`m`	magenta
`y`	yellow
`k`	black
`w`	white

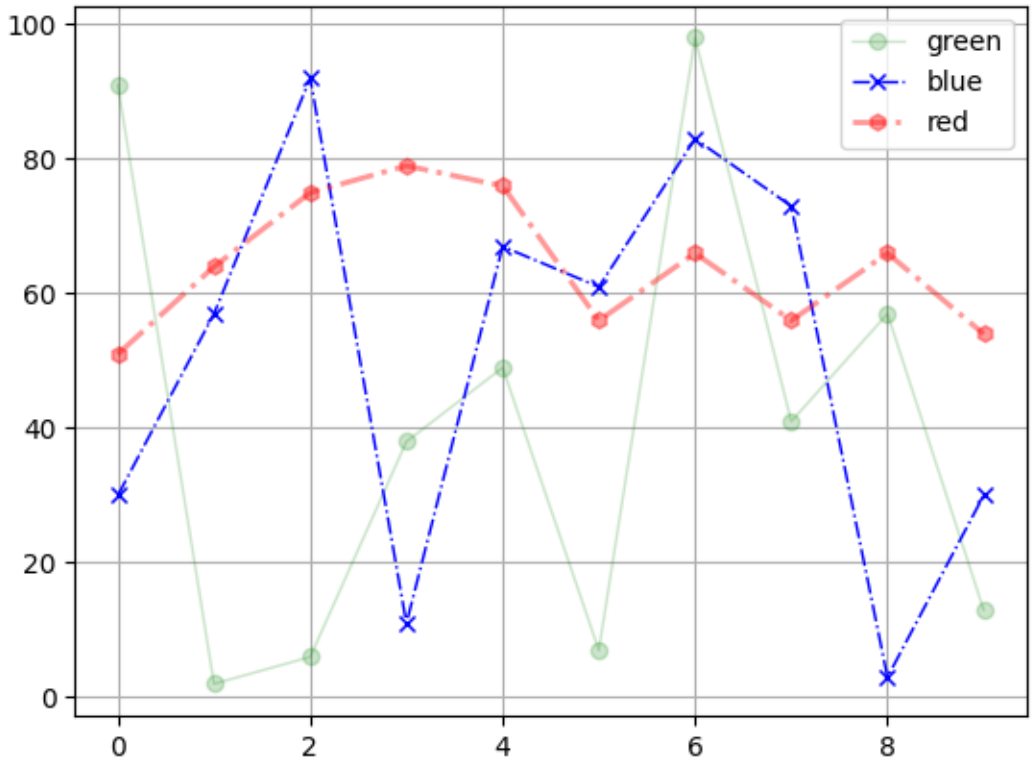
Marcador	Descripción
`.`	point marker
`,`	pixel marker
`o`	circle marker
`v`	triangle_down marker
`^`	triangle_up marker
`<`	triangle_left marker

Marcador	Descripción
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker
4	tri_right marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker
_	hline marker

Línea	Descripción
-	solid line style
--	dashed line style
-.	dash-dot line style
:	dotted line style

0.3.1 Representar más de un gráfico en una subfigura

[15]:
```
x_data =np.arange(10)
fig,axis= plt.subplots(1,1)
axis.plot(x_data,np.random.
 ↪randint(100,size=(10)),'go-',label='green',linewidth=1,alpha=0.2)#creación␣
 ↪del gráfico
axis.plot(x_data,np.random.randint(100,size=(10)),'bx-.
 ↪',label='blue',linewidth=1)#creacióndelgráfico
axis.plot(x_data,np.random.randint(50,80,size=(10)),'rh-.
 ↪',label='red',linewidth=2,alpha=0.4) #creacióndelgráfico
axis.legend(loc='best')
plt.grid()
plt.show()
```



0.3.2 Decoradores

Cambian el aspecto general de la figura

```
axes.set_xticks(<lista>) #etiquetas del eje
axes.set_title('titulo') #titulo
axes.set_ylabel('label')
axes.set_frame_on(bool)
```

Más métodos

0.4 Anotaciones

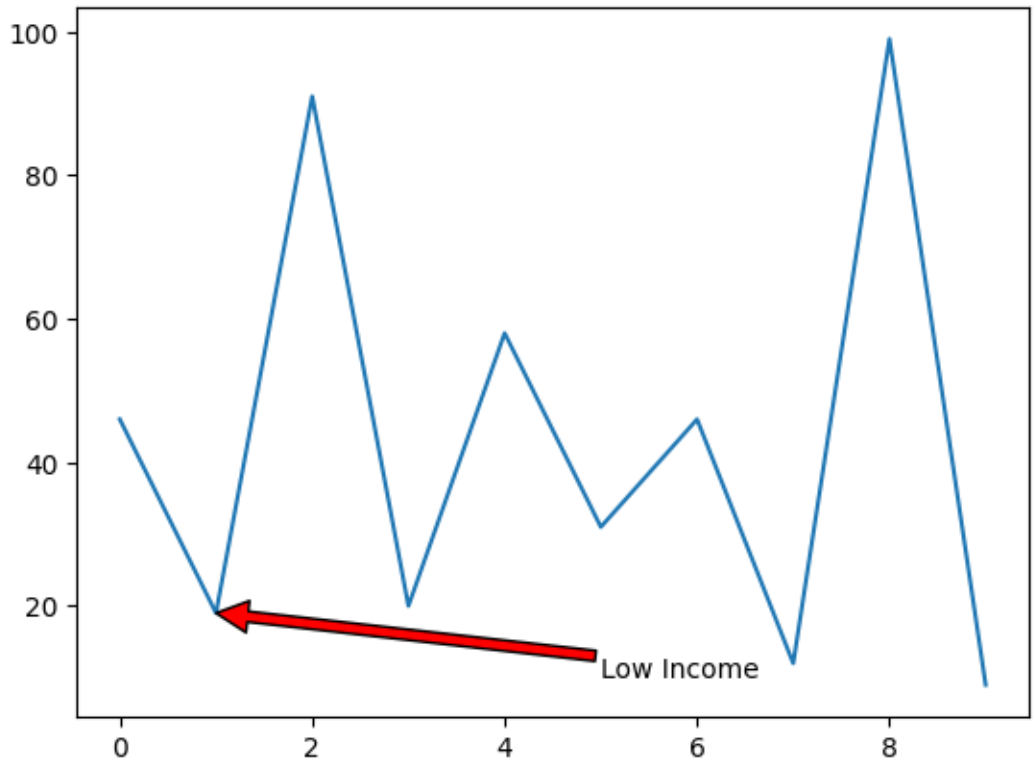
Formas de añadir información sobre el gráfico

```
axes.text(x,y,'texto')
axes.annotate(texto, xy, xytext, xycoords, arrowprops)
```

Más métodos

[16]:
```
y_data = np.random.randint(100, size=(10)) # Cuidado con las dimensiones␣
  ↪absolutas
x_data = np.arange(10)
```

[17]:
```
fig,axis = plt.subplots(1,1)
axis.plot(x_data,y_data)
# axis.text(x_data[1],y_data[1],'Low income') # coordenadas igual a los datos
axis.annotate('Low Income', xy=(x_data[1],y_data[1]), xytext=(5, 10),arrowprops␣
 ↪= dict(facecolor='red', shrink=1.0)) # anotación completa
plt.show()
```

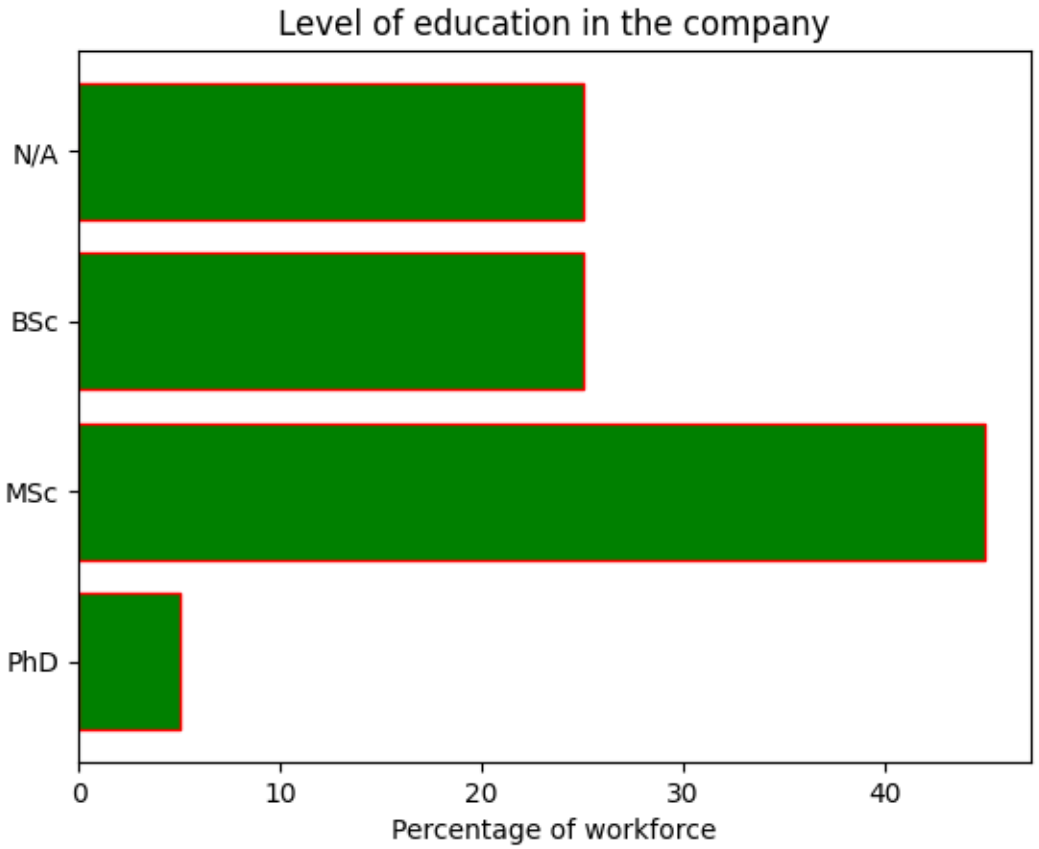


0.5 Tipos de gráficos

Ejemplos disponibles como referencia https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

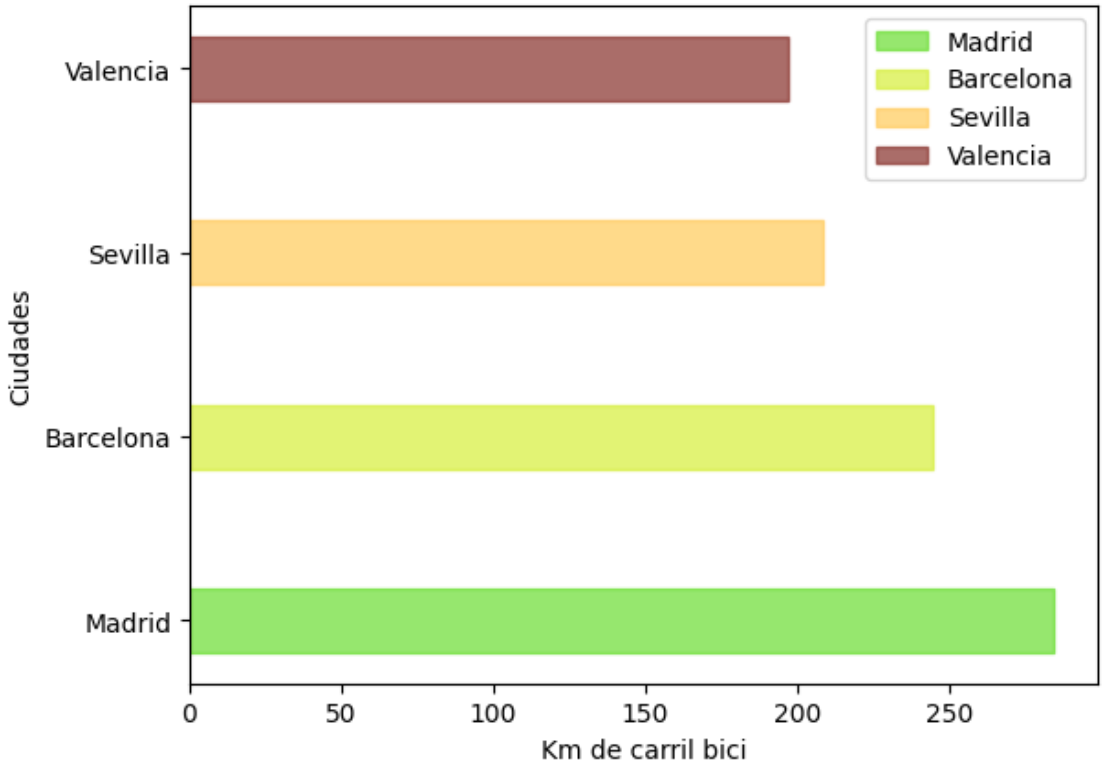
[18]:
```
# Horizontal bar
fig, ax = plt.subplots()
# Example data
people = ('PhD', 'MSc', 'BSc', 'N/A')
y_pos = np.arange(len(people))
percentage = [5,45,25,25]

# print(y_pos)
ax.barh(y_pos, percentage, align='center',
    color='green', edgecolor='red')
```

```
ax.set_yticks(y_pos)
ax.set_yticklabels(people)
#ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Percentage of workforce')
ax.set_title('Level of education in the company')
plt.show()
```



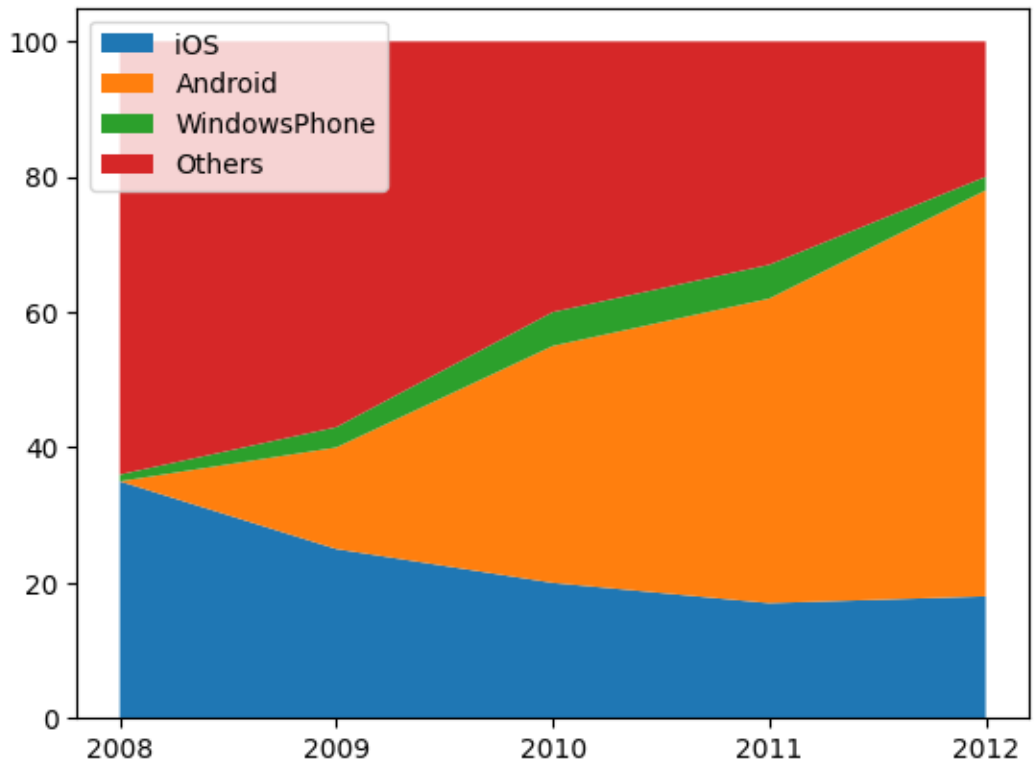
[19]:
```
ls_count = (284487, 244560, 208493, 196764)
cities = ('Madrid', 'Barcelona', 'Sevilla', 'Valencia')
data = np.array(ls_count)/1000
y_pos = np.arange(data.size)

width = 0.35  # the width of the bars
opacity = 0.7

fig, ax = plt.subplots()
rects1 = ax.barh(y_pos, data, width, alpha=opacity)
```

```
rects1[3].set_color('#872f29')
rects1[2].set_color('#ffcb59')
rects1[1].set_color('#d6ee39')
rects1[0].set_color('#6ade30')

ax.set_yticks(y_pos)
ax.set_yticklabels(cities)

# add some text for labels, title and axes ticks
ax.set_ylabel('Ciudades')
ax.set_xlabel('Km de carril bici')

ax.legend((rects1[0], rects1[1], rects1[2], rects1[3]), cities)

plt.show()
```



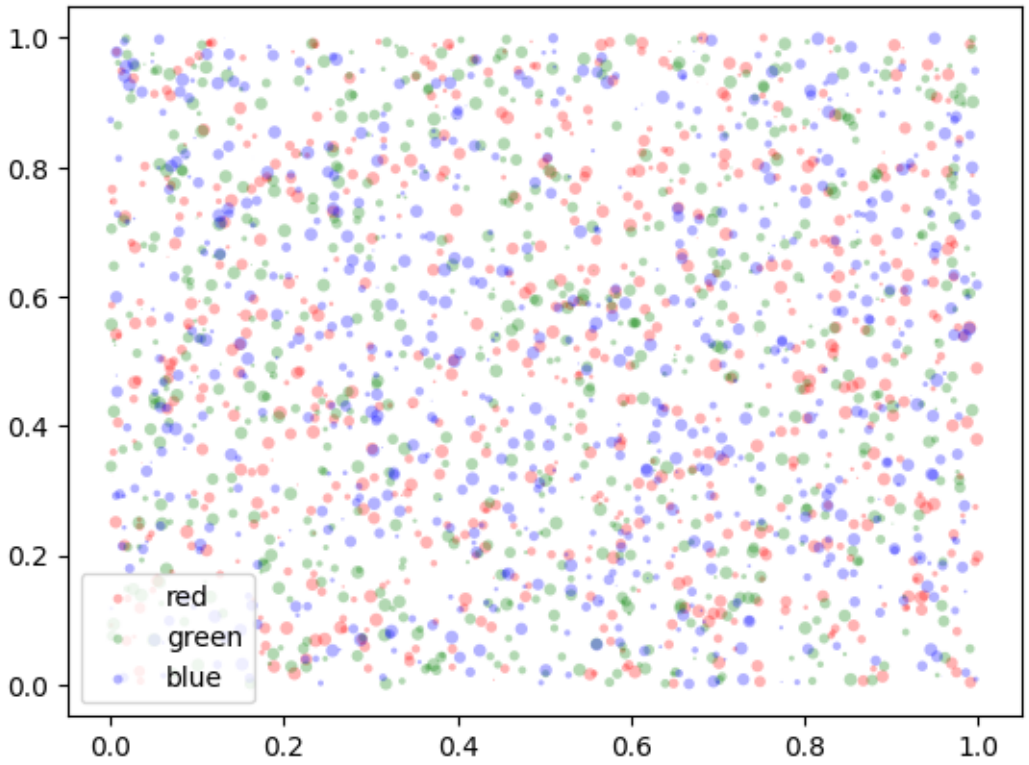
[20]:
```
# stack plot
x = [2008, 2009, 2010, 2011, 2012]
iOS = [35, 25, 20, 17, 18]
Android = [0, 15, 35, 45, 60]
WindowsPhone = [1, 3, 5, 5, 2]
Others = [64,57,40,33,20]
```

```
# y = np.vstack([iOS, Android, WindowsPhone,Others])
# print(y)

labels = ["iOS ", "Android", "WindowsPhone", "Others"]

fig, ax = plt.subplots()
ax.stackplot(x, iOS, Android, WindowsPhone, Others,labels=labels)
# ax.stackplot(x, y, labels=labels)
ax.set_xticks(x)
ax.legend(loc='upper left')
plt.show()
```



[21]:
```
# Scatter plot
from numpy.random import rand

fig, ax = plt.subplots()
n = 700
for color in ['red', 'green', 'blue']:
    x, y = rand(2, n)
    scale = 25.0 * rand(n)
    ax.scatter(x, y, c=color, s=scale, label=color,
        alpha=0.3, edgecolors='none')
```

```
ax.legend()
plt.show()
```



0.5.1 Grabar a archivo

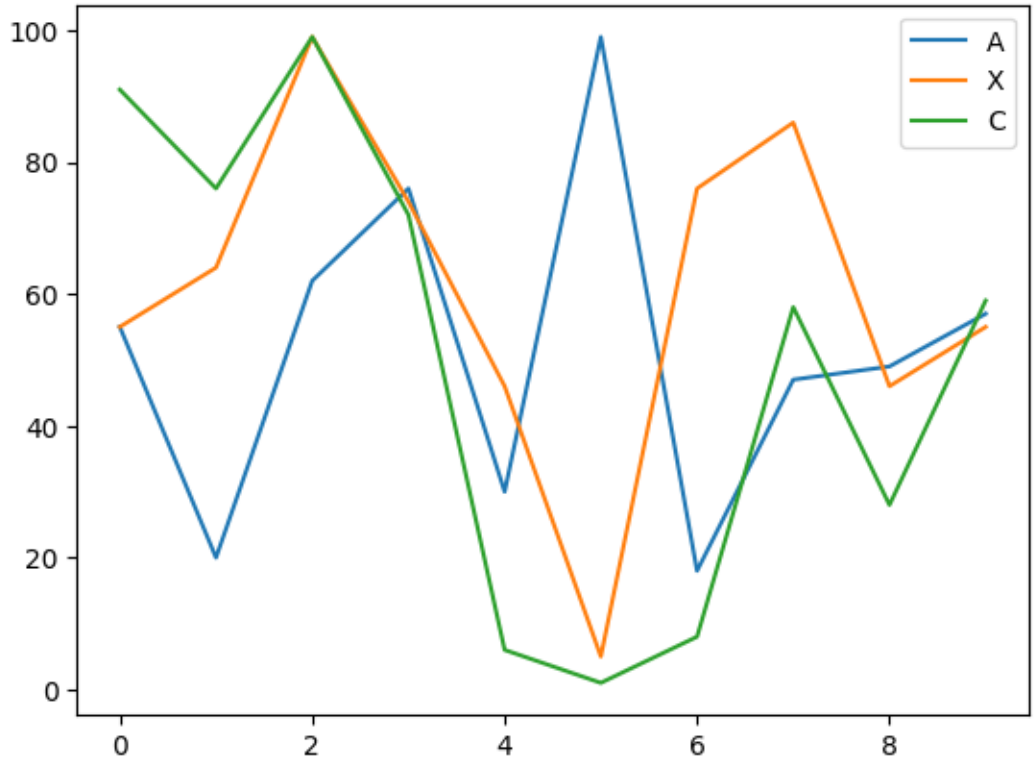
`figure.savefig(<filename>,dpi=None, facecolor='w', edgecolor='w', format=None, bbox_inches=None`

[22]:
```
# guardar la figura (puede ser múltiples gráficos)
import os
ruta = os.path.join("res", "o_figure.png")

fig.savefig(ruta, format='png') # varios formatos aceptados ('pdf',' png',␣
 ↪'svg', 'eps'...)
```

0.6 Representación gráfica en pandas

- pandas ofrece interfaz para dibujar Series y DataFrame
- Usa matplotlib internamente
- Fácil de usar

[23]:
```
rand_matrix = np.random.randint(100,size=(10,3))
frame = pd.DataFrame(rand_matrix, columns=list('AXC'))
frame.plot()
plt.show() #aúnesnecesariollamarashow()paramostrarlosgráficos //TODO encontrar
 ↪una maneradenousarpltdesdepandas

display(frame)#datos enejex,ejey,leyenda,series?
```



	A	X	C
0	55	55	91
1	20	64	76
2	62	99	99
3	76	74	72
4	30	46	6
5	99	5	1
6	18	76	8
7	47	86	58
8	49	46	28
9	57	55	59

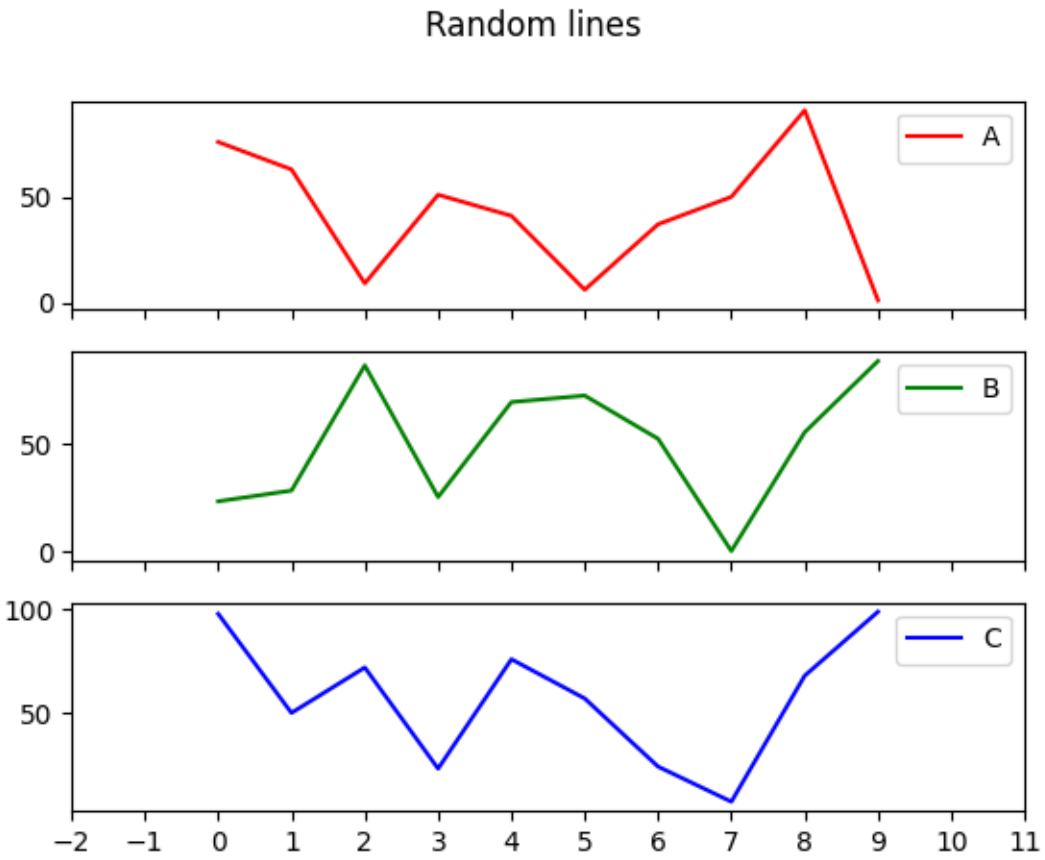
0.7 frame.plot()

```
frame.plot(x,y,label=<Series>,ax,style,kind,xticks,yticks,title,subplots)
```

- (soloparaSeries)label: referenciaparalaleyenda
- ax: subfigura en la que dibujar los datos
- style: estilo de la línea
- kind: tipo de gráfico ('bar', 'pie', 'hist', 'area' 'line', 'barh', 'density', 'kde').
- xticks y yticks: valores en los ejes X e Y
- title: string como título
- (solo para DataFrame) subplots: bool si se desean subfiguras separadas para cada columna.
- (solo para DataFrame) 'x' e 'y' se pueden utilizar para seleccionar columnas para cada eje

Lista completa de argumentos

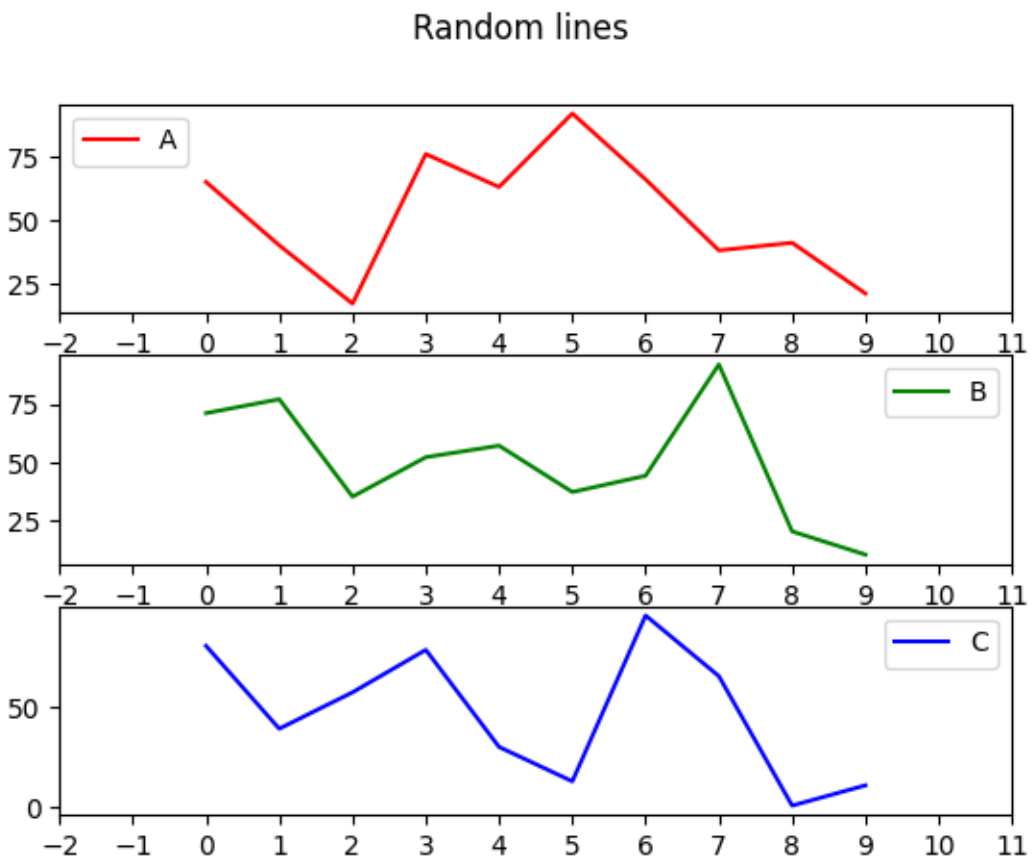
[24]:
```
rand_matrix = np.random.randint(100, size=(10,3))
frame = pd.DataFrame(rand_matrix, columns=list('ABC'))
frame.plot(style=['r-','g-','b-'], xticks=range(-2,12), title='Random␣
 ↪lines',subplots=True)
plt.show()
display(frame)
```



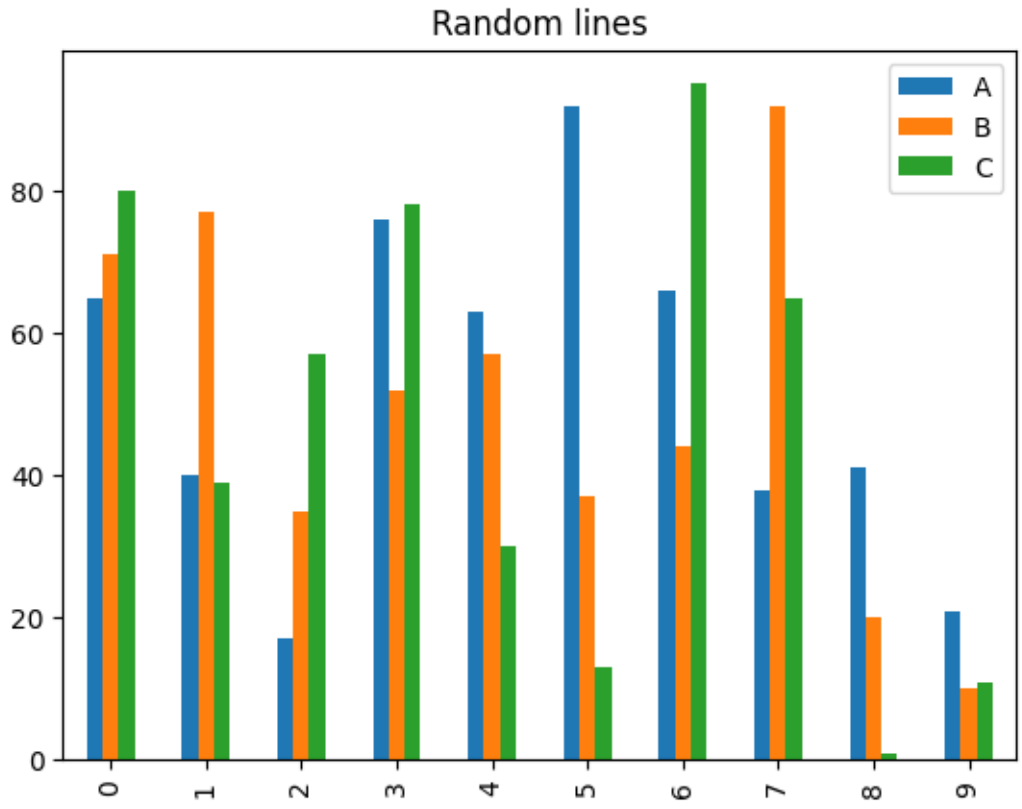
	A	B	C
0	76	23	98
1	63	28	50

```
2   9  86  72
3  51  25  23
4  41  69  76
5   6  72  57
6  37  52  24
7  50   0   7
8  91  55  68
9   1  88  99
```

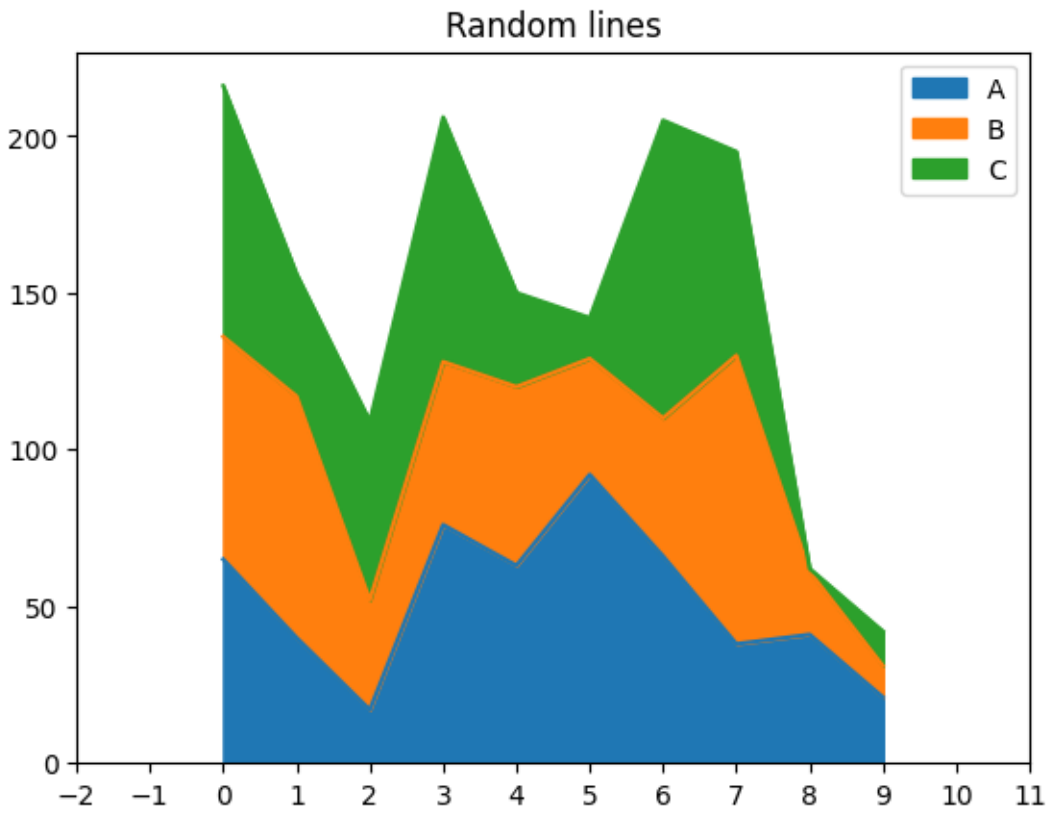
[25]:
```
# combinar pandas plot con matplotlib
rand_matrix = np.random.randint(100, size=(10,3))
frame = pd.DataFrame(rand_matrix, columns=list('ABC'))

fig, axis = plt.subplots(3,1)
ax = frame.plot(style=['r-','g-','b-'], xticks=range(-2,12), title='Random␣
 ↪lines', subplots=True, ax=axis)

ax[0].legend(loc='upper left')
plt.show()
```



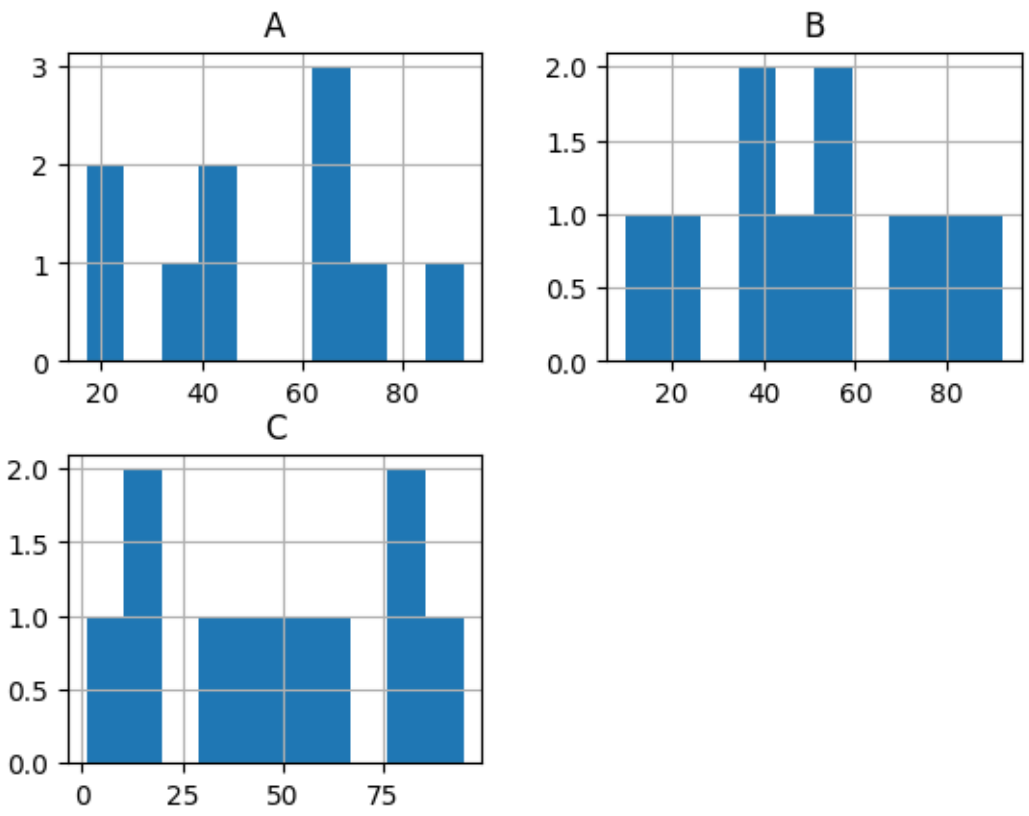
[26]:
```
# Gráfico de barras
frame.plot(kind='bar', title='Random lines')
plt.show()
```



[27]:
```
# Gráfico de área
frame.plot(kind='area', xticks=range(-2,12), title='Random lines')
plt.show()
```

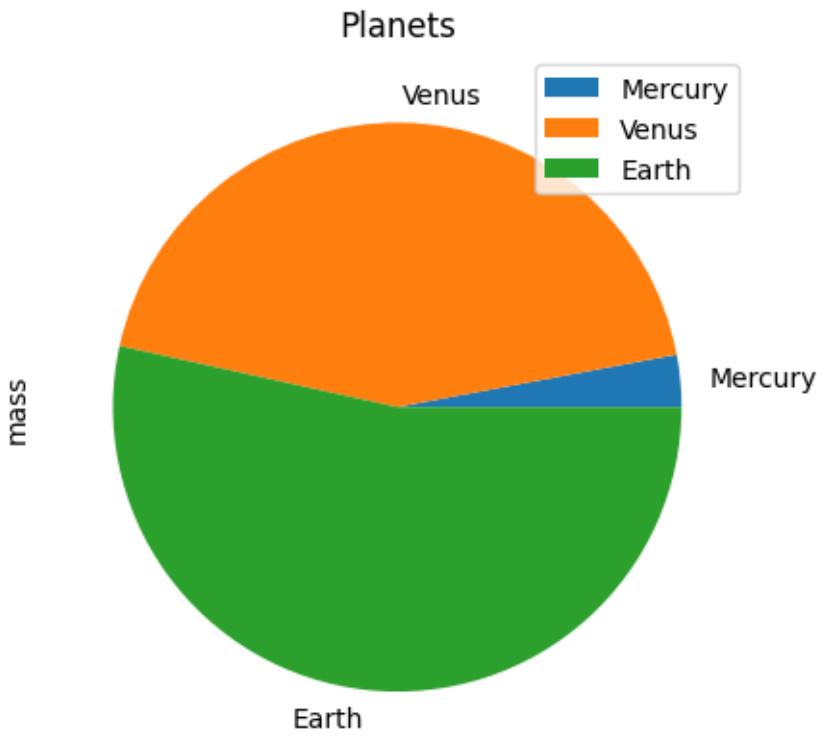


[28]:
```
# Histogramas
frame.hist()
plt.show()
```



[29]:
```
# Gráfico de pastel
df = pd.DataFrame({'mass': [0.330, 4.87 , 5.97],
                   'radius': [2439.7, 6051.8, 6378.1]},
                   index=['Mercury', 'Venus', 'Earth'])
display(df)
df.plot(y='mass', kind='pie', title='Planets') # y variable a usar
plt.show()
```

	mass	radius
Mercury	0.33	2439.7
Venus	4.87	6051.8
Earth	5.97	6378.1



0.8 Seaborn

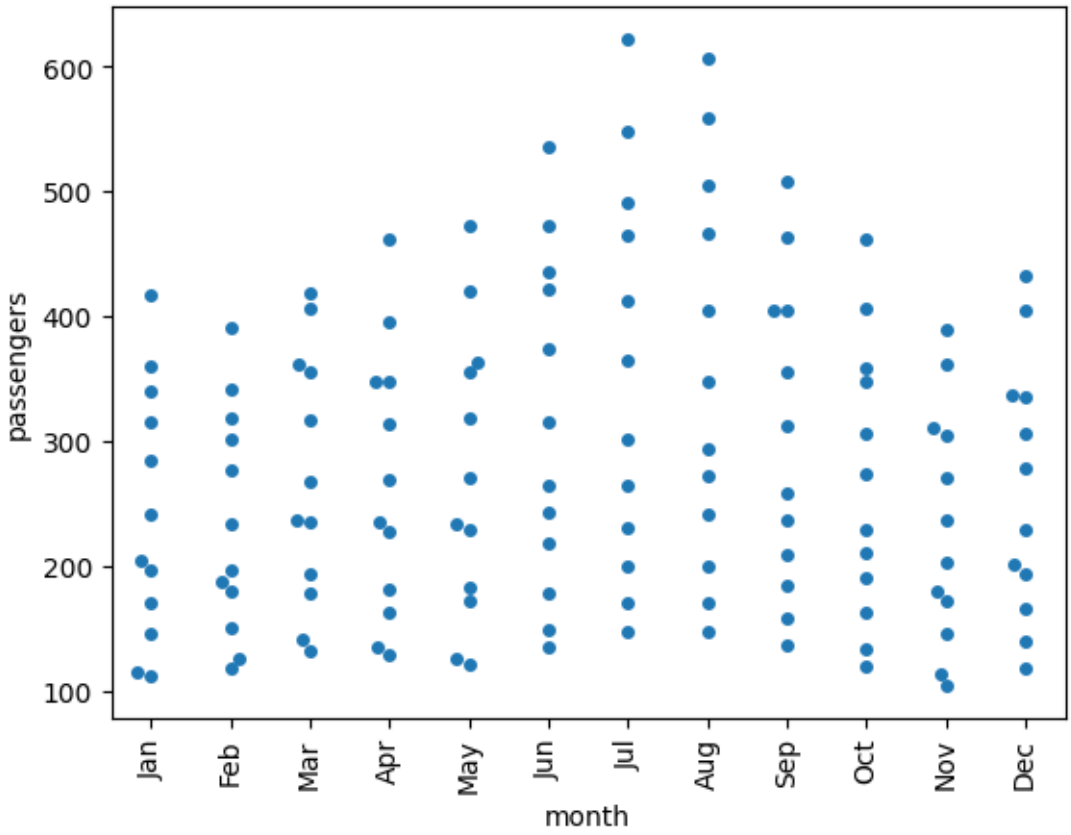
- Abstracción de matplotlib
- Facilidad de uso y personalización
- Excelente para exploración y visualización de relaciones entre variables

Viene con varios datasets predefinidos https://github.com/mwaskom/seaborn-data

Documentación oficial

[41]:
```
import seaborn as sns
```

[42]:
```
# Load data
flights_data = sns.load_dataset("flights")
# display(flights_data)

# Construct plot
sns.swarmplot(x="month", y="passengers", data=flights_data)
# Show plot
plt.xticks(rotation=90)
plt.show()
```



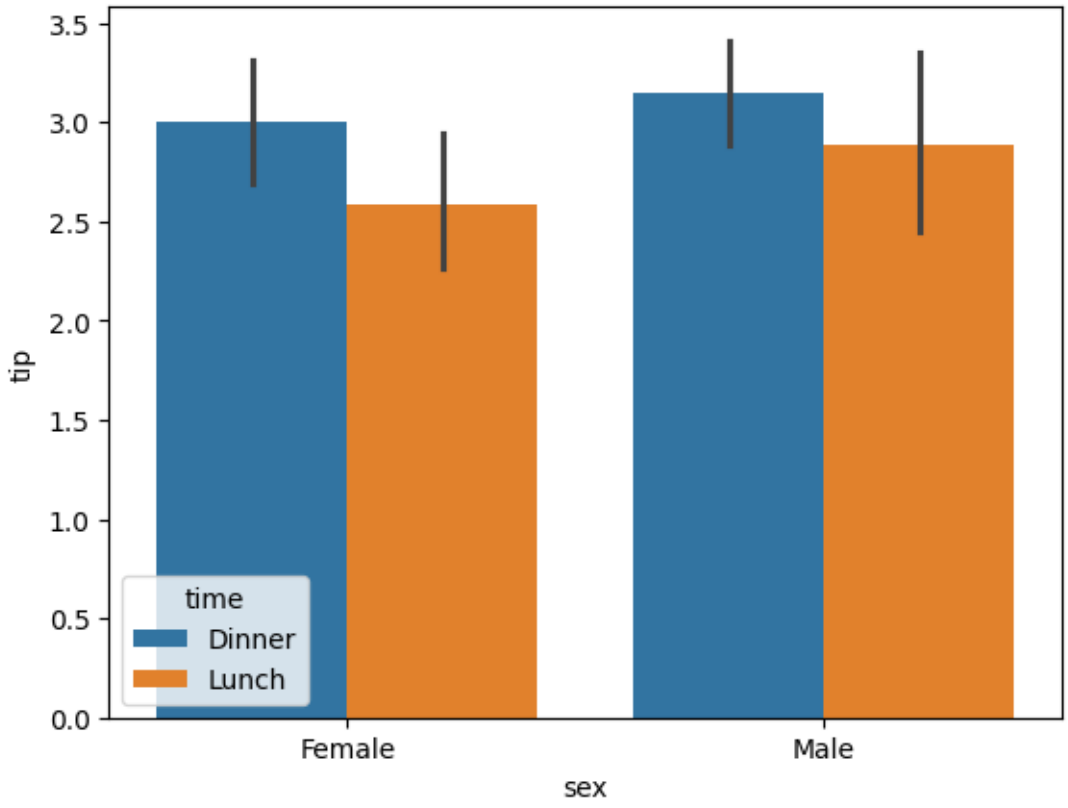
Compatible con pandas

[43]:
```
# Load in data
tips = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/
↪master/tips.csv")
tips.sample(5)
```

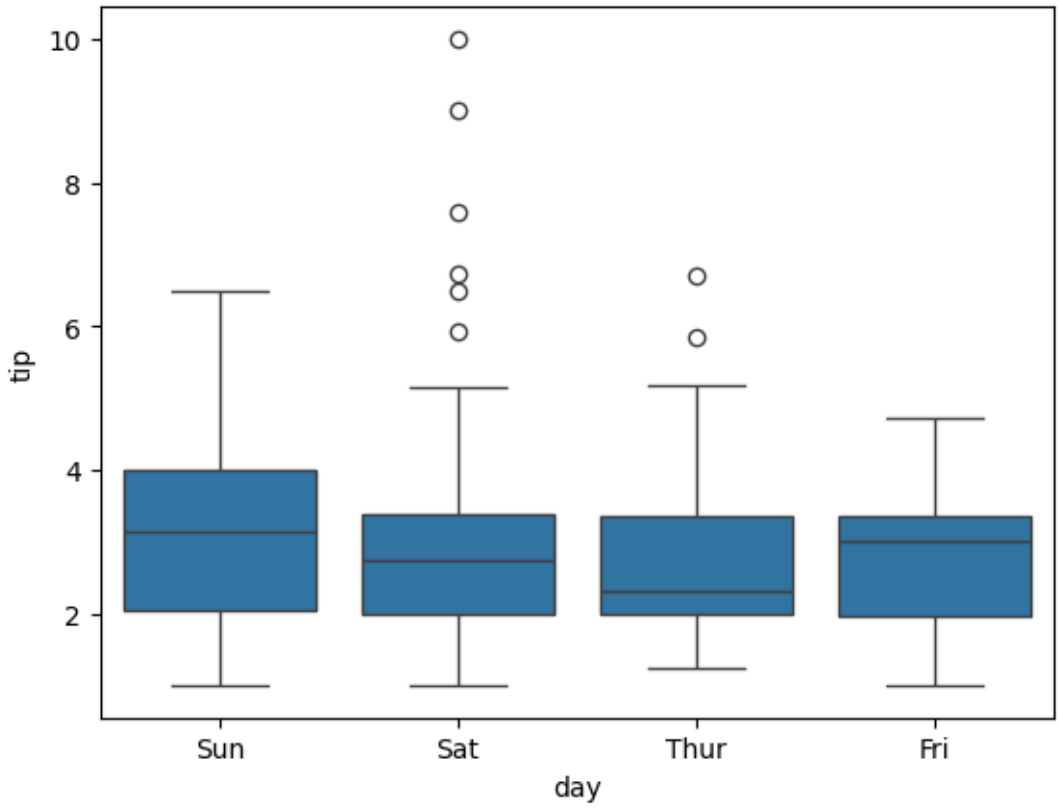
[43]:

	total_bill	tip	sex	smoker	day	time	size
25	17.81	2.34	Male	No	Sat	Dinner	4
48	28.55	2.05	Male	No	Sun	Dinner	3
73	25.28	5.00	Female	Yes	Sat	Dinner	2
194	16.58	4.00	Male	Yes	Thur	Lunch	2
57	26.41	1.50	Female	No	Sat	Dinner	2

[44]:
```
sns.barplot(x='sex', y='tip', hue='time', data=tips) # palette para color(negro␣
↪desviacion)
plt.show()
```



[45]:
```
# Boxplot (cuartiles)
sns.boxplot(x='day', y='tip', data=tips)
# sns.boxplot(x=tips['day'], y=tips.tip)
plt.show()
```



0.8.1 Relaciones entre variables

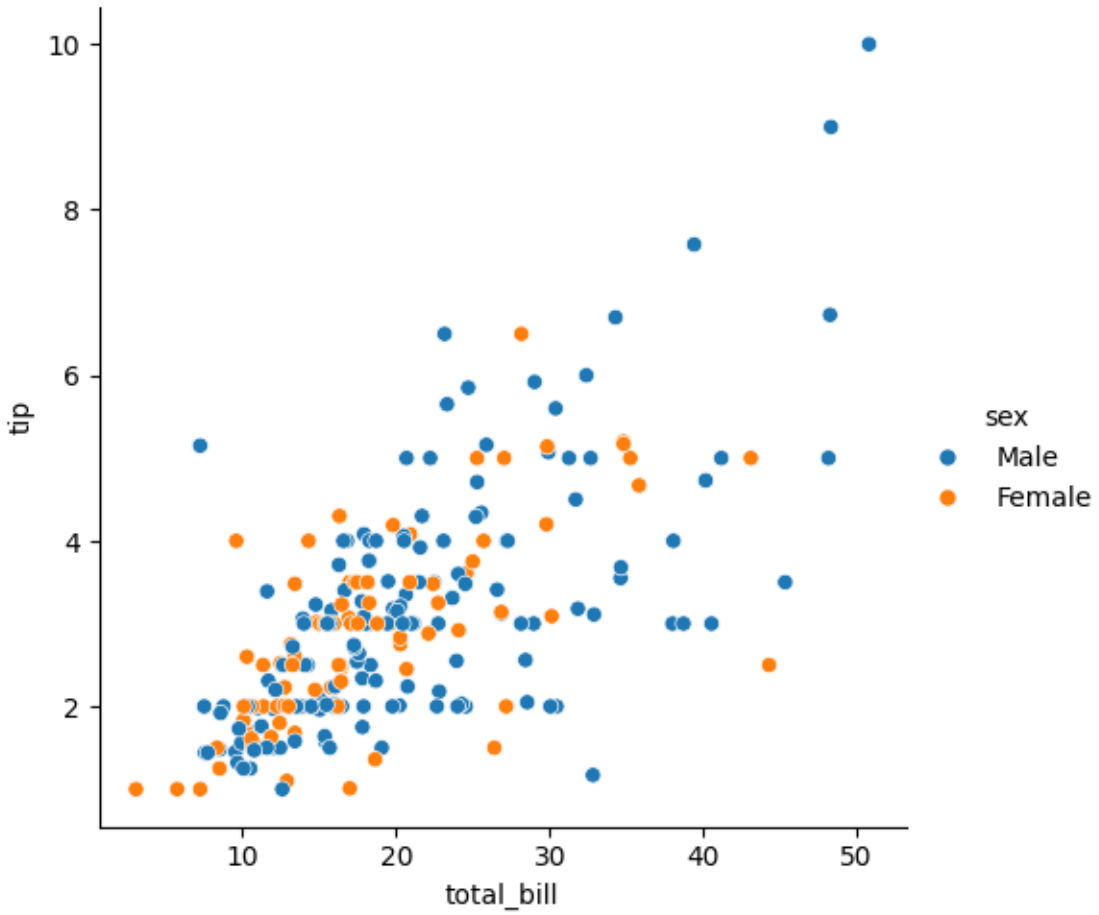
sns.relplot()

[46]:
```
tips = sns.load_dataset("tips")
tips.sample(5)
```

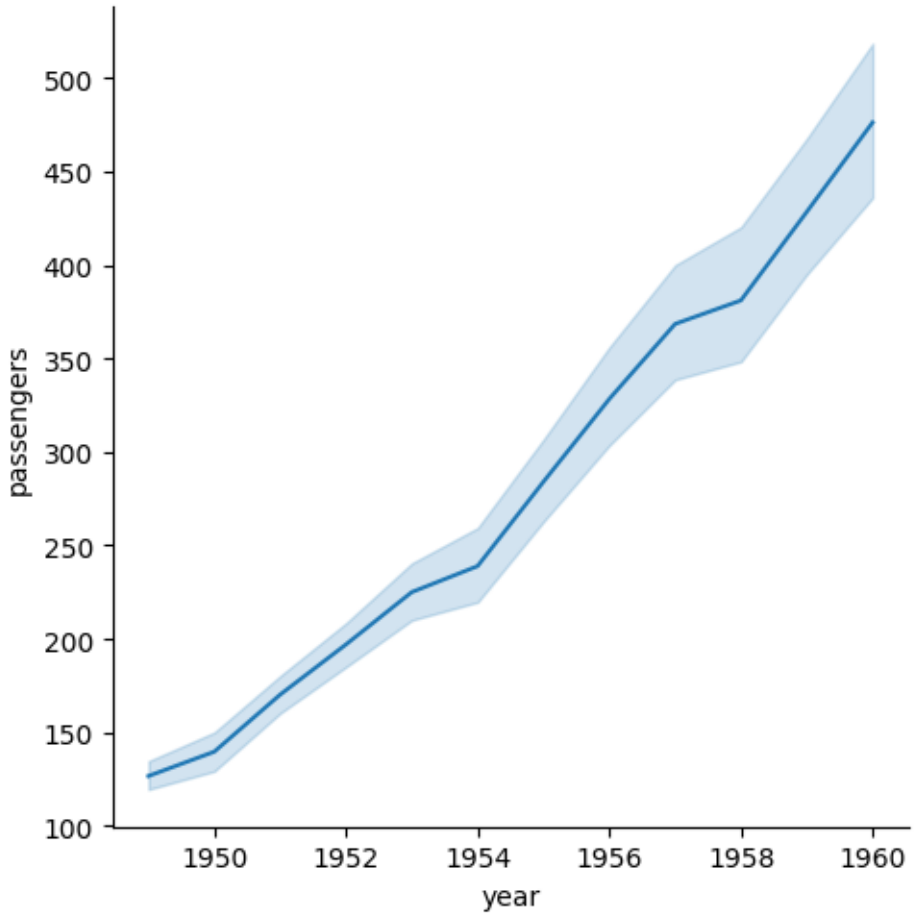
[46]:

	total_bill	tip	sex	smoker	day	time	size
111	7.25	1.00	Female	No	Sat	Dinner	1
26	13.37	2.00	Male	No	Sat	Dinner	2
76	17.92	3.08	Male	Yes	Sat	Dinner	2
78	22.76	3.00	Male	No	Thur	Lunch	2
182	45.35	3.50	Male	Yes	Sun	Dinner	3

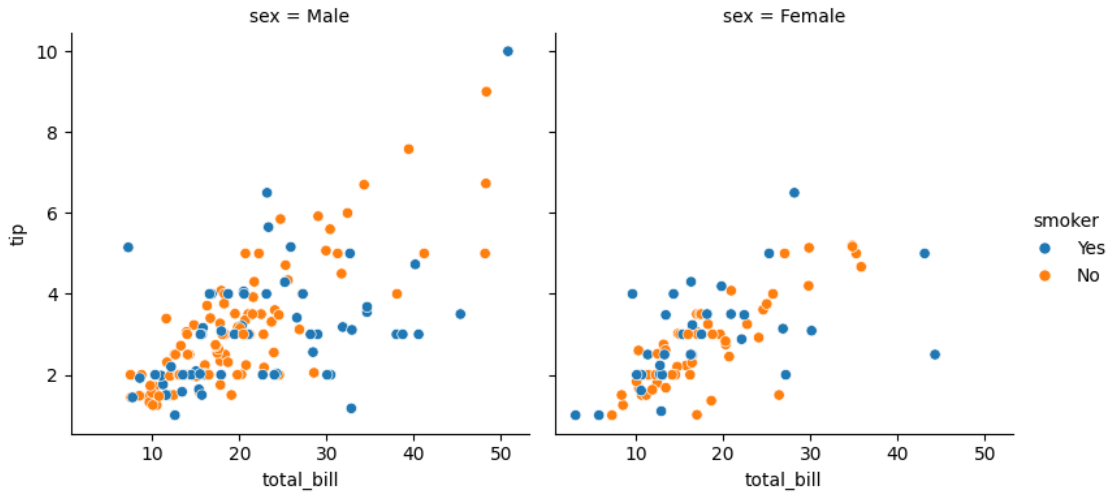
[47]:
```
sns.relplot(x='total_bill', y='tip', hue='sex', data=tips, kind='scatter')
plt.show()
```



[48]:
```
# representar la media y la variación de una variable en función de otra
data = sns.load_dataset('flights')
sns.relplot(x="year", y="passengers", kind="line", data=data)
plt.show()
```

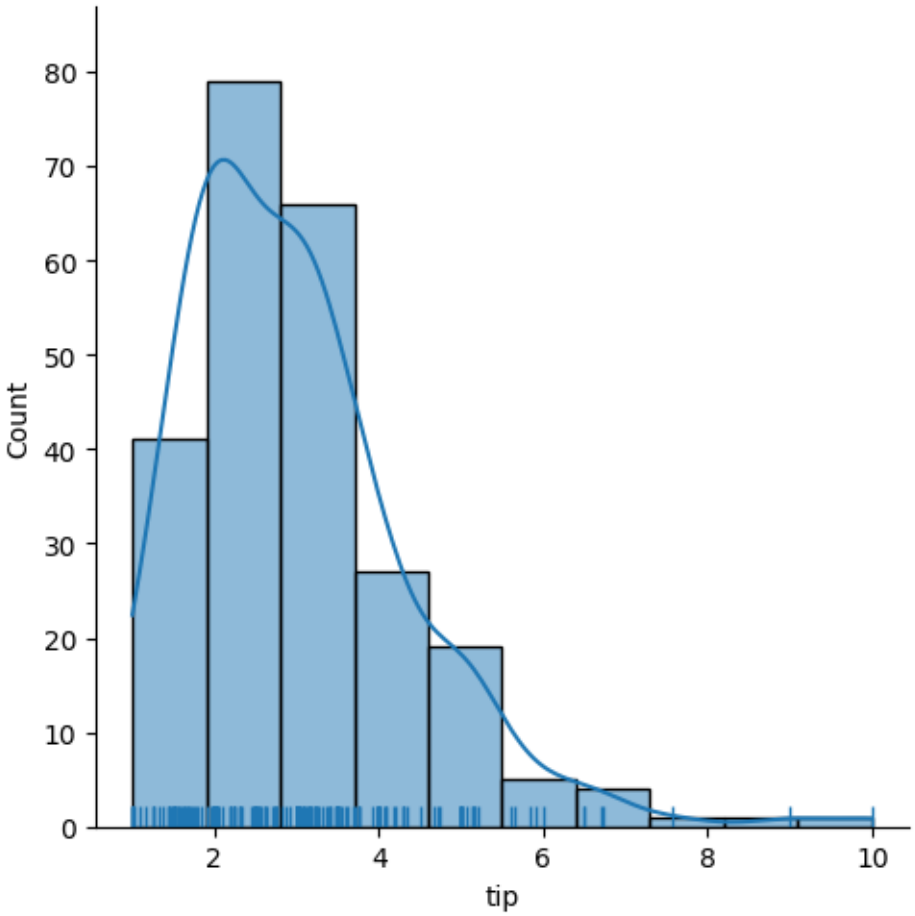


[49]:
```
# relacionar hasta 4 variables automáticamente
sns.relplot(x="total_bill", y="tip", hue="smoker",
            col="sex", height=4,
            kind="scatter",data=tips); #kind ('line','scatter'...)
plt.show()
```

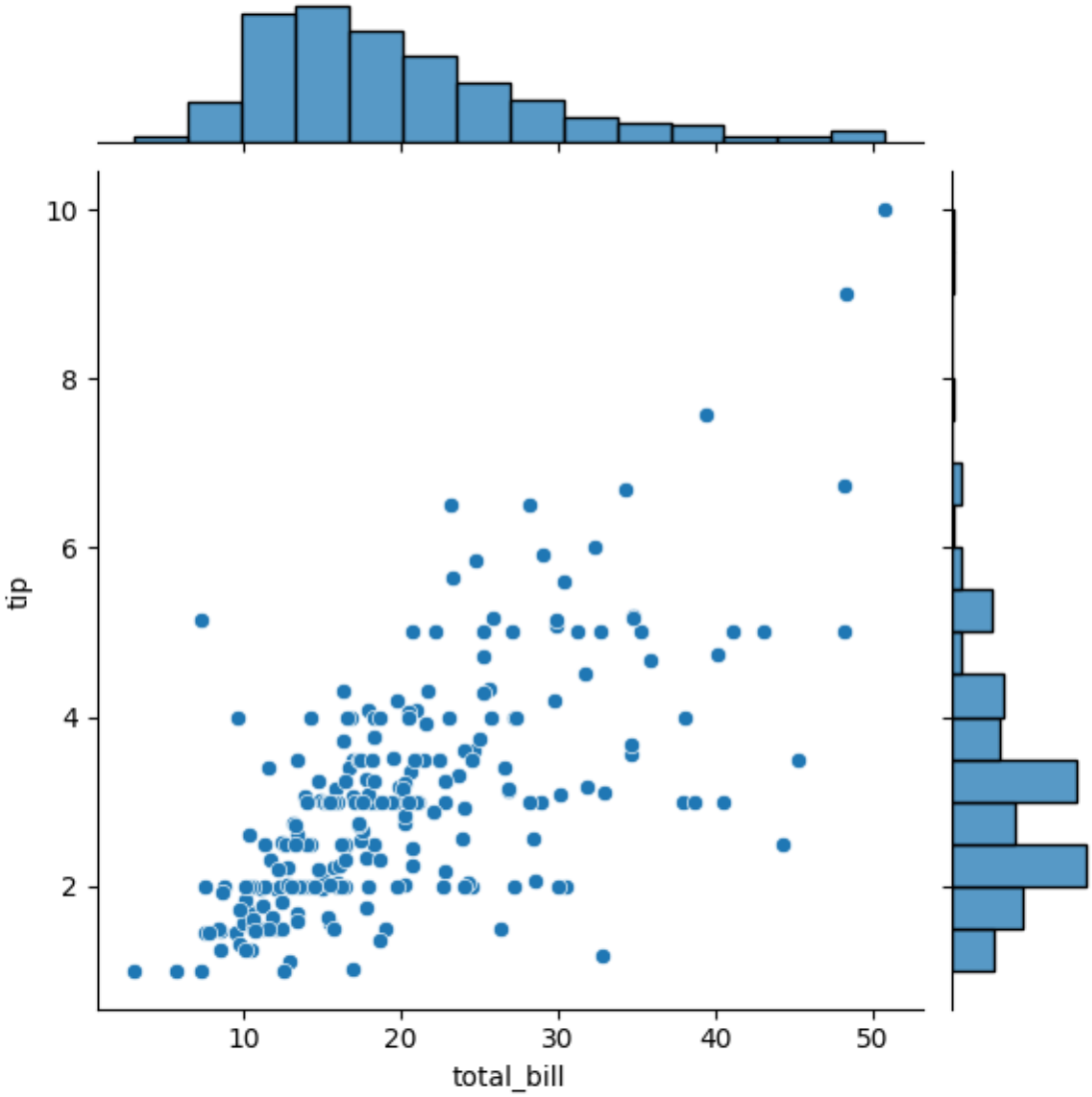


0.8.2 Distribución de un dataset

[50]:
```
# histograma y density plot
# kde = gaussian kernel density estimate, linea de densidad
# bins = tamaño de los contenedores
# rug = densidad de los datos
sns.displot(tips['tip'], kde=True, bins=10, rug=True)
plt.show()
```



[51]:
```
# Distribución bi-variable
sns.jointplot(x='total_bill', y='tip', data=tips)
plt.show()
```



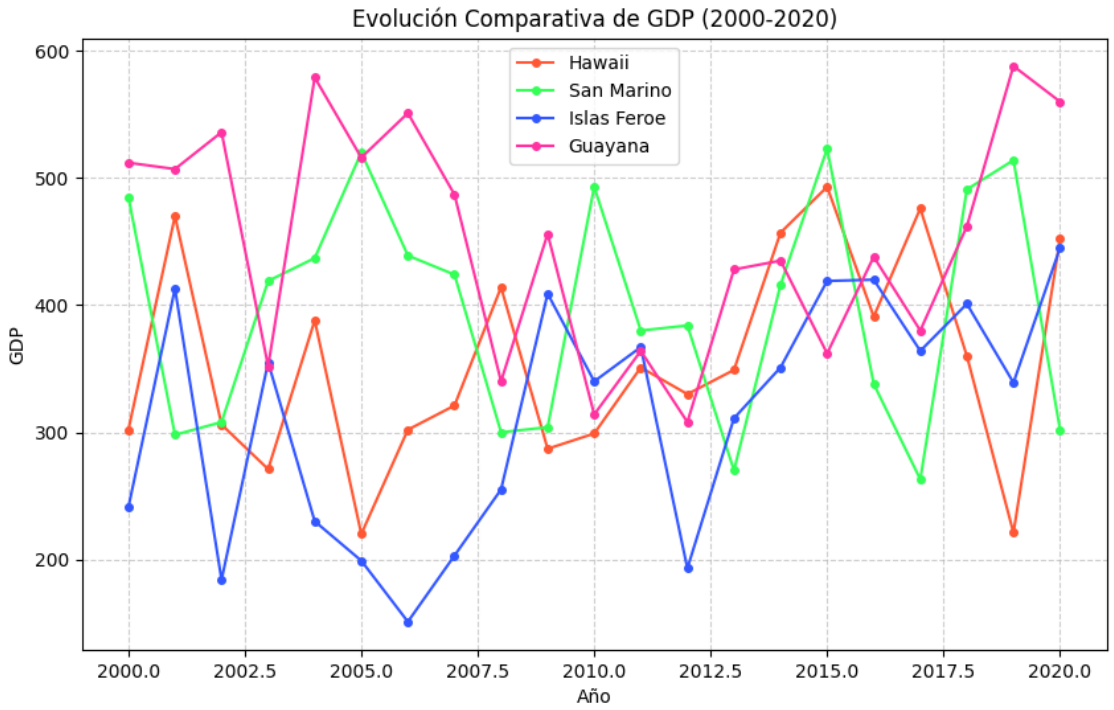
0.9 Ejercicios

[52]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np



#Crear lista de años (2000 a 2020)
years = np.arange(2000, 2021)
```

```
#Generar datos aleatorios
np.random.seed(42)  # Semilla para reproducibilidad
y0 = np.random.randint(200, 500, size=len(years))
y1 = np.random.randint(200, 500, size=len(years)) + 50
y2 = np.random.randint(200, 500, size=len(years)) - 50
y3 = np.random.randint(200, 500, size=len(years)) + 100

datos = [y0, y1, y2, y3]
nombres = ["Hawaii", "San Marino", "Islas Feroe", "Guayana"]
colores = ['#FF5733', '#33FF57', '#3357FF', '#FF33A1']

#Representar las 4 secuencias en una sola figura
plt.figure(figsize=(10, 6))
for i in range(4):
    plt.plot(years, datos[i], label=nombres[i], color=colores[i], marker='o',␣
 ↪markersize=4)

plt.title("Evolución Comparativa de GDP (2000-2020)")
plt.xlabel("Año")
plt.ylabel("GDP")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

#Representar en 4 subfiguras distintas
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
axes_flat = axes.flatten()

for i, ax in enumerate(axes_flat):
    # Gráfico
    ax.plot(years, datos[i], color=colores[i], label=nombres[i])

    # Info
    ax.set_title("Datos aleatorios")
    ax.set_xlabel("Año")
    ax.set_ylabel("GDP")
    ax.legend(loc='upper left')
    ax.grid(True, alpha=0.3)

    # 5. Añadir anotación
    if i == 0:
        max_val = np.max(datos[i])
        max_year = years[np.argmax(datos[i])]
        ax.annotate(f'Máximo: {max_val}',
                    xy=(max_year, max_val),
                    xytext=(max_year - 5, max_val + 50),
                    arrowprops=dict(facecolor='black', shrink=0.05),
```

```
                    fontsize=10, color='darkred')

plt.tight_layout()
plt.show()



#  SEABORN


# Cargar dataset (usamos 'penguins', distinto a tips/iris/flights)
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/penguins.
 ↪csv"
df_penguins = pd.read_csv(url)

print("Primeras filas del dataset:")
display(df_penguins.head())

# Representar visualmente la relación de cada pareja de atributos
# Usamos pairplot para ver todas las relaciones
sns.pairplot(df_penguins, hue="species", palette="viridis")
plt.suptitle("Relación entre parejas de atributos", y=1.02)
plt.show()

# Escoger una pareja y demostrar su correlación
# Elegimos longitud de aleta vs masa corporal
plt.figure(figsize=(8, 6))
sns.regplot(x="flipper_length_mm", y="body_mass_g", data=df_penguins,␣
 ↪line_kws={"color": "red"})
plt.title("Correlación: Largo de Aleta vs Masa Corporal")
plt.grid(True, alpha=0.3)
plt.show()

# 4. Representar dos variables numéricas en base a otra categórica
plt.figure(figsize=(10, 6))
sns.scatterplot(x="bill_length_mm", y="bill_depth_mm", hue="island",␣
 ↪style="species", s=100, data=df_penguins)
plt.title("Dimensiones del Pico por Isla y Especie")
plt.show()

#Gráficos de distribución
# Distribución de valores (media, outliers) de una variable continua (Masa␣
 ↪Corporal)
plt.figure(figsize=(8, 5))
sns.boxplot(x=df_penguins["body_mass_g"], color="lightblue")
plt.title("Distribución de Masa Corporal (Detectando Outliers)")
plt.xlabel("Masa Corporal (g)")
```

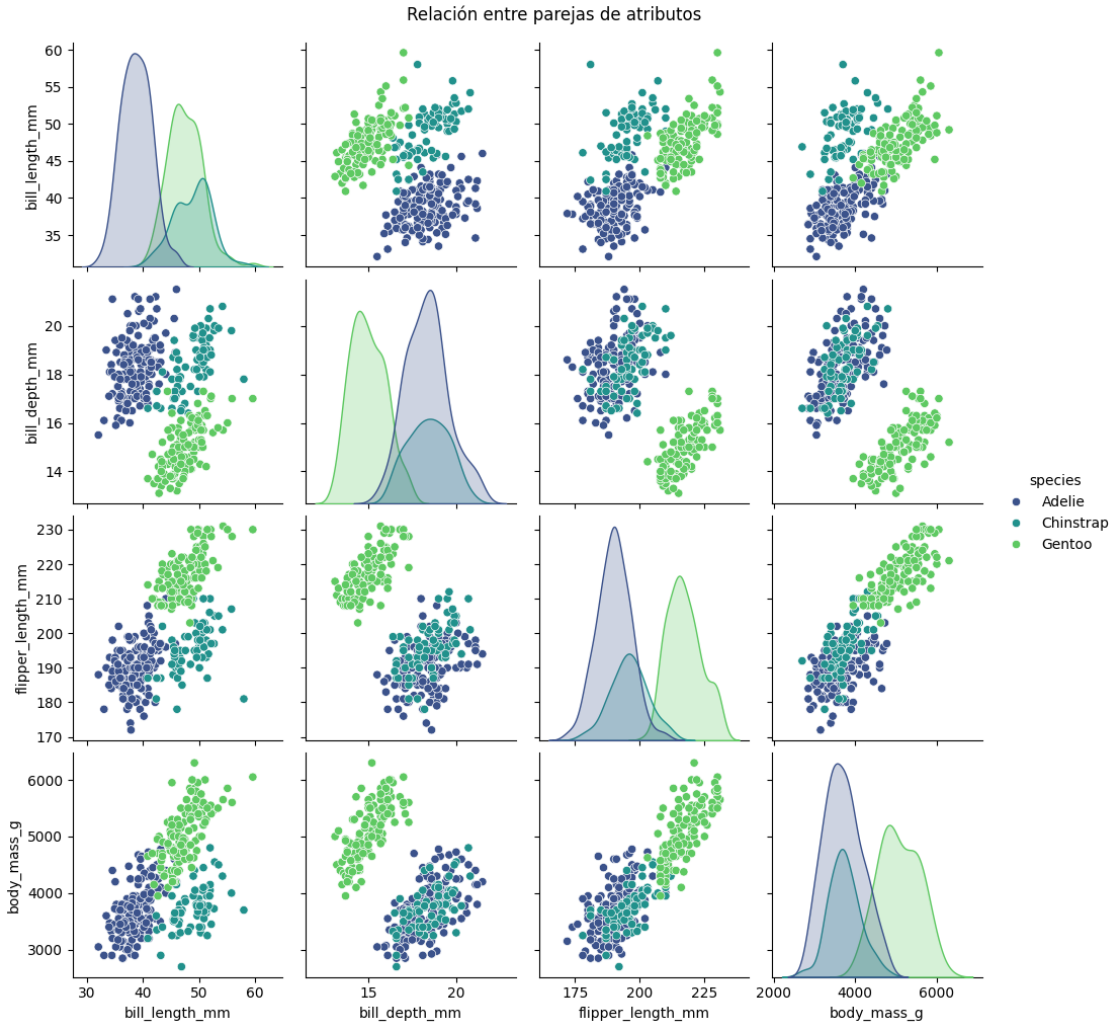
```
plt.show()

# Comparar distribución de esa variable con otra para densidad
plt.figure(figsize=(10, 6))
sns.kdeplot(df_penguins["body_mass_g"], fill=True, label="Masa Corporal",␣
 ↪alpha=0.5)
# Para comparar magnitudes distintas, a veces es mejor normalizar o usar ejes␣
 ↪dobles,
# Aquí comparamos la densidad de longitud de pico vs profundidad (ambas en mm)
sns.kdeplot(df_penguins["bill_length_mm"], fill=True, label="Largo Pico (mm)",␣
 ↪alpha=0.5)
sns.kdeplot(df_penguins["bill_depth_mm"], fill=True, label="Profundidad Pico␣
 ↪(mm)", alpha=0.5)
plt.title("Comparación de Densidad: Largo vs Profundidad del Pico")
plt.legend()
plt.show()

#Regresión lineal de orden > 1 .
plt.figure(figsize=(8, 6))
sns.regplot(x="bill_length_mm", y="body_mass_g", data=df_penguins, order=2,
            scatter_kws={"alpha": 0.5}, line_kws={"color": "darkorange"})
plt.title("Regresión Polinómica (Orden 2): Largo Pico vs Masa")
plt.show()
```

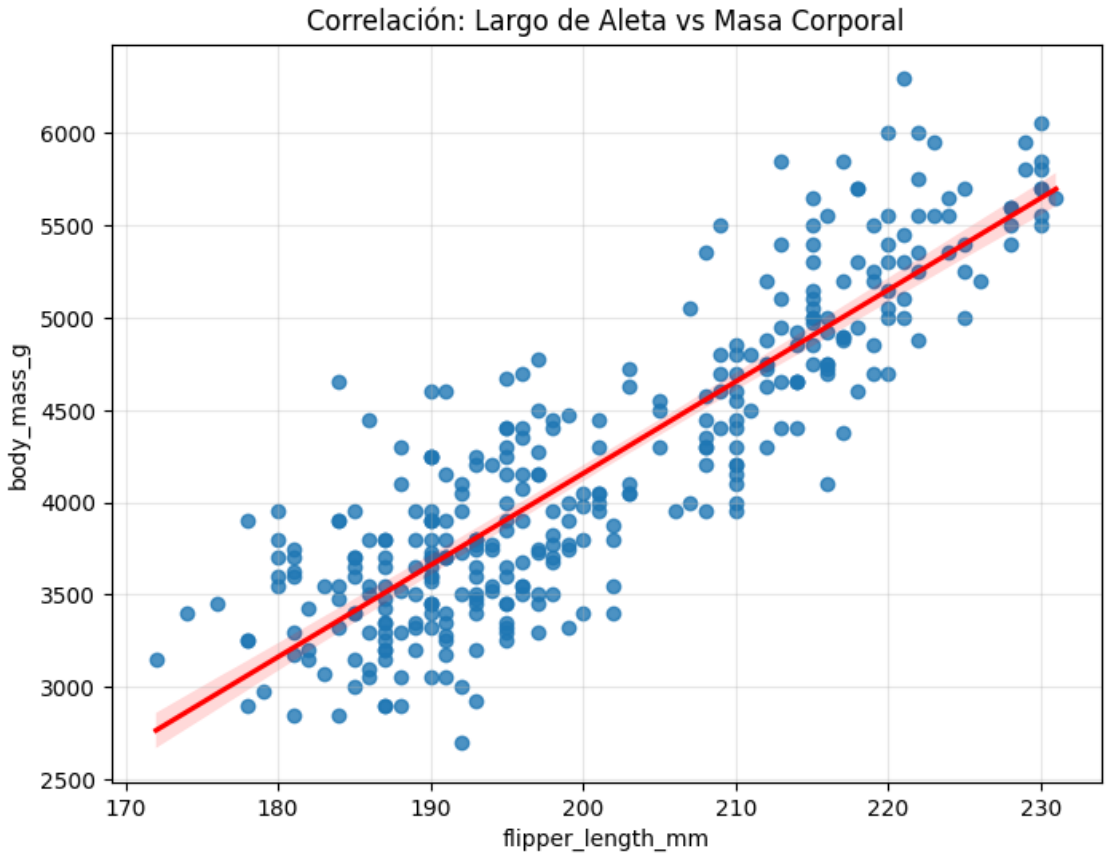


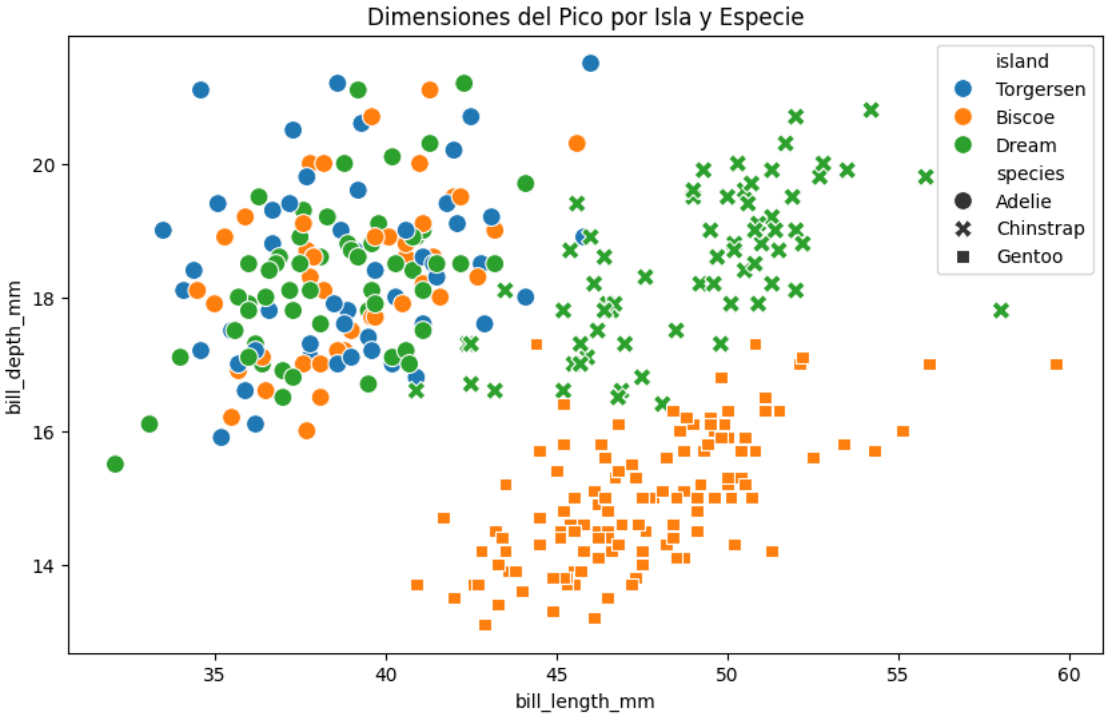
Primeras filas del dataset:

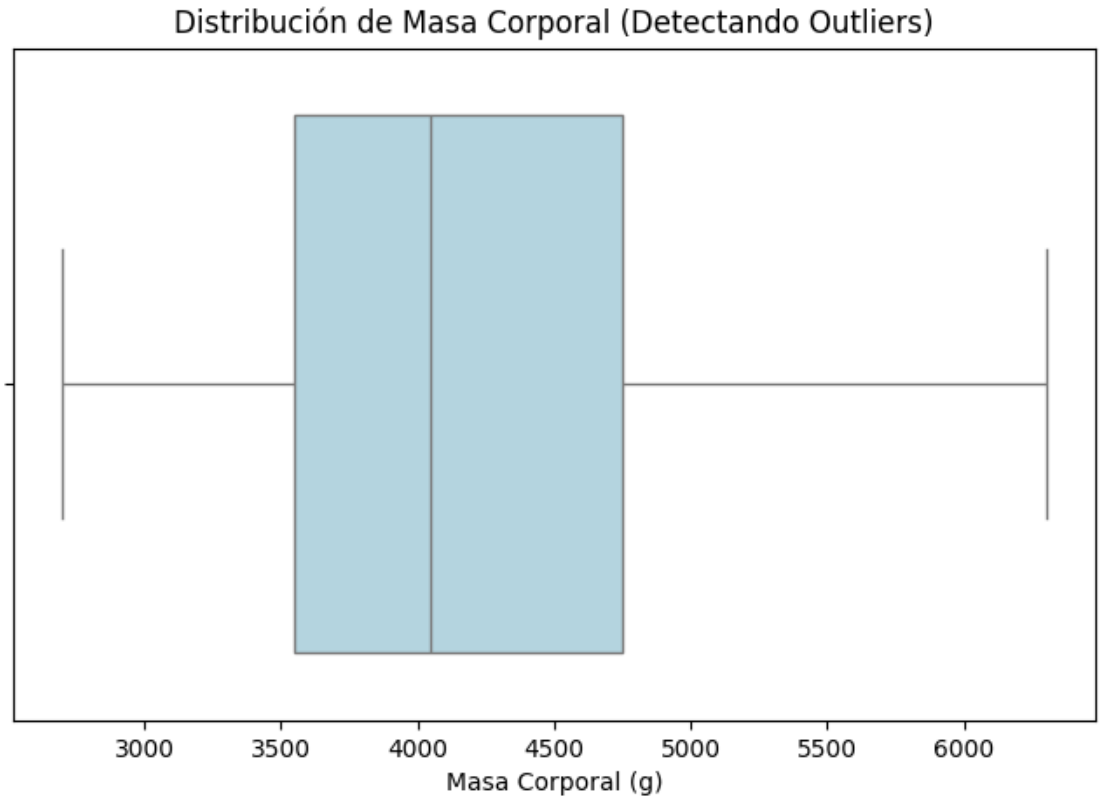
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	\
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	

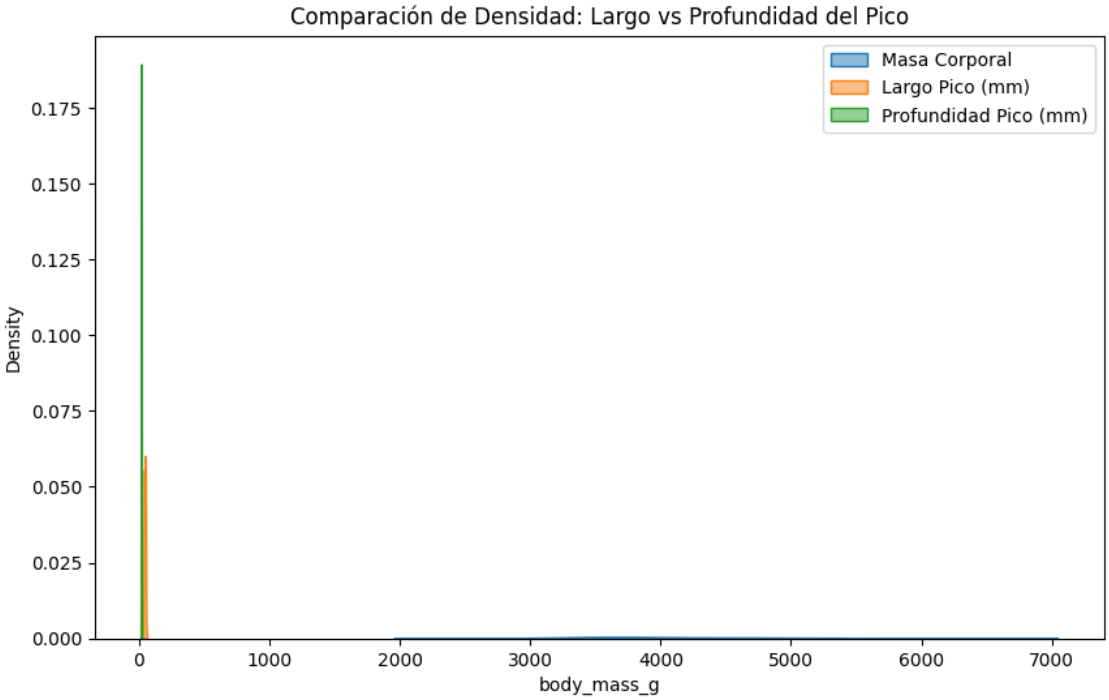
	body_mass_g	sex
0	3750.0	MALE
1	3800.0	FEMALE
2	3250.0	FEMALE
3	NaN	NaN
4	3450.0	FEMALE

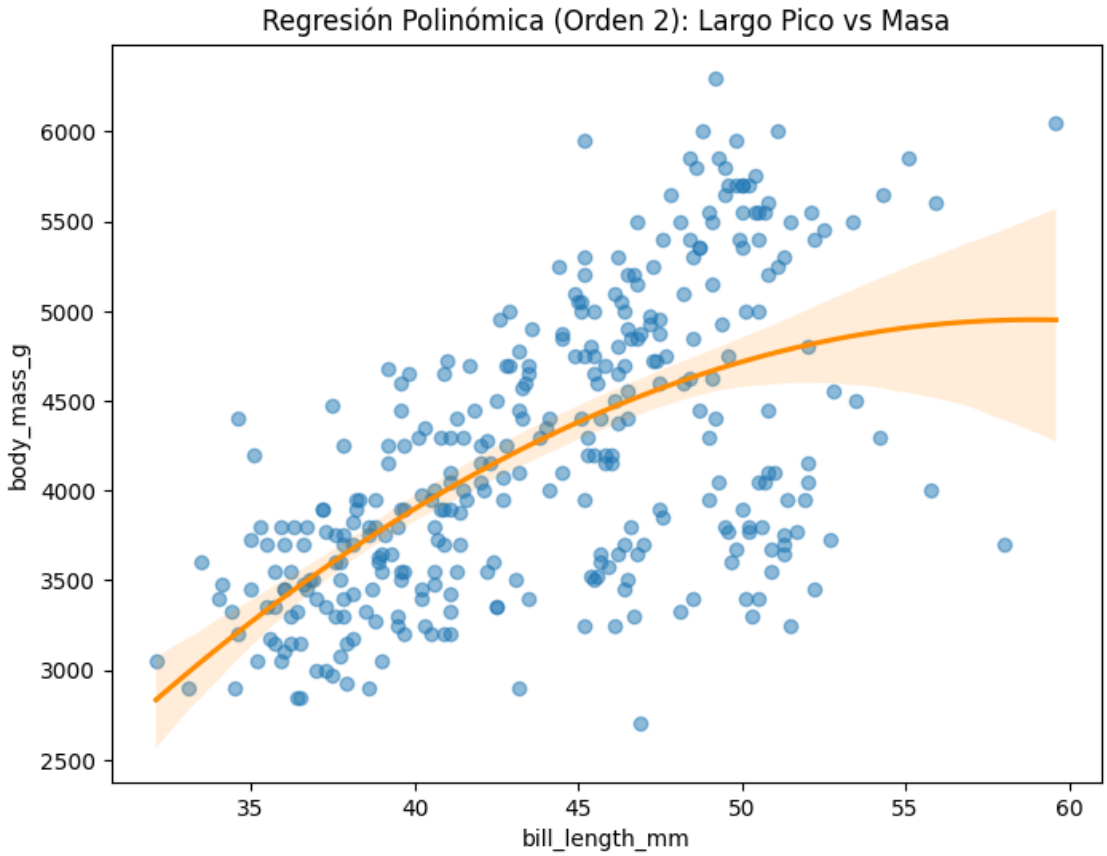

Relación entre parejas de atributos
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g
species
Adelie
Chinstrap
Gentoo


Correlación: Largo de Aleta vs Masa Corporal
body_mass_g
flipper_length_mm
2500
3000
3500
4000
4500
5000
5500
6000
170
180
190
200
210
220
230


Dimensiones del Pico por Isla y Especie
bill_depth_mm
bill_length_mm
island
Torgersen
Biscoe
Dream
species
Adelie
Chinstrap
Gentoo
14
16
18
20
35
40
45
50
55
60


Distribución de Masa Corporal (Detectando Outliers)
3000
3500
4000
4500
5000
5500
6000
Masa Corporal (g)


Comparación de Densidad: Largo vs Profundidad del Pico
Masa Corporal
Largo Pico (mm)
Profundidad Pico (mm)
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
1000
2000
3000
4000
5000
6000
7000
body_mass_g


Regresión Polinómica (Orden 2): Largo Pico vs Masa
body_mass_g
6000
5500
5000
4500
4000
3500
3000
2500
35
40
45
50
55
60
bill_length_mm

0.10 Repostorio

GitHub: https://github.com/reromash1/machines2026

[ ]: