INSTITUTO SUPERIOR TECNOLÓGICO QUITO

Excelencia en Educación Superior

0.1 Tercer Modulo- Python , Representación Gráfica

python™

Erick Mejia

Representación Gráfica

- matplotlib básico (trama, leyenda, decoradores y anotaciones, grabar)
- dibujo con pandas (tipos de gráficos)
- introducción 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

1. matplotlib

Librería estándar de representación gráfica

• Librería estándar de representación gráfica

• Low level: fácil de usar, difícil de dominar

• Control

• Gráficos y diagramas

1.0.1 Elementos de visualización

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

In [84]:
```
# Importar matplotlib
import matplotlib.pyplot as plt
```

In [85]:
```
import numpy as np
import pandas as pd
```

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

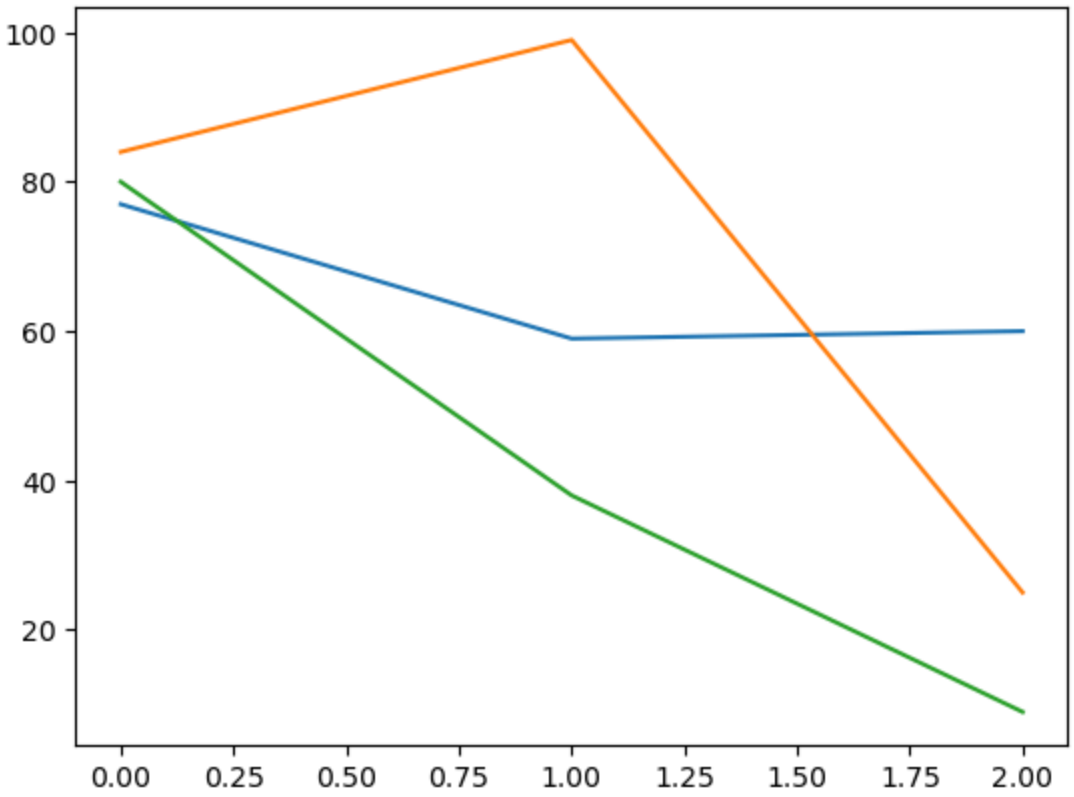
In [86]:
```
%matplotlib --list
```

```
Available matplotlib backends: ['agg', 'auto', 'cairo', 'gtk3', 'gtk3agg', 'gtk3cair
o', 'gtk4', 'gtk4agg', 'gtk4cairo', 'inline', 'ipympl', 'macosx', 'module://matplotl
ib_inline.backend_inline', 'nbagg', 'notebook', 'osx', 'pdf', 'pgf', 'ps', 'qt', 'qt
5', 'qt5agg', 'qt5cairo', 'qt6', 'qtagg', 'qtcairo', 'svg', 'template', 'tk', 'tkag
g', 'tkcairo', 'webagg', 'widget', 'wx', 'wx', 'wxagg', 'wxcairo']
```

In [87]:
```
# magic command para visualizar gráficos en jupyter en modo no interactivo
%matplotlib inline
```

In [88]:
```
data = np.random.randint(100,size=(3,3))
display(data)
plt.plot(data) # creación del gráfico
plt.show() # comando que
```

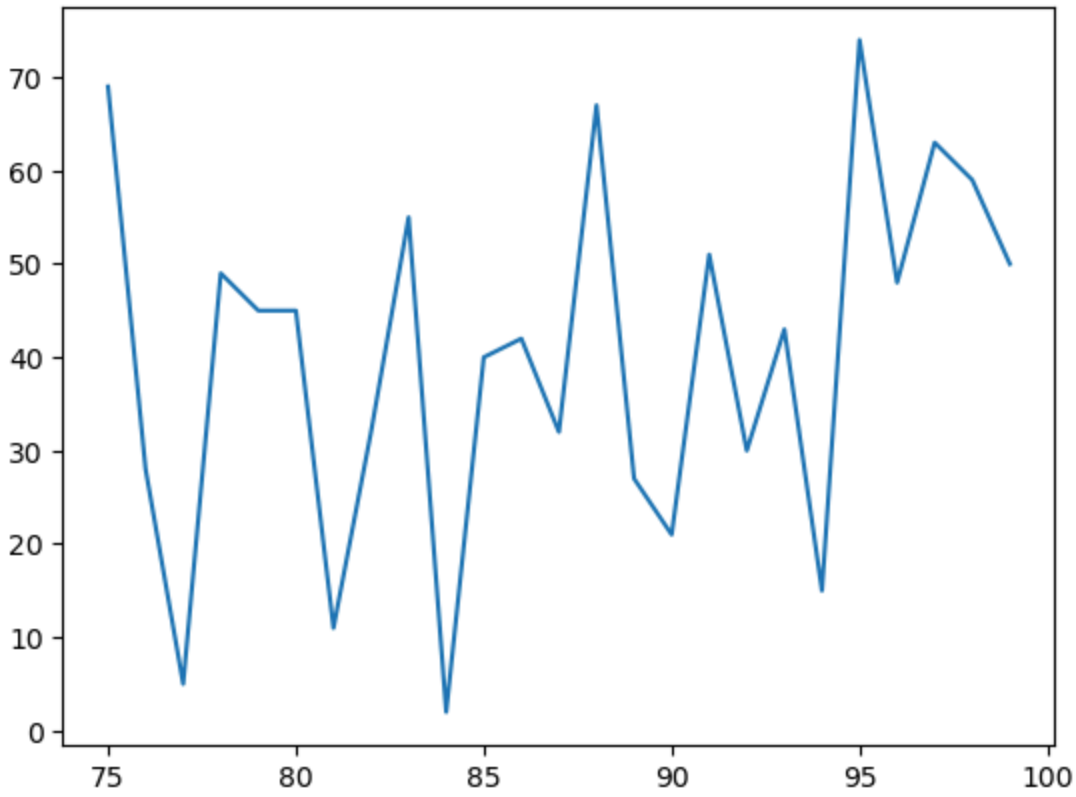
```
array([[77, 84, 80],
       [59, 99, 38],
       [60, 25,  9]], dtype=int32)
```



In [89]:
```
# activar modo interactivo
%matplotlib notebook
plt.ioff() # evitar que se sobreescriban los gráficos
%matplotlib inline
```

In [90]:
```
# 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] [69 28  5 49 45 45 11 32 55  2 40 42 32 67 27 21 51 30 43 15 74 48 63 59
 50]
```

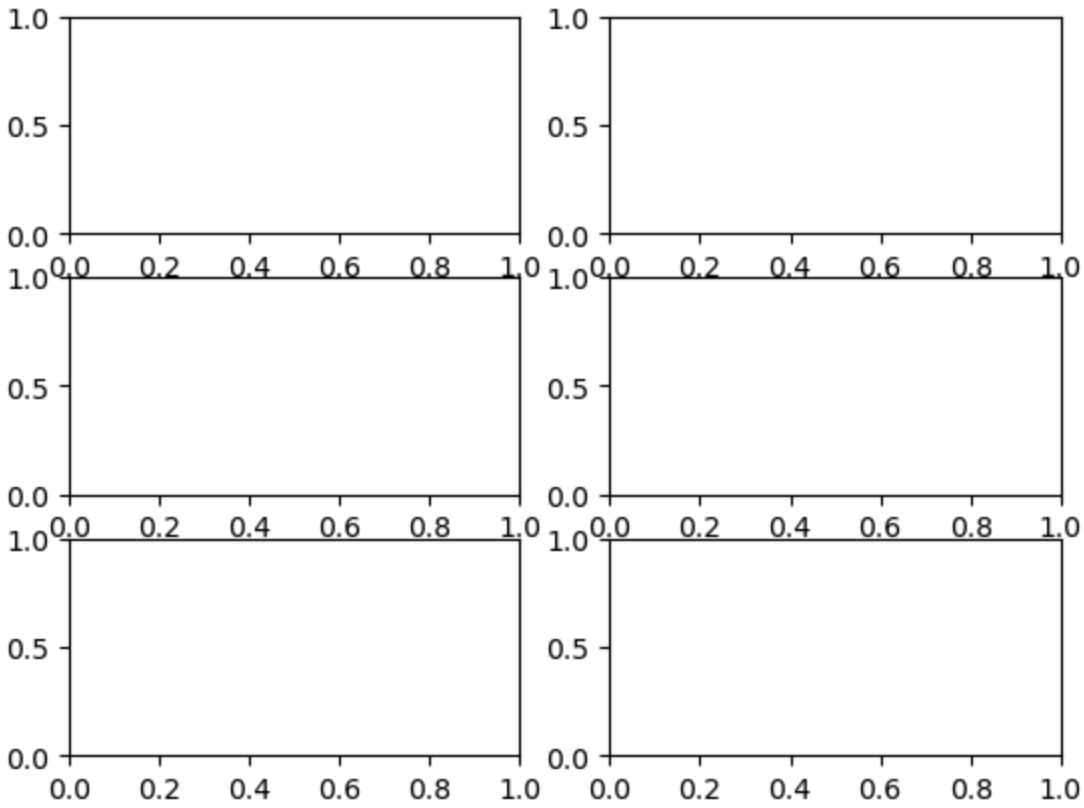


1.1 Figura y Ejes

- **Figura**: objeto en el que residen todos los gráficos (uno o múltiple)
- **Ejes**: cada uno de los subtramas (o gráficos) dentro de Figura

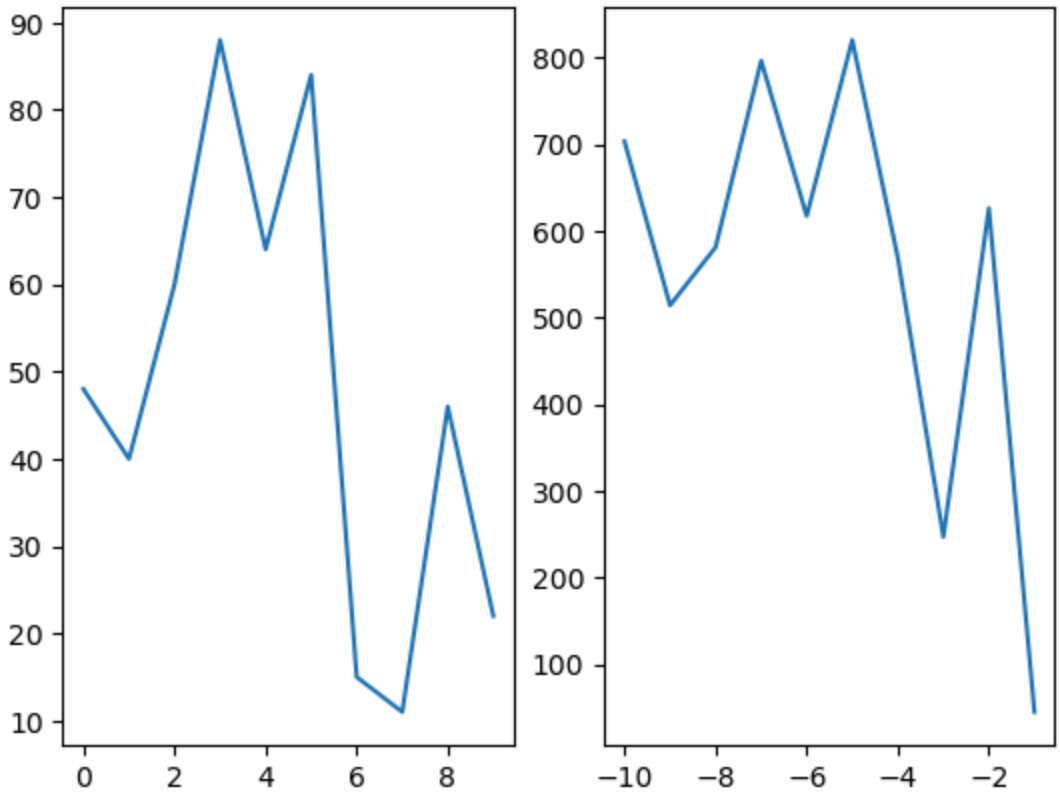
In [91]:
```
# Crear una figura con múltiples subplots
fig, axes = plt.subplots(3, 2)  # crea una ndarray de 3x2 axes (o subplots)
print(type(axes))
print(axes.shape)
```

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

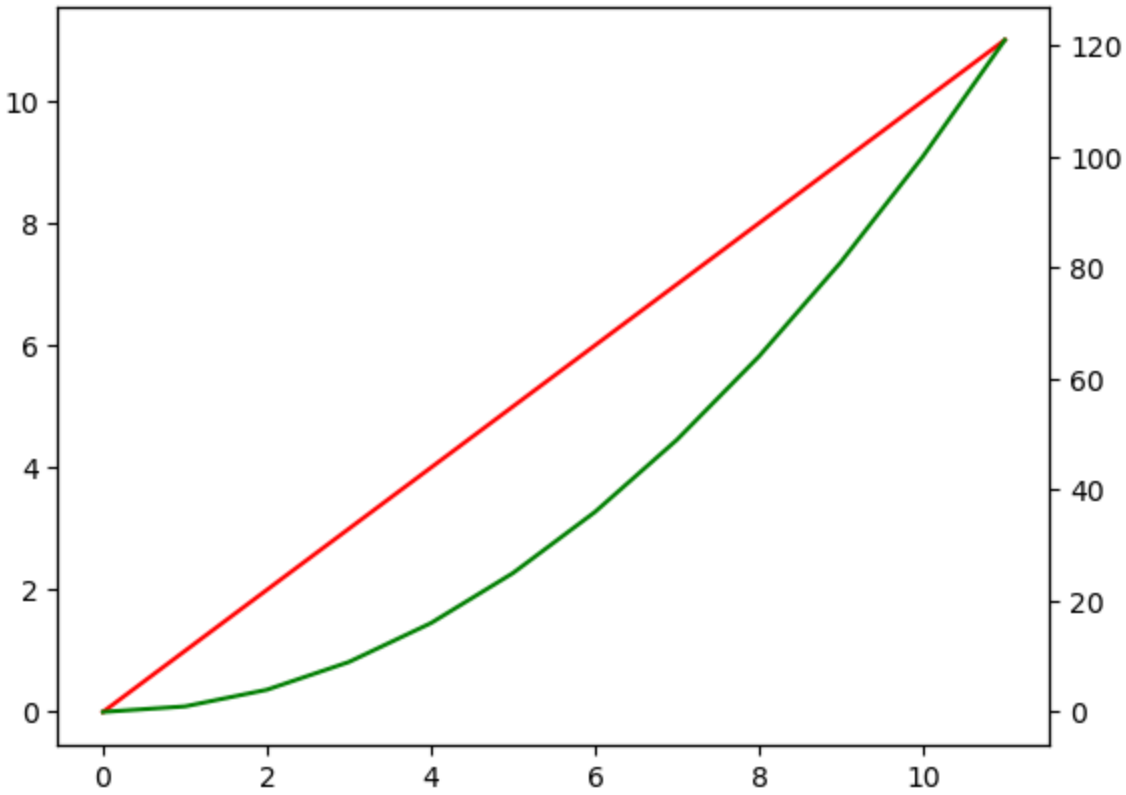


In [92]:
```
# Generar los datos
x_data1 = np.arange(0, 10)
x_data2 = np.arange(-10, 0)
y_data1 = np.random.randint(100, size=(10))
y_data2 = np.random.randint(1000, size=(10))
```

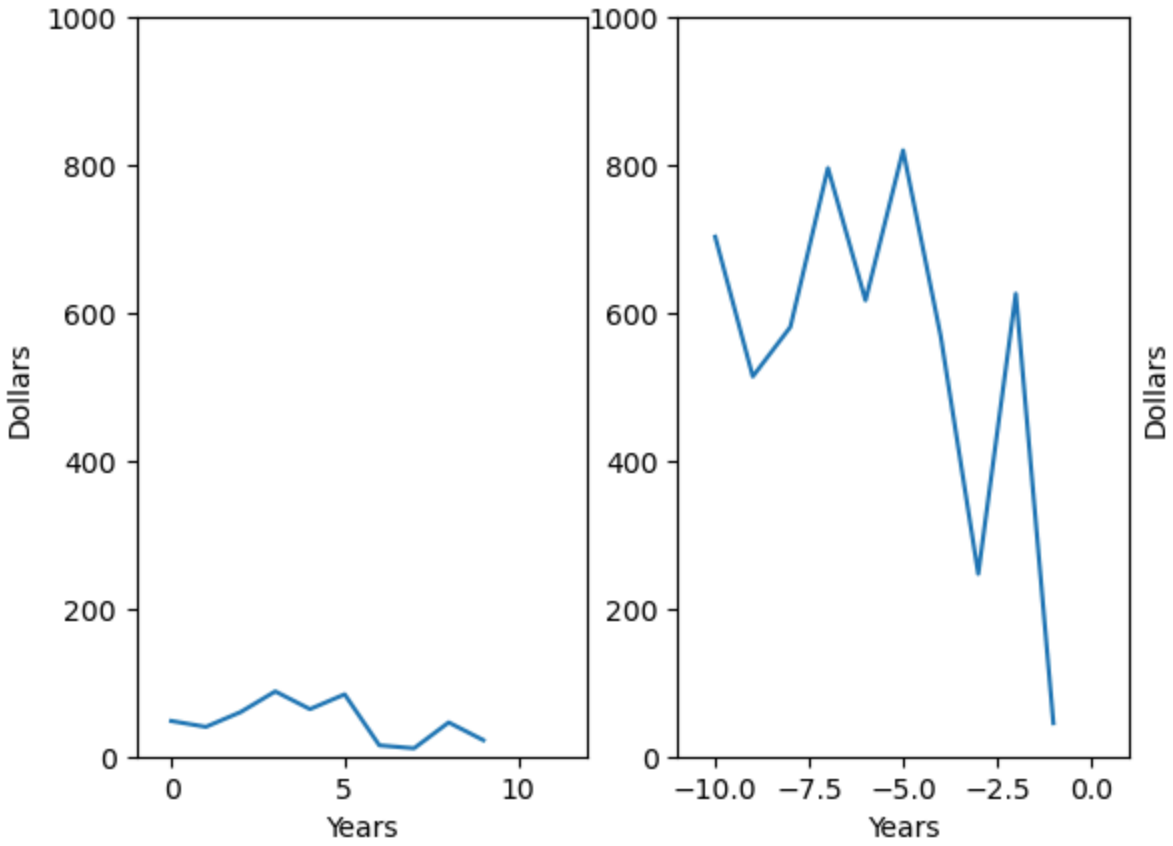
In [93]:
```
# Crear la figura y los subplots
fig, axes = plt.subplots(1,2)
# preparar cada subplot
axes[0].plot(x_data1,y_data1)
axes[1].plot(x_data2,y_data2)
#mostrar ambos axes
plt.show() # Recomendable usar plt y no fig.show()
```



In [94]:
```
# 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()
```



In [95]:
```
# 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()
```



1.1.1 Más métodos

2. plot()

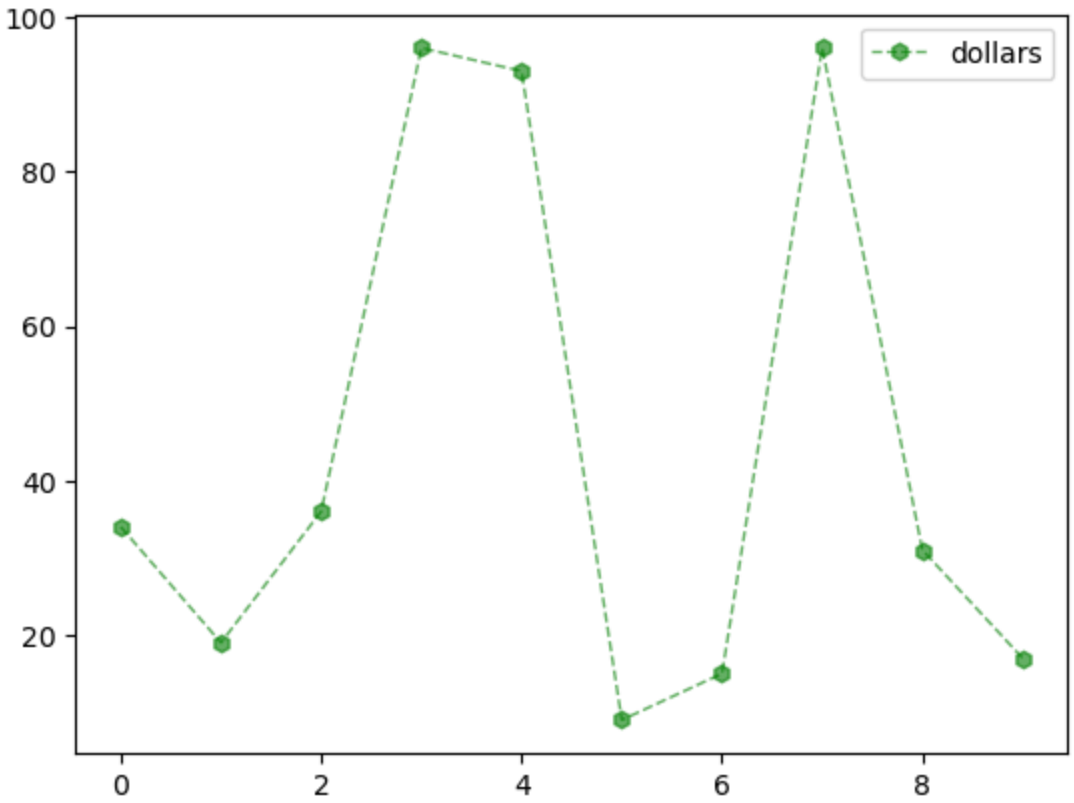
Múltiples parámetros

```
plt.plot(x, y, estilo, etiqueta, linewidth=1, alpha=1)
```

- **estilo**: cadena de texto con formato (color, tipo...) [color][marcador][línea]
- **etiqueta**: referencia para leyenda
- **linewidth**: grosor de línea
- **alpha**: transparencia (0-1)

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

In [96]:
```
# Ejemplo con diferentes estilos
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)  # creac
axis.legend(loc='best')  # representar la leyenda
plt.show()
```



Colores

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

Marcadores

Código	Descripción
.	marcador de punto
,	marcador de píxel
o	marcador de círculo

Código	Descripción
v	marcador de triángulo_abajo
^	marcador de triángulo_arriba
<	marcador de triángulo_izquierda
>	marcador triángulo_derecha
s	marcador cuadrado
p	marcador pentágono
*	marcador de estrella
h	marcador hexágono1
H	marcador hexágono2
+	marcador plus
x	marcador x
D	marcador de diamante
d	marcador de diamante fino
_	hline marker

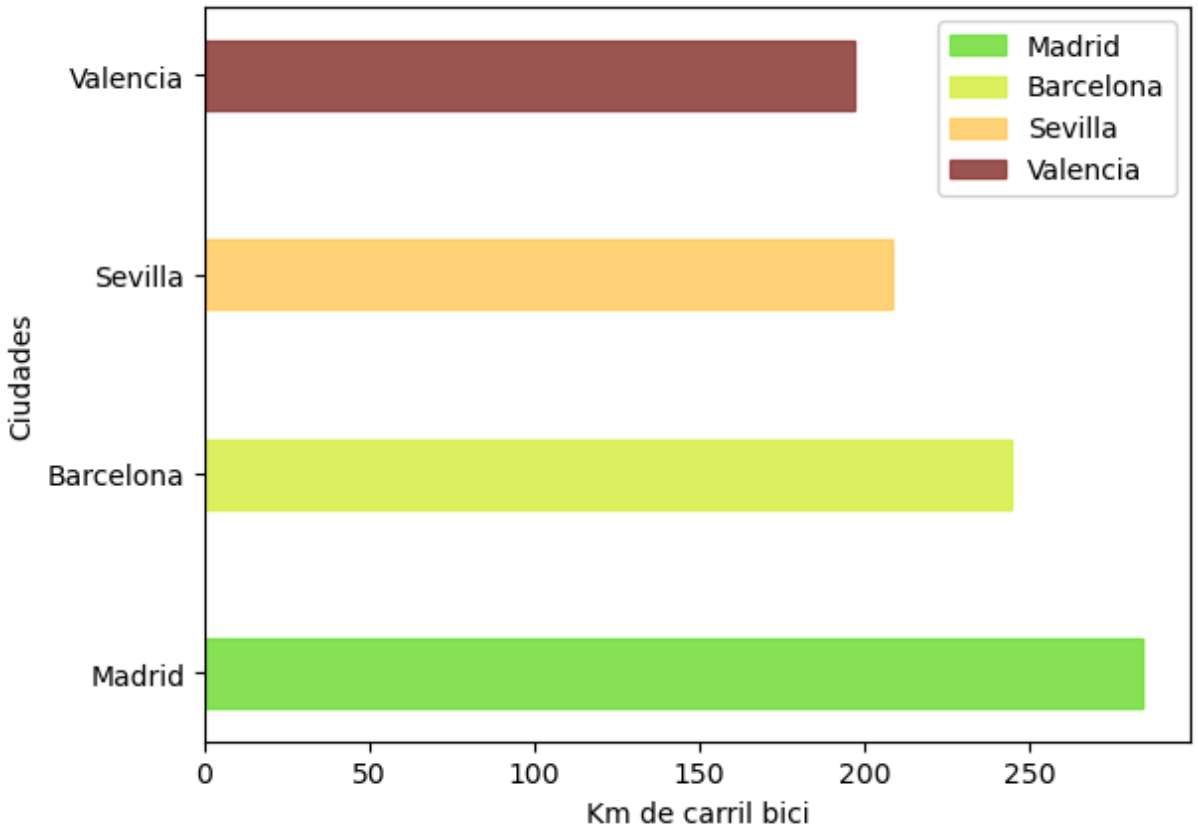
Estilos de Línea

Código	Descripción
-	estilo de línea sólida
--	estilo de línea discontinua
-.	estilo de línea de guiones y puntos
:	estilo de línea de puntos

In [132…

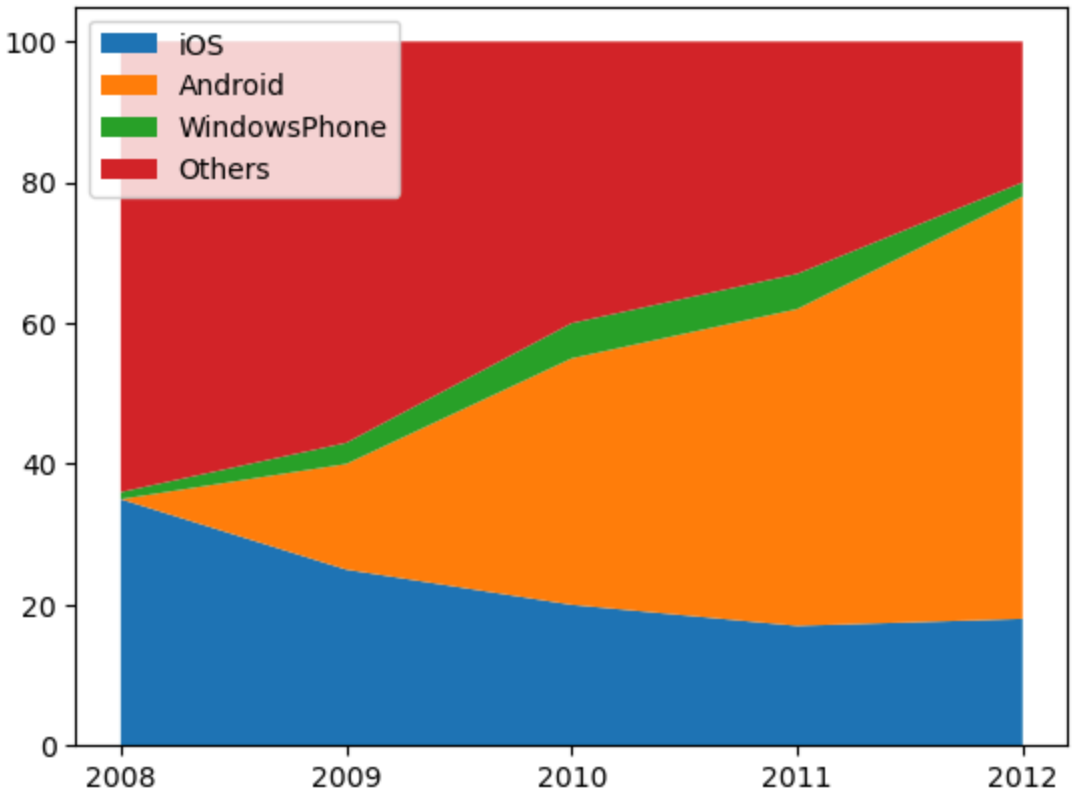
```
# Gráfico de barras con colores personalizados
ls_count = (284487, 244560, 208493, 196764)
cities = ('Madrid', 'Barcelona', 'Sevilla', 'Valencia')
datos = np.array(ls_count) / 1000
y_pos = np.arange(datos.size)

width = 0.35  # the width of the bars
opacity = 0.8

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

# Asignar colores
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)
ax.set_ylabel('Ciudades')
ax.set_xlabel('Km de carril bici')
ax.legend((rects1[0], rects1[1], rects1[2], rects1[3]), cities)
plt.show()
```



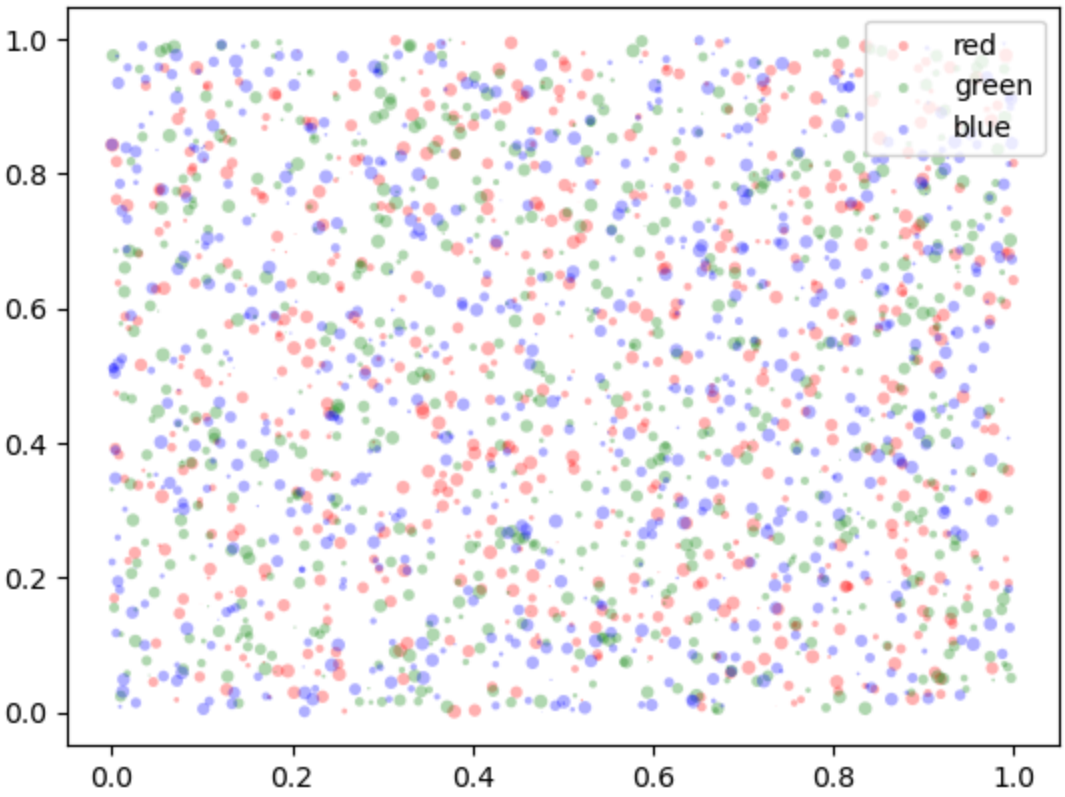
In [100…

```
# Diagrama de pila
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]

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

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



In [101…

```
# Scatter plot (Gráfico de dispersión)
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()
```



4.1 Grabar un archivo

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

In [102…

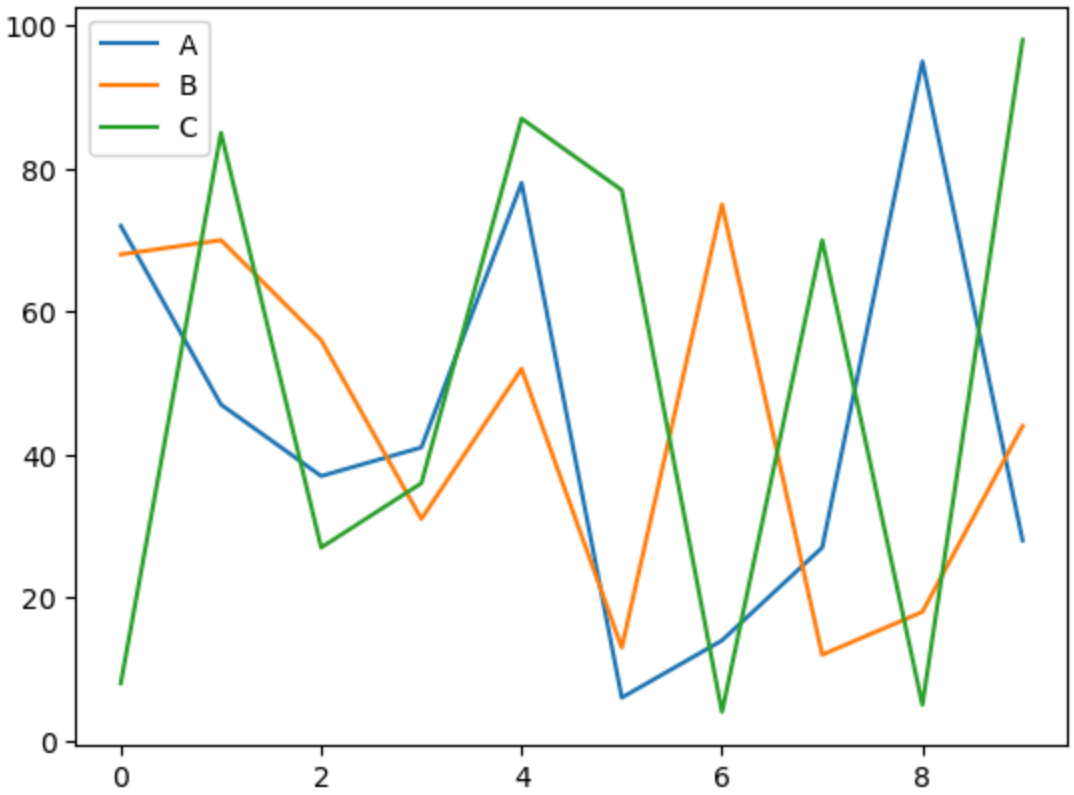
```
# Guardar la figura (puede ser múltiples gráficos)
import os

ruta = os.path.join("res", "o_figure.png")
fig.savefig(ruta, format='png')  # varios formatos aceptados ('pdf', 'png', 'svg',
```

5. Representación gráfica en pandas

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

In [103…

```
rand_matrix = np.random.randint(100, size=(10, 3))
frame = pd.DataFrame(rand_matrix, columns=list('ABC'))

frame.plot()
plt.show()

display(frame)
```



	A	B	C
0	72	68	8
1	47	70	85
2	37	56	27
3	41	31	36
4	78	52	87
5	6	13	77
6	14	75	4
7	27	12	70
8	95	18	5
9	28	44	98

6 Frame.plot()

Parámetros principales

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

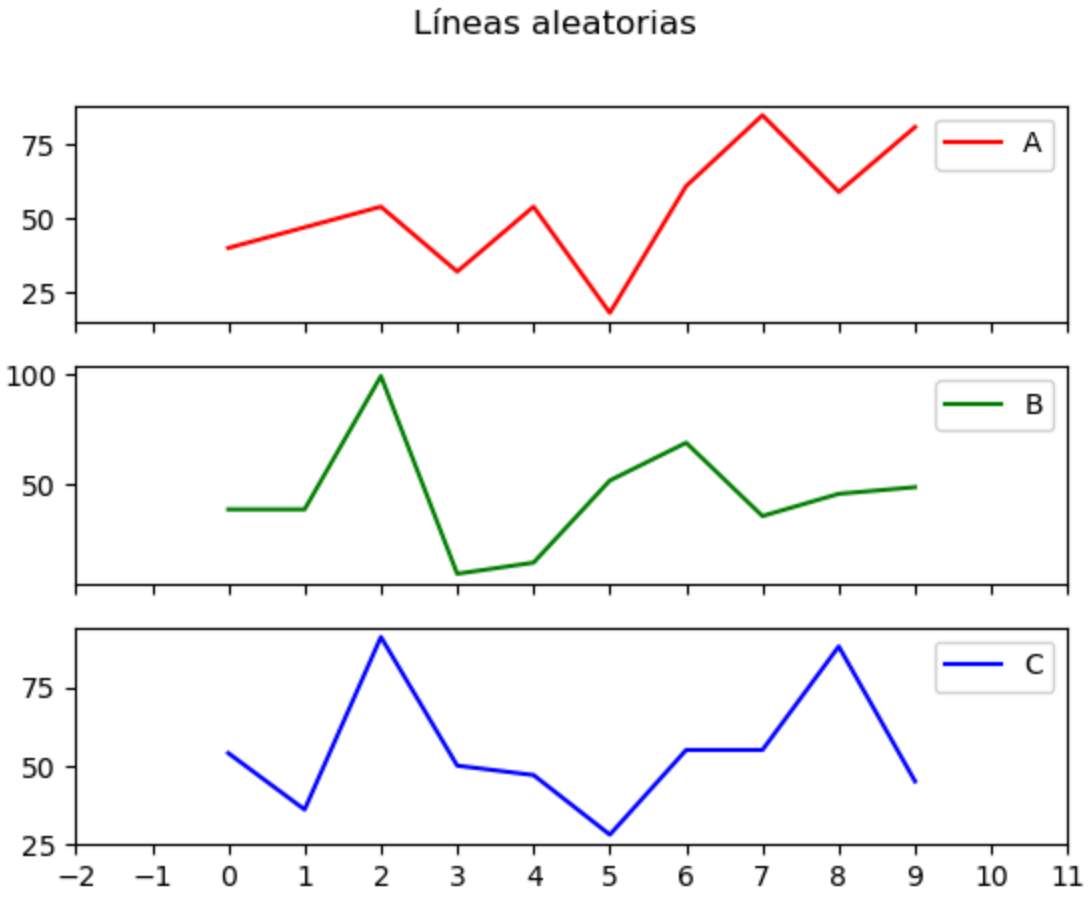
- **label** (solo para Series): referencia para la leyenda
- **ax**: subfigura en la que dibujar los datos

- **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**: cadena como título
- **subplots** (solo para DataFrame): bool si se desean subfiguras separadas para cada columna
- **x** e **y**: se pueden utilizar para seleccionar columnas para cada eje

Lista completa de argumentos

In [104…

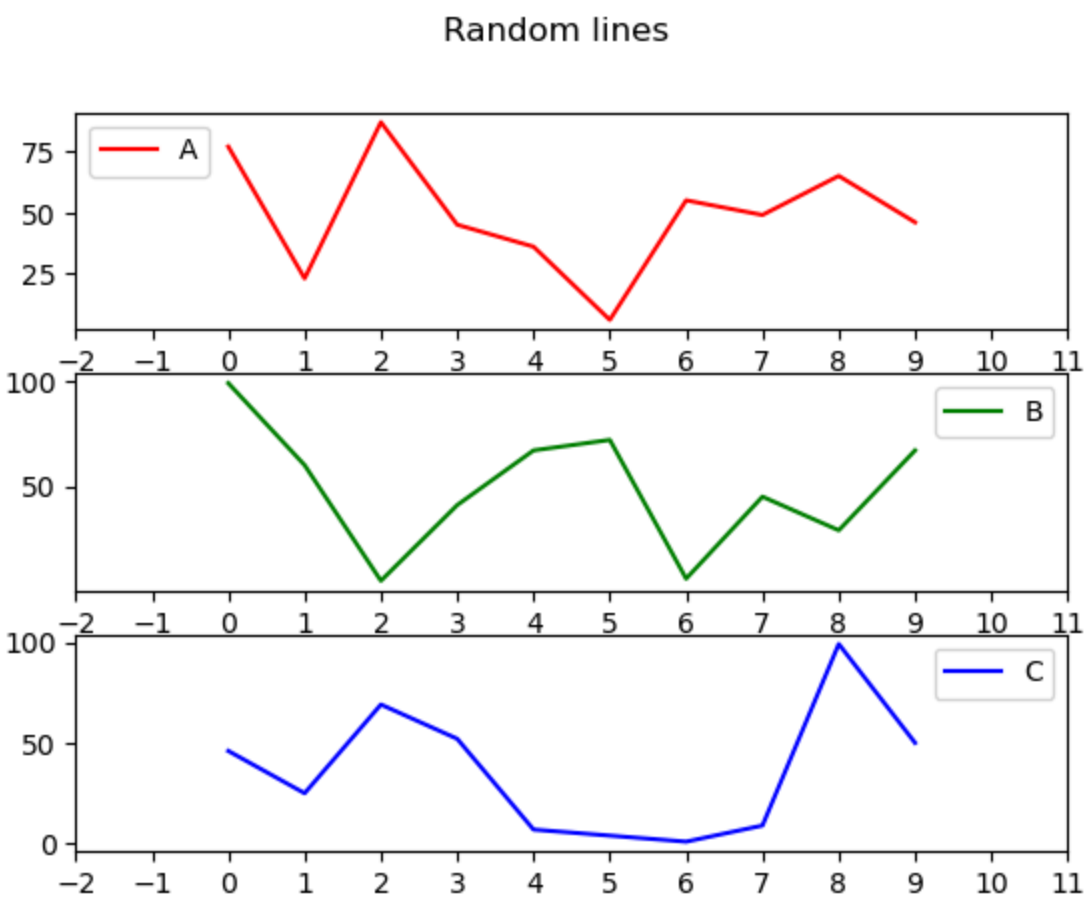
```
rand_matrix = np.random.randint(100, size=(10, 3))
frame = pd.DataFrame(rand_matrix, columns=list('ABC'))

frame.plot(style=['r-', 'g-', 'b-'], xticks=range(-2, 12),
           title='Líneas aleatorias', subplots=True)
plt.show()
display(frame)
```



	A	B	C
0	40	39	54
1	47	39	36
2	54	99	91
3	32	10	50
4	54	15	47
5	18	52	28
6	61	69	55
7	85	36	55
8	59	46	88
9	81	49	45

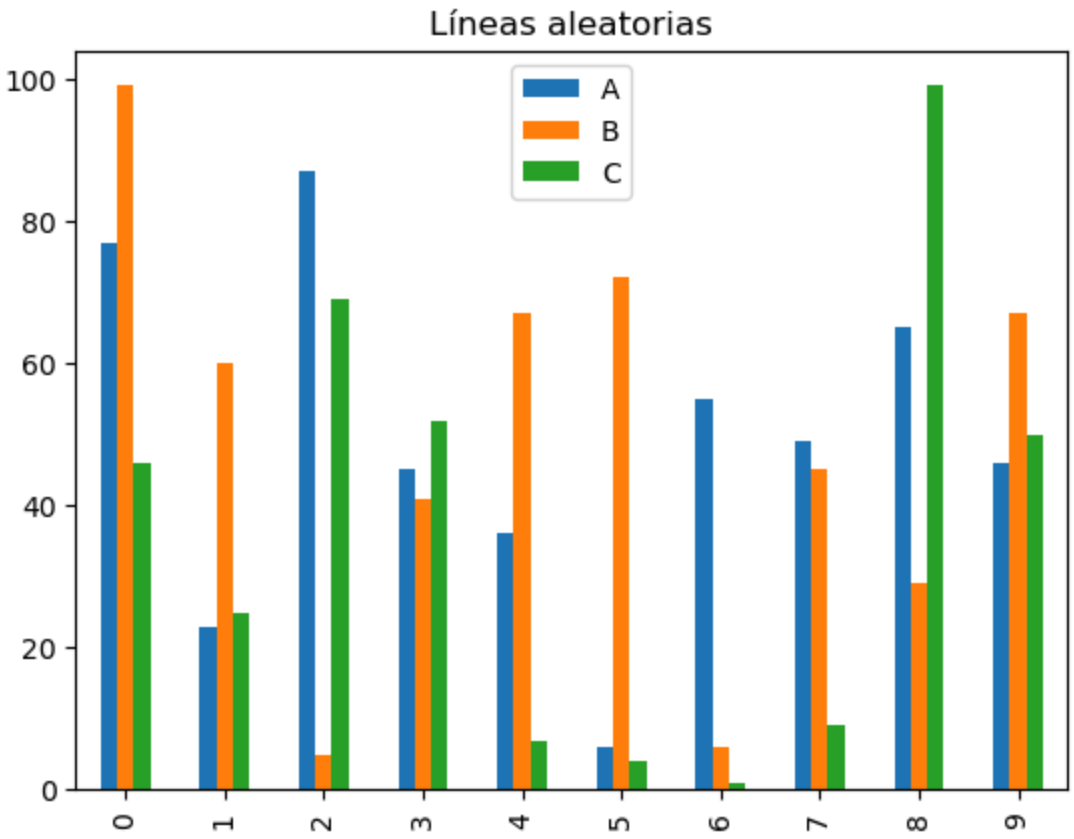
In [105…

```
# 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()
```



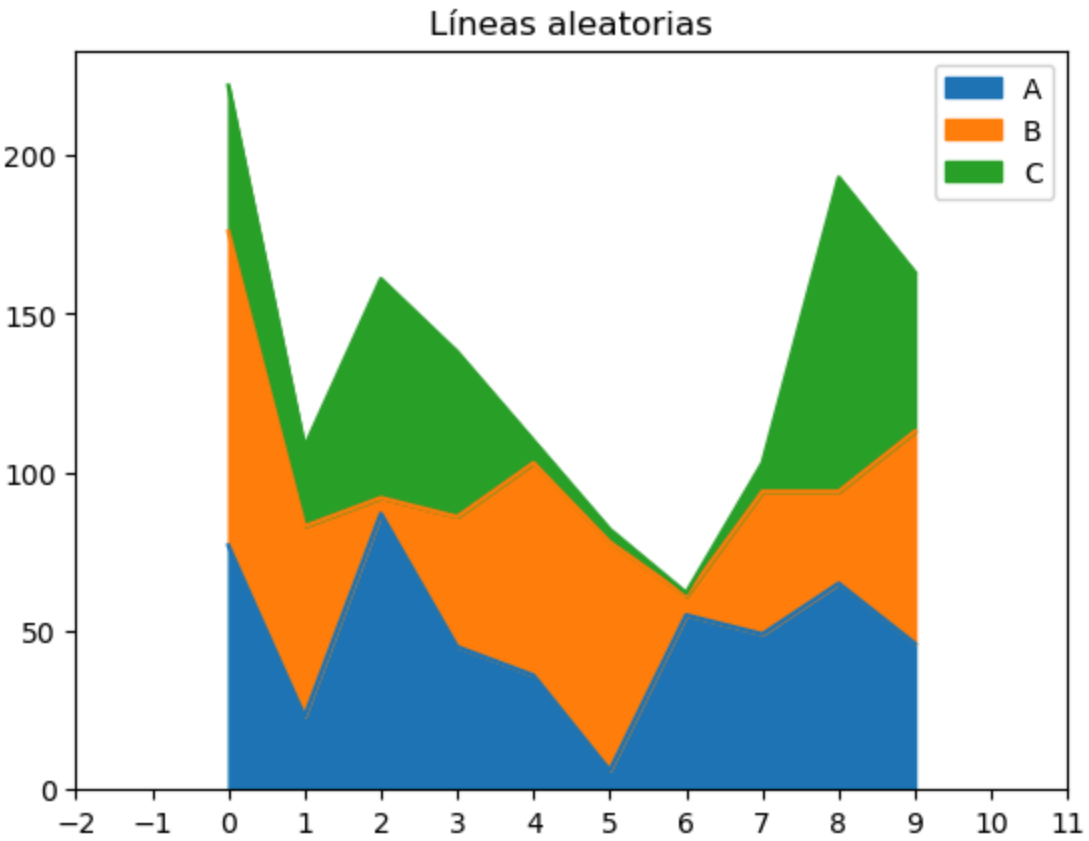
In [106…

```
# Gráfico de barras
frame.plot(kind='bar', title='Líneas aleatorias')
plt.show()
```



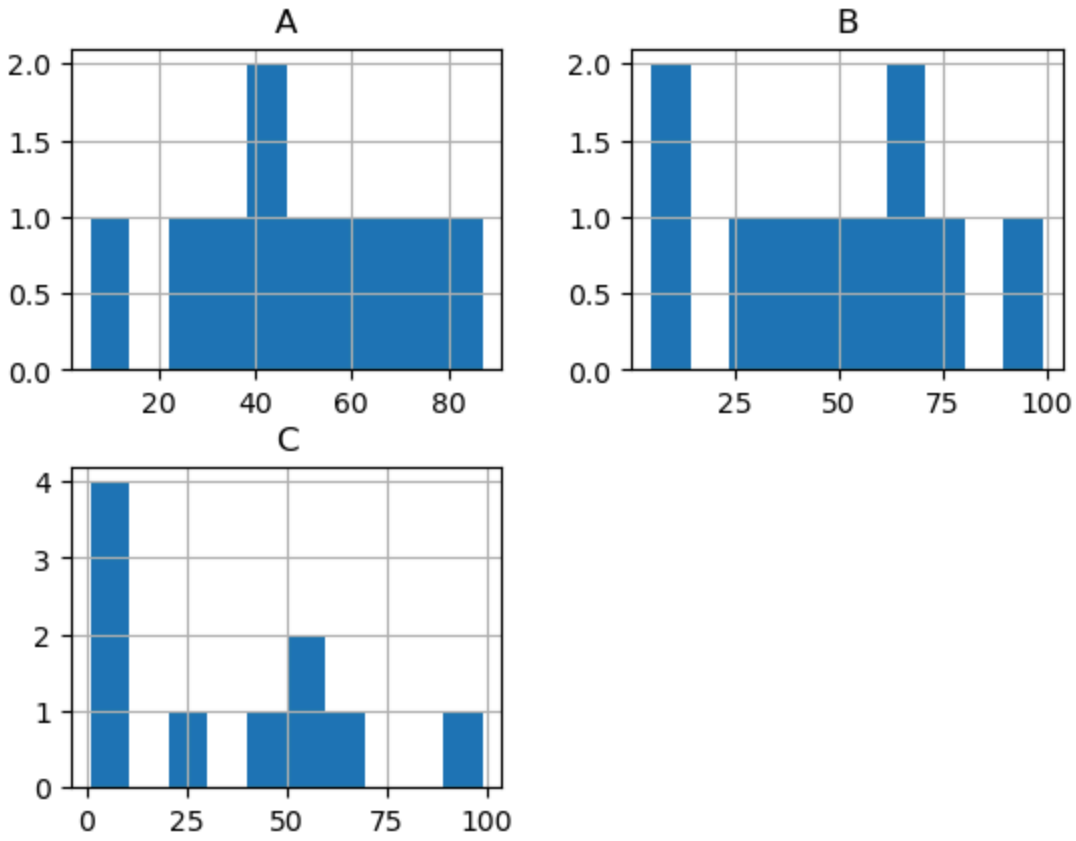
In [107…

```
# Gráfico de área
frame.plot(kind='area', xticks=range(-2, 12), title='Líneas aleatorias')
plt.show()
```



In [108…

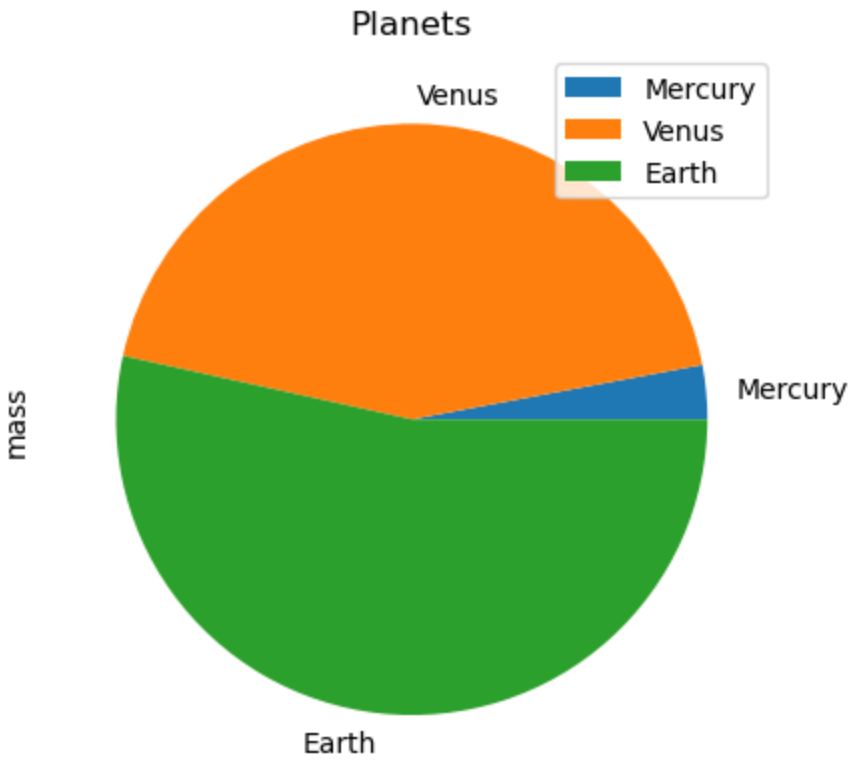
```
# Histogramas
frame.hist()
plt.show()
```



In [109…

```
# Gráfico de pastel (pie chart)
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')
plt.show()
```

	mass	radius
Mercury	0.33	2439.7
Venus	4.87	6051.8
Earth	5.97	6378.1



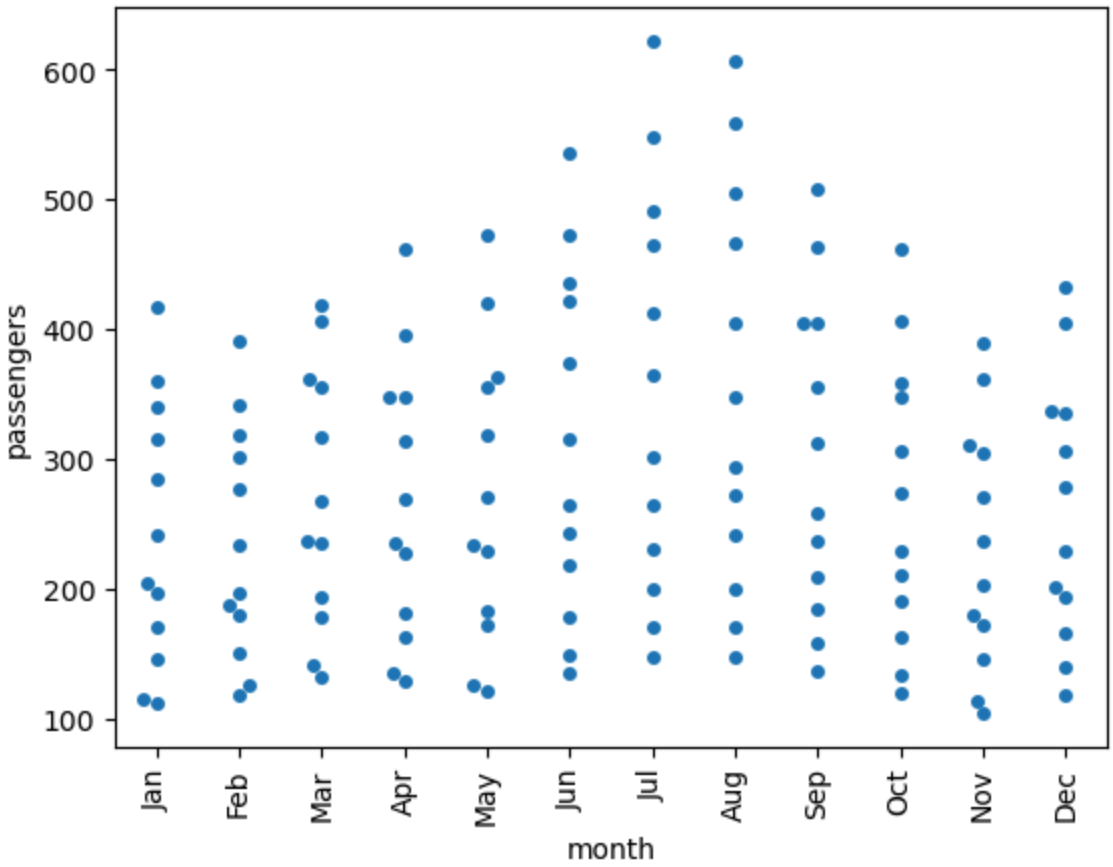
7. Seaborn

- 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

In [110…
```
# Importar seaborn
import seaborn as sns
```

In [111…
```
# Cargar datos de seaborn
flights_data = sns.load_dataset("flights")

# Construir gráfico
sns.swarmplot(x="month", y="passengers", data=flights_data)

# Mostrar gráfico
plt.xticks(rotation=90)
plt.show()
```



Compatible con pandas

In [112…

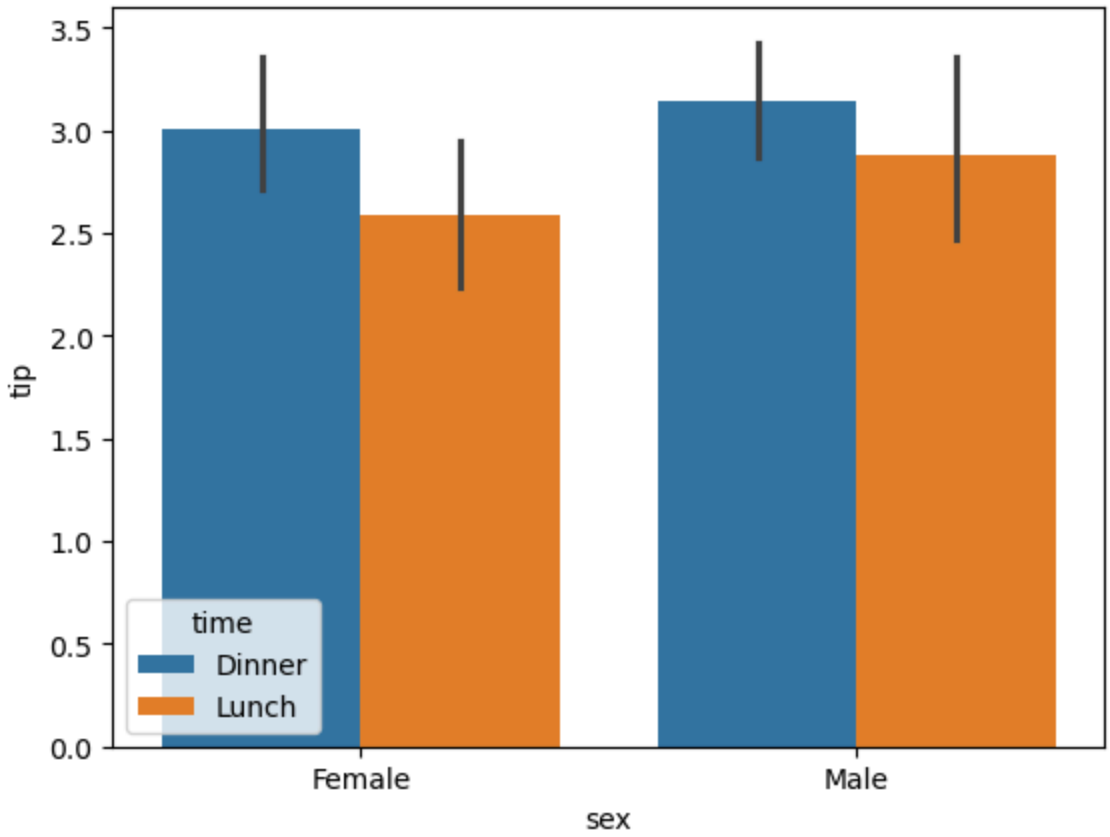
```
# Cargar datos desde URL
tips = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/t
tips.sample(5)
```

Out[112…

	total_bill	tip	sex	smoker	day	time	size
176	17.89	2.0	Male	Yes	Sun	Dinner	2
27	12.69	2.0	Male	No	Sat	Dinner	2
2	21.01	3.5	Male	No	Sun	Dinner	3
210	30.06	2.0	Male	Yes	Sat	Dinner	3
45	18.29	3.0	Male	No	Sun	Dinner	2

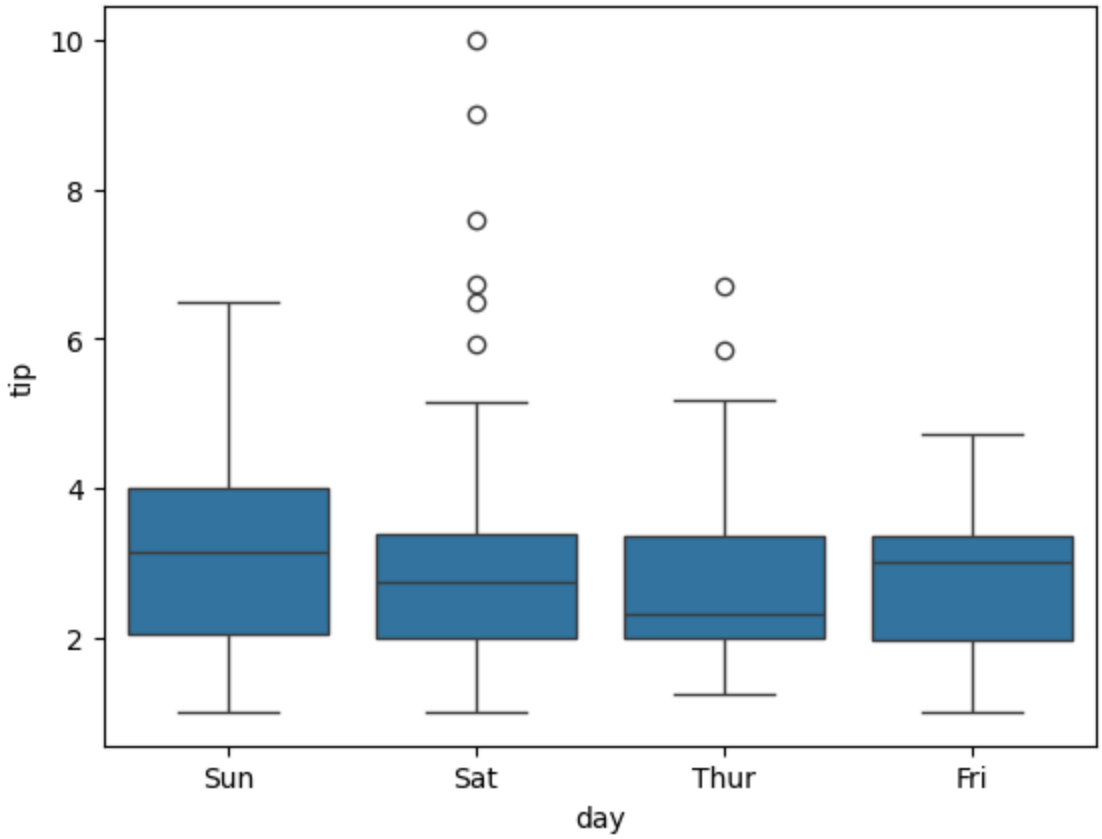
In [113…

```
# Barplot con seaborn
sns.barplot(x='sex', y='tip', hue='time', data=tips)
plt.show()
```



In [114…

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



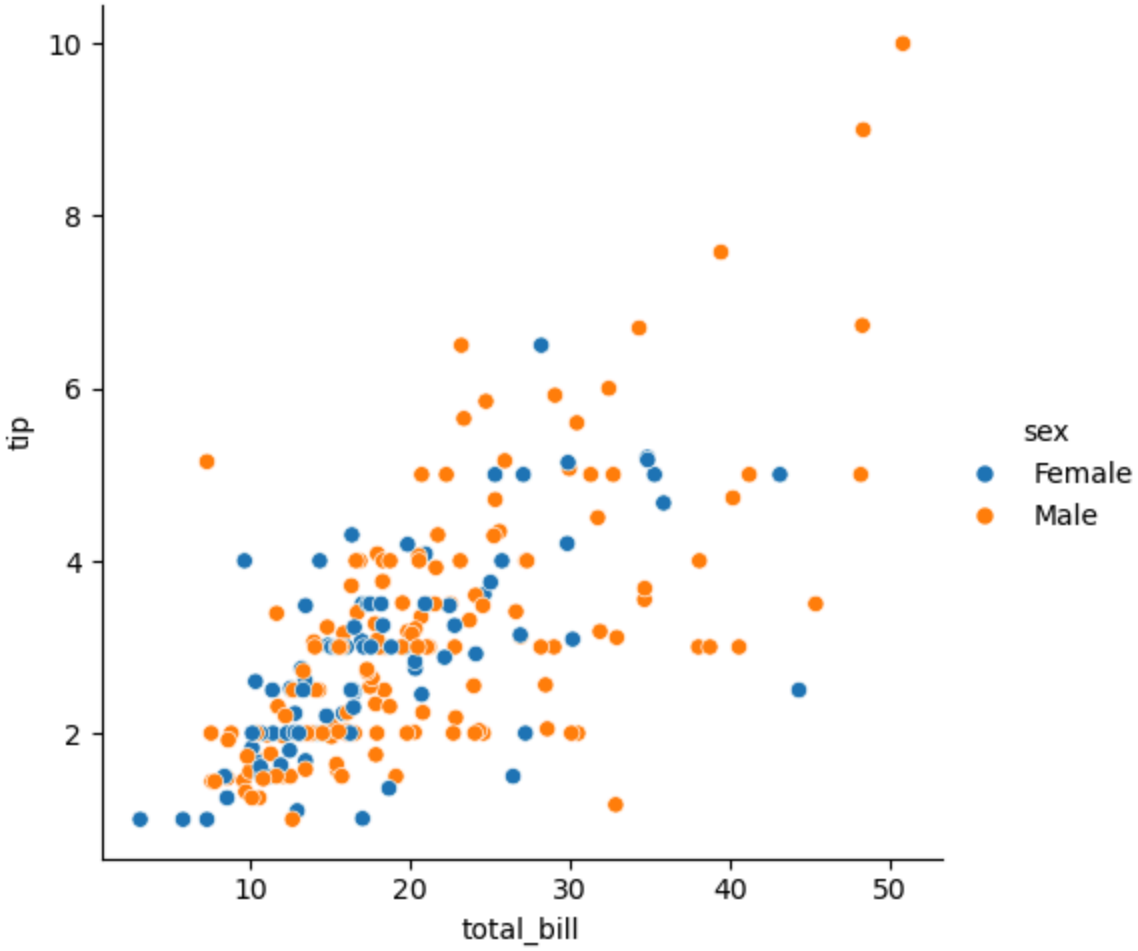
8.2.6 Comparación de distribuciones bivariables

Enunciado:

1. Comparar la distribución de dos variables (continua y numérica)
2. Representar cuantitativamente la densidad de elementos por valor en ambas variables
3. Utilizar `jointplot` con KDE para visualizar la densidad conjunta

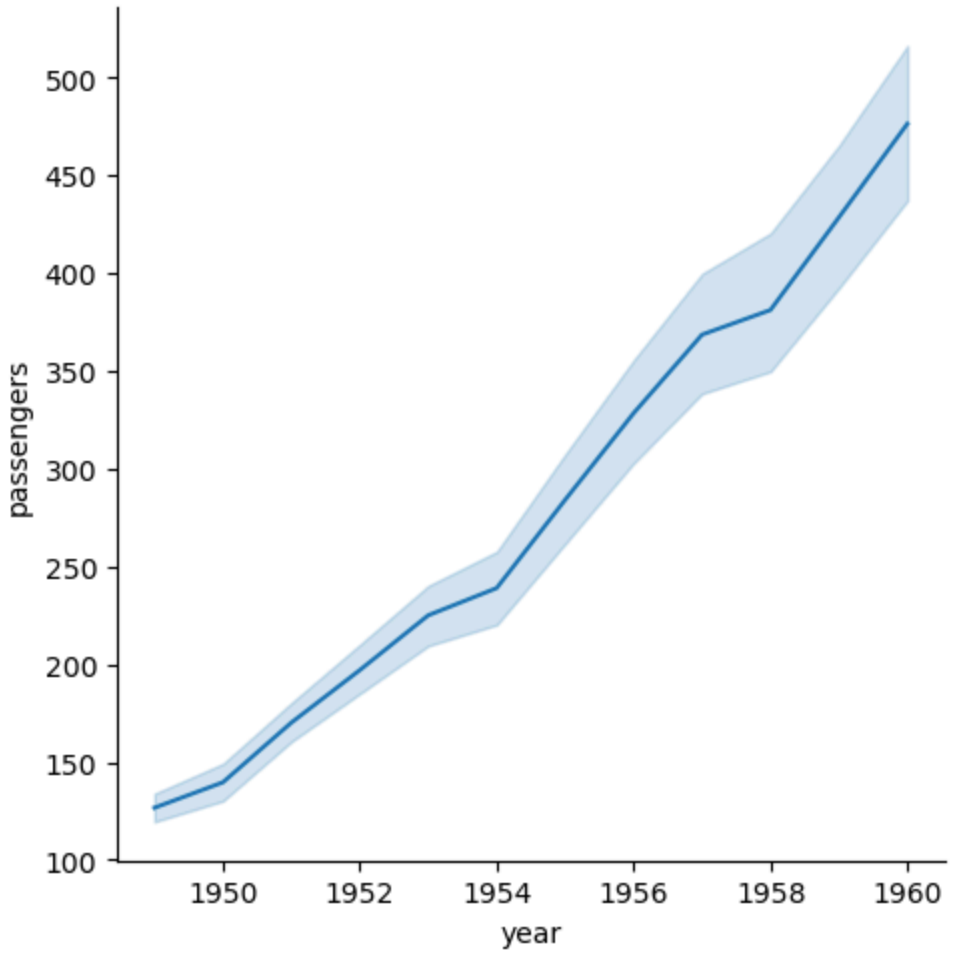
In [133…

```
# Relplot - Scatter plot básico
sns.relplot(x='total_bill', y='tip', hue='sex', data=tips, kind='scatter')
plt.show()
```



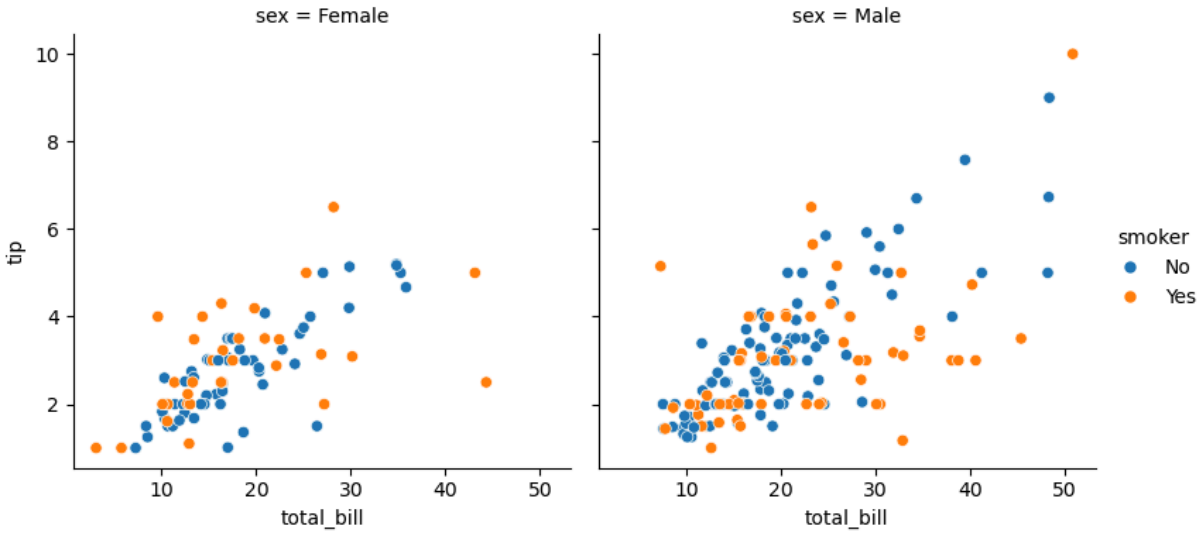
In [134…

```
# Relplot - Gráfico de línea
data = sns.load_dataset('flights')
sns.relplot(x="year", y="passengers", kind="line", data=data)
plt.show()
```



In [135…

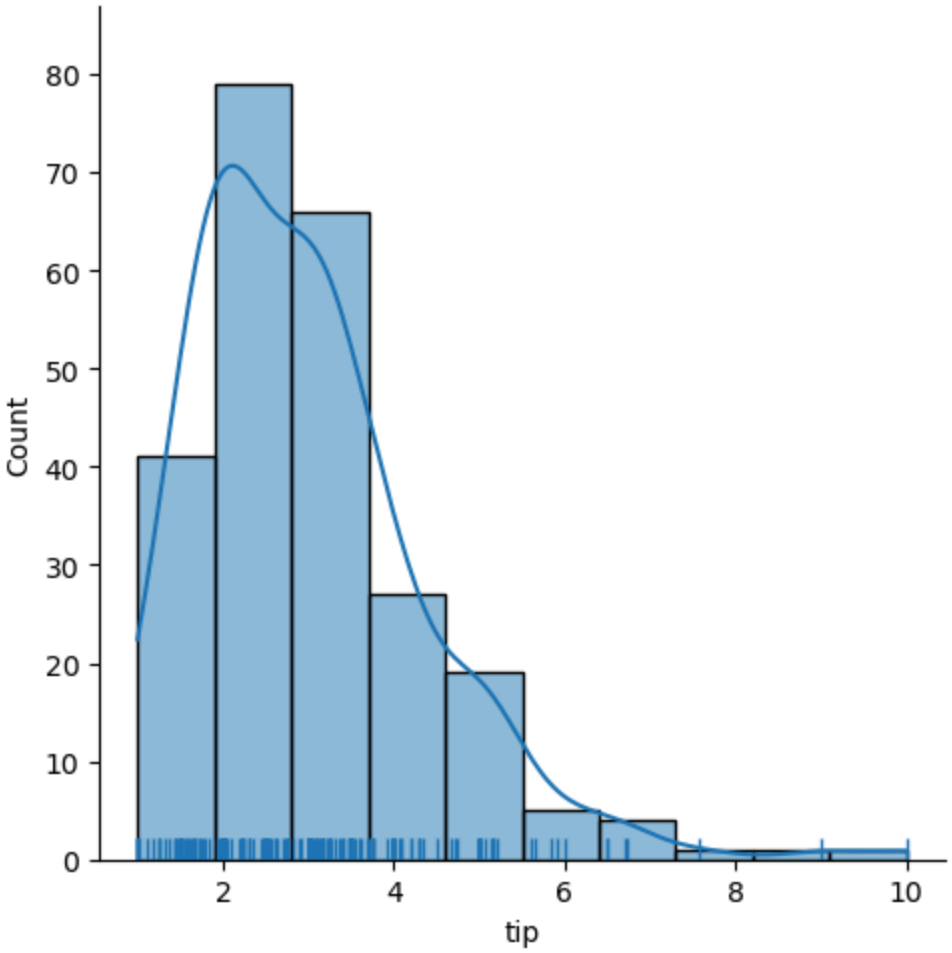
```
# Relplot - Relacionar hasta 4 variables automáticamente
sns.relplot(x="total_bill", y="tip", hue="smoker",
            col="sex", height=4,
            kind="scatter", data=tips)
plt.show()
```



7.2 Distribución de un conjunto de datos

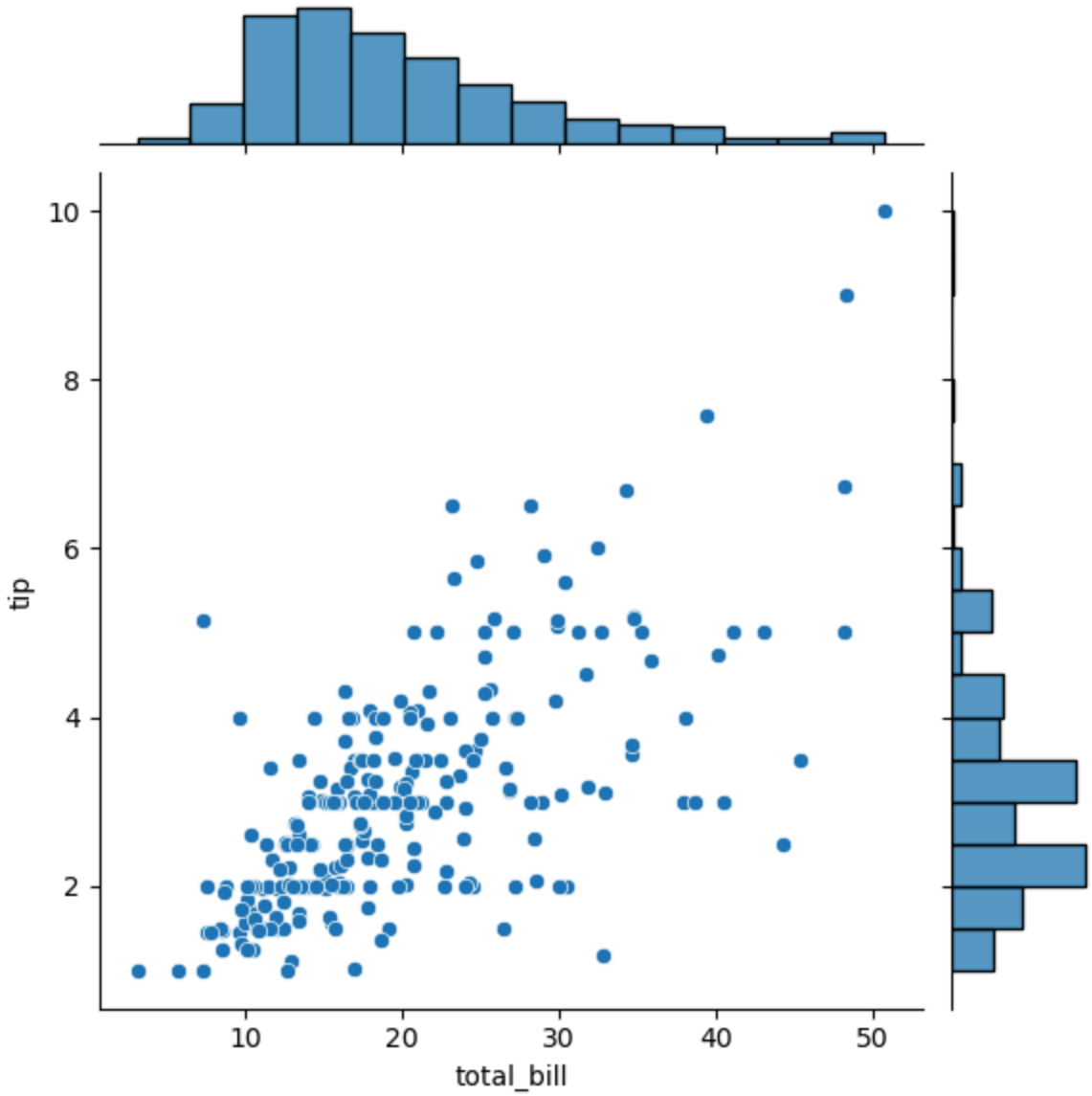
In [136…

```
# Histograma y diagrama de densidad
# kde = estimación de densidad del núcleo gaussiano (línea de densidad)
# bins = tamaño de los contenedores
# rug = densidad de los datos

sns.displot(tips['tip'], kde=True, bins=10, rug=True)
plt.show()
```



In [137…

```
# Distribución bivariable
sns.jointplot(x='total_bill', y='tip', data=tips)
plt.show()
```

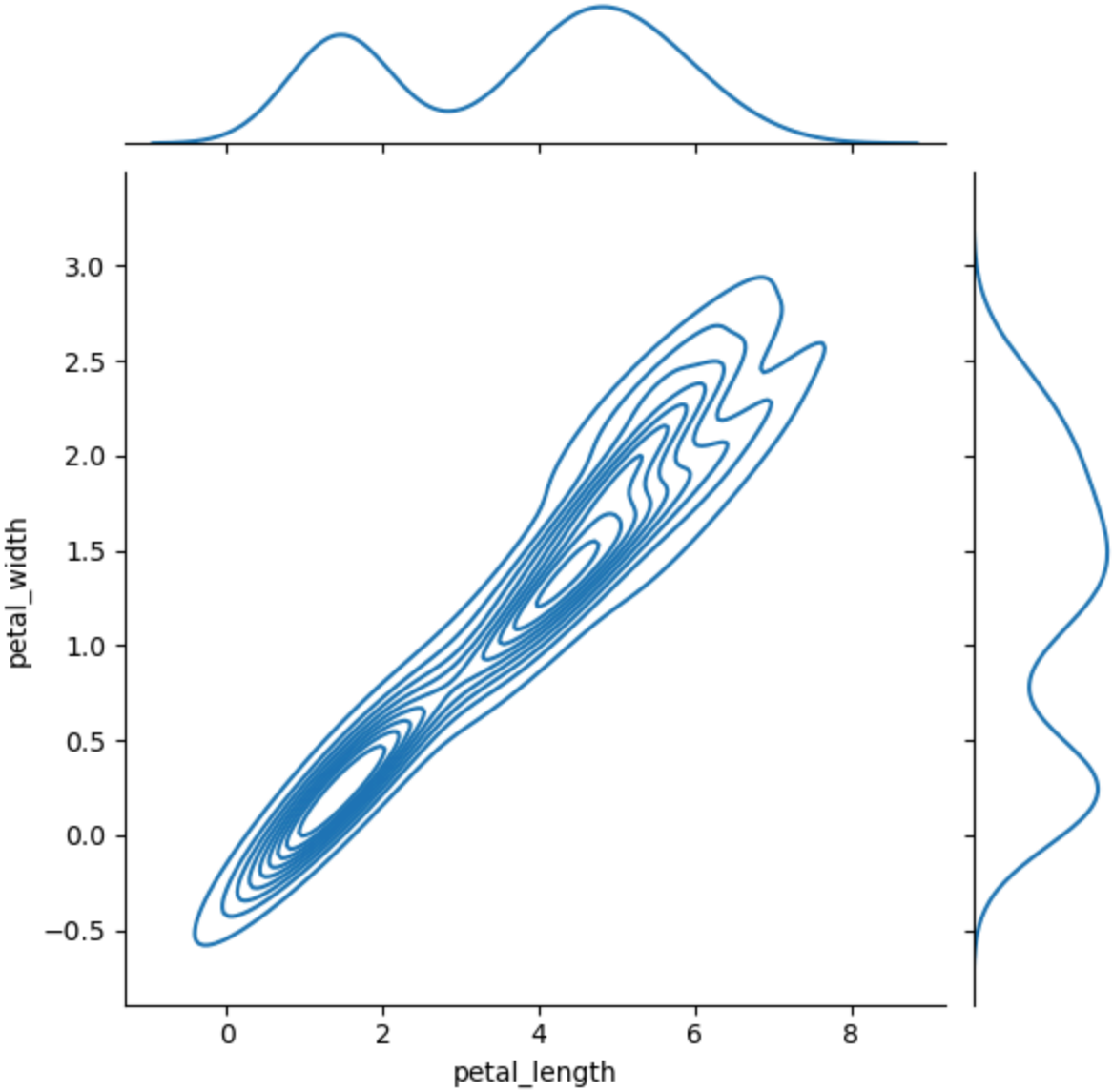


In [138…

```
# Gráfico de densidad bivariable con KDE
iris = sns.load_dataset('iris')
display(iris.head(5))
display(iris["species"].unique())

sns.jointplot(x='petal_length', y='petal_width', kind="kde", data=iris)
plt.show()
```

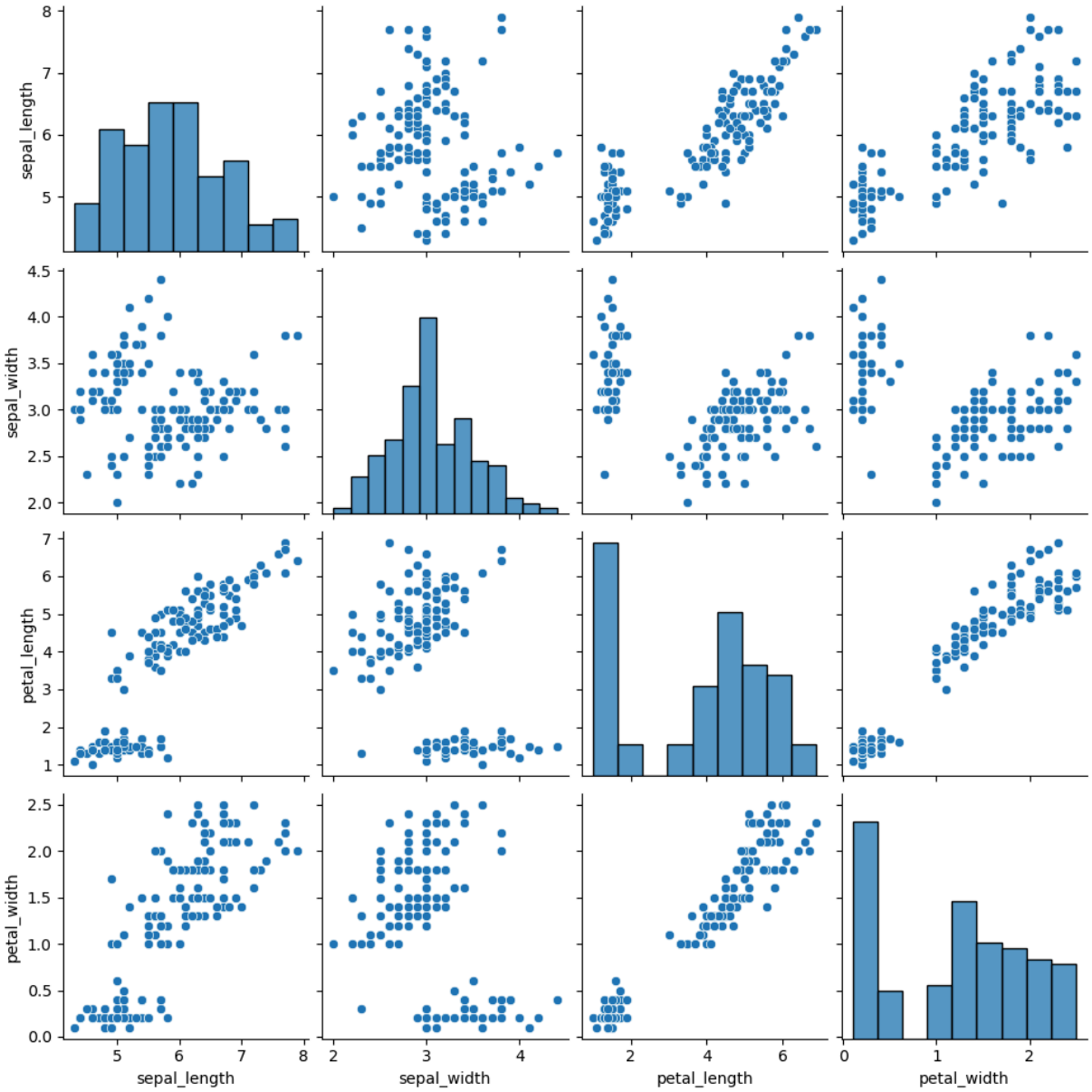
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```



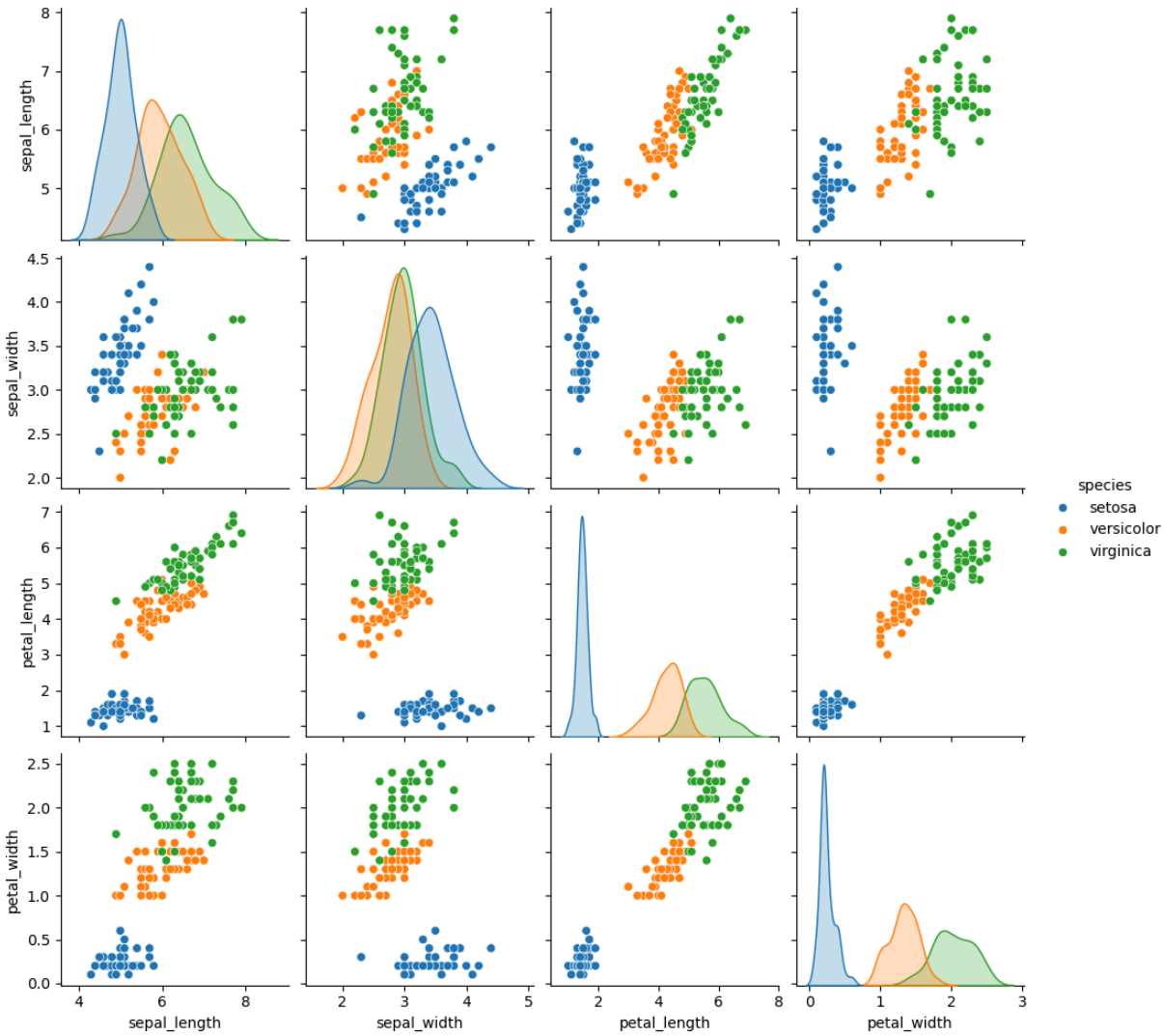
In [139…

```
# Relación de cada pareja (pairplot sin colores)
sns.pairplot(iris)
plt.show()
```



In [140…

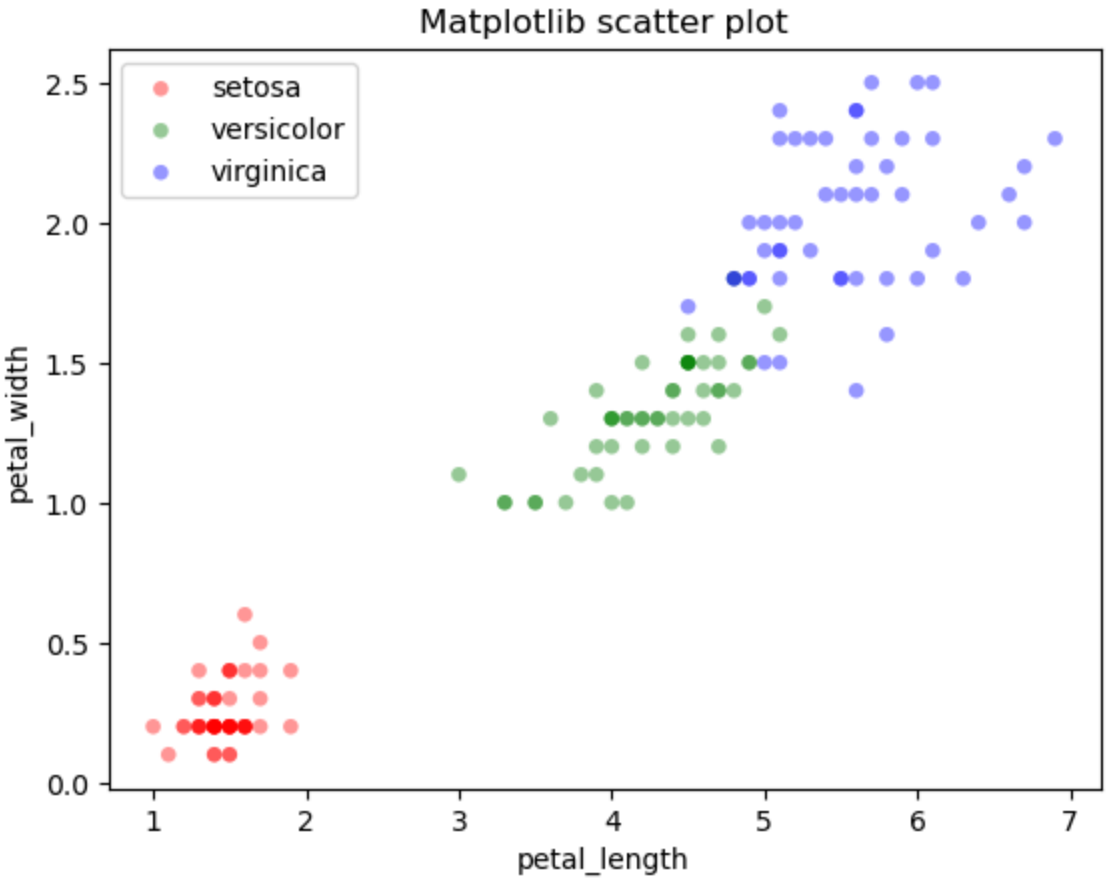
```
# Relación de cada pareja coloreando según clase (hue)
sns.pairplot(iris, hue="species")
plt.show()
```



In [141…

```
# Scatter plot de clases, coloreando según clase en matplotlib
iris = sns.load_dataset('iris')

# Preparación de los datos
data = []
colors = ["red", "green", "blue"]
groups = ["setosa", "versicolor", "virginica"]
columns = ["petal_length", "petal_width"]

for i in groups:
    # Búsqueda por clase definida
    data.append(iris[iris["species"] == i][columns].values)

# Creación de la gráfica
fig = plt.figure()
ax = fig.add_subplot()

for count, d in enumerate(data):
    # Separar las variables del gráfico
    x, y = d[:, 0], d[:, 1]
    ax.scatter(x, y, alpha=0.4, c=colors[count], edgecolors='none', s=30, label=gro

# Etiquetas
```

```
ax.set_xlabel(columns[0])
ax.set_ylabel(columns[1])
plt.legend(loc='best')
plt.title('Matplotlib scatter plot')
plt.show()
```



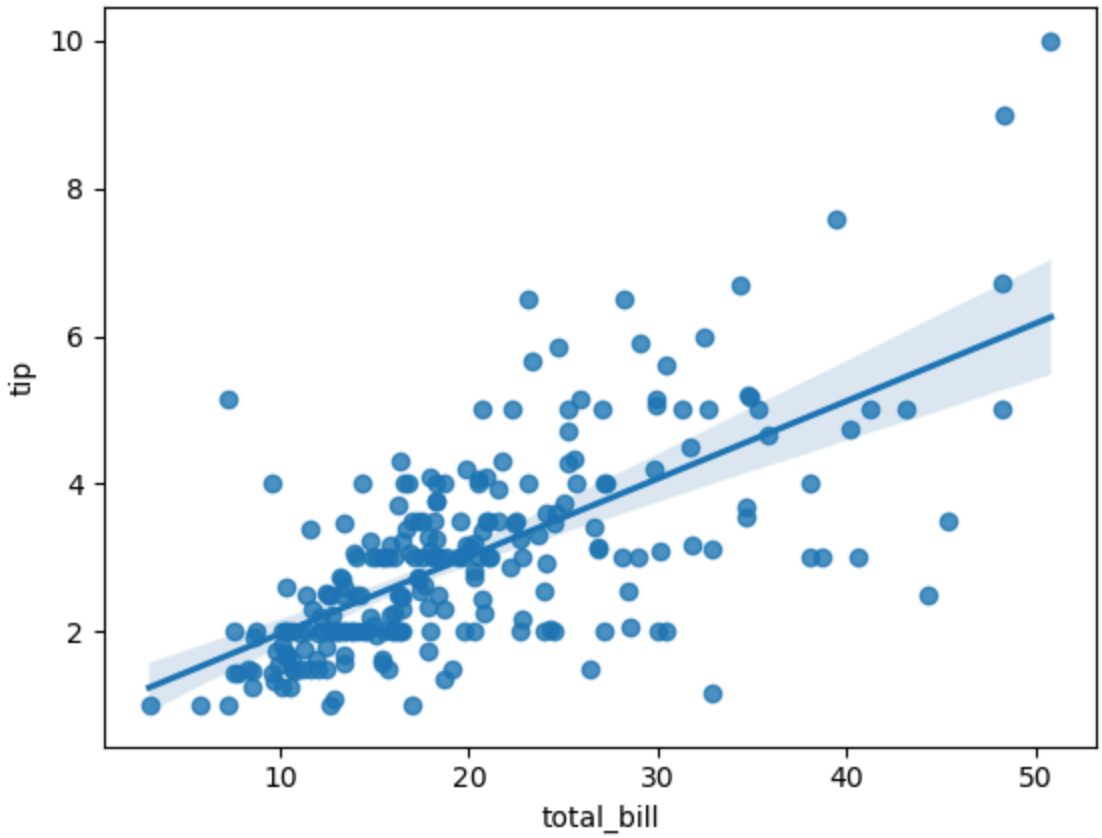
7.3 Visualización de correlaciones

• Regresión lineal con variables

• Correlaciones numéricas

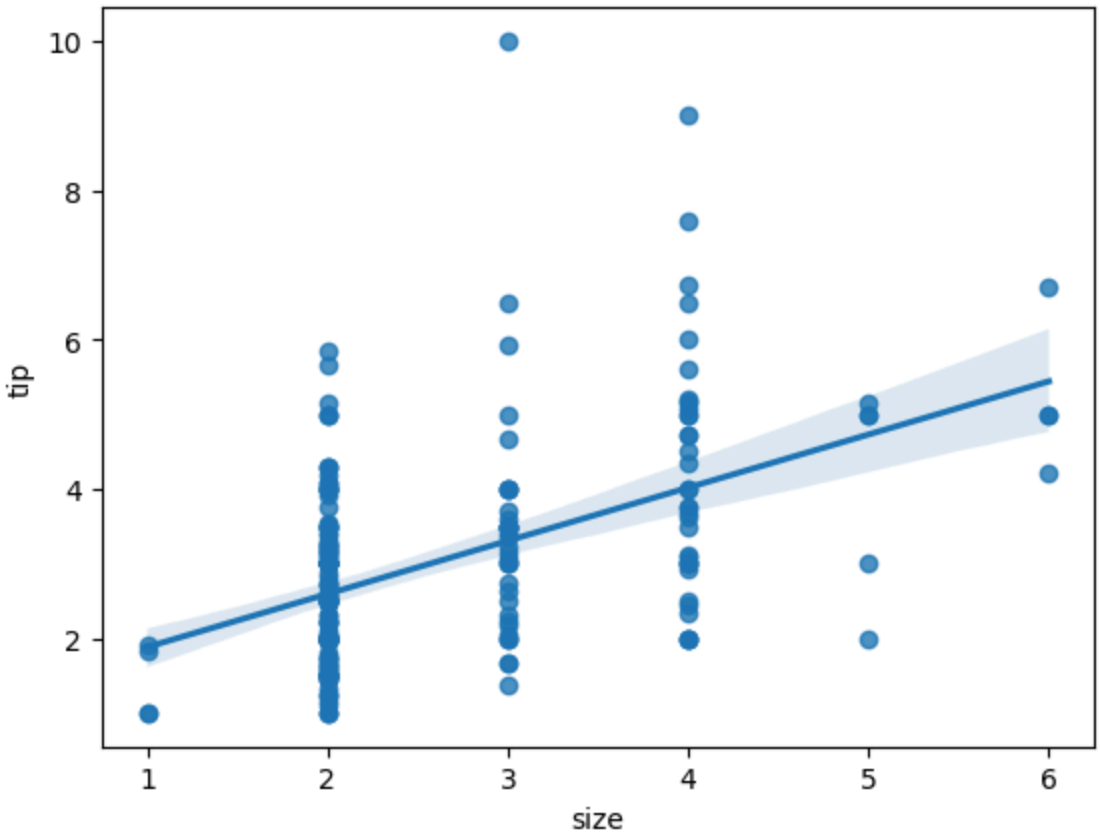
In [142…

```
# Correlación entre total_bill y tip: recta de regresión e intervalo de confianza
sns.regplot(x="total_bill", y="tip", data=tips)
plt.show()
```



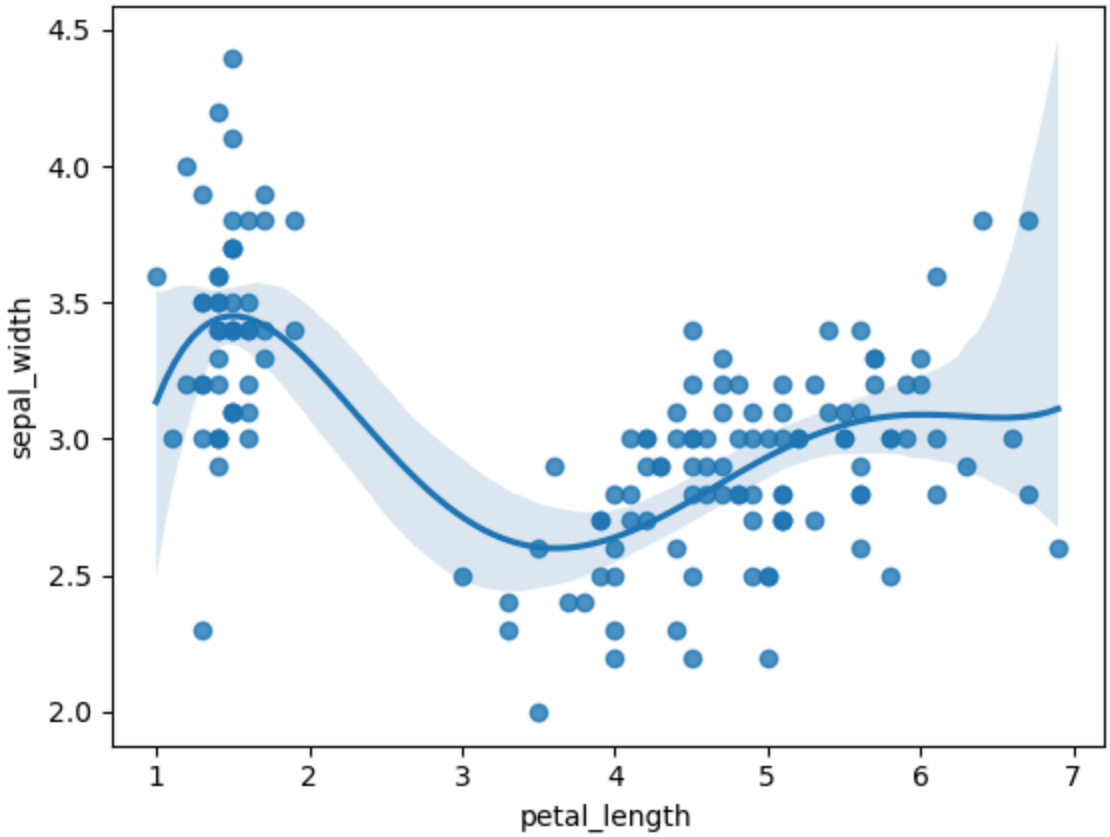
In [143…

```
# Correlación con valores categóricos (variable discreta)
# Nota: una variable debe ser numérica/continua
sns.regplot(x="size", y="tip", data=tips)
plt.show()
```



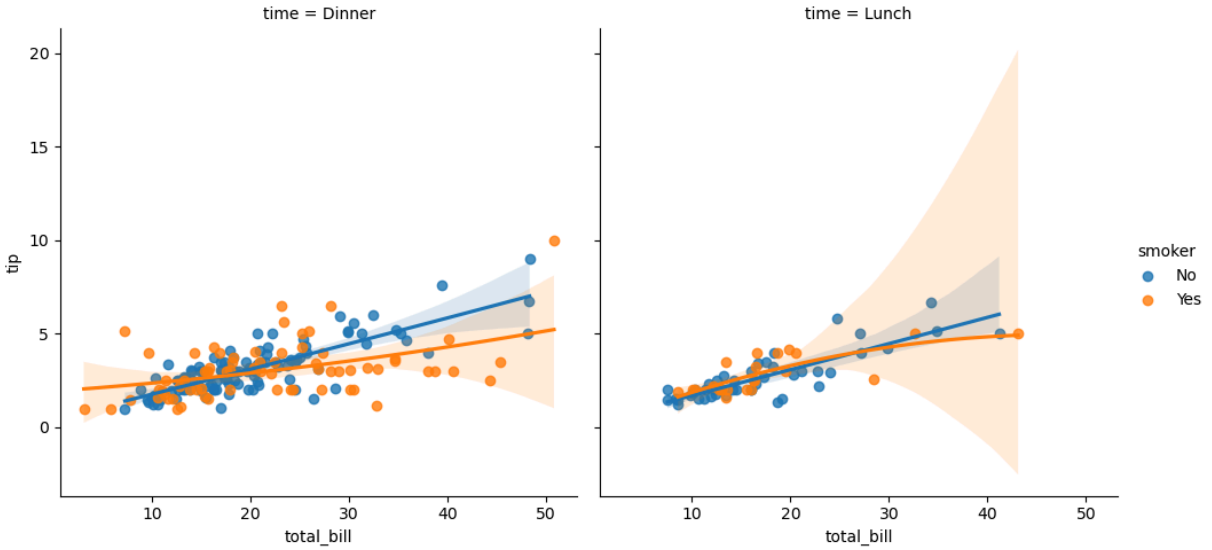
In [144…

```
# Regresión con polinomios de grados superiores
sns.regplot(x="petal_length", y="sepal_width", data=iris, order=5)
plt.show()
```



In [145…

```
# Correlaciones con varias variables
sns.lmplot(x="total_bill", y="tip", hue="smoker", data=tips, col='time', order=2)
plt.show()
```



8. Ejercicios

• Representación gráfica con matplotlib y seaborn

8.1 matplotlib

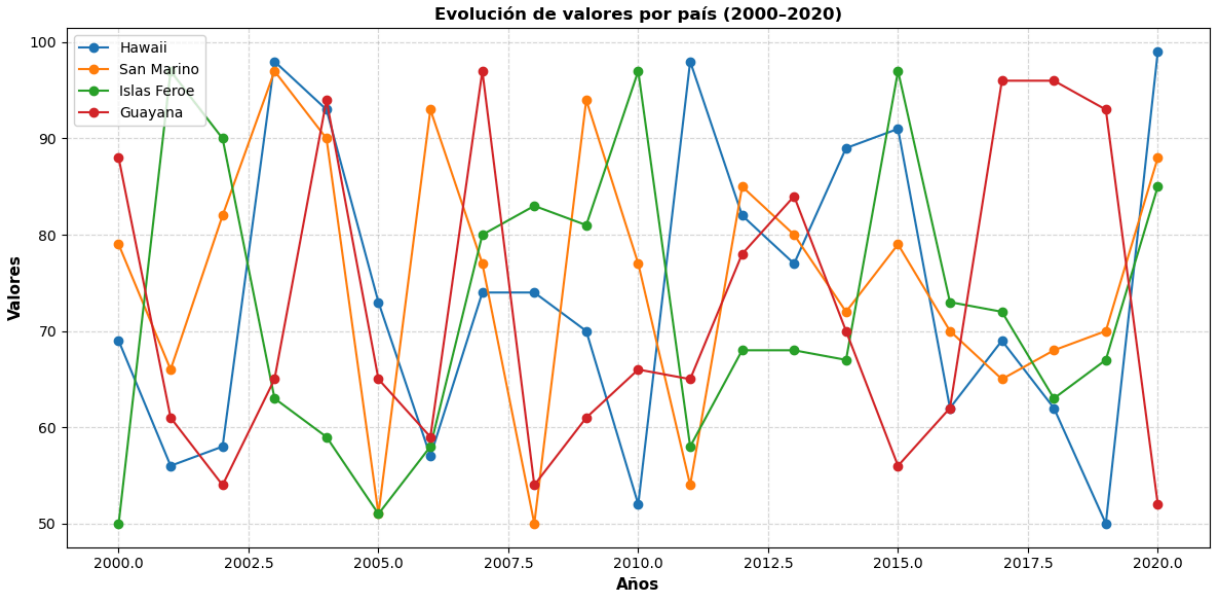
1. Crear una lista de años de 2000 a 2020 (eje x)
    - Generar datos aleatorios para el eje y (hacerlo 4 veces distintas: y0, y1, y2 e y3)

In [146…

```
# Generar datos: años de 2000 a 2020
years = np.arange(2000, 2021)
y0 = np.random.randint(50, 100, size=len(years))
y1 = np.random.randint(50, 100, size=len(years))
y2 = np.random.randint(50, 100, size=len(years))
y3 = np.random.randint(50, 100, size=len(years))

print(f"Años: {years}")
print(f"y0 (Hawaii): {y0}")
print(f"y1 (San Marino): {y1}")
print(f"y2 (Islas Feroe): {y2}")
print(f"y3 (Guayana): {y3}")

plt.figure(figsize=(12, 6))

plt.plot(years, y0, marker='o', label='Hawaii')
plt.plot(years, y1, marker='o', label='San Marino')
plt.plot(years, y2, marker='o', label='Islas Feroe')
plt.plot(years, y3, marker='o', label='Guayana')
```

```
plt.xlabel('Años', fontsize=11, fontweight='bold')
plt.ylabel('Valores', fontsize=11, fontweight='bold')
plt.title('Evolución de valores por país (2000-2020)', fontsize=12, fontweight='bol
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```

```
Años: [2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
 2014 2015 2016 2017 2018 2019 2020]
y0 (Hawaii): [69 56 58 98 93 73 57 74 74 70 52 98 82 77 89 91 62 69 62 50 99]
y1 (San Marino): [79 66 82 97 90 51 93 77 50 94 77 54 85 80 72 79 70 65 68 70 88]
y2 (Islas Feroe): [50 97 90 63 59 51 58 80 83 81 97 58 68 68 67 97 73 72 63 67 85]
y3 (Guayana): [88 61 54 65 94 65 59 97 54 61 66 65 78 84 70 56 62 96 96 93 52]
```



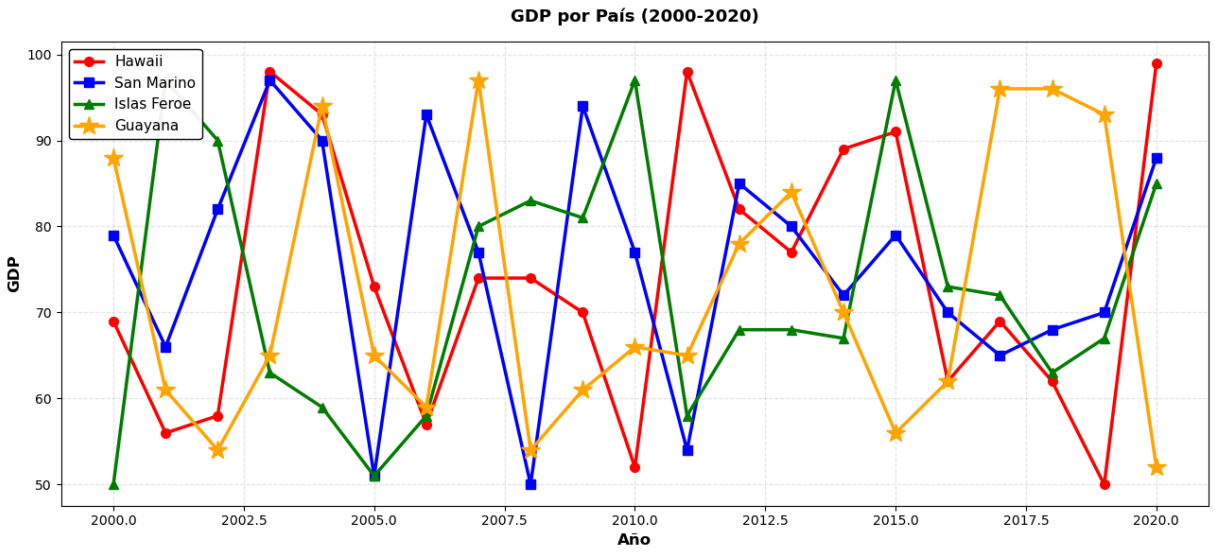
2. Representar las 4 secuencias de datos aleatorios en una sola figura utilizando matplotlib

- Añade una leyenda para poder identificar cada secuencia en el gráfico
- Nombra las secuencias de la siguiente forma: “Hawaii”, “San Marino”, “Islas Feroe”,“Guayana”

In [147…

```
# Representar las 4 secuencias en una sola figura
fig, ax = plt.subplots(figsize=(13, 6))

ax.plot(years, y0, label='Hawaii', marker='o', linewidth=2.5, markersize=7, color='
ax.plot(years, y1, label='San Marino', marker='s', linewidth=2.5, markersize=7, col
ax.plot(years, y2, label='Islas Feroe', marker='^', linewidth=2.5, markersize=7, co
ax.plot(years, y3, label='Guayana', marker='*', linewidth=2.5, markersize=15, color

ax.set_xlabel('Año', fontsize=12, fontweight='bold')
ax.set_ylabel('GDP', fontsize=12, fontweight='bold')
ax.set_title('GDP por País (2000-2020)', fontsize=13, fontweight='bold', pad=15)
ax.legend(loc='best', fontsize=11, framealpha=0.95, edgecolor='black')
ax.grid(True, alpha=0.3, linestyle='--')
```

```
plt.tight_layout()
plt.show()
```



3. Representar los mismos datos en 4 subfiguras (axes) distintas, parte de la misma figura

Añade la siguiente info:

– Título: “Datos aleatorios”

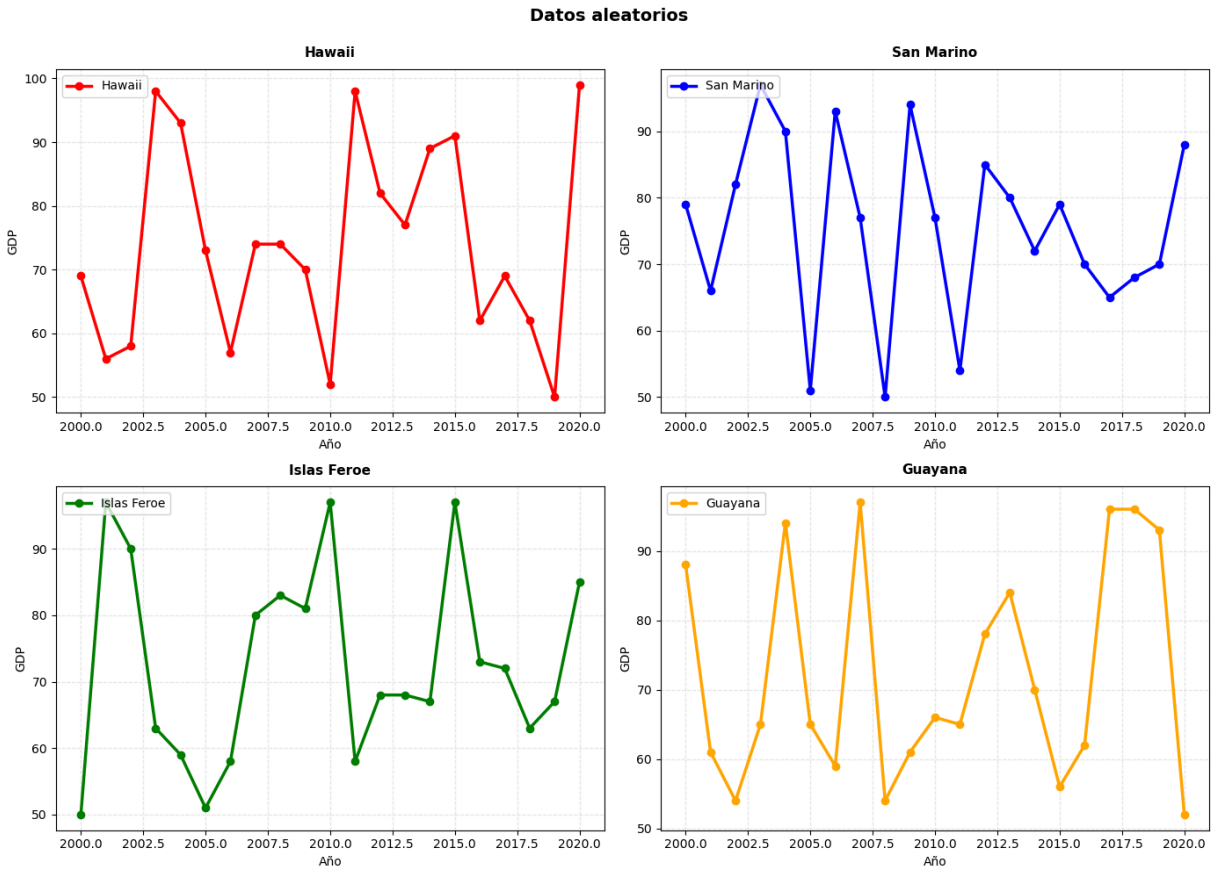
– Leyenda

– Nombre del eje x: “Año”

– Nombre del eje y: “GDP”

In [148…

```
# Crear 4 subfiguras (2x2)
countries = ['Hawaii', 'San Marino', 'Islas Feroe', 'Guayana']
y_data_list = [y0, y1, y2, y3]
colors = ['red', 'blue', 'green', 'orange']

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Datos aleatorios', fontsize=14, fontweight='bold', y=0.995)

axes_flat = axes.flatten()

for idx, (ax, country, y_data, color) in enumerate(zip(axes_flat, countries, y_data
    ax.plot(years, y_data, marker='o', linewidth=2.5, color=color, markersize=6)
    ax.set_title(country, fontsize=11, fontweight='bold', pad=10)
    ax.set_xlabel('Año', fontsize=10)
    ax.set_ylabel('GDP', fontsize=10)
    ax.grid(True, alpha=0.3, linestyle='--')
    ax.legend([country], loc='upper left', fontsize=10)

plt.tight_layout()
plt.show()
```



4. Añade al menos una anotación (texto y flecha) a uno de los gráficos

In [149…

```
# Añadir anotación en uno de los gráficos (destacar máximo)
countries = ['Hawaii', 'San Marino', 'Islas Feroe', 'Guayana']
y_data_list = [y0, y1, y2, y3]
colors = ['red', 'blue', 'green', 'orange']

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Datos aleatorios', fontsize=14, fontweight='bold', y=0.995)

axes_flat = axes.flatten()

for idx, (ax, country, y_data, color) in enumerate(zip(axes_flat, countries, y_data
    ax.plot(years, y_data, marker='o', linewidth=2.5, color=color, markersize=6)
    ax.set_title(country, fontsize=11, fontweight='bold', pad=10)
    ax.set_xlabel('Año', fontsize=10)
    ax.set_ylabel('GDP', fontsize=10)
    ax.grid(True, alpha=0.3, linestyle='--')
    ax.legend([country], loc='upper left', fontsize=10)

    # Anotación: resaltar máximo
    max_idx = np.argmax(y_data)
    max_year = years[max_idx]
    max_value = y_data[max_idx]

    # Punto destacado
    ax.scatter(max_year, max_value, s=250, color='red', zorder=5,
               edgecolors='darkred', linewidth=2.5, marker='*')
```

```
    # Anotación con flecha
    ax.annotate('MÁXIMO',
                xy=(max_year, max_value),
                xytext=(max_year-3, max_value+10),
                arrowprops=dict(arrowstyle='->', color='red', lw=2.5,
                                connectionstyle='arc3,rad=0.3'),
                fontsize=9, fontweight='bold', color='white',
                bbox=dict(boxstyle='round,pad=0.6', facecolor='red',
                          edgecolor='darkred', linewidth=2, alpha=0.9))

plt.tight_layout()
plt.show()
```

8.2 Seaborn

1. Ejercicio: Carga de dataset

Objetivo: Cargar un dataset de seaborn distinto a 'tips', 'iris' o 'flights'. Usar el dataset 'planets' disponible en: https://github.com/mwaskom/seaborn-data

In [150…

```
# Cargar dataset 'planets'
planets = sns.load_dataset('planets')

print("Dataset 'planets' cargado exitosamente")
print(f"Tamaño del dataset: {planets.shape[0]} filas x {planets.shape[1]} columnas"
print("\nPrimeras 5 filas:")
```

```
display(planets.head())

print("\nColumnas disponibles:")
print(planets.columns.tolist())

print("\nTipos de datos:")
print(planets.dtypes)
```

```
Dataset 'planets' cargado exitosamente
Tamaño del dataset: 1035 filas x 6 columnas

Primeras 5 filas:
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

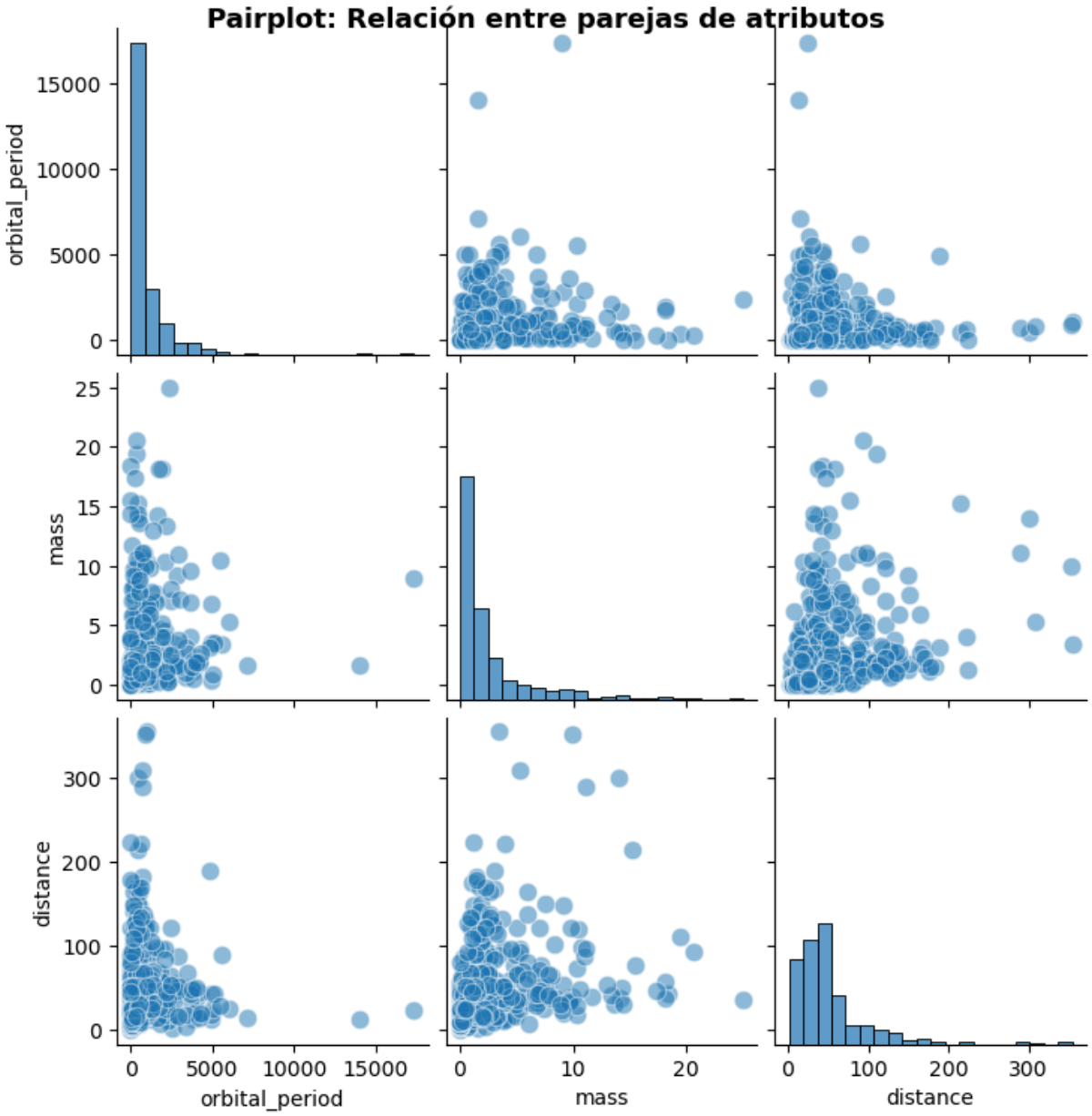
```
Columnas disponibles:
['method', 'number', 'orbital_period', 'mass', 'distance', 'year']

Tipos de datos:
method             object
number              int64
orbital_period    float64
mass              float64
distance          float64
year                int64
dtype: object
```

2. Representa visualmente la relación de cada pareja de atributos (variables)

In [151…

```
# Pairplot: relación de cada pareja de atributos
planets_numeric = planets[['orbital_period', 'mass', 'distance']].dropna()

print(f"Observaciones limpias: {len(planets_numeric)}")

sns.pairplot(planets_numeric, diag_kind='hist', plot_kws={'alpha': 0.5, 's': 80},
             diag_kws={'bins': 20, 'edgecolor': 'black', 'alpha': 0.7})
plt.suptitle('Pairplot: Relación entre parejas de atributos', fontsize=13, fontweig
plt.show()
```

```
Observaciones limpias: 498
```

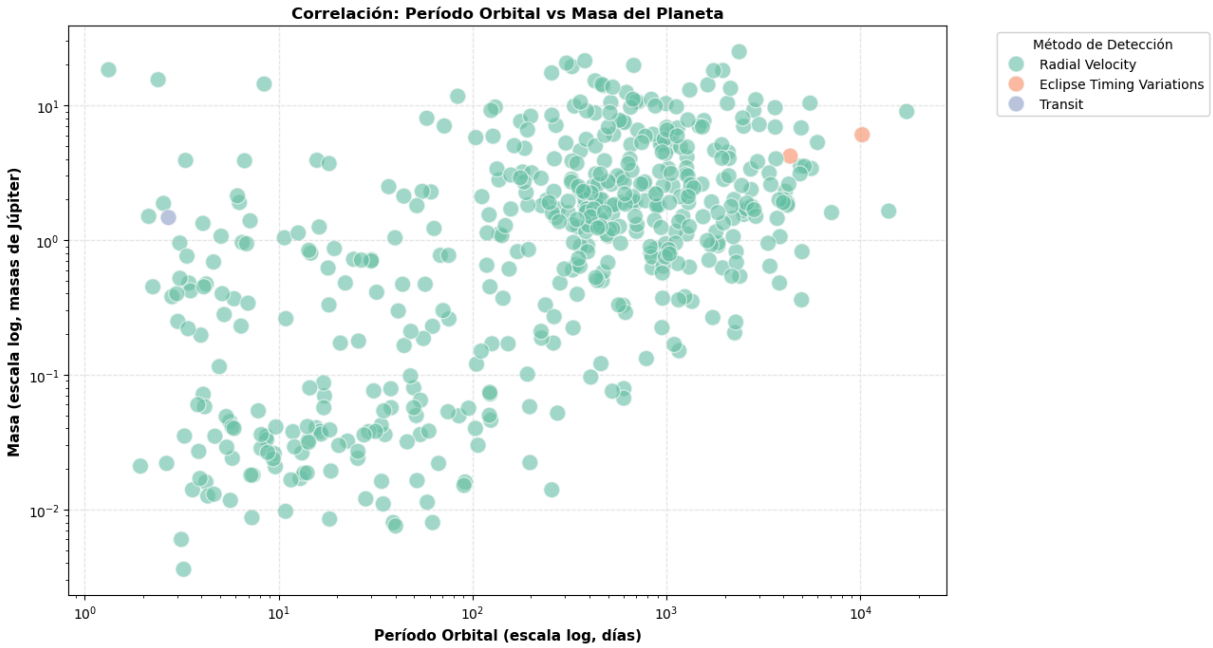


3. Escoge una pareja (ambos valores contínuos y numéricos) y demuestra su correlación

In [152…

```
planets_clean1 = planets.dropna(
    subset=['orbital_period', 'mass', 'method']
)

fig, ax = plt.subplots(figsize=(13, 7))

sns.scatterplot(
    x='orbital_period',
    y='mass',
    hue='method',
    data=planets_clean1,
    s=150,
    alpha=0.6,
    ax=ax,
    palette='Set2'
```

```
)

ax.set_xscale('log')
ax.set_yscale('log')
ax.set_xlabel('Período Orbital (escala log, días)', fontsize=11, fontweight='bold')
ax.set_ylabel('Masa (escala log, masas de Júpiter)', fontsize=11, fontweight='bold'
ax.set_title('Correlación: Período Orbital vs Masa del Planeta', fontsize=12, fontw
ax.legend(title='Método de Detección', bbox_to_anchor=(1.05, 1), loc='upper left',
ax.grid(True, alpha=0.3, linestyle='--')

plt.tight_layout()
plt.show()
```

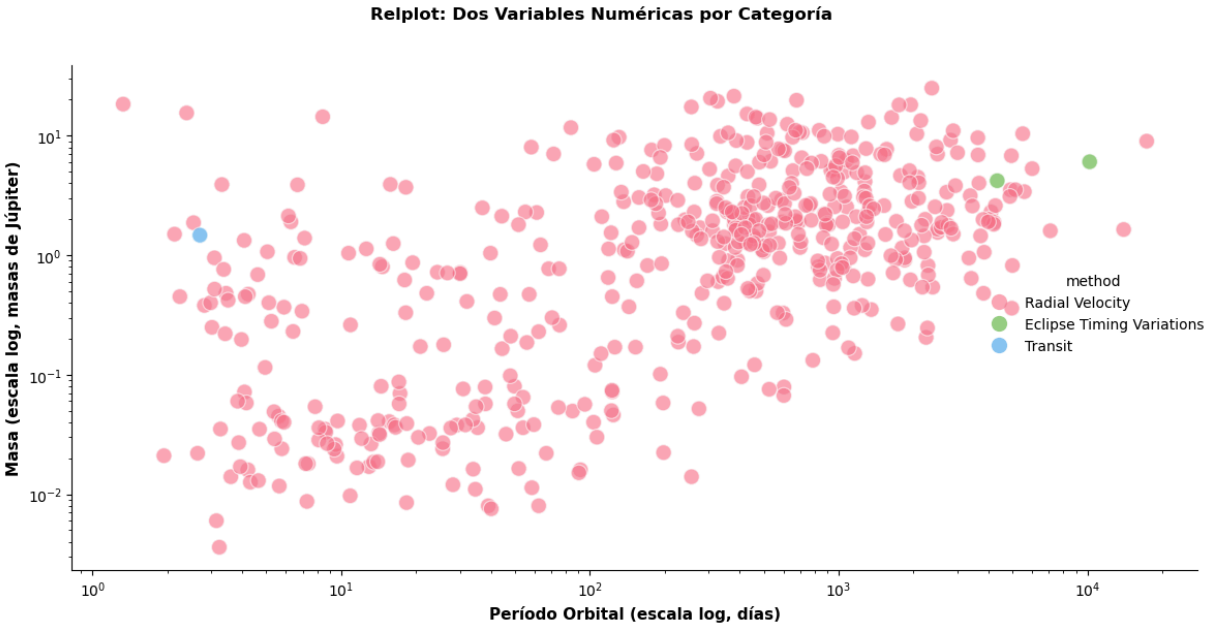


4. Representa los valores de dos variables (x e y, numéricas) en base a otra variable (categórica)

In [153…

```
planets_clean2 = planets.dropna(
    subset=['orbital_period', 'mass', 'method']
)

sns.relplot(
    x='orbital_period',
    y='mass',
    hue='method',
    data=planets_clean2,
    kind='scatter',
    height=6,
    aspect=1.6,
    s=120,
    alpha=0.6,
    palette='husl'
)

plt.xscale('log')
```

```
plt.yscale('log')
plt.xlabel('Período Orbital (escala log, días)', fontsize=11, fontweight='bold')
plt.ylabel('Masa (escala log, masas de Júpiter)', fontsize=11, fontweight='bold')
plt.suptitle(
    'Relplot: Dos Variables Numéricas por Categoría',
    fontsize=12,
    fontweight='bold',
    y=1.02
)

plt.tight_layout()
plt.show()
```



5. Crea los gráficos que demuestren lo siguiente en el dataset seleccionado:

– La distribución de valores (media, distribución, outliers) de una variable numérica contínua

– Compara la distribución de esa variable con otra (contínua y numérica) para representar cuantitativamente la densidad de elementos por valor en ambas variables

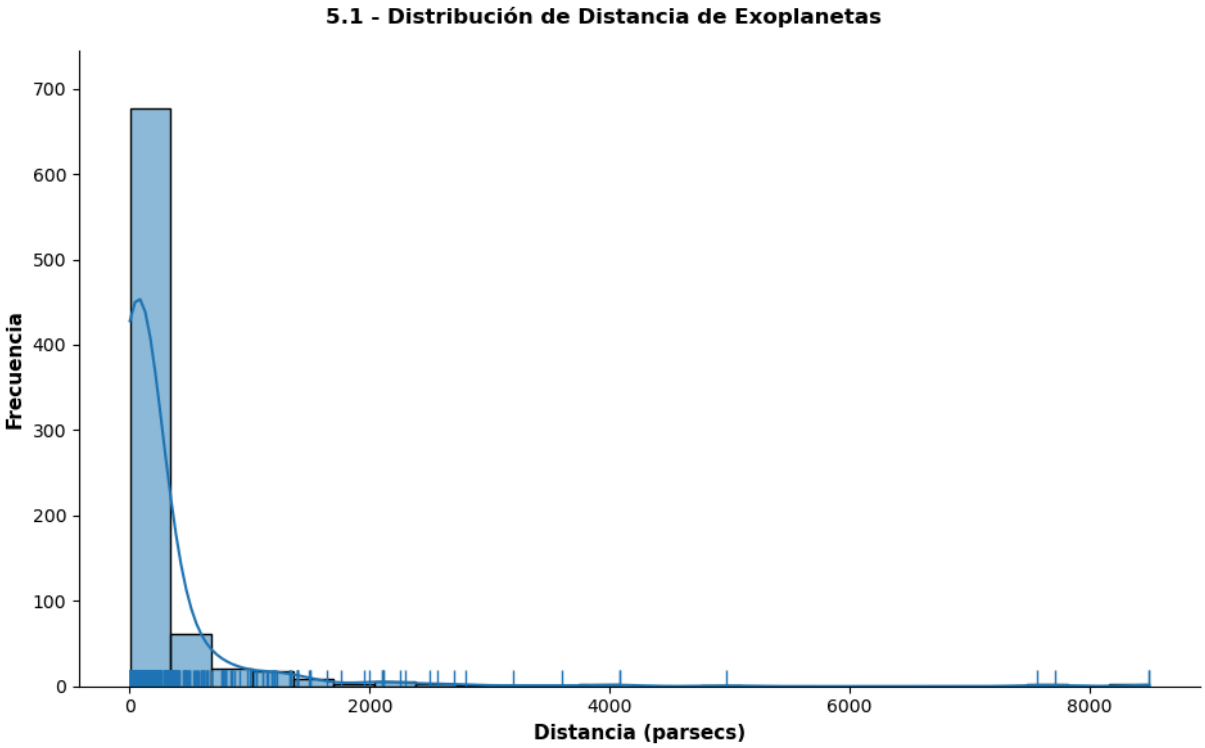
– Muestra un ejemplo en el dataset de una variable a la que un modelo de regresión lineal de orden > 1 sea más ajustado que uno de orden == 1

In [154…

```
# 5.1 Distribución de una variable continua
planets_clean3 = planets.dropna(subset=['distance'])

sns.displot(data=planets_clean3, x='distance', kde=True, bins=25, rug=True, height=
plt.xlabel('Distancia (parsecs)', fontsize=11, fontweight='bold')
plt.ylabel('Frecuencia', fontsize=11, fontweight='bold')
plt.suptitle('5.1 - Distribución de Distancia de Exoplanetas', fontsize=12, fontwei
plt.tight_layout()
plt.show()

print("Estadísticas de Distancia:")
```

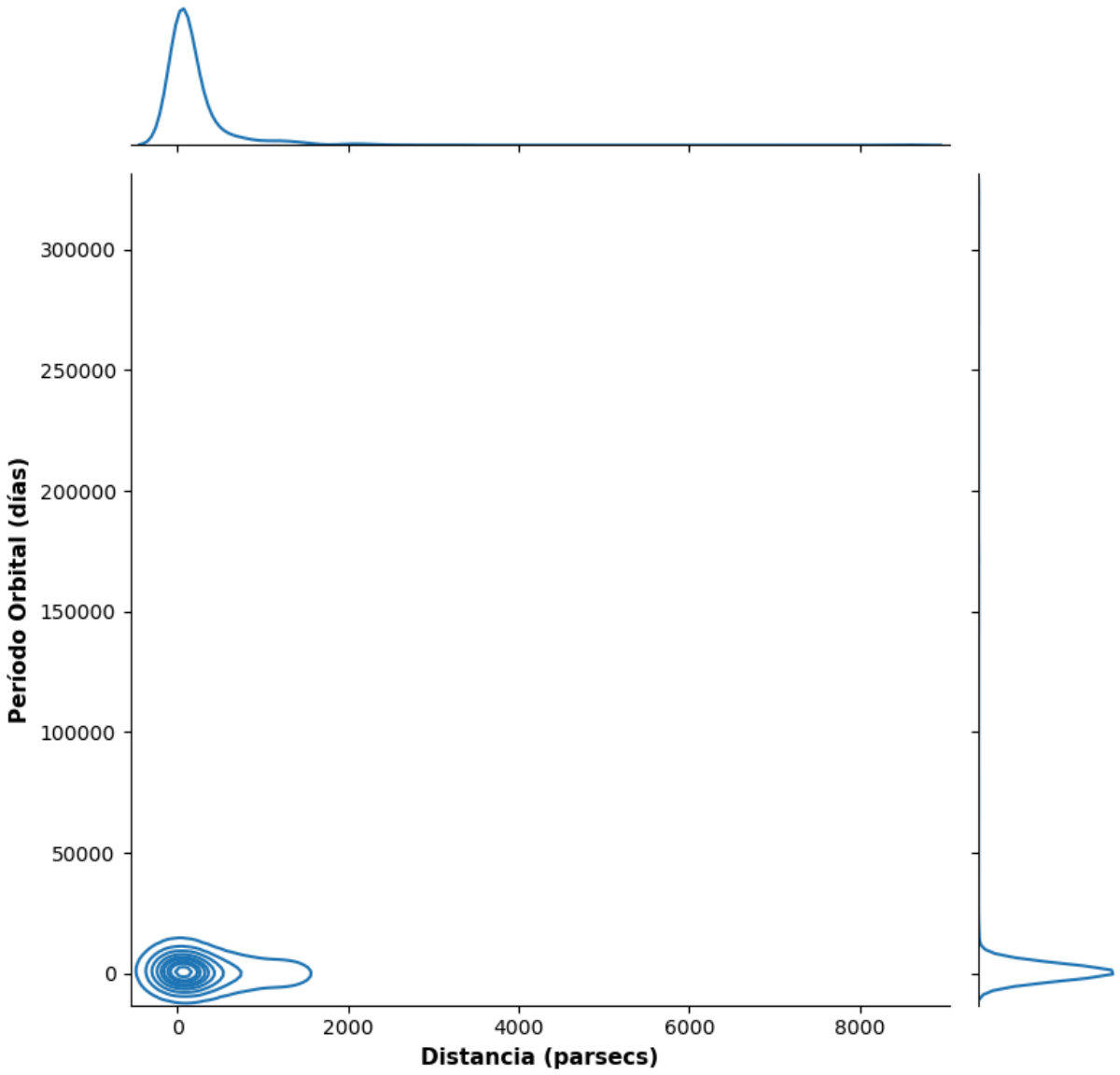
```
print(f"Media: {planets_clean3['distance'].mean():.4f} pc")
print(f"Mediana: {planets_clean3['distance'].median():.4f} pc")
print(f"Desv. Est.: {planets_clean3['distance'].std():.4f} pc")
print(f"Mínimo: {planets_clean3['distance'].min():.4f} pc")
print(f"Máximo: {planets_clean3['distance'].max():.4f} pc")

# 5.2 Comparación de distribuciones bivariables
print("\n" + "="*70)
print("5.2 - DENSIDAD BIVARIABLE")
print("="*70 + "\n")

planets_clean4 = planets.dropna(subset=['distance', 'orbital_period'])

sns.jointplot(x='distance', y='orbital_period', data=planets_clean4, kind='kde', he
plt.xlabel('Distancia (parsecs)', fontsize=11, fontweight='bold')
plt.ylabel('Período Orbital (días)', fontsize=11, fontweight='bold')
plt.suptitle('5.2 - Densidad Bivariable: Distancia vs Período Orbital', fontsize=12
plt.tight_layout()
plt.show()

# 5.3 Regresión: Orden 1 vs Orden 3
print("\n" + "="*70)
print("5.3 - REGRESIÓN LINEAL vs POLINOMIAL")
print("="*70 + "\n")

planets_clean5 = planets.dropna(subset=['orbital_period', 'mass'])

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Orden 1
sns.regplot(x='orbital_period', y='mass', data=planets_clean5, order=1, ax=axes[0],
            scatter_kws={'alpha': 0.5, 's': 100}, line_kws={'linewidth': 3, 'color':
axes[0].set_title('Regresión LINEAL (Orden 1)', fontsize=12, fontweight='bold')
axes[0].set_xlabel('Período Orbital (días)', fontsize=11, fontweight='bold')
axes[0].set_ylabel('Masa (masas de Júpiter)', fontsize=11, fontweight='bold')
axes[0].set_xscale('log')
axes[0].set_yscale('log')
axes[0].grid(True, alpha=0.3, linestyle='--')
axes[0].text(0.05, 0.95, 'Ajuste: BÁSICO', transform=axes[0].transAxes, fontsize=11
             verticalalignment='top', fontweight='bold', bbox=dict(boxstyle='round',

# Orden 3
sns.regplot(x='orbital_period', y='mass', data=planets_clean5, order=3, ax=axes[1],
            scatter_kws={'alpha': 0.5, 's': 100}, line_kws={'linewidth': 3, 'color':
axes[1].set_title('Regresión POLINOMIAL (Orden 3)', fontsize=12, fontweight='bold')
axes[1].set_xlabel('Período Orbital (días)', fontsize=11, fontweight='bold')
axes[1].set_ylabel('Masa (masas de Júpiter)', fontsize=11, fontweight='bold')
axes[1].set_xscale('log')
axes[1].set_yscale('log')
axes[1].grid(True, alpha=0.3, linestyle='--')
axes[1].text(0.05, 0.95, 'Ajuste: MEJORADO ✓', transform=axes[1].transAxes, fontsiz
             verticalalignment='top', fontweight='bold', bbox=dict(boxstyle='round',

fig.suptitle('5.3 - Comparación: Regresión Lineal vs Polinomial', fontsize=13, font
plt.tight_layout()
plt.show()
```

```
print("CONCLUSIÓN: La regresión polinomial (orden 3) se ajusta mejor a los datos qu
```

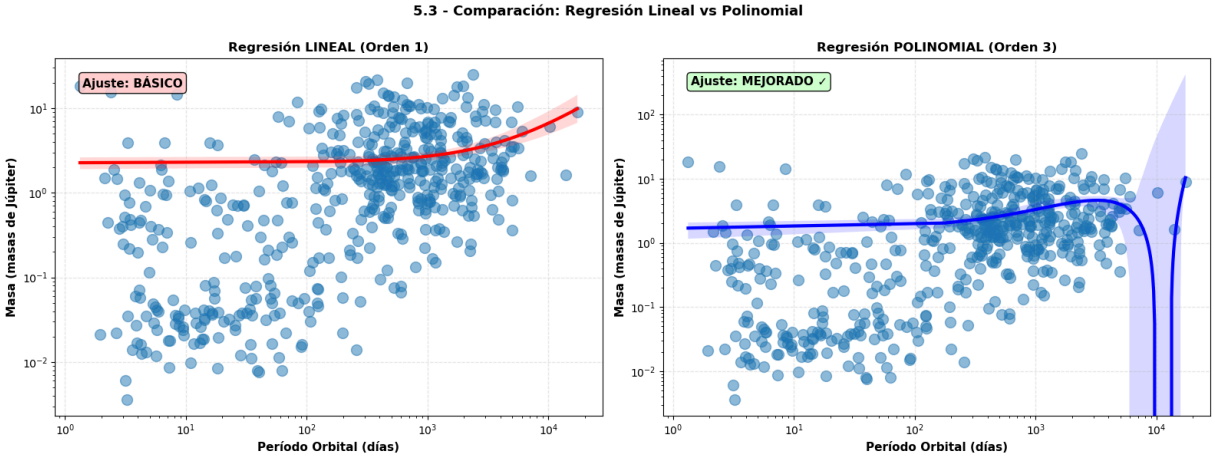


Estadísticas de Distancia:
Media: 264.0693 pc
Mediana: 55.2500 pc
Desv. Est.: 733.1165 pc
Mínimo: 1.3500 pc
Máximo: 8500.0000 pc

==========
5.2 - DENSIDAD BIVARIABLE
==========




```
======================================================================
5.3 - REGRESIÓN LINEAL vs POLINOMIAL
======================================================================
```



```
CONCLUSIÓN: La regresión polinomial (orden 3) se ajusta mejor a los datos que la lin
eal.
```

GitHub

https://github.com/Erick-305/machine-learning.git