

## VISUALIZACIÓN

### Librerías:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

### Tipos de datos:

**Cualitativos:** categóricos

- Ordinal: con un orden lógico → Nivel de satisfacción: baja, media, alta
- Nominal: sin orden → Tipos de animales en una granja: vaca, cerdo, gallina

**Cuantitativos:** numéricos

- Discreto: número limitado, se pueden contar individualmente → número de amigos, calificación, cantidad de likes
- Continuo: número infinito en un rango, cantidades que se pueden medir → peso, altura, temperatura, edad, volumen

**Outliers:** valores atípicos, que se encuentran muy alejados del resto de datos.

### Tipos de análisis

**Univariado:** promedio de edad, cuántas veces aparece cada color de ojos.

**Bivariado:** relación entre tiempo de estudio y calificación en un examen.

**Multivariado:** cómo afectan la dieta, el ejercicio y el sueño en la salud.

`plt.xlabel("etiqueta")` Etiqueta del eje x

`plt.ylabel("etiqueta")` Etiqueta del eje y

`plt.title("título", fontsize = 8)` Título gráfica y tamaño fuente

`plt.gca().spines['right'].set_visible(False)` Quitamos la línea de la derecha

`plt.gca().spines["top"].set_visible(False)` Quitamos la línea de arriba

`plt.xlim([min, max])` Mínimo y máximo de los valores del eje x

`plt.ylim([min, max])` Mínimo y máximo de los valores del eje y

`plt.savefig("mi_grafica_bonita.jpg")` Guardar gráfica en jpg

`plt.xticks(ticks, labels, rotation=90)` Modificar el eje x

`plt.yticks(ticks, labels, rotation=90)` Modificar el eje y

## HISTOGRAMA O HISTPLOT:

Representa la **distribución** de una **variable numérica**.

Para **identificar** sesgos, valores atípicos, **outliers**.

### Seaborn:

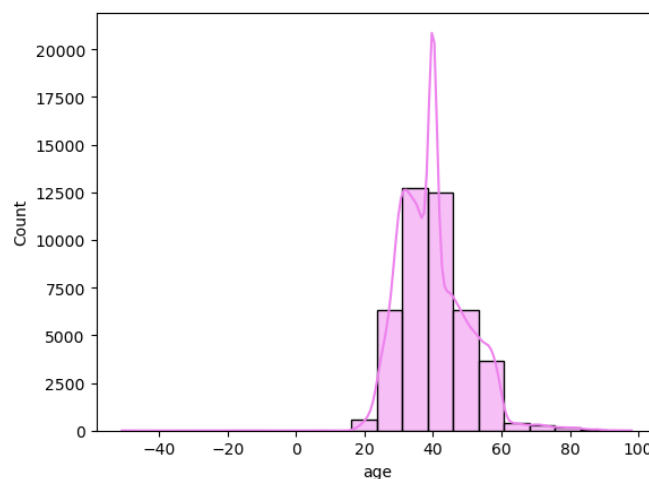
```
sns.histplot(data=df, x="variable", bins=10, kde=False, color="blue")
```

- data: DataFrame
- x: variable que analizar
- bins: (op) nº de barras que muestra para representar los valores
- kde: (op) línea sobre las barras que muestra el patrón de los valores
- color: (op) por defecto blue

### Matplotlib:

```
plt.hist(data=df, x="variable", bins=None, range=None, density=False, cumulative=False, color=None, edgecolor=None)
```

- data: DataFrame
- x: variable que analizar
- bins: (op) nº de barras que muestra para representar los valores
- range: (op) lista de valores mínimo y máximo del eje x
- density: (op) True densidad, False frecuencia, valores eje y
  - densidad: proporción de datos en ese bin
  - frecuencia: nº de datos en el bin
- cumulative: (op) False distribución normal, True mostrará distribución acumulativa
- color: (op) de las barras
- edgecolor: (op) del borde de las barras



## GRÁFICO DE DISPERSIÓN O SCATTERPLOT:

Identificar patrones, tendencias o **correlaciones** entre **dos variables numéricas**.  
Para **identificar** sesgos, valores atípicos, **outliers**.

### Seaborn:

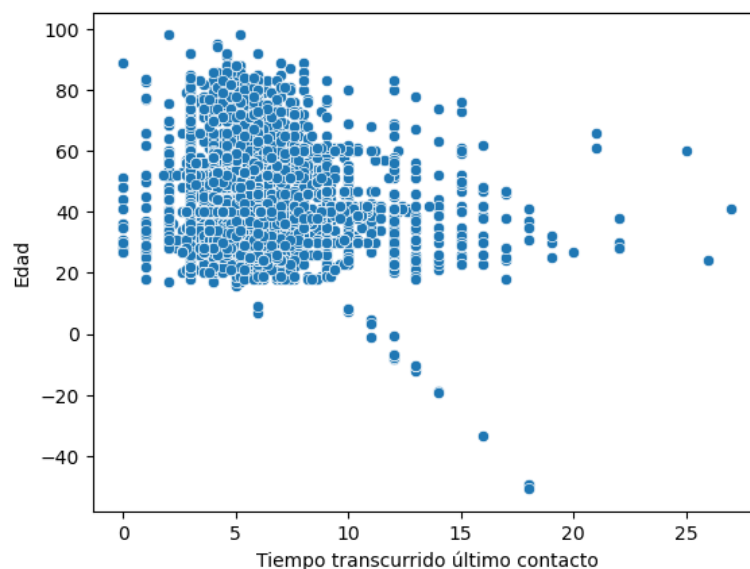
```
sns.scatterplot(x = "variable", y = "variable", data = df, hue: "variable", size = 5,  
marker= "v", color= "purple")
```

- x: variable eje x
- y: variable eje y
- data: DataFrame
- hue: (op) una tercera variable para hacer grupos
- size: (op) tamaño de los puntos
- marker: (op) tipo de marcador, por defecto círculo

### Matplotlib:

```
plt.scatter(x = "variable", y = "variable", data = df, marker= "^", c = "purple")
```

- x: variable eje x
- y: variable eje y
- data: DataFrame
- marker: (op) tipo de marcador, por defecto círculo
- c: (op) color
- s: (op) tamaño



## GRÁFICO DE REGRESIÓN O REGPLOT:

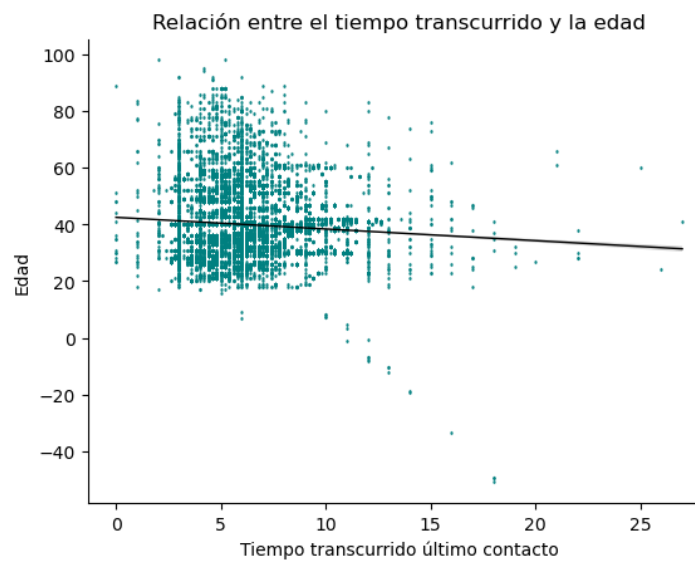
Explorar la **relación** entre **dos variables numéricas** y ver si existe una **correlación lineal** entre ellas.

### Seaborn:

```
sns.regplot(x = "variable", y = "variable", data = df, marker= "d",  
scatter_kws={"color": "purple", "s": 1}, line_kws={"color": "green", "linewidth": 1})
```

- x: variable eje x
- y: variable eje y
- data: DataFrame
- marker: (op) tipo de marcador, por defecto círculo
- scatter\_kws: color y tamaño de los puntos
- line\_kws: color y tamaño de las líneas

Para este tipo de gráfico no tenemos método en matplotlib



## SUBPLOTS: crear en una figura varias gráficas

`fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 5))`

- `nrows`: número filas
- `ncols`: número columnas
  - 1, 2 muestra las dos gráficas en horizontal
  - 2, 1 muestra las dos gráficas en vertical
- `figsize`: pulgadas de la figura

Poner el primer Scatterplot o Regplot poniendo en los parámetros `ax = axes[0]`

`axes[0].xlabel("etiqueta")` Etiqueta del eje x

`axes[0].ylabel("etiqueta")` Etiqueta del eje y

`axes[0].title("título", fontsize = 8)` Título gráfica

`axes[0].gca().spines['right'].set_visible(False)` Quitamos la línea de la derecha

`axes[0].gca().spines["top"].set_visible(False)` Quitamos la línea de arriba

Poner el segundo Scatterplot o Regplot, poniendo `axes[1]` en lugar de `sns.` o `plt.`

`axes[1].xlabel("etiqueta")` Etiqueta del eje x

`axes[1].ylabel("etiqueta")` Etiqueta del eje y

`axes[1].title("título", fontsize = 8)` Título gráfica

`axes[1].gca().spines['right'].set_visible(False)` Quitamos la línea de la derecha

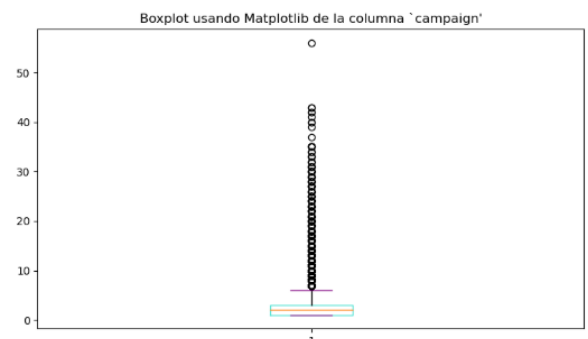
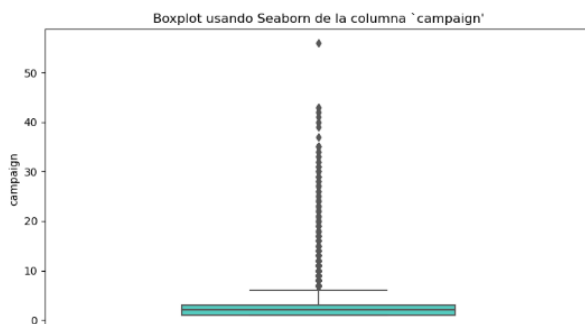
`axes[1].gca().spines["top"].set_visible(False)` Quitamos la línea de arriba

`axes[n].set_xticks(ticks, labels, rotation)` Modificar eje x

`axes[n].set_yticks(ticks, labels, rotation)` Modificar eje y

`axes[n].set_xlim([mix, max])` Limitar los valores del eje x

`axes[n].set_ylim([mix, max])` Limitar los valores del eje



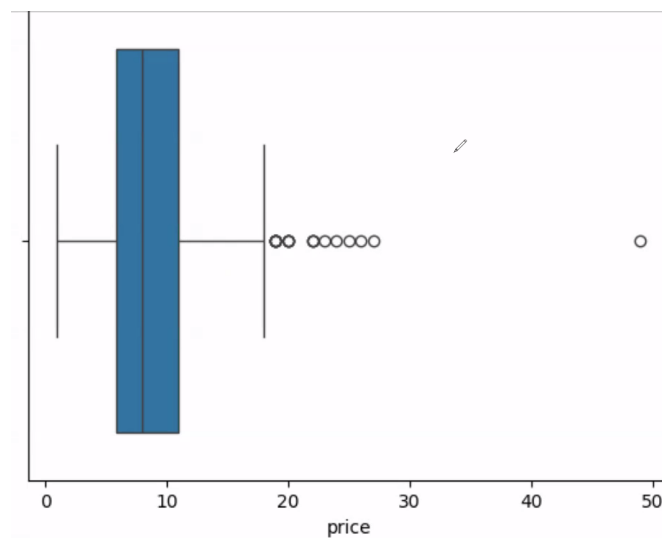
## GRÁFICO DE CAJAS O BOXPLOT Y VIOLINPLOT:

Explorar la **distribución** y **valores atípicos** en variables **numéricas**.

### Seaborn:

```
sns.boxplot(x = "variable", data = df, width = 0.5, color = "turquoise")
```

- x: variable eje x
- y: variable eje y
- data: DataFrame
- width: (op) ancho de las cajas



La caja tiene una línea central que es la mediana. La caja significa que el 50% de los datos está entre esos valores. Hacia la izquierda de la caja, Q1 el 25% de los valores es más bajo. Hacia la izquierda, Q3 el 25% de los valores es más alto. Lo que está fuera de las líneas son outliers.

### Matplotlib:

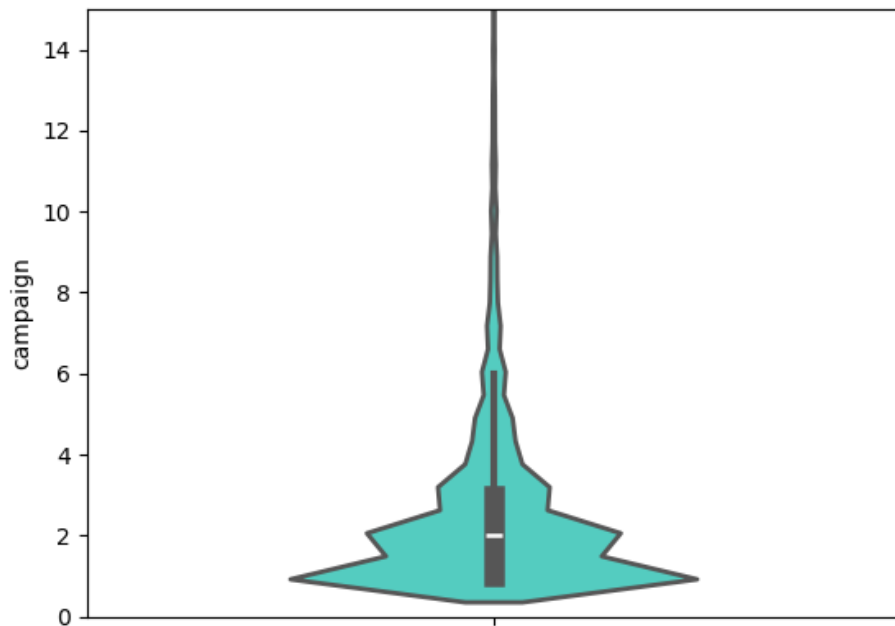
```
plt. boxplot (x = "variable", data = df, boxprops = {"color": "turquoise"},  
capprops={'color':'purple'})
```

- x: variable eje x
- y: variable eje y
- data: DataFrame
- boxprops: color de la línea
- capprops: color de los bigotes

### Violinplot:

`sns.violinplot(y = "variable", data = df, width = 0.5, color = "turquoise", linewidth = 2)`

- x: variable eje x
- y: variable eje y
- data: DataFrame
- width: ancho de las cajas
- linewidth: ancho del contorno



Punto blanco: la mediana

Línea gorda negra Q1 y Q3

## COUNTPLOT:

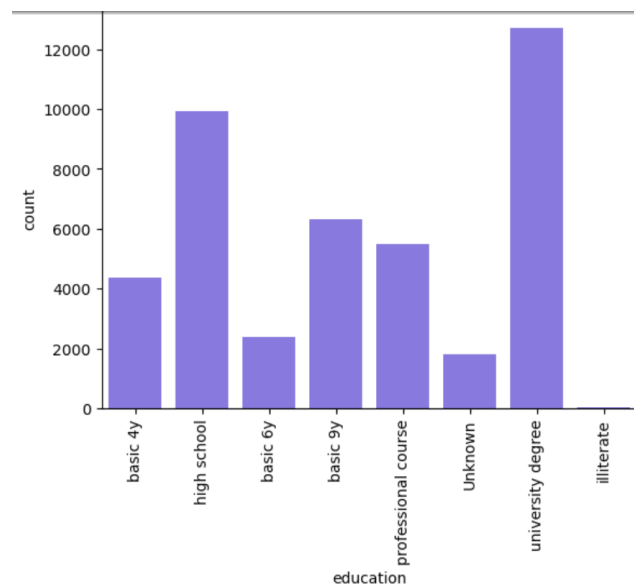
### Seaborn:

Explorar la **frecuencia** en que se repite una **variable categórica**. En el eje x la variable que medimos, en el eje y el número de veces que aparece

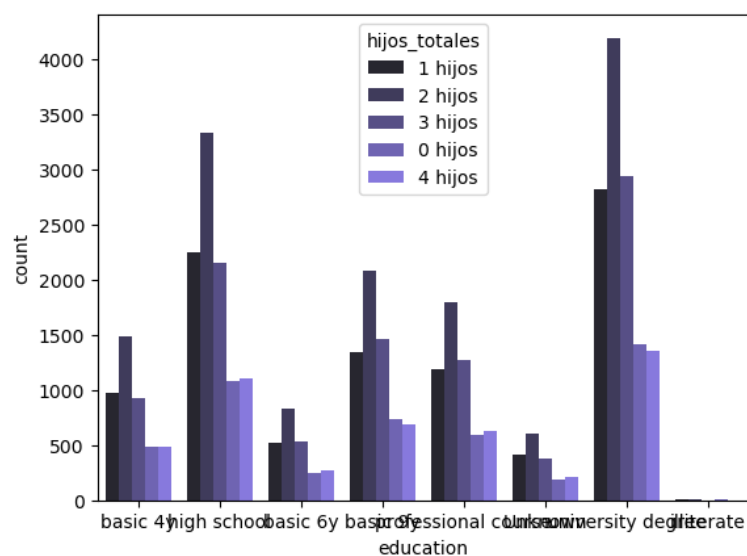
`sns.countplot(x, data, palette, color, hue, order, orient)`

- hue: permite añadir otra variable categórica, funciona como un group by agrupado por la variable de x
- order y hue\_order: lista con orden de las barras
- orient: orientación de las barras

Sin hue:



Con hue:





## GRÁFICO DE QUESITOS O PIEPLOT:

### Matplotlib:

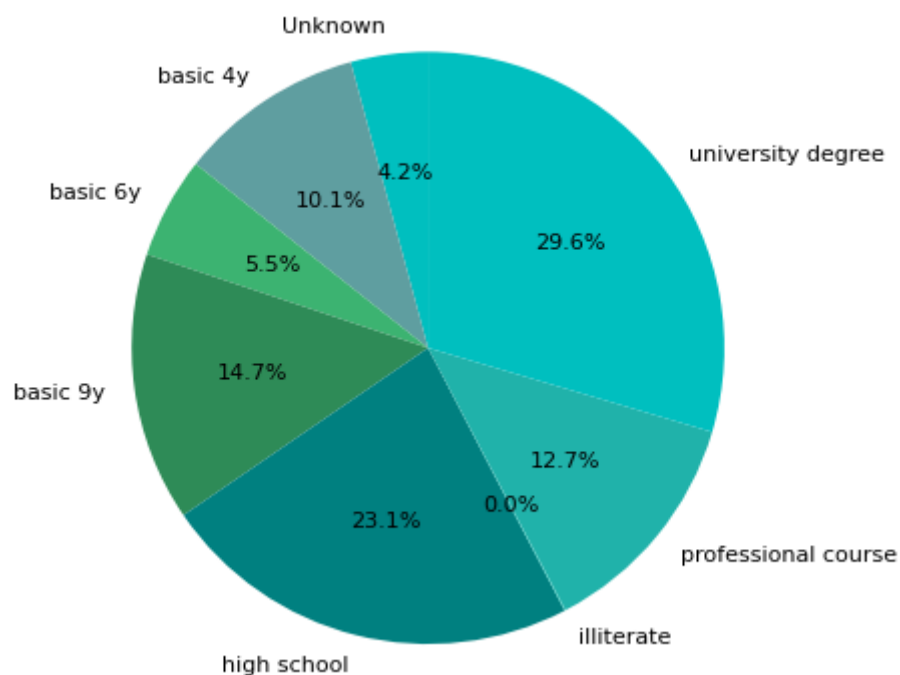
Muestra la proporción de una categoría dentro del conjunto.

Crear un DF con la agrupación que se quiere mostrar en la gráfica.

```
plt.pie(valores, labels, autopct = '%1.1f%%', startangle, data, colors)
```

- valores: de los que se muestra la proporción
- labels: categorías/datos a mostrar en la gráfica por agrupación
- autopct: % añade símbolo, 1 al menos un dígito, .1 un decimal
- colors: lista con tantos colores como categorías

	education	id
0	Unknown	1807
1	basic 4y	4356
2	basic 6y	2386
3	basic 9y	6309
4	high school	9925
5	illiterate	18
6	professional course	5477
7	university degree	12722



## BARPLOT:

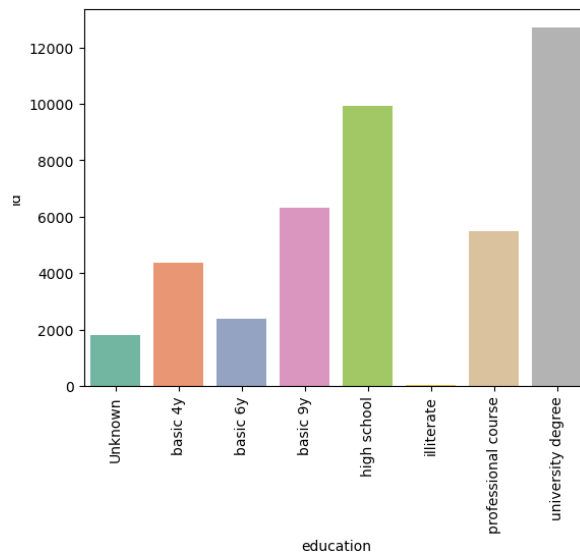
Muestra la distribución, **comparación** y relaciones entre diferentes categorías. En el eje x variable **categorica**, en el eje y variable **numérica**.

Crear un DF con la agrupación que se quiere mostrar en la gráfica.

## Seaborn:

`sns.barplot(x, y, palette, ci, hue)`




- `palette`: muestra una barra de cada color
- `hue`: añadir variable



## Matplotlib:

`plt.bar(categorias, height, data, color)`

- `categorias`: Representa las categorías o etiquetas en el eje X del gráfico de barras.
- `height`: Corresponde a los valores numéricos asociados con cada categoría en el eje X.

	black		linen		forestgreen		slategray
	k		bisque		limegreen		lightsteelblue
	dimgray		darkorange		darkgreen		cornflowerblue
	dimgray		burlywood		green		royalblue
	grey		antiquewhite		g		ghostwhite
	gray		tan		lime		lavender
	darkgray		navajowhite		seagreen		midnightblue
	darkgrey		blanchedalmond		mediumseagreen		navy
	silver		papayawhip		springgreen		darkblue
	lightgrey		moccasin		mintcream		mediumblue
	lightgray		orange		mediumspringgreen		blue
	gainsboro		wheat		mediumaquamarine		b
	whitesmoke		oldlace		aquamarine		slateblue
	white		floralwhite		turquoise		darkslateblue
	w		darkgoldenrod		lightseagreen		mediumslateblue
	snow		goldenrod		mediumturquoise		mediumpurple
	rosybrown		cornsilk		azure		rebeccapurple
	lightcoral		gold		lightcyan		blueviolet
	indianred		lemonchiffon		paleturquoise		indigo
	brown		khaki		darkslategray		darkorchid
	firebrick		palegoldenrod		darkslategrey		darkviolet
	maroon		darkkhaki		teal		mediumorchid
	darkred		ivory		darkcyan		thistle
	red		beige		c		plum
	r		lightyellow		cyan		violet
	mistyrose		lightgoldenrodyellow		aqua		purple
	salmon		olive		darkturquoise		darkmagenta
	tomato		y		cadetblue		m
	darksalmon		yellow		powderblue		magenta
	coral		olivedrab		lightblue		fuchsia
	orangered		yellowgreen		deepskyblue		orchid
	lightsalmon		darkolivegreen		skyblue		mediumvioletred
	sienna		greenyellow		lightskyblue		deeppink
	seashell		chartreuse		steelblue		hotpink
	chocolate		lawngreen		aliceblue		lavenderblush
	saddlebrown		honeydew		dodgerblue		palevioletred
	sandybrown		darkseagreen		lightslategray		crimson
	peachpuff		palegreen		lightslategray		pink
	peru		lightgreen		slategray		lightpink