

ACADEMICA

---

**¡Bienvenidos/as  
a Data Science!**



# Agenda

---

¿Cómo anduvieron?

Repaso: Visualización de datos

Explicación: Visualización de datos - Seaborn

Hands-on training

Break

¿Sabías que...?

Hands-on training

Cierre



# ¿Dónde estamos?



# ¿Cómo anduvieron?

A



# Repasso: Visualización de Datos



# ¿Por qué visualizar?

**Visualizar** los datos es una parte fundamental del análisis en ciencia de datos.



## ¿Por qué visualizar?

No solo sirve para **comunicar** (que es una parte fundamental del trabajo) sino que también es una herramienta esencial para **comprender el dataset** con el que estamos trabajando.

# ¿Por qué visualizar?

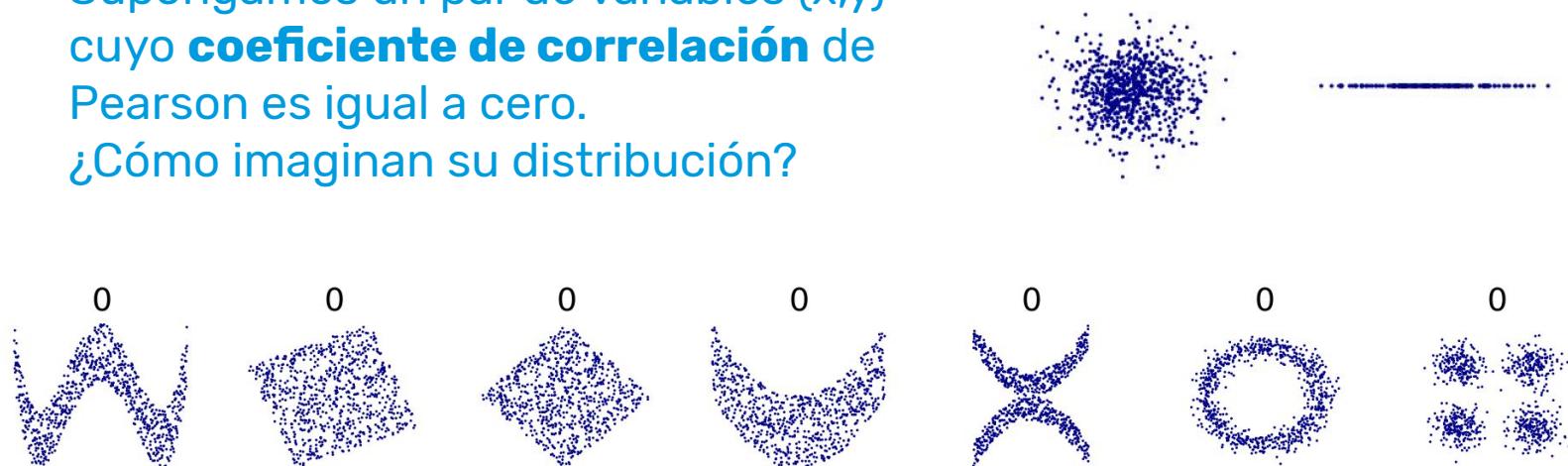
Hay veces que sólo indicadores numéricos no alcanzan para describir las características principales de nuestro dataset.

Supongamos un par de variables ( $x,y$ )  
cuyo **coeficiente de correlación** de  
Pearson es igual a cero.  
¿Cómo imaginan su distribución?

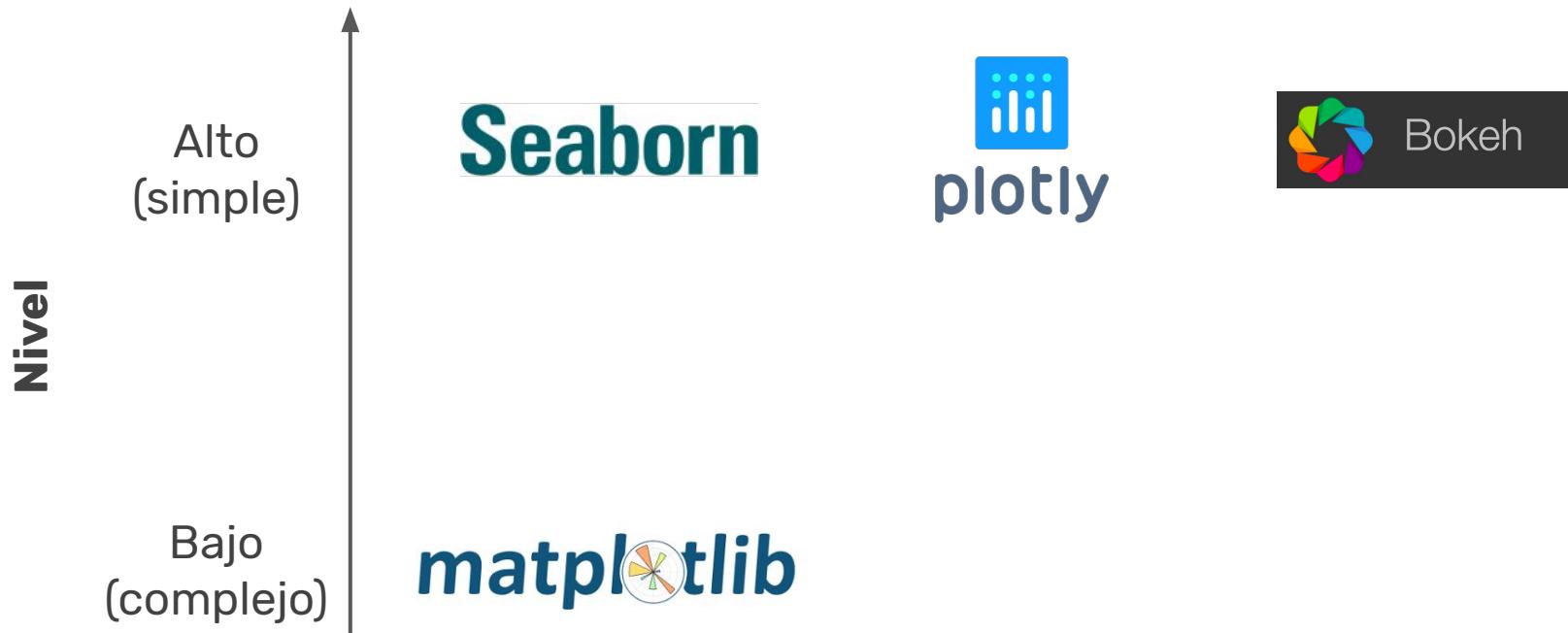
# ¿Por qué visualizar?

Hay veces que sólo indicadores numéricos no alcanzan para describir las características principales de nuestro dataset.

Supongamos un par de variables ( $x,y$ ) cuyo **coeficiente de correlación** de Pearson es igual a cero.  
¿Cómo imaginan su distribución?



# Herramientas de Visualización



# Herramientas de Visualización



# Documentación

## Gallery

This gallery contains examples of the many things you can do with Matplotlib. Click on any image to see the full code.

For longer tutorials, see our [tutorials page](#). You can also find [external resources](#) and a [FAQ](#) in our [user guide](#).

### Lines, bars and markers



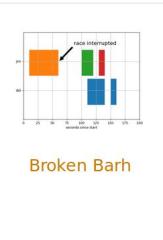
Stacked Bar Graph



Grouped bar chart with labels



Horizontal bar chart



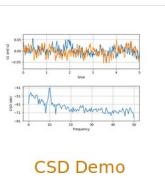
Broken Barh



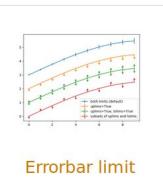
Plotting



Plotting the



CSD Demo



Errorbar limit

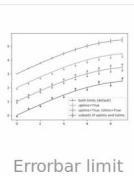
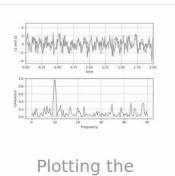
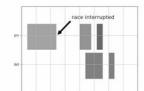
# Documentación

## Gallery

This gallery contains examples of the many things you can do with Matplotlib. Click on any image to see the full code.

For longer tutorials, see our [tutorials page](#). You can also find external resources and a FAQ in our user guide.

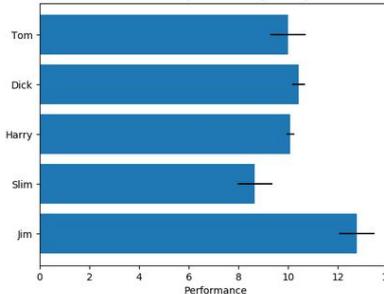
### Lines, bars and markers



### Horizontal bar chart

This example showcases a simple horizontal bar chart.

How fast do you want to go today?



```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)

plt.rcParams()

fig, ax = plt.subplots()

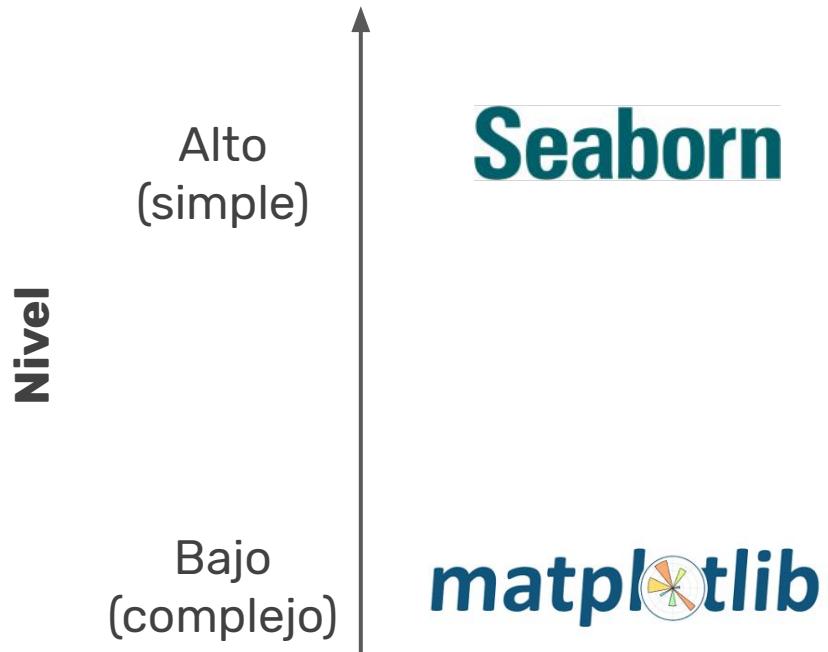
# Example data
people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')
y_pos = np.arange(len(people))
performance = 3 + 10 * np.random.rand(len(people))
error = np.random.rand(len(people))

ax.broken_barh(y_pos, performance, xerr=error, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(people)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Performance')
ax.set_title('How fast do you want to go today?')

plt.show()
```

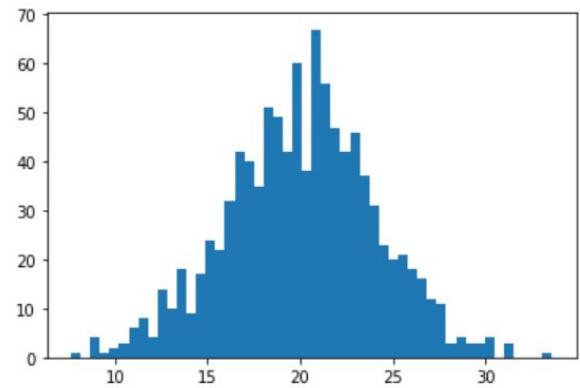
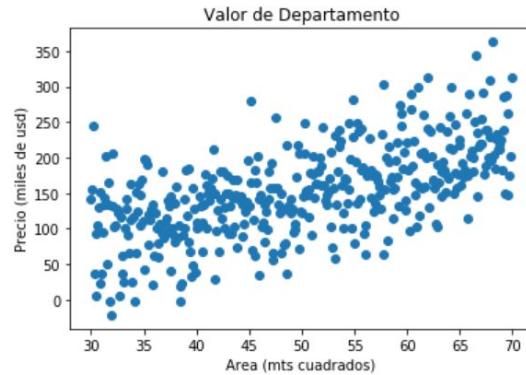
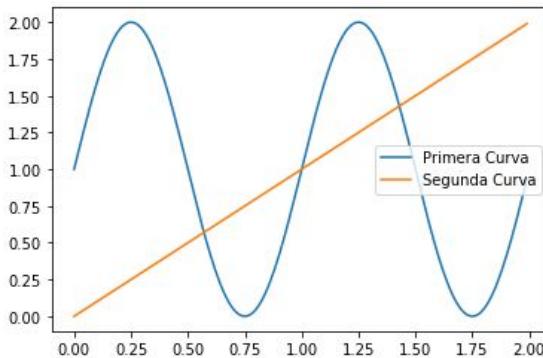


# Tipos de gráfico



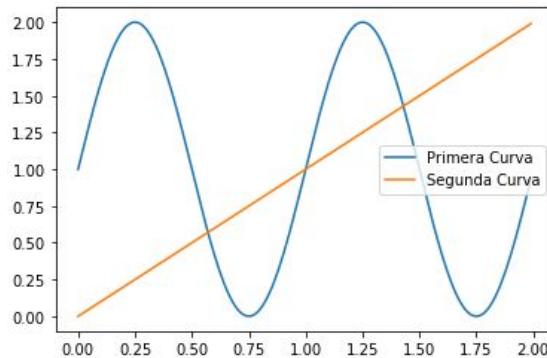
¡Usaremos  
éstas dos!

# Tipos de Gráficos

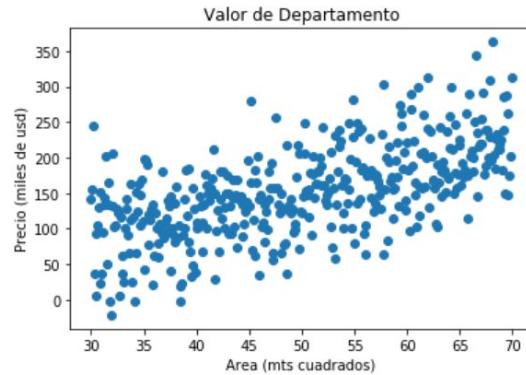


¿Cómo llamamos a cada gráfico?  
¿Cuándo usarían uno u otro?

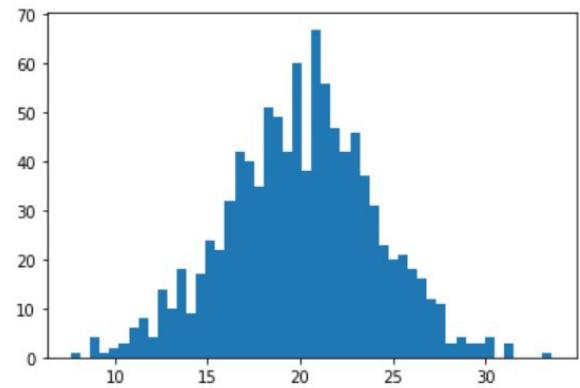
# Tipos de Gráficos



Line Plot



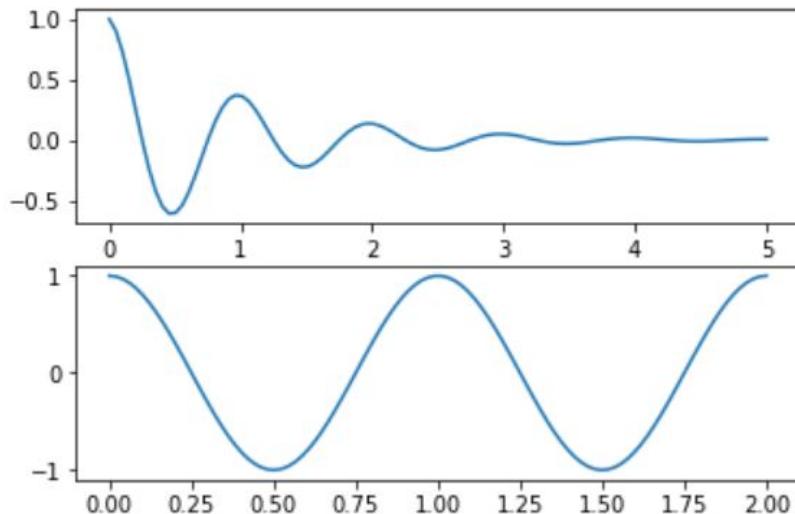
Scatter  
Plot



Histograma

# Tipos de Gráficos

Graficos Alineados en forma vertical



¿Y este tipo de gráfico?  
¿Cuándo será útil?

# Visualización de Datos - Seaborn



¿ Seaborn o matplotlib ?



# ¿ Seaborn o matplotlib ?



- **Seaborn corre sobre Matplotlib.** Es por esto que trabaja con objetos definidos en esa librería, como por ejemplo figuras y ejes.
- La **manera de utilizarlo** es muy parecida (sabiendo usar Matplotlib resulta fácil usar Seaborn).
- La **principal diferencia** radica en la habilidad de Seaborn de poder importar los datos directamente desde un DataFrame.

# Seaborn vs. matplotlib

## VENTAJAS

- Facilita el trabajo con DataFrames
- Mejora automáticamente la estética de los gráficos
- Sintaxis sencilla para algunos gráficos complejos.

## DESVENTAJAS

- Precisa la instalación de una librería adicional (esto puede ser perjudicial en algunos contextos)
- Menos flexible (configurable) que Matplotlib

# Seaborn y DataFrames

Supongan que estamos trabajando con el dataset Iris:

```
[30]: iris_data = pd.read_csv('DS_Clase_04_iris.csv')
iris_data.head()
```

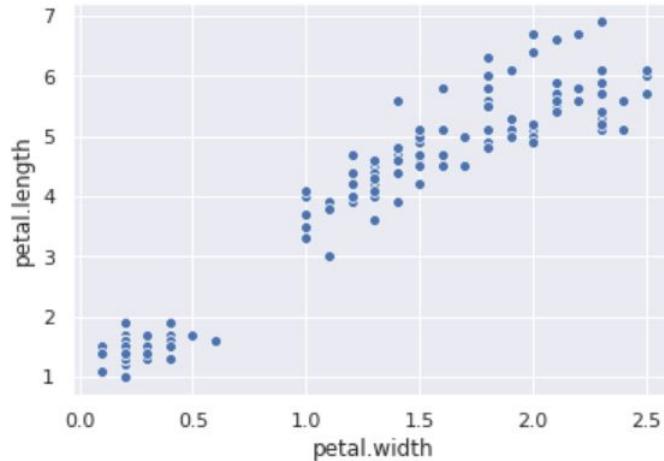
```
[30]:   sepal.length  sepal.width  petal.length  petal.width  variety
      0           5.1          3.5          1.4          0.2    Setosa
      1           4.9          3.0          1.4          0.2    Setosa
      2           4.7          3.2          1.3          0.2    Setosa
      3           4.6          3.1          1.5          0.2    Setosa
      4           5.0          3.6          1.4          0.2    Setosa
```

# Seaborn y DataFrames

Con Seaborn podemos graficar tomando directamente los datos desde el DataFrame que los contiene:

```
[31]: sns.scatterplot(x="petal.width", y="petal.length", data=iris_data)
```

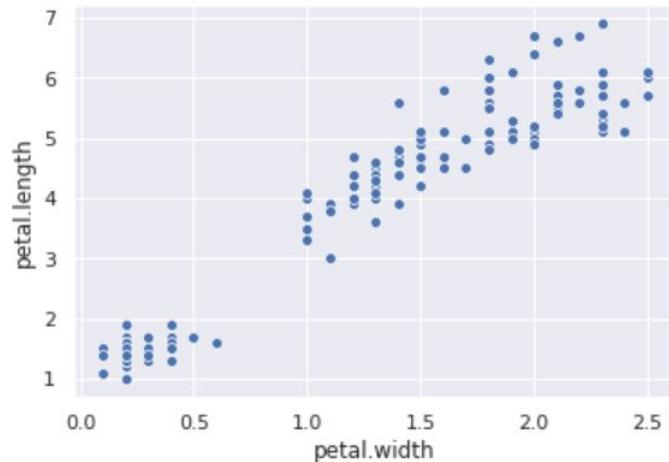
```
[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f48a8d4f208>
```



# Seaborn y DataFrames

En '**x**' e '**y**' le decimos que columnas del DataFrame queremos que tome para graficar.

```
[31]: sns.scatterplot(x="petal.width", y="petal.length", data=iris_data)  
[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f48a8d4f208>
```



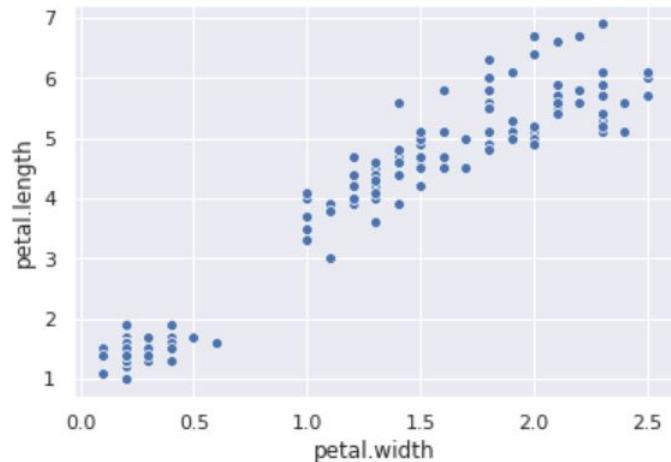
En 'data' le indicamos el nombre del **DataFrame**

# Seaborn y DataFrames

Acá pasamos un **string**, con el nombre de las columnas del **DataFrame** que queremos que tome para graficar en el eje 'x' y el eje 'y'.

```
[31]: sns.scatterplot(x="petal.width", y="petal.length", data=iris_data)  
[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f48a8d4f208>
```

En 'data' le indicamos el nombre del **DataFrame**



# Seaborn • Parámetros

Una de las características de **Seaborn** es que a través de los **parámetros** de sus funciones, nos permite diferenciar distintos **subsets** de los datos fácilmente.

seaborn

0.9.0

Gallery

Tutorial

API

Site ▾

Page ▾

## seaborn.scatterplot

```
seaborn.scatterplot (x=None, y=None, hue=None, style=None, size=None, data=None, palette=None, hue_order=None,  
hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=True, style_order=None, x_bins=None, y_bins=None, units=None,  
estimator=None, ci=95, n_boot=1000, alpha='auto', x_jitter=None, y_jitter=None, legend='brief', ax=None, **kwargs)
```

# Seaborn • Parámetros

Estos parámetros son: **hue**, **style** y **size**.

The screenshot shows the Seaborn API documentation page. At the top, there is a navigation bar with links for 'seaborn 0.9.0', 'Gallery', 'Tutorial', 'API', 'Site ▾', and 'Page ▾'. Below the navigation bar, the title 'seaborn.scatterplot' is displayed in a large, bold, dark font. Underneath the title, the function signature 'seaborn.scatterplot (x=None, y=None, hue=None, style=None, size=None, data=None, palette=None, hue\_order=None, hue\_norm=None, sizes=None, size\_order=None, size\_norm=None, markers=True, style\_order=None, x\_bins=None, y\_bins=None, units=None, estimator=None, ci=95, n\_boot=1000, alpha='auto', x\_jitter=None, y\_jitter=None, legend='brief', ax=None, \*\*kwargs)' is shown in a monospaced font. A red arrow points from the text 'Estos parámetros son: **hue**, **style** y **size**.' to the word 'hue' in the function signature.

```
seaborn.scatterplot (x=None, y=None, hue=None, style=None, size=None, data=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=True, style_order=None, x_bins=None, y_bins=None, units=None, estimator=None, ci=95, n_boot=1000, alpha='auto', x_jitter=None, y_jitter=None, legend='brief', ax=None, **kwargs)
```

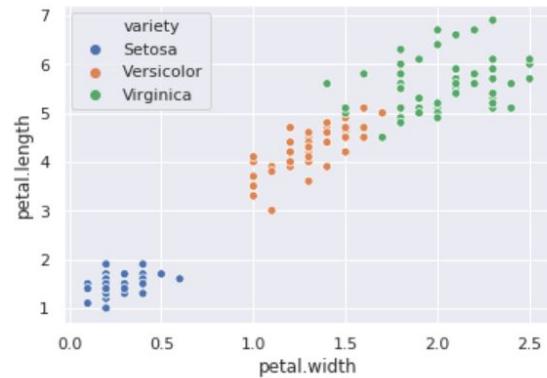
# Seaborn • Parámetro HUE

## HUE

El parámetro **hue** nos permite diferenciar nuestros datos según alguna **variables categórica** de nuestro dataset:

```
sns.scatterplot(x="petal.width", y="petal.length", hue="variety", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8c8b208>
```



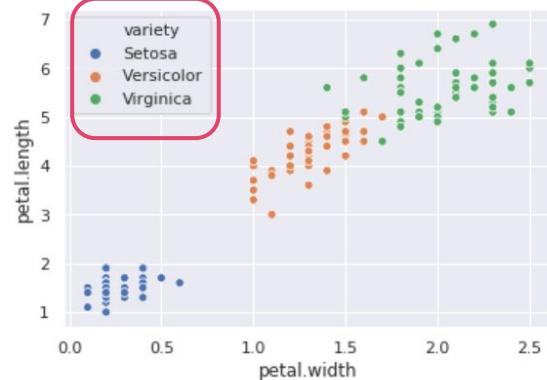
# Seaborn • Parámetro HUE

## HUE

El parámetro **hue** nos permite diferenciar nuestros datos según alguna **variables categórica** de nuestro dataset:

```
sns.scatterplot(x="petal.width", y="petal.length", hue="variety", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8c8b208>
```



	sepal.length	sepal.width	petal.length	petal.width	variety
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

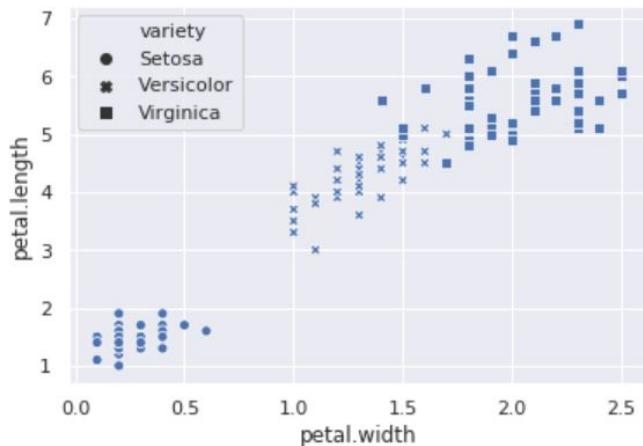
# Seaborn • Parámetro STYLE

## STYLE

El parámetro **style** también nos permite diferenciar nuestros datos según algunas **variables categóricas** de nuestro dataset.

```
sns.scatterplot(x="petal.width", y="petal.length", style="variety", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8cc39e8>
```



EN VEZ DE CAMBIAR EL  
COLOR, CAMBIA EL  
**ESTILO** DE LOS PUNTOS.

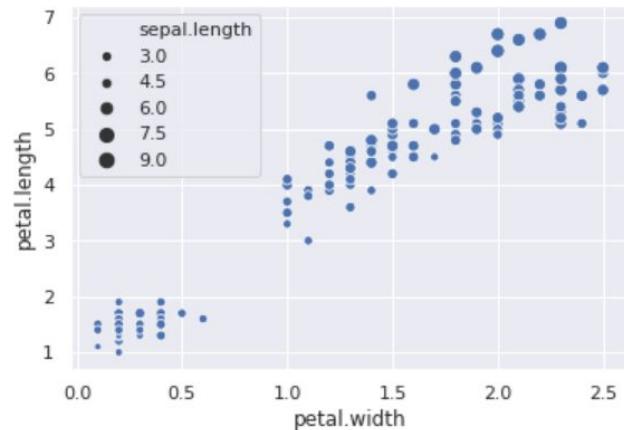
# Seaborn • Parámetro SIZE

## SIZE

El parámetro **size** también nos permite diferenciar nuestros datos según algunas **variables numéricas** (o categóricas) de nuestro dataset.

```
sns.scatterplot(x="petal.width", y="petal.length", size="sepal.length", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8f80780>
```

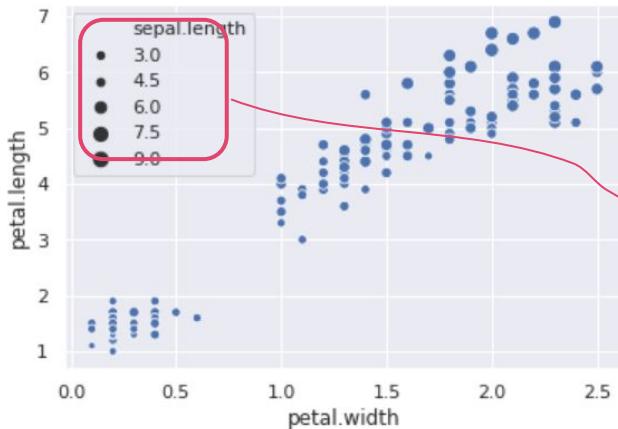


# Seaborn • Parámetro SIZE

## SIZE

El parámetro **size** también nos permite diferenciar nuestros datos según algunas **variables numéricas** (o categóricas) de nuestro dataset.

```
sns.scatterplot(x="petal.width", y="petal.length", size="sepal.length" data=iris_data)  
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8f80780>
```



	sepal.length	sepal.width	petal.length	petal.width	variety
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

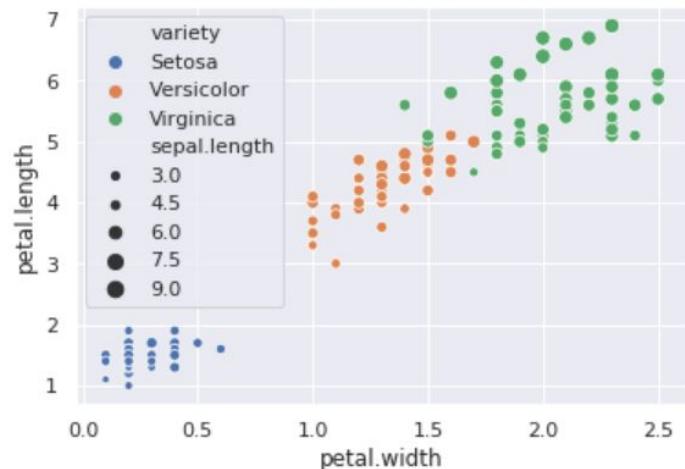
referencia de valor según  
tamaño

# Seaborn • Parámetros COMBINADOS

Podemos **combinar parámetros** y representar **varias características** en un único gráfico.

```
sns.scatterplot(x="petal.width", y="petal.length", size="sepal.length", hue="variety", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8fa0dd8>
```

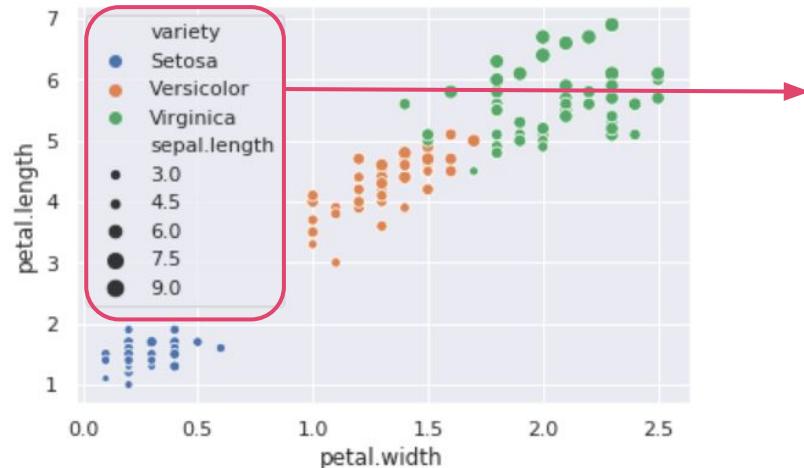


# Seaborn • Parámetros COMBINADOS

Podemos **combinar parámetros** y representar **varias características** en un único gráfico.

```
sns.scatterplot(x="petal.width", y="petal.length", size="sepal.length", hue="variety", data=iris_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a8fa0dd8>
```



Noten que la **leyenda** se genera automáticamente, y nos explicita los dos **features** que están representados en el gráfico pero no en sus ejes.

# Hands-on training



# DS\_Clase\_07\_Seaborn.ipynb

Scatter Plots





**;BREAK!**

---



# Visualización de Datos - Seaborn...

## ¡SEGUIMOS!

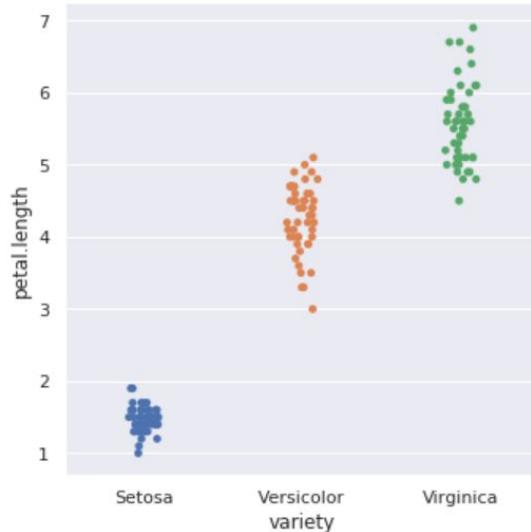


# Categorical plots

Como su nombre indica, esta función de Seaborn nos permite graficar utilizando una **variable categórica** como eje.

# Categorical plots

```
: sns.catplot(x="variety", y="petal.length", data=iris_data)
: <seaborn.axisgrid.FacetGrid at 0x7f48a8c941d0>
```



**Seaborn** realmente se **destaca** en este tipo de gráficos respecto a **Matplotlib**.

Noten que no “amontona” los puntos, permitiendo distinguir la cantidad.

# Categorical plots

## seaborn.catplot

```
seaborn.catplot(x=None, y=None, hue=None, data=None, row=None, col=None, col_wrap=None, estimator=<function mean>, ci=95,  
n_boot=1000, units=None, order=None, hue_order=None, row_order=None, col_order=None, kind='strip', height=5, aspect=1, orient=None,  
color=None, palette=None, legend=True, legend_out=True, sharex=True, sharey=True, margin_titles=False, facet_kws=None, **kwargs)
```

Categorical scatterplots:

- `stripplot()` (with `kind="strip"`; the default)
- `swarmplot()` (with `kind="swarm"`)

Categorical distribution plots:

- `boxplot()` (with `kind="box"`)
- `violinplot()` (with `kind="violin"`)
- `boxenplot()` (with `kind="boxen"`)

Categorical estimate plots:

- `pointplot()` (with `kind="point"`)
- `barplot()` (with `kind="bar"`)
- `countplot()` (with `kind="count"`)

# Categorical plots

## seaborn.catplot

```
seaborn.catplot(x=None, y=None, hue=None, data=None, row=None, col=None, col_wrap=None, estimator=<function mean>, ci=95,  
n_boot=1000, units=None, order=None, hue_order=None, row_order=None, col_order=None, kind='strip', height=5, aspect=1, orient=None,  
color=None, palette=None, legend=True, legend_out=True, sharex=True, sharey=True, margin_titles=False, facet_kws=None, **kwargs)
```

Categorical scatterplots:

- `stripplot()` (with `kind="strip"`; the default)
- `swarmplot()` (with `kind="swarm"`)

Categorical distribution plots:

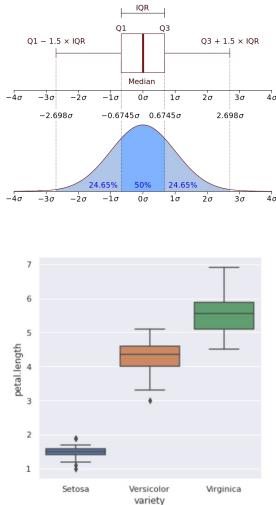
- `boxplot()` (with `kind="box"`)
- `violinplot()` (with `kind="violin"`)
- `boxenplot()` (with `kind="boxen"`)

Categorical estimate plots:

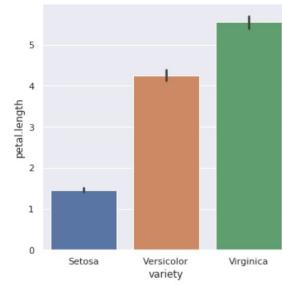
- `pointplot()` (with `kind="point"`)
- `barplot()` (with `kind="bar"`)
- `countplot()` (with `kind="count"`)

# Categorical plots

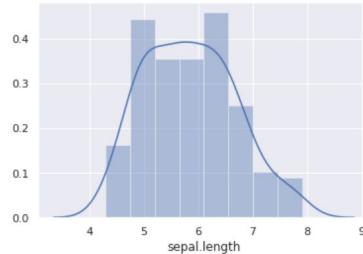
## Boxplot



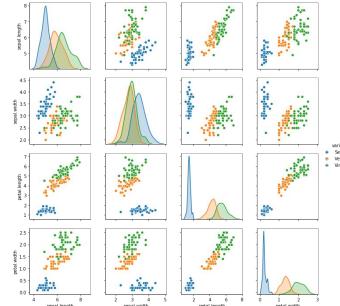
## Barplot



## Histogram



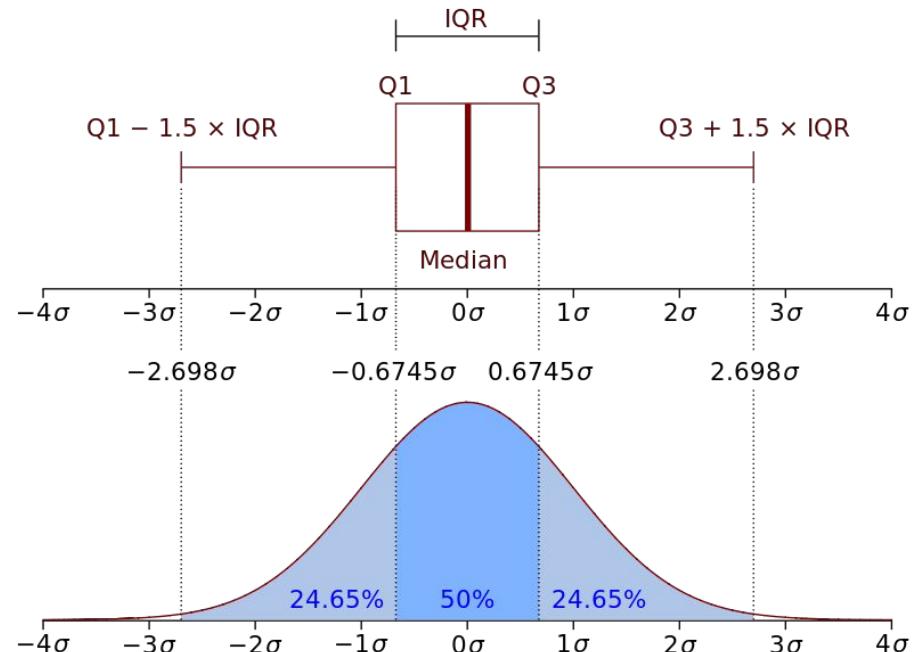
## Pair plots



# Categorical plots: Boxplot

El **diagrama de cajas** es una forma de visualizar un conjunto de valores.

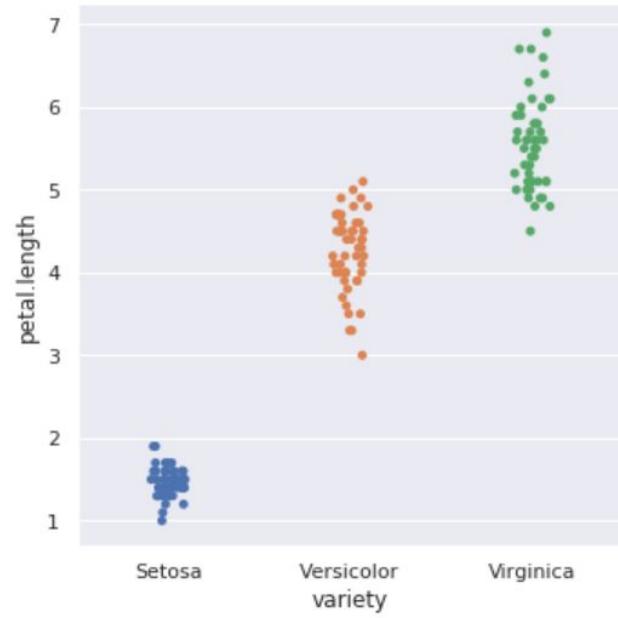
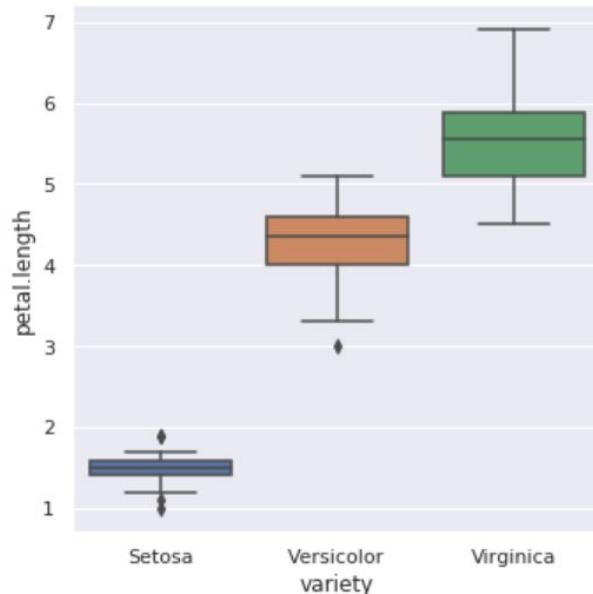
Muchas veces resulta más **informativa** que simplemente dibujar un punto por cada valor, ya que nos permite tener una idea de como es la distribución subyacente.



# Categorical plots: Boxplot

```
sns.catplot(x="variety", y="petal.length", kind='box', data=iris_data)
```

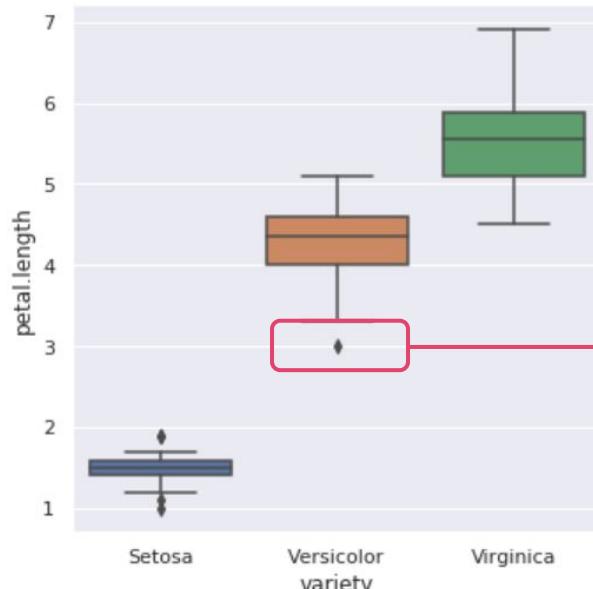
```
<seaborn.axisgrid.FacetGrid at 0x7f48a8b58e48>
```



# Categorical plots: Boxplot

```
sns.catplot(x="variety", y="petal.length", kind='box', data=iris_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f48a8b58e48>
```



Los “**whiskers**” indican el max y min valor de los datos ( hasta 1,5 veces el RIC =  $Q_3 - Q_1 = \text{len}(\text{box})$ ).

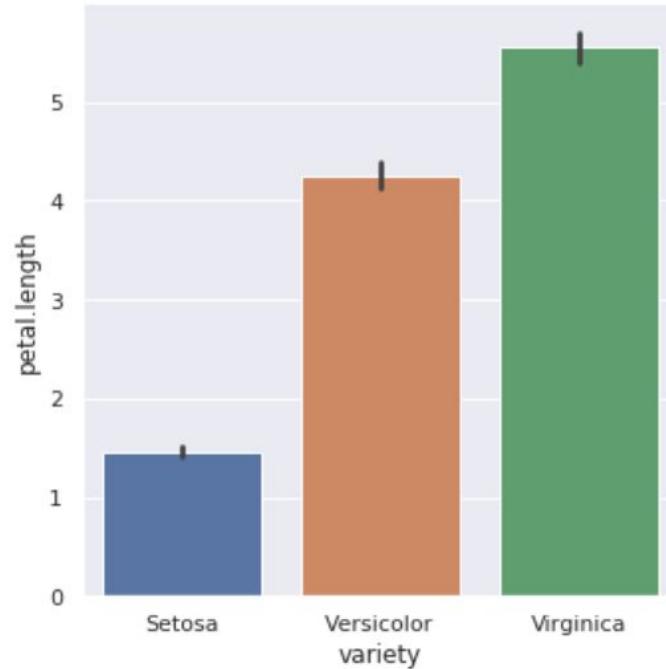
Los puntos fuera de los whiskers pueden considerarse **Outliers**.

valores extremadamente atípicos son aquellos que exceden:  $Q_1 - 3 \cdot \text{RIC}$  o  $Q_3 + 3 \cdot \text{RIC}$ .

# Categorical plots: Barplot

```
sns.catplot(x="variety", y="petal.length", kind='bar', data=iris_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f48a8e85c88>
```



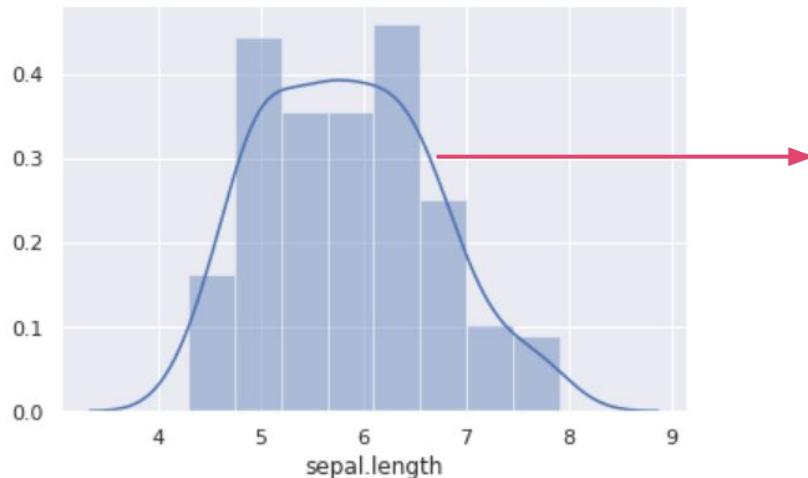
El **gráfico de barras** puede ser útil para ciertas circunstancias (por ej. al tener una única instancia), pero nos da menos información que el Boxplot.

# Histogramas

**Seaborn** también tiene la opción de graficar automáticamente **histogramas**, para hacerlo utiliza la función **hist** de **Matplotlib**.

```
sns.distplot(iris_data['sepal.length'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48a89d5e80>
```



Por **default**, la función nos incluye una curva que intenta aproximar la distribución de la que vienen los datos. Su nombre es **KDE** ([kernel density estimation](#)).

# Histogramas

## seaborn.distplot

```
seaborn.distplot (a, bins=None, hist=True, kde=True, rug=False, fit=None, hist_kws=None, kde_kws=None, rug_kws=None,  
fit_kws=None, color=None, vertical=False, norm_hist=False, xlabel=None, label=None, ax=None)
```

Flexibly plot a univariate distribution of observations.

This function combines the matplotlib `hist` function (with automatic calculation of a good default bin size) with the seaborn `kdeplot()` and `rugplot()` functions. It can also fit `scipy.stats` distributions and plot the estimated PDF over the data.

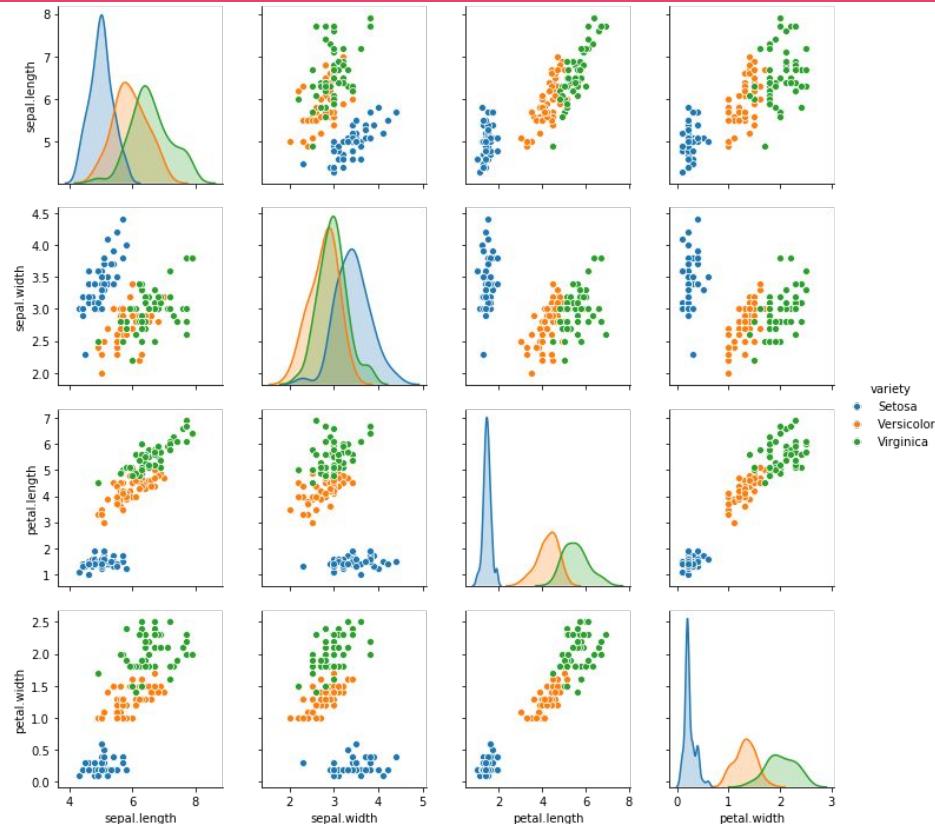
# Pair Plots

**Pairplot** es una función de Seaborn que nos va a resultar **muy útil** a la hora de explorar los features de un dataset.

Compara todas las variables numéricas del dataset entre sí (gráficos scatter 2D). Al mismo tiempo nos permite colorear (**hue**) según una variable categórica.

# Pair Plots

```
sns.pairplot(data=iris_data, hue="variety");
```



# DS\_Clase\_07\_Seaborn.ipynb

Categorical Plots, Histogramas y Pair plots



# Buenas prácticas de un data scientist



Un buen Análisis Exploratorio de Datos no pierde de vista las particularidades del problema en el que estamos trabajando.<sup>1</sup>

<sup>1</sup> Es decir, depende del problema.

# ABC del Análisis Exploratorio de Datos

1. Fijarse tipos de datos y qué valores toman.
2. Métricas básicas del dataset
3. Graficar la distribución de los datos (histograma de cada columna, etc.).
4. Explorar valores faltantes y outliers<sup>2</sup>
5. Graficar features vs. features. Explorar correlaciones.

<sup>2</sup> Aún no lo vimos, pero pronto.

## Recursos



### 1. EDA:

- <https://medium.com/@swethalakshmanan14/exploratory-data-analysis-in-python-ebdf643a33f6>
- <https://towardsdatascience.com/exploratory-data-analytics-in-python-c9a77dfa39ce>

2. **An Introduction to Seaborn:** la documentación oficial de Seaborn ofrece un muy buena introducción a la librería, así como también un tutorial muy completo y accesible

3. **Capítulo 5, “Visualization With Matplotlib”, de Python Data Science Handbook.** Ese capítulo también tiene una entrada sobre Seaborn.

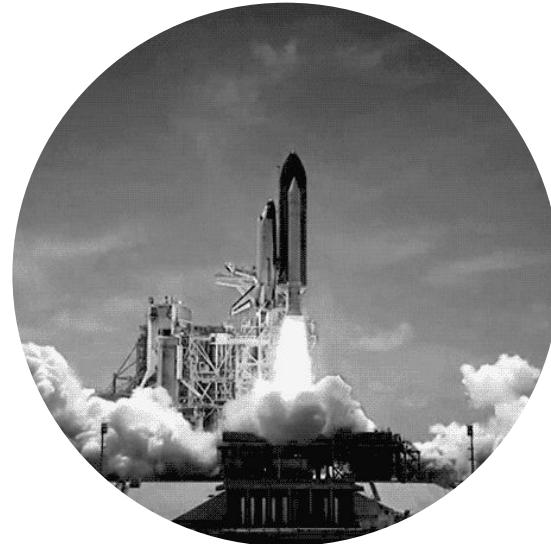
4. **10 tips to improve your plotting:** una nota interesante con consejos generales sobre visualización en el ámbito de Data Science.

# Lanzamiento Entrega 01

---

¿Alguna duda con:

1. Notebook
1. Datos
1. Checklist?



# Para la próxima

---

1. ¡Clase integradora! También nos pondremos al día si estamos atrasados con alguna explicación.
2. Avanzar con los notebooks que tengan atrasados y traer dudas
3. Si pueden, trabajar en la Entrega 01

ACÁMICA