# Ecosistema Python: Pandas, Matplotlib – Seaborn

Programación 2023-II

UNIVERSIDAD EIA®

Lamentablemente, el nombre Pandas no tiene nada que ver con este lindo animal




El nombre se deriva del término **PAN**el **DA**ta, un término econométrico para datos que incluyen observaciones durante múltiples períodos de tiempo.

# Pandas

- Pandas, al igual que NumPy, es una de las bibliotecas de Python más populares para el análisis de datos
- Es una abstracción de alto nivel de NumPy (bajo nivel), que está escrita en C puro
- Pandas proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar
- Hay dos estructuras principales utilizadas por pandas: *data frames* y *series*

# Estructura *series*

- Una serie pandas es similar a una lista, pero se diferencia en que una serie asocia una etiqueta a cada elemento. Esto lo hace parecer un diccionario
- Si el usuario no proporciona explícitamente un índice, pandas crea un **`RangeIndex`** que va de 0 a N – 1
- Cada objeto de la serie también tiene un tipo de datos

```
import pandas as pd
new_series = pd.Series([5, 6, 7, 8, 8, 10])
print(new_series)
```

# Acceso a los datos

- Los valores de una serie se pueden extraer todos, así como individualmente

```
print(serie_1.values)
print('====================')
print(serie_1[4])
```

- Los índices también se pueden personalizar

```
serie_2 = pd.Series([5, 6, 7, 8, 9, 10], index = ['a', 'b', 'c', 'd', 'e', 'f'])
print(serie_2.values)
print('====================')
print(serie_2['f'])
```

# Filtrado y operaciones matemáticas

```
serie_3 = serie_2[serie_2>6]
print(serie_3.values)
print('====================')
serie_3 = serie_3**2 + 2
print(serie_3)
```

# Estructura data frame

- De manera simple, un *data frame* es una tabla
- Cada columna de un *data frame* es un objeto serie
- Las filas constan de elementos dentro de series

| Case ID | Variable 1 | Variable 2 | Variable 3 |
|---|---|---|---|
| 1 | 123 | ABC | 10 |
| 2 | 456 | DEF | 20 |
| 3 | 789 | XYZ | 30 |

# Creación de *data frames*

Un *data frame* se puede crear a partir de diccionarios

```
df_1 = pd.DataFrame({
    'País':['Kasajistán','Rusia','Bielorrusia','Ucrania'],
    'Población':[17.04, 143.5, 9.5, 45.5],
    'Superficie':[2724902, 17125191, 207600, 603628]
})
print(df_1)
```

O a partir de listas

```
lista = [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
df_2 = pd.DataFrame(lista)
print(df_2)
print('====================')
df_2.columns = ['V1', 'V2', 'V3']
print(df_2)
```

Un objeto *data frame* de Pandas como dos índices:

- un índice de columna
- un índice de fila

```
print(df_1.columns)
print('====================')
print(df_1.index)
```

Nuevamente, si no proporciona uno, Pandas creará un **`RangeIndex`** de 0 a N – 1

# Hay diferentes formas de indicar el índice de fila

```
df_3 = pd.DataFrame({
    'País':['Kasajistán','Rusia','Bielorrusia','Ucrania'],
    'Población':[17.04, 143.5, 9.5, 45.5],
    'Superficie':[2724902, 17125191, 207600, 603628]},
    index = ['KZ', 'RU', 'KY', 'UA'])
print(df_3)
```

```
df_4 = pd.DataFrame({
    'País':['Kasajistán','Rusia','Bielorrusia','Ucrania'],
    'Población':[17.04, 143.5, 9.5, 45.5],
    'Superficie':[2724902, 17125191, 207600, 603628]})
print(df_4)
print('====================')
df_4.index = ['KZ', 'RU', 'KY', 'UA']
df_4.index.name = 'Código de país'
print(df_4)
```

El acceso a las filas mediante el índice se puede realizar de varias formas

Primero, usando `.loc()` y proporcionar una etiqueta de índice

```
print(df_4.loc['KZ'])
```

Segundo, usando `.iloc()` y proporcionar un índice numérico

```
print(df_4.iloc[0])
```

# Filtrado

El filtrado se realiza mediante las matrices booleanas

```
print(df_4[df_4.Población>10][['País', 'Superficie']])
```

# Eliminación de columnas

Una columna se puede eliminar usando la función `.drop()`

```
import numpy as np
df_5 = pd.DataFrame(np.arange(12).reshape(3, 4),
                    columns=['A', 'B', 'C', 'D'])
print(df_5)
df_5 = df_5.drop(['B', 'C'], axis=1)
print('====================')
print(df_5)
```

# Lectura y escritura de archivos

Pandas admite muchos formatos de archivos populares, incluidos **CSV**, XML, HTML, Excel, SQL, JSON, etc.

Puede leer los datos de un archivo CSV utilizando la función `read_csv()` y escribir el archivo la función `to_csv()`

```
df_6 = pd.read_csv('MonthlyData.csv', sep =';')
df_6.head()
```

```
df_5.to_csv('prueba.csv')
```

Pandas tiene la capacidad de hacer mucho más de lo que hemos cubierto aquí, como agrupar datos e incluso visualizarlos

Sin embargo, al igual que con NumPy, no tenemos tiempo suficiente para cubrir todos los aspectos de Pandas aquí

# Análisis exploratorio de datos

Explorar los datos es un paso crucial en el análisis de datos

Implica:

- Organizar el conjunto de datos
- Graficar el conjunto de datos.
- Aplicar estadística descriptiva: medidas de centralidad y dispersión

*"El análisis exploratorio de datos nunca puede ser la historia completa, pero nada más puede servir como piedra fundamental".*

***- John Tukey***

# Descarga de la base de datos

https://github.com/aquinteroz/Programacion/blob/main/Files/pokemon.csv

# Lectura de los datos en Python

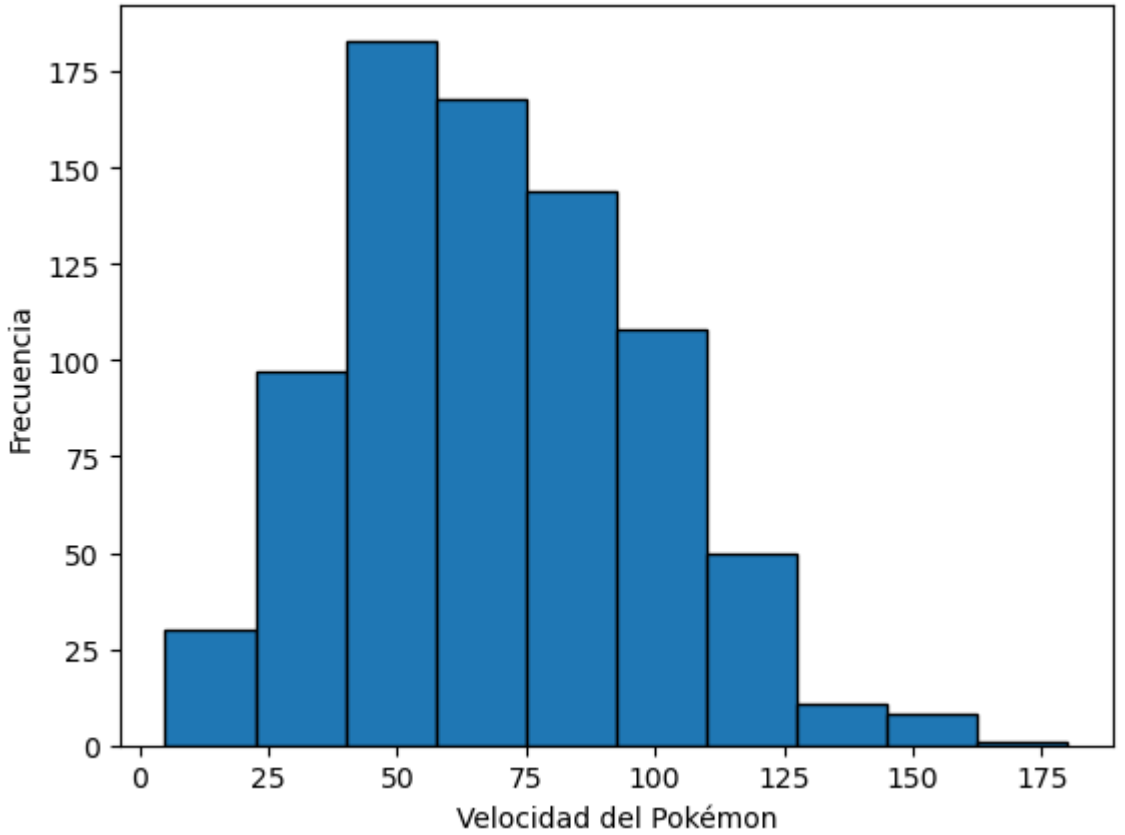
```
import pandas as pd
import numpy as nop
from matplotlib import pyplot as plt
import seaborn as sns

data1 = pd.read_csv('pokemon.csv')
data1.head()
```

Estadísticas descriptivas: `data1.describe()`

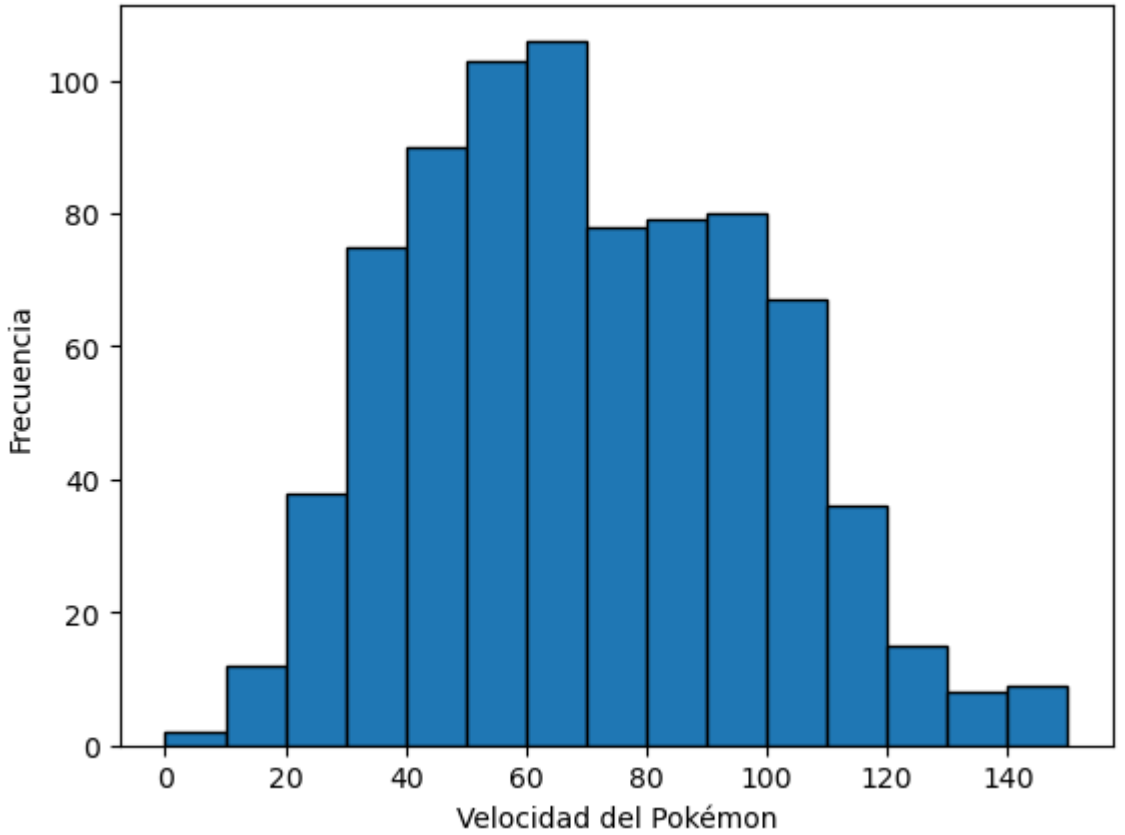
# Gráfica de histograma

```
plt.hist(data1['Speed'],
         histtype = 'bar',
         ec = 'black')
plt.xlabel('Velocidad del Pokémon')
plt.ylabel('Frecuencia')
plt.show()
```



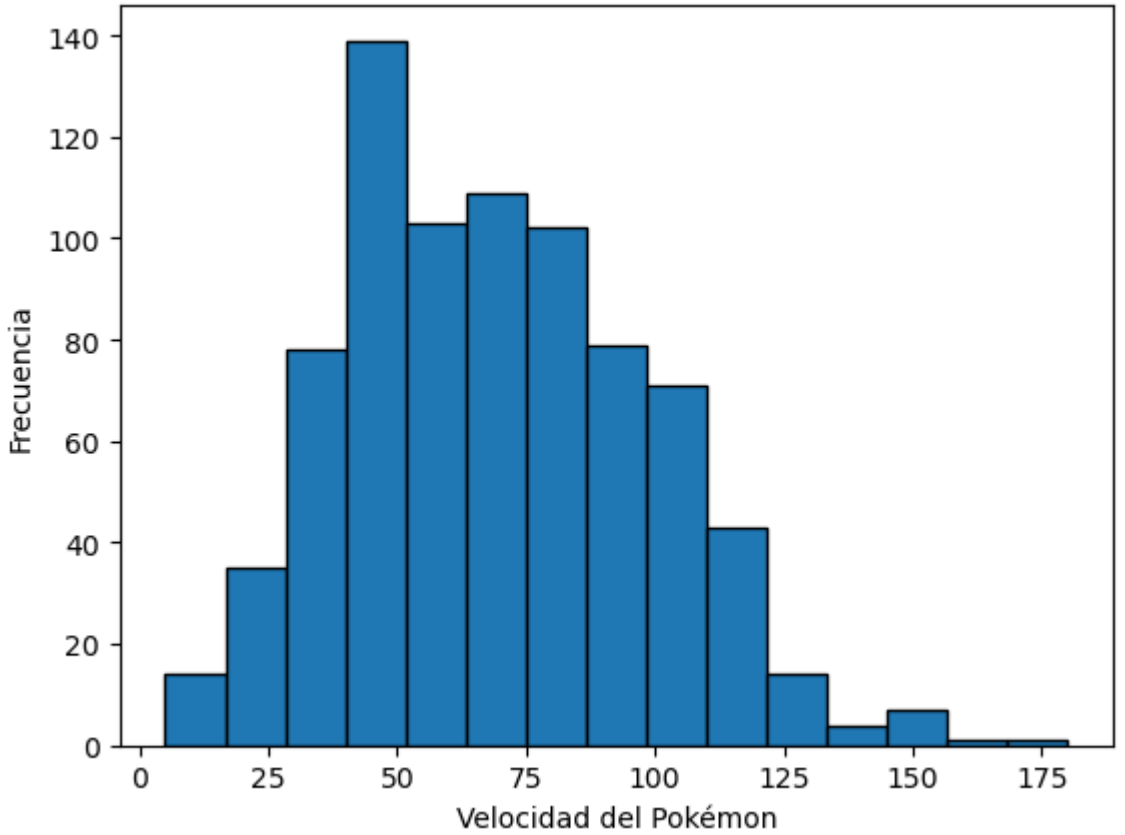
# Especificación de bins

```
plt.hist(data1['Speed'],
         histtype = 'bar',
         ec = 'black',
         bins = list(range(0,160,10)))
plt.xlabel('Velocidad del Pokémon')
plt.ylabel('Frecuencia')
plt.show()
```



# Especificación de bins

```
plt.hist(data1['Speed'],
         histtype = 'bar',
         ec = 'black',
         bins = 15)
plt.xlabel('Velocidad del Pokémon')
plt.ylabel('Frecuencia')
plt.show()
```



# Seaborn

Matplotlib es una biblioteca de Python poderosa, pero a veces difícil de manejar

Seaborn proporciona una interfaz de alto nivel para Matplotlib y facilita la producción de gráficos como el de la derecha

# Ventajas de Seaborn

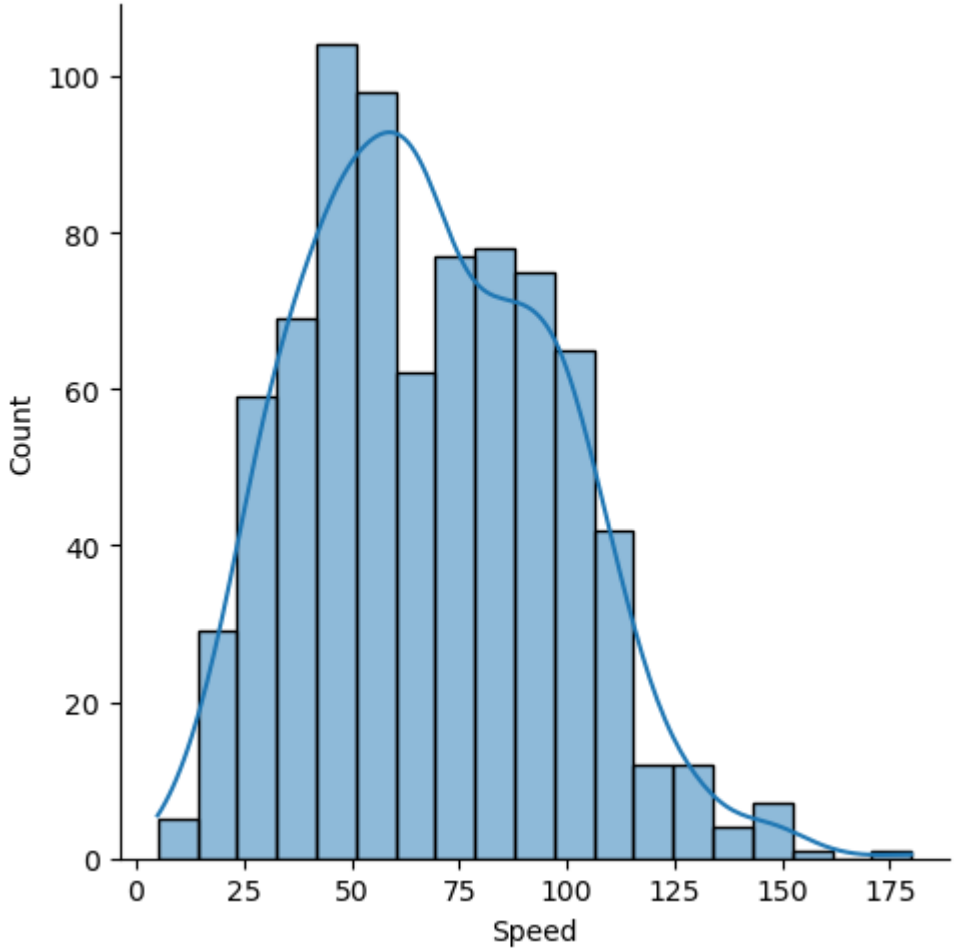
Seaborn ofrece:

- Uso de temas predeterminados que sean estéticamente agradables
- Configuración de paletas de colores personalizadas
- Gráficas estadísticas atractivas
- Visualización de distribuciones de forma fácil y flexible
- Visualización de información a partir de matrices y DataFrames.

Los últimos tres puntos han llevado a Seaborn a convertirse en la herramienta de análisis de datos exploratorios elegida por muchos usuarios de Python

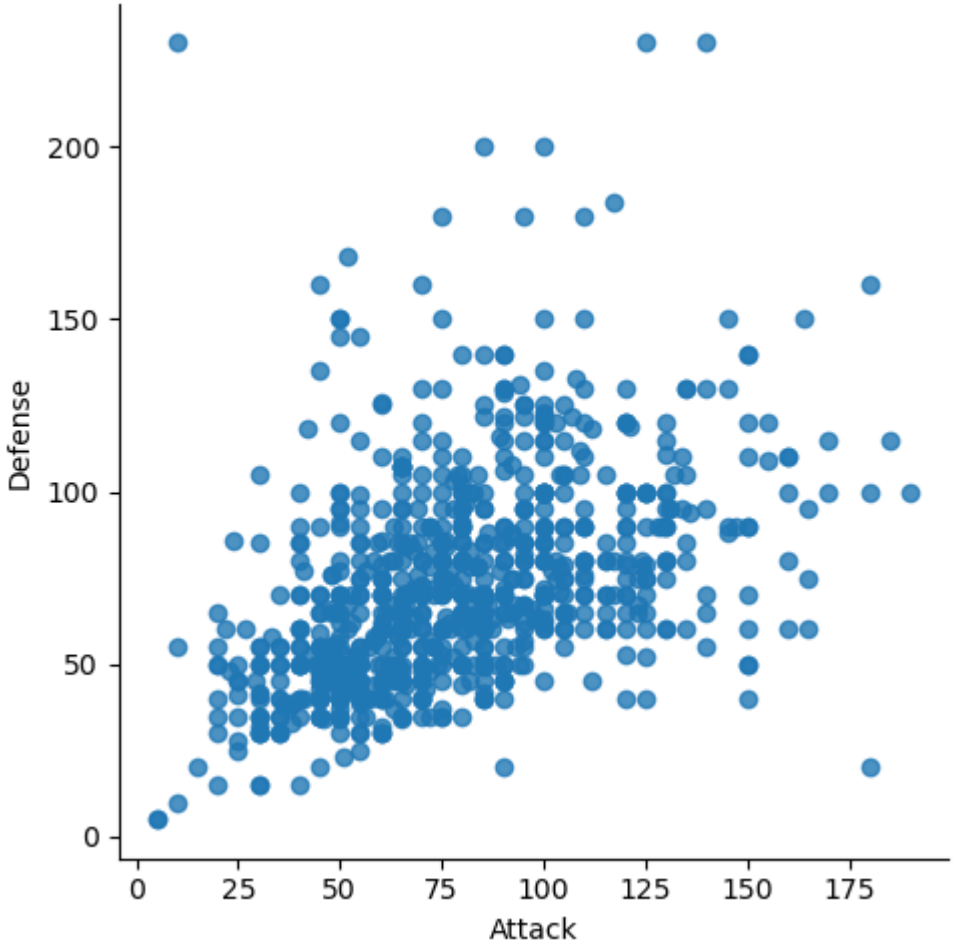
# Histograma con Seaborn

```
sns.set_style()
sns.displot(data = data1.Speed,
            kde = True)
```



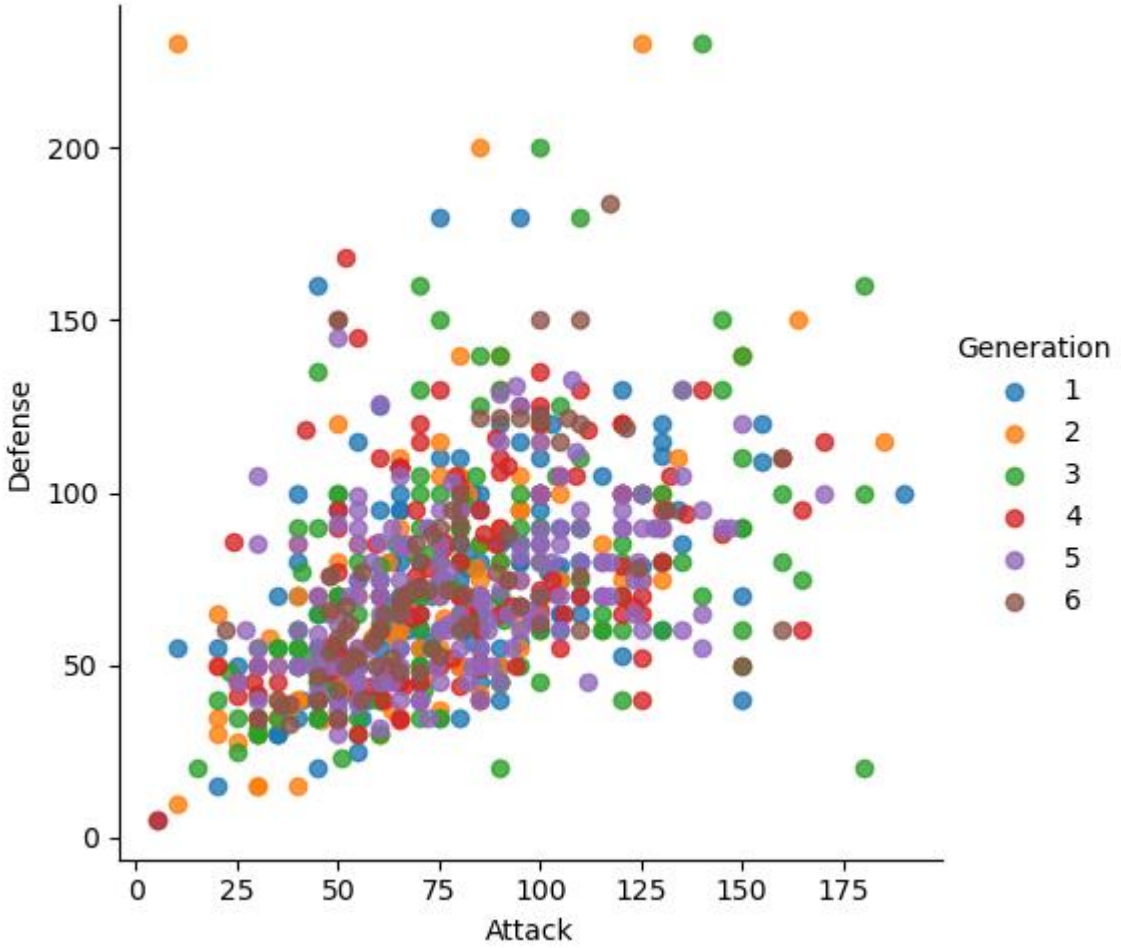
# Scatter plot

```
sns.lmplot(x = 'Attack',
           y = 'Defense',
           data = data1,
           fit_reg = False)
```



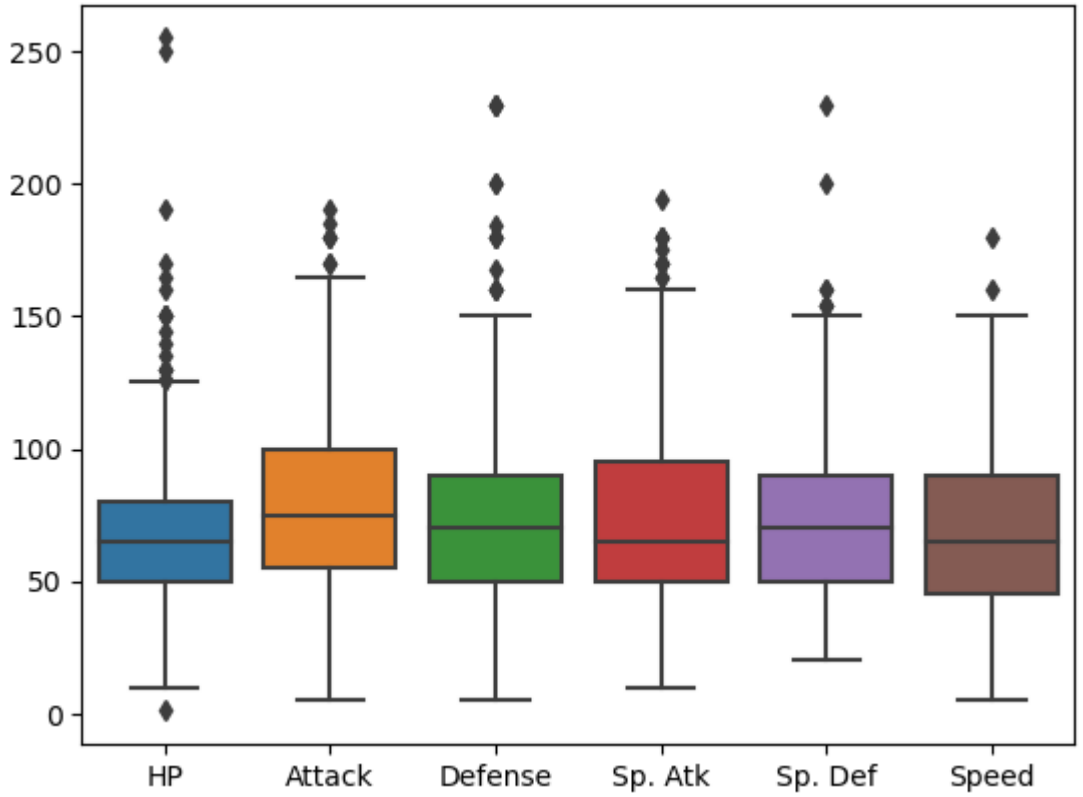
# *Scatterplot* por categorías

```
sns.lmplot(x = 'Attack',
           y = 'Defense',
           data = data1,
           hue = 'Generation',
           fit_reg = False)
```



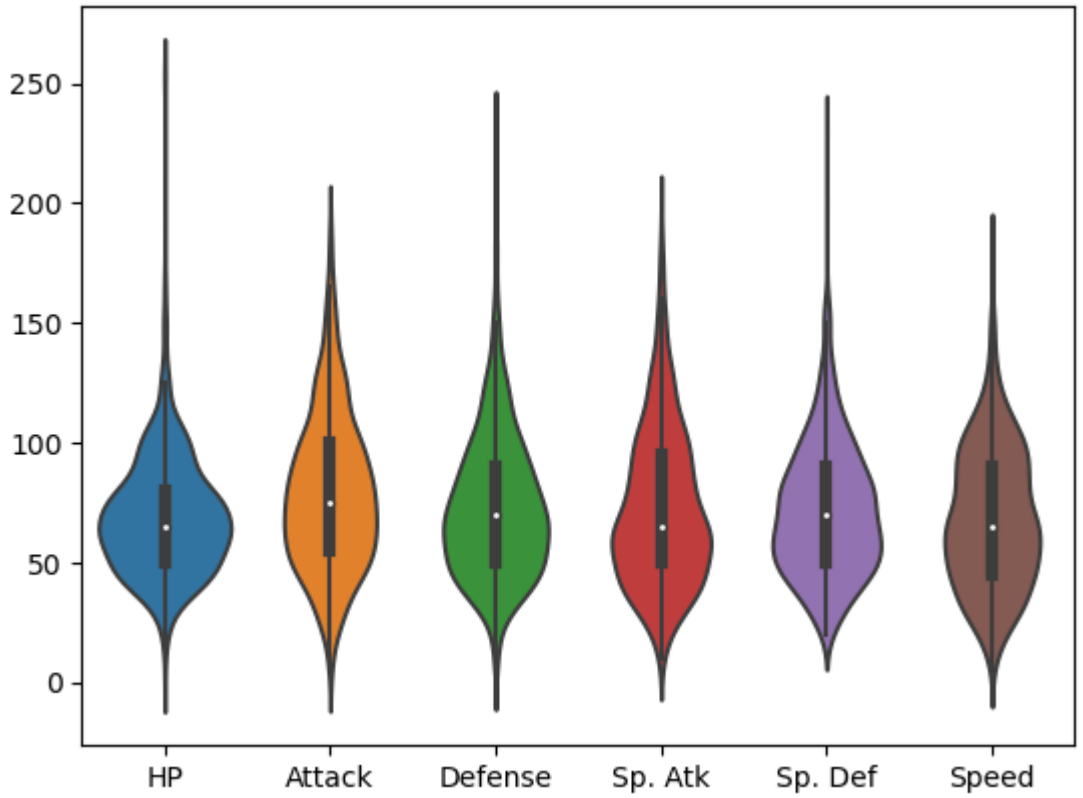
# Box plot

```
sns.boxplot(data = data1[['HP', 'Attack',
                          'Defense', 'Sp. Atk',
                          'Sp. Def', 'Speed']])
```



# Gráfico de violín

```
sns.violinplot(data = data1[['HP', 'Attack',
                             'Defense', 'Sp. Atk',
                             'Sp. Def', 'Speed']])
```



# Varias gráficas en un eje

```
plt.figure(figsize = (15,6))
sns.violinplot(x = 'Type 1',
               y = 'Attack',
              data =data1,
              inner = None,
              palette = 'Set2')
sns.swarmplot(x = 'Type 1',
              y = 'Attack',
              data =data1,
              color = 'k',
              alpha = 0.7)
```