

# Introducción

## Markdown

- Una celda "markdown" corresponde a una celda de texto, en este espacio usted puede escribir notas o ecuaciones en entorno ".tex" utilizando el símbolo \$.
- Es importante que en cada sección utilice títulos adecuados para cuando se imprima el documento quede ordenado y separado por secciones.

**negrito** *cursiva*

## Libreria Numpy

Esta libreria permite realizar algunas operaciones matemáticas de aritmética básica y álgebra. En el contexto de las aplicaciones de estadística y matemática se destaca:

- Operaciones con números racionales.
- Creación de matrices y vectores.
- Partición de un intervalo cerrado o lista de números aleatorios.

In [ ]:

```
import numpy
numpy.e
numpy.pi
```

Out[ ]: 3.141592653589793

```
import numpy as np
```

```
np.e
```

Out[ ]: 2.718281828459045

```
np.pi
```

Out[ ]: 3.141592653589793

```
np.sqrt(5)
```

Out[ ]: 2.23606797749979

```
e=np.e
e
```

Out[ ]: 2.718281828459045

```
np.log(e)
```

```
Out[ ]: 1.0
```

```
In [ ]: x=[1,2,3,4,5,6,7,8,9]    # lista de números
x
```

```
Out[ ]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [ ]: x=np.linspace(1,10,100)
x
```

```
Out[ ]: array([ 1.          ,  1.09090909,  1.18181818,  1.27272727,  1.36363636,
                1.45454545,  1.54545455,  1.63636364,  1.72727273,  1.81818182,
                1.90909091,  2.          ,  2.09090909,  2.18181818,  2.27272727,
                2.36363636,  2.45454545,  2.54545455,  2.63636364,  2.72727273,
                2.81818182,  2.90909091,  3.          ,  3.09090909,  3.18181818,
                3.27272727,  3.36363636,  3.45454545,  3.54545455,  3.63636364,
                3.72727273,  3.81818182,  3.90909091,  4.          ,  4.09090909,
                4.18181818,  4.27272727,  4.36363636,  4.45454545,  4.54545455,
                4.63636364,  4.72727273,  4.81818182,  4.90909091,  5.          ,
                5.09090909,  5.18181818,  5.27272727,  5.36363636,  5.45454545,
                5.54545455,  5.63636364,  5.72727273,  5.81818182,  5.90909091,
                6.          ,  6.09090909,  6.18181818,  6.27272727,  6.36363636,
                6.45454545,  6.54545455,  6.63636364,  6.72727273,  6.81818182,
                6.90909091,  7.          ,  7.09090909,  7.18181818,  7.27272727,
                7.36363636,  7.45454545,  7.54545455,  7.63636364,  7.72727273,
                7.81818182,  7.90909091,  8.          ,  8.09090909,  8.18181818,
                8.27272727,  8.36363636,  8.45454545,  8.54545455,  8.63636364,
                8.72727273,  8.81818182,  8.90909091,  9.          ,  9.09090909,
                9.18181818,  9.27272727,  9.36363636,  9.45454545,  9.54545455,
                9.63636364,  9.72727273,  9.81818182,  9.90909091, 10.          ])
```

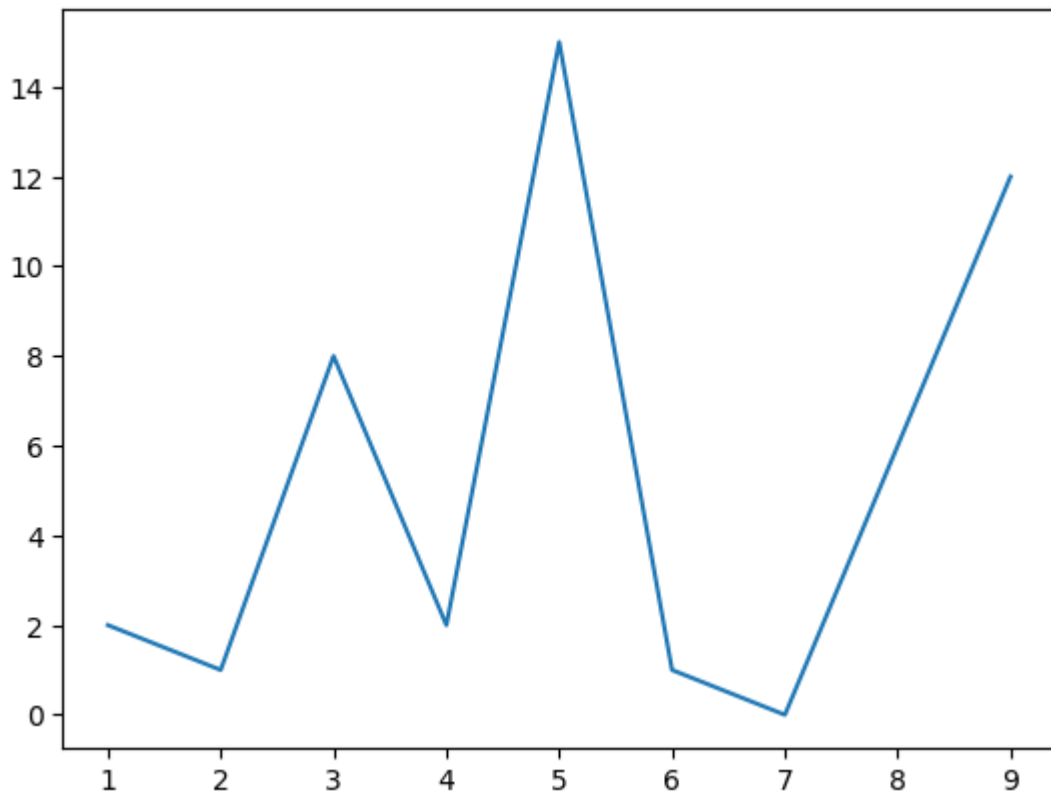
## Libreria Matplotlib

La libreria matplotlib es la columna vertebral de las gráficas en python. Con esta libreria podremos visualizar curvas en el espacio R2 o R3, y desde el contexto estadístico nos permite realizar: - Gráficos de cajas. - Gráficos de torta. - Histogramas.

- Nubes de puntos o scatterplot. - Mapas de correlaciones. - Gráficos de violin. Véamos la construcción de un gráfico paso a paso:

```
In [ ]: import matplotlib.pyplot as plt
```

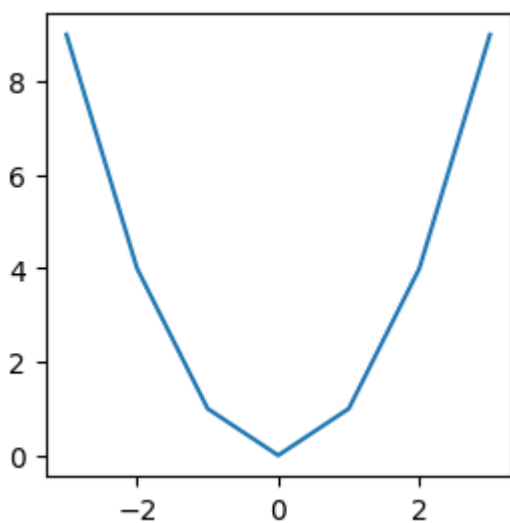
```
In [ ]: # definiendo las variables x e y o dominio y recorrido.
x=[1,2,3,4,5,6,7,8,9] # variable x
y=[2,1,8,2,15,1,0,6,12] # variable y
plt.plot(x,y) #x: variable de entrada, y: variable de salida.
plt.show()
#obs: El código que aparece antes del gráfico se puede eliminar agregando la ext
```



## Buscando una función cuadrática

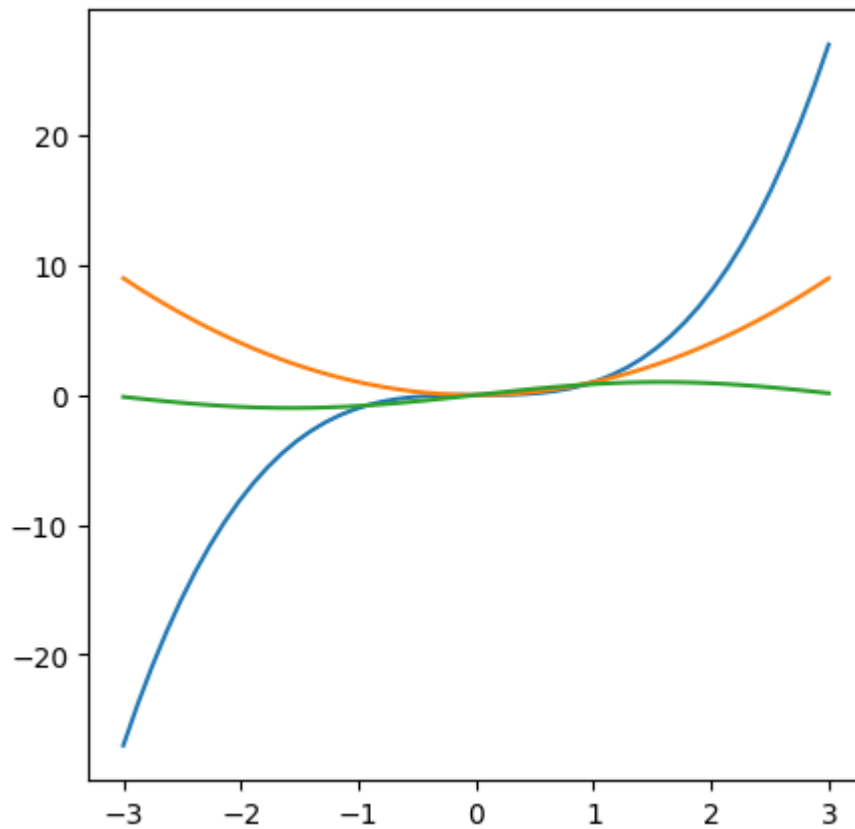
Utilizando el entorno array, logramos crear una lista de números con un orden específico. De esta forma podemos relacionar dicha lista con una nueva variable y crear una relación de dependencia con el objetivo de gráficar.

```
In [ ]: x=np.array([-3,-2,-1,0,1,2,3])
plt.figure(figsize=(3,3))
plt.plot(x,x**2)
plt.show()
```

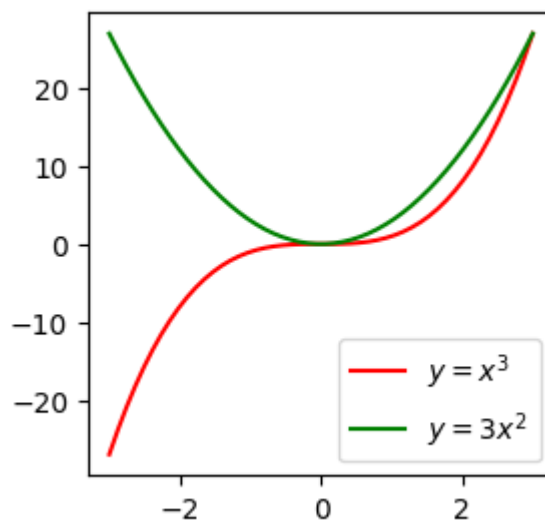


En estos casos, es mejor utilizar la función "linspace" dado que de forma automática se crea un arreglo de números los cuales se pueden utilizar para crear una relación más fina.

```
In [ ]: X=np.linspace(-3,3,50)
plt.figure(figsize=(5,5))
plt.plot(X,X**3)
plt.plot(X,X**2)
plt.plot(X,np.sin(X))
plt.show()
```

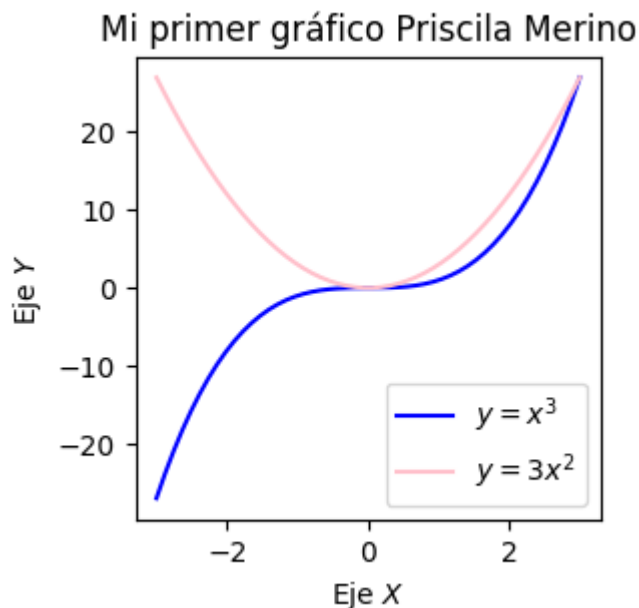


```
In [ ]: plt.figure(figsize=(3,3))
plt.plot(X,X**3,color="red",label="$y=x^3$")
plt.plot(X,3*X**2,color="green",label="$y=3x^2$")
plt.legend()
plt.show()
```



```
In [ ]: plt.figure(figsize=(3,3))
plt.plot(X,X**3,color="blue",label="$y=x^3$")
plt.plot(X,3*X**2,color="pink",label="$y=3x^2$")
```

```
plt.xlabel("Eje $X$")
plt.ylabel("Eje $Y$")
plt.title("Mi primer gráfico Priscila Merino")
plt.legend()
plt.show()
```



## Librería Pandas

Pandas es la libreria que permite analizar y procesar datos. Utilizando esta libreria se pueden realizar las siguientes acciones:

- Procesar bases de datos.
- Filtrar datos.
- Realizar informes descriptivos.
- Agrupar datos y crear nuevas variables.
- entre otros.

```
In [ ]: import pandas as pd
```

Para Colab y Jupyter notebook , es distinto la carga de datos En Colab se carga cada vez en el icono a la izquierda carpeta En Jupyter debo tener el archivo en la misma carpeta donde está el notebook

data=pd.read\_csv("ejemplo1.csv", sep=";", decimal=",") imprimir cabecera de los datos --> las cinco primeras filas

- data.head() # para ver la cabecera de los datos
- data.info() # para saber que tipo de datos tiene
- data.tail() # para ver los últimos elementos
- data.head(10) mostrar los 10 primeros elementos
- data.describe() #resumen estadístico

```
In [ ]: data=pd.read_csv("ejemplo1.csv",encoding='latin-1',sep=";",decimal=",")
```

```
In [ ]: data.head()
```

```
Out[ ]:
```

	<b>Mortalidad_Infantil</b>	<b>Tasa_Alfabetismo_Femenino</b>	<b>PIB_per_cápita</b>	<b>Tasa_fecundidad</b>
<b>0</b>	128	37	1870	6.66
<b>1</b>	204	22	130	6.15
<b>2</b>	202	16	310	7.00
<b>3</b>	197	65	570	7.25
<b>4</b>	96	76	2050	3.81

```
In [ ]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 64 entries, 0 to 63  
Data columns (total 4 columns):  
#   Column                                Non-Null Count  Dtype  
---  -  
0   Mortalidad_Infantil                  64 non-null     int64  
1   Tasa_Alfabetismo_Femenino           64 non-null     int64  
2   PIB_per_cápita                      64 non-null     int64  
3   Tasa_fecundidad                     64 non-null     float64  
dtypes: float64(1), int64(3)  
memory usage: 2.1 KB
```

```
In [ ]: data.tail()
```

```
Out[ ]:
```

	<b>Mortalidad_Infantil</b>	<b>Tasa_Alfabetismo_Femenino</b>	<b>PIB_per_cápita</b>	<b>Tasa_fecundidad</b>
<b>59</b>	115	62	1470	3.89
<b>60</b>	186	45	300	6.90
<b>61</b>	47	85	3630	4.10
<b>62</b>	178	45	220	6.09
<b>63</b>	142	67	560	7.20

```
In [ ]: data.head(10)
```

Out[ ]:	Mortalidad_Infantil	Tasa_Alfabetismo_Femenino	PIB_per_cápita	Tasa_fecundidad
0	128	37	1870	6.66
1	204	22	130	6.15
2	202	16	310	7.00
3	197	65	570	7.25
4	96	76	2050	3.81
5	209	26	200	6.44
6	170	45	670	6.19
7	240	29	300	5.89
8	241	11	120	5.89
9	55	55	290	2.36

```
In [ ]: data.describe() #resumen estadístico
```

Out[ ]:	Mortalidad_Infantil	Tasa_Alfabetismo_Femenino	PIB_per_cápita	Tasa_fecundidad
count	64.000000	64.000000	64.000000	64.000000
mean	139.921875	51.187500	1399.687500	5.565469
std	77.260291	26.007859	2726.156064	1.521400
min	12.000000	9.000000	120.000000	1.690000
25%	78.500000	29.000000	300.000000	4.615000
50%	138.500000	48.000000	575.000000	6.040000
75%	192.500000	77.250000	1317.500000	6.660000
max	312.000000	95.000000	19830.000000	8.490000

## Observación:

para realizar alguna modificación del archivo , filtros  
columnas adicionales ,etc

realizar una copia para no perder los datos originales


```
In [ ]: data2=data.copy()
```

```
In [ ]: data2["nueva"]=data2["Mortalidad_Infantil"]+data2["Tasa_Alfabetismo_Femenino"]
```

```
In [ ]: data2.head()
```

Out[ ]:

	Mortalidad_Infantil	Tasa_Alfabetismo_Femenino	PIB_per_cápita	Tasa_fecundidad	nu
0	128	37	1870	6.66	·
1	204	22	130	6.15	·
2	202	16	310	7.00	·
3	197	65	570	7.25	·
4	96	76	2050	3.81	·



In [ ]: *# De La data anterior dejar solo las columnas 1 y 5.*  
data3=data2[["Mortalidad\_Infantil","nueva"]]  
data3.head(15)

Out[ ]:

	Mortalidad_Infantil	nueva
0	128	165
1	204	226
2	202	218
3	197	262
4	96	172
5	209	235
6	170	215
7	240	269
8	241	252
9	55	110
10	75	162
11	129	184
12	24	117
13	165	196
14	94	171

In [ ]: data4=data3[data3["nueva"]>150] *# filtro*  
data4.info()  
data4.describe()

```
<class 'pandas.core.frame.DataFrame'>
Index: 49 entries, 0 to 63
Data columns (total 2 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Mortalidad_Infantil    49 non-null    int64
1   nueva                  49 non-null    int64
dtypes: int64(2)
memory usage: 1.1 KB
```



Out[ ]:

	Mortalidad_Infantil	nueva
<b>count</b>	49.000000	49.000000
<b>mean</b>	168.918367	214.020408
<b>std</b>	63.083224	45.524943
<b>min</b>	67.000000	152.000000
<b>25%</b>	121.000000	176.000000
<b>50%</b>	165.000000	204.000000
<b>75%</b>	209.000000	236.000000
<b>max</b>	312.000000	333.000000

```
In [ ]: import xlrd
renta=pd.read_excel("renta1.xlsx", 0) # archivo en excel
renta
```

Out[ ]:

	renta	año	población	estudiantes	ingreso
<b>0</b>	197	1	75211	15303	11537
<b>1</b>	342	0	77759	18017	19568
<b>2</b>	323	1	106743	22462	19841
<b>3</b>	496	0	141865	29769	31885
<b>4</b>	216	1	36608	11847	11455
<b>...</b>	<b>...</b>	<b>...</b>	<b>...</b>	<b>...</b>	<b>...</b>
<b>123</b>	352	0	56856	10640	24735
<b>124</b>	220	1	48347	9051	13458
<b>125</b>	344	0	51003	9961	21947
<b>126</b>	243	1	170616	37475	16510
<b>127</b>	472	0	191262	44601	29420

128 rows × 5 columns

## Libreria Seaborn

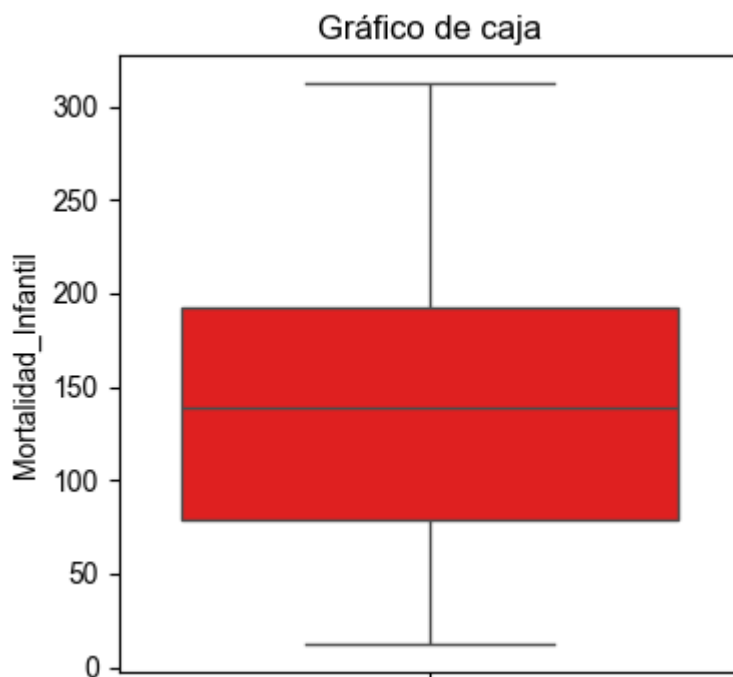
Esta libreria permite aplicar gráficos estadísticos de una forma simple, es un atajo de la libreria matplotlib por lo que se puede ahorrar mucho tiempo en visualizar características de las bases de datos.

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

```
In [ ]: data["Mortalidad_Infantil"].describe()
```

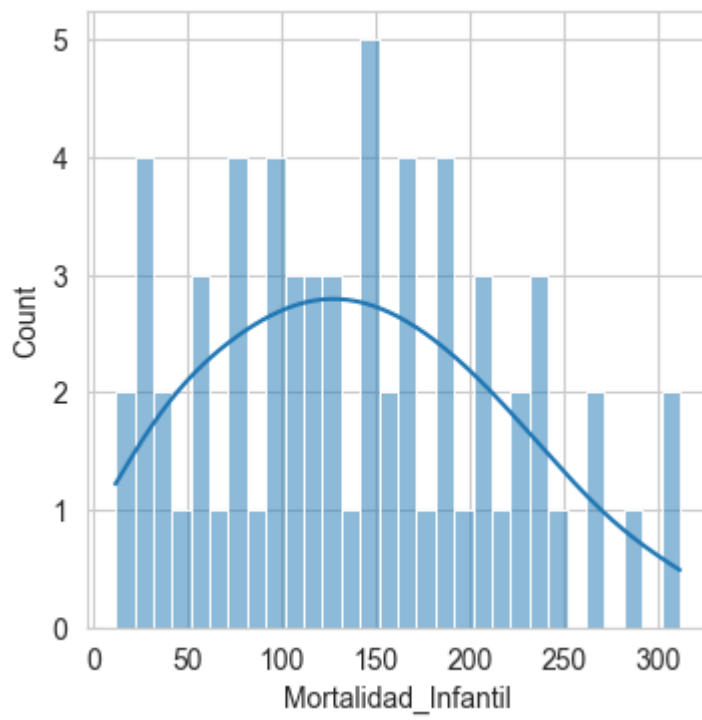
```
Out[ ]: count      64.000000
        mean      139.921875
        std       77.260291
        min       12.000000
        25%       78.500000
        50%      138.500000
        75%      192.500000
        max      312.000000
        Name: Mortalidad_Infantil, dtype: float64
```

```
In [ ]: #para realizar boxplot
        plt.figure(figsize=(4,4))
        sns.boxplot(data=data,y="Mortalidad_Infantil",color="red")
        plt.title("Gráfico de caja")
        sns.set_style("whitegrid")
        plt.show()
```



```
In [ ]: sns.set_style("whitegrid")
```

```
In [ ]: plt.figure(figsize=(4,4))
        sns.histplot(data=data,x="Mortalidad_Infantil",bins=30,kde=True)
        plt.savefig("histograma.png")
        plt.show()
```



```
In [ ]: plt.figure(figsize=(8,8))
sns.boxplot(data=data,y="Mortalidad_Infantil",color="blue")
plt.title("Gráfico de caja Mortalidad infantil")
plt.savefig("boxplotMI.png")
plt.show()
```

Gráfico de caja Mortalidad infantil

