

Python

Breve historia

- Creado por Guido Van Rossum en 1991.
- Es sucesor del lenguaje ABC
- Su nombre esta inspirado en el programa de televisión de la BBC “Monty Python Flying Circus”

¿Qué es?

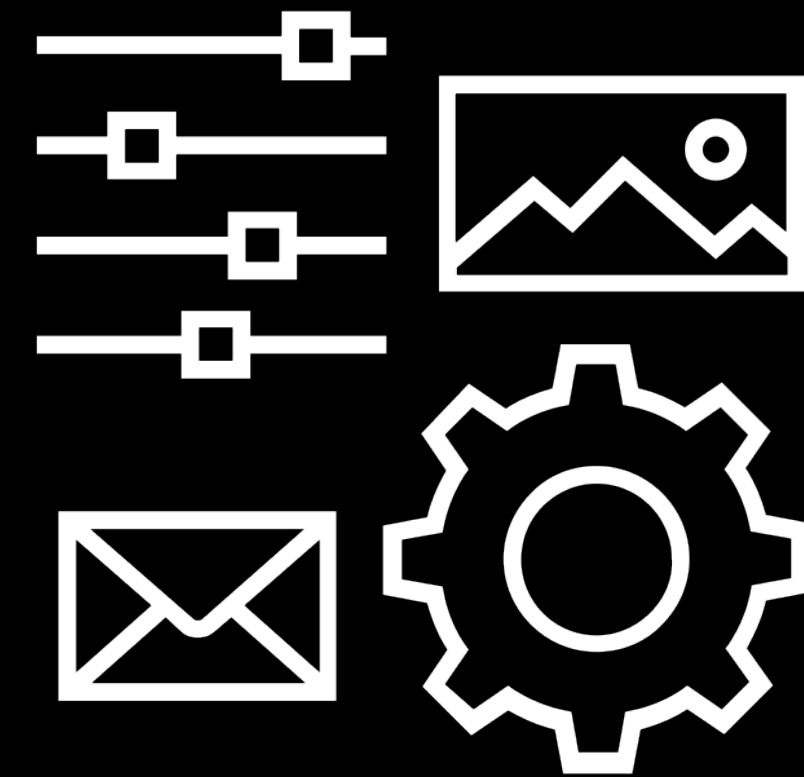
- Interpretado
- Multiparadigma
- Tipado dinámico
- Sintaxis de código legible
- Open source
- Multiplataforma

¿Porqué aprenderlo?

- Curva de aprendizaje baja; esto lo vamos a ver desde el primer ejercicio de este taller. Una sola línea de código nos da el resultado directamente
- Sintaxis legible (simple)
- Utilidad en muchos entornos de trabajo. Servidor, páginas web, aplicaciones desktop, etc; agricultura, algoritmos de estimación de datos perdidos, reconstrucción de series de datos, etc, etc

¿Big Data?

Es un término para conjuntos de datos que son tan grandes o complejos que el software de aplicación de procesamiento de datos tradicional es inadecuado para tratar con ellos.



¿Aplicación?



Detección de fraudes en
tiempo real



Análisis de
sentimientos del
consumidor



Gestión de inteligencia del tráfico

¿Características?

- Volumen: demasiados datos para manejar fácilmente
- Velocidad: la velocidad de entrada y salida de datos dificulta su análisis
- Variedad: el rango y el tipo de fuentes de datos son demasiado grandes para asimilar

Inicio del taller

Quiz

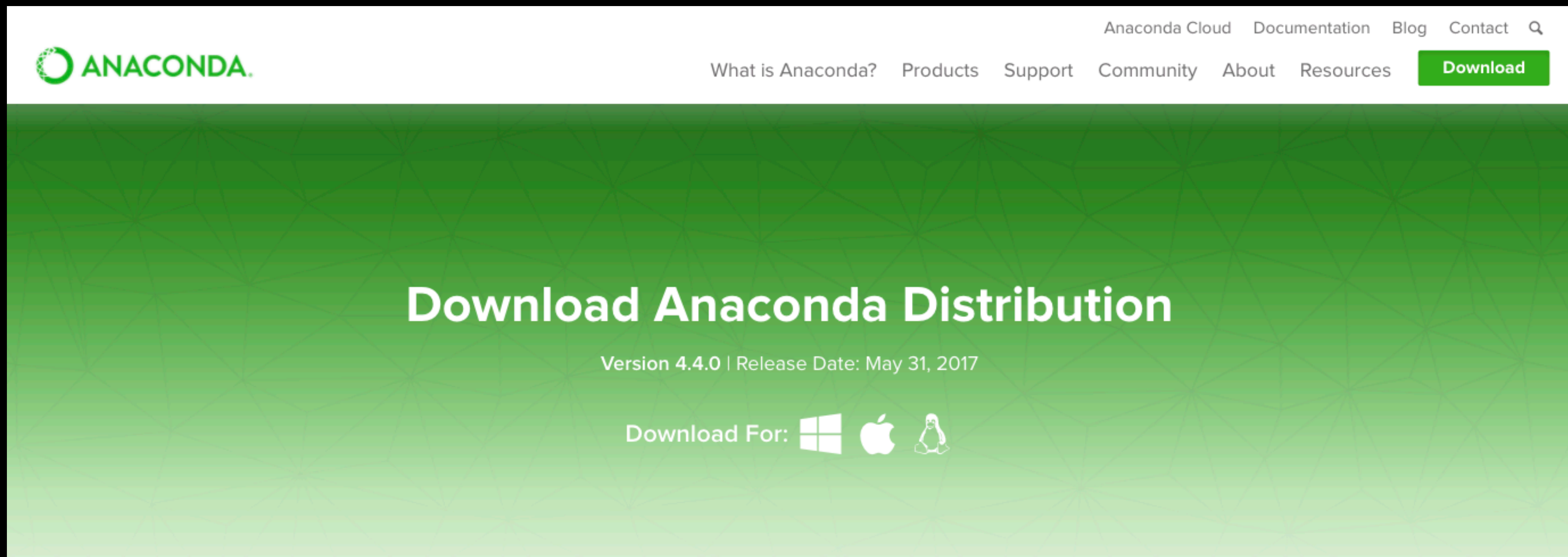
Ejercicio

Para el siguiente ejercicio vamos a ocupar la base de datos que se encuentra en la carpeta LINK con el nombre de `db_rain.csv` la cual contiene los siguientes campos:

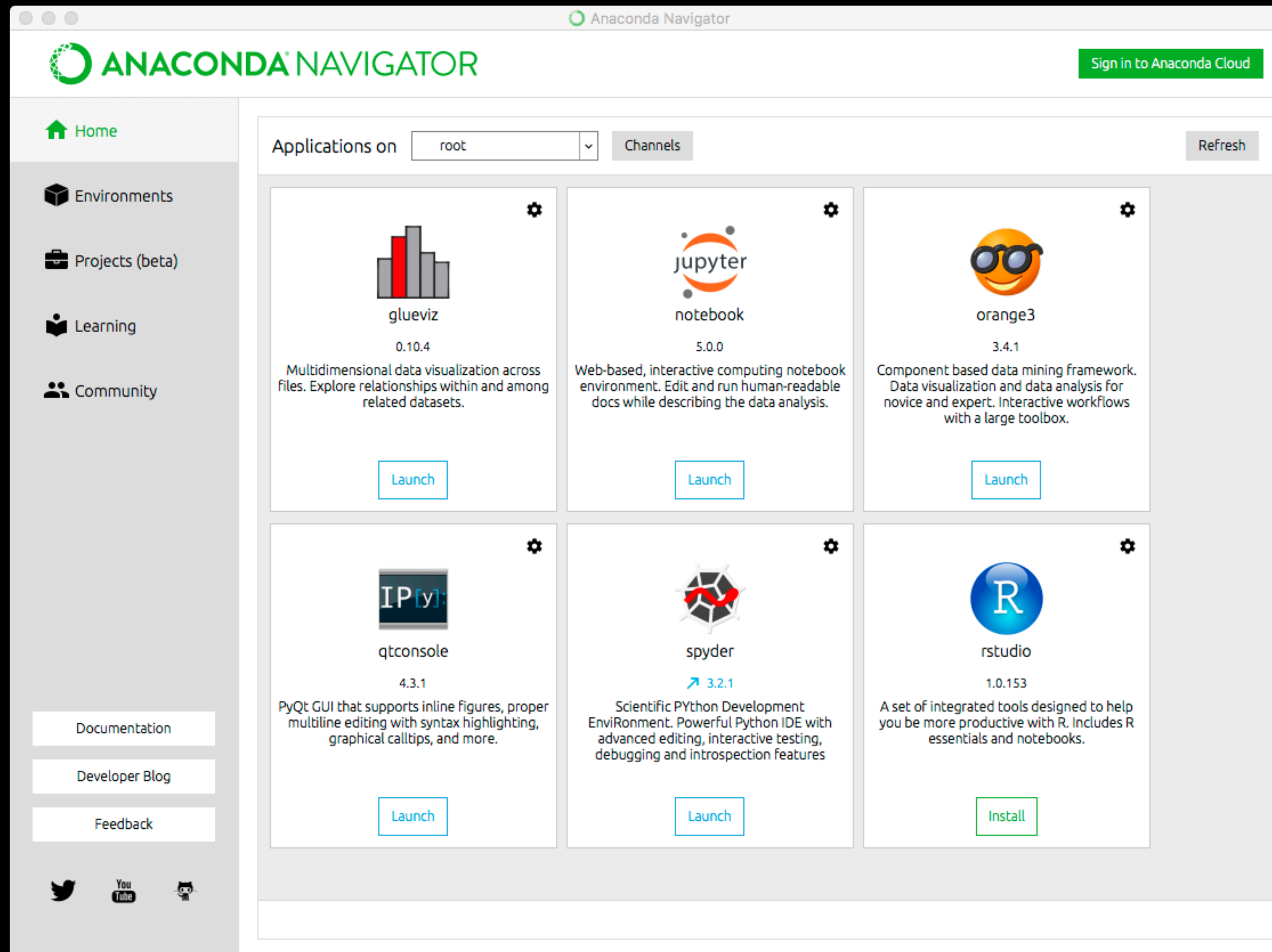
- * name : Nombre de la estación agrometeorológica
 - * number : Número de la estación agrometeorológica
 - * lat : Latitud de la estación agrometeorológica
 - * long : Longitud de la estación agrometeorológica
 - * rain : Precipitación en mm
 - * date : Fecha en que se registro el dato
- **Tienes 10 minutos para resolver las siguientes preguntas****
- * Número total de estaciones en la db?
 - * Tabla y gráfica de precipitación acumulada por años?
 - * Precipitación acumulada en la base de datos

Instalación Anaconda

- www.anaconda.com/downloads
- Windows, macOS, Linux
- Windows-macOS: asistente, clic, clic, clic, Finalizar
- Linux: descargar el archivo .sh, ejecutarlo.

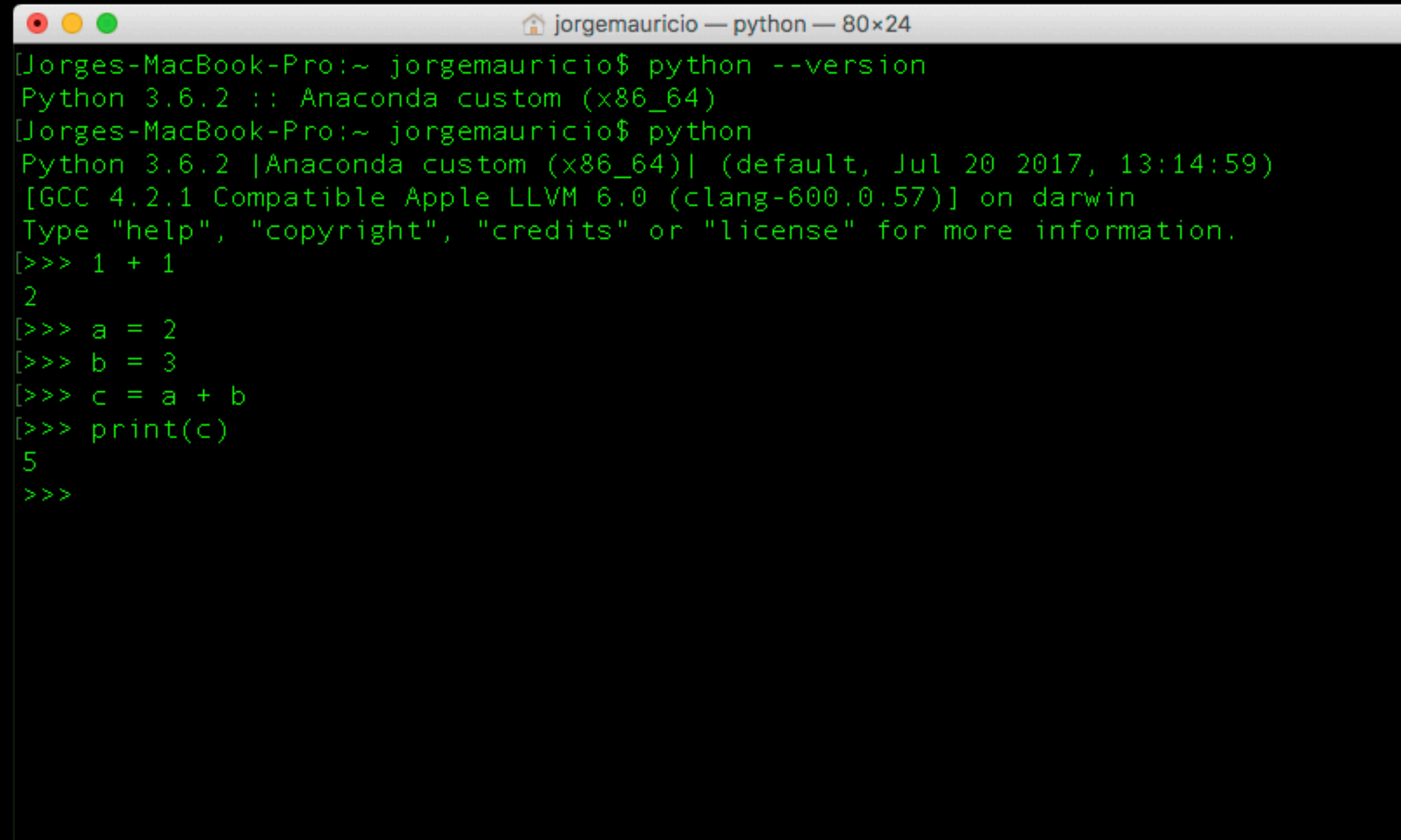


Anaconda Navigator



Modo Interactivo

- Para emplear el modo interactivo, se debe de ingresar el comando **python** en la terminal de comandos. De ahí en adelante las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa.

A screenshot of a macOS terminal window titled "jorgemaurocio — python — 80x24". The terminal shows the execution of the Python command line interface. The user enters "python --version" and "python". The output shows Python 3.6.2 running on a macOS machine. The user then enters several lines of Python code in the interactive shell: "1 + 1", "a = 2", "b = 3", "c = a + b", and "print(c)". The results of these operations are displayed: "2", and "5".

```
jorgemaurocio — python — 80x24
[Jorges-MacBook-Pro:~ jorgemaurocio$ python --version
Python 3.6.2 :: Anaconda custom (x86_64)
[Jorges-MacBook-Pro:~ jorgemaurocio$ python
Python 3.6.2 [Anaconda custom (x86_64)] (default, Jul 20 2017, 13:14:59)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> a = 2
>>> b = 3
>>> c = a + b
>>> print(c)
5
>>>
```

Modo IDE

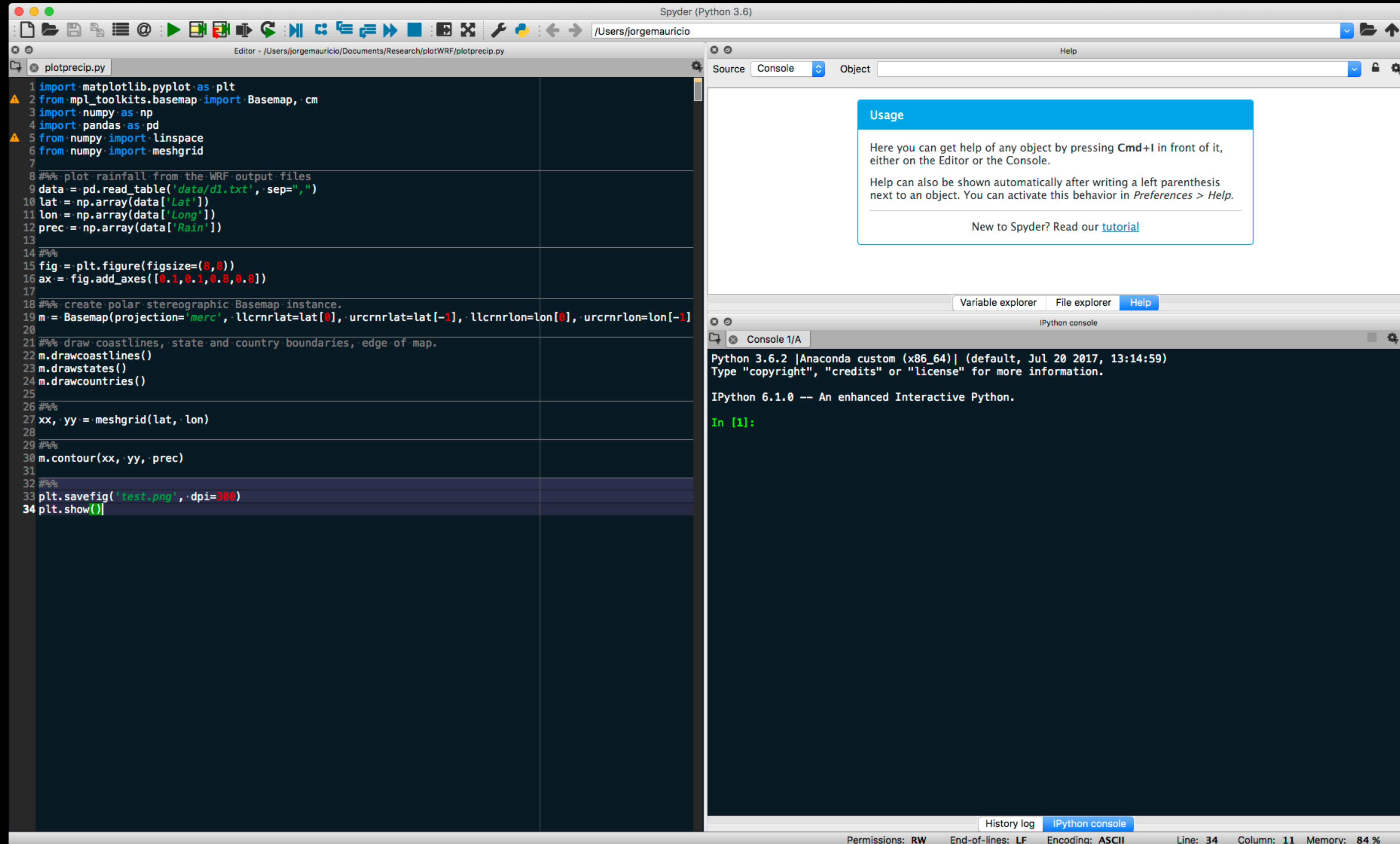
- Existen varios IDE (Interface Development Enviroment), los más populares son:
 - Spyder
 - Jupyter



Spyder

- Potente entorno de desarrollo interactivo para el lenguaje Python con funciones avanzadas de edición, pruebas interactivas, depuración e introspección.
- Entorno de computación numérica gracias al soporte de IPython (intérprete interactivo mejorado de Python) y bibliotecas populares de Python como:
 - NumPy (álgebra lineal)
 - SciPy (procesamiento de señales e imágenes)
 - Matplotlib (trazado interactivo 2D / 3D)

IDE



Editor

The image shows the Spyder Python IDE interface. The main window is divided into three panes: a code editor on the left, a help panel on the top right, and an IPython console on the bottom right.

Code Editor: The file `plotprecip.py` is open. The code is as follows:

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.basemap import Basemap, cm
3 import numpy as np
4 import pandas as pd
5 from numpy import linspace
6 from numpy import meshgrid
7
8 """ plot rainfall from the WRF output files
9 data = pd.read_table('data/d1.txt', sep=",")
10 lat = np.array(data['Lat'])
11 lon = np.array(data['Long'])
12 prec = np.array(data['Rain'])
13
14 """
15 fig = plt.figure(figsize=(8,8))
16 ax = fig.add_axes([0.1,0.1,0.8,0.8])
17
18 """ create polar stereographic Basemap instance.
19 m = Basemap(projection='merc', llcrnrlat=lat[0], urcrnrlat=lat[-1], llcrnrlon=lon[0], urcrnrlon=lon[-1])
20
21 """ draw coastlines, state and country boundaries, edge of map.
22 m.drawcoastlines()
23 m.drawstates()
24 m.drawcountries()
25
26 """
27 xx, yy = meshgrid(lat, lon)
28
29 """
30 m.contour(xx, yy, prec)
31
32 """
33 plt.savefig('test.png', dpi=300)
34 plt.show()
```

Help Panel: The 'Usage' tab is selected. It contains the following text:

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

[New to Spyder? Read our tutorial](#)

IPython Console: The console shows the following output:

```
Python 3.6.2 |Anaconda custom (x86_64)| (default, Jul 20 2017, 13:14:59)
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: LF, Encoding: ASCII, Line: 34, Column: 11, Memory: 84 %.

Visualizador de archivos

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.6)". The left pane contains a code editor with the file "plotprecip.py" open. The code is as follows:

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.basemap import Basemap, cm
3 import numpy as np
4 import pandas as pd
5 from numpy import linspace
6 from numpy import meshgrid
7
8 """ plot rainfall from the WRF output files
9 data = pd.read_table('data/d1.txt', sep=",")
10 lat = np.array(data['Lat'])
11 lon = np.array(data['Long'])
12 prec = np.array(data['Rain'])
13
14 """
15 fig = plt.figure(figsize=(8,8))
16 ax = fig.add_axes([0.1,0.1,0.8,0.8])
17
18 """ create polar stereographic Basemap instance.
19 m = Basemap(projection='merc', llcrnrlat=lat[0], urcrnrlat=lat[-1], llcrnrlon=lon[0], urcrnrlon=lon[-1])
20
21 """ draw coastlines, state and country boundaries, edge of map.
22 m.drawcoastlines()
23 m.drawstates()
24 m.drawcountries()
25
26 """
27 xx, yy = meshgrid(lat, lon)
28
29 """
30 m.contour(xx, yy, prec)
31
32 """
33 plt.savefig('test.png', dpi=300)
34 plt.show()
```

The right pane is divided into two sections. The top section is titled "Help" and contains a "Usage" box with the following text:

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

The bottom section of the right pane is titled "IPython console" and contains the following text:

```
Python 3.6.2 [Anaconda custom (x86_64)] (default, Jul 20 2017, 13:14:59)
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:
```

The bottom status bar of the IDE shows the following information:

- Permissions: RW
- End-of-lines: LF
- Encoding: ASCII
- Line: 34
- Column: 11
- Memory: 84 %

Consola

The image shows the Spyder Python IDE interface. The main window is divided into three panes: a code editor on the left, a help panel on the top right, and an IPython console on the bottom right.

Code Editor: The file `plotprecip.py` is open. The code is as follows:

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.basemap import Basemap, cm
3 import numpy as np
4 import pandas as pd
5 from numpy import linspace
6 from numpy import meshgrid
7
8 """ plot rainfall from the WRF output files
9 data = pd.read_table('data/d1.txt', sep=",")
10 lat = np.array(data['Lat'])
11 lon = np.array(data['Long'])
12 prec = np.array(data['Rain'])
13
14 """
15 fig = plt.figure(figsize=(8,8))
16 ax = fig.add_axes([0.1,0.1,0.8,0.8])
17
18 """ create polar stereographic Basemap instance.
19 m = Basemap(projection='merc', llcrnrlat=lat[0], urcrnrlat=lat[-1], llcrnrlon=lon[0], urcrnrlon=lon[-1])
20
21 """ draw coastlines, state and country boundaries, edge of map.
22 m.drawcoastlines()
23 m.drawstates()
24 m.drawcountries()
25
26 """
27 xx, yy = meshgrid(lat, lon)
28
29 """
30 m.contour(xx, yy, prec)
31
32 """
33 plt.savefig('test.png', dpi=300)
34 plt.show()
```

Help Panel: The "Usage" tab is selected. It contains the following text:

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

IPython console: The console shows the following output:

```
Python 3.6.2 [Anaconda custom (x86_64)] (default, Jul 20 2017, 13:14:59)
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:
```



The bottom status bar shows the following information:

- Permissions: RW
- End-of-lines: LF
- Encoding: ASCII
- Line: 34
- Column: 11
- Memory: 84 %












Jupyter

- Es una aplicación web open source que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto explicativo.
- Los usos más comunes para este IDE son: limpieza y transformación de datos, simulación numérica, modelado estadístico, aprendizaje automático entre otros.

IDE

 jupyter 1_Arreglos Numpy (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

          Code 

In [3]:

```
# librerias
import numpy as np
```

Crear Numpy Arrays

De una lista de python

Creamos el arreglo directamente de una lista o listas de python

In [4]:

```
my_list = [1,2,3]
my_list
```

Out[4]:

```
[1, 2, 3]
```

In [5]:

```
np.array(my_list)
```

Out[5]:

```
array([1, 2, 3])
```

In [6]:


```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
my_matrix
```

Out[6]:


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

Métodos

Barra de herramientas











 jupyter

1_Arreglos Numpy (autosaved)


 Logout

FileEditViewInsertCellKernelWidgetsHelp

TrustedPython 3



Code



In [3]:

```
# librerias
import numpy as np
```

Crear Numpy Arrays

De una lista de python

Creamos el arreglo directamente de una lista o listas de python

In [4]:

```
my_list = [1,2,3]
my_list
```

Out[4]:

```
[1, 2, 3]
```

In [5]:

```
np.array(my_list)
```

Out[5]:

```
array([1, 2, 3])
```

In [6]:

```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
my_matrix
```

Out[6]:

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

Métodos

Línea de código

The screenshot shows a Jupyter Notebook window titled "1_Arreglos Numpy (autosaved)". The top bar includes the Jupyter logo, the title, and a "Logout" button. Below the title bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are two buttons: "Trusted" and "Python 3". Below the menu bar is a toolbar with icons for saving, adding new files, undo, redo, copy, paste, up/down arrows, previous/next cell, run, and a keyboard icon. The main area of the notebook contains three code cells. The first cell, labeled "In [3]:", contains the code `# librerias` and `import numpy as np`. The second cell, labeled "In [4]:", contains the code `my_list = [1,2,3]` followed by `my_list` on the next line. The output for this cell, labeled "Out[4]:", is `[1, 2, 3]`. The third cell, labeled "In [5]:", contains the code `np.array(my_list)`. The output for this cell, labeled "Out[5]:", is `array([1, 2, 3])`. A fourth cell, labeled "In [6]:", contains the code `my_matrix = [[1,2,3],[4,5,6],[7,8,9]]` followed by `my_matrix` on the next line. The output for this cell, labeled "Out[6]:", is `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`. The notebook interface has a light gray background, and the code cells have a white background with a blue border on the left side.

Línea de resultado

The image shows a Jupyter Notebook interface. At the top, there's a header bar with the Jupyter logo, the text "jupyter 1_Arreglos Numpy (autosaved)", and a "Logout" button. Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help. To the right of the menu bar are two buttons: "Trusted" and "Python 3". Below the menu bar is a toolbar with icons for saving, adding new files, undo, redo, copy, paste, up/down arrows, previous/next cell, run, and a keyboard icon. The main area contains three code cells. The first cell has the prompt "In [3]:" followed by the code "# librerias\nimport numpy as np". The second cell has the prompt "In [4]:" followed by the code "my_list = [1,2,3]\nmy_list". Below it is the output "Out[4]: [1, 2, 3]". The third cell has the prompt "In [5]:" followed by the code "np.array(my_list)". Below it is the output "Out[5]: array([1, 2, 3])". The fourth cell has the prompt "In [6]:" followed by the code "my_matrix = [[1,2,3],[4,5,6],[7,8,9]]\nmy_matrix". Below it is the output "Out[6]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]". The bottom of the notebook shows the title "Métodos" in a large font.

Markdown

jupyter

1_Arreglos Numpy (autosaved)

Logout

File

Edit

View

Insert

Cell

Kernel

Widgets

Help

Trusted

Python 3

+

Code

In [3]:

```
# librerias
import numpy as np
```

Crear Numpy Arrays

De una lista de python

Creamos el arreglo directamente de una lista o listas de python

In [4]:

```
my_list = [1,2,3]
my_list
```

Out[4]:

```
[1, 2, 3]
```

In [5]:

```
np.array(my_list)
```

Out[5]:

```
array([1, 2, 3])
```

In [6]:

```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
my_matrix
```

Out[6]:

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

Métodos

Sintaxis Markdown

Título

Subtítulo

Título normal

**** Negrita ****

* Lista

` Código `

_ Itálica _

Título

Subtítulo

Título normal

Negrita

- Lista
- `Código`
- *Itálica*

Sintaxis

Indentación

El contenido de los bloques de código es delimitado mediante espacios o tabuladores, conocidos como indentación.

C (Indentación opcional)

```
int numeroMayor (int x, int y)
{
    if (x == y){
        printf("Los dos valores son iguales")
    }else if (x > y){
        printf("El primer valor es mayor")
    }else (x > y){
        printf("El primer valor es mayor")
    }
}
```

Python (Indentación obligatoria)

```
def numeroMayor(x,y):  
    if x == y:  
        print("Los dos valores son iguales")  
    elif x > y:  
        print("El primer valor es mayor")  
    else:  
        print("El segundo valor es mayor")
```


Python (Indentación obligatoria)

```
def numeroMayor(x,y):  
    ***if x == y:  
    *****print("Los dos valores son iguales")  
    ***elif x > y:  
    *****print("El primer valor es mayor")  
    ***else:  
    *****print("El segundo valor es mayor")
```

Comentarios

Existen dos formas. La primera y más apropiada para comentarios largos es utilizando la notación

''' comentario ''', tres apóstrofes de apertura y tres de cierre.

La segunda notación utiliza el símbolo **#**, y se extienden hasta el final de la línea.

```
'''  
Comentario multilinea  
'''  
  
print("Hola Mundo") # Comentario en línea
```

Imprimir variables

```
a = 2
print(a)
# imprime 2

print("Hola mundo")
# imprime Hola mundo

a = 2
b = 3

print("Multiplicar {} x {} = {}".format(a,b,a*b))
# imprime Multiplicar 2 x 3 = 6
```

Imprimir variables

```
a = 2
print(a)
# imprime 2

print("Hola mundo")
# imprime Hola mundo

a = 2
b = 3

print("Multiplicar {} x {} = {}".format(a,b,a*b))
# imprime Multiplicar 2 x 3 = 6
```

Variables

Las variables se definen de forma dinámica, lo que significa que no se tiene que especificar cuál es su tipo de antemano y puede tomar distintos valores en otro módulo, función o proceso, incluso de un tipo diferente al que tenía previamente.

Se utiliza el símbolo “=” para asignar valores

```
x = 1
```

```
x = "Texto"
```

```
# Esto es posible por que los tipos son asignados dinámicamente
```

Operadores lógicos

!	not
	or
&&	and

Tipos de datos

Tipo	Clase	Notas	Ejemplo
str	Cadena	Inmutable	Cadena'
unicode	Cadena	Versión Unicode de str	u'Cadena'
list	Secuencia	Mutable, puede contener objetos de diversos tipos	[4.0, 'Cadena', True]
tuple	Secuencia	Inmutable, puede contener objetos de diversos tipos	(4.0, 'Cadena', True)
set	Conjunto	Mutable, sin orden, no contiene duplicados	set([4.0, 'Cadena', True])
frozenset	Conjunto	Inmutable, sin orden, no contiene duplicados	frozenset([4.0, 'Cadena', True])
dict	Mapping	Grupo de pares clave:valor	{'key1': 1.0, 'key2': False}
int	Número entero	Precisión fija, convertido en long en caso de overflow.	42
long	Número entero	Precisión arbitraria	42L ó 456966786151987643L
float	Número decimal	Coma flotante de doble precisión	3.1415927
complex	Número complejo	Parte real y parte imaginaria j.	(4.5 + 3j)
bool	Booleano	Valor booleano verdadero o falso	True o False

Lista

Para declarar una lista se usan los corchetes “[]”.

Para acceder a los elementos de una lista se utiliza un índice entero (empezando por "0", no por "1"). Se pueden utilizar índices negativos para acceder elementos a partir del final.

Las listas se caracterizan por ser mutables, es decir, se puede cambiar su contenido en tiempo de ejecución.

```
>>>> lista = ["abc", 42, 3.1416]

>>>> lista[0] # Acceder a un elemento por su índice

'abc'

>>>> lista[-1] # Acceder a un elemento usando un índice negativo

3.1416

>>>> lista.append(True) # Añadir un elemento al final de la lista

>>>> lista

['abc', 42, 3.1416, True]

>>>> del lista[3] # Borra el elemento número 3 de la lista

>>>> lista[0] = "xyz" # Re-asignar el valor del primer elemento de la lista

>>>> lista[0:2] # Mostrar los elementos de la lista del índice 0 al 2

["xyz", 42]

>>>> lista_anidada = [lista, [True, 42L]]

>>>> lista_anidada

[['xyz', 42, 3.1416], [True, 42L]]

>>>> lista_anidada[1][0] # Acceder a un elemento de una lista dentro de otra lista
```

Diccionarios

Para declarar un diccionario se usan las llaves “{ }”. Contienen elementos separados por comas, donde cada elemento está formado por un par **clave : valor** (el símbolo : separa la clave de su valor correspondiente).

Los diccionarios son mutables, es decir, se puede cambiar el contenido de un valor en tiempo de ejecución.

En cambio, las claves de un diccionario deben ser inmutables. Esto quiere decir, por ejemplo, que no podremos usar ni listas ni diccionarios como claves.

El valor asociado a una clave puede ser de cualquier tipo de dato, incluso un diccionario.

```
>>>> diccionario = {"cadena": "abc", "numero": 42, "lista": [True, 42L]}
# Diccionario que tiene diferentes valores por cada clave
>>>> diccionario["cadena"] # Accede a su valor usando clave
'abc'
>>>> diccionario["lista"][0] # Acceder a un elemento de una lista dentro de un valor
True
>>>> diccionario["cadena"] = "xyz" # Re-asignar el valor de una clave
>>>> diccionario["cadena"]
"xyz"
>>>> diccionario["decimal"] = 3.1416 # Insertar un nuevo elemento clave:valor
>>>> diccionario["decimal"]
3.1416
```

Clases

Las clases se definen con la palabra clave `class`, seguida del nombre de la clase y, si hereda de otra clase, el nombre de esta.

En una clase un "método" equivale a una "función", y un "atributo" equivale a una "variable".

"`__init__`" es un método especial que se ejecuta al instanciar la clase, se usa generalmente para inicializar atributos y ejecutar métodos necesarios. Al igual que todos los métodos en Python, debe tener al menos un parámetro, generalmente se utiliza `self`. El resto de parámetros serán los que se indiquen al instanciar la clase.

Los atributos que se desee que sean accesibles desde fuera de la clase se deben declarar usando `self`. delante del nombre.

En python no existe el concepto de encapsulación, por lo que el programador debe ser responsable de asignar los valores a los atributos

```

>>>> class Persona(Object):
...     def __init__(self, nombre, edad):
...         self.nombre = nombre # atributo
...         self.edad = edad # atributo
...     def mostrar_edad(self): # Es necesario que, al menos, tenga un parámetro, generalmente "self"
...         print(self.edad) # mostrar un atributo
...     def modificar_edad(self, edad): #modificando Edad
...         if edad < 0 or edad > 150: # Comprobar que la edad no sea menor de 0, ni mayor a 150
...             return False
...         else: # Si esta en el rango 0-150, entonces se modifica la variable
...             self.edad = edad # modificar la edad

>>>> p = Persona("Jorge", 20) # Instanciar la clase, como se puede ver, no se especifica el valor de "self"
>>>> p.nombre # la variable "nombre" del objeto sí es accesible desde fuera
'Jorge'
>>>> p.nombre = 'Ernesto' # Y por tanto, se puede cambiar su contenido
'Ernesto'
>>>> p.mostrar_edad() # Se llama a un método de la clase
20
>>>> p.modificar_edad(21)
# Es posible cambiar la edad usando el método específico que hemos hecho para hacerlo de forma controlada
>>>> p.mostrar_edad()
21

```

Condicionales

Una sentencia condicional **if** ejecuta su bloque de código interno sólo si se cumple cierta condición. Se define usando la palabra clave `if` seguida de la condición, y el bloque de código.

Condiciones adicionales, si las hay, se introducen usando **elif** seguida de la condición y su bloque de código. Todas las condiciones se evalúan secuencialmente hasta encontrar la primera que sea verdadera, y su bloque de código asociado es el único que se ejecuta.

Opcionalmente, puede haber un bloque final (la palabra clave es **else** seguida de un bloque de código) que se ejecuta sólo cuando todas las condiciones fueron falsas.


```
>>>> verdadero = True

>>>> if verdadero: # No es necesario poner "verdadero == True"

...     print("Verdadero")

... else:

...     print("Falso")

Verdadero

>>>> lenguaje = "Python"

>>>> if lenguaje == "C": # Lenguaje no es "C", por lo que este bloque se obviará y evaluará la siguiente condición

...     print("Lenguaje de programación: C")

...     elif lenguaje == "Python": # se pueden añadir tantos bloques "elif" como se quiera

...         print("Lenguaje de programación: Python")

...     else: # en caso de que ninguna de las anteriores condiciones fuera cierta, se ejecutaría este bloque

...         print("Lenguaje de programación: Indefinido")

...

Lenguaje de programación: Python

>>>> if verdadero and lenguaje == "Python": # Uso de "and" para comprobar que ambas condiciones son verdaderas

...     print("Verdadero y Lenguaje de programación: Python")

...

Verdadero y Lenguaje de programación: Python
```

Ciclos

For

Es similar a **foreach** en otros lenguajes.

Recorre un objeto iterable, como una lista, una tupla o un generador, y por cada elemento del iterable ejecuta el bloque de código interno.

Se define con la palabra clave **for** seguida de un nombre de variable, seguido de **in**, seguido del iterable, y finalmente el bloque de código interno.

En cada iteración, el elemento siguiente del iterable se asigna al nombre de variable especificado

```
>>>> lista = ["a", "b", "c"]
>>>> for i in lista: # iteramos sobre una lista
...     print(i)
...
a
b
c
```

```
>>>> cadena = "abcdef"
for i in cadena: # iteramos sobre una cadena
...     print(i)
...
a
b
c
d
e
f
```

While

Evalúa una condición y, si es verdadera, ejecuta el bloque de código interno. Continúa evaluando y ejecutando mientras la condición sea verdadera. Se define con la palabra clave **while** seguida de la **condición**, y a continuación el bloque de código interno

```
>>>> numero = 0
```

```
>>>> while numero < 5:
```

```
...     print(numero)
```

```
...         numero += 1 # un buen programador modificará las variables
```

```
...         # de control al finalizar el ciclo while
```

```
...
```

```
0
```

```
1
```

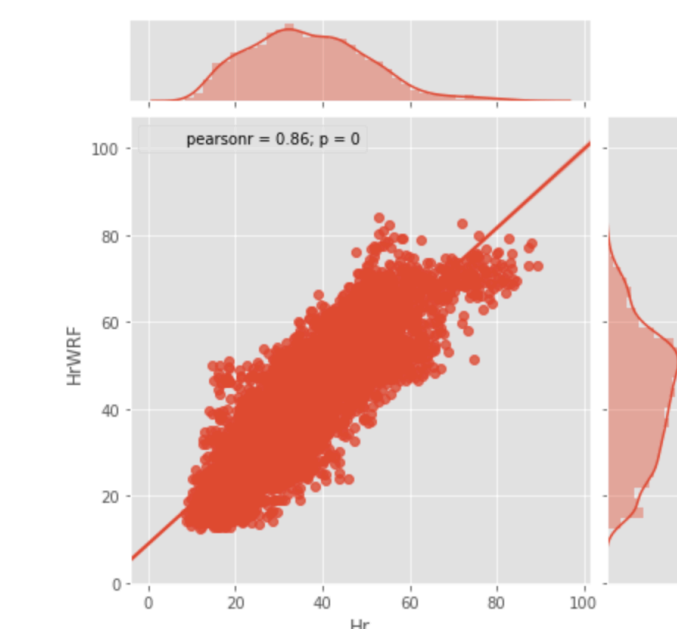
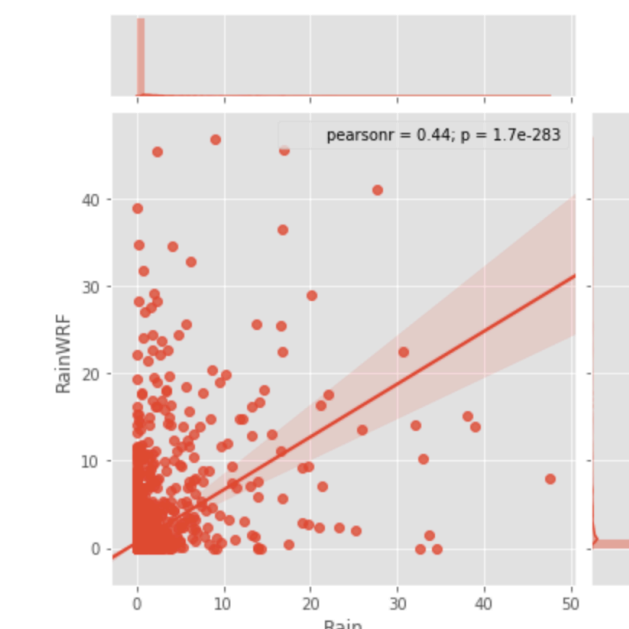
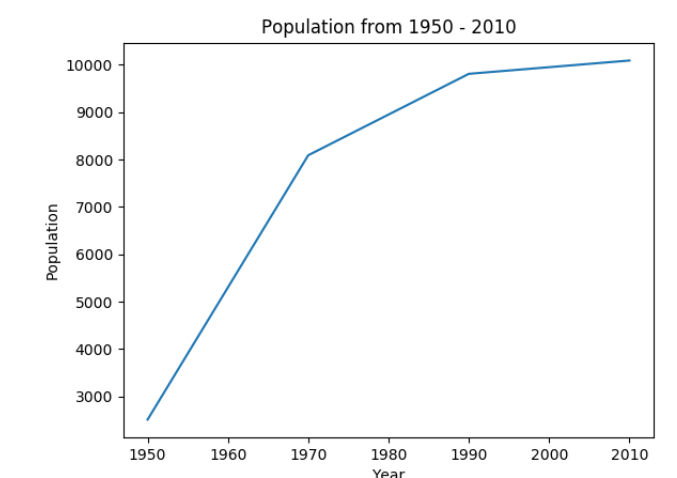
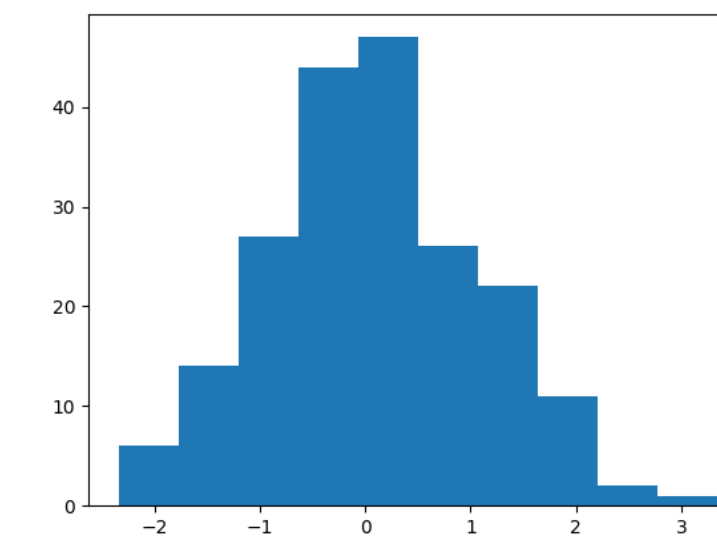
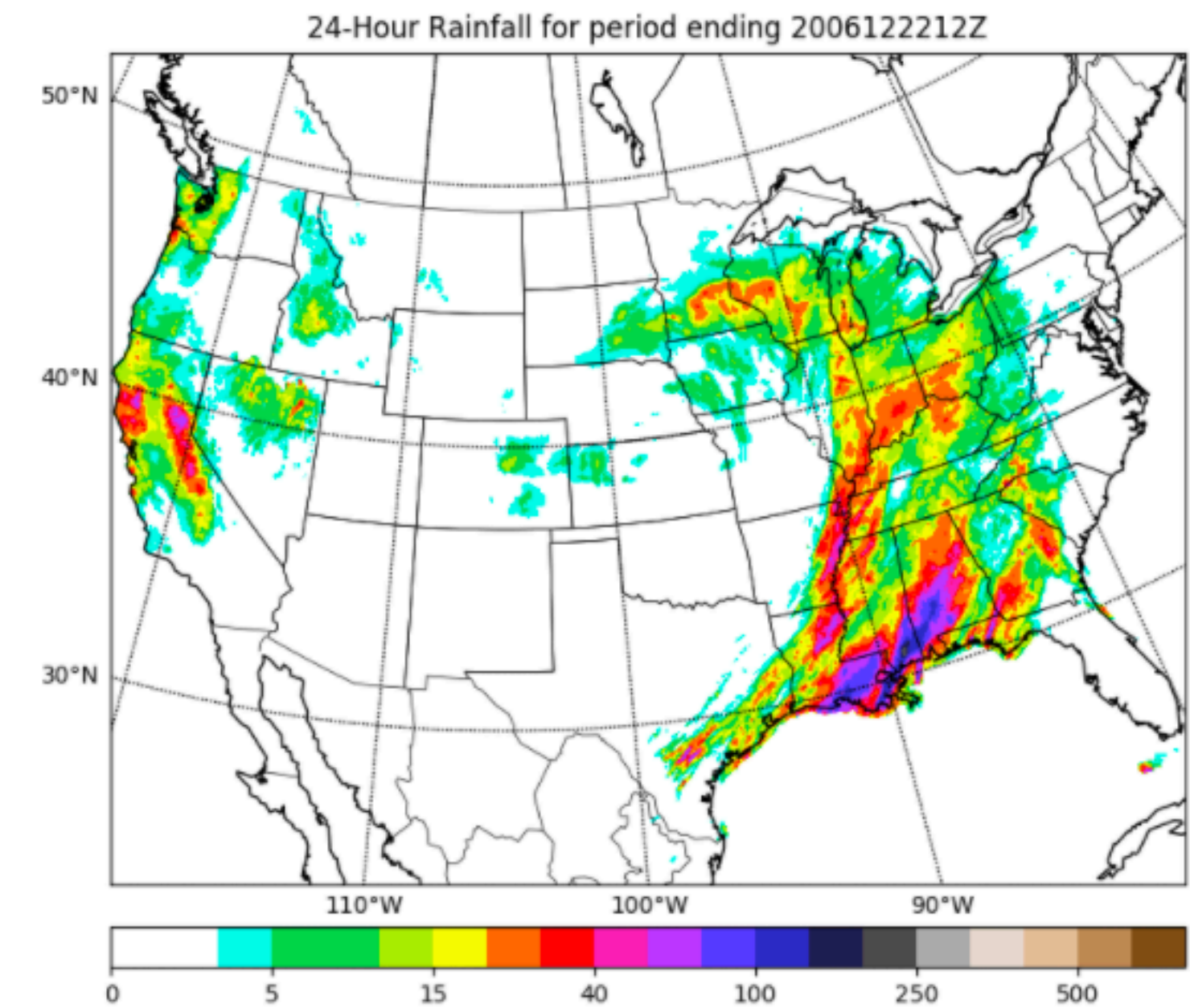
```
2
```

```
3
```

```
4
```

Módulos

- Existen muchas propiedades que se pueden agregar al lenguaje importando módulos, que son "minicódigos" (la mayoría escritos también en Python) que proveen de ciertas funciones y clases para realizar determinadas tareas.
- Un ejemplo es el módulo Basemap, que nos permite crear mapas.
- Otro ejemplo es el módulo os, que provee acceso a muchas funciones del sistema operativo. Los módulos se agregan a los códigos escribiendo import seguida del nombre del módulo que queremos usar.




```
>>>> import os # módulo que provee funciones del sistema operativo

>>>> os.name # devuelve el nombre del sistema operativo

'posix'

>>>> os.mkdir("tmp/ejemplo") # crea un directorio en la ruta especificada

>>>> import time # módulo para trabajar con fechas y horas

.... time.strftime("%Y-%m-%d %H:%M:%S")

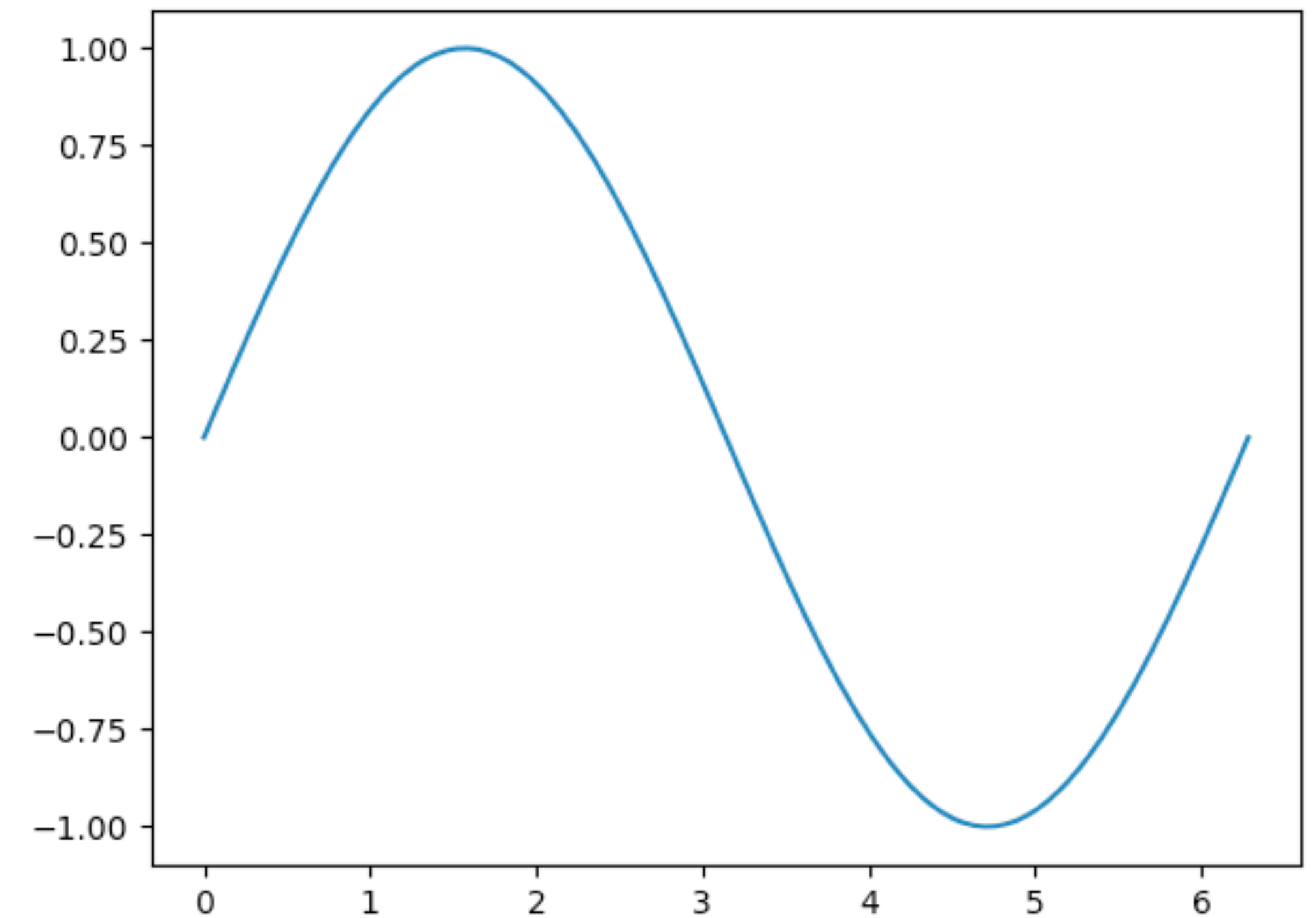
# dándole un cierto formato, devuelve la fecha y/u hora actual

'2017-09-15 10:52:01'
```

Numpy

Es un módulo de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

```
import numpy as np
from matplotlib import pyplot
x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)
pyplot.plot(x,y)
pyplot.show()
```



Arreglos

```
import numpy as np

a = np.array([1,2,3]) # arreglo de rank 1

print(type(a)) # imprime "<class 'numpy.ndarray'>"

print(a.shape) # imprime "(3,)"

print(a[0], a[1], a[2]) # imprime "1 2 3"

a[0] = 5 # cambiar un elemento del arreglo

print(a) # imprime "[5,2,3]"

b = np.array([[1,2,3],[4,5,6]]) # arreglo de rank 2

print(b.shape) # imprime ("2,3")

print(b[0,0], b[0,1], b[1,0]) # imprime "1, 2, 4"
```

Funciones

```
import numpy as np

a = np.zeros((2,2)) # crea un arreglo de 0's

print(a)

# imprime "[[ 0.  0.]
           [ 0.  0.]]"

b = np.ones((1,2)) # crea un arreglo de 1's

print(b) # imprime "[[1. 1.]]"

c = np.full((2,2),7) # crea un arreglo de una constante

print(c) # imprime "[[7. 7.
                    [7. 7.]]"

d = np.eye(2) # crea una matrix de 2x2

print(d) # imprime "[[1. 0]
                    [0. 1.]]"

e = np.random.random((2,2)) # crea un arreglo de números random

print(e) # imprime "[[0.91209392 0.98736452
                    [0.87594037 0.892384719]]]"
```

Índices

```
import numpy as np

# Crea un arreglo de rank 2 con la siguiente forma (3,4)

# [[1  2  3  4]
#
#    [5  6  7  8]
#
#    [9 10 11 12]]

a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

# crear un subarreglo de los 2 primeros renglones y la primer columna

# [[2 3
#
#    [6 7]]

b = a[:2, 1:3]

# una división del arreglo es una vista de los mismos datos, por lo tanto cualquier

# modificación, modificara el arreglo original

print(a[0,1]) #imprime 2

b[0,0] = 77 # b[0,0] es el miso dato a a[0,1]

print(a[0,1]) # imprime 77
```

Funciones básicas matemáticas

Suma

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
print(x + y)
```

```
print(np.add(x,y))
```

```
# [[6.0  8.0]
   [10.0 12.0]]
```

Resta

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
print(x - y)
```

```
print(np.subtract(x,y))
```

```
# [[-4.0 -4.0]
   [-4.0 -4.0]]
```

Multiplicación

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
print(x * y)
```

```
print(np.multiply(x,y))
```

```
# [[ 5.0 12.0]
   [21.0 32.0]]
```

División

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
print(x / y)
```

```
print(np.divide(x,y))
```

```
# [[ 0.2          0.33333333]
   [ 0.42857143  0.5         ]]
```

Raíz cuadrada

```
import numpy as np
```

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
```

```
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

```
print(np.sqrt(x))
```

```
# [[ 1.          1.41421356]
   [ 1.73205081  2.          ]]
```

Pandas

Es un módulo de Python que proporciona estructuras de datos rápidas, flexibles y expresivas.

Esta diseñado para que trabajar con datos "relacionales" o "etiquetados".

Su objetivo es ser el módulo fundamental de alto nivel para realizar análisis de datos reales.

Crear un DataFrame

```
import pandas as pd
```

```
data = {'Country': ['Belgium', 'India', 'Brazil'],  
        'Capital': ['Brussels', 'New Delhi', 'Brasília'],  
        'Population': [11190846, 1303171035, 207847528]}
```

```
df = pd.DataFrame(data, columns=['Country', 'Capital', 'Population'])
```

```
df = pd.read_csv('file.csv', header=None, nrows=5)
```

```
df = pd.read_excel('file.xlsx')
```


Información del DataFrame

`df.shape` # (filas, columnas)

`df.index` # describe los indices

`df.columns` # describe las columnas

`df.info()` # información del dataframe

`df.count()` # número de valores no NaN

Resumen del DataFrame

`df.sum()` # suma de los valores

`df.cumsum()` # acumulado de la sumatoria de los valores

`df.min()` / `df.max()` # mínimo / máximo de los valores

`df.idxmin()` / `df.idxmax()` mínimo / máximo de los índices

`df.describe()` # resumen estadístico

`df.mean()` # media de los valores

`df.median()` # mediana de los valores

Eliminar registros del DataFrame

```
s.drop(['a', 'c']) # eliminar datos de filas (axis=0)  
df.drop('Country', axis=1) # eliminar datos de una columna (axis=1)
```

Ordenar el DataFrame

```
df.sort_index() # ordenar por indice
```

```
df.sort_values(by='Country') # ordenar por valores de la columna Country
```

```
df.rank() # asignar un ranking
```

Matplotlib

Matplotlib es un módulo de trazado 2D de Python que produce gráficas de calidad de publicación en una variedad de formatos impresos y entornos interactivos a través de plataformas.

Matplotlib se puede utilizar en shell, scripts, IPython, Jupyter, servidores de aplicaciones web.

Gráfica de línea

```
import matplotlib.pyplot as plt

year = [1950, 1970, 1990, 2010]

pop = [2510, 8090, 9809, 10090]

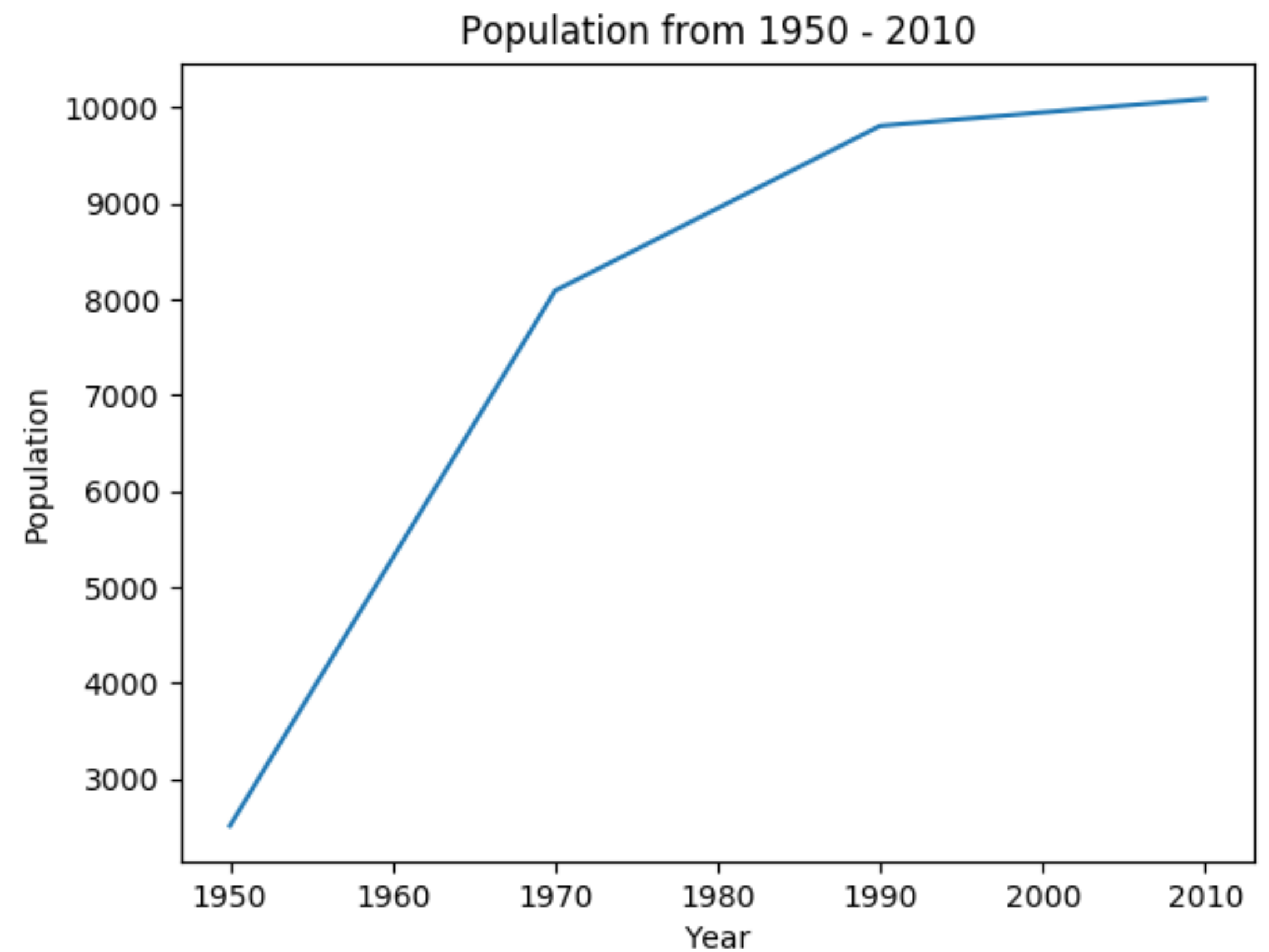
plt.plot(year, pop)

plt.xlabel('Year')

plt.ylabel('Population')

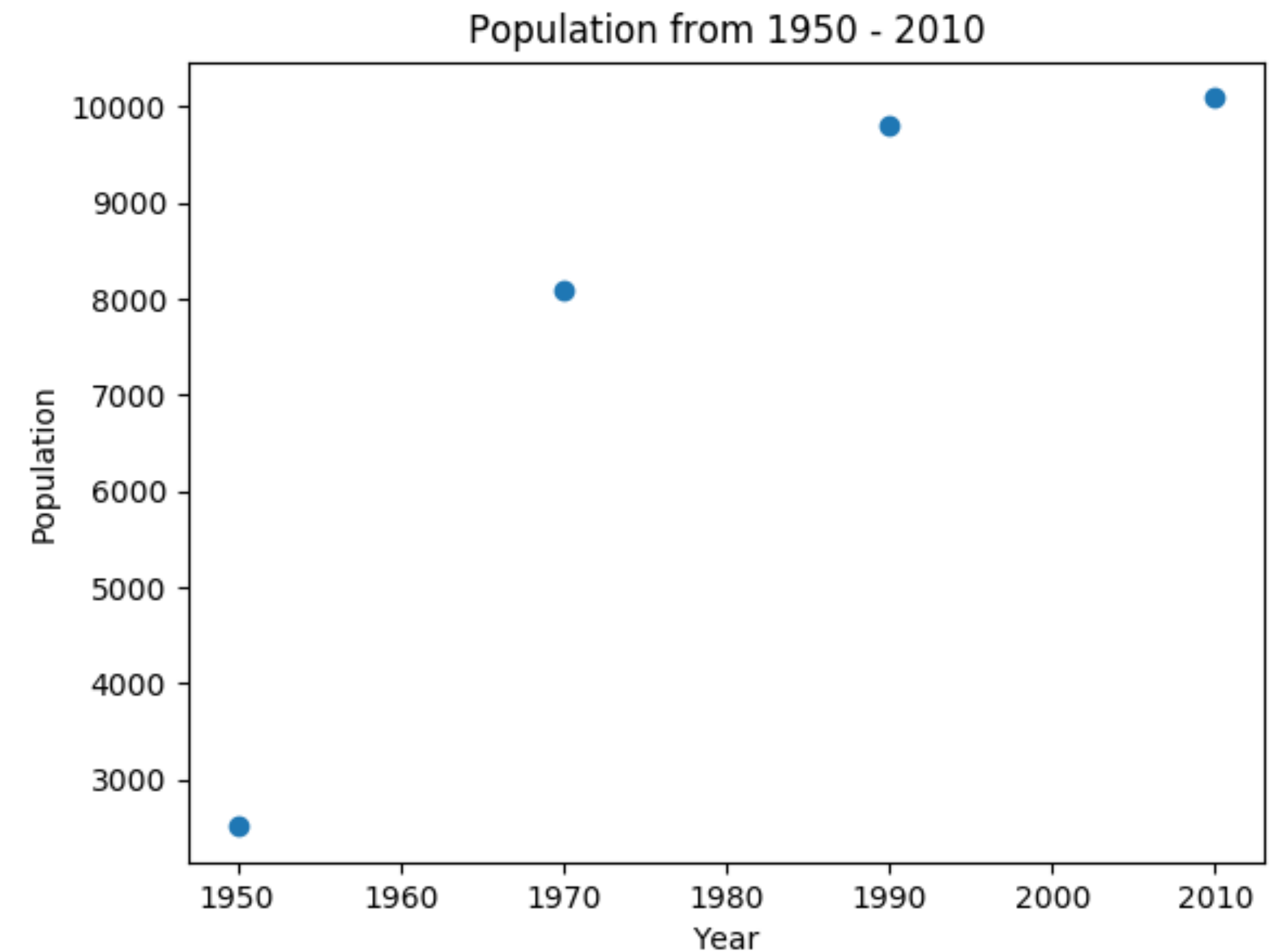
plt.title('Population from 1950 - 2010')

plt.show()
```



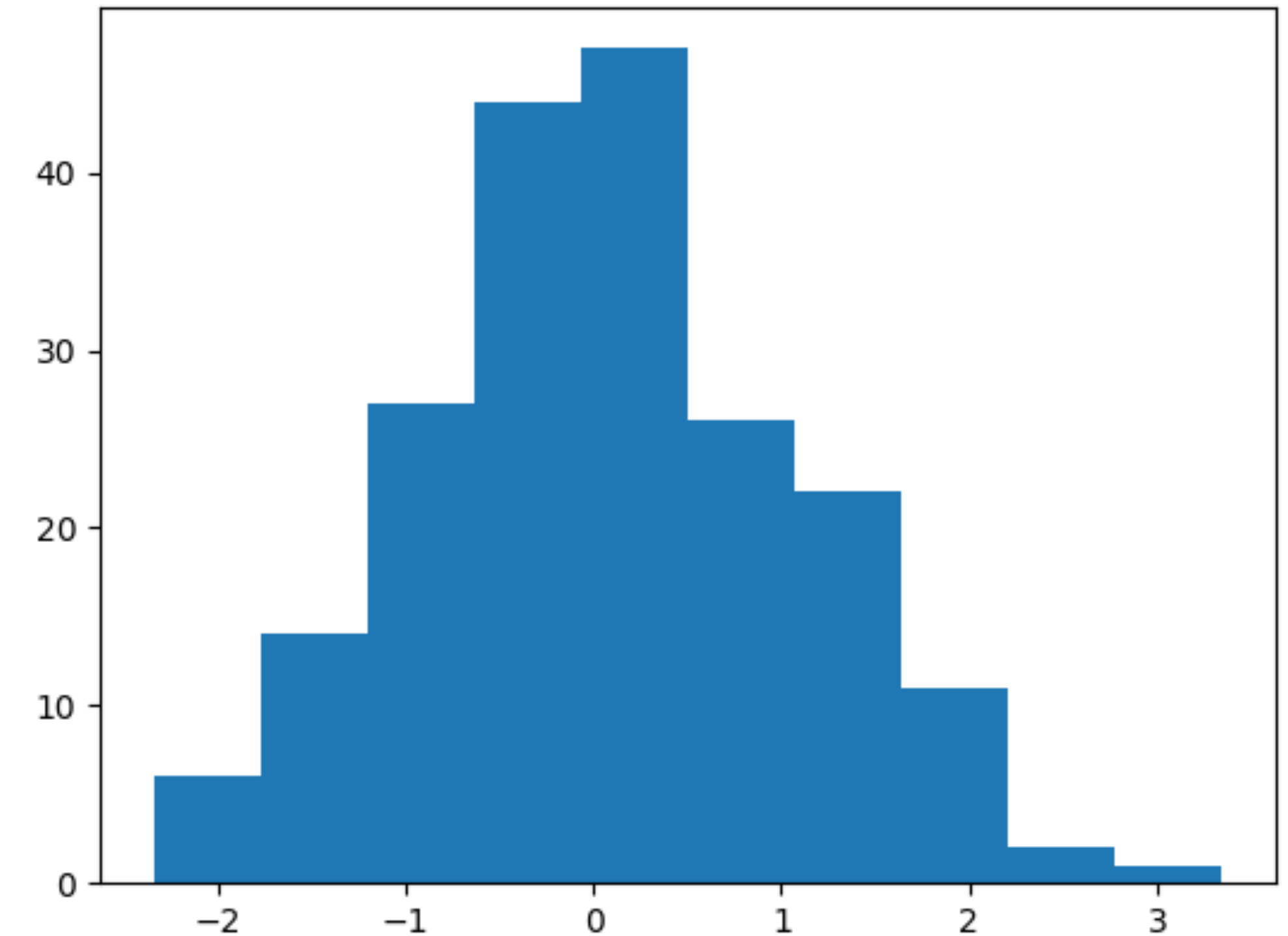
Gráfica de puntos

```
import matplotlib.pyplot as plt  
  
year = [1950, 1970, 1990, 2010]  
pop = [2510, 8090, 9809, 10090]  
  
plt.scatter(year, pop)  
  
plt.xlabel('Year')  
  
plt.ylabel('Population')  
  
plt.title('Population from 1950 - 2010')  
  
plt.show()
```



Histograms

```
import numpy as np
import matplotlib.pyplot as plt
values = np.random.randn(200)
plt.hist(values)
plt.show()
```



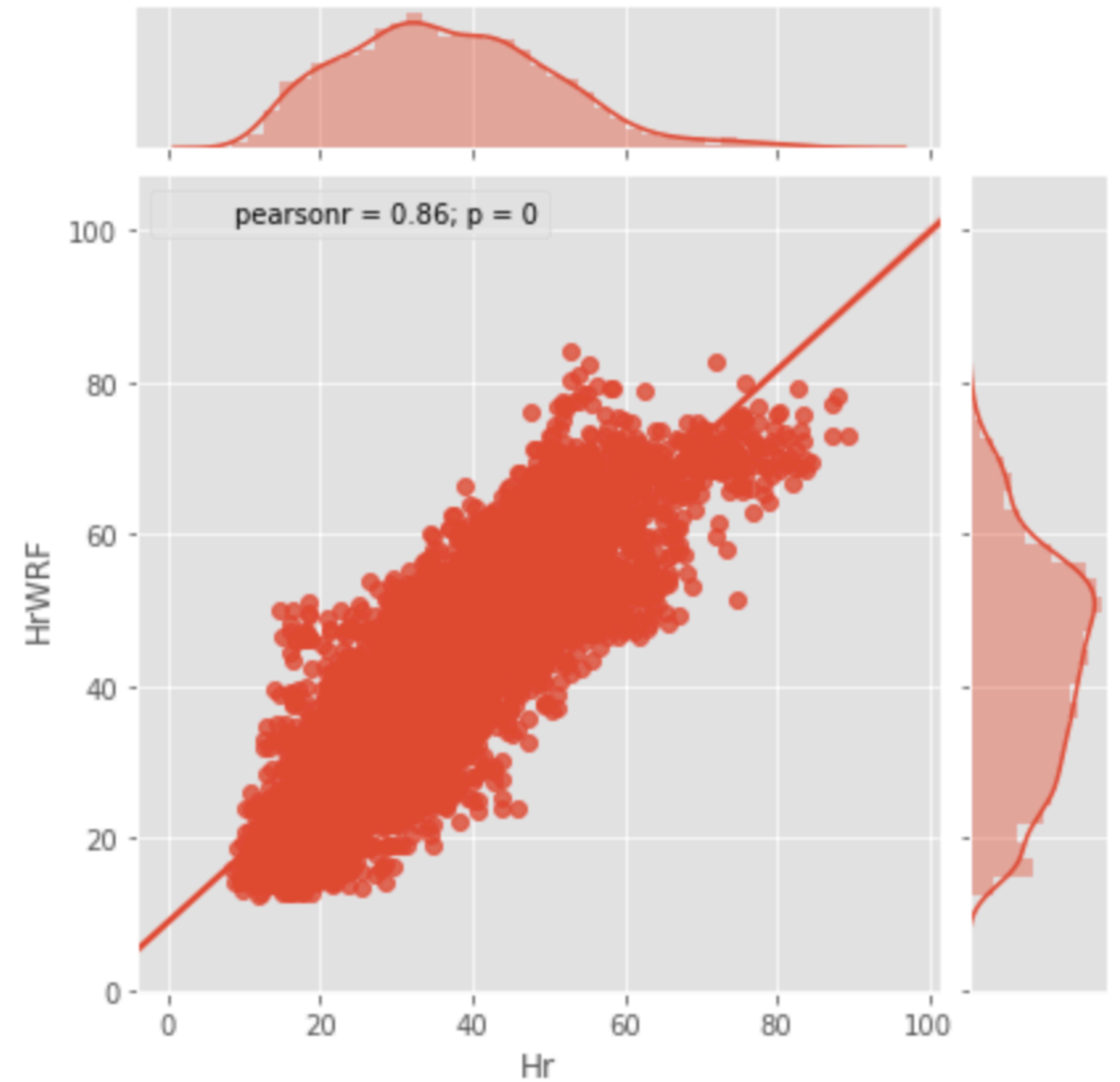
Seaborn

Es una biblioteca para hacer gráficos estadísticos atractivos e informativos en Python.

Está construido sobre matplotlib, está estrechamente integrado con PyData, incluye soporte para estructuras de datos numpy y pandas, así como de rutinas estadísticas de scipy y modelos de estadísticas.

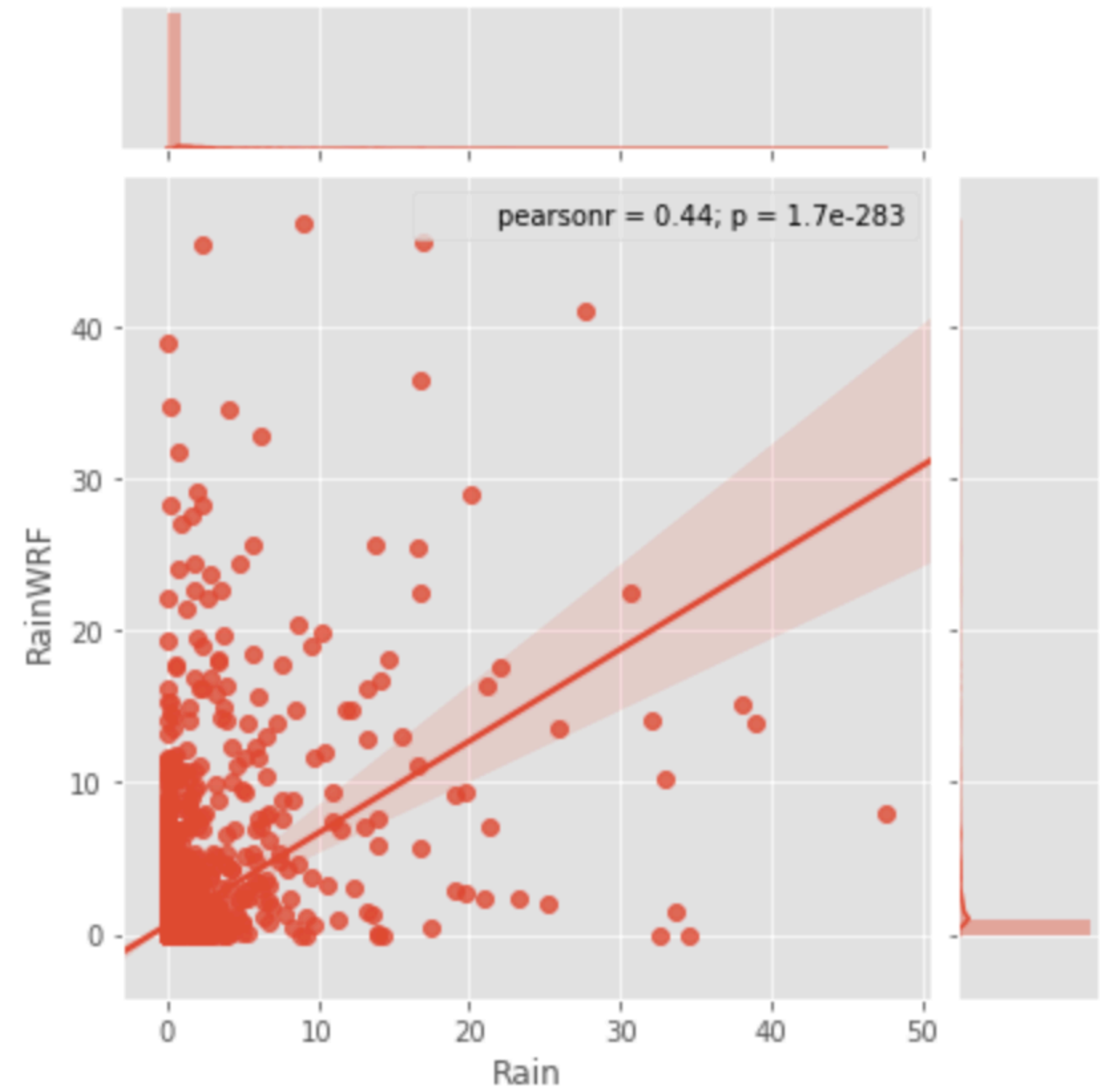
Regresión lineal con p y pearsonr

```
import seaborn as sns  
sns.jointplot("Hr", "HrWRF", data=data, kind="reg")
```



Regresión lineal con p y pearsonr

```
import seaborn as sns  
sns.jointplot("Rain", "RainWRF", data=data, kind="reg")
```



ZEN OF PYTHON

Keep in mind Python's philosophy as you code

```
>>> import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess

...

@Tim Peters

<https://github.com/python/peps/blob/master/pep-0020.txt>

Estadística

Tipos de datos

Numéricos

Categóricos

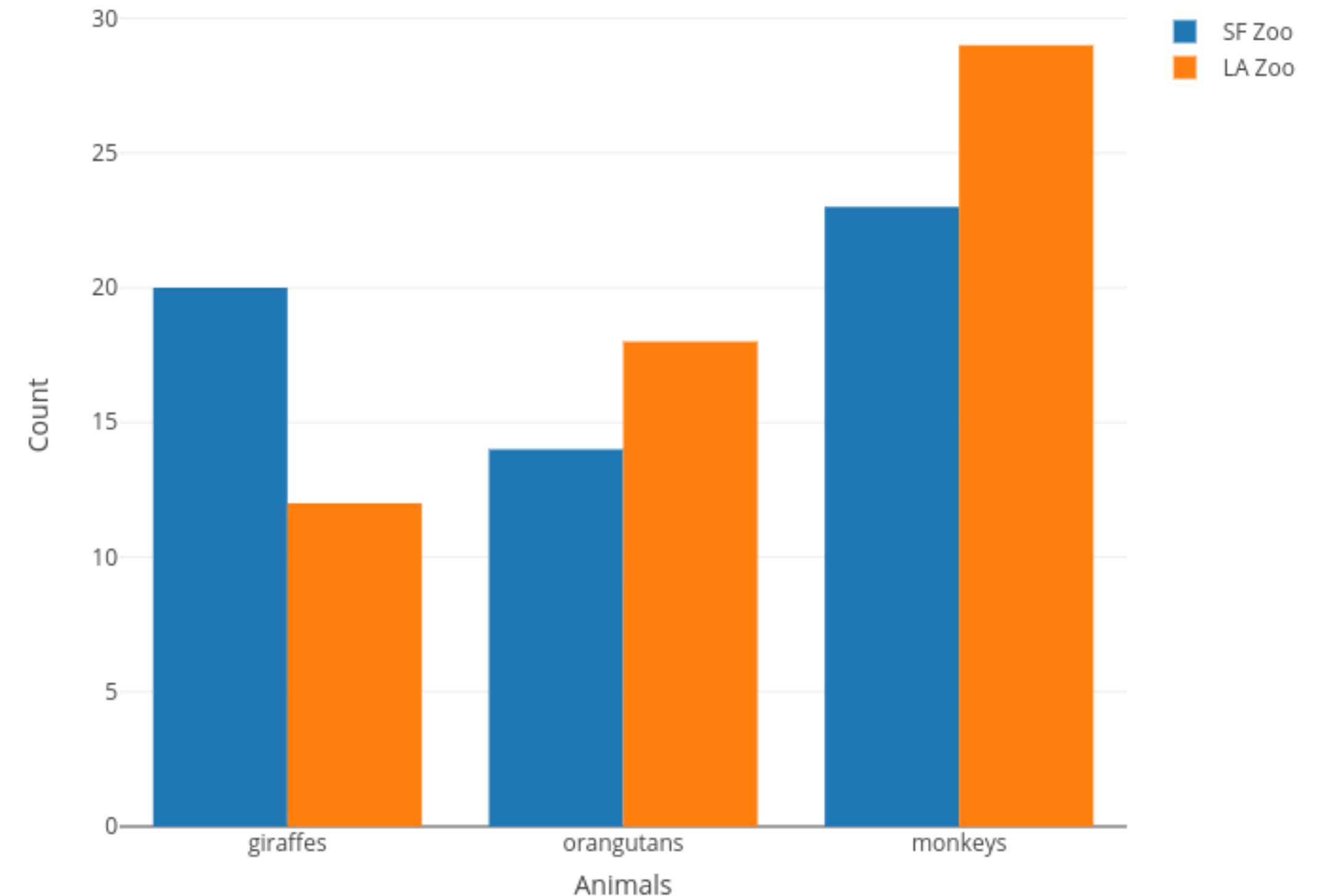
Ordinal

Numéricos

Representan valores cuantificables como la altura de una persona, el valor de una acción, etc.

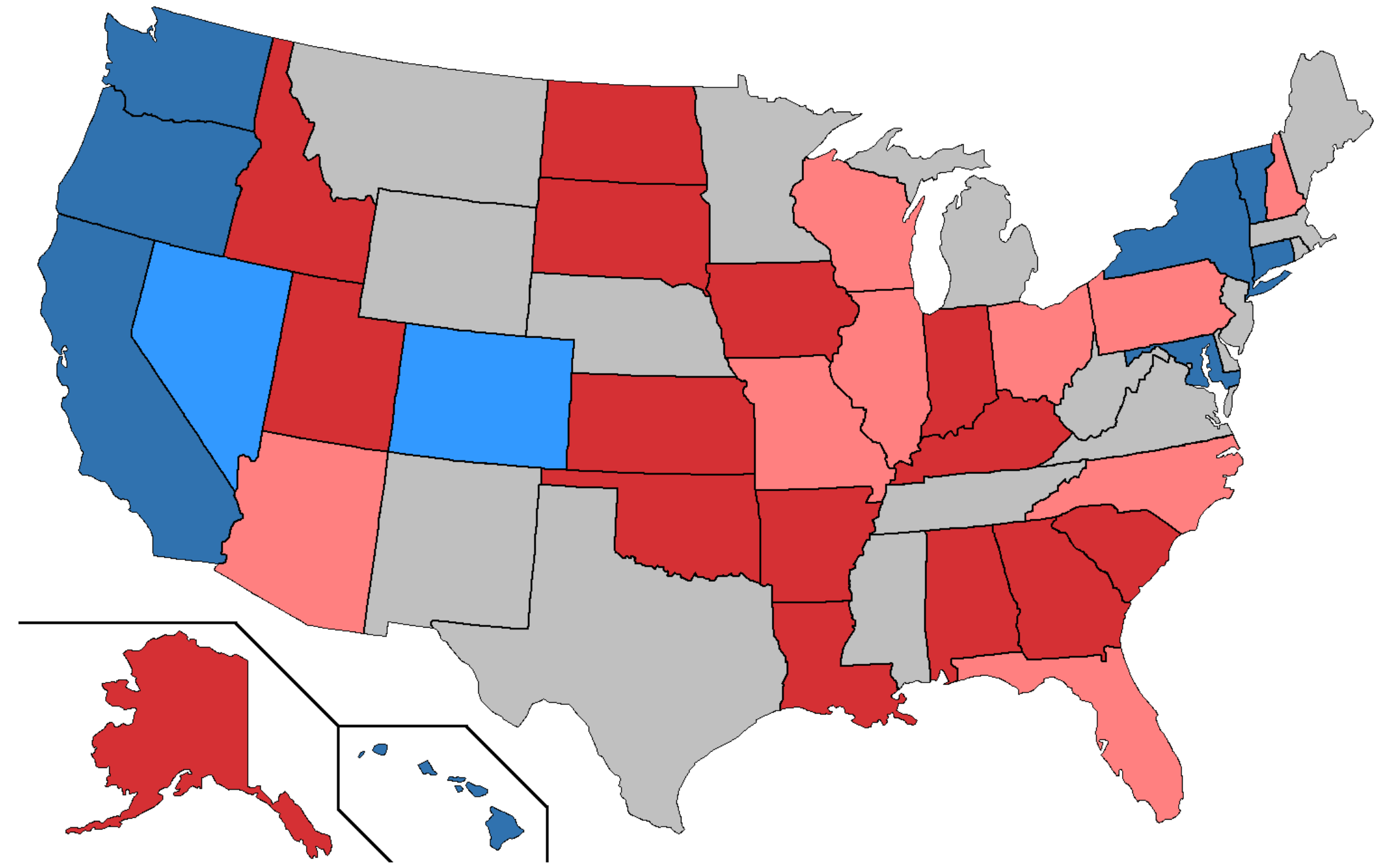
Valor discreto: Usualmente es un valor que cuenta un evento, por ejemplo, Cuantos días vas al gym en un mes?

Valor continuo: Tiene un número infinito de posibilidades para sus valores, por ejemplo, Cuanta lluvia puede caer en un día?



Categorico

Información cualitativa
que no tienen un
significado matemático, se
pueden asignar números a
categorías.



Ordinal

Combinación de numéricos y
categóricos

Información categórica que
tiene un significado
matemático.

Ejemplo:

Rating de las películas en
Netflix



Quiz

Qué tipo de datos son las respuestas de las siguientes preguntas?

La cantidad de café que tiene tu termo?

El servicio de tu compañía de cable donde 1,2,3,4 corresponden a "bajo", "moderado", "bueno" y "excelente"

Dinero qué gastas al hacer el super?

La edad de una persona en años?

Media

Promedio

Sumatoria / Número de muestras

Ejemplo

Calificación de un grupo de estudiantes

9,10,8,8,7,6,10,10,10,9

La MEDIA es: $(9+10+8+8+7+6+10+10+10+9)/10$

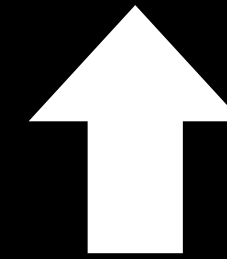
Mediana

Ordenar los valores y tomar el valor que este justo en el centro.

Ejemplo:

Calificaciones: 9,10,8,8,7,6,10,10,10,9

Ordenar : 6,7,8,8,9,9,10,10,10,10



Mediana

Si se tiene un número par de ejemplares, toma el promedio de los dos valores intermedios.

La media es menos susceptible a los valores atípicos, como suele suceder con la media.

Ejemplo: la media del ingreso en US es de \$72,641, pero la mediana es de solo \$51,939, por que la media esta sesgada por la cantidad de billonarios que existen.

En este caso en particular la mediana representa mejor el "típico" Americano

Moda

El valor más común en una serie de datos.

Calificación: 9,10,8,8,7,6,10,10,10,9

Moda

El valor más común en una serie de datos.

Calificación: 9,10,8,8,7,6,10,10,10,9

Cuántas veces se repite una calificación?

6:1, 7:1, 8:2, 9:2, 10:4

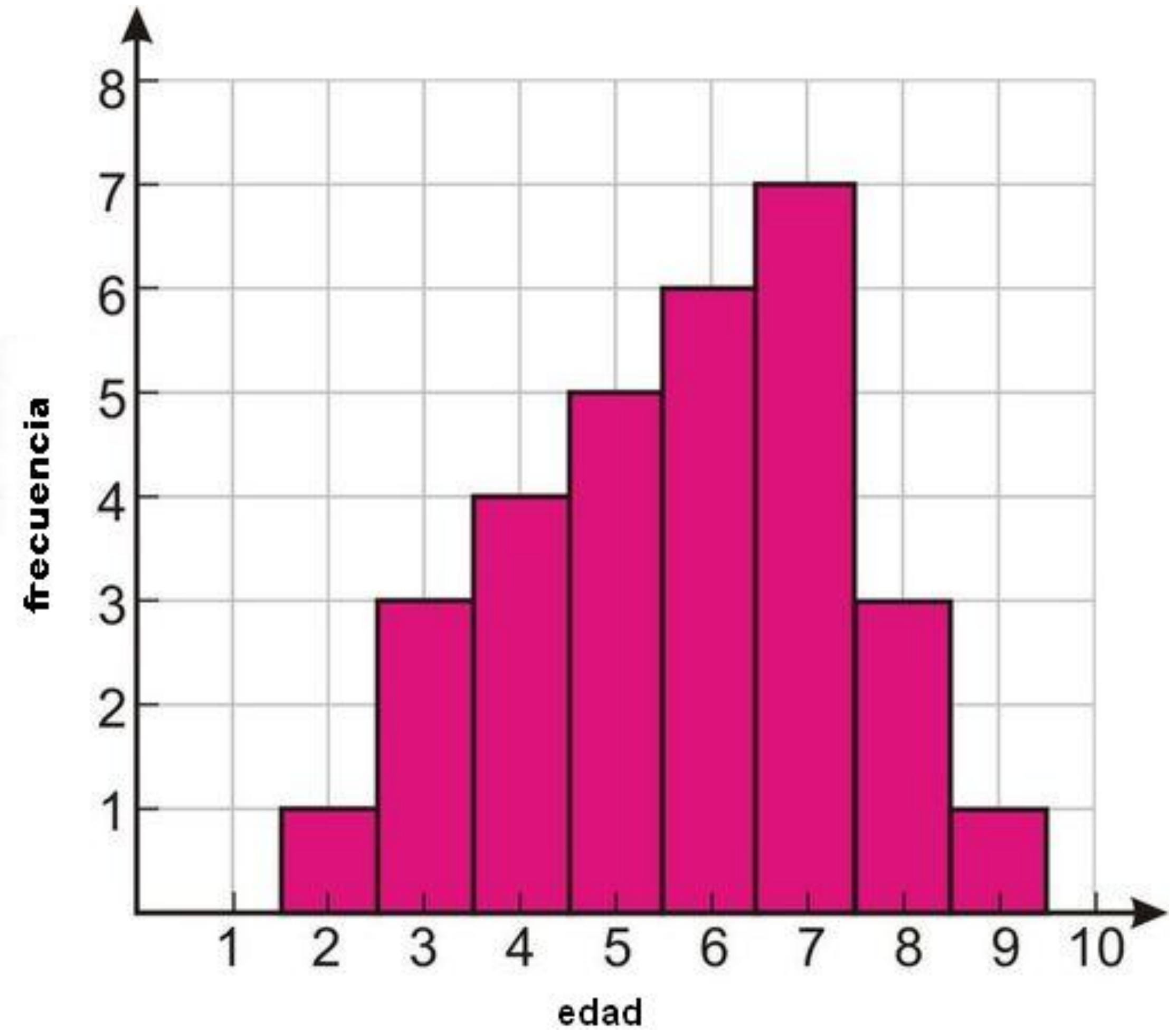
La MODA es 10

Desviación estándar y varianza

Histograma

Edades de alumnos de un grupo de futbol infantil:

2,3,3,3,4,4,4,5,5,5,5,5,6,6,6,6,
6,6,7,7,7,7,7,7,7,8,8,8,9



Varianza

La varianza mide como se “extiende” la información

Varianza(σ^2) es el promedio de las diferencias al cuadrado con respecto a la media

Ejemplo: Cual es la varianza de los siguientes datos (1,4,5,4,8)

1. Calcular la media: $(1+4+5+4+8)/5=4.4$

2. Diferencia con respecto a la media $(-3.4, -0.4, 0.6, -0.4, 3.6)$

3. Cuadrado de las diferencias $(11.56, 0.16, 0.36, 0.16, 12.96)$

4. Promedio del cuadrado de las diferencias:

$$\sigma^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04$$

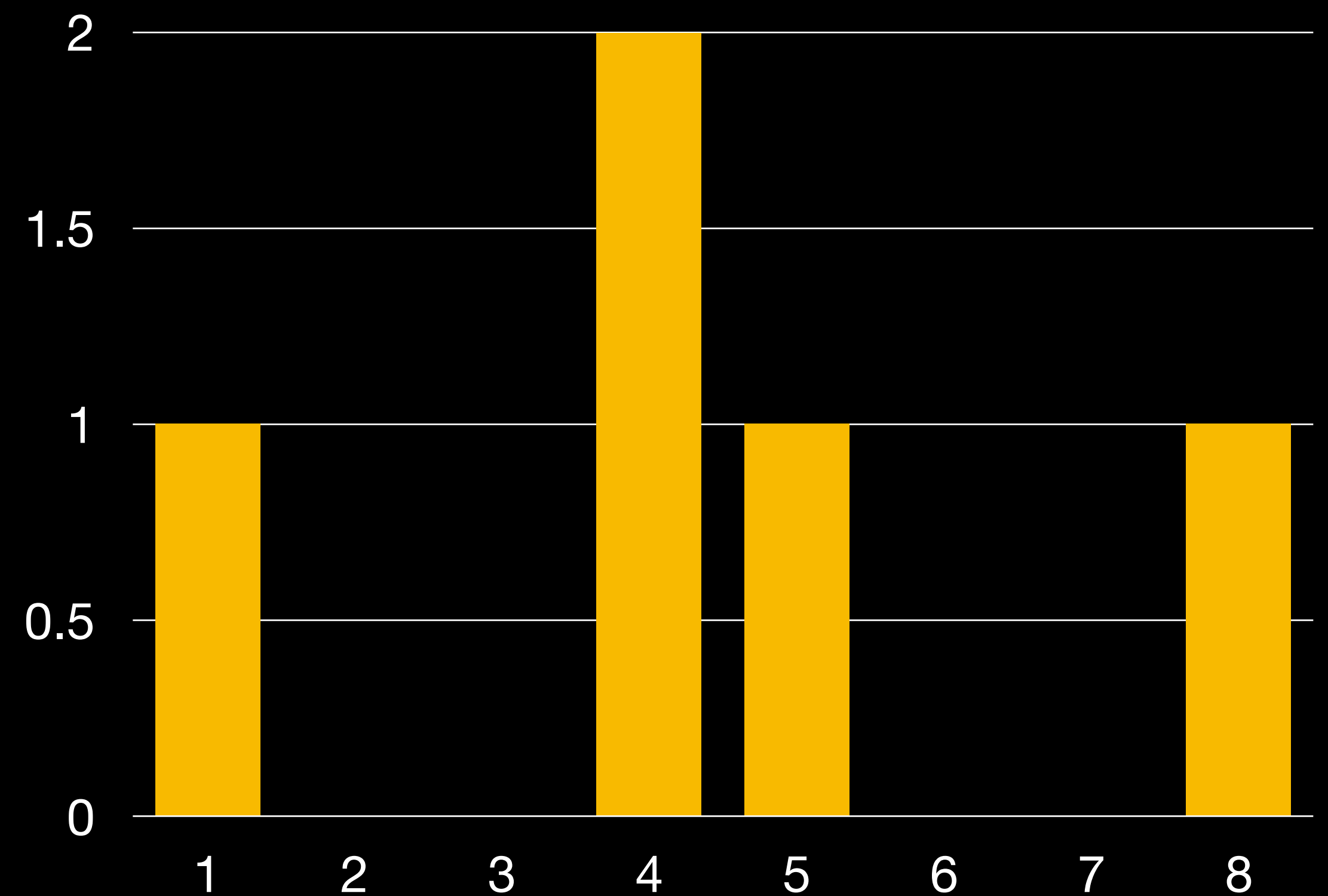
Desviación estándar (σ)

Es la raíz cuadrada de la varianza.

$$\sigma^2 = 5.04$$

$$\sigma = \sqrt{5.04} = 2.24$$

La desviación estándar de $(1, 4, 5, 4, 8)$ es 2.24, usualmente se utiliza para identificar datos atípicos. Aquellos datos que se encuentran más allá de una desviación estándar de la media son considerados inusuales.



Población vs Muestra

Si se esta trabajando con una muestra de toda la información.

Se debe de utilizar la varianza muestra en vez de la varianza de la población

Para "n" muestras, se debe de dividir las variantes entre N-1 en vez de N

Varianza: $\sigma^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04$

Varianza Muestra: $s^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 4 = 6.3$

Varianza

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

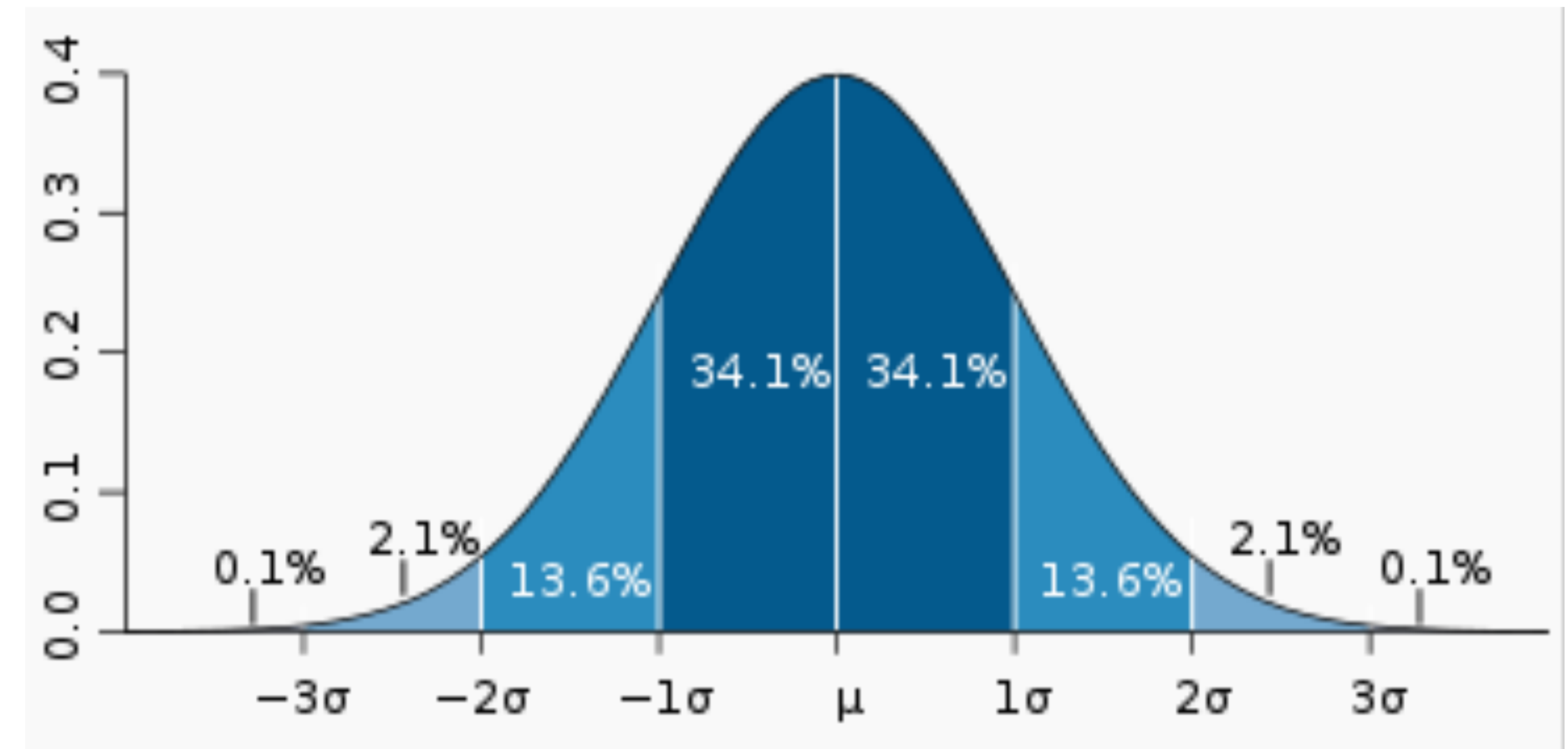
Varianza muestral

$$s^2 = \frac{\Sigma(X-M)^2}{N-1}$$

Funciones de densidad de probabilidad

Distribución Normal

Muestra la probabilidad de que un valor caiga sobre un determinado rango de valores



Percentiles

Percentiles

En una serie de datos, como se comportan los datos a un determinado porcentaje.

Example: You are the fourth tallest person in a group of 20

80% of people are shorter than you:



That means you are at the **80th percentile**.

If your height is 1.85m then "1.85m" is the 80th percentile height in that group.

Momentos

Primer momento

El primer momento es la media.

Segundo momento

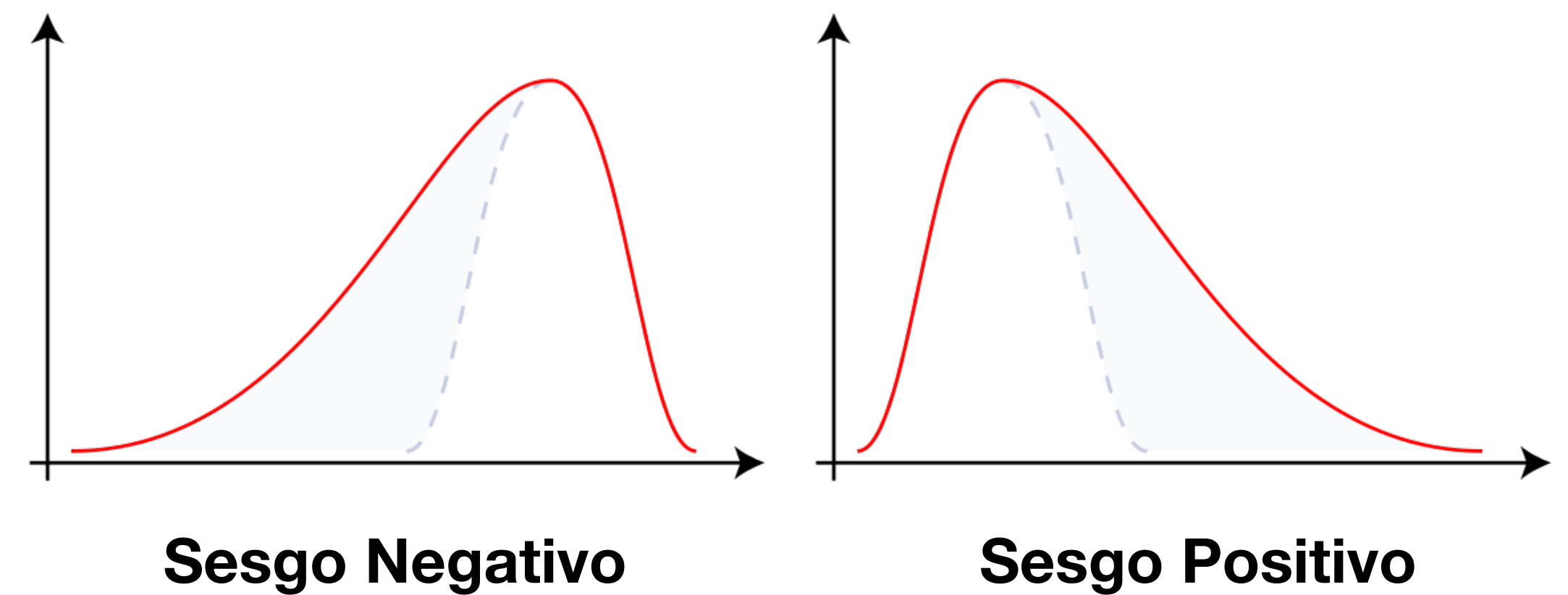
El segundo momento es la varianza

Tercer momento

El tercer momento es el sesgo de la distribución

Una distribución con un sesgo a la izquierda, tendrá un sesgo negativo.

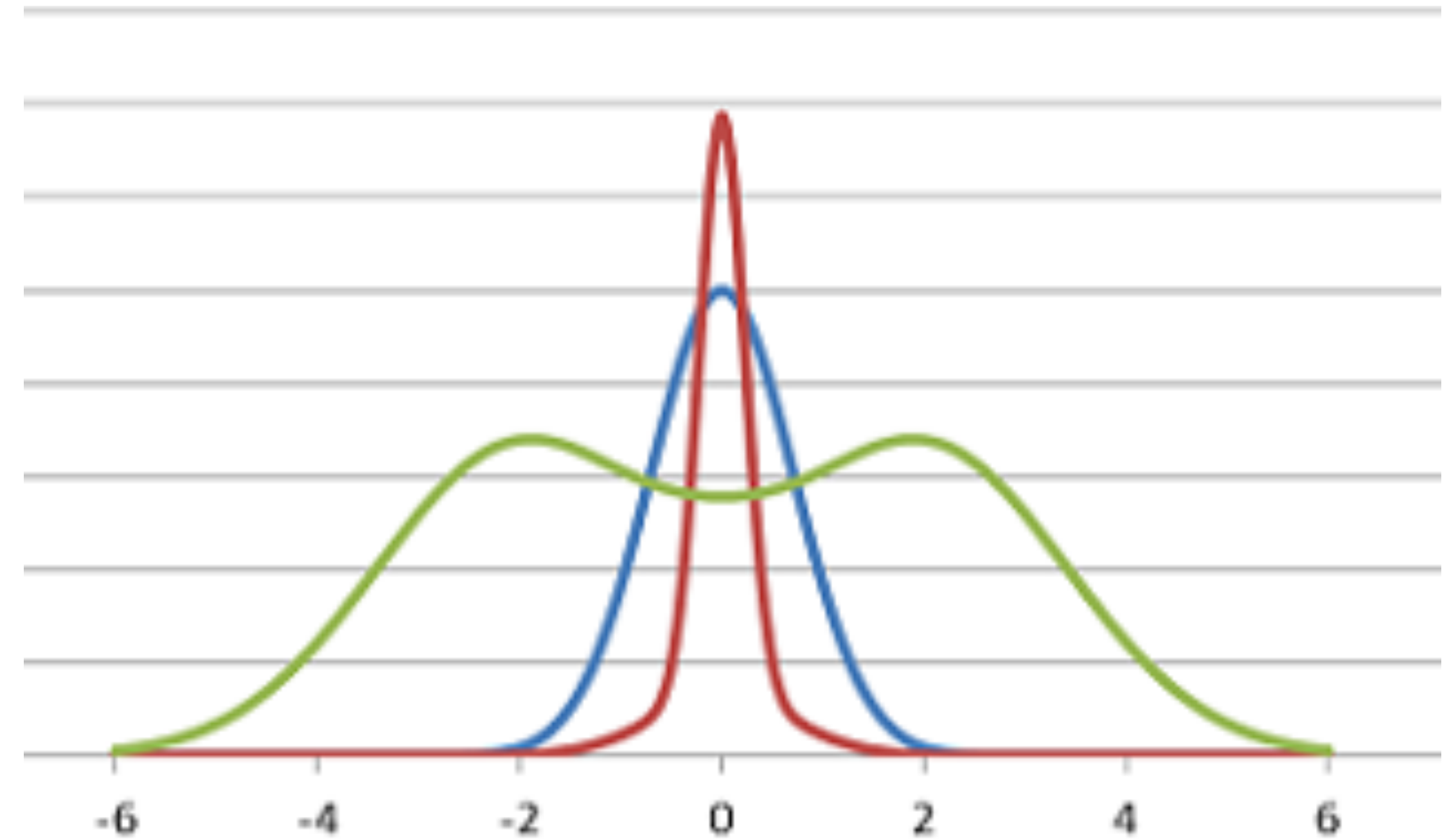
Una distribución con un sesgo a la derecha, tendrá un sesgo positivo



Cuarto momento

El cuarto momento es la kurtosis, nos permite identificar lo ancho o puntiagudo de la curva comparado contra una distribución normal.

Ejemplo: una curva muy puntiaguda indica una alta kurtosis.



Distribución Normal

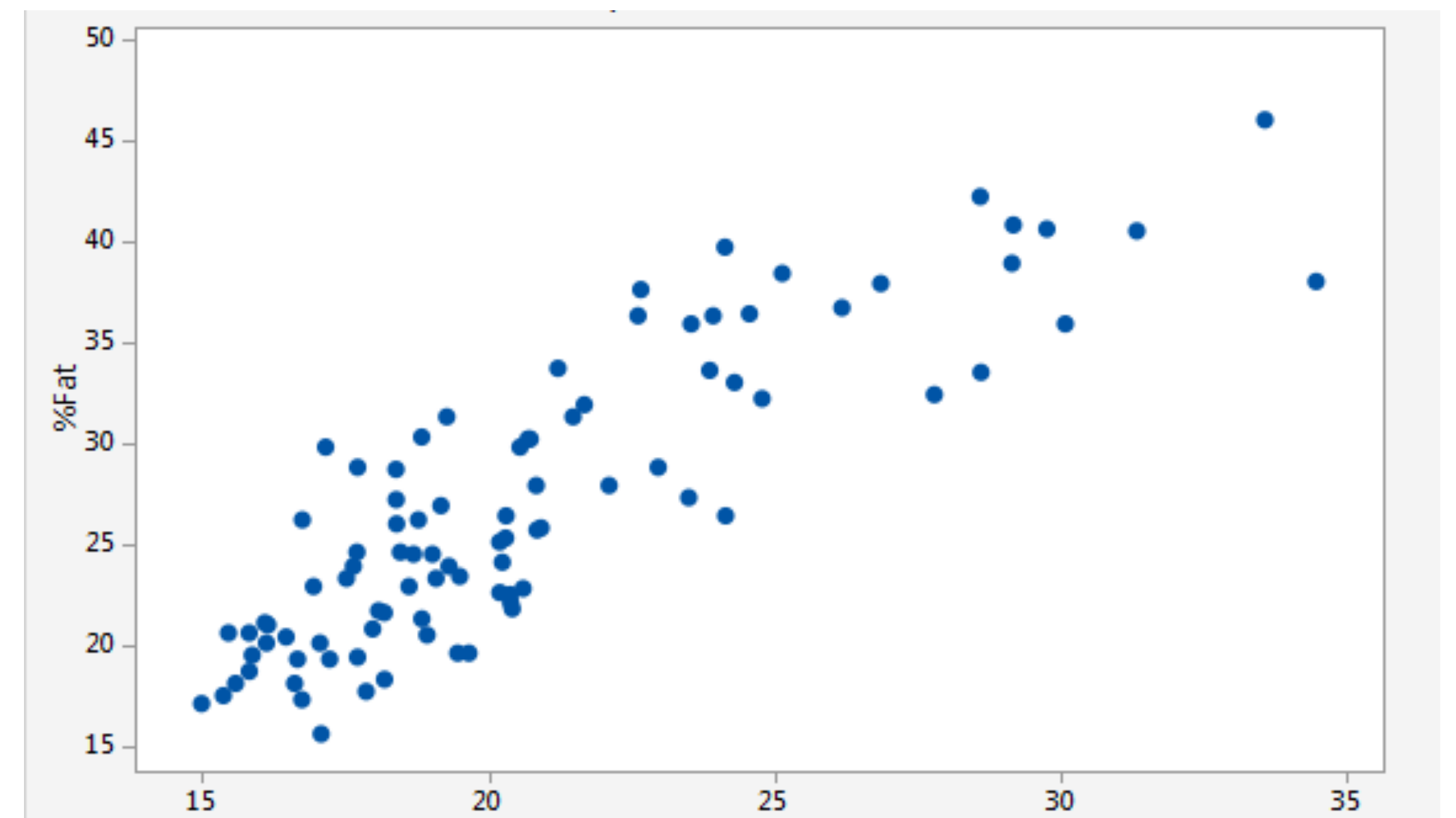
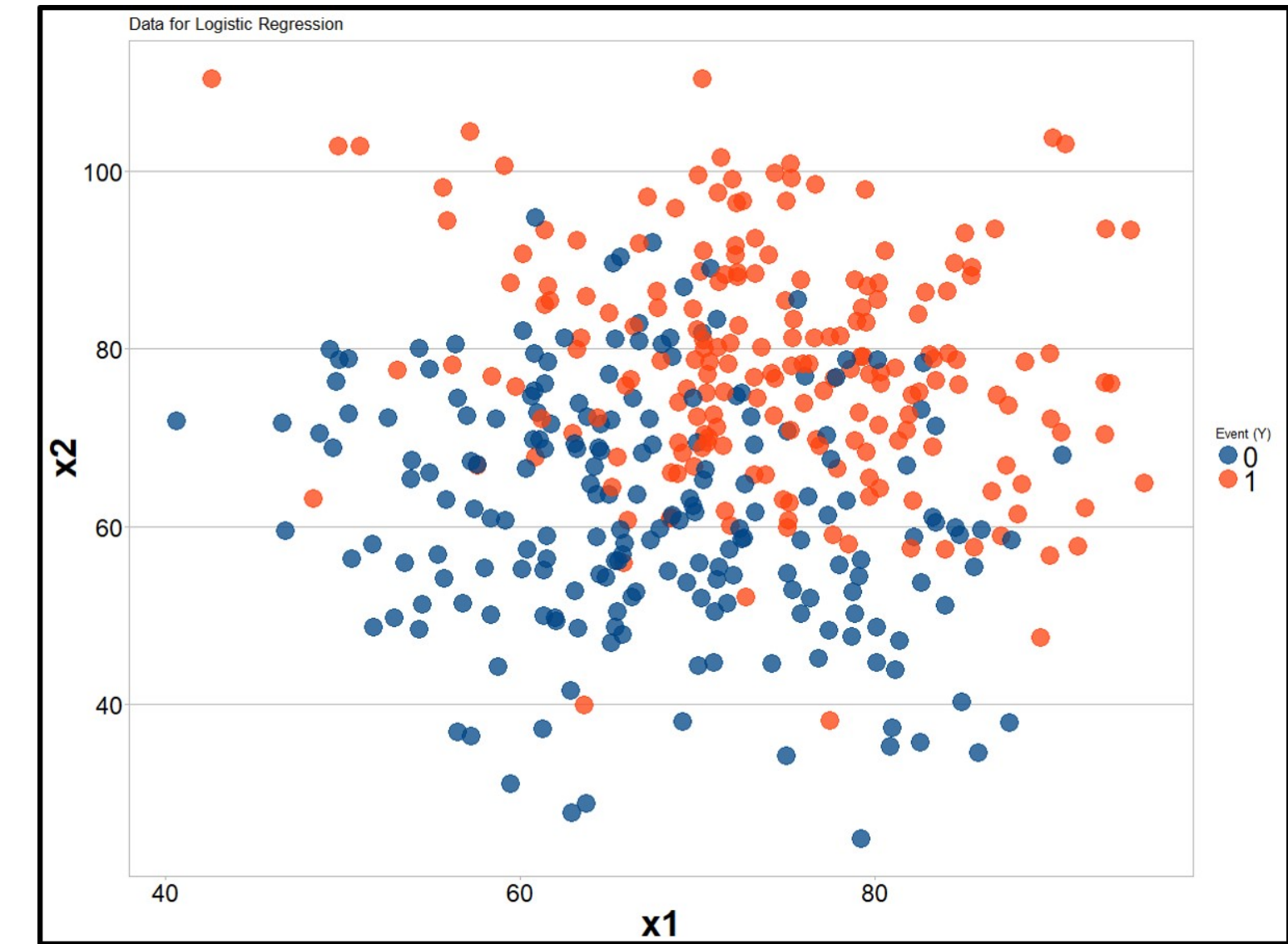
Kurtosis positiva

Kurtosis negativa

Covarianza y correlación

Covarianza

Mide como dos variables varían entre sí desde sus medias



Interpretación

Interpretar la covarianza no es fácil, hay que un valor cercano a 0, nos indica que no hay mucha correlación entre las dos variables.

Un valor lejano a 0 nos puede indicar que existe una fuerte correlación ya sea un valor positivo o negativo.

Que tan lejano a 0 es lejano?

Correlación

Debemos de dividir la covarianza entre la desviación estándar de las dos variables.

-1: significa una perfecta correlación inversa

0: no existe una correlación

1: una perfecta correlación

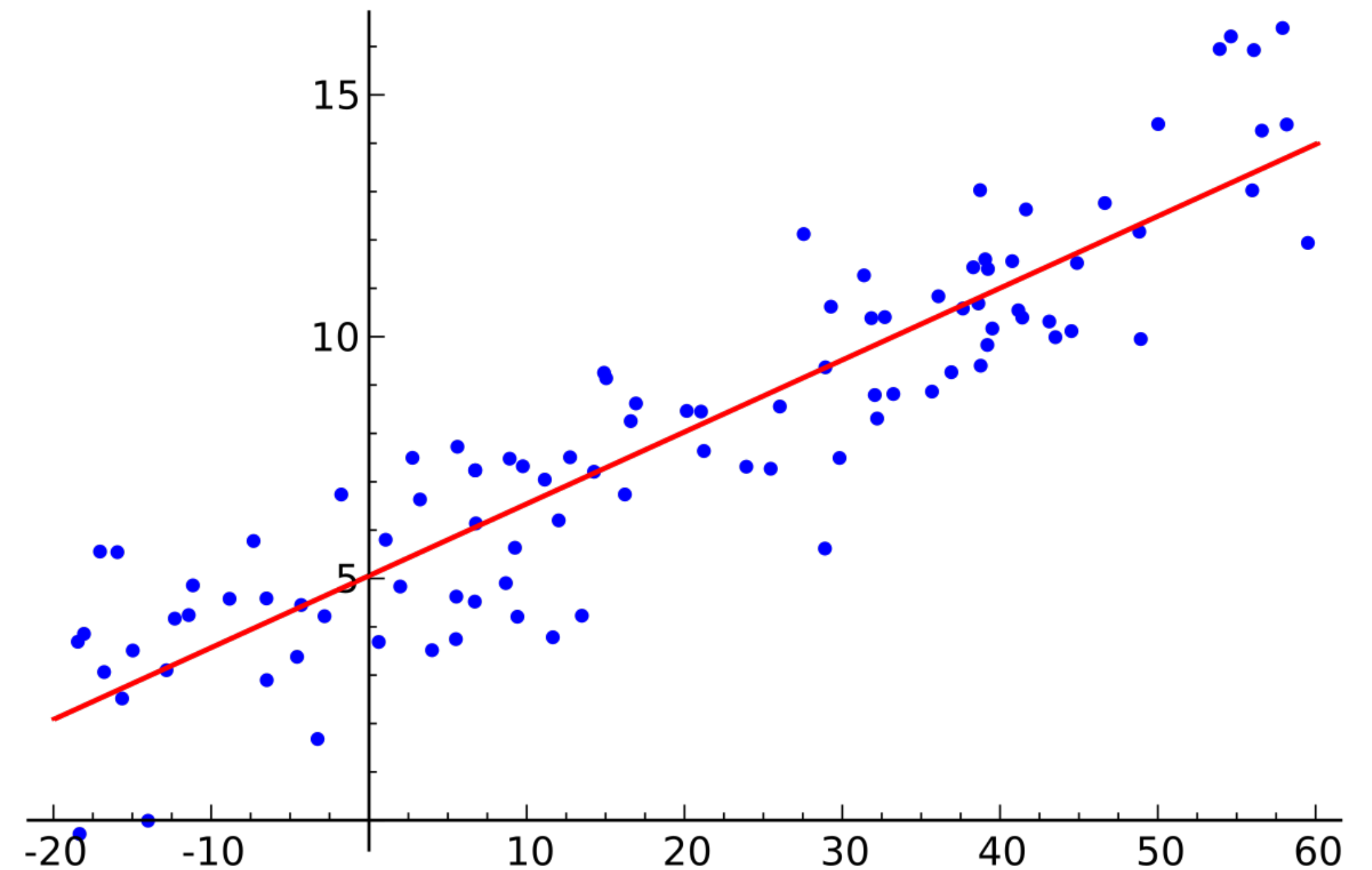
Regresión Lineal

Regresión Lineal

Ajuste de una línea a un grupo de observaciones

Se puede utilizar esta línea para predecir valores

hacia el pasado o futuro.



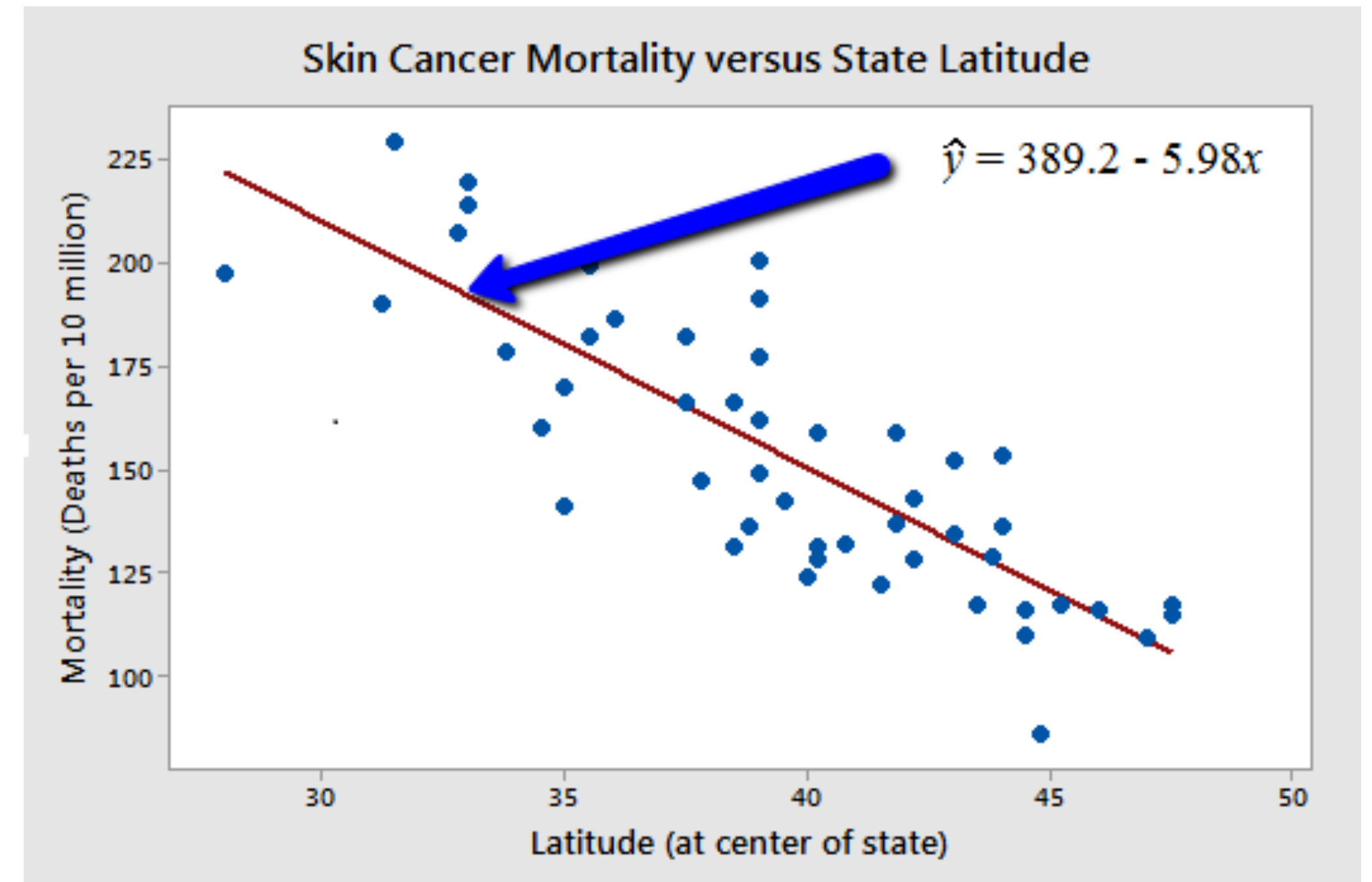
Cómo funciona?

Minimizar el error cuadrado entre cada punto y la línea

Utilizamos ecuación de la recta $y=mx+b$

La curva es la correlación entre las dos variables multiplicada por la desviación estándar en Y, todo esto dividido entre la desviación estándar en X

La intervención es la media de Y menos la recta multiplicada por la media de X



Medir el error con r-cuadrada(r-squared)

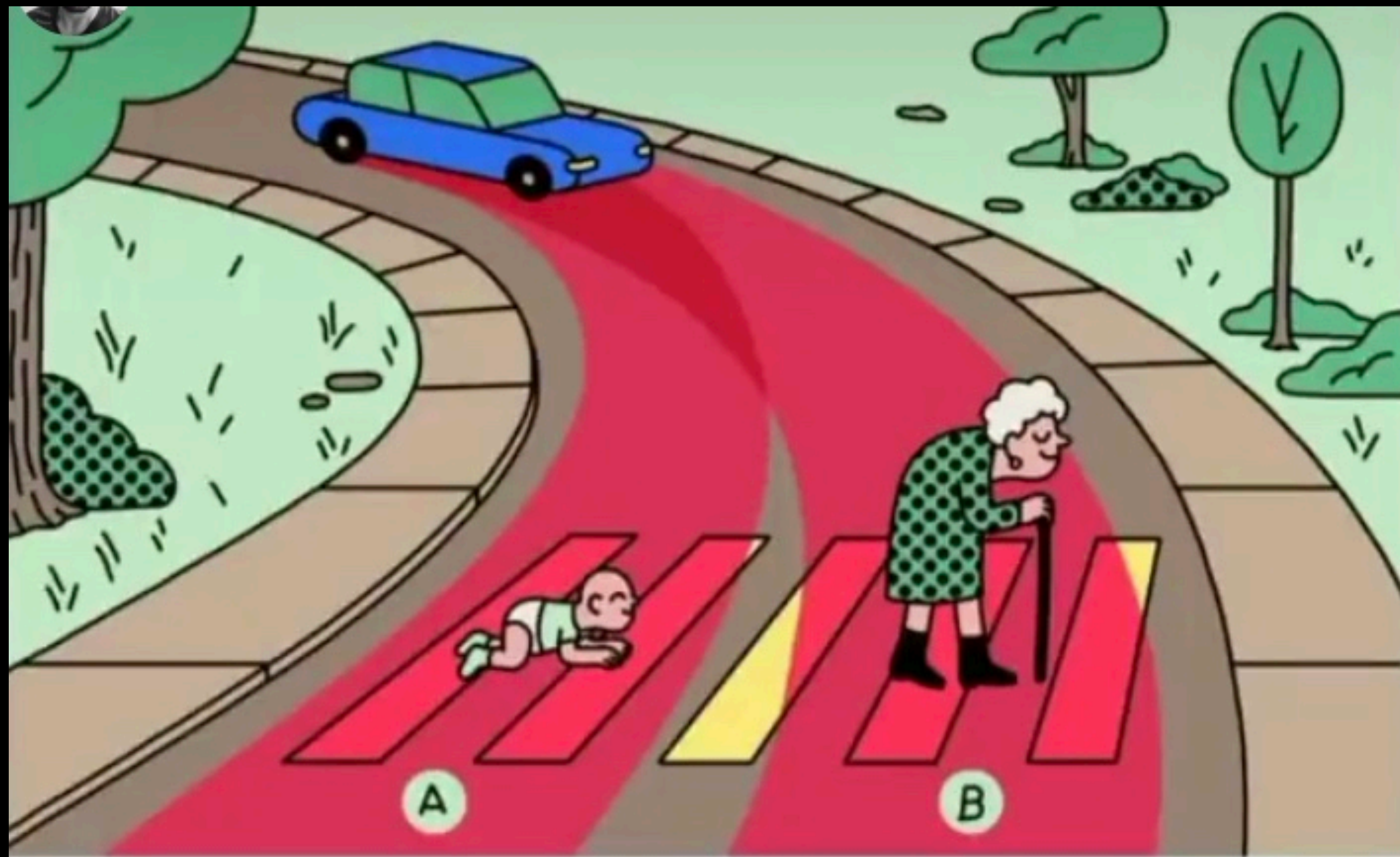
Como es que se puede medir que tan bien se ajusta la curva a nuestros datos?

r-squared es el que nos ayuda a medir.

La fracción del total de la variación en Y que es captada por el modelo.

Calcular r-squared

1.0 - Sum of squared errors / Sum of squared variation from mean



Regresión polinomial

Regresión Polinomial

No todas las relaciones son lineales

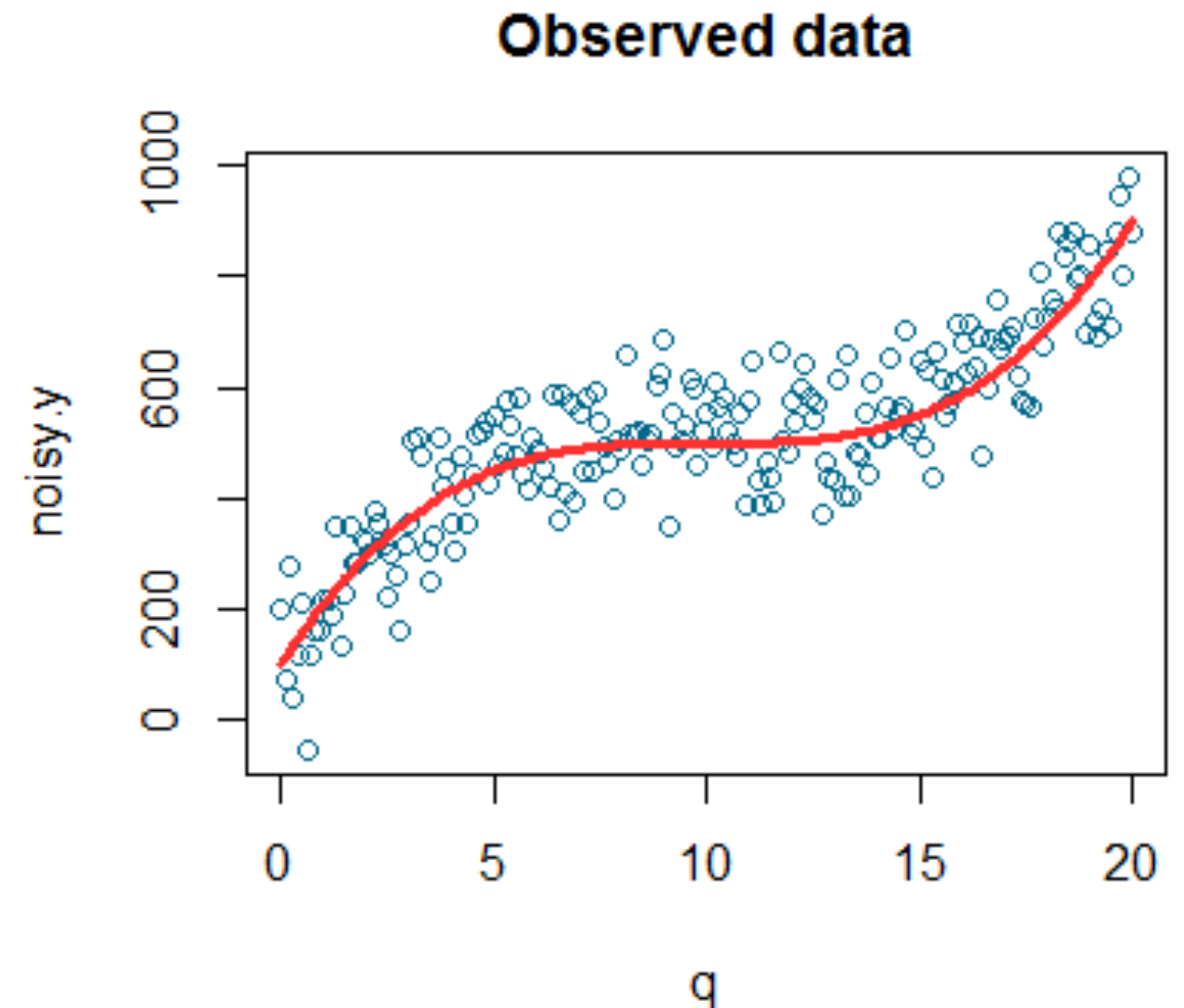
Formula lineal: $y=mx+b$

Considerada de primer orden o primer grado, ya que x está elevado a 1.

Segundo orden: $y= ax^2 + bx + c$

Tercer orden: $y = ax^3 + bx^2 + cx + d$

Ordenes más alto pueden generar curvas más complejas



No overfitting

No usar más grados de los que necesitamos

Visualizar la información para determinar el grado necesario.

Un valor alto de r -squared significa que la curva se ajusta bien a los datos de entrenamiento, pero no quiere decir que genere una buena predicción.

Regresión Múltiple

Regresión multiple

Que podemos hacer si más de una variable tiene influencia en la variable de interés?

Ejemplo: predecir el precio de un carro utilizando como base sus atributos(marca, kilometraje, tipo, etc)

Utilizando least squares

Vamos a utilizar coeficientes para cada uno de los factores, como por ejemplo:

$$\text{precio} = a + b1 * \text{kilometraje} + b2 * \text{antigüedad} + b3 * \text{puertas}$$

Se puede utilizar r-squared para medir su ajuste

Se debe de asumir que los diferentes factores no dependen unos de otros.

Contacto

IIS Jorge Ernesto Mauricio Ruvalcaba

jorge.ernesto.mauricio@gmail.com

Github: @jorgemauricio

Muchas gracias!