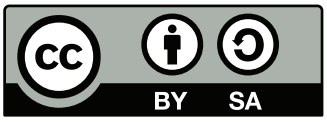
Universidad Complutense de Madrid

Laboratorio de Computación Científica
Laboratory for Scientific Computing
Introducción a Matplotlib ※ Introduction to Matplotlib

Juan Jiménez
Héctor García de Marina
Lía García

16 de junio de 2025



Índice general

Índice de figuras

Índice de cuadros

Capítulo/Chapter 1

Tratamiento estadístico de datos Statistical data analysis

Dicen que las buenas mozas en Madrid han decidido, el gastar en vez de lengua una espada de dos filos.
Y si hay guerra en este invierno , los Valones y los Suizos llevarán en vez de espada guardapiés y rebocillo.

El barberillo de Lavapiés. Luis Mariano de Larra

1.1. Secuencias de números aleatorios

Se entiende por secuencia de número aleatorios, aquella en la que no es posible encontrar relación alguna entre un número de la secuencia y los que le preceden.

No es posible general secuencias de números aleatorios con un computador. La razón es que un computador es una máquina completamente determinista; la salidas que puede producir cualquier programa son completamente predecibles. Solo los procesos físicos pueden ser realmente —intrínsecamente— aleatorios.

Sin embargo, es posible con un ordenador

1.1. Random number sequences

A random number sequence is such that it is not possible to find any relationship among a number of the sequence and the previous ones.

It is impossible to generate random numbers using a computer, because a computer is a deterministic machine and then, the outputs of any computer program are always thoroughly predictable. Only physical process could be actually –intrinsically– unpredictable.

Nevertheless, it is possible to use the computer for generating sequences of number that

generar secuencias de números que simulan ser aleatorios. Estas secuencias reciben el nombre de secuencias de números *pseudoaleatorios*.

El primer algoritmo para obtener números pseudoaleatorios, lo desarrolló John Von Newman en 1946. Se conoce con el nombre de *Middle Square*. La idea consiste en elegir un número inicial, formado por n cifras, que recibe el nombre de semilla. A continuación se eleva el número al cuadrado. Por último, se extraen las n cifras centrales del número así generado, que constituye el segundo número en la secuencia de números aleatorios. Este número se vuelve a elevar al cuadrado y se extraen de el las n cifras centrales que constituirán el tercer número aleatorio. Este proceso se repite tantas veces como números aleatorios se deseen generar.

Veamos un ejemplo con una semilla formada por $n = 8$ dígitos,

seem to be random. Such sequences are called *pseudo-random* number sequences.

The first algorithm for obtaining pseudo-random numbers was developed by John Von Newman in 1946. Its is known as the *Middle Square* method. It starts from an initial number made up of n digits which is called the seed. Then, the square of the number is calculated and the n central digits of the result are taken as a new number that will be the second one of the random number sequence. This new number is squared and the n central digit of the result are taken to form the three number of the random sequence. The process is repeated so many times as random numbers you wish to generate.

Let see an example, using a n=8 digits seed,

$$
\begin{aligned}
\textit{semilla/seed} = & 87289689 \rightarrow 87289689^2 = 7619\ 48980571\ 6721 \\
& 48980571 \rightarrow 48980571^2 = 2399\ 09633548\ 6041 \\
& 09633548 \rightarrow 09633548^2 = 0092\ 80524706\ 8304 \\
& 80524706 \rightarrow 80524706^2 = 6484\ 22827638\ 6436 \\
& 22827638 \\
& \vdots
\end{aligned}
$$

El siguiente código de Python, permite obtener una lista de números pseudoaleatorios mediante el método *Middle Square*

The next Python code allows us to get a list a pseudo-random number using the Middle Square method,

middlesquare.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 26 15:20:07 2025
Middle Square method to generate psudo-random numbers
@author: chapgpt
@Juan: Beware: the function do not check the number of digits of the seed.
So is up to the user to introduce a seed with the proper size
"""
def middle_square(seed, num_digits, num_iterations):
    # Initialize the current number with the seed
    current_number = seed

    # List to store the random numbers generated
    random_numbers = []
```

```
16
17      for _ in range(num_iterations):
18          # Square the current number
19          squared = current_number * current_number
20
21          # Convert squared number to string to extract middle digits
22          # Calculate the number of digits in the squared result
23          squared_str = str(squared)
24
25          # Calculate the start and end index for the middle digits
26          mid_start = (len(squared_str) - num_digits) // 2
27          mid_end = mid_start + num_digits
28
29          # Extract the middle digits (as integer)
30          middle_digits = int(squared_str[mid_start:mid_end])
31
32          # Add the middle digits to the list of random numbers
33          random_numbers.append(middle_digits)
34
35          # Update current_number to the middle digits for the next iteration
36          current_number = middle_digits
37
38      return random_numbers
39
40  # Example usage
41  seed = 3708  # Initial seed
42  num_digits = 4  # Number of digits to extract as the random number
43  num_iterations = 10  # Number of iterations to generate
44
45  random_sequence = middle_square(seed, num_digits, num_iterations)
46  print("Random Sequence:", random_sequence)
```

Las secuencias generadas mediante el método *middle square* no son aleatorias, cada número viene completamente determinado por el número anterior. Sin embargo, dichas secuencias parecen aleatorias. La figura 1.1 muestra una secuencia de 100 valores pseudoaleatorios generados mediante *middle square*. Aparentemente, cada valor parece no guardar ninguna relación con los anteriores.

El método descrito, presenta sin embargo serios inconvenientes: Si la semilla contiene ceros o unos a la derecha, estos tienden a repetirse indefinidamente. Además todos los métodos inducen ciclos que hacen que los números generados comiencen a repetirse periódicamente. Por ejemplo, si empezamos con la semilla de cuatro dígitos 3708, La secuencia cae, tras generar los primeros cuatro números aleatorios, en un ciclo en el que los cuatro siguientes

Sequences generated using the middle square method are not random, each number is thoroughly determined by the previous one. Nevertheless, these sequences seems random. Figure 1.1 shows a sequence of 100 pseudo-random values generated using the middle square. Apparently, each value doesn't seem to have any relationship with the previous ones.

However, this method has significant drawbacks: if the seed contains zeros or ones on the right side, these values tend to repeat indefinitely. Besides, all methods induce cycles, which make the generated numbers begin to repeat periodically. For instance, If we begin with the four-digit seed 3708, The sequence gets into a cycle after it generates the first four random numbers. The four subsequent numbers repeat indefinitely.

números generados se repiten indefinidamente.

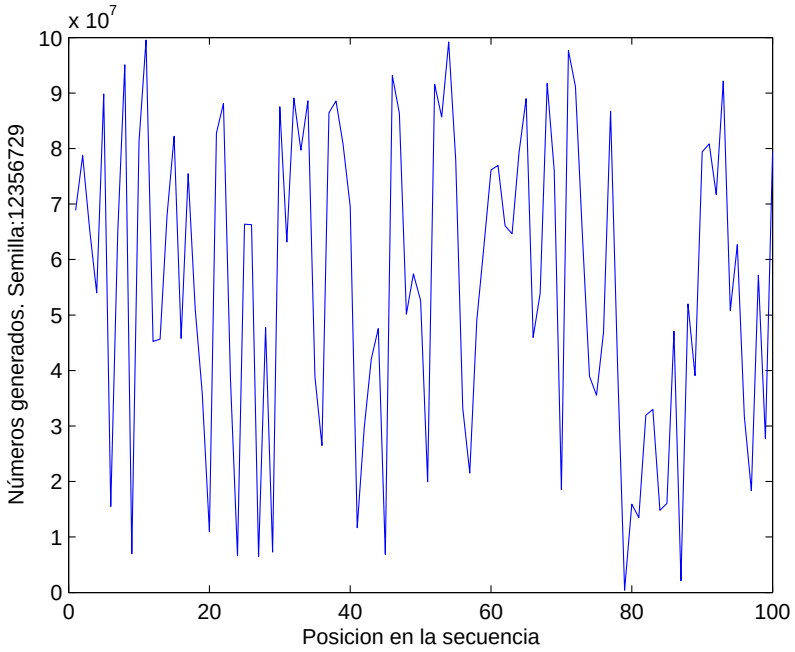


Figura 1.1: Secuencia de 100 números pseudoaleatorios generada mediante el método *middle square*.
Figure 1.1: 100 pseudo-random numbers sequence generates using the *middle square* method

$$
\begin{aligned}
semilla/seed = 3708 \rightarrow 3708^2 &= 13\ 7492\ 64 \\
7492^2 &= 56\ 1300\ 64 \\
1300^2 &= 01\ 6900\ 00 \\
6900^2 &= 47\ 6100\ 00 \\
6100^2 &= 37\ 2100\ 00 \\
2100^2 &= 04\ 4100\ 00 \\
4100^2 &= 16\ 8100\ 00 \\
8100^2 &= 65\ 6100\ 00 \\
6100^2 &= 37\ 2100\ 00 \\
2100^2 &= 04\ 4100\ 00 \\
4100^2 &= 16\ 8100\ 00 \\
&\vdots
\end{aligned}
$$

Además, en la secuencia generada puede observarse cómo los ceros situados a las dere-

Besides, it is possible to see how the the zeros locate at the right size, once they turn

cha una vez que aparecen en la secuencia, se conservan siempre.

En los algoritmos de generación de números aleatorios, la idea es obtener números en el intervalo $[0, 1]$. Para ello, se generan números entre 0 y un número natural m y luego se dividen los números generados, X_n entre m,

up, remain for ever.

Algorithms for random number generation are usually designed to generate numbers in the interval $[0, 1]$. This is carried out generating a number between 0 an a natural number m and, after, dividing the generated numbers X_n by m,

$$U_n = \frac{X_n}{m}$$

Usualmente, se elige para m un valor tan grande como sea posible. En 1948 Lehmer propuso un algoritmo conocido con el nombre de *linear congruential*. El algoritmo emplea cuatro elementos numéricos,

- I_0, Semilla. Análoga a la del método *middle square*.
- $m > I_0$, Módulo. Se elige tan grande como sea posible. Cuanto mayor es el módulo, mayor es el periodo del ciclo inducido.
- a, Multiplicador. Debe elegirse de modo que $0 \leq a < m$
- c, Incremento. Debe elegirse de modo que $0 \leq c < m$

El algoritmo de obtención de la secuencia de números pseudoaleatorios se define en este caso como,

Usually, a value as large as possible is chosen for m. In 1948 Lehmmer proposed an algorithm known as *linear congruential*. This algorithm uses four numerical items,

- I_0, Seed. Similar to *middle square* method.
- $m > I_0$, modulus. it is chossen as large as posible. The larger the modulus the lager the period of the induced cycle.
- a, Multiplier. Should be chosen to meet that $0 \leq a < m$
- c, Increment. Should be chosen to meet that $0 \leq c < m$

Then, the algorithm to obtain the sequence of random number is defined as,

$$I_{n+1} = \text{rem}(a \cdot I_n + c, m)$$

Es decir, cada número se obtiene como el resto de la división entera del producto del multiplicador por el número aleatorio anterior más el incremento, divido entre el módulo. Por tanto, se trata de un número comprendido entre 0 y $m - 1$.

El siguiente código permite calcular una secuencia de números aleatorios por el método *linear congruential*

Each number is calculated as the remainder of the product of the multiplier and the previous random number, added to the increment, all divided by the modulus. This results in a number between 0 and m-1.

The following code calculates a sequence of random numbers using the *linear congruential* method.

linearcong.py

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Feb 27 16:51:02 2025
5 Generados aleatorios linear congruential
```

```
@author: juan
"""
import matplotlib.pyplot as pl
def linear_con(i0,a,c,m,l):
    """



    Parameters
    ----------
    i0 : TYPE integer
        DESCRIPTION. seed /semilla
    a : TYPE integer
        DESCRIPTION. multiplier 0 <= a < m
    c : TYPE integer
        DESCRIPTION. Increment 0 <= c < m
    m : TYPE int
        DESCRIPTION. Module > I0 (the larger the better)
    l : TYPE int
        DESCRIPTION. number of elements in the random sequence

    Returns
    -------
    secuencia : TYPE List of double 0<=secuencia[i]<=1
        DESCRIPTION. Sequence o l pseudo-random numbers

    """
    secuencia=[]
    for i in range(1,l):
        i0 = (i0*a+c)%m
        secuencia.append(i0/m)
    return secuencia
#ejemplo/example
sec = linear_con(1000,3,10,100000,1000)
pl.plot(sec)
```

Una variante del algoritmo descrito, frecuentemente empleada, se obtiene haciendo el incremento igual a cero, $c = 0$. Dicha variante se conoce con el nombre de *multiplicative congruential*, su expresión general sería,

We can get a frequently used variation of the linear congruential algorithm taking the increment equal to zero, $c = 0$. Such variation is known as the *multiplicative congruential* algorithm; a general expression for it would be,

$$I_{n+1} = \text{rem}(a \cdot I_n)$$

Dado que el resto de la división entera de un número cualquiera entre el módulo m solo puede tomar valores enteros comprendidos entre 0 y $m - 1$, es fácil deducir que, en el mejor de los casos, podríamos obtener un ciclo de longitud m, antes de que secuencia de números aleatorios comenzara a repetirse. De ahí la importancia de elegir m lo más grande

Since the integer division of whatever number by the modulus m can yield results between 0 and $m-1$, it becomes clear that, in the best-case scenario, we can achieve a cycle of length m before the sequence of random numbers starts to repeat. Hence, the importance of choosing a large as possible value for m. On the other hand, the values taken by I_0 and c

posible. Por otra parte, los valores de I_0 e c, también influyen en la longitud de los ciclos inducidos.

Si ahora dividimos los numero aleatorios obtenidos por el modulo,

also influence the length of the induced cycles.

If we now divide the random number achieved by the modulus,

$$x_n = \frac{I_n}{m} \Rightarrow x_n \in [0, 1]$$

Los números así obtenidos pertenecen al intervalo $[0, 1]$ y están regularmente distribuidos.

En la década de los 60 del siglo pasado, La compañía IBM, desarrolló una función, conocida como *Randu*, que utilizaba el algoritmo *multiplicative congruential*. *Randu* tenía definido un multiplicador $c = 2^{16} + 3$ y un módulo $m = 2^{31}$. Se han desarrollado diferentes variante de *Randu*, mejorando progresivamente su rendimiento. Una de las mejoras introducidas fue aumentar el número de semillas empleadas,

The number obtained belong to the interval $[0, 1]$ and are regularly distributed.

In the last century sixties, IBM company developed a function known as *Randu* which employed the *multiplication congruential* algorithm. *Randu* had defined a multiplier $c = 2^{16} + 3$ and modulus $m = 2^{31}$. Several variations of *Randu* have been developed, gradually improving its performance. One of these improvements was to increase the number of seeds the algorithm uses.

$$I_{n+1} = \text{rem}(a \cdot I_n + b \cdot I_{n-1} + \cdots)$$

Este método permite obtener periodos mayores.

This method leads to get longer periods

1.1.1. El generador de números aleatorios de Numpy.

Se trata de un generador mucho mas evolucionado que los ejemplo descritos hasta ahora. Es muy versátil y se puede emplear de muchas maneras. Aquí vamos a ver solo una introducción. Para una visión detallada, lo mejor es leer la documentación de numpy, (*random sampling*).

Numpy contiene un módulo `numpy.random`, que contiene todas las funciones necesarias para generar secuencias de números aleatorios. Lo primero que debemos hacer, es crear un 'generador' (*generator*) de numeros aleatorios. La manera más sencilla de hacerlo es emplear la función `default_rng()`,

1.1.1. Numpy's random numbers generator.

The numpy's random number generator is fairly more evolved than the example generators described so far. It is highly versatile and can be use in many different ways. Here, we are going to offer only an introduction. For a detailed knowledge it is advisable to read numpy documentation, (*random sampling*).

Numpy has a module `numpy.random`, which has all the functions needed to generate sequences of random numbers. The first step is to create a *generator* of random numbers. The easiest way to do it is by using the function `default_rng()`,

```
import numpy as np

#lo primero de todo es crear un generador de números aleatorios
#si no le indicamos semilla, el la toma del sistema operativo de modo no
#determinista, es decir la secuencia de números generada no podrá volver
#a reproducirse.
```

```
gen_num = np.random.default_rng()
```

El *objeto* creado `gen_num` es un generador de números aleatorios. Lo hemos creado de la forma más sencilla posible. Como no le hemos indicado ningún valor para la semilla, el programa la toma directamente del sistema operativo de un modo aleatorio. Para obtener un número aleatorio en el intervalo $[0, 1)$, llamamos directamente al generador que hemos creado.

The *object* so created `gen_num` is a random number generator. We have create it in the most simple way. A we haven't supplied any value for the seed, the program takes it directly from the operative system in aleatory mode. To get a random number un the interval $[0, 1)$, we call the generator we have created directly,

```
In [2]: n1 = gen_num.random()
...: print('n1=',n1)
n1= 0.15431941570206298
```

Si llamamos al generador con un valor entero como variable de entrada, nos generará un array de numpy con tantos elementos como indique el valor entero,

If we call the generator with an integer number as input variable, it will generate a numpy array with the a number of elements equal to the integer supplied.

```
In [7]: v = gen_num.random(3)
In [8]: v
Out[8]: array([0.07971535, 0.38433405, 0.03229333])
```

y si introducimos una tupla o una lista con dos elementos $[m, n]$, devolverá una matriz de números aleatorios de dimensión $n \times m$,

and if we use a tuple or a list with two elements $[m, n]$, the generator will cast a matrix of random number with dimensions $n \times m$.

```
In [9]: v = gen_num.random((3,4))
In [10]: v
Out[10]:
array([[0.06821989, 0.6214247 , 0.07793149, 0.99665013],
[0.11869072, 0.6999295 , 0.8089612 , 0.02035995],
[0.67626972, 0.48921183, 0.38843264, 0.71577686]])
```

Cada vez, que ejecutamos, el generador, éste avanza en la secuencia de números aleatorios generados, a partir de la semilla de inicialización. El método empleado por el generador para obtener la secuencia de números aleatorios, se conoce con el nombre de *Permuted Congruential Generator*. Es el método por defecto, y el único que podemos emplear si creamos nuestro generador empleando la función `default_rng()`. Podemos comprobar cual es método empleado por un generador empleando la función print de python,

Any time we run the generator, it goes ahead in the sequence of random numbers generated from the initial seed. The method used by the generator to get the sequence of random numbers in known as *Permuted Congruential Generator*. It is the default method and the only one we can use if we create a generator using the `default_rng()`. We can determine which method a generator is using by utilizing Python's print function.

```
In [12]: print(gen_num)
Generator(PCG64)
```

Nos devuelve las iniciales del método y el número 64 que hace referencia al tamaño en bits de los números enteros empleados en el proceso de generación. EL periodo de este método es de 2^{128}.

Podemos controlar la secuencia de números aleatorios suministrando una semilla a la función `default_rng`, de este modo, podemos reproducir cualquier secuencia de numeros aleatorios, siempre que creemos un nuevo generador con la misma semilla,

It retrieves the acronym of the method and the number 64 which refers to the bit size of the number used in the random generation process. The period of this method is 2^{128}.

We can control the random number sequence supplying a seed to function `default_rng`; in this way, we can reproduce any sequence of random numbers, provided we create a new generator with the same seed.

```
In [48]:
...: gen_num_1 = np.random.default_rng(100)
...: #generamos una secuencia de números aleatorios
...: for i in range(10):
...:     print(gen_num_1.random())
...: #volvemos a inicializar el generador con la misma semilla
...: gen_num_1 = np.random.default_rng(100)
...: #y nos reproduce exactamente la misma secuencia
...: print('\n Secuencia repetida. Repited Sequence \n')
...: for i in range(10):
...:     print(gen_num_1.random())
0.8349816305020089
0.5965540269678873
0.2888632416912036
0.042951570694211405
0.9736543951062142
0.5964717040646884
0.7902631644187212
0.9103393812954528
0.6881544475917452
0.18999147338772315

Secuencia repetida. Repited sequence

0.8349816305020089
0.5965540269678873
0.2888632416912036
0.042951570694211405
0.9736543951062142
0.5964717040646884
0.7902631644187212
0.9103393812954528
0.6881544475917452
0.18999147338772315
```

Las posibilidades del generador de números aleatorios de numpy son muy amplias. Puede generar números aleatorios empleando dis-

Numpy's random number generator has an extensive range of possibilities. It can generate random numbers using different kinds of pro-

tintos tipos de distribuciones de probabilidad (Ver sección siguiente), permutar aleatoriamente arrays de números, seleccionar números de modo aleatorio de entre una secuencia de enteros, etc. Para conocer a fondo el proceso, se aconseja consultar la ayuda de numpy.

1.2. Probabilidad y distribuciones de probabilidad

La probabilidad es una propiedad asociada a los sucesos aleatorios. Decimos que un suceso es aleatorio cuando no depende de una causa determinista que nos permite predecir cuándo va a suceder.

1.2.1. Sucesos aleatorios discretos.

Obtener *cara* o *cruz* al lanzar una moneda al aire es un ejemplo sencillo de proceso aleatorio, ya que no es posible conocer de antemano cuál será el resultado del lanzamiento. La probabilidad da una medida de las posibilidades de que un determinado suceso aleatorio tenga lugar.

Volviendo al ejemplo de la moneda lanzada al aire, las posibilidades son dos:

1. Que salga cara.
2. Que salga cruz.

Ambos sucesos son igualmente probables tras un lanzamiento, uno de los dos debe darse necesariamente y no hay, en principio, ningún otro suceso posible. Podemos entonces asociar valores numéricos a la probabilidad con la que se dan las distintas posibilidades:

- La probabilidad de que salga cara o cruz —cualquiera de las dos— debe tener asociado el valor máximo ya que *necesariamente* ha de salir cara o cruz. Habitualmente se toma como valor máximo de un suceso aleatorio el valor 1, que estaría asociado con el suceso necesario o cierto,

$$P(\text{cara o cruz}) = 1$$

babability distributions (see the next section), permuting random number arrays, randomly choosing a number from a sequence of integers, etc. For a complete description of Numpy's random number generator, it is advisable to check Numpy's help.

1.2. Probability and probability distributions.

The probability is a property associated with random events. We say that an event is random whenever it does not depend on a deterministic cause, allowing us to forecast when the event will occur.

1.2.1. Discrete random events.

Tossing a coin to obtain either a *head* or a *tail* is a simple example of a random process, as it is impossible to know the result beforehand. Probability measures the odds of a specific event occurring.

Comming back to the coin tossing, there are two possibilities:

1. that you get head
2. that you get tail

Both events can occur with the same probability after the tossing, but, necessary, one of them has to occur, and there is not, in principle, any other possible event. We can then associate numerical values with the probability that any possible event will occur:

- The probability of getting head or tail –whatever of them– should have associated the maximum value because *necessarily* we have to get head or tail. Usually, we take as the maximum value for the probability of a random event the value 1, which would be associated with the necessary or *true* event,

$$P(\text{head or tail}) = 1$$

- The probability of not getting either heads or tails in a coin flip represents an impossible event, as we know that we must

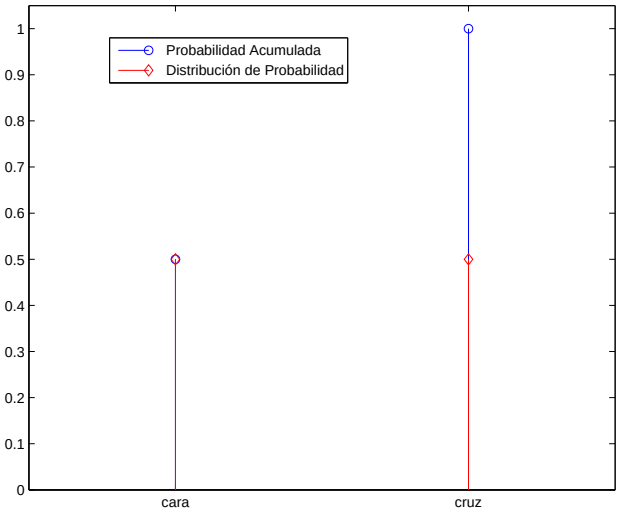


Figura 1.2: Distribución de probabilidad y probabilidad acumulada de los resultados de lanzar una moneda al aire.
Figure 1.2: Probability distribution and cumulated probability of the results of tossing a coin.

- La probabilidad de que no salga ni cara ni cruz. Puesto que hemos establecido que ha de salir necesariamente cara o cruz, estamos ante un suceso imposible. A los sucesos imposibles se les asigna probabilidad 0,

$$P(\emptyset) = 0$$

- La probabilidad de que salga cara. Como en una moneda cara y cruz son igualmente probables, la probabilidad de sacar cara, sera la *mitad* de la probabilidad de sacar cara o cruz,

$$P(\text{cara}) = \frac{1}{2} = 0.5$$

- La probabilidad de sacar cruz. De modo análogo al caso anterior será la mitad de la probabilidad de sacar cara o cruz,

$$P(\text{cruz}) = \frac{1}{2} = 0.5$$

obtain either heads or tails. Since this event cannot occur, we assign a probability value of 0 to it.

$$P(\emptyset) = 0$$

- The probability of getting head. In a *fair* coin, the probability of getting a head or tail is the same, so the probability of getting a head would be half the probability of getting a head or tail,

$$P(\text{head}) = \frac{1}{2} = 0.5$$

- The probability of getting a tail, following the same argument as in the previous case, should also be half the probability of getting a head or tail

$$P(\text{tail}) = \frac{1}{2} = 0.5$$

Lanzar una moneda o un dado, elegir una carta al azar, rellenar una quiniela, son ejemplos de fenómenos aleatorios discretos. Reciben este nombre, porque los sucesos posibles asociados al fenómeno son numerables, es decir, se pueden contar. Así por ejemplo en el lanzamiento de una moneda solo hay dos sucesos posibles (cara o cruz), en el de un dado hay seis (cada una de sus caras), en el de la elección de una carta hay cuarenta (si se trata de una baraja española) etc. El conjunto de sucesos posibles recibe el nombre de variable aleatoria.

Una distribución de probabilidad discreta, es una función que asocia cada caso posible de un fenómeno aleatorio con su probabilidad. Si vamos sumando sucesivamente las probabilidades de todos los fenómenos posibles, lo que obtenemos es una nueva función que recibe el nombre de probabilidad acumulada (Más adelante volveremos sobre esta idea). La figura 1.2 muestra la distribución de probabilidad y la probabilidad acumulada asociadas al lanzamiento de una moneda al aire.

Si todos los sucesos posibles dentro de un fenómeno aleatorio discreto tienen la misma probabilidad de suceder, es posible determinar directamente la probabilidad de cada uno de ellos sin más que dividir los casos favorables entre los casos posibles,

$$P(a) = \frac{\text{casos en que se da } a}{\text{numero total de casos posibles}}$$

Así por ejemplo la probabilidad de sacar un 6 cuando se laza un dado será,

$$P(a) = \frac{\text{casos en que se da } a = 1}{\text{numero total de casos posibles} = 6} = \frac{1}{6}$$

Una propiedad que deben cumplir los sucesos aleatorios es que la suma de las probabilidades de todos los sucesos posibles debe ser igual a la unidad,

Tossing a coin, rolling a dice, drawing a card at random, or filling in a pool coupon are examples of random discrete phenomena. They are called *discrete* because the outcomes of these activities are numerable, i.e., they can be counted. For instance, there are only to events associated to tossing a coin (head or tail), in the case of rolling a dice we have six events (any one of the dice faces), drawing a card has 40 events associated (well, it depends on the deck, 52 for a french playing card). The set of possible events of a random discrete phenomena are called random variable.

A discrete probability distribution, us a functions which associates each possible event of a random phenomena with its probability. If we add on the probabilities of every possible event, we obtain a new function called the cumulated probability (More on this later). Figure 1.2 shows the probability distribution and the cumulated probability distribution associated with tossing a coin.

If all possible events belonging to a specific random discrete phenomenon have the same probability of occurring, the probability of any of them can be determined by dividing the number of favorable cases by the number of possible cases,

$$P(a) = \frac{\text{cases where } a \text{ occurs}}{\text{total number of possible cases}}$$

So, for instance, the probability of getting a 6 when rolling a dice would be,

$$P(a) = \frac{\text{cases where } a \text{ occurs} = 1}{\text{total number of possible cases} = 6} = \frac{1}{6}$$

A property that all aleatory events should fulfill is that the sum of the probability of all possible events must equal one.

$$\sum_{i=1}^{n} P(i) = 1$$

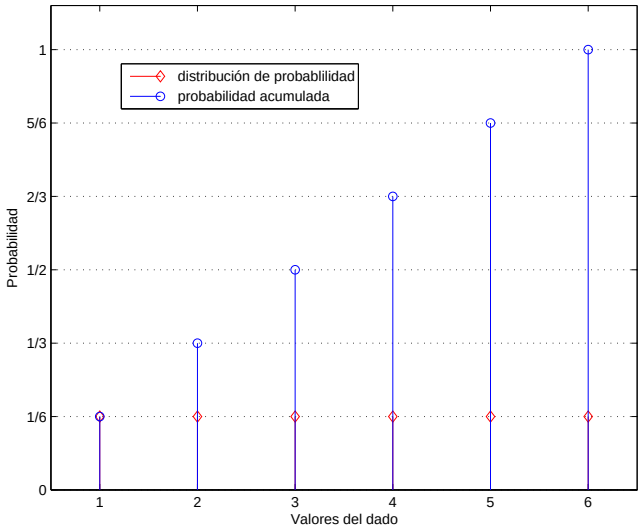


Figura 1.3: Distribución de probabilidad y probabilidad acumulada de los resultados de lanzar un dado al aire.
Figure 1.3: Probability distribution and cumulated probability of the resutls of rolling a dice.

Es decir, *necesariamente* debe darse alguno de los casos posibles.

Esto nos lleva al carácter aditivo o acumulativo de la probabilidad. Si elegimos un conjunto de casos posibles de un fenómeno aleatorio cualquiera, la probabilidad de que se de alguno de ellos será la suma de las probabilidades de los casos individuales. Por ejemplo la probabilidad al lanzar un dado de obtener un 2, un 3 ó un 6 será,

That is, we have to get *necessarily* one of the possible cases.

This brings us to an important aspect of probability: its additive or accumulative nature. If we select a set of potential events from any random phenomenon, the probability of any one of those events occurring is equal to the sum of the probabilities of each individual case. For example, the probability of rolling a 2, a 3, or a 6 on a dice is,

$$P(2,3,6) = P(2) + P(3) + P(6) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{3}{6} = \frac{1}{2}$$

lo que coincide con lo que intuitivamente cabía esperar, que la probabilidad de sacar uno cualquiera de los tres números fuera 0.5. La figura 1.3 muestra la distribución de probabilidad y la probabilidad acumulada para el lanzamiento de un dado. La probabilidad acumulada guarda relación con el carácter aditivo de la probabilidad que acabamos de describir. Si consideramos el conjunto de valores posibles en orden: $1 < 2 < 3 < 4 < 5 < 6$, la probabilidad acumulada para cada valor es la probabilidad de sacar un número al lanzar el

This result coincides with what we may intuitively expect; the probability of getting any one of the three numbers would be 0.5. Figure 1.3 shows the probability distribution and the cumulate probability for the dice-rolling example. The cumulated probability is related to the additive nature of probability we have already discussed. If we consider the set of possible outcomes in order: $1 < 2 < 3 < 4 < 5 < 6$, the cumulated probability for each value represents the likelihood of rolling a number less than or equal to this value. For ins-

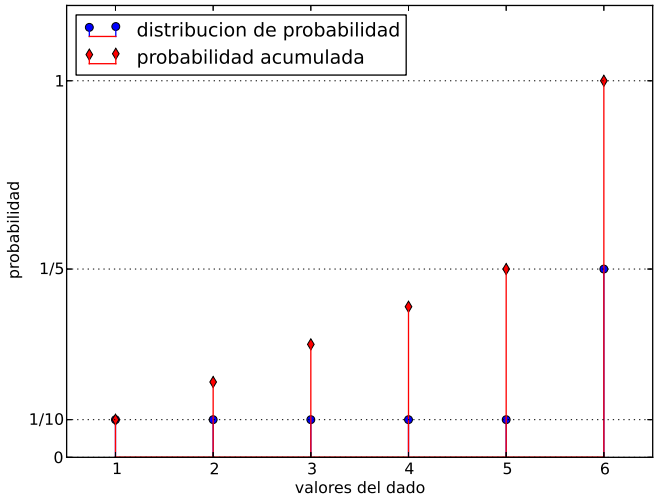


Figura 1.4: Distribución de probabilidad y probabilidad acumulada de los resultados de lanzar un dado trucado al aire.
Figure 1.4: Probability distribution and cumulated probability of the results of rolling a biased dice.

dado menor o igual que dicho valor. Así por ejemplo la probabilidad de sacar un número menor o igual que tres sería,

tance, the probability of getting a number less than or equal to three will be,

$$P(n \leqslant 3) = P(1) + P(2) + P(3) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{3}{6} = \frac{1}{2}$$

Hasta ahora, todas las distribuciones discretas de probabilidad que hemos visto —moneda, dado— asignan la misma probabilidad a todos los sucesos posibles. En general esto no tiene porqué ser así. Supongamos un dado trucado en el que la probabilidad de sacar un 6 valga 0.5, es decir en promedio tiende a salir un 6 la mitad de las veces que se lanza el dado. Supongamos además que todos los demás números salen con la misma probabilidad,

All discrete probability distributions we have encountered so far, —such as coins and dice–, assign equal probability to their possible outcomes. In general, a probability distribution do not have to do so. Consider a biased die where the probability of rolling a 6 is 0.5. This means we will likely obtain a 6 half the times we roll the die. Moreover, suppose that the remaining numbers roll with equal probability.

$$\begin{aligned} P(1) = P(2) = P(3) = P(4) = P(5) &= \frac{1}{10} \\ P(6) &= \frac{1}{2} \\ \sum_i P(i) &= 1 \end{aligned}$$

La figura 1.4 muestra la distribución de probabilidad y la probabilidad asociada al lanzamiento del dato trucado del ejemplo.

1.2.2. Distribuciones de probabilidad continuas

Hasta ahora, hemos considerado fenómenos en los que los casos posibles —la variable aleatoria— son finitos y numerables. Supongamos ahora un fenómeno aleatorio en que su variable aleatoria, x, es continua. Esto es: puede tomar todos los valores reales comprendidos en cierto intervalo, $x \in [a, b]$. Supongamos que todos los números contenidos en dicho intervalo pueden ser elegidos con igual probabilidad. Podríamos representar la distribución de probabilidad como una línea horizontal continua $f(x) = c$ que cubriera todo el intervalo $[a, b]$. Para obtener el valor constante c de dicha distribución de probabilidad bastaría *sumar* la probabilidad asociada por la distribución a cada valor del intervalo, igualar el resultado a uno y despejar c.

En realidad, no podemos sumar los infinitos valores que contiene el intervalo. De hecho lo que hacemos es sustituir la suma por una integral,

Figure 1.4 shows the probability distribution associated the example of the biased dice rolling.

1.2.2. Continuous probability distributions

So far, we have considered phenomena in which the possible cases– the random variable— are finite and enumerable. Let us suppose now a random phenomenon for which the random variable, x, is continuous. It can take every real value in an interval $[a, b]$. Suppose every number in this interval can be chosen with equal probability. We could represent the probability as a continuous horizontal line $f(x) = c$ which would cover the whole interval $[a, b]$. To obtain the constant value c for such probability distribution would be enough to sum up the probability assigned by the distribution to each value in the interval, equal the result to one, and find c.

We can not sum up the infinite values contained in the interval. In fact, we have to replace the sum with an integral.

Actually we can not sum up the infinite values contained in the interval. In fact, what we have to do is replace the sum for a in integral.

$$\int_a^b c \cdot dx = 1 \Rightarrow c = \frac{1}{b-a}$$

La figura 1.5 muestra la distribución de probabilidad resultante

Es importante darse cuenta de la diferencia con el caso discreto. El intervalo $[a, b]$ contiene infinitos números. Si tratáramos de asignar un valor a la probabilidad de obtener un número dividiendo casos favorables entre casos posibles, como hacíamos en el caso discreto, obtendríamos un valor 0 para todos los números.

La distribución de probabilidad $f(x) \geqslant 0$ representa en el caso continuo una *densidad* de probabilidad. Así, dado un número cualquiera $x_0 \in [a, b]$, el valor que toma la distribución de probabilidad $f(x_0)$ representa la probabilidad de obtener un número al azar en un intervalo dx en torno a x_0,

Figure 1.5 shows the resulting probability distribution.

It is impointronumpy.texrtant to realize that the difference when we compare with the discrete case. The interval $[a, b]$ contains infinite numbers. If we try to assign a value to the probability dividing the favorable cases by the possible cases, as we did in the discrete case, we would obtain a value 0 for every number.

The probability distribution $f(x) \geqslant 0$ represents in the continuous case the a *density* of probability. $f(x_0)$ represents the probability of obtaining a random number in an interval dx around x_0,

$$P\left(x_0 - \frac{dx}{2} \leqslant x_0 \leqslant x_0 + \frac{dx}{2}\right) = f(x_0)$$

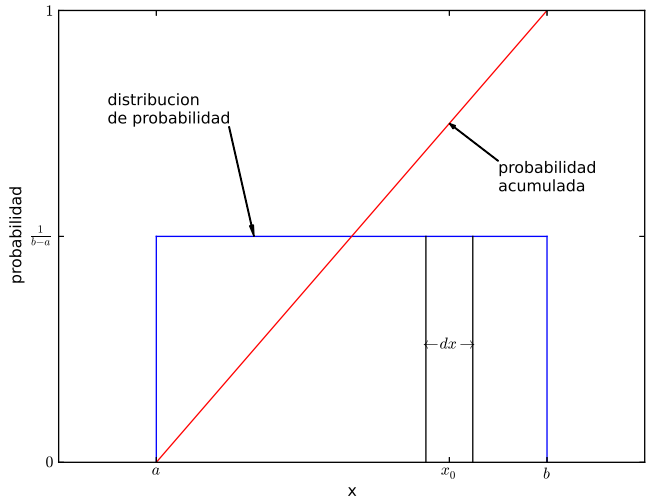


Figura 1.5: Distribución de probabilidad uniforme para un intervalo $[a, b]$ y probabilidad acumulada correspondiente.
Figure 1.5: Uniform probability distribution and cumulated probability.

La probabilidad, vendrá siempre asociada a un intervalo, y se obtendrá integrando la distribución de probabilidad en dicho intervalo. Así por ejemplo,

The probability will be always associated to an interval, and it will be obtained integrating the probability distribution on this interval. So, for instance,

$$P(x_1 \leqslant x \leqslant x_2) = \int_{x_1}^{x_2} f(x)dx$$

representa la probabilidad de obtener un número al azar en intervalo $[x_1, x_2]$. La probabilidad acumulada, se obtiene también mediante integración,

represents the probability of get a random number in the interval $[x_1, x_2]$. The cumulated probability can also be obtained by integration,

$$P(x) = \int_{a}^{x} f(x)dx$$

Para el caso de la distribución de probabilidad constante descrita más arriba obtendríamos una línea recta que corta el eje de abcisas en a y el valor de ordenada 1 en b,

For the constant probability distribution described above, we will obtain a straight line that cuts the abscissa axis on a and takes the ordinate value 1 on b.

$$P(x) = \int_{a}^{x} f(x)dx = \int_{a}^{x} \frac{1}{b-a}dx = \frac{x-a}{b-a},\ x \in [a, b]$$

La figura 1.5 muestra la probabilidad acumulada para el ejemplo de la distribución uniforme.

Figure 1.5 shows the cumulated probability for the case of an uniform distribution.

Si integramos la densidad de probabilidad

If we integrate the probability density over the whole definition interval of the random va-

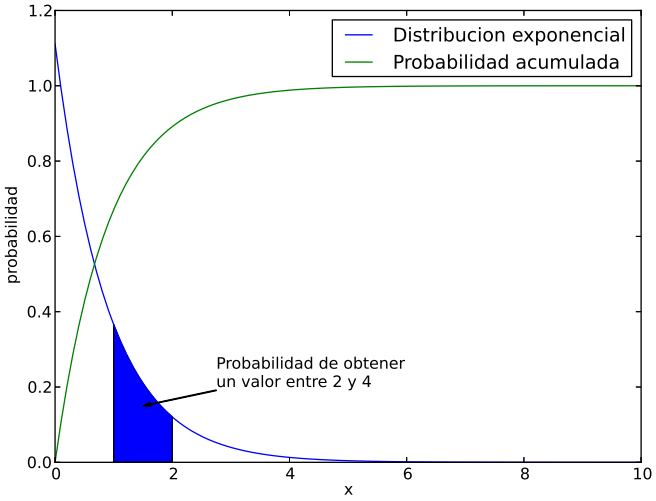


Figura 1.6: Distribución de probabilidad exponencial
Figure 1.6: Exponential probability distribution

sobre todo el intervalo de definición de la variable aleatoria, el resultado debe ser la unidad, puesto que dicha integral representa la probabilidad de obtener un número cualquiera entre todos los disponibles,

riable, the result should be one due to such integral represents the probability of getting whatever value among the available numbers,

$$\int_a^b f(x)dx = 1,\ a \leqslant x \leqslant b$$

En probabilidad se emplean muchos tipos de distribuciones para representar el modo en que se dan los suceso aleatorios en la naturaleza. A continuación presentamos dos ejemplos muy conocidos:

Distribución exponencial La distribución exponencial está definida para sucesos que pueden tomar cualquier valor real positivo ó 0, $x \in [0, \infty)$. se representa mediante una función exponencial decreciente,

We use many different kind of distributions to represent how random events take place in nature. Let see now two very well known examples:

Exponential distribution Exponetial distribution are define for events that can take a real positive value or 0, $x \in [0, \infty)$. It is represent by a decreasing exponetial function,

$$f(x) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

La figura 1.6 muestra un ejemplo de distribución exponencial. Es interesante observar como su probabilidad acumulada tiende a 1 a medida que x tiende a ∞.

Figure 1.6 shows an example of an exponential distribution. It is interesting to notice how its cumulative probability tends to 1 as x tends to ∞.

$$P(0 \leqslant x < \infty) = \int_0^\infty \frac{1}{\mu} e^{-\frac{x}{\mu}} dx = 1$$

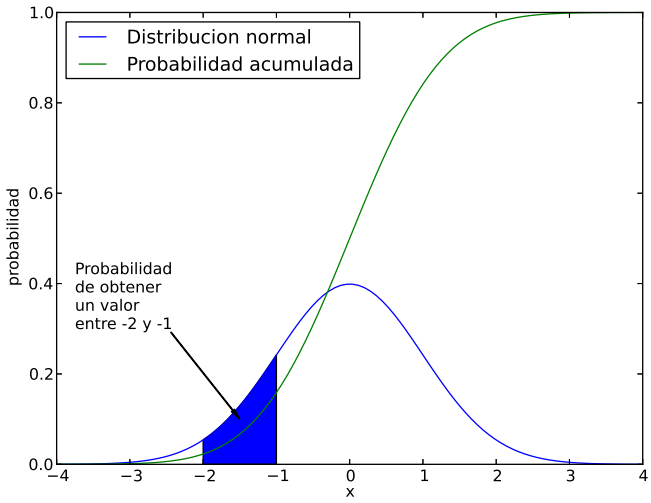


Figura 1.7: Distribución de probabilidad normal
Figure 1.7: Normal probability distribution

Distribución Normal Aparece con gran frecuencia en la naturaleza. En particular, como veremos más adelante, está relacionada con la incertidumbre inevitable en la medidas experimentales. Esta definida para sucesos aleatorios que pueden tomar cualquier valor real, $-\infty < x < \infty$. Se representa mediante la función de Gauss,

Normal distribution It is very frequent to find this distribution in nature. It is related, as we shall see later, with the uncertainty always present in experimental measurements. It is defined for random events that can take any real value, $-\infty < x < \infty$. It is represented by Gauss' function.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

La figura 1.7 muestra un ejemplo de distribución normal.

Figure 1.7 shows an example of normal distribution.

Parámetros característicos de una distribución Los parámetros característicos permiten definir propiedades importantes de las distribuciones de probabilidad. Nos limitaremos a definir los dos más importantes:

- **Media ó valor esperado.** El valor esperado de una distribución se obtiene integrando el producto de cada uno de los valores aleatorios sobre los que está definida la distribución, por su densidad de probabilidad,

Characteristic parameters of Distributions The characteristic parameter allow us to define important properties of probability distributions. We will define only the two main distributions parameters:

- **Distribution mean or expected value** We get the expected value of a distribution integrating the product of the ramdom variable the distribution is defined on, by its probability density,

$$\mu = \int_a^b x f(x) dx, \ a \leqslant x \leqslant b$$

Así, para una distribución uniforme definida en un intervalo $[a, b]$,

So, for an uniform distribution defined in the interval $[a, b]$

$$\mu = \int_a^b x \frac{1}{b-a} dx = \frac{a+b}{2}$$

para una distribución exponencial,

for an exponential distribution

$$\mu = \int_0^\infty x \frac{1}{\mu} e^{-\frac{x}{\mu}} dx = \mu$$

y para la distribución normal,

and for a normal distribution,

$$\mu = \int_{-\infty}^\infty x \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \mu$$

Es interesante observar como en el caso de las distribuciones exponencial y normal la media es un parámetro que forma parte de la definición de la función de distribución.

It is interesting to notice how for the cases of the exponential and normal distribution the mean is a parameter that is included in the definitions of the distribution function,

- **Varianza** La varianza da una medida de la dispersión de los valores de la distribución en torno a la media. Se define como,

- **Variance** The variance supplies an measurement of the distribution values dispersion around the mean. It is defined as,

$$\sigma^2 = \int_a^b (x-\mu)^2 f(x) dx, \ a \leqslant x \leqslant b$$

Para una distribución uniforme definida en el intervalo $[a, b]$, tomará el valor,

For an uniforme distribution defined in the interval $[a, b]$ it will take the value,

$$\sigma^2 = \int_a^b (x-\mu)^2 \frac{1}{b-a} dx = \frac{(b-a)^2}{12}$$

para una distribución exponencial,

for an exponential distribution,

$$\sigma^2 = \int_0^\infty (x-\mu)^2 \frac{1}{\mu} e^{-\frac{x}{\mu}} dx = \mu^2$$

y para la distribución normal,

and for a normal distribution,

$$\sigma^2 = \int_{-\infty}^\infty x \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \sigma^2$$

- **Desviación estándar.** Es la raíz cuadrada de la varianza: $\sigma = \sqrt{\sigma^2}$

- **Standard deviation** It is the variance square root : $\sigma = \sqrt{\sigma^2}$

1.3. El teorema del límite central

El teorema del límite central juega un papel muy importante a la hora de analizar y evaluar resultados experimentales. Nos limitaremos a enunciarlo, pero no daremos una demostración.

Supongamos que tenemos un fenómeno aleatorio continuo que viene caracterizado por una variable aleatoria x. Además el fenómeno aleatorio viene descrito por una distribución de probabilidad arbitraria $f(x)$ de media μ y varianza σ^2.

Supongamos que realizamos un experimento consistente en obtener n valores de x al azar. Los valores así obtenidos, deberán seguir la distribución de probabilidad $f(x)$. Para caracterizar nuestro experimento, calcularemos la media de los n valores obtenidos,

$$\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Si repetimos el experimento muchas veces, obtendremos al final una colección de medias; tantas, como veces hemos repetido el experimento, $\{\bar{x}_{n1}, \bar{x}_{n2} \cdots \bar{x}_{nj} \cdots\}$. El teorema del límite central, establece que las medias así obtenidas constituyen a su vez valores de una variable aleatoria que sigue una distribución normal, cuya media coincide con la media μ de la distribución original $f(x)$ y cuya varianza coincide con el valor de la varianza de la distribución original σ^2 dividida por el número n de valores de x obtenidos al azar en cada uno de los experimentos. (En todos los experimentos se obtiene siempre el mismo número de valores de la variable aleatoria x).

$$x,\ f(x), \mu, \sigma^2 \Rightarrow \bar{x}_n,\ f(\bar{x}_n) = \frac{1}{\sqrt{2\pi\sigma^2/n}} e^{-\frac{(x-\mu)^2}{2\sigma^2/n}}$$

Veamos un ejemplo para ilustrar el teorema. Hemos visto en la sección 1.1.1 que podemos construir un generador que genera números aleatorio uniformemente distribuidos en el intervalo $[0, 1)$. Podemos considerar que el ordenador genera número aleatorios que siguen aproximadamente una distribución continua

1.3. The Central Limit Theorem

The central limit theorem plays a very important role in experimental data analysis. We are going to present it without demonstration.

Suppose we have a continuous random phenomenon with is defined by a random variable x. Besides, the phenomenon is described by an arbitrary probability distribution $f(x)$, with mean μ and variance σ^2.

Suppose we carry out an experiment and obtain n random samples of x. The values so obtained should follow the probability distribution $f(x)$. Tu characterize our experiment we will calculate the mean of the n samples.

If we repeat the experiment many times, we eventually, will obtain a collection of mean values; so many as times we have repeated the experiment, $\{\bar{x}_{n1}, \bar{x}_{n2} \cdots \bar{x}_{nj} \cdots\}$. The central limit theorem establishes that the set of means so obtained are in turn values of a random variable that follows a normal distribution. The mean of such normal distributions meets the mean μ of the original distribution $f(x)$ and the its variance is equal to the variance σ^2 of the original distribution divided by the number n of values of x randomly generated in each one of the experiments. (You should obtain the same number of samples of the random variable x in every experiment).

Let see an example to show the theorem's result. We have seen in section 1.1.1 that we can build generator to obtain random numbers uniformly distributed in the interval $[0, 1)$. We can consider the computer to generate number with approximately follows a continuous distribution in this interval. We can expand

en dicho intervalo. Podemos alterar dicho intervalo multiplicando por un número cualquiera el resultado de `gen_num.random`. Así, por ejemplo,

the interval multiplying by any number the result of `gen_num.random` . So, for instance,

```
In [53]: l = 5*gen_num.random(100)
```

genera un array de 100 números aleatorios en el intervalo $(0, 5)$. Además podemos desplazar el centro del intervalo sumando o restando al resultado anterior un segundo número. Así,

generates a 100 random number array in the interval $(0, 5)$. Moreover, we can shift the center of the interval adding or subtracting a secon number. So,

```
In [53]: l = 5*gen_num.random(100)-2
```

genera un vector de 100 números aleatorios en el intervalo $(-2, 3)$.

Para comprobar el teorema del límite central, podemos construir un bucle que genere un vector de n números aleatorios en un determinado intervalo, calcule su media y guarde el resultado en un vector de medias. El siguiente código de Python, realiza dicho cálculo primero generando un vector de 10 números aleatorios ($n = 10$) y después generando un vector de 100, ($n = 100$). El programa genera en cada caso un millón de vectores distintos.

generates a 100 random numbers array in the interval $(-2, 3)$.

We can now check the central limit theorem, building a loop that generates a vector of n random number in an specific interval, calculates their mean value and saves the result in a vector of 'means'. The following python code, makes this calculation generating firs a vector of 10 random numbers ($n = 10$) and then generating a vector of 100 random numbers, ($n = 100$). IN both cases, the program generates one million of different vectors.

central_limit

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 28 16:13:56 2025
EStudiamos el teorema central del límite, generando números aleatorios
We study the centrar limit theorem using random numbers
@author: juan
"""

import numpy as np
import matplotlib.pyplot as pl
#creamos un generador de números aleatorios,
#create a random generator
#usamos una semilla para que se pueda reproducir tal cual
#we use a seed, so the secuence of random number is alwaus the same
gen_num = np.random.default_rng(1000)

#generamos un ,millón de vectores, formados por número aletorios distribuidos
#en el intervalo [-1,1). calculamos el valor medio de los 10 elementos de
#cada vector y guardamos los resultados en un nuevo vector.
#esto puede llevar un rato
#We generate a millon of vectors built from random number distributed in the
#interval [-1,1). We calculte the mean of the vector elements and save the
#result in a new vector.
```

```
#this could take time
media10 = np.zeros(1000000) #vector para guardar las medias/vector forsaving the means
for i in range(1000000):
    media10[i] = np.mean(2*gen_num.random([10,1])-1)

#repetimos el cálculo pero ahora con vectores de 100 elementos
#repeat the computing but now we take vectors of 100 elements
media100 = np.zeros(1000000) #vector para guardar las medias
for i in range(1000000):
    media100[i] = np.mean(2*gen_num.random([100,1])-1)

#creamos las distribuciones a las que pertenecen nuestras medias
#we create the distributions our means belong to
def normal(x,mu,s2):
    y = np.exp(-(x-mu)**2/2/s2)/np.sqrt(2*np.pi*s2)
    return(y)
#Para medias obtenidas de 10 medidas
#for means getting from 10 data
d10 = np.array([i for i in  np.arange(-0.9,0.9,0.01)])
#para medias obtenidas de 100 medidas
#for means getting from 100 data
d100 = np.array([i for i in  np.arange(-0.9,0.9,0.01)])
y10 = normal(d10,0,1./30)
y100 = normal(d100,0,1./300)

#creamos histogramas de 50 barras de la misma anchura
#buiding a histogram of 50 bars with the same width
counts10, bins10 = np.histogram(media10,50)
#calculamos el ancho de las barras para normalizar el histograma y así
#poder comparar con la distribución a la que pertenece

#computing the width of the bars to normalizate the histogram and so can
#compare it with the distribution it belongs to
anch10 = (max(media10)-min(media10))/50
#normalizamos la altura de las barras
#normalizing the bars height
cn10 = counts10/1000000/anch10

#same story for the other distrbutions
counts100, bins100 = np.histogram(media100,50)
anch100 = (max(media100)-min(media100))/50
cn100 = counts100/1000000/anch100

#ploting de results.
ax1 = pl.subplot(1,2,1)
ax1.hist(bins10[:-1],bins10,weights= cn10)
ax1.plot(d10,y10)
pl.xlabel('x')
pl.ylabel('Probabilidad')
ax2 = pl.subplot(1,2,2)
ax2.hist(bins100[:-1],bins100,weights= cn100)
ax2.plot(d100,y100)
```

```
77 pl.xlabel('x')
78 pl.ylabel('Probabilidad')
79
```

De acuerdo con el teorema del límite central, los dos vectores de medias obtenidos, `media10` y `media100`, deben seguir distribuciones normales tales que,

1. La media de la distribución normal debe coincidir con la media de la distribución a la que pertenecían los datos originales. Como hemos tomado datos distribuidos uniformemente en el intervalo $[-1, 1]$, la media de dicha distribución es,

According with the central limit theorem, both means vectors obtained by this procedure, `media10` y `media100`, should follow normal distributions such that,

1. The mean of the normal distribution should meet the mean of the original distribution the data belong to. as we have taken data uniformly distributed in the interval $[-1, 1]$, The mean value of this distribution is,

$$\mu = \frac{(b+a)}{2} = \frac{1+(-1)}{2} = 0$$

2. La varianza de las distribuciones normales a las que pertenecen las medias obtenidas será en cada caso el resultado de dividir la varianza de la distribución uniforme original entre el número de datos empleados para calcular dichas medias,

2. The variance of the normal distributions the means obtained belongs to, will be in each case the result of dividing the variance of the original uniform distribution by the number of data used to calculate the means,

$$\sigma^2_{10} = \frac{\sigma^2}{10} = \frac{\frac{(b-a)^2}{12}}{10} = \frac{\frac{(1-(-1))^2}{12}}{10} = \frac{1}{30}$$
$$\sigma^2_{100} = \frac{\sigma^2}{100} = \frac{\frac{(b-a)^2}{12}}{100}\frac{\frac{(1-(-1))^2}{12}}{10} = \frac{1}{300}$$

Por tanto, las distribuciones normales a las que pertenecen las medias calculadas son,

Thus, the normal distributions the means calculated belong to are,

$$f(\bar{x}_{10}) = \frac{1}{\sqrt{2\pi\frac{1}{30}}}e^{-\frac{(x-\mu)^2}{2\frac{1}{30}}},\ f(\bar{x}_{100}) = \frac{1}{\sqrt{2\pi\frac{1}{300}}}e^{-\frac{(x-\mu)^2}{2\frac{1}{300}}}$$

La figura 1.8, muestra un histograma normalizado de las medias obtenidas con el código que acabamos de describir [1]. Sobre dicho

[1] El histograma normalizado se obtiene dividiendo el número de puntos que pertenecen a cada barra del histograma entre el número total deintronumpy.tex puntos y el ancho de la barra. De esta manera, si sumamos los valores representados en cada barra multiplicados por su ancho, el resultado es la unidad. Solo normalizando el histograma es posible compararlo con la distribución a que pertenecen los datos, que cumple por definición tener área unidad.

Figure 1.8 shows a normalized histogram of the means obtained with the code described above[1]. On top of the histogram we have

[1] We can obtain a normalized histogram dividing the number of values belonging to a bar by the width of the bar and the total number of value that compose the histogram. IN this way if we sum up the values represented in each bar multiplied by the bar width the result is one. Only if we normalize the histogram can we compare it with the distribution the data belong two which, by definition should have unit area.

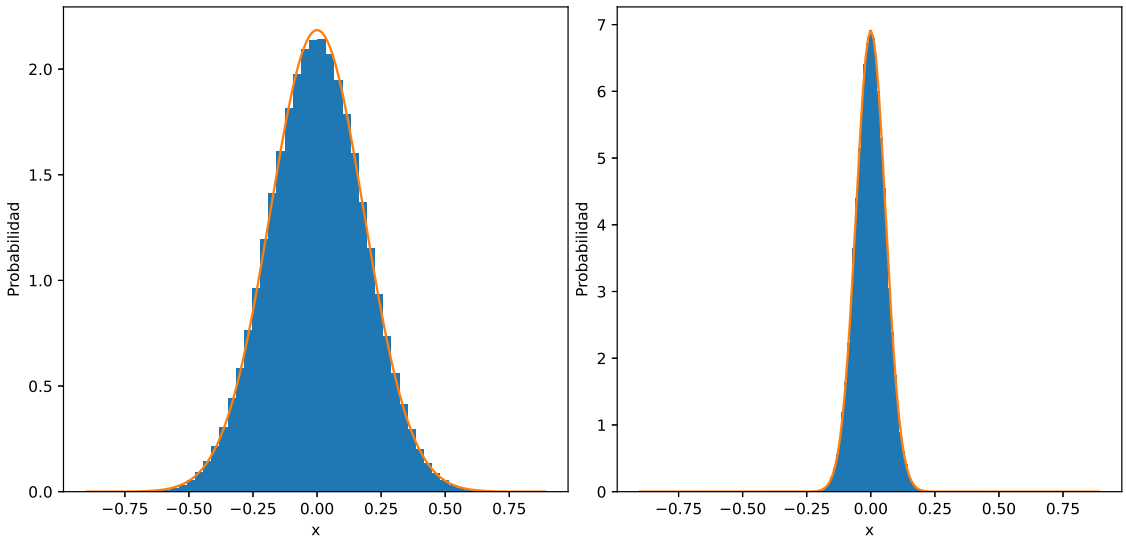


Figura 1.8: Teorema del límite central: Comparación entre histogramas normalizados para un millón de medias y la distribución normal a que pertenecen. Izquierda medias de 10 muestras. Derecha medias de 100 muestras

Figure 1.8: Central Limit Theorem: Comparison between the normalised histograms for a million of means and the normal distribution they belong to. Left, mean of 10 random samples. Right, means of 100 random samples

histograma se ha trazado mediante una línea roja, la función de Gauss que representa en cada caso la distribución normal a que pertenecen las medias. Como puede observarse en ambos casos el histograma normalizado y la distribución coinciden bastante bien.

De las figuras, es fácil deducir que cuantos más datos empleemos en el cálculo de las medias, más centrados son los resultados obtenidos en torno a la media de la distribución original. Por otro lado, esto es una consecuencia lógica del hecho de que la varianza de la distribución normal que siguen las medias, se obtenga dividiendo la varianza de la distribución original, por el número de datos n; cuantos mayor es n más pequeña resulta la varianza de la distribución normal resultante.

represented, using a orange line, The Gauss function which represent in each case the normal distribution the means belong to. As can be seen, the normalize histograms and the normal distributions feet well.

From the figures it is easy to observe that as many data are uses to calculate the means, more center are the results around the mean of the original distribution. On the other hand, this is a logical consequence of the fact that the variance of the normal distribution the means follow, are obtained dividing the variance of the original distribution by the number of data n; as larger is n smaller is the variance of the resulting normal distribution.

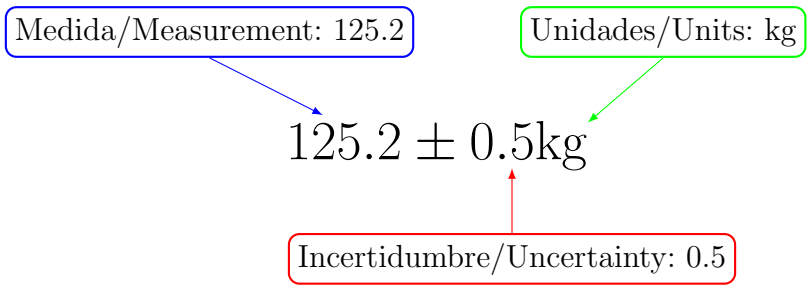


Figura 1.9: Modo correcto de expresar una medida experimental.
Figure 1.9: A right way to display an experimental measurement.

1.4. Incertidumbre en las medidas experimentales

Siempre que realizamos una medida de una cantidad física dicha medida estará afectada por un cierto error. Dado que no conocemos cual es el valor verdadero que toma la magnitud que estamos midiendo, no podemos tampoco conocer el error que cometemos al medirla. Decimos entonces que toda medida está afectada por un cierto grado de incertidumbre.

Lo que sí es posible hacer, es acotar el grado de incertidumbre de una medida, es decir estimar unos límites con respecto a la medida realizada dentro de los cuales debe estar contenido el verdadero valor que toma la magnitud medida.

Cualquier medida experimental debe incluir junto al resultado de la medida el valor de su incertidumbre y, por supuesto, las unidades empleadas en la medida. La figura 1.9 muestra un modo correcto de expresar una medida experimental, aunque no es el único. Como puede observarse, la medida y su incertidumbre se separan en la representación mediante el símbolo ±. La incertidumbre se representa como un entorno alrededor del valor medido, dentro del cual estaría incluido el valor real de la medida[2].

[2]Como se verá más adelante se debe indicar también el *grado de confianza* con el que esperamos que la medida caiga dentro del intervalo indicado por la incertidumbre

1.4. Experimental measurement uncertainty

Whenever we measure a physical magnitude, such measurement is affected by some error. As we do not know the actual value taken by the magnitude we are measuring, we cannot know the error we make when we take the measure. We say then, that every measurement is affected by some degree of uncertainty.

But we can, a least, calculate bounds on the degree of measurement uncertainty. This means we can estimate limits around the measurement that contain the actual value of the measured quantity.

Any experimental measurement should include, besides the measure's result, the value of its uncertainty and, of course, the units used in the measurement. Figure 1.9 shows the right way of expressing an experimental measurement, although it is not the only one. As seen, the measurement and its uncertainty are separated in the representation by the symbol ±. So, the uncertainty is represented as a contour around the value of the measurement; inside this contour, the actual value of the measured magnitude should be. [2]

[2]As we shall later, we should also indicate the *degree of confidence* when we expect the actual value to fall inside the interval indicated by the uncertainty.

1.4.1. Fuentes de incertidumbre.

Antes de entrar en el estudio de las fuentes de incertidumbre, es importante destacar que el estudio de la medición constituye por sí mismo una ciencia, conocida con el nombre de metrología. Lo que vamos a describir a continuación tanto referido a las fuentes de incertidumbre como al modo de estimarla, es incompleto y representa tan solo una primera aproximación al problema. La incertidumbre de una medida tiene fundamentalmente dos causas:

Incertidumbre sistemática de precisión. La primera es la precisión limitada de los aparatos de medida. Cualquier aparato de medida tiene una precisión que viene determinada por la unidad o fracción de unidad más pequeña que es capaz de resolver. Por ejemplo, una regla normal, es capaz de resolver (distinguir) dos longitudes que se diferencien en $0.5mm$, un reloj digital sencillo es capaz de resolver tiempos con una diferencia de $1s$. etc.

La incertidumbre debida a la precisión finita de los aparatos de medida recibe el nombre de *Incertidumbre sistemática de precisión.* Se llama así porque depende exclusivamente de las características del aparato de medida que, en principio, son siempre las mismas.

La incertidumbre sistemática de precisión suele expresarse añadiendo a la medida la mitad de la división mínima de la escala del aparato de medida empleado. Así, por ejemplo, si medimos $123mm$ con una regla graduada en milímetros, podríamos expresar la medida como $123 \pm 0.5mm$ o también como $12,3 \pm 0.05cm$.

La incertidumbre sistemática de precisión representa una distribución uniforme de media el valor de la medida realizada y anchura la división mínima de la escala. Es decir, se considera que el valor real de la medida está dentro de dicho intervalo intervalo con una probabilidad del 100 %. Volviendo al ejemplo anterior de la regla, el valor real de nuestra medida estaría comprendido en el intervalo [122.5, 123.5], y podría tomar cualquier valor de dicho intervalo con igual probabilidad.

1.4.1. Sources of Uncertainty

Before beginning to study the sources of uncertainty, it is important to notice that the study of the 'measure' is science, known as metrology. What we are to present in the following paragraphs, in reference to the sources of uncertainty and the means to estimate this last, is incomplete and only represents a first approach to the problem. The uncertainty of a measurement has mainly two causes:

Systematic accuracy uncertainty. The first one is the limited accuracy of the sensors. Any sensor's accuracy is determined by the least unit or fraction of the units the device is able to resolve. For instance, a standard rule can resolve (distinguish) two lengths that are different only by $0.5mm$. A simple digital watch can display times with a resolution of one second, etc.

The uncertainty due to the finite accuracy of the sensors is known as *Systematic accuracy uncertainty.* It is named so because it depends only on the features of the sensor device, which are always the same, at least in principle.

Systematic accuracy uncertainty is usually shown by adding half the least division of the Sensor device measurement scale to the measurement. So, for instance, if we are taking a measurement of $123mm$ with a rule graduated in millimeters, we can present the measurement as $123 \pm 0.5mm$ or $12,3 \pm 0.05cm$.

The systematic accuracy uncertainty represents an uniform probability density distribution. The mean is the value measured and the width is the minimum division of the sensor measurement scale. This means, that we consider that the real value of the magnitude is inside this interval (defined by the minimum division value) with a probability of 100 %. Coming back to the example of the rule, The real value of our measurement will be inside the interval [122.5, 123.5] and it could take whatever value, inside this interval, with the same probability.

Statistical uncertainty The second source of uncertainty arises from environmental fac-

Incertidumbre estadística. La segunda fuente de incertidumbre se debe a factores ambientales, que modifican de una vez para otra las condiciones en que se realiza una medida experimental. Estos factores hacen que la medidas realizadas sean sucesos aleatorios. En principio podría describirse mediante una distribución de probabilidad $f(x)$, pero en la práctica desconocemos de qué distribución se trata y ni siquiera sabemos cual es su valor esperado μ o su varianza σ.

Afortunadamente, el teorema del límite central, que describimos en al sección 1.3 proporciona un sistema de estimar la incertidumbre estadística.

Supongamos que repetimos n veces la misma medida experimental. Por ejemplo: tenemos un recipiente con agua, lo calentamos y tomamos la temperatura del agua cuando ésta rompe a hervir. Obtenemos un conjunto $\{x_1, x_2, \cdots x_n\}$ de medidas de la temperatura de ebullición del agua. Cabe esperar que, debido a la incertidumbre estadística, los valores obtenidos sean distintos y, a la vez próximos entre sí[3].

Sabemos que dichos valores deben seguir una cierta distribución de probabilidad. Podríamos estimar su valor esperado μ mediante el cálculo de la media aritmética de las medidas tomadas.

[3]De hecho, si aparecen algunos valores claramente alejados del resto, lo habitual es considerar que están afectados por algún error desconocido y desecharlos. Dichos valores suelen recibir el nombre de valores o datos aberrantes.

tors that constantly change the conditions under which a measurement is taken. These factors render the measurement subject to random variations. While we could theoretically describe this variation using a probability distribution $f(x)$, in practice, the specific distribution is unknown. Consequently, we do not know its mean value μ or its variance σ.

But, fortunately, the central limit theorem, that we described in section 1.3, supplies a method to estimate the statical uncertainty.

Let's suppose we repeat the same experimental measurement n times. For example, we have a bowl of water; we heat it and measure the water temperature when it begins to boil. We obtain a set $\{x_1, x_2, \cdots x_n\}$ of boiling water temperature measurements. Due to statistical uncertainty, it is likely that, the values we obtain will differ from one to another but they will be close together[3].

We know that this data should follow some probability distribution. We can estimate the expected value, μ, by calculating the arithmetic mean of the collected data.

[3]In fact, when we obtain data points that are significantly different from the rest, we often consider them to be influenced by some unknown error and will discard them. Such data are known as outlayers

$$\mu \approx \bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

La media así calculada se toma como el valor resultante de la medida realizada.

De acuerdo con el teorema del límite central, $\bar{x}_n$ es una variable aleatoria que debe seguir una distribución normal de media μ y de varianza σ^2/n. Como tampoco conocemos el valor de la varianza σ^2 de la distribución de probabilidad que siguen los datos medidos, la estimamos a partir de los propios datos medidos, de acuerdo con la siguiente ecuación,

We take the mean so calculated as the resulting value of the performed measure.

According to the central limit theorem, the sample mean $\bar{x}_n$ is a random variable that follows a normal distribution with a mean of μ and a variance of σ^2/n. Since we do not know the variance σ^2 of the underlying probability distribution from which the measured data is drawn, we estimate it using the measured data itself, following this equation,

$$\sigma^2 \approx s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Dividiendo s^2 por el número de muestras empleado se puede aproximar la varianza de la distribución normal que debe seguir $\bar{x}_n$,

Dividing s^2 by the number of samples, we can approximate the variance of the normal distribution that $\bar{x}_n$ follows,

$$\sigma_{xn}^2 \approx s_{xn}^2 = \frac{s^2}{n}$$

La incertidumbre, se puede asociar con la raíz cuadrada la varianza que acabamos de estimar,

We can associate the uncertainty with the square root of the variance just estimated.,

$$s_{xn} = \sqrt{s_{xn}^2} = \sqrt{\frac{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}$$

esta cantidad recibe el nombre de desviación estándar.

this quantity is known as the standard deviation.

1.4.2. Intervalos de confianza.

Como hemos visto, a partir del teorema central del límite podemos asociar la media de los valores obtenidos al repetir una medida y su desviación estándar con una distribución normal. Podemos ahora, en un segundo paso, asociar la incertidumbre de la medida realizada con la probabilidad representada por dicha distribución normal.

Si el valor obtenido tras realizar n repeticiones de una medida es $\bar{x}_n$ y su desviación típica es s_{xn} ¿Cuál es la probabilidad de que el valor medido esté realmente comprendido en el intervalo $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$?

Como vimos en el apartado 1.2.2, dicha probabilidad puede calcularse integrando la distribución normal obtenida entre $\bar{x}_n - s_{xn}$ y $\bar{x}_n + s_{xn}$,

intronumpy.tex

1.4.2. Confidence intervals.

As we have seen, using the central limit theorem, it is possible to relate the mean of the values obtained by repeating a measure and their standard deviation to a normal distribution. In the second step, we can associate the uncertainty of the performed measure with the probability represented by this normal distribution.

If the mean value after carrying out n repetitions of measurement is $\bar{x}_n$ and the standard deviation is s_{xn} What is the probability that the measured value will be actually inside the interval $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$?

As we see in section 1.2.2, such probability can be calculated by integrating the normal distribution obtained between the limits $\bar{x}_n - s_{xn}$ y $\bar{x}_n + s_{xn}$,

$$P(\bar{x}_n - s_{xn} \leqslant x \leqslant \bar{x}_n + s_{xn}) = \frac{1}{\sqrt{2\pi\sigma_{xn}^2}} \int_{\bar{x}_n - s_{xn}}^{\bar{x}_n + s_{xn}} e^{\frac{-(x-\bar{x}_n)^2}{2s_{xn}^2}}$$

El valor de dicha integral es 0.6827. Si expresamos este valor como un tanto por ciento, concluimos que la probabilidad de que el valor medido este en el intervalo $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$ es del 68.27 %. Dicho de otra manera: el intervalo $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$ corresponde con el *intervalo de confianza* del 68.27 %. La figura 1.10 muestra el área bajo la distribución normal, correspondiente a dicha probabilidad.

Volviendo al problema de la medida, la

The value taken by this integral is 0.6827. If we represent this value using a percentage, We can conclude that the probability of the mean value falling within the interval $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$ is a 68.27 %. In other words, the interval $[\bar{x}_n - s_{xn}, \bar{x}_n + s_{xn}]$ represents a confidence interval of 68.27 %. Figure 1.10 shows the area beneath the normal distribution corresponding to this probability.

Coming back to the measurement problem,

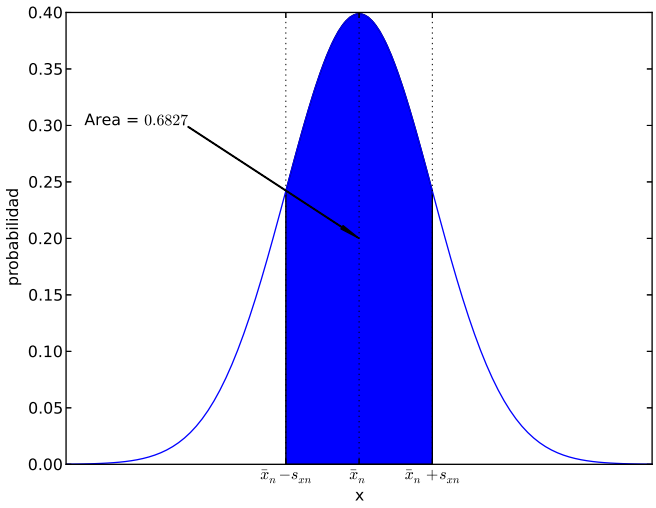


Figura 1.10: Intervalo de confianza del 68.27 %

manera correcta de expresarla sería en nuestro caso: $\bar{x} \pm s_{xn}$ (unidades). Indicando que la incertidumbre corresponde con el intervalo de confianza del 68.27 %.

Observando la figura 1.10, es fácil plantearse la siguiente cuestión: ¿Qué valor de la incertidumbre contendría el valor de real de la medida con una probabilidad de, por ejemplo, el 95 %? O, dicho de otra manera, ¿Cuánto tengo yo que *alargar* el intervalo de integración para que el área contenida bajo la distribución normal valga 0.95?

Para estimarlo se emplea la inversa de la función de probabilidad acumulada. Como se explicó en el apartado 1.2.2, la función de probabilidad acumulada se obtiene por integración de la distribución de probabilidad correspondiente.

Dicha función representa la probabilidad de obtener un resultado entre el límite inferior para el que está definida la distribución de probabilidad y el valor x para el que se calcula la función. La inversa de la función de probabilidad acumulada nos indica, dado un valor de la probabilidad comprendido entre 0 y 1, a qué valor x corresponde dicha probabilidad acumulada.

Dado que la distribución normal no tiene

the appropriate way to represent it in our example would be: $\bar{x} \pm s_{xn}$ (units). Indicating that the uncertainty corresponds to a confidence interval of 68.27 %.

When examining figure 1.10, one might wonder: What uncertainty value will encompass the true measurement value with a 95 % probability? In other words, how much do we need to *stretch* the integration interval to achieve an area of 0.95 beneath the normal distribution?

We can estimate this interval by using the inverse of the cumulated probability distribution, as we explained in section 1.2.2. We obtain the cumulated probability distribution by integrating the corresponding probability density distribution.

The cumulative probability function provides the probability of obtaining a result that falls between the lower limit of the probability distribution and a specific value, x, for which we are calculating the function's result. The inverse of the cumulative probability function determines which value x corresponds to a given cumulative probability, where that probability is a value between 0 and 1.

The normal function has not primitive and it can only be integrated numerically. Thus, its

primitiva, solo puede integrarse numéricamente. Por tanto, su inversa, solo puede obtenerse también numéricamente.

Hay varias posibilidades en Python para obtener tanto los valores de la distribución normal acumulada como los de su inversa. Todos ellos forman parte de módulos especializados en realizar cálculos estadísticos. En esta notas nos vamos a centrar en el uso del módulo Scipy y, en particular, su submódulo stats. Se trata de un módulo especializado en cálculo estadistico, que implementa un buen número de funciones de distribución.

En nuestro caso particular, nos centraremos en la distribución normal. Podemos emplearla de muchas maneras, la más sencilla es importarla directamente desde su módulo, `from scipy.stats import norm`.

`norm` suministra todas las funciones necesarias para trabajar con distribuciones normales. Por defecto, la distribución normal con la que trabaja es la estádar, es decir una distribucion de media 0 y desviación estandar 1. Si queremos cambiar los valores de media y desviaciación estándar, podemos hacerlo empleando los parámetros `loc` (media) y `scale` (desviación estándar). Entre las funciones sumnistradas, nos vamos a centrar de momento en las tres principales,

- `norm.pdf(x,loc=0,scale=1)`, suministra valores de la distribucion normal.
- `norm.cdf(x,loc=0,scale=1)`, suministra valores de la distribución normal acumulada
- `norm.ppf(p,loc=0,scale=1)`, $0 \leq q \leq 1$, suministra la inversa de la distribución normal acumulada

veamos algunos ejemplos del uso de estas funciones,

inverse, can only be obtain also by numerical methods.

There are several means in Python to calculate the values of the normal cumulative distribution and the values of its inverse function. All this methods belong to Python modules devoted to make statistical calculations. In this notes we are going to focus on Scipy module and, specifically in Scipy's submodule stats; a submodule specifically designed to perform statistical computing, which implement a large number of distributions functions.

In our case, we are going to focus on the normal distribution. we can use it in many different ways but perhaps the simplest is just to import in from its module,
`from scipy.stats import norm`.

`norm` supplies all necessary functions to work with normal distributions. It works by default with an standard normal distribution, i.e. a normal distribution of mean 0 and standard deviation 1. If we want to change the values of mean and standard deviation, we can do it using the parameters `loc` (mean) y `scale` (standard deviation). Between the function supplied by `norm` we are going to focus now on the three main ones,

- `norm.pdf(x,loc=0,scale=1)`, it supplies normal distribution values .
- `norm.cdf(x,loc=0,scale=1)`, it supplies normal cumulated distributions values.
- `norm.ppf(p,loc=0,scale=1)`, $0 \leq q \leq 1$, it supplies inverse normal cumulated distributions values.

Let's see for these functions some examples of use,

normal

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Apr  7 14:51:17 2025
Normal distribution. Examples
Ejemplos distribución normal
@author: juan
```

```
8  """
9  import numpy as np
10 from scipy.stats import norm
11
12 #we firs call the functions of norm directly
13 x = np.linspace(-5, 5,7)
14 p = np.linspace(0,1,7)
15 print('x=',x,'\n')
16 print('p=',p,'\n')
17 #normal distribution standard
18 y = norm.pdf(x)
19 print('pdf standard',y,'\n')
20
21 #normal cumulative distribution
22 ycum = norm.cdf(x)
23 print('cdf standard',ycum,'\n')
24
25 #normal inverse cumulative distribution
26 val = norm.ppf(p)
27 print('ppf standard',val,'\n')
28
29 #now, we introduce a mean value and a std deviation
30 mean =10 #mean
31 std = 5 #std dev
32
33 #normal distribution mean = 10 std dev = 5
34 y = norm.pdf(x,mean,std)
35 print('pdf m=10,std=5',y,'\n')
36
37 #normal cumulative distribution
38 ycum = norm.cdf(x,mean,std)
39 print('cdf m=10,std=5',ycum,'\n')
40
41 #normal inverse cumulative distribution
42 val = norm.ppf(p,mean, std)
43 print('ppf m=10,std=5',val,'\n')
44
45 #alternatively we can define a normal distribution with a predefined mean and
46 #standar deviation
47 norm_10_5 = norm(loc=10,scale=5)
48 print('mean and std predefined \n')
49 #normal distribution mean = 10 std dev = 5
50 y = norm_10_5.pdf(x)
51 print('pdf m=10,std=5',y,'\n')
52
53 #normal cumulative distribution
54 ycum = norm_10_5.cdf(x)
55 print('cdf m=10,std=5',ycum,'\n')
56
57 #normal inverse cumulative distribution
58 val = norm_10_5.ppf(p)
```

```
59 print('ppf m=10,std=5',val,'\n')
```

```
x= [-5. -3.33333333 -1.66666667  '0.' 1.66666667  3.33333333 5.]

p= [0.   0.16666667 0.33333333 '0.5'  0.66666667 0.83333333 1.]

pdf standard [1.48671951e-06 1.54227900e-03 9.94771388e-02 3.98942280e-01
9.94771388e-02 1.54227900e-03 1.48671951e-06]

cdf standard [2.86651572e-07 4.29060333e-04 4.77903523e-02 '5.00000000e-01'
9.52209648e-01 9.99570940e-01 9.99999713e-01]

ppf standard [-inf -0.96742157 -0.4307273 '0.' 0.4307273 0.96742157 inf]

pdf m=10,std=5 [0.00088637 0.0022792  0.00524438 0.01079819 0.01989543 0.03280201
0.04839414]

cdf m=10,std=5 [0.0013499  0.00383038 0.00981533 0.02275013 0.04779035 0.09121122
0.15865525]

ppf m=10,std=5 [-inf 5.16289217 7.8463635  10. 12.1536365 14.83710783 inf]

mean and std predefined

pdf m=10,std=5 [0.00088637 0.0022792  0.00524438 0.01079819 0.01989543 0.03280201
0.04839414]

cdf m=10,std=5 [0.0013499  0.00383038 0.00981533 0.02275013 0.04779035 0.09121122
0.15865525]

ppf m=10,std=5 [-inf 5.16289217 7.8463635 10. 12.1536365  14.83710783 inf]
```

Si analizamos el código del ejemplo, las líneas 18 y 22 nos dan los valores que toma la ditribución normal estándar y su integral (la distribución normal acumulada) en los puntos $-5, -3.33, -1.66, 0, 1.66, 3.33, 5$. Para la distribución acumulada, podemos fijarnos en el valor correspondiente a $x = 0$, el resultado obtenido es 0.5. Efectivamente, si obtenemos un número aleatorio, empleando una distribución de probabilidad normal, 0.5 —un 50 %— es la probabilidad de obtener un número en el intervalo $(-\infty, 0]$.

La línea 26, nos calcula la inversa de la distribución normal acumulada. Es decir nosotros damos un valor para la probabilidad y la función nos devuelve el límite derecho del in-

If we analyze the example code, lines 18 and 22 compute the values taken for the standard normal distribution ant its integral (the standard cumulated normal distribution) on points $-5, -3.33, -1.66, 0, 1.66, 3.33, 5$. Focus on the cumulated distribution, we can look at the value corresponding to $x = 0$, the result obtained is 0.5. Indeed, if we get a random number using a normal distribution, de value 0.5 —a 50 %— is the probability to obtain a number in the interval $(-\infty, 0]$.

Line 26 calculates the inverse of the cumulated normal distribution. That is, we give to the function a probability value and the function yields the value of the right limit of the interval this probability belongs to. So, for ins-

tervalo al que corresponde dicha probabilidad. Así por ejemplo, para el valor $p = 0.5$ que corresponde a una probabilidad del 50 %, la función `ppf` nos devuelve el valor 0. Es decir, para una distribución normal, tenemos una probabilidad del 50 % de obtener aleatoriamente un valor en el intervalo $(-\infty, 0]$. La figura 1.11 muestra la función inversa de probabilidad acumulada y en particular el punto correspondiente al valor 0.5.

Los ejemplos incluidos en las líneas de código 33 a 58, repiten los cálculos anteriores pero ahora para una distribución de probabilidad normal de media 10 y desviación estándar 5. Lo hace de dos maneras distintas, primero indicando expresamente el valor de la media y la desviación y luego creando una nueva función con dichos valores incluidos por defecto.

tance, for $p = 0.5$ the function `ppf` casts back the value 0. That is, for a normal distribution the probability of obtain randomly a number in the interval $(-\infty, 0]$ is 50 %. Figure 1.11 shows the inverse standard cumulated normal distribution and, in particular, the point corresponding to the value 0.5.

The examples included in code lines 33 to 58, repeat the calculation performed before, but now using normal distributions with mean 10 and standard deviation 5. We have obtain the results using to different methods. First we use the function `norm` passing to it the values of the mean and the deviation and after, we repeat these calculation but now, first we create a new function with the desired values of mean and standard deviation included by default.

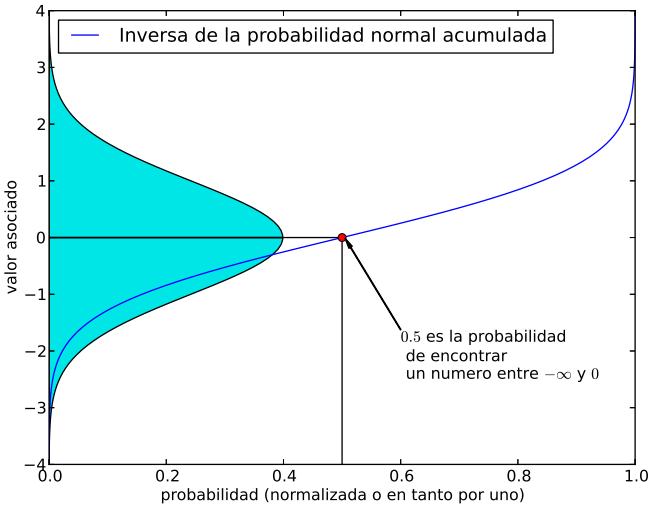


Figura 1.11: Función inversa de probabilidad normal acumulada
Figure 1.11: Inverse standard cumulated normal distribution

s

Retomando el hilo de nuestro problema, lo que a nosotros realmente nos interesa, es encontrar intervalos de confianza, es decir conocer el intervalo en torno al valor medio $\bar{x}_n$ que corresponde a una determinada probabilidad. La función inversa de probabilidad acumulada nos da el intervalo entre $-\infty$ y x correspondiente a una determinada probabilidad. Si

Coming back to our problem, we are actually interested in finding confidence intervals. That is, get to know the interval around a mean value $\bar{x}_n$ associated to a specific probability. The inverse cumulated normal probability function gives us the interval between $-\infty$ and x associated to a specific probability. If we combine the results of this function with the

combinamos los resultados de dicha función con las propiedades de simetría y área unidad de la función normal podemos obtener el intervalo centrado en torno a la media correspondiente a una determinada probabilidad.

Así el intervalo $[-x, x]$ correspondiente a una probabilidad del $P\,\%$, abarca un área bajo la curva $P/100$. Por tanto el área que queda en las colas —a derecha e izquierda del intervalo— es $1 - x/100$. Como la distribución normal es simétrica, dicha área debe dividirse en dos, una correspondiente a la cola $[-\infty, -x]$ y la otra correspondiente a la cola $[x, \infty]$.

symmetry and unit area characteristic of the normal function, we can obtain the interval centered around a measurement, associated to a specific probability.

So the interval $[-x, x]$ corresponding to a $P\,\%$ probability, Cover an area beneath the curve of a normal funtion equals to $P/100$. Thus, the remaining area in the sides — on the left and right the interval— is $1 - x/100$. Moreover, as the normal distribution is symmetric we should divide this area by two. One half correspond to the interval $[-\infty, -x]$ and the other to the interval $[x, \infty]$.

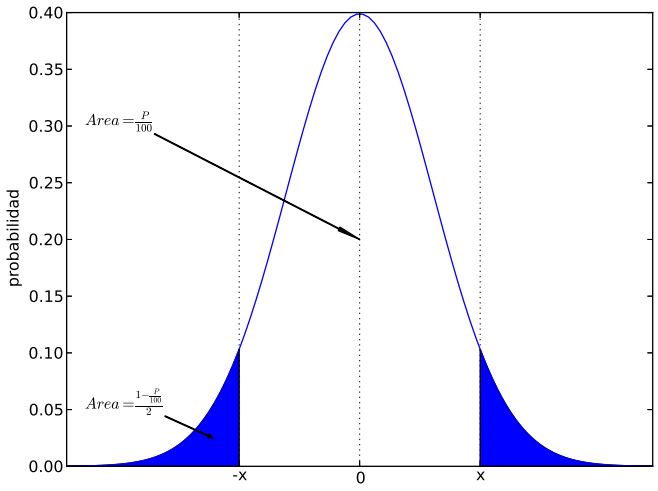


Figura 1.12: Intervalo de probabilidad P %
Figure 1.12: Probability interval P%

La figura 1.12 muestra gráficamente la relación de áreas que acabamos de describir; las dos colas aparecen pintadas en azul.

Para obtener el límite inferior, $-x$, del intervalo de probabilidad $P\,\%$ basta emplear la función `norm.ppf`, introduciendo como variable de entrada el área de la cola, `x=norm.ppf((1-P/100)/2)`. Así por ejemplo, para obtener el límite inferior del intervalo de confianza correspondiente a una probabilidad del 90 % siguiendo una distribución normal,

Figure 1.12 shows graphically the relationship among the areas we have just described; both tip-sides are in blue.

To obtain the lower limit, $-x$, for a probability interval $P\,\%$ we can use the function `norm.ppf` giving as input the area of the tip-side. `x=norm.ppf((1-P/100)/2)`. So for instance, to obtain the lower limit of the confidence interval for a 90 % probability, following a normal distribution,

```
In [4]: p = (1-0.90)/2

In [5]: x = norm.ppf(p)

In [6]: prin2t(x)
-1.6448536269514729
```

Por tanto el intervalo de confianza correspondiente a una probabilidad del 90 % para una distribución normal de media cero y desviación estándar uno es,
$I_{90\,\%} = [-1.6449,\ 1.6449]$

Supongamos ahora que hemos realizado un conjunto de n medidas de una determinada magnitud física, por ejemplo temperatura en $^{\text{o}}$C, y obtenemos su media $\bar{x}_n = 6.5\ ^{\text{o}}$C y su desviación estándar $s_{xn} = 2$. Si queremos dar el resultado de la medida con una incertidumbre correspondiente a un intervalo de confianza del 95 %, calculamos el intervalo de confianza empleando la función inversa de probabilidad acumulada normal, y después, multiplicamos el resultado por s_{xn} para ajustarlo a la distribución normal que sigue la media de nuestros datos,

Thus the confidence interval corresponding to a 90 % probability for a normal distribution with mean cero and deviation standard one is
$I_{90\,\%} = [-1.6449,\ 1.6449]$

Suppose we have taken a set of n measures of a physical magnitude, for instance temperature in $^{\text{o}}$C, and we calculate it mean value $\bar{x}_n = 6.5\ ^{\text{o}}$C and standard deviation $s_{xn} = 2$. If we can represent the result of the measure with an uncertainty corresponding to a 95 % confidence interval, we calculate the such confidence intervale using the inverse cumulated normal probability am after, we multiply the result for s_{xn} to fit it to the normal distribution our data belong to,

```
In [10]: xm = 6 #media de las medidas //measures mean

In [11]: sigma = 2 #desviacion de las mendidad // measures deviation

In [12]: P = (1-0.95)/2 #probalidad acumulada hasta el límite inferior del
#intervalo// cumulated probability up to the lower limit of the interval

In [18]: print(P)
0.025000000000000022

In [15]: from scipy.stats import norm

In [16]: i95 = norm.ppf(P) #limite inferior del intervalo //Lower limit of the interval

In [19]: print(i95)
-1.959963984540054

In [20]: incert = i95*sigma #limite inferior de la
#incertidumbre//lower limit of the uncertainty

In [21]: print(incert)
-3.919927969080108
```

Por tanto, en este ejemplo, debemos expresar la medida como $6.5 \pm 3.92\ ^{\text{o}}$C indicando que

Thus, in this example, we have to represent the measurement as $6.5 \pm 3.92\ ^{\text{o}}$C indicating

el intervalo de confianza es del 95 %.

La distribución T de Student. Habitualmente, cuando el número de medidas que se han tomado para estimar el valor de $\bar{x}_n$ es pequeño, los intervalos de confianza se calculan empleando la distribución t de Student de $n-1$ grados de libertad, donde n representa el número de medidas tomadas.

that the confidence interval is 95 %.

The Student's T distribution Usually, when the number of measurements taken to estimate the value of $\bar{x}_n$ is small, we calculate the confidence intervals using the Studen's T distribution of $n-1$ degrees of freedom, where n represent the number of measurements taken.

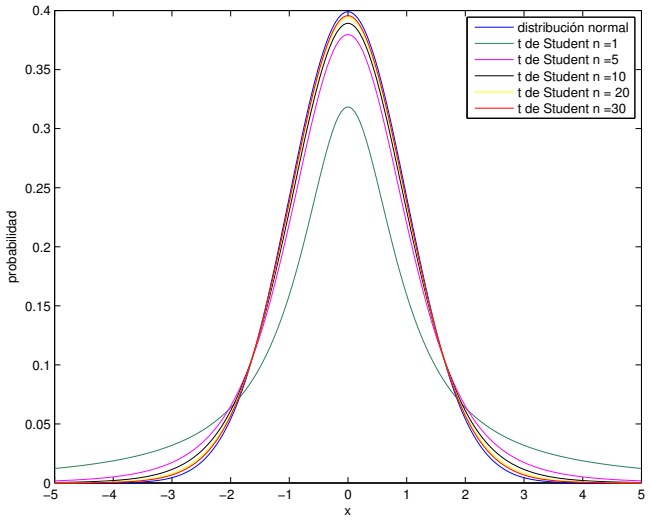


Figura 1.13: Comparación entre las distribuciones t de Student de 1, 5, 10, 20 y 30 grados de libertad y la distribución normal.
Figure 1.13: Comparison among the Student's T dsitributions of 1, 2,10, 20 and 30 dergrees of fredom and the normal distribution

Para un número pequeño de muestras, la distribución t de Student da una aproximación mejor que la distribución normal. Para un número de muestras grande, ambas distribuciones son prácticamente iguales. La figura 1.13 muestra una comparación entre la distribución normal y la distribución t de Student, calculada para distintos grados de libertad. De la figura se puede concluir en primer lugar que la desviación estándar de la distribución t de Student es en todos los casos mayor que la de la distribución normal, con lo cual los intervalos de confianza para un valor dado de la probabilidad son siempre mayores. En segundo lugar se observa también que para 30

For a small number of samples, the Student's T distributions yields a better approximation than the normal distribution. For a large number of samples, both distributions are quite the same. Figure 1.13 shows a comparison among the normal distribution a Student's T distribution calculated for several degrees of freedom. Looking at the figure we may conclude that the standard deviation of the Student's T distribution is in every case larger than the normal distribution standard deviation. This means that the confidence intervals associated with a specific probability value are always larger too. Moreover, we can see also that for 30 or more measurements — $n \geqslant 30$

o más medidas — $n \geqslant 30$ — la diferencia entre las distribuciones t de Student y normal es prácticamente despreciable.

El procedimiento de cálculo es el mismo que si empleamos la distribución normal, basta sustituir la inversa de la distribución normal acumulada por la inversa de la distribución t de Student acumulada. En Python, podemos importar la distribción t de Student de modo análogo a como hicimos para la distribución normal: `scipy.stats.t`. Por defecto obtendremos una distribución de media cero y varianza uno y podremos cambiar dichos parámetros empleando las variables `log` y `scale`. Disponemos de las mismas funciones `pdf, cdf, ppf, etc` que para la distribución normal. La única diferencia es que al llamar a estas funciones debemos indicar los grados de libertad de la distribución. `t.pdf(x,df)`.

Así, para el ejemplo anterior de las medidas de temperatura de media $\bar{x}_n = 6.5$ ºC y desviación estándar $s_{xn} = 2$ si suponemos que el número de medidas tomadas es $n = 5$. El intervalo de confianza correspondiente a un 95 % lo calcularíamos mediante la inversa de la distribución de probabilidad t de Student de $n - 1$ grados de libertad acumulada como,

— the difference between the normal and the Student's T distributions is negligible.

The method to calculate a confidence interval is identical to the normal distribution case; we just use the inverse cumulated Student's T distribution instead of the inverse cumulated normal distribution. IN python we can import the the Student's T distribution using the same wise that in the normal distribution case: `scipy.stats.t`. We will obtain a mean zero variance one distribution by default and we can change these parameters using using the variables, `log` y `scale`. The same functions `pdf, cdf, ppf, etc` as for the normal distribution are available The main difference is that now we can input the distribution degrees of freedom to the functions `t.pdf(x,df)`.

So, for the previous example of temperature measurements, with mean $\bar{x}_n = 6.5$ ºC and standard deviation $s_{xn} = 2$ supposing that we have taken $n = 5$ measurements. We calculate the 95 % confident interval using the inverse cumulated Student's T distribution of five degrees of freedom as follows,

```
In [22]: from scipy.stats import t #importamos la distribucion T de Student//
#import the Student's T #distribution

In [23]: P = (1-0.95)/2 #probalidad acumulada hasta el límite inferior del
#intervalo// cumulated probability up to the lower limit of the interval

In [24]: i95 = t.ppf(P,4) #calculamos con 5-1 grados de libertad// we calculate
#with 5-1 degrees of freedom

In [25]: print(i95) #límite inferior del itervalo de confianza//
#Confidence interval lower limit
-2.7764451051977987

In [26]: incert = i95*sigma #límite inferior de la incertidumbre//
#Uncertainty lower limit

In [27]: print(incert)
-5.5528902103955975
```

Por tanto deberíamos expresar el valor de la medida realizada como 6.5 ± 5.55 ºC. Si comparamos este resultado con el obtenido

Therefore, we should represent the measurement value as 6.5 ± 5.55 ºC. Comparing this result with the one obtained using the normal

empleando la distribución normal, vemos que el intervalo de confianza y, por tanto, la incertidumbre han aumentado.

1.4.3. Propagación de la incertidumbre: Estimación de la incertidumbre de medidas indirectas.

Supongamos que deseamos obtener la incertidumbre de una determinada magnitud física y que no hemos medido directamente, sino que se determina a partir de otras cantidades $x_1, x_2, \cdots, x_n$ empleando una relación funcional f,

distribution, we can see that the confidence interval and, consequently, the uncertainty have increased.

1.4.3. Uncertainty propagation: uncertainty estimation of indirect measurements.

Suppose we wish to get the uncertainty of a physical magnitude y that we have not measured directly, but that is determined from other measurements $x_1, x_2, \cdots, x_n$ using a functional relationship f,

$$y = f(x_1, x_2, \cdots, x_n)$$

La función f suele recibir el nombre de ecuación de medida, y no tiene por qué representar simplemente una ley física; de hecho, debería incluir entre sus entradas cualquier cantidad que pueda contribuir a modificar significativamente la incertidumbre de y. Por ejemplo: diferentes observadores, laboratorios, experimentos, horas a las que se han realizado las observaciones, etc.

La incertidumbre asociada al valor de y, a la que llamaremos $u_c(y)$, se expresa en función de la desviación estándar de y,

Function f is referred to as a measure equation; it does not necessarily represent a physical law. In fact, f should encompass any quantity that can significantly alter the uncertainty of y; for instance, different observers, laboratories, experiments, the time the observations have been carried out, etc.

The uncertainty associated to the value taken for y, that we will denote $u_c(y)$, is described using the standard deviation of y,

$$u_c(y) = \sqrt{s_c^2(y)},$$

donde $s_c^2(y)$ se define como la varianza *combinada* de y, que se estima en primera aproximación a partir de la expansión en serie de Taylor[4] de la función f.

where $s_c^2(y)$ is defined as the *combined* variance of y, that can be estimated in a first approximation using the Taylor's expansion[4] of function f.

[4]En primera aproximación empleamos un desarrollo de Taylor de primer orden. En realidad, lo adecuado o no de este método depende de las características de la función f. No discutiremos aquí este problema.

[4]In first approach we use the first-order Taylor's expansion. Actually, this method could be valid or not depending on the features of function f. We do not discusse thsi problem here.

$$s_c^2(y) = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 s^2(x_i) + 2 \sum_{i=1}^{n-1} \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} c_v(x_i, x_j)$$

Donde $s^2(x_i)$ representa la varianza asociada a la entrada x_i y $c_v(x_i, x_j)$ la covarianza entre las entradas x_i y x_j.

La covarianza entre dos variables z y t de las que se tienen n muestras, puede estimarse

Were $s^2(x_i)$ represents the variance associated to the input x_i and $c_v(x_i, x_j)$ de covariance between inputs x_i and x_j.

If we have n samples of a pair of variables z y t , we can estimate their covariance using

a partir de las muestras como,

the samples as follows,

$$c_v(z,t) = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})(t_i - \bar{t})$$

De modo análogo a como hicimos en el caso de la varianza de la media de un conjunto de medidas, podemos relacionar la covarianza entre las medias de dos variables, con la covarianza entre n medidas de dichas variables como,

Following the same steps that in the variance calculation for the mean of a set of measurements, we may relate the covariance between the means of two variables with the covariance among the n measurements of such variables as,

$$c_v(\bar{z},\bar{t}) = \frac{c_v(z,t)}{n}$$

En muchos casos de interés no existe correlación entre las variables de entrada. En estos casos, los valores de la covarianza será nulos y podemos simplificar la ecuación empleada en el cálculo de $s_c^2(y)$,

Frequently there is no correlation among input variables. In these cases, the covariance values are zero and we can simplify the equation used for calculating $s_c^2(y)$,

$$s_c^2(y) = \sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^2 s^2(x_i)$$

Un ejemplo simple lo podemos obtener de la ecuación para la potencia disipada en una resistencia eléctrica. En este caso, el valor de la potencia disipada depende del voltaje, V, de la resistencia R_0 medida a una temperatura de referencia t_0, de la temperatura real a que se encuentra la resistencia t y, en primera aproximación, de un coeficiente que b, que establece una relación linea entre temperatura y resistencia.

We will show a easy example using the equation to determine the power dissipated in an electric resistance. In this case, the value of the power dissipated depends on the voltage, V, on the resistance R_0 measured at a reference temperature t_0, on the actual resistance temperature t and, in a first approximation, on a coefficient b that defines a linear dependence between the temperatura and the resistance.

$$P = f(V, R_o, b, t) = \frac{V^2}{R_0\left(1 + b(t - t_0)\right)}$$

Supongamos que b es un coeficiente numérico no sujeto a incertidumbre y que las variables V, R_0, t y t_0 no están correlacionadas entre sí. Podemos entonces expresar la varianza en la potencia disipada, a partir de las varianzas de las variables de entrada como,

Let's suppose that b is a numerical coefficient with none uncertain and that variables V, R_0, t y t_0 are not correlated. Then, we can determine the variance of the power dissipated from the variance of the input variables as follows,

$$s_c^2(P) = \left(\frac{\partial f}{\partial V}\right)^2 s^2(V) + \left(\frac{\partial f}{\partial R_0}\right)^2 s^2(R_0) + \left(\frac{\partial f}{\partial t}\right)^2 s^2(t) + \left(\frac{\partial f}{\partial t_0}\right)^2 s^2(t_0)$$

Por tanto,

Thus,

$$s_c^2(P) = \left(\frac{2V}{R_0\left(1 + b(t - t_0)\right)}\right)^2 s^2(V) + \left(\frac{-V^2\left(1 + b(t - t_0)\right)}{R_0^2\left(1 + b(t - t_0)\right)^2}\right)^2 s^2(R_0) +$$
$$+ \left(\frac{-V^2 R_0 b}{R_0^2\left(1 + b(t - t_0)\right)^2}\right)^2 s^2(t) + \left(\frac{V^2 R_0 b}{R_0^2\left(1 + b(t - t_0)\right)^2}\right)^2 s^2(t_0)$$

y la incertidumbre en la potencia disipada será la raíz cuadrada de la expresión que acabamos de obtener: $u_c(P) = \sqrt{s_c^2(P)}$

Establecer los intervalos de confianza, para una incertidumbre propagada no es tan sencillo como en el caso de una incertidumbre obtenida directamente de una medida experimental. En primera aproximación, podemos suponer que y sigue una distribución normal de media el valor obtenido, a partir de la función f y desviación $u_c(p)$ y estimar los intervalos de confianza del mismo modo que se describió anteriormente para una medida directa. Pero esto no es siempre correcto. No podemos establecer en realidad ninguna ley general que relacione las distribuciones de probabilidad de las incertidumbres de las variables de entrada con la distribución de probabilidad de la incertidumbre de la variable de salida.

1.4.4. Ejemplo de estimación de la incertidumbre con Python.

Para aclarar los métodos descritos en los apartados anteriores, vamos obtener la incertidumbre de la potencia disipada en una resistencia, siguiendo la ecuación descrita en el apartado anterior. La tabla 1.1 muestras los datos obtenidos tras realizar 20 medidas experimentales de la tensión y la temperatura en una resistencia.

Para obtener la incertidumbre en el valor de la potencia disipada por la resistencia debemos en primer lugar calcular la varianza de las medidas directas suministradas en la tabla.

•Varianza del valor nominal de la resistencia R_0: En este caso, la tabla nos suministra como dato el intervalo de confianza correspondiente al 98 %. Podemos emplear la función de Scipy `norm.ppf()` para calcular la desviación estándar de R_0. Como hemos visto, el límite inferior del intervalo de confianza del 98 %, corresponde con una probabilidad acumulada de $(1 - 0.98)/2$,

And the dissipated power uncertainty would be the square root of the expression just obtained: $u_c(P) = \sqrt{s_c^2(P)}$

To stablish the confidence intervals for a propagated uncertainty is not so easy as in the case o a experimental measurement straightforwardly obtained. On first approximation, we may suppose that y will follow a normal distribution which mean is the value obtained using function f and deviation $u_c(p)$ and obtain the confidence intervals in the same way we used for a straightforward measure. But that it is not always right. We can not stablish a general law that cast the relationship among the input variables probability distributions and the uncertainty of the output variable.

1.4.4. Example of uncertainty estimation using Python.

To clarify the methods included in previous sections, we are going to obtain the uncertainty of the power dissipated in a resistance, following the equation described in the previous section. Table 1.1 shows the data obtained after performing 20 experimental measurements of the Voltage and temperature of a resistance.

To obtain the uncertainty of the resistance dissipated power we need first to calculate the variance of the straightforward measurements supplied on the table.

•Variance of the resistance nominal value R_0: In this case the table supplied the confidence interval corresponding to 98 %. we can use the Scipy function `norm.ppf()` to calculate the standard deviation of R_0. As we have seen, the lower limit of the 98 % confidence interval is associated with a cumulated probability of $(1 - 0.98)/2$,

```
In [1]: from scipy.stats import norm

In [2]: i98 = norm.ppf((1-0.98)/2)

In [3]: i98
Out[3]: -2.3263478740408408
```

Tabla 1.1: Mediciones de Temperatura y Voltaje, sobre una resistencia de prueba de 100 Ω
Table 1.1: Temperature and Voltage Measurements on a 100 Ω test resistence

Valor nominal de la resistencia Resistence nominal value	$R_0 = 100 \pm 5\Omega$ (intervalo confianza 98 %) (confidence interval 98 %)
Temperatura de referencia Reference temperature	$t_0 = 50 \pm 0.1^oC$ (división menor del sensor 0.2^oC) (sensor least count 0.2^oC)
Valor del parámetro b b parameter value	$b = 0.4\Omega/^oC$ (incertidumbre despreciable) (negligible uncertainty)

Temperatura ºC Temperature ºC	Votaje mV Voltage mV
55.7	761
62.8	897
58.1	879
60.9	767
66.3	758
59.4	763
61.8	762
62.5	761
59.4	790
61.4	852
55.7	816
65.3	752
56.9	848
57.8	920
62.9	743
59.9	858
62.6	761
59.1	788
57.9	775
53.8	791

Dicha cantidad, multiplicada por la desviación estándar de R_0, corresponde con el limite inferior del intervalo de confianza suministrado en la tabla para R_0. Por tanto,

This quantity multiplied by R_0 standard deviation casts the confidence interval lower limit supplied by the table for R_0. Then,

```
In [5]: sR0 = -5/i98

In [6]: sR0 = -5/i98 #R0 standard deviation

In [7]: s2R0 = sR0**2 #R0 variance

In [8]: print('sR0=',sR0,'s2R0=', s2R0)
sR0= 2.149291623919966 s2R0= 4.619454484652525
```

luego $s^2(R_0) = 4.6195$

•Varianza de la temperatura de referencia t_0: En este caso se ha expresado la in-

•Reference temperature variance t_0: In this case, the uncertainty has been expressed as

certidumbre en función de la mitad de la división más pequeña de la escala del aparato de medida. Podemos considerar por tanto que la incertidumbre estaría representada por una distribución uniforme definida en el intervalo $[-0.1\ 0.1]$. En esta caso, podemos asociar la varianza de la medida directamente con dicha distribución uniforme,

half the sensor least count. We can consider then that the uncertainty is represented by an uniform distribution defined in the interval $[-0.1\ 0.1]$. In this case, we can associate straightforwardly the measurement variance with this uniform distribution.

```
In [14]: s2t0 = (1-(-1))**2/12

In [15]: print(s2t0)
0.3333333333333333
```

Así, $s^2(t_0) = 0.3333$

•Varianzas de las medidas de temperatura y voltaje: En primer lugar promediamos los valores obtenidos de voltaje y temperatura, para calcular el valor de la medida. Numpy tiene definida la función `mean()` que permite obtener directamente el valor medio de los elementos de una array de Numpy. Podemos por tanto definir en Python dos vectores, unos para los datos de temperatura y otro para los datos de voltaje y calcular directamente las medias,

Thus, $s^2(t_0) = 0.3333$

•temperature measurements and voltage measurements variances: First we calculate the mean of the values we have obtained for temperature and voltage, taking these means as the measurement value. Numpy defines the function `mean()` which allows us directly to calculate the mean value of a Numpy array. So, we may define in Python two vectors one with the temperature data and other for the voltage data and obtain the means straightforwardly,

```
In[29] t=np.array([55.7, 62.8, 58.1, 60.9, 66.3, 59.4, 61.8, 62.5, 59.4, 61.4,
55.7, 65.3, 56.9, 57.8, 62.9, 59.9, 62.6, 59.1, 57.9, 53.8])

In[30]: t
Out[30]:
array([55.7, 62.8, 58.1, 60.9, 66.3, 59.4, 61.8, 62.5, 59.4, 61.4, 55.7,
65.3, 56.9, 57.8, 62.9, 59.9, 62.6, 59.1, 57.9, 53.8])

In[31]: t.mean()
Out[31]: 60.00999999999999

In [32]: v=np.array([761, 897, 879, 767, 758, 763, 762, 761, 790, 852,
816, 752, 848, 920, 743, 858, 761, 788, 775, 791])

In [33]: v
Out[33]:
array([761, 897, 879, 767, 758, 763, 762, 761, 790, 852, 816, 752, 848,
920, 743, 858, 761, 788, 775, 791])

In [34]: v.mean()
Out[34]: 802.1
```

Una vez estimadas las medias, que emplearemos como valores de las medidas de la temperatura y el voltaje, podemos ahora estimar la varianza o la desviación estándar de las medias haciendo uso de las función de Numpy `std()`, para la desviación estándar, y de `var()`, para la varianza. Además, debemos aplicar el teorema central del límite, dividiendo por el número de datos empleados. Un último detalle técnico es que debemos llamar a las funciones con el parámetro `var(ddof=1), std(ddof=1)`, para que en el cálculo de la varianza y de la desviación estandar se divida por el número de datos menos uno $(n-1)$, en lugar de entre n. (ver sección 1.4.1) ,

Once mean values have been estimated, we will employ them as the values of the temperature and voltage measurements. We can now estimate the variance or the standard deviation of the mean values using the Numpy functions `std()` for the standard deviation and `var()` for the variance. Besides, we should apply the central limit theorem, dividing the result by the number of data used. A last technical detail is that we have to call the functions with the parameter `var(ddof=1), std(ddof=1)`, in order to have the calculations of the variance and the standard deviation divide by the number of data less one $(n-1)$ instead than by n. (see section 1.4.1),

```
In [65]: s2t= t.var(ddof=1)/len(t) #mean temp. variance

In [66]: print(s2t)
0.5328368421052628

In [67]: s2v= v.var(ddof=1)/len(v) #mean volt. variance

In [68]: print(s2v)
145.58368421052631

In [71]: st= t.std(ddof=1)/np.sqrt(len(t)) #mean volt. standard dev

In [72]: print(st)
0.7299567398861818

In [73]: sv= v.std(ddof=1)/np.sqrt(len(v)) #mean volt. standard dev

In [74]: print(sv)
12.065806405314412
```

Lógicamente, los resultados muestran que las desviaciones estándar son las raíces cuadradas de las varianzas. Tenemos por tanto, $\bar{t} = 60.01$, $\bar{V} = 802,01$, $s^2(t) = 0.5382$, $s^2(V) = 145.5837$.

En este momento, hemos reunido ya toda la información necesaria para estimar la incertidumbre de la potencia consumida en la resistencia de nuestro ejemplo. La tabla 1.2 contiene los datos calculados.

Obviously, the result show that the standard deviation are the square roots of the variances. So, eventually we have, $\bar{t} = 60.01$, $\bar{V} = 802,01$, $s^2(t) = 0.5382$, $s^2(V) = 145.5837$.

Now we have gathered all needed information to estimate the uncertainty of the power consumed in the resistance of our example. Table 1.2 holds of the computed data.

We can now calculate the power,

$$P = \frac{V^2}{R_0\,(1 + b(t - t_0))} = \frac{0.80201^2}{100\,(1 + 0.4(60.01 - 50))} = 0.001285W = 1.285mW$$

Tabla 1.2: Medidas y varianzas estimadas
Table 1.2: Estimated means and variances

Variable (unidades) Variable (units)	Valor Value	Varianza Variance
$R_0(\Omega)$	100	4.6195
b ($\Omega/^{\mathrm{o}}$C)	0.4	–
t_0 ($^{\mathrm{o}}$C)	50	0.3333
$\bar{t}$ ($^{\mathrm{o}}$C)	60.01	0.5328
$\bar{V}$(mV)	802.01	145.5837

Donde hemos introducido el voltaje en voltios, para obtener la potencia directamente en vatios.

A continuación, propagamos la incertidumbre de las variables de entrada a la de salida,

Where we have introduced the voltage in volts in order to obtain the power directly in watts.

Subsequently, we propagate the uncertainty o the input variables to the output,

$$s_c^2(P) = \left(\frac{2V}{R_0\left(1+b(t-t_0)\right)}\right)^2 s^2(V) + \left(\frac{-V^2\left(1+b(t-t_0)\right)}{R_0^2\left(1+b(t-t_0)\right)^2}\right)^2 s^2(R_0) +$$
$$+ \left(\frac{-V^2 R_0 b}{R_0^2\left(1+b(t-t_0)\right)^2}\right)^2 s^2(t) + \left(\frac{V^2 R_0 b}{R_0^2\left(1+b(t-t_0)\right)^2}\right)^2 s^2(t_0) =$$
$$= \left(\frac{2\cdot 0.80201}{100\left(1+0.4(60.01-50)\right)}\right)^2 \cdot 1.455836\cdot 10^{-4} + \left(\frac{-0.80201^2\left(1+0.4(60.01-50)\right)}{100^2\left(1+0.4(60.01-50)\right)^2}\right)^2 \cdot 4.619 +$$
$$+ \left(\frac{-0.80201^2\cdot 100\cdot 0.4}{100^2\left(1+0.4(60.01-50)\right)^2}\right)^2 \cdot 0.5328 + \left(\frac{0.80201^2\cdot 100\cdot 0.4}{100^2\left(1+0.4(60.01-50)\right)^2}\right)^2 0.3333 =$$
$$= 1.4959\cdot 10^{-09} + 7.6327\cdot 10^{-10} + 5.6252\cdot 10^{-09} + 3.5189\cdot 10^{-09} = 1.1403\cdot 10^{-08}$$

En este caso, hemos multiplicado la varianza del voltaje $s^2(V)$ por 10^{-6}, para obtener $s_c^2(P)$ en W^2. Podemos expresar ahora la incertidumbre de la potencia consumida en función de la desviación estándar como,

In this case, we have multiply the voltage variance $s^2(V)$ by 10^{-6}, to obtain the $s_c^2(P)$ in W^2. We can now represent the uncertainty of the consumed power as a function of the standard deviation as,

$$u_c(P) = \sqrt{s_c^2(P)} = 1.0678\cdot 10^{-4} W$$

Con lo que finalmente expresaríamos la medida indirecta de la potencia consumida como, $P = 1.285 \pm 0.10678$ mW. Si suponemos que la distribución de probabilidad asociada a la incertidumbre de la potencia es normal, la incertidumbre así expresada (mediante la desviación estándar) representaría un intervalo de confianza del 68.27 %.

And eventually we can represent the indirect measurement of the consumed power as, $P = 1.285 \pm 0.10678$ mW. If we suppose that the probability distribution asociated to the power uncertainty is normal, the uncertainty so represented (using the standard deviation) represents a confidence intervar of 68.27 %. $s_c^2(P)$ en W^2.

1.5. Una nota sobre Pandas.

Python cuenta con un módulo específico para tratar con datos. Se trata de Pandas. Una descripción detallada de este módulo queda fuera del alcance de estos apuntes. Aquí simplemente daremos una muy breve introducción con un ejemplo de uso. Para ello tomaremos datos de la generación de energía electrica en España durante el 19 de marzo de 2025. Los datos están disponibles en la página web de Red Eléctrica,

1.5. A note on Pandas

Python has a specific module to deal with data, named Pandas. A detailed description o this module is far beyond the reach of these notes. We are only to supply a very brief introduction to it, following a use case. To do this, we will the take data of the electrical generation in Spain during the 19th of March, 2025. These data are available in the Red Eléctrica Web Page

https://demanda.ree.es/visiona/peninsula/nacionalau/total/2025-03-19

Para obtenerlos los hemos descargado en formato csv, en un archivo llamado *generacion20250319MW.csv*. Este formato –'comma-separated values'– es compatible con los programas de hojas de cálculo estándar.

Los datos contenidos en el fichero csv, corresponden a la producción total de potencia eléctrica en Megawatios, entre las 21.00 horas del días 18.03.2025 y las 03.00 horas del día 20.03.2025. Los datos se suministran a intervalos de 5 minutos y contienen las cantidades generadas por los distintos sistemas de producción: Eólica, Nuclear, etc.

A continuación, incluimos un script de Python en el que se muestra un análisis my sencillo de los datos descargados.

We have download the data into a file called *generacion20250319MW.csv*. The file has csv format –'comma separated values'– and it is compatible with standard spread sheet programs.

Data contained in the csv file, correspond to the total Power generation in Megawatts, since the 21.00 hour of 2025.03.18 to 03.00 hour of 2025.03.20. Data are supplied in regular intervals of 5 minutes and hold the quantities generated by the different production systems Nuclear power, wind power, etc.

Next, we present a Python's script that shows a simple analysis of the downloaded data.

datos_electrica_pandas.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 20 17:04:52 2025
Playing with the electrical network data
@author: juan
"""

import pandas as pd
import matplotlib.pyplot as pl

generacion_mw = pd.read_csv('../../datos/generacion20250319MW.csv')
generacion_mw.rename(columns=generacion_mw.iloc[1], inplace = True)
generacion_mw.drop([0,1],inplace=True)
lista = generacion_mw.columns
for i in lista[1:]:
    generacion_mw[i] = generacion_mw[i].astype(float)
```

```
18 #hacemos un analisis estadístico general
19 des = generacion_mw.describe()
20
21 #hacemos un diagrama de pastel con las medias de la producción
22 ax=des.loc['mean'][des.loc['mean']>0].plot.pie(rotatelabels=True)
23 #cambio los índices a las horas que es bastante sensato
24 gn_Mw_h = generacion_mw.set_index('Hora')
25
26 #pido que me grafique todas, frente al tiempo queda un tanto faragoso
27 #pero da una idea de lo que hay
28 pl.figure()
29 ax = gn_Mw_h.plot(rot=40,ylabel='Mw')
30 ax.legend(loc='upper left', bbox_to_anchor=(1.0, 1.0))
31 pl.figure()
32 gn_Mw_h.boxplot(rot=90,ylabel='Mw')
33 #vamos a hacer un diagrama de pastel pero eliminado los negativos que propiamente
34 #no son produccion Seleccionamos una hora
35
36 datos1005 = gn_Mw_h.loc['2025-03-19 01:05']
37 datos1230 = gn_Mw_h.loc['2025-03-19 12:30']
38 datosprod1005 = datos1005[datos1005>0]
39 datosprod1230 = datos1230[datos1230>0]
40 pl.figure()
41 datosprod1005.plot.pie(rotatelabels=True)
42 pl.figure()
43 datosprod1230.plot.pie(rotatelabels=True)
```

Veamos en detalle lo que hace éste codigo.

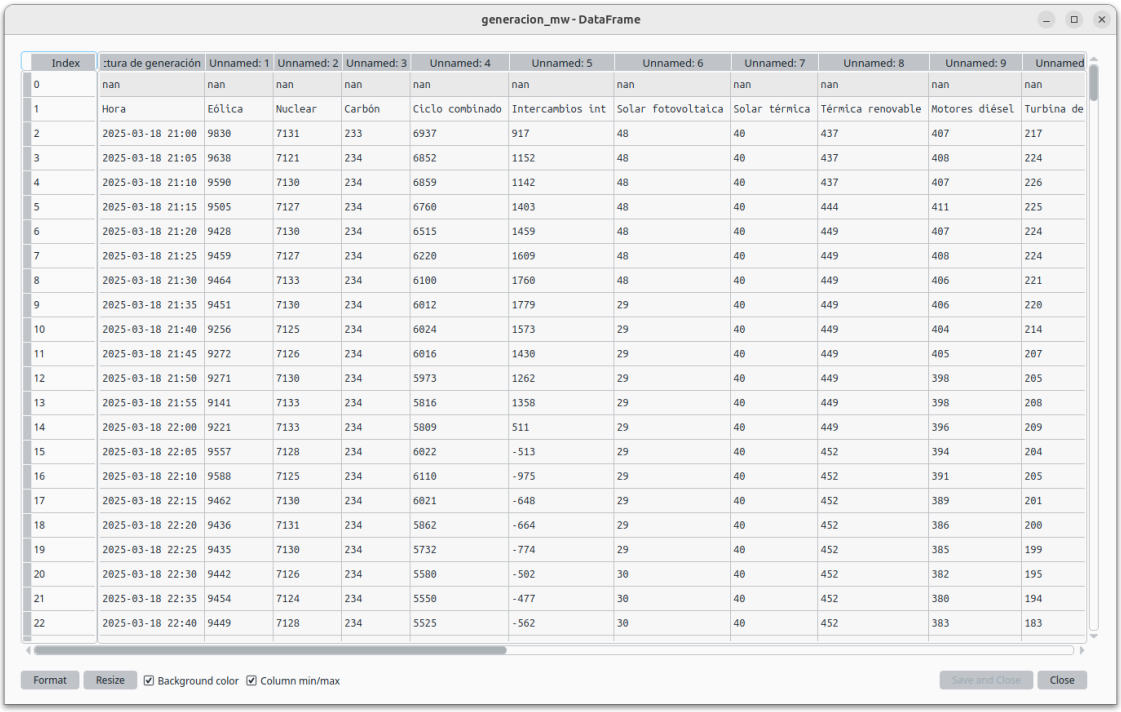
En la línea 9 importamos Pandas, del mismo modo que importaríamos cualquier otro módulo de Python.

En la línea 12 cargamos el fichero de datos en Python. Para ello, empleamos una función específica de Pandas `pd.read_csv`. Este comando permite leer un fichero tipo csv y lo carga en una variable especial de Pandas, cuyo tipo se conoce como *DataFrame*, llamada `generacion_mw`. La figura 1.14, muestra una vista de dicha variable empleando el visor de variables de Spyder. La primera fila muestra una cabecera, que en este caso, por la estructura del fichero csv cargado, no contiene nombres significativos (Unnamed:#). Además, tenemos una línea de datos que contiene NaNs. Una de las ventajas de Panda, es que, los *DataFrames* contienen un gran número de funciones propias para su manipulación. Sin nos fijamos, es en la fila dos de la tabla donde aparecen los nombre de las variables contenidas en cada columna.

Let see the performance of this program in more detail. In line 9 we import Pandas in the same way that we would import any other Python module.

In line 12 we load the data file in Python. To do this, we use a specific Pandas function `pd.read_csv`. This command allows us to read a csv file and load its contend in a special Pandas variable named generacion_mv, which type is known as *DataFrame*. Figure 1.14 shows a view of this variable using the spider variable scope. The first row, shows a header that, in this case and due to the data structure, has not meaningful names (Unnamed:#). In addition, we have got a row that contains NaNs. One advantage of Panda is that DataFrames include many built-in functions for manipulation. If we look at the table we see that row two has the names of the variables contained in each column.

We would like the data from row two to be used as the column names. We carry out this change using the `rename` command, shown in



Index	ctura de generación	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed
0	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
1	Hora	Eólica	Nuclear	Carbón	Ciclo combinado	Intercambios int	Solar fotovoltaica	Solar térmica	Térmica renovable	Motores diésel	Turbina de
2	2025-03-18 21:00	9830	7131	233	6937	917	48	40	437	407	217
3	2025-03-18 21:05	9638	7121	234	6852	1152	48	40	437	408	224
4	2025-03-18 21:10	9590	7130	234	6859	1142	48	40	437	407	226
5	2025-03-18 21:15	9505	7127	234	6760	1403	48	40	444	411	225
6	2025-03-18 21:20	9428	7130	234	6515	1459	48	40	449	407	224
7	2025-03-18 21:25	9459	7127	234	6220	1609	48	40	449	408	224
8	2025-03-18 21:30	9464	7133	234	6100	1760	48	40	449	406	221
9	2025-03-18 21:35	9451	7130	234	6012	1779	29	40	449	406	220
10	2025-03-18 21:40	9256	7125	234	6024	1573	29	40	449	404	214
11	2025-03-18 21:45	9272	7126	234	6016	1430	29	40	449	405	207
12	2025-03-18 21:50	9271	7130	234	5973	1262	29	40	449	398	205
13	2025-03-18 21:55	9141	7133	234	5816	1358	29	40	449	398	208
14	2025-03-18 22:00	9221	7133	234	5809	511	29	40	449	396	209
15	2025-03-18 22:05	9557	7128	234	6022	-513	29	40	452	394	204
16	2025-03-18 22:10	9588	7125	234	6110	-975	29	40	452	391	205
17	2025-03-18 22:15	9462	7130	234	6021	-648	29	40	452	389	201
18	2025-03-18 22:20	9436	7131	234	5862	-664	29	40	452	386	200
19	2025-03-18 22:25	9435	7130	234	5732	-774	29	40	452	385	199
20	2025-03-18 22:30	9442	7126	234	5580	-502	30	40	452	382	195
21	2025-03-18 22:35	9454	7124	234	5550	-477	30	40	452	380	194
22	2025-03-18 22:40	9449	7128	234	5525	-562	30	40	452	383	183

Figura 1.14: Datos de generación de energía eléctrica, Red Eléctrica 19.03.2025, cargados en un *Data Frame* de Pandas

Figure 1.14: Electrical Power generation, Red Eléctrica 19.03.2025, loaded into a Pandas' Data Frame

Nos gustaría que los datos de esta fila dos fueran los que dieran nombre a las columnas. Para hacer este cambio, utilizamos el comando `rename` mostrado en la línea 13 del script que estamos comentado. El comando pide que se renombren la columnas, empleando los datos de las segunda fila de la tabla mediante el uso de `generacion_mw.iloc`. El parámetro `inplace = True`, obliga a que el cambio se realice sobre el propio *DataFrame* en lugar de generar uno nuevo con los cambios.

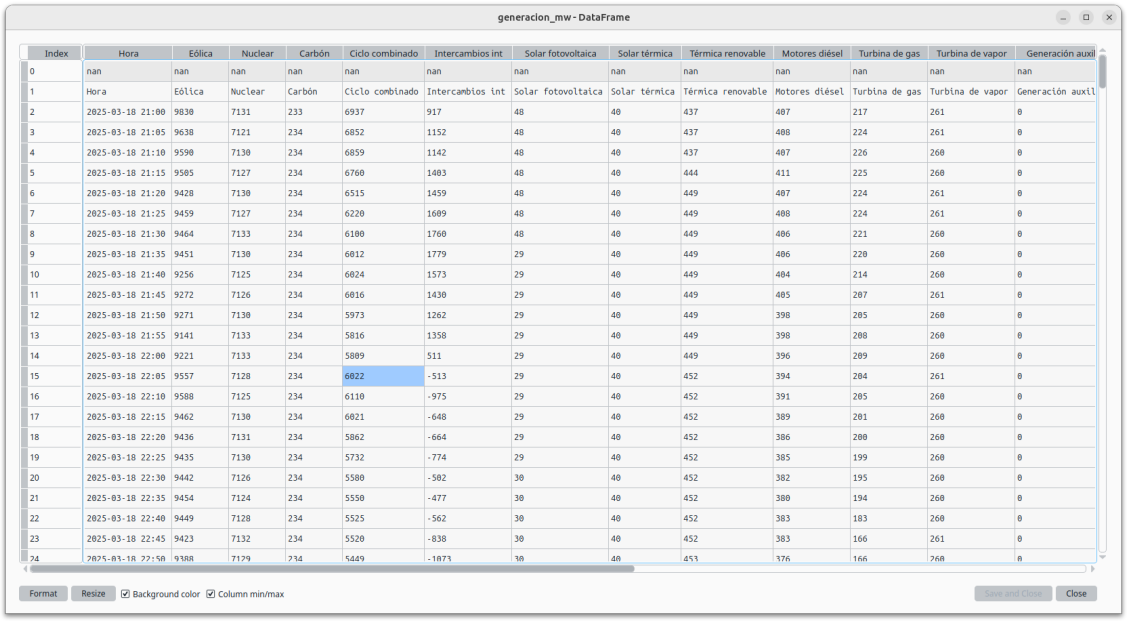
Podemos ver los resultados de este cambio en la figura 1.15. Evectivamente, las columnas han sido renombradas con los valores de los distintos sistemas de producción de energía eléctrica, pero dichos nombres siguen presentes en la fila uno, desde la que se han copiado a la cabecera. Además la fila cero sigue conteniendo NaNs.

Nos gustaría eliminar estas dos filas para

line 13 of the script we are commenting. The command changes the name of the columns using the data from the second row of the table using `generacion.iloc`. The parameter `inplace = True` forces the change to be made onto the same *DataFrame* instead of creating a new *DataFrame* with the changes.

We can see the results of the change in figure 1.15. The column are now renamed using the names of the different power generation sources, but these names are still presented in the first row, where they were before be copied on the header. Besides, the row zero still has NaNs.

We would like to delete these two lines and



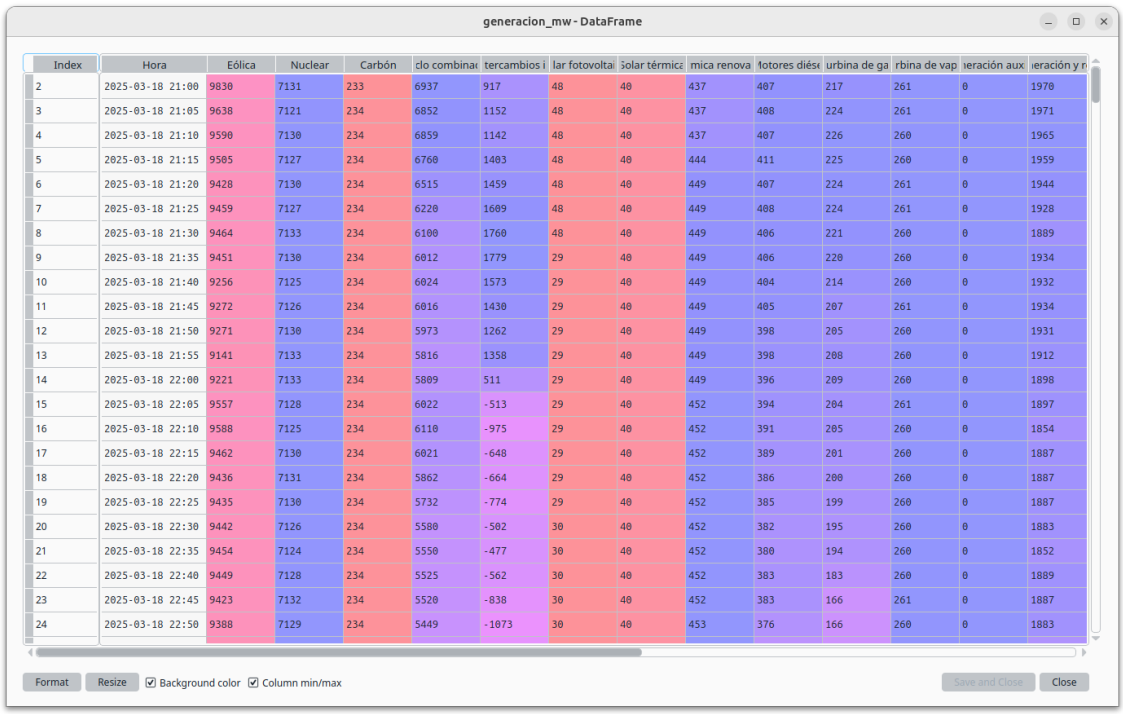
Index	Hora	Eólica	Nuclear	Carbón	Ciclo combinado	Intercambios int	Solar fotovoltaica	Solar térmica	Térmica renovable	Motores diésel	Turbina de gas	Turbina de vapor	Generación auxil
0	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
1	Hora	Eólica	Nuclear	Carbón	Ciclo combinado	Intercambios int	Solar fotovoltaica	Solar térmica	Térmica renovable	Motores diésel	Turbina de gas	Turbina de vapor	Generación auxil
2	2025-03-18 21:00	9830	7131	233	6937	917	48	40	437	407	217	261	0
3	2025-03-18 21:05	9638	7121	234	6852	1152	48	40	437	408	224	261	0
4	2025-03-18 21:10	9590	7130	234	6859	1142	48	40	437	407	226	260	0
5	2025-03-18 21:15	9505	7127	234	6760	1403	48	40	444	411	225	260	0
6	2025-03-18 21:20	9428	7130	234	6515	1459	48	40	449	407	224	261	0
7	2025-03-18 21:25	9459	7127	234	6220	1609	48	40	449	408	224	261	0
8	2025-03-18 21:30	9464	7133	234	6100	1760	48	40	449	406	221	260	0
9	2025-03-18 21:35	9451	7130	234	6012	1779	29	40	449	406	220	260	0
10	2025-03-18 21:40	9256	7125	234	6024	1573	29	40	449	404	214	260	0
11	2025-03-18 21:45	9272	7126	234	6016	1430	29	40	449	405	207	261	0
12	2025-03-18 21:50	9271	7130	234	5973	1262	29	40	449	398	205	260	0
13	2025-03-18 21:55	9141	7133	234	5816	1358	29	40	449	398	208	260	0
14	2025-03-18 22:00	9221	7133	234	5809	511	29	40	449	396	209	260	0
15	2025-03-18 22:05	9557	7128	234	6022	-513	29	40	452	394	204	261	0
16	2025-03-18 22:10	9588	7125	234	6110	-975	29	40	452	391	205	260	0
17	2025-03-18 22:15	9462	7130	234	6021	-648	29	40	452	389	201	260	0
18	2025-03-18 22:20	9436	7131	234	5862	-664	29	40	452	386	200	260	0
19	2025-03-18 22:25	9435	7130	234	5732	-774	29	40	452	385	199	260	0
20	2025-03-18 22:30	9442	7126	234	5580	-502	30	40	452	382	195	260	0
21	2025-03-18 22:35	9454	7124	234	5550	-477	30	40	452	380	194	260	0
22	2025-03-18 22:40	9449	7128	234	5525	-562	30	40	452	383	183	260	0
23	2025-03-18 22:45	9423	7132	234	5520	-838	30	40	452	383	166	261	0
24	2025-03-18 22:50	9388	7129	234	5449	-1073	30	40	453	376	166	260	0

Figura 1.15: *Data Frame* con los nombres de columna corregidos pero repetidos en la fila uno. Además, las fila 0 contiene Nans

Figure 1.15: Data Frame with the column names fixed but repeated in row 1. Besides, row 0 holds NaNs

dejar nuestro *DataFrame* limpio. Para ello, en la línea 14 del scripts, hemos incluido el comando `drop`. Este comando nos permite eliminara líneas enteras de la tabla, empleando directamente su número de índice. En nuestro caso queremos eliminar las filas uno y dos; `generacion_mw.drop([0,1],inplace=True))`. Donde, de nuevo hemos empleado la opción `inplace = True`, para que los cambios se lleven a cabo sobre el propio *DataFrame*. Podemos ver el resultado de la tabla completa en la figura 1.16. Las filas cero y uno se han eliminado, dejándonos una tabla de datos coherente. Sin embargo, los datos, contenidos en las columnas no son numéricos. Para convertirlos en números y poder trabajar con ellos, En la línea 15 copiamos los nombres de las columnas, en una lista,

so, leave our *DataFrame* clean. To do it, we have included the command `drop` in the script line number 14. This command allows us to eliminate complete rows from the table, using directly the row index number. In our case, we want to delete rows zero and one from the table; `generacion_mw.drop([0,1],inplace=True))`. Where, one more time, we have used the option `inplace = True`, to perform the changes on the same *DataFrame*. WE can see the result of the complete table in figure 1.16. Rows zero and one have been deleted casting a coherent data table. Nevertheless, the data contained in the columns are not numerical. We need to change them to numbers to work with. We perform it in two steps, first in line 15 we copy the columns names in a list,



Index	Hora	Eólica	Nuclear	Carbón	clo combinac	tercambios i	lar fotovolta	olar térmica	mica renova	lotores diése	urbina de ga	rbina de vap	eración aux	eración y r
2	2025-03-18 21:00	9830	7131	233	6937	917	48	40	437	407	217	261	0	1970
3	2025-03-18 21:05	9638	7121	234	6852	1152	48	40	437	408	224	261	0	1971
4	2025-03-18 21:10	9590	7130	234	6859	1142	48	40	437	407	226	260	0	1965
5	2025-03-18 21:15	9505	7127	234	6760	1403	48	40	444	411	225	260	0	1959
6	2025-03-18 21:20	9428	7130	234	6515	1459	48	40	449	407	224	261	0	1944
7	2025-03-18 21:25	9459	7127	234	6220	1609	48	40	449	408	224	261	0	1928
8	2025-03-18 21:30	9464	7133	234	6100	1760	48	40	449	406	221	260	0	1889
9	2025-03-18 21:35	9451	7130	234	6012	1779	29	40	449	406	220	260	0	1934
10	2025-03-18 21:40	9256	7125	234	6024	1573	29	40	449	404	214	260	0	1932
11	2025-03-18 21:45	9272	7126	234	6016	1430	29	40	449	405	207	261	0	1934
12	2025-03-18 21:50	9271	7130	234	5973	1262	29	40	449	398	205	260	0	1931
13	2025-03-18 21:55	9141	7133	234	5816	1358	29	40	449	398	208	260	0	1912
14	2025-03-18 22:00	9221	7133	234	5809	511	29	40	449	396	209	260	0	1898
15	2025-03-18 22:05	9557	7128	234	6022	-513	29	40	452	394	204	261	0	1897
16	2025-03-18 22:10	9588	7125	234	6110	-975	29	40	452	391	205	260	0	1854
17	2025-03-18 22:15	9462	7130	234	6021	-648	29	40	452	389	201	260	0	1887
18	2025-03-18 22:20	9436	7131	234	5862	-664	29	40	452	386	200	260	0	1887
19	2025-03-18 22:25	9435	7130	234	5732	-774	29	40	452	385	199	260	0	1887
20	2025-03-18 22:30	9442	7126	234	5580	-502	30	40	452	382	195	260	0	1883
21	2025-03-18 22:35	9454	7124	234	5550	-477	30	40	452	380	194	260	0	1852
22	2025-03-18 22:40	9449	7128	234	5525	-562	30	40	452	383	183	260	0	1889
23	2025-03-18 22:45	9423	7132	234	5520	-838	30	40	452	383	166	261	0	1887
24	2025-03-18 22:50	9388	7129	234	5449	-1073	30	40	453	376	166	260	0	1883

Figura 1.16: *Data Frame* con los nombres de las columnas corregidos y NaNs eliminados
Figure 1.16: Data Frame with the column names fixed and NaNs eliminated

```
In [9]: lista = generacion_mw.columns

In [10]: lista
Out[10]:
Index(['Hora', 'Eólica', 'Nuclear', 'Carbón', 'Ciclo combinado',
'Intercambios int', 'Solar fotovoltaica', 'Solar térmica',
'Térmica renovable', 'Motores diésel', 'Turbina de gas',
'Turbina de vapor', 'Generación auxiliar', 'Cogeneración y residuos',
'Andorra exportación', 'Marruecos exportación', 'Portugal exportación',
'Francia exportación', 'Francia importación', 'Portugal importación',
'Marruecos importación', 'Andorra importación', 'Hidráulica'],
dtype='object')
```

Despues, empleamos un bucle `for` para recorrer las columnas y cambiar el tipo de los datos a `float`. Es interesante observar que, tal y como hemos escrito el código, cambiamos los tipos de todos los datos de cada columna en un solo paso.

El comando `describe` de la línea 19, nos permite hacernos una idea de la potencia de Pandas.

After, we use a loop `for` to rerun the columns and change the data type to `float`. It in interesting to note that the code has be written to change the type of all data in a column in a single step.

The command `describe` shows us the power of Pandas module.

```
In [3]: des = generacion_mw.describe()

In [4]: des
Out[4]:
              Eólica      Nuclear  ...  Andorra importación   Hidráulica
count     361.000000   361.000000  ...                361.0   361.000000
mean    11429.587258  6041.326870  ...                  0.0  5518.722992
std      3260.205128   508.590267  ...                  0.0  1199.772571
min      7740.000000  5640.000000  ...                  0.0  3440.000000
25%      8727.000000  5667.000000  ...                  0.0  4306.000000
50%      9830.000000  5673.000000  ...                  0.0  5640.000000
75%     15167.000000  6410.000000  ...                  0.0  6467.000000
max     17810.000000  7133.000000  ...                  0.0  8061.000000
```

Con una única instrucción, Pandas nos devuelve un estudio estadístico básico de los datos de nuestra tabla. En el terminal de python se no muestran los nombres de las columnas, con los distintos tipos de generación de energía eléctrica, y en cada columna `count,mean, std, min, 25%, 50%, 75%, max` los valores del número total de datos, el valor medio, el valor mínimo, la desviación estándar, y los valores que marcan los cuartiles 25 %, 50 % (la mediana),75 % y el valor máximo. En el terminal de python no se muestran todos los datos de la tabla `des`, solo las dos primeras columnas y las dos últimas y unos puntos suspensivos entre medias para indicarnos que hay más datos.

Veamos con detalle el código contenido en la línea 22. Se trata de un comando asociado a los *DataFrames* que permite realizar gráficos tipo 'pastel'. Lo que queremos es obtener una comparativa de la producción media obtenida de las distintas fuentes de energía. El problema es que la tabla que estamos estudiando contiene valores de producción negativos. Se trata en realidad de los intercambios energéticos como, por ejemplo, `'Francia exportación'` estos datos se toman como negativos porque se trata de energía que 'sale' del sistema español y, por tanto no está disponible para el consumo interno. El problema es que un gráfico tipo pastel, nos da gráficamente una distribución de valores respecto a un total. Debemos excluir los valores negativos. Para ello, dentro del *Dataframe* `des`, escogemos los datos de la fila `mean` pero solo aquellos que

With a single instruction, Pandas renders a basic statistical study of our table data. On the python terminal we can see the names of the different kinds of Electrical power generators and in each column `count,mean, std, min, 25%, 50%, 75%, max` the values corresponding to the total number of data, the mean value, the minimum, the standard deviation and the values taken by the 25 %, 50 % (the median) and 75 % quartiles and the maximum. On the Python terminal no every data of table `des` are shown, but only the two first and two last columns and a ellipsis in between to show us that there are more data available.

Let see in detail the code on line 22. It contains a command associated to *DataFrames* Who allows us to draw Pie charts. We want to get a comparison among the mean power generation from the different generation sources. The problem is that the table we want to study holds negative production values. In fact these correspond to electrical power interchanges with other countries, such France `'Francia exportación'`. These data are taken negative because it is energy that 'gets out' of the Spanish system and, thus, it is not available for inner consumption. The problem is that a pie chart gives a graphic distribution of values that sum up a total quantity. We should exclude the negative value. To do this we chose from the *Dataframe* `des`, the data on the row `mean`, but only those that are greater than zero: `des.loc['mean']>0`. Eventually, we call the command `plot.pie` to draw the graphic. The result is shown if figure 1.17.

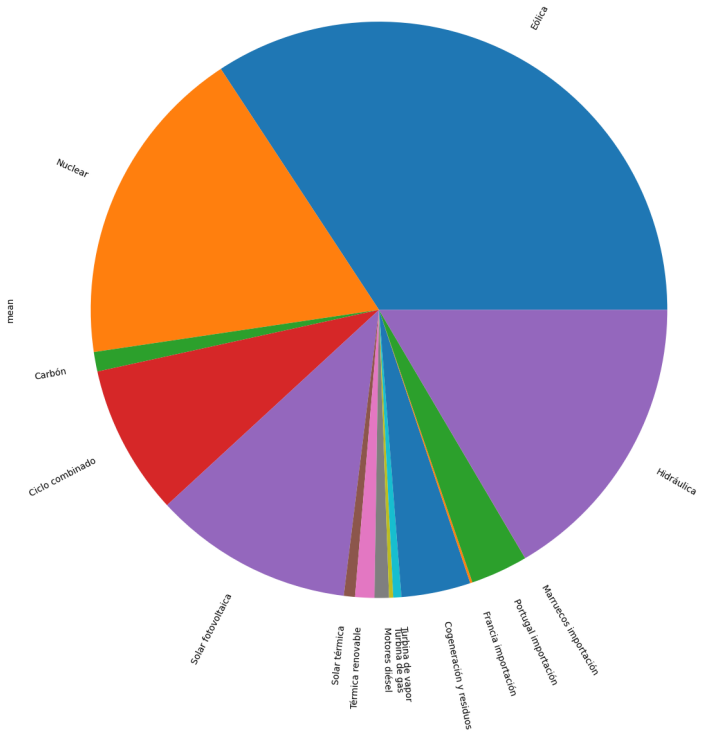


Figura 1.17: Distribución de la producción media de energía eléctrica en Mw en el periodo 18.03.2025:21.00h - 20.03.2025:03.00h
Figure 1.17: Electrical mean energy during the period 03.18.2025:21.00h - 03.20.2025:03.00h

sean mayores que cero: `des.loc['mean']>0`. Por último llamamos al comando `plot.pie`, para que nos dibuje el gráfico. EL resultado se puede ver en la figura 1.17. Es fácil ver la proporción de cada tipo de generación empleado en el periodo que cubren los datos y, en particular, el dominio del consumo de energía eólica.

A continuación, en la línea 24 sustituimos la columna de índices por la columna de horas. De este modo, los datos están directamente etiquetados por el tiempo en que se tomaron. El resultado es un *DataFrame* nuevo, `gn_Mw_h`, en el que la primera columna son las horas.

En la línea 29 empleamos el comando `plot`, asociado al nuevo *Dataframe*. Por defecto, no representará los valores de cada columna con respecto a la columna de índices que ahora

It very clear to see the distribution of each kind of power generation source used in the period covered by the data and, in particular, the dominance of wind power consumption.

Following with the code of the example, in line 24 we change the column of indices by the column of hours. In this way, data are directly label with the time when they were measured. The result is a new *DataFrame*, `gn_Mw_h` where the first columns are the hours.

In line 29 we use the command `plot`, associated to the new *DataFrame*. By default, it will represent de values each column versus the index column, which, now, has the times when the measurements were taken. The pa-

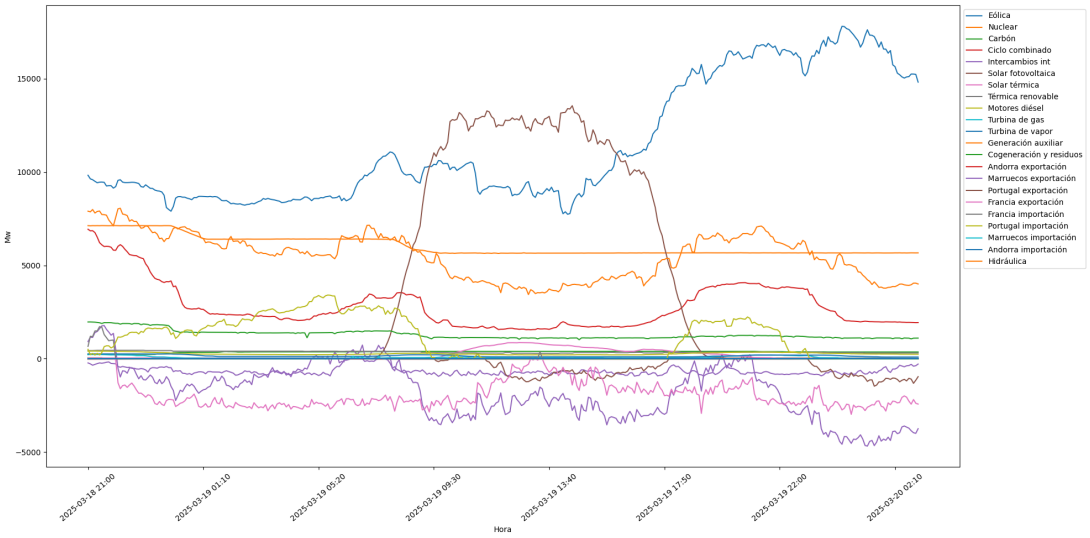


Figura 1.18: *Data Frame* Potencia en megawatios frente al tiempo
Figure 1.18: Electrical Power in megawatts vs time

contiene los tiempos en los que se han realizado las correspondientes medidas. El parámetro `rot`, inclina las etiquetas del eje x de modo que no se solapen y sean legibles. Es imposible cubrir aquí todas las posibilidades de modificación que admiten los gráficos de Pandas que, a su vez, dependen del módulo matplotlib. Para más información se aconseja consultar la documentación de Pandas. La figura 1.18 muestra la gráfica resultante. El eje y representa la potencia en Mw generada. Los valores negativos representan exportaciones de electricidad a otros países, como ya se comentó anteriormente.

La línea de código 32 utiliza el comando `boxplot` para dibujar un diagrama de 'bigotes' de los datos . Se trata de un diagrama muy empleado en estadística para analizar como es la distribución de un conjunto de muestras. Para ello, se obtienen los tres primeros cuartiles, es decir los valores de las muestras que superan al 25 % (Q_1), al 50 % (Q_2 o mediana) y al 75 % (Q_3) de los datos. los cuartiles Q_1 y Q_3 se representa en el gráfico respectivamente como los límites inferior y superior de la 'caja', ademas se señala tambien dentro de la caja la posición de la mediana. El valor $Q_2 - Q_1$ recibe el nombre de rango inrameter `rot` adjusts the tilt of the x-axis labels to prevent overlapping and ensure they remain legible. It is impossible to cover here all possible modifications admitted by Pandas graphics which, in turn, depends on Matplotlib module. For more information it is valuable to consult Panda documentation. Figure 1.18 shows the resulting graphic. The y-axis represents the power generated in Mw. As we said before, negative values are the electrical power exported to other countries.

Line 32 in the code uses command `boxplot` to draw a box and whisker plot. This kind of plot is widely used in statistics to analyse the distribution of a set of samples. To perform it, we obtain the value of the first three quartiles, that is, the sample values that surpass the 25 % (Q_1), the 50 % (Q_2 or median) and the 75 % (Q_3) of the sample data. Quartiles Q_1 y Q_3 are represented in the box plot as the upper and lower limit of the 'box'. In addition, the position of the median is also represented in the box. The value $Q_2 - Q_1$ is known as the interquartile rank (IQR) and usually is taken to draw a pair of lines —the whiskers of

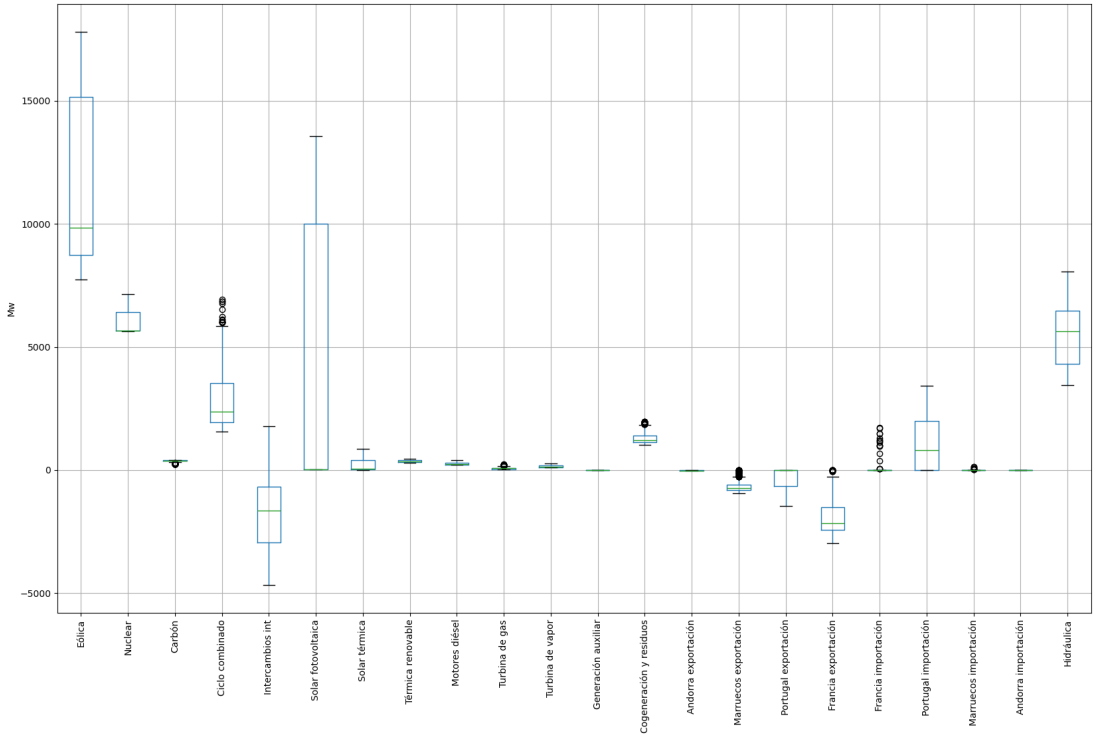
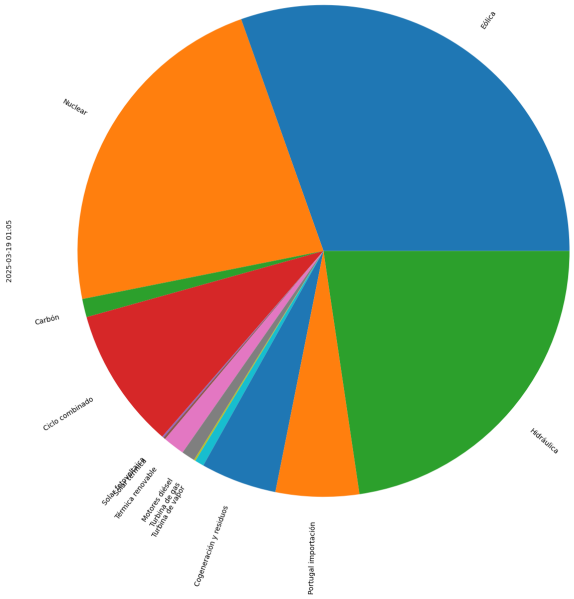


Figura 1.19: Diagrama de bigotes de la potencia generada en Megawatios, distribuido por origen de producción
Figure 1.19: Boxplot of Power generation in megawatts, distributed by production source.

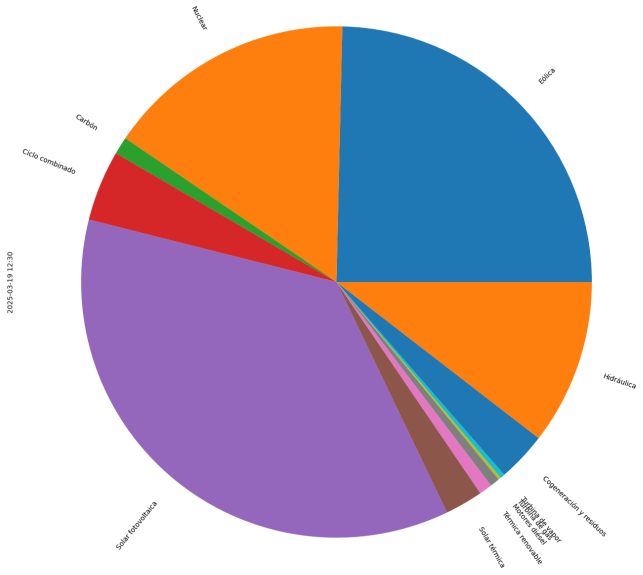
tercuartílico (RIC) y habitualmente se emplea para dibujar unas líneas, —los bigotes del diagrama— que se extienden desde la caja. Para obtener el límite de estas líneas, se calculan los valores $Q_3 + 1.5RIC$ y $Q_1 - 1.5RIC$. De este modo, el límite superior L_s lo marca el valor mayor que cumpla $L_s <= Q_3 + 1.5RIC$ y el límite inferior L_i se elige como el valor menor que cumpla $L_i >= Q_1 - 1.5RIC$.

La figura 1.19 muestra el diagrama de bigotes obtenidos a partir del *DataFrame* `gn_Mw_h`. Los datos aparecen distribuidos por la fuente de generación de energía, es decir, por las columnas del *DataFrame*. Si nos fijamos en la primera entrada, `Eólica`, observamos que la caja del diagrama, para este tipo de energía va desde apróximadamente desde los 8000 hasta un poco más de 15000 Mw. La mediana, aparece representada como una línea horizontal en la mitad inferior de la caja, estaría alrededor de los 9000 Mw. El límite del bigote inferior —los bigotes del diagrama— que se extienden desde la caja. To obtain the limit of these lines, we compute the values $Q_3 + 1.5RIC$ y $Q_1 - 1.5RIC$. So, the upper limit L_s is defined by the largest value of the sample, which fulfils that $L_s <= Q_3 + 1.5RIC$ and the lower limit L_i is chosen as the lower value, which fulfils that $L_i >= Q_1 - 1.5RIC$.

Figure 1.19 shows a box and whisker plot drawn with the *DataFrame* `gn_Mw_h`. Data are distributed according to the generation source, that is, according to the *DataFrame* columns. If we focus on the first entry, `Eólica`, We ca see that the box, for this type of energy cover approximately from 8000 to a bit more than 15000 Mw. The median is represented as a horizontal line in the lower middle of the box. It would be around 9000 Mw. The whisker lower limit would be close to 6000 Mw and the upper one near to 18000 . In some columns



(a) Distribución de la generación a las 01.05h am
(a) Generation distribution at 01.05h am



(b) Distribución de la generación a las 12.30h am
(b) Generation distribution at 12.30h am

Figura 1.20: Comparación entre la distribución de generación de energía eléctrica a la 01h am y al las 12.30h pm
Figure 1.20: Comparison between the distribution of electricity generation at 01h am and 12.30h pm

rior estaría cerca de los 6000 Mw y wl límite del bigote superior cerca de 18000. En algunas columnas, como por ejemplo en la que representa la generación de electricidad mediante turbinas de ciclo combinado, se observan unos puntos negros más allá de límite de los bigotes. Representan datos atípicos, es decir, valores que no encajan con la distribución estadística del resto de los datos.

La línea de código 36 extrae la fila de datos correspondientes al tiempo `'2025-03-19 01:05'` y la línea 37 la fila correspondiente al tiempo `'2025-03-19 12:30'`. A continuación las líneas 41 y 42, dibujan sendos diagramas de pastel, que nos permiten comparar como cambia la distribución de la generación entre ambas horas (una de la madrugada, doce y media del mediodía. Algunas diferencias obvias son la gran generación de energía solar fotovoltaica a mediadía, inexistente a la una de la madrugada.

we can see some black dots beyond the whisker limits. See, for instance, the ciclo combinado turbines generation. These black dots represent outlayers. That is, data that do not fit with the statistical distribution of the remaining data.

The code line 36 extracts the data row corresponding to time `'2025-03-19 01:05'` and line 37 the data line corresponding to time `'2025-03-19 12:30'`. Then, lines 41 and 42, draw two pie charts with allow us to compare how the power generation distribution changes between these two times (one in the morning and noon). Some obvious differences are the huge generation of solar photovoltaic at noon that does not exists at night.

Índice alfabético

Alphabetic Index