

Apunts CE_5074 2.1

Iloc: [Institut d'Ensenyaments a Distància de les Illes Balears](#)
Curs: Sistemes de Big Data
Llibre: Apunts CE_5074 2.1

Imprès per: Carlos Sanchez Recio
Data: dilluns, 28 d'octubre 2024, 16:06

Taula de continguts

1. Introducció
2. Histograma i valors extrems
3. Tendències centrals
4. Dispersió
5. Correlació
6. La paradoxa de Simpson i els factors de confusió
7. Distribució normal d'una variable aleatòria
8. La taula normal estàndard
9. Mostres i intervals de confiança
10. La distribució binomial
11. Més informació

1. Introducció

Quan tenim un dataset (conjunt de dades) petit, les pròpies dades ens poden donar la millor descripció possible. Potser amb una simple ullada a les dades ja podem extreure conclusions rellevants. Però quan el dataset creix, necessitam altres eines. És per això que utilitzam l'estadística.

En aquest lliurament introduirem diferents conceptes estadístics. I ho farem d'una manera pràctica, utilitzant Python i algunes de les llibreries que s'han vist en el segon lliurament del mòdul de Programació d'Intel·ligència Artificial.

L'objectiu d'aquest llibre d'apunts no és donar-vos un tractat formal i matemàticament rigorós dels fonaments de l'estadística, sinó introduir els conceptes que necessitarem conèixer, tant per a l'anàlisi de dades massives com per a l'aprenentatge automàtic. De fet, al llarg d'aquest tema es faran algunes simplificacions de la teoria matemàtica que hi ha per sota dels conceptes que es van introduint, per tal de fer més senzilla la seva comprensió.

En el següent vídeo pots veure un resum del que tractarem en aquest lliurament.

SBD L2: Fonaments d'estadística per a l'anàlisi de d...



Vídeo: Resum del Lliurament 2

2. Histograma i valors extrems

Una primera aproximació per entendre les dades d'un dataset és veure el seu **histograma**.



L'histograma és una gràfica que ens mostra quantes vegades apareix cada un dels possibles valors d'una variable. Té dues dimensions: en l'eix X (horizontal) tenim els valors que pot prendre la variable, mentre que a l'eix Y (vertical) tenim el número de repeticions.

DEFINICIÓ

Vegem un exemple. Imaginem que tenim un dataset amb les edats dels alumnes de l'IEDIB. Això ho podem representar en Python mitjançant una variable *edats*, de tipus llista:

```
edats = [33, 27, 48, 63, 46, 60, 57, 48, 40, 33, 65, 62, 26, 23, 28, 32, 22, 22, 55, 38, 24, 33, 31, 49, 59, 63, 43, 49, 45, 23, 21, 32, 24, 18, 60, 17, 53, 49, 57, 48, 26, 49, 20, 65, 44, 24, 44, 20, 65, 26, 25, 45, 48, 19, 43, 25, 32, 29, 35, 26, 36, 30, 17, 38, 35, 32, 33, 27, 19, 44, 49, 20, 40, 21, 50, 30, 23, 34, 21, 24, 25, 50, 23, 18, 31, 25, 35, 31, 32, 28, 17, 39, 21, 48, 33, 43, 47, 38, 26, 42, 48, 33, 23, 34, 44, 46, 33, 46, 48, 34, 39, 25, 40, 34, 45, 22, 37, 47, 47, 29, 20, 35, 36, 44, 23, 48, 30, 21, 34, 31, 49, 44, 47, 46, 44, 33, 37, 50, 31, 25, 28, 39, 32, 24, 24, 23, 32, 38, 39, 38, 38, 38, 39, 36, 37, 40, 39, 40, 40, 38, 37, 35, 42, 38, 35, 44, 43, 43, 44, 44, 36, 45, 43, 43, 38, 35, 41, 36, 38, 44, 37, 45, 38, 38, 38, 41, 39, 45, 35, 35, 41, 42, 40, 40, 37, 45]
```



En aquests apunts representarem els datasets mitjançant llistes de Python. D'aquesta manera, per fer més senzilla l'explicació, assumim que un dataset conté les dades d'una única variable estadística (l'edat dels alumnes en aquest exemple; més endavant farem feina amb altres variables estadístiques com la nota o les hores de dedicació).

En la realitat, és més habitual tenir dades que provenen, per exemple, d'un fitxer CSV i treballar amb diverses columnes (on cada columna és una variable estadística diferent). En aquest cas, empraríem els *dataframes* de la llibreria *pandas*. En qualsevol cas, és molt senzill convertir un dataframe (*df*) a una llista:

```
df.values.tolist()
```

I també convertir una columna concreta (*nom_columna*), és a dir, una variable estadística, d'un dataframe a una llista:

```
df['nom_columna'].values.tolist()
```

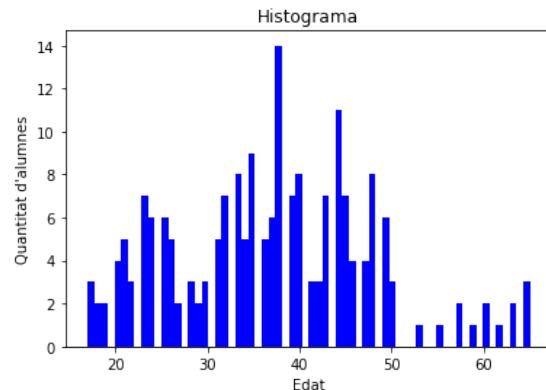
En aquests apunts normalment donarem la implementació dels conceptes que anam introduint mitjançant codi Python i les llibreries *matplotlib* per representar les gràfiques i *numpy* per fer càculs. Si voleu els equivalents amb *pandas*, podeu consultar la documentació de la llibreria.

ALERTA

Al segon lliurament del mòdul de Programació d'Intel·ligència Artificial ja hem vist com generar un histograma fent servir el mètode [*hist*](#) de *matplotlib.pyplot*:

```
max_edat = max(edats)
import matplotlib.pyplot as plt
plt.title("Histograma")
plt.xlabel("Edat")
plt.ylabel("Quantitat d'alumnes")
plt.hist(edats, bins = max_edat+1, color = 'blue')
plt.show()
```

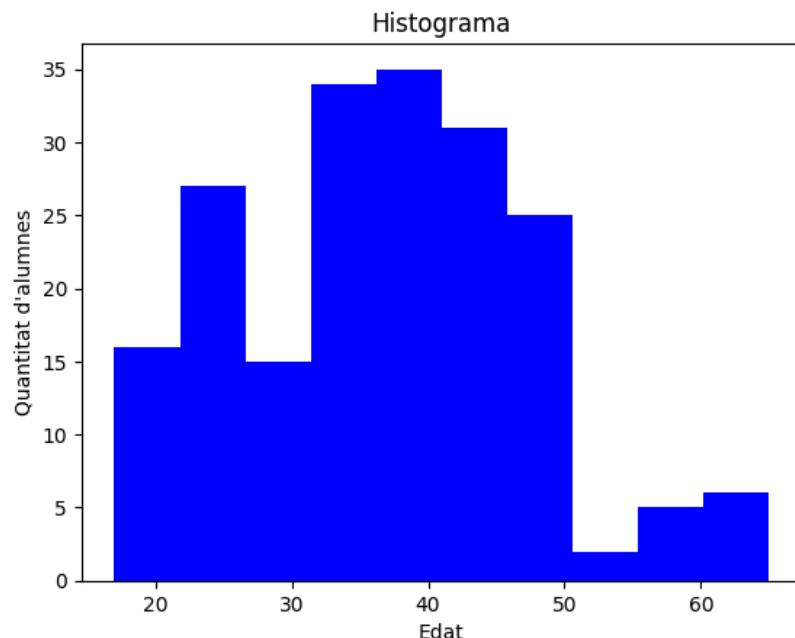
Vegem el resultat:



Imatge: Histograma del dataset d'edats

Aquesta imatge ja ens dona algunes pistes sobre el nostre dataset. A primera vista, ja detectam que hi ha alguns alumnes menors de 20 anys, també alguns de més de 60 i sembla que la majoria estan concentrats en el rang entre 30 i 50.

Fixau-vos que hem emprat un espai (*bin*) per a cada possible valor de l'edat (des de 0 fins al màxim d'edat). Si empràssim un número menor de *bins*, veuríem les dades agrupades per rangs d'edat. Vegem, per exemple, l'histograma amb 10 *bins*:



Imatge: Histograma del dataset d'edats amb 10 bins



Aquests espais (*bins*) de l'eix X de l'histograma reben diversos noms com ara grup, classe o interval.

ACLARIMENT

En anglès, a més de *bin*, també es servir molt sovint *bucket*, així com *group*, *class* o *interval*.



A la pràctica, sovint no és fàcil decidir el nombre de classes o *bins* d'un histograma. Si utilitzam poques classes, es poden amagar algunes de les característiques de les dades. I, en canvi, si n'utilitzam moltes, ens queden moltes classes sense cap element o instància.

Molts paquets estadístics, per defecte, fan servir l'anomenada regla de Sturges: si tenim N elements (196 en el nostre exemple), es trien k classes o bins, on k es calcula segons la següent fórmula:

$$k = 1 + \log_2 N$$

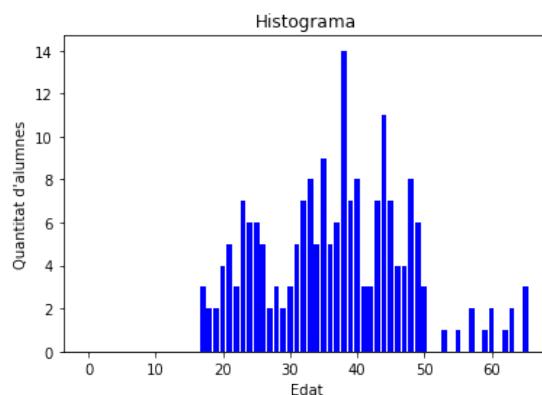
AMPLIACIÓ

En el nostre cas, el $\log_2 196$ és 7,6. Així que, arrodonint, un número adequat de classes segons Sturges seria 9.

Vegem una altra implementació que potser ajuda a entendre millor el concepte d'histograma. Ara emprarem la classe *Counter*, una subclasse del tipus *dict* (diccionari). Amb la línia `edats_comptador = Counter(edats)` ens genera un comptador dels valors: a la posició x tendrem el número d'instàncies del valor x dins la llista `edats`. Això és precisament el que volem representar amb un histograma, així que només hem de posar a l'eix x els valors entre 0 i l'edat més gran (`xs = range(max_edat+1)`) i en l'eix y els valors del nostre comptador. I a partir d'això feim un diagrama de barres, amb el mètode `bar` de la classe *matplotlib.pyplot*:

```
from collections import Counter
edats_comptador = Counter(edats)
plt.title("Histograma")
plt.xlabel("Edat")
plt.ylabel("Quantitat d'alumnes")
xs = range(max_edat+1)
ys = [edats_comptador[x] for x in xs]
plt.bar(xs, ys, color = 'blue')
plt.show()
```

El resultat és semblant a l'anterior (la única petita diferència és que en l'anterior l'eix x s'ajusta millor al rang de les dades):



Imatge: Una altra versió de l'histograma de les edats

A més de l'histograma, sol ser important conèixer el **nombre d'instàncies** del nostre dataset, així com els valors extrems, és a dir el **màxim** i el **mínim**:

```
num_alumnes = len(edats)
print("Número d'alumnes: ", num_alumnes)
min_edats = min(edats)
print("Edat de l'alumne més jove: ", min_edats)
max_edats = max(edats)
print("Edat de l'alumne més vell: ", max_edats)
```

En el nostre cas, tenim 196 alumnes, on el més jove té 17 anys i el més gran 65.

3. Tendències centrals

Normalment ens interessa conèixer on estan centrades les nostres dades. Normalment empram la **mitjana** (*media* en castellà, *mean* o *average* en anglès), que és la suma de tots els valors, dividit pel número de valors:

```
from typing import List
def mitjana(llista: List[float]) -> float:
    return sum(llista) / len(llista)
edat_mitjana = mitjana(edats)
print(edat_mitjana)
```

En el nostre cas, l'edat mitjana és de 36,91 anys.

La llibreria numpy, que ofereix un gran ventall d'eines per a l'estadística, ens proporciona una funció [mean](#) per calcular directament la mitjana:

```
import numpy as np
edat_mitjana = np.mean(edats)
print(edat_mitjana)
```

Si tenim un dataset amb només dos valors, la mitjana és simplement el punt a mig camí entre ells dos. A mesura que anam afegint més punts, la mitjana canvia, però sempre depèn del valor de cada punt. Per exemple, si tenim 10 valors i augmentam un d'ells en 1, la mitjana augmentarà en 0,1 (1/10).

La mitjana és molt sensible als **valors atípics**, el que en anglès es denominen **outliers**. Un exemple molt conegut és el d'una estadística que va fer la Universitat de Carolina del Nord a mitjans dels anys 80 sobre els salaris que tenien els seus titulats quan començaven a fer feina. La manera de fer-ho és senzilla: s'agrupen els alumnes per les seves titulacions i es calcula la mitjana dels salaris per cada una d'elles. Sorprenentment, la titulació amb uns majors ingressos va resultar ser Geografia. Hi havia una raó: Michel Jordan va ser estudiant de Geografia d'aquesta universitat. El seu salari d'entrada a la NBA era tan alt, que distorsionava completament la mitjana de la seva titulació. Cobraven més la majoria dels geògrafs que els titulats en Dret? No. Però, gràcies a (o per culpa de) Michael Jordan, la mitjana sí era superior. Aquest exemple ens mostra que hem d'anar sempre molt alerta a extreure conclusions quan xerrem de mitjanes.

La mitjana molt sovint es representa amb la lletra grega μ (mu).

Existeixen altres mesures que intenten ser més tolerants als *outliers*. Una d'elles és la **mediana** (*mediana* en castellà i *median* en anglès). No hem de confondre mitjana i mediana (*media* i *mediana* en castellà). Per calcular la mediana, primer ordenam les dades, de menor a major, i després agafam la que ens queda just al mig (si el número de valors és senar). Si el número de valors és parell, emprarem la mitjana entre els dos valors que ens queden al mig.

Imaginem que tenim una llista ordenada amb 5 valors (les posicions van del 0 al 4). La mitjana serà l'element que queda a la posició 2 ($5 // 2$, és a dir, divisió entera, sense decimals, de 5 entre 2). Si en tengués 6 valors, agafaríem la mitjana entre les posicions 2 i 3: agafam $6//2$ (3) i l'anterior, $6//2-1$ (2).

En el cas de la nostra llista amb edats, que teníem 196 valors, una vegada ordenats, haurem de prendre la mitjana entre els valors de les posicions 98 i 97.

Escriurem dues funcions per als casos senars i parells i les combinarem:

```

def _mediana_imparell(llista: List[float]) -> float:
    return sorted(xs)[len(llista) // 2]
def _mediana_parell(llista: List[float]) -> float:
    llista_ordenada = sorted(llista)
    punt_mig = len(llista) // 2
    return (llista_ordenada[punt_mig - 1] + llista_ordenada[punt_mig]) / 2
def mediana(llista: List[float]) -> float:
    return _mediana_parell(llista) if len(llista) % 2 == 0 else _mediana_imparell(llista)
edat_mediana = mediana(edats)
print(edat_mediana)

```

El resultat és 37,5 anys.

També podem emprar la funció [median](#) de numpy:

```

edat_mediana = np.median(edats)
print(edat_mediana)

```

És important entendre que la mediana no depèn completament de cada un dels valors de les dades. Per exemple, si modificam el valor més gros o més petit, la mediana no es veu afectada, ja que els punts del mig segueixen essent els mateixos. És a dir, si l'alumne més gran té, en lloc de 65, 70 anys, la mediana seguiria essent de 37,5. En canvi, aquest canvi (com qualsevol altre), sí que afectaria a la mitjana.

En el cas que hem comentat abans d'en Michael Jordan, el seu alt salari no influeix en la mediana, perquè quedarà com el valor més gran i no afectarà a quins són els que estan en mig de la llista ordenada. Així doncs, Geografia no era la titulació amb una mediana més gran.

Emprarem normalment medianes en lloc de mitjanes? No necessàriament. El concepte de mitjana és més intuïtiu que el de mediana. I amb dades més o menys uniformes, reflecteix millor la realitat i recull els canvis que es vagin produint. A més, la mitjana és més simple de calcular, ja que la mediana necessita una ordenació prèvia de la llista de valors.

Una generalització del concepte de mediana són els **percentils** i **quantils**. Mentre que la mediana ens diu el valor per sota del qual tenim el 50% de les dades, el percentil s'aplica a qualsevol percentatge donat de les dades. D'aquesta manera, el percentil 90 de les dades ens diu el valor per sota del qual tenim el 90% de les dades. En general, el percentil p (on p és un número entre 0 i 100) és el valor que marca el tall de manera que el $p\%$ dels valors són iguals o inferiors a aquest valor. En concret, el percentil 50 és el mateix que la mediana.

El quantil és el mateix que el percentil, però expressant els percentatges no amb un número entre 0 i 100, sinó mitjançant un número real entre 0 i 1 (per exemple, el 50% és el 0.5). Així, el quantil d'ordre 0,9 és el mateix que el percentil 90.

Vegem una implementació del quantil en Python:

```

def quantil(llista: List[float], p: float) -> float:
    p_index = int(p * len(llista))
    return sorted(llista)[p_index]

```

Alguns exemples d'ús habitual són:

- quantil(edats, 0.1) retorna 23: un 10% dels alumnes tenen 23 anys o menys
- quantil(edats, 0.25) retorna 29: un 25% dels alumnes tenen 29 anys o menys
- quantil(edats, 0.75) retorna 44: un 75% dels alumnes tenen 44 anys o menys
- quantil(edats, 0.9) retorna 49: un 90% dels alumnes tenen 49 anys o menys

La llibreria numpy també ofereix una funció [quantile](#) per a calcular els quantils. I també una funció [percentile](#) per als percentils, on aquí el paràmetre serà un número entre 0 i 100:

```
quantil_09 = np.quantile(edats, 0.9)
print(quantil_09)
percentil_90 = np.percentile(edats, 90)
print(percentil_90)
```

En ambdós casos s'imprimeix el valor 49.0.

Un altra mesura estadística que de vegades resulta rellevant és la **moda**, que ens diu quin és el valor més repetit. Segurament haureu sentit xerrar del salari moda, és a dir el salari que més persones tenen, que pot ser més significatiu que el salari mitjà, el qual es pot veure desvirtuat pels *outliers*. Per exemple, en un territori on hi resideixen algunes poques persones amb grans fortunes fa que el salari mitjà de tot el territori pugi moltíssim, tot i que la majoria de la gent té un sou per davall d'aquesta mitjana. Vegem una implementació de la moda en Python:

```
def moda(llista: List[float]) -> List[float]:
    frequencies = Counter(llista)
    mes_frequent = max(frequencies.values())
    modes = [clau for clau, valor in frequencies.items()
             if valor == mes_frequent]
    return modes
edat_moda = moda(edats)
print(edat_moda)
```

En aquest cas imprimeix [38]. És 38 perquè hi ha 14 alumnes amb 38 anys, mentre que la segona edat amb més alumnes és 44 anys, amb 11 alumnes. Notau que la funció moda retorna una llista, perquè pot ser que tenguem diverses edats amb el mateix número d'alumnes.

La llibreria numpy no té cap funció per obtenir la moda. En canvi, sí que la podem trobar en la llibreria **SciPy**, que conté un conjunt d'utilitats per resoldre diversos problemes de computació científica, relacionats amb estadística, àlgebra lineal, tractament d'imatges, dades espacials, etc. En concret, trobarem la funció [mode](#) en el paquet d'estadístiques (**stats**):

```
import scipy.stats as stats
stats.mode(edats)
```

Això ens retorna un objecte *ModeResult* que conté el valor més repetit (atribut *mode*) i el número de vegades que es repeteix (atribut *count*).



La mesura més utilitzada és la mitjana, però és important entendre que no sempre serà la més significativa, ja que es pot veure alterada per la presència d'*outliers*. En aquests casos pot resultar més rellevant la mediana o la moda.

RESUM

4. Dispersió

Mentre que en l'apartat anterior hem vist com triar un valor representatiu de les dades, normalment la mitjana, ara volem saber si els valors estan més o menys concentrats al voltant d'aquesta mitjana. És el que anomenam dispersió.

Suposem un cas extrem en què tots els alumnes tenen la mateixa edat, per exemple 40 anys. En aquest cas, no hi ha cap dispersió. Però podríem tenir un altre conjunt de dades on la meitat d'alumnes tenen 20 i l'altra meitat tenen 60. La mitjana és la mateixa, però les dades són més disperses que en el cas anterior.

Un primer indicador de la dispersió és el **rang** de valors, que ve definit per la diferència entre el valor més gran i el més petit.

```
def rang(llista: List[float]) -> float:
    return max(llista) - min(llista)
print(rang(edats))
```

Per a la llista d'edats dels alumnes de l'IEDIB, ens donaria un valor de 48. En el cas en què tots els alumnes tenen 40 anys, el rang seria 0, perquè el mínim i el màxim són iguals. En l'altre cas que hem comentat, la meitat de 20 i la meitat de 60, el rang seria de 40.

De totes formes, aquesta no és una mesura massa precisa, perquè no ens permet veure si són molts els valors que estan als extrems o pocs. Si tots els alumnes tenen una edat de 40, excepte un que té 80, el rang també és 40. Però realment les dades estan molt concentrades en els 40 anys, només un alumne se'n surt. Necessitam tenir més detalls sobre com estan repartides les dades dins d'aquest rang.

És per a això que es defineix la **variància**, que ens dona una idea més precisa de com d'enfora estan les dades respecte de la mitjana. La idea és calcular, per a cada valor, la diferència amb la mitjana. Com que de vegades la diferència pot ser positiva i d'altres negativa, s'utilitza el quadrat d'aquesta diferència, que sempre serà positiu. Aleshores, sumam tots aquests quadrats de les diferències i ho dividim pel número de punts. El resultat és la variància. Aquesta és la seva fórmula matemàtica:

$$v = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_N - \bar{x})^2}{N} = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

on

- N és el nombre de valors de la llista x
- x_1, x_2, \dots, x_N són tots els valors de la llista
- \bar{x} és la mitjana

El sumatori

La fórmula que hem vist abans incorpora un símbol matemàtic (\sum) que ens interessa aprendre, perquè es fa servir molt en matemàtiques i en estadística. Es tracta d'un sumatori. A la part de baix posam el valor inicial des d'on volem sumar i a la de dalt el valor final. Vegem un exemple. Tenim una variable x amb N valors, x_1, x_2, \dots, x_N , aleshores

$$\sum_{i=1}^N x_i$$

Vol dir que per a $i=1$ fins a N sumarem totes les x_i . Per tant:

$$\sum_{i=1}^N x_i = x_1 + x_2 + \dots + x_n$$

Per tant, podem escriure aquest sumatori en Python mitjançant el següent bucle:

```
sumatori = 0
for i in range(1, N)
    sumatori += x[i]
```

Vegem ara el codi Python per obtenir la variància:

```
def variancia(llista: List[float]) -> float:
    xm = mitjana(llista)
    sumatori = 0
    for xi in llista:
        sumatori += (xi - xm)**2
    return sumatori / len(llista)
edats_variancia = variancia(edats)
print(edats_variancia)
```

El resultat és 115,364926...

La llibreria *numpy* també incorpora una funció [var](#) per calcular la variància:

```
import numpy as np
print(np.var(edats))
```

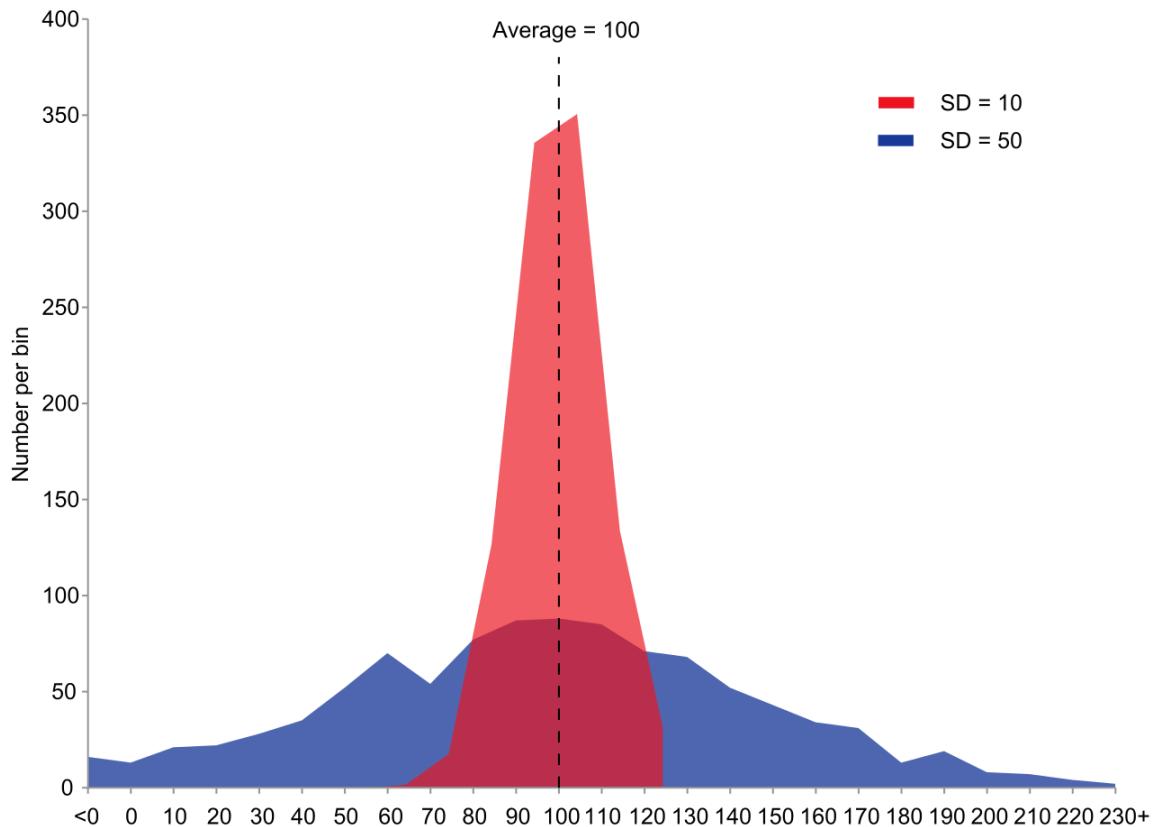
El problema de la variància és que, atès que hem fet una operació d'elevar al quadrat, les seves unitats no són massa intuïtives. Si, com en el nostre exemple, les dades venen expressades en anys, la variància queda expressada en anys al quadrat (any^2). Per evitar això, se li sol aplicar l'arrel quadrada, de manera que les unitats tornen a estar en anys. Aquesta nova mesura, l'arrel quadrada de la variància, es denomina **desviació estàndard** (o **desviació tipus** o **desviació típica**) i molt sovint se la representa mitjançant la lletra grega σ (sigma) o la lletra S. També és freqüent emprar les sigles SD (de *standard deviation*). Vegem el codi Python corresponent:

```
def sd(llista: List[float]) -> float:
    return math.sqrt(edats_variancia)
edats_sd = sd(edats)
print(edats_sd)
```

La llibreria *numpy* inclou la funció [std](#) per calcular la desviació estàndard:

```
import numpy as np
print(np.std(edats))
```

Com més baixa sigui la desviació estàndard, més concentrats estaran els valors al voltant de la mitjana. La següent imatge mostra dues llistes de valors, on les dues tenen una mateixa mitjana (100). La vermella està molt més concentrada i té una desviació estàndard de 10, mentre que la blava, més dispersa, la té de 50.



Imatge: Dues variables amb igual mitjana però diferent desviació estàndard. Font: Wikipedia Commons

La desviació estàndard és la mesura més emprada per definir la dispersió de les dades. Però, al igual que la mitjana, també és molt sensible als *outliers*. És per això que, de vegades, també s'empra la diferència entre el percentil 25 i el percentil 75, ja que així, es descarten els valors atípics que estan molt enfora de la mitjana.

```
def rang_interquartils(llista: List[float]) -> float:
    return quantil(llista, 0.75) - quantil(llista, 0.25)
print(rang_interquartils(edats))
```

5. Correlació

Ens han passat un altre dataset, que conté el número d'hores de dedicació setmanal de mitjana per mòdul o assignatura de cada alumne. Tenim les dades en la següent llista de Python:

```
hores = [3.4, 3.1, 4.7, 6.4, 4.9, 5.9, 5.9, 5.0, 4.5, 3.5, 6.8, 6.0, 2.2,
2.4, 3.2, 3.3, 1.8, 2.1, 5.3, 3.8, 2.5, 3.3, 3.6, 4.8, 5.8, 6.3, 4.4, 4.5,
4.8, 1.9, 2.5, 3.3, 2.2, 1.8, 6.1, 2.0, 5.0, 4.7, 5.2, 5.0, 2.3, 5.0, 2.1,
6.8, 4.4, 2.5, 4.6, 2.3, 6.0, 3.1, 2.8, 4.4, 5.2, 1.6, 4.8, 3.0, 3.6, 2.4,
3.0, 2.5, 3.4, 3.5, 2.1, 4.1, 3.2, 3.5, 3.1, 2.9, 1.7, 4.8, 4.7, 2.3, 4.5,
1.6, 4.7, 3.5, 2.8, 3.9, 1.9, 2.6, 2.5, 5.5, 1.8, 1.3, 2.8, 2.4, 3.6, 3.0,
3.1, 2.7, 1.8, 3.4, 2.0, 5.2, 3.4, 4.1, 5.2, 3.7, 2.5, 4.1, 4.5, 3.7, 2.3,
3.9, 4.1, 5.0, 3.3, 4.5, 5.3, 3.4, 4.3, 2.5, 4.5, 3.3, 5.0, 2.2, 3.9, 4.2,
4.2, 3.2, 1.8, 3.0, 3.7, 4.5, 2.6, 4.3, 2.9, 2.1, 3.4, 2.7, 4.5, 4.8, 4.9,
4.3, 4.2, 3.3, 3.8, 5.3, 2.7, 2.5, 2.7, 3.4, 3.6, 2.8, 2.2, 2.7, 3.0, 4.2,
4.3, 3.9, 3.7, 4.3, 3.5, 3.5, 3.4, 3.8, 4.2, 4.3, 4.5, 3.7, 3.8, 3.4, 4.3, 4.0,
4.3, 3.6, 3.9, 4.5, 4.8, 4.4, 4.8, 3.9, 4.4, 4.7, 3.9, 3.8, 3.4, 4.3, 3.2,
3.6, 4.6, 4.0, 4.7, 3.7, 4.0, 4.3, 4.3, 3.7, 4.0, 3.6, 3.3, 3.9, 4.6, 3.9,
4.0, 3.3, 4.2]
```



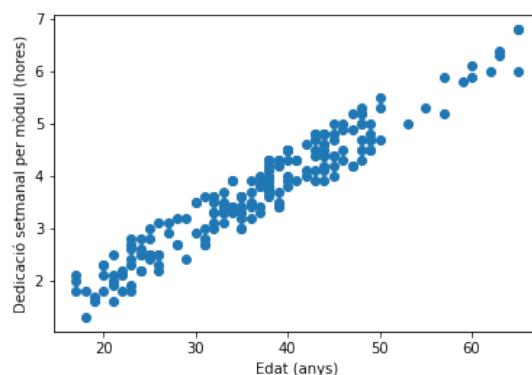
Les llistes *hores* i *edats* fan referència als mateixos alumnes i estan ordenades de la mateixa manera.

És a dir, *hores[x]* fa referència al mateix alumne que *edats[x]*.

Pel que ens han comentat alguns professors, tenim la impressió que els alumnes més grans dediquen més temps que els més joves i ens agradarà confirmar aquesta hipòtesi.

Primer de tot, anam a veure gràficament les dades, posant en un eix l'edat i en l'altre les hores de dedicació:

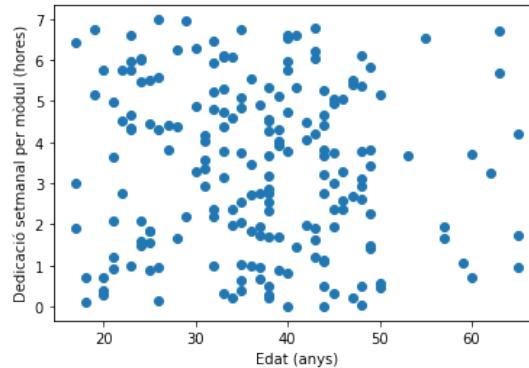
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.set_xlabel('Edat (anys)')
ax.set_ylabel('Dedicació setmanal per mòdul (hores)')
ax.scatter(edats, hores)
```



imatge: Gràfica de les edats i dedicació setmanal per mòdul: relació lineal

Si ens fixam en la gràfica, podem veure clarament que les dades estan totes al voltant d'una recta. Això ens indica que les dues variables estan relacionades. A valors petits d'edat tenim sempre valors petits de dedicació, mentre que a valors grans d'edat, tenim valors grans de dedicació. És a dir, a mesura que ens allunyam de la mitjana d'una de les variables, ens allunyam també de la mitjana de l'altra, en el mateix sentit. En conseqüència ja podem confirmar que existeix el que s'anomena una **relació lineal** entre les dues variables. Tot això confirma la nostra hipòtesi inicial que els alumnes més grans dediquen més temps de treball als mòduls/assignatures.

Vegem un exemple de gràfica on no existeix relació lineal. Per fer-la, hem emprat altres valors (generats aleatoriament) per a la variable hores. Aquest és el resultat:

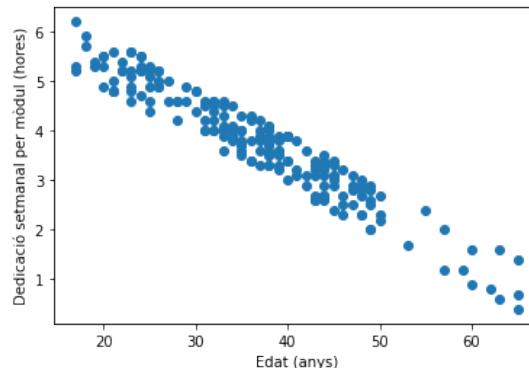


Imatge: Gràfica de les edats i una altra dedicació setmanal per mòdul: sense relació lineal

Amb aquestes noves dades, podem veure clarament que no existeix cap relació lineal entre les dues variables.

Es pot donar un altre cas, com el que podem veure a la següent gràfica, amb uns altres valors de la variable hores:

```
hores2 = [3.6, 4.6, 2.9, 1.6, 2.7, 1.6, 1.2, 2.9, 3.4, 4.1, 1.4, 0.8, 5.2, 5.1, 4.2, 4.6, 5.2, 5.2, 2.4, 3.5, 5.2, 4.0, 4.6, 2.9, 1.2, 0.6, 2.6, 2.5, 3.0, 5.6, 4.8, 4.2, 5.5, 5.9, 0.9, 6.2, 1.7, 2.8, 2.0, 3.0, 4.9, 2.6, 5.5, 0.7, 2.7, 4.7, 3.4, 4.9, 0.4, 5.2, 4.9, 3.4, 2.3, 5.4, 2.6, 4.4, 4.0, 4.9, 3.7, 4.9, 3.4, 4.4, 5.3, 4.0, 3.5, 4.0, 3.9, 5.0, 5.3, 2.6, 2.0, 5.5, 3.0, 5.6, 2.3, 4.8, 5.6, 3.8, 4.8, 5.3, 5.2, 2.7, 5.2, 5.7, 4.0, 4.6, 4.0, 4.0, 4.4, 4.6, 5.2, 3.9, 5.0, 2.9, 4.3, 3.2, 3.1, 4.1, 5.0, 2.9, 2.8, 4.5, 4.8, 4.0, 2.7, 2.3, 4.0, 2.5, 2.3, 4.5, 3.6, 5.3, 3.9, 3.9, 3.1, 5.4, 3.9, 2.5, 2.8, 4.6, 5.3, 3.7, 3.8, 3.5, 4.6, 2.7, 4.8, 4.8, 4.1, 4.2, 2.0, 3.3, 2.9, 3.2, 2.9, 4.6, 3.6, 2.2, 4.5, 5.1, 4.6, 3.8, 4.6, 5.2, 5.5, 4.9, 4.5, 3.9, 3.2, 3.8, 3.5, 3.6, 3.4, 4.0, 3.9, 3.3, 3.9, 3.9, 3.7, 3.3, 4.0, 3.6, 3.8, 4.3, 3.3, 3.3, 2.7, 3.1, 3.2, 4.2, 2.9, 3.4, 3.1, 3.3, 3.6, 3.1, 4.3, 4.1, 2.7, 3.8, 3.1, 3.8, 3.3, 3.7, 3.2, 3.5, 3.3, 4.0, 3.8, 3.1, 3.9, 3.4, 4.2, 2.4]
```



Imatge: Gràfica de les edats i una altra dedicació setmanal per mòdul: relació lineal inversa

Aquí podem veure clarament que les dades també estan totes al voltant d'una recta. Però fixau-vos que la inclinació (pendent) de la recta és diferent a la que hem vist abans. D'aquesta recta se'n diu que té un pendent negatiu (mirant-la d'esquerra a dreta, va cap avall), mentre que l'anterior tenia un pendent positiu (d'esquerra a dreta, anava cap amunt). En aquest cas, per a valors grans de l'edat, tenim valors petits de dedicació, mentre que per a valors petits de l'edat, tenim valors grans de dedicació. És a dir, que quan en una variable ens allunyam de la mitjana, en l'altra també ens allunyam, però en el sentit contrari. En conseqüència, aquestes dues variables tenen el que anomenam una **relació lineal inversa**.

Per confirmar numèricament el que ens estan mostrant aquestes gràfiques, necessitam tenir un indicador per saber si les dades de dues variables tenen alguna relació lineal entre elles.

Començarem introduint la **covariància**, que és l'equivalent a la variància amb un parell de llistes de dades. Mentre que la variància mesura com una variable es desvia de la seva mitjana, la covariància mesura com dues variables varien conjuntament respecte de les seves mitjanes. Vegem la seva definició matemàtica, per a dues variables (o llistes de valors) x i y :

$$COV(x,y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

On:

- N és el número de valors de les variables x i y
- x_1, x_2, \dots, x_N són tots els valors de la variable x i y_1, y_2, \dots, y_N són tots els valors de la variable y
- \bar{x} és la mitjana de la variable x i \bar{y} és la mitjana de la variable y

Molt sovint la covariància es representa mitjançant la lletra grega σ (sigma), igual que la desviació típica d'una variable:

$$\sigma_{x,y} = COV(x,y)$$

Vegem la seva implementació en Python:

```
def covariancia(x: List[float], y: List[float]) -> float:
    assert len(x) == len(y)
    xm = mitjana(x)
    ym = mitjana(y)
    n = len(x)
    sumatori = 0
    for i in range(n):
        sumatori += (x[i] - xm)*(y[i] - ym)
    return sumatori / n
covariancia_edats_hores = covariancia(edats, horas)
print(covariancia_edats_hores)
```

El resultat que ens dona és 11.44715...

En particular, la covariància d'una variable amb ella mateixa és igual a la seva variància.

Un valor gran de la covariància indica que les dues variables estan molt relacionades. És a dir, quan un valor està a prop de la mitjana en una variable, també ho està en l'altra. I quan un valor s'allunya de la mitjana en una variable, també s'allunya de la mitjana en la mateixa direcció en l'altra variable. És a dir, les dues variables estan disperses d'una manera semblant.

El problema de la covariància és que és mala d'interpretar perquè les seves unitats són el producte del quadrat de les unitats de les variables d'entrada. En el nostre exemple: anys * hores setmanals. En conseqüència, és complicat saber quan una covariància és "gran" o no.

Per solucionar aquest problema, s'ha definit el **coeficient de correlació de Pearson**, que a més de la covariància fa servir les desviacions estàndard de les dues variables. El coeficient de correlació de Pearson se sol representar mitjançant la lletra grega ρ (rho) o amb una r o R . Es defineix d'aquesta manera:

$$R(x,y) = \rho_{x,y} = \frac{\sigma_{x,y}}{\sigma_x \sigma_y} = \frac{\text{Covariància}(x,y)}{\text{Desviació estàndard}(x) \cdot \text{Desviació estàndard}(y)}$$

La particularitat d'aquest coeficient és que ens dona un valor normalitzat, que no té unitats, i que sempre està entre -1 i 1.

Si aquest coeficient ens dona un valor proper a 0, ens indica que les dues variables no estan correlacionades.

En canvi, si el coeficient està a prop d'1, vol dir que sí estan correlacionades, que existeix una relació lineal entre elles. Com més a prop d'1 més forta és la correlació entre elles. Quan dona 1 tenim el que s'anomena correlació perfecta. Per exemple, una variable està perfectament correlacionada amb ella mateixa (el seu coeficient de Pearson sempre dona 1).

Per últim, si el coeficient està a prop de -1, ens indica que les dues variables estan correlacionades, però de manera inversa (anticorrelacionades), és a dir, quan un valor d'una variable s'allunya de la mitjana en un sentit, l'altra ho fa en la mateixa proporció però en sentit contrari: quan un valor creix, en l'altra variable decreix en la mateixa proporció. Al coeficient amb valor -1 li denominam anticorrelació perfecta.

Vegem la implementació en Python:

```
def pearson(x: List[float],y: List[float]) -> float:
    return(covariancia(x,y)/(sd(x)*sd(y)))
r = pearson(edats,hores)
print(r)
```

El resultat és 0,9616...

Aquest valor, molt proper a 1, ens indica que l'edat i les hores setmanals de dedicació per mòdul estan molt correlacionades. És a dir, que els alumnes de més edat li dediquen més hores i els de menys edat, menys hores. La nostra hipòtesi inicial era certa.

Vegem ara una llista de valors amb les notes mitjanes i calculem el coeficient de correlació:

```
notes = [6, 5, 8, 8, 5, 0, 6, 6, 10, 0, 5, 1, 5, 7, 8, 5, 9, 1, 6, 3, 9, 8,
        7, 6, 9, 4, 10, 1, 7, 3, 5, 9, 4, 1, 6, 8, 1, 3, 0, 0, 5, 3, 6, 0, 5, 8, 0,
        4, 4, 4, 9, 10, 4, 1, 8, 2, 5, 10, 8, 9, 4, 0, 2, 8, 3, 4, 2, 8, 0, 4, 9, 4,
        6, 8, 1, 2, 7, 8, 7, 1, 10, 7, 0, 0, 0, 9, 7, 10, 8, 8, 0, 0, 4, 0, 0, 0, 7,
        9, 4, 3, 10, 8, 4, 2, 8, 10, 8, 4, 5, 7, 0, 5, 1, 1, 4, 0, 10, 6, 8, 8, 9, 8,
        4, 4, 10, 10, 9, 6, 7, 0, 0, 7, 7, 9, 7, 5, 1, 8, 7, 0, 6, 6, 1, 1, 7, 6, 1,
        2, 0, 2, 4, 5, 8, 6, 2, 5, 5, 1, 5, 3, 0, 8, 0, 6, 7, 8, 4, 9, 0, 5, 0, 4,
        5, 3, 3, 2, 4, 3, 2, 1, 5, 2, 1, 10, 9, 7, 8, 5, 2, 2, 0, 6, 2, 4, 3]
print(pearson(edats,notes))
```

El resultat ara és -0.029297..., un valor molt proper a zero, que ens indica que no hi ha cap correlació entre les notes i les edats. És a dir, són dues variables independents, una no depèn de l'altra.

En la llibreria *numpy* també tenim una funció [***corrcoef***](#) que ens retorna la **matriu de correlacions** entre dues variables. Aquesta matriu té 2 files i 2 columnes:

- En la casella [0][0] tenim la correlació entre la primera variable i ella mateixa (que sempre dona 1):
- En la casella [0][1] tenim la correlació entre la primera variable i la segona

- En la casella [1][0] tenim la correlació entre la segona variable i la primera (que donarà el mateix resultat que el de la casella [0][1])
- En la casella [1][1] tenim la correlació entre la segona variable i ella mateixa (que sempre dona 1)

Vegem el codi:

```
print(np.corrcoef(edats,hores))
```

i el resultat:

```
[[1.          0.96164994]
 [0.96164994 1.          ]]
```

Podem veure que el coeficient de Pearson obtingut és 0.96164994.

De fet, *numpy* també té la funció [cov](#) per obtenir la matriu de covariàncies, amb la mateixa estructura que l'anterior:

```
print(np.cov(edats,hores))
```

Ens imprimeix la matriu:

```
[[115.95654108 11.50585819]
 [ 11.50585819 1.23455024]]
```

Com podem veure, la covariància també és simètrica, no importa l'ordre: la covariància per a *edats, hores* és la mateixa que per a *hores, edats*. I recordem que la covariància d'una variable amb ella mateixa ens dona la seva variància.

6. La paradoxa de Simpson i els factors de confusió

De vegades ens podem trobar amb sorpreses analitzant les dades. En un estudi que comparava dos tractaments per als càlculs renals fet als anys 1980s es va trobar que el tractament A tenia un percentatge d'èxit del 78%, mentre que el tractament B el tenia del 83%. L'estudi es va fer amb el mateix número de pacients (350) per als dos tractaments. Semblaria lògic recomanar la utilització del tractament B, ja que té un major percentatge d'èxit.

	Tractament A	Tractament B
Càlculs renals	78% (273/350)	83% (289/350)

Taula: Percentatges d'èxit en dos tractaments per als càlculs renals

En canvi, aprofundint en l'estudi, resulta que paradoxalment el tractament A és més efectiu tant per als càlculs petits com per als càlculs grans.

	Tractament A	Tractament B
Càlculs petits	93% (81/87)	87% (234/270)
Càlculs grans	73% (192/263)	69% (55/80)
Global	78% (273/350)	83% (289/350)

Taula: Percentatges d'èxit en dos tractaments per als càlculs renals, segons la seva mida

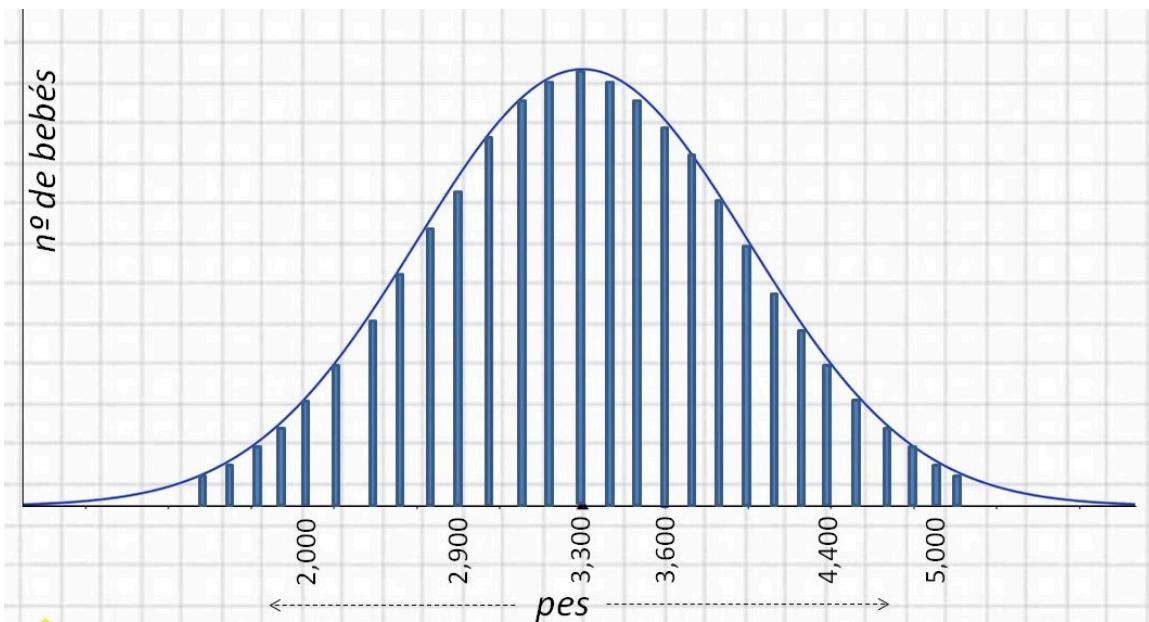
Aquesta és l'anomenada **paradoxa de Simpson**: tot i que globalment el tractament B és més efectiu, el tractament A és més efectiu tant per als càlculs grans com per als petits. Això és degut a que hi ha el que s'anomena una **variable amagada o factor de confusió**, en aquest cas, la mida del càlcul, que fins aleshores no s'havia tengut en compte, i que altera completament l'anàlisi de les dades.

Perquè passa això en aquest cas? Per dues raons. La primera és que, com és lògic, els percentatges d'èxit en els casos més lleus (càlculs petits) són més alts que per als casos més greus (càlculs grans). I la segona és que els metges no han aplicat els dos tractaments d'una manera uniforme. Han decidit, segurament a partir d'estudis previs o per un tema de cost dels tractaments, utilitzar principalment el tractament A per als casos més greus i deixar el tractament B per als casos més lleus. Així doncs, aquests dos grups (tractament A per als càlculs grans i tractament B per als petits) tenen molt més pes en el resultat global que els altres dos. Tenint en compte això i el que hem comentat abans, que els càlculs petits (tractats majoritàriament amb el tractament B) tenen major probabilitat d'èxit, acaba resultant que globalment el tractament B té un percentatge d'èxit superior.

Això és un exemple molt conegut (en podeu trobar molts més a la literatura científica) que mostra que hem de tenir molta cura a l'hora de treure conclusions a partir d'indicadors estadístics, i que sempre hem d'intentar descartar l'existència de possibles variables amagades o factos de confusió.

7. Distribució normal d'una variable aleatòria

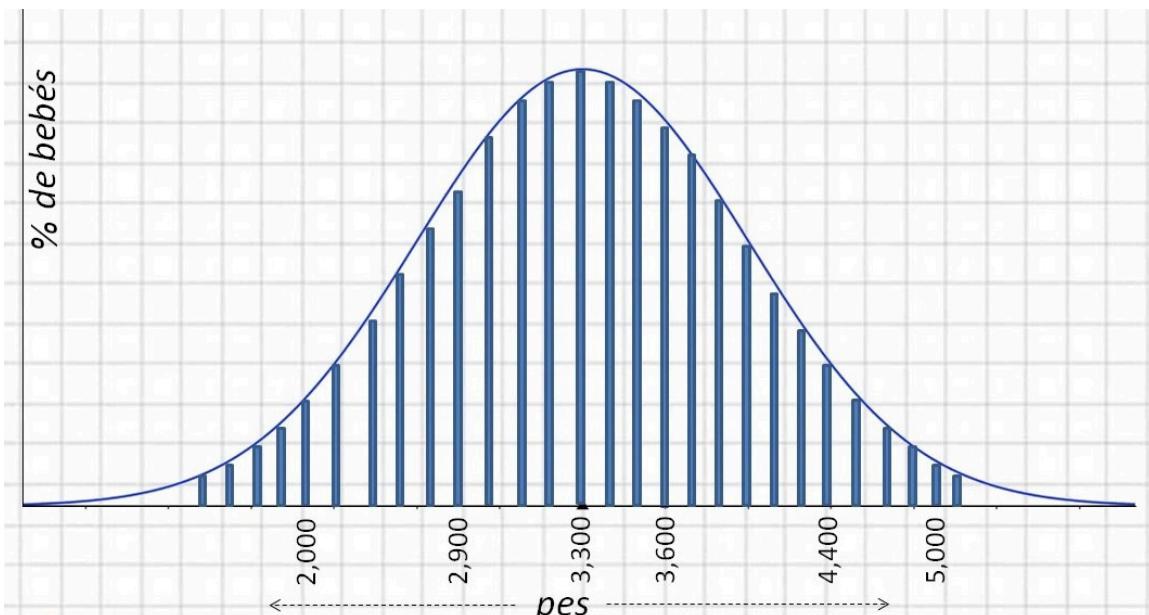
Si estudiam l'histograma del pes dels bebés quan neixen, ens trobarem que tenim casos extrems, tant de bebés que pesen molt poc (menys de 2 Kg) com d'altres que pesen molt (fins i tot de més de 5 Kg). Però la majoria dels bebés estan al voltant de la mitjana, que són uns 3,3 Kg. Ho podem veure en la imatge següent:



Imatge: Histograma i funció de freqüència del pes dels bebés. Font: Canal de YouTube Píldoras Matemáticas

Si unim els punts més alts de cada barra de l'histograma (o el que és el mateix, la freqüència de cada possible valor, és a dir, el número de vegades que apareix el valor en les nostres dades), obtenim el que s'anomena la seva funció de freqüències. Podem observar que aquesta funció té la típica forma d'una campana, a la que denominam **campana de Gauss**.

Anem a modificar una mica la gràfica anterior, perquè ara en lloc de posar en l'eix Y el número de bebés, hi posarem el percentatge dels bebés. Els percentatges els representam amb un número entre 0 i 1 (per exemple, 0,5 és el 50%, 0,75 el 75%). Així doncs, si per exemple hi ha un 10% dels bebés que pesen 3 kg, tendrem que el valor per al 3 serà 0,1.

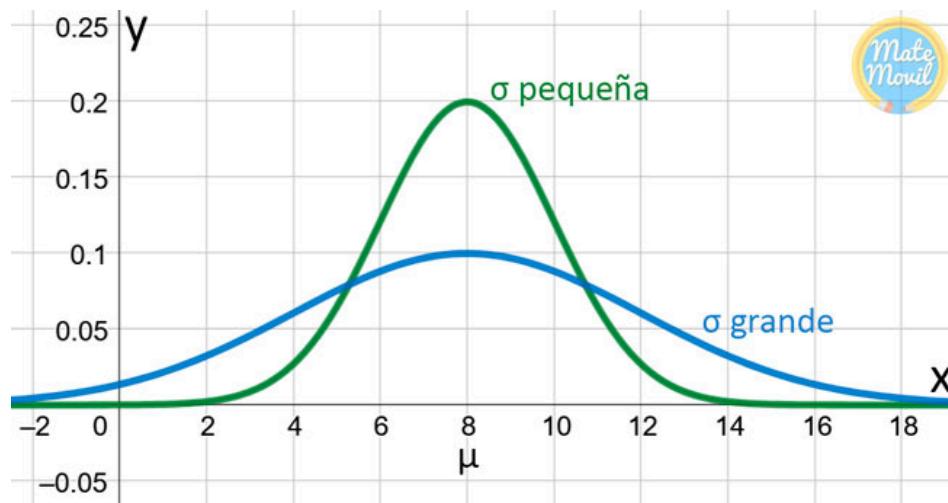


Imatge: Funció de densitat del pes dels bebés. Font: Canal de YouTube Píldoras Matemáticas (modificat)

La línia que ens apareix ara rep el nom de **funció de densitat** (o de densitat de probabilitats) i continua tenint la mateixa forma d'una campana de Gauss. Es diu, llavors, que aquesta variable (el pes dels bebés) segueix una **distribució normal** (també anomenada **distribució gaussiana**). Aquesta distribució té algunes característiques importants que veurem a continuació i ens ajuda a modelar una variable estadística i a predir les probabilitats dels seus valors.

Són moltes les variables estadístiques que segueixen aquesta distribució normal, és a dir, que la seva funció de densitat té la forma d'una campana de Gauss. Alguns exemples són l'altura de la població d'un país, les temperatures d'un lloc, el temps d'espera en una aturada de bus, o el renou en la transmissió d'un senyal, entre molts altres fenomèns naturals, socials o fins i tot psicològics.

La forma d'aquesta campana de Gauss ve determinada per dos paràmetres: la mitjana (μ) i la desviació estàndard (σ). La mitjana marca el centre (i punt més alt) de la campana, la qual deixarà la meitat de valors a un costat i l'altra meitat a l'altre (té una forma simètrica). Per tant, en una distribució normal, la mitjana coincideix amb la mediana. La desviació estàndard ens determina l'amplada de la campana: com més petita la desviació estàndard, més prima és la campana, concentrant més valors a prop de la mitjana. Per contra, com més gran la desviació estàndard, més ampla (més aplanada) serà la campana.



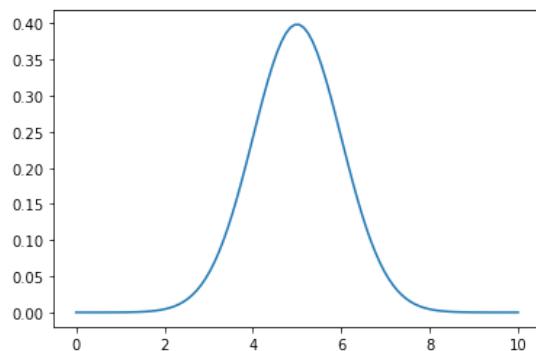
Imatge: Influència de la desviació estàndard en la forma de la campana de Gauss. Font: matemovil.com

Molt sovint es denota una distribució normal de mitjana μ i desviació estàndard σ com a $N(\mu, \sigma)$. Per exemple, una distribució normal amb mitjana 5 i desviació estàndard 1 es denota com a $N(5, 1)$.

La llibreria **SciPy**, que ja hem mencionat anteriorment, en el seu paquet d'estadístiques (stats), conté un objecte **`scipy.stats.norm`** per representar una variable aleatòria normal, com les que estam descriuint en aquest apartat. Té un mètode **`pdf`** (de *probability density function*) per obtenir la funció de densitat per a una distribució normal. El següent codi genera i dibuixa una distribució normal entre els valors 0 i 10 de l'eix X, amb mitjana 5 i desviació estàndard 1, és a dir $N(5, 1)$:

```
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
x = np.linspace(0, 10, 100)
mitjana=5
desviacio_estandard=1
plt.plot(x, stats.norm.pdf(x, mitjana, desviacio_estandard))
plt.show()
```

I el resultat és aquest gràfic:

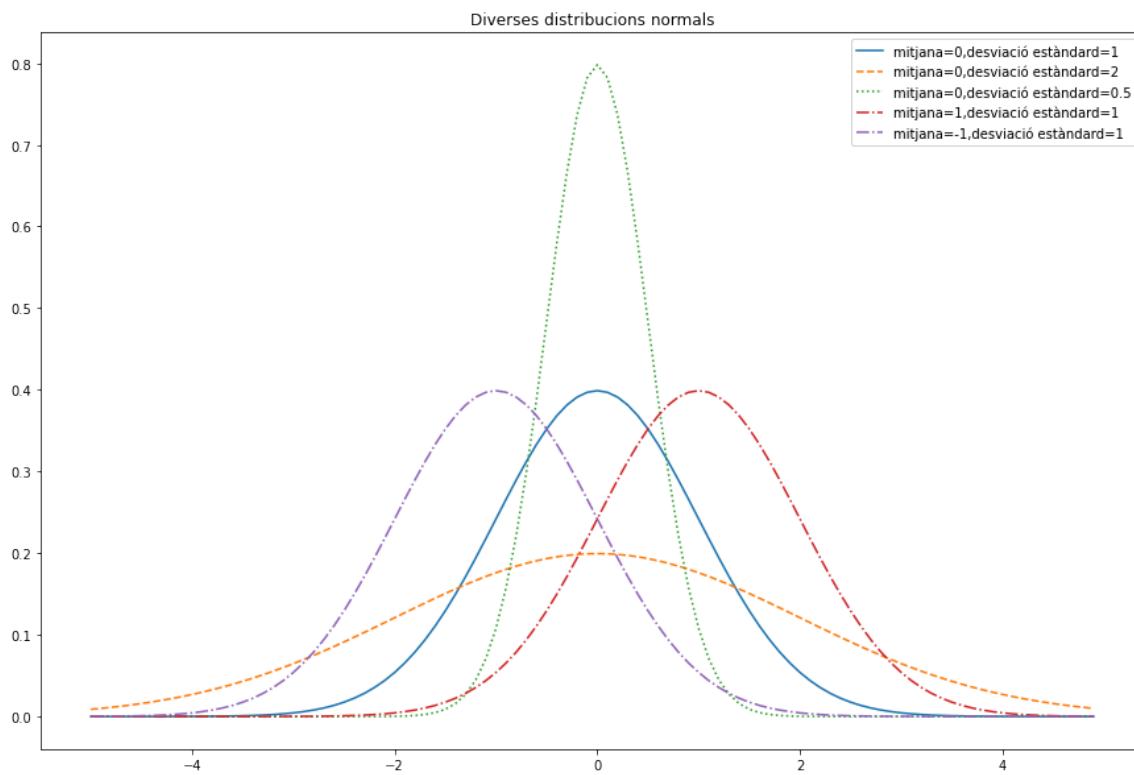


Imatge: Distribució normal amb mitjana 5 i desviació estàndard 1

Vegem un exemple que dibuixa diverses distribucions normals, amb diferents mitjanes i desviacions estàndard i així podrem veure com canvia la forma de la funció segons aquests dos paràmetres:

```
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
fig = plt.gcf()
fig.set_size_inches(15, 10)
x = np.linspace(-5, 5, 100)
plt.plot(x, stats.norm.pdf(x, 0, 1), '--', label='mitjana=0,desviació estàndard=1')
plt.plot(x, stats.norm.pdf(x, 0, 2), '--', label='mitjana=0,desviació estàndard=2')
plt.plot(x, stats.norm.pdf(x, 0, 0.5), ':', label='mitjana=0,desviació estàndard=0.5')
plt.plot(x, stats.norm.pdf(x, 1, 1), '-.', label='mitjana=1,desviació estàndard=1')
plt.plot(x, stats.norm.pdf(x, -1, 1), '-.', label='mitjana=-1,desviació estàndard=1')
plt.legend()
plt.title("Diverses distribucions normals")
plt.show()
```

Aquest és el resultat:



Imatge: Alguns exemples de distribucions normals

Si ens fixam en la forma de la cada una de les funcions, podem veure que comença creixent, primer essent una corba còncava, que després passa a ser convexa. Una vegada arriba al màxim comença a decreixir, primer seguint la corba convexa, que després acaba essent còncava. Els dos punts d'inflexió d'una distribució normal (on passa de còncava a convexa i on passa de convexa a còncava), són sempre els mateixos, independentment de com d'ampla sigui la campana, i venen determinats per la mitjana i la desviació estàndard. El primer punt d'inflexió és la mitjana menys la desviació estàndard ($\mu - \sigma$), mentre que el segon és la mitjana més la desviació estàndard ($\mu + \sigma$).

Una altra de les característiques importants d'una distribució normal és que l'àrea que queda per sota la funció sempre suma 1, independentment de la mitjana i la desviació estàndard. Aquest fet ens serveix per calcular les probabilitats per a un rang de valors. Si som capaços de calcular l'àrea de la funció entre dos valors donats, podrem saber quina probabilitat tenim de que un valor concret es trobi dins d'aquest rang. Si per exemple, entre dos valors tenim que l'àrea suma 0.7, sabrem que tenim un 70% de probabilitat que un valor que triem a l'atzar estigui dins d'aquest rang (i per tant, un 30% de que no hi estigui).

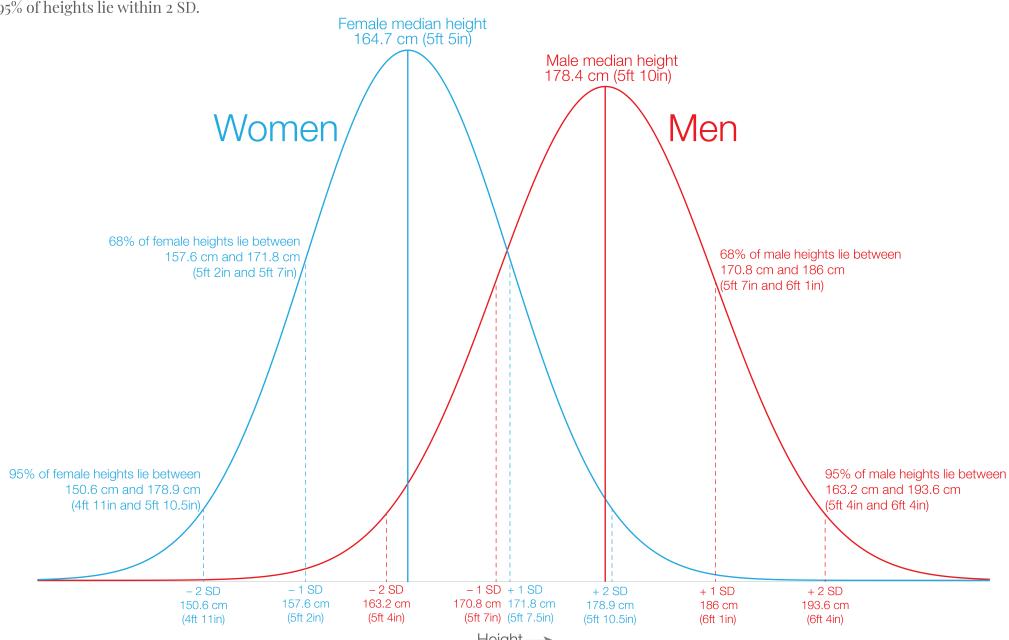
La següent imatge mostra un exemple real, l'altura d'una gran mostra d'homes i dones de 20 països d'Amèrica del Nord, Europa, Àsia oriental i Australia, nascuts entre 1980 i 1994:

The distribution of male and female heights

Our World
in Data

The distribution of adult heights for men and women based on large cohort studies across 20 countries in North America, Europe, East Asia and Australia. Shown is the sample-weighted distribution across all cohorts born between 1980 and 1994 (so reaching the age of 18 between 2008 and 2012).

Since human heights within a population typically form a normal distribution:
- 68% of heights lie within 1 standard deviation (SD) of the median height;
- 95% of heights lie within 2 SD.



Note: this distribution of heights is not globally representative since it does not include all world regions due to data availability.

Data source: Jelenkovic et al. (2016). Genetic and environmental influences on height from infancy to early adulthood: An individual-based pooled analysis of 45 twin cohorts.

This is a visualization from OurWorldInData.org, where you find data and research on how the world is changing.

Licensed under CC-BY by the author Cameron Appel.

Imatge: Distribució de les altures d'homes i de dones. Font: ourworldindata.org

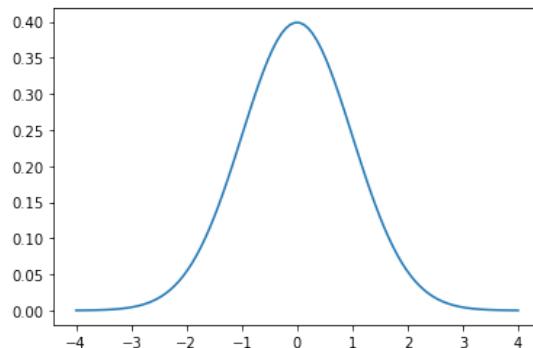
Mirem, per exemple, la funció relativa als homes. La mitjana és 178,4 cm i la desviació estàndard és 7,6 cm. Podem veure que aproximadament el 68% dels homes estan en el tram de la campana entre la mitjana menys la desviació estàndard (170,8 cm) i la mitjana més la desviació estàndard (186 cm). Si ens fixam en el tram entre la mitjana menys dues vegades la desviació estàndard (163,2 cm) i la mitjana més dues vegades la desviació estàndard (193,6 cm), hi trobam aproximadament el 95% dels homes. És a dir, només un 2,5% dels homes d'aquesta mostra fan més de 193,6 cm.

Aquestes probabilitats que hem comentat en funció de la mitjana i la desviació estàndard són sempre les mateixes, per a qualsevol distribució normal:

- La probabilitat de que un valor estigui entre $\mu - \sigma$ i $\mu + \sigma$ sempre és 68,26%
- La probabilitat de que un valor estigui entre $\mu - 2\sigma$ i $\mu + 2\sigma$ sempre és 95,4%
- La probabilitat de que un valor estigui entre $\mu - 3\sigma$ i $\mu + 3\sigma$ sempre és 99,7%

8. La taula normal estàndard

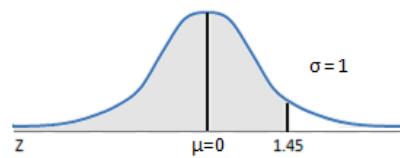
Es denomina distribució normal estàndard a la distribució normal que té mitjana 0 i desviació estàndard 1. Es denota amb $N(0,1)$.



imatge: La funció de distribució normal estàndard

Aquesta distribució és molt coneguda i podem trobar les taules de probabilitat per a qualsevol dels seus valors (entre -4 i 4).

La taula que apareix a continuació ens dona la probabilitat acumulada des de l'origen (menys infinit) fins a z . És a dir, la probabilitat de que una observació sigui menor que z . Així, per a $z=1,45$, es tractaria de calcular l'àrea de la zona grisa de la següent gràfica:



imatge: Significat de la taula normal estàndard per a probabilitats acumulades des de l'origen

Cercam la fila corresponent a 1,4. Una vegada allà, com que cercam el valor de 1,45 (no de 1,4) ens hem d'anar a la columna corresponent al 0,05. Podem veure que el valor de la cel·la és 0.92647. És a dir, podem afirmar que amb un 92,647% de probabilitats, una mesura serà inferior a 1,45.

z	+0.00	+0.01	+0.02	+0.03	+0.04	+0.05	+0.06	+0.07	+0.08	+0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56360	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891

1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87286	0.87493	0.87698	0.87900	0.88100	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91308	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.92220	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.94520	0.94630	0.94738	0.94845	0.94950	0.95053	0.95154	0.95254	0.95352	0.95449
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.96080	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.97320	0.97381	0.97441	0.97500	0.97558	0.97615	0.97670
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.98030	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.98300	0.98341	0.98382	0.98422	0.98461	0.98500	0.98537	0.98574
2.2	0.98610	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.98840	0.98870	0.98899
2.3	0.98928	0.98956	0.98983	0.99010	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.99180	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.99430	0.99446	0.99461	0.99477	0.99492	0.99506	0.99520
2.6	0.99534	0.99547	0.99560	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.99720	0.99728	0.99736
2.8	0.99744	0.99752	0.99760	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99896	0.99900
3.1	0.99903	0.99906	0.99910	0.99913	0.99916	0.99918	0.99921	0.99924	0.99926	0.99929
3.2	0.99931	0.99934	0.99936	0.99938	0.99940	0.99942	0.99944	0.99946	0.99948	0.99950
3.3	0.99952	0.99953	0.99955	0.99957	0.99958	0.99960	0.99961	0.99962	0.99964	0.99965
3.4	0.99966	0.99968	0.99969	0.99970	0.99971	0.99972	0.99973	0.99974	0.99975	0.99976
3.5	0.99977	0.99978	0.99978	0.99979	0.99980	0.99981	0.99981	0.99982	0.99983	0.99983
3.6	0.99984	0.99985	0.99985	0.99986	0.99986	0.99987	0.99987	0.99988	0.99988	0.99989
3.7	0.99989	0.99990	0.99990	0.99990	0.99991	0.99991	0.99992	0.99992	0.99992	0.99992
3.8	0.99993	0.99993	0.99993	0.99994	0.99994	0.99994	0.99994	0.99995	0.99995	0.99995
3.9	0.99995	0.99995	0.99996	0.99996	0.99996	0.99996	0.99996	0.99996	0.99997	0.99997
4.0	0.99997	0.99997	0.99997	0.99997	0.99997	0.99997	0.99998	0.99998	0.99998	0.99998

Taula: La taula normal estàndard per a probabilitats acumulades des de l'origen

La utilitat d'aquesta taula és que, quan tenim una altra mitjana i desviació estàndard (com en els exemples de l'apartat anterior), amb una senzilla conversió podem utilitzar també aquesta taula. Així doncs, si tenim una variable aleatòria x amb una distribució normal amb mitjana μ i desviació estàndard σ , la z que necessitam per a emprar la taula normal estàndard s'obté aplicant aquesta fórmula:

$$z = \frac{x - \mu}{\sigma}$$

Per exemple, en la gràfica que hem vist a l'apartat anterior de l'altura dels homes, volem calcular quina és la probabilitat de que un home tengui una altura menor a 185 cm (recordem que la mitjana és 178,4 cm i la desviació estàndard és 7,6 cm). Per tant, anam a mirar què val la z i després cercarem el valor a la taula:

$$z = \frac{185 - 178,4}{7,6} = 0,868421052631578 \approx 0,87$$

El valor de la taula per al 0,87 és 0,80785. Per tant, la probabilitat de que un home sigui més baix que 1,85 m és del 80,785%.

Ja hem vist abans que el paquet stats de SciPy proporciona un objecte `norm` per representar una variable aleatòria normal. Aquest objecte té un mètode, `cdf` (*cumulative distribution function*, funció de distribució acumulativa), que ens retorna els valors d'aquesta taula. Vegem com calcular la probabilitat per $z=1,45$:

```
import scipy.stats as stats
stats.norm.cdf(1.45)
```

Que retorna:

```
0.9264707403903516
```

A més, podem fer la conversió automàtica si treballam amb una altra mitjana i desviació estàndard. Si tornam a l'exemple de les altures dels homes, per calcular la probabilitat de que un individu sigui més baix que 1,85 m:

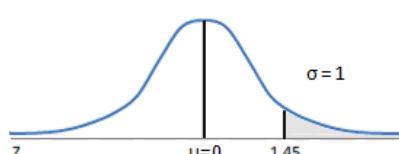
```
stats.norm.cdf(185, 178.4, 7.6)
```

Que ens retorna un 80,74%:

```
0.8074180630347813
```

Per últim, una altra taula ens dona el complementari d'aquesta darrera. Així, en lloc de donar-nos la probabilitat de que l'observació sigui menor que z , ens dona la probabilitat de que sigui major que z . És a dir, ens donarà la probabilitat de l'interval entre z i infinit. Fixau-vos que en qualsevol casella de la taula, el valor que tenim és 1 menys el valor que hi ha a l'altra taula, perquè la suma de la probabilitat que una observació sigui menor que un valor z (taula anterior) i la probabilitat que sigui major que z (aquesta nova taula) sempre serà 1 (100%).

Així, per a $z=1,45$, hauríem de calcular l'àrea de la zona en gris de la següent gràfica:



Imatge: Significat de la taula normal estàndard per a probabilitats acumulades complementàries

z	+0.00	+0.01	+0.02	+0.03	+0.04	+0.05	+0.06	+0.07	+0.08	+0.09
0.0	0.50000	0.49601	0.49202	0.48803	0.48405	0.48006	0.47608	0.47210	0.46812	0.46414
0.1	0.46017	0.45620	0.45224	0.44828	0.44433	0.44038	0.43640	0.43251	0.42858	0.42465
0.2	0.42074	0.41683	0.41294	0.40905	0.40517	0.40129	0.39743	0.39358	0.38974	0.38591
0.3	0.38209	0.37828	0.37448	0.37070	0.36693	0.36317	0.35942	0.35569	0.35197	0.34827
0.4	0.34458	0.34090	0.33724	0.33360	0.32997	0.32636	0.32276	0.31918	0.31561	0.31207
0.5	0.30854	0.30503	0.30153	0.29806	0.29460	0.29116	0.28774	0.28434	0.28096	0.27760
0.6	0.27425	0.27093	0.26763	0.26435	0.26109	0.25785	0.25463	0.25143	0.24825	0.24510
0.7	0.24196	0.23885	0.23576	0.23270	0.22965	0.22663	0.22363	0.22065	0.21770	0.21476
0.8	0.21186	0.20897	0.20611	0.20327	0.20045	0.19766	0.19489	0.19215	0.18943	0.18673
0.9	0.18406	0.18141	0.17879	0.17619	0.17361	0.17106	0.16853	0.16602	0.16354	0.16109
1.0	0.15866	0.15625	0.15386	0.15151	0.14917	0.14686	0.14457	0.14231	0.14007	0.13786
1.1	0.13567	0.13350	0.13136	0.12924	0.12714	0.12507	0.12302	0.12100	0.11900	0.11702
1.2	0.11507	0.11314	0.11123	0.10935	0.10749	0.10565	0.10383	0.10204	0.10027	0.09853
1.3	0.09680	0.09510	0.09342	0.09176	0.09012	0.08851	0.08692	0.08534	0.08379	0.08226
1.4	0.08076	0.07927	0.07780	0.07636	0.07493	0.07353	0.07215	0.07078	0.06944	0.06811
1.5	0.06681	0.06552	0.06426	0.06301	0.06178	0.06057	0.05938	0.05821	0.05705	0.05592
1.6	0.05480	0.05370	0.05262	0.05155	0.05050	0.04947	0.04846	0.04746	0.04648	0.04551
1.7	0.04457	0.04363	0.04272	0.04182	0.04093	0.04006	0.03920	0.03836	0.03754	0.03673
1.8	0.03593	0.03515	0.03438	0.03362	0.03288	0.03216	0.03144	0.03074	0.03005	0.02938
1.9	0.02872	0.02807	0.02743	0.02680	0.02619	0.02559	0.02500	0.02442	0.02385	0.02330
2.0	0.02275	0.02222	0.02169	0.02118	0.02068	0.02018	0.01970	0.01923	0.01876	0.01831
2.1	0.01786	0.01743	0.01700	0.01659	0.01618	0.01578	0.01539	0.01500	0.01463	0.01426
2.2	0.01390	0.01355	0.01321	0.01287	0.01255	0.01222	0.01191	0.01160	0.01130	0.01101
2.3	0.01072	0.01044	0.01017	0.00990	0.00964	0.00939	0.00914	0.00889	0.00866	0.00842
2.4	0.00820	0.00798	0.00776	0.00755	0.00734	0.00714	0.00695	0.00676	0.00657	0.00639
2.5	0.00621	0.00604	0.00587	0.00570	0.00554	0.00539	0.00523	0.00508	0.00494	0.00480
2.6	0.00466	0.00453	0.00440	0.00427	0.00415	0.00402	0.00391	0.00379	0.00368	0.00357
2.7	0.00347	0.00336	0.00326	0.00317	0.00307	0.00298	0.00289	0.00280	0.00272	0.00264
2.8	0.00256	0.00248	0.00240	0.00233	0.00226	0.00219	0.00212	0.00205	0.00199	0.00193
2.9	0.00187	0.00181	0.00175	0.00169	0.00164	0.00159	0.00154	0.00149	0.00144	0.00139

3.0	0.00135	0.00131	0.00126	0.00122	0.00118	0.00114	0.00111	0.00107	0.00104	0.00100
3.1	0.00097	0.00094	0.00090	0.00087	0.00084	0.00082	0.00079	0.00076	0.00074	0.00071
3.2	0.00069	0.00066	0.00064	0.00062	0.00060	0.00058	0.00056	0.00054	0.00052	0.00050
3.3	0.00048	0.00047	0.00045	0.00043	0.00042	0.00040	0.00039	0.00038	0.00036	0.00035
3.4	0.00034	0.00032	0.00031	0.00030	0.00029	0.00028	0.00027	0.00026	0.00025	0.00024
3.5	0.00023	0.00022	0.00022	0.00021	0.00020	0.00019	0.00019	0.00018	0.00017	0.00017
3.6	0.00016	0.00015	0.00015	0.00014	0.00014	0.00013	0.00013	0.00012	0.00012	0.00011
3.7	0.00011	0.00010	0.00010	0.00010	0.00009	0.00009	0.00008	0.00008	0.00008	0.00008
3.8	0.00007	0.00007	0.00007	0.00006	0.00006	0.00006	0.00006	0.00005	0.00005	0.00005
3.9	0.00005	0.00005	0.00004	0.00004	0.00004	0.00004	0.00004	0.00004	0.00003	0.00003
4.0	0.00003	0.00003	0.00003	0.00003	0.00003	0.00003	0.00002	0.00002	0.00002	0.00002

Taula: La taula normal estàndard per a probabilitats acumulades complementàries

Tornem a cercar el valor per a 1,45. Anam a la fila 1,4 i columna 0,05 i trobam el resultat: 0,07353. Tal i com hem comentat abans, 0,92647+0,07353 és igual a 1.

Aquesta darrera taula ens serà útil per calcular els intervals de confiança que introduirem en el següent apartat.

L'objecte *norm* del paquet *stats* de SciPy també té un mètode, **sf** (*survival function*, funció de supervivència) que ens permet calcular aquesta probabilitat acumulada complementària. Vegem-ho de nou per a z=1,45:

```
stats.norm.sf(1.45)
```

I per obtenir la probabilitat de que un home sigui més alt que 1,85 m:

```
stats.norm.sf(185, 178.4, 7.6)
```

Que retorna un 19,258%:

```
0.19258193696521875
```

9. Mostres i intervals de confiança

Suposem que volem saber l'alçada mitjana de la població adulta masculina espanyola. Per tenir el valor exacte, hauríem de conèixer l'alçada dels al voltant de 20 milions d'homes adults del país. Això, evidentment, és impossible. En canvi, el que hauríem de fer és prendre una **mostra**, és a dir, agafar un conjunt d'individus, que intenten representar de la millor manera al conjunt de la població.

El problema és que si agaf diferents mostres, el més probable és que em surtin totes amb una mitjana diferent. Per exemple, suposem que per a la meva mostra he agafat als jugadors d'un equip de bàsquet. És possible que em surti una mitjana de més de 2 metres. Evidentment, la mitjana real de la població serà molt més baixa, perquè la mostra que he pres no és gens representativa. És un exemple extrem, però la realitat és que molt difícilment la mitjana de dues mostres coincidiran, sempre hi haurà diferències, encara que siguin petites.

Aleshores, com puc saber si la mostra que jo he triat és prou representativa de la població real? Aquest és un problema molt complex. Aquí explicarem breument, sense entrar en els fonaments matemàtics, el que passa quan la variable segueix una distribució normal, com a priori seria el nostre exemple de l'alçada de la població espanyola.

El que volem és donar un interval de dos valors entre els quals, amb una determinada probabilitat (normalment del 90%, del 95% o del 99%) es trobarà la mitjana real de la població. És a dir, donada una mostra, no puc saber quina és la mitjana real de tota la població, però puc assumir que amb, per exemple, un 95% de probabilitat, la mitjana es trobarà entre dos valors (en un interval). Aquest interval s'anomena **interval de confiança** i a la probabilitat que triam (per exemple el 95%) li denominam **nivell de confiança**. Anam a veure ara com obtenir aquest interval de confiança a partir d'una mostra.

Donada una mostra, podrem obtenir les següents variables amb el que hem vist en apartats anteriors:

- n és la mida de la mostra (número de punts o valors)
- m és mitjana de les dades de la mostra
- d és la desviació estàndard de les dades de la mostra

Agafam un nivell de nivell de confiança que sigui 0,9 (90%), 0,95 (95%) o 0,99 (99%). A partir de la següent taula, obtenim un valor al qual anomenarem z , per a diferents nivells de confiança:

Nivell de confiança	z
0,9	1,64
0,95	1,96
0,99	2,58



D'on surten els valors d'aquesta taula?

Li direm p al nivell de confiança. Suposem que volem un $p=0,95$. Això vol dir que un 5% (0,05) de les possibles observacions quedarien fora del rang que volem, la meitat (0,025) per dalt i l'altra meitat (0,025) per baix. Aleshores, per saber què val z , hem de cercar a la taula normal estàndard per a probabilitats acumulades complementàries quina és la primera casella amb un valor igual o inferior a 0,025. I el trobam a la casella corresponent a 1,96 (que val exactament 0,025). Per això, per a $p=0,95$, z val 1,96.

Aleshores, l'interval de confiança per a la mitjana de la població ve determinat per aquests dos límits:

$$\text{Límit inferior} = m - z \frac{d}{\sqrt{n}}$$

$$\text{Límit superior} = m + z \frac{d}{\sqrt{n}}$$



És important tenir en compte que per poder aplicar aquestes fórmules, la mostra ha de tenir un **mínim de 30 exemplars** o individus. Per a mostres més petites, no es pot assumir que les dades segueixin una distribució normal (seria una altra, anomenada *t de Student*) i per tant, s'ha de calcular d'una altra manera, que ja surt de l'objectiu d'aquest lliurament.

IMPORTANT

Vegem un exemple. Suposem que tenim la següent mostra amb l'alçada de 30 homes:

175.9, 173.1, 177.7, 178.3, 183.2, 171.1, 184.6, 169.4, 168.3, 189.1,

177.2, 175.3, 186.3, 173.6, 169.0, 173.8, 188.1, 167.0, 192.0, 175.7,

196.9, 171.9, 179.7, 176.4, 174.1, 174.2, 168.9, 172.6, 175.5, 172.4

Volem conèixer l'interval de confiança de la mitjana amb un nivell de confiança del 95%. Per tant, z serà igual a 1,96.

Vegem el codi Python per obtenir l'interval de confiança:

```
import numpy as np
import math
mostra = [175.9, 173.1, 177.7, 178.3, 183.2, 171.1, 184.6, 169.4, 168.3, 189.1,
          177.2, 175.3, 186.3, 173.6, 169.0, 173.8, 188.1, 167.0, 192.0, 175.7,
          196.9, 171.9, 179.7, 176.4, 174.1, 174.2, 168.9, 172.6, 175.5, 172.4]
n = len(mostra)
m = np.mean(mostra)
d = np.std(mostra)
z = 1.96
liminf = m - z*d/math.sqrt(n)
limsup = m + z*d/math.sqrt(n)
print(f"Interval [{liminf:.4f}, {limsup:.4f}]")
```

El resultat serà l'interval [174.4228, 179.6639]. Per tant, podem assumir, amb una probabilitat del 95%, que la mitjana de la població adulta masculina espanyola estarà dins aquest interval.

10. La distribució binomial

En els apartats anteriors hem estudiat la distribució normal o gaussiana. Però n'hi ha moltes altres. Una d'elles és la distribució binomial. Aquesta potser no té gaire interès en l'àmbit de l'anàlisi de dades, però sí en determinats problemes d'aprenentatge automàtic.

S'anomena una **variable aleatòria de Bernoulli** a aquella que només pot tenir dos possibles valors: èxit o fracàs. És, per tant, una variable discreta, a diferència d'una variable aleatòria normal, com per exemple l'altura dels homes, que és contínua, ja que pot prendre qualsevol valor dins un interval.

Un exemple típic de variable de Bernoulli és el llançament d'una moneda: només pot ser cara o creu (a una la considerarem èxit i a l'altra fracàs). La probabilitat d'èxit en un llançament és 0,5 (50%). Però què passa si feim molts llançaments? Per exemple, si llançam 100 monedes a l'aire, quina probabilitat n'hi ha que surtin exactament 50 cares? Això es modela mitjançant una variable aleatòria que té el que anomenam una **distribució binomial**.

Una distribució binomial es caracteritza per dos paràmetres n , el número d'experiments o observacions (llançaments en l'exemple de la moneda) i p , la probabilitat d'èxit (0,5 en el cas de la moneda). Ho representam com $B(n,p)$.

La funció de probabilitats d'una distribució $B(n,p)$ per calcular si hi ha k èxits es calcula de la següent manera:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

On n sobre k es calcula de la següent manera:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

I el factorial de n és:

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

Per exemple:

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

En el paquet stats de SciPy tenim l'objecte ***binom*** per representar una variable aleatòria binomial. L'objecte ***binom*** té un mètode ***pmf*** (*probability mass function*), amb tres paràmetres k , n i p , que calcula la probabilitat de tenir k èxits amb n experiments i una probabilitat d'èxit individual de p . Per exemple, si llançam 100 monedes, per saber la probabilitat de treure 50 cares seria:

```
import scipy.stats as stats
stats.binom.pmf(50, 100, 0.5)
```

Que dona com a resultat aproximadament un 7,96%:

```
0.07958923738717875
```

I si llançam un dau 100 vegades, quina probabilitat hi ha de treure 20 cincs? Aquí un èxit és treure un 5 i la probabilitat p és d'una entre 6 (1/6):

```
stats.binom.pmf(20, 100, 1/6)
```

Que dona aproximadament un 6,79%:

```
0.06786199515844837
```

També, igual que vàrem veure amb les variables aleatòries normals, podem obtenir la funció de probabilitat acumulada, mitjançant el mètode **cdf** (*cumulative distribution function*) de l'objecte *binom*. Així, podem saber, per exemple, quina probabilitat hi ha de treure fins a 50 cares si llançam una moneda 100 vegades:

```
stats.binom.cdf(50, 100, 0.5)
```

Que dona una mica més del 50%:

```
0.5397946186935895
```

I també tenim la funció de probabilitat acumulada complementària (també anomenada funció de supervivència), que seria 1 menys la funció de probabilitat acumulada. Ho tenim al mètode **sf** (*survival function*) de l'objecte *binom*. Per exemple, si volem saber la probabilitat de treure més de 50 cares en 100 llançaments:

```
stats.binom.sf(50, 100, 0.5)
```

Que dona:

```
0.46020538130641053
```

Fixa't que $0.5397946186935895 + 0.46020538130641053$ dona 1.



En aquest [article](#) pots trobar una explicació sobre la distribució binomial i la seva utilitat importància en l'aprenentatge automàtic. També pots veure més detalls sobre la distribució binomial, així com altres distribucions de probabilitat que tenen aplicació en l'aprenentatge automàtic a aquest altre [article](#).

En l'[activitat d'ampliació](#) veurem què és una **distribució multinomial**, una generalització de la distribució binomial, quan en cada experiment tenim més de dos possibles resultats. Mentre que la distribució binomial ens pot servir per caracteritzar un problema de classificació binària, una distribució multinomial ens serveix per a un problema de classificació multiclasse.

11. Més informació

Tal i com ja hem comentat anteriorment, amb l'objectiu de fer més senzilla la seva comprensió, s'han fet algunes simplificacions de la teoria matemàtica que hi ha per sota dels conceptes que hem introduït. Aquells que vulgueu tenir un coneixement més profund, podeu acudir a una extensa bibliografia sobre el tema.

Aquest tema està basat en part en la segona edició del llibre **Data Science from Scratch: First Principles with Python**, de Joel Grus, publicat per l'editorial O'Reilly en 2019.

Hi ha molts d'altres llibres d'introducció a l'estadística. Alguns d'ells els podeu trobar en obert a la web:

- [Introductory Statistics](#). Douglas Shafer i Zhiyi Zhang. Saylor Foundation.
- [OnlineStatBook](#). David Lane. Rice University
- [Introductory Statistics](#). OpenStax. OpenStax College

També podreu trobar multitut de recursos de cursos universitaris d'introducció a l'estadística en castellà o català.

Un exemple n'és aquest curs de bioestadística de la Universidad de

Málaga: <https://www.bioestadistica.uma.es/baron/apuntes/>