

# Apunts CE\_5074 5.1

Iloc: [Institut d'Ensenyaments a Distància de les Illes Balears](#)  
Curs: Sistemes de Big Data  
Llibre: Apunts CE\_5074 5.1

Imprès per: Carlos Sanchez Recio  
Data: dilluns, 27 de gener 2025, 21:43

## Taula de continguts

### 1. Introducció

### 2. Tècniques d'anàlisi basades en la visualització

- 2.1. Visualitzar quantitats
- 2.2. Visualitzar proporcions
- 2.3. Visualitzar relacions
- 2.4. Visualitzar distribucions
- 2.5. Visualitzar dades textuals
- 2.6. Visualitzar dades espacials

### 3. Biblioteques de visualització en Python

- 3.1. Matplotlib
- 3.2. Seaborn
- 3.3. Plotly
- 3.4. Datashader

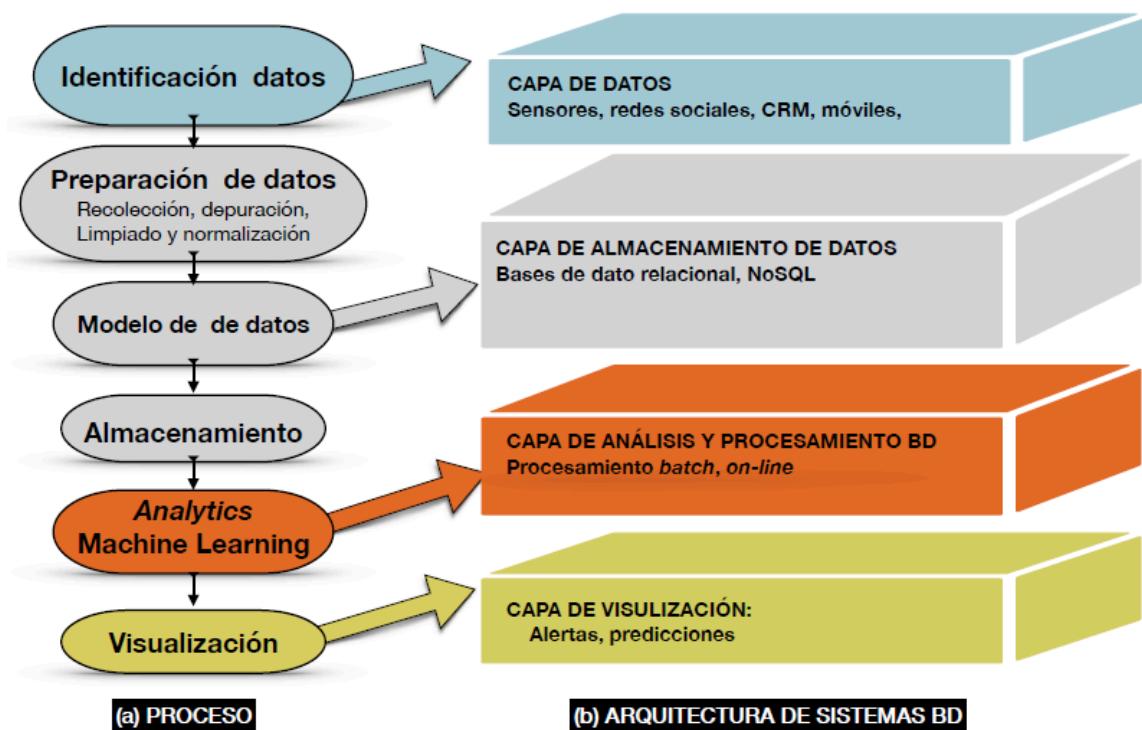
### 4. Bibliografia

## 1. Introducció

En el primer lliurament d'aquest mòdul vàrem identificar les principals etapes del procés d'extreure informació rellevant a partir de dades massives:

- Identificació de les dades
- Processament de les dades, amb tres subfases: preparació o pre-processament, modelat i emmagatzematge
- Anàlisi
- Visualització

Així mateix, també vàrem veure que la majoria de sistemes de big data s'organitzen a partir d'aquestes etapes, donant lloc a una estructura de diverses capes:



**Imatge:** Estructura multicapa dels sistemes de Big Data. Font: Universidad de Castilla-La Mancha

En els lliuraments anteriors, llevat del capítol dedicat als fonaments estadístics (que relament són la base de la capa d'anàlisi), ens hem centrat en la capa d'emmagatzematge de dades, on hem vist diversos models de dades: el model basat en documents, el model basat en grafs i el model multidimensional. En aquest lliurament ens centrarem en la darrera de les capes (i etapes), la de visualització. En el lliurament 6, xerrarem del que entenem per *business intelligence* i els quadres de comandament, que integren les capes d'anàlisi i visualització. Per últim, els dos darrers lliuraments els dedicarem íntegrament a la capa d'anàlisi.

### Què entenem per visualització de dades?

Les dades contenen informació concreta sobre fets, elements, etc., que permet estudiar-los, analitzar-los o coneixer-los. La visualització de dades és la presentació d'aquestes dades en format gràfic, mitjançant l'ús d'elements visuals com a taules, gràfics o mapes. El camp de la visualització de dades barreja tant l'art com la ciència de dades. Una bona visualització de dades ha de ser creativa i agradable de mirar, però també ha de ser funcional per a aconseguir la comunicació visual i efectiva de les dades. Les eines de visualització de dades proporcionen una forma accessible de veure i comprendre les tendències, els valors atípics i els patrons de les dades.

Les dades, especialment quan xerram de grans volums, poden resultar difícils d'entendre. En el context del Big Data, les eines i tecnologies de visualització de dades es converteixen en eines essencials perquè els analistes puguin estudiar grans quantitats d'informació i prendre decisions basades en aquesta. La visualització de dades, per tant, té diversos objectius. Primer, ens permet **explicar i interpretar les dades** dins d'un determinat context; també ens permet **resoldre determinats problemes** a través d'aquestes visualitzacions; ens permet **explorar les dades, esdeveniments o successos** que d'una altra forma podrien passar desapercebuts; finalment, també ens permet **realitzar prediccions** de manera senzilla.

## 2. Tècniques d'anàlisi basades en la visualització

Les tècniques d'anàlisis basades en la visualització impliquen la representació gràfica de les dades per a permetre o millorar la seva percepció visual. La vista és un dels sentits més especialitzats dels éssers humans, la qual cosa ens permet extreure patrons, entendre i treure conclusions a partir de gràfics d'una manera molt més ràpida que des d'altres formats, com per exemple el text. Per això, l'anàlisi visual s'utilitza com una eina dins del camp del Big Data, ja que és essencial per a analitzar quantitats massives d'informació i prendre decisions sobre la base de les dades.

Aquestes tècniques se centraran, per tant, a obtenir una compressió més profunda de les dades que estam analitzant, ajudant-nos a identificar patrons, correlacions i anomalies de manera senzilla. També solen emparar-se aquest tipus de tècniques dins de l'anàlisi exploratòria de les dades, la qual cosa permet formular preguntes des d'angles diferents. Finalment, les visualitzacions ens permetran comunicar les dades i les operacions aplicades sobre aquestes de manera efectiva a altres persones.

Les tècniques de visualització de dades en Big Data no es distingeixen necessàriament de les visualitzacions tradicionals en entorns amb dades petites. Per exemple, en el cas de visualitzacions que ens permetin resumir categories, com en els gràfics de barres o circulars, s'apliquen les mateixes visualitzacions tant a nivell de Big Data com de dades petites. No obstant això, algunes tècniques tradicionals de visualització aplicades per a petites dades han de ser lleugerament alterades per a incorporar-les amb èxit en el Big Data, per exemple a l'hora de treballar amb dades que provenen de fonts no estructurades. També ha de tenir-se en compte la grandària del conjunt de dades a l'hora d'aplicar les visualitzacions, de manera que no se saturi el canal visual dels receptors (els éssers humans no som bons visualitzant gran quantitat d'informació simultàniament) o la memòria de vídeo del dispositiu utilitzat per a realitzar aquesta visualització.

En aquest apartat descriurem una de les principals tècniques d'anàlisis de dades visuals, els **gràfics**.

Els gràfics permeten als usuaris/clients/parts interessades (*stakeholders*) endinsar-se en la història que volem contar. Unes certes imatges, com el pendent d'un gràfic de línies o l'agrupació de punts en un gràfic de dispersió, poden transmetre la informació de forma més eficaç que el text o les taules.

La decisió sobre el tipus de gràfic a utilitzar es basa en dos factors principals: el format de les dades i el tipus d'història que es vol contar. A continuació presentarem alguns dels gràfics més utilitzats habitualment per a visualitzar i analitzar les dades.



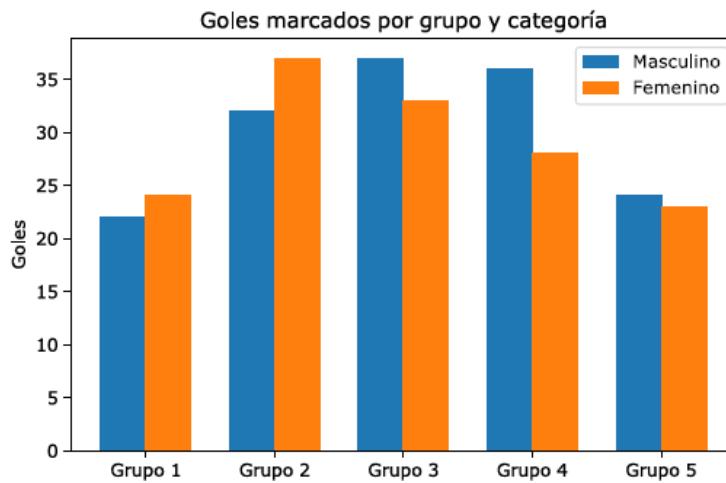
La recopilació de diferents tipus de gràfics que es mostren a continuació no pretén ser un llistat exhaustiu, sinó simplement una enumeració d'algunes de les tècniques de visualització més utilitzades.

## 2.1. Visualitzar quantitats

En alguns escenaris, estam interessats a representar la magnitud d'un conjunt de variables. Per exemple, visualitzar el volum total de vendes de la nostra companyia o el salari de cadascun dels empleats. En aquests casos, tenim una variable categòrica (per exemple els articles que s'han venut o el nom de cadascun dels empleats) i una altra variable quantitatativa associada amb cada categoria. Vegem les principals visualitzacions d'aquest tipus.

### ■ Gràfic de columnes i de barres

Els **gràfics de columnes** s'utilitzen per a visualitzar les diferències numèriques entre diferents categories i poder comparar una al costat de les altres.

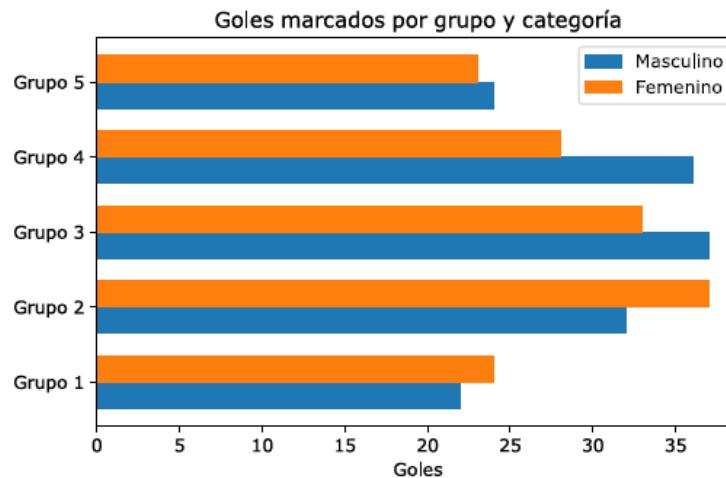


**Imatge:** Gràfic de columnes. Font: Universidad de Castilla-La Mancha

Quan les etiquetes de les categories són llargues se sol intercanviar l'eix x i l'eix y, de manera que les barres s'orienten horitzontalment, donant lloc a un **gràfic de barres**.

ACLARIMENT

Molt sovint ens referim a gràfic de barres, tant si aquestes són horizontals com verticals (columnes).

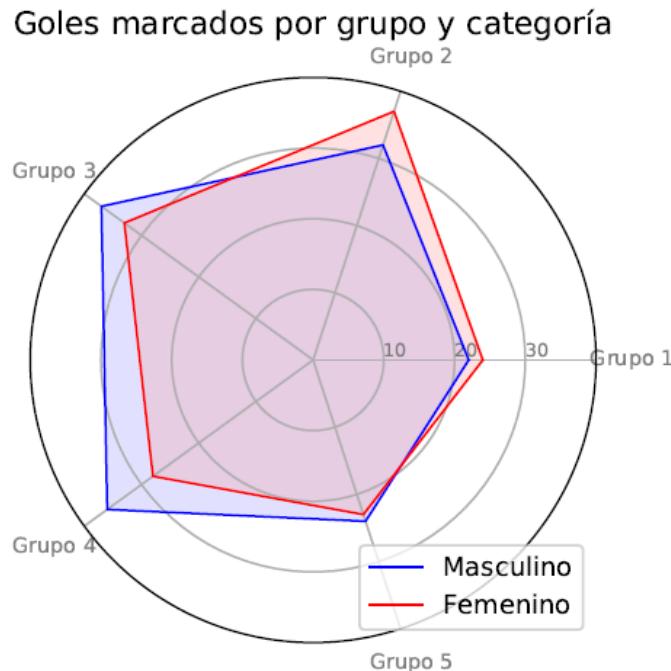


**Imatge:** Gràfic de barres. Font: Universidad de Castilla-La Mancha

Quan estam interessats a comparar dues variables categòriques a la mateixa vegada, llavors podem agrupar o apilar les barres unes damunt o al costat de les altres. Així és com s'ha fet en els gràfics anteriors, on hem agrupat per masculí (blau) i femení (taronja). En un **gràfic de columnes agrupades**, primer definim un grup de barres en cada posició al llarg de l'eix x, el qual està determinat per una variable categòrica, i després tracem barres dins de cada grup segons la segona variable categòrica. I farem el mateix, però en l'eix y, per a obtenir un **gràfic de barres agrupades**.

### ■ Gràfic de radar

Una altra opció per a visualitzar quantitats de diverses categories és utilitzar un gràfic de radar, com el de la imatge següent. Aquests gràfics permeten la visualització de dades multivariades en forma de gràfic bidimensional de tres o més variables quantitatives representades en eixos que parteixen d'un mateix punt.



**Imatge:** Gràfic de radar. Font: Universidad de Castilla-La Mancha

### ■ Mapa de calor

Una altra alternativa als gràfics de columnes o barres és assignar els valors de les dades a diferents colors. És el que es coneix com a mapes de calor. En la imatge següent hem representat les mateixes dades, emprant els tons roses per als valors més alts i blaus per als més baixos, quedant els tons violats al mig.



**Imatge:** Mapes de calor. Font: Universidad de Castilla-La Mancha

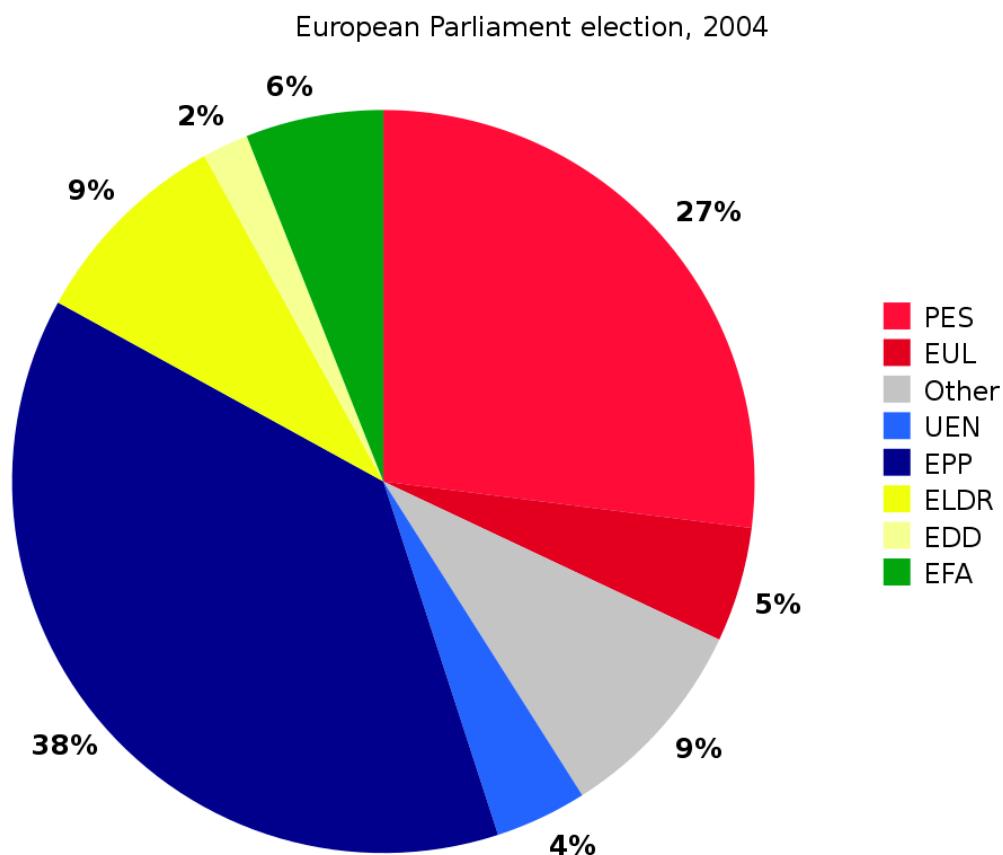


## 2.2. Visualitzar proporcions

### ■ Gràfic circular o de pastís

La manera més habitual de representar proporcions entre diverses variables quantitatives és mitjançant l'anomenat **gràfic circular o de pastís**. Normalment s'utilitza per representar percentatges de diferents categories, que soLEN ser més de 3 (però no necessàriament).

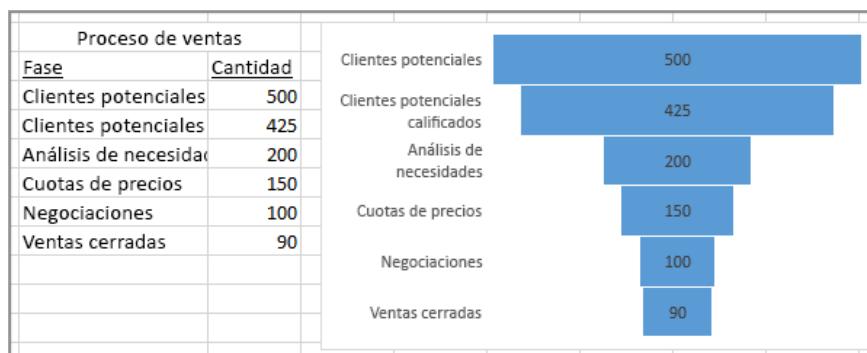
En l'exemple següent podem veure el percentatge de vots obtinguts per diversos partits a les eleccions al Parlament Europeu de 2004:



Imatge: Gràfic circular. Font: Wikipedia Commons

### ■ Gràfic d'embut

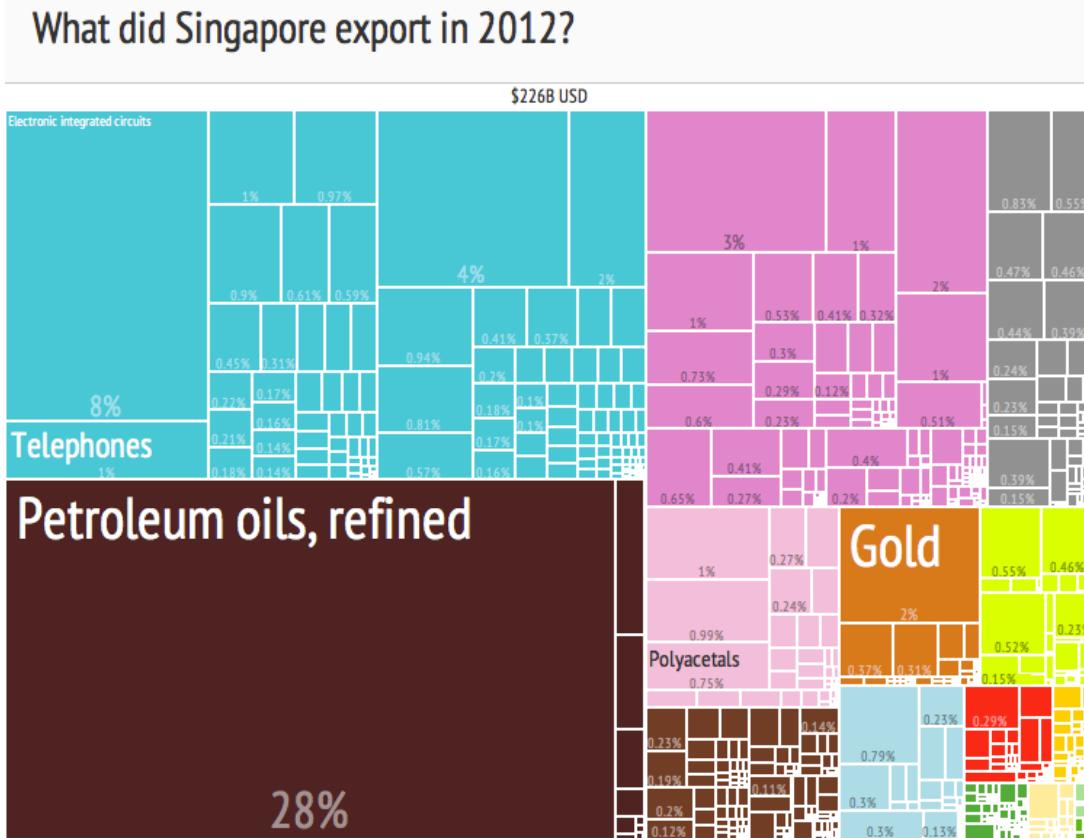
Una altra manera de representar proporcions és mitjançant l'anomenat **gràfic d'embut**. És un gràfic similar al de pastís, però les diferents categories es mostren apilades i ordenades de major a menor o viceversa, creant una forma d'embut.



Imatge: Gràfic d'embut. Font: Suport tècnic de Microsoft (<https://support.microsoft.com/>)

## ■ Treemap

Una altra representació de les proporcions és l'anomenat **treemap** (mapa d'arbre), semblant al mapa de calor, però on la mida de cada categoria és proporcional al seu valor. En l'exemple podem veure les exportacions de Singapur en l'any 2012.



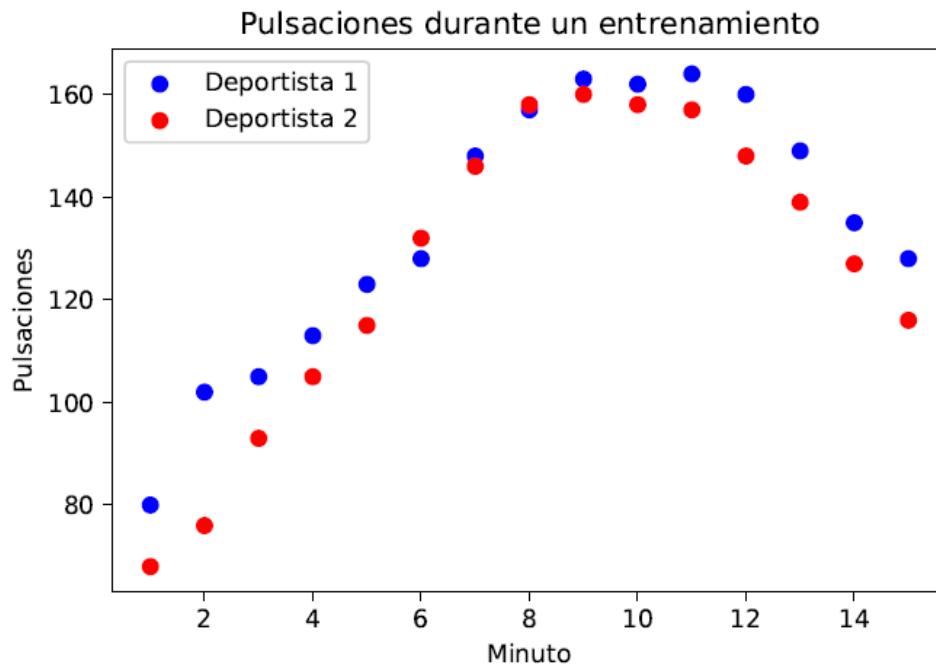
Imatge: Treemap. Font: Wikipedia Commons

## 2.3. Visualitzar relacions

Els gràfics que veurem a continuació permeten visualitzar la relació entre dues variables quantitatives.

### ■ Gràfic de dispersió

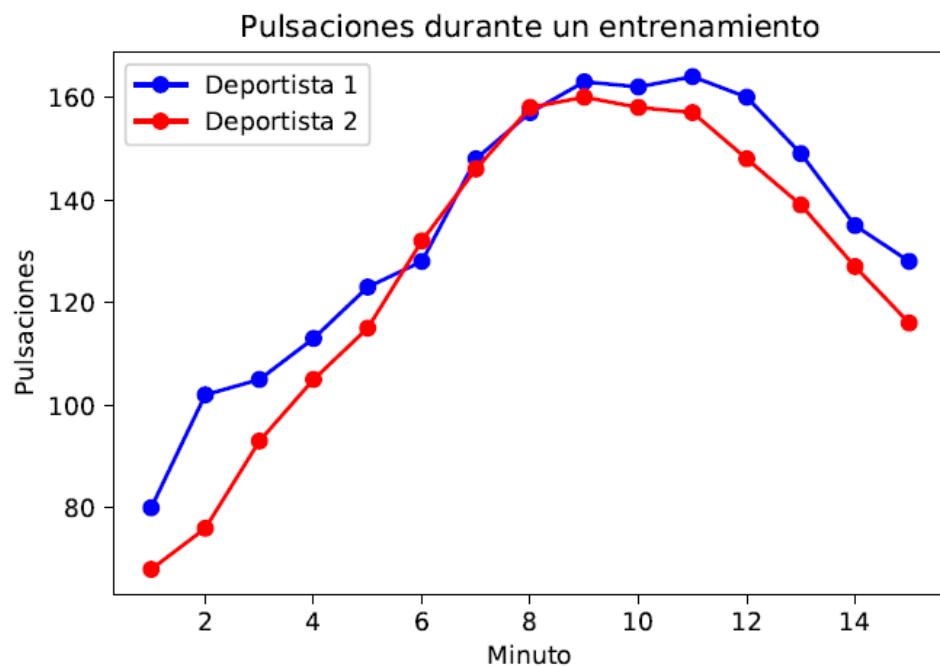
Els gràfics de dispersió són la visualització més habitual quan volem mostrar com es relacionen dues variables quantitatives. Per a això, se sol elaborar un gràfic amb dos eixos (un per cada variable) i es dibuixa un punt (o un altre element similar) per a cadascuna de les dades del nostre conjunt de dades. D'aquesta forma, els punts formen un nívol dispers (d'aquí el terme gràfic de dispersió). La imatge següent mostra un exemple de gràfic de dispersió que mostra les pulsacions de dos esportista al llarg d'una sessió d'entrenament de 15 minuts.



Imatge: Gràfic de dispersió. Font: Universidad de Castilla-La Mancha

### ■ Gràfic de línies

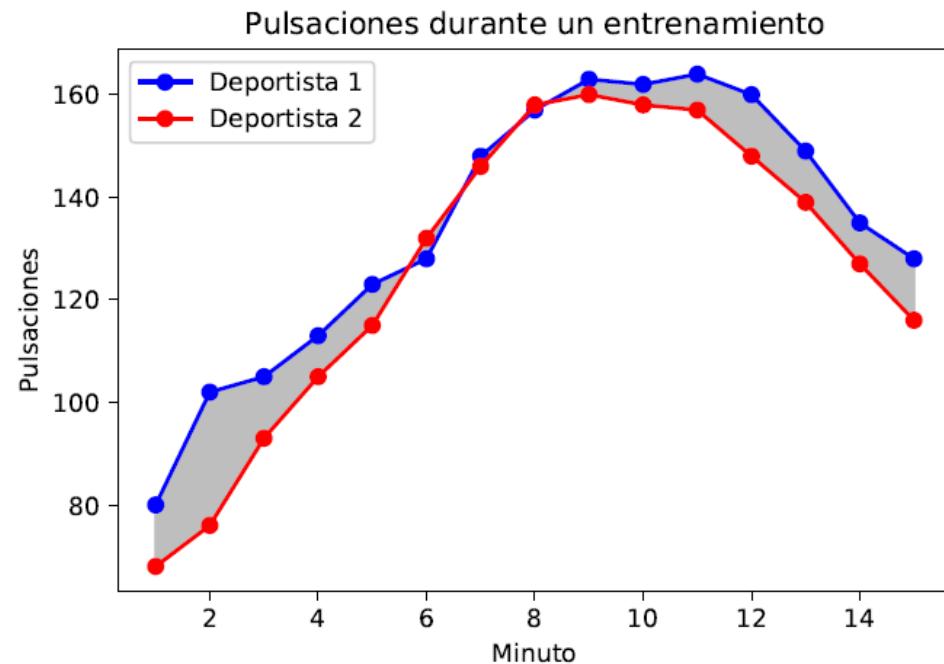
Quan l'eix horitzontal representa el temps o una quantitat estrictament creixent, és habitual unir els punts del gràfic de dispersió mitjançant línies, obtenint el que anomenam com a gràfic de línies. La imatge següent representa l'exemple anterior utilitzant ara un gràfic de línies.



Imatge: Gràfic de línies. Font: Universidad de Castilla-La Mancha

### ■ Gràfic d'àrea

Aquest tipus de gràfics és en essència un gràfic de línies en el qual s'ombreja d'un color determinat l'àrea entre la línia i l'eix o bé entre dues línies. La imatge següent mostra un gràfic d'àrea en el qual s'ha ombrerat l'àrea entre les pulsacions dels dos esportistes.



Imatge: Gràfic d'àrea. Font: Universidad de Castilla-La Mancha

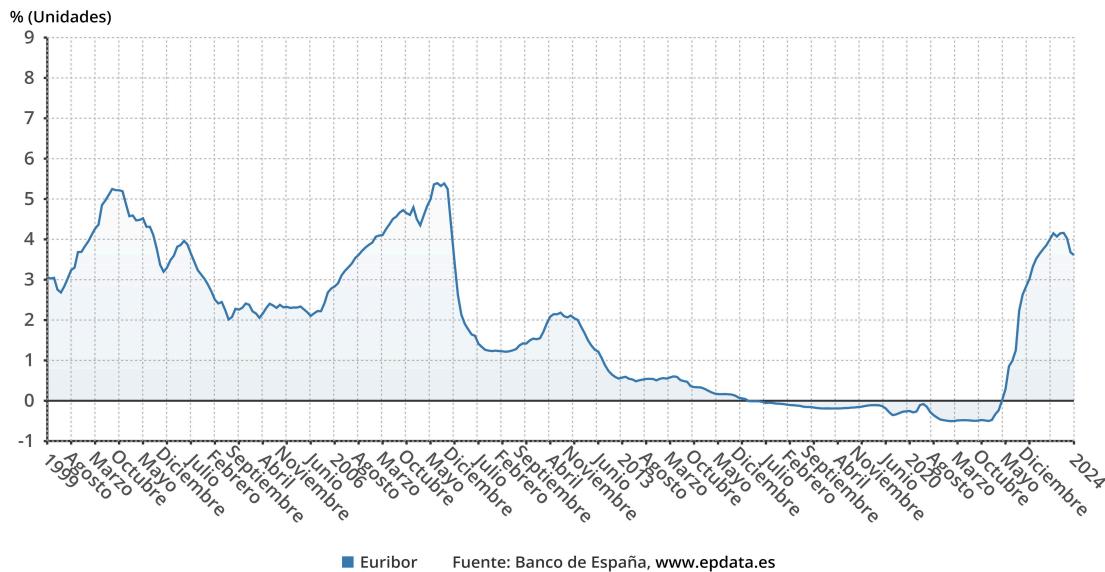
### ■ Gràfic de sèries temporals

Aquest tipus de gràfics representen les dades en intervals temporals successius. Són similars als gràfics de línies però únicament representen dades en les quals en l'eix horitzontal es tenim intervals de data i hora i en l'eix vertical es representen els valors que es volen mesurar en aquest interval. Aquest tipus de gràfics solen utilitzar per a realitzar ànalisis de sèries temporals, el qual tracta de predir valors futurs d'una sèrie temporal sobre la base

de valors passats coneixuts de la mateixa sèrie o altres sèries. En realitat, el gràfic de línies que hem vist anteriorment correspondria a una sèrie temporal. Però normalment quan xerram de sèries temporals, ens referim a intervals molt més grans, amb moltes més medicions.

Podem trobar nombrosos exemples molt presents en el nostre dia a dia: l'evolució de la pandèmia o de la vaccinació, l'evolució de l'Euríbor (en la imatge següent), l'evolució de temperatures o precipitació, o l'evolució en la cotització bursàtil o de facturació o beneficis d'una empresa.

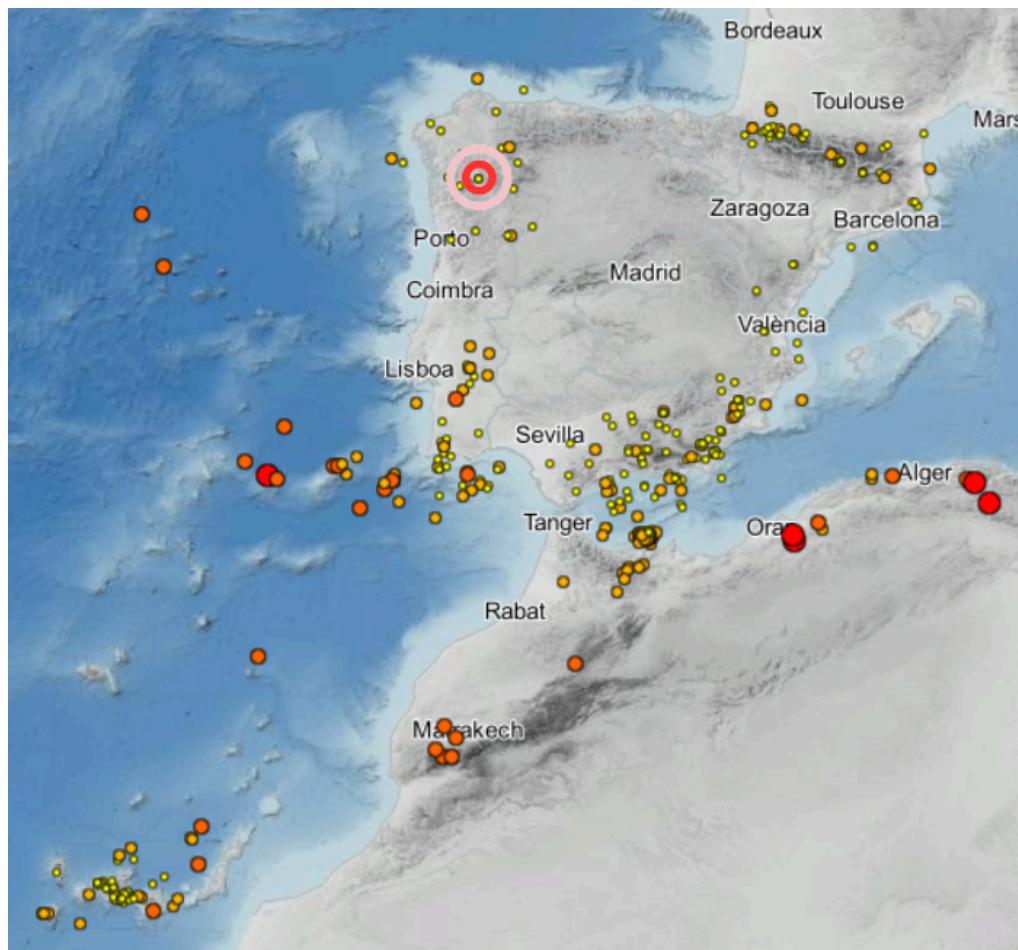
### Evolución del Euríbor mensual



**Imatge:** Evolució de l'Euribor des de 1999. Font: Europa Press Data, a partir de dades del Banc d'Espanya

### ■ Gràfic de bimboles

Aquest tipus de gràfics són una variació del gràfic de dispersió en el qual els diferents punts de dades es reemplacen per bimboles que afegeixen una dimensió addicional (la grandària de les bimboles). Per tant, els eixos verticals i horizontals representen dues variables quantitatives, però a més d'aquests valors, s'afegeix un valor que es representa mitjançant la mida de cada bimbolla. Per tant, aquest tipus de gràfics permeten representar tres dimensions de les dades. La imatge següent mostra un gràfic de bimboles amb la localització (eixos horitzontal i vertical) dels terratrèmols durant els darrers 30 dies i la seva intensitat (grandària i color de les bimboles). En lloc de la mida, també es freqüent aplicar diferents colors que marquen una gradació. El punt destacat a Galícia correspon al darrer terratrèmol enregistrat.



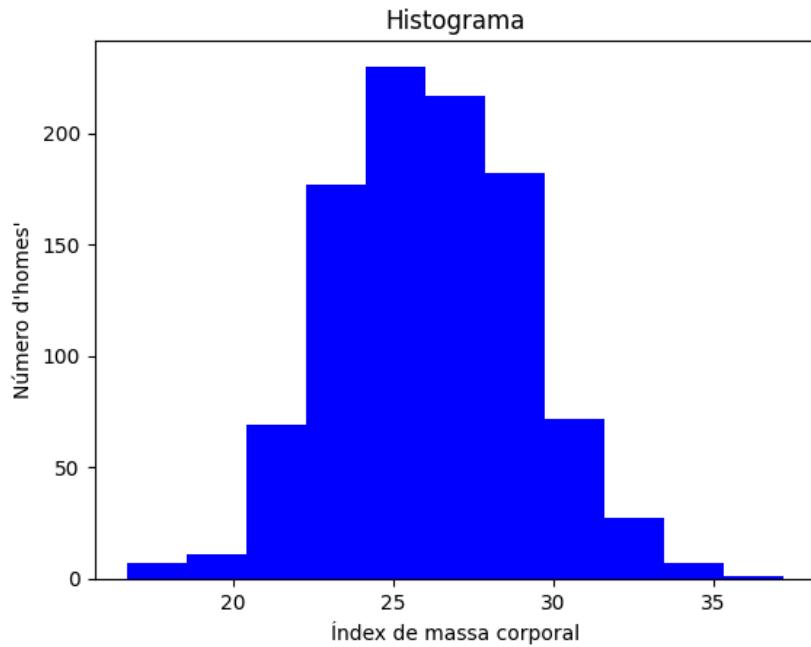
**imatge:** Terremotèmols en els darrers 30 dies. Visualitzador de terremotèmols propers de l'Institut Geogràfic Nacional

## 2.4. Visualitzar distribucions

El gràfic més emprat per a visualitzar distribucions és l'histograma, del qual ja n'hem parlat amb detall en el lliurament 2. Però n'existeixen d'altres.

### Histograma

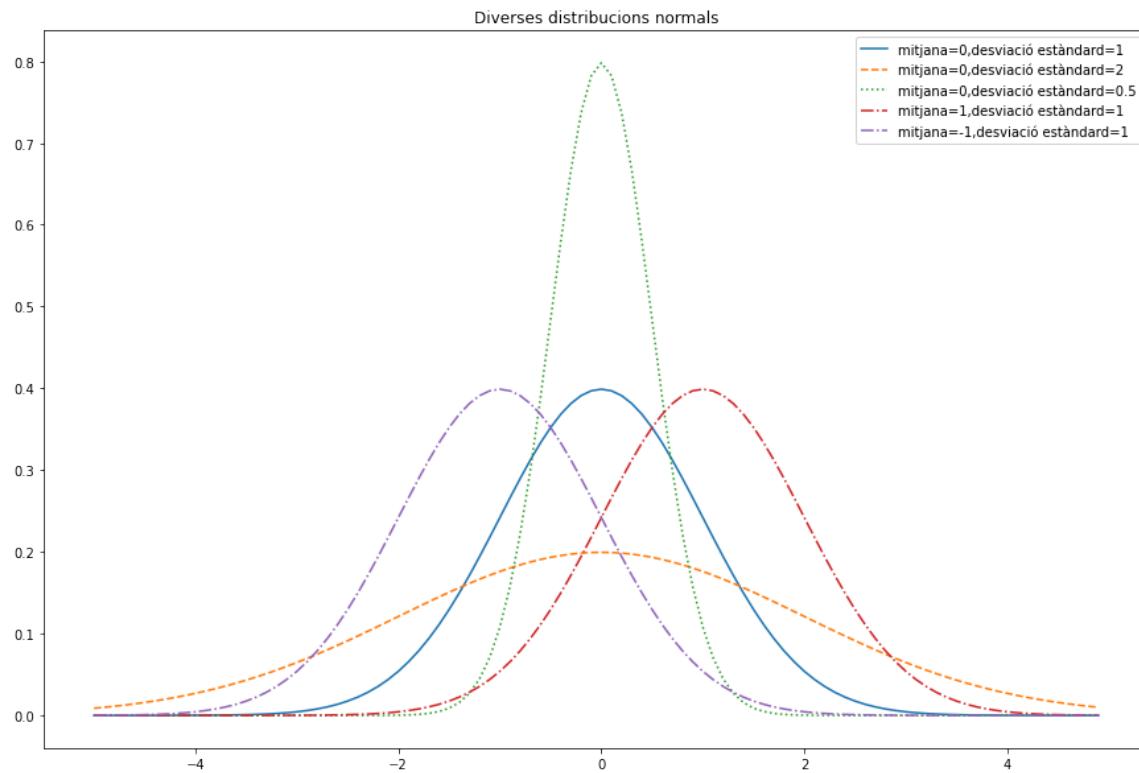
Sovint ens trobem amb la situació de voler entendre com es distribueix una determinada variable en un conjunt de dades. Per exemple, com es distribueix l'altura o pes de les persones en una ciutat, regió o país. En la imatge següent mostram l'histograma de l'índex de massa corporal d'una mostra sintètica de 1000 homes espanyols entre 25 i 34 anys d'edat, que vàrem obtenir en la tasca del lliurament 2.



**imatge:** Histograma de l'índex de massa corporal (mostra de 1000 homes espanyols d'entre 25 i 34 anys)

En un histograma, totes les franges han de tenir la mateixa amplària perquè l'histograma sigui vàlid.

De manera similar a l'histograma, existeixen els denominats **diagrames de densitat**. Aquests són com un histograma, però suavitzats. Aquest tipus de diagrames s'utilitzen per a realitzar una estimació de la funció de densitat de probabilitat de la variable subjacent. Recordem també del lliurament 2 que vàrem xerrar de la distribució de probabilitat normal. En la imatge següent podem veure les funcions de distribució normal amb diverses mitjanes i desviacions estàndard.



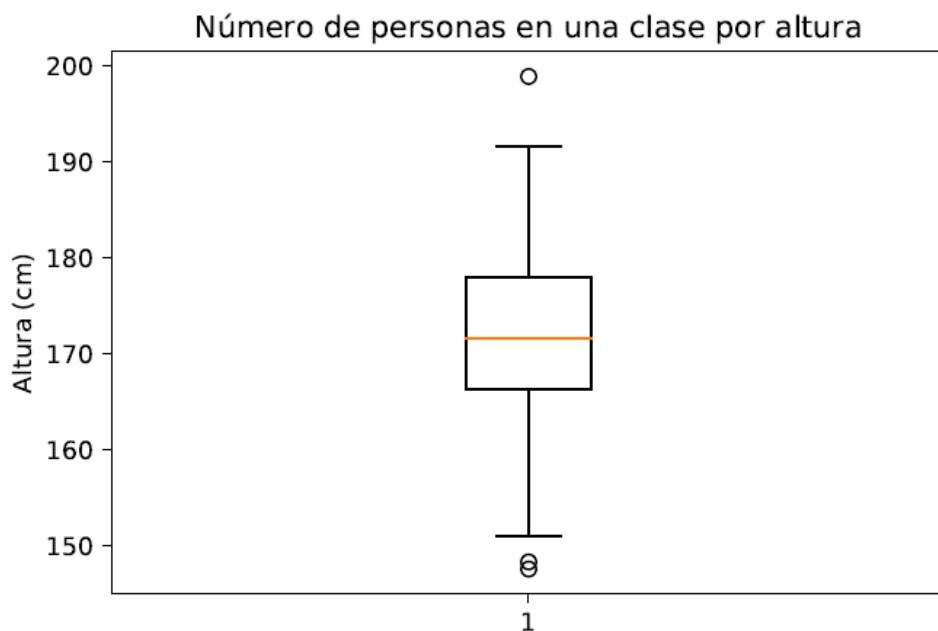
**Imatge:** Alguns exemples de distribucions normals

## ■ Diagrama de caixa

En anglès es denomina **box plot**, també se'ls anomena **diagrama de caixa i bigotis**. Aquest tipus de diagrames permeten mostrar la distribució dels valors d'una variable al llarg d'un eix. Per a això, aquest tipus de diagrames mostren diversos elements en l'eix y:

- Primer, mostren una línia horitzontal (normalment d'un altre color) que representa la mitjana de la distribució.
- També es mostra un quadre (denominat **caixa**) que indica entre què rangs es troba el 50% central de les dades, en altres paraules, els dos quartils centrals de la distribució.
- Addicionalment, es mostren unes línies verticals (denominades **bigotis**) que representen 1,5 vegades el rang intercuartil en el qual es troben totes les dades.
- Finalment, tots els punts que queden fora dels bigotis seran els outliers del conjunt de dades sota estudi i es representaran com a cercles en el diagrama.

La imatge següent mostra un diagrama de caixa per a representar la distribució de l'altura dels alumnes d'una classe.



**imatge:** Diagrama de caixa representant l'altura dels alumnes d'una classe. Font: Universidad de Castilla-La Mancha

## 2.5. Visualitzar dades textuais

El **gràfic de nigul de paraules** (o **d'etiquetes**), **word cloud** o **tag cloud** en anglès, permet visualitzar dades de text. Per a això, es mostren certes paraules o frases amb diferents grandàries i colors en funció d'un valor determinat. El més habitual és utilitzar com a variable a mesurar el nombre d'aparicions de cada cadena de text, tot i que també es pot emprar la importància en un determinat context, l'ordre alfabètic, o qualsevol altre índex que l'usuari desitgi. Per exemple, la imatge següent mostra un gràfic de nigul de paraules dels 1.000 articles més vistos de Wikipedia (a més repeticions d'una paraula, més gran apareixerà aquesta).



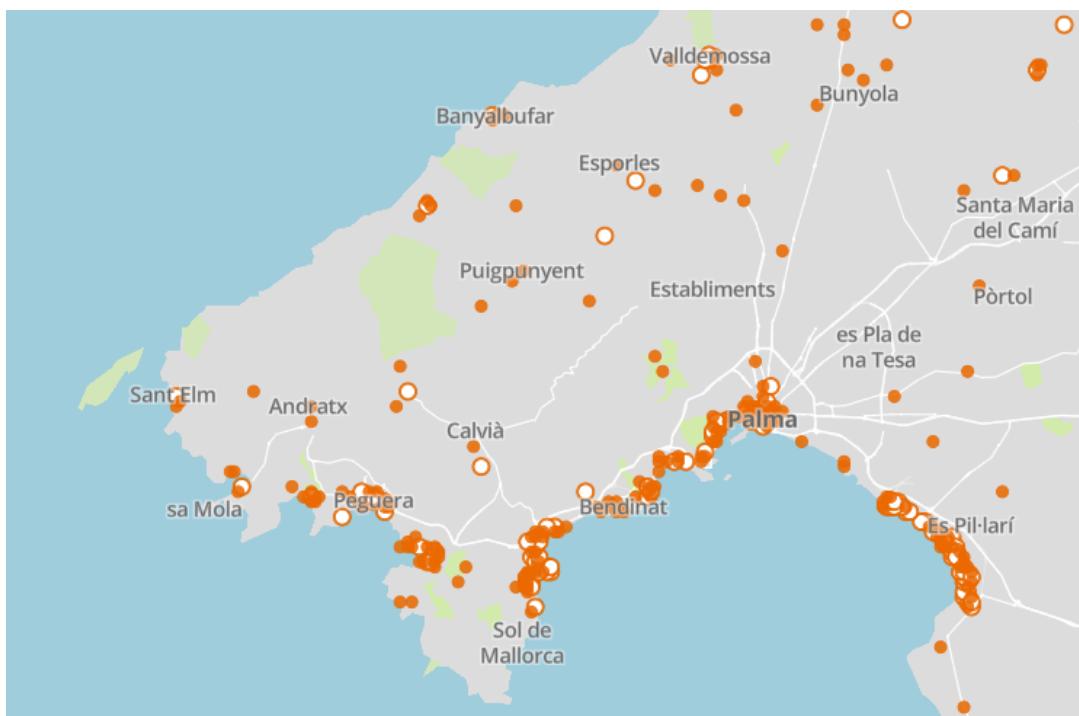
**Imatge:** Gràfic de nigul de paraules dels 1.000 articles més vistos de Wikipedia. Font: Wikipedia Commons

## 2.6. Visualitzar dades espacials

Les dades amb un component espacial es representen normalment mitjançant un mapa. Les dades es poden agregar per municipi, regió, país, etc. per a generar el que anomenam un mapa de coropletes.

### ■ Mapa

La forma principal de mostrar les dades geoespacials és en forma de mapa. Un mapa pren les coordenades del globus terraquí i les projecta sobre una superfície plana en la qual es representarà. D'aquesta manera, les formes i les distàncies del globus terraquí es representen aproximadament amb les formes i les distàncies de la representació plana en 2 dimensions. Existeixen diversos sistemes de projecció i cada un utilitza un sistema de coordenades diferent.



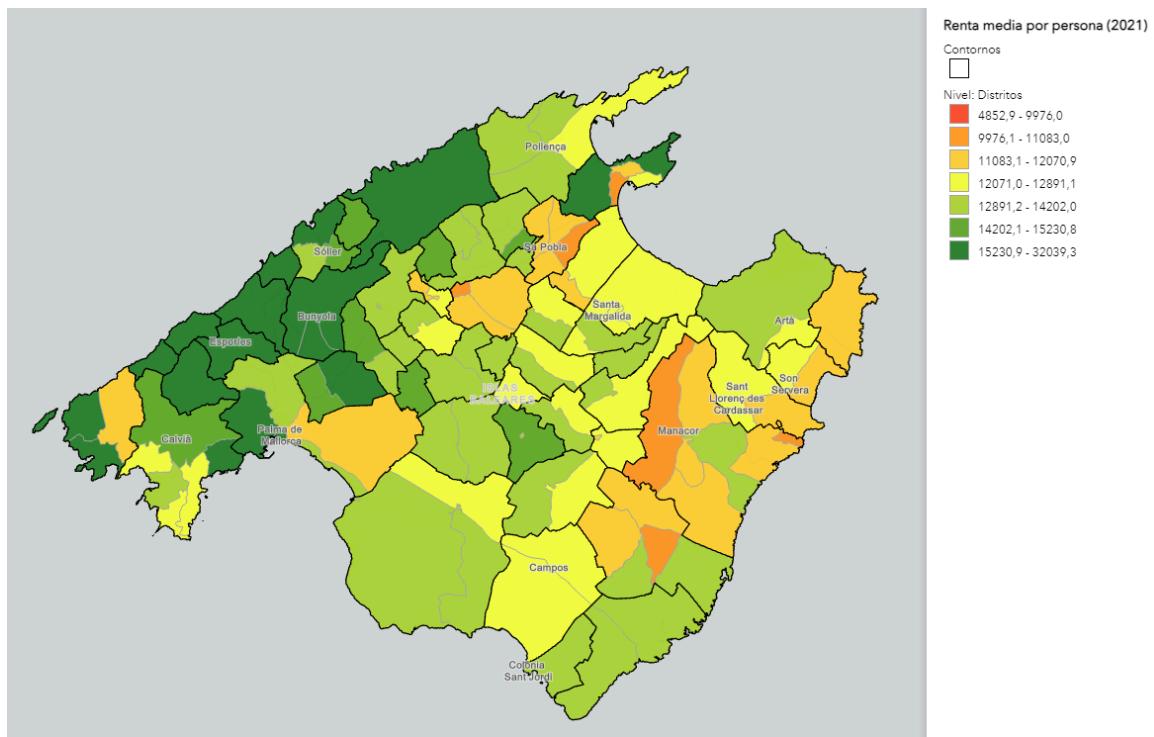
**imatge:** Mapa dels allotjaments turístics de Mallorca. Catàleg de dades obertes del Govern de les Illes Balears

### ■ Mapa de coropletes

En algunes ocasions es vol mostrar com varia alguna variable agregant els seus valors per municipi, comunitat autònoma, país, etc.

Per a això, es poden mostrar els valors de les dades en diferents regions acolorint-les en el mapa segons les seves dades agregades. Aquest tipus de mapa es denomina mapa de coropletes o mapa coroplètic.

La imatge següent mostra el mapa de la renda mitjana per persona de l'illa de Mallorca, extret de l'*Atlas de Distribución de Renta de los Hogares 2021*, del Instituto Nacional de Estadística. Podem veure que no es representa un punt per al valor de renda de cada un dels habitants de Mallorca, sinó que les dades s'agreguen, en aquest cas per districte censal, i es calcula la mitjana de cada districte. Cada un dels districtes es farceix d'un color, en funció dels intervals que veim a la llegenda.



**Imatge:** Mapa de la renda mitjana per persona. Atlas de Distribución de renta de los hogares 2021 de l'INE

### 3. Biblioteques de visualització en Python

En aquest apartat ens centrem en la visualització gràfica de dades mitjançant Python. Python proporciona diverses biblioteques per a la visualització. Matplotlib, que ja vàrem emprar en el lliurament 2, així com en altres mòduls del curs, n'és la més bàsica. Però n'hi ha d'altres, com Seaborn, que ofereix moltes opcions de gràfics avançats d'una manera senzilla, Plotly, que permet interactuar amb els gràfics, i Databricks, especialitzada en el tractament de grans volums de dades. I encara n'hi ha més que no veurem en aquest curs, com per exemple HoloViews o Bokeh.

D'altra banda, en el lliurament 6 veurem eines de visualització integrades en entorns de *Bussiness Intelligence*, sense necessitat de programar. Són els anomenats **quadres de comandament** o **dashboards**, eines que mostren de manera gràfica i interactiva els principals indicadors d'una organització. Aquestes eines s'utilitzen com a suport per a la presa de decisions.

### 3.1. Matplotlib

En el mòdul de Programació d'Inteligència Artificial ja vàrem introduir Matplotlib amb un [quadern Colab](#).

Recordem que dins Matplotlib, el mòdul *matplotlib.pyplot* (normalment important amb el prefix *p/t*) proporciona una interfície orientada a objectes per a crear i gestionar gràfics. Els gràfics sempre són dins una figura (objecte *Figure*), que es pot dividir en subplots o eixos. Per crear un gràfic bàsic, normalment seguim aquestes passes:

1. **Definir la figura:** la funció *plt.figure()* permet crear una nova figura (objecte *Figure*).
2. **Crear els eixos o subplots:** la funció *plt.subplot()* permet definir diverses zones (subplots o eixos) per a poder incloure un gràfic en cada una d'elles.
3. **Dibuixar les dades:** podem utilitzar les funcions com *plt.plot()*, *plt.hist()* o *plt.scatter()* per a dibuixar les dades als eixos o subplots.
4. **Personalitzar el gràfic:** tenim diverses funcions com *plt.title()*, *plt.xlabel()* o *plt.ylabel()* per a personalitzar el títol, les etiquetes dels eixos i altres aspectes del gràfic.
5. **Mostrar el gràfic:** finalment, cridant la funció *plt.show()* es mostrerà el gràfic a la pantalla.

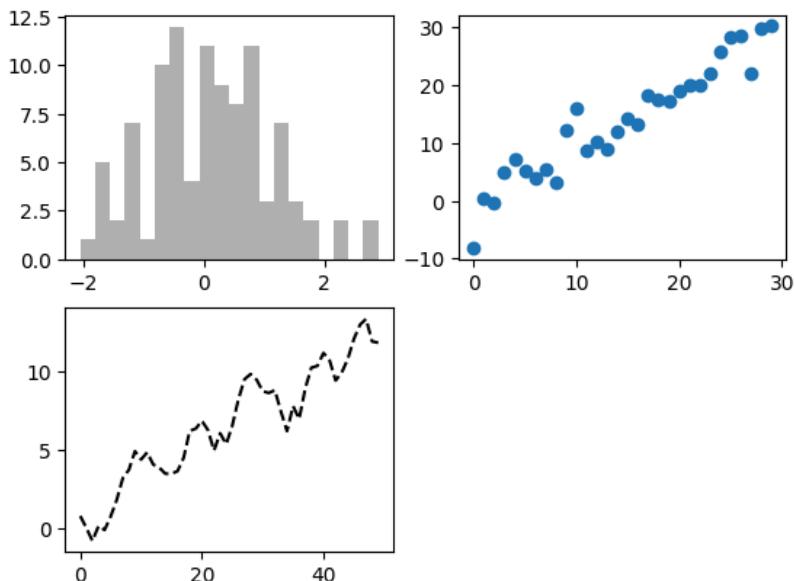
En el quadern de Colab que hem mencionat, podem veure un exemple on definim 3 subplots i hi dibuixam un histograma (amb la funció *plt.hist*), un gràfic de dispersió o nigrul de punts (amb la funció *plt.scatter*) i un gràfic de línies (amb la funció *plt.plot*):

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(2,2,2)
ax3 = fig.add_subplot(2,2,3)

ax1.hist(np.random.randn(100), bins = 20, color = 'k', alpha= 0.3)
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
ax3.plot(np.random.randn(50).cumsum(), 'k--')

plt.show()
```



Però amb Matplotlib podem fer molts més tipus de gràfics. Vegem els probablement dos més habituals: el diagrama de barres, per al qual emprarem les funcions *plt.bar* (barres verticals) o *plt.barh* (barres horizontals), i el diagrama circular o de pastís, mitjançant la funció *plt.pie*.



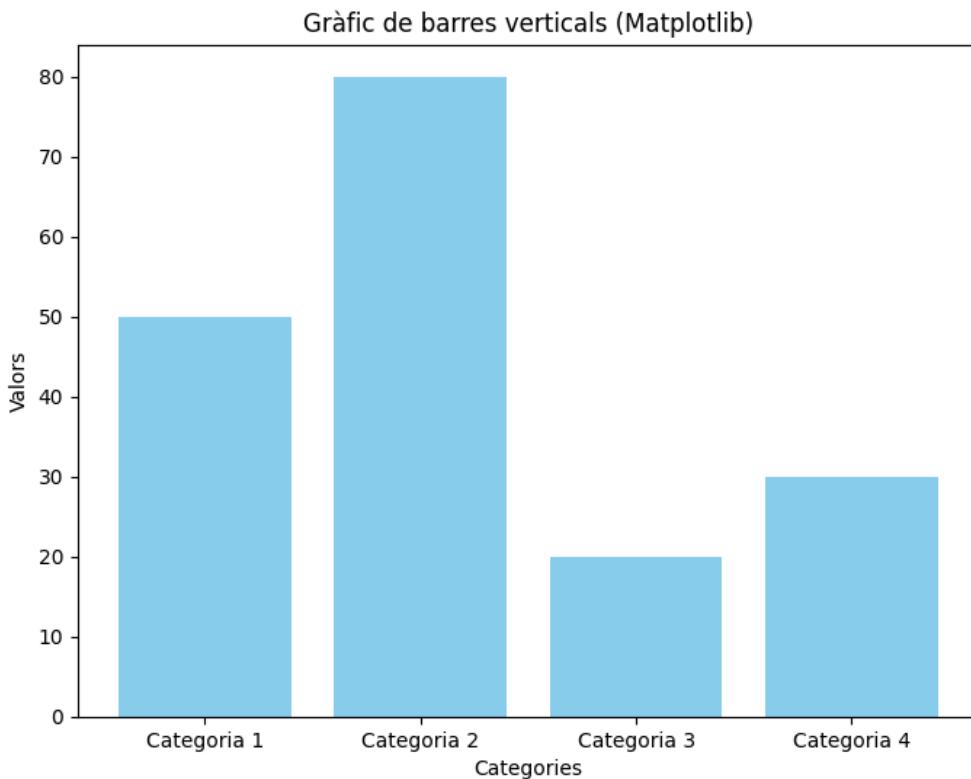
Podeu veure els exemples d'aquest apartat en aquest nou [quadern de Colab](#), on també s'ha inclòs l'exemple dels subplots que acabam de mencionar.

Per generar aquests dos tipus de gràfics farem feina amb aquestes dades:

```
etiquetes = ['Categoria 1', 'Categoria 2', 'Categoria 3', 'Categoria 4']
valors = [50, 80, 20, 30]
```

Vegem com generar el gràfic de barres verticals, emprant la funció ***plt.bar***:

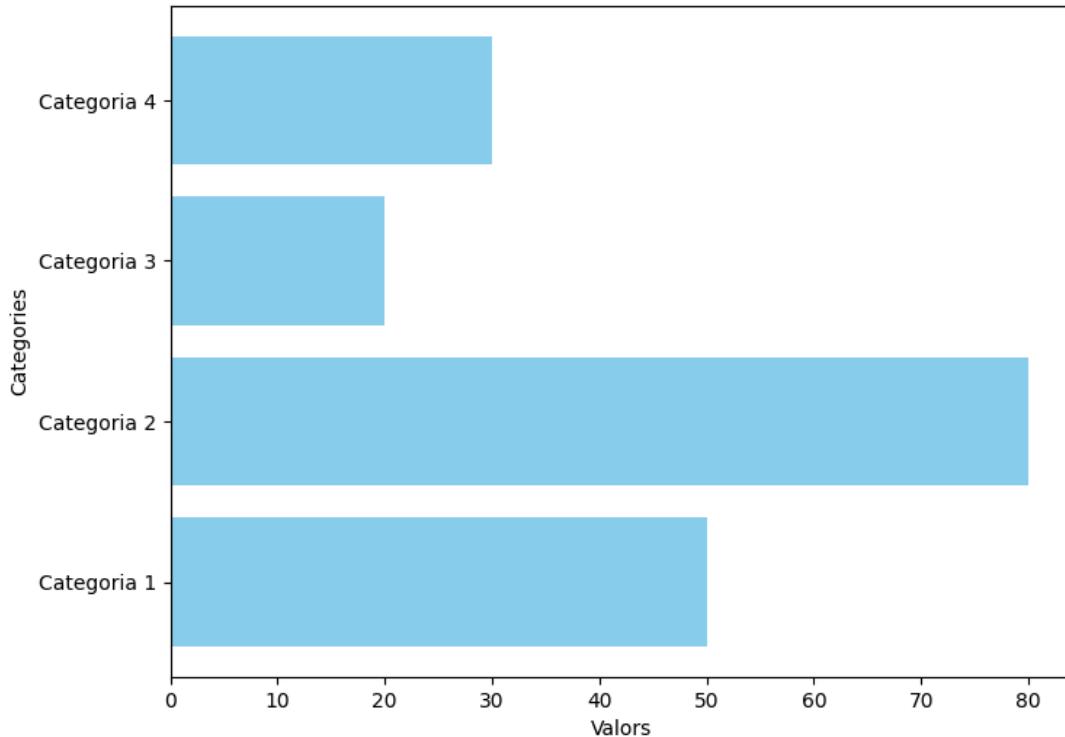
```
plt.figure(figsize=(8, 6))
plt.bar(etiquetes, valors, color='skyblue')
plt.title('Gràfic de barres verticals (Matplotlib)')
plt.xlabel('Categories')
plt.ylabel('Valors')
plt.show()
```



Si volem que les barres siguin horizontals, simplement hem d'emprar la funció ***plt.bart*** en lloc de *plt.bar*. A més, haurem d'intercanviar les etiquetes dels eixos.

```
plt.figure(figsize=(8, 6))
plt.bart(etiquetes, valors, color='skyblue')
plt.title('Gràfic de barres horizontals (Matplotlib)')
plt.xlabel('Valors')
plt.ylabel('Categories')
plt.show()
```

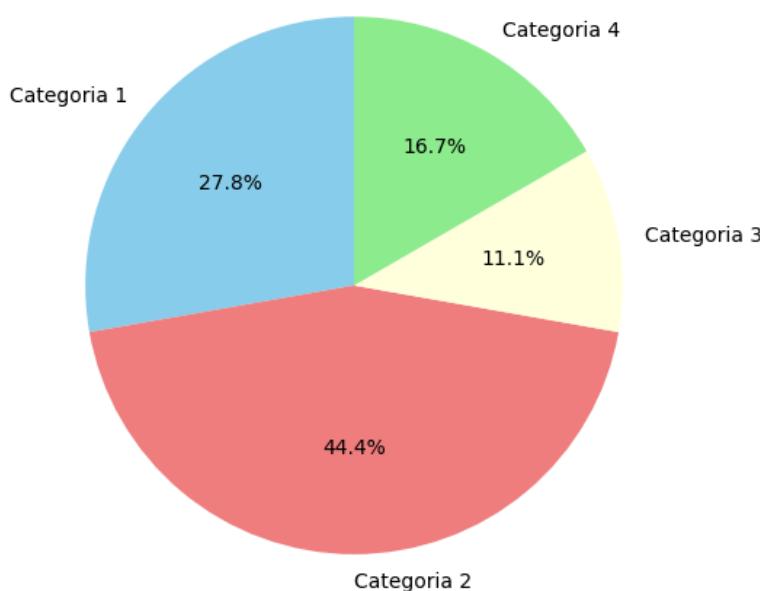
### Gràfic de barres horizontals (Matplotlib)



Finalment, vegem com generar el gràfic circular amb la funció `plt.pie`, on també hem triat un conjunt de colors determinat.

```
plt.figure(figsize=(6, 6))
plt.pie(valors, labels=etiquetes, autopct='%.1f%%', startangle=90,
        colors=['skyblue', 'lightcoral', 'lightyellow', 'lightgreen'])
plt.title('Gràfic circular (Matplotlib)')
plt.show()
```

### Gràfic circular (Matplotlib)



Sobre Matplotlib, s'han fet molts desenvolupaments per generar millors gràfics d'una manera senzilla. Alguns són biblioteques molt completes com Seaborn. Però d'altres són petits mòduls per implementar funcionalitats concretes. Per exemple, el mòdul **squarify** permet crear treemaps o el mòdul **wordcloud** per a crear niguls de paraules. Abans d'emprar aquests mòduls, és necessari instal·lar-los, per exemple amb pip (**wordcloud** ja està instal·lat en Colab).

Aquest codi genera un *treemap*, fent servir la funció **squarify.plot**.

```
import matplotlib.pyplot as plt
import squarify

# Dades d'exemple
etiquetes = ['Categoria 1', 'Categoria 2', 'Categoria 3',
             'Categoria 4', 'Categoria 5', 'Categoria 6']
valors = [250, 120, 280, 320, 140, 95]
colors = ['#91DCEA', '#64CDCC', '#5FBB68', '#F9D23C', '#F9A729', '#FD6F30']

# Treemap
squarify.plot(sizes = valors, label = etiquetes, color = colors )

# Eliminar els eixos:
plt.axis("off")

plt.show()
```



I aquest genera un nigul de paraules. Amb la classe *WordCloud* generam una imatge que carregam després amb Matplotlib.

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

text = "En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho \
tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, \
rocín flaco y galgo corredor."

wc = WordCloud(width = 300, height = 300).generate(text)

# Elimina els eixos i mostra les dades com una imatge
plt.axis("off")
plt.imshow(wc, interpolation = "bilinear")

plt.show()
```



A la <https://matplotlib.org/stable/> trobareu la documentació oficial de la biblioteca **Matplotlib**, amb exemples i tutorials de molts tipus diferents de gràfics.

A <https://github.com/laserson/squarify> trobareu la documentació del mòdul **squarify** i a [https://amueller.github.io/word\\_cloud/](https://amueller.github.io/word_cloud/) la del mòdul **wordcloud**.

Un bon recurs on trobar exemples de gràfics atractius és l'apartat de Matplotlib de la web Python charts: <https://python-charts.com/es/matplotlib/>

CONSELL

## 3.2. Seaborn

Matplotlib permet crear una gran varietat de tipus de gràfics i ofereix una gran varietat d'opcions per a personalitzar-los. A més és de codi obert i està suportat per una gran comunitat d'usuaris i desenvolupadors molt activa. No obstant això, dissenyar gràfics atractius amb Matplotlib pot arribar a ser complex. És per això que han sorgit moltes alternatives. **Seaborn** n'és probablement la més popular.

Seaborn és una biblioteca de Python d'alt nivell per a la visualització de dades, construïda sobre Matplotlib i que s'integra a la perfecció amb Pandas, facilitant la creació de gràfics atractius i informatius a partir de dades estructurades.

Seaborn proporciona una interfície més simple i intuitiva que Matplotlib, cosa que el fa ideal per a usuaris novells. A més, Seaborn ofereix una sèrie de temes i paletes de colors predefinits que permeten crear gràfics atractius amb un mínim esforç. Seaborn també incorpora funcions especialitzades que simplifiquen la visualització de dades estadístiques. Així, Seaborn suporta molts tipus de gràfics que sovint es fan servir en l'anàlisi de dades, com poden ser gràfics de barres amb barres d'error o gràfics de caixa (box plots). Per últim, també ha estat molt important per a la seva popularitat el fet que Seaborn s'integra perfectament amb Pandas, facilitant la visualització de DataFrames i Series.



Tots els exemples d'aquest apartat els trobareu a un [quadern de Colab](#).

Abans d'emprar Seaborn, l'hem d'installar amb pip (o altre gestor de dependències com conda). En el cas de Colab, ja està installat als servidors de Google.

```
!pip install seaborn
```

A continuació, importam Seaborn, usant el prefix `sns`.

```
import seaborn as sns
```

El mòdul `seaborn` proporciona una gran quantitat de funcions per a elaborar molts de tipus de gràfics diferents. Alguns dels més utilitzats són `sns.barplot` per a gràfics de barres, `sns.piechart` per a gràfics circulars, `sns.scatterplot` per a gràfics de dispersió de punts, `sns.boxplot` per a gràfics de caixes o `sns.heatmap` per a mapes de calor.

A més, Seaborn proporciona cinc [estils preconfigurats](#) per controlar l'estètica dels gràfics: "darkgrid" (per defecte), "whitegrid", "dark", "white" y "ticks".

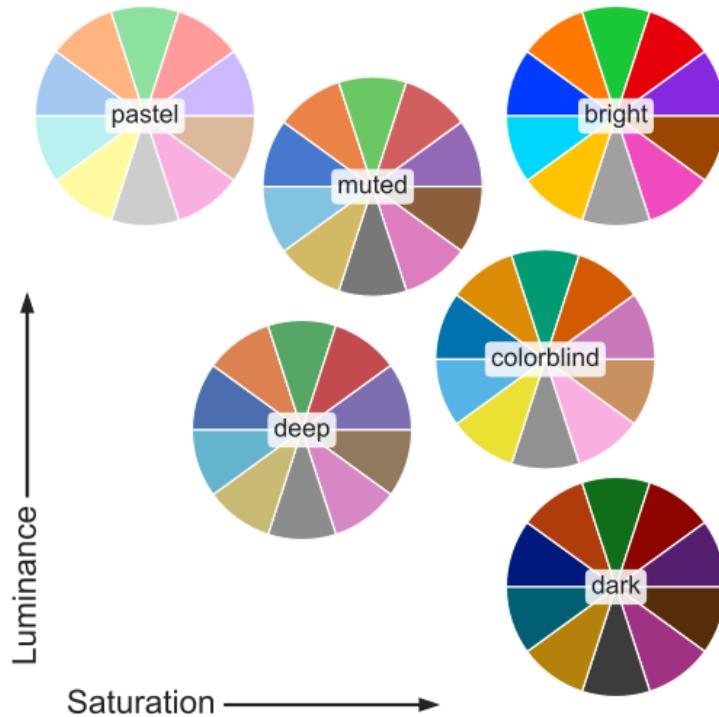
I també tenim una sèrie de [paletes de colors](#) predefinides. Aquesta és la paleta de colors per defecte:

```
paleta = sns.color_palette()  
sns.palplot(paleta)
```



**Imatge:** Paleta de colors per defecte de Seaborn

Són els mateixos colors i en el mateix ordre que la paleta per defecte de Matplotlib (anomenada "tab10"), tot i que amb colors una mica menys intensos. Sobre aquesta paleta bàsica, podem obtenir diverses variacions, modificant la luminància i saturació:



**Imatge:** Paletes de colors de Seaborn. Font: documentació de Seaborn.

A la documentació trobareu com obtenir altres combinacions de colors.

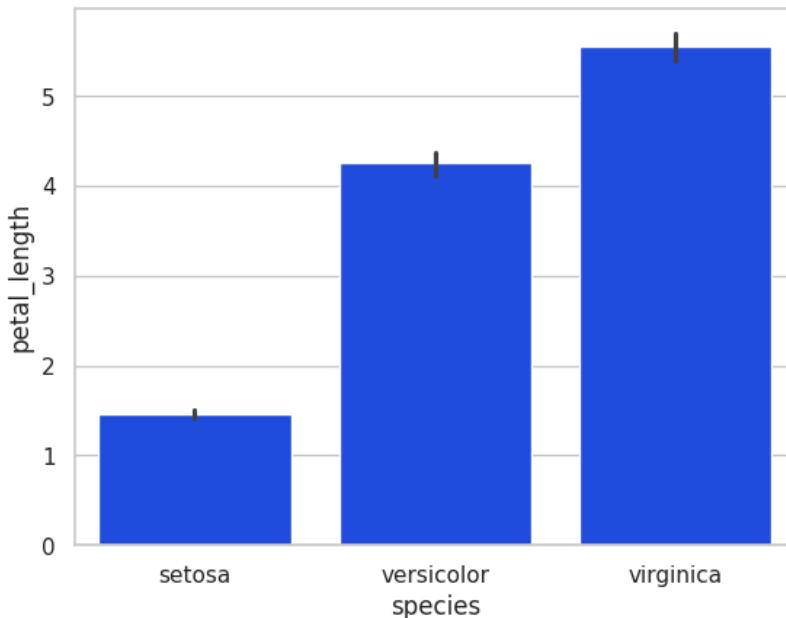
Vegem com crear un gràfic de barres, a partir del famós dataset iris. Representarem les 3 espècies de flors en l'eix X i la longitud dels pètals en l'eix Y:

```
import pandas as pd

# Carregam les dades de flors iris
df = sns.load_dataset("iris")

# Assignem un estil al gràfic (opcional)
sns.set_style("whitegrid")

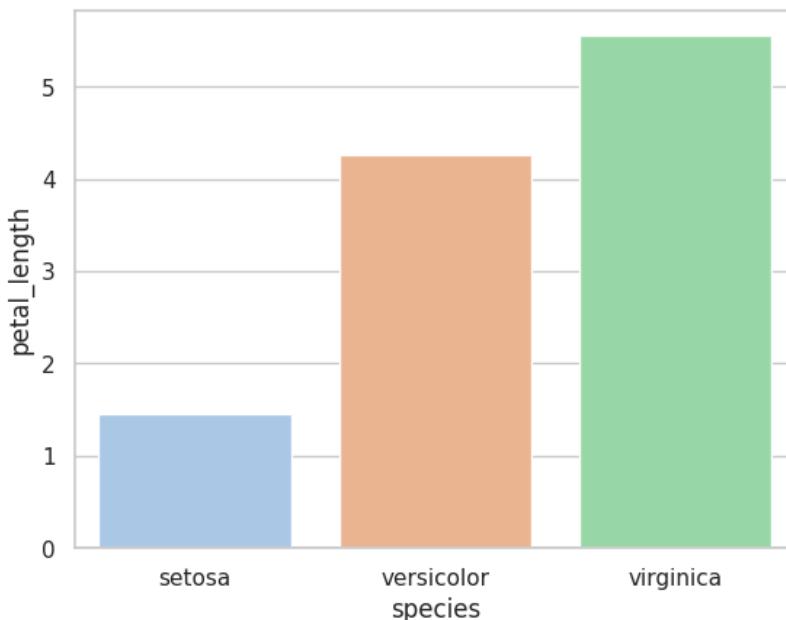
# Cream un diagrama de barres
sns.barplot(data=df, x='species', y='petal_length')
```



Podem observar que hem emprat l'estil whitegrid i la paleta per defecte. Totes les barres surten del primer color de la paleta. També veim que a sobre les barres blaves ens mostra una barra negra que mostra l'error, que ens ajuda a determinar si les diferències entre valors de la classe són significatives estadísticament.

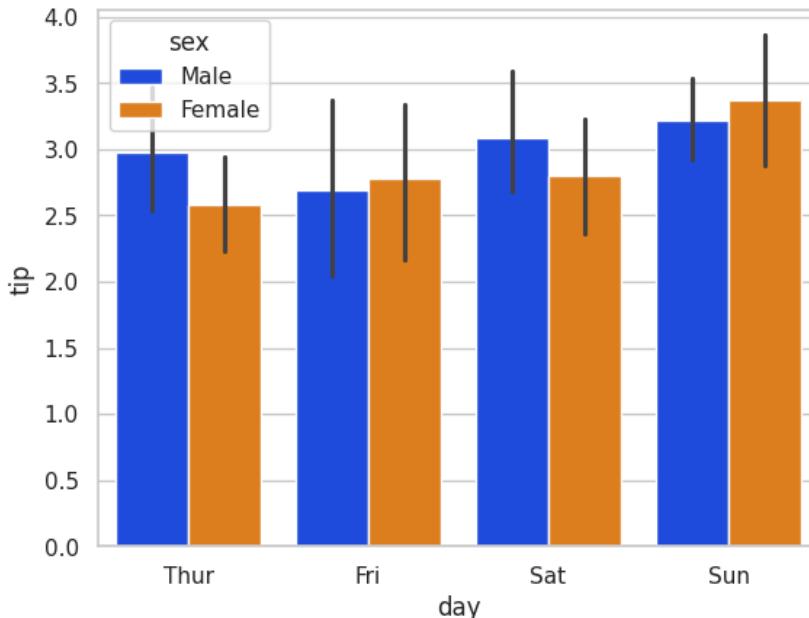
Per entendre millor com funciona la biblioteca, anam ara a fer que cada barra tengui un color diferent de la paleta "pastel" i que no es mostri la barra d'errors.

```
sns.barplot(data=df, x='species', y='petal_length',
             palette='pastel', errorbar=None)
```



Quan ho executam, podem veure que ens dona una advertència que indica que si empram el paràmetre 'palette', també hauríem de definir el 'hue'. El *hue* és un agrupament de valors en l'eix X. Vegem un altre exemple, per entendre-ho millor, amb un altre dataset. En aquest cas és un que conté els comptes i propines cobrades en un restaurant. Posarem el dia de la setmana (obren de dijous a diumenge) en l'eix X, la propina cobrada en l'eix Y i anam a distingir entre homes i dones (això és el *hue*).

```
df2 = sns.load_dataset("tips")
sns.barplot(data=df2, x='day', y='tip', hue='sex')
```

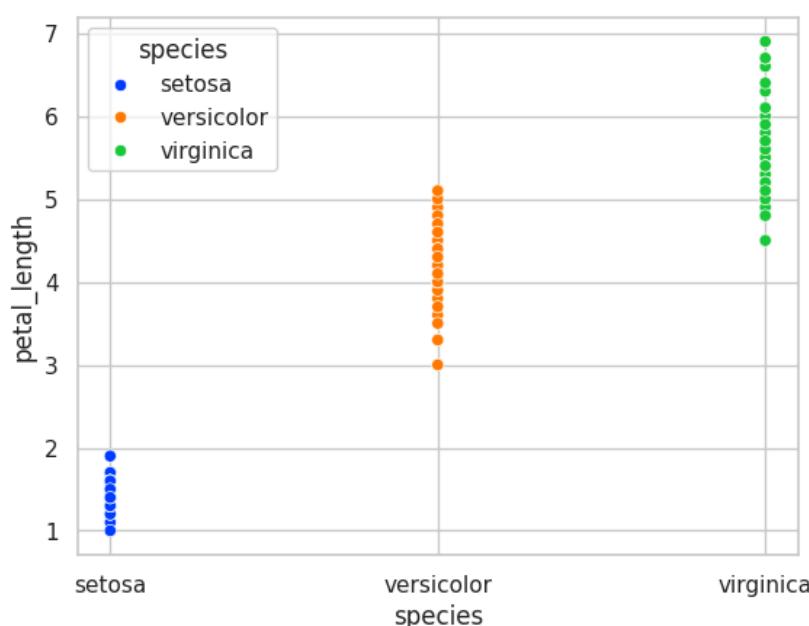


Així doncs, anam a tornar a fer el nostre gràfic de barres de les flors iris, idèntic a l'anterior, però sense cap warning:

```
sns.barplot(data=df, x='species', y='petal_length', hue='species', palette='pastel',
            errorbar=None)
```

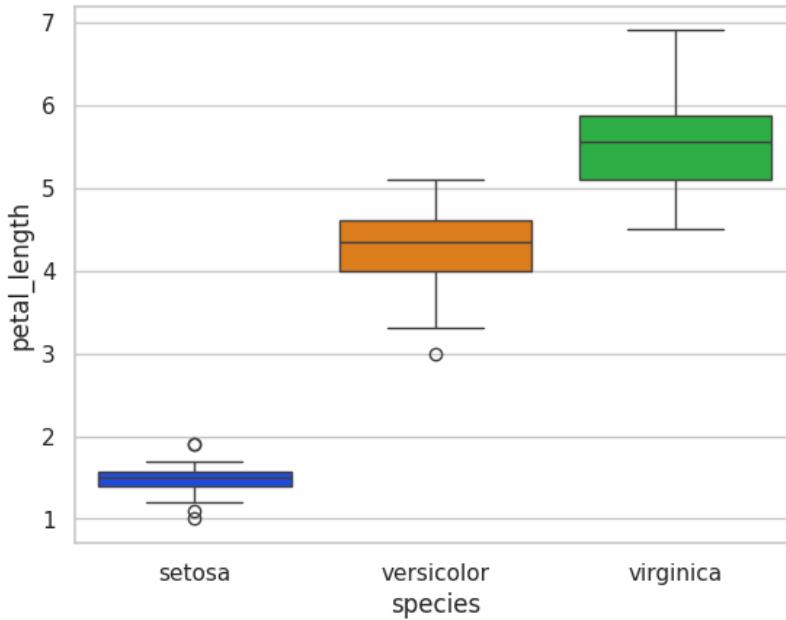
Un gràfic relacionat amb aquest molt interessant és el de dispersió, `sns.scatterplot`, que ens mostra com es situa cada un dels valors en les tres classes. Ens permet apreciar clarament com la setosa té un pètal molt més curt que les altres i que la majoria de les virginica tenen el pètal més llarg que la versicolor.

```
sns.scatterplot(data=df, x='species', y='petal_length', hue='species',
                 palette='bright')
```



Una informació semblant ens dona el gràfic de caixes, amb `sns.boxplot`:

```
sns.boxplot(data=df, x='species', y='petal_length', hue='species',
             palette='bright')
```

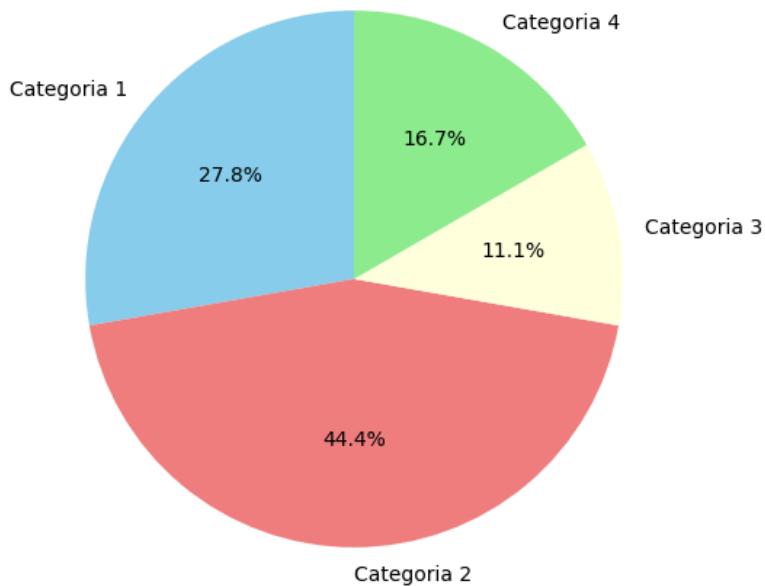


Per acabar amb aquest apartat, parlarem del gràfic circular. Curiosament, Seaborn no incorpora cap funció per dibuixar un gràfic d'aquest tipus, així que hem d'utilitzar el de Matplotlib. Però podem emprar les paletes de color de Seaborn en Matplotlib. Tornem a veure el gràfic circular de l'apartat anterior, ara amb la paleta "pastel":

```
import matplotlib.pyplot as plt

etiquetes = ['Categoria 1', 'Categoria 2', 'Categoria 3', 'Categoria 4']
valors = [50, 80, 20, 30]
colors = sns.color_palette('pastel')
sns.set_style("dark")
plt.figure(figsize=(6, 6))
plt.pie(valors, labels=etiquetes, autopct='%1.1f%%', startangle=90, colors=colors)
plt.title('Gràfic circular (Matplotlib i Seaborn)')
plt.show()
```

### Diagrama circular (Matplotlib)



A la <https://seaborn.pydata.org> trobareu la documentació oficial de la biblioteca **Seaborn**, amb exemples i tutorials de molts tipus diferents de gràfics.

Un bon recurs on trobar exemples de gràfics atractius és l'apartat de Seaborn de la web Python charts: <https://python-charts.com/es/seaborn/>

CONSELL

### 3.3. Plotly

Plotly és una altra biblioteca de Python d'alt nivell per a la visualització de dades. A diferència de Matplotlib i Seaborn, Plotly està dissenyada per a crear gràfics interactius que es poden explorar i manipular a la web.

Plotly genera gràfics en format HTML, JavaScript i CSS, el que permet visualitzar-los i interactuar-hi en un navegador web. A més, ofereix una API de Python per a crear i personalitzar els gràfics. La contrapartida és que generar els gràfics de Plotly pot ser més lent que els de Matplotlib i Seaborn, especialment quan treballam amb grans conjunts de dades.

Aquí podem veure una comparativa entre les tres biblioteques de visualització de dades:

| Característica            | Plotly                       | Matplotlib                  | Seaborn                       |
|---------------------------|------------------------------|-----------------------------|-------------------------------|
| Interactivitat            | Alta                         | Baixa                       | Baixa                         |
| Format                    | HTML, JavaScript, CSS        | PNG, SVG, PDF               | PNG, SVG, PDF                 |
| Enfocament                | Visualitzacions interactives | Creació de gràfics estàtics | Visualitzacions estadístiques |
| Dificultat d'aprenentatge | Moderada                     | Alta                        | Moderada                      |
| Rendiment                 | Mitjà                        | Alt                         | Alt                           |

Taula: Comparativa entre Plotly, Matplotlib i Seaborn

Com sempre, abans d'utilitzar la biblioteca Plotly, l'hem d'installar amb pip (o un altre gestor de dependències com conda). En el cas de Colab, ja està instal·lat en els servidors de Google.

```
!pip install plotly
```



Tots els exemples d'aquest apartat els trobareu a un [quadern de Colab](#).

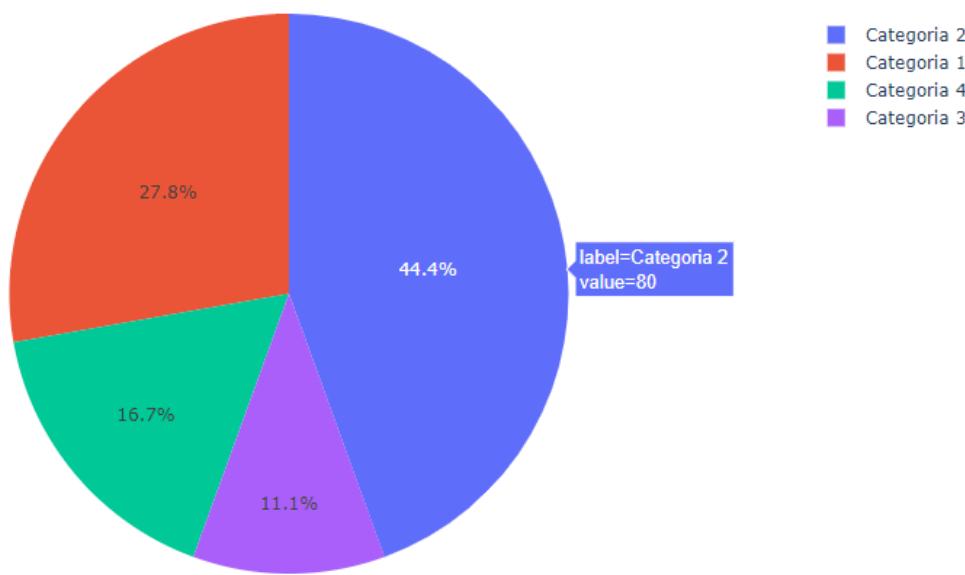
I després importarem el mòdul `plotly.express`, amb el prefix `px`:

```
import plotly.express as px
```

Vegem com dibuixar un gràfic circular amb les dades que hem emprat en els apartats anteriors:

```
etiquetes = ['Categoria 1', 'Categoria 2', 'Categoria 3', 'Categoria 4']
valors = [50, 80, 20, 30]

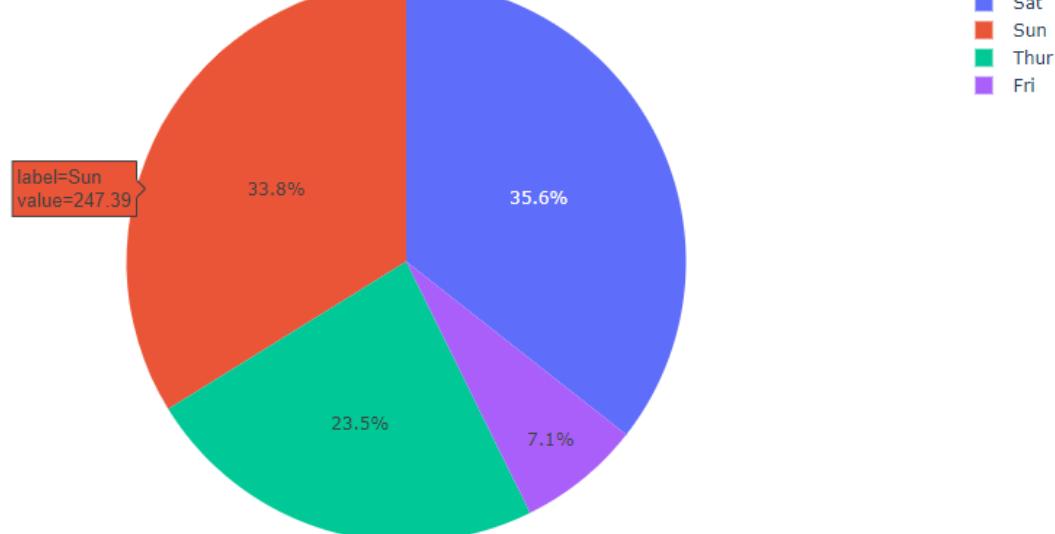
fig = px.pie(values = valors, names = etiquetes)
fig.show()
```



Podem comprovar que quan passam el ratolí per damunt d'un dels sectos del gràfic, apareix un *tooltip* amb els detalls: en la imatge veim que la Categoría 2, té un valor de 80. A més, també apareix una barra de botons sobre la imatge. En aquest cas, només té dos botons, un per a guardar la figura com a PNG i l'altre que indica que el gràfic està fet amb Plotly i enllaça a la seva pàgina web.

Anam a fer el mateix gràfic circular per al dataset de propines, que també està disponible en Plotly:

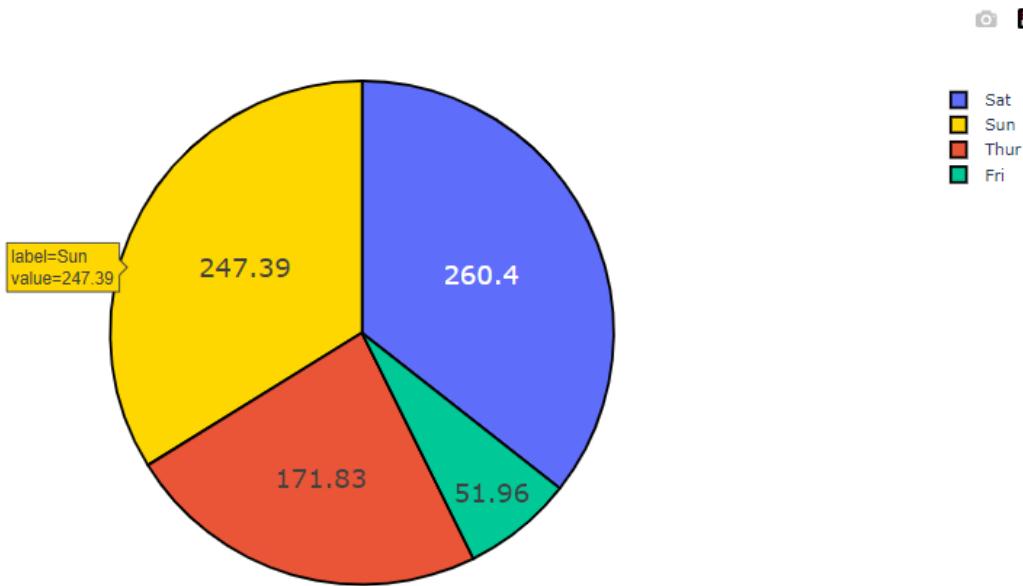
```
df = px.data.tips()
fig = px.pie(df, values='tip', names='day')
fig.show()
```



Vegeu un exemple de com podem personalitzar aquest gràfic:

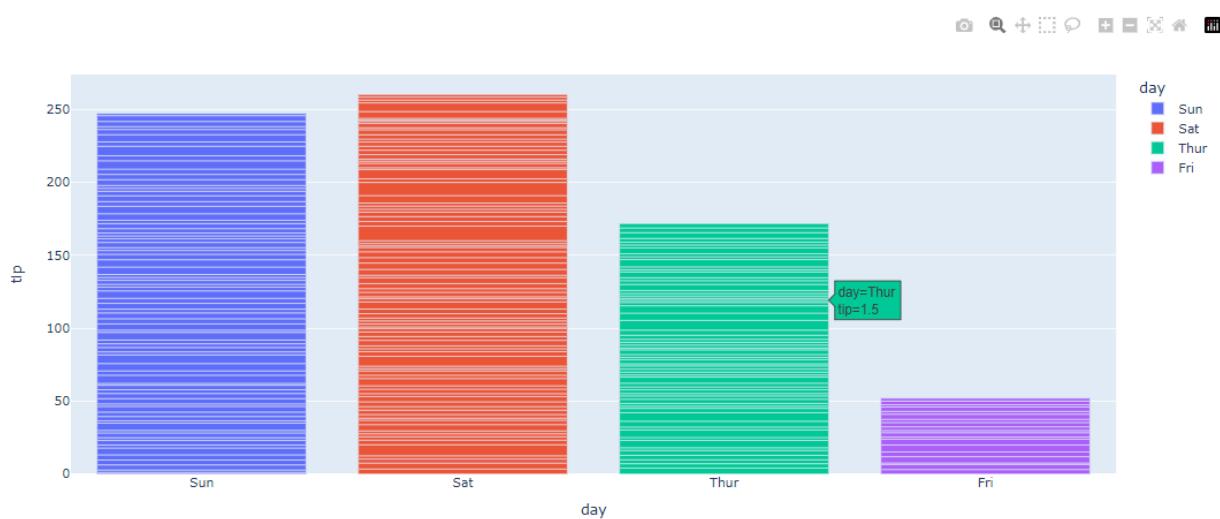
```
colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']

fig.update_traces(textinfo='value', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=2)))
fig.show()
```



Anem a veure ara un diagrama de barres a partir del dataset de propines.

```
df = px.data.tips()
fig = px.bar(df, x = 'day', y = 'tip', color='day')
fig.show()
```



Podem observar que ara tenim més botons en la barra d'eines. Entre d'altres ens apareixen botons de *zoom* i de *pan*.

## ■ Dash

Si volem donar més interactivitat als nostres gràfics, podem emprar Dash. Dash és un framework de codi obert, desenvolupat pels creadors de Plotly, per a crear quadres de comandament (*dashboards*) interactius, interfícies d'usuari personalitzades i aplicacions web amb dades.

Dash està construït sobre Plotly i Flask i funciona combinant aquests tres components principals:

- Plotly: Per a crear gràfics i visualitzacions de dades.
- Flask: Per a crear la infraestructura web de l'aplicació.
- Dash Components: Per a crear interfícies d'usuari interactives amb elements com ara botons, menús desplegables, controls lliscants i caixes de text.

Per emprar-ho, haurem d'installar Dash amb pip (o altre). Dash, no està installat en els servidors de Google Colab:

```
!pip install dash
```

I ara vegem com feim una interfície d'usuari que ens permet seleccionar el dia de la setmana, a partir del qual ens mostrerà les propines segons el sexe, i dins el sexe, separades per fumadors i no fumadors.

```
from dash import Dash, dcc, html, Input, Output
import plotly.express as px

app = Dash(__name__)

app.layout = html.Div([
    html.H4('Propines per dia'),
    dcc.Dropdown(
        id="bar-chart-x-dropdown",
        options=[
            {'label': 'Dijous', 'value': 'Thur'},
            {'label': 'Divendres', 'value': 'Fri'},
            {'label': 'Dissabte', 'value': 'Sat'},
            {'label': 'Diumenge', 'value': 'Sun'}
        ],
        value="Thur",
        clearable=False,
    ),
    dcc.Graph(id="bar-chart-x-graph"),
])

@app.callback(
    Output("bar-chart-x-graph", "figure"),
    Input("bar-chart-x-dropdown", "value"))
def update_bar_chart(day):
    df = px.data.tips()
    mask = df["day"] == day
    fig = px.bar(df[mask], x="sex", y="total_bill",
                 color="smoker", barmode="group")
    return fig

if __name__ == "__main__":
    app.run_server(debug=True)
```

## Propines per dia



**CONSELL**

A la <https://plotly.com/python/> trobareu la documentació oficial de la biblioteca **Plotly**, amb exemples i tutorials de molts tipus diferents de gràfics. A <https://plotly.com/dash/> trobareu la documentació de **Dash**.

Un bon recurs on trobar exemples de gràfics atractius és l'apartat de Plotly de la web Python charts:  
<https://python-charts.com/es/plotly/>

### 3.4. Datashtader

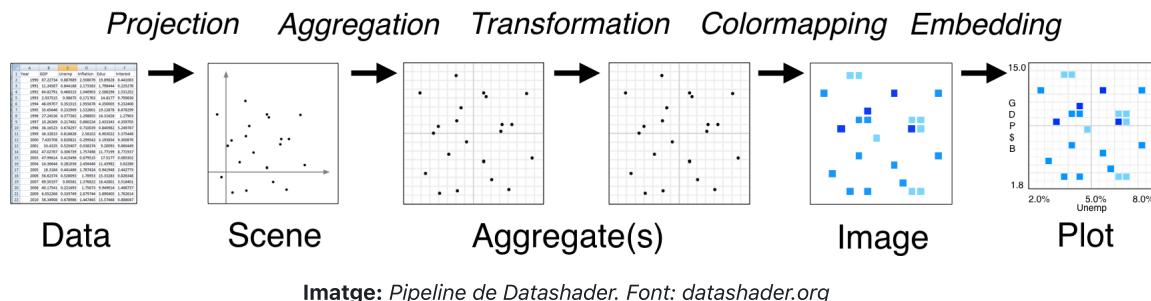
Datashtader és una biblioteca de codi obert per a Python 2 i 3 que permet realitzar visualitzacions de grans conjunts de dades. En concret, Datashtader està dissenyada per a dibuixar (rasteritzar) conjunts de dades en quadrícules regulars que poden analitzar-se més a fons o visualitzar-se com a imatges, la qual cosa permet veure de manera senzilla i ràpida les propietats i els patrons de les dades. Per a fer-se una idea de l'optimització del paquet Datashtader per a Big Data, en un portàtil de 16 GB de memòria RAM, Datashtader pot arribar a traçar mil milions de punts en un segon. A més, aquest paquet s'adapta fàcilment al processament distribuït o a la utilització d'una GPU, optimitzant el seu rendiment per a conjunts de dades encara majors.

Com sempre, necessitam instal·lar Datashtader per poder utilitzar-lo. Podem emprar pip o un altre gestor de dependències com conda. En els servidors de Google Colab no està instal·lat per defecte.

```
!pip install datashader
```

El funcionament de Datashtader es basa en un *pipeline*, una sèrie flexible d'etapes de processament que, a partir de dades en brut acaben obtenint imatges visibles. Entendre cadascuna d'aquestes etapes ajuda a treure el màxim profit de la biblioteca.

Aquestes són les etapes del pipeline de Datashtader:



Vegem breument què es fa a cada una de les cinc fases:

- **Projecció:** en aquesta fase, les dades es transformen en un espai de coordenades bidimensional. Això és necessari per a poder visualitzar les dades en una pantalla. Per a dades espacials, s'utilitza una projecció cartogràfica per a convertir les coordenades latitud/longitud en coordenades x/y. Per a dades de major dimensió, es pot utilitzar una tècnica de reducció de la dimensionalitat, com ara PCA o t-SNE, per a projectar les dades en un espai bidimensional.
- **Agregació:** aquesta és una fase crucial del *pipeline* de Datashtader. En aquesta etapa, les dades es divideixen en petits subconjunts, anomenats "agregats". Cada agregat conté un resum estadístic de les dades dins d'una regió espacial específica, que corresindrà a un píxel de la imatge resultant. Això pot ser, per exemple, la suma, la mitjana o el recompte de tots els punts de dades dins d'una regió. L'agregació permet visualitzar grans conjunts de dades de manera eficient, ja que només cal processar i renderitzar els agregats, en lloc de tots els punts de dades individuals.
- **Transformació:** en aquesta fase, es poden aplicar diverses transformacions als agregats. Això pot incloure operacions matemàtiques, com ara sumar, restar o multiplicar, o operacions lògiques, com ara seleccionar o filtrar. Les transformacions permeten modificar els agregats per a crear visualitzacions més específiques, o per a resaltar característiques particulars de les dades.
- **Mapejat de colors:** el mapejat de colors és també una fase molt important per a la visualització de dades. En aquesta etapa, es defineix una paleta de colors que s'utilitzarà per a representar els valors dels agregats en la visualització final. La paleta de colors pot ser lineal, seqüencial o divergent, i pot incloure colors discrets o gradients. La elecció de la paleta de colors és important per a comunicar de manera efectiva el significat de les dades.

- **Incrustació:** aquesta és la fase final del *pipeline* de Datashader. En aquesta etapa, els agregats colorejats s'incrosten en una imatge final. Això pot implicar superposar-los en un mapa, dibuixar-los com a punts o crear una trama de contorn. La incrustació permet crear visualitzacions de dades atractives i informatives que faciliten la comprensió dels patrons i tendències en les dades.

En aquest apartat anam a seguir diversos quadern de Colab.

Començarem amb un primer quadern amb una [introducció](#) a Datashader, on generarem un dataframe amb 50.000 punts aleatoris i veurem com funciona un *pipeline* de Datashader.

En el segon quadern treballarem amb [sèries temporals](#), una de les fonts de dades més habituals quan treballam amb dades massives. També generarem unes dades sintètiques, en aquest cas 3 sèries de 100.000 elements cadascuna.

En el tercer quadern, lligant amb el lliurament anterior, veurem com Dataframe també permet representar [grafs](#).

Per últim, acabarem amb un [cas pràctic](#), on analitzarem gràficament els viatges en taxi dins la ciutat de Nova York en el mes de gener de 2015 (més de 10 milions de moviments).



En aquest article, [Large Scale Visualizations and Mapping with Datashader](#), publicat per Finn Qiao en towardsdatascience.com, s'explica la importància d'utilitzar una biblioteca com Datashader quan treballam amb grans conjunts de dades. I es fa una anàlisi dels comerços de la ciutat de San Francisco, molt semblant a la del nostre cas pràctic.

Molt recomanable!

AMPLIACIÓ

## 4. Bibliografia

Una part d'aquests materials està basada en els apunts de *Sistemas de big data* (capítol *Visualización de datos*) de la Universidad de Castilla-La Mancha.

Com s'ha anat mencionant al llarg de tot el llibre d'apunts, a la documentació oficial de les diferents eines es poden trobar molts més exemples, d'una amplíssima varietat de gràfics:

- Matplotlib: <https://matplotlib.org/stable/>
- Mòdul squarify: <https://github.com/laserson/squarify>
- Mòdul wordcloud: [https://amueller.github.io/word\\_cloud/](https://amueller.github.io/word_cloud/)
- Seaborn: <https://seaborn.pydata.org>
- Plotly: <https://plotly.com/python/>
- Dash: <https://plotly.com/dash/>
- Datashader: <https://datashader.org/>

Un altre web, que també hem mencionat, amb molt de material útil sobre Matplotlib, Seaborn i Plotly és <https://python-charts.com/es/>