Anàlisi de Dades amb Python

Exploració, Supervivència i Clustering

Iker Pérez

2 de juliol del 2026

Contingut del curs

1. Situació en l'Ecosistema de Python
2. Exploració de Dades amb Pandas i Seaborn
3. Anàlisi de Supervivència amb Lifelines
4. Clustering amb Scikit-learn

Contingut del curs

1. Situació en l'Ecosistema de Python
2. Exploració de Dades amb Pandas i Seaborn
3. Anàlisi de Supervivència amb Lifelines
4. Clustering amb Scikit-learn

Per què Python? Motivació per a la migració

- Python és un dels llenguatges de programació més utilitzats al món.
- Ecosistema ampli: des de web fins a IA, passant per estadística i bioinformàtica.
- Gratuït i de codi obert, amb una gran comunitat i documentació extensa.
- Integració nativa amb eines modernes d'IA i Machine Learning.
- Aplicacions directes en l'anàlisi d'imatges i molt emprat en l'àmbit científic.
- Permet crear executables per tal d'usar-los sense la necessitat de recompilar el codi.

Objectiu del curs

Familiaritzar-vos amb l'equivalent Python de les eines que ja coneixeu en R, amb exemples pràctics de bioestadística.

Familiarització amb Google Colab

Google Colab és un entorn Jupyter al núvol, gratuït, que no requereix instal · lació local.

Enllaç: https://colab.research.google.com/drive/1TxwlqXBsP4VM8avl1On3d24XdlByvjyk?usp=sharing

Com s'utilitza

- Accediu a colab.research.google.com per crear un notebook nou.
- Un notebook conté **cel · les de codi** (Python) i **cel · les de text** (Markdown).
- Podeu executar les cel · les de codi amb el botó de "play".
- Instal · lar paquets addicionals: `pip install nom_paquet`.

Treballar en Local: Spyder i Executables

- **Spyder:** És l'IDE (Entorn de Desenvolupament) ideal per a bioestadístics. La seva interfície és pràcticament idèntica a **RStudio**.
- Proporciona explorador de variables, pestanya de gràfics i consola interactiva.
- Instal · lació recomanada via **Anaconda** (inclou Spyder i la majoria de paquets científics).

Com compartir els meus programes?

A diferència de scripts de R, en Python podem usar eines com `PyInstaller` per convertir el nostre codi en un **executable** (`.exe` a Windows o `.app` a Mac). Així, podem compartir eines amb el personal clínic sense que ells hagin d'instal · lar Python!

Sistema de Llibreries de Python

Les Llibreries Clau per a Bioestadística

Propòsit	Llibreria R	Llibreria Python	Funció principal
Manipulació de dades	`dplyr`, `tidyr`	`pandas`	DataFrames, filtrat, agrupació
Càlcul numèric	base R	`numpy`	Vectors, matrius, operacions
Visualització	`ggplot2`	`seaborn`, `matplotlib`	Gràfics estadístics
Tests estadístics	`stats` (base)	`scipy.stats`	t-test, chi-quadrat, ANOVA
Models estadístics	`lm()`, `glm()`	`statsmodels`	Regressió, GLM
Supervivència	`survival`, `survminer`	`lifelines`, `sksurv`	KM, Cox, ML Supervivència
Machine Learning	`caret`, `tidymodels`	`scikit-learn`	Clustering, classificació
Bioinformàtica	`Bioconductor`	`Biopython`	Seqüències, alineament

Com Importar Llibreries

R

```r
# Instal·lar
install.packages("dplyr")
install.packages("survival")

# Carregar
library(dplyr)
library(survival)
library(ggplot2)
```

Python

```python
# Instal·lar (des de terminal/Colab)
pip install pandas lifelines

# Carregar
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot
```

Sintaxi Bàsica de Python: Diferències Clau

Concepte	R	Python
Assignació	`x <- 5` o `x = 5`	`x = 5`
Indexació	1-based: `v[1]`	0-based: `v[0]`
Booleans	`TRUE`, `FALSE`	`True`, `False`
Valor nul	`NULL`, `NA`	`None`, `np.nan`
Pipe / Encadenat	`df %>% filter()`	`df.query()` o `.method()`
Definir funció	`f <- function(x) {...}`	`def f(x):`
Vector	`c(1, 2, 3)`	`[1, 2, 3]` o `np.array()`
Diccionari/Llista	`list(a=1, b=2)`	`{"a": 1, "b": 2}`
Bloc de codi	`{ ... }`	Indentació (4 espais)

Atenció: indexació basada en 0

A la gran majoria de llenguatges de programació, com és el cas de Python, el primer element d'una llista o columna és `[0]`, a diferència de R, que és `[1]`.

Carregar un Dataset

R

```r
# Des d'arxiu local
df <- read.csv("dades.csv")

# Des de paquet
data(lung, package="survival")

# Primeres files
head(df)
```

Python

```python
# Des d'arxiu local
df = pd.read_csv("dades.csv")

# Des d'URL publica (GitHub, etc.)
url = "https://raw.githubusercontent..."
df = pd.read_csv(url)

# Des de lifelines (datasets d'exemple)
from lifelines.datasets import load_rossi
df = load_rossi()

# Primeres files
df.head()
```

El Canvi de Sintaxi Més Important: Mètodes

Com es criden les funcions

A **R**, les funcions envolten l'objecte: `head(df)`, `summary(df)`.
A **Python**, els mètodes s'apliquen *des de* l'objecte: `df.head()`, `df.describe()`.

```
# R: funcio(objecte)          Python: objecte.metode()
# head(df)              ->     df.head()
# tail(df)              ->     df.tail()
# summary(df)           ->     df.describe()
# str(df)               ->     df.info()
# nrow(df)              ->     len(df)  o  df.shape[0]
```

Cheatsheet d'Importacions

Aquestes són les importacions que usarem durant tot el curs. Les trobareu a la primera cel · la de codi del notebook del curs.

```
# Dades i Calcul numeric
import pandas as pd
import numpy as np

# Visualitzacio
import matplotlib.pyplot as plt
import seaborn as sns

# Analisi de supervivencia
from lifelines import KaplanMeierFitter, CoxPHFitter
from lifelines.statistics import logrank_test

# Clustering
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

Contingut del curs

1 Situació en l'Ecosistema de Python

2 Exploració de Dades amb Pandas i Seaborn

3 Anàlisi de Supervivència amb Lifelines

4 Clustering amb Scikit-learn

Context i Objectius del dataset de Rossi

Referència Principal

Rossi, Berk i Lenihan (1980). *Money, Work, and Crime: Experimental Evidence*.

Pregunta d'Investigació

Pot un subsidi econòmic post-penitenciari reduir la taxa de reincidència criminal a curt termini?

Disseny de la Mostra

- $N = 432$ exreclusos (Maryland, EUA).
- Alliberats durant la dècada de 1970.
- Seguiment setmanal longitudinal (màxim de 52 setmanes).

Disseny Experimental i Variables

Metodologia Experimental

- **Grup de Tractament (50%):** Assignació aleatòria a rebre ajuda financera.
- **Grup de Control (50%):** Sense ajut financer.

Diccionari de Variables

Variable	Descripció	Rol en el model
`week`	Setmana del primer arrest o final de l'estudi	Variable dependent (t)
`arrest`	L'individu ha estat arrestat? (1=Sí, 0=No)	Indicador d'esdeveniment
`fin`	Ha rebut subsidi financer? (1=Sí, 0=No)	Variable de tractament
`prio`	Nombre d'arrestos previs	Covariable de control
`age`	Edat en el moment de l'alliberament	Covariable de control

Pandas: Exploració Bàsica del DataFrame

```python
# Carregar el dataset (usem el rossi com a exemple)
from lifelines.datasets import load_rossi
df = load_rossi()

# Informació general (equivalent a str() en R)
df.info()  # Mostra: nombre de files, columnes, tipus de dades, NaN

# Estadístiques descriptives (equivalent a summary() en R)
df.describe()  # Mostra: count, mean, std, min, 25%, 50%, 75%, max

# Dimensions
df.shape         # (432, 9) -> equivalent a dim(df)

# Primers/últims registres
df.head()        # 5 primeres files (head(df) en R)
df.tail(3)       # 3 últimes files (tail(df, 3) en R)

# Noms de columnes
df.columns       # equivalent a names(df) o colnames(df)
```

Pandas: Resum d'Equivalències d'Exploració

Funcions fonamentals per inspeccionar dades

R	Python (Pandas)	Què fa
`str(df)`	`df.info()`	Estructura: tipus, NaN, mida
`summary(df)`	`df.describe()`	Estadístiques descriptives
`dim(df)`	`df.shape`	(files, columnes)
`head(df)`	`df.head()`	Primeres files
`tail(df)`	`df.tail()`	Últimes files
`names(df)`	`df.columns`	Noms de columnes
`nrow(df)`	`len(df)`	Nombre de files
`table(df$col)`	`df['col'].value_counts()`	Freqüència de valors

Indexació i Filtratge

```
# Seleccionar columnes (equivalent a select() en dplyr)
df['age']                       # Una columna
df[['age', 'fin']]              # Múltiples columnes

# Filtrar files (equivalent a filter() en dplyr)
df[df['age'] > 30]              # Files on age > 30
df[(df['age'] > 30) & (df['fin'] == 1)]  # Múltiples filtres

# Indexació amb .loc i .iloc
df.loc[0:5, 'age':'prio']       # Per etiqueta (dels elements [0-5), agafa desde la variable age fins
    prio)
df.iloc[0:5, 2:5]               # Per posició (dels elements [0-5), agafa les variables [2:5))

# Equivalent al pipe (%>%) de dplyr
df.loc[df['age'] > 30, ['age', 'fin']] # Dels elements que compleixen la condició inicial, agafa desde
    la variable age fins fin).
```

Gestió de Valors Perduts (NaN)

En tenir valors desconeguts o perduts, aquests són representats com NaN (Not a Number). Tenim diverses tècniques per a gestionar-los.

```
# Detectar valors perduts
df.isna().sum()           # Comptar NaN per columna
# equivalent a: sapply(df, function(x) sum(is.na(x)))

# Eliminar files amb NaN
df_clean = df.dropna()    # equivalent a na.omit(df). El dataset "net" és df_clean

# Substituir NaN per un valor
df['col'] = df['col'].fillna(0)                  # Per 0
df['col'] = df['col'].fillna(df['col'].mean())   # Per la mitjana

# Filtrat boolea: files sense NaN en una columna
df[df['col'].notna()]
```

Atenció: `NaN` no és comparable directament

En Python, `NaN != NaN` dóna `True`. Utilitzeu sempre `.isna()` o `.notna()`, mai `== np.nan`.

Operacions de Neteja de Dades

```python
# Eliminar columnes
df = df.drop(columns=['columna_innecessaria'])

# Canviar el nom de columnes
df = df.rename(columns={'old_name': 'new_name'})

# Canviar tipus de dades
df['edat'] = df['edat'].astype(int)
df['grup'] = df['grup'].astype('category')  # equivalent a factor()

# Eliminar duplicats
df = df.drop_duplicates()

# Recodificar valors
df['sexe'] = df['sexe'].map({0: 'Home', 1: 'Dona'}) # Els valors 0 -> "Home", els valors 1 -> "Dona"
```

Introducció a Seaborn per a Visualització

Seaborn és l'equivalent més proper a `ggplot2` de R:

- S'integra directament amb els DataFrames de pandas.
- Estil visual atractiu per defecte i dissenyat per a visualització estadística.

Patró universal de Seaborn

La majoria de gràfics de Seaborn segueixen la mateixa estructura:
`sns.tipus_grafic(data=df, x='col_x', y='col_y', hue='grup')`

```
import seaborn as sns
import matplotlib.pyplot as plt

# Configurar l'estil global per als gràfics (https://python-graph-gallery.com/104-seaborn-themes/)
sns.set_theme(style="whitegrid")
```

Visualització: Boxplots (R vs Python)

R (ggplot2)

```r
ggplot(df, aes(x=factor(fin),
               y=age)) +
  geom_boxplot(fill="steelblue") +
  labs(x="Grup", y="Edat",
       title="Distribució d'edat")
```

Python (Seaborn)

```python
sns.boxplot(data=df,
            x='fin',
            y='age',
            palette='Blues')
plt.xlabel('Grup')
plt.ylabel('Edat')
plt.title("Distribució d'edat")
plt.show()
```

Visualització: Clustermap

```
# Clustermap - equivalent a heatmap.2() en R (gplots) o a pheatmap()

sns.clustermap(
    df.corr(),                    # Matriu de correlació (ha de ser un dataframe net!)
    method='ward',                # Mètode de linkage
    cmap='coolwarm',              # Paleta de colors
    annot=True,                   # Mostrar valors
    fmt='.2f',                    # Format numèric (2 decimals)
    figsize=(8, 6),
    standard_scale=1              # Estandarditzar per columnes
)
plt.title('Clustermap de Correlacions')
plt.show()
```

Visualització: Altres Gràfics Útils

```
# Histograma
sns.histplot(data=df, x='age', kde=True)

# Gràfic de dispersió (scatter) amb color segons un grup
sns.scatterplot(data=df, x='age', y='prio', hue='fin')

# Gràfic de violí (combinació de boxplot i estimació de densitat)
sns.violinplot(data=df, x='fin', y='age')
```

Contingut del curs

1. Situació en l'Ecosistema de Python
2. Exploració de Dades amb Pandas i Seaborn
3. Anàlisi de Supervivència amb Lifelines
4. Clustering amb Scikit-learn

Introducció a Lifelines

- És la llibreria d'anàlisi de supervivència estàndard en Python.
- Equivalent al paquet `survival` + `survminer` de R.
- Proporciona: Kaplan-Meier, Cox PH, Log-Rank, i models paramètrics.
- Incorpora visualització nativa de les corbes i models.

Instal · lació i Importació

```
pip install lifelines

# Principals eines que utilitzarem
from lifelines import KaplanMeierFitter, CoxPHFitter
from lifelines.statistics import logrank_test
```

Kaplan-Meier

R

```r
library(survival)
library(survminer)

# Crear objecte de supervivència
fit <- survfit(
  Surv(week, arrest) ~ 1,
  data = rossi
)

# Visualitzar
ggsurvplot(fit,
  xlab = "Setmanes",
  ylab = "Prob. Supervivència")
```

Python

```python
from lifelines import KaplanMeierFitter

# Crear i ajustar estimador
kmf = KaplanMeierFitter()
kmf.fit(
    durations=df['week'],
    event_observed=df['arrest']
)

# Visualitzar
kmf.plot_survival_function()
plt.xlabel('Setmanes')
plt.ylabel('Prob. Supervivència')
plt.title('Corba Kaplan-Meier')
plt.show()
```

Anàlisi de la informació del Kaplan-Meier

```python
# Mediana de supervivencia
print(f"Mediana: {kmf.median_survival_time_}")

# Taula de supervivencia completa
print(kmf.survival_function_)

# Intervals de confianca
print(kmf.confidence_interval_survival_function_)

# Probabilitat de supervivencia en un temps concret
# Exemple: probabilitat de sobreviure (no reincidir) a les 30 setmanes
prob_30 = kmf.predict(30)
print(f"Probabilitat S(t=30): {prob_30:.4f}")
```

Atributs clau del `KaplanMeierFitter`

- `median_survival_time_` — Mediana de supervivència.
- `survival_function_` — Taula $S(t)$ completa.
- `confidence_interval_survival_function_` — IC al 95%.
- `predict(t)` — $S(t)$ per a un temps concret.

Comparació de Grups amb KM

```python
# Comparar dos grups
fig, ax = plt.subplots(figsize=(8, 5))

for name, grouped_df in data.groupby('fin'):
    kmf = KaplanMeierFitter()
    label = 'Amb ajuda financera' if name == 1 else 'Sense ajuda'
    kmf.fit(
        durations=grouped_df['week'],
        event_observed=grouped_df['arrest'],
        label=label
    )
    kmf.plot_survival_function(ax=ax)
    print(f"{label}: Mediana = {kmf.median_survival_time_}")

plt.title('Corbes KM per Grup de Tractament')
plt.xlabel('Temps (setmanes)')
plt.ylabel('Probabilitat de Supervivència')
plt.legend(loc='lower left')
plt.grid(alpha=0.3)
plt.show()
```

Test de Log-Rank — Comparació de Grups

R

```r
survdiff(
  Surv(week, arrest) ~ fin,
  data = rossi
)
```

Python

```python
from lifelines.statistics import logrank_test

# Separar els DataFrames per grup
g1 = data[data['fin'] == 1]
g0 = data[data['fin'] == 0]

# Executar test log-rank
result = logrank_test(
    durations_A=g1['week'],  # Temps a esperar
    durations_B=g0['week'],
    event_observed_A=g1['arrest'],  #
        Esdeveniment a observar
    event_observed_B=g0['arrest']
)
result.print_summary()
```

Interpretació dels Resultats del Log-Rank

```
# Accedir als valors clau del test
print(f"chi²: {result.test_statistic:.4f}")
print(f"P-valor: {result.p_value:.4f}")

# Resum formatat (similar a l'output de R)
result.print_summary()
# O bé amb print(result)
```

Principals valors de retorn

- `result.test_statistic` retorna la χ^2
- `result.p_value` retorna el p-valor
- `result.print_summary()` retorna la taula completa

CoxPHFitter — Anàlisi Multivariant (Model de Cox)

R

```r
cox_model <- coxph(
  Surv(week, arrest) ~
    fin + age + race +
    wexp + mar + paro + prio,
  data = rossi
)

# Resultats
summary(cox_model)
```

Python

```python
from lifelines import CoxPHFitter

# Crear i ajustar/entrenar el model
cph = CoxPHFitter()
cph.fit(
    df,
    duration_col='week',
    event_col='arrest'
)
# S'usen per defecte totes les
# columnes excepte temps i esdeveniment

cph.check_assumptions(df) # Verificació dels
     residus de Schoenfeld

# Resultats
cph.print_summary()
```

Interpretació del Summary de Cox

Exemple de taula de sortida (simplificada):

	coef	exp(coef)	se(coef)	z	p	lower	upper
fin	-0.38	0.68	0.19	-1.98	0.048	-0.75	-0.003
age	-0.06	0.94	0.02	-2.61	0.009	-0.10	-0.01
prio	0.09	1.10	0.03	3.19	0.001	0.04	0.15

Com interpretar les columnes clau:

- `coef`: Coeficient de regressió ($\log$ del hazard ratio).
- `exp(coef)`: Hazard Ratio (HR).
- `p`: P-valor.
- `lower`/`upper`: Interval de confiança al 95%.

Anàlisi del Summary amb filtres

```
# Obtenir el summary complet com un DataFrame
summary_df = cph.summary

# Filtrar i seleccionar variables significatives (p < 0.05)
significant = summary_df[summary_df['p'] < 0.05]
print("Variables significatives:")
print(significant[['exp(coef)', 'p']])

# C-index
print(f"C-index: {cph.concordance_index_:.4f}")
# El C-index mesura la capacitat predictiva global del model.
```

Forest Plot (Gràfic de Hazard Ratios)

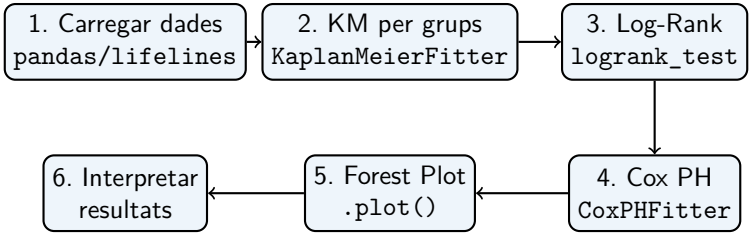
```
# Forest Plot - Visualització dels Hazard Ratios i els IC
fig, ax = plt.subplots(figsize=(8, 5))
cph.plot(ax=ax)
plt.title('Forest Plot - Hazard Ratios amb IC 95%')
plt.axvline(x=0, color='black', linestyle='--') # Línia en log(HR)=0 (HR=1)
plt.tight_layout()
plt.show()
```

Interpretació visual del Forest Plot:

- Cada punt central és el log(HR) de la covariable.
- La línia horitzontal representa l'interval de confiança al 95%.

En R s'obté amb `ggforest(cox_model, data = rossi)` del paquet `survminer`.

Resum del Flux d'Anàlisi de Supervivència



Contingut del curs

1. Situació en l'Ecosistema de Python
2. Exploració de Dades amb Pandas i Seaborn
3. Anàlisi de Supervivència amb Lifelines
4. Clustering amb Scikit-learn

Scikit-learn — L'Estàndard de ML a Python

- `scikit-learn` és la llibreria principal de Machine Learning en Python.
- Equivalent a paquets com `caret` o `tidymodels` en R.
- Ofereix una API molt consistent per a **tots** els algorismes (KMeans, SVM, Random Forest...):
  1. **Importar** la classe de l'algorisme.
  2. **Instanciar** el model amb els paràmetres desitjats.
  3. **Ajustar** el model a les dades amb `.fit(X)`.
  4. **Predir** o **Transformar** amb `.predict(X)` o `.transform(X)`.

Importacions típiques per a Clustering

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

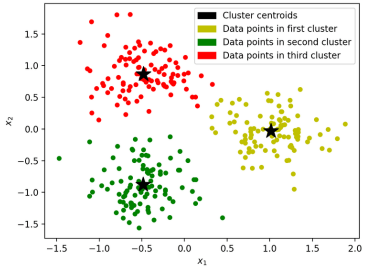
Introducció al Clustering: K-Means

Què és un Clúster?

- Un agrupament d'observacions amb característiques similars.
- L'objectiu és aconseguir una alta cohesió interna i una gran separació amb la resta de grups.

Com funciona el K-Means?

- Particiona les dades en k grups predefinits.
- Assigna cada observació al seu **centroide** més proper (utilitzant distàncies com l'euclidiana).
- Cerca minimitzar la **inèrcia** (la suma de distàncies de cada punt respecte al seu centre).



Cas Pràctic: El Dataset "Wine"

Aquest conjunt de dades clàssic prové d'una anàlisi química de vins cultivats a la mateixa regió d'Itàlia, però derivats de tres varietats de raïm diferents.

Característiques del Dataset

- **Mida:** 178 mostres (vins).
- **Variables (*Features*):** 13 atributs exclusivament numèrics (Grau d'alcohol, Àcid Màlic, Magnesi, Prolina, etc.).
- **Classes Reals:** 3 varietats de raïm.

Consideracions

Les diferents variables d'aquest dataset es troben en diferents unitats. L'alcohol es mesura en percentatge (%), el magnesi en (mg/l), ...

Escalat de Dades — StandardScaler

R

```r
# Estandarditzar dades
X_scaled <- scale(X)
```

Python

```python
from sklearn.preprocessing import StandardScaler

# Instanciar el scaler
scaler = StandardScaler()

# Ajustar i transformar
X_scaled = scaler.fit_transform(X)
```

Fòrmula aplicada (Z-score): $$z = \frac{x - \mu}{\sigma}$$

On: x valor original, μ mitjana de la columna i σ desviació estàndard.

Per què cal escalar les dades?

K-Means usa la distància euclidiana. L'escalat uniforme iguala totes les escales, fent que la contribució de totes les variables sigui igual.

K-Means Clustering

R

```r
result <- kmeans(
  X_scaled,
  centers = 3,
  nstart = 10
)

result$cluster    # Etiquetes
result$centers    # Centroides
result$tot.withinss # Inercia
```

Python

```python
from sklearn.cluster import KMeans

# Instanciar amb els parametres
kmeans = KMeans(
    n_clusters=3,
    random_state=42,
    n_init=10
)

# Ajustar a les dades
kmeans.fit(X_scaled)

kmeans.labels_           # Etiquetes
kmeans.cluster_centers_  # Centroides
kmeans.inertia_          # Inercia
```

Resultats del K-Means

- `n_clusters`: nombre de grups (equivalent a `centers` en R).
- `random_state`: fixa la llavor d'aleatorietat de l'algorisme per garantir la reproduïbilitat de l'anàlisi.
- `n_init`: nombre de repeticions de K-Means. L'algorisme es quedarà amb el millor resultat.

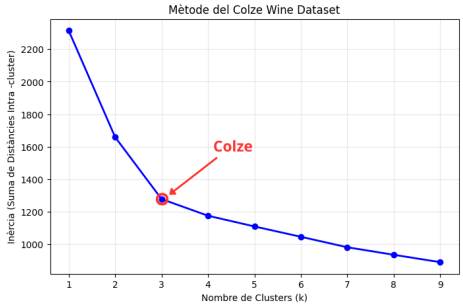
```
# Com assignar els clusters obtinguts al dataframe original
df['cluster'] = kmeans.labels_

# Càlcul de la mitjana de cada cluster
print(df.groupby('cluster').mean(numeric_only=True))

# Elements de cada cluster
print(df['cluster'].value_counts())
```

Triar el Nombre Òptim de Clusters: Mètode del Colze

Interpretació:

El punt on la corba fa un **"colze"** (canvi brusc de pendent) indica el nombre òptim de clusters.

A partir d'aquest punt, afegir més grups no millora substancialment l'homogeneïtat interna (inèrcia), ja que la reducció del valor és marginal.



Triar el Nombre Òptim de Clusters: Mètode del Colze

```
inertias = []
K_range = range(1, 10)

for k in K_range:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    km.fit(X_scaled)
    inertias.append(km.inertia_) # Guardem la inèrcia per cada k

# Gràfic
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertias, 'bo-', linewidth=2)
plt.xlabel('Nombre de Clusters (k)')
plt.ylabel('Inèrcia (Suma de Distàncies Intra-cluster)')
plt.title('Mètode del Colze')
plt.grid(alpha=0.3)
plt.show()
```

Visualització Final dels Clusters

```
millor_k = 3 # Cal definir el valor k òptim
kmeans = KMeans(n_clusters=millor_k, random_state=42, n_init=10)
kmeans.fit(X_scaled)

# Visualitzar
fig, ax = plt.subplots(figsize=(8, 5))
scatter = ax.scatter(
    X_scaled[:, 0], X_scaled[:, 1],
    c=kmeans.labels_, cmap='viridis', alpha=0.6, s=50
)

# Afegir els centroides al gràfic
ax.scatter(
    kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
    c='red', marker='X', s=200, edgecolors='black',
    linewidths=2, label='Centroides'
)
plt.colorbar(scatter, label='Cluster')
plt.title(f'K-Means Clustering (k={millor_k})')
plt.legend()
plt.show()
```

Aplicació Clínica del Clustering

Aplicacions reals en investigació biomèdica i tica:

- **Estratificació de pacients**: Agrupar pacients a partir de múltiples variables clíniques o analítiques per definir perfils (ex: risc alt, mig, baix).
- **Patrons de resposta al tractament**: Descobrir de forma no supervisada quins subgrups de pacients tenen evolucions similars.
- **Descobriment de fàrmacs**: Classificar grans llibreries de compostos químics segons la seva similitud estructural per optimitzar la selecció de candidats per a assajos de laboratori.

Resum i taula d'equivalències

Bloc d'Anàlisi	**Eines Python**	**Equivalent R**
Exploració de dades	`pandas`	`dplyr` + `tidyr`
Visualització	`seaborn` / `matplotlib`	`ggplot2`
Supervivència	`lifelines`	`survival` + `survminer`
Machine Learning / Clustering	`scikit-learn`	`caret` + base `stats`

Recursos extra

- **Pandas**: pandas.pydata.org/docs
- **Seaborn**: seaborn.pydata.org
- **Lifelines**: lifelines.readthedocs.io
- **Scikit-learn**: scikit-learn.org
- **Google Colab**: colab.research.google.com
- **Kaggle**: kaggle.com/datasets
- **Video sobre clustering**:https://www.youtube.com/watch?v=iNIZ3IU5Ffw

Gràcies per la vostra atenció!

iker.perez@estudiantat.upc.edu