Taula de contingut

Format d´entrega

Aquest document s´ha realitzat mitjançant **Markdown**[1] amb l´ajuda de l´entorn de desenvolupament **RStudio**[2] utilitzant les característiques que aquest ofereix per a la creació de documents **R** reproduïbles.

La documentació generada en la realització de la pràctica es troba allotjada en **GitHub** al següent repositori:

- **https://github.com/rsanchezs/data-minig**

En aquest repositori es poden trobar els següents fitxers:

- Aquest document en formats **pdf** i **docx** amb el nom rsanchezs_PAC2.
- Un document **R Markdown**[3] que es pot utilitzar per a reproduir tots els exemples presentats a la PAC.
- El conjunt de dades utilitzades.

[1] La documentació oficial es pot trobar a: **http://www.sthda.com/english/rpkgs/factoextra**.

[2] Per a més informació: **https://es.wikipedia.org/wiki/K-medoids**

[3] **https://rmarkdown.rstudio.com/**

1. **Nota:** Propietat intel·lectual

Sovint és inevitable, al produir una obra multimèdia, fer ús de recursos creats per terceres persones. És per tant comprensible fer-lo en el marc d'una pràctica dels Estudis, sempre que això es documenti clarament i no suposi plagi en la pràctica.
Per tant, al presentar una pràctica que faci ús de recursos aliens, s'ha de presentar juntament amb ella un document en quin es detallin tots ells, especificant el nom de cada recurs, el seu autor, el lloc on es va obtenir i el seu estatus legal: si l'obra està protegida pel copyright o s'acull a alguna altra llicència d'ús (Creative Commons, llicència GNU, GPL ...).
L'estudiant haurà d'assegurar-se que la llicència no impedeix específicament el seu ús en el marc de la pràctica. En cas de no trobar la informació corresponent haurà d'assumir que l'obra està protegida per copyright.
Hauríeu de, a més, adjuntar els fitxers originals quan les obres utilitzades siguin digitals, i el seu codi font si correspon.

Exercici 1

En relació amb el cas pràctic que vaig desenvolupar a la PAC1 es tractava d´implementar un *recommender system* (de l´anglès, sistemes de recomanació). És per això que, utilitzar els mètodes no supervistats no seria un bona elecció.

Avui dia, l´algoritme *Nearest Neighborhood* (de l´anglès, algoritme del veí més proper), és el més utilitzat avui dia en els sistemes de recomanació.

Els algoritmes de veïns més pròxims s´han desenvolupat en dues perspectives possibles: recomanació de veïns pròxims per usuari i per ítem:

- **Centrats en usuari *(User kNN)*:** Es recomanen a l'usuari ítems que han agradat a usuaris similars.
- **Centrats en ítem *(Item kNN)*:** Es recomanen a l'usuari ítems que es pareixen a ítems que li han agradat.

Exercici 2

Requisits

Per començar, per a la realització del nostre anàlisi necessitarem els següents paquets:

- `cluster` per a la computació dels algoritmes d´agregació.
- `factoextra` per a la visualització de resultats d´agregació i que es fonamenta en el paquet `ggplot2`.[4]
- `clValid` que s´utilitza per a comparar els mètodes d´agregació.

El paquet `factoextra` conté funcions per anàlisi de *clustering* i visualització dels resultats:

Funció	Descripció
`dist(fviz_dist, get_dist)`	Visualització i computació de la matriu de distàncies
`get_clust_tendency`	Avaluació de la tendència d´agregació
`fviz_nbclust(fviz_gap_stat)`	Determinació del nombre òptim de clústers
`fviz_dend`	Visualització de dendrogrames

[4] La documentació oficial es pot trobar a: **http://www.sthda.com/english/rpkgs/factoextra**.

`fviz_cluster`	Visualització dels resultats d´agrupament
`fviz_mclust`	Visualització dels resultats del model d´agrupament
`fviz_silhouette`	Visualització de la informació de la silueta
`hkmeans`	K-means jeràrquic
`eclust`	Visualització de l´anàlisi de agrupament

Podem instal·lar els dos paquets com es mostra en la següent línia de codi:

```
# Instalació paquets clustering
install.packages(c("cluster", "factoextra", "clValid"))
```

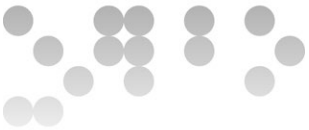
En acabat, ens caldrà carregar les llibreries a la sessió R:

```
# Carreguem les llibreries
library(cluster)
library(factoextra)
library(clValid)
```

Preparació de les dades

D´entrada, per a realitzar una anàlisi d´agregació en R cal assegurar-se d´unes quantes coses:

- Que les files es corresponen a observacions (individuals) i les columnes a variables.
- Qualsevol valor desconegut en el nostre conjunt de dades ha de ser o bé eliminat o bé substituït per exemple amb el valor de la mitjana o per el valor més freqüent.
- Les dades han de ser estar discretitzades.



Per il·lustrar l´anàlisi d´agregació farem ús del conjunt de dades `USArrests`, que conté dades estadístiques d´agressions, assassinats i violacions en cada un dels 50 estats d´USA l´any 1973.

```
# Carreguem les dades
data("USArrests")
df <- USArrests
```

En primer lloc, podem eliminar els valors desconeguts en el nostre conjunt de dades com es mostra a continuació:

```
# Eliminem valors desconeguts
df <- na.omit(df)
```

En segon lloc, discretitzarem les nostres dades estandaritzant-les amb l´ajuda de la funció `scale()`:

```
# Estandaritzem les variables
df <- scale(df)
head(df, n = 3)
```

```
##              Murder   Assault   UrbanPop         Rape
## Alabama 1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska  0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona 0.07163341 1.4788032  0.9989801  1.042878388
```

Determinació del nombre de clústers

Per a determinar el nombre de clústers farem ús de la funció `fviz_nbclust()` del paquet `factoextra` que calcula els mètodes **Elbow**, **Silhouhette** i **Gap**.

El prototip de la funció es el següent:

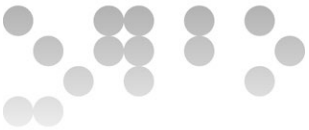
```
fviz_nbclust(x, FUNcluster, method = c("silhouette", "wss", "gap_stat
"))
```

on els arguments són els següents:

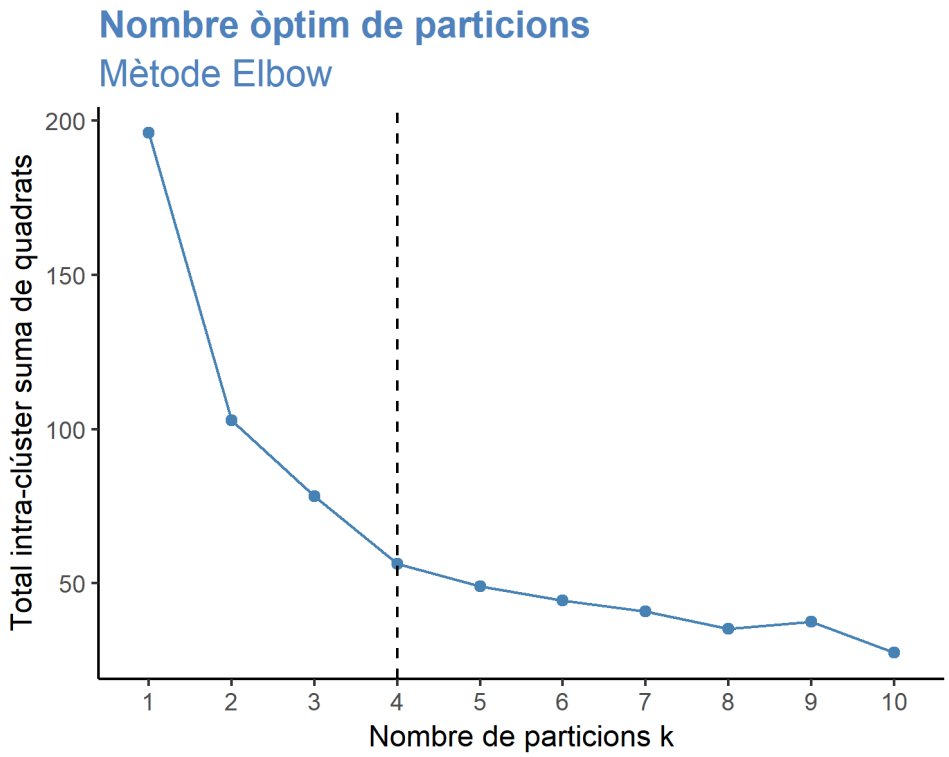
- **x:** matriu o data frame.
- **FUNcluster:** una funció d´agregació. Valors possibles: kmeans, pam, clara i hcut.
- **method:** mètode per a determinar el nombre òptim de clústers. Valors possibles: **Elbow**, **Silhouhette** i **Gap**

A continuació, es mostra com determinar el nombre òptim de particions per al mètode *k-means*:

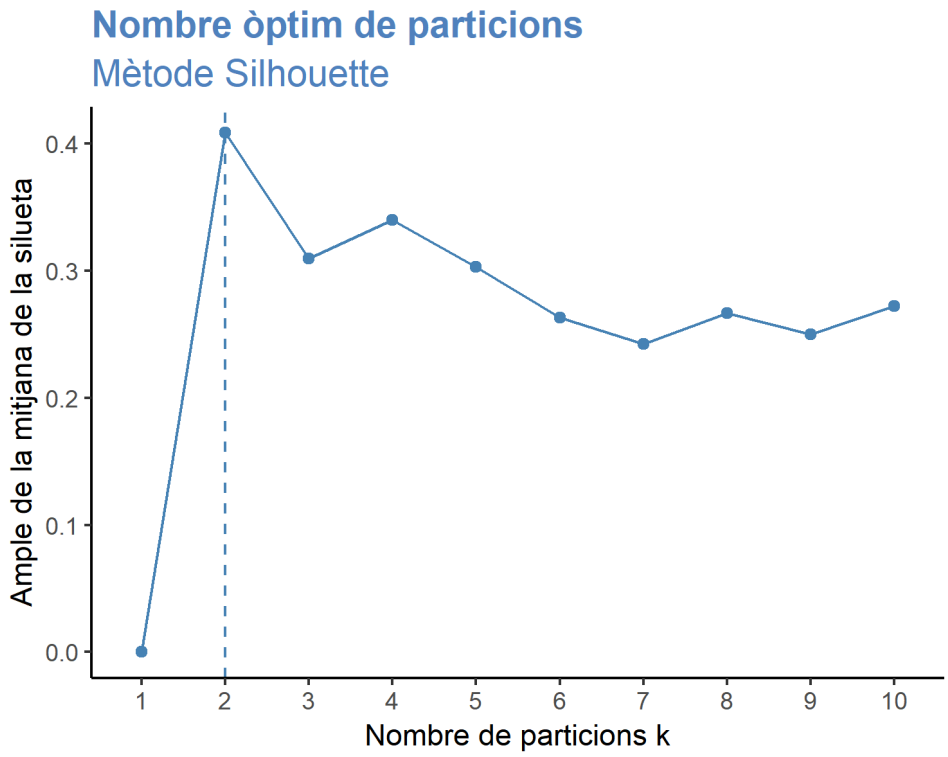
```
# Mètode elbow
fviz_nbclust(df, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2) +
  labs(x = "Nombre de particions k",
       y = "Total intra-clúster suma de quadrats",
       title = "Nombre òptim de particions",
       subtitle = "Mètode Elbow") +
  theme_classic() +
```



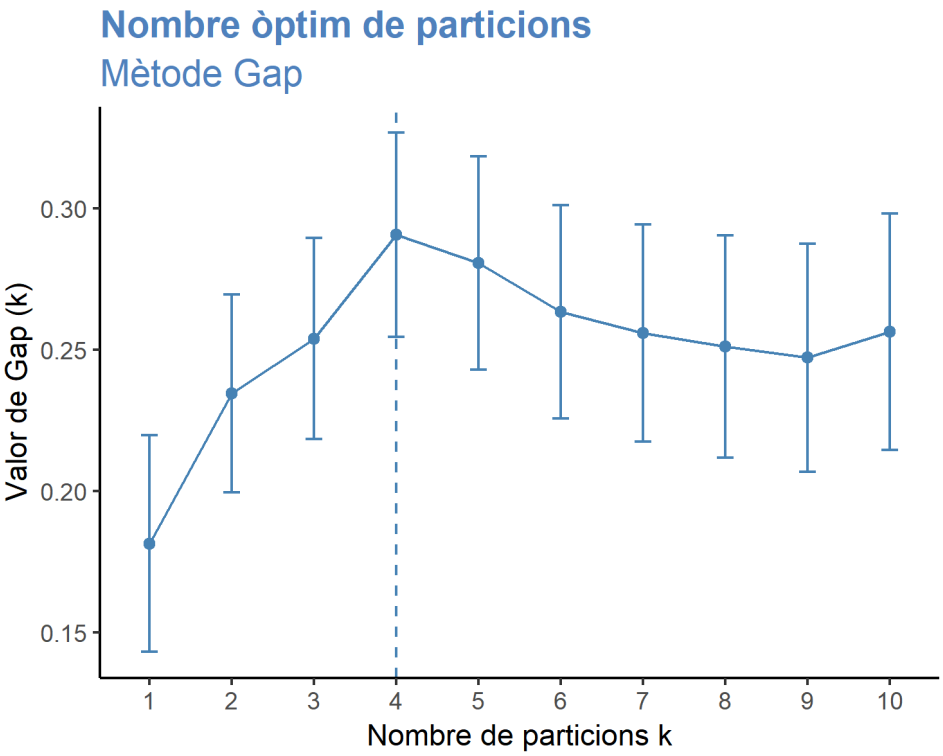
```
  theme(plot.title = element_text(color="#4F81BD", size=14, face="bold
"),
        plot.subtitle = element_text(color="#4F81BD", size=14))
```




```
# Mètode Silhouette
fviz_nbclust(df, kmeans, method = "silhouette") +
  labs(x = "Nombre de particions k",
       y = "Ample de la mitjana de la silueta",
       title = "Nombre òptim de particions",
       subtitle = "Mètode Silhouette") +
  theme_classic() +
  theme(plot.title = element_text(color="#4F81BD", size=14, face="bold
"),
        plot.subtitle = element_text(color="#4F81BD", size=14))
```




```
# Mètode Gap
set.seed(123)
fviz_nbclust(df, kmeans, nstart = 25,
             method = "gap_stat", nboot = 500) +
  labs(x = "Nombre de particions k",
       y = "Valor de Gap (k)",
       title = "Nombre òptim de particions",
       subtitle = "Mètode Gap") +
  theme_classic() +
  theme(plot.title = element_text(color="#4F81BD", size=14, face="bold
"),
        plot.subtitle = element_text(color="#4F81BD", size=14))
```



Com podem observar en els gràfics:

- El mètode Elbow ens suggereix 4 clústers.
- El mètode Silhoutte ens suggereix 2 clústers.
- El mètode Gap ens sugereix 4 clústers.

Així és que, segons aquestes observacions podem considerar $k = 4$ com el nombre òptim de clústers.

Mètode d´agregació *k-means*

A causa de que, l´algoritme *k-means* comença seleccionant un centroide aleatòriament, es recomanable fer ús de la funció `set.seed()` a l´efecte de aconseguir resultats reproduïbles. Així el lector d´aquest document obtindrà els mateixos resultats que es presenten tot seguit.

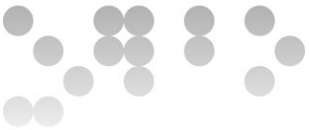
A continuació es mostra com aplicar l´algorisme k-means amb k = 4:

```
# Execució k-means amb k = 4
set.seed(123)
kmeansFit <- kmeans(df, 4, nstart = 25)
```

Podem mostrar per pantalla els resultats amb la següent línia de codi:

```
# Mostrem els resultats
print(kmeansFit)
```

```
## K-means clustering with 4 clusters of sizes 13, 16, 13, 8
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 2 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 3  0.6950701  1.0394414  0.7226370  1.27693964
## 4  1.4118898  0.8743346 -0.8145211  0.01927104
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas
California
##              4              3              3              4
3
##       Colorado    Connecticut       Delaware        Florida
Georgia
##              3              2              2              3
4
```



```
##         Hawaii          Idaho       Illinois        Indiana
Iowa
##              2              1              3              2
1
##         Kansas       Kentucky      Louisiana          Maine
Maryland
##              2              1              4              1
3
##  Massachusetts       Michigan      Minnesota    Mississippi
Missouri
##              2              3              1              4
3
##        Montana       Nebraska         Nevada  New Hampshire
New Jersey
##              1              1              3              1
2
##     New Mexico       New York North Carolina   North Dakota
Ohio
##              3              3              4              1
2
##       Oklahoma         Oregon   Pennsylvania   Rhode Island Sou
th Carolina
##              2              2              2              2
4
##   South Dakota      Tennessee          Texas           Utah
Vermont
##              1              4              3              2
1
##       Virginia     Washington  West Virginia      Wisconsin
Wyoming
##              2              2              1              1
2
##
## Within cluster sum of squares by cluster:
## [1] 11.952463 16.212213 19.922437  8.316061
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
```

```
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Podem observar en la sortida el següent:

- La mitjana de clústers: una matriu, on les files són el nombre de clúster i les columnes són les variables.
- El vector de particions: un vector d´enters (de 1:k) que indica el clúster on cada observació ha sigut agrupada.

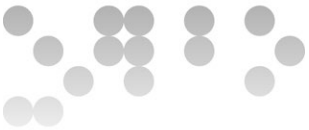
Així mateix, és recomanable realitzar un gràfic amb els resultats del model. Ja sigui, per a escollir el nombre de clústers, ja sigui per a comparar diferents anàlisis.

Una possible opció és visualitzar les dades en un diagrama de dispersió acolorint cada observació d'acord al grup assignat.

El problema és que el nostre conjunt de dades conté més de 2 variables i no és possible representar el model en dues dimensions.

Convé fer ressaltar que, una possible solució és reduir la dimensionalitat fent ús d´un algoritme de reducció del nombre d´atributs, com per exemple **Principal Component Analysis (PCA)**.

En aquest sentit, farem ús de la funció `fviz_cluster()` que ens permetrà visualitzar els clústers i que utilitza PCA quan el nombre de variables és més gran de 2. Passarem com a arguments el resultat del model i el conjunt de dades original:



```
# Visualitzem els clústers
fviz_cluster(kmeansFit, data = df,
     main = "Gràfic de clústers",
     palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
     ellipse.type = "euclid",
     star.plot = TRUE,
     repel = TRUE,
     ggtheme = theme_minimal()) +
  theme(legend.position = "bottom",
        plot.title = element_text(color="#4F81BD", size=16, face="bold
"))
```

Gràfic de clústers

Podem observar en el gràfic que les observacions són representades mitjançant punts i que en el nostre cas s´ha usat PCA. A més, s´han dibuixat el.lipses per tal de diferenciar cada clúster.

Exercici 3

Mètode d´agregació *k-medoids*

El següent apartat tracta del mètode d´agregació **k-medoids** i la seva implementació mitjançant l´algoritme de **Partició al voltant de Medoids PAM)**.

Igual com k-means, K-medoid és una tècnica clàssica de partició de grups que divideix les dades conformades per n objectes en k grups (amb k fixat *a priori*).

És més robust davant el soroll i valors atípics que k-means perquè minimitza una suma de dissimilituds (entre parells de punts) en comptes d'una suma de distàncies euclidiana quadrades.

Un **medoid** pot ser definit com l'objecte d'un grup on la seva dissimilitud mitjana a tots els objectes en el grup és mínima. És el punt situat més cap al centre en tot el grup.[5]

Per a determinar el nombre de clústers farem ús de la funció `pam()` del paquet `cluster`.

El prototip de la funció es el següent:

```
pam(x, k, metric = "euclidean", stand = FALSE)
```

on els arguments són els següents:

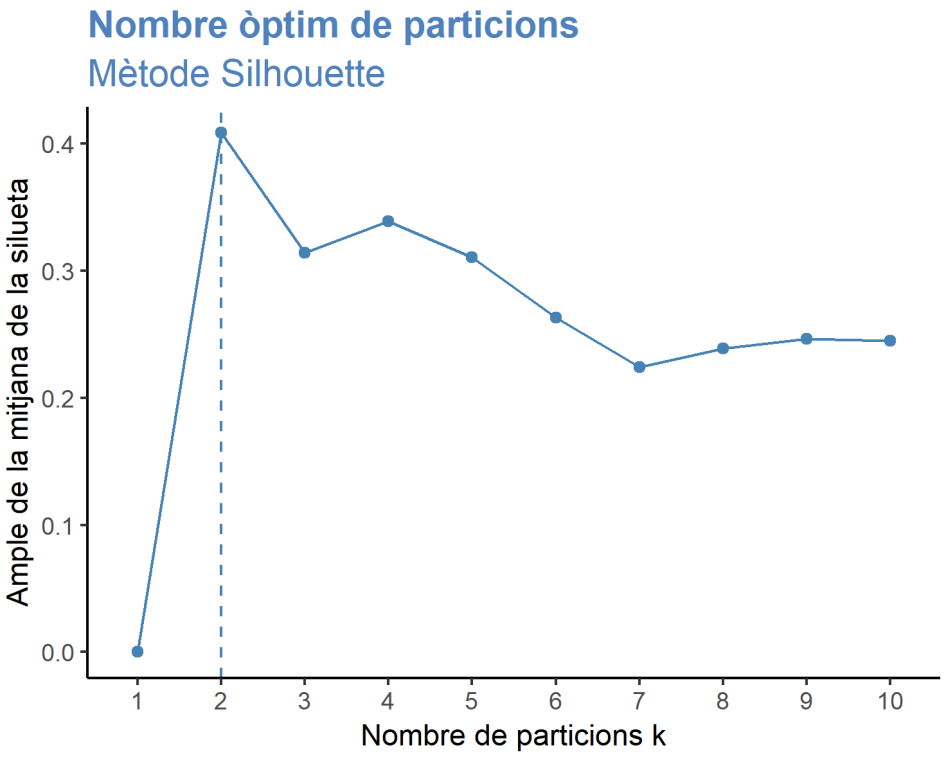
[5] Per a més informació: **https://es.wikipedia.org/wiki/K-medoids**

- **x:** on x pot ser:
- Una matriu o data frame de tipus numèric: cada fila correspon a una observació i cada columna a una variable.
- Una matriu de dissimilituds: en aquest cas x es normalment la sortida o bé de la funció `daisy()` o bé de `dist()`.
- **K:** el nombre de clústers.
- **metric:** la mesura de similitud. O bé "euclidiana" o bé "manhattan".
- **stand:** un valor de tipus lògic; si es TRUE, les variables en x són estandarditzades.

Estimació del nombre òptim de clùsters

Per a estimar el nombre òptim de clústers utilitzarem la mètrica de Silhoutte. La idea central és calcular l´algoritme PAM amb diferents valors de k. Per a la realització d´aquesta tasca farem ús de la funció `fviz_nbclust()`:

```
# Obtenció de k utilitzant la mètrica Silhoutte
fviz_nbclust(df, pam, method = "silhouette") +
  labs(x = "Nombre de particions k",
       y = "Ample de la mitjana de la silueta",
       title = "Nombre òptim de particions",
       subtitle = "Mètode Silhouette") +
  theme_classic() +
  theme(plot.title = element_text(color="#4F81BD", size=14, face="bold
"),
        plot.subtitle = element_text(color="#4F81BD", size=14))
```



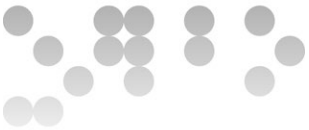
Com podem observar en el gràfic el resultat per al mètode Silhoutte ens suggereix 2 clústers.

Càlcul del mètode PAM

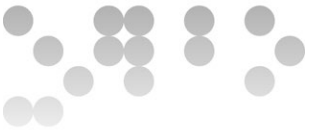
El següent codi calcula el mètode PAM amb k = 2:

```
# Execució de l´algoritme PAM
pamFit <- pam(df, 2)
# Visualització de resultats
print(pamFit)

## Medoids:
##            ID     Murder    Assault   UrbanPop       Rape
## New Mexico 31  0.8292944  1.3708088  0.3081225  1.1603196
```



```
## Nebraska   27 -0.8008247 -0.8250772 -0.2445636 -0.5052109
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas
California
##              1              1              1              2
1
##       Colorado    Connecticut       Delaware        Florida
Georgia
##              1              2              2              1
1
##         Hawaii          Idaho       Illinois        Indiana
Iowa
##              2              2              1              2
2
##         Kansas       Kentucky      Louisiana          Maine
Maryland
##              2              2              1              2
1
##  Massachusetts       Michigan      Minnesota    Mississippi
Missouri
##              2              1              2              1
1
##        Montana       Nebraska         Nevada  New Hampshire
New Jersey
##              2              2              1              2
2
##     New Mexico       New York North Carolina   North Dakota
Ohio
##              1              1              1              2
2
##       Oklahoma         Oregon   Pennsylvania   Rhode Island Sou
th Carolina
##              2              2              2              2
1
##   South Dakota      Tennessee          Texas           Utah
Vermont
##              2              1              1              2
2
##       Virginia     Washington  West Virginia      Wisconsin
Wyoming
##              2              2              2              2
```



```
2
## Objective function:
##    build     swap
## 1.441358 1.368969
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isola
tion"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

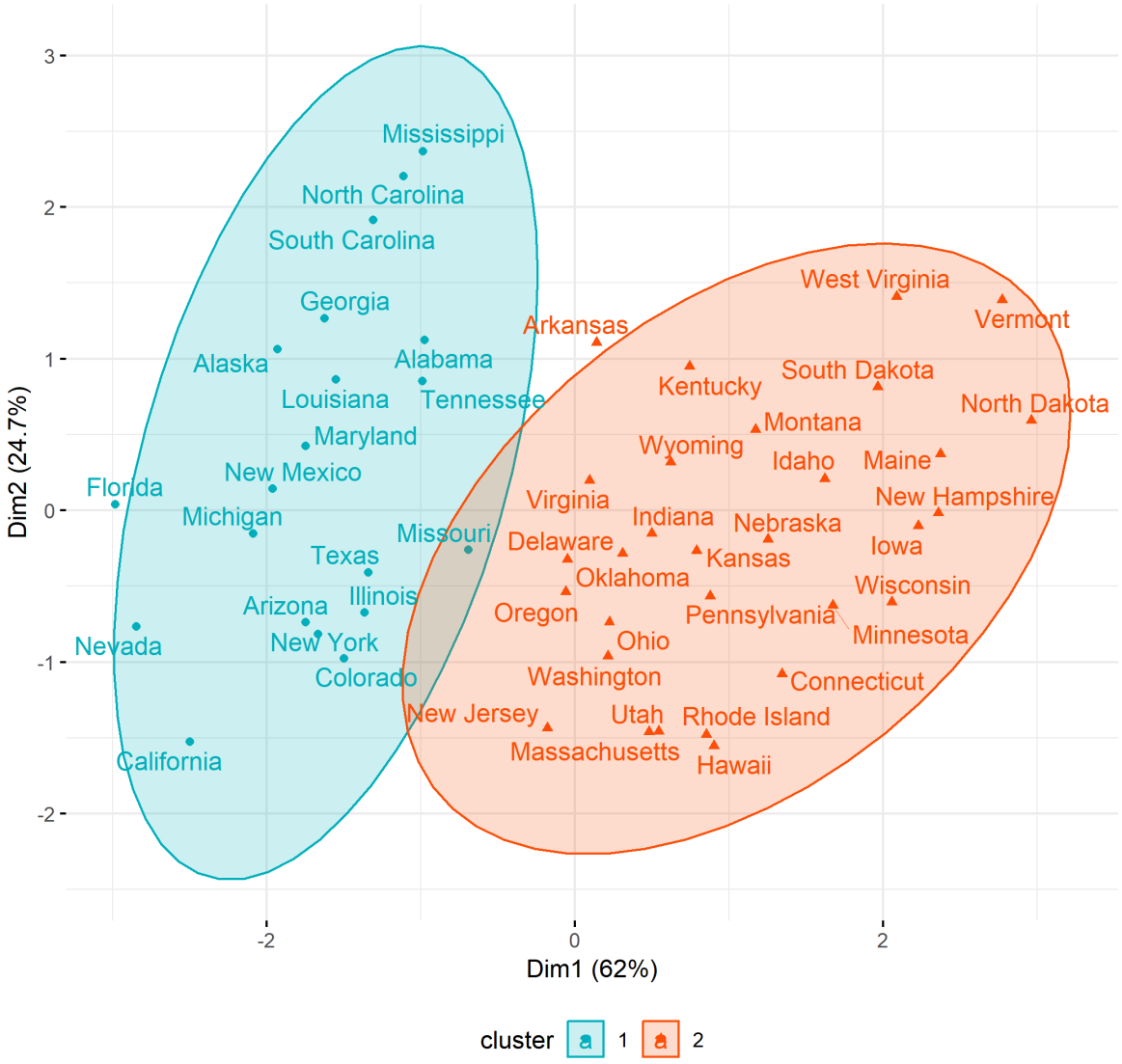
Podem observar en la sortida el següent:

- Els grups medoids: una matriu, on les files són els medoids i les columnes són les variables.
- El vector de particions: un vector d´enters (de 1:k) que indica el clúster on cada observació ha sigut agrupada.

Igual com hem fet en l´exercici anterior utilitzarem la funció `fviz_cluster()` del paquet `factoextra` per a visualitzar les particions:

```
# Visualitzem els clústers
fviz_cluster(pamFit,
    main = "Gràfic de clústers",
    palette = c("#00AFBB", "#FC4E07"),
    ellipse.type = "t",
    repel = TRUE,
    ggtheme = theme_minimal()) +
  theme(legend.position = "bottom",
      plot.title = element_text(color="#4F81BD", size=16, face="bold")
)
```

Gràfic de clústers



El mètode aglomeradors *AGNES*

En el mètode de *k-means* sempre es comença per un nombre fix de grups coneguts *a priori*. Això fa que sigui útil quan tenim alguna idea de quants grups hi ha en realitat.

Ara bé, hi ha situacions en què el nostre desconeixement del domini d'aplicació és encara més gran. Per tant, ni tan sols és possible fixar una quantitat *k* de punts inicials entorn dels quals anar formant els grups.

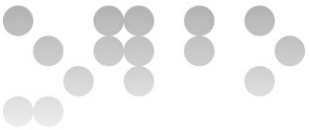
En aquest cas, cal reflectir aquest desconeixement adoptant una actitud neutra respecte a les dades. Una de les maneres de resoldre el problema és mitjançant els *mètodes aglomeradors*.

Els **mètodes aglomeradors** (en àngles, *Agglomerative Nesting*) comencen considerant que cada objecte forma un grup per si mateix (un grup d'un sol element) i llavors avaluen les distàncies entre grups (o objectes, en el primer pas) i creen per aglomeració els diversos grups finals.

Mesures de similitud

Amb la finalitat de decidir quins objectes han de ser agrupats i quins clústers dividits, hem de fer ús de les mesures de similitud entre els elements.

Tal com s´ha estudiat en l'apartat 3 dels apunts de l´assignatura existeixen diferents mètodes per a calcular la di(similitud), com per exemple les distancies Euclidià i *Hamming*.



Podem calcular les distancies entre cada par d´elements amb l´ajuda de la funció `dist():

```
# Calcula la matriu de disimilitud
matrixDist <- dist(df, method = "euclidean")
```

El resultat de la línia de codi anterior és la matriu de distàncies o dissimilituds. Per defecte, la funció `dist()` calcula la distància Euclidià, no obstant podem indicar altres mètriques passant-les a l'argument `method`.

Càlcul de l´arbre jeràrquic

Així doncs, donada una matriu de distàncies calculada amb la funció `dist()`, podem crear un arbre jeràrquic amb l´ajuda de la funció `hclust()`:

```
hcFit <- hclust(d = matrixDist, method = "ward.D2")
```

Cal fer una especial referència, als diferents tipus de mètodes aglomeratius. Tot seguit, es mostren els més habituals:

- Maximum o *complete linkage*: La distància entre dos clústers es defineix com el màxim valor de totes les distàncies de cada par d´elements entre els elements del clúster 1 i els elements del clúster 2. Aquest mètode tendeix a produir particions compactes.

- Minium o *single linkage*: La distància entre dos clústers es defineix com el mínim valor de totes les distàncies de cada par d´elements entre els elements del clúster 1 i els elements del clúster 2. Aquest mètode tendeix a produir particions molt grans.

- Mean o *average linkage*: La distància entre dos clústers es defineix com la distància mitjana entre els elements de la partició 1 i els elements de la partició 2.

- *Centroid linkage*: La distància entre dos clústers es defineix com la distància entre el centroide de la particio 1 i el centroide de la partició 2.

- *Ward´s minium variance method*: Aquest mètode minimitza la variància intra-clústers.

Convé destacar que, es recomanable utilitzar el mètode *Ward´s* o *complete linkage*.

Tallar el dendograma en diferents grups

Un dels problemes amb el clustering jeràrquic és que com no proporcionem el nombre de clústers *a priori*, no sabem on tallar l´arbre per a formar clústers.

Podem fer ús de la funció `cutree()` per a tallar un arbre passant el nombre de particions o l´altura de l´arbre:

```
# Talla l´argre en 4 groups
grp <- cutree(hcFit, k = 4)
head(grp, n = 4)

##  Alabama   Alaska  Arizona Arkansas
##        1        2        2        3
```

Podem conèixer el nombre d´elements en cada partició com es mostra a continuació:

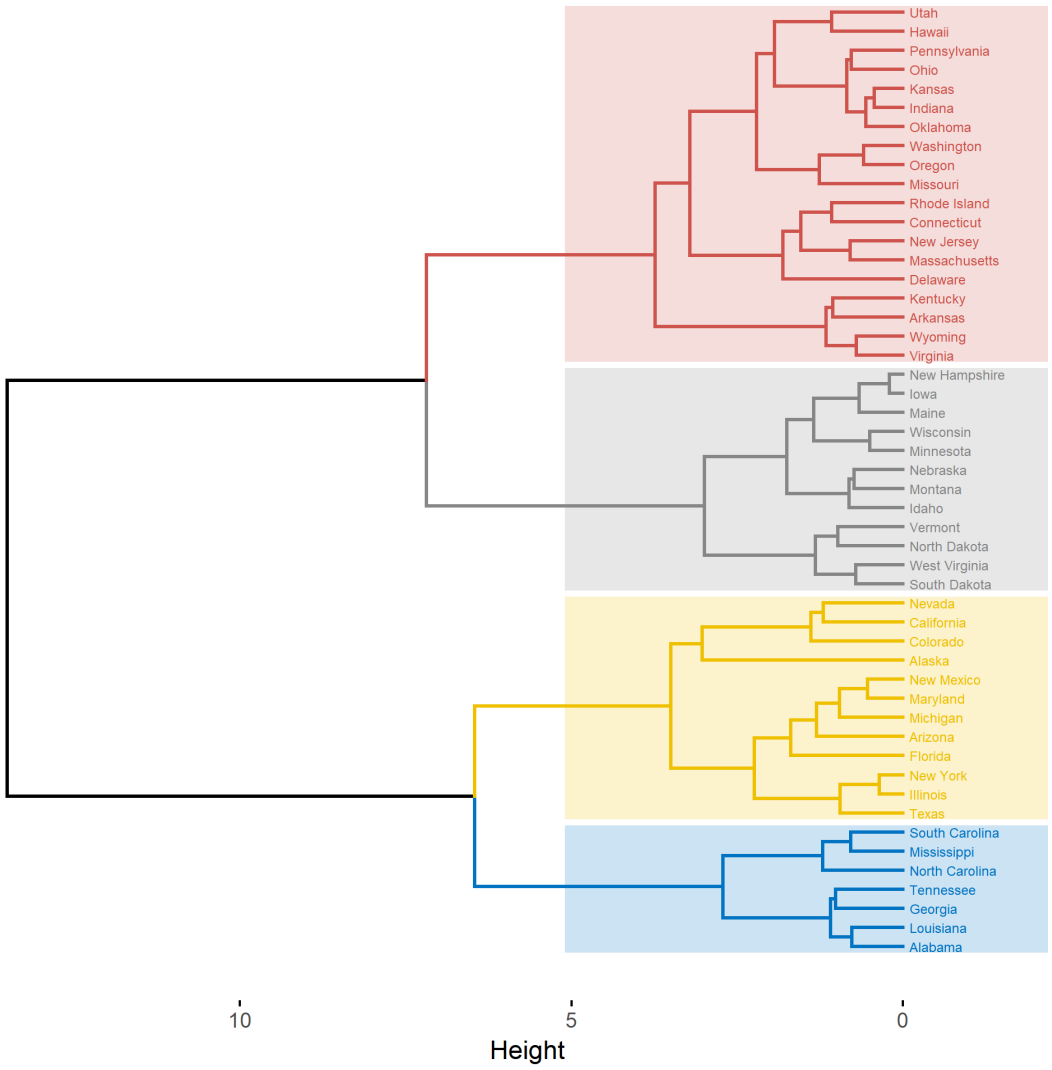
```
# Nombre de elements en cada partició
table(grp)
```

```
## grp
##  1  2  3  4
##  7 12 19 12
```

A continuació, es mostra la representació gràfica de l´arbre jeràrquic. Aquest gràfic es coneix com a dendrograma:

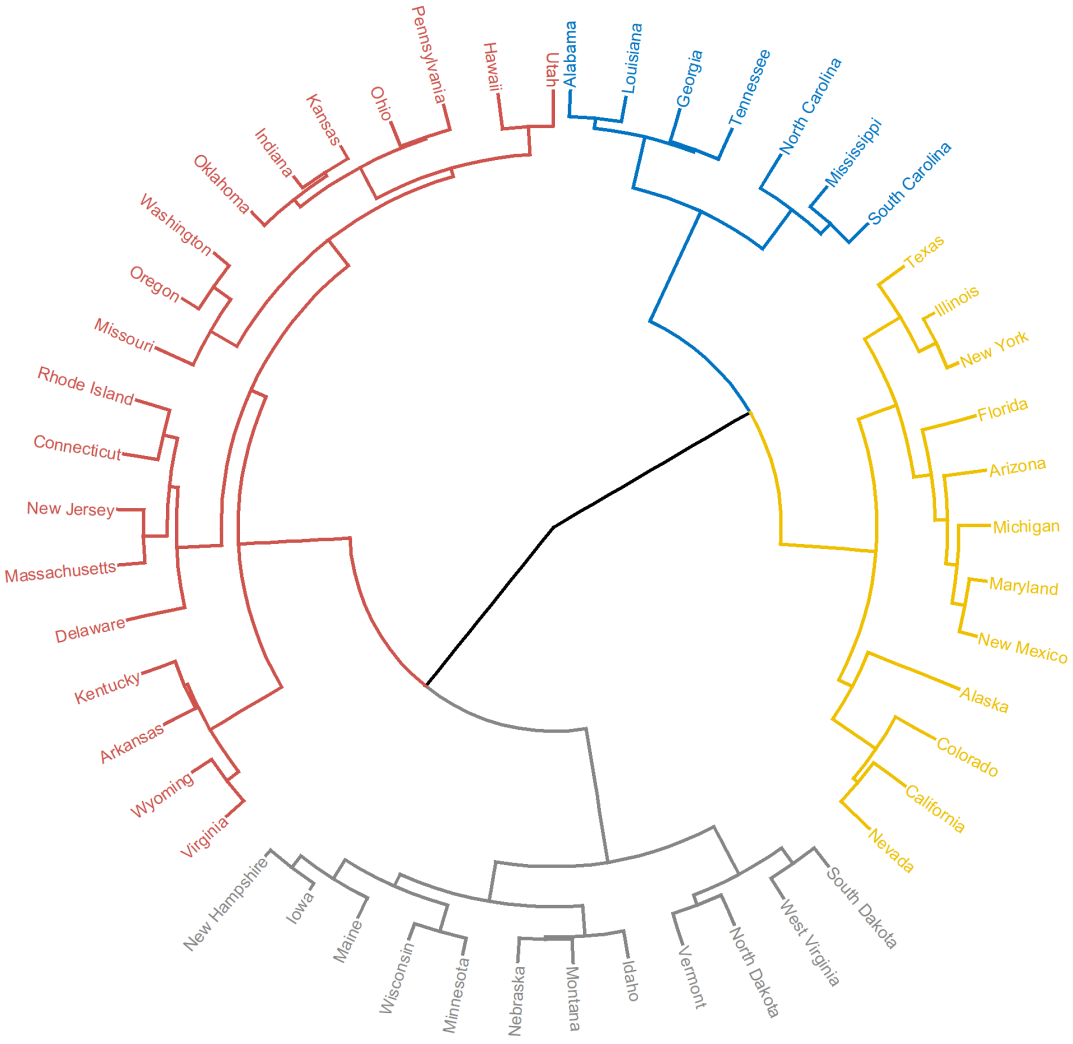
```
# Talla en 4 groups i acoloreix per grups
fviz_dend(hcFit, k = 4,
  main = "Dendograma de clústers",
  horiz = TRUE,
  cex = 0.4,
  k_colors = "jco",
  color_labels_by_k = TRUE,
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE) +
  theme(legend.position = "bottom",
      plot.title = element_text(color="#4F81BD", size=16, face="bold")
)
```

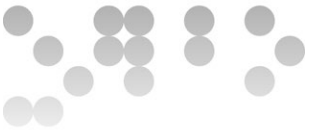
Dendograma de clústers



També, podem obtindre un dendrograma circular mitjançant la opció `type = "circular"`:

```
fviz_dend(hcFit, cex = 0.5, k = 4,
          k_colors = "jco", type = "circular")
```

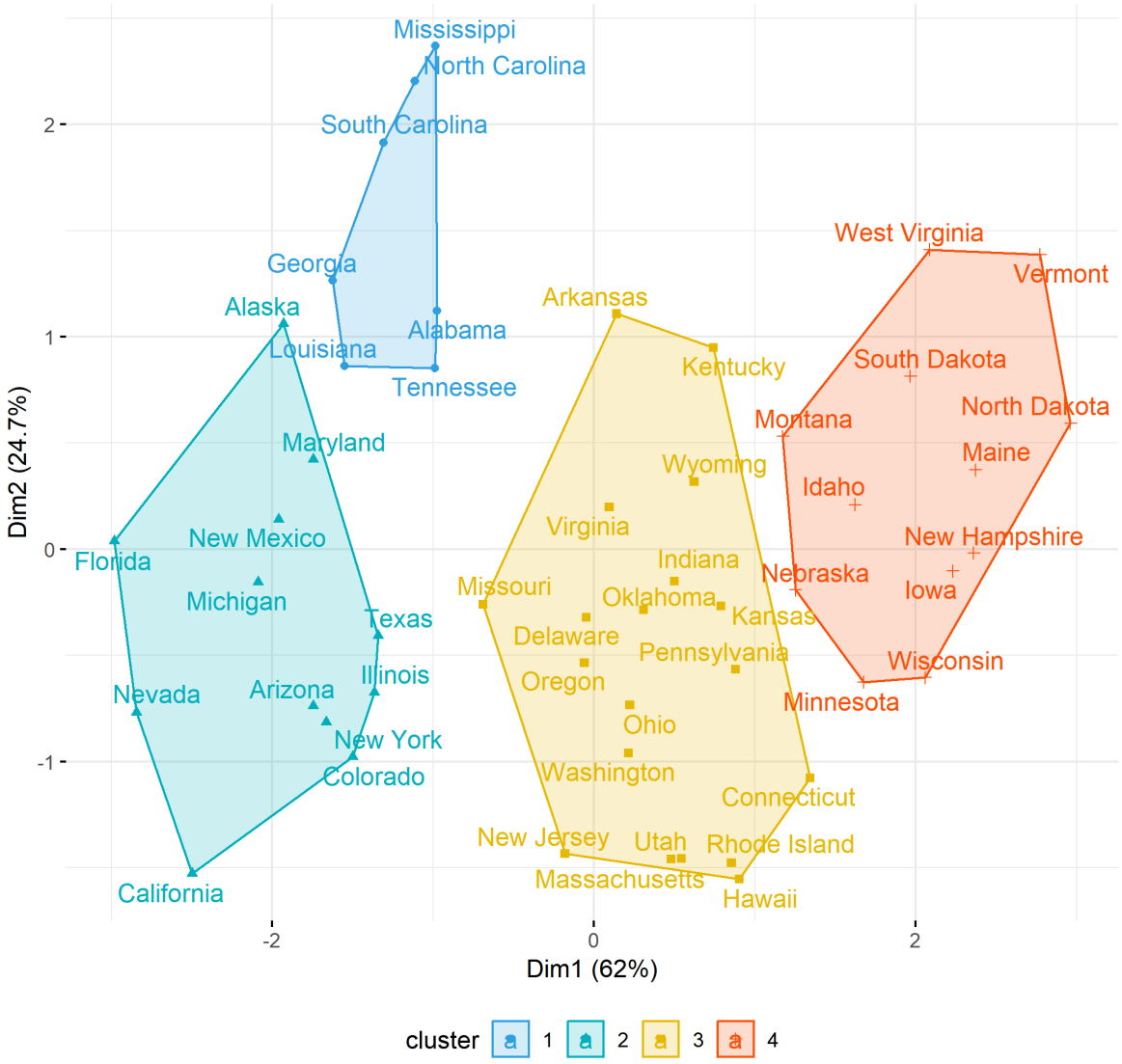

Pennsylvania
Hawaii
Utah
Alabama
Louisiana
Georgia
Tennessee
North Carolina
Mississippi
South Carolina
Ohio
Kansas
Indiana
Oklahoma
Washington
Oregon
Missouri
Rhode Island
Connecticut
New Jersey
Massachusetts
Delaware
Kentucky
Arkansas
Wyoming
Virginia
Texas
Illinois
New York
Florida
Arizona
Michigan
Maryland
New Mexico
Alaska
Colorado
California
Nevada
New Hampshire
Iowa
Maine
Wisconsin
Minnesota
Nebraska
Montana
Idaho
Vermont
North Dakota
West Virginia
South Dakota



Amb l´ajuda de la funció `fviz_cluster()`, podem visualitzar el resultat en un diagrama de dispersió. Els elements són representats mitjançant punts, a més s´utilitza PCA:

```
# Diagrama de clústers
fviz_cluster(list(data = df, cluster = grp),
  main = "Gràfic de clústers",
  palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
  ellipse.type = "convex",
  repel = TRUE,
  show.clust.cent = FALSE, ggtheme = theme_minimal()) +
 theme(legend.position = "bottom",
    plot.title = element_text(color="#4F81BD", size=16, face="bold"))
```

Gràfic de clústers



Comparació dels mètodes d´agrupació

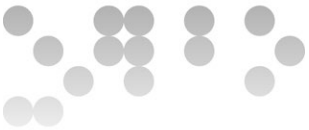
Recollint tot el que s´ha dit, farem ús de la funció `clValid()` del paquet `clValid` per a comparar els diferents mètodes d´agrupació.

El prototip de la funció és el següent:

```
clValid(obj, nClust, clMethods = "hierarchical",
validation = "stability", maxitems = 600,
metric = "euclidean", method = "average")
```

on els arguments són:

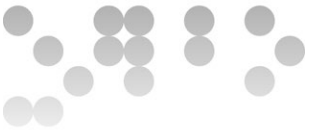
- **obj**: Una matriu o data frame numèric. Les files son els elements que s´han d´agrupar i les columnes són les observacions.
- **nClust**: Un vector numèric especificant el nombre de particions que s´han d´avaluar.
- **validation**: El tipus de mètrica de validació. Possibles valors són "internal", "stability", and "biological. També, permet escollir varies mètriques a la vegada.
- **maxitems**: El nombre màxim d´elements (files en la matriu) que han de ser agrupades.
- **metric**: La mètrica utilitzada per a determinar la matriu de distàncies. Possibles valors són "euclidean", "correlation", i "manhattan".
- **method**: Per a agrupacions jeràrquiques, el mètode d´aglomeració a utilitzar. Les opcions disponibles són "ward", "single", "complete" i "average".



Per exemple, considerant el conjunt de dades `USArrests` que hem estat utilitzant per a il·lustrar els diferents mètodes d´aglomeració, podem utilitzar la funció `clValid()` com es mostra tot seguit per a calcular les diferents mètriques internes:

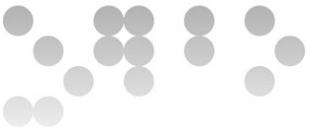
```
# comparació dels mètodes d´agrupació
clmethods <- c("hierarchical", "kmeans", "pam")
intern <- clValid(df, nClust = 2:6,
          clMethods = clmethods, validation = "internal")
# Resúm
summary(intern)

##
## Clustering Methods:
##  hierarchical kmeans pam
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5
6
##
## hierarchical Connectivity   6.6437  9.5615 13.9563 22.5782 31.2
873
##              Dunn           0.2214  0.2214  0.2224  0.2046  0.2
126
##              Silhouette     0.4085  0.3486  0.3637  0.3213  0.2
720
## kmeans       Connectivity   6.6437 13.6484 16.2413 24.6639 33.7
194
##              Dunn           0.2214  0.2224  0.2224  0.1983  0.2
231
##              Silhouette     0.4085  0.3668  0.3573  0.3377  0.3
079
## pam          Connectivity   6.6437 13.8302 20.4421 29.5726 38.2
643
##              Dunn           0.2214  0.1376  0.1849  0.1849  0.2
019
##              Silhouette     0.4085  0.3144  0.3390  0.3105  0.2
```



```
630
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 6.6437 hierarchical 2
## Dunn         0.2231 kmeans       6
## Silhouette   0.4085 hierarchical 2
```

Com es pot observar el métode jerárquic amb dos clústers obté valors òptims en les métriques *Connectivity* i *Silhouette* . Per altra banda, el mètode k-means obté una bona puntuació en la mètrica *Dunn* amb un valor òptim de $k = 6$.



Bibliografia

[1] Daniel T. Larouse, Chantal D. Larouse: Data Mininig and Predictive Analytics.USA, John Wiley & Sons,2015,ISBN 978-1-118-11619-7

[2] Jordi Gironés Roig, Jordi Casas Roma, Julià Minguillón Alfonso, Ramon Caihuelas Quiles : Minería de Datos: Modelos y Algoritmos. Barcelona, Editorial UOC, 2017, ISBN: 978-84-9116-904-8.

[3] Jiawe Han, Michellie Chamber & Jian Pei: Data mining : concepts and techniques. 3º Edition. USA, Editorial Elsevier, 2012, ISBN 978-0-12-381479-1