Algoritmi non gerarchici: K-means e PAM

Esistono molti algoritmi di clustering diversi. Questo laboratorio si concentra sul metodo delle K medie o K-means.

Algoritmo K-Means

La funzione `kmeans()` implementa il metodo delle K medie in R, prendendo come input i dati e il numero di clusters desiderato.

L'esempio che segue utilizza i dati in `food_spending.csv`, riguardanti il numero medio (mensile) di pasti consumati fuori casa (ad es., al ristorante) e la spesa per i pasti consumati a casa (annuale) per un insieme di 100 famiglie.

```
dat<-read.csv("food_spending.csv", header=TRUE)
head(dat)
```

```
##    FreqAway      Home
## 1  6.951280 10965.524
## 2  7.415451  9854.342
## 3 12.462907  6824.662
## 4 14.136932  8982.304
## 5  5.969204  9276.950
## 6  7.976161 10131.322
```

Per eseguire l'algoritmo K-means standardizziamo prima i dati e poniamo $K = 2$ nella funzione `kmeans()` (la funzione `set.seed()` viene utilizzata per garantire che il risultato sia riproducibile).

```
x.scaled<-scale(dat)
set.seed(4)
km.out <- kmeans(x.scaled, 2)
km.out   # see the output
```

```
## K-means clustering with 2 clusters of sizes 43, 57
##
## Cluster means:
##     FreqAway       Home
## 1 -1.1045122  0.9206786
## 2  0.8332285 -0.6945470
##
## Clustering vector:
##   [1] 1 1 2 2 1 1 1 1 2 2 1 1 2 2 1 2 2 1 2 1 1 1 2 2 1 2 2 1 2 2 2 2 1 1 2 1 2
##  [38] 2 1 1 1 1 2 1 1 1 2 2 2 1 2 2 2 1 1 1 1 2 2 2 2 1 1 1 2 1 2 1 1 2 2 2 1 2
##  [75] 1 2 2 1 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2
##
## Within cluster sum of squares by cluster:
## [1] 21.96630 20.05714
##  (between_SS / total_SS =  78.8 %)
##
## Available components:
##
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

L'output stampato mostra: - i vettori delle medie di gruppo (Cluster means) - il vettore del clustering, cioè un vettore di numeri interi (da 1 a K) che indica il cluster di appartenenza di ciascuna unità - la devianza *within* per ciascun gruppo

$$WSS_k = \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

($\bar{x}_{kj}$ è la media della j-esima variabile nel gruppo k) e il rapporto "devianza TRA i gruppi"/"devianza TOT" (BSS/DEV_{TOT}), che rappresenta la percentuale di varianza spiegata tramite il clustering

$$\frac{BSS}{BSS + WSS} = \frac{BSS}{DEV_{TOT}} = \frac{\sum_{k=1}^{K} \sum_{j=1}^{p} |C_k| (\bar{x}_{kj} - \bar{x}_j)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2}$$

dove $WSS = \sum_{k=1}^{K} WSS_k$.

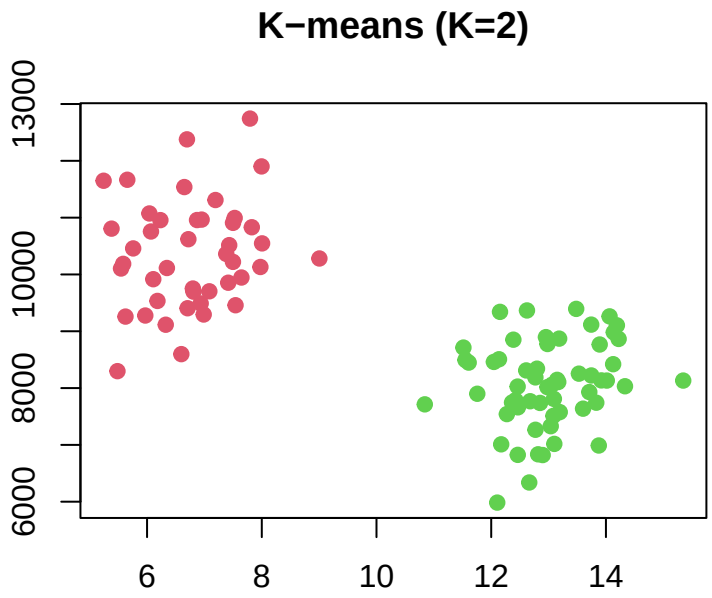
La partizione finale è contenuta nel vettore `km.out$cluster`:

```
str(km.out$cluster)
```

```
##  int [1:100] 1 1 2 2 1 1 1 1 2 2 ...
```

I due cluster ottenuti contengono rispettivamente 56 e 44 osservazioni, le unità sono raggruppate in due cluster ben separati, come ci si aspettava. I due gruppi possono essere riportati con colori diversi nel grafico di dispersione:

```
plot(dat, col =(km.out$cluster+1), main="K-means (K=2)", xlab ="", ylab="",pch=19)
```

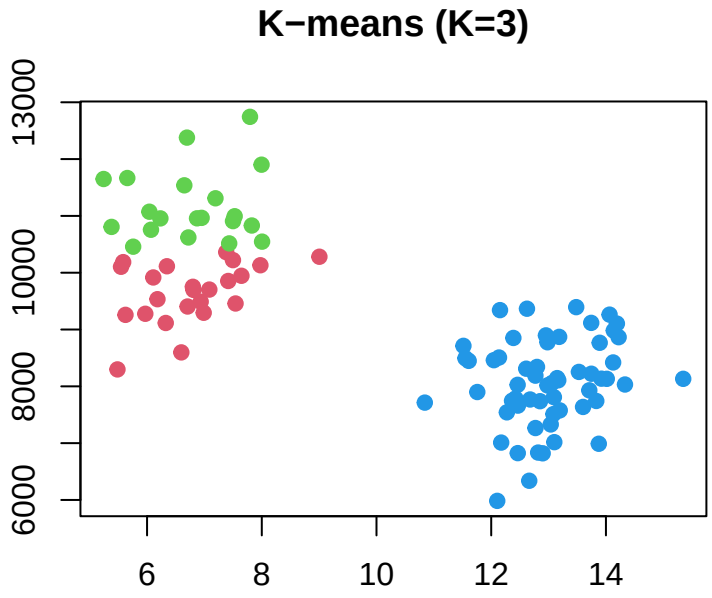


In questo esempio con dati simulati sapevamo che in realtà c'erano due cluster ben distinti. Tuttavia, supponendo di non conoscere il numero di gruppi, avremmo potuto eseguire l'algoritmo con $K = 3$

```r
set.seed(5)
km.out<-kmeans(x.scaled, 3)
km.out
```

```
## K-means clustering with 3 clusters of sizes 23, 20, 57
##
## Cluster means:
##     FreqAway       Home
## 1 -1.1065155  0.4232302
## 2 -1.1022083  1.4927443
## 3  0.8332285 -0.6945470
##
## Clustering vector:
##   [1] 2 1 3 3 1 1 1 1 3 3 2 2 3 3 2 3 3 1 3 1 2 1 3 3 2 3 3 2 3 3 3 3 1 1 3 2 3
##  [38] 3 1 1 2 1 3 2 1 1 3 3 3 2 3 3 3 1 1 1 2 3 3 3 3 2 1 2 3 1 3 2 2 3 3 3 1 3
##  [75] 1 3 3 2 3 3 3 3 3 2 3 3 3 3 3 2 3 3 3 3 1 3 3 2 3 3
##
## Within cluster sum of squares by cluster:
## [1]  4.671572  5.057885 20.057140
##  (between_SS / total_SS =  85.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
plot(dat, col =(km.out$cluster+1), main="K-means (K=3)", xlab ="", ylab="", pch=19)
```



La funzione `kmeans()` consente di eseguire l'assegnazione iniziale un certo numero di volte e scegliere la configurazione iniziale migliore attraverso l'argomento "nstart'' che può essere incrementato arbitrariamente:

```
set.seed(6)
km.out<-kmeans(dat, 3, nstart=1)
km.out$tot.withinss
```

```
## [1] 31535467
```

```
km.out<-kmeans(dat, 3, nstart=20)
km.out$tot.withinss
```

```
## [1] 31385541
```

Esercizio *Si considerino i dati nel file "sanita.txt" che si riferiscono ad un rilevamento ISTAT sui servizi sanitari nelle regioni italiane (i dati sono del 1998). I dati sono utilizzati in Rapallo & Rogantin 'Statistica descrittiva multivariata' (2003). Le variabili su cui effettuare l'analisi sono: il numero di medici per 10.000 abitanti; il numero di posti letto ospedalieri per 10.000 abitanti; il numero di pediatri per 10.000 abitanti di età inferiore ai 14 anni; il numero annuo di interventi di pronto soccorso per 1.000 abitanti; la percentuale di persone che si sono dichiarate soddisfatte dell'assistenza medica ospedaliera. Si importino i dati con il seguente comando*

```
sanita<-read.table("sanita.txt", header =TRUE, row.names=1)
```

Si effettui una analisi di raggruppamento utilizzando il metodo delle K-medie.

Partitioning Around Medoids (PAM)

L'obiettivo comune degli algoritmi K-means e PAM è di determinare una partizione ammissibile che sia ottimale rispetto alla omogeneità interna dei cluster ed alla eterogeneità tra i gruppi.

Nell'algoritmo *Partitining Around Medoids* (PAM) si adoperano le distanze tra le unità. I K centroidi diventano K medoidi, che sono K delle unità tra le n considerate.

La funzione `pam()` del pacchetto **cluster** implementa l'algoritmo PAM prendendo la matrice dei dati oppure la matrice di dissimilarità in input, e consentendo l'eventuale standardizzazione preliminare dei dati attraverso l'argomento `stand = TRUE` (il valore di default è FALSE).

Se l'argomento è la matrice dei dati la distanza utilizzata di default è la distanza euclidea (la documentazione fornisce le alternative possibili):

```
library(cluster)
pam.out<-pam(dat, 2, metric="euclidean", stand=TRUE)
pam.out
```

```
## Medoids:
##      ID  FreqAway      Home
## [1,] 42  7.380646 10362.114
## [2,] 53 13.059628  8058.363
## Clustering vector:
##   [1] 1 1 2 2 1 1 1 1 2 2 1 1 2 2 1 2 2 1 2 1 1 1 2 2 1 2 2 1 2 2 2 2 1 1 2 1 2
##  [38] 2 1 1 1 1 2 1 1 1 2 2 2 1 2 2 2 1 1 1 1 2 2 2 2 1 1 1 2 1 2 1 1 2 2 2 1 2
##  [75] 1 2 2 1 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2
## Objective function:
##     build      swap
## 0.8438882 0.6608854
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

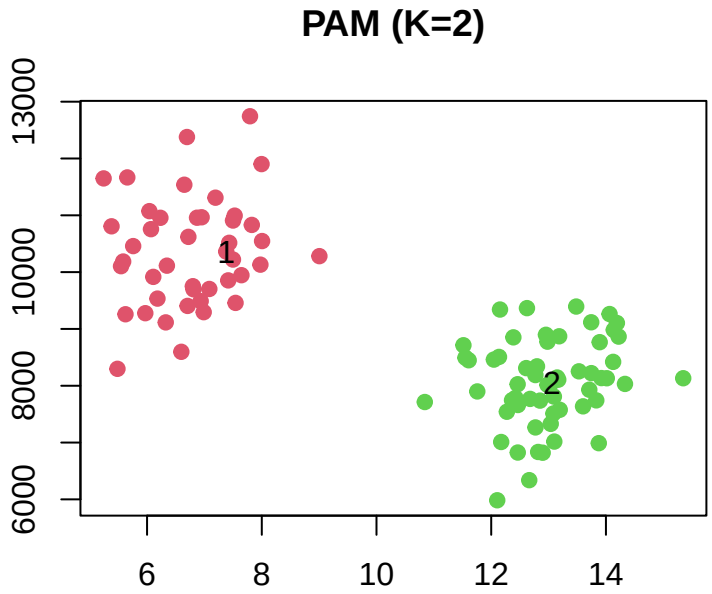
I medoidi e il vettore del clustering si ottengono come segue:

```
pam.out$medoids
```

```
##       FreqAway      Home
## [1,]  7.380646 10362.114
## [2,] 13.059628  8058.363
```

```
#pam.out$clustering
```

```
plot(dat, col =(pam.out$cluster+1), main="PAM (K=2)", xlab ="", ylab="", pch=19)
points(pam.out$medoids, pch=as.character(pam.out$cluster[pam.out$id.med]))
```



Nota Per dataset di dimensioni elevate si veda la funzione `clara()`, che risulta essere computazionalmente più efficiente.

Scelta del numero di gruppi

La scelta del numero di gruppi negli algoritmi di cluster analysis è un problema cruciale per questo tipo di analisi e non sempre risulta semplice individuare una soluzione. Negli algoritmi non gerarchici, il numero di gruppi è stabilito dal principio e pertanto occorre una strategia per la validazione della procedura di clustering adottata.

Vengono descritti e implementati di seguito due degli indici *interni* più utilizzati per determinare il numero ottimale di gruppi: l'indice **silhouette** e l'**indice di Calinski e Harabasz**. Si veda il pacchetto di R **fpc** per molti altri indici comunemente utilizzati.

L'indice *silhouette* (Rousseeuw, 1987) si basa sul grado di coesione dell'unità su cui viene calcolata con il guppo di appartenenza e sulla sua separazione con gli altri gruppi. Tale indice viene calcolato per ogni unità statistica attraverso la formula

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

dove $a(i)$ è la distanza media dall'osservazione i da tutte le unità appartenenti al medesimo gruppo; $b(i)$ è la più piccola tra le distanze medie dell'i-esima osservazione con le unità degli altri gruppi.

Si noti che $s(i) \in [-1, 1]$: un $s(i)$ vicino a 1 significa che l'unità i è ben classificata, un $s(i)$ vicino a 0 implica che l'i-esima osservazione è in una situazione borderline tra due gruppi, infine un $s(i)$ vicino a -1 implica che l'i-esima osservazione è classificata nel gruppo sbagliato.

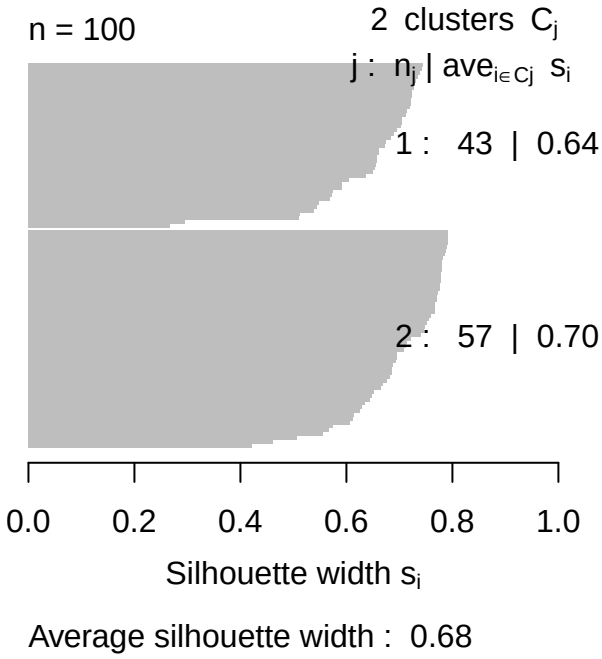
La *silhouette media* (ASW) S_K è data da

$$S_K = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

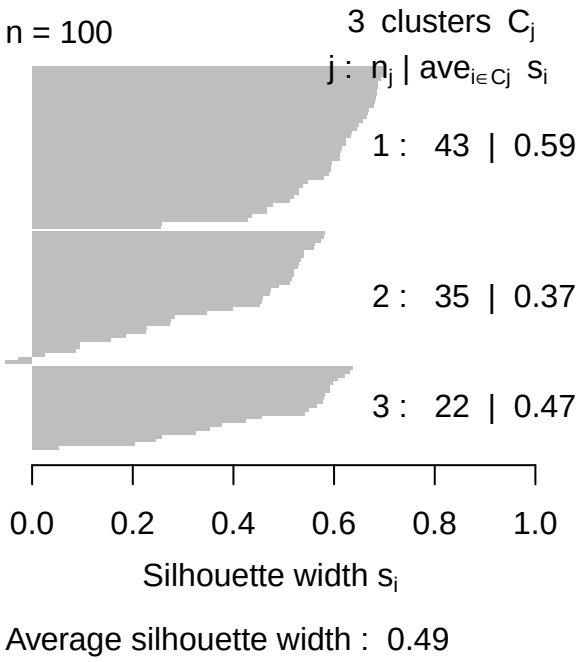
dove K è il numero di gruppi della partizione. Ottenendo la *silhouette media* al variare di K, la scelta di K dovrà ricadere sul valore che massimizza S_K.

L'output di `pam` consente di ottenere facilmente l'indice ASW:

```
plot(pam.out, which=2, main="")  #silhouette plot
```




```
plot(pam(dat, 3, metric="euclidean", stand=TRUE), which=2, main="")
```



L'indice di *Calinski e Harabasz* (Calinski and Harabasz, 1974) valuta la qualità della partizione ottenuta in termini di rapporto tra la varianza entro i gruppi e la varianza tra i gruppi. Per una data partizione è definito da

$$CH_K = \frac{BSS}{WSS}\frac{n-K}{K-1}$$

dove n e K sono rispettivamente il numero totale di osservazioni e il numero di cluster. Il numero di gruppi K che risulta ottimale secondo tale indice 'è quello che lo massimizza in quanto al numeratore compare un indicatore di separazione tra i gruppi e al denominatore un un indicatore di coesione.

```
require(fpc)
```

```
## Caricamento del pacchetto richiesto: fpc
```

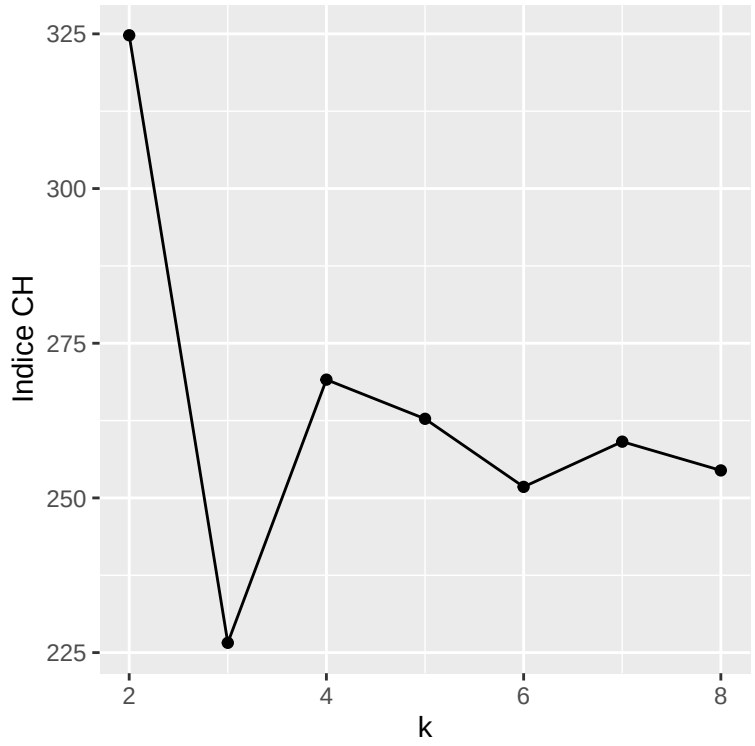
```
require(ggplot2)
```

```
## Caricamento del pacchetto richiesto: ggplot2
```

```
Diss<-pam(dat, 2, stand = TRUE, keep.diss = TRUE)$diss
minC <- 2
maxC <- 8
res <- sapply(minC:maxC,
       function(nc, data)
        cluster.stats(Diss,
                      pam(data, nc, stand = TRUE, cluster.only=TRUE))$ch, data = dat)

ggp <- ggplot(data=data.frame(x=2:(length(res)+1), y=res),
              mapping = aes(x=x,y=y)) +
  geom_point() +
  geom_line() +
```

```
  xlab("k") +
  ylab("Indice CH")
print(ggp)
```



Esercizio *La funzione* `pam` *consente di utilizzare una matrice di dissimilarità arbitraria impostando* `diss=TRUE`*. Si implementi il metodo PAM per effettuare l'analisi di clustering dei dati in "claims.csv", che riguardano le caratteristiche di 1000 sinistri. Si ricordi che il dataset contiene tre variabili binarie e una variabile numerica. Si utilizzi una opportuna matrice di dissimilarità.*