Clustering

Lab – R – 12/11/2020

K-means: Usando o Pacote R

```
> set.seed(2)
> x=matrix(rnorm(50*2), ncol=2)
> x[1:25,1]=x[1:25,1]+3
> x[1:25,2]=x[1:25,2]-4
```

Fornece uma semente para gerar realizações de uma Variável aleatória Normal Padrão.

Gera 50 pares ordenados, cujas componentes são realizações de uma Variável aleatória Normal Padrão .

matrix(...,...) gera uma matriz com 2 colunas e com os pares ordenados em cada coluna.

x[1:25,1] se refere à coluna 1. Somando 3 em cada realização temos que a média da normal fica deslocada para 3.
x[1:25,2] se refere à coluna 2. Somando -4 em cada realização temos que média da normal fica deslocada para -4.

```
set.seed (2)
x=matrix (rnorm (50*2) ,
ncol =2)
> x
            [,1]        [,2]
 [1,] -0.896914547 -0.838287148
 [2,]  0.184849185  2.066301356
 [3,]  1.587845331 -0.562247053
 [4,] -1.130375674  1.275715512
 [5,] -0.080251757 -1.047572627
 [6,]  0.132420284 -1.965878241
 [7,]  0.707954729 -0.322971094
 [8,] -0.239698024  0.935862527
 [9,]  1.984473937  1.139229803
[10,] -0.138787012  1.671618767
[11,]  0.417650751 -1.788242207
[12,]  0.981752777  2.031242519
[13,] -0.392695356 -0.703144333
[14,] -1.039668977  0.158164763
[15,]  1.782228960  0.506234797
[16,] -2.311069085 -0.819995106
[17,]  0.878604581 -1.998846995
[18,]  0.035806718 -0.479292591
[19,]  1.012828692  0.084179904
[20,]  0.432265155 -0.895486611
[21,]  2.090819205 -0.921275666
[22,] -1.199925820  0.330449503
[23,]  1.589638200 -0.141660809
[24,]  1.954651642  0.434847762
[25,]  0.004937777 -0.053722626
[26,] -2.451706388 -0.907110376
[27,]  0.477237303  1.303512232
[28,] -0.596558169  0.771789776
[29,]  0.792203270  1.052525595
[30,]  0.289636710 -1.410038341
[31,]  0.738938604  0.995984590
[32,]  0.318960401 -1.695764903
[33,]  1.076164354 -0.533372143
[34,] -0.284157720 -1.372269451
[35,] -0.776675274 -2.207919779
[36,] -0.595660499  1.822122519
[37,] -1.725979779 -0.653393411
[38,] -0.902584480 -0.284681219
[39,] -0.559061915 -0.386949604
[40,] -0.246512567  0.386694975
[41,] -0.383586228  1.600390852
[42,] -1.959103175  1.681154956
[43,] -0.841705060 -1.183606388
[44,]  1.903547467 -1.358457254
[45,]  0.622493930 -1.512670795
[46,]  1.990920436 -1.253104899
[47,] -0.305483725  1.959357077
[48,] -0.090844235  0.007645872
[49,] -0.184161452 -0.842615198
[50,] -1.198767765 -0.601160105
```

Exercício 1: Gere uma outra matriz x, na sequência, sem usar set.seed(2). As matrizes são iguais? Agora coloque set.seed(2) novamente e compare com a matriz ao lado.

We now perform K-means clustering with $K = 2$.

```
> km.out=kmeans(x,2,nstart=20)
```

nstart é o número de configurações iniciais que serão testadas. Lembram da aula?

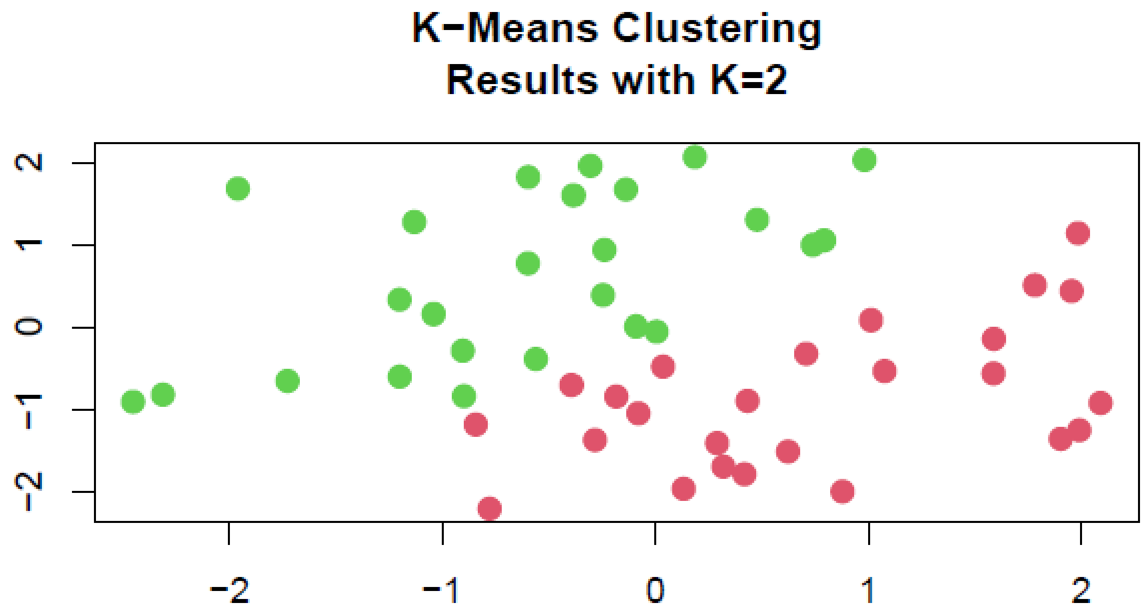
Clustering vector nstart=1, 20 e 50.

2 2 1 2 1 1 1 2 1 2 1 2 1 2 1 2 1 1 1 1 1 2 1 1 2 2 2 2 2 1 2 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2 1 2

2 1 2 1 2 2 2 1 1 1 2 1 2 1 1 2 2 2 2 2 2 1 2 1 2 2 1 1 1 2 1 2 2 2 2 1 2 2 2 1 1 1 2 2 2 2 1 2 2 2

2 1 2 1 2 2 2 1 1 1 2 1 2 1 1 2 2 2 2 2 2 1 2 1 2 2 1 1 1 2 1 2 2 2 2 1 2 2 2 1 1 1 2 2 2 2 1 2 2 2

```
> plot(x, col=(km.out$cluster+1), main="K-Means Clustering
    Results with K=2", xlab="", ylab="", pch=20, cex=2)
```



E se tivéssemos proposto K=3?

Clustering vector: (1)

3 1 2 1 3 2 2 1 1 1 2 1 3 3 2 3 2 3 2 2 2 3 2 2 3 3 1 1 1 2 1 2 2 3 3 1 3 3 3 1 1 1 3 2 2 2 1 3 3 3

Clustering vector: (20)

2 3 1 3 2 2 1 3 1 3 2 3 2 2 1 2 2 2 1 2 1 3 1 1 2 2 3 3 3 2 3 2 1 2 2 3 2 2 2 3 3 3 2 1 2 1 3 2 2 2

Clustering vector: (50)

1 3 2 3 1 1 2 3 2 3 1 3 1 1 2 1 1 1 2 1 2 3 2 2 1 1 3 3 3 1 3 1 2 1 1 3 1 1 1 3 3 3 1 2 1 2 3 1 1 1

Clustering vector: (100)

1 2 3 2 1 1 3 2 3 2 1 2 1 1 3 1 1 1 3 1 3 2 3 3 1 1 2 2 2 1 2 1 3 1 1 2 1 1 1 2 2 2 1 3 1 3 2 1 1 1

Clustering vector: (1000)

2 1 3 1 2 2 3 1 3 1 2 1 2 2 3 2 2 2 3 2 3 1 3 3 2 2 1 1 1 2 1 2 3 2 2 1 2 2 2 1 1 1 2 3 2 3 1 2 2 2

Exercício: **reproduzir esses dados e gerar os gráficos ("plotar"). O que você pode dizer com relação à estabilização da classificação das observações? Compare com os resultados para K=2.**

Analisando as demais informações fornecidas pela função k-means

->**km.out =kmeans (x,2)**
> km.out
K-means clustering with 2 clusters of sizes 25, 25

Cluster means: (as coordenadas finais dos centróides de cada cluster)

```
       [,1]       [,2]
1  0.7299706 -0.8812778
2 -0.5916948  0.6202094
```

Clustering vector:
2 2 1 2 1 1 1 2 1 2 1 2 1 2 1 2 1 1 1 1 2 1 1 2 2 2 2 2 1 2 1 1 1 1 2 2 2 2 2 2 2 1 1 1 1 2 2 1 2

Within cluster sum of squares by cluster: W(C_1) e W(C_2)
[1] 37.64374 44.60166

https://uc-r.github.io/kmeans_clustering#kmeans

Recall that, the basic idea behind cluster partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation (known as **total within-cluster variation or total within-cluster sum of square**) is minimized:

tot.withiness

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}.$$

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}.$$

- **totss:** The total sum of squares.
- **withinss:** Vector of within-cluster sum of squares, one component per cluster.
- **tot.withinss:** Total within-cluster sum of squares, i.e. sum(withinss).
- **betweenss:** The between-cluster sum of squares, i.e. $totss-tot.withinss$.
- **size:** The number of points in each cluster.

O comando "str(km.out)" gera:

List of 9
 $ **cluster**   : int [1:50] 1 2 2 1 1 1 2 2 2 2 ...
 $ **centers**   : num [1:2, 1:2] -0.543 0.914 -0.656 0.596
  ..- attr(*, "dimnames")=List of 2
  .. ..$: chr [1:2] "1" "2"
  .. ..$: NULL
 $ **totss**   : num 132
 $ **withinss**   : num [1:2] 46.3 41
 $ **tot.withinss:** num 87.3
 $ **betweenss**   : num 44.9
 $ **size**   : int [1:2] 29 21
 $ **iter**   : int 1
 $ **ifault**   : int 0
 - attr(*, "class")= chr "kmeans"