# K-means method for clustering

## K-means Examples

STAT 32950-24620

Spring 2025 (wk7)



# K-means method for clustering

For a given $K$ – number of desired clusters, the optimal clustering minimizes the **total within-cluster sums of squares**:

$$\sum_{i=1}^{K}\sum_{j=1}^{n_i} d^2_{i_j,c(i)} = \sum_{i=1}^{K}\sum_{j=1}^{n_i}(x_{i_j} - c(i))^2$$

where

$K$ the total number of clusters (usually preassigned in the K-means method)

$n_i$ the number of members in cluster $i$ (varies at each step of the algorithm)

$c(i)$ is the centroid of cluster $i$ (varies at each step of the algorithm)

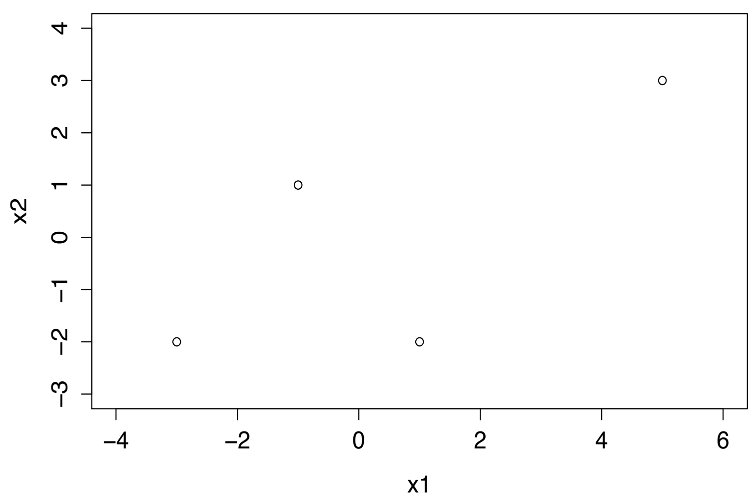
$x_{i_j}$ the $j$th member of cluster $i$ (varies at each step of the algorithm)

$d_{i_j,c(i)}$ the distance between $x_{i_j}$ and $c(i)$ (Euclidean distance most common)



# Example 1 Consider various initialization and clustering methods

4 points in $\mathbb{R}^2$, to form 2 clusters



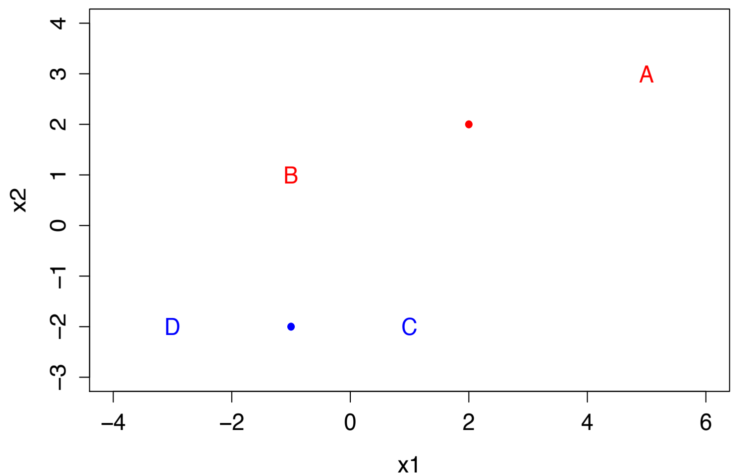



Start with a random assignment:

- Either, assign K cluster centroids randomly:
  - Assign all objects to the cluster of nearest centroid
  - Recalculate the cluster centroids
  - Repeat
- Or, assign cluster membership randomly:
  - Calculate the cluster centroids
  - Re-assign all objects to the cluster of nearest centroid
  - Repeat

Ex1: Start with randomly assigned membership (vs start with centroids)

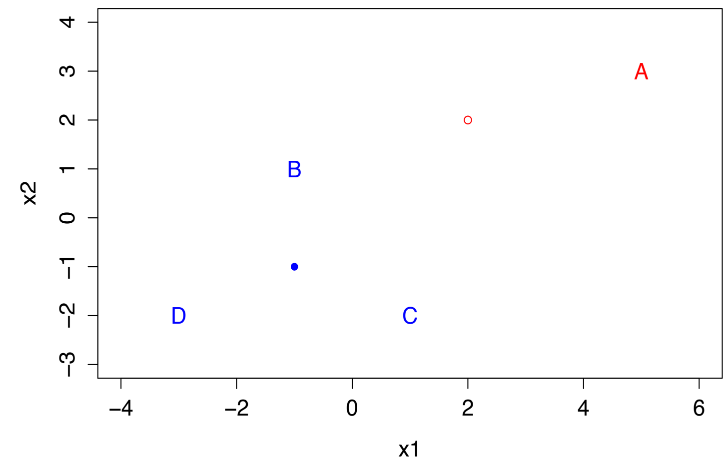
Start with clusters (AB) and (CD).
Next: Calculate cluster centroids.






"MacQueen" method (earlier days)

Re-assign element to nearest centroids.
Reassign A, C, D - rejected.
Reassigning B - accepted. (Early MacQueen: Update affected cluster centroids right away.)



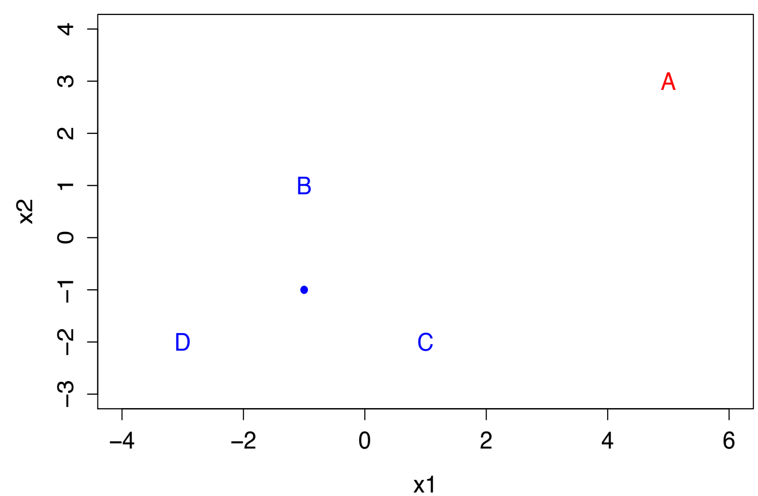



With the new clusters,

- Re-calculate the new cluster centroids.
- Check the distance of each items to the centroids
  (Early Floyd: update centroids after checking all items)
- Re-assign items to the cluster with the nearest centroid.
- Repeat.
- Stop when no appreciable improvement of total within-cluster sum of squares.



Final clusters: (A) and (BCD)



All possible combinations of $k = 2$ clusters in the example:

(A) (BCD)
(B) (ACD)
(C) (ABD)
(D) (ABC)

(AB) (CD)
(AC) (BD)
(AD) (BC)

Of all 2-clusters, the partition {(A), (BCD)} minimizes the total within-cluster sums of squares

$$\sum_{i=1}^{2}\sum_{j=1}^{2} d^2_{i_j,c(i)} = \sum_{i=1}^{2}\sum_{j=1}^{2}(x_{i_j} - c(i))^2$$

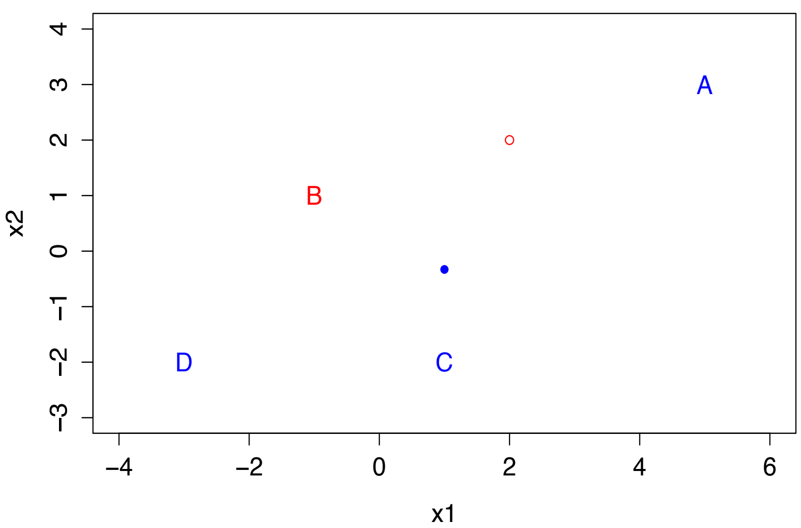


## The kmeans function in R

```
help(kmeans)

kmeans {stats}
K-Means Clustering

Description

Perform k-means clustering on a data matrix.

Usage

kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace=FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)

Arguments
...

Value

kmeans returns an object of class "kmeans" which has a print and a fitted method.
It is a list with at least the following components:

cluster

centers

...
```



Discussion: Different algorithms. (Lloyd, vs MacQueen vs. Hartigan-Wong: min ESS update)





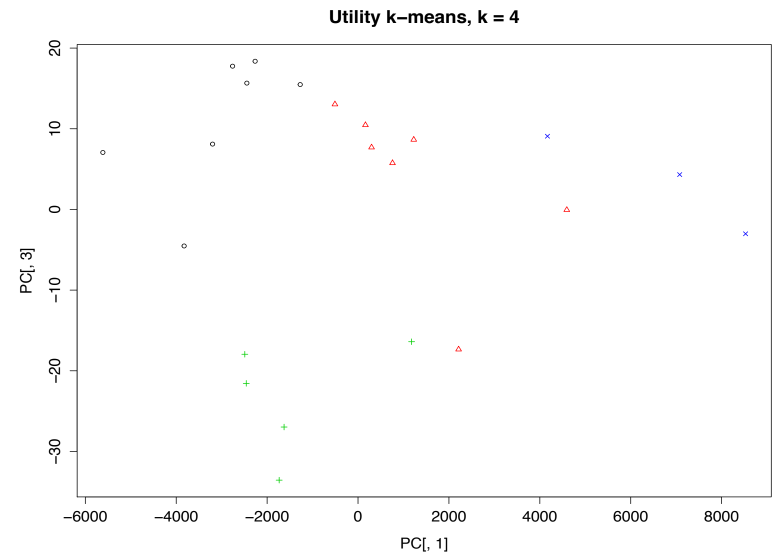

## Using the kmeans function in R for Example 1

```
x1=c(5,-1,1,-3); x2=c(3,1,-2,-2)
M0 = kmeans(cbind(x1,x2),2)

M0$cluster
[1] 1 2 2 2   # (A), (BCD)
M0$size
[1] 1 3

M0$center
  x1 x2
1  5  3
2 -1 -1

M0$withinss
[1]  0 14
M0$tot.withinss
[1] 14
M0$betweenss
[1] 39
M0$totss
[1] 53

dist(M0$center)
         1
2 7.211103


Available components:

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

## Example 2: K-means method applied to the utility data: First, let $K = 4$

```
data = read.table("T12-4.dat"); X = data[,1:8]
NormX = as.matrix(X)%*%solve(diag(sqrt(diag(var(X)))))
Mkm = kmeans(NormX,4)

Mkm$cluster
[1] 3 4 3 1 4 3 4 2 3 1 2 4 1 3 4 2 4 3 3 1 4 1
Mkm$size
[1] 5 3 7 7

print(Mkm$center, digits=3)
  [,1] [,2] [,3] [,4]  [,5] [,6]  [,7] [,8]
1 6.03 5.12 4.30 12.4 1.206 2.11 2.280 1.39
2 5.44 3.95 5.42 12.3 2.031 4.37 0.000 1.02
3 6.54 5.56 3.10 12.4 0.550 2.86 0.191 1.57
4 5.80 4.13 4.34 13.6 0.985 1.65 0.426 3.23

Mkm$withinss
[1] 10.177094  9.533522 26.507769 34.164812
Mkm$betweenss
[1] 87.6168

print(dist(Mkm$center),digits=3)
     1    2    3
2 3.75
3 2.70 3.75
4 3.08 4.07 3.15
```








## Ex2: K-means method applied to the utility data, now $K = 5$

```
Mkm5 = kmeans(NormX,5)

Mkm5$cluster
[1] 4 3 4 2 1 4 3 5 4 2 5 3 2 4 3 5 3 4 4 2 3 2
Mkm5$size
[1] 1 5 6 7 3

print(Mkm5$center, digits=3)
  [,1] [,2] [,3] [,4]  [,5] [,6]  [,7] [,8]
1 8.08 3.92 4.66 11.5 0.321 0.93 0.929 3.68
2 6.03 5.12 4.30 12.4 1.206 2.11 2.280 1.39
3 5.42 4.16 4.28 13.9 1.096 1.77 0.342 3.16
4 6.54 5.56 3.10 12.4 0.550 2.86 0.191 1.57
5 5.44 3.95 5.42 12.3 2.031 4.37 0.000 1.02

Mkm5$withinss
[1]  0.000000 10.177094 21.187976 26.507769  9.533522
Mkm5$betweenss
[1] 100.5936

print(dist(Mkm5$center),digits=3)
1    2    3    4
2 3.98
3 3.89 3.25
4 4.14 2.70 3.30
5 5.56 3.75 4.04 3.75
```



## The effect of initial assignments

Ex2: K-means method applied to the utility data with $K = 4$ again.

```
> Mkm = kmeans(NormX,4)
> Mkm

K-means clustering with 4 clusters of sizes 1, 6, 3, 12

Cluster means:
      [,1]     [,2]     [,3]     [,4]      [,5]     [,6]      [,7]     [,8]
1 4.118992 2.851987 3.301664 13.87535 2.8862340 1.609585 0.4942854 3.452628
2 6.079126 4.337397 4.511465 13.52044 0.6681097 1.657378 0.4148821 3.196079
3 5.437792 3.951191 5.421850 12.29131 2.0310536 4.367531 0.0000000 1.017207
4 6.327530 5.377184 3.599057 12.42580 0.8231112 2.548814 1.0615225 1.495390

Clustering vector:
 [1] 4 2 4 4 2 4 2 3 4 4 3 2 4 4 2 3 1 4 4 4 2 4

Within cluster sum of squares by cluster:
[1]  0.000000 23.336839  9.533522 58.012322
 (between_SS / total_SS =  45.9 %)
```

## The effect of initial assignments (cont.)

Ex2: And again, K-means method applied to the utility data with $K = 4$.

```
> Mkm = kmeans(NormX,4)
> Mkm

K-means clustering with 4 clusters of sizes 11, 4, 4, 3

Cluster means:
      [,1]     [,2]     [,3]     [,4]      [,5]     [,6]      [,7]     [,8]
1 6.242539 5.339374 3.555468 12.32459 0.6880315 2.565261 1.1580245 1.490417
2 4.945500 3.665249 4.284880 13.61757 1.1705282 1.684515 0.5136399 3.142881
3 7.018545 5.002118 4.327365 13.51670 1.1304417 1.795924 0.2322546 2.901916
4 5.437792 3.951191 5.421850 12.29131 2.0310536 4.367531 0.0000000 1.017207

Clustering vector:
 [1] 1 2 1 1 3 1 3 4 3 1 4 3 1 1 2 4 2 1 1 1 2 1

Within cluster sum of squares by cluster:
[1] 51.590389 12.574083 18.083154  9.533522
 (between_SS / total_SS =  45.4 %)
```



## The effect of initial assignments (cont.)

Ex2: One more time, K-means method applied to the utility data with $K = 4$.

```
> Mkm = kmeans(NormX,4)

 Mkm
K-means clustering with 4 clusters of sizes 5, 6, 1, 10

Cluster means:
      [,1]     [,2]     [,3]     [,4]      [,5]      [,6]      [,7]     [,8]
1 6.557869 5.810924 2.786993 12.26142 0.2052433 3.0230559 0.2679860 1.665534
2 5.419726 4.159148 4.284880 13.92018 1.0956999 1.7707122 0.3424266 3.158915
3 8.075392 3.921482 4.661173 11.47687 0.3206927 0.9295816 0.9290183 3.675611
4 5.945439 4.732516 4.551927 12.46764 1.4944278 2.8573086 1.1398339 1.266863

Clustering vector:
 [1] 4 2 1 4 3 1 2 4 4 4 4 2 4 1 2 4 2 1 1 4 2 4

Within cluster sum of squares by cluster:
[1] 15.15613 21.18798  0.00000 57.53424
 (between_SS / total_SS =  44.1 %)
```



## The effect of initial assignments (cont.) - the worst?

Ex2: K-means method applied to the utility data with $K = 4$. The worst case?

```
> kmeans(NormX,4)


K-means clustering with 4 clusters of sizes 5, 4, 4, 9

Cluster means:
      [,1]     [,2]     [,3]     [,4]      [,5]     [,6]      [,7]     [,8]
1 6.557869 5.810924 2.786993 12.26142 0.2052433 3.023056 0.2679860 1.665534
2 7.018545 5.002118 4.327365 13.51670 1.1304417 1.795924 0.2322546 2.901916
3 4.945500 3.665249 4.284880 13.61757 1.1705282 1.684515 0.5136399 3.142881
4 5.799107 4.614674 4.604527 12.34859 1.4039212 2.911688 1.2664821 1.235394

Clustering vector:
 [1] 4 3 1 4 2 1 2 4 2 4 4 2 4 1 3 4 3 1 1 4 3 4

Within cluster sum of squares by cluster:
[1] 15.15613 18.08315 12.57408 50.29653
 (between_SS / total_SS =  42.8 %)
```



## The effect of initial assignments (cont.) - the best?

Ex2: K-means method applied to the utility data with $K = 4$. The best case?

```
> kmeans(NormX,4)


K-means clustering with 4 clusters of sizes 7, 5, 3, 7

Cluster means:
      [,1]     [,2]     [,3]     [,4]      [,5]     [,6]      [,7]     [,8]
1 6.542384 5.563921 3.097044 12.43434 0.5497589 2.862870 0.1914186 1.572436
2 6.026735 5.115752 4.301874 12.41385 1.2058044 2.109136 2.2796679 1.387525
3 5.437792 3.951191 5.421850 12.29131 2.0310536 4.367531 0.0000000 1.017207
4 5.799107 4.125196 4.338636 13.57114 0.9849846 1.650551 0.4262254 3.232729

Clustering vector:
 [1] 1 4 1 2 4 1 4 3 1 2 3 4 2 1 4 3 4 1 1 2 4 2

Within cluster sum of squares by cluster:
[1] 26.507769 10.177094  9.533522 34.164812
 (between_SS / total_SS =  52.2 %)
```

Remarks: There is always some output, even if the results are not optimal.
Example: `kmeans(distance-matrix, k)`