Data Clustering using R

Aparajita Khan[1] Avisek Gupta[2]

Junior Research Fellow,
[1]Machine Intelligence Unit,
[2]Electronics and Communication Sciences Unit,
Indian Statistical Institute.

Workshop on Big Data and R
Institute of Engineering and Management, Kolkata
February 4, 2017

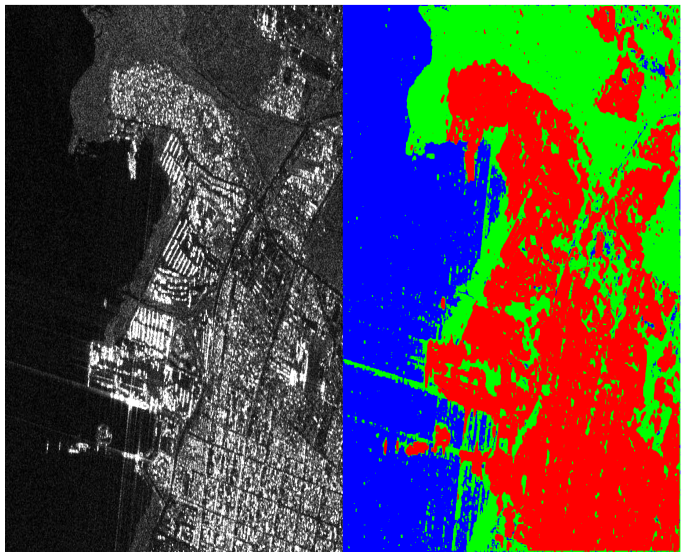
Data Clustering

- Objective - group together similar data.

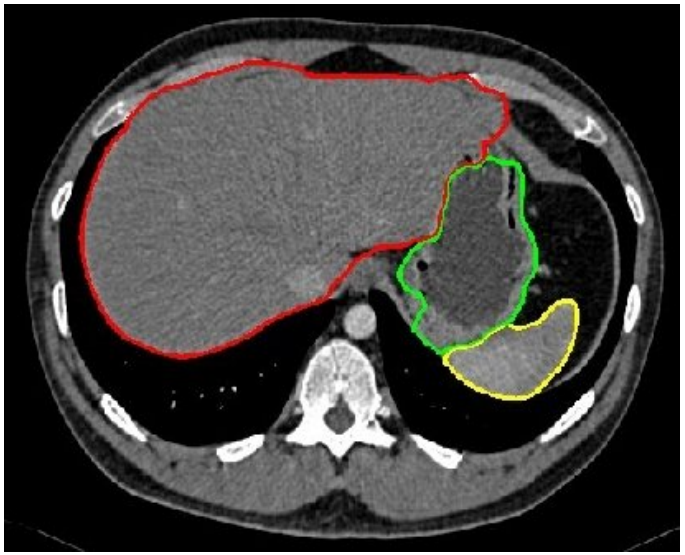
$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \\ 3 & 3 & 3 \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \\ 3 & 3 & 3 \end{bmatrix}$$

Applications

- Image Segmentation



(a) Remote Sensing



(b) Biomedical: Brain MRI

- Text Mining : Document clustering [Scientific, Sports, Political]

Applications II

- Network Clustering

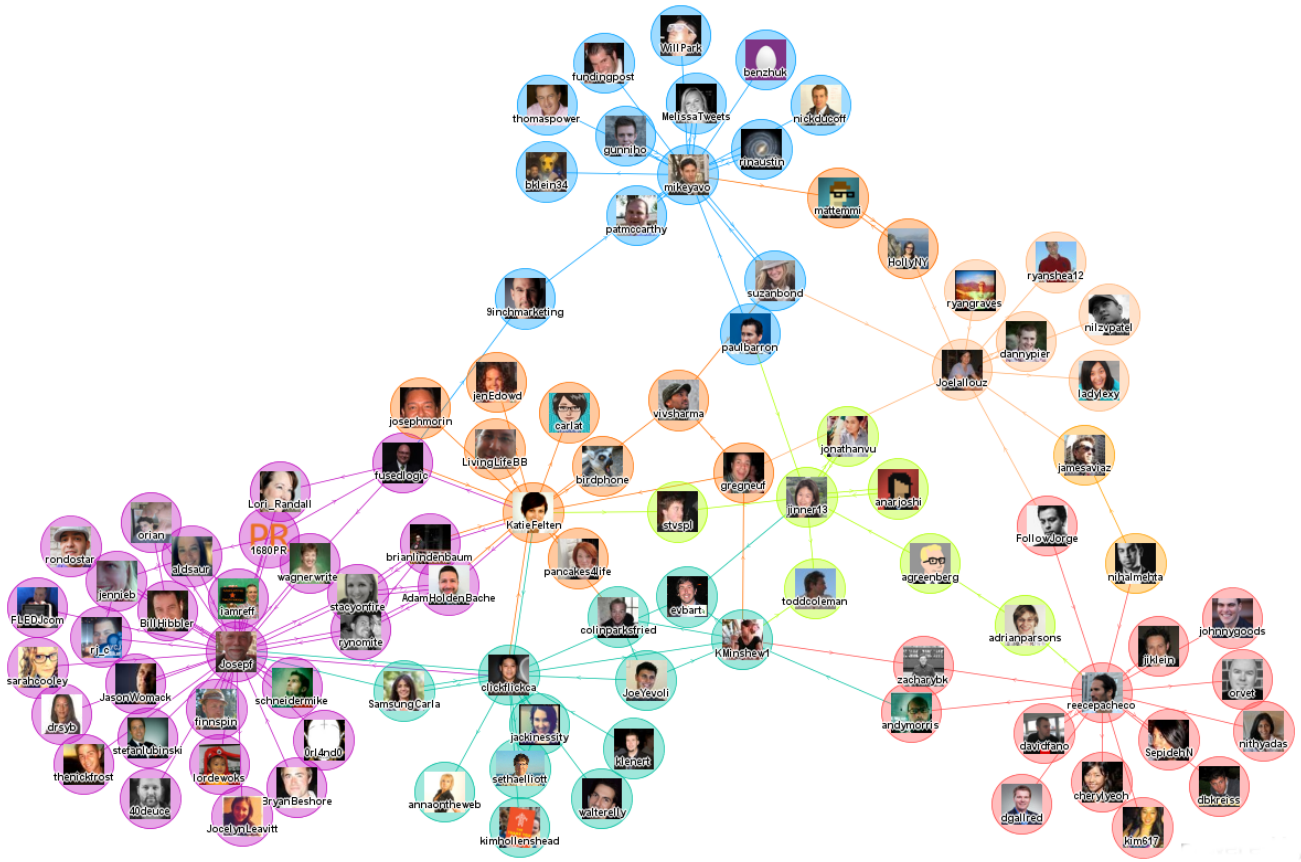


Figure: Social Networks : Community Detection

Applications III

- Bioinformatics

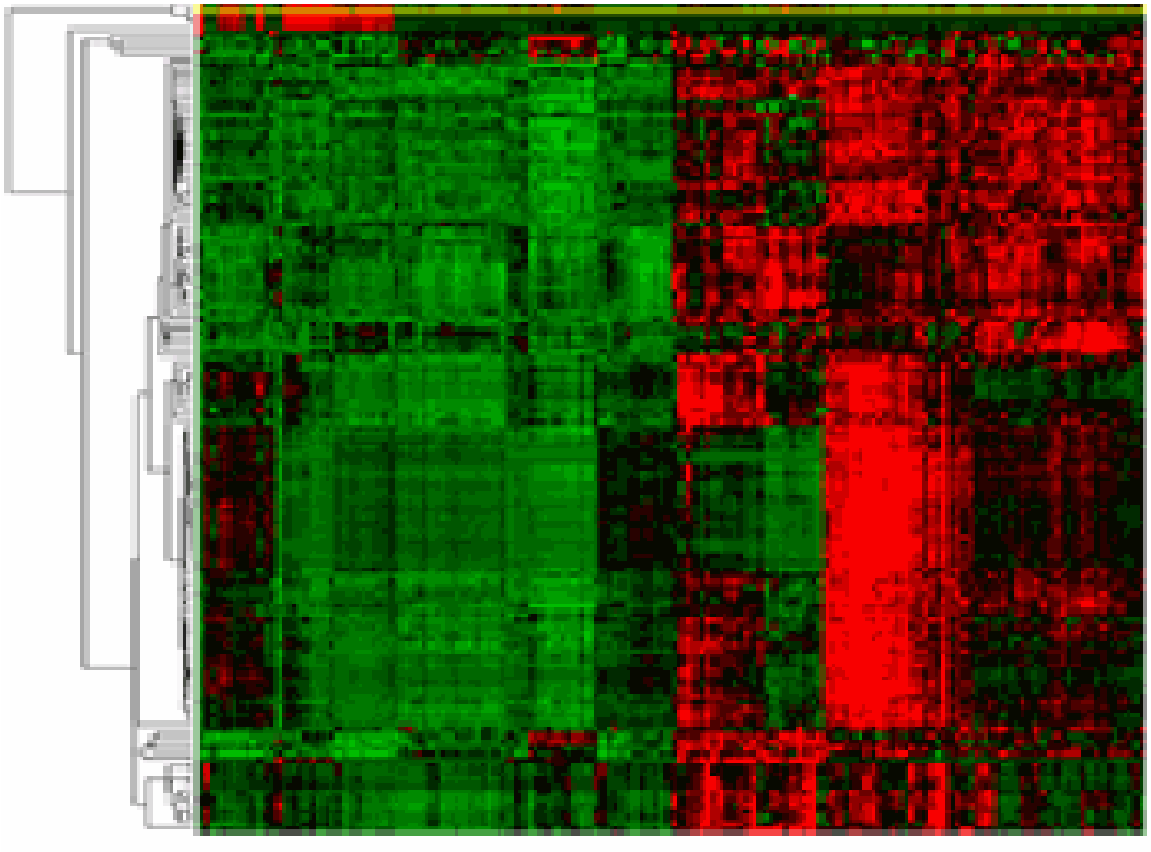


Figure: Grouping similar genes

Data Clustering

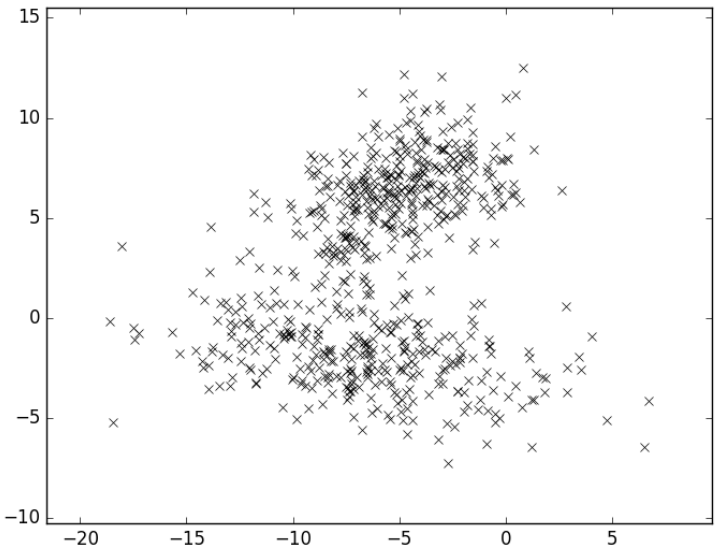
- Data Matrix X of dimension $(n \times d)$.
- Each row is a data point x_i of dimension d.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{bmatrix}$$

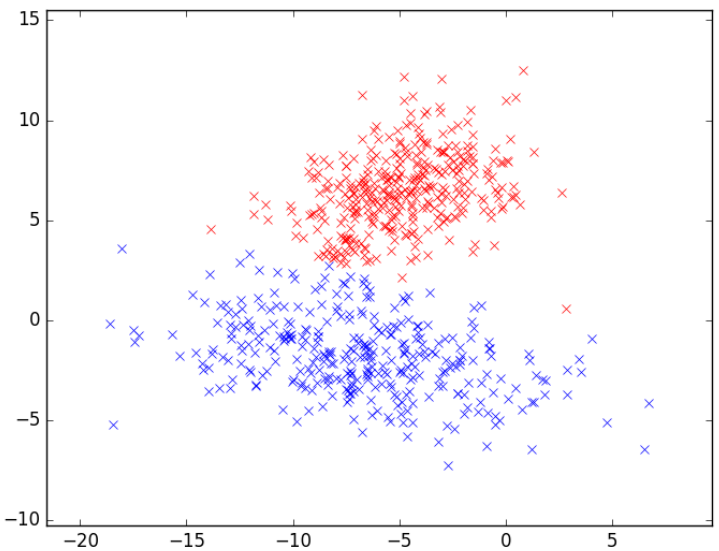
- Objective - Assign n data points to K clusters.

Data Clustering

- Input : n data points : $\{x_i \in \Re^d \ : \ i = 1, ..., n\}$
  Output : K Clusters : $C_1, C_2, ..., C_K$



(a) Original Data



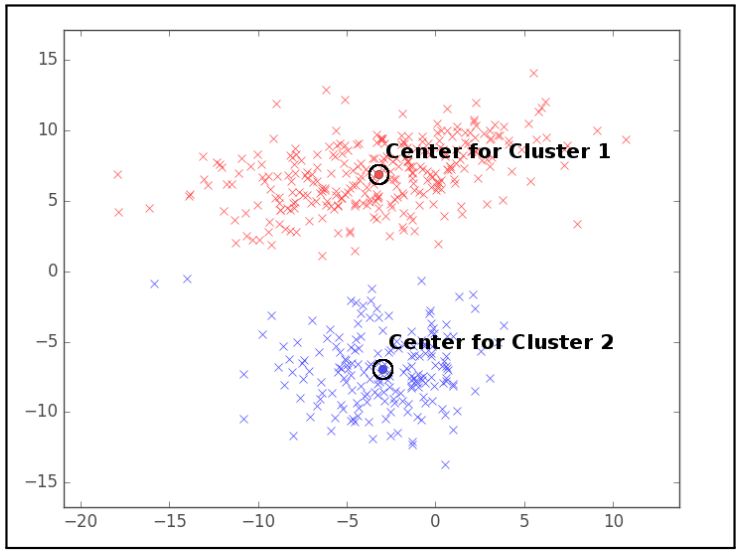
(b) Clustered Data (K=2)

Conents

- Hard Partitional Clustering - K-means
- Soft Clustering - Fuzzy c-Means Clustering
- Hierarchical Clustering
- Clustering for Big Data -
  - Large-scale datasets
  - High dimensional datasets

K-Means Clustering

- Cluster centers m_j represent each cluster C_j, $j = 1, .., K$.



- Distance Metric : Squared Euclidean Distance

$$dist(x_i, m_j) = ||x_i - m_j||^2$$

- **K-Means Cost Function :**

$$Cost_{KM} = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - m_j||^2$$

K-Means Clustering : Defining Centers

- The cost for each cluster is,

$$\sum_{x_i \in C_j} ||x_i - m_j||^2$$

- Finding m_j that minimizes the cost,

$$\frac{\partial}{\partial m_j}\left(\sum_{x_i \in C_j} ||x_i - m_j||^2\right) = 0$$
$$\implies \sum_{x_i \in C_j} 2(x_i - m_j) = 0$$
$$\implies m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

The cluster center is the mean of all points present in a cluster

A K-Means Clustering Algorithm : Lloyd's Algorithm

- Input :

1. N data points : $\{x_i \in \Re^d \ : \ i = 1, ..., N\}$
2. The number of clusters : K
3. A termination criteria

- Output : K centroids : $\{m_j \in \Re^d \ : \ j = 1, ..., K\}$

Llyod's Algorithm

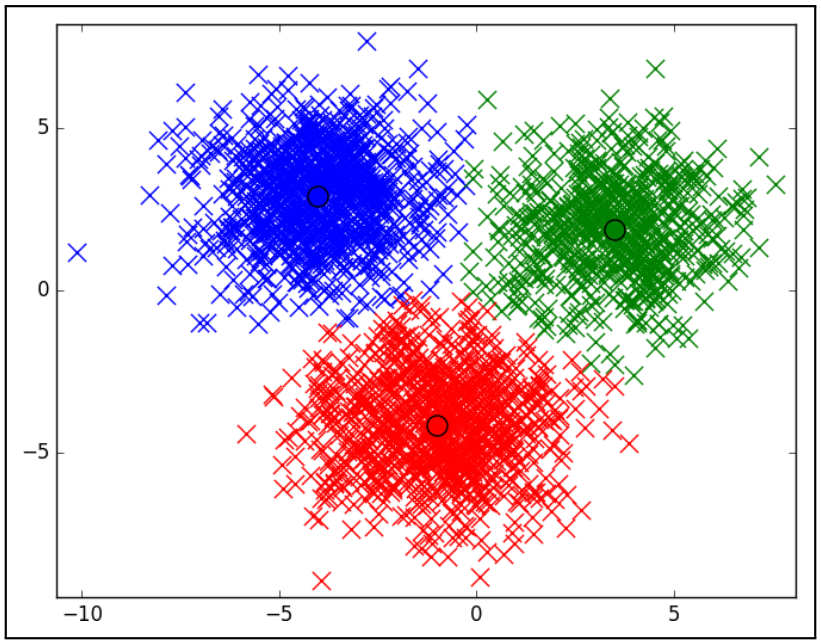
Step 1 Randomly initialize K centroids $m_j^{(0)} \in \Re^d \ , \ j = 1, ..., K$

Step 2 Assign points to the nearest cluster

$$C_j = \{x_i \ : \ ||x_i - m_j||^2 < ||x_i - m_t|| \ , \ t = 1, ..., K\}$$

Step 3 Update cluster centroids $m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$

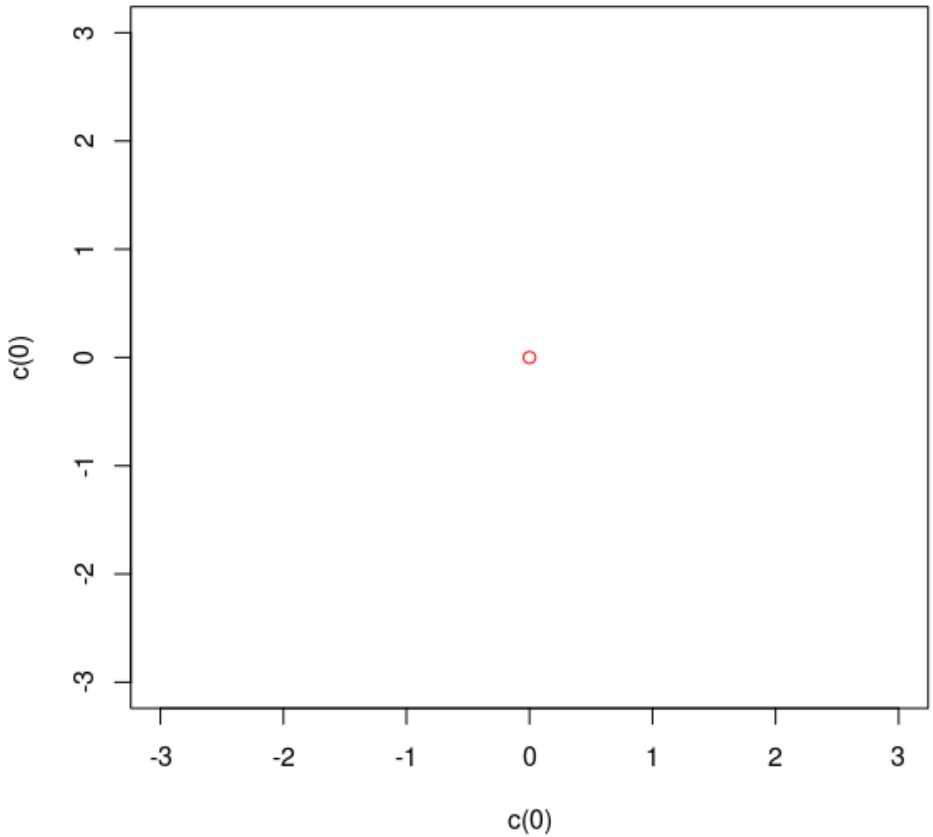
Step 4 Stop if termination criteria is satisfied, else go to Step 2.



Ideal output of the algorithm

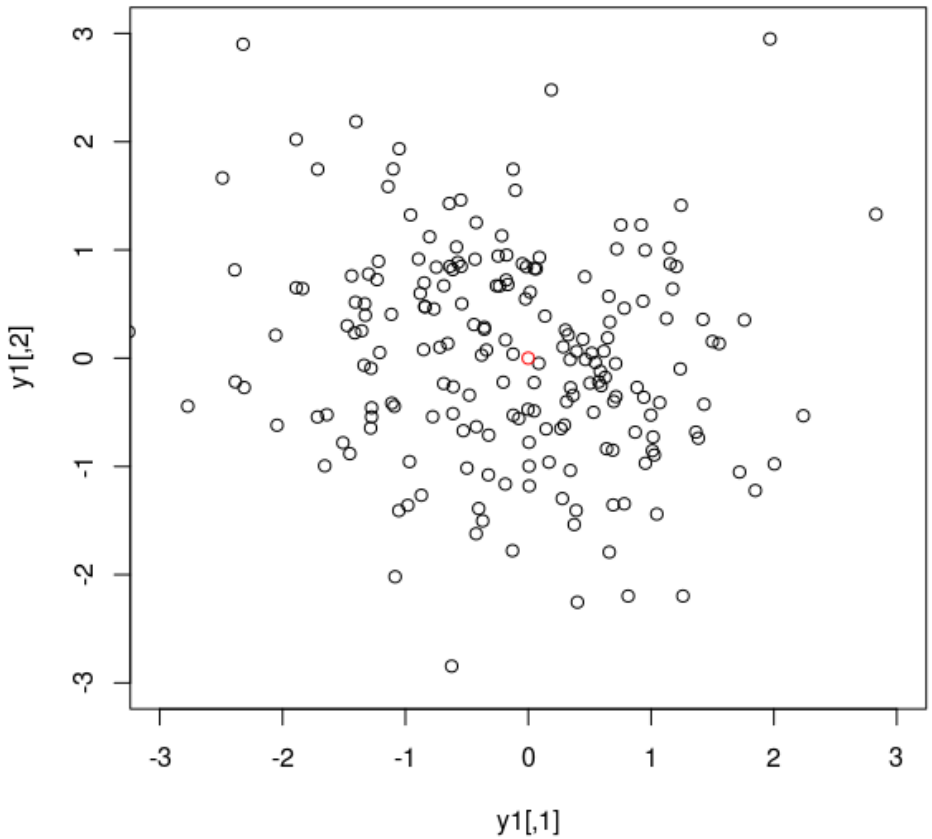
Generate Random Data : Multivariate Normal Distribution

```
plot(c(0),c(0),col='red')
```



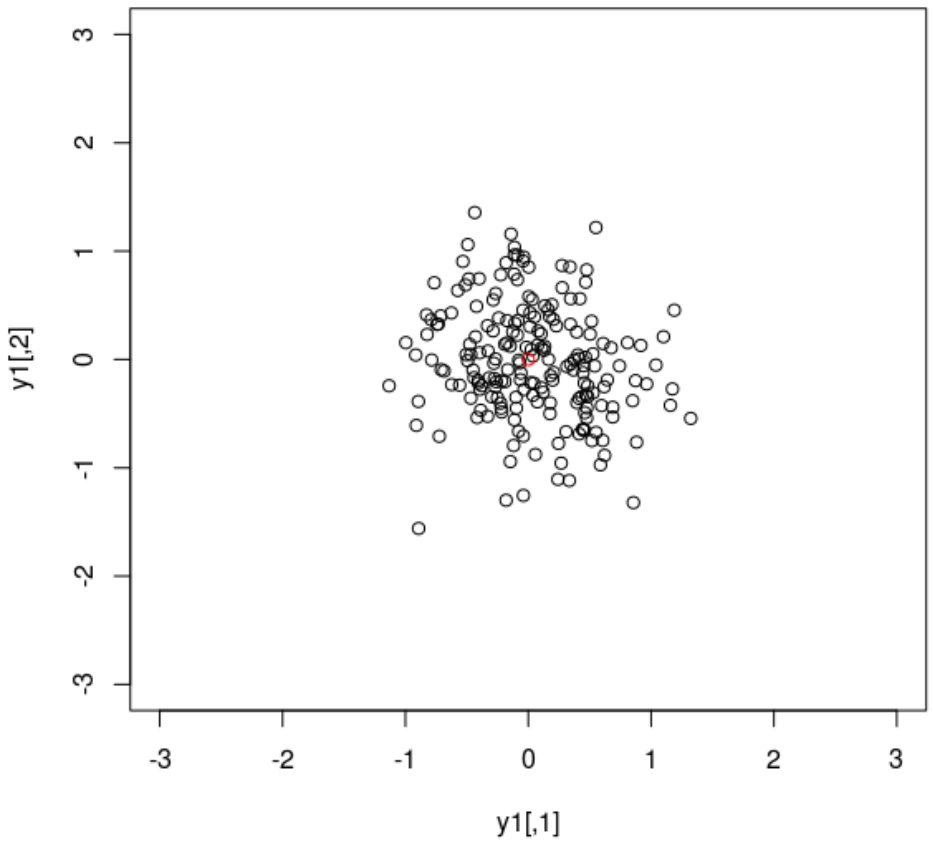
Generate Random Data : Multivariate Normal Distribution

```
y1 = cbind(rnorm(200, mean=0, sd=1), rnorm
    (200, mean=0, sd=1))
plot(y1, xlim=c(-3.0,3.0), ylim=c(-3.0,3.0))
points(c(0),c(0),col='red')
```



Generate Random Data : Multivariate Normal Distribution

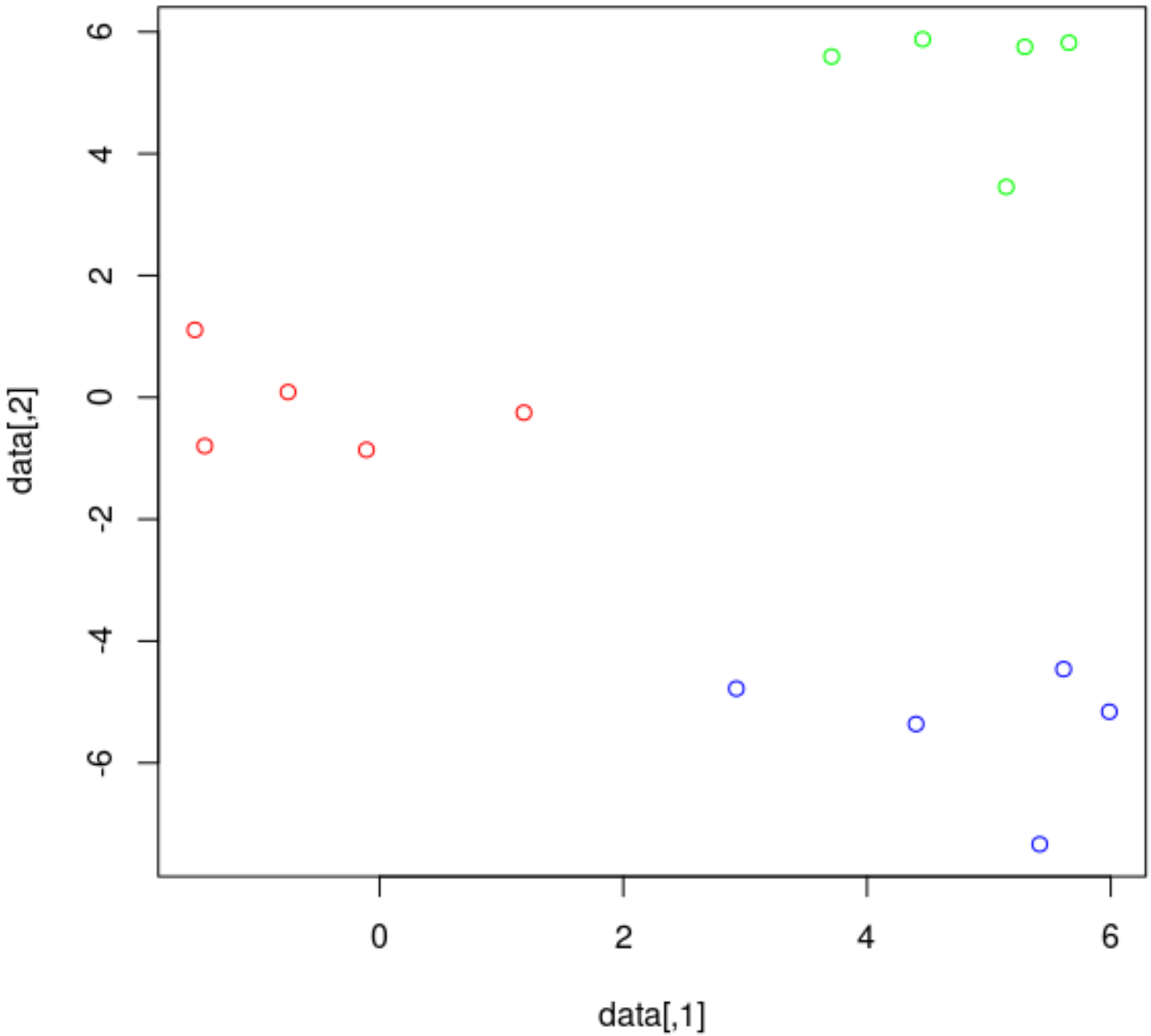
```
y1 = cbind(rnorm(200, mean=0, sd=0.5), rnorm
    (200, mean=0, sd=0.5))
plot(y1, xlim=c(-3.0,3.0), ylim=c(-3.0,3.0))
points(c(0),c(0),col='red')
```



Generate Random Data

```
x1 = cbind(rnorm(5, mean=0, sd=1), rnorm(5,
    mean=0, sd=1))
x2 = cbind(rnorm(5, mean=5, sd=1), rnorm(5,
    mean=5, sd=1))
x3 = cbind(rnorm(5, mean=5, sd=1), rnorm(5,
    mean=-5, sd=1))

data = rbind(x1,x2,x3)

label = c(rep(1,5), rep(2,5), rep(3,5))

K = 3

colvec = c("red","green","blue")[label]

plot(data, col=colvec)
```

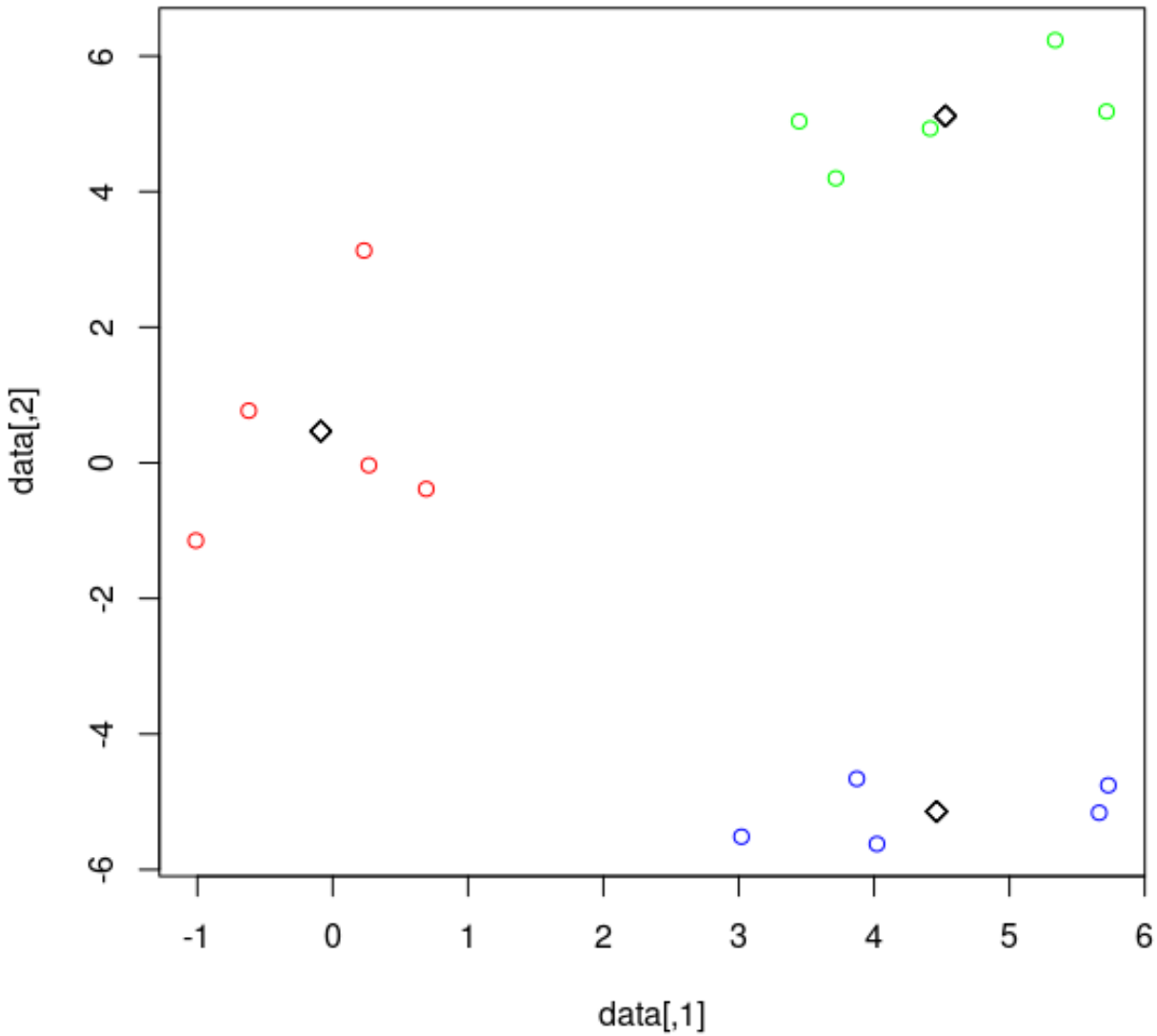
Generate Random Data



K-Means in R

```
km = kmeans(x=data, centers=K, algorithm = "
    Lloyd", nstart=20, iter.max=100)

print(km$cluster)

print(km$centers)

points(km$centers, pch=5)
```

K-Means in R



Llyod's Algorithm

Step 1 Randomly initialize K centroids $m_j^{(0)} \in \Re^d \ , \ j = 1, ..., K$

Step 2 Assign points to the nearest cluster
$C_j = \{x_i \ : \ ||x_i - m_j||^2 < ||x_i - m_t|| \ , \ t = 1, ..., K\}$

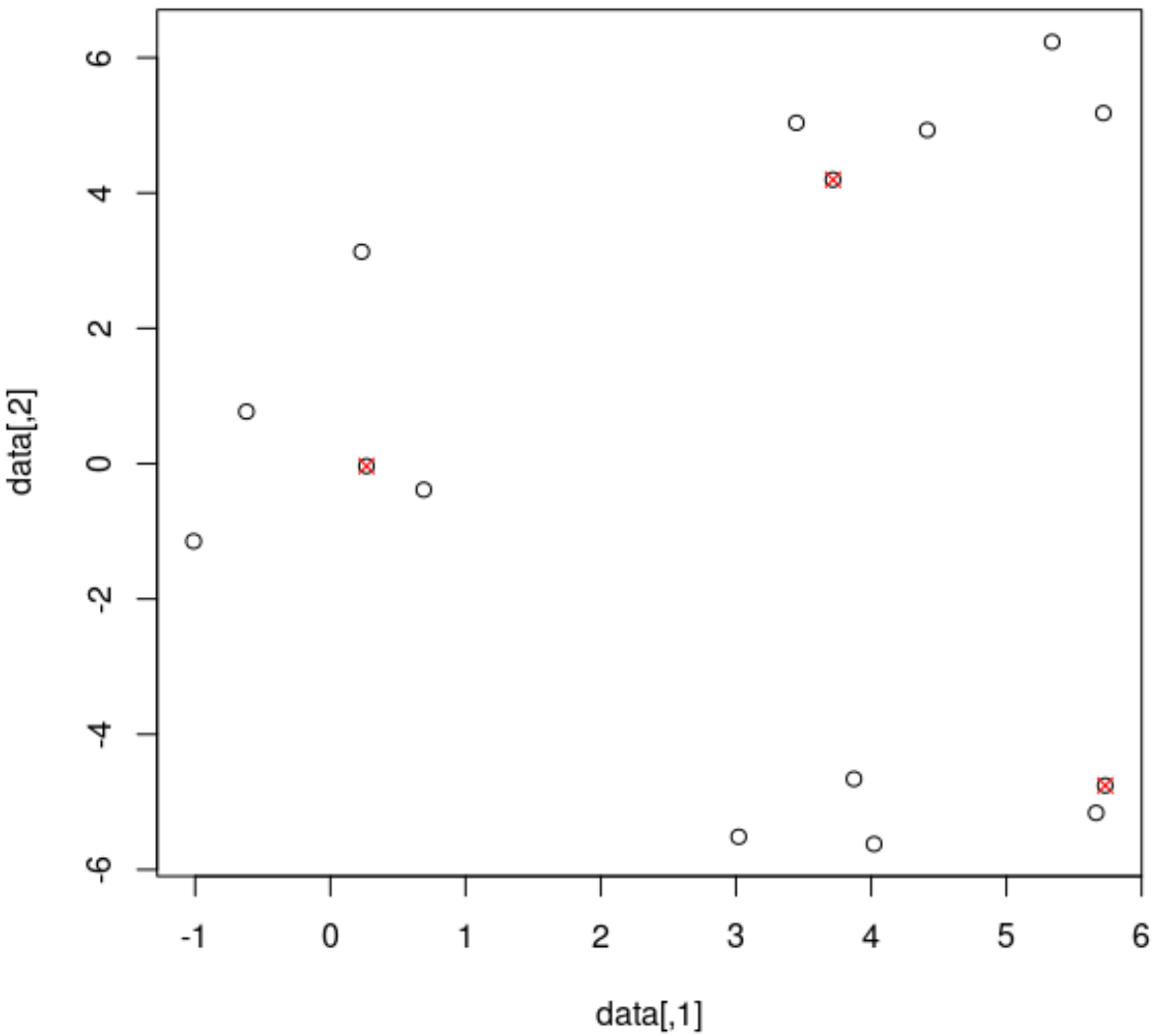
Step 3 Update cluster centroids $m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$

Step 4 Stop if termination criteria is satisfied, else go to Step 2.

Step 1 : Initialization

```
rownum = dim(data)[1]

seed = sample(1:rownum, K)

center = data[seed,]

print(center)

plot(data)

points(center, pch=4, col="red")
```

Step 1 : Initialization



Step 2 : Assign Points

```
mem = rep(0,rownum)

for(i in 1:rownum)
{
  temp = rep(0,K)
  for(j in 1:K)
  {
    temp[j] = sum((data[i,]-center[j,])^2)
  }
  mem[i] = which.min(temp)
}

print(mem)
```

Step 3 : Update Centers

```
prev_center = center

for(j in 1:K)
{

  if (sum(mem == j) > 1)

    center[j,] = colMeans(data[mem==j,])

  else

      center[j,] = data[mem==j]

}

print(center)
```

Step 4 : Test for Convergence

```
diff_centers = sum((center - prev_center)^2)

epsilon = 1e-6

print(diff_centers < epsilon)
```

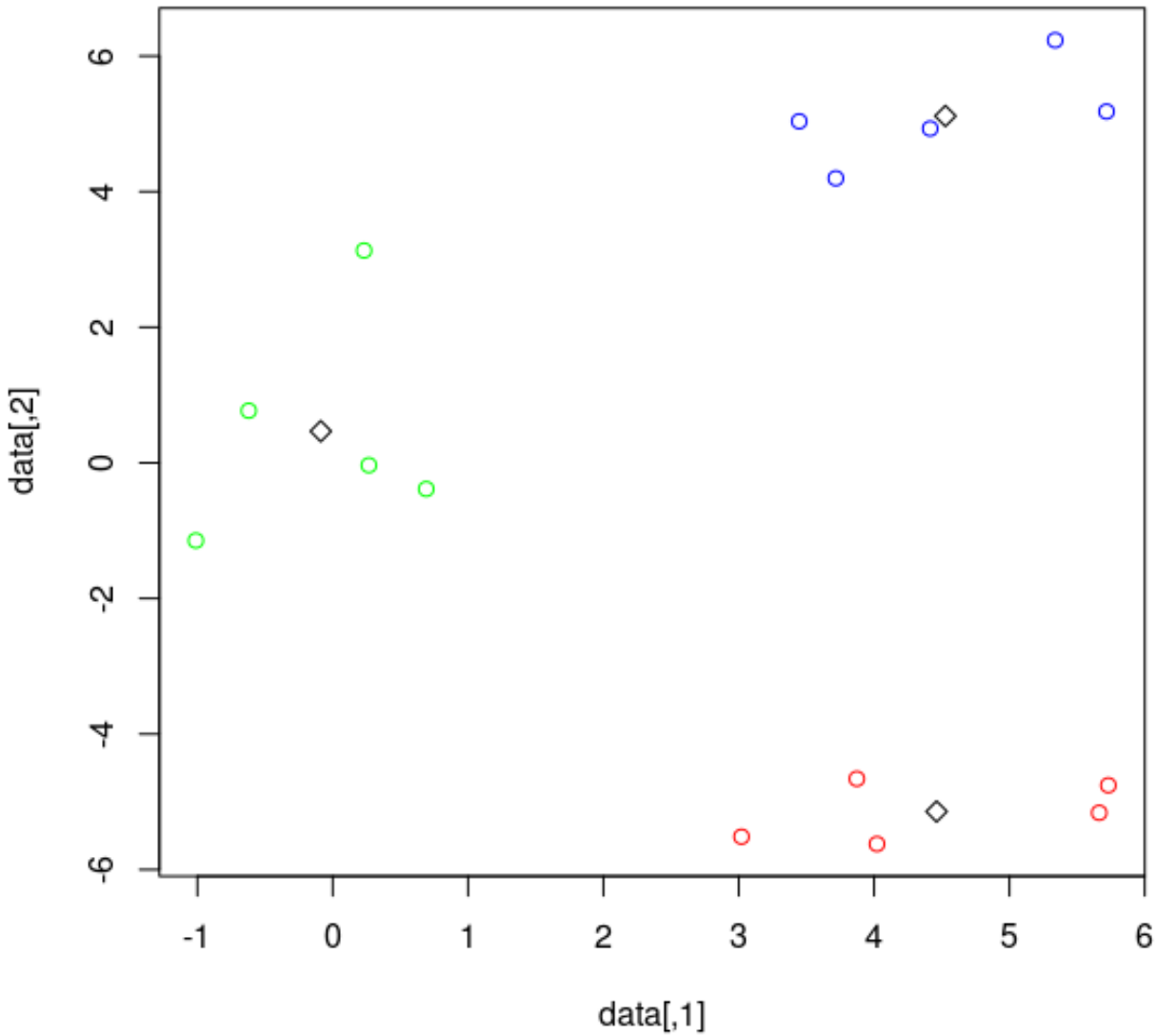
Step 5 : Loop till convergence I

```
num_iterations = 100
epsilon = 1e-6

mem = rep(0,rownum)
for(i in 1:num_iterations)
{
  for(i in 1:rownum)
  {
    temp = rep(0,K)
    for(j in 1:K)
    {
      temp[j] = sum((data[i,]-center[j,])^2)
    }
    mem[i] = which.min(temp)
  }
```

Step 5 : Loop till convergence II

```
    prev_center = center
    for(j in 1:K)
    {
      if (sum(mem == j) > 1)
        center[j,] = colMeans(data[mem==j,])
      else
        center[j,] = data[mem==j]
    }

    diff_centers = sum((center - prev_center)^2)
    if (diff_centers < epsilon)
    {     break     }
}
```

K-Means in R



K-Means : Output Parameters

```
km = kmeans(x=data, centers=K, algorithm = "
    Lloyd", nstart=20, iter.max=100)

print(km)
```

Cluster Evaluation I

- Internal Criteria : Percentage of Variance Explained

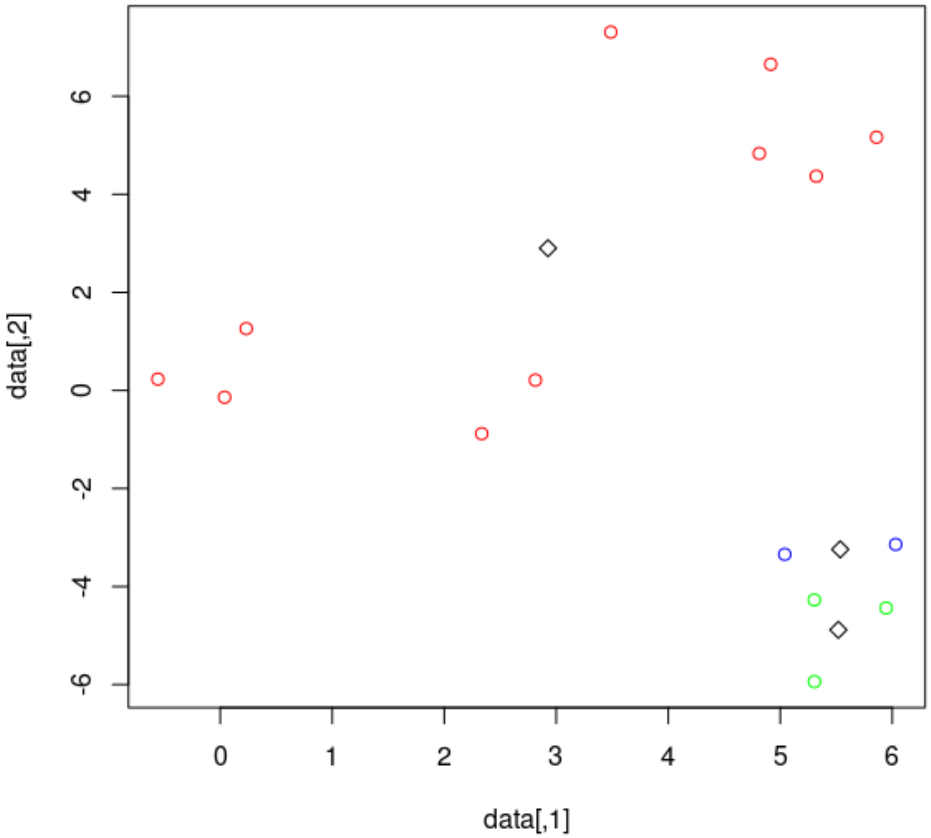
```
dmean=colMeans(data)

print(dmean)

#Total Sum of Squared distance from Data
    mean

print(km$totss)

#Total Within-cluster Sum of Squares from
    cluster centroid

print(km$withinss)
```

Cluster Evaluation II

```
#Total Between Cluster Sum of Squares

print(km$betweenss)

#Percentage of Variance Explained

print(km$betweenss / km$totss)
```

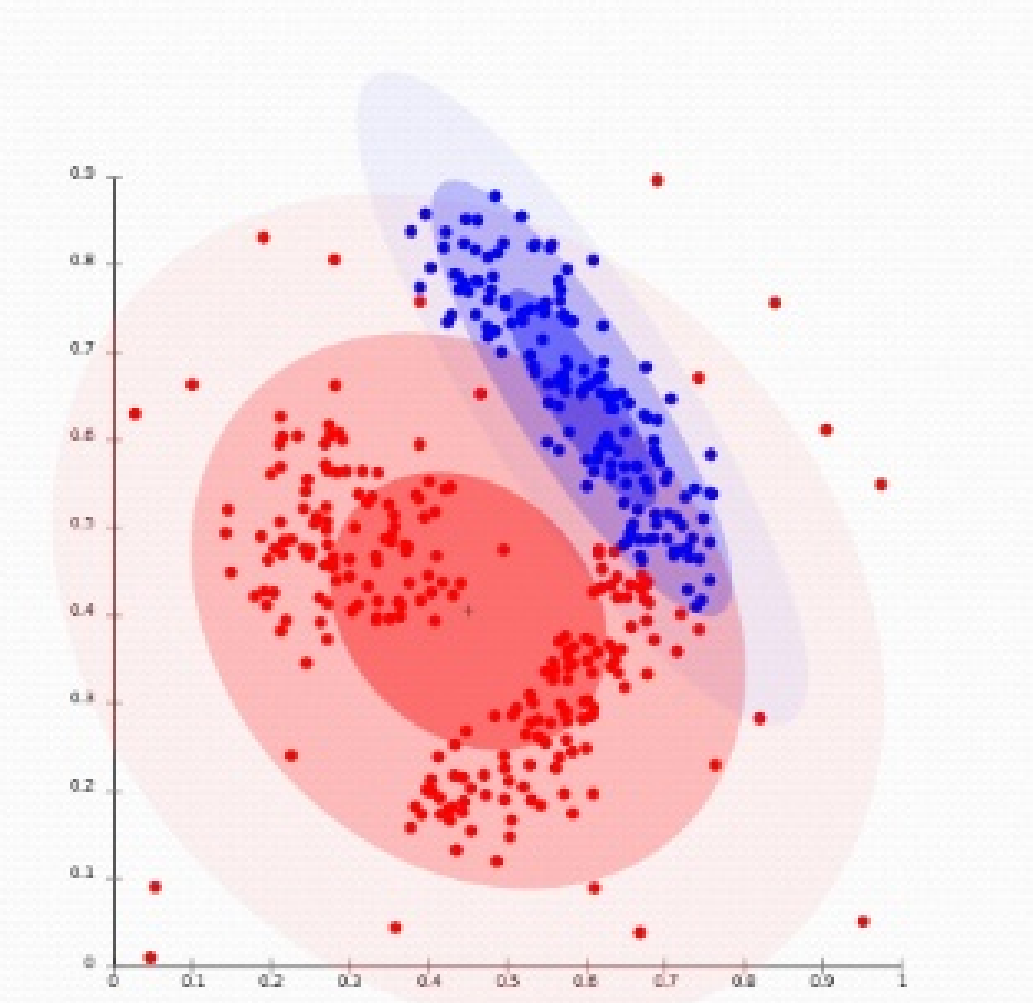
Challenges faced in K-means Clustering

- Different initial centroids lead to different local optima.



- No guarantee of the algorithm terminating at a global optima.
- Outliers affect centroid location.

Fuzzy c-Means



https://image.slidesharecdn.com/fuatsunum-111129050437-phpapp01/95/fuat-a-fuzzy-clustering-analysis-tool-4-728.jpg?cb=1322544452

Fuzzy c-Means

- Objective Function :

$$minimize \sum_{i=1}^{n} \sum_{j=1}^{c} \mu_{ij}^{m} ||x_i - v_j||^2$$

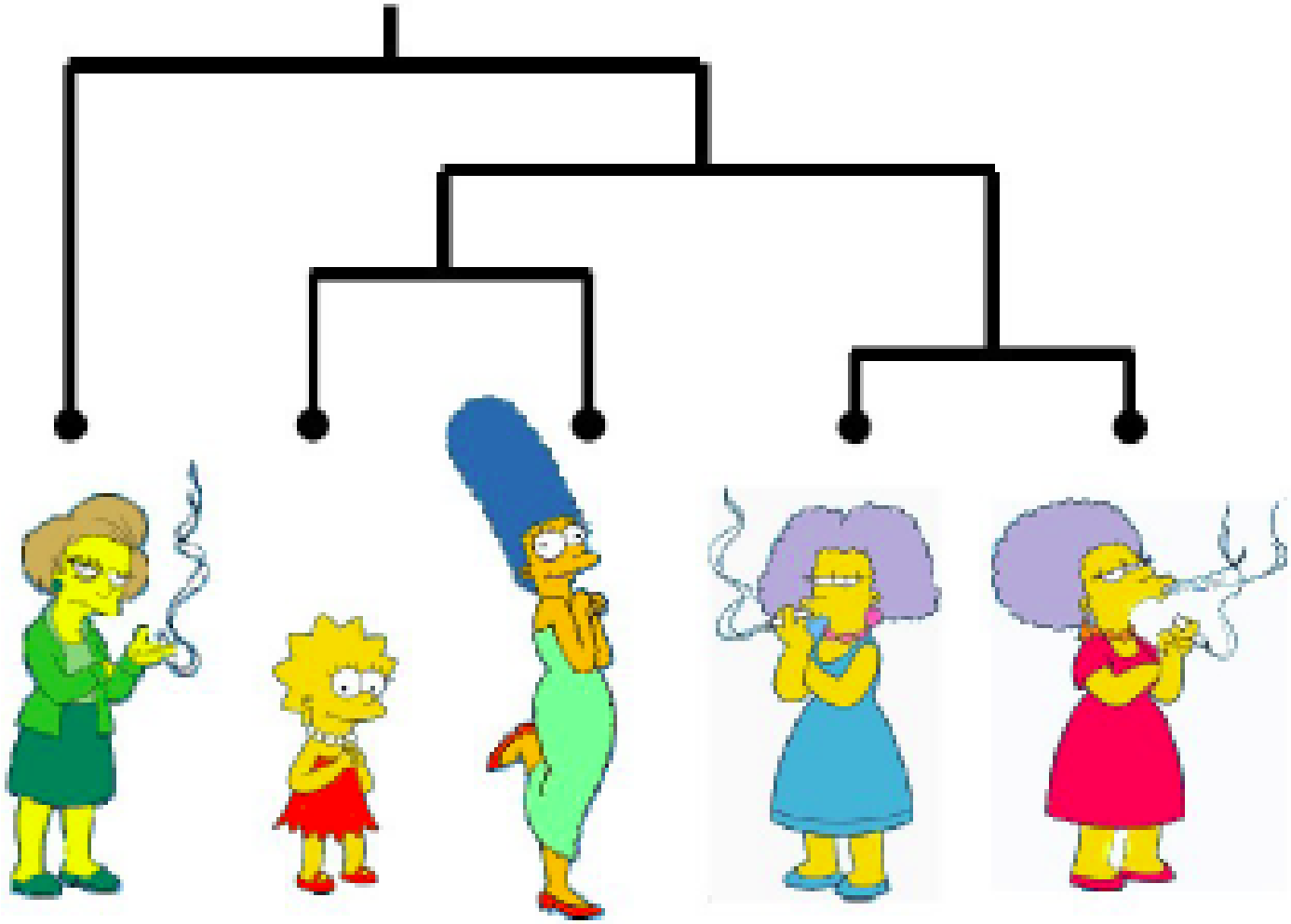
- $\mu_{ij} \in [0, 1]$ are membership values, representing how close point x_i is to center v_j.

Fuzzy c-Means

```
library("e1071")

result = cmeans(x=data, centers=3, iter.max
    =100, method="cmeans")

print(result)

print(result$membership)
```

Hierarchical Clustering

- Creates a hierarchy decomposition of a set of objects



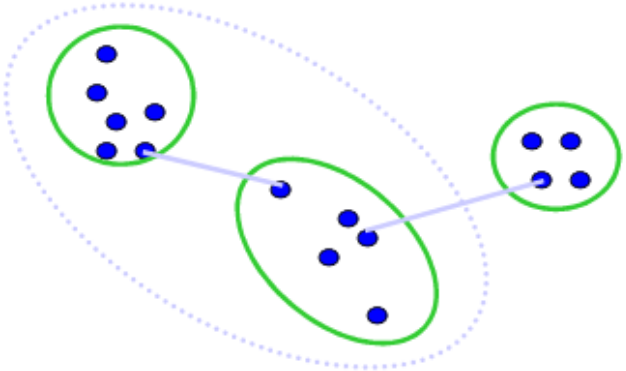
Hierarchical Clustering Approaches

2 Approaches

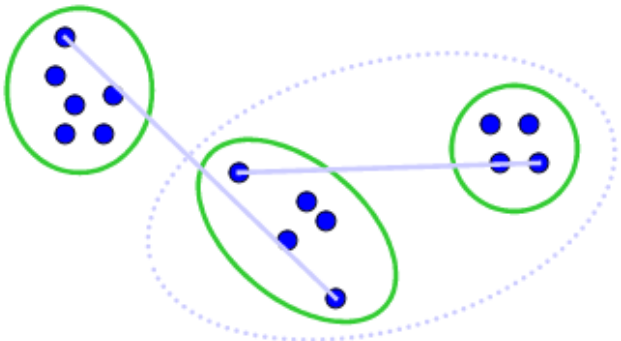
- **Bottom-Up / Agglomerative**
  - Sart with n single element clusters
  - Find best pair to merge into new cluster
  - Repeat untill all clusters fused together
- **Top-Down / Divisive**
  - Sart with all data in single cluster
  - Consider every possible way to divide cluster into 2
  - Choose best division
  - Recurse on both new clusters

Distance Criteria

- **Single Linkage** : distance between nearest pair of points in two clusters. [Long skinny clusters]

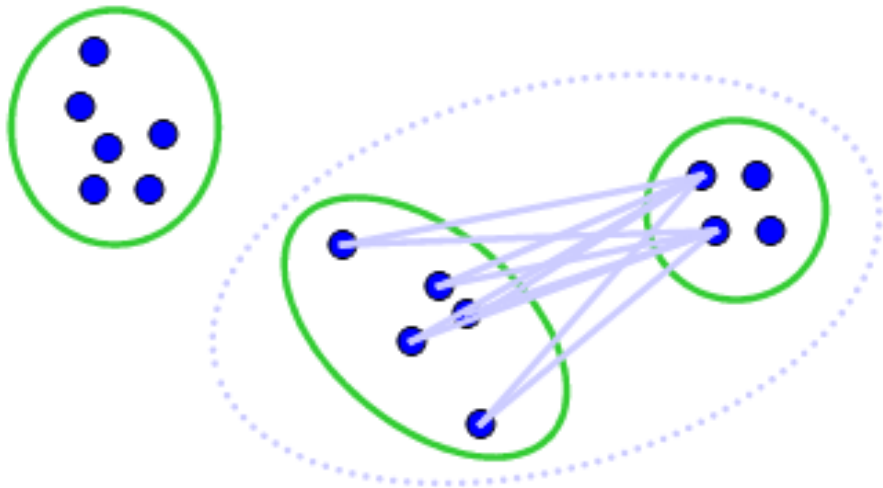


- **Complete Linkage** : distance between farthest pair of points in two clusters [Tight clusters]

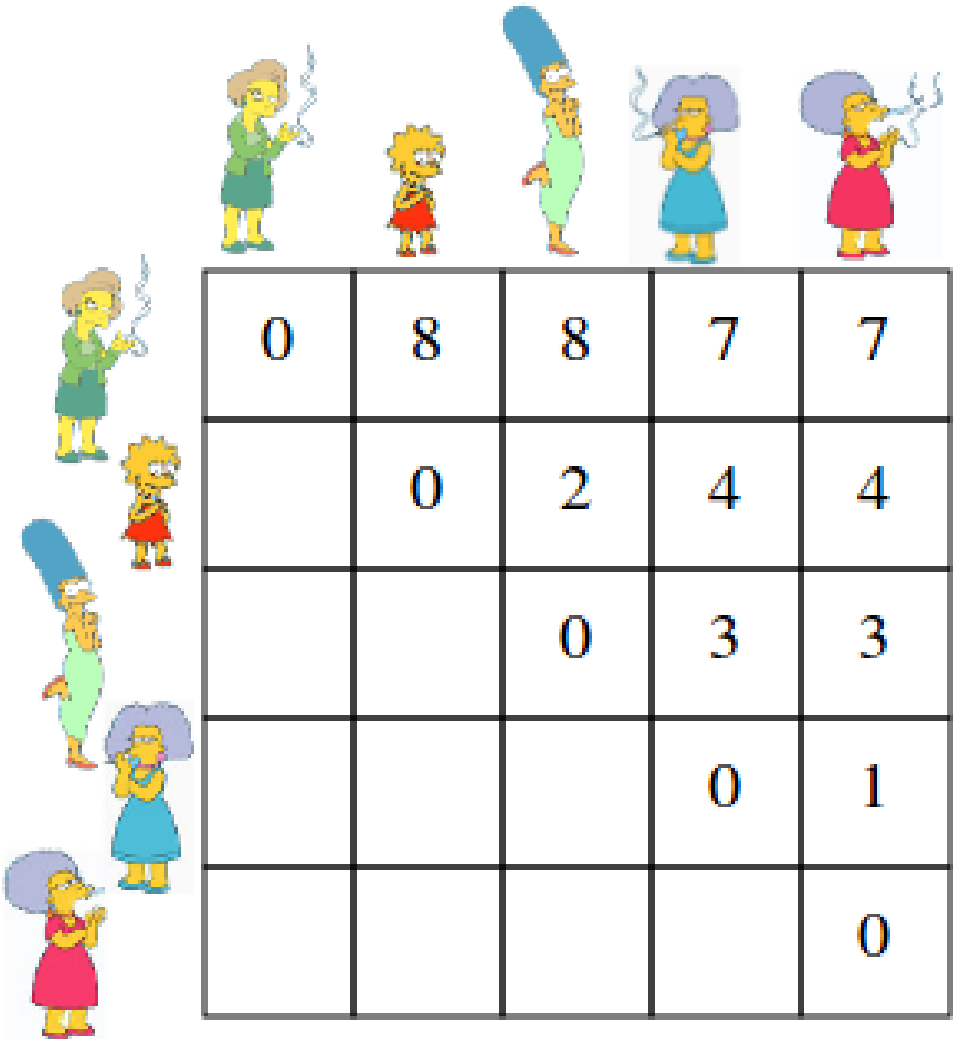


Distance Criteria II

- **Average Linkage** : average distance between all pair of points in the two clusters [Tight clusters, Robust to noise]



Pairwise Dissimilarity Matrix



0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Hierarchical Clustering in R

```
h_cluster = hclust(dist(data), method="
    complete")

plot(h_cluster)

clusterCut = cutree(h_cluster, 3)

table(clusterCut, label)
```

Clustering for Big Data I

- Large sample dataset ($n >> d$).
- Hierarchical Clustering not feasible $\mathcal{O}(n^2)$.
- K-Means runs faster $\mathcal{O}(nkd)$.
- For very large datasets, iterating too many times using K-Means can also be too expensive.
- Mini-batch K-Means often used for such large datasets. (**MiniBatchKmeans** Available in **clusterR**. package)

Clustering for Big Data II

- High Dimensional Datasets : $n << d$
- Bioinformatics : Gene Expression/ DNA Methylation Data Clustering
- TCGA Brain (LGG) Cancer Dataset
  - No. of Features : 27578 DNA Methylation Markers in human genome
  - No. of Sample : 267 LGG Brain cancer patients
  - No. of Clusters : 3 subtypes LGG
- Approach :
  - Feature Selection : Most varying features
  - Dimensionality Reduction : Principal Component Analysis (PCA)

Dimension Reduction Example

```
print(iris)
Data=iris[,1:4]

npc=2
PC=prcomp(Data,center=T,scale=F,retx=T)$x[,1:
    npc]

plot(PC,main="PRINCIPAL COMPONENTS OF IRIS")
Clusters=kmeans(PC,3)$cluster
print(Clusters)

colvec = c("red","green","blue")[Clusters]
plot(PC, col=colvec)
```

Thank You