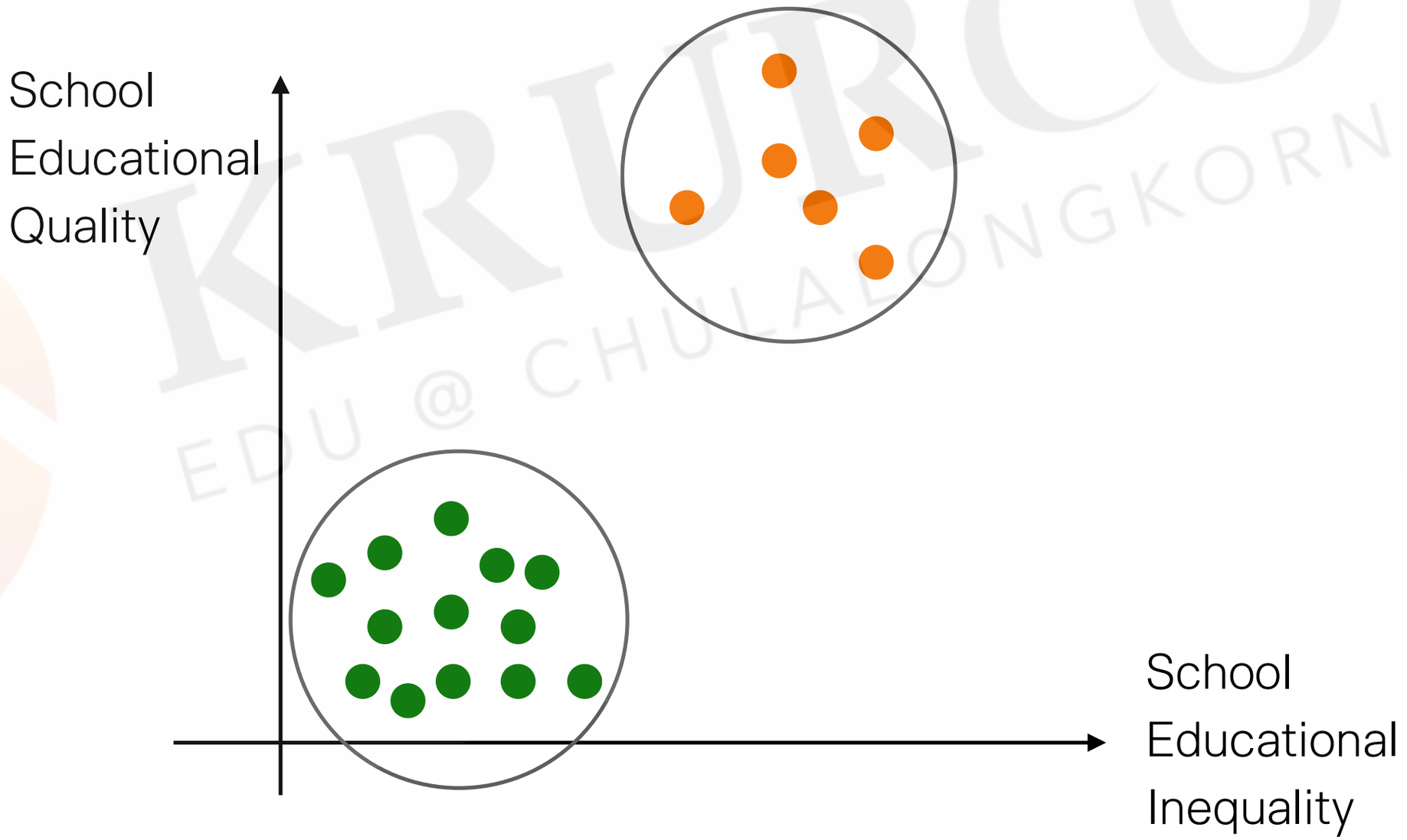
Clustering

- K-means clustering
- Hierarchical clustering
- Density-based clustering

Clustering

- Clustering is a set of methods or algorithms that are used to find natural groupings according to predefined properties of variables in a dataset.



Clustering

Clustering is mostly used for data without labels and with predefined classes for training models.

1. **Exploratory data analysis** — for unlabelled data, clustering is used to explore underlying structure and categories in the dataset.
2. **Generate training data** — sometimes after processing unlabelled data with clustering method, it can be labeled for further training with supervised learning.
3. **Recommender systems** — with the help of clustering we can find properties of similar items and use these properties to make recommendations.
4. **Anomaly Detection** — This is used to find outliers with clustering
5. **Natural Language Processing** — grouping similar words, texts, articles or tweets without labelled data.

Types of Clustering

- K-means clustering
- K-medoids clustering
- K-modes clustering
- Agglomerative hierarchical clustering
- Divisive clustering
- Density based clustering

K-means clustering

K-means clustering

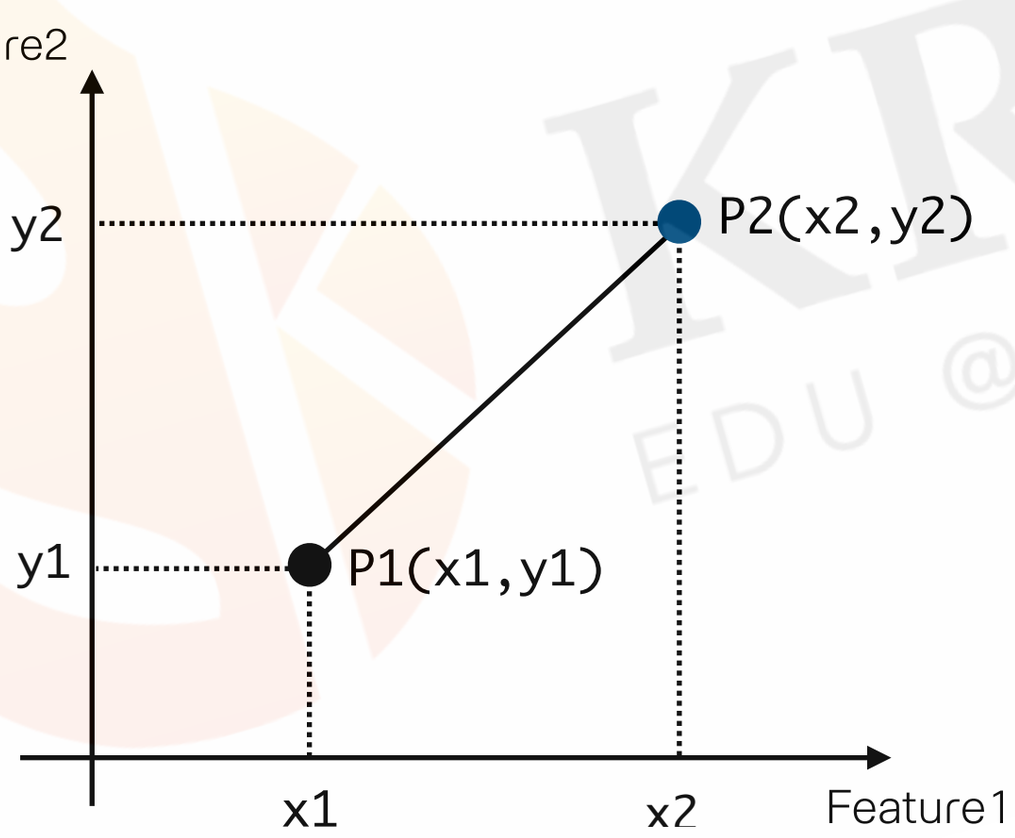
- K-means clustering is a basic type of unsupervised learning that find natural groupings in accordance to a predefined **similarity or distance measure.**

Distance	Euclidean distance
	Manhattan distance
	Cosine distance
	Hamming distance

Euclidean distance

A straight line distance between any two points in any n-dimensional space (not just two dimensional space).

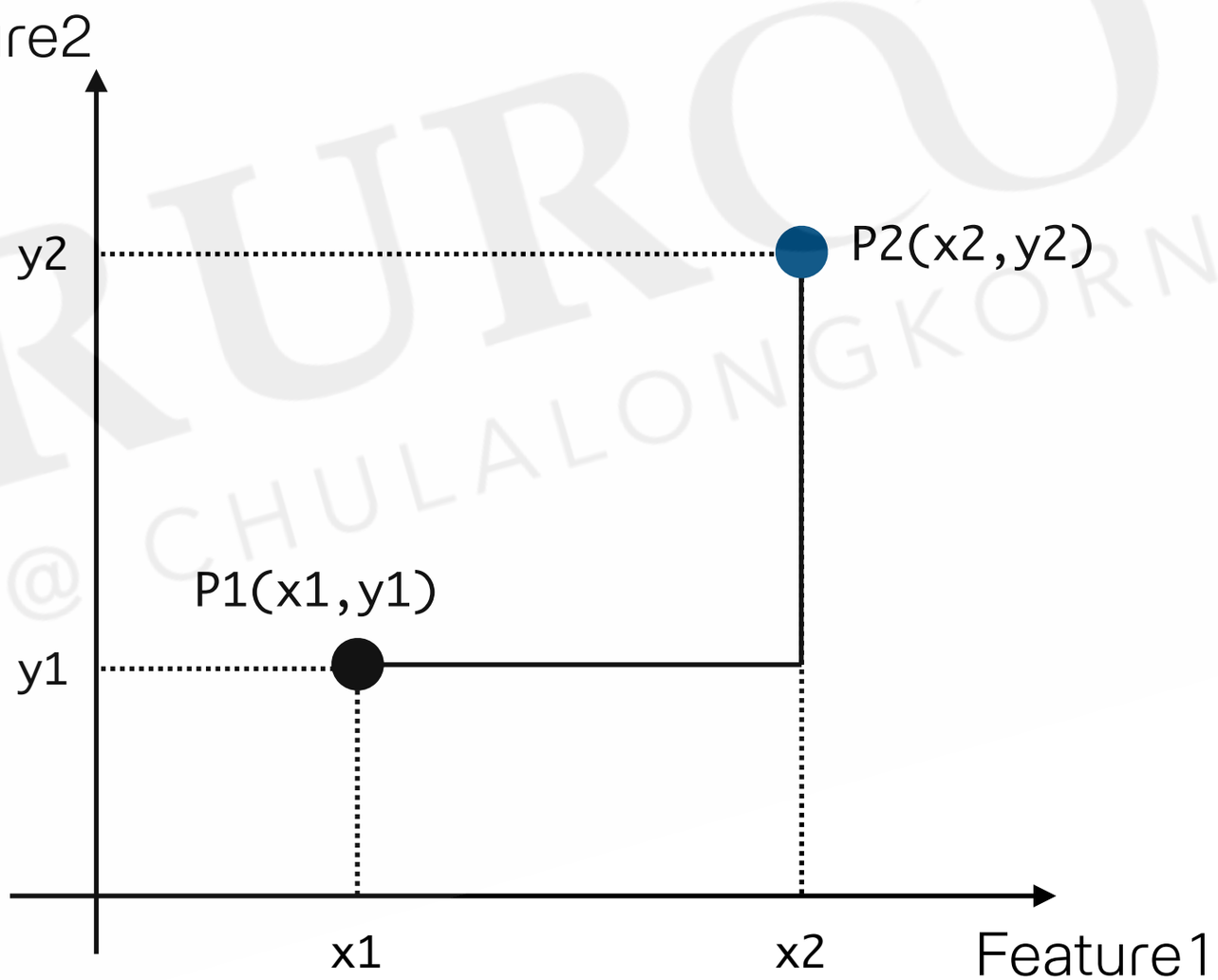
$$D(P_1, P_2) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \ldots + (x_n - y_n)^2}$$



Manhattan distance

Distance between two points measured along right angle to the axes.

$$D(P_1, P_2) = \sum_{i=1}^{n} |x_i - y_i|$$



Cosine similarity

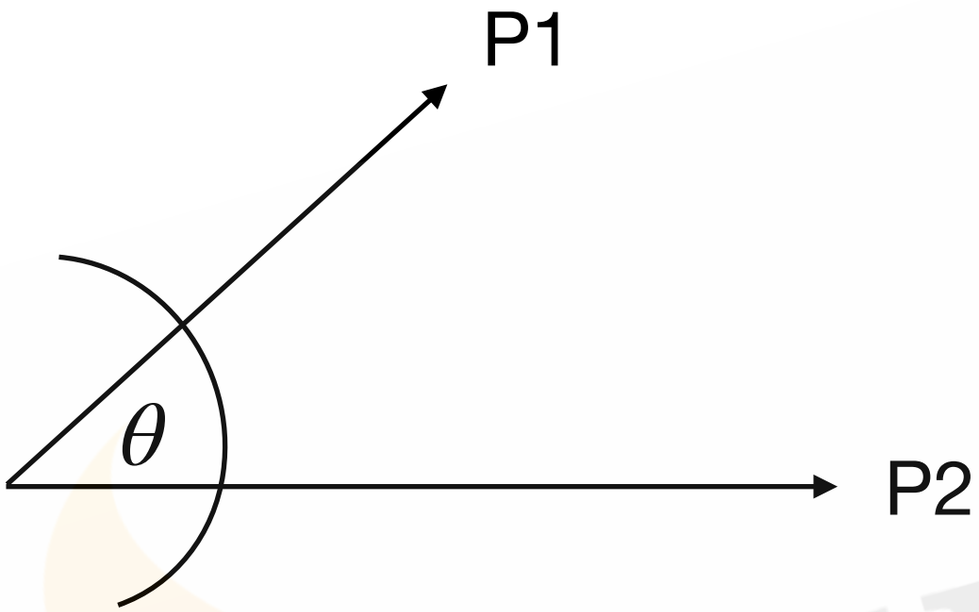
Cosine distance is the similarity between any two points is defined as the cosine of angle between any two points with origin as its vertex

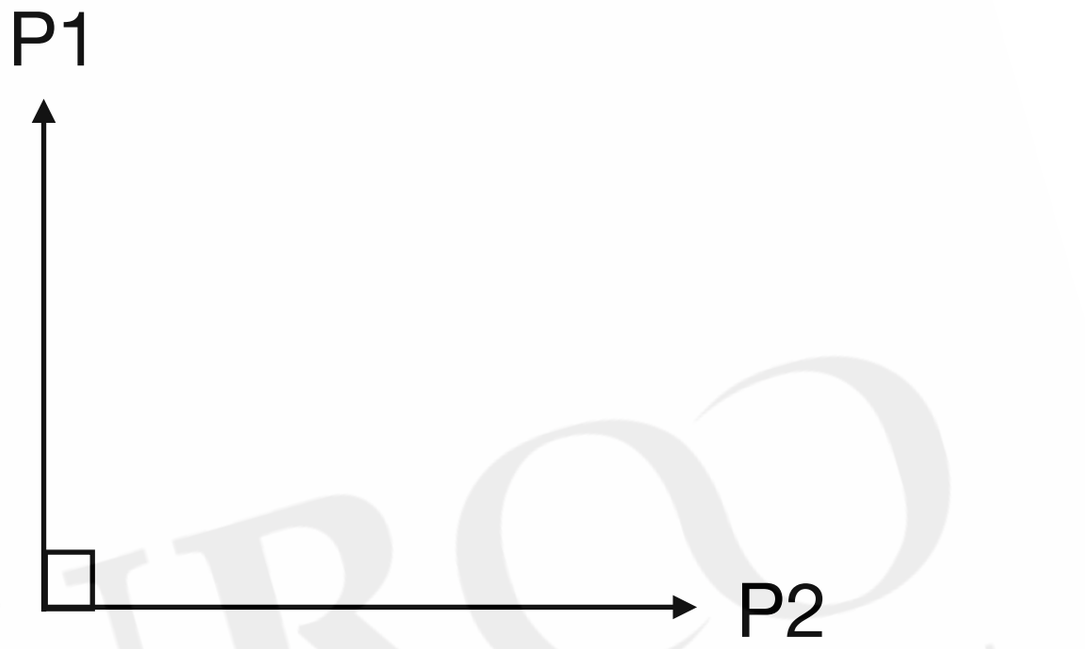
$$u \cdot v = |u||v| cos\theta \implies cos\theta = \frac{u \cdot v}{|u||v|} \in [-1,1]$$

Cosine distance

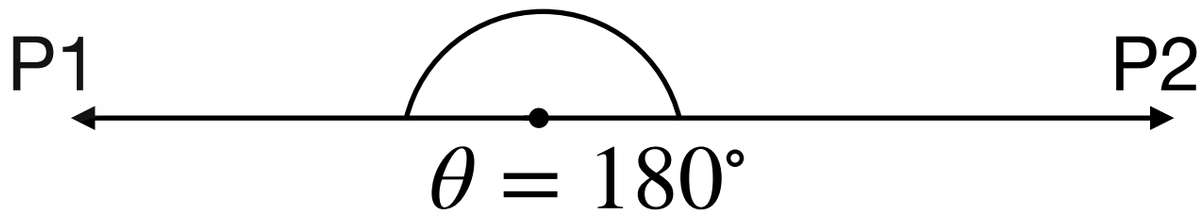
$$1 - cos\theta = 1 - \frac{u \cdot v}{|u||v|}$$

Cosine similarity





Similar



Not similar

Pearson correlation vs Cosine similarity

$$\rho_{xy} = \frac{\sum((x_i - \bar{x}) * (y_i - \bar{y}))}{\sqrt{\sum(x_i - \bar{x})^2} * \sqrt{\sum(y_i - \bar{y})^2}}$$

$$cos(\theta) = \frac{\sum x_i * y_i}{\sqrt{\sum x_i^2} * \sqrt{\sum y_i^2}}$$

K-means clustering - algorithm

1. Specify the number of clusters (K) that will be generate in the final solution.
2. Randomly selecting K subjects from the dataset as the initial cluster centers or centroids.
3. Assigned each remaining subjects to their closest centroid, based on euclidean distance between subject and the centroid. (cluster assignment step)
4. For each of the clusters update the cluster centroid by calculating the new mean values of all data point in the cluster. (centroid update step)
5. Iteratively minimise the total within sum squared by iterate step3 and 4 until the cluster assignments stop changing (or maximum number of iteration is reached).

K-means clustering

- The basic idea behind k-means clustering consists of defining clusters so that the total intra-cluster variation (known as total within-cluster variation) is minimized.

$$Total_{withinSS} = \sum_{k=1}^{K} WSS_k$$

where

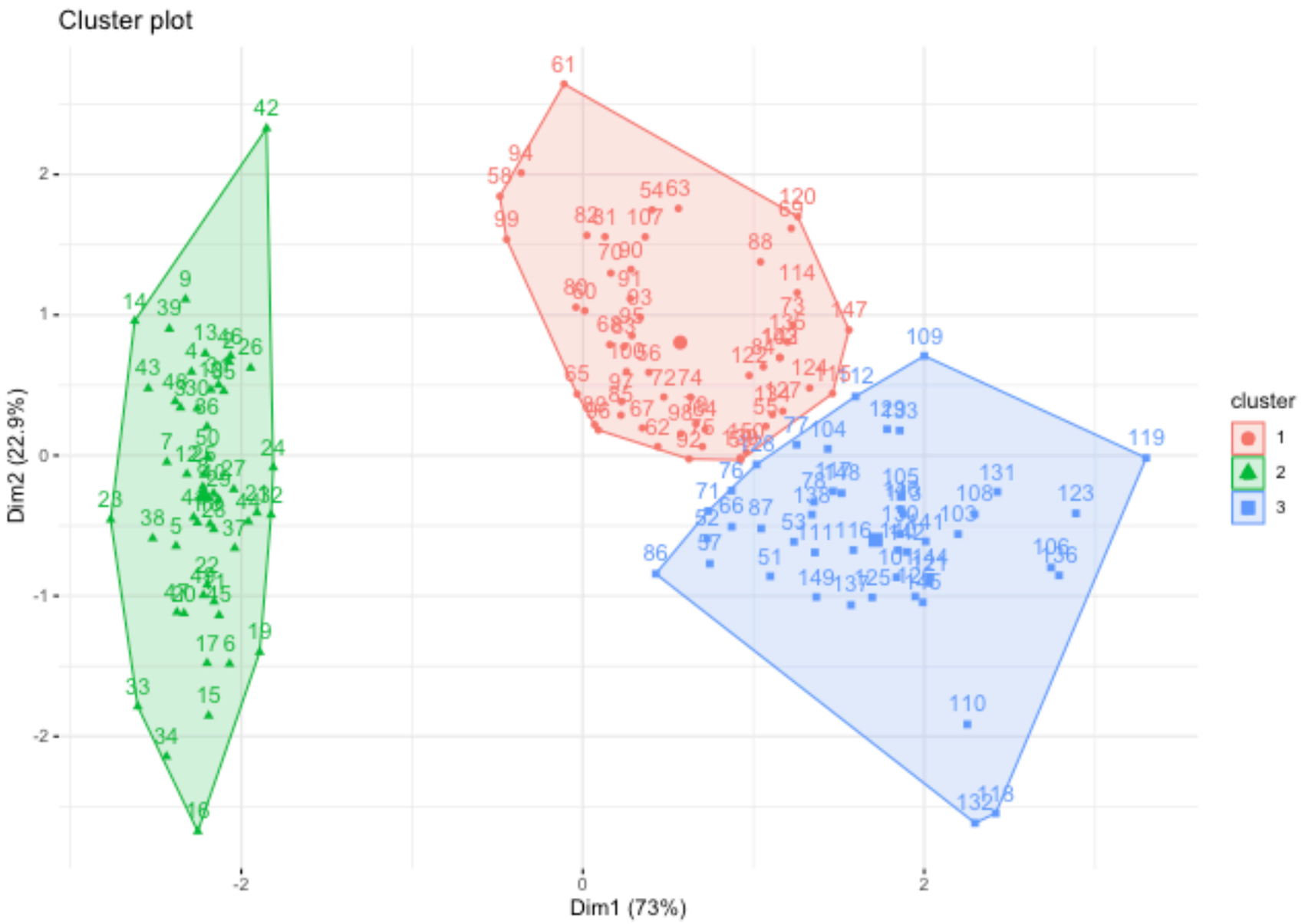
- $WSS_k = \sum_{x_i \in C_k} (x_i - \mu_k)^2$ is euclidean distance or other distance.
- x_i is a vector of clustering features (data point) belonging to the cluster C_k
- μ_k is a vector mean of the data points assigned to the cluster C_k (called centroids)

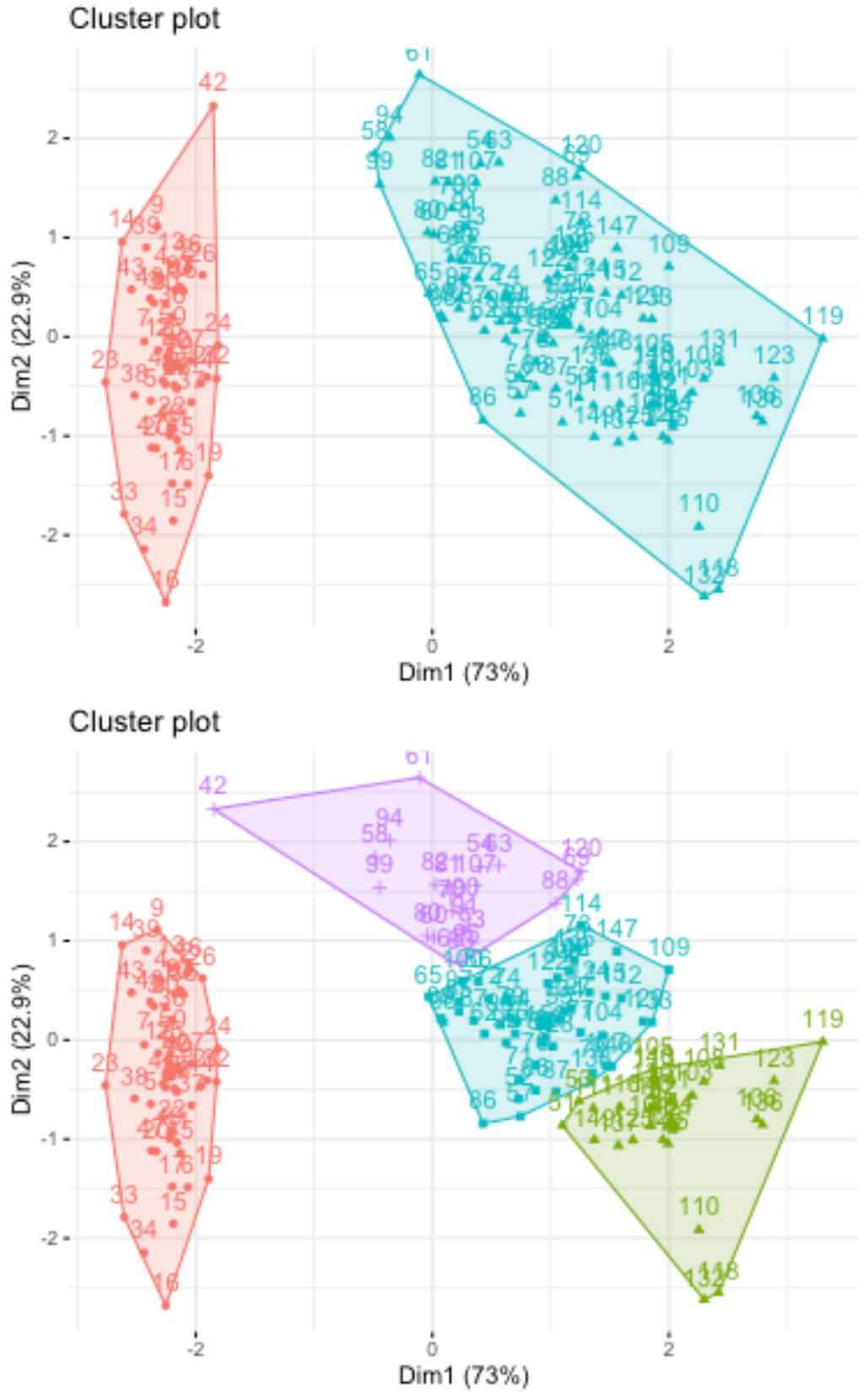
(Hartigan and Wong, 1979)

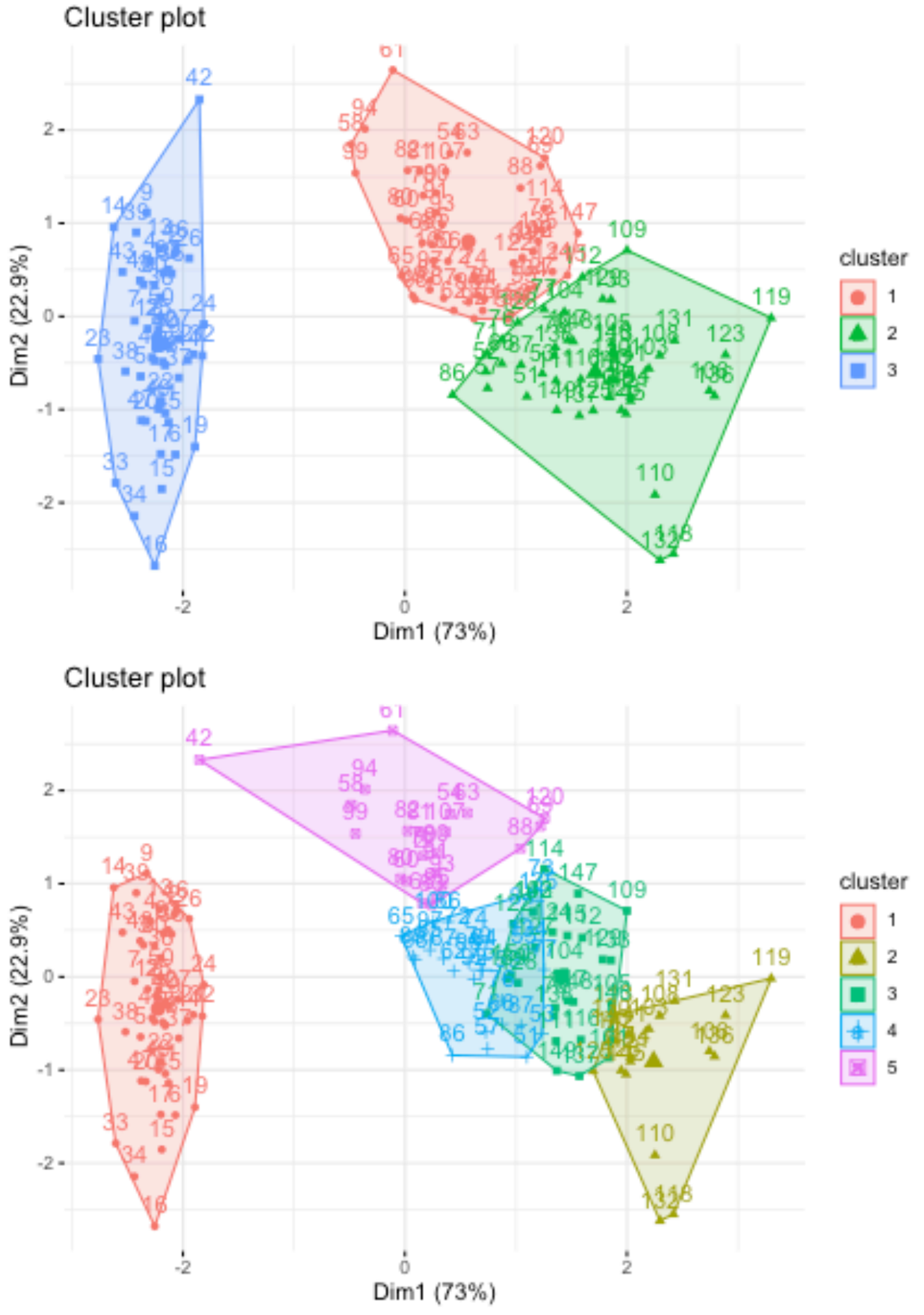
```
> km3<-kmeans(dat,centers=3)
> km3
K-means clustering with 3 clusters of sizes 33, 96, 21

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   -0.8135055   1.3145538   -1.2825372  -1.2156393
2    0.5690971  -0.3705265    0.6888118   0.6609378
3   -1.3232208  -0.3718921   -1.1334386  -1.1111395

Clustering vector:
  [1] 1 3 3 3 1 1 1 1 3 3 1 1 3 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 3 3 1 1 1 3 3 1 1 3 1 1 3 3 1
 [45] 1 3 1 3 1 1 2 2 2 2 2 2 2 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [89] 2 2 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[133] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  17.33362 149.25899  23.15862
 (between_SS / total_SS =  68.2 %)

Available components:

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

K-means clustering - Visualizing cluster

```
> fviz_cluster(km3,data=dat,ggtheme=theme_minimal())
```




Cluster plot
Dim2 (22.9%)
Dim1 (73%)
cluster
1
2


Cluster plot
Dim2 (22.9%)
Dim1 (73%)
cluster
1
2
3

Cluster plot
Dim2 (22.9%)
Dim1 (73%)
cluster
1
2
3
4

Cluster plot
Dim2 (22.9%)
Dim1 (73%)
cluster
1
2
3
4
5

Determining the number of clusters

- Using a priori knowledge
- Rule of thumb: $k = \sqrt{n/2}$; where n is sample size
- Statistical methods
  - Elbow method
  - Silhouette method
  - Gap statistic

See Charrad et al. (2015)

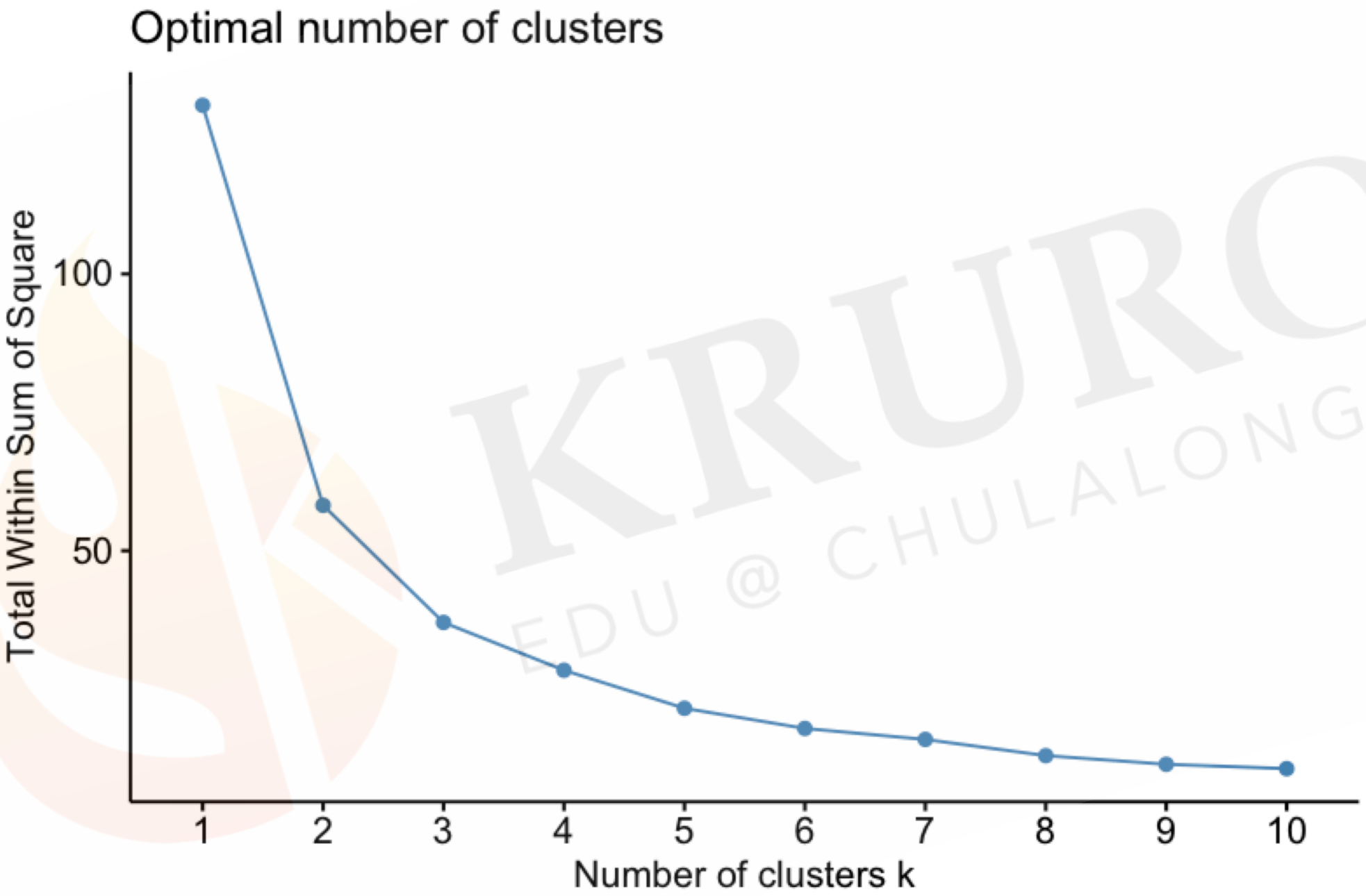
http://www.jstatsoft.org/v61/i06/paper

Elbow Method

Minimizes - $Total_{withinSS} = \sum_{k=1}^{K} WSS_k$

1. Compute clustering algorithm (e.g., k-means clustering) for different values of *k*. For instance, by varying *k* from 1 to 10 clusters
2. For each *k*, calculate the total within-cluster sum of square (wss)
3. Plot the curve of wss according to the number of clusters *k*.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

```
> fviz_nbclust(x=iris[,1:2],FUNcluster=kmeans,method="wss")
```



Average Silhouette Method

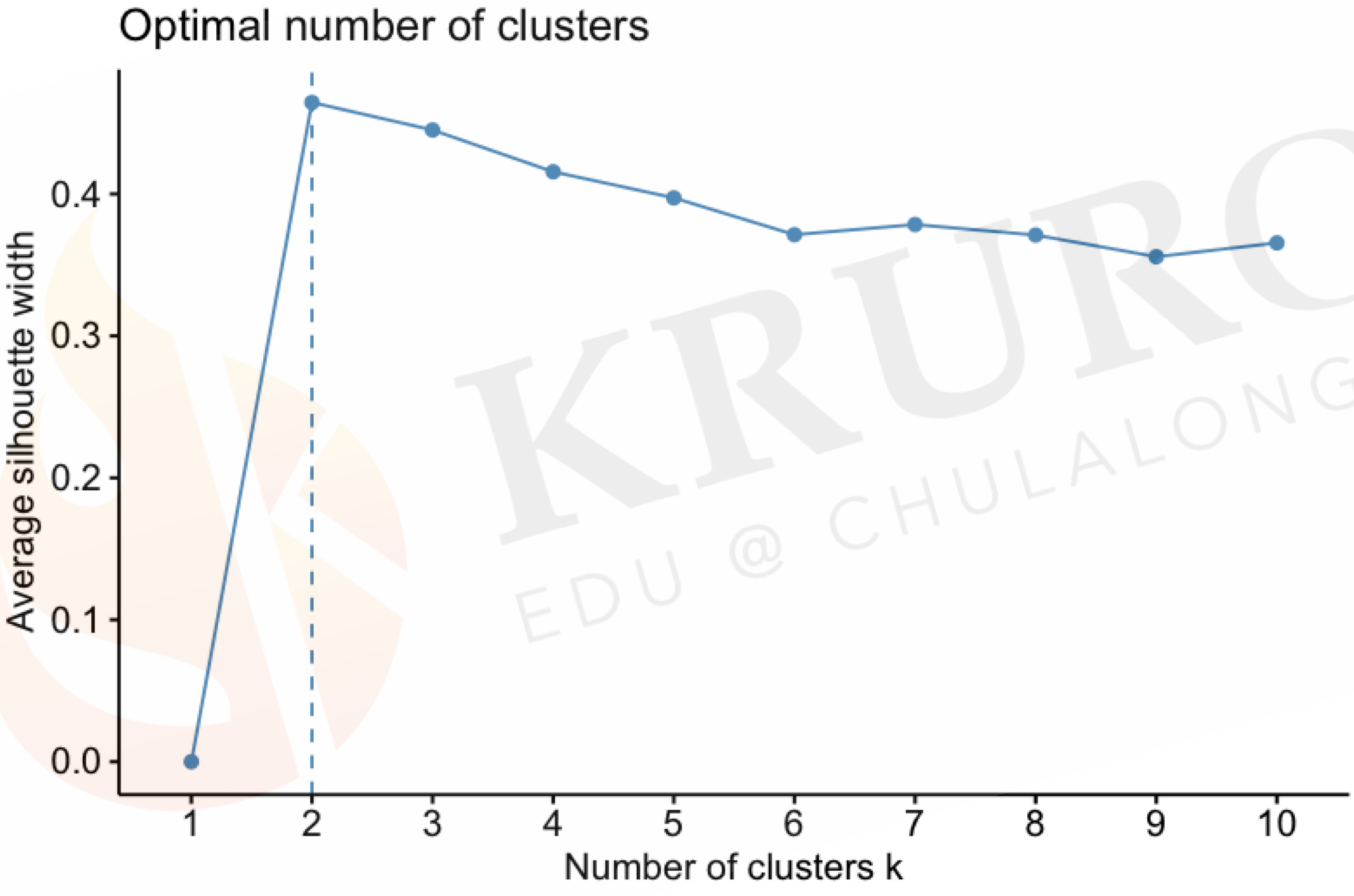
The silhouette analysis measures how well an observation is clustered and it estimates the **average distance between clusters**. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters.

For each observation i, the silhouette width s_i is calculated by:

1. For each observation i, calculate the average dissimilarity a_i, between observation i and all other points of the cluster to which i belong.
2. For all other cluster C, to which i does belong, calculate the average dissimilarity $d(i, C)$ of i to all observations of C. The smallest of these $d(i, C)$ is defined as $b_i = min_C d(i, C)$. The value of b_i can be seen as the dissimilairy between i and its neightbor cluster.
3. Finally the sihouette width of the observation i is defined by the formula

$$S_i = \frac{b_i - a_i}{max(a_i, b_i)}$$

```
> fviz_nbclust(df, kmeans, method = "silhouette")
```



Gap Statistic Method

1. Cluster the observed data, varing the number of clusters from $k = 1,2,...,k_{max}$ and compute the corresponding within cluster variation W_k.
2. Generate B reference datasets and cluster each of them with varing number of cluster $k = 1,2,...,k_{max}$ and compute the corresponding within cluster variation W_k^*.
3. Compute the estimated gap statistics

$$Gap_n(k) = E_n^* log(W_k) - log(W_k)$$

Where $E_n^* = avg(log(W_k^*))$. The gap statistics measures the deviation between observed W_k and its expected value of the reference W_k^*

4. Let $\overline{\omega} = \frac{1}{B}\sum_b log(W_{kb}^*)$, and compute standard deviation

$$sd(k) = \sqrt{(1/b)\sum_b (log(W_{kb}^*) - \overline{\omega})^2}$$ and define $s_k = sd_k \times \sqrt{1 + 1/B}$

5. Choose the number of clusters as the smallest k such that

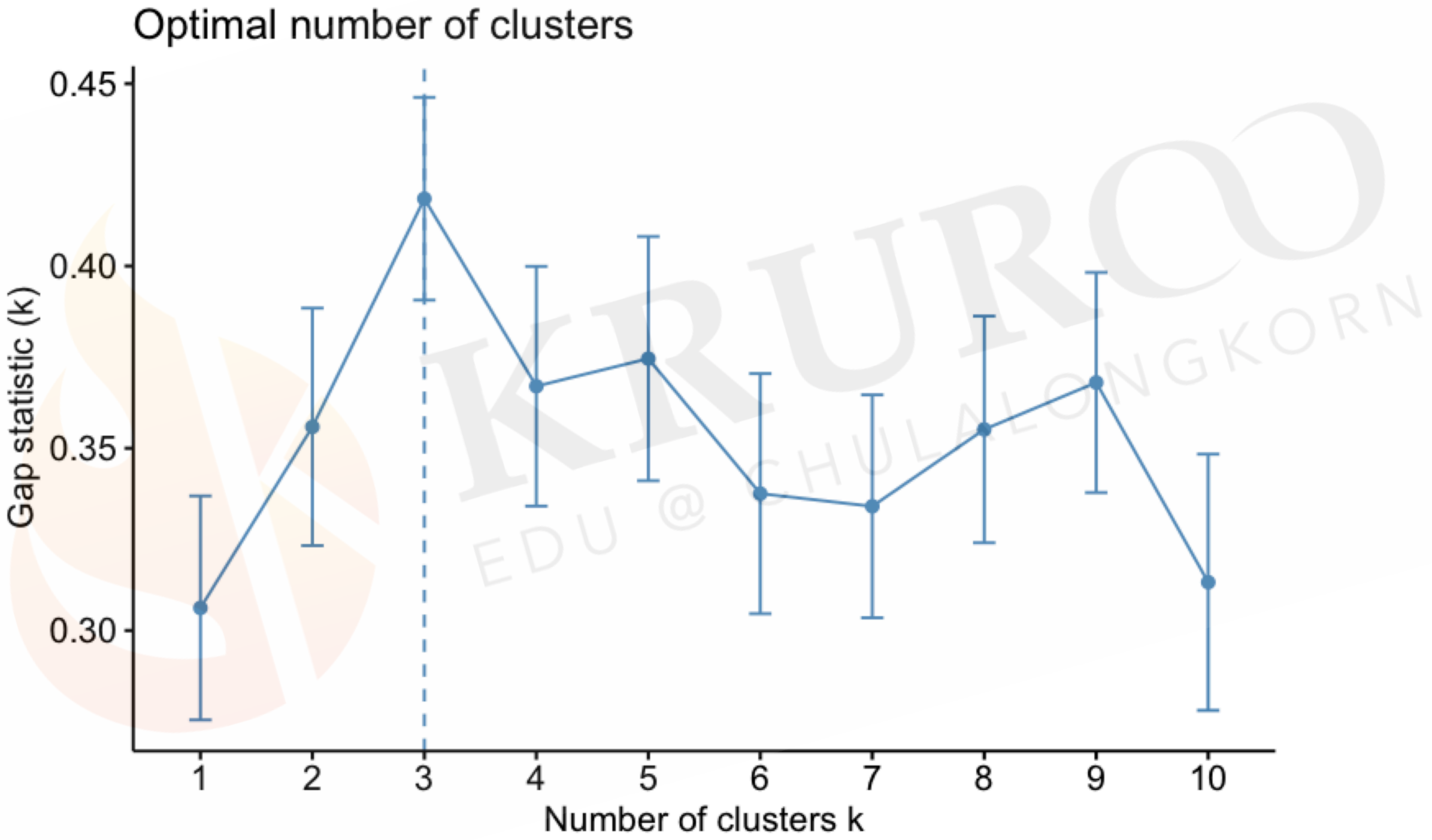
$$Gab(k) \geq Gap(k+1) - s_{k+1}$$

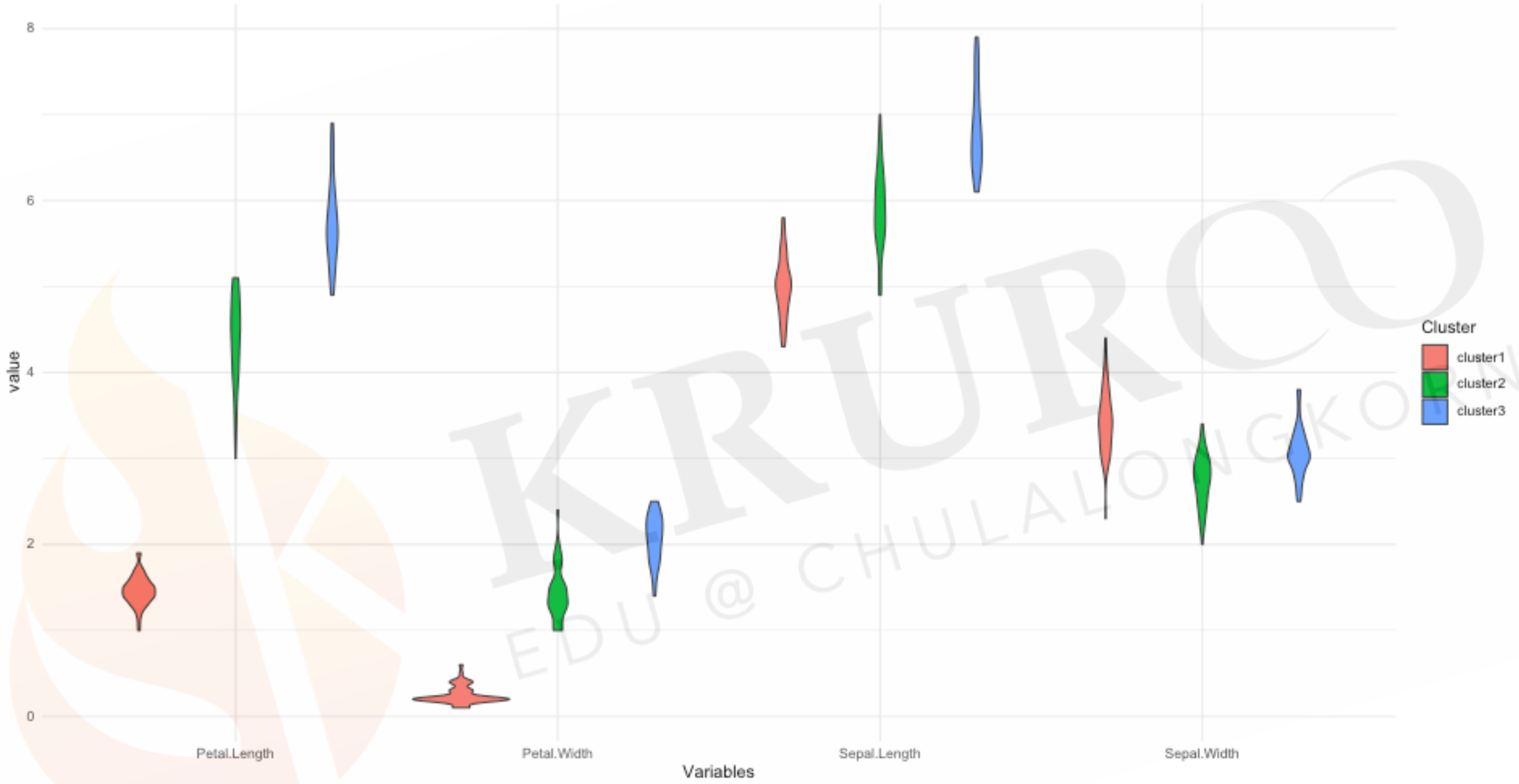
To compute gap statistics we can use

```
gap<-clusGap(x,kmeans,K.max,B)
```

```
clusGap(x=iris[,1:2],kmeans,K.max=10,B=50)
Clustering k = 1,2,..., K.max (= 10): .. done
Bootstrapping, b = 1,2,..., B (= 50)  [one "." per sample]:
.................................................. 50
Clustering Gap statistic ["clusGap"] from call:
clusGap(x = iris[, 1:2], FUNcluster = kmeans, K.max = 10, B = 50)
B=50 simulated reference sets, k = 1..10; spaceH0="scaledPCA"
 --> Number of clusters (method 'firstSEmax', SE.factor=1): 3
          logW   E.logW       gap     SE.sim
 [1,] 3.760434 4.063606 0.3031719 0.02872942
 [2,] 3.353856 3.709839 0.3559831 0.03181575
 [3,] 3.103750 3.522686 0.4189354 0.02814096
 [4,] 2.974516 3.346057 0.3715409 0.02842358
 [5,] 2.834807 3.206930 0.3721228 0.02935773
 [6,] 2.739778 3.098067 0.3582885 0.03575695
 [7,] 2.679592 3.011480 0.3318873 0.03934559
 [8,] 2.610213 2.931722 0.3215091 0.03121702
 [9,] 2.567336 2.860146 0.2928101 0.02832294
[10,] 2.464985 2.801821 0.3368364 0.03197032
```

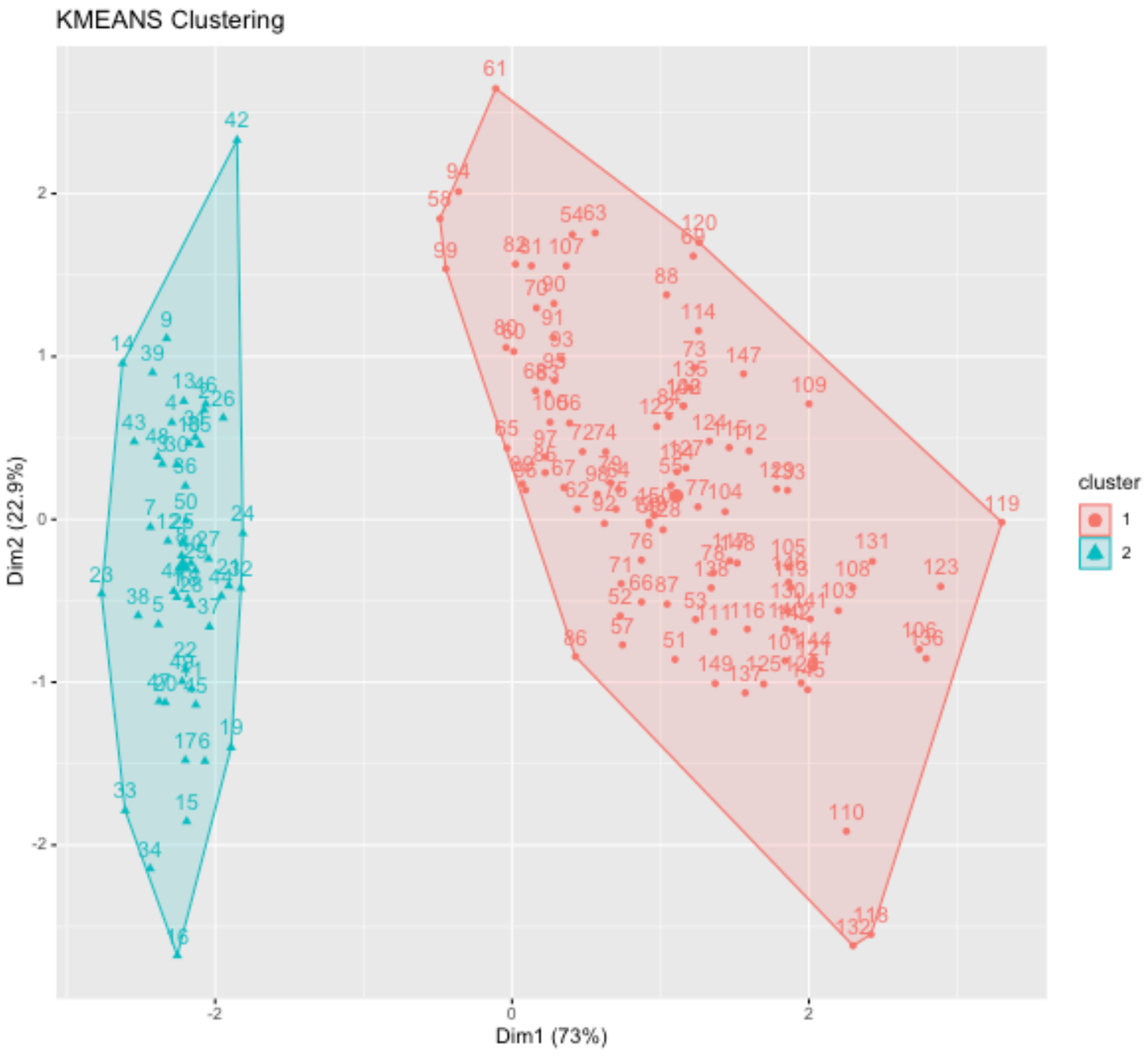
```
> gap<-clusGap(x=iris[,1:2],kmeans,K.max=10,B=50)
> fviz_gap_stat(gap)
```




8
6
4
2
0
value
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Variables
Cluster
cluster1
cluster2
cluster3

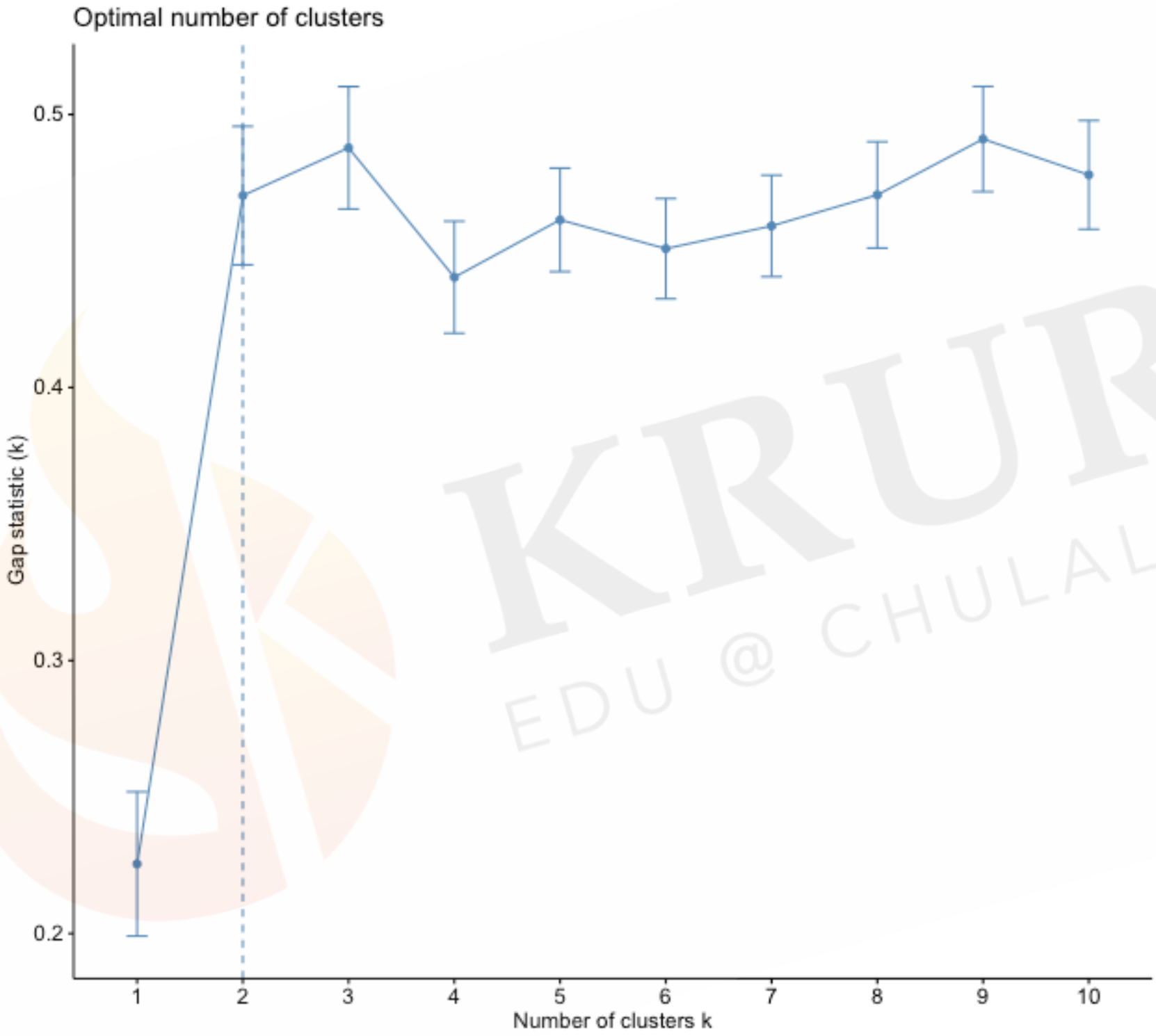
K-means clustering using “eclust”

```
> out.euclid<-eclust(dat, "kmeans", nstart = 25,hc_metric="euclidean")
> out.man<-eclust(dat, "kmeans", nstart = 25,hc_metric="manhattan")
> out.cos<-eclust(dat, "kmeans", nstart = 25,hc_metric="pearson")
```



```
> out.man
K-means clustering with 2 clusters of sizes 100, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1    0.5055957  -0.4252069     0.650315   0.6253518
2   -1.0111914   0.8504137    -1.300630  -1.2507035

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [45] 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [89] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[133] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 173.52867  47.35062
 (between_SS / total_SS =  62.9 %)

Available components:

 [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
 [6] "betweenss"    "size"         "iter"         "ifault"       "clust_plot"
[11] "silinfo"      "nbclust"      "data"         "gap_stat"
```

```
> fviz_gap_stat(out.man$gap_stat)
```



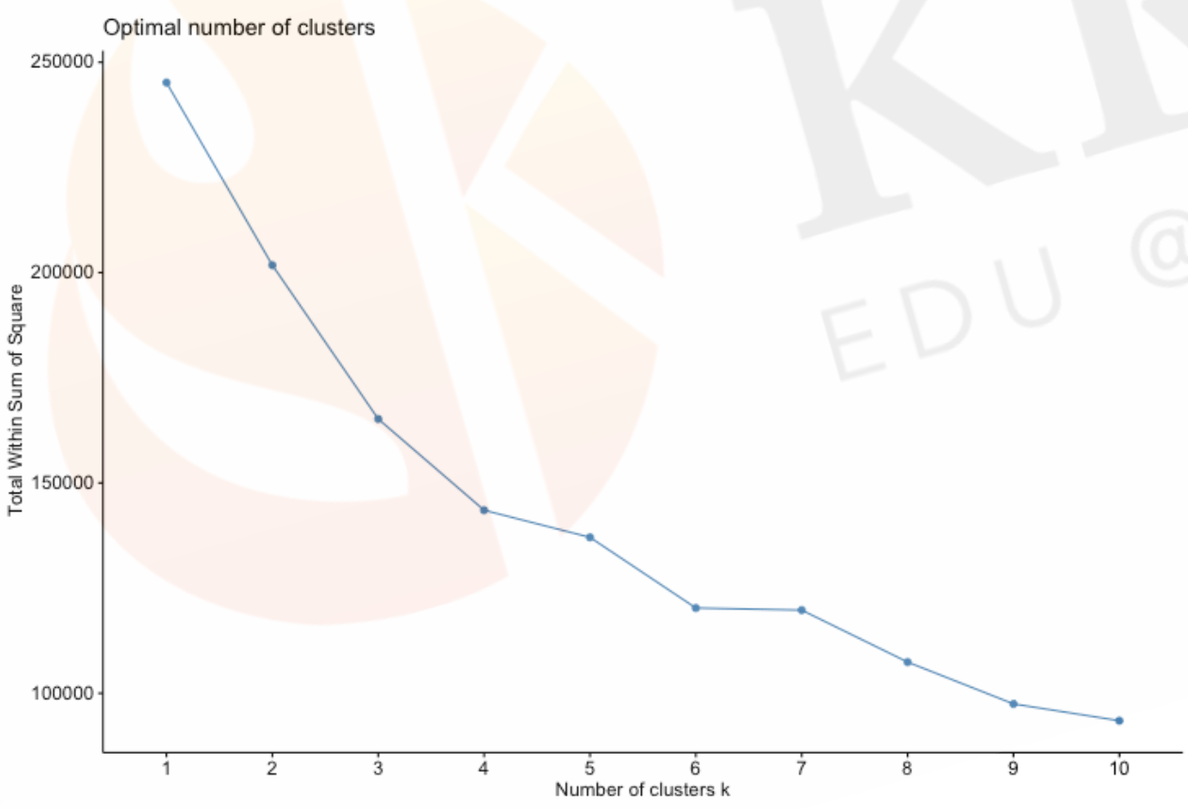
K-means clustering with mixed data

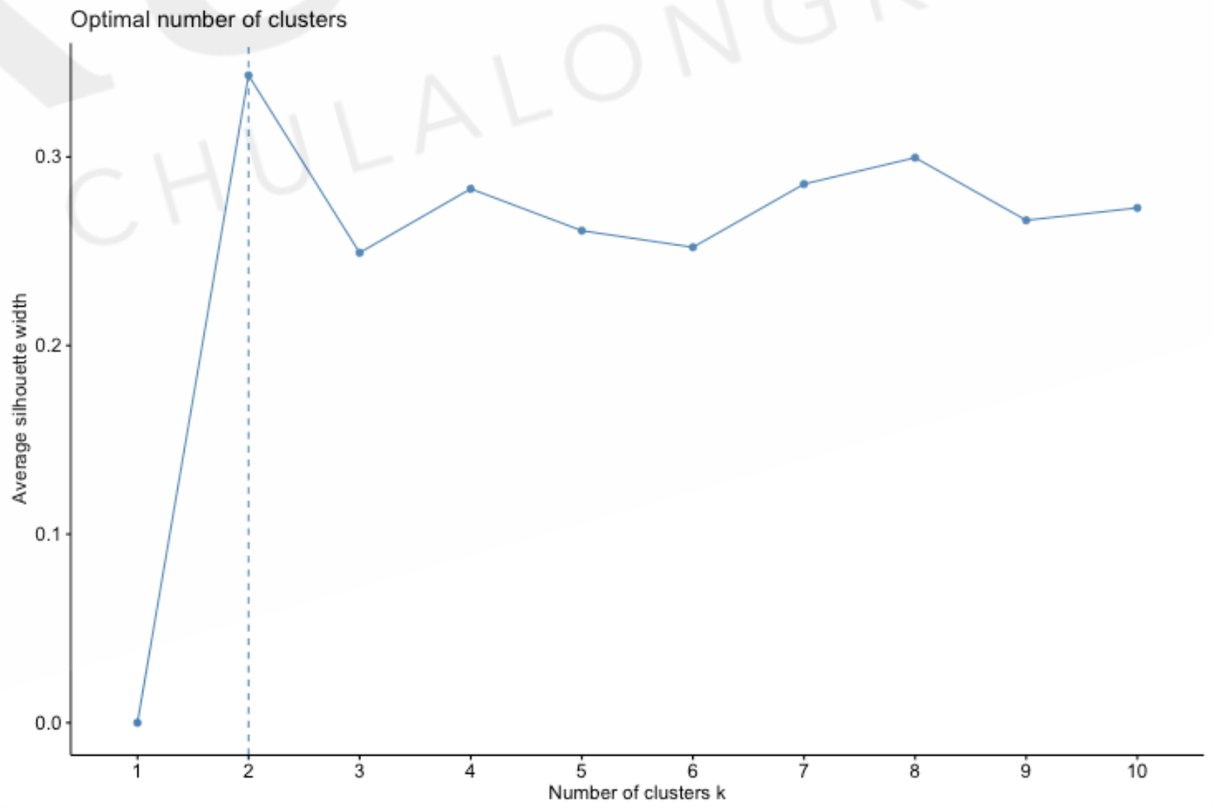
One way to perform k-means clustering on mixed data we can convert any ordinal variables to numeric and convert nominal variable to dummies.

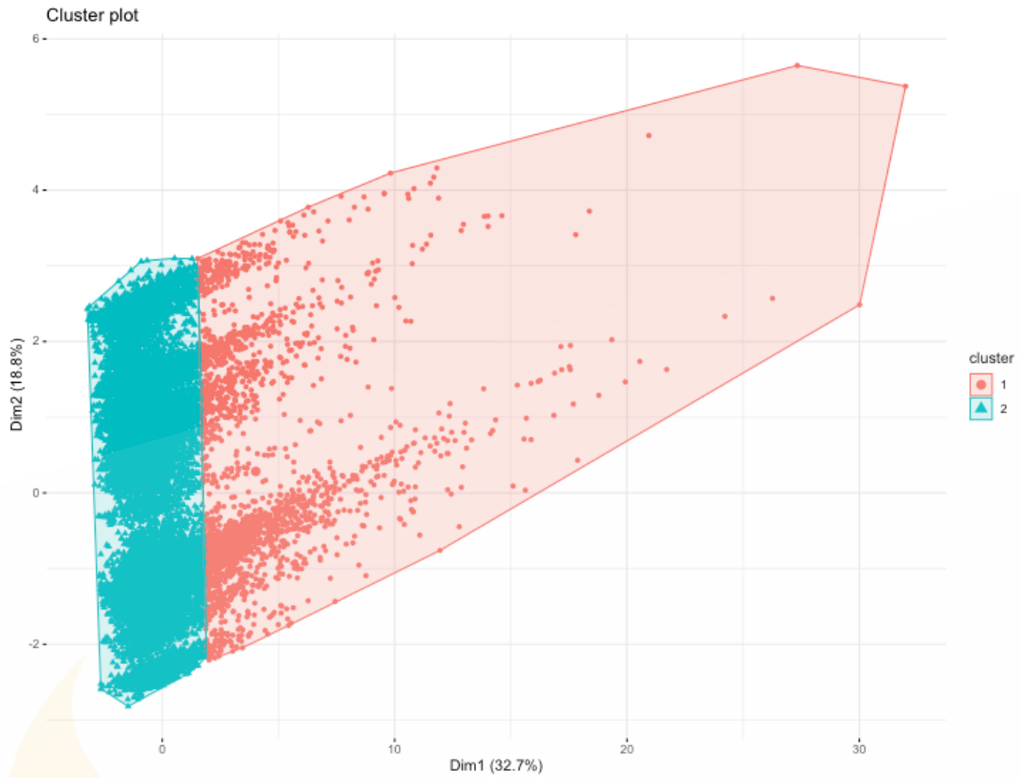
```
> dat<-read.csv("/Users/choat/Documents/2758688 ML/
                ep1_simple regression/dataset/housing.csv")
> glimpse(dat)

 Rows: 20,640
Columns: 10
$ longitude          <dbl> -122.23, -122.22, -122.24, -122.25, -122.25, -122.…
$ latitude           <dbl> 37.88, 37.86, 37.85, 37.85, 37.85, 37.85, 37.84, 3…
$ housing_median_age <dbl> 41, 21, 52, 52, 52, 52, 52, 52, 42, 52, 52, 52, 52…
$ total_rooms        <dbl> 880, 7099, 1467, 1274, 1627, 919, 2535, 3104, 2555…
$ total_bedrooms     <dbl> 129, 1106, 190, 235, 280, 213, 489, 687, 665, 707,…
$ population         <dbl> 322, 2401, 496, 558, 565, 413, 1094, 1157, 1206, 1…
$ households         <dbl> 126, 1138, 177, 219, 259, 193, 514, 647, 595, 714,…
$ median_income      <dbl> 8.3252, 8.3014, 7.2574, 5.6431, 3.8462, 4.0368, 3.…
$ median_house_value <dbl> 452600, 358500, 352100, 341300, 342200, 269700, 29…
$ ocean_proximity    <chr> "NEAR BAY", "NEAR BAY", "NEAR BAY", "NEAR BAY", "N…
```

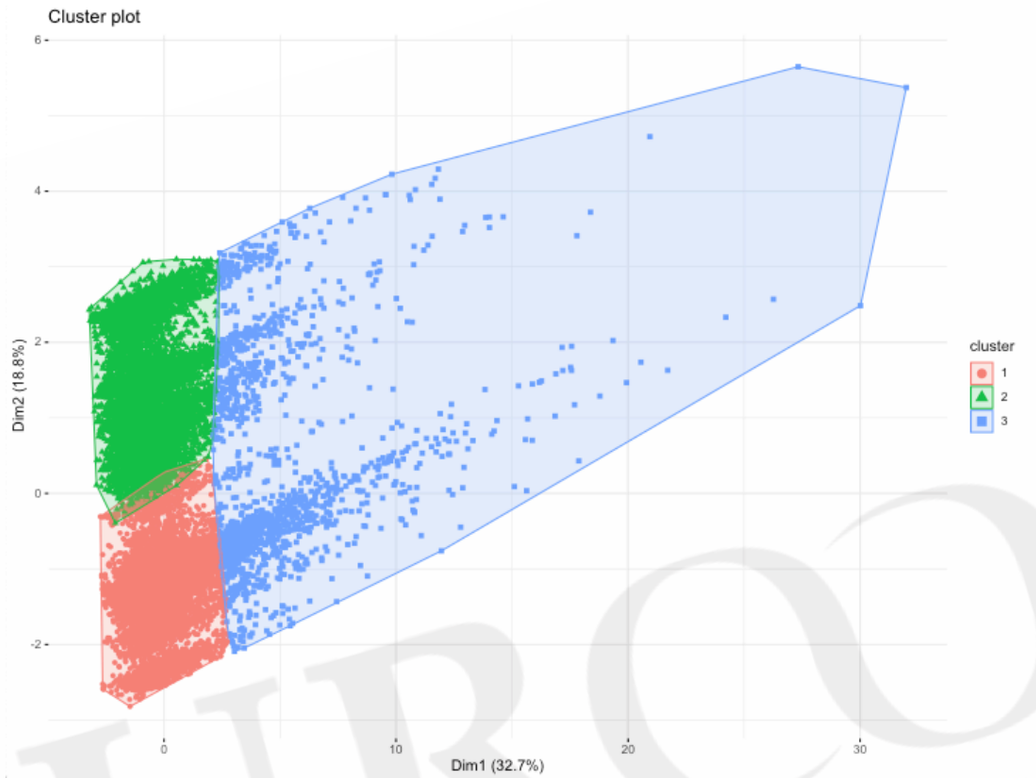
K-means clustering with mixed data

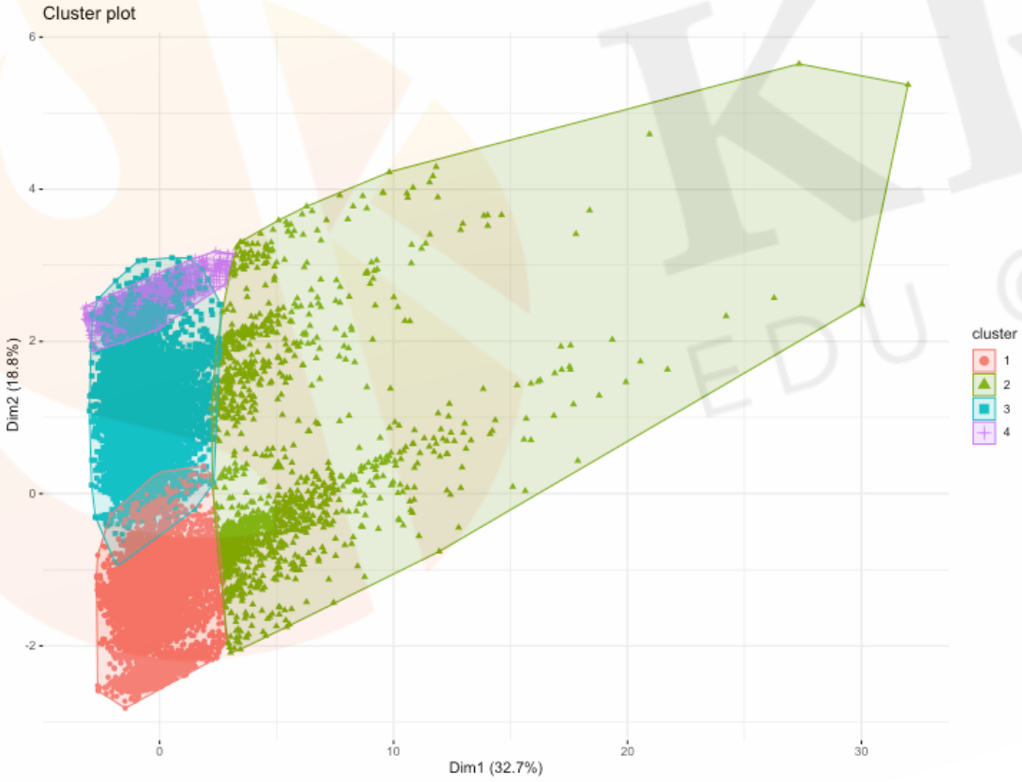
```
> dat<-dat%>%mutate_if(is.character,as.factor)
> dat<-model.matrix(median_house_value~.,data=dat)
> dat<-data.frame(scale(dat[,-1],center=T,scale=T))

>  fviz_nbclust(x=dat,
               kmeans,
               method="wss",
               k.max=10,
               diss=dist(dat,method="euclidean"))
```

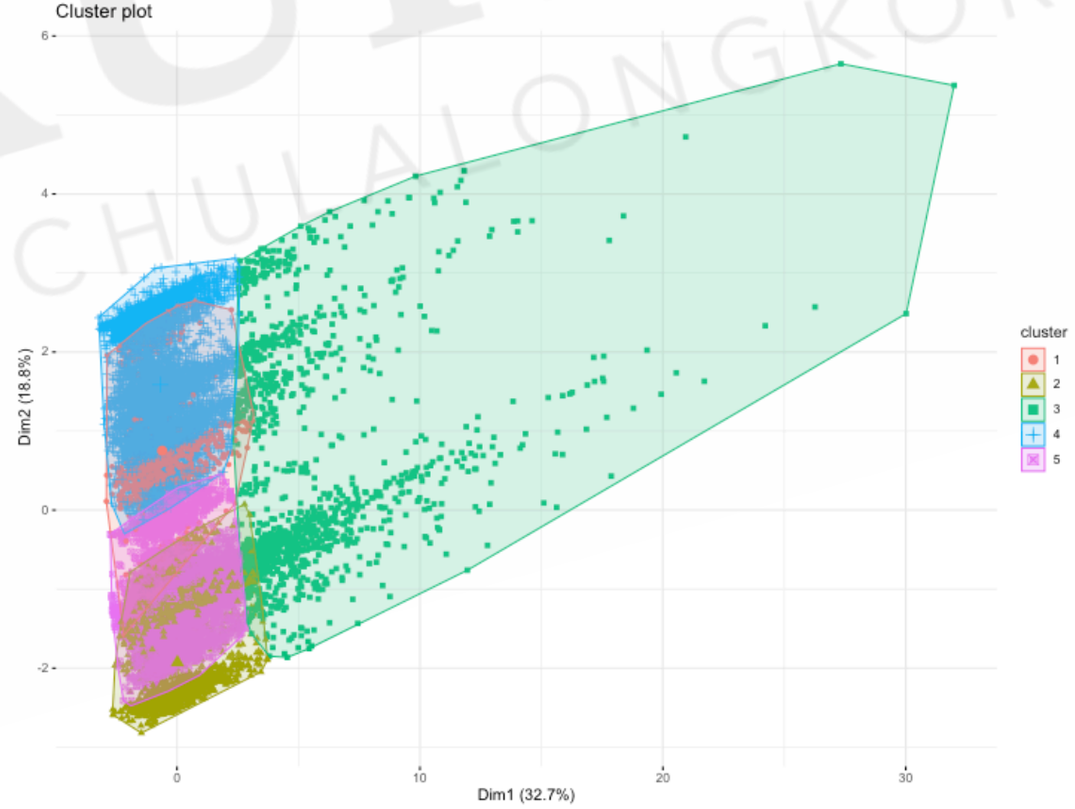





Cluster plot
Dim2 (18.8%)
Dim1 (32.7%)
cluster
1
2


Cluster plot
Dim2 (18.8%)
Dim1 (32.7%)
cluster
1
2
3


Cluster plot
Dim2 (18.8%)
Dim1 (32.7%)
cluster
1
2
3
4


Cluster plot
Dim2 (18.8%)
Dim1 (32.7%)
cluster
1
2
3
4
5

K-means clustering

- Very simple and fast algorithm
- Efficiently deal with very large data sets.
- It requires user to pre-specify the number of clusters.

  Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of clusters.
- Sensitive to outliers and different results can occur if you change the ordering of your data. In this case the **K-medoids** approach is less sensititive to outliers and provides a robust alternative to k-means to deal with these situations

K-medoids clustering

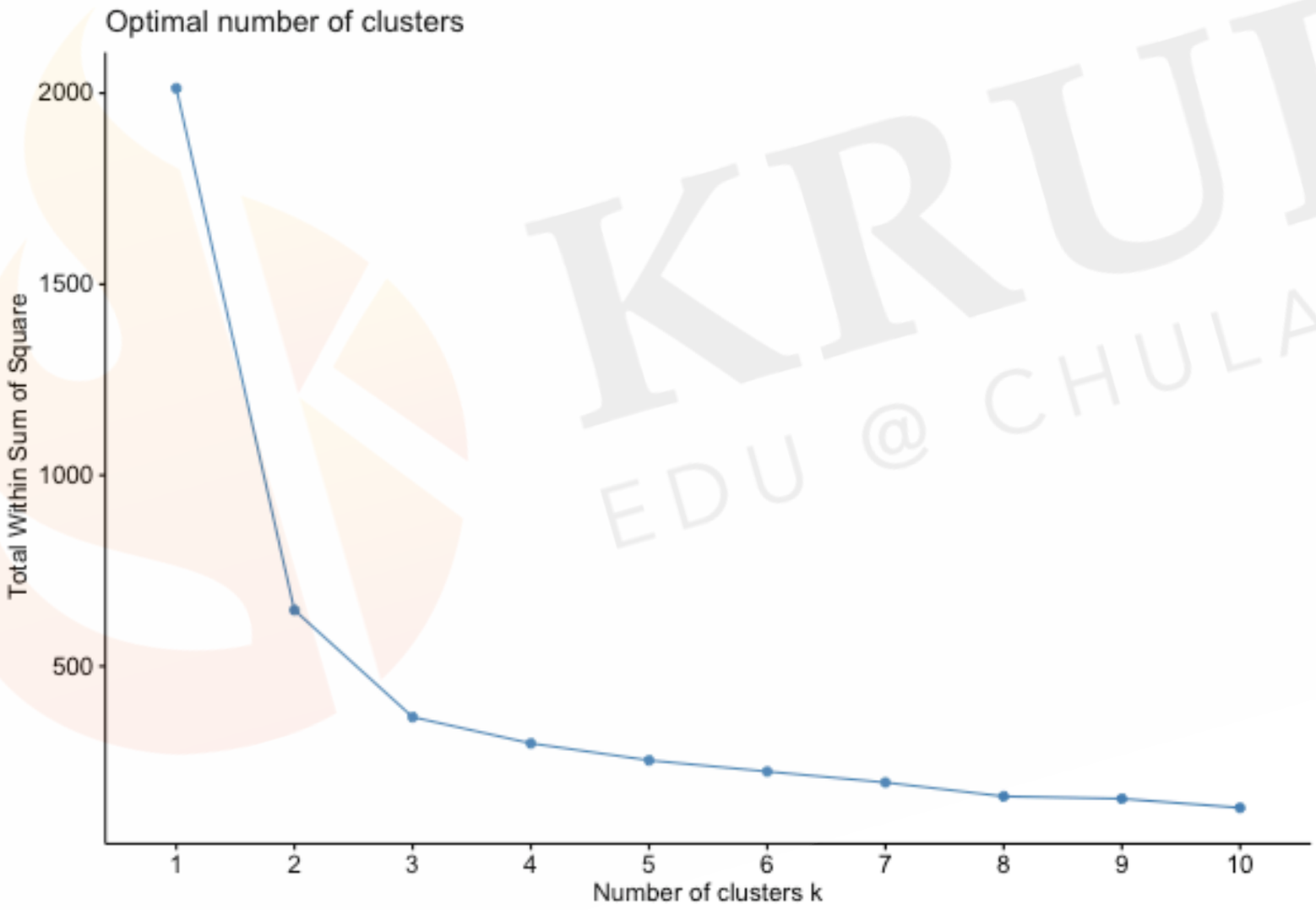
K-medoids clustering

K-medoids has the same algorithm as K-meansbut uses the median rather than mean to determine the centroid.

1. Specify the number of clusters (K) that will be generate in the final solution.
2. Randomly selecting K subjects from the dataset as the initial cluster centers or centroids.
3. Assigned each remaining subjects to their closest centroid, based on euclidean distance between subject and the centroid. (cluster assignment step)
4. For each of the clusters update the cluster centroid by calculating the new mean values of all data point in the cluster. (centroid update step)
5. Iteratively minimise the total within sum squared by iterate step3 and 4 until the cluster assignments stop changing (or maximum number of iteration is reached).

K-medoids clustering

```
> fviz_nbclust(x=dat,
        pam,
        method="wss",
        k.max=10,
        diss=dist(dat,method="manhattan"))
```



K-medoids clustering

```
> out2<-pam(dat,metric="manhattan",k=2)
> out3<-pam(dist.man,diss=T,k=3)
```

```
> out2
Medoids:
      ID Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,]   8   -1.0184372   0.7861738   -1.2791040  -1.3110521
[2,] 134    0.5514857  -0.5903951    0.7602115   0.3944526
Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [45] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [89] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[133] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Objective function:
   build     swap
2.109068 1.785022

Available components:
 [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"  "clusinfo"
 [7] "silinfo"    "diss"       "call"       "data"
```

K-medoids clustering

```
> out3
Medoids:
      ID
[1,]   8   8
[2,] 148 148
[3,]  95  95
Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [45] 1 1 1 1 1 1 2 2 2 3 2 3 2 3 3 3 3 3 3 3 2 3 3 3 3 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 3 2 3
 [89] 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2
[133] 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Objective function:
     build       swap
0.08426868 0.08127864

Available components:
[1] "medoids"    "id.med"     "clustering" "objective"  "isolation"  "clusinfo"
[7] "silinfo"    "diss"       "call"
```

K-modes clustering

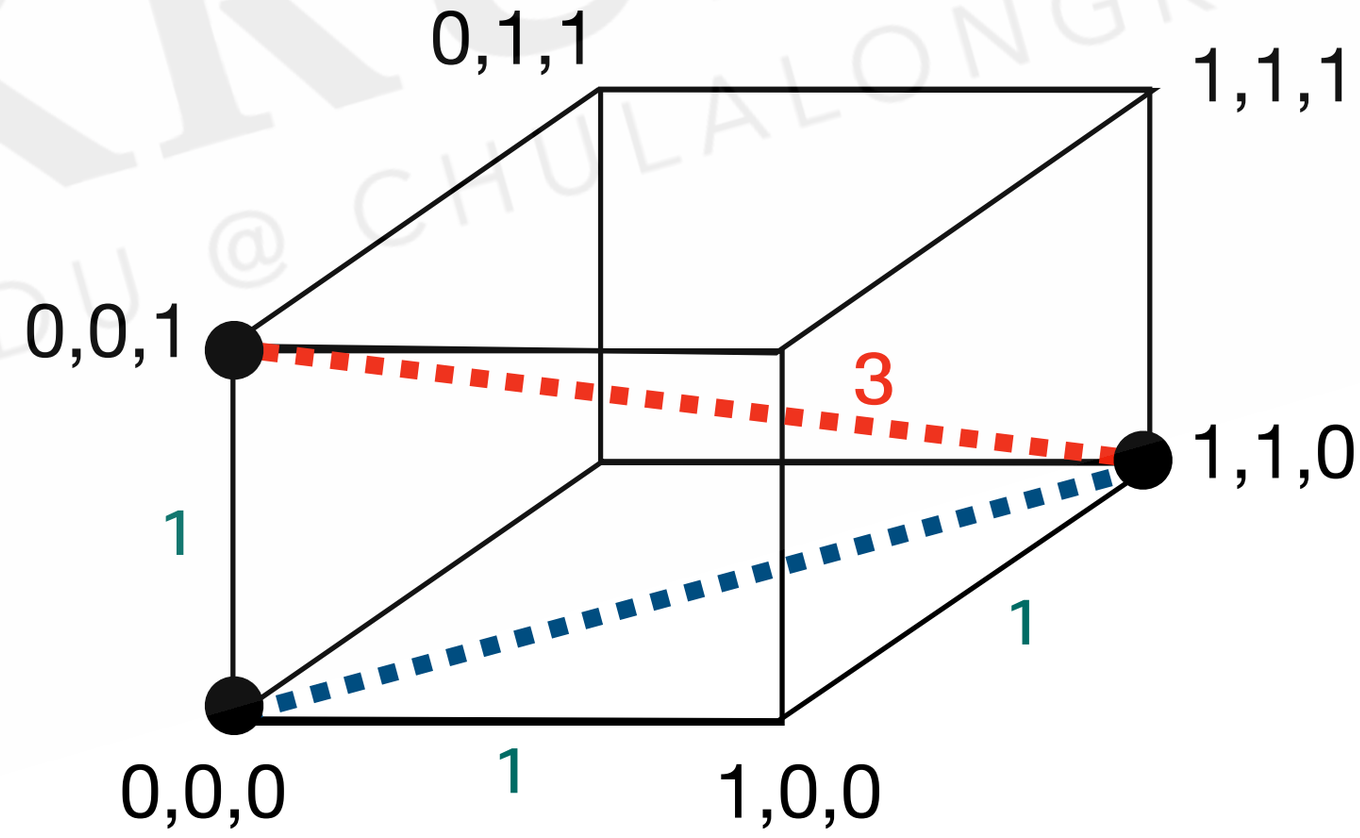
K-modes clustering

When variable in dataset is not possible to measure distance in traditional sense like in nominal variables. In this case we can use K-modes clustering

1. Specify the number of clusters (K) that will be generate in the final solution.
2. Randomly selecting K subjects from the dataset as the initial cluster centers or centroids.
3. Find the **Hamming distance** for each point from center. Assigned each remaining subjects to their closest centroid, based on euclidean distance between subject and the centroid. (cluster assignment step)
4. For each of the clusters update the cluster centroid by calculating the new mean values of all data point in the cluster. (centroid update step)
5. Iteratively minimise the total within sum squared by iterate step3 and 4 until the cluster assignments stop changing (or maximum number of iteration is reached).

Hamming distance

- Hamming distance is a special type of distance that is used for categorical variables.
- Given two points of equal dimensions, hamming distance is defined as the number of coordinates differing from one another.



```r
### k-mode clustering
install.packages("klaR")
library(klaR)

dat<-read.csv("/Users/choat/Desktop/breast_cancer.csv")
glimpse(dat)
dat<-dat[,-1]
kmode<-kmodes(dat,modes=2,iter.max=10)
```

```
K-modes clustering with 2 clusters of sizes 161, 124

Cluster modes:
  X30.39 premeno X30.34 X0.2 no X3  left left_low no.1
1  50-59    ge40  20-24  0-2 no  2 right  left_up   no
2  40-49 premeno  30-34  0-2 no  3  left left_low   no

Clustering vector:
  [1] 1 2 1 1 1 2 1 2 1 2 1 2 1 2 2 2 2 2 1 1 1 1 1 1 2 2 1 2 2 1 1 2 1 1 2 1 1
 [38] 2 2 1 1 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 2 1 1 1 2
 [75] 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 2 1 1 2 2 2 1 1 2 1 1 1
[112] 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 2 2 2 1 1 1 2 2 2 1 2 2
[149] 1 1 2 2 1 1 1 1 2 1 2 2 2 1 1 1 1 1 1 1 1 1 2 1 1 2 2 2 1 1 2 2 2 2 1 2 1
[186] 1 1 1 1 2 2 1 2 1 1 1 2 1 2 1 2 2 1 1 1 2 1 2 1 2 1 2 1 1 1 1 2 1 2 2 1 2
[223] 1 2 1 2 1 1 1 1 1 2 2 1 2 2 2 2 1 1 2 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 1 1
[260] 1 2 1 1 1 2 2 2 2 2 2 2 2 1 1 2 2 1 1 2 2 2 2 1 2 2

Within cluster simple-matching distance by cluster:
[1] 584 438

Available components:
[1] "cluster"    "size"       "modes"      "withindiff" "iterations"
[6] "weighted"
```

Hierarchical clustering

Hierarchical clustering

- Hierarchical clustering is an alternative approach to K-means clustering for identifying groups in dataset.
- In contrast to K-means, hierarchical clustering will create a hierarchy of cluster and therefore **does not require us to pre-specify the number of cluster.**
- Hierarchical clustering haas more advantage over K-means clustering in that its results can be easily visualised using an dendrogram.

Algorithms

- **Agglomerative clustering (AGNES)**
    - Bottom-up algorithm —> each observation is initially considered as a cluster.
    - At each step, the two clusters that are the most similar are combined into a new bigger cluster.
    - The algorithm is repeated until all observations are a member of only one single cluster.
    - The result is a tree called dendogram.
- **Divisive hierarchical clustering (DIANA)**

Agglomerative clustering (AGNES)

- Bottom-up algorithm —> each observation is initially considered as a cluster.
- At each step, the two clusters that are the most similar are combined into a new bigger cluster.
- The algorithm is repeated until all observations are a member of only one single cluster.
- The result is a tree called dendogram.

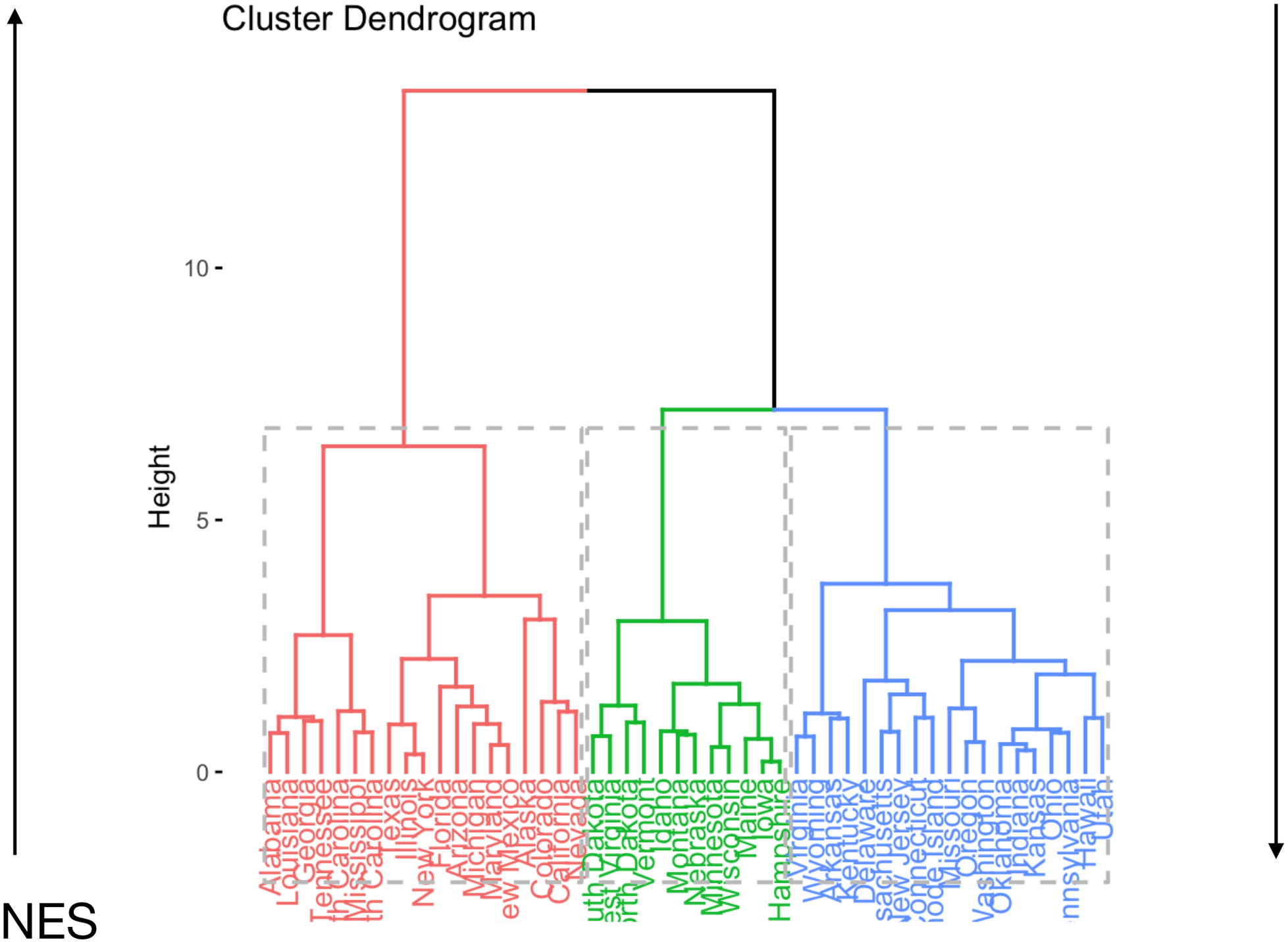
N**ote:** this algorithm is good for small clusters.

Divisive hierarchical clustering (DIANA)

- Top-down algorithm: DIANA is reverse of AGNES.
- It begin with the one big single cluster.
- At each step of algorithm, the current cluster is split into two clusters that are considered most heterogenous.
- The algorithm is repeated until all observations are a member of their own cluster.

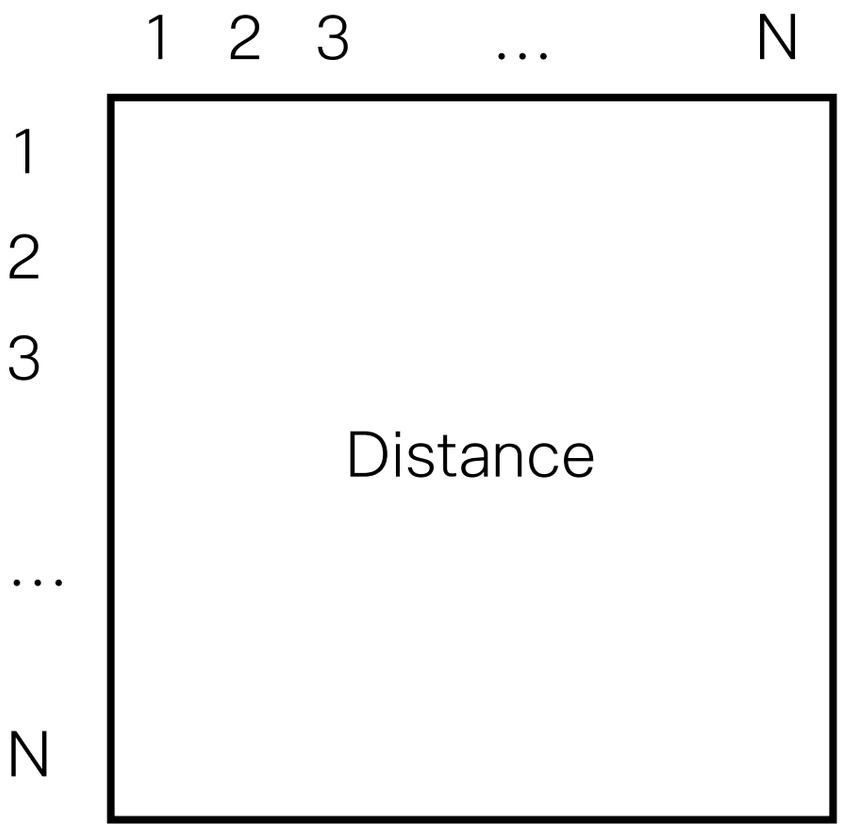
N**ote:** this algorithm is good for large clusters.

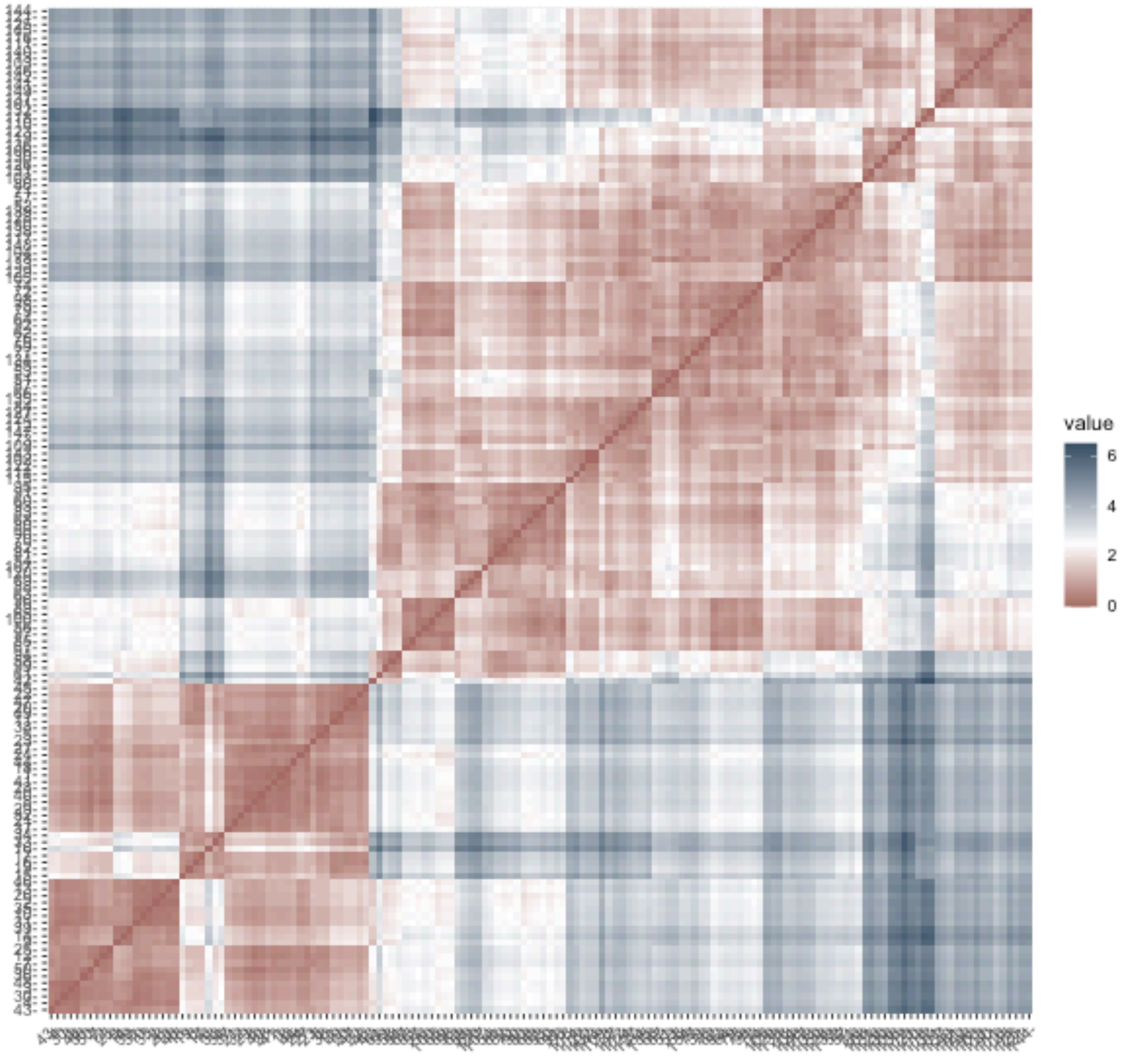
DIANA


Cluster Dendrogram
Height
10
5
0
Alabama
Louisiana
Georgia
Tennessee
South Carolina
Mississippi
North Carolina
Texas
Illinois
New York
Florida
Arizona
Michigan
Maryland
New Mexico
Alaska
Colorado
California
Nevada
South Dakota
West Virginia
North Dakota
Vermont
Idaho
Montana
Nebraska
Minnesota
Wisconsin
Maine
Iowa
New Hampshire
Virginia
Wyoming
Arkansas
Kentucky
Delaware
Massachusetts
New Jersey
Rhode Island
Connecticut
Missouri
Oregon
Washington
Oklahoma
Indiana
Kansas
Ohio
Pennsylvania
Hawaii
Utah

AGNES

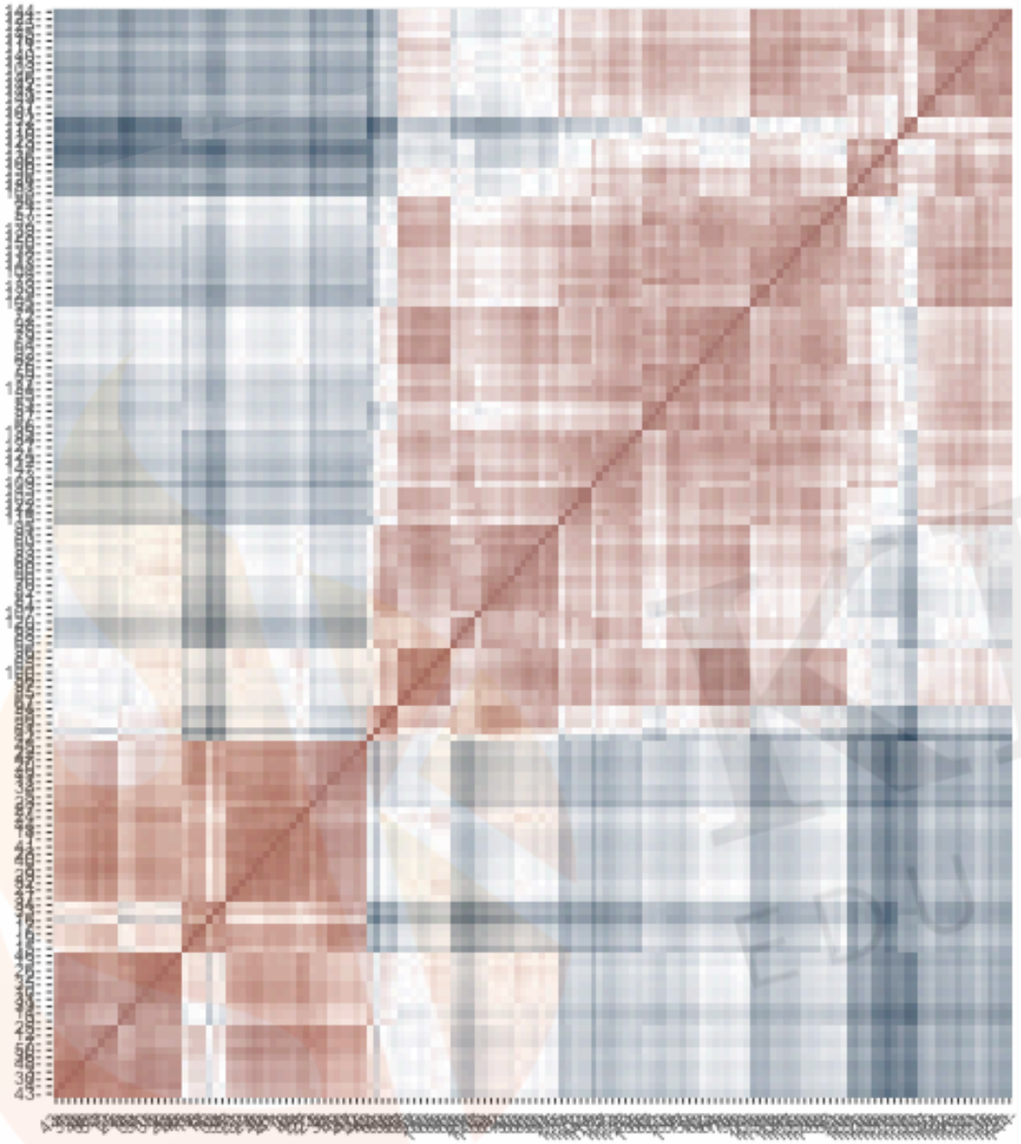
Distance matrix



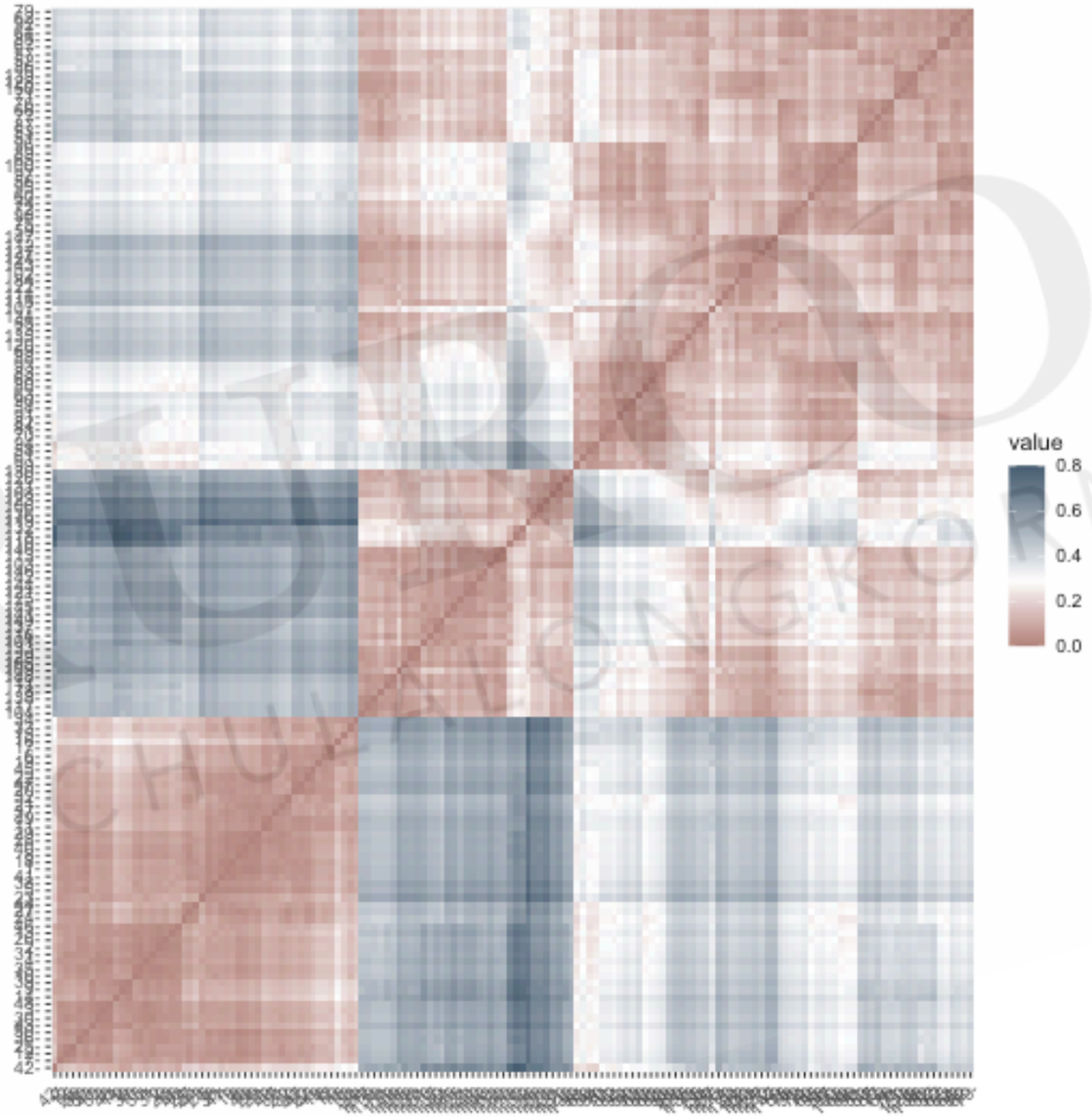



```
> dat<-iris
> dat<-dat[,-5]
> dat<-scale(dat)

> dist.euclid<-dist(dat,method="euclidean")
> dist.man<-dist(dat,method="manhattan")
> dist.cos<-factoextra::get_dist(dat,method="pearson")
> dist.gower<-cluster::daisy(dat,metric="gower")
> dist.gower<-as.dist(dist.gower)
```

Euclidean distance matrix



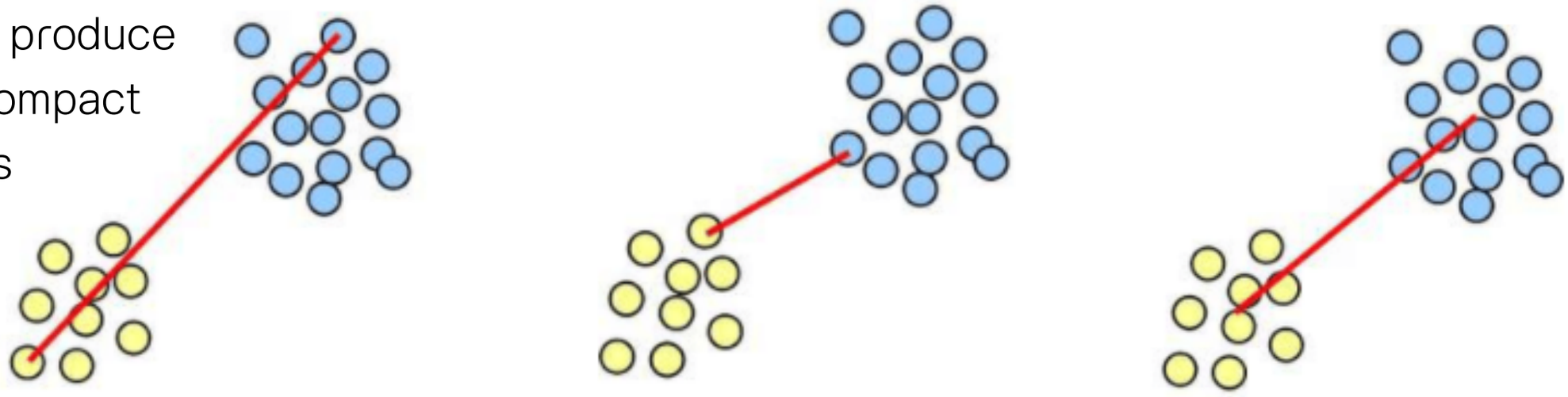
Manhattan distance matrix



How to measure the dissimilarity between two clusters?

- Maximum or complete linkage clustering
- Minimum or single linkage clustering
- Mean or average linkage clustering
- Centroid linkage
- Ward's minimum variance —-> minimised $Total_{withinSS} = \sum_{k=1}^{K} WSS_k$

tend to produce more compact clusters



AGNES using R

There are many function available in R for hierarchical clustering

- `stats::hclust()`
- `cluster::agnes()`

DIANA using R

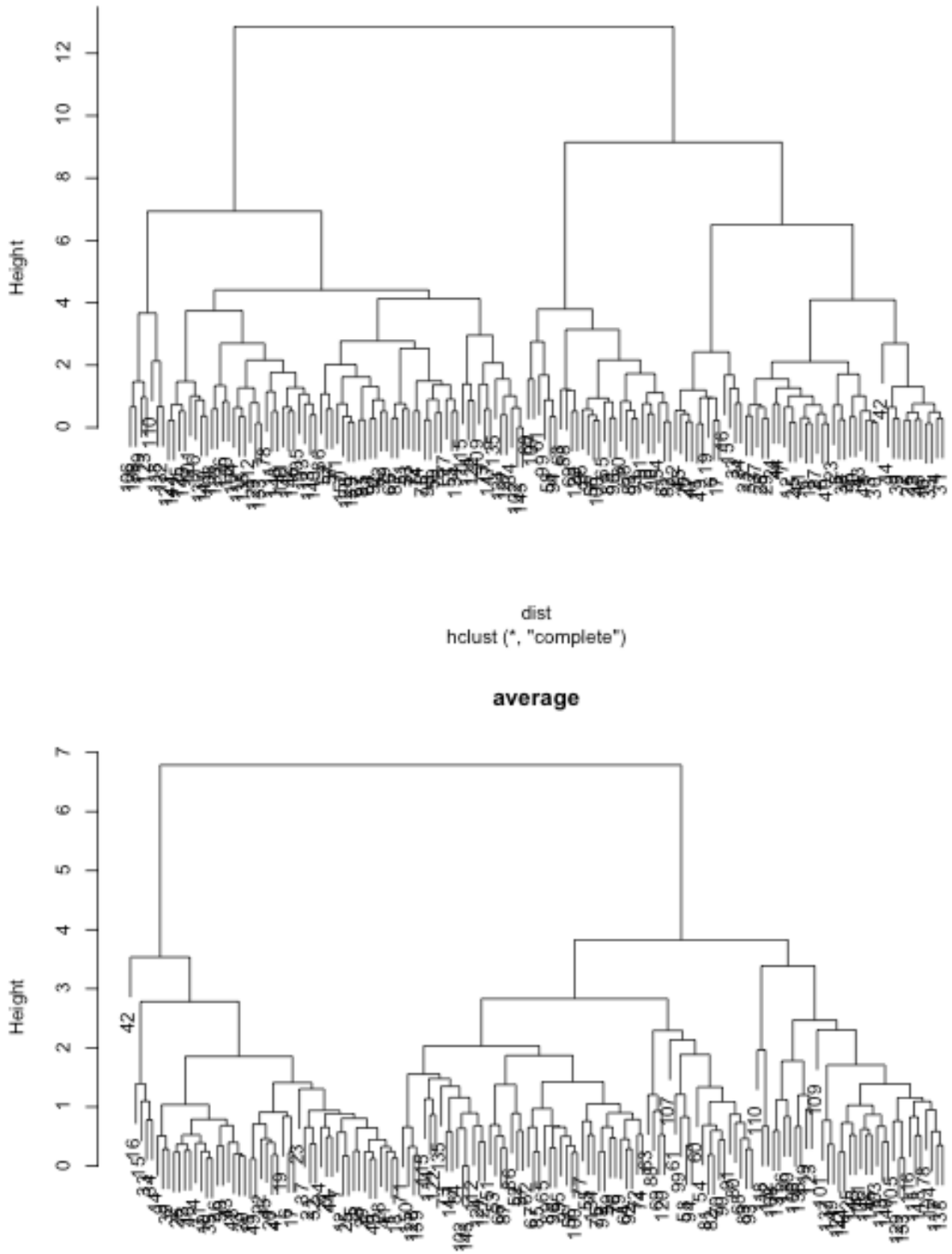
There are many function available in R for hierarchical clustering

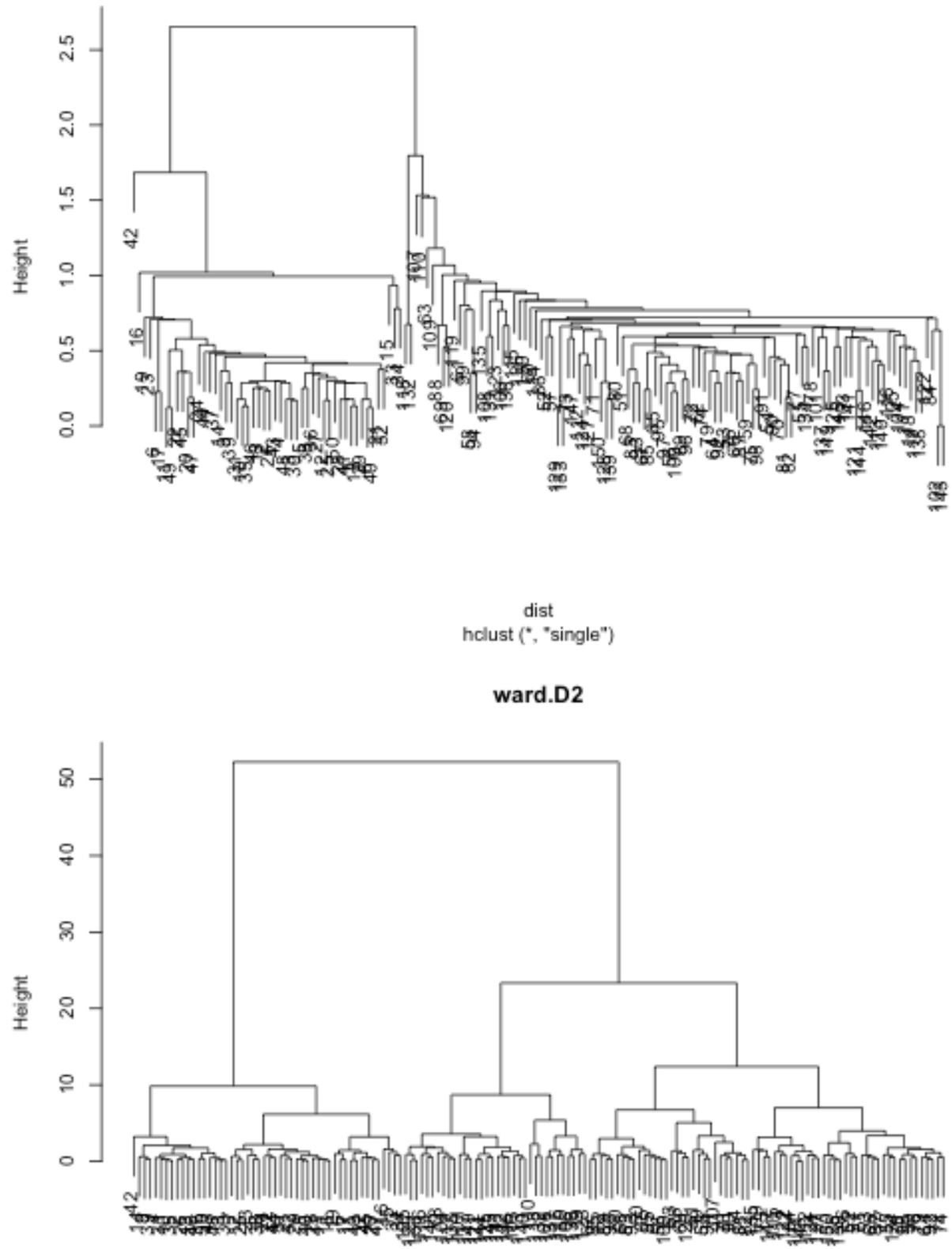
- `cluster::diana()`

Step 1: compute dissimilarity matrix

Step 2: feed the dissimilarity matrix in to `hclust()` or `diana()` and specify the agglomeration method to be used

AGNES

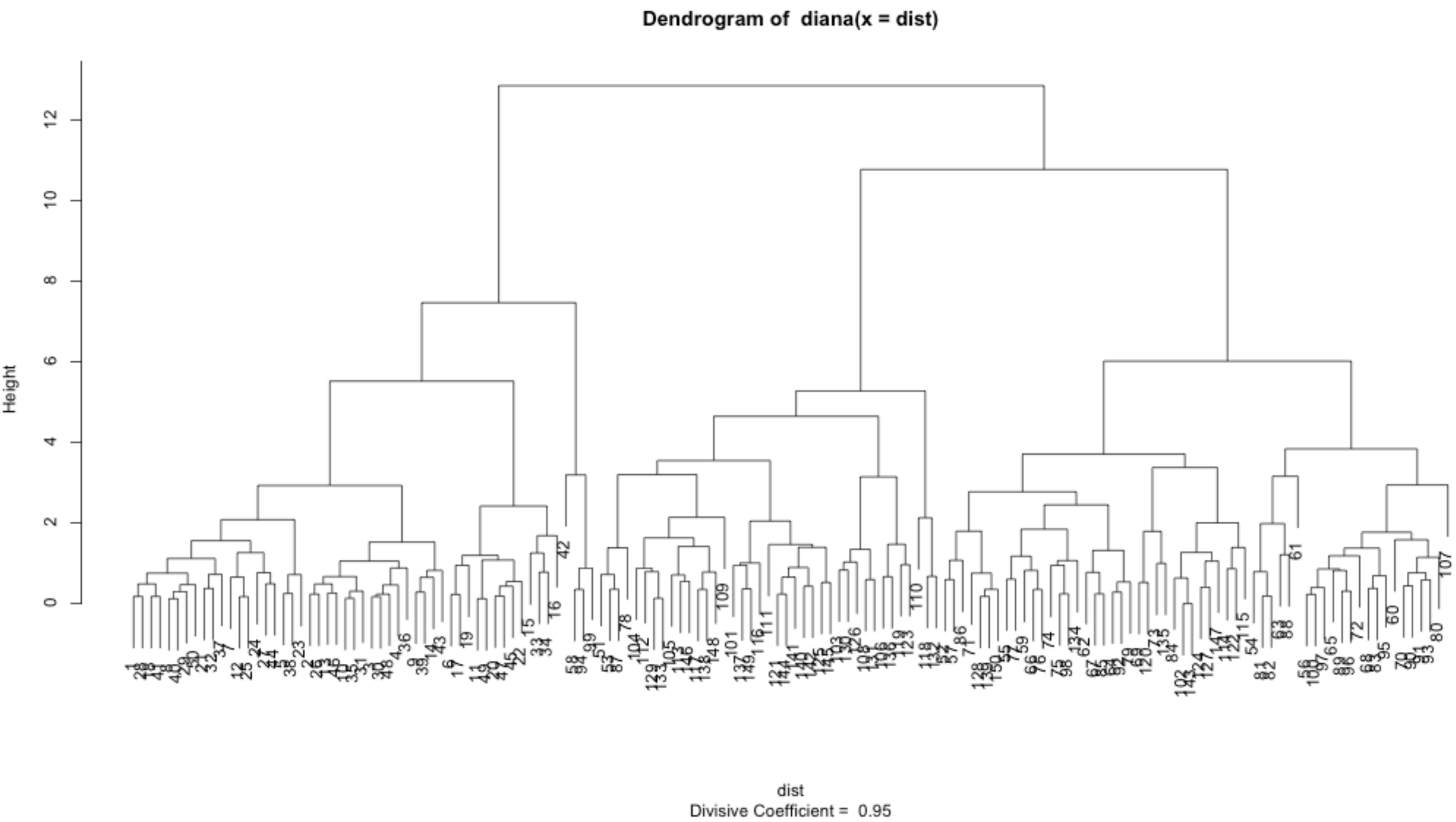

complete
Height
0
2
4
6
8
10
12
dist
hclust (*, "complete")


single
Height
0.0
0.5
1.0
1.5
2.0
2.5
dist
hclust (*, "single")

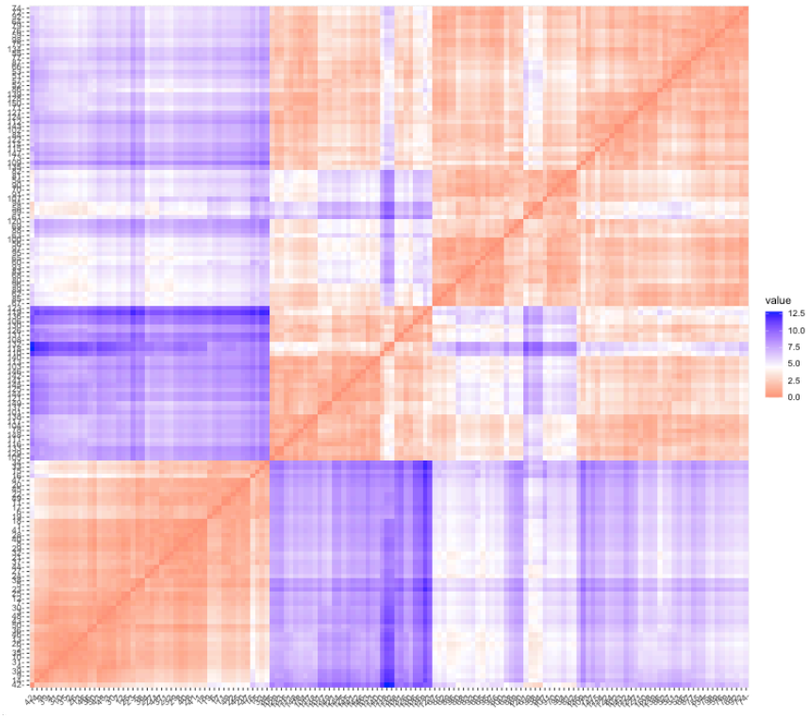
average
Height
0
1
2
3
4
5
6
7
dist
hclust (*, "average")

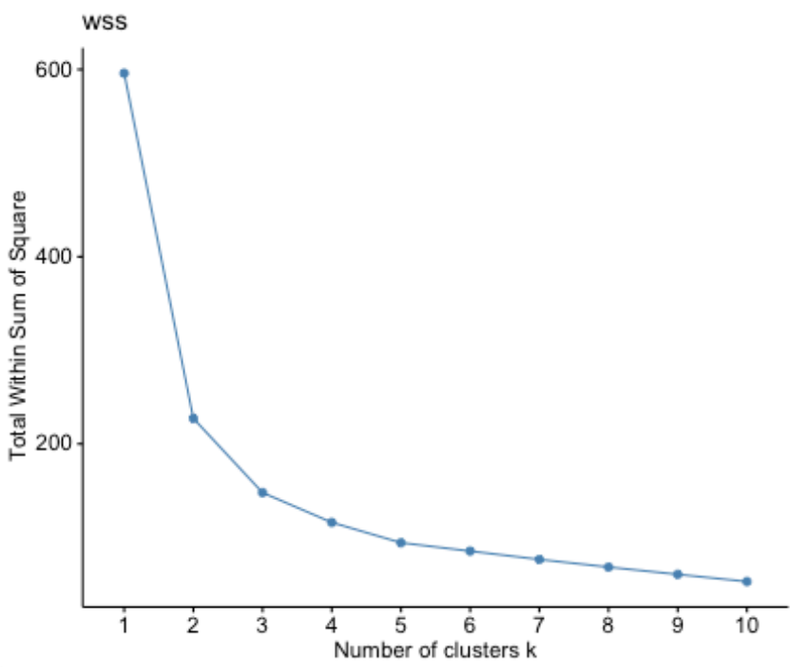
ward.D2
Height
0
10
20
30
40
50
dist
hclust (*, "ward.D2")

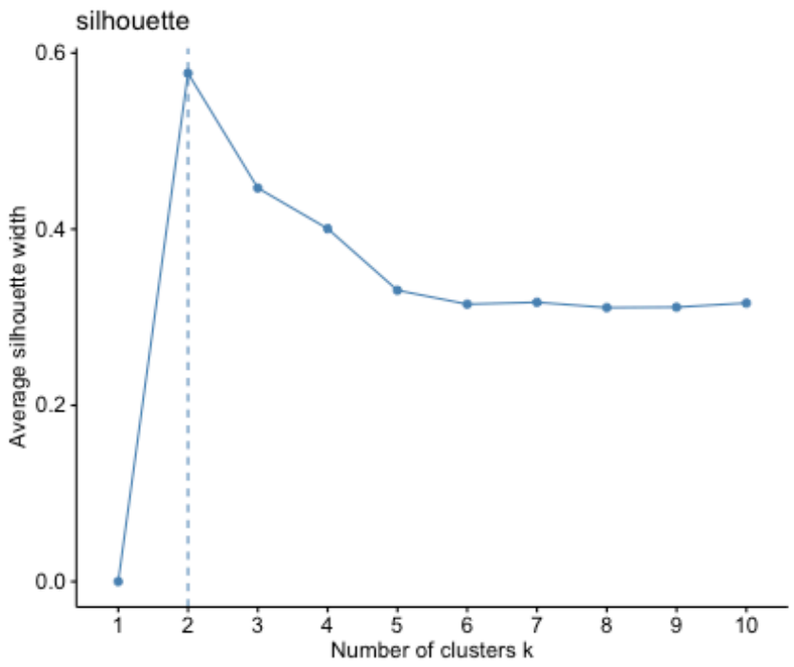
DIANA

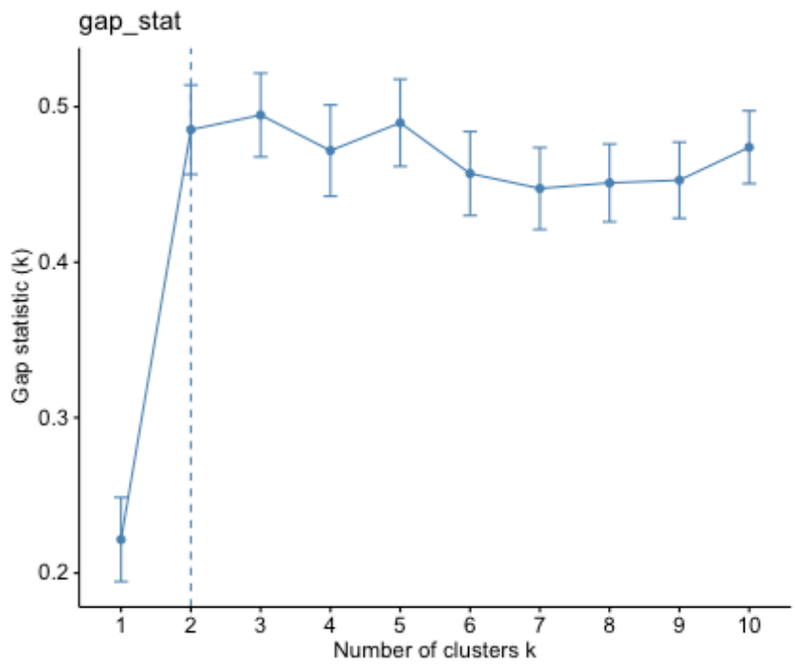

Dendrogram of diana(x = dist)
Height
0
2
4
6
8
10
12
dist
Divisive Coefficient = 0.95

```
> dat<-iris
> dat<-dat[,-5]
> dat<-scale(dat)
> dist<-dist(dat,method="manhattan")
> fviz_dist(dist)
> hc<-hclust(dist,method="complete")
> plot(hc)
> clus3<-cutree(hc,k=3)
> table(clus3,iris$Species)
```

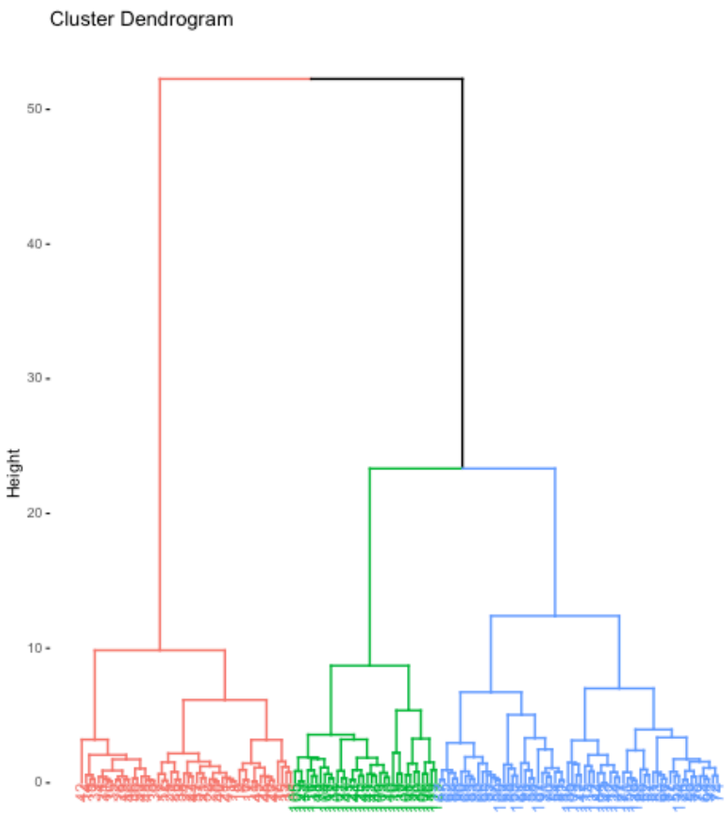


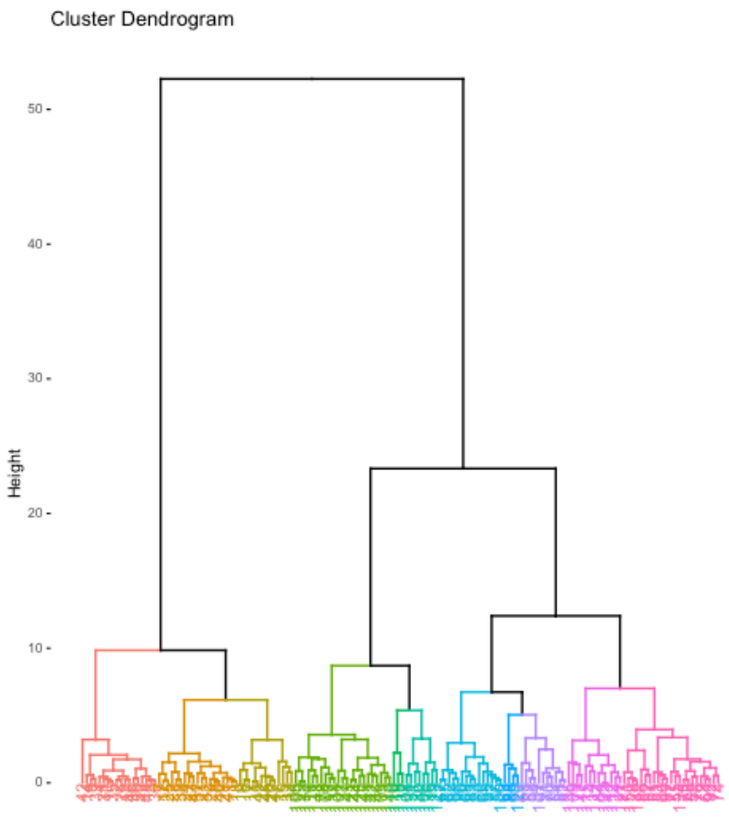


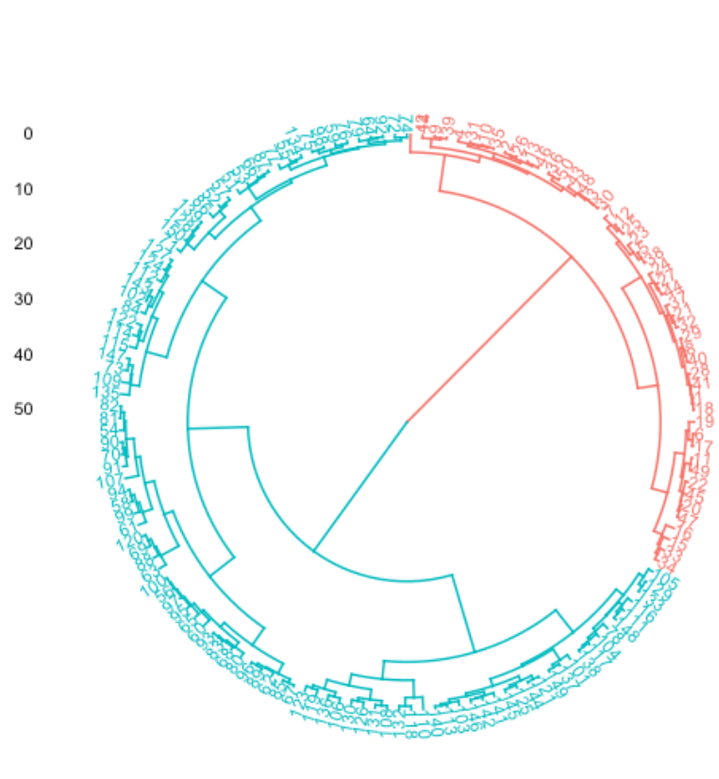




```r
hc1<-hclust(dist,method="complete")
hc2<-hclust(dist,method="single")
hc3<-hclust(dist,method="average")
hc4<-hclust(dist,method="centroid")
hc5<-hclust(dist,method="ward.D2")


p1<-fviz_dend(hc5,k=3)
p2<-fviz_dend(hc5,h=5)
p3<-fviz_dend(hc5,k=2,type="circular")
grid.arrange(p1,p2,p3,nrow=1)
```

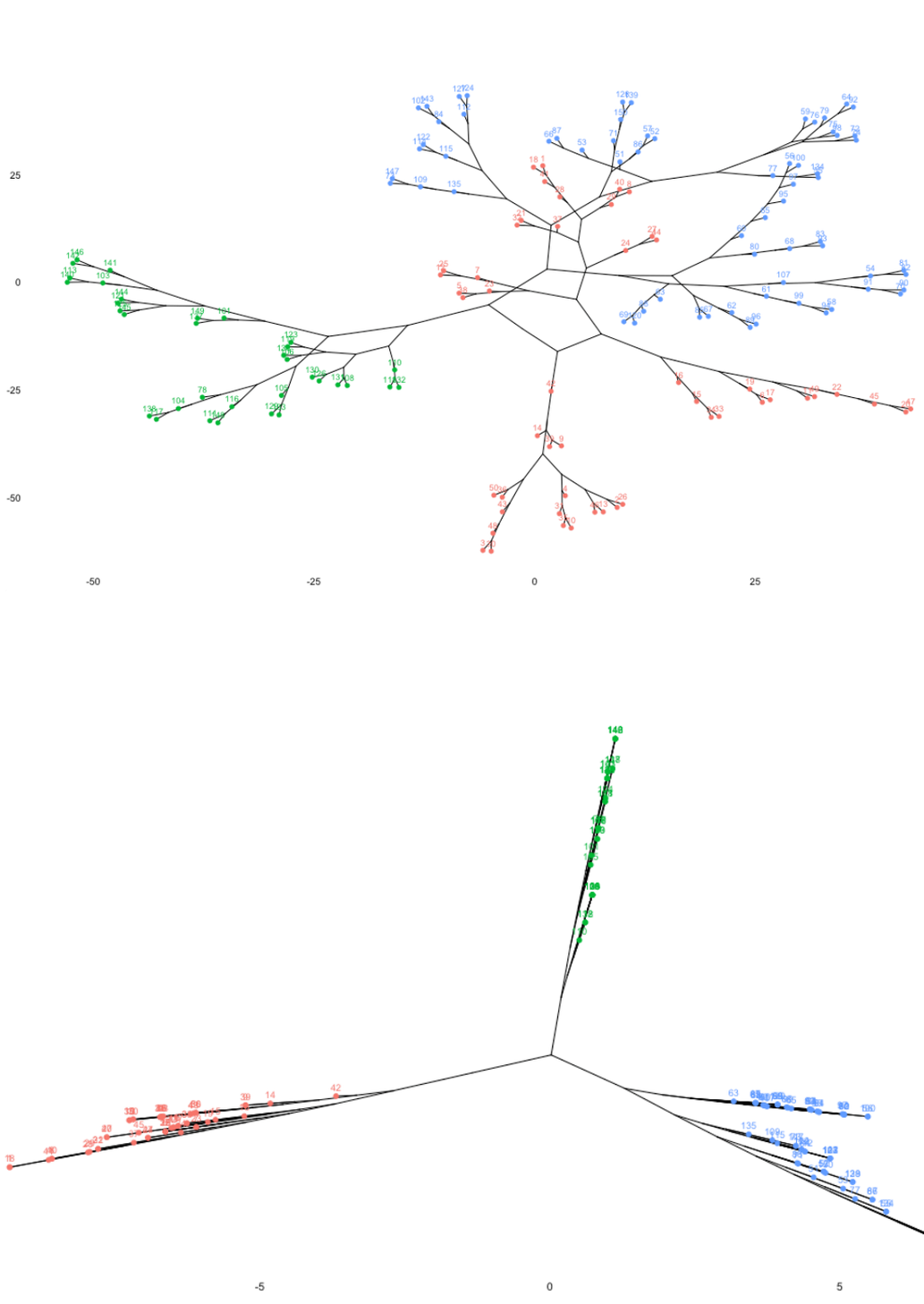


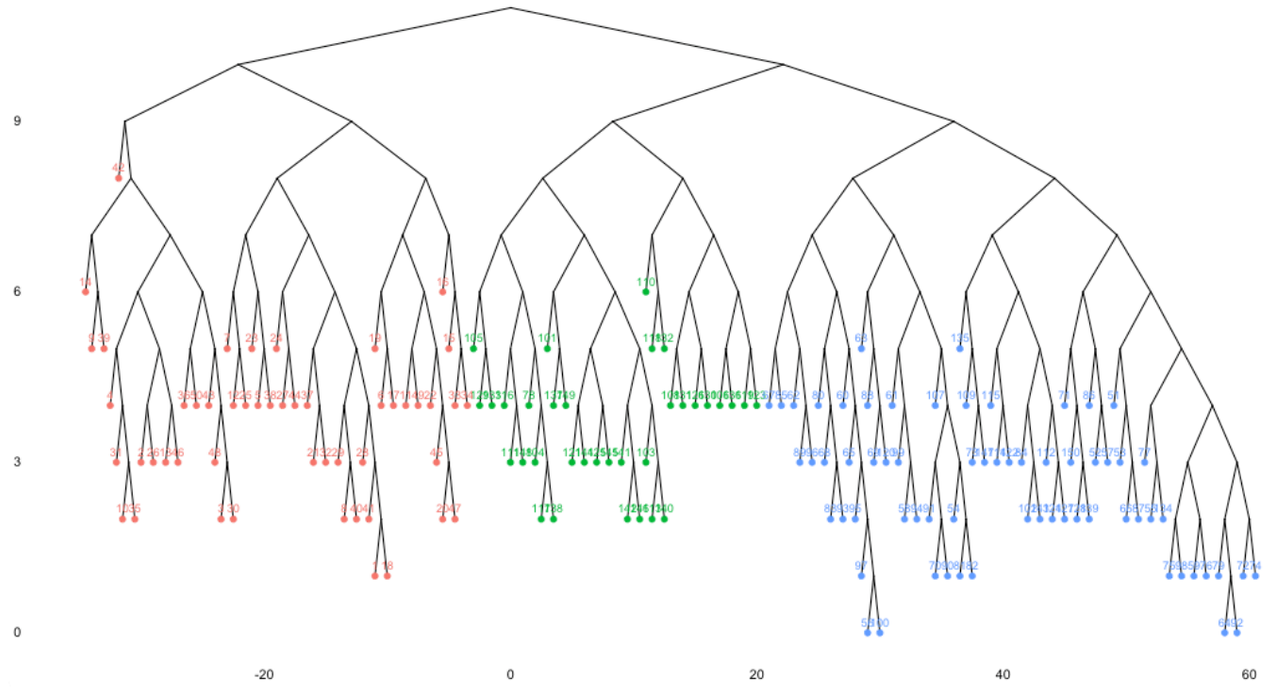


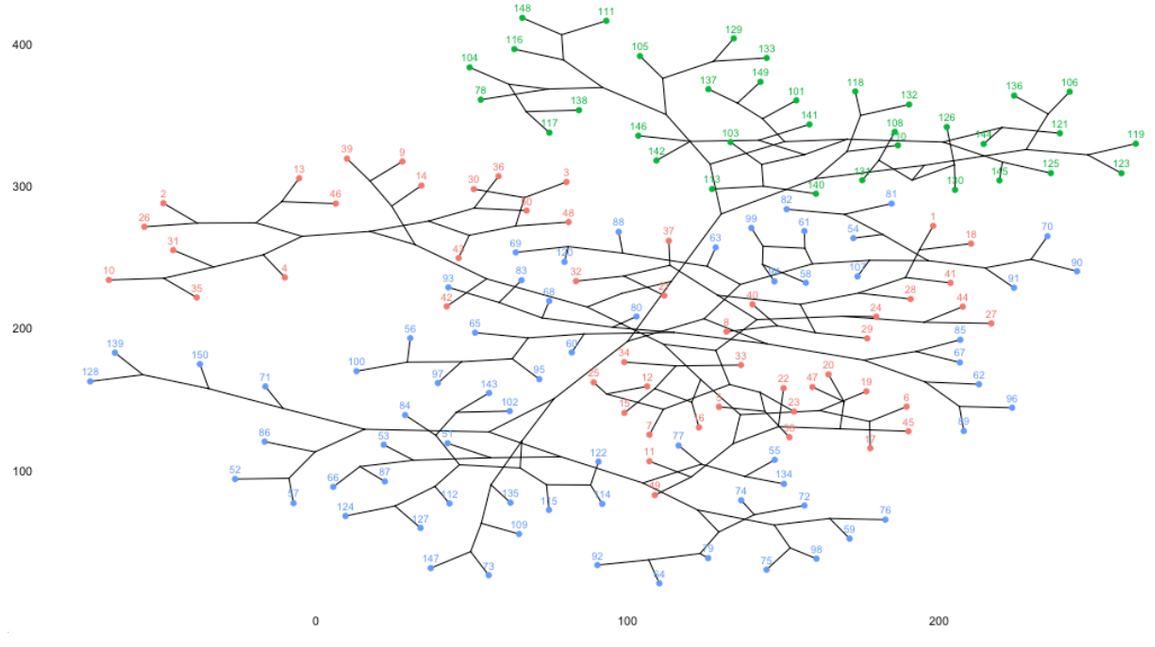


type type of plot. Allowed values are one of "rectangle", "triangle", "circular", "phylogenic".

phylo_layout the layout to be used for phylogenic trees. Default value is "layout.auto". Allowed values include: `layout.auto`, `layout_with_drl`, `layout_as_tree`, `layout.gem`, `layout.mds` and `layout_with_lgl`.



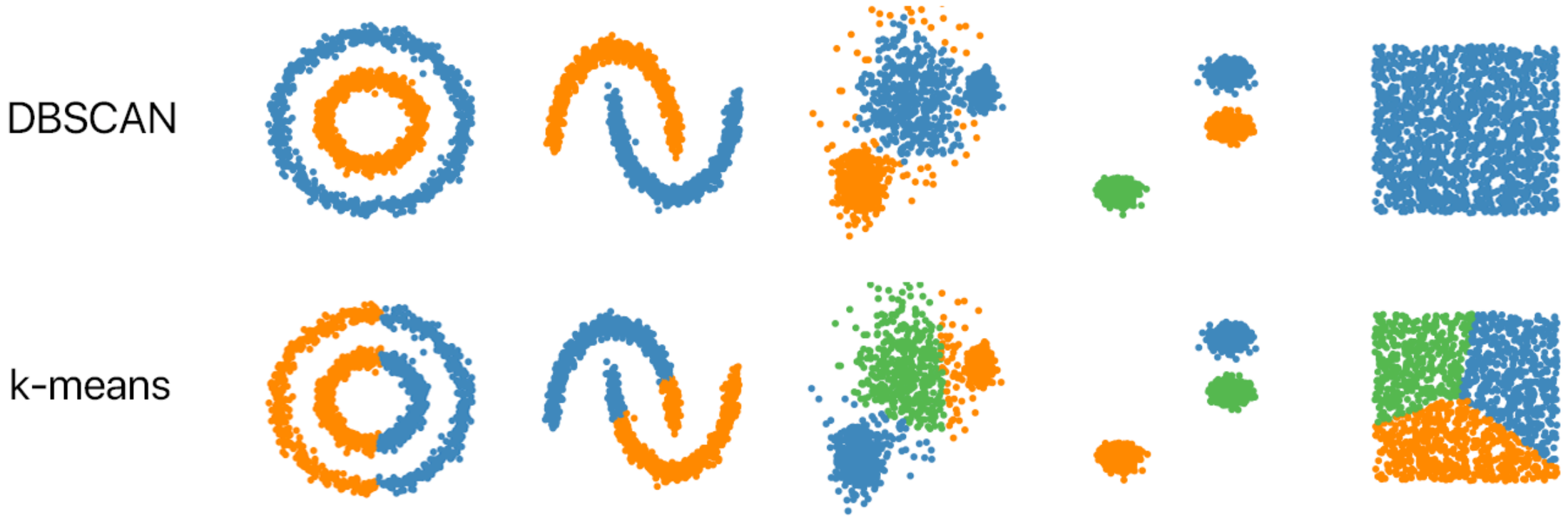




Density Based Clustering (DBSCAN)

Density Based Clustering

- DBSCAN doesn't require to define number of cluster.
- It can automatically detect the number of clusters based on your input data and parameters.
- More importantly, DBSCAN can find arbitrary shape clusters that k-means are not able to find

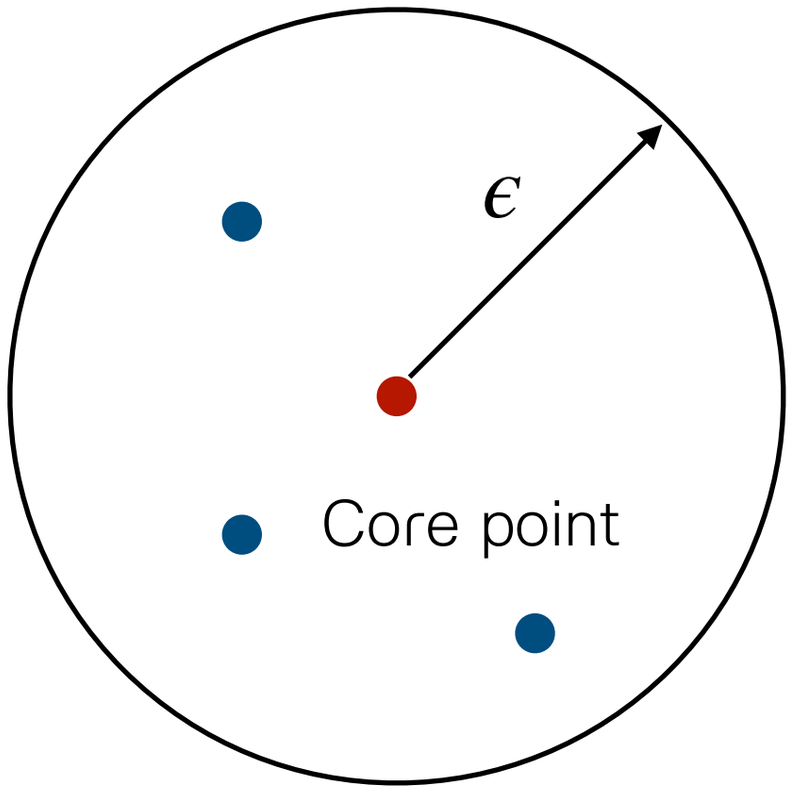


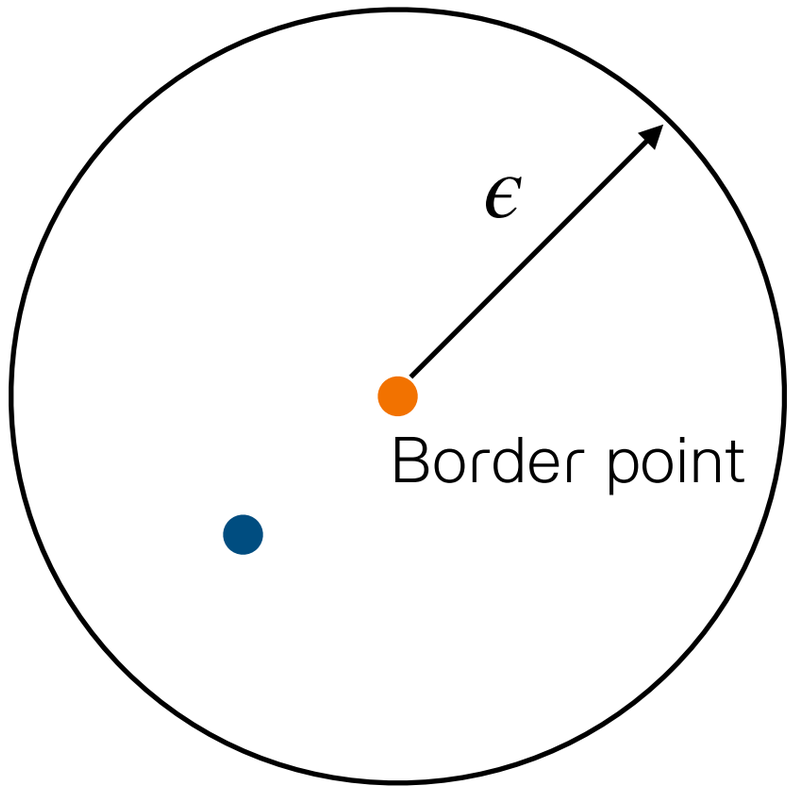
https://github.com/NSHipster/DBSCAN

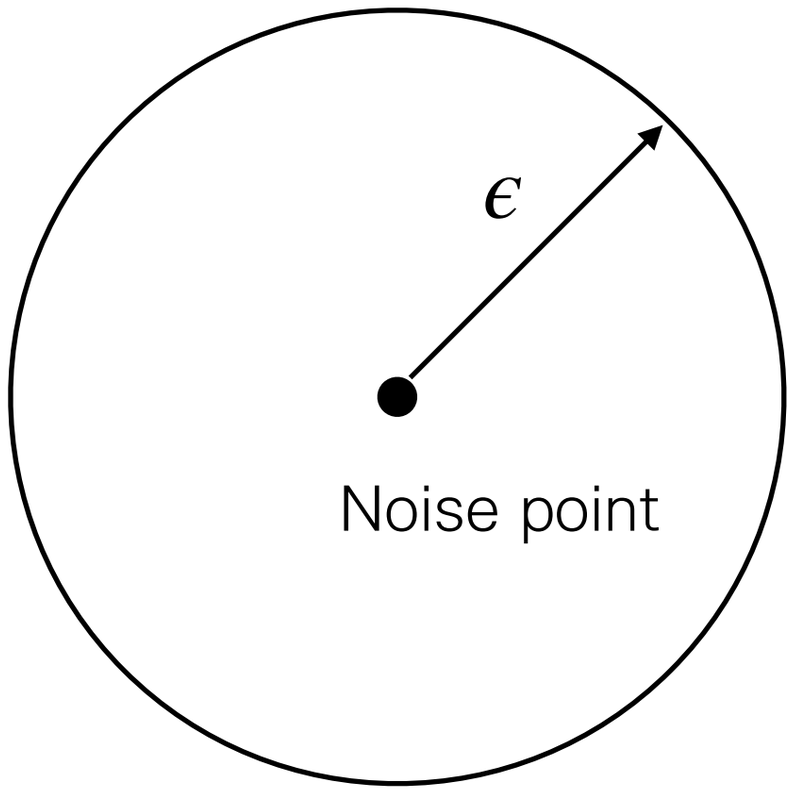
Density Based Clustering

- DBSCAN can handle noise and outliers. All the outliers will be identified and marked without been classified into any cluster.
- Therefore, DBSCAN can also be used for Anomaly Detection (Outlier Detection)
- DBSCAN requires to choose two parameters to perform,
    - the first is ϵ, which denotes maximum distance between two points in the same cluster.
    - Another parameter is is the minimum number of points in the cluster. —-> `minPts`

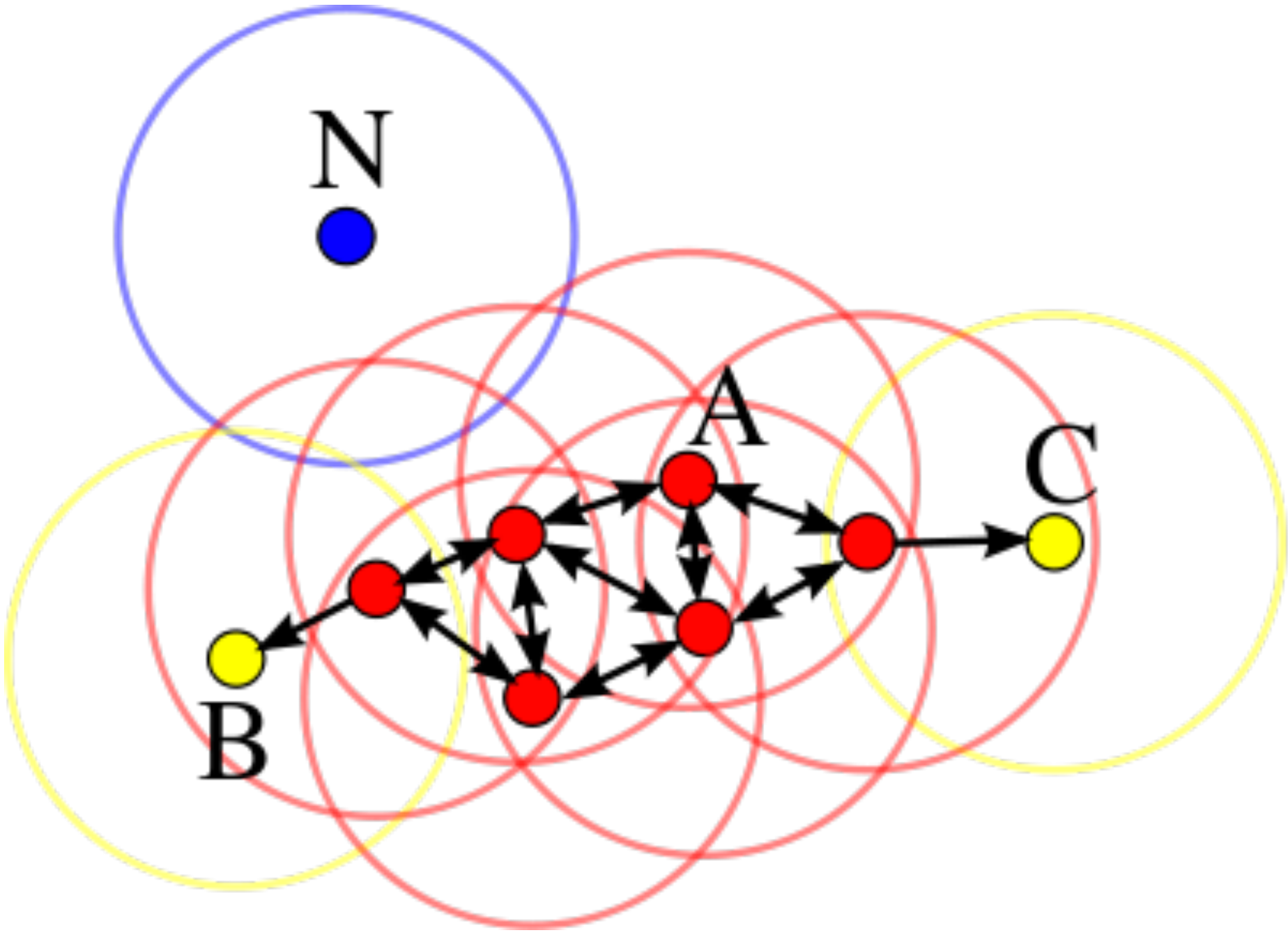
$\epsilon = 3 \quad minPts = 4$







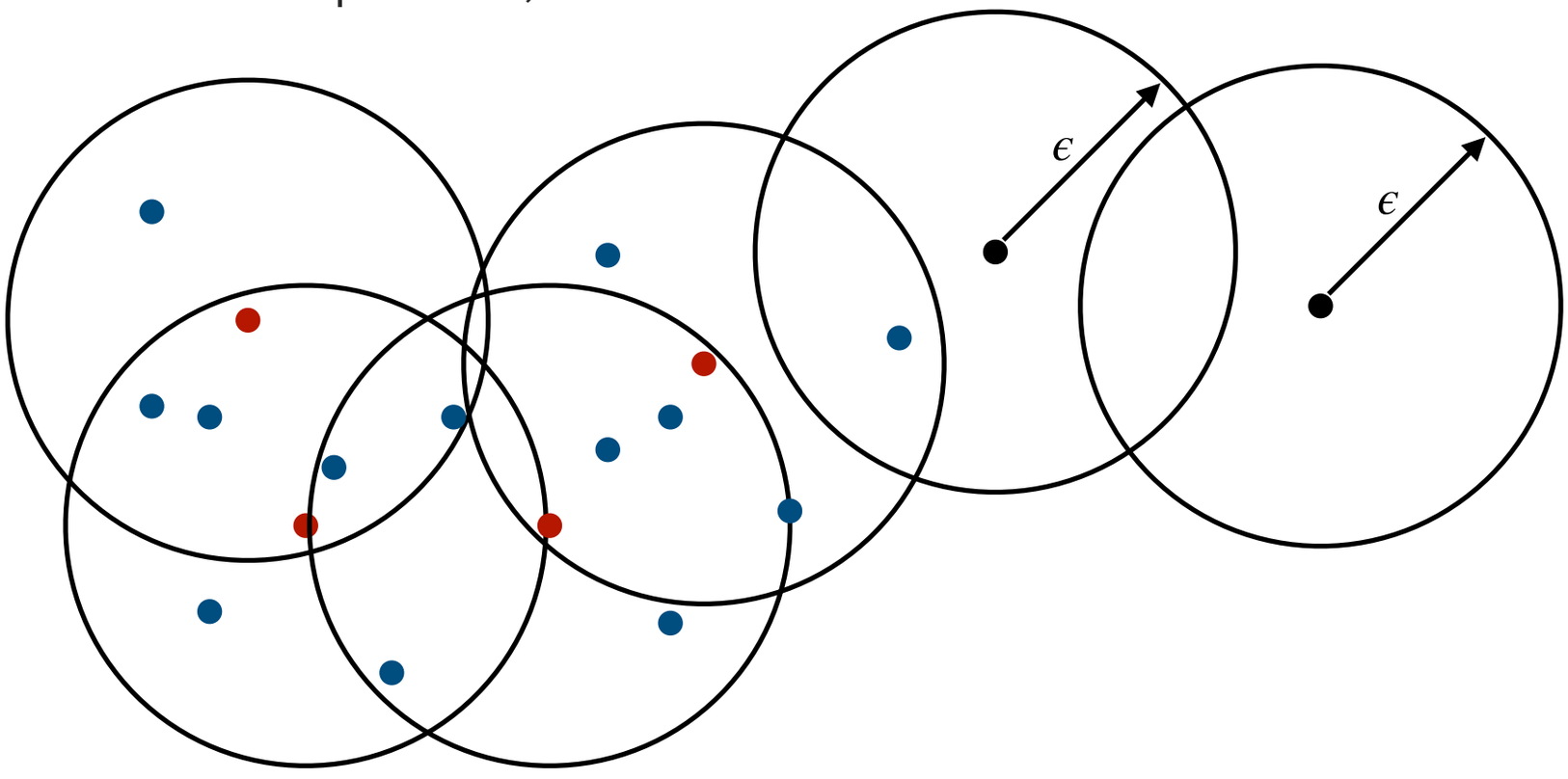
$\epsilon = 3 \quad minPts = 4$



https://en.wikipedia.org/wiki/DBSCAN

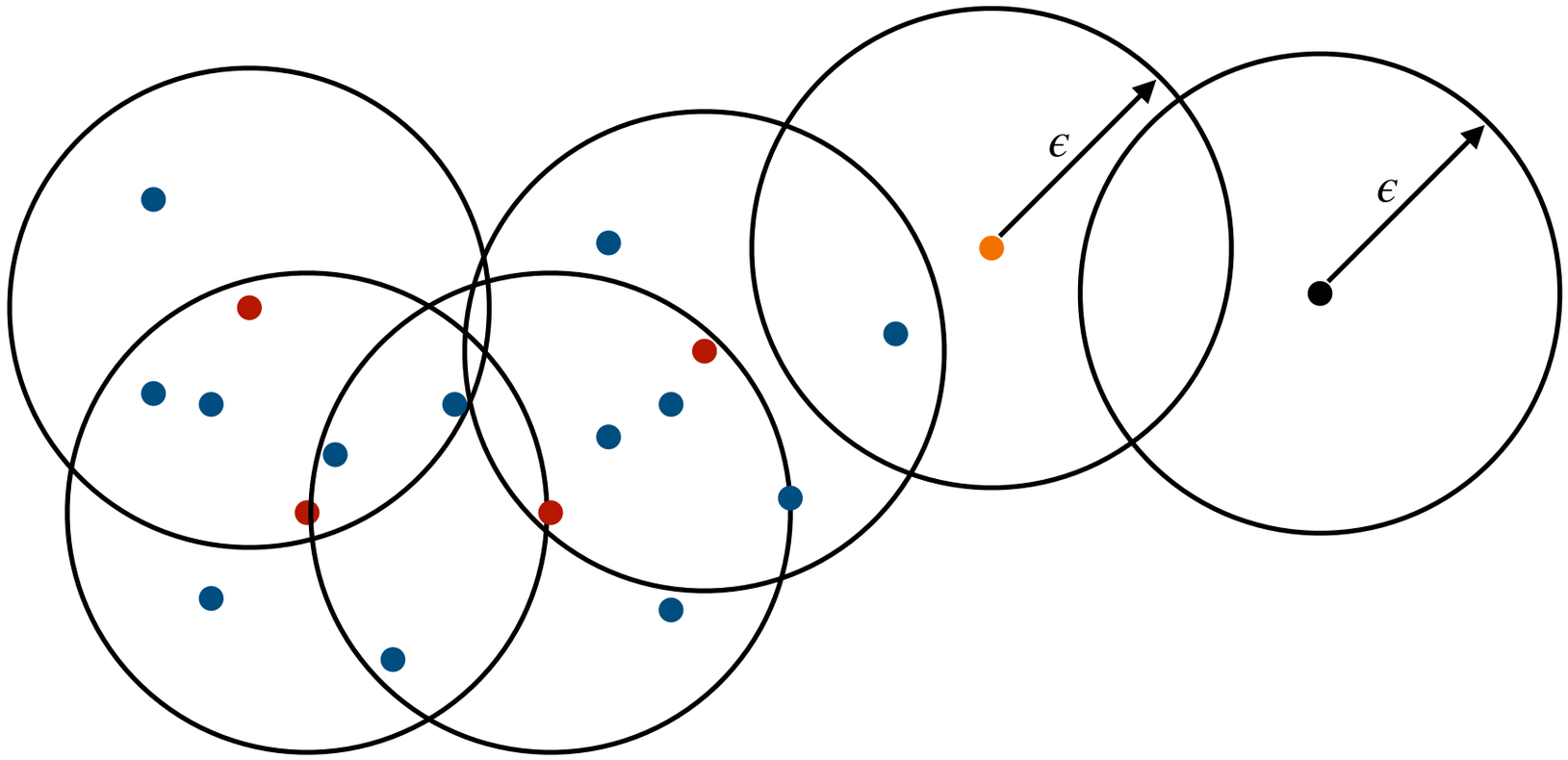
Density-Reachability

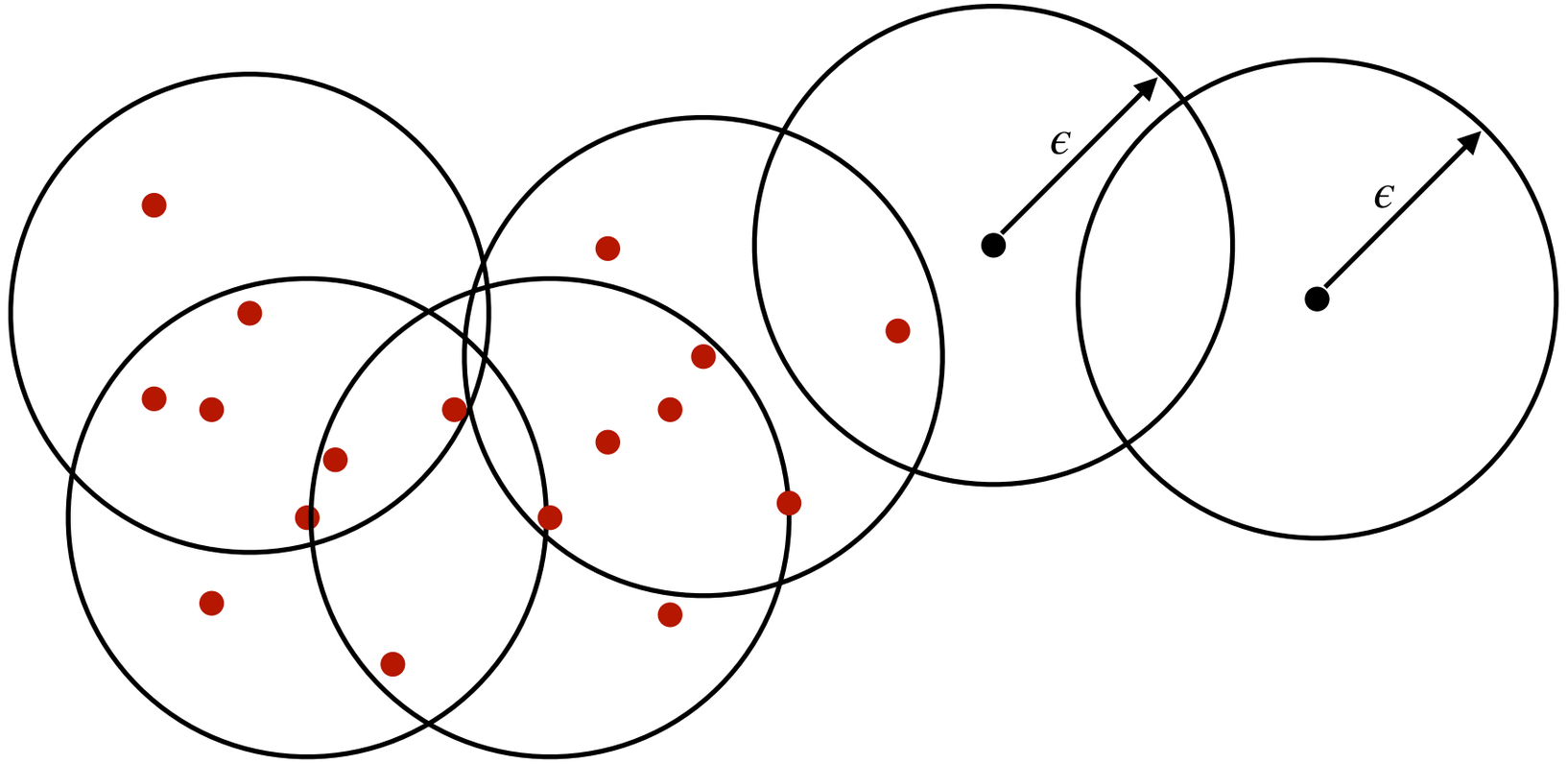
- Point y is directly density-reachable from point x, if x is a core point and y is in x's ϵ.
- Point y is (indirectly) dentist-reachable from x, if there is a path p1, p2, …, pn with p1=x and pn=y, where each p on the path must be core points,



Density Based Clustering

- Select any point p in the dataset
- Determine the point p if it is core point, if not label the point as outlier.
- Once the core point has been found, add all directly reachable to its cluster. Then do neighbour jumps to each reachable point and add them to the cluster.
- Repeat these steps until all point are assigned a cluster or label as outlier.


ϵ
ϵ


ϵ
ϵ

```r
install.packages("dbscan")
library(dbscan)
dat<-iris[,-5]
dat<-scale(dat)

db<-dbscan(dat,eps=0.5,minPts=5)
fviz_cluster(db,data=dat)
```