Cluster Analysis: Iris dataset

Nathacia Nathacia

2022-10-09

```
library(cluster)
library(factoextra)
```

Loading necessary packages and dataset

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(dbscan)

data("iris")
View(iris)
```

Investigating data

We investigate the iris dataset by displaying its name, class, structure, summary, and the first and last few rows.

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
class(iris)
```

```
## [1] "data.frame"
```

```
View(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
```

```
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
tail(animals)
```

```
##     war fly ver end gro hai
## lob   1   1   1   1  NA   1
## man   2   1   2   2   2   2
## rab   2   1   2   1   2   2
## sal   1   1   2   1  NA   1
## spi   1   1   1  NA   1   2
## wha   2   1   2   2   2   1
```

Preprocessing

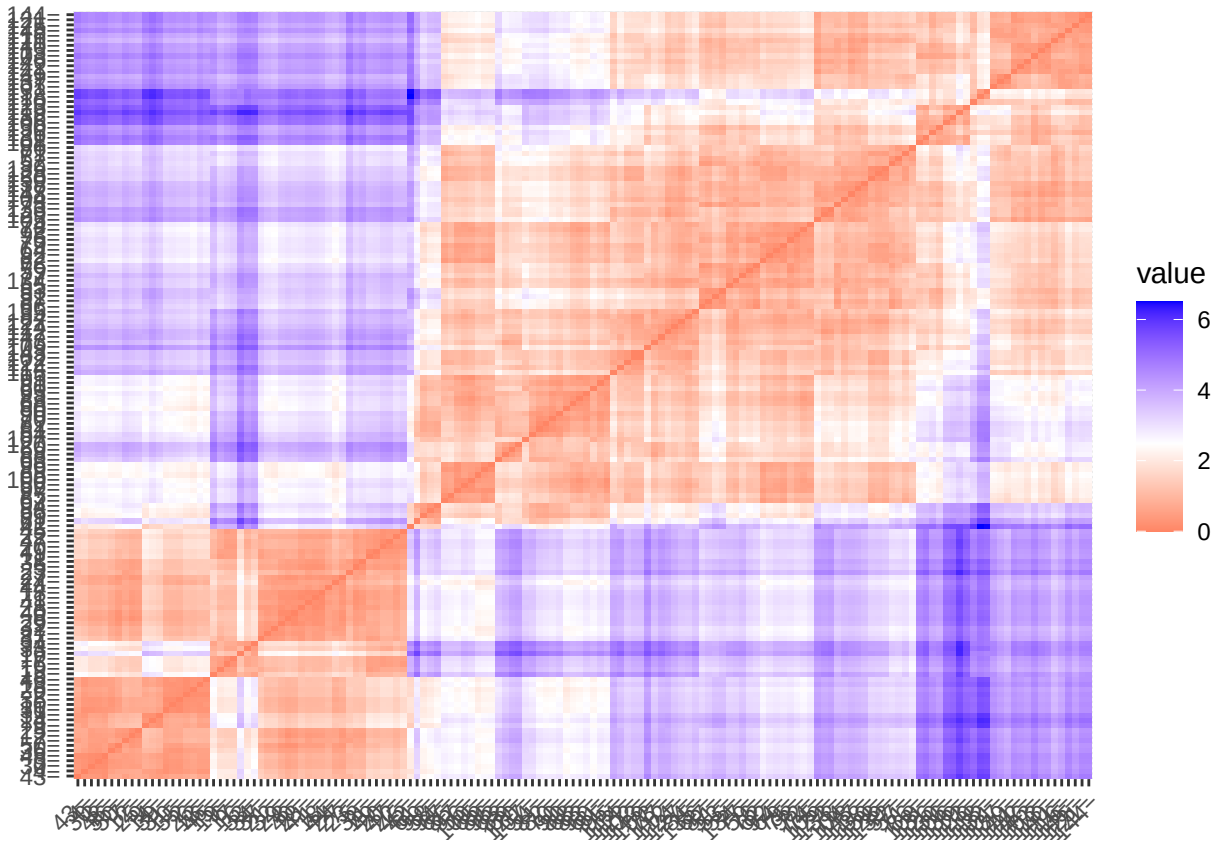
We preprocess the iris dataset by removing missing values, scaling the numerical variables, and storing the preprocessed dataset as iris_c.

```
iris_c <- iris[,1:4]
iris_c <- na.omit(iris_c)
iris_c <- scale(iris_c)
#iris_c
```

Kmeans clustering

We apply k-means clustering to the preprocessed iris_c dataset by calculating the pairwise Euclidean distances between the observations, visualizing the distance matrix using a heatmap, and performing k-means clustering with k=2 centers and 25 random starts. The result showed that the dataset can be clustered into two distinct groups.

```
distance <- get_dist(iris_c)
#distance
fviz_dist(distance)
```



Initial clustering

```
k1 <- kmeans(iris_c, centers = 2, nstart = 25)
str(k1)
```

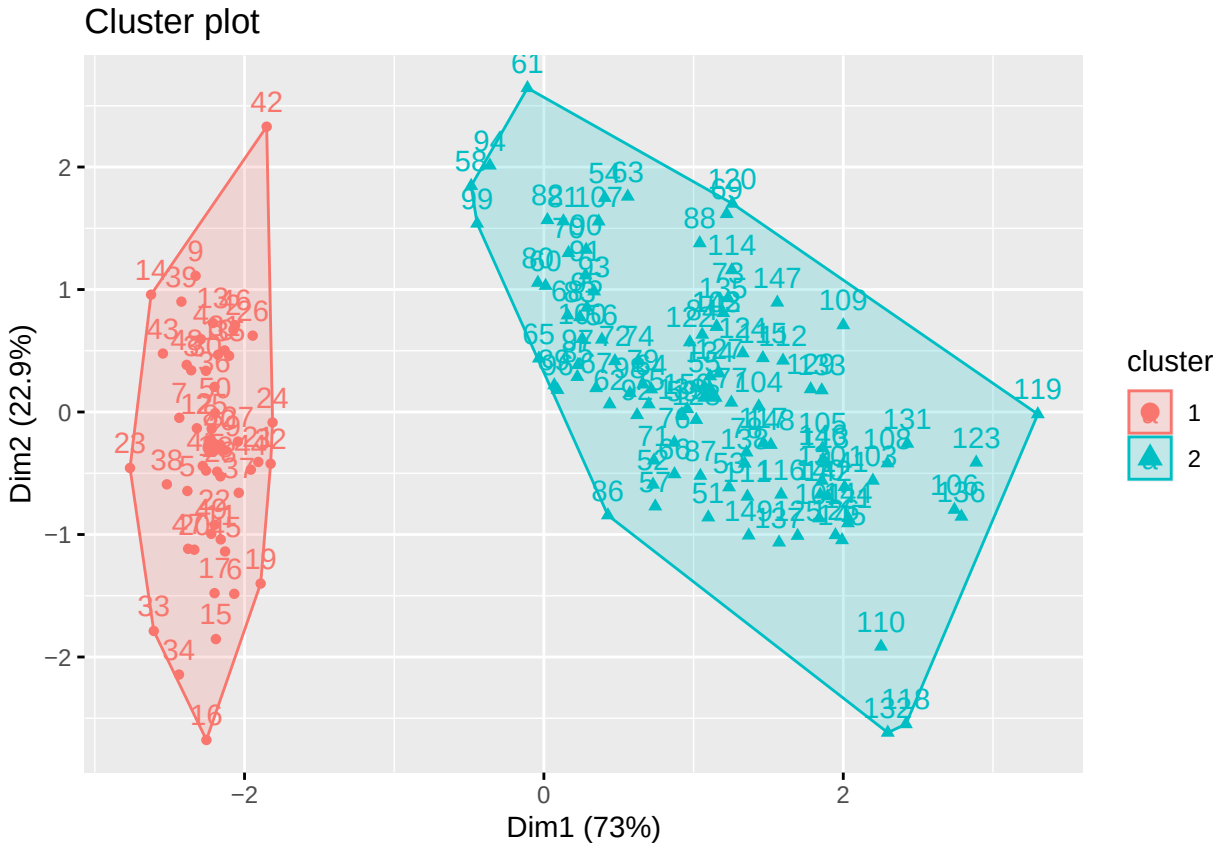
```
## List of 9
##  $ cluster     : Named int [1:150] 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
##  $ centers     : num [1:2, 1:4] -1.011 0.506 0.85 -0.425 -1.301 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
##  $ totss       : num 596
##  $ withinss    : num [1:2] 47.4 173.5
##  $ tot.withinss: num 221
##  $ betweenss   : num 375
##  $ size        : int [1:2] 50 100
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
k1
```

```
## K-means clustering with 2 clusters of sizes 50, 100
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1   -1.0111914   0.8504137    -1.300630  -1.2507035
```

```
## 2    0.5055957  -0.4252069     0.650315   0.6253518
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 141 142 143 144 145 146 147 148 149 150
##   2   2   2   2   2   2   2   2   2   2
##
## Within cluster sum of squares by cluster:
## [1]  47.35062 173.52867
##  (between_SS / total_SS =  62.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

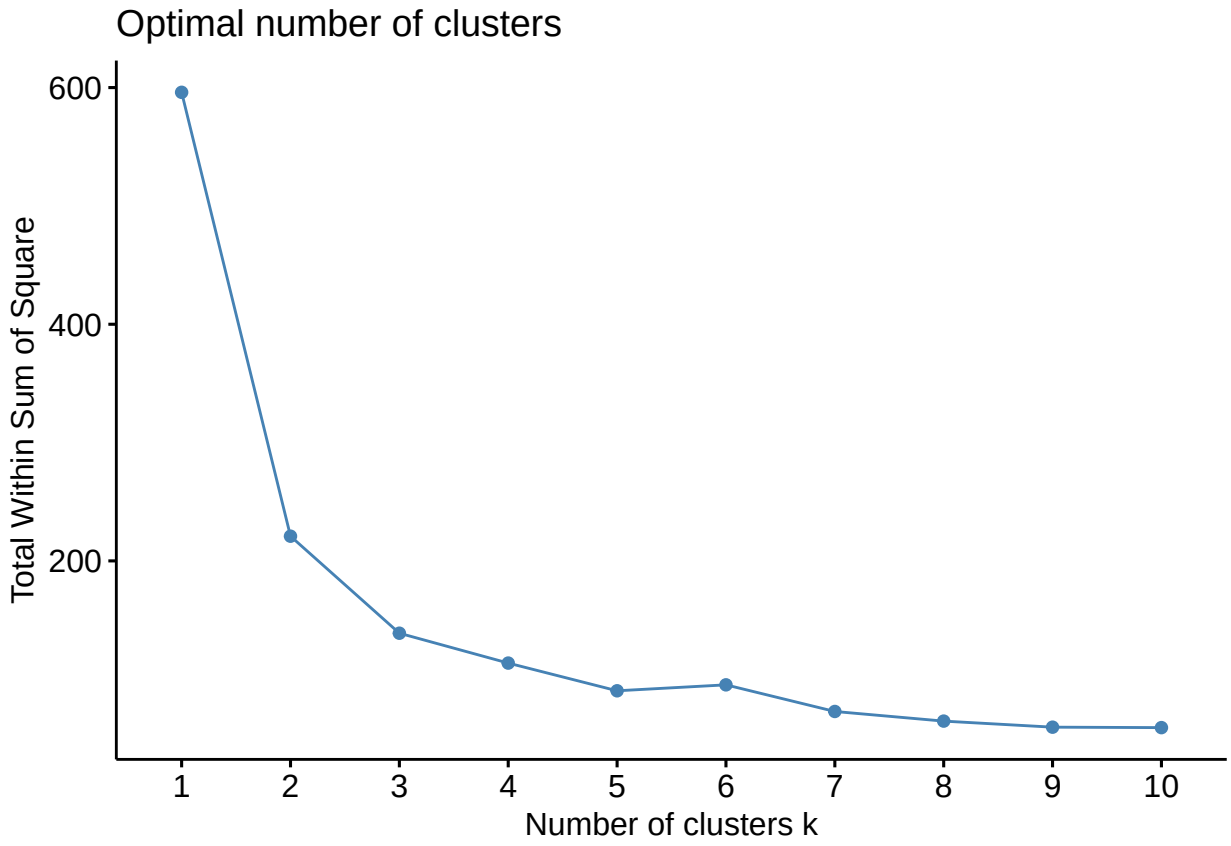
```
fviz_cluster(k1, data = iris_c)
```



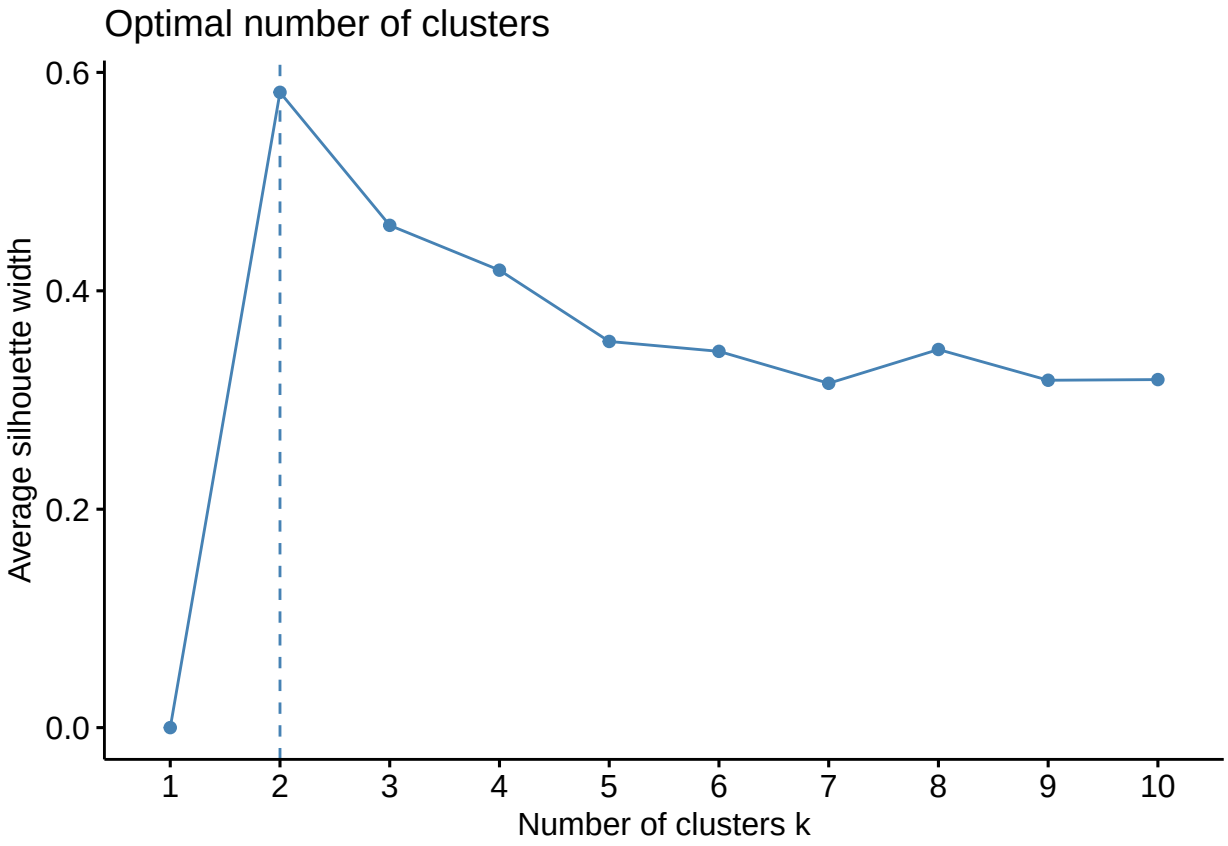
Determining the right number of clusters

To determine the optimal number of clusters, we used the elbow method and silhouette method. The elbow method plots the total within-cluster sum of squares (WSS) as a function of the number of clusters. We can use the "elbow" in the plot to determine the optimal number of clusters. The silhouette method, on the other hand, measures how similar an object is to its own cluster compared to other clusters. The optimal number of clusters can be determined by choosing the value that maximizes the silhouette width. Based on both methods, we found that the optimal number of clusters is 3.

```
set.seed(123)
fviz_nbclust(iris_c, kmeans, method = 'wss')
```




```
fviz_nbclust(iris_c, kmeans, method = 'silhouette')
```



Final kmeans clustering

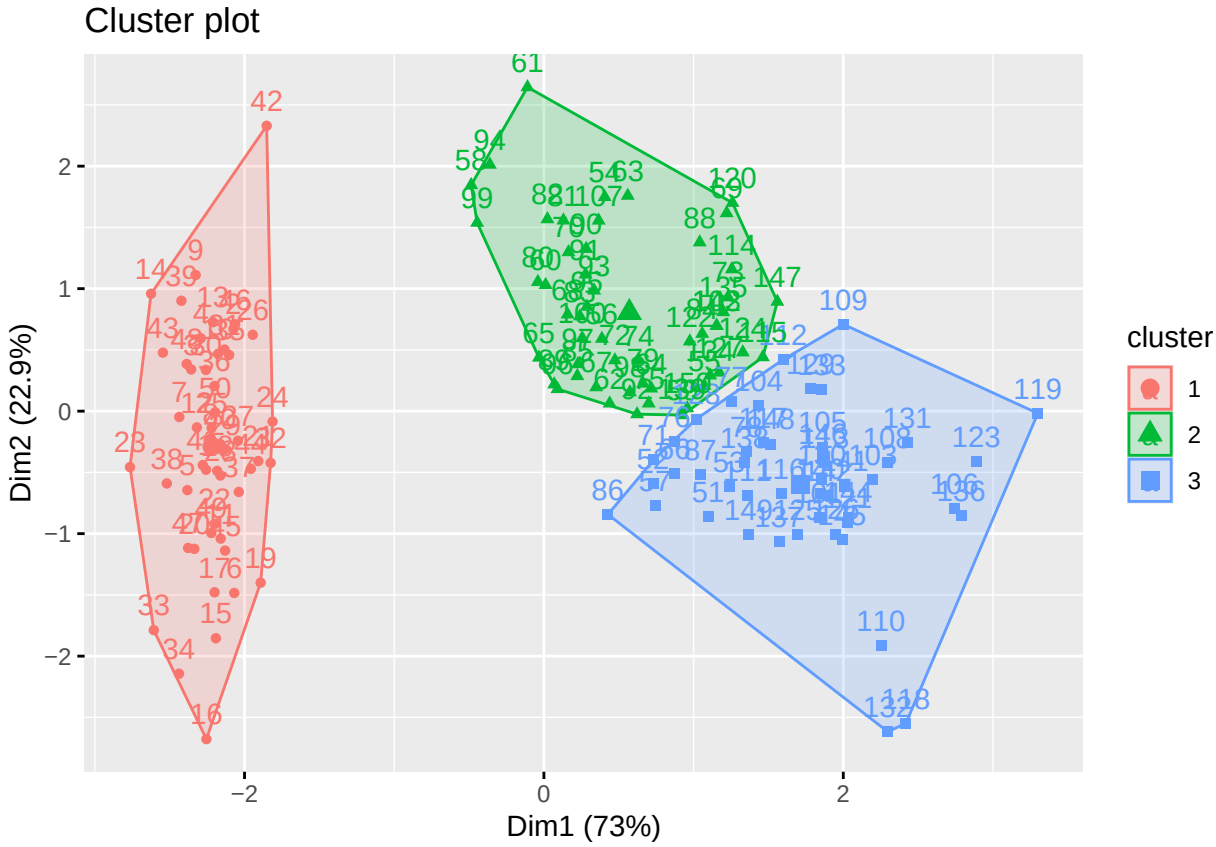
We applied k-means clustering again to the dataset, this time with the optimal number of clusters (3) identified in the previous step. We used the kmeans() function to cluster the data into three clusters, with a total of 25 starting positions. We then visualized the clusters using the fviz_cluster() function from the factoextra package. The result showed that the dataset can be clustered into three distinct groups.

```
finalk <- kmeans(iris_c, 3, nstart = 25)
finalk
```

```
## K-means clustering with 3 clusters of sizes 50, 53, 47
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1  -1.01119138  0.85041372   -1.3006301  -1.2507035
## 2  -0.05005221 -0.88042696    0.3465767   0.2805873
## 3   1.13217737  0.08812645    0.9928284   1.0141287
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   1   1   1   1   1   1   1   3   3   3   2   2   2   3   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   2   2   3   2   2   2   2   3   2   2   2   2   3   3   3   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

```
##   2   2   2   2   2   3   3   2   2   2   2   2   2   2   2   2   2   2   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   2   3   3   3   3   2   3   3   3   3   3   3   2   2   3   3   3   3   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   2   3   2   3   3   2   3   3   3   3   3   3   2   2   3   3   3   2   3
## 141 142 143 144 145 146 147 148 149 150
##   3   3   2   3   3   3   2   3   3   2
##
## Within cluster sum of squares by cluster:
## [1] 47.35062 44.08754 47.45019
##  (between_SS / total_SS =  76.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
fviz_cluster(finalk, data = iris_c)
```



Hierarchical clustering

We also applied hierarchical clustering to the dataset using the agnes() function from the cluster package. We used three different linkage methods: complete, single, and average. The agnes() function returns an object of class agnes, which contains information about the hierarchical clustering. We then plotted the dendrogram using the plot() function. The agglomerative coefficient values were also computed for each linkage method, with values closer to 1 suggesting a stronger clustering structure.

```
hc1 <- agnes(iris_c, method = 'complete')
hc1
```

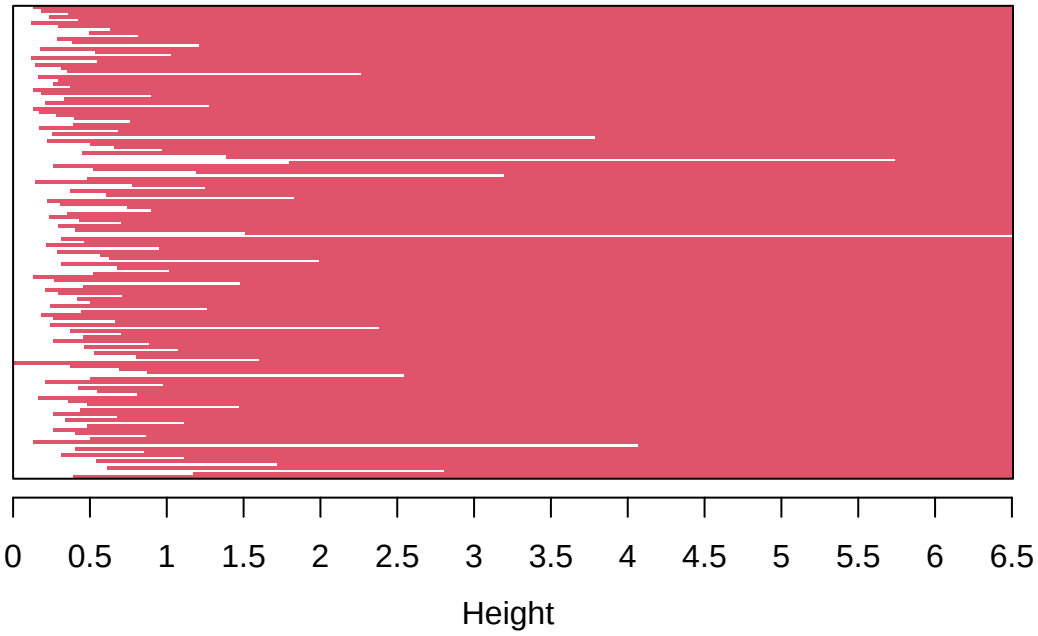
```
## Call:     agnes(x = iris_c, method = "complete")
## Agglomerative coefficient:  0.9438858
## Order of objects:
##   [1] 1   18  41  28  29  8   40  27  24  44  21  32  37  5   38  23  11  49
##  [19] 20  47  45  22  2   26  13  46  10  35  31  9   14  39  3   48  30  4
##  [37] 43  7   12  25  36  50  6   17  19  15  33  34  16  42  58  94  99  61
##  [55] 54  81  82  63  69  120 88  56  100 95  60  68  83  93  80  70  90  91
##  [73] 107 51  53  66  87  59  76  77  78  52  57  86  71  128 139 150 62  64
##  [91] 79  92  72  74  75  98  65  89  96  97  67  85  55  134 112 124 127 84
## [109] 135 73  147 109 102 143 122 115 114 101 137 149 111 116 125 121 144 141
## [127] 145 103 113 140 142 146 104 117 138 148 105 129 133 106 136 108 131 126
## [145] 130 119 123 110 118 132
## Height (summary):
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2858  0.4613  0.7550  0.8539  6.5075
##
## Available components:
## [1] "order"     "height"    "ac"        "merge"     "diss"      "call"
## [7] "method"    "order.lab" "data"
```

```
hc1$ac #agglomerative coefficient. values closer to 1 suggest strong clustering structure
```

```
## [1] 0.9438858
```

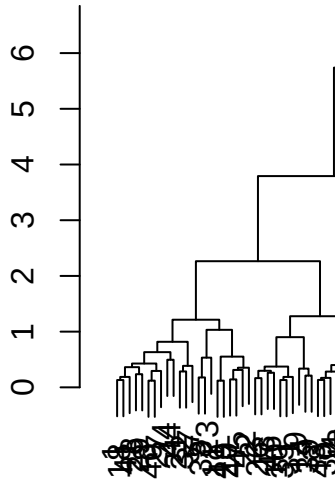
```
plot(hc1)
```

Banner of agnes(x = iris_c, method = "complete")



Agglomerative Coefficient = 0.94

Dendrogram



```
hc2 <- agnes(iris_c, method = 'single')
hc2$ac
```

```
## [1] 0.8023794
```

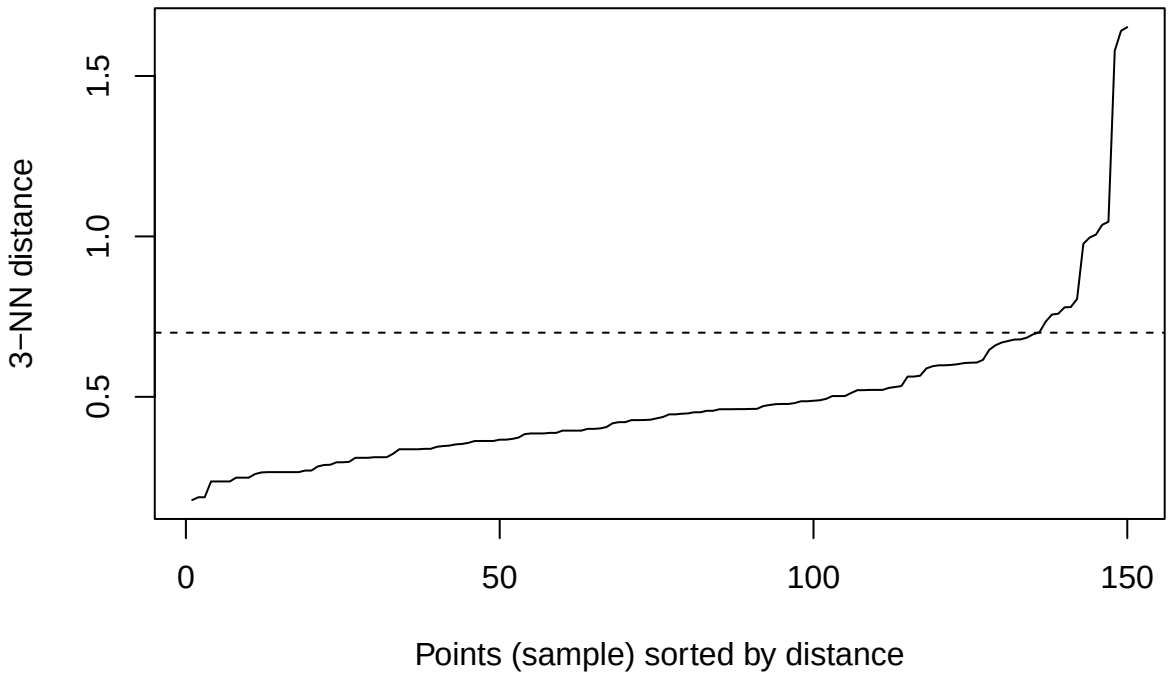
```
hc3 <- agnes(iris_c, method = 'average')
hc3$ac
```

```
## [1] 0.9035705
```

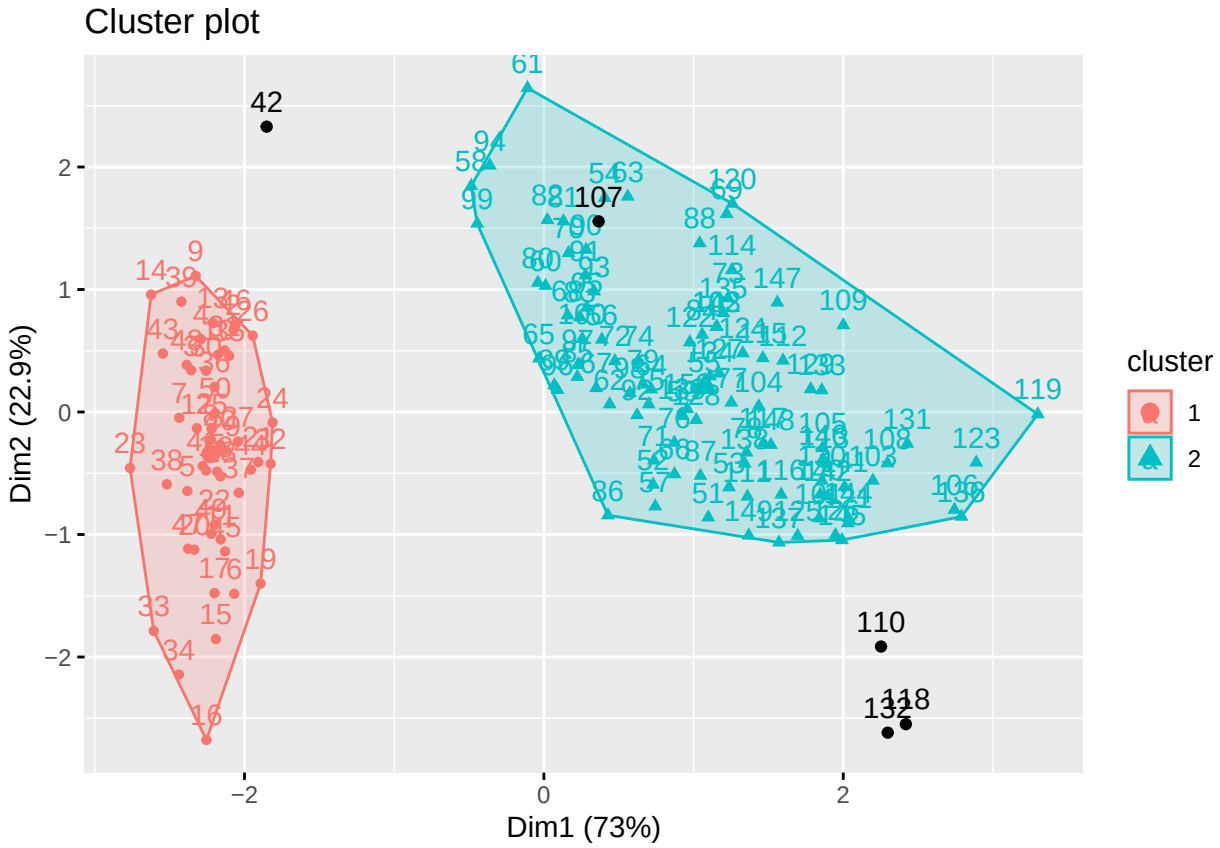
Density based clustering

Lastly, we performed density-based clustering using the dbscan() function from the dbscan package. We first used the kNNdistplot() function to determine the value of the eps parameter, which represents the maximum distance between two points in the same cluster. We then used the dbscan() function with a minPts parameter of 3 to cluster the data. We visualized the clusters using the fviz_cluster() function from the factoextra package. The result showed that the dataset can be clustered into three distinct groups.

```
kNNdistplot(iris_c, k=3)
abline(h=.7, lty=2)
```




```
db <- dbscan(iris_c, .7, minPts = 3)
fviz_cluster(db, data = iris_c)
```




```
print(db)
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.7, minPts = 3
## The clustering contains 2 cluster(s) and 5 noise points.
##
##  0  1  2
##  5 49 96
##
## Available fields: cluster, eps, minPts
```