Cluster Analysis II

Mike Strube

November 1, 2018

1 Preliminaries

In this section, the RStudio workspace and console panes are cleared of old output, variables, and other miscellaneous debris. Packages are loaded and any required data files are retrieved.

```
options(replace.assign = TRUE, width = 65, digits = 4, scipen = 4, fig.width = 4,
    fig.height = 4)
# Clear the workspace and console.
rm(list = ls(all = TRUE))
cat("\f")
```

```
# Turn off showing of significance asterisks.
options(show.signif.stars = F)
# Set the contrast option; important for ANOVAs.
options(contrasts = c("contr.sum", "contr.poly"))
how_long <- Sys.time()
set.seed(123)
library(knitr)
```

```
library(psych)
library(ggplot2)

##
## Attaching package:  'ggplot2'
## The following objects are masked from 'package:psych':
##
##     %+%, alpha

library(MASS)
library(sciplot)
library(ggplot2)
library(vegan)

## Warning:  package 'vegan' was built under R version 3.5.1
## Loading required package:  permute
## Warning:  package 'permute' was built under R version 3.5.1
## Loading required package:  lattice
## This is vegan 2.5-2

library(smacof)
```

```
## Warning: package 'smacof' was built under R version 3.5.1
## Loading required package: plotrix
##
## Attaching package: 'plotrix'
## The following object is masked from 'package:psych':
##
##     rescale
##
## Attaching package: 'smacof'
## The following object is masked from 'package:base':
##
##     transform
```

```r
library(ape)
library(ade4)
```

```
## Warning: package 'ade4' was built under R version 3.5.1
```

```r
library(scatterplot3d)
library(cluster)
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.5.1
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library(ggdendro)
```

```
## Warning: package 'ggdendro' was built under R version 3.5.1
```

```r
library(plyr)
library(fpc)
library(clusterGenomics)
library(clustertend)
library(e1071)
library(Gmedian)
```

```
## Warning: package 'Gmedian' was built under R version 3.5.1
```

```r
library(protoclust)
library(dbscan)
```

```
## Warning: package 'dbscan' was built under R version 3.5.1
##
## Attaching package: 'dbscan'
## The following object is masked from 'package:fpc':
##
##     dbscan
```

```r
library(ppclust)
```

```
## Warning: package 'ppclust' was built under R version 3.5.1
##
## Attaching package: 'ppclust'
## The following object is masked from 'package:fpc':
##
##     plotcluster
## The following object is masked from 'package:psych':
##
##     pca
```

```r
library(vegan)
```

1.1 Data Entry

```
setwd("C:\\Courses\\Psychology 516\\PowerPoint\\2018")

Iris <- read.table("iris.csv", sep = ",", header = TRUE)
Iris <- as.data.frame(Iris)
Iris$Species[Iris$Species == "1"] <- "Setosa"
Iris$Species[Iris$Species == "2"] <- "Versicolor"
Iris$Species[Iris$Species == "3"] <- "Virginica"
Iris_Dist <- dist(Iris[, 1:4], method = "euclidean")
```

2 Additional Clustering Methods

Additional clustering methods attempt to improve upon limitations in classical methods:

- ***Partitioning Around Medoids (PAM)***
- ***Minimax Clustering***
- ***Density-Based Clustering (DBSCAN)***
- ***Fuzzy Sets***
- ***Minimum Spanning Trees***

2.1 Partitioning Around Medoids

A medoid is the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal. In other words, it is the most centrally located or representative object in the cluster. The goal is to find k medoids that minimize the sum of the dissimilarities of the observations to their closest medoid. The pam( ) function in the cluster library can perform this calculation with the number of clusters required as input.

The cluster information that is provided includes size of the cluster, the maximal and average dissimilarity between the observations in the cluster and the cluster's medoid, the maximal dissimilarity between two observations of the cluster (called the diameter of the cluster), and the minimal dissimilarity between an observation of the cluster and an observation of another cluster (called the separation of the cluster). The silhouette scores and coefficients are also provided (described later).

```
Iris_P <- pam(Iris[, 1:4], k = 3, diss = FALSE, metric = "euclidean")
Iris_P$clustering

##   [1] 1 2 3 2 3 1 2 3 3 1 3 3 2 3 2 3 2 1 3 2 2 3 2 2 3 1 2 3 3 3
##  [31] 1 3 3 3 2 1 1 3 2 1 2 1 3 1 2 3 1 3 3 2 1 1 3 1 1 1 2 2 1 1
##  [61] 1 3 3 1 1 3 3 1 1 3 3 1 1 2 2 3 3 2 1 1 2 2 3 2 3 3 3 1 1 2
##  [91] 3 1 3 3 3 1 1 3 3 3 1 1 2 3 2 3 1 1 3 3 2 2 1 3 3 1 3 3 3 3
## [121] 3 3 2 2 1 1 2 2 3 3 2 2 2 3 1 1 1 3 1 1 3 3 3 1 1 1 3 2 2 1

Iris_P$clusinfo

##      size max_diss av_diss diameter separation
## [1,]   50    12.37   4.846    24.29     16.401
## [2,]   38    17.23   7.260    24.19      2.646
## [3,]   62    18.38   7.470    26.78      2.646

Iris_P$medoids

##      Sepal_Length Sepal_Width Petal_Length Petal_Width
## [1,]           50          34           15           2
## [2,]           68          30           55          21
## [3,]           60          29           45          15

Iris_P$id.med

## [1]  96 133  70
```

```
Iris_P$silinfo

## $widths
##     cluster neighbor sil_width
## 96        1        3   0.85391
## 65        1        3   0.85296
## 1         1        3   0.85210
## 126       1        3   0.85102
## 107       1        3   0.85033
## 108       1        3   0.84942
## 73        1        3   0.84930
## 36        1        3   0.84364
## 146       1        3   0.84202
## 37        1        3   0.84189
## 135       1        3   0.83591
## 68        1        3   0.83347
## 56        1        3   0.83225
## 101       1        3   0.82932
## 89        1        3   0.82877
## 145       1        3   0.82591
## 113       1        3   0.82529
## 6         1        3   0.82165
## 140       1        3   0.82030
## 69        1        3   0.81902
## 150       1        3   0.81823
## 47        1        3   0.81785
## 64        1        3   0.81549
## 31        1        3   0.81519
## 52        1        3   0.81340
## 136       1        3   0.81056
## 51        1        3   0.80978
## 102       1        3   0.80501
## 116       1        3   0.80310
## 42        1        3   0.80024
## 79        1        3   0.79899
## 55        1        3   0.79866
## 26        1        3   0.79415
## 18        1        3   0.79411
## 10        1        3   0.79297
## 40        1        3   0.78658
## 144       1        3   0.78418
## 125       1        3   0.77568
## 54        1        3   0.77504
## 44        1        3   0.76857
## 80        1        3   0.76273
## 97        1        3   0.75215
## 92        1        3   0.74828
## 61        1        3   0.74699
## 59        1        3   0.74615
## 88        1        3   0.72225
## 139       1        3   0.70686
## 60        1        3   0.70259
## 72        1        3   0.64377
## 137       1        3   0.63900
```

```
## 17        2        3   0.61325
## 81        2        3   0.61194
## 103       2        3   0.60703
## 41        2        3   0.58015
## 50        2        3   0.57818
## 4         2        3   0.57023
## 23        2        3   0.56708
## 127       2        3   0.56152
## 105       2        3   0.56017
## 111       2        3   0.55917
## 123       2        3   0.55778
## 133       2        3   0.55510
## 74        2        3   0.55187
## 24        2        3   0.54384
## 124       2        3   0.53445
## 78        2        3   0.51609
## 45        2        3   0.51237
## 149       2        3   0.49928
## 21        2        3   0.49487
## 39        2        3   0.48442
## 90        2        3   0.48341
## 2         2        3   0.46255
## 75        2        3   0.45550
## 58        2        3   0.45434
## 112       2        3   0.44076
## 27        2        3   0.42514
## 15        2        3   0.42111
## 7         2        3   0.41026
## 35        2        3   0.39825
## 82        2        3   0.38878
## 84        2        3   0.36076
## 20        2        3   0.35245
## 13        2        3   0.31493
## 128       2        3   0.26063
## 132       2        3   0.22965
## 148       2        3   0.11798
## 131       2        3   0.05340
## 57        2        3   0.05329
## 122       3        2   0.63064
## 86        3        2   0.62754
## 43        3        2   0.62445
## 77        3        2   0.62206
## 100       3        2   0.61973
## 134       3        2   0.61436
## 14        3        2   0.61158
## 67        3        2   0.61073
## 117       3        2   0.60716
## 87        3        2   0.60657
## 48        3        2   0.59561
## 33        3        2   0.59466
## 141       3        2   0.59294
## 19        3        2   0.59221
## 30        3        2   0.58969
## 106       3        2   0.58710
```

```
## 94        3        2   0.57829
## 130       3        1   0.56310
## 99        3        1   0.56076
## 104       3        2   0.55780
## 66        3        2   0.55751
## 70        3        2   0.55294
## 110       3        2   0.55108
## 38        3        2   0.54077
## 71        3        1   0.50698
## 11        3        2   0.50353
## 120       3        2   0.50005
## 119       3        2   0.49425
## 115       3        2   0.48936
## 129       3        2   0.48217
## 25        3        2   0.46682
## 8         3        2   0.46150
## 95        3        2   0.41599
## 142       3        2   0.38549
## 29        3        2   0.38119
## 98        3        1   0.37479
## 3         3        2   0.36885
## 83        3        2   0.35125
## 93        3        2   0.34419
## 109       3        2   0.33356
## 9         3        2   0.32910
## 91        3        2   0.32869
## 62        3        2   0.32459
## 118       3        2   0.31046
## 63        3        2   0.30904
## 49        3        1   0.29362
## 138       3        2   0.28503
## 22        3        2   0.28233
## 85        3        1   0.26525
## 12        3        2   0.26087
## 53        3        2   0.25227
## 16        3        2   0.23225
## 76        3        2   0.23225
## 46        3        2   0.20297
## 32        3        2   0.18544
## 114       3        2   0.16655
## 121       3        1   0.14132
## 5         3        2   0.13900
## 143       3        2   0.12629
## 34        3        2   0.10417
## 28        3        2   0.02672
## 147       3        2   0.02636
##
## $clus.avg.widths
## [1] 0.7981 0.4511 0.4173
##
## $avg.width
## [1] 0.5528
```

```
Iris_Class <- as.data.frame(cbind(Iris_P$clustering, Iris$Species))
```

```r
names(Iris_Class) <- c("Cluster", "Species")
table(Iris_Class$Species, Iris_Class$Cluster)
```

```
##
##              1  2  3
##   Setosa     50  0  0
##   Versicolor  0  2 48
##   Virginica   0 36 14
```

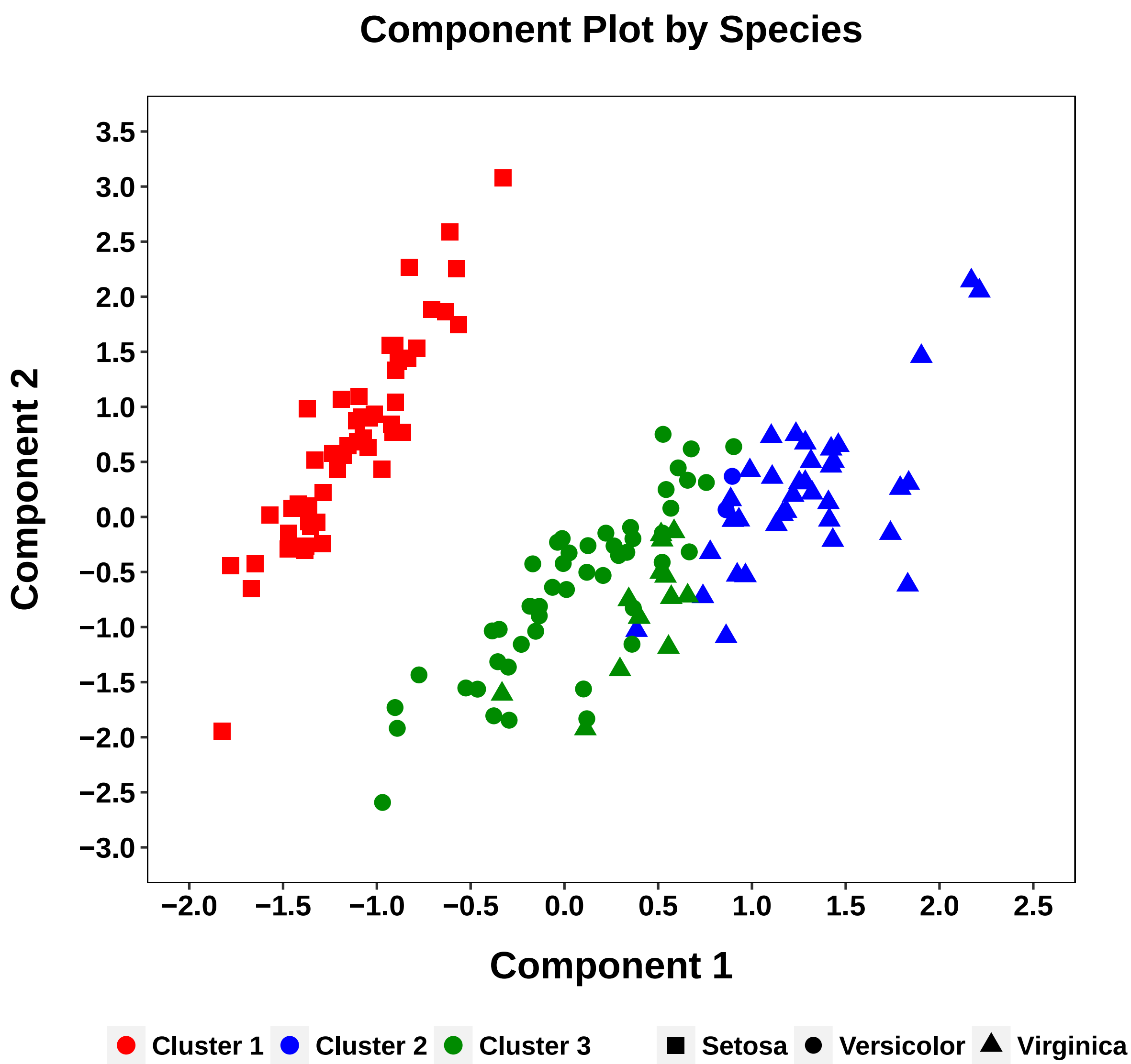
```r
# The following table compares the clustering done by pam( ) and
# that done by kmeans( ).
Iris_K <- kmeans(Iris[, 1:4], centers = 3, iter.max = 1000, nstart = 10)
table(Iris_K$cluster, Iris_P$clustering)
```

```
##
##      1  2  3
##   1  0  0 62
##   2  0 38  0
##   3 50  0  0
```

2.1.1 Dimensional Plot for Iris Data

```r
# Use PCA to show potential clustering along two dimensions.
PCA <- principal(Iris[, 1:4], nfactors = 2, rotate = "varimax", scores = TRUE)
Iris_Cluster <- Iris_P$clustering
plot_data <- cbind(Iris, PCA$scores, Iris_Cluster, Iris$Species)
plot_data$Cluster_F <- factor(plot_data$Iris_Cluster, levels = c(1,
    2, 3), labels = c("Cluster 1", "Cluster 2", "Cluster 3"))
plot_data$Species_F <- factor(plot_data$Species)
```

```r
ggplot(plot_data, aes(x = RC1, y = RC2, color = Cluster_F, shape = Species_F)) +
    geom_point(size = 3) + scale_color_manual(values = c("red", "blue",
    "green4", "orange", "black")) + scale_shape_manual(values = c(15,
    16, 17, 18)) + scale_y_continuous(breaks = c(seq(-3, 3.5, 0.5))) +
    scale_x_continuous(breaks = c(seq(-2, 2.5, 0.5))) + coord_cartesian(xlim = c(-2,
    2.5), ylim = c(-3, 3.5)) + xlab("Component 1") + ylab("Component 2") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Component Plot by Species")
```



2.2 Minimax Clustering

Minimax clustering is an alternative hierarchical method developed by Bien and Tibshirani (2011). It has some resemblance to the medoid approach in that at each step the method identifies the most highly representative object (the prototype) for the cluster that has been formed. The linkage in this method is the radius of the smallest enclosing ball, centered at a point chosen from the two clusters being considered for joining. This is done by identifying the object whose farthest distance from another object (max) is the closest (min). This central object is the prototype for the newly formed cluster.

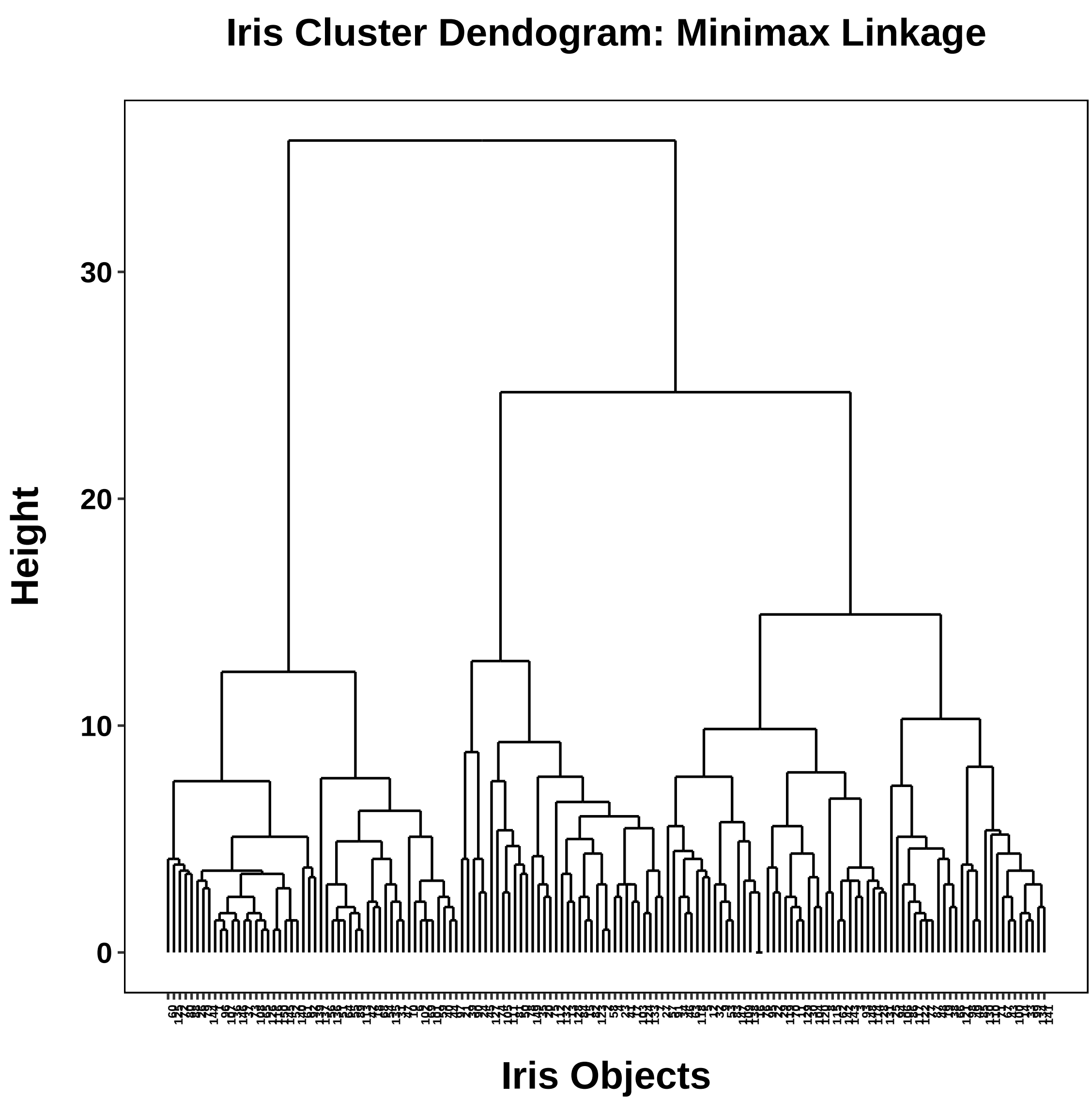
One desirable feature of this method, apart from identifying prototypes for each cluster, is that when a dendrogram is cut at a particular height, H, every observation in the dataset is within H of its cluster's prototype.

Bien, J., & Tibshirani, R. (2011). Prototype selection for interpretable classification. Annals of Applied Statistics, 5, 2403-2424.

```
Iris_Proto <- protoclust(Iris_Dist)

Iris_Proto$protos

##   [1]  16  96   2  89  65 116  96  69  69  44  31  53  51  64  36
##  [16]  37 126  52 140  11  33  15  77  77  43  62  49  89 107  73
##  [31]  46  33  77 124  44  89  18  38  11 134 104  69  83  47  68
##  [46]  13  17  77   3   4   7  44  65  20  43 129  46  84   8  22
##  [61]  24  28 138 105 150  79 131 135  64  66 103 103  77  33  53
##  [76]  20   2  40  79  93  93  16 131   5 104  92  80 146  50  13
##  [91]   5  14 124  88  49 146  22  92 114  50  88  49  68  21  19
## [106]  45  46  88  75 104  14  58 118  19  50 135 109  15 101  86
## [121] 150  14  14 105   4  70  12 109 133   6 133   3  38  81 125
## [136]  51   2  16   3  99  24 103  63  94  96  81  77   5  87

clustnumber <- protocut(Iris_Proto, k = 3)[[1]]
protocut(Iris_Proto, k = 3)[[2]]

## [1] 96 81 77

Iris_Class <- as.data.frame(cbind(clustnumber, Iris$Species))
names(Iris_Class) <- c("Cluster", "Species")
table(Iris_Class$Species)

##
##     Setosa Versicolor  Virginica
##         50         50         50

table(Iris_Class$Cluster)

##
##  1  2  3
## 50 35 65

table(Iris_Class$Species, Iris_Class$Cluster)

##
##               1  2  3
##   Setosa     50  0  0
##   Versicolor  0  0 50
##   Virginica   0 35 15
```

```
ggdendrogram(Iris_Proto, theme_dendro = FALSE) + xlab("Iris Objects") +
    ylab("Height") + theme(text = element_text(size = 14, family = "sans",
    color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 5, face = "bold", angle = 90), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16, angle = 90), axis.line.x = element_blank(),
    axis.line.y = element_blank(), plot.title = element_text(size = 16,
        face = "bold", margin = margin(0, 0, 20, 0), hjust = 0.5),
    panel.background = element_rect(fill = "white", linetype = 1,
```

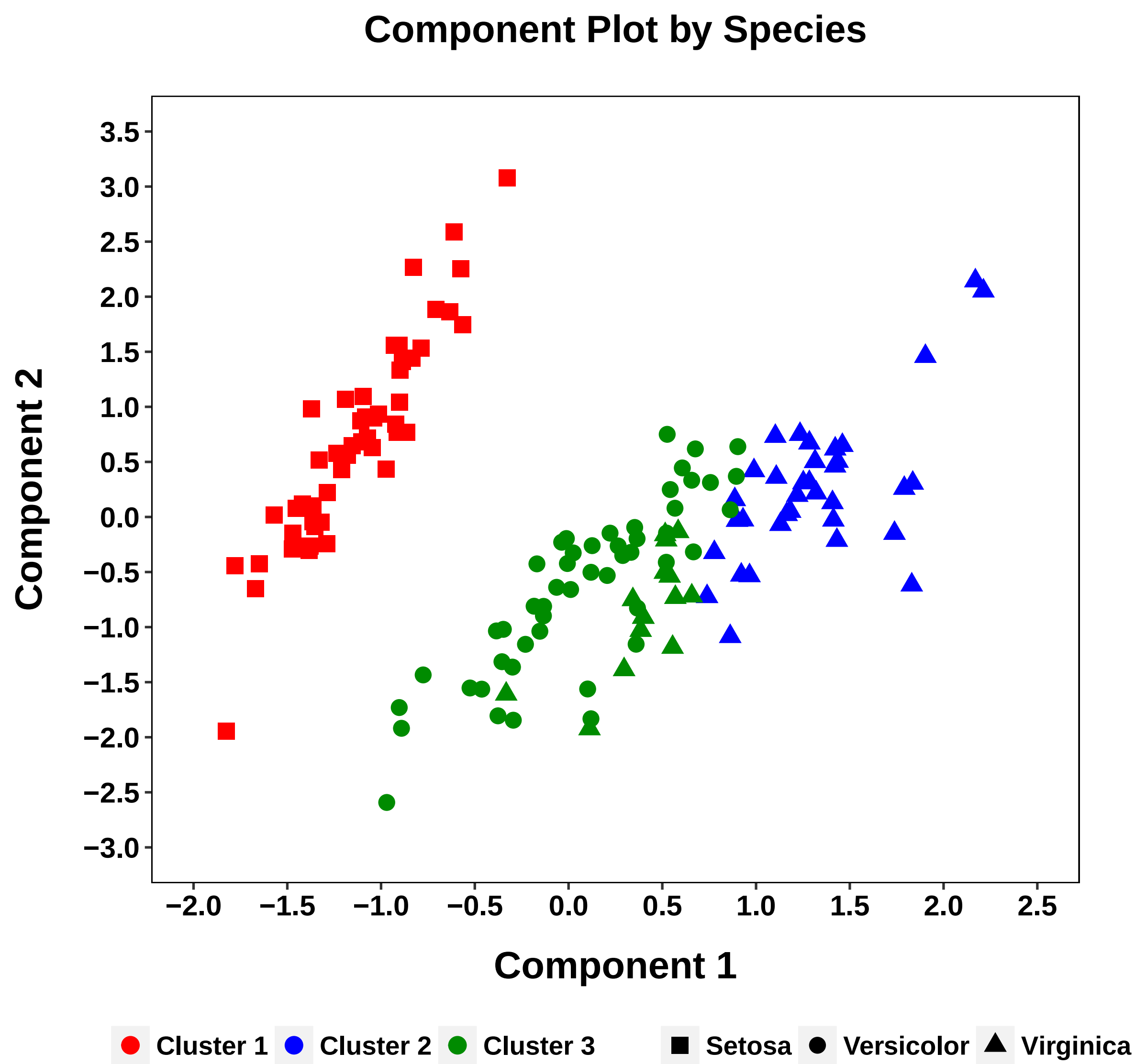
```
        color = "black"), panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
    plot.background = element_rect(fill = "white"), plot.margin = unit(c(1,
        1, 1, 1), "cm"), legend.position = "bottom", legend.title = element_blank()) +
    ggtitle("Iris Cluster Dendogram: Minimax Linkage")

## Warning in if (dataClass %in% c("dendrogram", "hclust")) {:  the condition has length >
1 and only the first element will be used
## Warning in if (dataClass %in% c("dendrogram", "hclust")) {:  the condition has length >
1 and only the first element will be used
```



2.2.1 Dimensional Plot for Iris Data

```r
# Use PCA to show potential clustering along two dimensions.
PCA <- principal(Iris[, 1:4], nfactors = 2, rotate = "varimax", scores = TRUE)
Iris_Cluster <- clustnumber
plot_data <- cbind(Iris, PCA$scores, Iris_Cluster, Iris$Species)
plot_data$Cluster_F <- factor(plot_data$Iris_Cluster, levels = c(1,
    2, 3), labels = c("Cluster 1", "Cluster 2", "Cluster 3"))
plot_data$Species_F <- factor(plot_data$Species)
```

```r
ggplot(plot_data, aes(x = RC1, y = RC2, color = Cluster_F, shape = Species_F)) +
    geom_point(size = 3) + scale_color_manual(values = c("red", "blue",
    "green4", "orange", "black")) + scale_shape_manual(values = c(15,
    16, 17, 18)) + scale_y_continuous(breaks = c(seq(-3, 3.5, 0.5))) +
    scale_x_continuous(breaks = c(seq(-2, 2.5, 0.5))) + coord_cartesian(xlim = c(-2,
    2.5), ylim = c(-3, 3.5)) + xlab("Component 1") + ylab("Component 2") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Component Plot by Species")
```



2.3 Density-Based Clustering

One limitation to hierarchical and partitioning methods is that they work best for circular, spherical, or convex clusters. When clusters have more unusual shapes, then these traditional methods struggle. Density-based clustering can provide a solution for finding unusual cluster shapes. The most popular algorithm is called DBSCAN, which stands for density-based spatial clustering and application with noise. The latter part is important because the method recognizes that some objects may not fit neatly into any clusters and resemble noise.

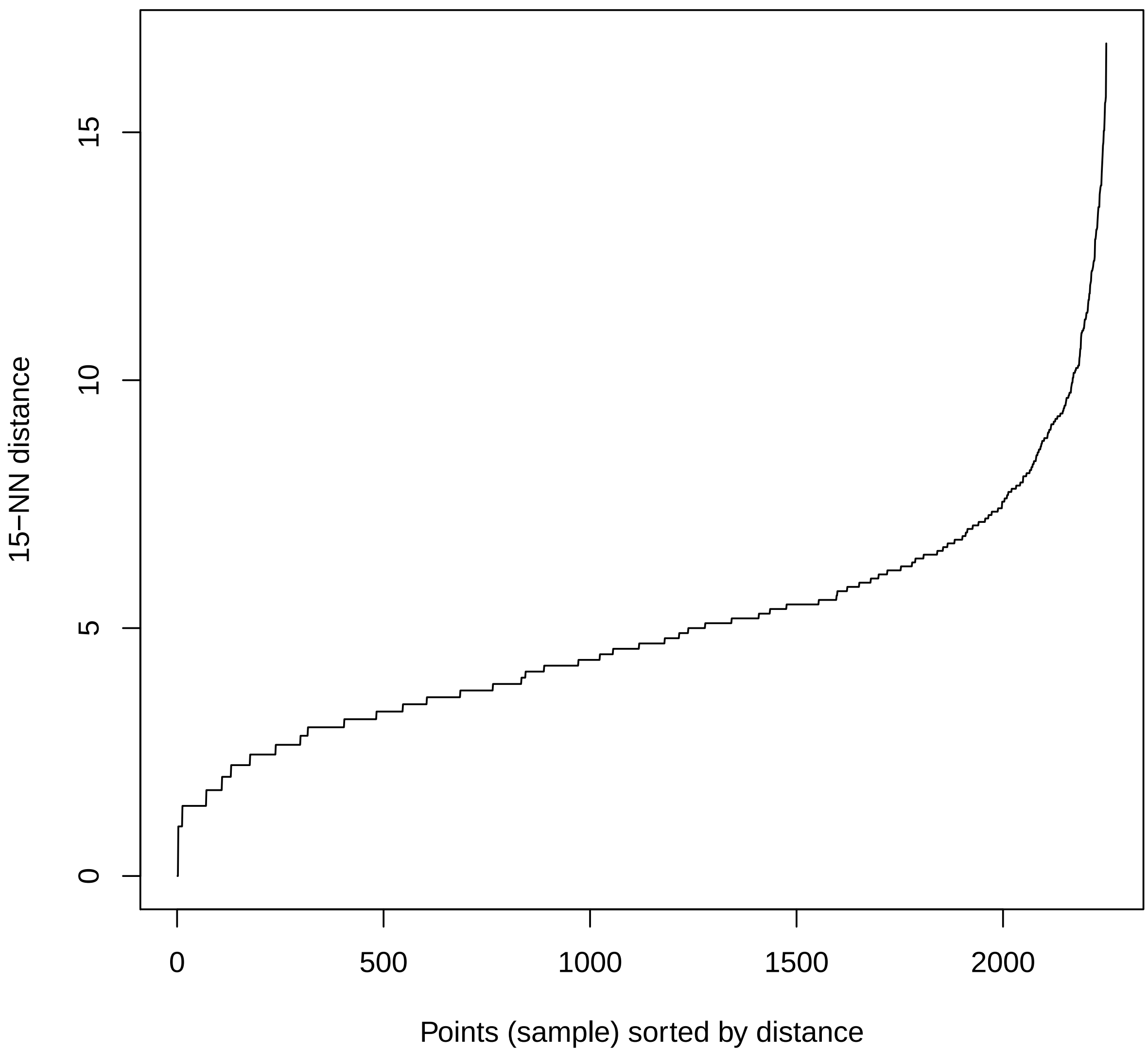
The goal of DBSCAN is to identify dense regions. Two parameters are required for DBSCAN: epsilon ("eps") and minimum points ("MinPts"). The parameter, eps, is the radius of neighborhood around a point, x. The area enclosed by this radius is called the ϵ*-neighborhood of x. The parameter, MinPts, is the minimum number of neighbors that* *must be within ϵ-neighborhood. Using these parameters, we can define different kinds of points. A core point has a neighbor count greater than or equal to MinPts. A border point has neighbors less than MinPts, but it belongs to the ϵ-neighborhood of another point. A noise point (also called an outlier) is neither a core nor a border point.*

These point definitions are then used to define three additional features: direct density reachable, density reachable, and density connected. A point "A" is directly density reachable from another point "B" if "A" is in the ϵ-neighborhood of "B" and "B" is a core point. A point "A" is density reachable from "B" if there are a set of core points leading from "B" to "A." Two points "A" and "B" are density connected if there is a core point "C", such that both "A" and "B" are density reachable from "C."

The DBSCAN algorithm works as follow:

For each point, x_i, compute the distance between x_i and the other points. Find all neighbor points within distance eps of the starting point (x_i). Each point, with a neighbor count greater than or equal to MinPts, is marked as a core point. For each core point, if it is not already assigned to a cluster, create a new cluster. Find recursively all its density-connected points and assign them to the same cluster as the core point. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are treated as noise.

```
dbscan::kNNdistplot(Iris[, 1:4], k = 15)
```




```
Iris_Density <- fpc::dbscan(Iris_Dist, eps = 5, MinPts = 10, method = "dist")
Iris_Cluster <- Iris_Density$cluster

Iris_Class <- as.data.frame(cbind(Iris_Cluster, Iris$Species))
names(Iris_Class) <- c("Cluster", "Species")
table(Iris_Class$Species)

##
##     Setosa Versicolor  Virginica
##         50         50         50

table(Iris_Class$Cluster)

##
```

```
##  0  1  2
## 27 48 75
```

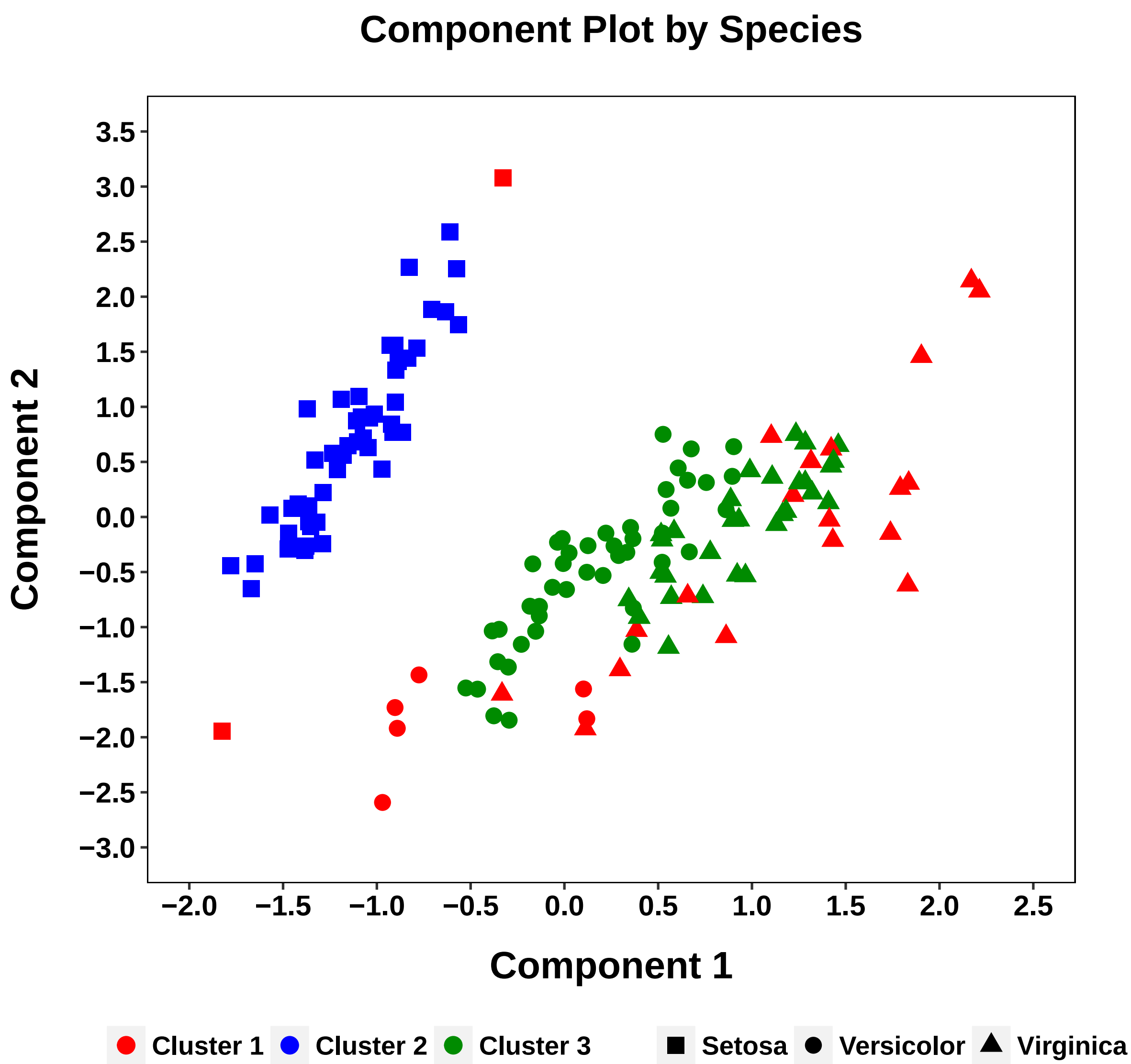
```r
table(Iris_Class$Species, Iris_Class$Cluster)
```

```
##
##              0  1  2
##   Setosa     2 48  0
##   Versicolor 6  0 44
##   Virginica 19  0 31
```

2.3.1 Dimensional Plot for Iris Data

```r
# Use PCA to show potential clustering along two dimensions.
PCA <- principal(Iris[, 1:4], nfactors = 2, rotate = "varimax", scores = TRUE)
Iris_Cluster <- Iris_Density$cluster
plot_data <- cbind(Iris, PCA$scores, Iris_Cluster, Iris$Species)
plot_data$Cluster_F <- factor(plot_data$Iris_Cluster, levels = c(0,
    1, 2), labels = c("Cluster 1", "Cluster 2", "Cluster 3"))
plot_data$Species_F <- factor(plot_data$Species)
```

```r
ggplot(plot_data, aes(x = RC1, y = RC2, color = Cluster_F, shape = Species_F)) +
    geom_point(size = 3) + scale_color_manual(values = c("red", "blue",
    "green4", "orange", "black")) + scale_shape_manual(values = c(15,
    16, 17, 18)) + scale_y_continuous(breaks = c(seq(-3, 3.5, 0.5))) +
    scale_x_continuous(breaks = c(seq(-2, 2.5, 0.5))) + coord_cartesian(xlim = c(-2,
    2.5), ylim = c(-3, 3.5)) + xlab("Component 1") + ylab("Component 2") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Component Plot by Species")
```

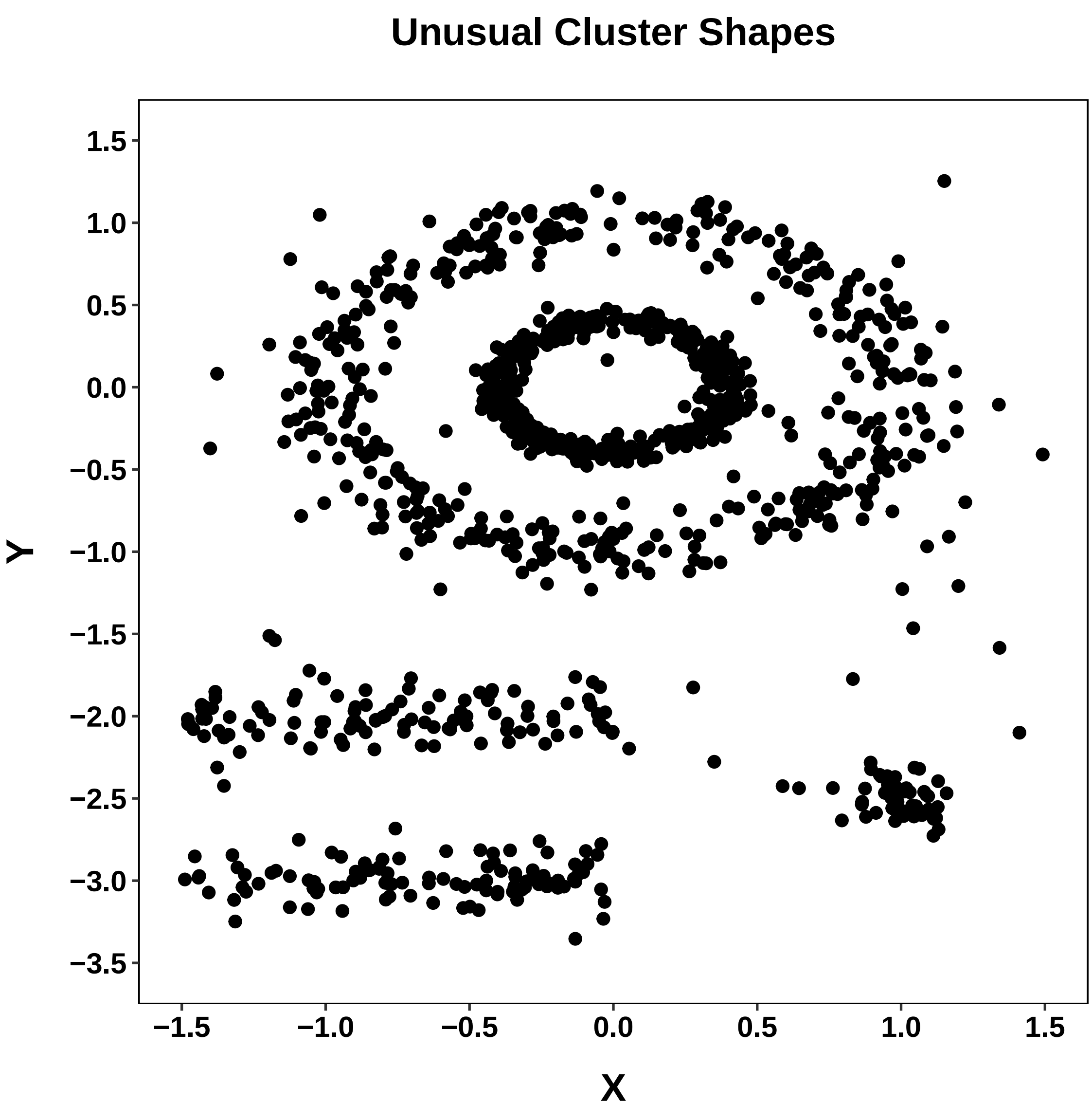


2.3.2 Odd Shapes Example

```
data("multishapes", package = "factoextra")
odd_shapes <- multishapes[, 1:2]
plot_data <- as.data.frame(odd_shapes)
names(plot_data) <- c("X", "Y")
```

```
ggplot(plot_data, aes(x = X, y = Y)) + geom_point(shape = 19, size = 2) +
    scale_y_continuous(breaks = c(seq(-3.5, 1.5, 0.5))) + scale_x_continuous(breaks = c(seq(-1.5,
    1.5, 0.5))) + coord_cartesian(xlim = c(-1.5, 1.5), ylim = c(-3.5,
    1.5)) + xlab("X") + ylab("Y") + theme(text = element_text(size = 14,
    family = "sans", color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
```

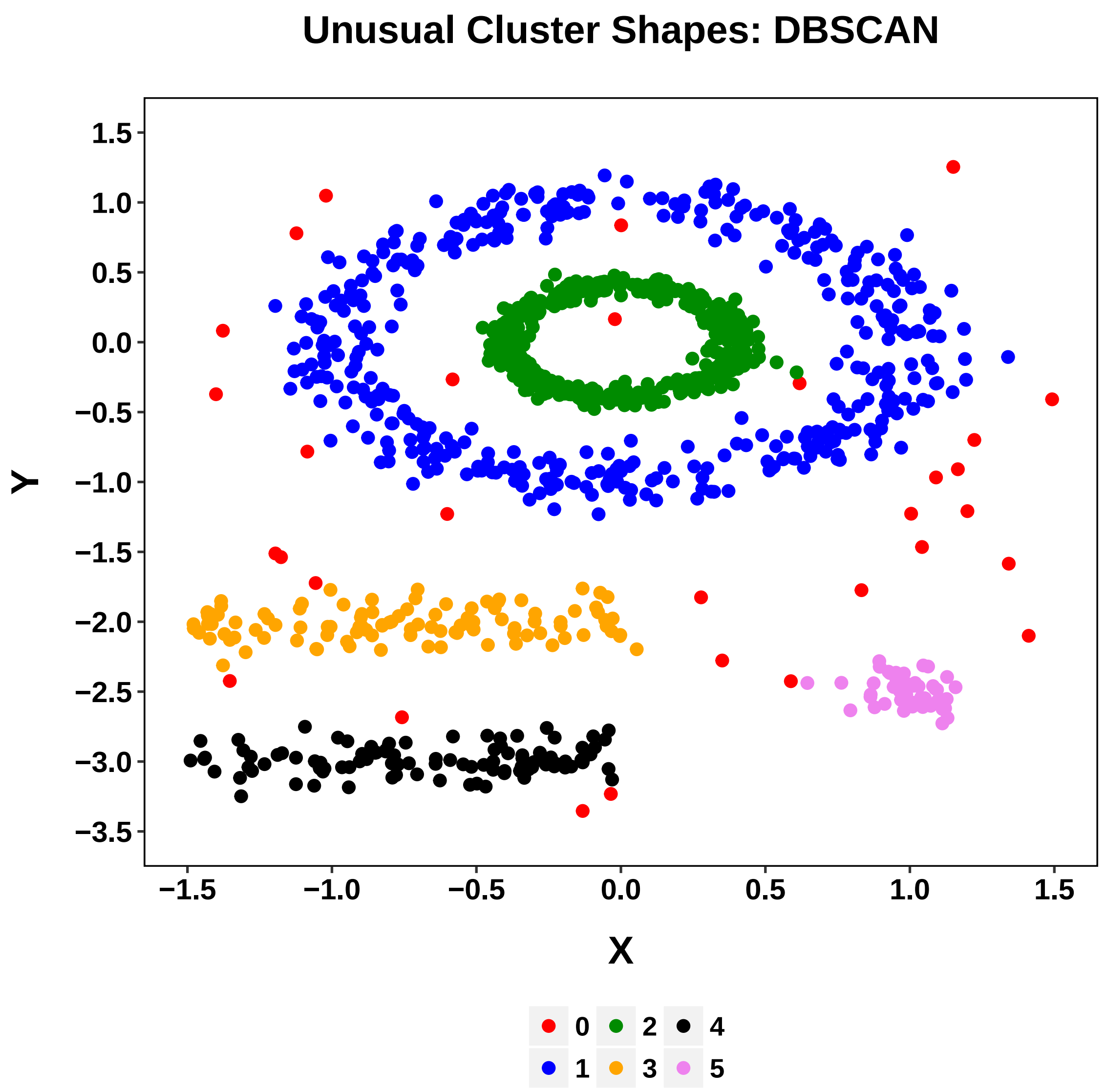
```
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
    plot.title = element_text(size = 16, face = "bold", margin = margin(0,
        0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
        linetype = 1, color = "black"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_blank()) + ggtitle("Unusual Cluster Shapes")
```




```
odd_shapes_fit <- fpc::dbscan(odd_shapes, eps = 0.15, MinPts = 5)
plot_data <- cbind(odd_shapes, odd_shapes_fit$cluster)
```

```r
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("X", "Y", "Cluster")
plot_data$Cluster_F <- factor(plot_data$Cluster)
```

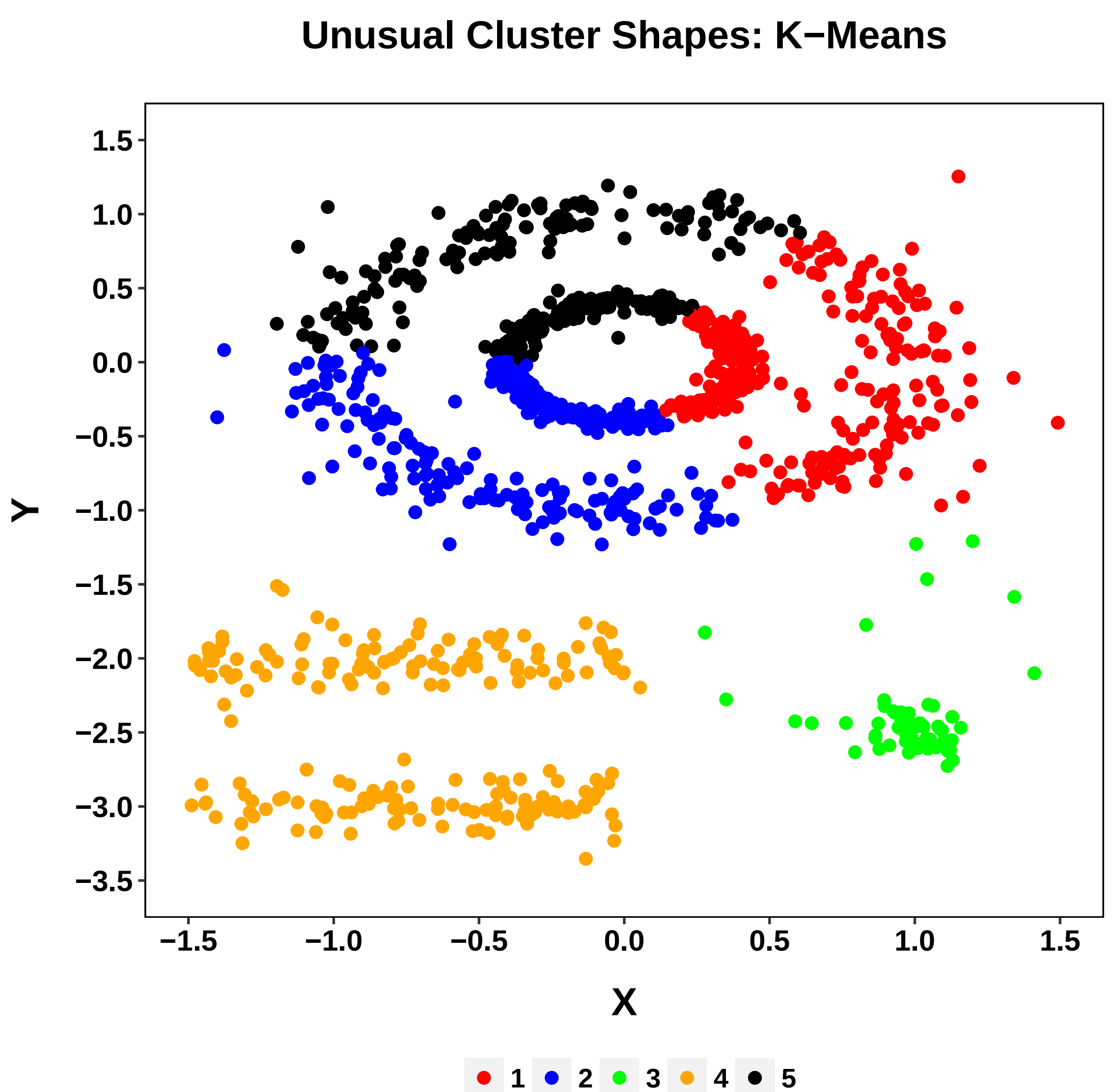
```r
ggplot(plot_data, aes(x = X, y = Y, color = Cluster_F)) + geom_point(shape = 19,
    size = 2) + scale_color_manual(values = c("red", "blue", "green4",
    "orange", "black", "violet")) + scale_y_continuous(breaks = c(seq(-3.5,
    1.5, 0.5))) + scale_x_continuous(breaks = c(seq(-1.5, 1.5, 0.5))) +
    coord_cartesian(xlim = c(-1.5, 1.5), ylim = c(-3.5, 1.5)) + xlab("X") +
    ylab("Y") + theme(text = element_text(size = 14, family = "sans",
    color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
    plot.title = element_text(size = 16, face = "bold", margin = margin(0,
        0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
        linetype = 1, color = "black"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_blank()) + ggtitle("Unusual Cluster Shapes: DBSCAN")
```




```r
odd_shapes_fit <- kmeans(odd_shapes, centers = 5, iter.max = 1000,
    nstart = 10)
plot_data <- cbind(odd_shapes, odd_shapes_fit$cluster)
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("X", "Y", "Cluster")
plot_data$Cluster_F <- factor(plot_data$Cluster)
```

```r
ggplot(plot_data, aes(x = X, y = Y, color = Cluster_F)) + geom_point(shape = 19,
    size = 2) + scale_color_manual(values = c("red", "blue", "green",
    "orange", "black", "violet")) + scale_y_continuous(breaks = c(seq(-3.5,
    1.5, 0.5))) + scale_x_continuous(breaks = c(seq(-1.5, 1.5, 0.5))) +
    coord_cartesian(xlim = c(-1.5, 1.5), ylim = c(-3.5, 1.5)) + xlab("X") +
    ylab("Y") + theme(text = element_text(size = 14, family = "sans",
```

```
    color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
    plot.title = element_text(size = 16, face = "bold", margin = margin(0,
        0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
        linetype = 1, color = "black"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_blank()) + ggtitle("Unusual Cluster Shapes: K-Means")
```



2.4 Fuzzy Clustering

Fuzzy clustering is different from other methods we have covered because data objects are treated as members of all clusters with varying degrees of fuzzy membership, indexed by a probability between 0 and 1. Objects closer to the centers of clusters have higher degrees of membership than objects nearer the borders of clusters. This provides a way to gauge the certainty or confidence with which objects can be classified into categories.

The Fuzzy C-Means (FCM) clustering algorithm is typically attributed to Bezdek (1974, 1981). In the algorithm the parameter, m, specifies the amount of 'fuzziness' of the clustering result. The default value is 2. As m increases, fuzzier clusters are produced. When m is 1, FCM produces the same results as the K-Means procedure.

Bezdek, J.C. (1974). Cluster validity with fuzzy sets. Journal of Cybernetics, 3, 58-73.
Bezdek J.C. (1981). Pattern recognition with fuzzy objective function algorithms. New York: Plenum.

```
Iris_Fuzzy <- fcm(Iris[, 1:4], centers = 3, m = 2, dmetric = "euclidean",
    iter.max = 1000, nstart = 10)
Iris_Fuzzy$u

##     Cluster 1 Cluster 2 Cluster 3
## 1     0.88099   0.06641   0.05260
## 2     0.07755   0.25879   0.66366
## 3     0.07690   0.55304   0.37006
## 4     0.09280   0.27308   0.63413
## 5     0.06521   0.37801   0.55678
## 6     0.80102   0.11077   0.08821
## 7     0.08893   0.28351   0.62755
## 8     0.10714   0.56618   0.32668
## 9     0.08488   0.51257   0.40255
## 10    0.73428   0.14637   0.11935
## 11    0.04952   0.75869   0.19180
## 12    0.07582   0.45634   0.46785
## 13    0.02896   0.11914   0.85191
## 14    0.18390   0.55211   0.26400
## 15    0.04934   0.18419   0.76647
## 16    0.08927   0.44135   0.46938
## 17    0.10694   0.29019   0.60287
## 18    0.88555   0.06456   0.04989
## 19    0.08082   0.70962   0.20957
## 20    0.09486   0.30531   0.59983
## 21    0.17057   0.34017   0.48926
## 22    0.08259   0.49450   0.42291
## 23    0.10778   0.29474   0.59748
## 24    0.15479   0.33486   0.51036
## 25    0.17848   0.49533   0.32619
## 26    0.77281   0.12618   0.10101
## 27    0.07425   0.25201   0.67375
## 28    0.11325   0.40439   0.48236
## 29    0.08687   0.56350   0.34963
## 30    0.11774   0.64525   0.23701
## 31    0.82871   0.09613   0.07516
## 32    0.07830   0.41773   0.50396
```

```
## 33    0.20963   0.51945   0.27092
## 34    0.07563   0.37757   0.54681
## 35    0.07636   0.26201   0.66163
## 36    0.87013   0.07243   0.05744
## 37    0.83330   0.09263   0.07408
## 38    0.12131   0.59637   0.28233
## 39    0.16956   0.34088   0.48956
## 40    0.73656   0.14626   0.11718
## 41    0.08950   0.26586   0.64464
## 42    0.85100   0.08359   0.06541
## 43    0.14198   0.61248   0.24554
## 44    0.71823   0.15661   0.12517
## 45    0.16024   0.33957   0.50020
## 46    0.06448   0.40200   0.53352
## 47    0.82432   0.09844   0.07724
## 48    0.12738   0.61230   0.26031
## 49    0.33082   0.40702   0.26216
## 50    0.12038   0.30994   0.56968
## 51    0.79493   0.11466   0.09041
## 52    0.81381   0.10375   0.08243
## 53    0.06642   0.44298   0.49060
## 54    0.82097   0.10095   0.07807
## 55    0.82299   0.09952   0.07749
## 56    0.81172   0.10465   0.08363
## 57    0.09832   0.37182   0.52986
## 58    0.07196   0.24705   0.68099
## 59    0.68796   0.17250   0.13955
## 60    0.65690   0.18851   0.15459
## 61    0.77226   0.12805   0.09969
## 62    0.10908   0.50829   0.38263
## 63    0.06511   0.48803   0.44686
## 64    0.80427   0.10942   0.08632
## 65    0.87565   0.06926   0.05508
## 66    0.09346   0.65885   0.24769
## 67    0.14373   0.60756   0.24871
## 68    0.94478   0.03099   0.02423
## 69    0.78831   0.11797   0.09372
## 70    0.02099   0.90613   0.07288
## 71    0.25012   0.48447   0.26541
## 72    0.61529   0.21130   0.17342
## 73    0.85609   0.08005   0.06386
## 74    0.14632   0.32925   0.52443
## 75    0.10280   0.30222   0.59498
## 76    0.08927   0.44135   0.46938
## 77    0.10784   0.67218   0.21998
## 78    0.11258   0.30940   0.57802
## 79    0.82774   0.09645   0.07582
## 80    0.72081   0.15434   0.12485
## 81    0.11020   0.29528   0.59452
## 82    0.05454   0.20501   0.74045
## 83    0.07106   0.52945   0.39948
## 84    0.06653   0.24701   0.68645
## 85    0.33929   0.40084   0.25987
## 86    0.12197   0.64126   0.23677
```

```
## 87    0.12008   0.64588   0.23404
## 88    0.67461   0.17918   0.14621
## 89    0.84302   0.08793   0.06906
## 90    0.16741   0.34273   0.48986
## 91    0.10437   0.49322   0.40240
## 92    0.73864   0.14592   0.11544
## 93    0.09164   0.52714   0.38122
## 94    0.20275   0.52106   0.27619
## 95    0.10019   0.57188   0.32793
## 96    0.91798   0.04584   0.03618
## 97    0.70936   0.16194   0.12870
## 98    0.29866   0.42433   0.27701
## 99    0.23174   0.49757   0.27069
## 100   0.15558   0.59550   0.24892
## 101   0.79765   0.11259   0.08976
## 102   0.78110   0.12235   0.09655
## 103   0.10076   0.28139   0.61785
## 104   0.07209   0.72755   0.20036
## 105   0.13094   0.32196   0.54710
## 106   0.08377   0.71355   0.20268
## 107   0.91745   0.04615   0.03640
## 108   0.85243   0.08204   0.06553
## 109   0.10427   0.49076   0.40496
## 110   0.17268   0.54144   0.28588
## 111   0.13464   0.32494   0.54042
## 112   0.09722   0.30078   0.60201
## 113   0.82926   0.09554   0.07519
## 114   0.08849   0.43141   0.48010
## 115   0.10986   0.58506   0.30508
## 116   0.79139   0.11615   0.09246
## 117   0.13748   0.61671   0.24581
## 118   0.07878   0.49483   0.42639
## 119   0.06916   0.68035   0.25049
## 120   0.09122   0.64061   0.26816
## 121   0.37638   0.37668   0.24694
## 122   0.12194   0.64879   0.22927
## 123   0.08894   0.26971   0.64134
## 124   0.07730   0.24509   0.67761
## 125   0.73603   0.14601   0.11796
## 126   0.88342   0.06498   0.05160
## 127   0.14169   0.32371   0.53460
## 128   0.05217   0.21263   0.73520
## 129   0.05115   0.72800   0.22085
## 130   0.21999   0.51519   0.26482
## 131   0.09059   0.35617   0.55324
## 132   0.05125   0.21720   0.73156
## 133   0.07216   0.23217   0.69568
## 134   0.16619   0.56962   0.26419
## 135   0.87467   0.07014   0.05519
## 136   0.78501   0.12002   0.09497
## 137   0.62476   0.20892   0.16632
## 138   0.10445   0.46628   0.42927
## 139   0.69574   0.16983   0.13443
## 140   0.81477   0.10306   0.08217
```

```
## 141   0.17582   0.54791   0.27628
## 142   0.09889   0.54909   0.35202
## 143   0.09203   0.41829   0.48968
## 144   0.82608   0.09776   0.07616
## 145   0.84216   0.08801   0.06983
## 146   0.88336   0.06514   0.05150
## 147   0.10792   0.39248   0.49959
## 148   0.05868   0.26265   0.67867
## 149   0.11983   0.31627   0.56390
## 150   0.81399   0.10359   0.08243
```

```
Iris_Fuzzy$v
```

```
##           Sepal_Length Sepal_Width Petal_Length Petal_Width
## Cluster 1        50.50       33.82        16.22        3.09
## Cluster 2        60.21       28.41        44.86       14.59
## Cluster 3        64.89       29.73        52.24       18.61
```

```
Iris_Fuzzy$d
```

```
##     Cluster 1 Cluster 2 Cluster 3
## 1       2.650   35.1548    44.389
## 2      46.436   13.9159     5.426
## 3      35.677    4.9609     7.414
## 4      47.958   16.2970     7.018
## 5      39.265    6.7738     4.599
## 6       5.023   36.3252    45.616
## 7      44.230   13.8743     6.268
## 8      35.244    6.6690    11.558
## 9      36.166    5.9891     7.626
## 10      8.055   40.4093    49.558
## 11     33.627    2.1947     8.682
## 12     38.900    6.4629     6.304
## 13     42.321   10.2862     1.438
## 14     26.260    8.7466    18.292
## 15     44.172   11.8335     2.844
## 16     39.568    8.0038     7.526
## 17     50.361   18.5581     8.933
## 18      2.275   31.2052    40.380
## 19     31.666    3.6064    12.212
## 20     44.229   13.7411     6.994
## 21     60.466   30.3191    21.080
## 22     35.652    5.9549     6.963
## 23     49.154   17.9740     8.867
## 24     58.854   27.2047    17.850
## 25     33.195   11.9609    18.163
## 26      5.759   35.2683    44.057
## 27     44.312   13.0551     4.883
## 28     38.080   10.6644     8.940
## 29     33.997    5.2408     8.446
## 30     28.423    5.1866    14.120
## 31      3.928   33.8659    43.312
## 32     38.973    7.3055     6.056
## 33     25.567   10.3174    19.782
## 34     40.352    8.0827     5.581
```

```
## 35     43.987   12.8193     5.077
## 36      2.894   34.7737    43.845
## 37      4.039   36.3318    45.431
## 38     31.577    6.4231    13.568
## 39     58.299   28.9984    20.191
## 40      7.558   38.0641    47.508
## 41     47.506   15.9933     6.596
## 42      3.189   32.4666    41.493
## 43     27.610    6.4002    15.965
## 44      8.270   37.9278    47.455
## 45     60.006   28.3158    19.223
## 46     38.726    6.2117     4.680
## 47      4.099   34.3293    43.752
## 48     30.538    6.3533    14.944
## 49     21.134   17.1779    26.670
## 50     51.036   19.8225    10.785
## 51      5.075   35.1847    44.622
## 52      4.357   34.1750    43.015
## 53     37.707    5.6535     5.105
## 54      3.903   31.7382    41.038
## 55      4.007   33.1344    42.554
## 56      4.746   36.8116    46.063
## 57     43.278   11.4444     8.031
## 58     46.038   13.4102     4.865
## 59     10.122   40.3675    49.899
## 60     10.650   37.1110    45.253
## 61      5.130   30.9424    39.742
## 62     34.220    7.3436     9.755
## 63     37.396    4.9894     5.449
## 64      4.788   35.1955    44.616
## 65      2.784   35.1967    44.257
## 66     31.860    4.5195    12.022
## 67     27.652    6.5417    15.981
## 68      1.078   32.8673    42.030
## 69      5.448   36.4052    45.826
## 70     32.920    0.7625     9.480
## 71     22.447   11.5886    21.154
## 72     12.175   35.4525    43.196
## 73      3.335   35.6708    44.710
## 74     55.844   24.8169    15.581
## 75     46.649   15.8682     8.060
## 76     39.568    8.0038     7.526
## 77     28.782    4.6176    14.110
## 78     48.879   17.7862     9.521
## 79      3.817   32.7613    41.676
## 80      7.726   36.0826    44.608
## 81     50.851   18.9777     9.425
## 82     43.778   11.6460     3.225
## 83     36.568    4.9081     6.505
## 84     44.546   11.9987     4.318
## 85     20.689   17.5122    27.011
## 86     28.977    5.5112    14.927
## 87     28.301    5.2616    14.520
## 88      9.659   36.3651    44.568
```

```
## 89      3.586  34.3831   43.780
## 90     62.814  30.6832   21.467
## 91     38.897   8.2311   10.089
## 92      6.367  32.2303   40.738
## 93     35.335   6.1430    8.495
## 94     26.207  10.1975   19.239
## 95     32.945   5.7716   10.065
## 96      1.719  34.4286   43.617
## 97      8.461  37.0624   46.636
## 98     24.325  17.1205   26.225
## 99     24.419  11.3728   20.905
## 100    26.479   6.9178   16.550
## 101     5.207  36.8868   46.268
## 102     5.557  35.4745   44.955
## 103    49.042  17.5616    7.998
## 104    31.159   3.0876   11.212
## 105    53.335  21.6906   12.764
## 106    29.889   3.5090   12.353
## 107     1.717  34.1455   43.286
## 108     3.466  36.0104   45.086
## 109    37.745   8.0197    9.719
## 110    29.038   9.2609   17.540
## 111    54.224  22.4685   13.510
## 112    48.146  15.5614    7.775
## 113     4.005  34.7617   44.169
## 114    37.007   7.5904    6.821
## 115    33.812   6.3493   12.176
## 116     5.003  34.0912   42.824
## 117    27.517   6.1344   15.391
## 118    38.085   6.0631    7.036
## 119    34.220   3.4787    9.449
## 120    31.952   4.5500   10.869
## 121    18.180  18.1654   27.709
## 122    28.079   5.2774   14.934
## 123    48.251  15.9116    6.691
## 124    45.808  14.4473    5.226
## 125     7.091  35.7441   44.244
## 126     2.563  34.8515   43.891
## 127    54.336  23.7842   14.402
## 128    41.343  10.1431    2.934
## 129    34.641   2.4338    8.023
## 130    23.302   9.9502   19.357
## 131    39.581  10.0678    6.482
## 132    42.834  10.1066    3.001
## 133    46.321  14.3960    4.804
## 134    27.601   8.0528   17.363
## 135     2.745  34.2264   43.503
## 136     5.488  35.8932   45.361
## 137    12.555  37.5452   47.162
## 138    39.334   8.8115    9.571
## 139     7.767  31.8205   40.199
## 140     4.386  34.6765   43.492
## 141    28.303   9.0820   18.011
## 142    33.847   6.0960    9.509
```

```
## 143    38.330    8.4333     7.204
## 144     3.750   31.6898    40.674
## 145     3.563   34.0896    42.963
## 146     2.512   34.0687    43.086
## 147    41.679   11.4604     9.003
## 148    40.268    8.9966     3.482
## 149    50.536   19.1466    10.738
## 150     4.363   34.2874    43.090
```

```
Iris_Fuzzy$cluster
```

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

```
Iris_Fuzzy$csize
```

```
##  1  2  3
## 50 50 50
```

```
Iris_Fuzzy$best.start
```

```
## [1] 2
```

```
Iris_Class <- as.data.frame(cbind(Iris_Fuzzy$cluster, Iris$Species))
names(Iris_Class) <- c("Cluster", "Species")
table(Iris_Class$Species)
```

```
##
##     Setosa Versicolor  Virginica
##         50         50         50
```

```
table(Iris_Class$Cluster)
```

```
##
##  1  2  3
## 50 50 50
```

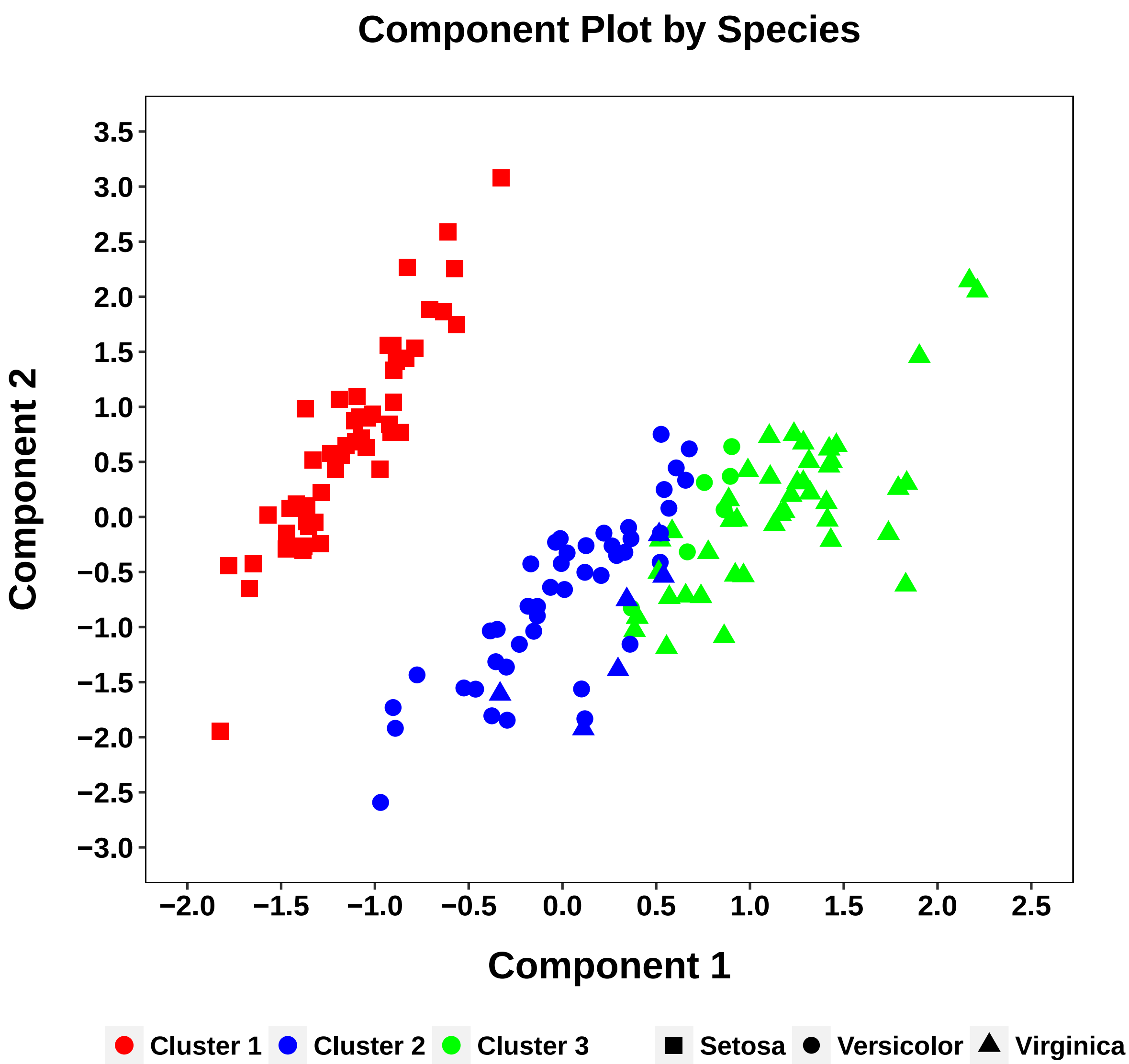
```
table(Iris_Class$Species, Iris_Class$Cluster)
```

```
##
##                1  2  3
##   Setosa      50  0  0
##   Versicolor   0 44  6
##   Virginica    0  6 44
```

2.4.1 Dimensional Plot for Iris Data

```
# Use PCA to show potential clustering along two dimensions.
PCA <- principal(Iris[, 1:4], nfactors = 2, rotate = "varimax", scores = TRUE)
Iris_Cluster <- Iris_Fuzzy$cluster
plot_data <- cbind(Iris, PCA$scores, Iris_Cluster, Iris$Species)
plot_data$Cluster_F <- factor(plot_data$Iris_Cluster, levels = c(1,
    2, 3), labels = c("Cluster 1", "Cluster 2", "Cluster 3"))
plot_data$Species_F <- factor(plot_data$Species)
```

```
ggplot(plot_data, aes(x = RC1, y = RC2, color = Cluster_F, shape = Species_F)) +
    geom_point(size = 3) + scale_color_manual(values = c("red", "blue",
    "green", "orange", "black")) + scale_shape_manual(values = c(15,
    16, 17, 18)) + scale_y_continuous(breaks = c(seq(-3, 3.5, 0.5))) +
    scale_x_continuous(breaks = c(seq(-2, 2.5, 0.5))) + coord_cartesian(xlim = c(-2,
    2.5), ylim = c(-3, 3.5)) + xlab("Component 1") + ylab("Component 2") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Component Plot by Species")
```

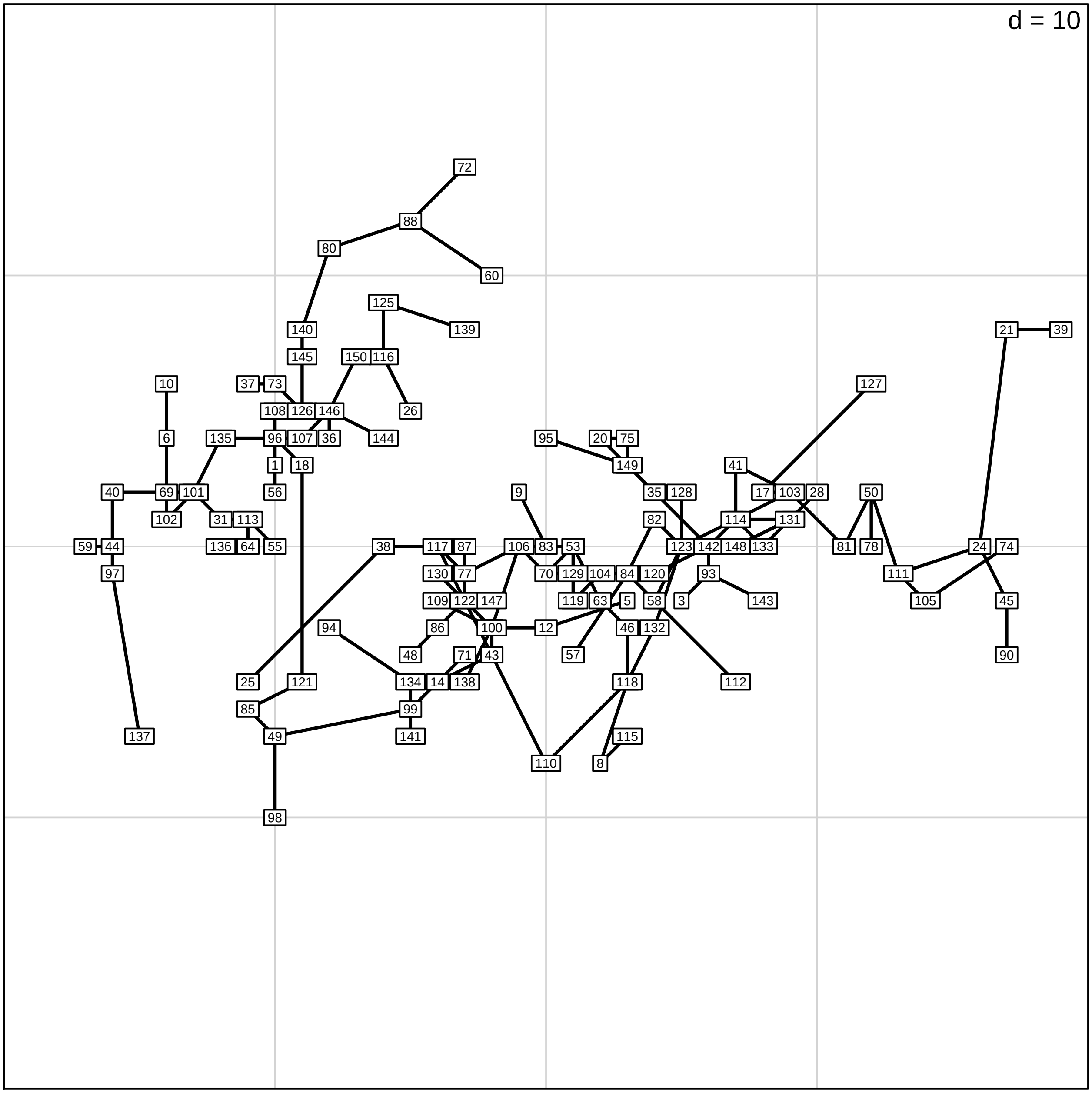


2.5 Minimum Spanning Trees

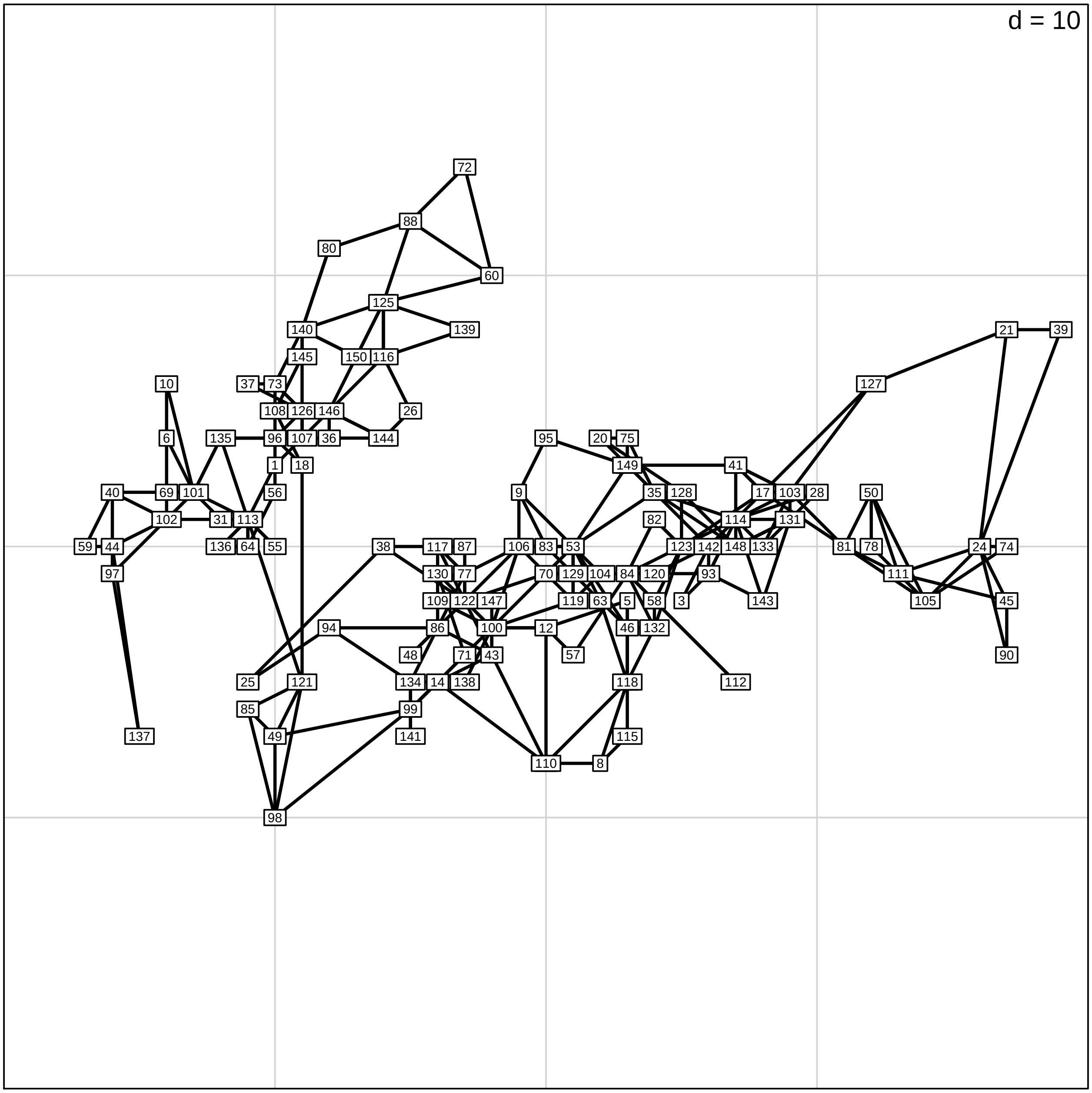
A dissimilarity matrix can be represented in graph theory as an undirected graph with objects as vertices (or nodes) and distances as edge weights. A minimum spanning tree (MST) is then the subset of the edges that connect all the vertices together without any cycles and with the minimum possible sum of edge weights. In simple terms, if the data have a cluster structure, a minimum spanning tree will have numerous interconnected paths within a cluster but few paths between clusters.

The restriction that there be no cycles can be relaxed and successive layers of minimum edges added to the graph. This can highlight clustering in the data to the extent that within-cluster edges outnumber between-cluster edges.

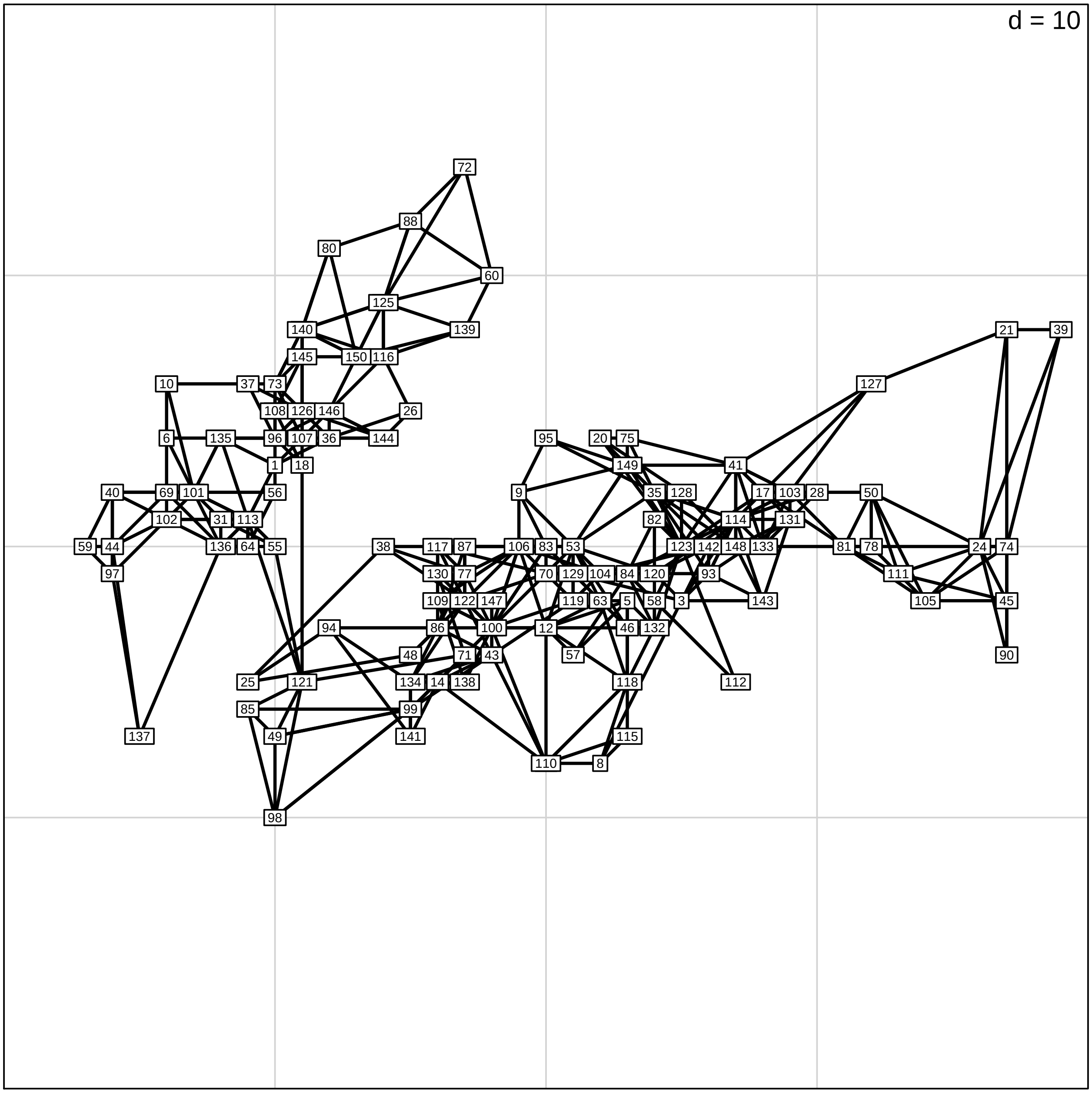
```r
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 1)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```



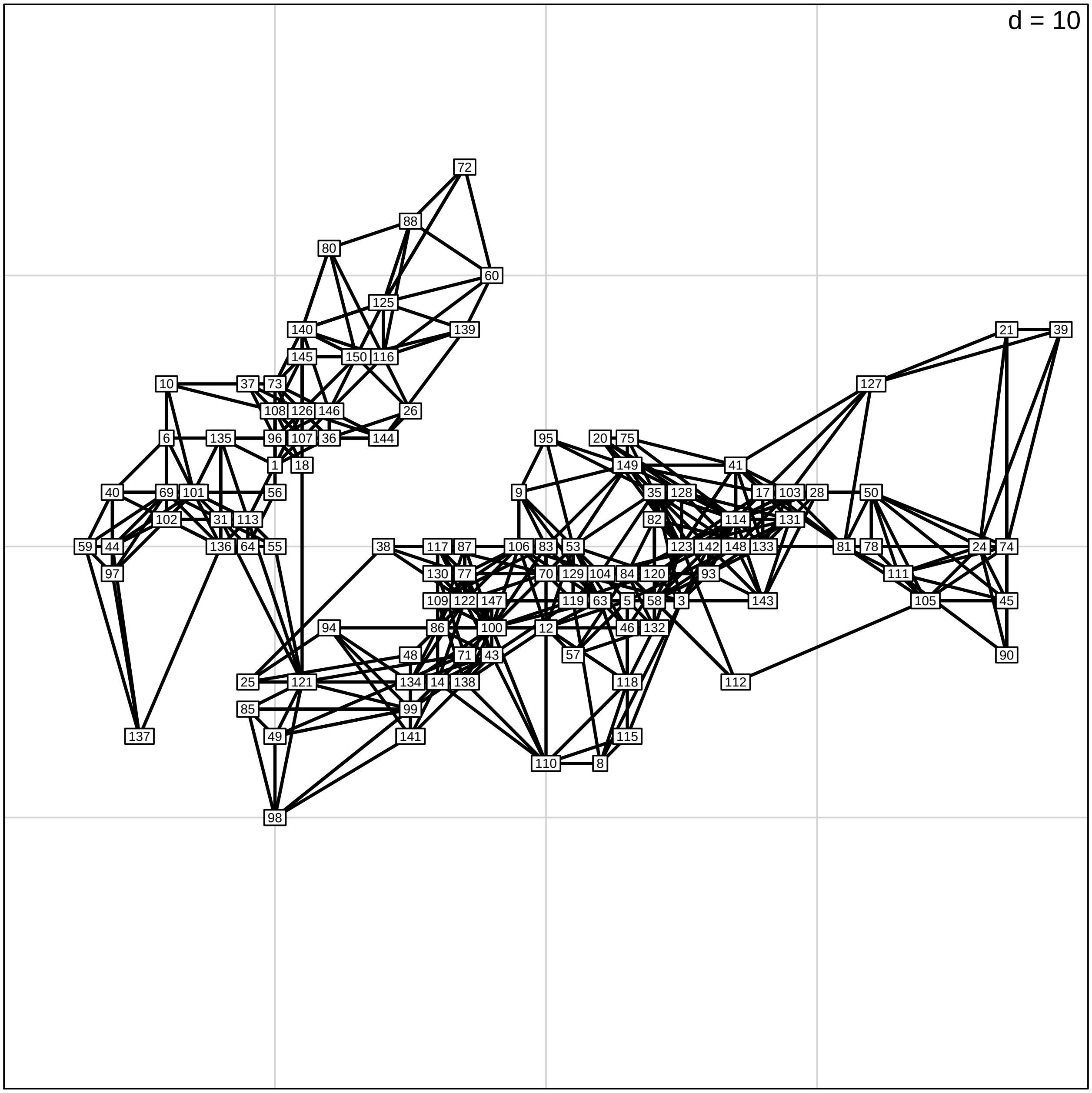

```r
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 2)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```



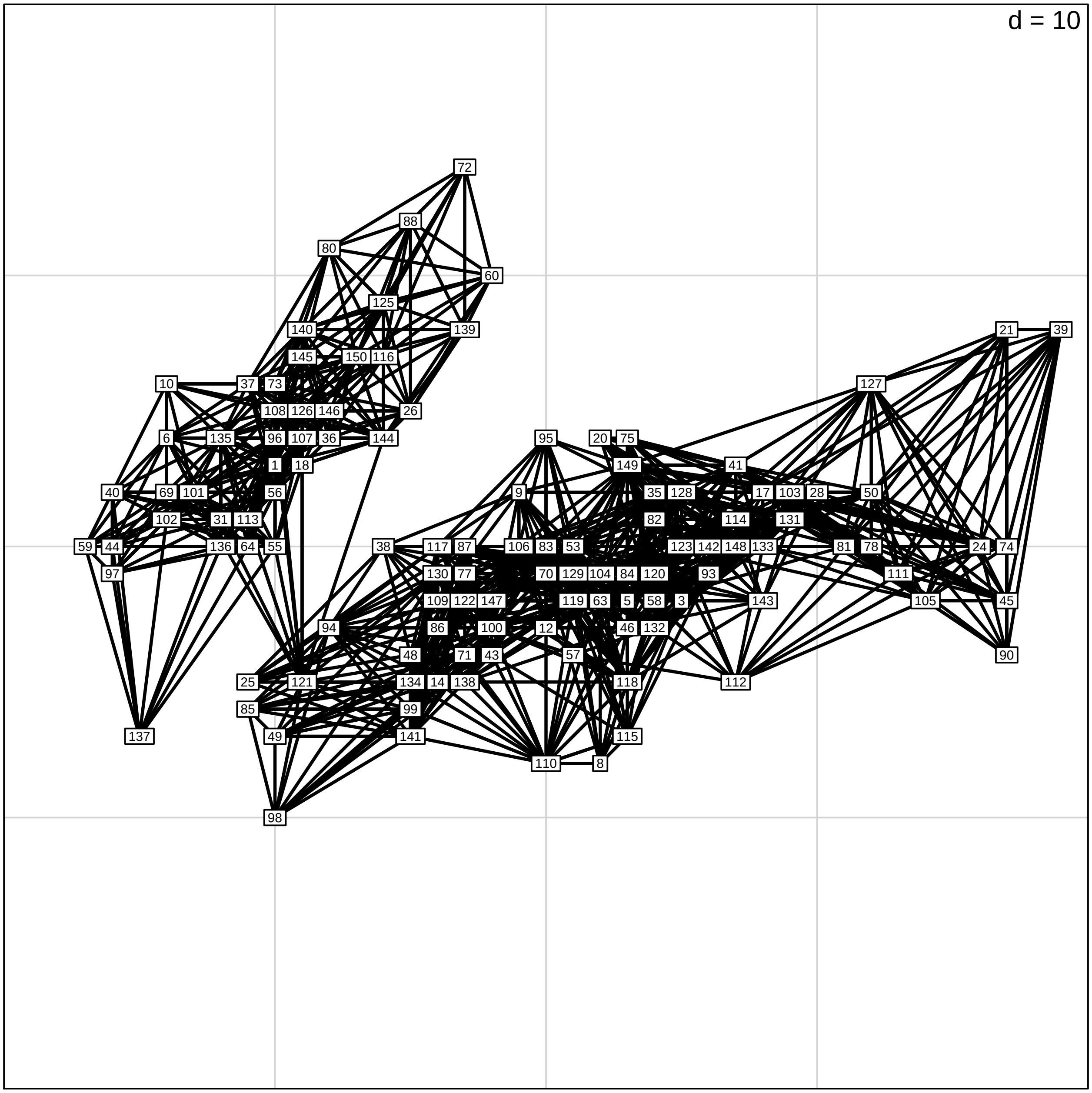

```
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 3)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```



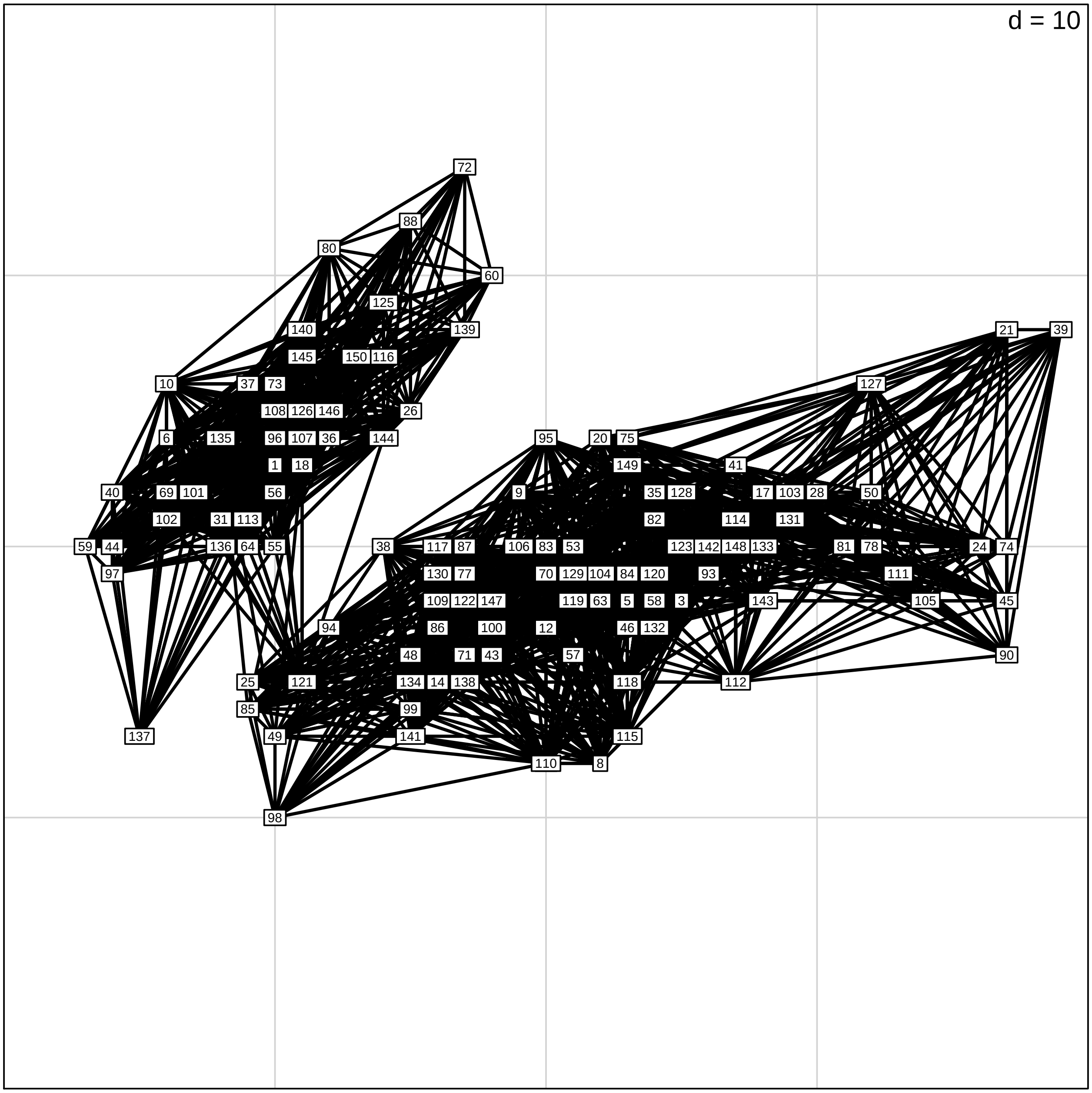

```r
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 4)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```



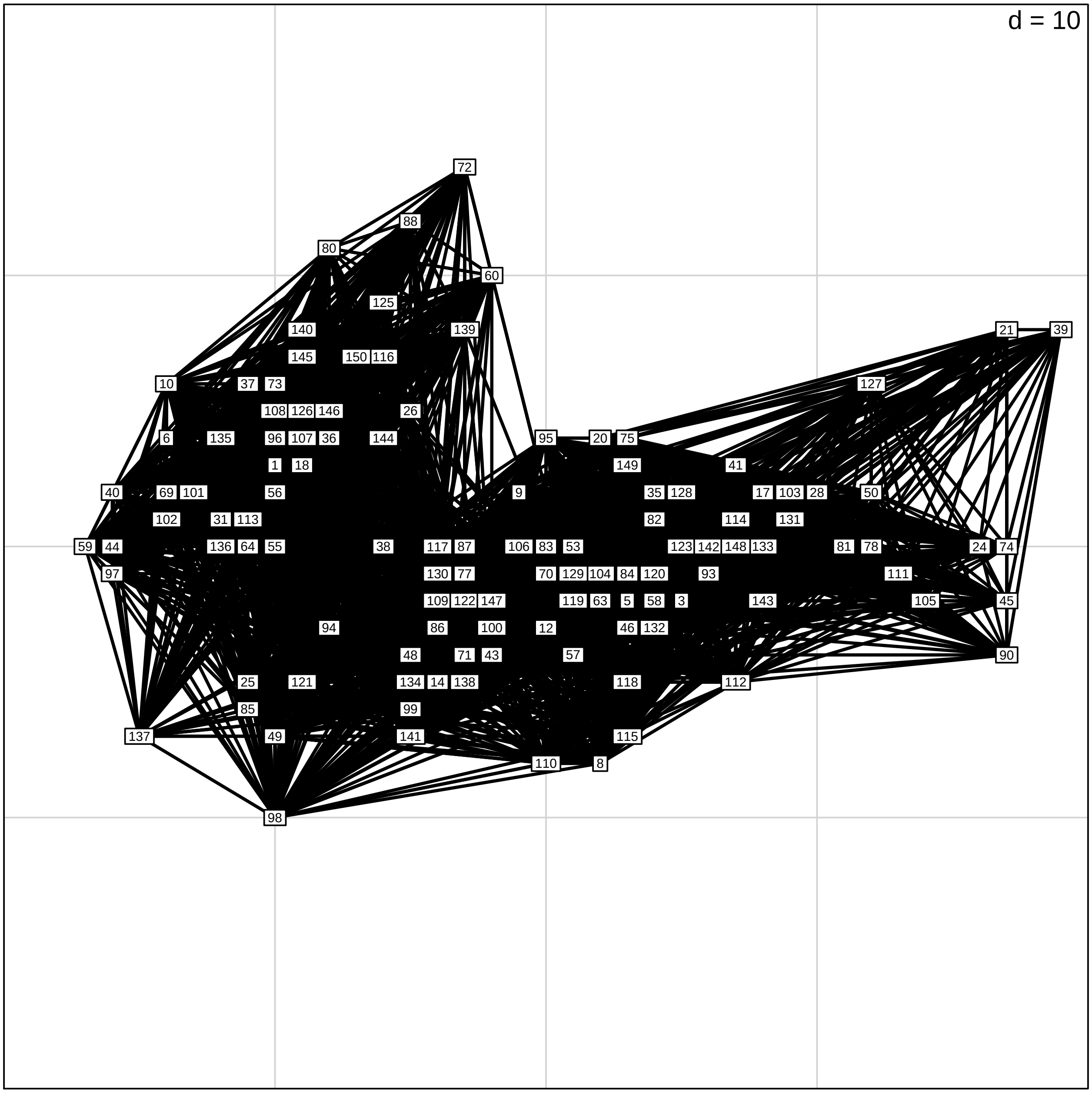
```
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 10)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```




```r
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 20)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```




```r
Iris_MST <- ade4::mstree(dist(Iris[, 1:4], method = "euclidean"),
    ngmax = 40)
s.label(Iris[, 1:4], xlim = c(50, 70), ylim = c(10, 50), addaxes = TRUE,
    neig = Iris_MST, clabel = 0.5, pch = 16)
```



3 Methods to Determine Cluster Quality or Number

The "quality" of a clustering solution can be assessed in a number of ways that often reflect the goal of a particular clustering method.

3.1 Silhouette Coefficient

The silhouette score is calculated using two values for each object, a_i ***and*** b_i***. The value,*** a_i***, is the average distance between*** $Object_i$ ***and all other objects in the same cluster. The value,*** b_i***, is the smallest average distance of*** $Object_i$ ***to all objects in another cluster. These two values are sometimes called cohesion and separation, respectively. The silhouette score (sometimes called its width),*** s_i***, is defined as:***

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

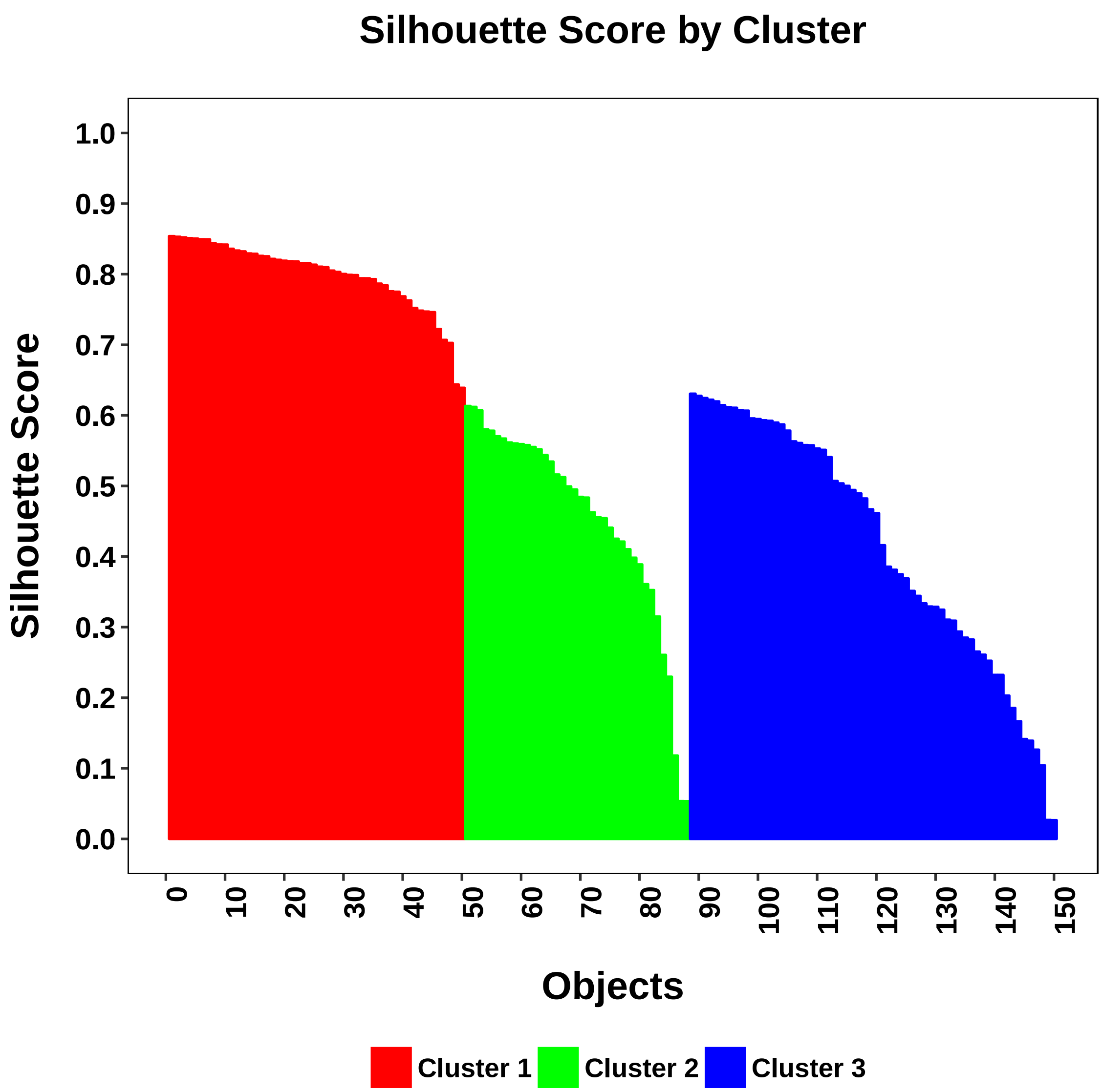
It can take on values between -1 and 1, with higher values indicating that an object is well matched to its cluster and a poor match to any neighboring clusters.

The average silhouette score is provided for each cluster and for the overall solution. The overall average, sometimes called the silhouette coefficient, is an index of cluster quality. Coefficients that approach 1 represent very clear evidence that the chosen cluster number produces a good cluster solution. Some common benchmarks for the average cluster silhouette coefficient:

- *.71 to 1.00: A strong structure has been found*
- *.51 to .70: A reasonable structure has been found*
- *.26 to .50: The structure is weak and could be artificial*
- *< .25: No substantial structure has been found*

The average score for different numbers of clusters can be plotted to give a visual display of cluster quality. The number of clusters with the highest average is the best solution. A range of clusters can be examined using the pamk( ) function form the fpc library, with the optimal number of clusters identified using the silhouette coefficient.

```
plot_data <- cbind(Iris_P$silinfo$widths[, 3], Iris_P$silinfo$widths[,
    1], seq(1, length(Iris[, 1]), 1))
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("Silhouette", "Cluster", "Object")
plot_data$Cluster_F <- factor(plot_data$Cluster, levels = c(1, 2,
    3), labels = c("Cluster 1", "Cluster 2", "Cluster 3"))


ggplot(plot_data, aes(x = Object, y = Silhouette, fill = Cluster_F,
    color = Cluster_F)) + geom_bar(stat = "identity") + scale_fill_manual(values = c("red",
    "green", "blue")) + scale_color_manual(values = c("red", "green",
    "blue")) + coord_cartesian(xlim = c(1, 150), ylim = c(0, 1)) +
    scale_y_continuous(breaks = seq(0, 1, 0.1)) + scale_x_continuous(breaks = seq(0,
    150, 10)) + xlab("Objects") + ylab("Silhouette Score") + theme(text = element_text(size = 14,
    family = "sans", color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 12, face = "bold", angle = 90, hjust = 1), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
    plot.title = element_text(size = 16, face = "bold", margin = margin(0,
        0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
        linetype = 1, color = "black"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_blank()) + ggtitle("Silhouette Score by Cluster")
```




```
pamk.best <- pamk(Iris[, 1:4])
pamk.best

## $pamobject
## Medoids:
##      ID Sepal_Length Sepal_Width Petal_Length Petal_Width
## [1,] 96           50          34           15           2
## [2,] 63           62          28           48          18
## Clustering vector:
##   [1] 1 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2
##  [31] 1 2 2 2 2 1 1 2 2 1 2 1 2 1 2 2 1 2 2 2 1 1 2 1 1 1 2 2 1 1
##  [61] 1 2 2 1 1 2 2 1 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 1 2
##  [91] 2 1 2 2 2 1 1 2 2 2 1 1 2 2 2 2 1 1 2 2 2 2 1 2 2 1 2 2 2 2
## [121] 1 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1 1 2 1 1 2 2 2 1 1 1 2 2 2 1
```

```
## Objective function:
## build  swap
## 9.901 8.622
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"
##  [5] "isolation"  "clusinfo"   "silinfo"    "diss"
##  [9] "call"       "data"
##
## $nc
## [1] 2
##
## $crit
##  [1] 0.0000 0.6858 0.5528 0.4897 0.4867 0.4704 0.3390 0.3319
##  [9] 0.2963 0.2963
```

```
cat("Number of clusters estimated by optimum average silhouette width:",
    pamk.best$nc)
```

```
## Number of clusters estimated by optimum average silhouette width: 2
```

3.2 Cophenetic Correlation

The cophenetic distance between two observations that have been clustered hierarchically is defined to be the intergroup dissimilarity at which the two observations are first combined into a single cluster. Note that this distance has many ties and restrictions. The correlation between the original distances and the cophenetic distances is an index of how well a dendrogram preserves the pairwise distances between the original objects.

```
d1 <- dist(Iris[, 1:4], method = "euclidean")
Iris_HC <- hclust(d1, "average")
d2 <- cophenetic(Iris_HC)
cor(d1, d2)
```

```
## [1] 0.877
```

```
Iris_HC <- hclust(d1, "single")
d2 <- cophenetic(Iris_HC)
cor(d1, d2)
```

```
## [1] 0.8639
```

```
Iris_HC <- hclust(d1, "complete")
d2 <- cophenetic(Iris_HC)
cor(d1, d2)
```

```
## [1] 0.7269
```

```
Iris_HC <- hclust(d1, "centroid")
d2 <- cophenetic(Iris_HC)
cor(d1, d2)
```

```
## [1] 0.8747
```

```r
Iris_HC <- hclust(d1, "ward.D2")
d2 <- cophenetic(Iris_HC)
cor(d1, d2)

## [1] 0.8728
```

3.3 Agglomerative Coefficient

For each object i, m(i) is its dissimilarity to the first cluster it is merged with, divided by the dissimilarity of the merger in the final step of the algorithm. The agglomerative coefficient is the average of all 1-m(i). This coefficient takes on values of 0 to 1. It grows with the number of observations, so this measure cannot be used to compare data sets of very different sizes.

```r
d1 <- dist(Iris[, 1:4], method = "euclidean")
Iris_HC <- hclust(d1, "average")
coef.hclust(Iris_HC)

## [1] 0.9296

Iris_HC <- hclust(d1, "single")
coef.hclust(Iris_HC)

## [1] 0.8493

Iris_HC <- hclust(d1, "complete")
coef.hclust(Iris_HC)

## [1] 0.9583

Iris_HC <- hclust(d1, "ward.D2")
coef.hclust(Iris_HC)

## [1] 0.9909
```

3.4 Pseudo F or Calinski-Harabasz Index

The Calinski-Harabasz index, also known as the Pseudo-F statistic, is used to help identify the proper number of clusters:

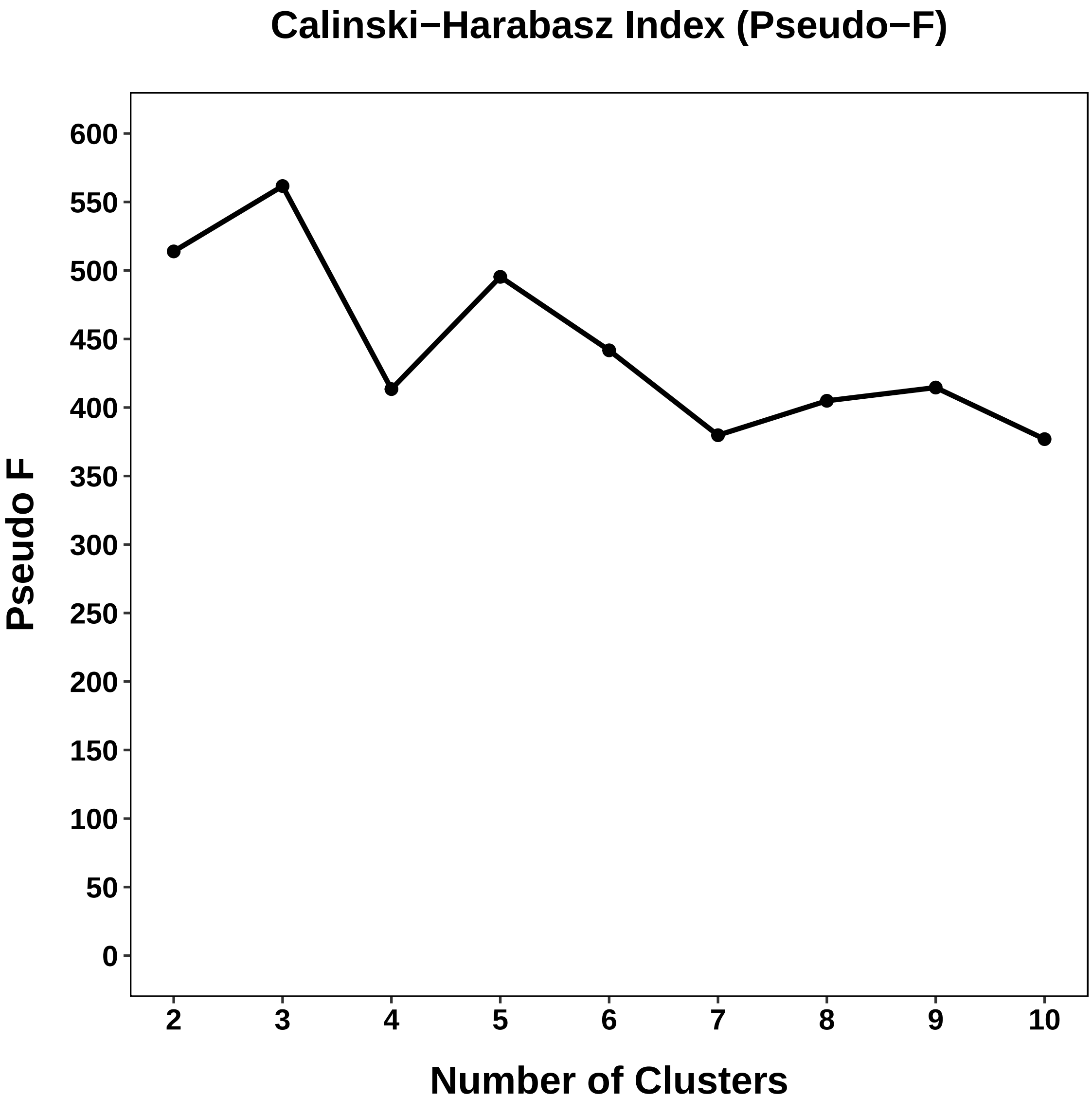
$$Pseudo\ \ F = \frac{\frac{SS_{BC}}{C-1}}{\frac{SS_{WC}}{N-C}}$$

It is best used with a method that assumes interval-level data because it resembles in form the calculation of an ANOVA, with the clusters representing the between-group structure and objects within the clusters representing the error structure. The calinhara( ) function is part of the fpc package.

```r
Pseudo_F <- matrix(NA, nrow = 10, ncol = 2)
for (j in 2:10) {
```

```
    Iris_K <- kmeans(Iris[, 1:4], centers = j)
    Pseudo_F[j, 1] <- j
    Pseudo_F[j, 2] <- calinhara(Iris[, 1:4], Iris_K$cluster)
}
```

```
plot_data <- Pseudo_F[2:10, ]
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("Number", "Pseudo_F")

ggplot(plot_data, aes(x = Number, y = Pseudo_F)) + geom_point(shape = 19,
    size = 2, color = "black", na.rm = TRUE) + geom_line(size = 1) +
    scale_x_continuous(breaks = c(seq(2, 10, 1))) + scale_y_continuous(breaks = c(seq(0,
    600, 50))) + coord_cartesian(xlim = c(2, 10), ylim = c(0, 600)) +
    xlab("Number of Clusters") + ylab("Pseudo F") + theme(text = element_text(size = 14,
    family = "sans", color = "black", face = "bold"), axis.text.y = element_text(colour = "black",
    size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
    size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
    0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
    15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
    plot.title = element_text(size = 16, face = "bold", margin = margin(0,
        0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
        linetype = 1, color = "black"), panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
    plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
    legend.title = element_blank()) + ggtitle("Calinski-Harabasz Index (Pseudo-F)")
```



3.5 Hopkins Statistic

This statistic examines whether objects in a data set differ significantly from the assumption that they are uniformly distributed in the multidimensional space. It compares the distances, x_i, between the real objects and their nearest neighbors to the distances, y_i, between artificial objects and their nearest neighbors, with the artificial objects uniformly generated over the data space. The H statistic is defined as:

$$H = \frac{\sum_{i=1}^{N} x_i}{\sum_{i=1}^{N} x_i + \sum_{i=1}^{N} y_i}$$

Values close to .5 indicate data are uniformly distributed. As H approaches 0, the data exhibit increasing clustering.

```
Iris_M <- as.matrix(Iris[, 1:4])
hopkins(Iris_M, n = 149, byrow = FALSE, header = FALSE)

## $H
## [1] 0.1613
```

3.6 Duda-Hart Statistic

The DudaHart test indicates if a data set should be split into two clusters. Variants exist for different clustering methods. The one used here appears suitable for interval level data for which sums of squares would be an appropriate calculation. The dh value calculated here is the ratio of the within-cluster sum of squares for two clusters to the overall sum of squares. The dudahart2( ) function is part of the fpc package. This is a very basic kind of test, but would indicate if the clustering effort should even be started.

```
Iris_K <- kmeans(Iris[, 1:4], centers = 2)
Duda_Hart <- dudahart2(Iris[, 1:4], Iris_K$cluster, alpha = 0.001)
Duda_Hart

## $p.value
## [1] 0
##
## $dh
## [1] 0.2236
##
## $compare
## [1] 0.6815
##
## $cluster1
## [1] FALSE
##
## $alpha
## [1] 0.001
##
## $z
## [1] 3.09
```

3.7 Gap Statistic

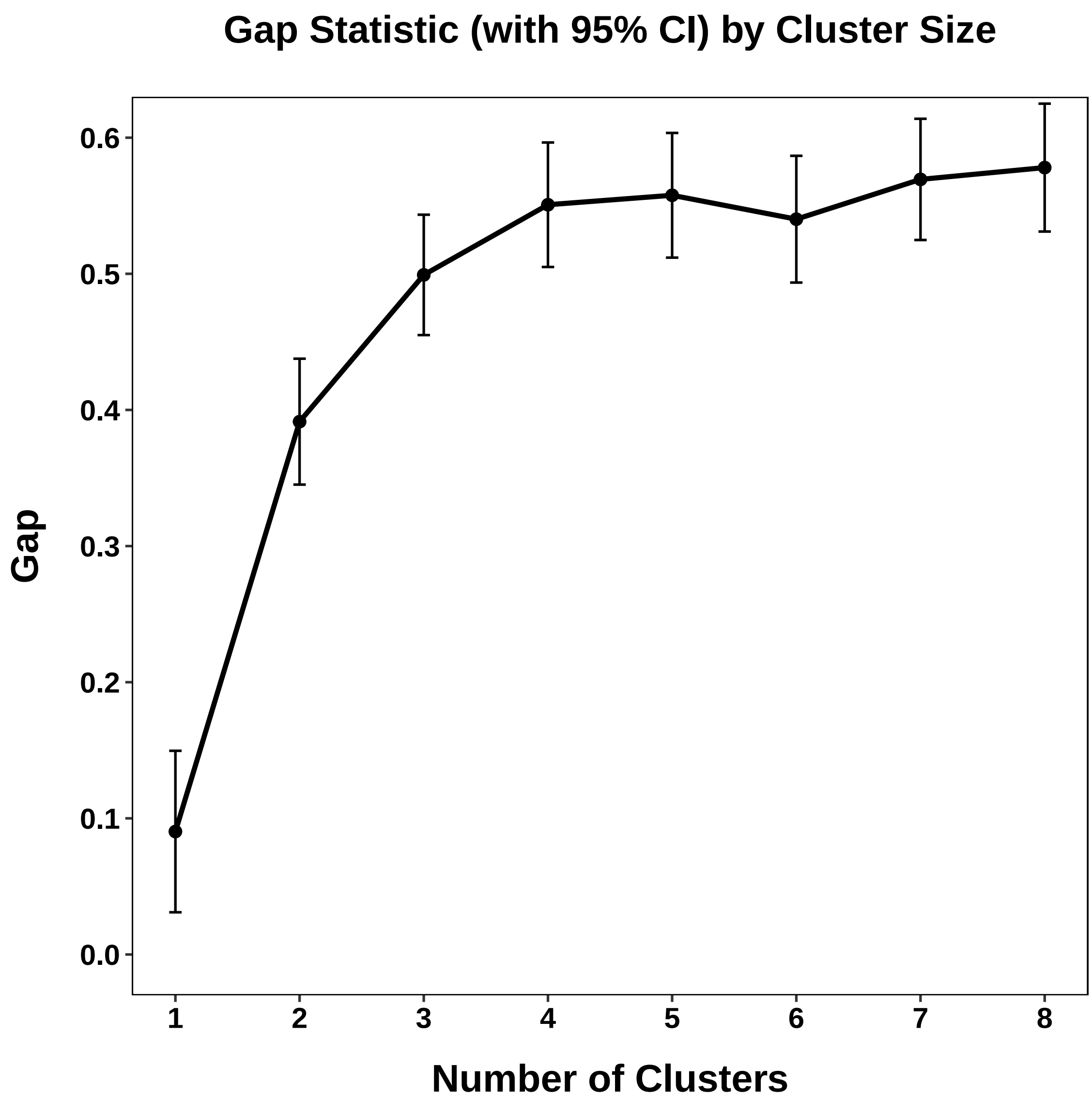
The gap statistic (Tibshirani et al., 2001) compares the within-cluster dispersion to that expected under an appropriate reference null distribution, which assumes random dispersion (e.g., uniform or Gaussian on the range of the original variables or a simplified space [e.g., PC]). The latter is defined by bootstrapping from the null reference distribution. As the obtained WSS curve departs ("gap") from that expected under the reference curve, there is evidence for non-random lumpiness in the data. The gap statistic can be applied to any clustering method and distance measure. The choice for the reference distribution can be important.

For each number of clusters k, it compares log(W(k)) with E[log(W(k))] where the latter is defined via bootstrapping (i.e., simulating from a reference distribution). The technique compares the change in within-cluster dispersion with that expected under an appropriate reference null distribution.

Tibshirani, R., Walther, G. & Hastie, T. (2001). Estimating the number of data clusters via the Gap statistic. Journal of the Royal Statistical Society B, 63, 411-423.

```
Iris_Gap <- clusGap(Iris[, 1:4], FUN = kmeans, nstart = 20, K.max = 10,
    B = 500)
Iris_Gap <- clusGap(Iris[, 1:4], FUN = pam, K.max = 8, B = 500)

print(Iris_Gap, method = "Tibs2001SEmax")

## Clustering Gap statistic ["clusGap"] from call:
## clusGap(x = Iris[, 1:4], FUNcluster = pam, K.max = 8, B = 500)
## B=500 simulated reference sets, k = 1..8; spaceH0="scaledPCA"
##  --> Number of clusters (method 'Tibs2001SEmax', SE.factor=1): 4
##       logW E.logW     gap  SE.sim
## [1,] 6.854  6.945 0.09031 0.03026
## [2,] 6.107  6.498 0.39141 0.02360
## [3,] 5.822  6.321 0.49920 0.02256
## [4,] 5.677  6.228 0.55074 0.02332
## [5,] 5.588  6.146 0.55768 0.02336
## [6,] 5.531  6.072 0.54012 0.02376
## [7,] 5.437  6.007 0.56936 0.02272
## [8,] 5.374  5.952 0.57802 0.02397
```

```
plot_data <- Iris_Gap$Tab[, 3:4]
plot_data <- as.data.frame(plot_data)
names(plot_data) <- c("Gap", "SE")
plot_data$Number <- seq(1, 8)

ggplot(plot_data, aes(x = Number, y = Gap)) + geom_point(shape = 19,
    size = 2, color = "black", na.rm = TRUE) + geom_line(size = 1) +
    geom_errorbar(aes(ymin = plot_data$Gap - 1.96 * plot_data$SE,
        ymax = plot_data$Gap + 1.96 * plot_data$SE), width = 0.1,
        position = position_dodge(0.5)) + scale_x_continuous(breaks = c(seq(1,
    8, 1))) + scale_y_continuous(breaks = c(seq(0, 0.6, 0.1))) + coord_cartesian(xlim = c(1,
    8), ylim = c(0, 0.6)) + xlab("Number of Clusters") + ylab("Gap") +
    theme(text = element_text(size = 14, family = "sans", color = "black",
        face = "bold"), axis.text.y = element_text(colour = "black",
        size = 12, face = "bold"), axis.text.x = element_text(colour = "black",
        size = 12, face = "bold", angle = 0), axis.title.x = element_text(margin = margin(15,
        0, 0, 0), size = 16), axis.title.y = element_text(margin = margin(0,
        15, 0, 0), size = 16), axis.line.x = element_blank(), axis.line.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", margin = margin(0,
            0, 20, 0), hjust = 0.5), panel.background = element_rect(fill = "white",
            linetype = 1, color = "black"), panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(), plot.background = element_rect(fill = "white"),
        plot.margin = unit(c(1, 1, 1, 1), "cm"), legend.position = "bottom",
        legend.title = element_blank()) + ggtitle("Gap Statistic (with 95% CI) by Cluster Size")
```



3.8 Rand Coefficient

The Rand coefficient can be used to compare two clustering methods. The simple Rand coefficient is given by:

$$R = \frac{a+b}{\binom{N}{2}}$$

in which a is the number of times a pair of objects is classified together across the two methods and b is the number of times a pair of objects is classified in different clusters across two methods. The denominator is the number of unique pairs of objects. There is a corrected version of the Rand Coefficient that takes chance agreement into account (in

much the same way that Cohen's kappa is a chance-corrected agreement statistic). That version is reported by the cluster.stats( ) function from the fpc library.

```
Iris_HC_1 <- hclust(d1, "single")
C1 <- cutree(Iris_HC_1, k = 3)
Iris_HC_2 <- hclust(d1, "ward.D2")
C2 <- cutree(Iris_HC_2, k = 3)
Iris_HC_3 <- hclust(d1, "average")
C3 <- cutree(Iris_HC_2, k = 3)
CS_1_2 <- cluster.stats(Iris_Dist, C2, C1, silhouette = TRUE, G2 = FALSE,
    G3 = FALSE, wgap = TRUE, sepindex = TRUE, sepprob = 0.1, sepwithnoise = TRUE,
    compareonly = FALSE, aggregateonly = FALSE)
CS_1_2$corrected.rand

## [1] 0.6069

CS_2_3 <- cluster.stats(Iris_Dist, C2, C3, silhouette = TRUE, G2 = FALSE,
    G3 = FALSE, wgap = TRUE, sepindex = TRUE, sepprob = 0.1, sepwithnoise = TRUE,
    compareonly = FALSE, aggregateonly = FALSE)
CS_2_3$corrected.rand

## [1] 1
```

4 Gower Index

It is not unusual to have measures on interval, ordinal, and nominal scales in the same data set. The Gower distance is a way to combine them all for use in the same cluster analysis. For each variable type, a particular distance metric that works well for that type is used and resulting distances scaled to fall between 0 and 1. A weighted linear combination of these distances for each pair of objects is calculated to create the final distance matrix. The weights are most often chosen to produce a simple average. The Gower distance is always a number between 0 (identical) and 1 (maximally dissimilar). The gower package can be used to get Gower distances using the gower_dist( ) function. The dist.ktab( ) function from the ade4 package can also be used.