Cluster Analysis

Peter Laurinec

March 12, 2018

Contents

What is clustering?

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

Unsupervised learning. It is used when no a priori information about data is available.

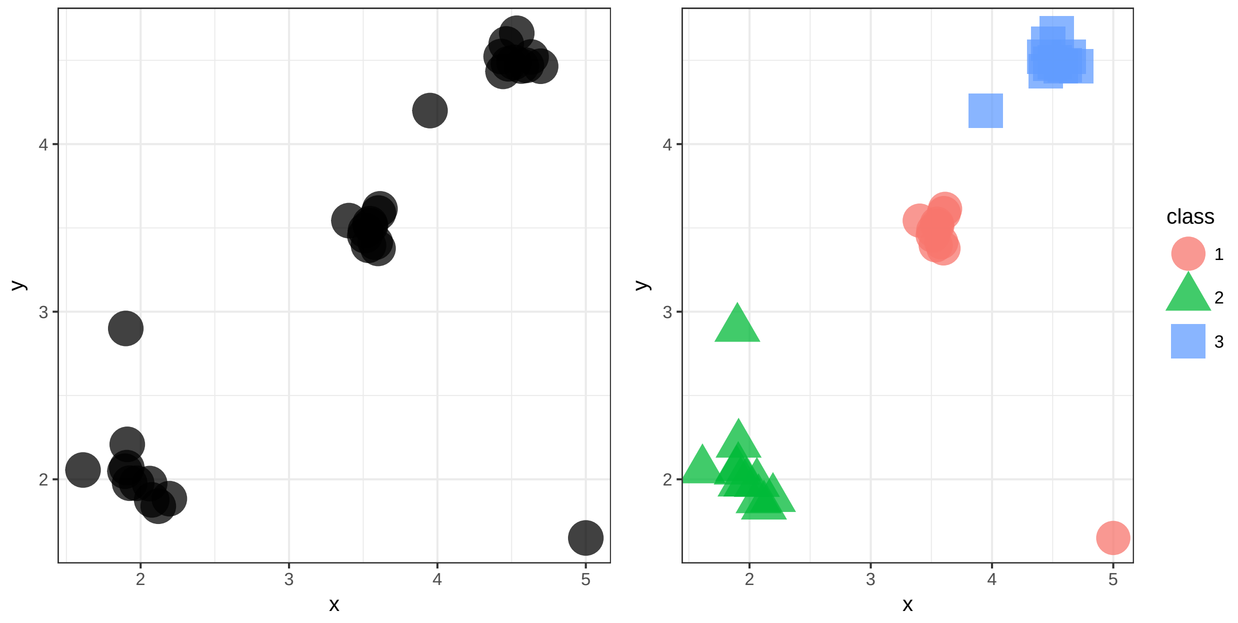
What is it good for?

- To gain insight into data, generate hypotheses, detect anomalies, and identify salient features,
- To identify the degree of similarity among objects (i.e. organisms),
- As a method for organising the data and summarising it through cluster prototypes (compression).

Classification to groups

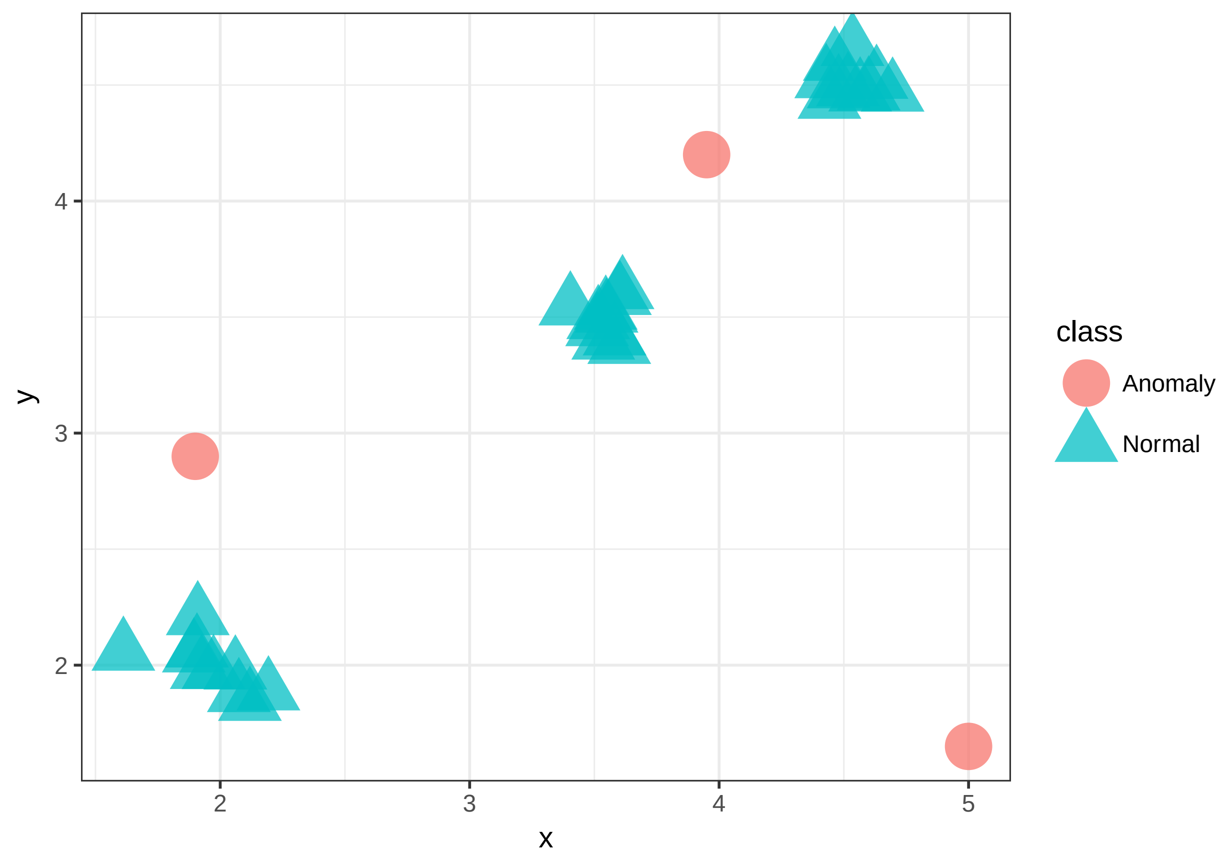
```r
library(data.table)
library(ggplot2)
library(gridExtra)
library(grid)
library(cluster)

datas <- data.table(x = c(rnorm(10, 3.5, 0.1), rnorm(10, 2, 0.1),
                          rnorm(10, 4.5, 0.1), c(5, 1.9, 3.95)),
                    y = c(rnorm(10, 3.5, 0.1), rnorm(10, 2, 0.1),
                          rnorm(10, 4.5, 0.1), c(1.65, 2.9, 4.2)))

gg1 <- ggplot(datas, aes(x, y)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw()

kmed_res <- pam(datas, 3)$clustering

datas[, class := as.factor(kmed_res)]

gg2 <- ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw()
```

```r
define_region <- function(row, col){
  viewport(layout.pos.row = row, layout.pos.col = col)
}

grid.newpage()
# Create layout : nrow = 2, ncol = 2
pushViewport(viewport(layout = grid.layout(1, 2)))
# Arrange the plots
print(gg1, vp = define_region(1, 1))
print(gg2, vp = define_region(1, 2))
```



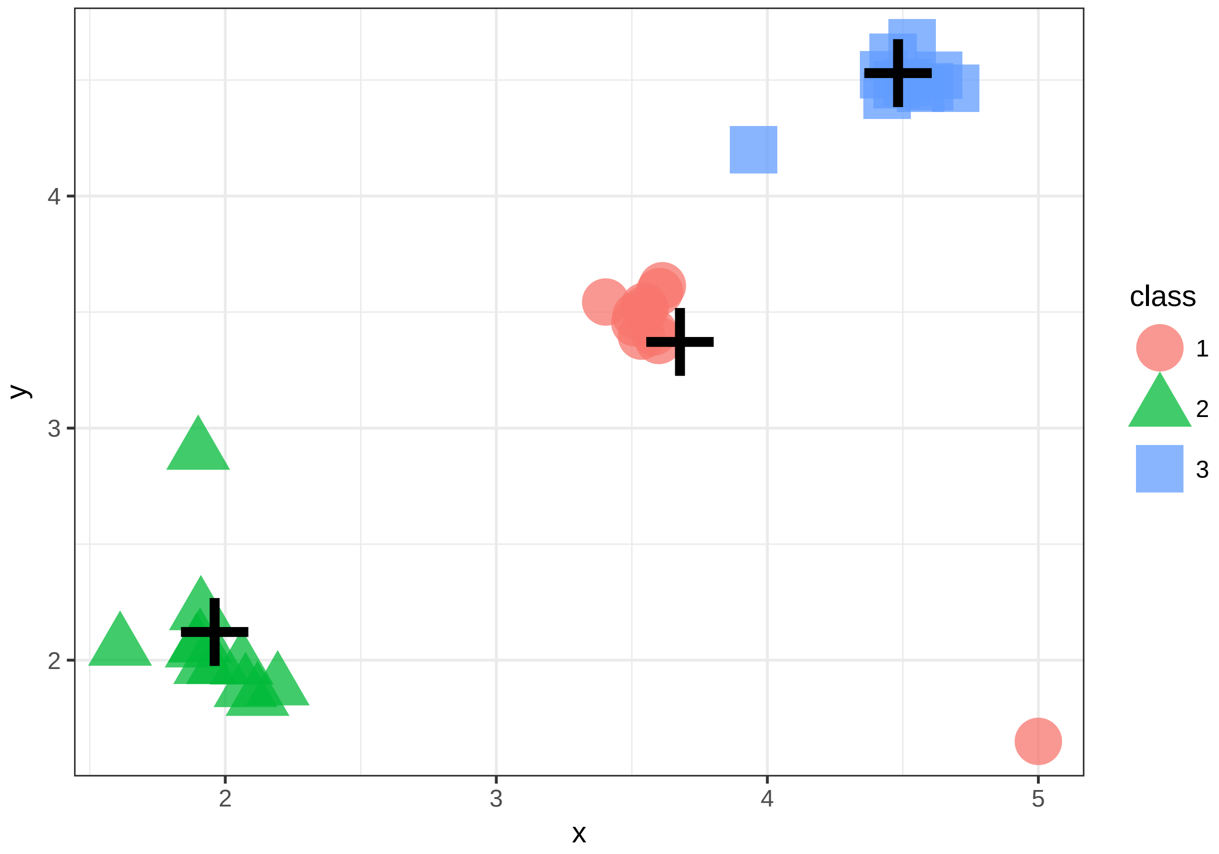
Anomaly detection

```r
anom <- c(rep(1, 30), rep(0, 3))
datas[, class := as.factor(anom)]
levels(datas$class) <- c("Anomaly", "Normal")

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw()
```



Data compression

```
datas[, class := as.factor(kmed_res)]

centroids <- datas[, .(x = mean(x), y = mean(y)), by = class]

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  geom_point(data = centroids, aes(x, y), color = "black", shape = "+", size = 18) +
  theme_bw()
```



Types of clustering methods

Non-hierarchical:

- Centroid-based
- Model-based
- Density-based
- Grid-based

Hierarchical:

- Agglomerative
- Divisive

Centroid-based

- K-means
- K-medians
- K-medoids

It creates prototypes: centroids or medoids.

K-means

Steps:

- Create randomly K clusters (centroids).
- Assign points to nearest centroids.
- Update centroids.
- Go to step 2 while centroids are changing.

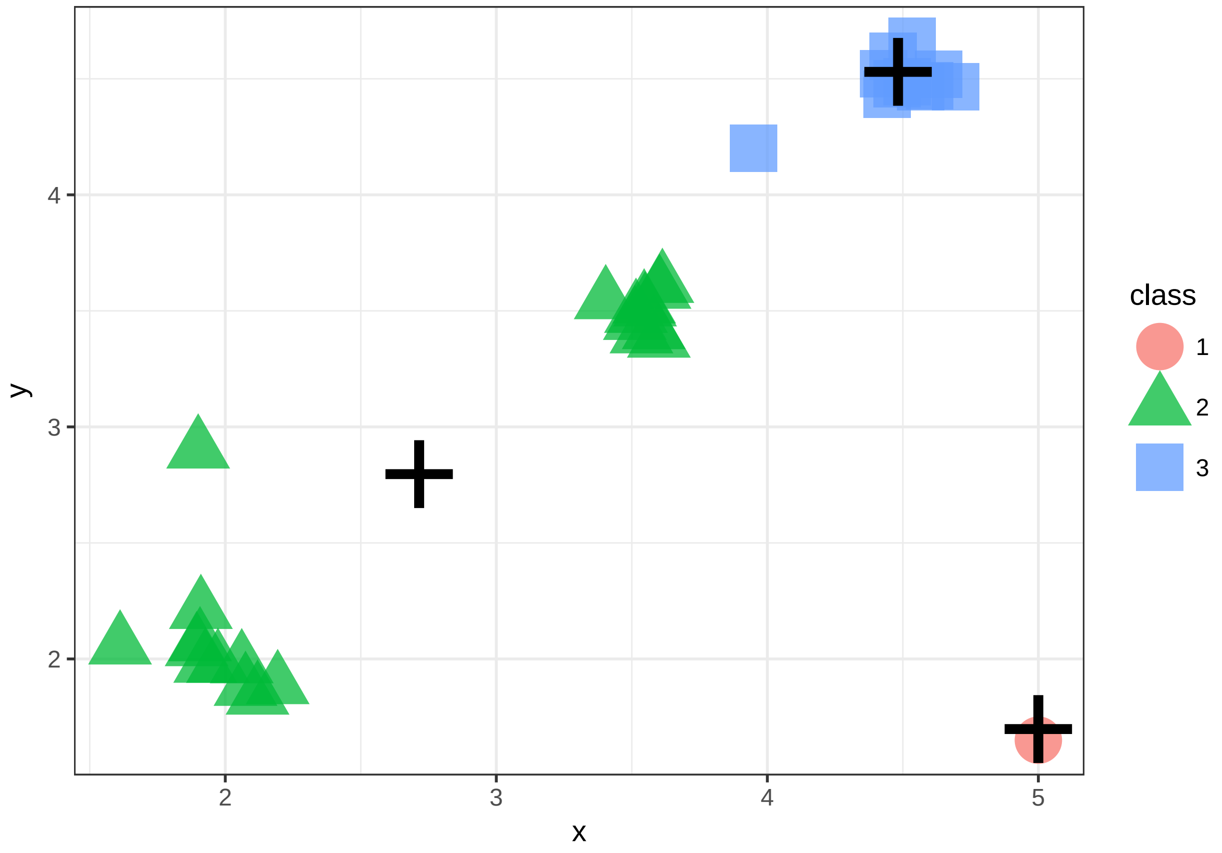
Pros and cons:

- [+] Fast to compute. Easy to understand.
- [-] Various initial clusters can lead to different final clustering.
- [-] Scale-dependent.
- [-] Creates only convex (spherical) shapes of clusters.
- [-] Sensitive to outliers.

K-means solved by Genetic Algorithm

K-means - example

```r
km_res <- kmeans(datas, 3)$cluster

datas[, class := as.factor(km_res)]

centroids <- datas[, .(x = mean(x), y = mean(y)), by = class]

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  geom_point(data = centroids, aes(x, y), color = "black", shape = "+", size = 18) +
  theme_bw()
```



K-medoids

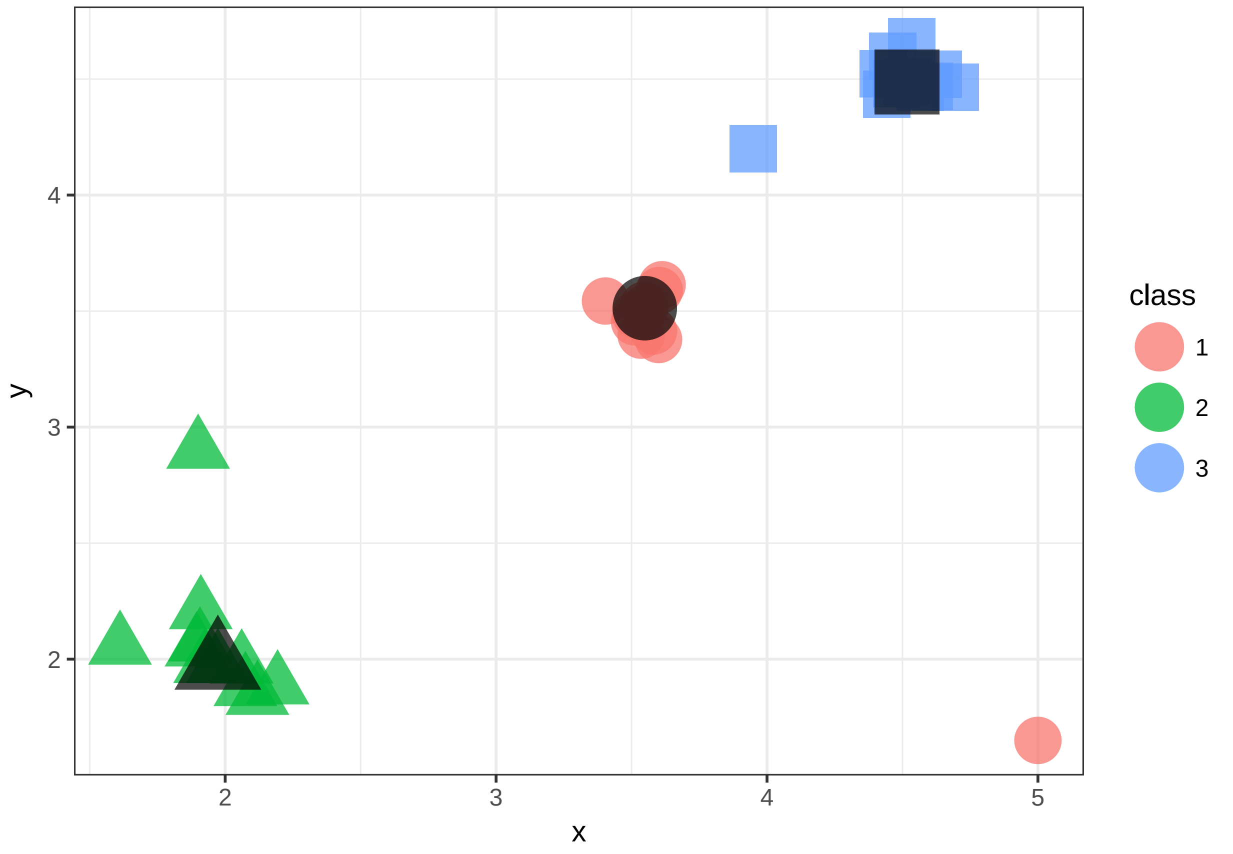
Prototypes are medoids - members of the dataset.

Pros and cons:

- [+] Easy to understand.
- [+] Less sensitive to outliers.
- [+] Possibility to use any distance measure.
- [-] Various initial clusters can lead to different final clustering.
- [-] Scale-dependent.
- [-] Slower than K-means.

K-medoids - example

```r
kmed_res <- pam(datas[, .(x, y)], 3)
datas[, class := as.factor(kmed_res$clustering)]

centroids <- data.table(kmed_res$medoids, class = as.factor(1:3))

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  geom_point(data = centroids, aes(x, y, shape = class), color = "black", size = 11, alpha = 0.7) +
  theme_bw() +
  guides(shape=FALSE)
```



The determination of number of clusters

Internal validation. Try many K.

Many indexes are there…

- Silhouette
- Davies-Bouldin index
- Dunn index
- etc.

Every index has similar characteristic:

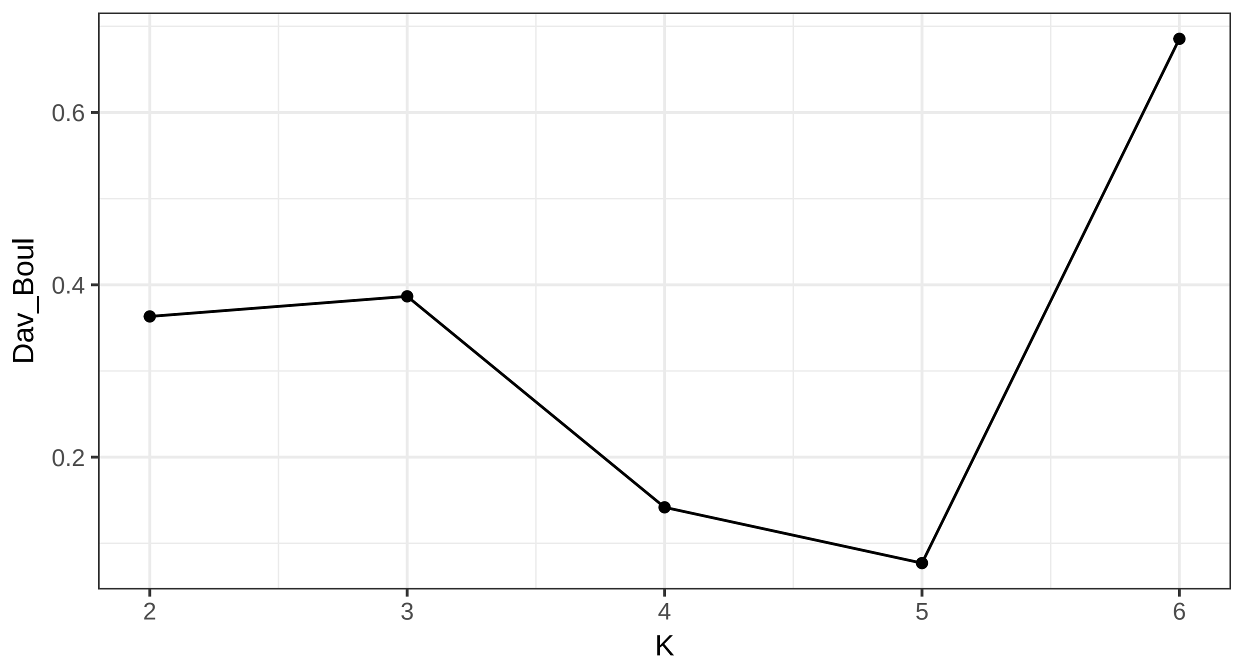
$$\frac{within - cluster - similarity}{between - clusters - similarity}.$$

Elbow diagram

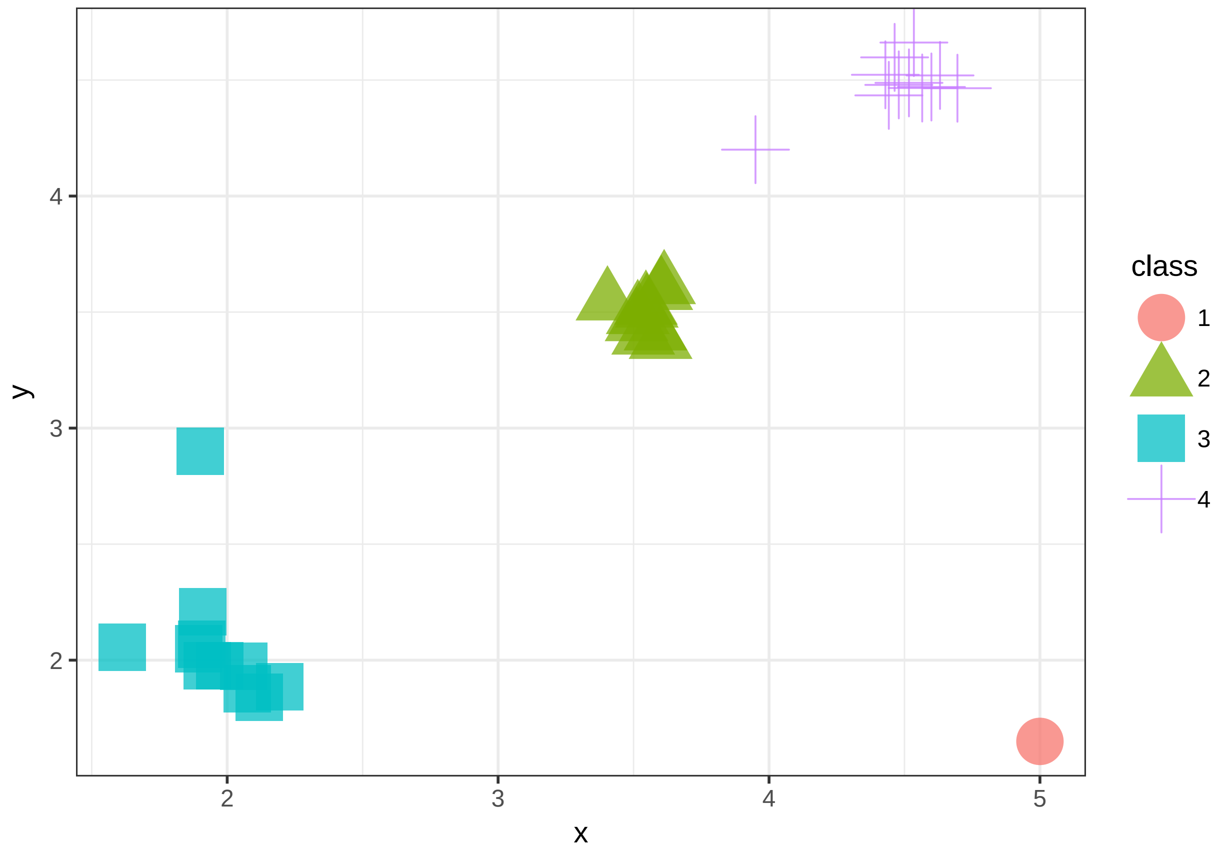
```r
library(clusterCrit)

km_res_k <- lapply(2:6, function(i) kmeans(datas[, .(x, y)], i)$cluster)
km_res_k
```

```
## [[1]]
##  [1] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 2
## 
## [[2]]
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 1 2 3
## 
## [[3]]
##  [1] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 1 3 4
## 
## [[4]]
##  [1] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 5 1 4
## 
## [[5]]
##  [1] 1 1 1 1 1 1 1 1 1 1 2 4 4 4 2 2 6 6 4 6 1 1 1 1 1 1 1 1 1 1 5 3 1
```

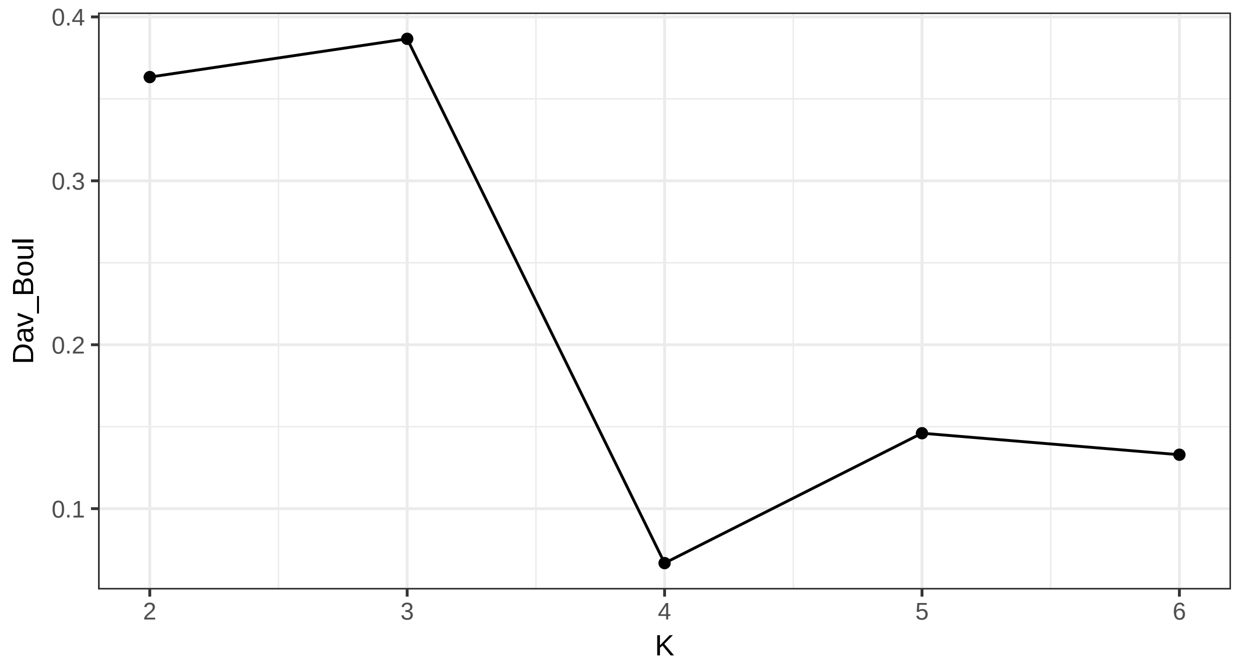
```r
db_km <- lapply(km_res_k, function(j) intCriteria(data.matrix(datas[, .(x, y)]),
                                                  j,
                                                  "Davies_bouldin")$davies_bouldin)

ggplot(data.table(K = 2:6, Dav_Boul = unlist(db_km)), aes(K, Dav_Boul)) +
  geom_line() +
  geom_point() +
  theme_bw()
```



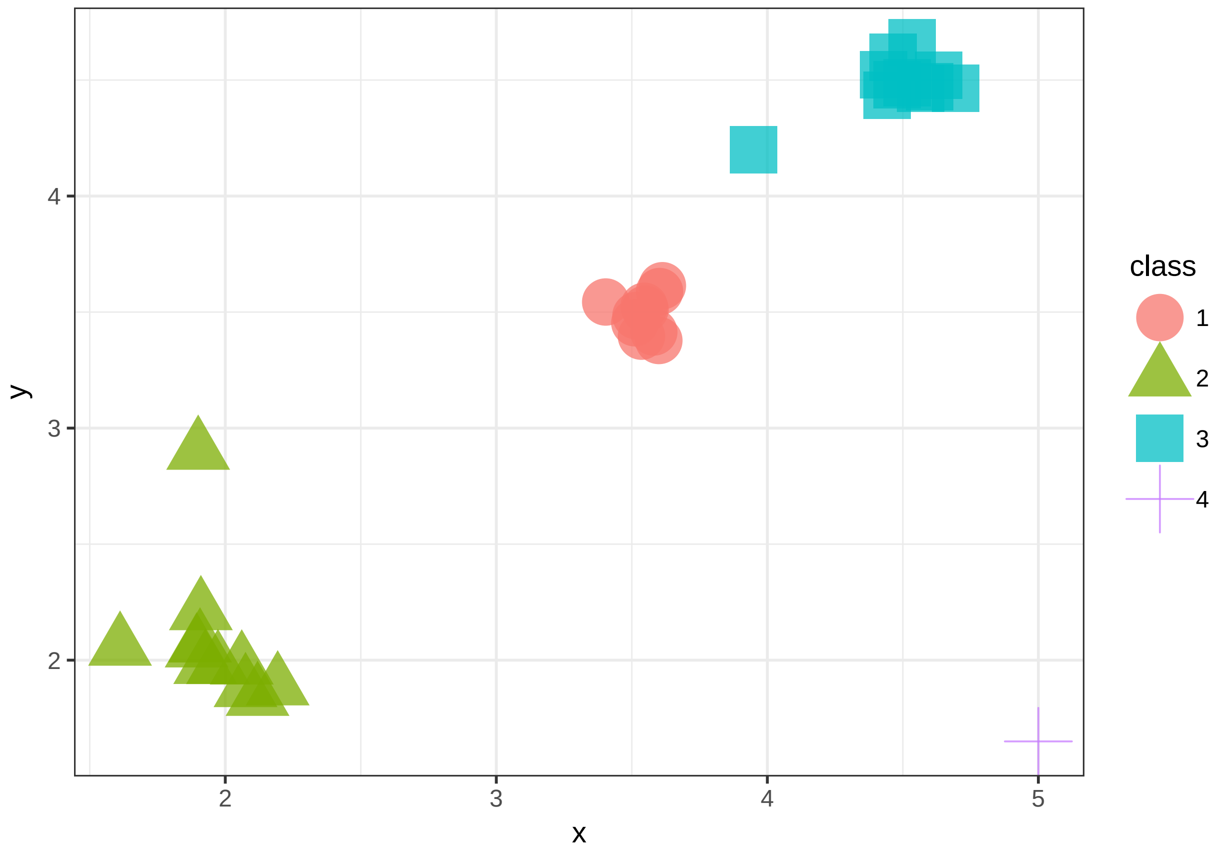
```
datas[, class := as.factor(km_res_k[[which.min(diff(unlist(db_km)))+1]])]

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw()
```



```
kmed_res_k <- lapply(2:6, function(i) pam(datas[, .(x, y)], i)$clustering)

db_kmed <- lapply(kmed_res_k, function(j) intCriteria(data.matrix(datas[, .(x, y)]),
                                                   j,
                                                   "Davies_bouldin")$davies_bouldin)

ggplot(data.table(K = 2:6, Dav_Boul = unlist(db_kmed)), aes(K, Dav_Boul)) +
  geom_line() +
  geom_point() +
  theme_bw()
```



```
datas[, class := as.factor(kmed_res_k[[which.min(diff(unlist(db_km)))+1]])]

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw()
```



Model-based

Clustering methods based on some probabilistic distribution.

- Gaussian normal distribution
- Poisson distribution
- etc.

Multivariate data -> Multivariate distributions -> Mixture of models -> **Gaussian Mixture Models (GMM)**.

GMM

Target is to maximise likelihood:

$$L(\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_1}, \ldots, \boldsymbol{\Sigma_k} | \boldsymbol{x_1}, \ldots, \boldsymbol{x_n}).$$

It is typically solved by **EM** algorithm (Expectation Maximization).

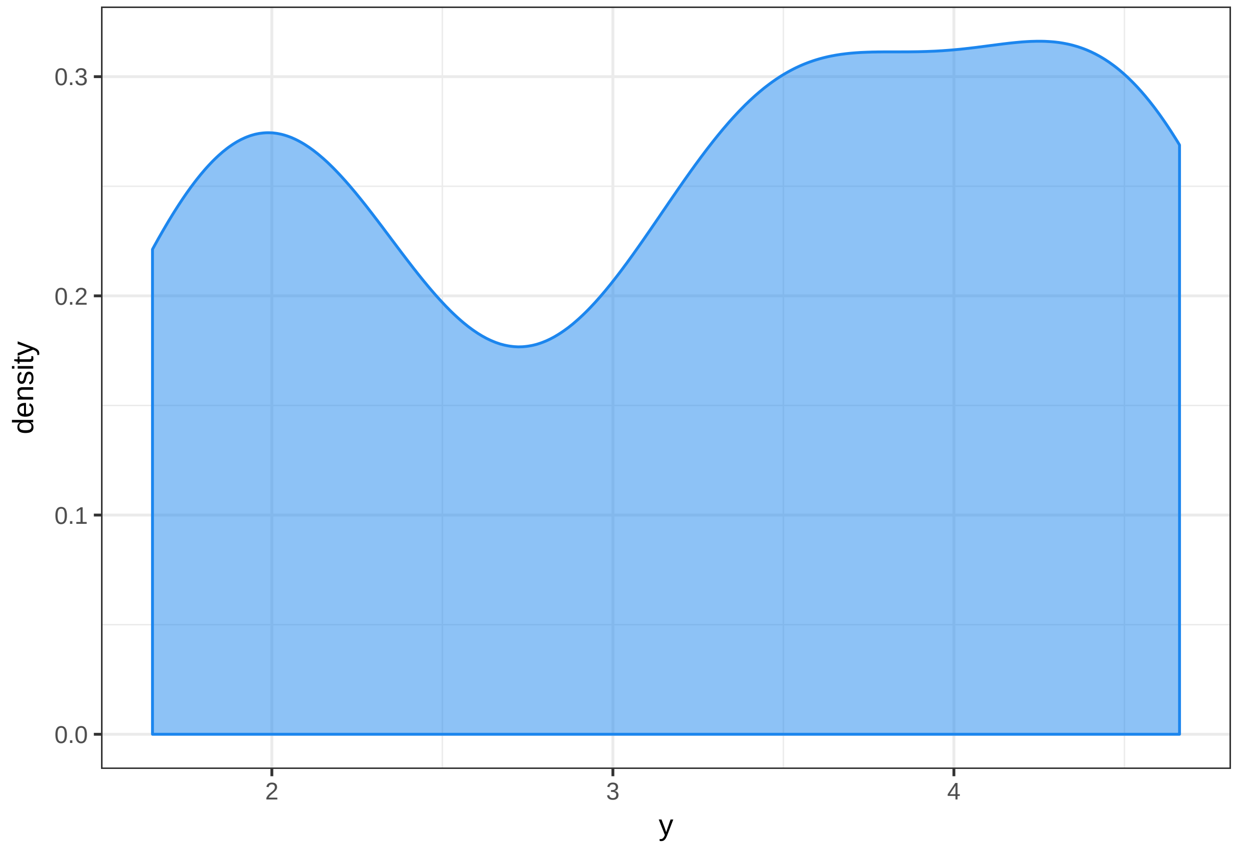
Cluster is represented by mean and covariance matrix.

Pros and cons:

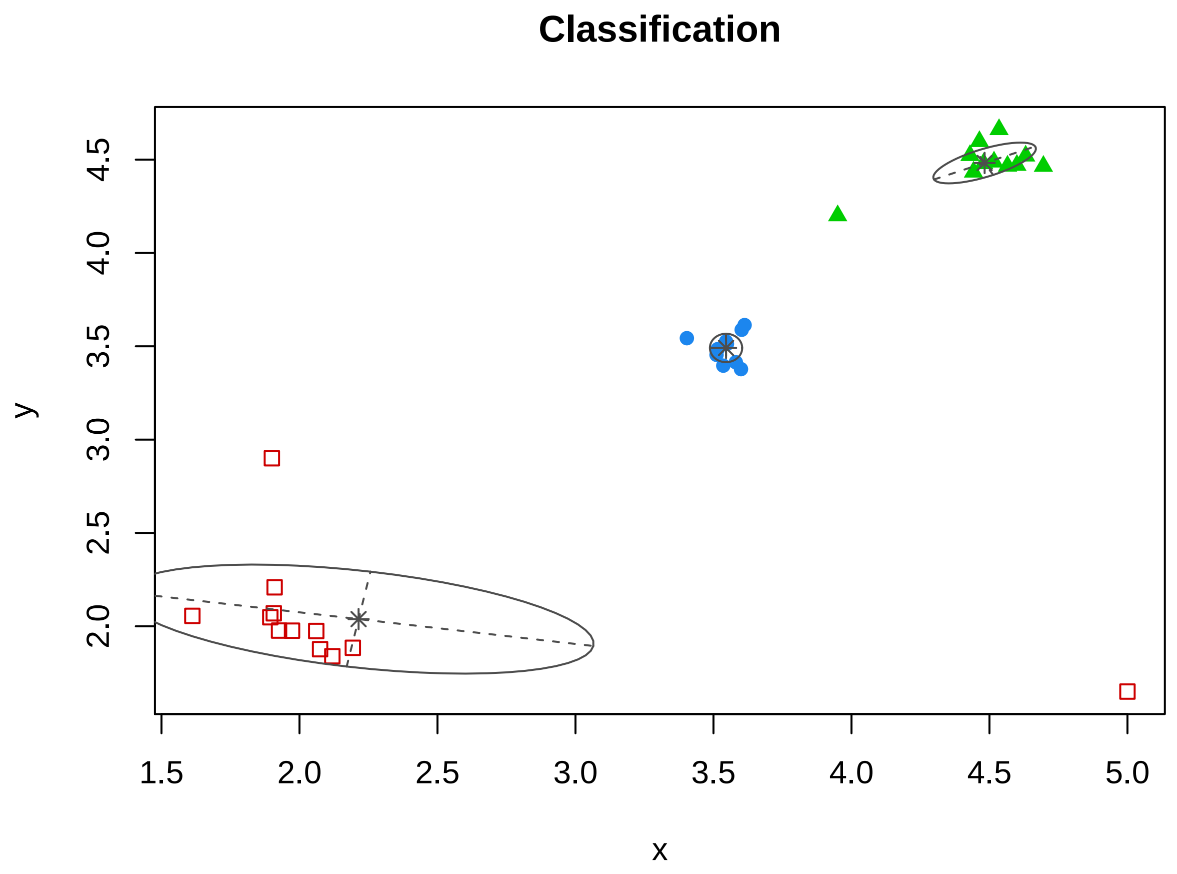
- [+] Ellipsoidal clusters,
- [+] Can be parameterised by covariance matrix,
- [+] Scale-independent,
- [-] Very slow for high-dimensional data,
- [-] Can be difficult to understand.

GMM - density

```
ggplot(datas, aes(y)) +
  geom_density(color = "dodgerblue2", fill = "dodgerblue2", alpha = 0.5) +
  theme_bw()
```



```
library(mclust)
res <- Mclust(datas[, .(x, y)], G = 3, modelNames = "VVV")

plot(res, what = "classification")
```



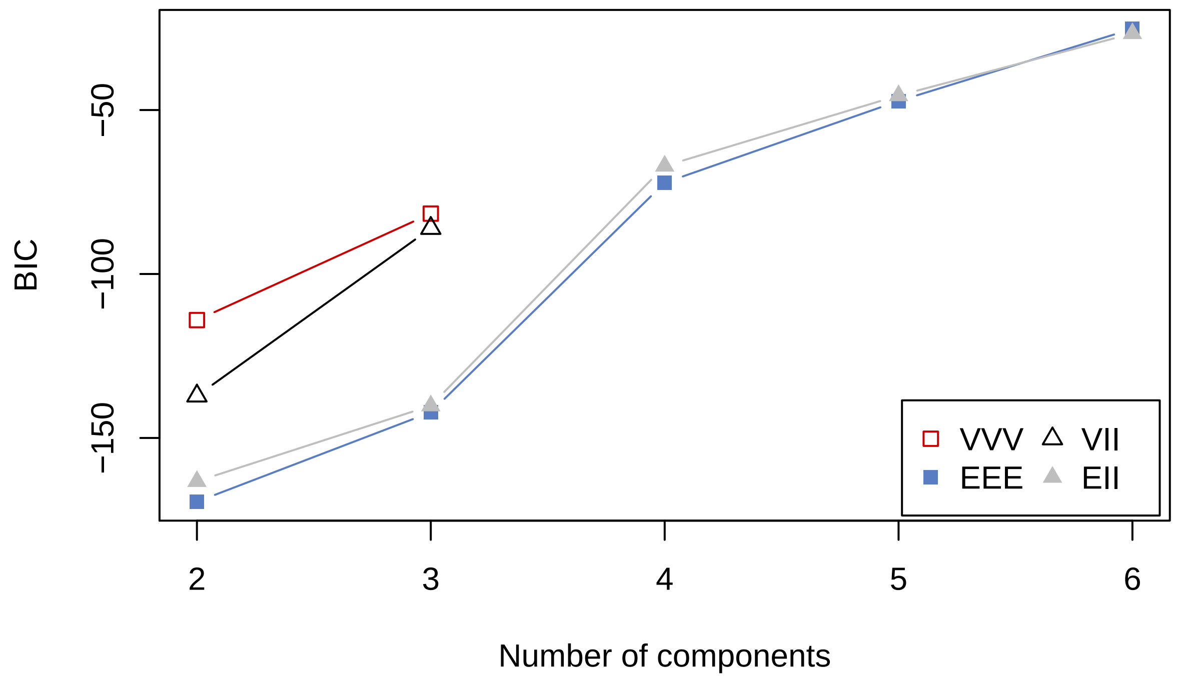
BIC

Bayesian Information Criterion (BIC) for optimal number of clusters choosing. We can try also vary dependency of covariance matrix $\boldsymbol{\Sigma}$.

```r
res <- Mclust(datas[, .(x, y)], G = 2:6, modelNames = c("VVV", "EEE", "VII", "EII"))
res
```

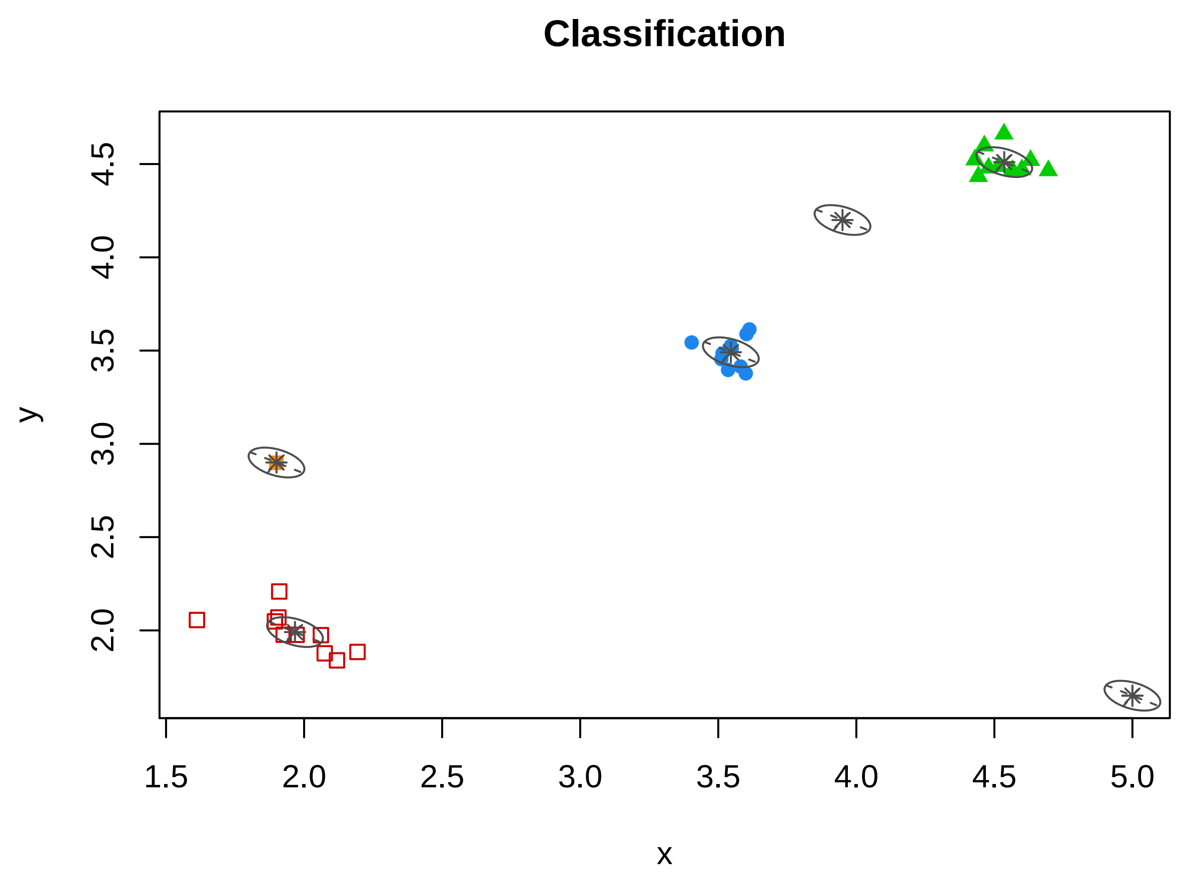
```
## 'Mclust' model object:
##  best model: ellipsoidal, equal volume, shape and orientation (EEE) with 6 components
```

```r
plot(res, what = "BIC")
```



The result:

```
plot(res, what = "classification")
```



Density-based

Neighborhoods (ϵ distance), minimal points in cluster.

Methods:

- **DBSCAN**
- **OPTICS**
- Multiple densities (Multi-density)

DBSCAN

Pros and cons:

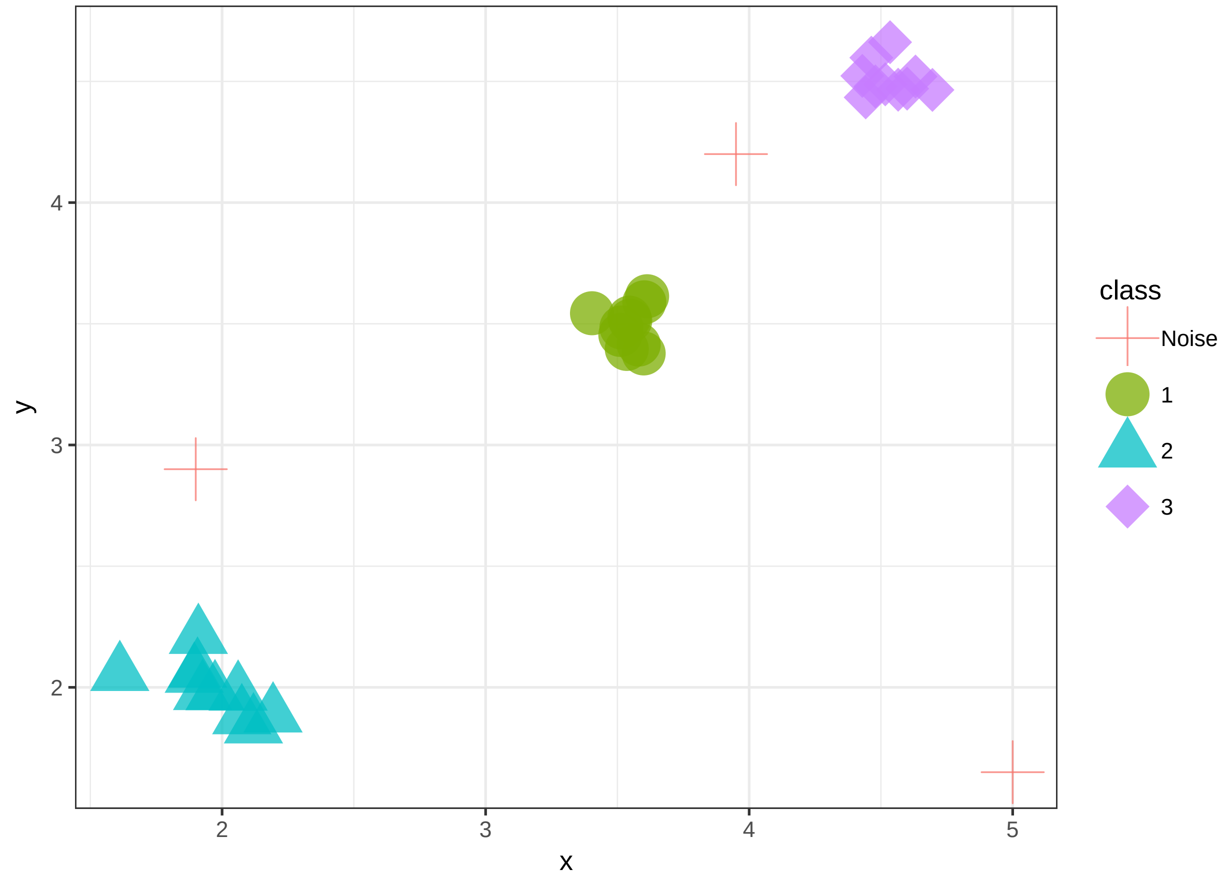
- [+] Extracts automatically outliers,
- [+] Fast to compute,
- [+] Can find clusters of arbitrary shapes,
- [+] Number of clusters is determined auto. based on data,
- [-] Parameters (ϵ, minPts) must be set by a practitioner,
- [-] Possible problem with neighborhoods - can be connected.

```
library(dbscan)
```

```r
res <- dbscan(datas[, .(x, y)], eps = 0.5, minPts = 5)
table(res$cluster)
```

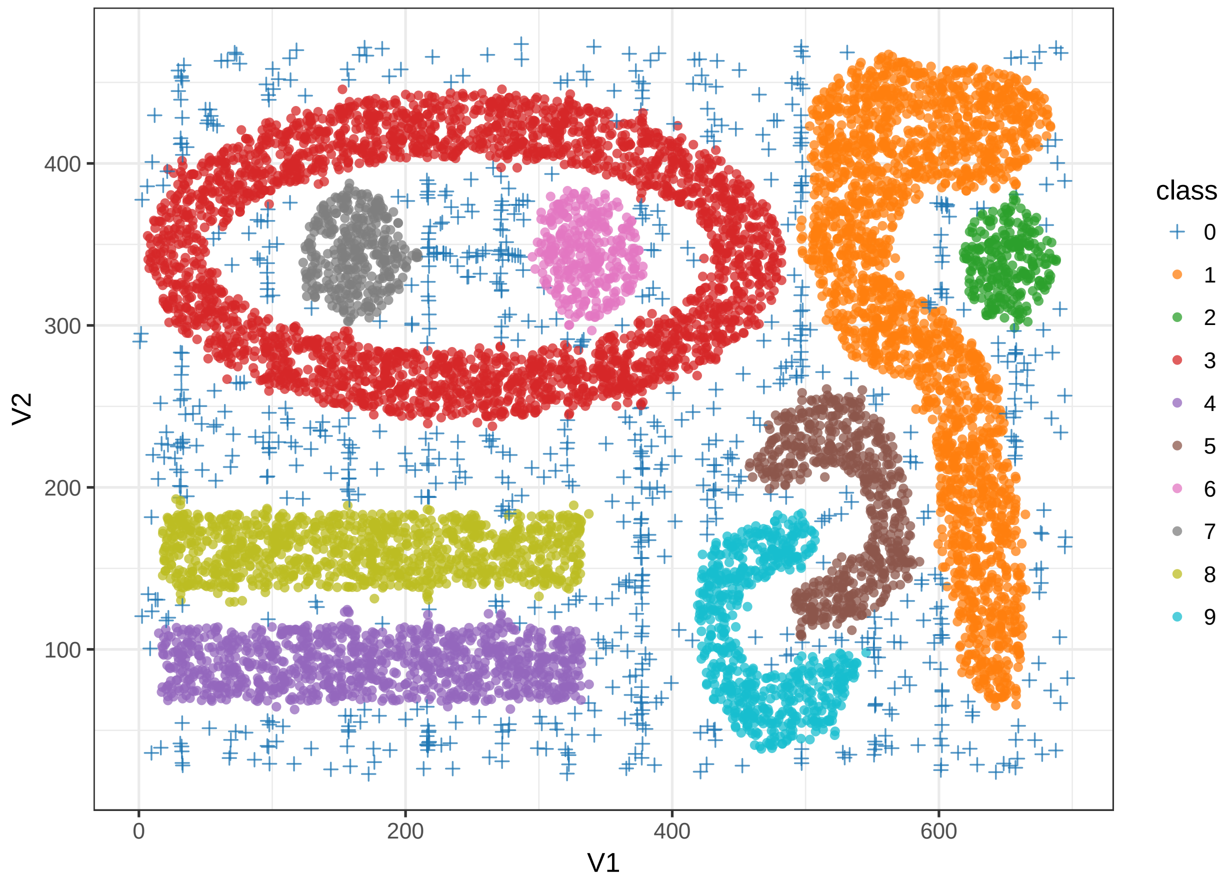
```
##
##  0  1  2  3
##  3 10 10 10
```

```r
datas[, class := as.factor(res$cluster)]
levels(datas$class)[1] <- c("Noise")

ggplot(datas, aes(x, y, color = class, shape = class)) +
  geom_point(alpha = 0.75, size = 8) +
  theme_bw() +
  scale_shape_manual(values = c(3,16,17,18))
```



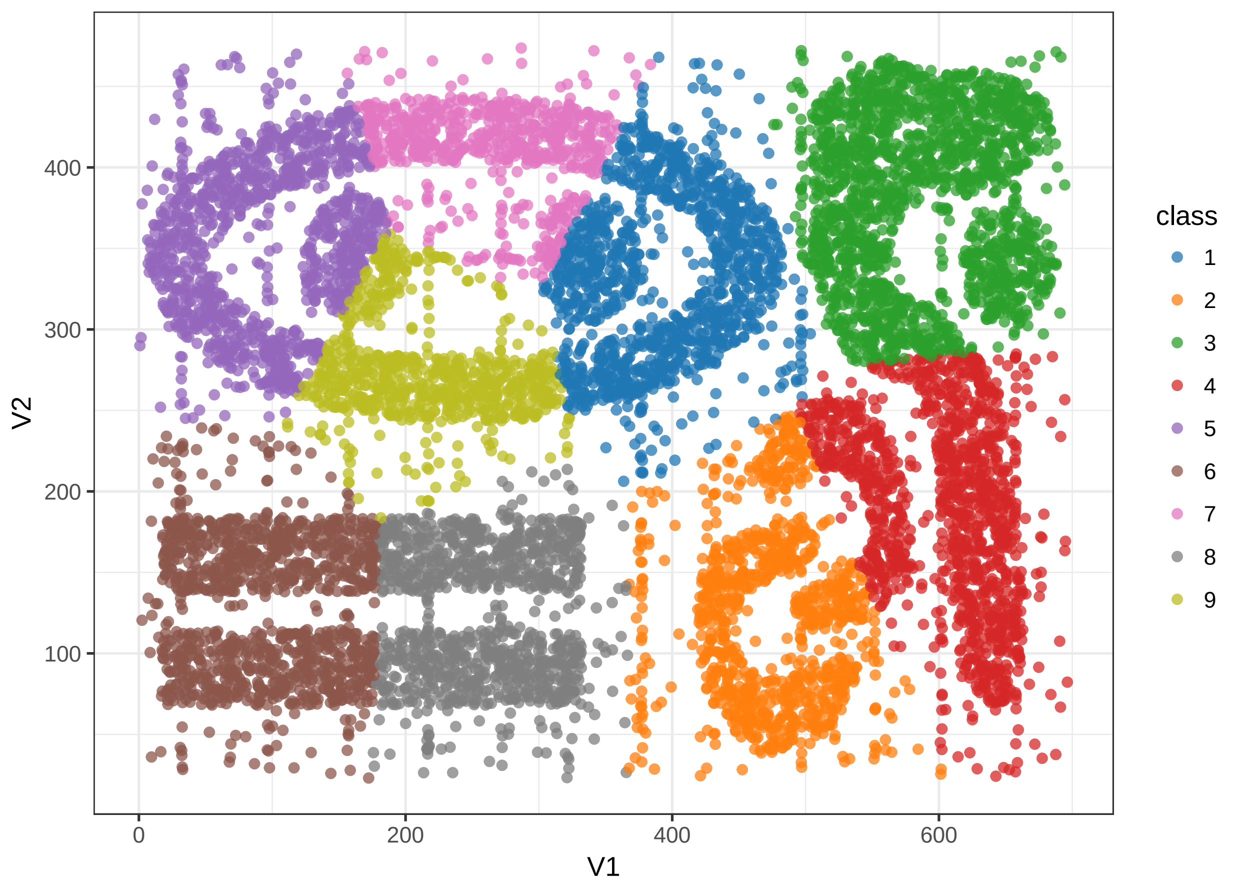
Bananas - DBSCAN result

```r
bananas <- fread("t7.10k.dat")
db_res <- dbscan(bananas, eps = 10, minPts = 15)
# table(db_res$cluster)

data_all <- data.table(bananas, class = as.factor(db_res$cluster))

library(ggsci)
ggplot(data_all, aes(V1, V2, color = class, shape = class)) +
  geom_point(alpha = 0.75) +
  scale_color_d3() +
  scale_shape_manual(values = c(3, rep(16, 9))) +
  theme_bw()
```



Bananas - K-means result

```
km_res <- kmeans(bananas, 9)
data_all[, class := as.factor(km_res$cluster)]

ggplot(data_all, aes(V1, V2, color = class)) +
  geom_point(alpha = 0.75) +
  scale_color_d3() +
  theme_bw()
```



Spectral clustering

Method based on spectral decomposition of data - creation of eigen vectors and eigen values.

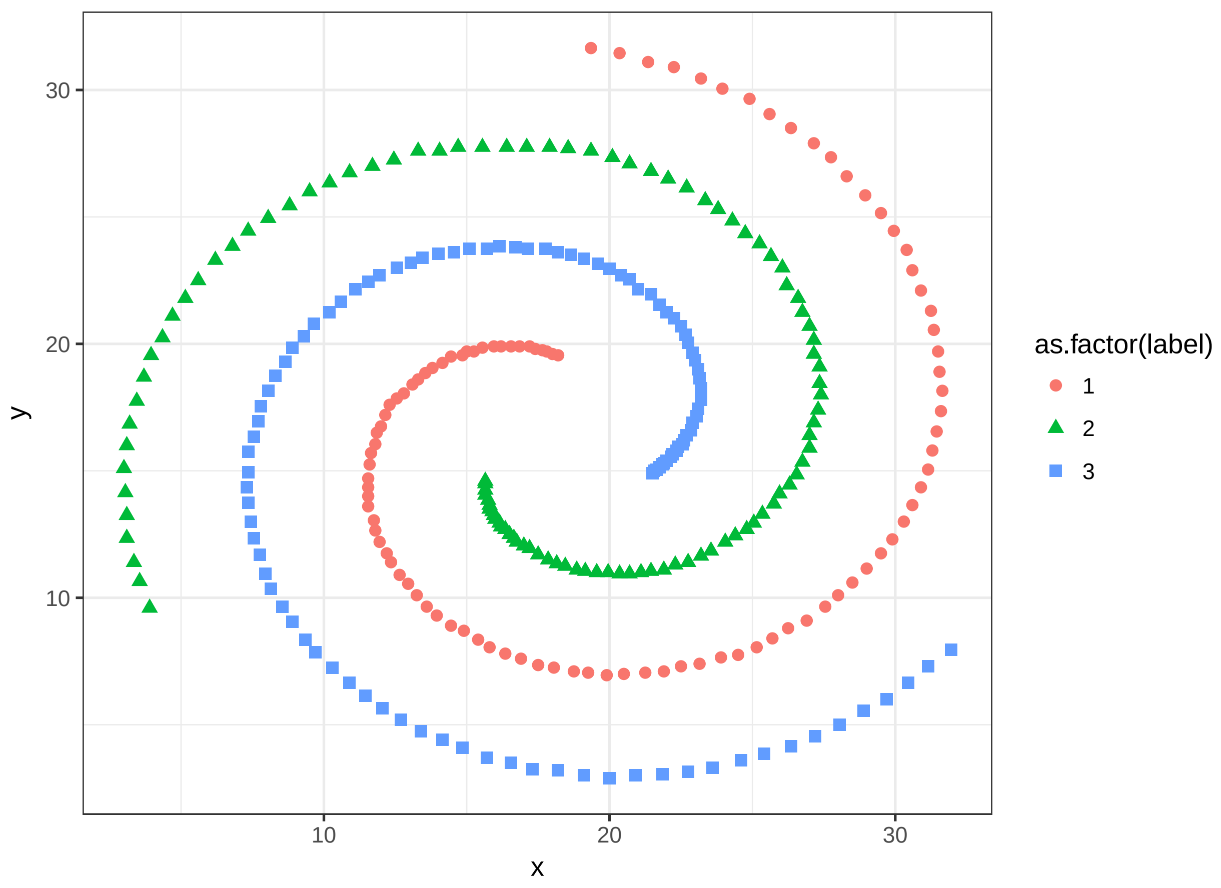
Steps:

1. N = number of data, d = dimension of data,
2. **A** = affinity matrix, $A_{ij} = \exp(-(data_i - data_j)^2/(2 * \sigma^2))$ - N by N matrix,
3. **D** = diagonal matrix whose (i,i)-element is the sum of **A** i-th row - N by N matrix,
4. $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ - N by N matrix,
5. **X** = union of k largest eigenvectors of **L** - N by k matrix,
6. Renormalising each of **X** rows to have unit length - N by k matrix,
7. Run K-means algorithm on **X**.

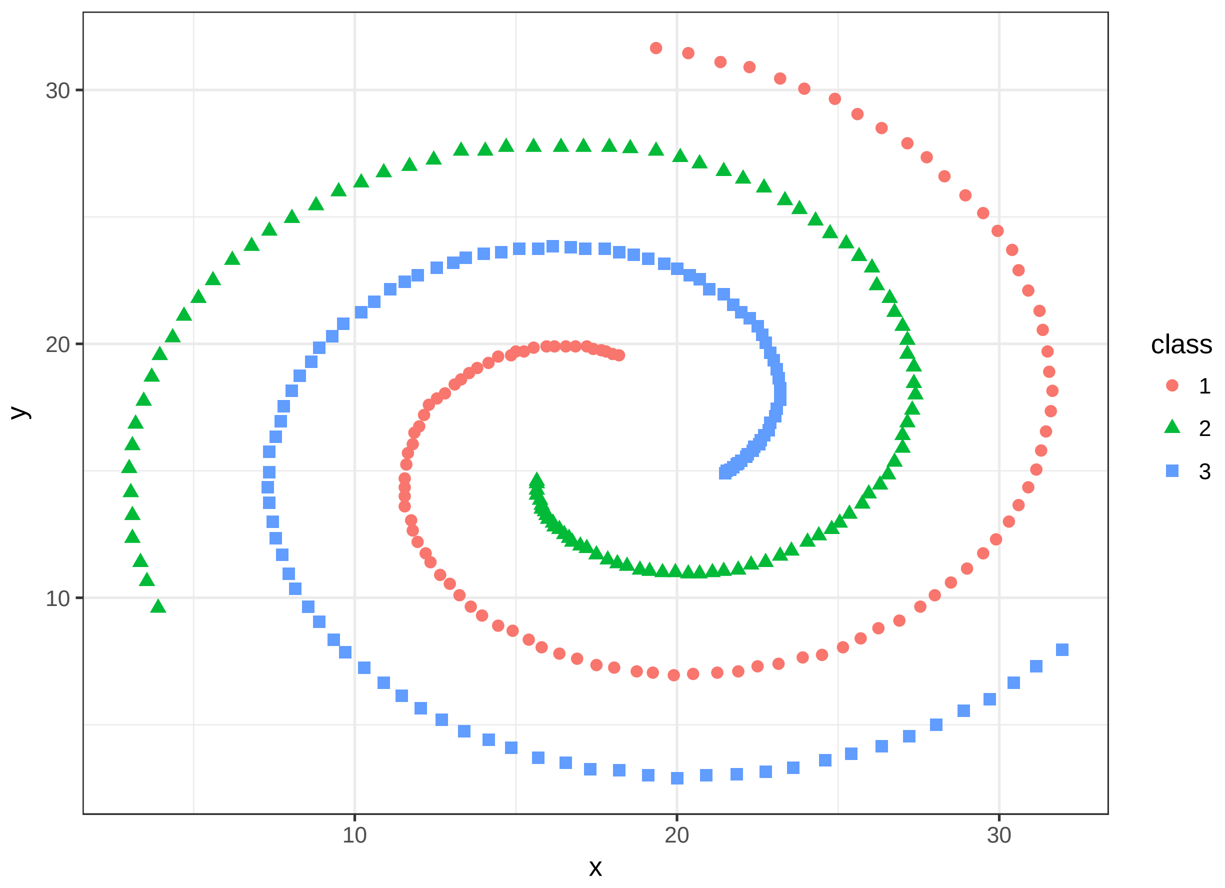
Typical use case for spectral clustering

Spirals.

```
data_spiral <- fread("data_spiral.csv")

ggplot(data_spiral, aes(x, y, color = as.factor(label), shape = as.factor(label))) +
  geom_point(size = 2) +
  theme_bw()
```

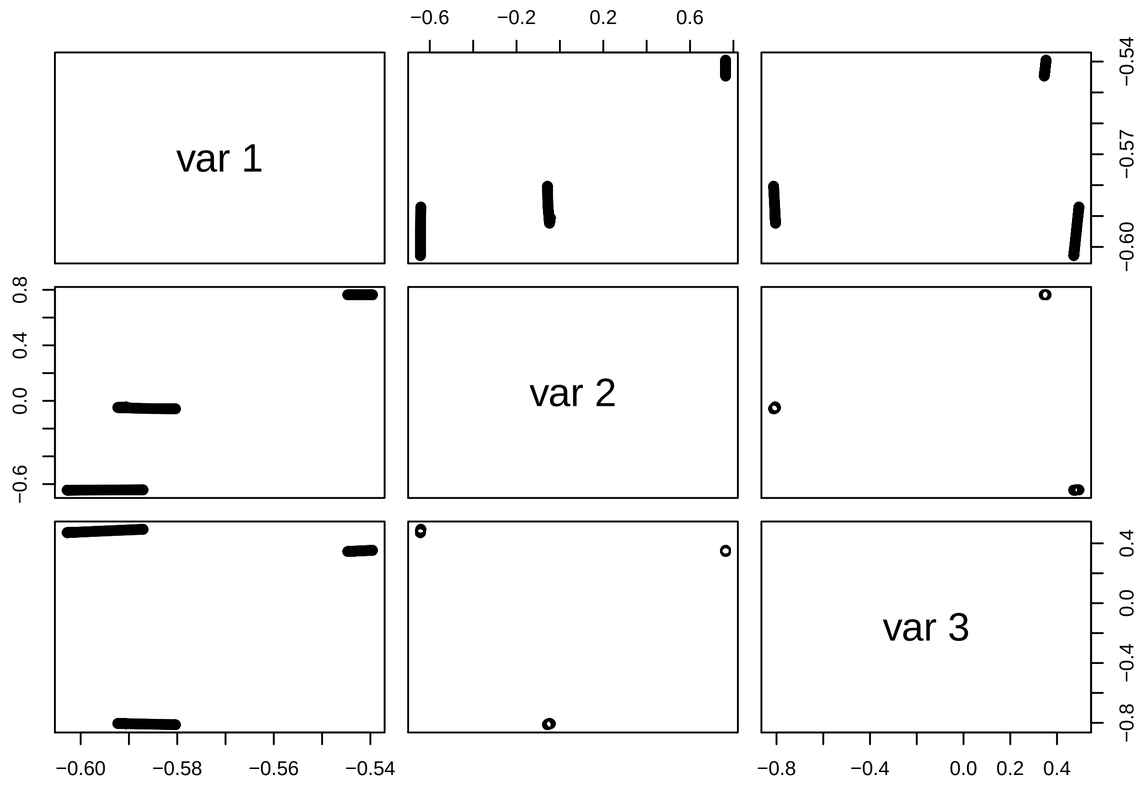


```r
library(kernlab)

res <- specc(data.matrix(data_spiral[, .(x, y)]), centers = 3)

data_spiral[, class := as.factor(res)]

ggplot(data_spiral, aes(x, y, color = class, shape = class)) +
  geom_point(size = 2) +
  theme_bw()
```



Why is it like that?

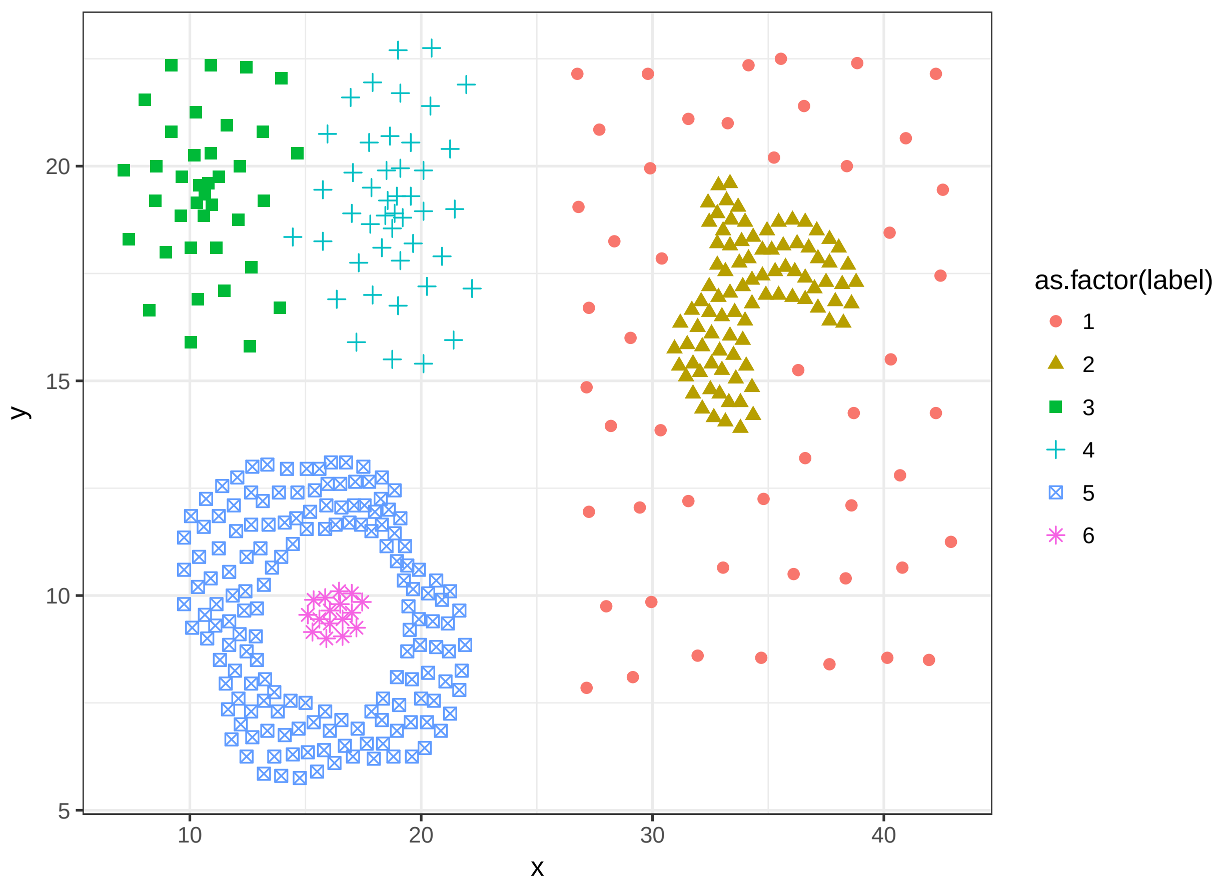
```r
library(expm)

# define kernel
rbf <- rbfdot(sigma = 0.6) # very important hyperparameter!
# calculate kernel matrix
aff_mat <- kernelMatrix(rbf, data.matrix(data_spiral))
# diagonal matrix
diag_mat <- diag(rowSums(aff_mat))
# ^(-1/2)
diag_mat_sqrt <- solve(sqrtm(diag_mat))
# L matrix
L_mat <- (diag_mat_sqrt %*% aff_mat %*% diag_mat_sqrt)
# calculate eigenvectors
X_eigen <- eigen(L_mat)$vectors[,1:3]
# normalising X
X_norm <- t(sapply(1:nrow(X_eigen), function(i) X_eigen[i,]/sqrt(sum(X_eigen[i,]^2))))

pairs(X_norm)
```



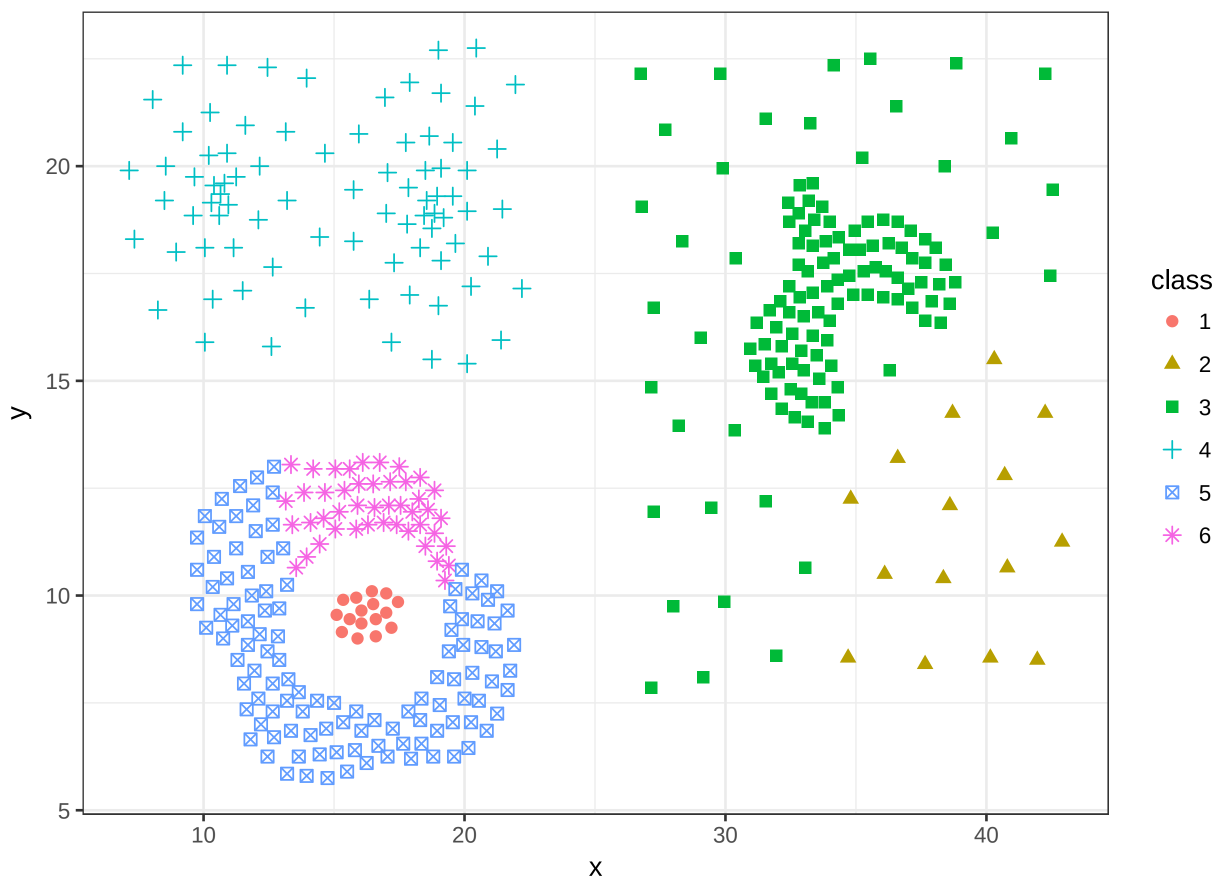
```r
# km_res_spec <- kmeans(X_norm, 3)$cluster
#
# data_spiral[, class := as.factor(km_res_spec)]
#
# ggplot(data_spiral, aes(x, y, color = class, shape = class)) +
#   geom_point(size = 2) +
#   theme_bw()
```

More advanced data.

```r
data_compound <- fread("data_compound.csv")

ggplot(data_compound, aes(x, y, color = as.factor(label), shape = as.factor(label))) +
  geom_point(size = 2) +
  theme_bw()
```

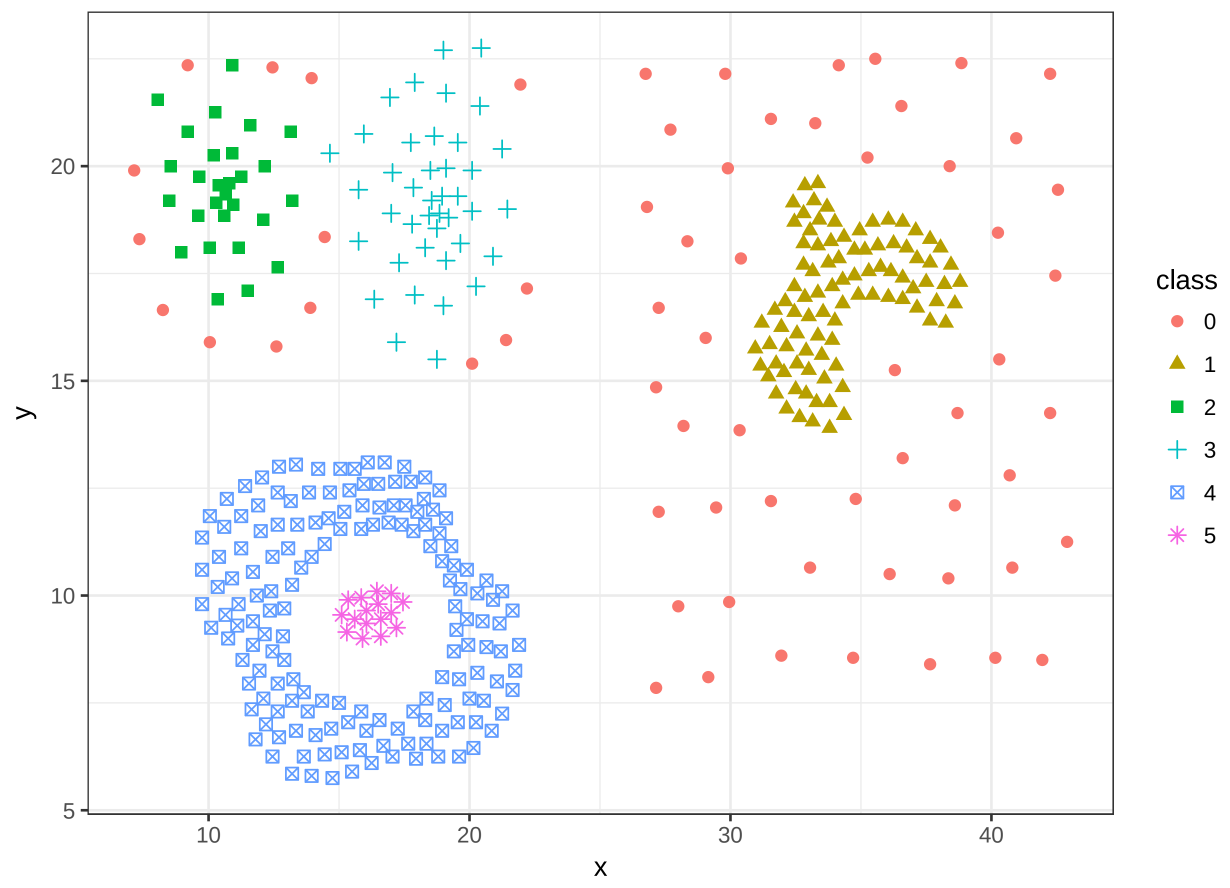


```r
res <- specc(data.matrix(data_compound[, .(x, y)]), centers = 6)

data_compound[, class := as.factor(res)]

ggplot(data_compound, aes(x, y, color = class, shape = class)) +
  geom_point(size = 2) +
  theme_bw()
```



Let's try DBSCAN.

```
db_res <- dbscan(data.matrix(data_compound[, .(x, y)]), eps = 1.4, minPts = 5)
# db_res

data_compound[, class := as.factor(db_res$cluster)]

ggplot(data_compound, aes(x, y, color = class, shape = class)) +
  geom_point(size = 2) +
  theme_bw()
```



Hierarchical clustering

Types:

- Single-linkage
- Complete-linkage
- Average-linkage
- Centroid-linkage
- Ward's minimum variance method
- etc.

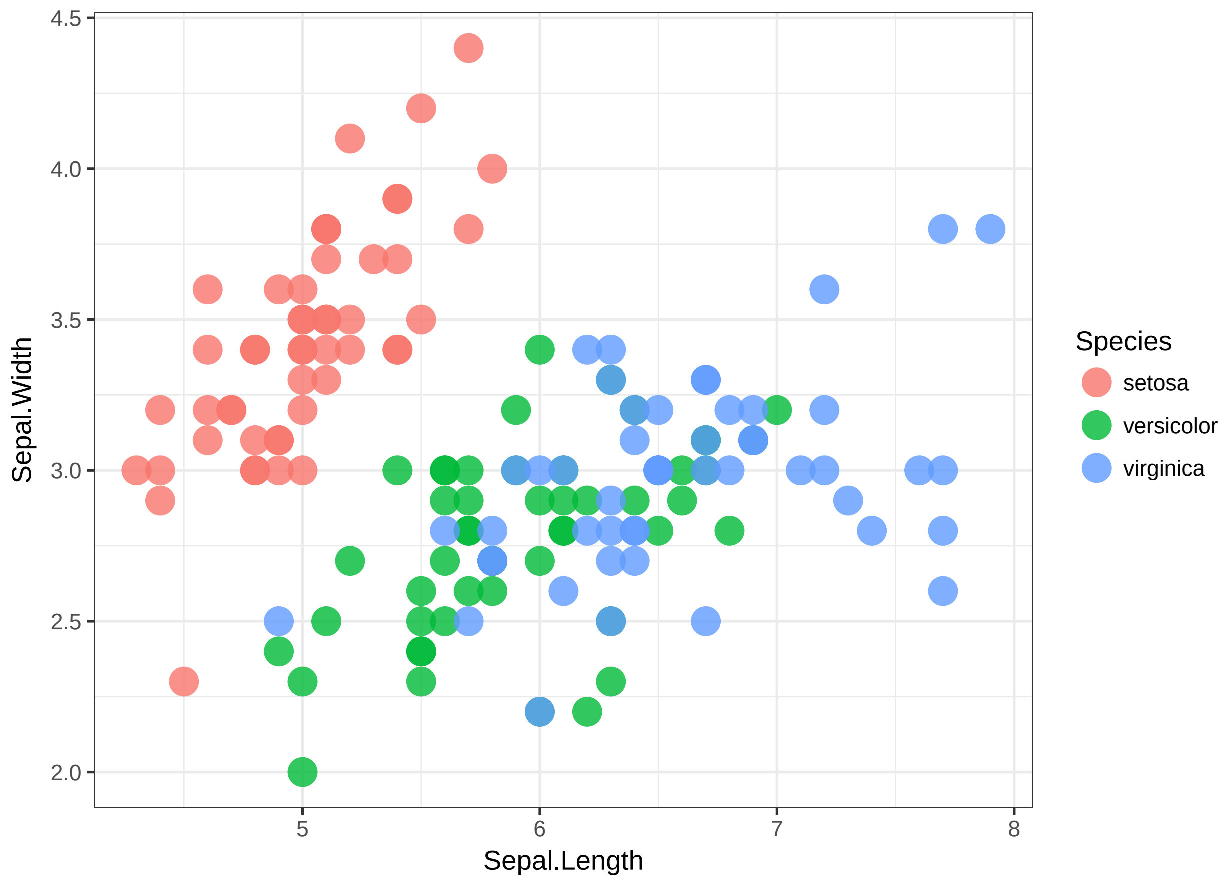
Result is a dendrogram.

Criteria:

- single-linkage: $\min\{d(a,b) : a \in A, b \in B\}$
- complete-linkage: $\max\{d(a,b) : a \in A, b \in B\}$
- average-linkage: $\frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$
- centroid-linkage: $||c_t - c_s||$, where c_s and c_t are the centroids of clusters s and t.

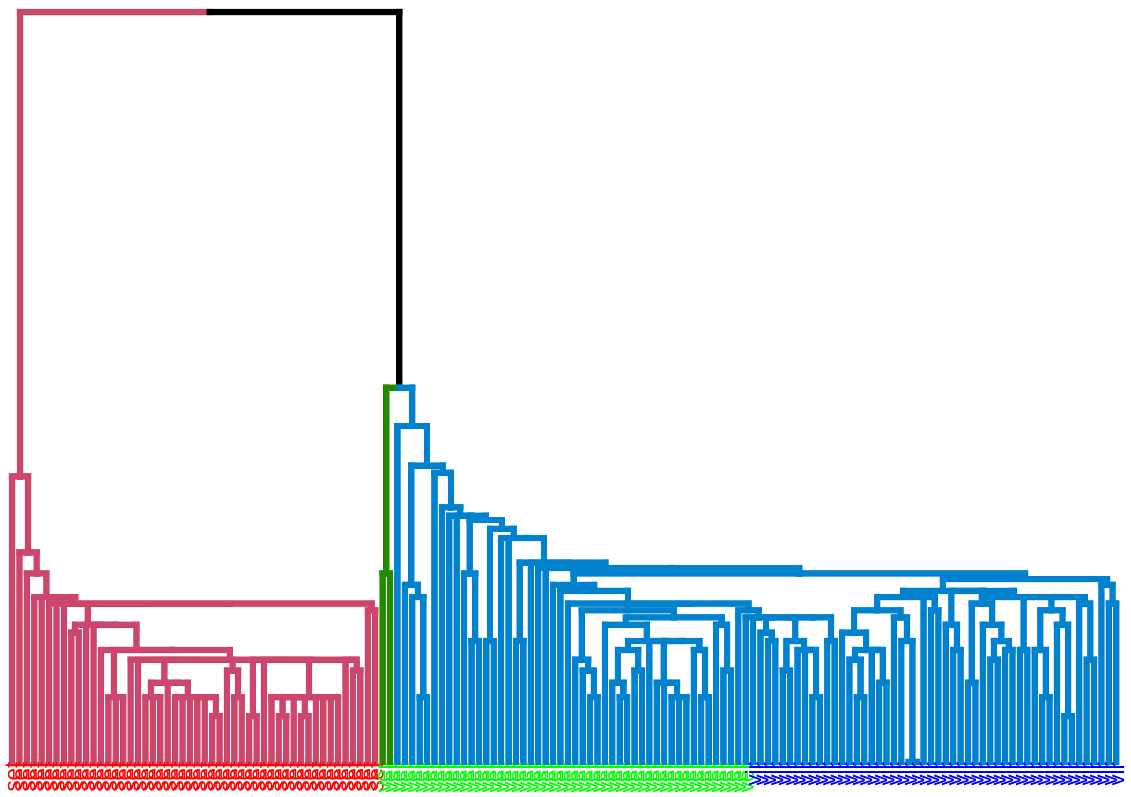
IRIS dataset use case

```r
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(alpha = 0.8, size = 5) +
  theme_bw()
```



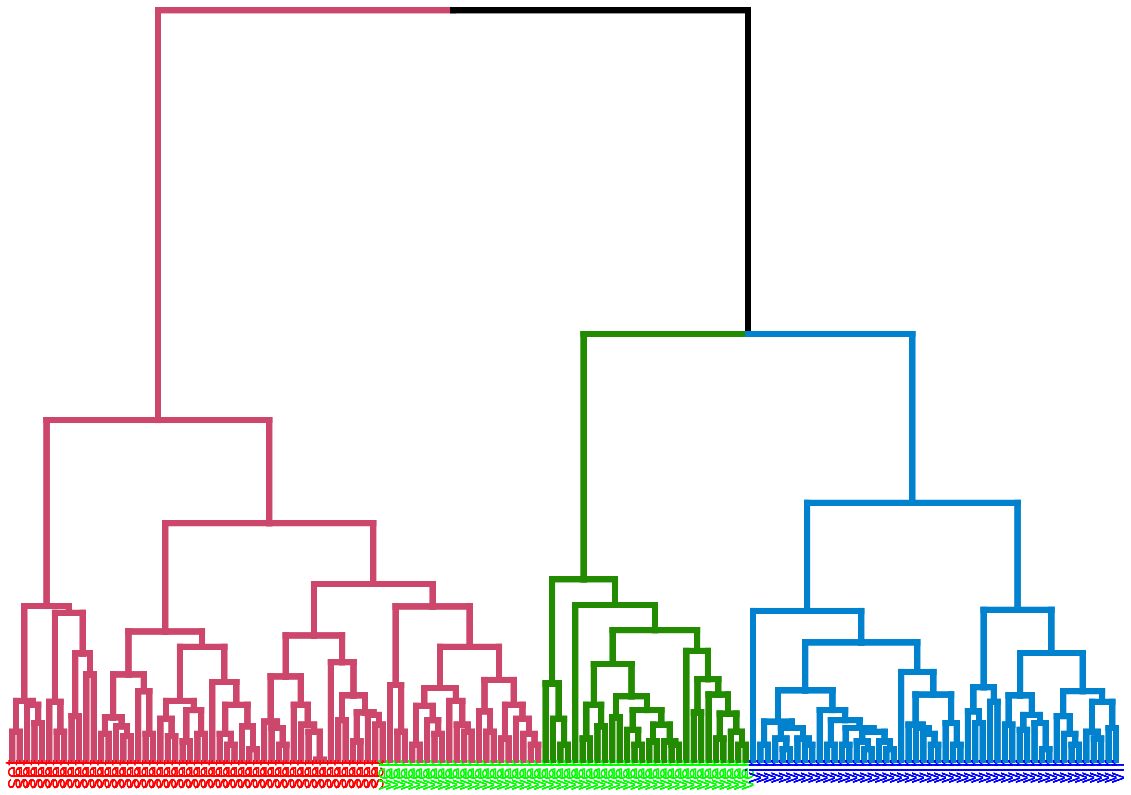
Single linkage:

```
library(ggdendro)
library(dendextend)

data_m <- iris[,-5]

hie_single <- hclust(dist(data_m), method = "single")
dend <- as.dendrogram(hie_single)
dend <- dend %>% set("branches_k_color", k = 3) %>%
  set("branches_lwd", 1.2) %>%
  set("labels", rep(c("set", "ver", "vir"), each = 50)) %>%
  set("labels_colors", rep(c("red", "green", "blue"), each = 50)) %>%
  set("labels_cex", 0.6)
ggd1 <- as.ggdend(dend)
ggplot(ggd1)
```



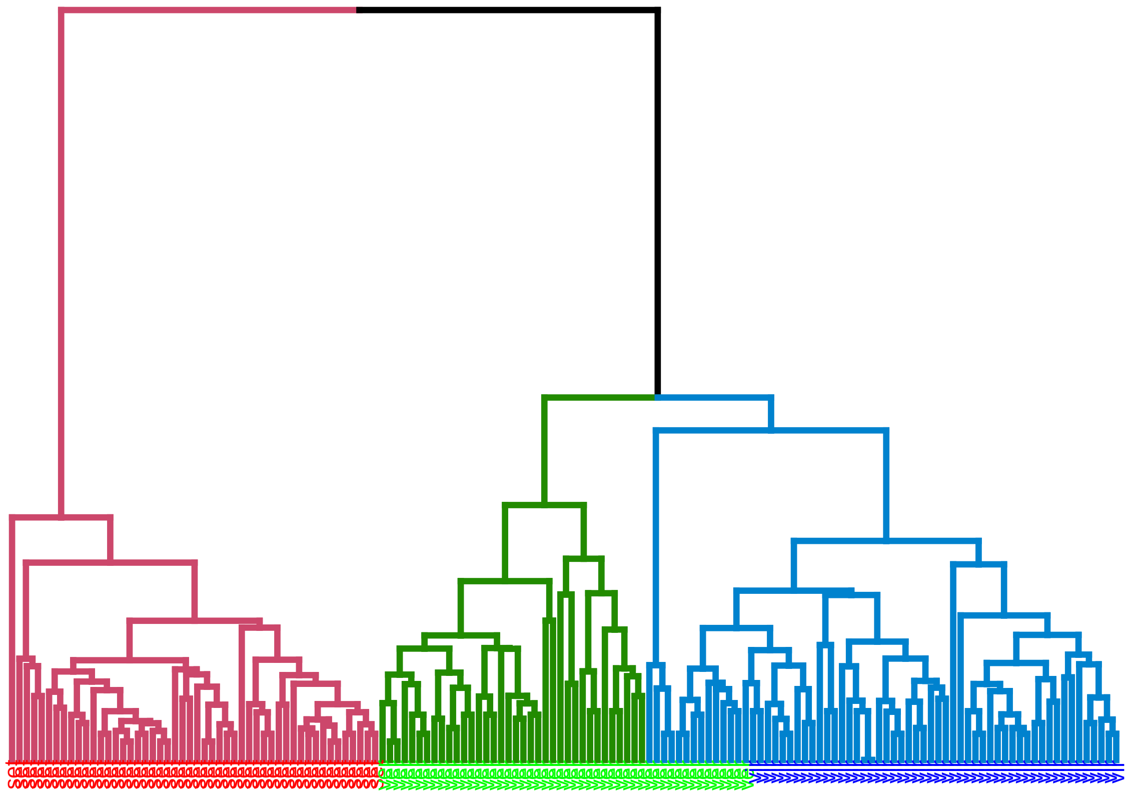
Complete linkage:

```
hie_complete <- hclust(dist(data_m), method = "complete")
dend <- as.dendrogram(hie_complete)
dend <- dend %>% set("branches_k_color", k = 3) %>%
  set("branches_lwd", 1.2) %>%
  set("labels", rep(c("set", "ver", "vir"), each = 50)) %>%
  set("labels_colors", rep(c("red", "green", "blue"), each = 50)) %>%
  set("labels_cex", 0.6)
ggd1 <- as.ggdend(dend)
ggplot(ggd1)
```



Average linkage:

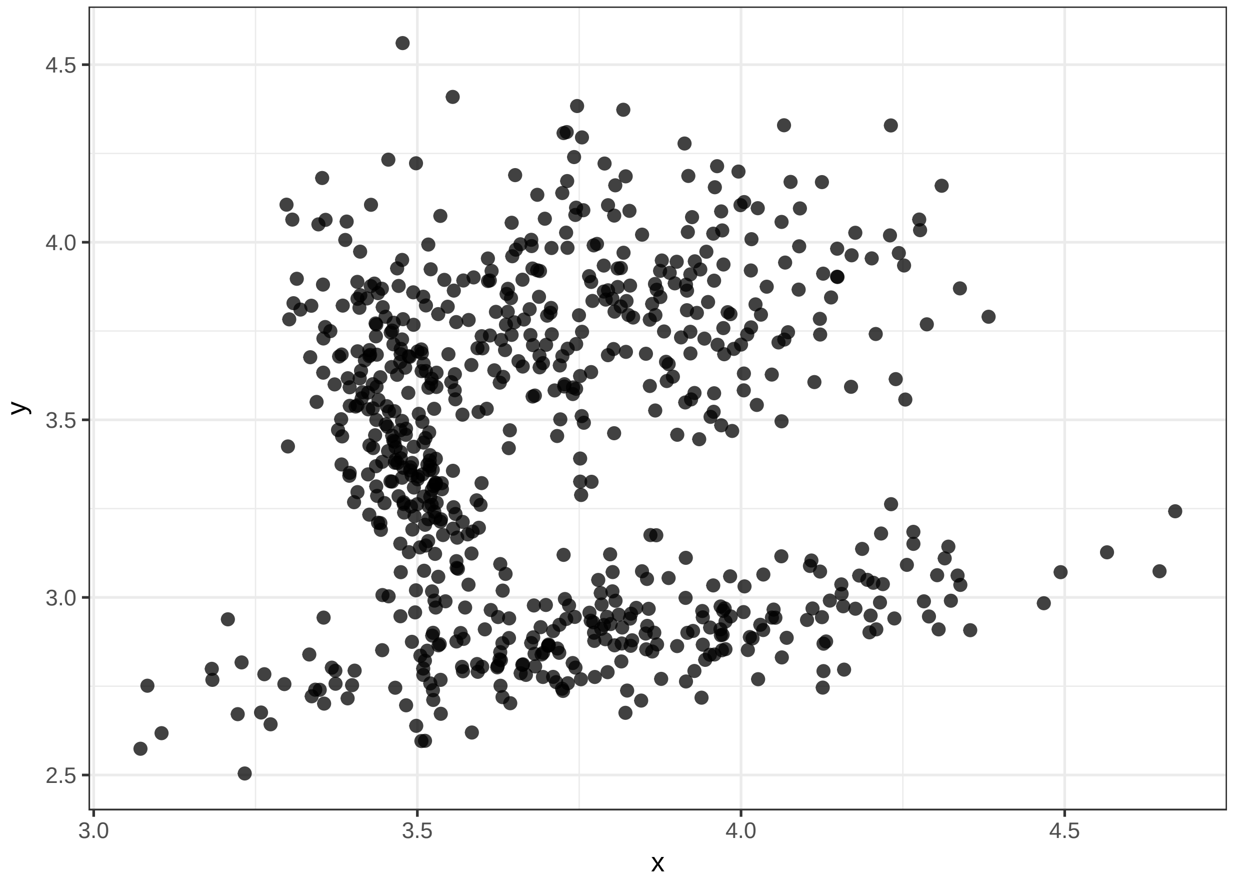
```
hie_ave <- hclust(dist(data_m), method = "average")
dend <- as.dendrogram(hie_ave)
dend <- dend %>% set("branches_k_color", k = 3) %>%
  set("branches_lwd", 1.2) %>%
  set("labels", rep(c("set", "ver", "vir"), each = 50)) %>%
  set("labels_colors", rep(c("red", "green", "blue"), each = 50)) %>%
  set("labels_cex", 0.6)
ggd1 <- as.ggdend(dend)
ggplot(ggd1)
```



Connected data

The most close scenario of real data.

```
set.seed(5)
library(MASS)

datas_2 <- as.data.table(rbind(
  mvrnorm(220, mu = c(3.48, 3.4), Sigma = matrix(c(0.005, -0.015, -0.01, 0.09), nrow = 2)),
  mvrnorm(280, mu = c(3.8, 3.8), Sigma = matrix(c(0.05, 0, 0, 0.05), nrow = 2)),
  mvrnorm(220, mu = c(3.85, 2.9), Sigma = matrix(c( 0.1, 0.03, 0.03, 0.017), nrow = 2))
  ))

setnames(datas_2, c("V1", "V2"), c("x", "y"))

ggplot(datas_2, aes(x, y)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```

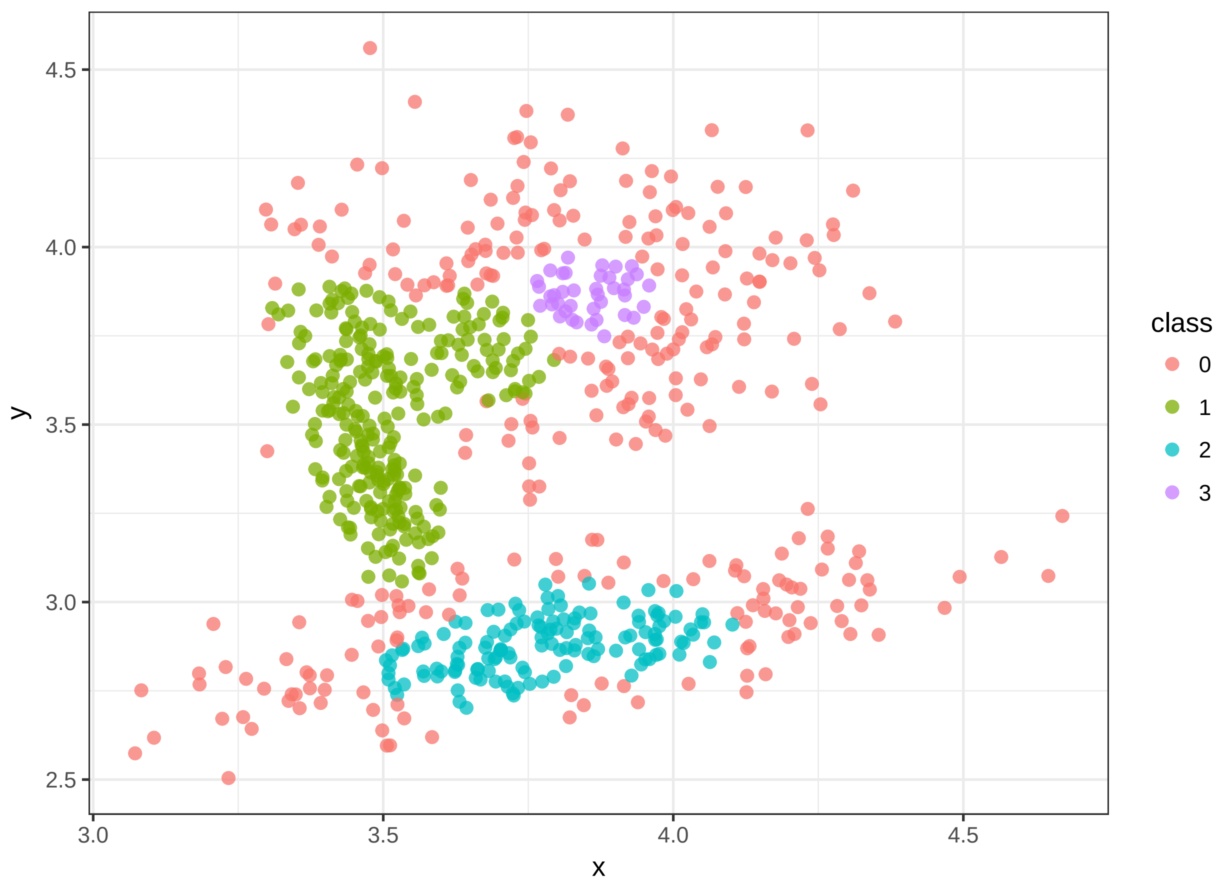


DBSCAN - result for connected data

$\epsilon = 0.08$, $minPts = 18$.

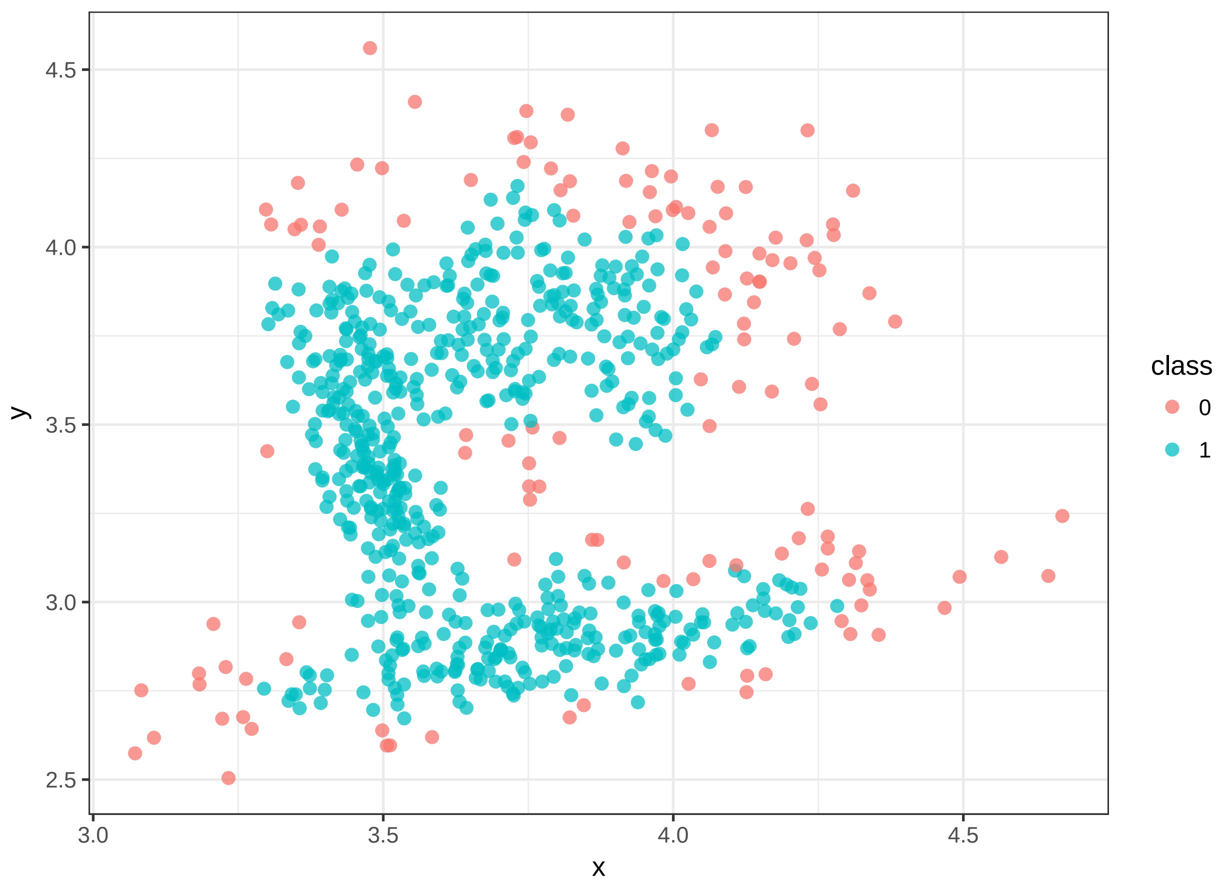
```
db_res <- dbscan(datas_2, eps = 0.08, minPts = 18)
```

```
data_all <- data.table(datas_2, class = as.factor(db_res$cluster))

ggplot(data_all, aes(x, y, color = class)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```



Change minPts to 10.

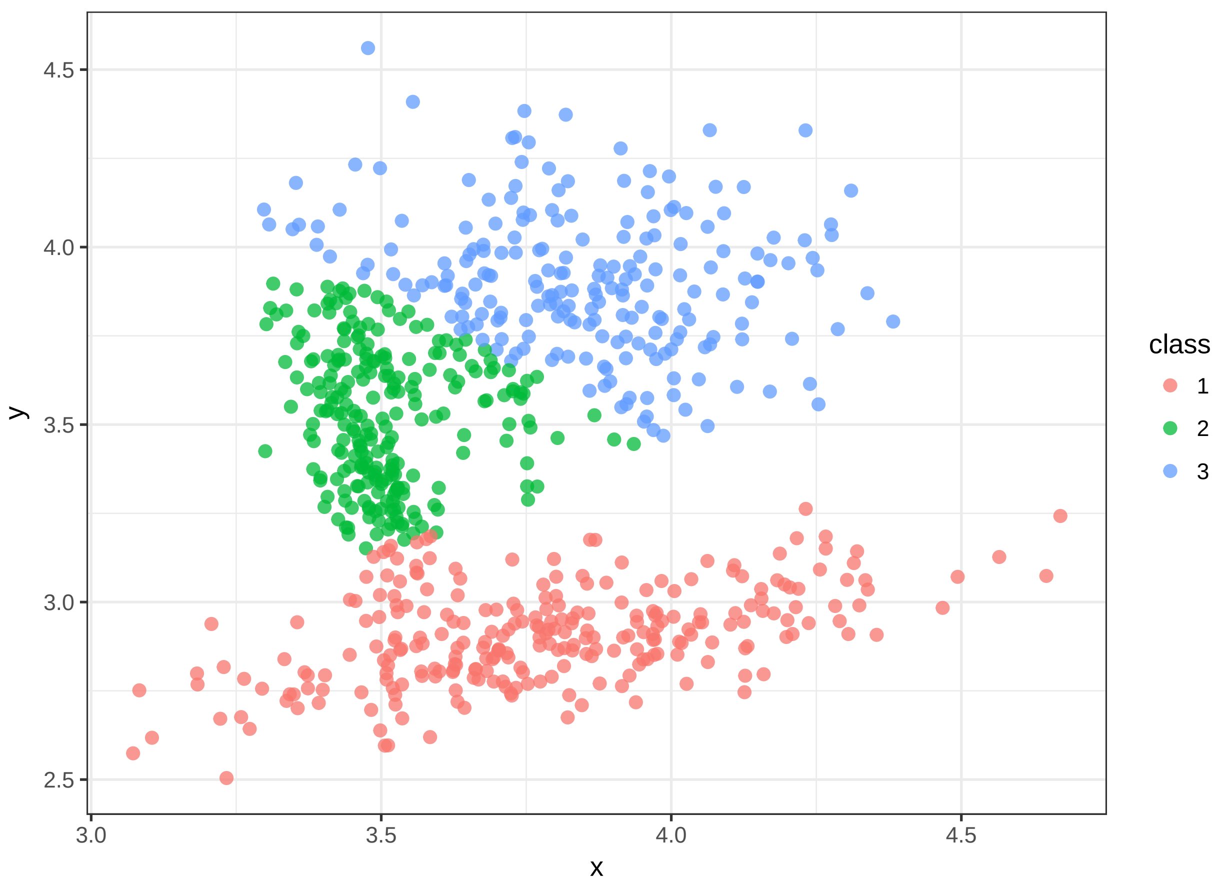
```
db_res <- dbscan(datas_2, eps = 0.08, minPts = 10)

data_all <- data.table(datas_2, class = as.factor(db_res$cluster))

ggplot(data_all, aes(x, y, color = class)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```



DBSCAN is very sensitive to parameter settings.

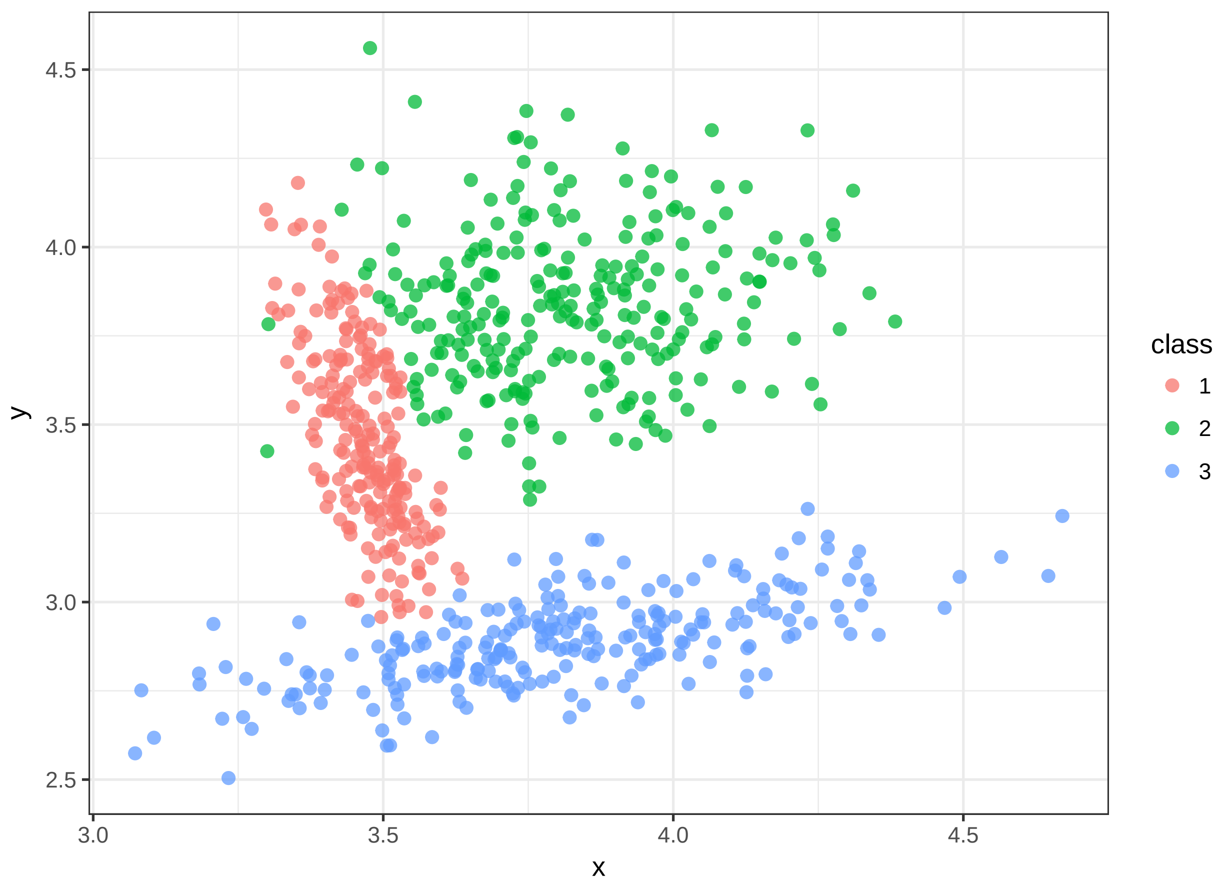
K-means - result for connected data

```
km_res <- kmeans(datas_2, 3)

data_all[, class := as.factor(km_res$cluster)]

ggplot(data_all, aes(x, y, color = class)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```



Gaussian model-based clustering result

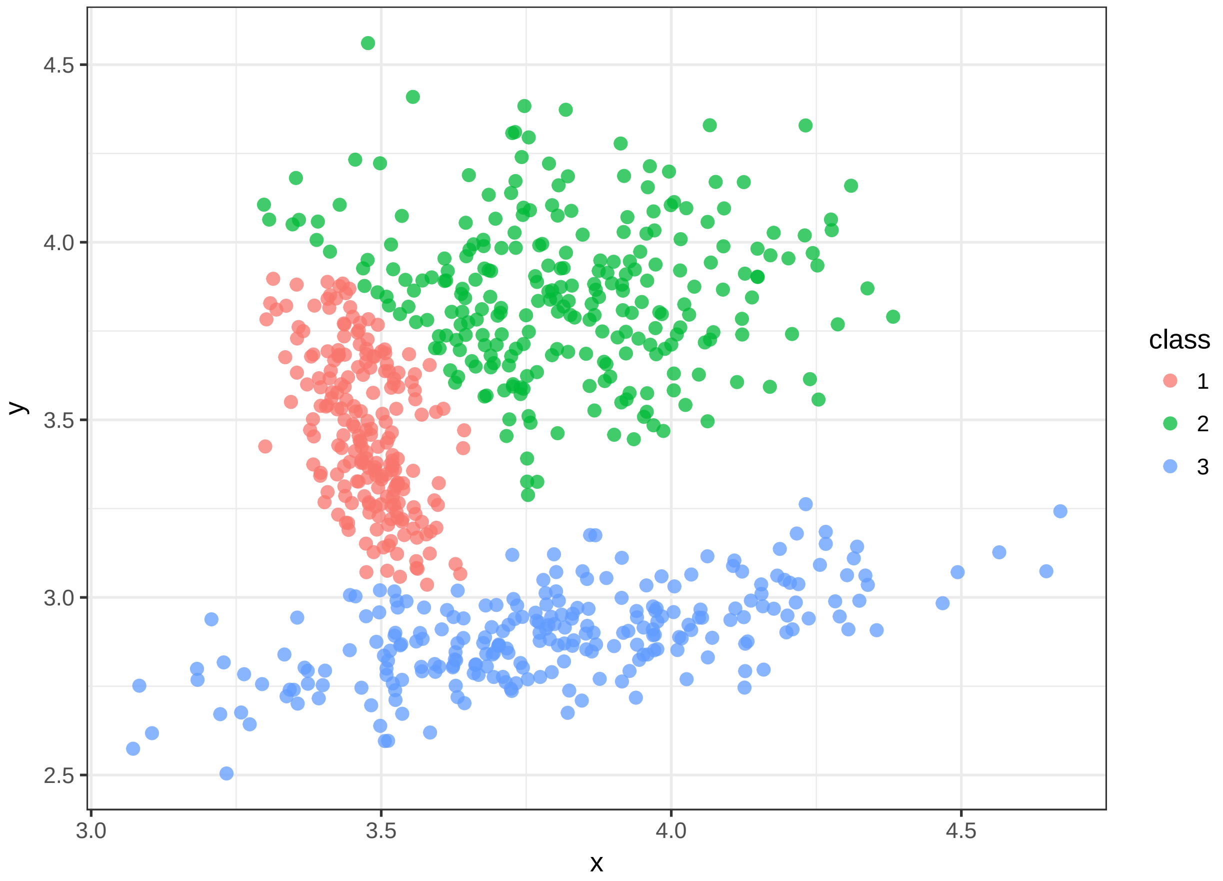
```
m_res <- Mclust(datas_2, G = 3, modelNames = "VVV")

data_all[, class := as.factor(m_res$classification)]

ggplot(data_all, aes(x, y, color = class)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```



Almost perfect due to normality of data.

Spectral clustering result

```
res <- specc(data.matrix(datas_2[, .(x, y)]), centers = 3)

data_all[, class := as.factor(res)]

ggplot(data_all, aes(x, y, color = class)) +
  geom_point(alpha = 0.75, size = 2) +
  theme_bw()
```



Very nice result!

Other types of clustering methods

- Grid-based
- Subspace clustering
- Multi-view clustering
- Based on artificial neural networks (e.g. SOM)
- Consensus (ensemble) clustering
- Data stream(s) clustering
- etc.

Conclusions

- We have many types of clustering methods,
- Different datasets needs different clustering methods,
- Automatic determination of a number of clusters can be tricky,
- Real datasets are usually connected - density-based methods can fail,
- Outliers (anomalies) can significantly influence clustering results - solution is to preprocess the data,
- Some methods are not suited for large datasets - model-based or spectral,