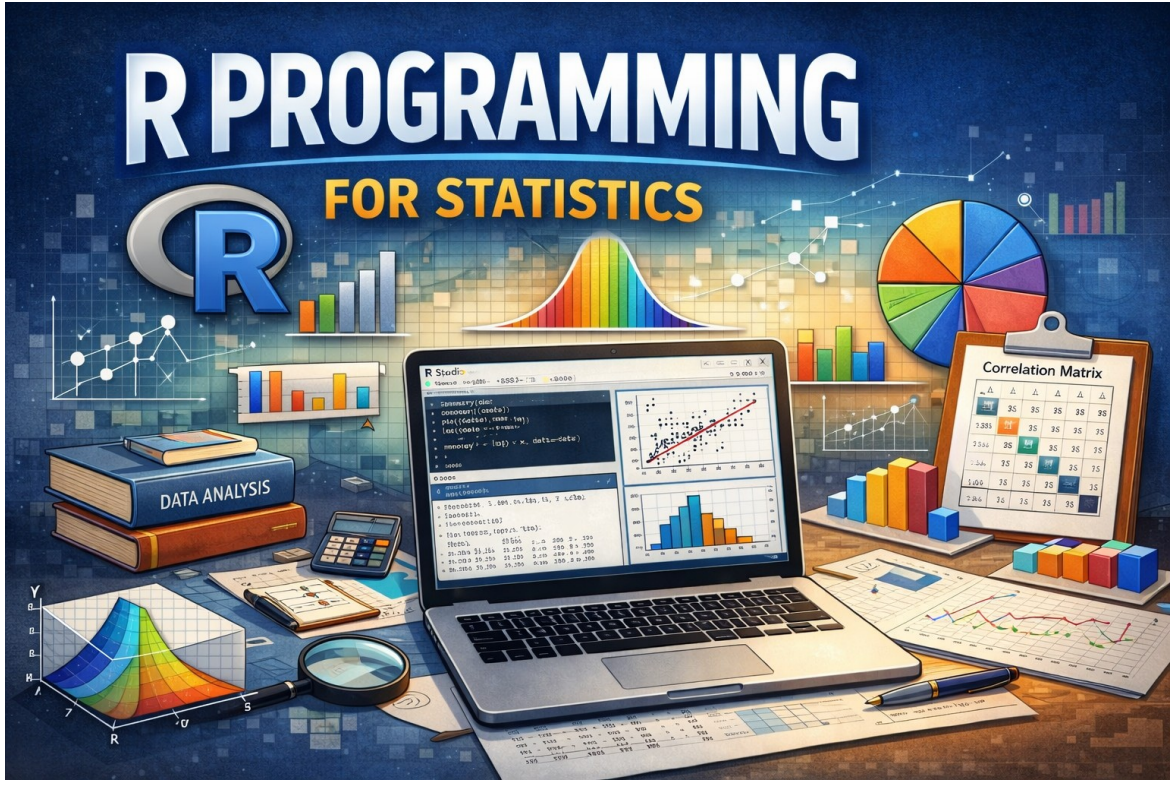
Basic Clustering in R

Nikolay Oskolkov, MRG Group Leader, LIOS, Riga, Latvia

R course, 12.02.2026



@NikolayOskolkov

@oskolkov.bsky.social

Personal homepage:
https://nikolay-oskolkov.com

Image generated by ChatGPT

Session content

Topics we'll cover in this session:

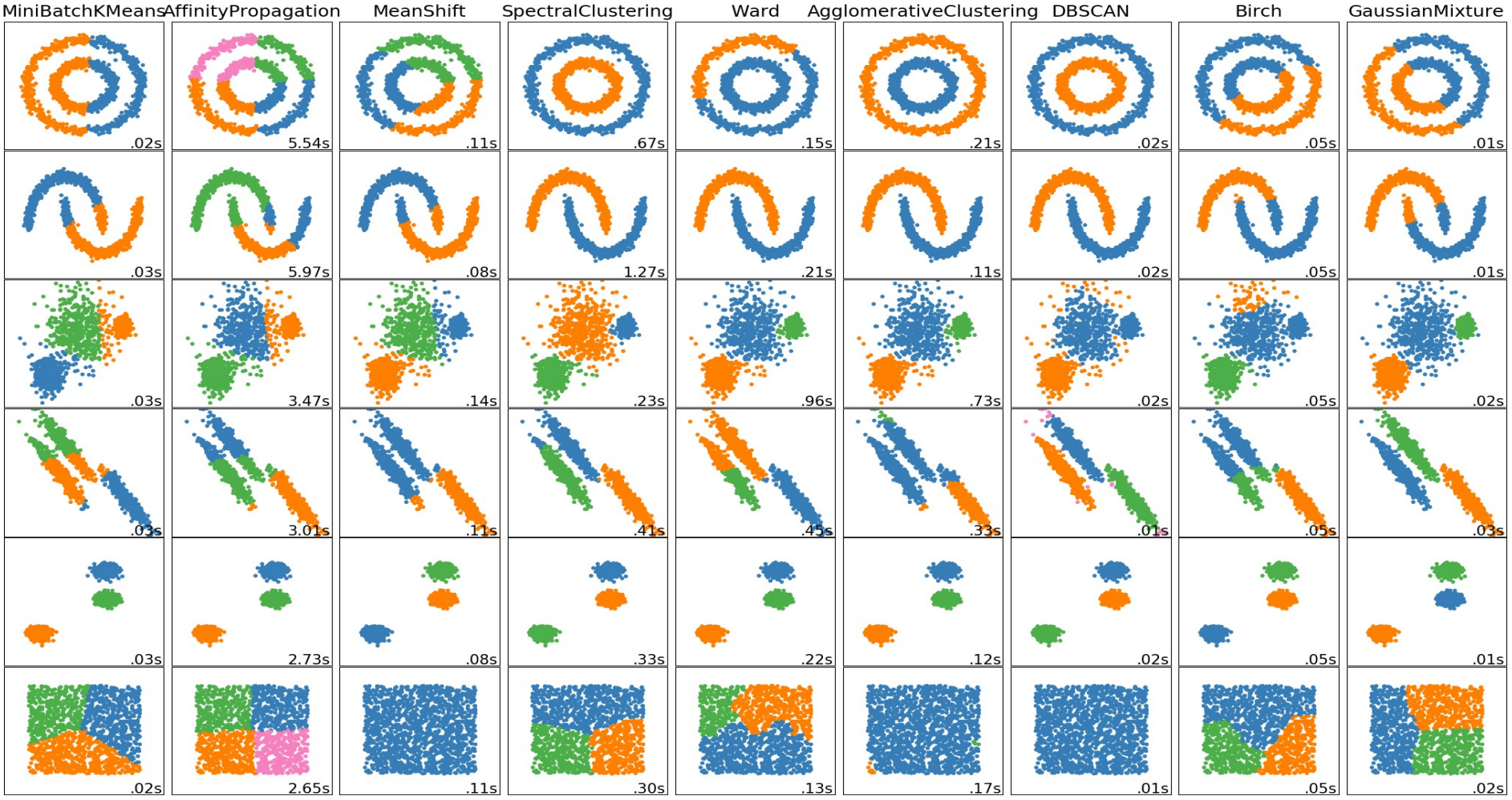
1) Hierachical agglomerative clustering

2) Partitioning (divisive) clustering

3) K-means and Gaussian Mixture Model (GMM)

4) Coding K-means and GMM from scratch in R and Python

Huge variety of clustering methods available

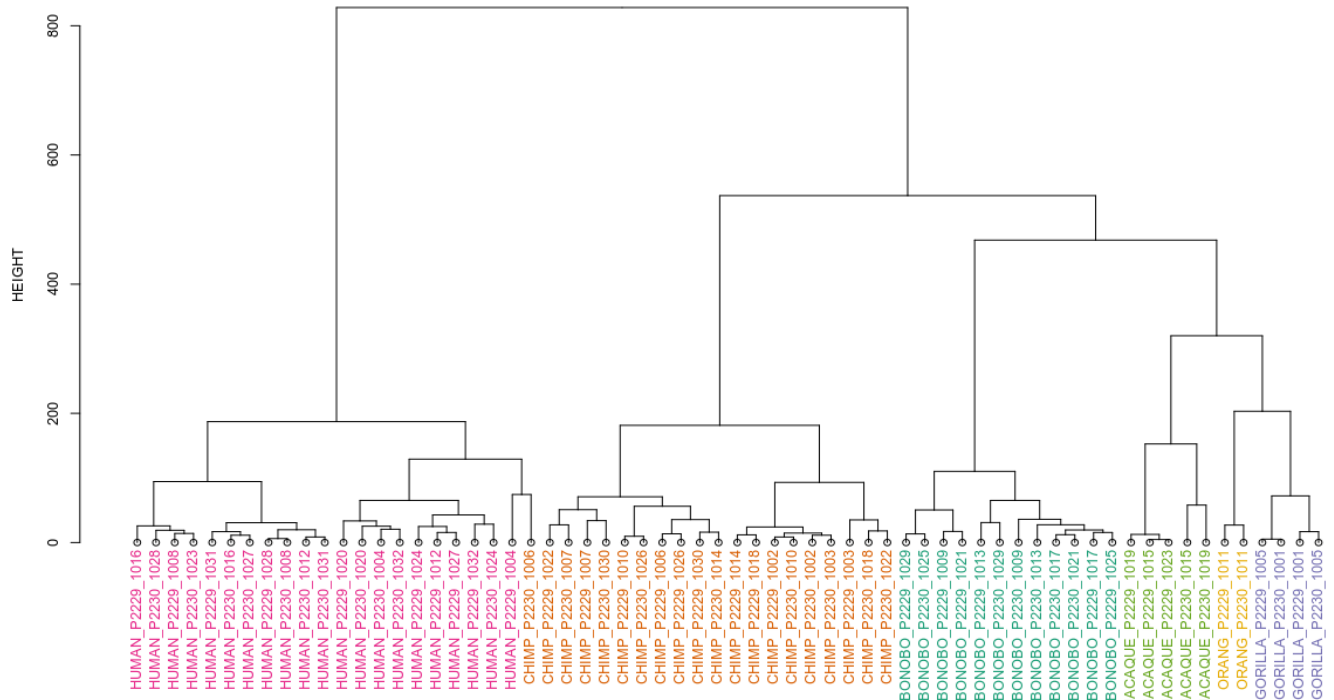


1) Hierarchical Clustering, 2) Partitioning Clustering, 3) Density-Based Clustering, 4) Graph-Based Clustering

Hierarchical clustering

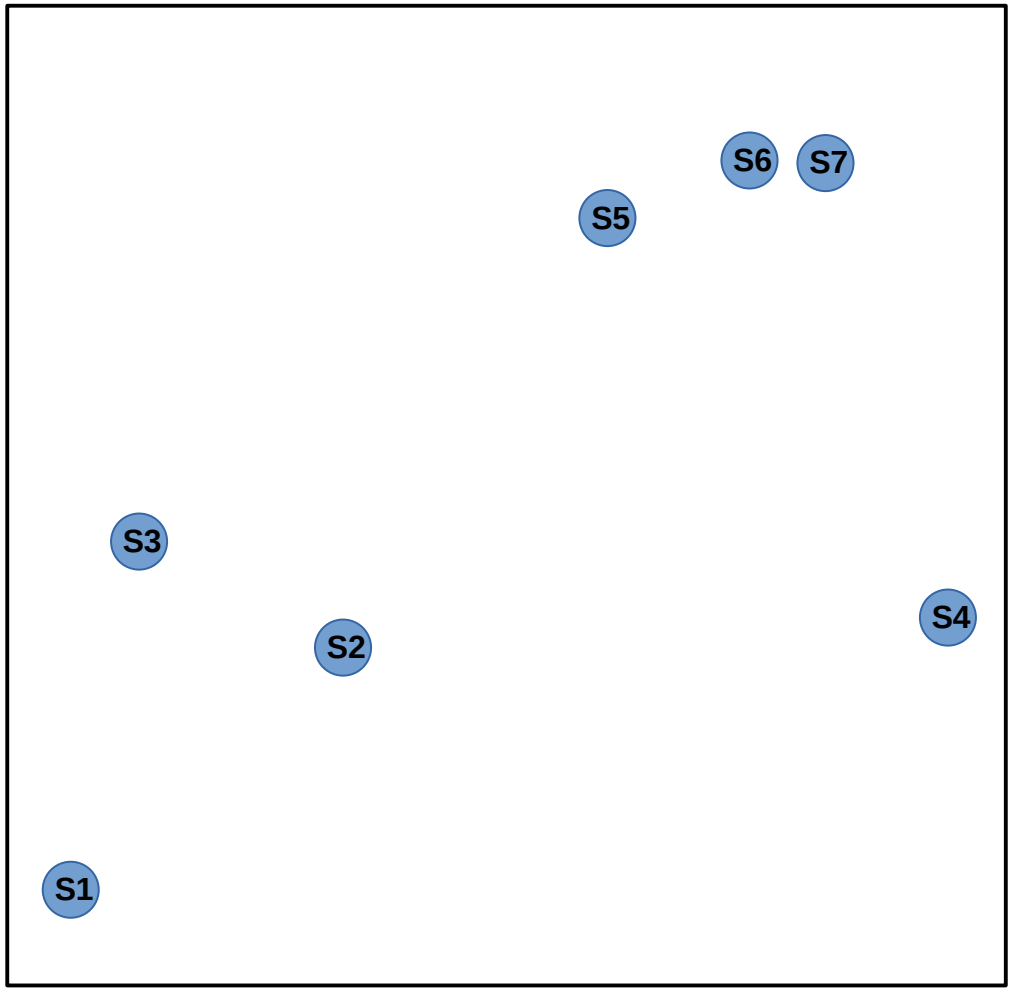
What is hierarchical clustering?

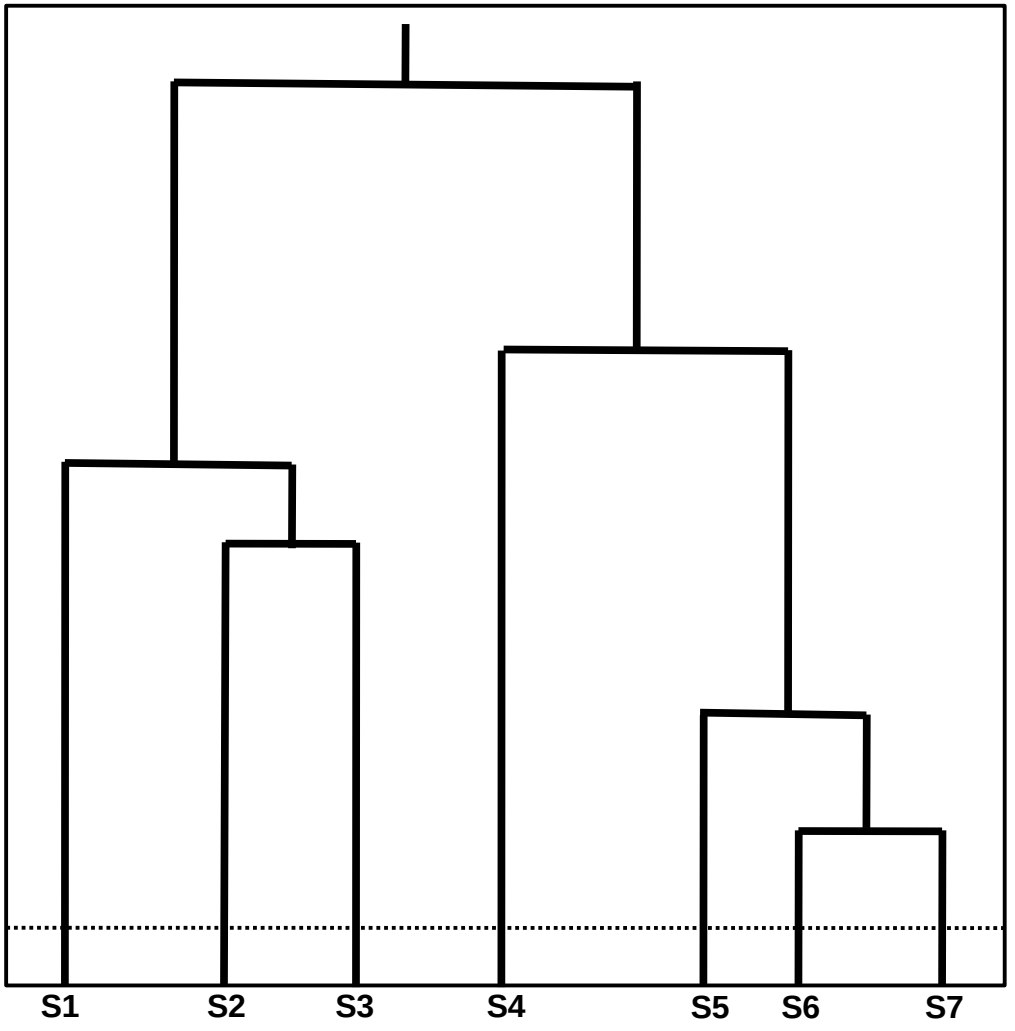
Clusters are nested (higher-ranked clusters contain lower-ranked clusters), in other words tree-like structure (dendrogram)



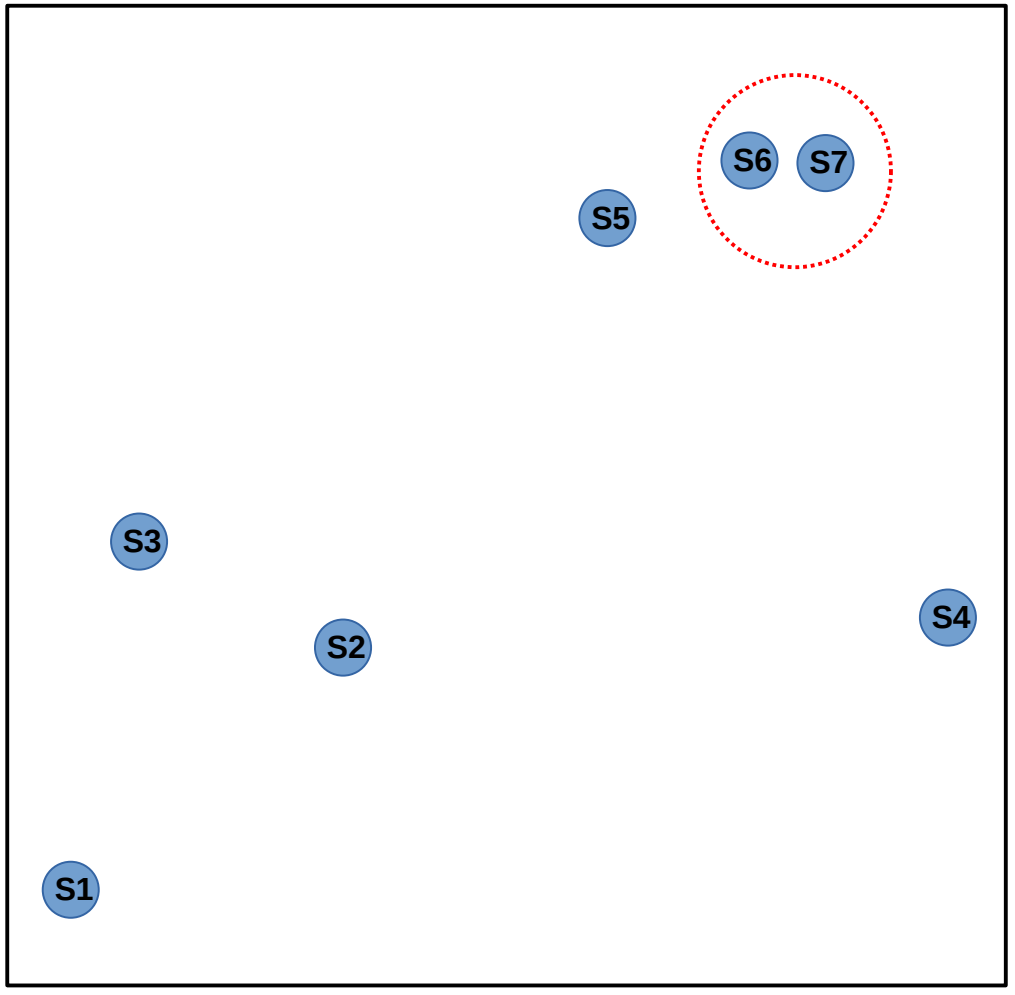
There are agglomerative (bottom-up approach) and divisive (top-down approach) hierachical types of hierachical clustering

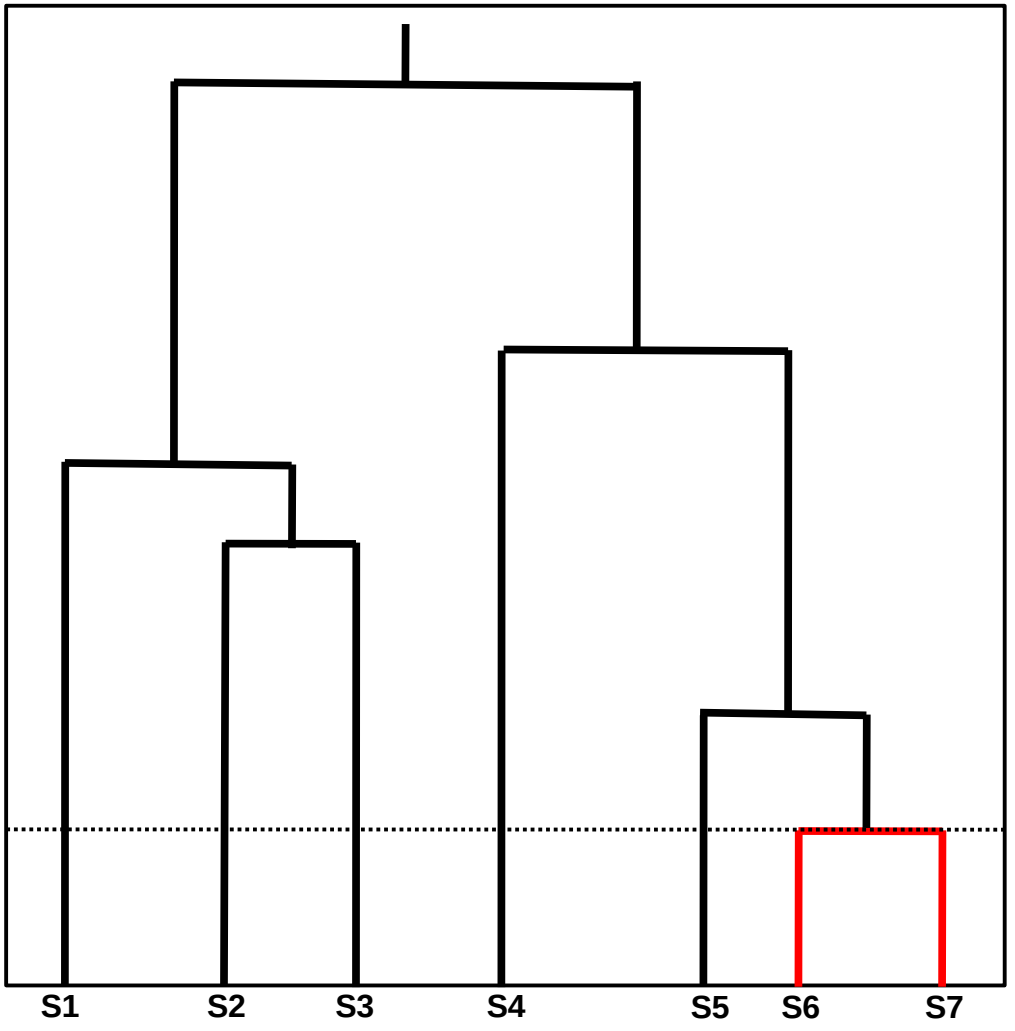
Agglomerative hierarchical clustering



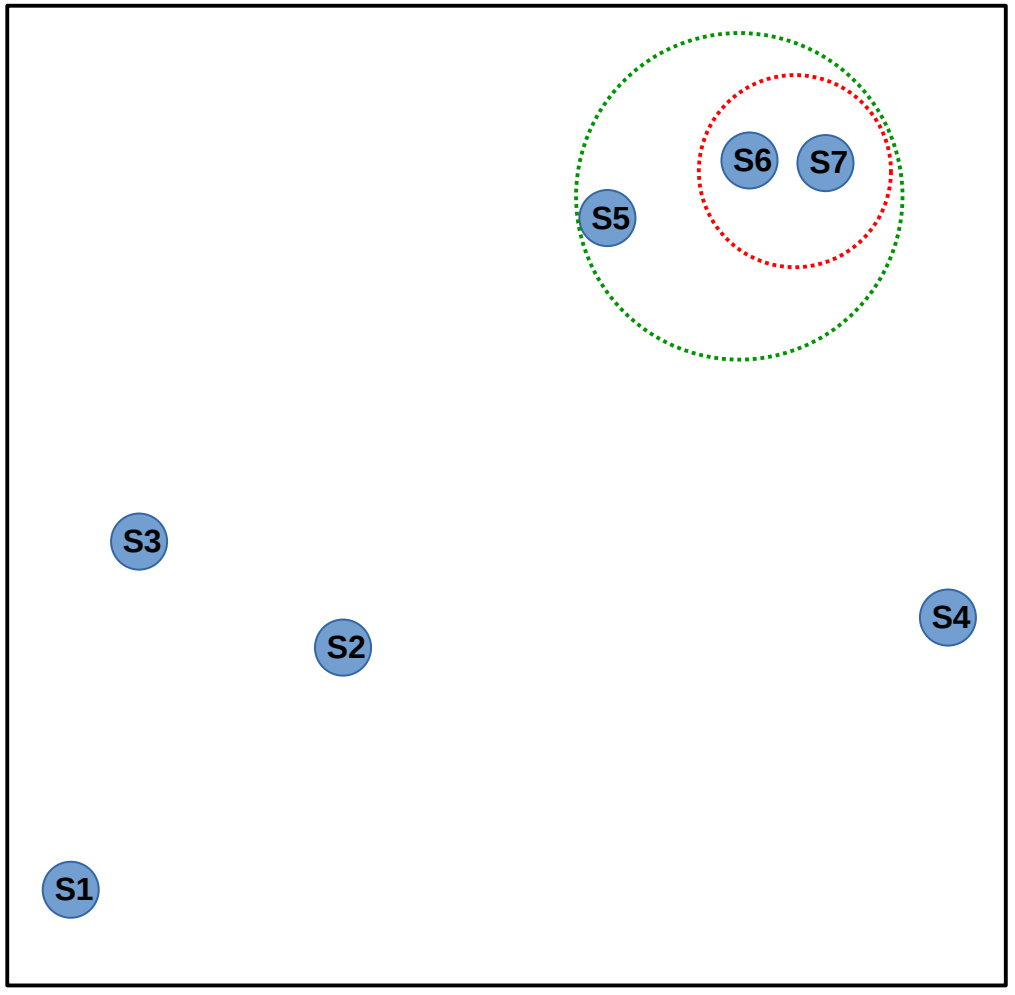


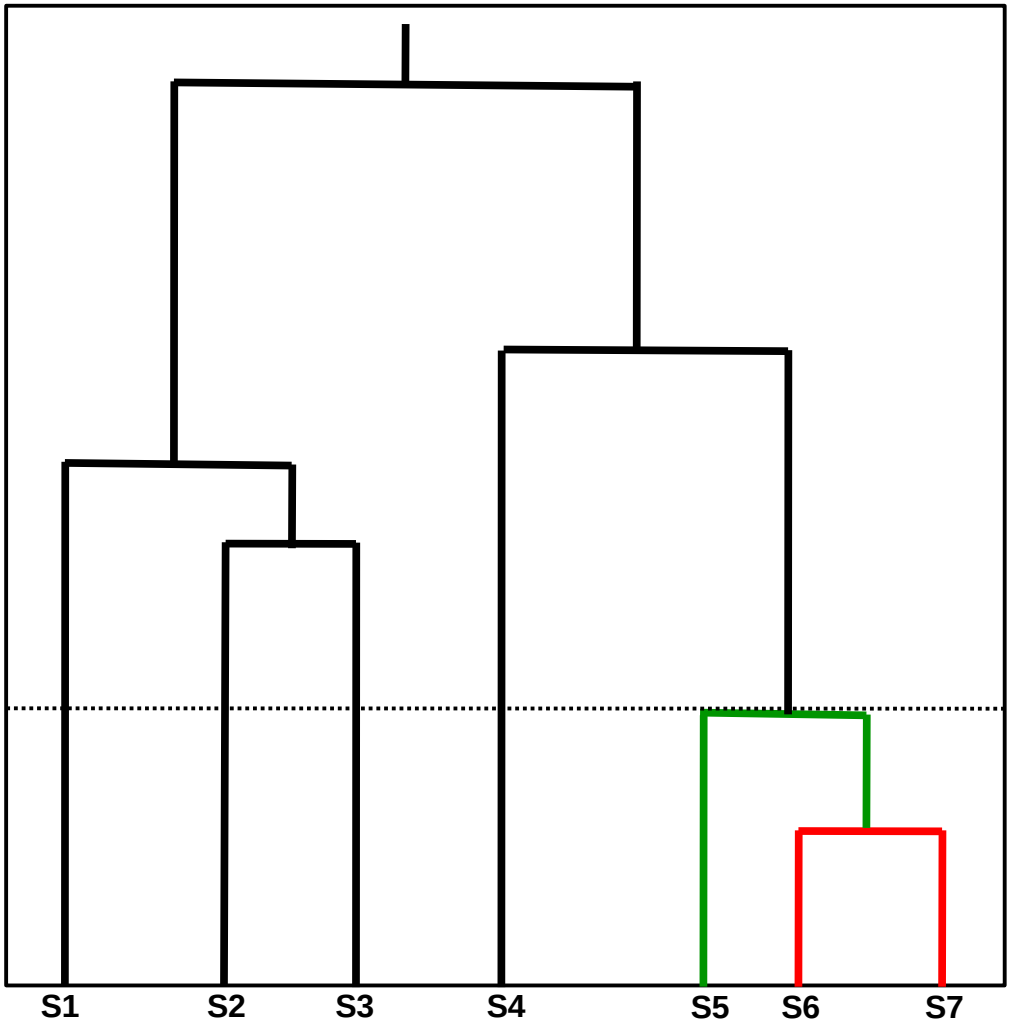
Agglomerative hierarchical clustering



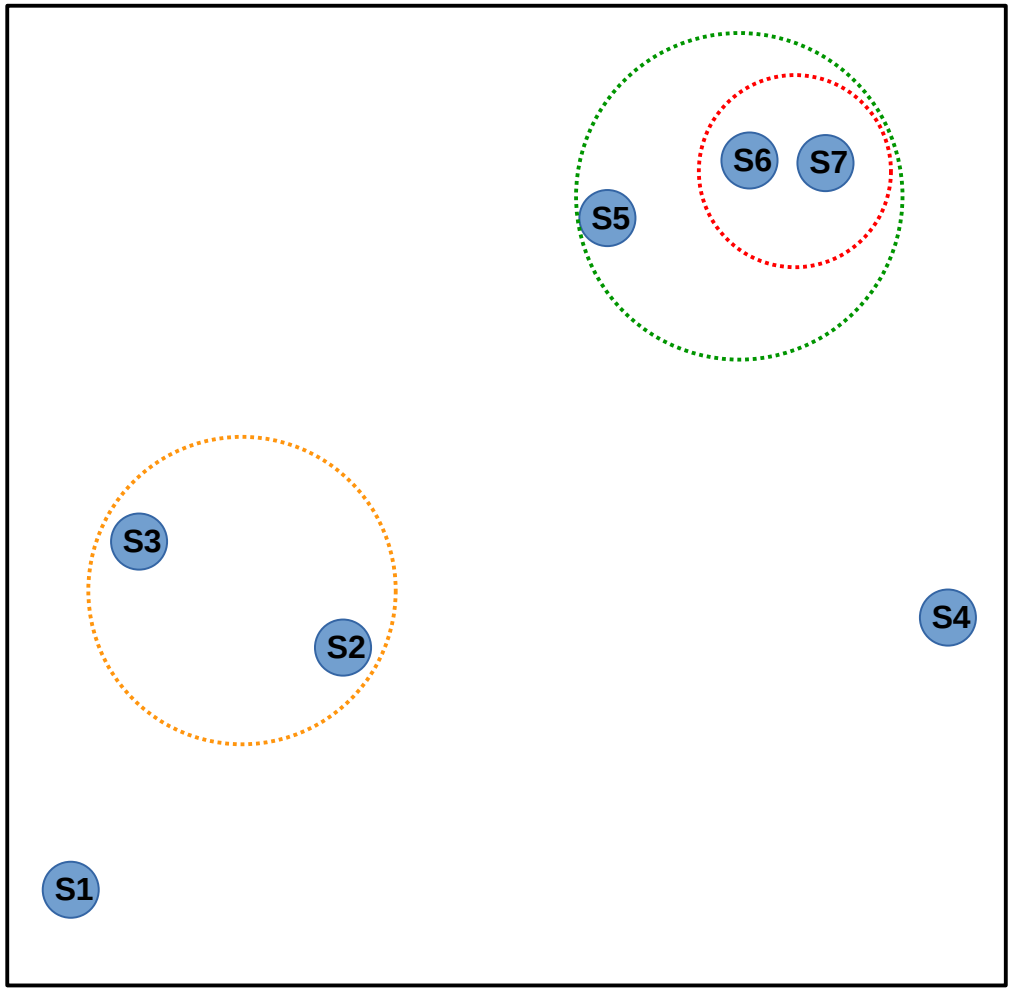


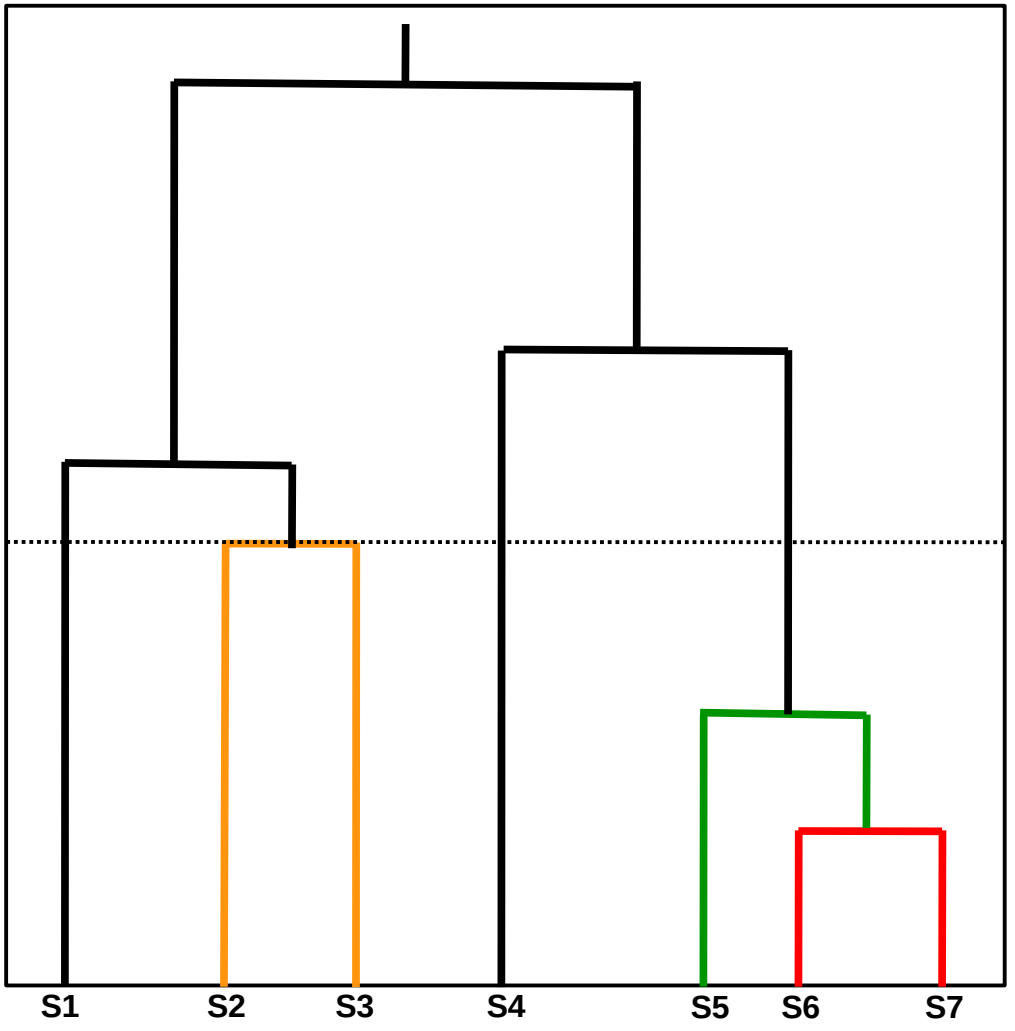
Agglomerative hierarchical clustering



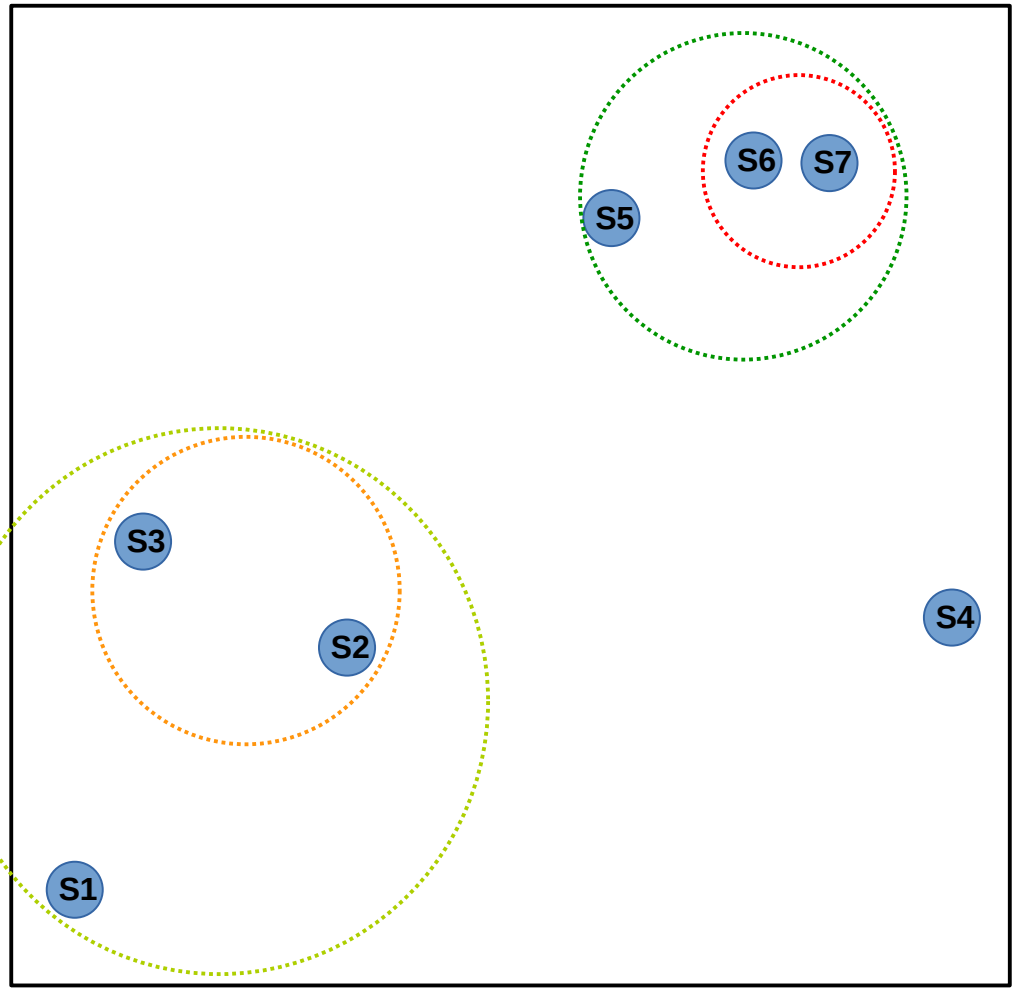


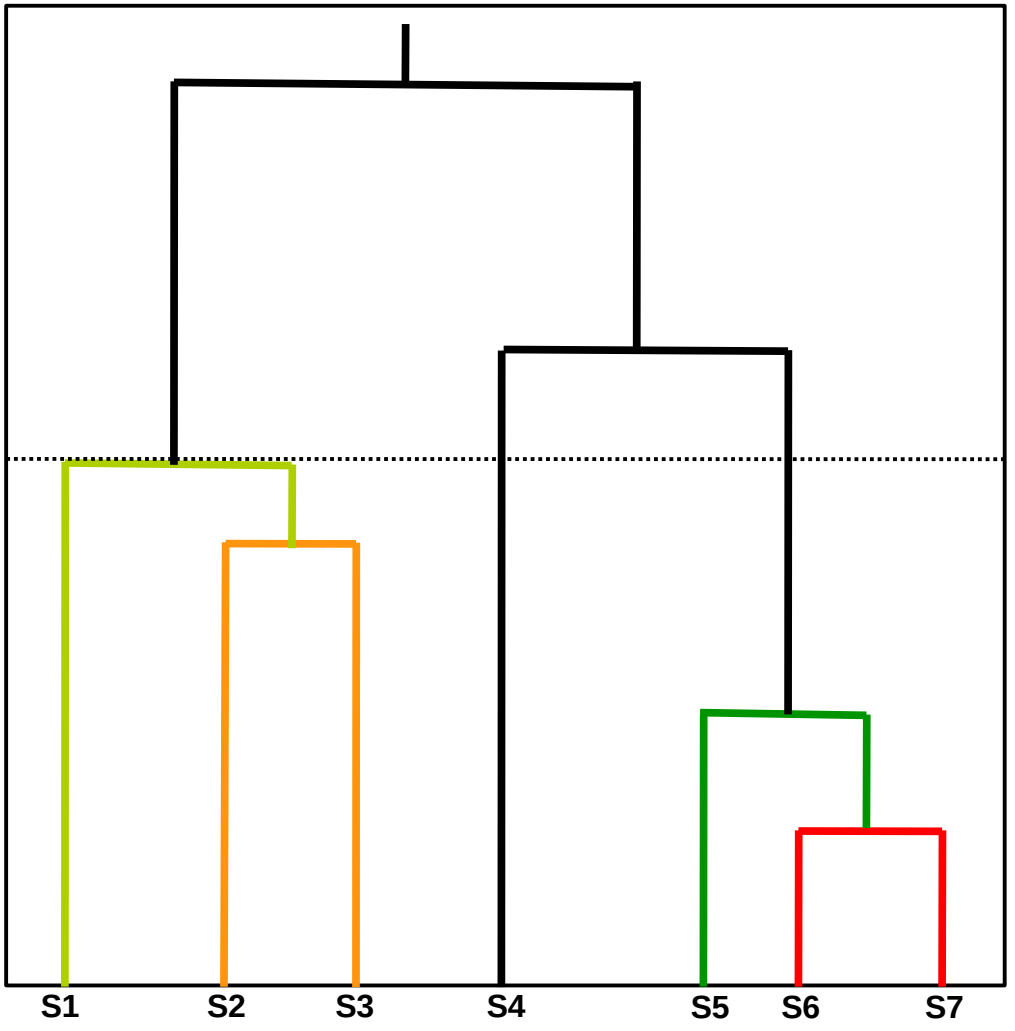
Agglomerative hierarchical clustering





Agglomerative hierarchical clustering





Activities RStudio sön 15:05 sv

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

hclust_toy.R

Source on Save Run Source

```
1  df <- data.frame(x = c(0, 5, 2, 12, 6, 8, 7), y = c(0, 3, 5, 3, 8, 9, 9))
2  df
3  plot(y ~ x, data = df, pch = 16, cex = 5, col = 2:8, xaxt="n", yaxt="n")
4  axis(1, at = seq(0, 12, by = 1)); axis(2, at = seq(0, 9, by = 1))
5  abline(h = 0:9, v = 0:12, lty = 1, col = "lightgrey")
6  points(y ~ x, data = df, pch = 16, cex = 5, col = 2:8)
7  points(y ~ x, data = df, pch = as.character(1:7))
8
9  dist(df)
10 clusters <- hclust(dist(df))
11 plot(as.dendrogram(clusters), ylab = "Euclidean distance")
12
```

12:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.2.3 · ~/

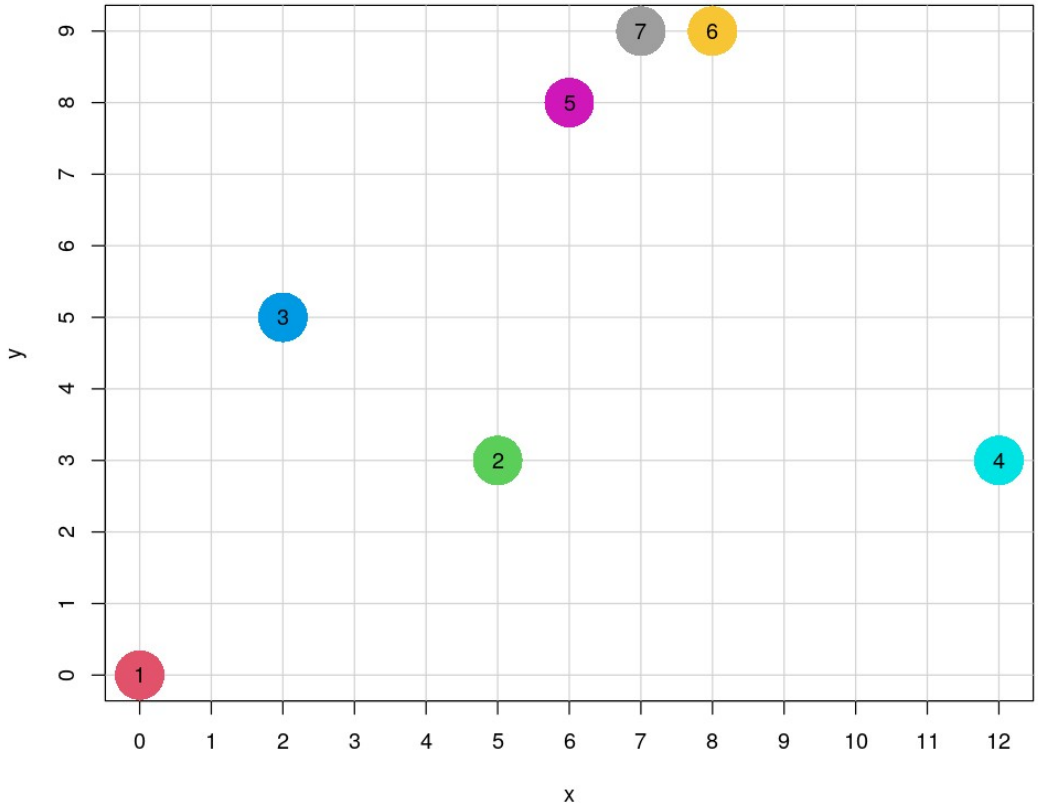
```
> df <- data.frame(x = c(0, 5, 2, 12, 6, 8, 7), y = c(0, 3, 5, 3, 8, 9, 9))
> df
   x y
1  0 0
2  5 3
3  2 5
4 12 3
5  6 8
6  8 9
7  7 9
> plot(y ~ x, data = df, pch = 16, cex = 5, col = 2:8, xaxt="n", yaxt="n")
> axis(1, at = seq(0, 12, by = 1)); axis(2, at = seq(0, 9, by = 1))
> abline(h = 0:9, v = 0:12, lty = 1, col = "lightgrey")
> points(y ~ x, data = df, pch = 16, cex = 5, col = 2:8)
> points(y ~ x, data = df, pch = as.character(1:7))
> dist(df)
         1        2        3        4        5        6
2  5.830952
3  5.385165  3.605551
4 12.369317  7.000000 10.198039
5 10.000000  5.099020  5.000000  7.810250
6 12.041595  6.708204  7.211103  7.211103  2.236068
7 11.401754  6.324555  6.403124  7.810250  1.414214  1.000000
> clusters <- hclust(dist(df))
> plot(as.dendrogram(clusters), ylab = "Euclidean distance")
>
```

Environment History Files Packages

Plots Connections Help Viewer Presentation

Zoom Export Publish

euc_dist_23 = sqrt ((2-5)^2 + (5-3)^2) = 3.605551



Activities RStudio sön 15:06 sv

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

hclust_toy.R

Source on Save Run Source

```
1  df <- data.frame(x = c(0, 5, 2, 12, 6, 8, 7), y = c(0, 3, 5, 3, 8, 9, 9))
2  df
3  plot(y ~ x, data = df, pch = 16, cex = 5, col = 2:8, xaxt="n", yaxt="n")
4  axis(1, at = seq(0, 12, by = 1)); axis(2, at = seq(0, 9, by = 1))
5  abline(h = 0:9, v = 0:12, lty = 1, col = "lightgrey")
6  points(y ~ x, data = df, pch = 16, cex = 5, col = 2:8)
7  points(y ~ x, data = df, pch = as.character(1:7))
8
9  dist(df)
10 clusters <- hclust(dist(df))
11 plot(as.dendrogram(clusters), ylab = "Euclidean distance")
12
```

12:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.2.3 · ~/

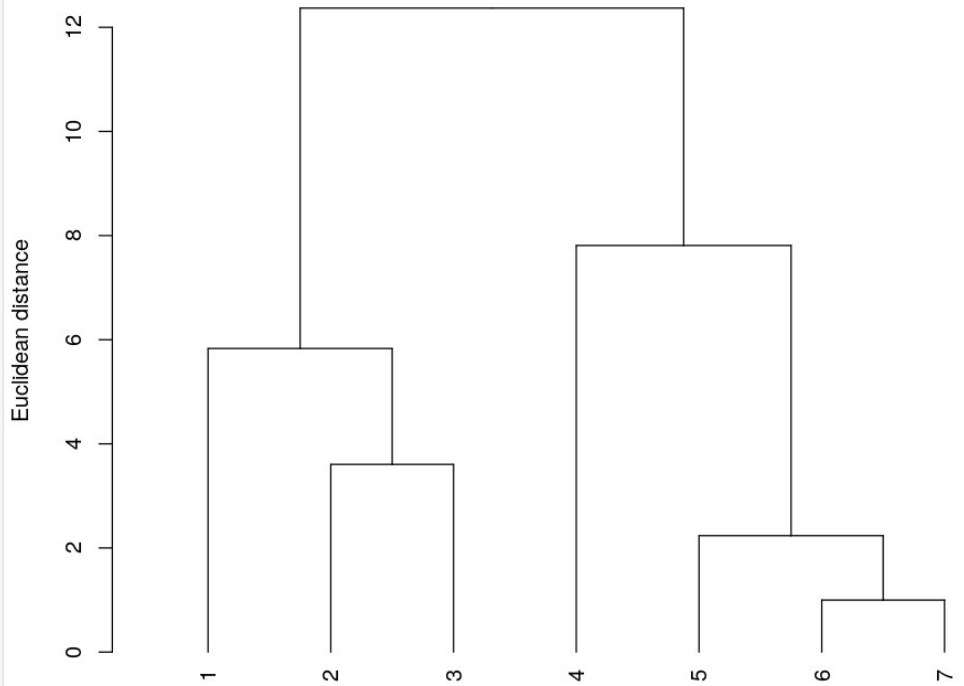
```
> df <- data.frame(x = c(0, 5, 2, 12, 6, 8, 7), y = c(0, 3, 5, 3, 8, 9, 9))
> df
   x y
1  0 0
2  5 3
3  2 5
4 12 3
5  6 8
6  8 9
7  7 9
> plot(y ~ x, data = df, pch = 16, cex = 5, col = 2:8, xaxt="n", yaxt="n")
> axis(1, at = seq(0, 12, by = 1)); axis(2, at = seq(0, 9, by = 1))
> abline(h = 0:9, v = 0:12, lty = 1, col = "lightgrey")
> points(y ~ x, data = df, pch = 16, cex = 5, col = 2:8)
> points(y ~ x, data = df, pch = as.character(1:7))
> dist(df)
         1        2        3        4        5        6
2  5.830952
3  5.385165  3.605551
4 12.369317  7.000000 10.198039
5 10.000000  5.099020  5.000000  7.810250
6 12.041595  6.708204  7.211103  7.211103  2.236068
7 11.401754  6.324555  6.403124  7.810250  1.414214  1.000000
> clusters <- hclust(dist(df))
> plot(as.dendrogram(clusters), ylab = "Euclidean distance")
>
```

Environment History Files Packages

Plots Connections Help Viewer Presentation

Zoom Export Publish

euc_dist_23 = sqrt ((2-5)^2 + (5-3)^2) = 3.605551



Types of hierarchical clustering

Add sample to cluster by:

1) Closest neighbor in cluster (single linkage)

2) Farthest neighbor in cluster (complete linkage)

3) Median / mean of cluster (UPGMA / UPGMC)

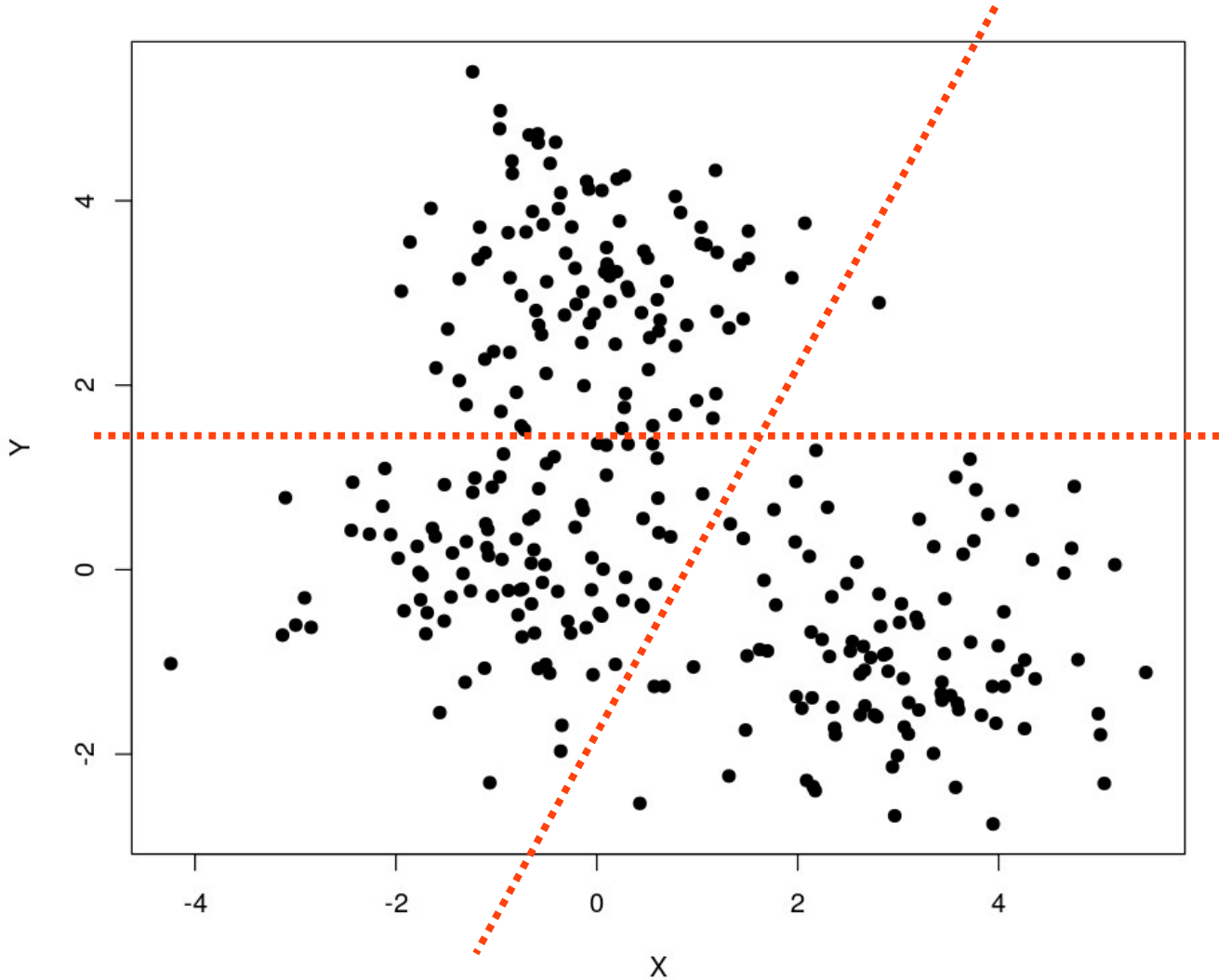
4) Least change in variation (Ward)

Disadvantages of hierachical clustering (no matter type):

Very sensitive to noise in data

Partitioning clustering

Partitioning principle



Assign points to K = 2, 3, 4, 5, … groups (clusters)

No dendrogram!

K-means, C-means, PAM, GMM, ...

Activities RStudio sön 15:35 sv

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Kmeans_from_scratch.R

Source on Save Run Source

```
1  library("MASS")
2
3  set.seed(123)
4
5  data1 <- as.data.frame(mvrnorm(n = 100, mu = c(-1, 0), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
6  data2 <- as.data.frame(mvrnorm(n = 100, mu = c(3, -1), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
7  data3 <- as.data.frame(mvrnorm(n = 100, mu = c(0, 3), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
8  data<-rbind(data1, data2, data3)
9  head(data)
10 plot(data, xlab = "X", ylab = "Y", pch = 19)
11
12 par(mfrow=c(2, 2))
13 K <- 3
14 set.seed(123)
15 centers <- data[sample(1:dim(data)[1], K), ]
16 plot(data, xlab = "X", ylab = "Y", pch = 19)
17 points(centers, col="blue", cex = 3, pch = 19)
18 for(t in 1:3)
```

12:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.2.3 · ~/

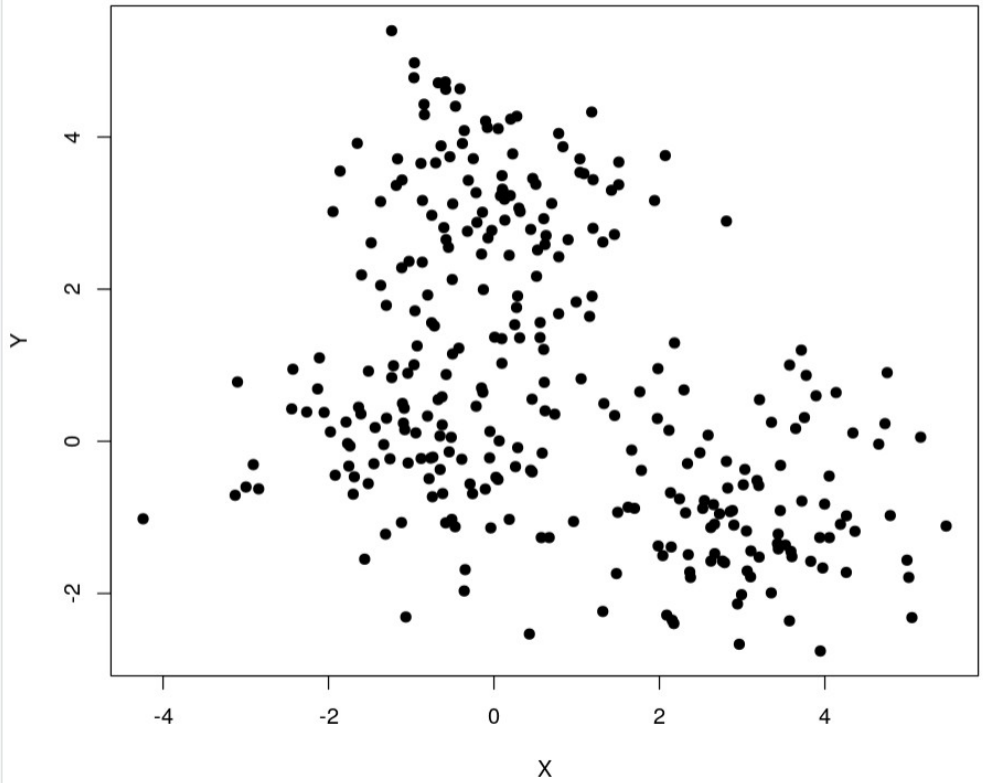
```
>
>
>
> library("MASS")
> set.seed(123)
> data1 <- as.data.frame(mvrnorm(n = 100, mu = c(-1, 0), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
> data2 <- as.data.frame(mvrnorm(n = 100, mu = c(3, -1), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
> data3 <- as.data.frame(mvrnorm(n = 100, mu = c(0, 3), Sigma=matrix(c(1, 0, 0, 1), ncol = 2)))
> data<-rbind(data1, data2, data3)
> head(data)
          V1          V2
1 -0.28959344 -0.56047565
2 -1.25688371 -0.23017749
3 -0.75330812  1.55870831
4 -0.65245740  0.07050839
5 -0.04838143  0.12928774
6 -0.95497228  1.71506499
> plot(data, xlab = "X", ylab = "Y", pch = 19)
>
```

Environment History Files Packages

R Global Environment

Plots Connections Help Viewer Presentation

Zoom Export Publish



Activities RStudio sön 15:36 sv

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins Project: (None)

Kmeans_from_scratch.R

Source on Save Run Source

```
12 par(mfrow=c(2, 2))
13 K <- 3
14 set.seed(123)
15 centers <- data[sample(1:dim(data)[1], K), ]
16 plot(data, xlab = "X", ylab = "Y", pch = 19)
17 points(centers, col="blue", cex = 3, pch = 19)
18 for(t in 1:3)
19 {
20 my_labels <- vector()
21 for(i in 1:dim(data)[1])
22 {
23   my_dist <- vector()
24   for(j in 1:K)
25   {
26     my_dist[j] <- sqrt((data[i, 1] - centers[j, 1])^2 + (data[i, 2] - centers[j, 2])^2)
27   }
28   my_labels[i] <- which.min(my_dist)
29 }
30 plot(data, xlab = "X", ylab = "Y", pch = 19)
31 points(data, col = my_labels, pch = 19)
32 points(centers, col = "blue", cex = 3, pch = 19)
33
34 new_centers <- list()
35 for(i in unique(my_labels))
36 {
37   new_centers[[i]] <- colMeans(data[my_labels == i,])
38 }
39 centers <- Reduce("rbind", new_centers)
40 }
```

33:1 (Top Level) R Script

Console Terminal Background Jobs

R 4.2.3 · ~/

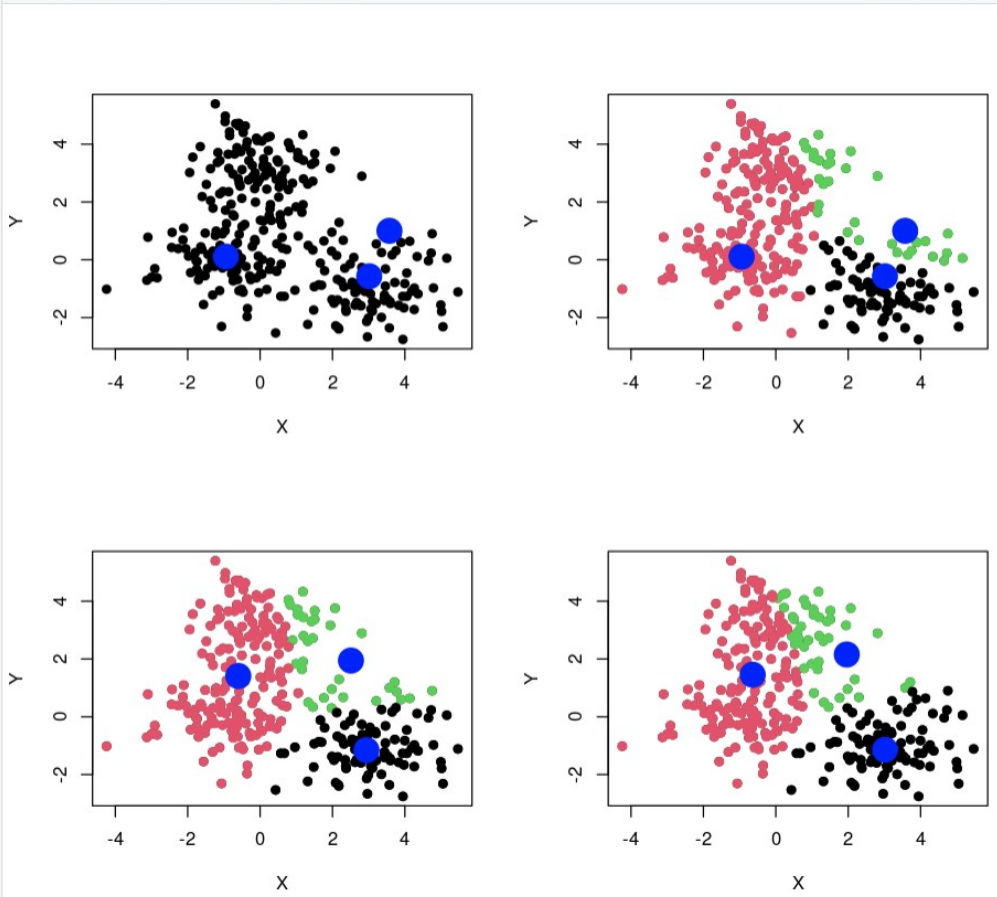
```
+
+ new_centers <- list()
+ for(i in unique(my_labels))
+ {
+   new_centers[[i]] <- colMeans(data[my_labels == i,])
+ }
+ centers <- Reduce("rbind", new_centers)
+ }
>
```

Environment History Files Packages

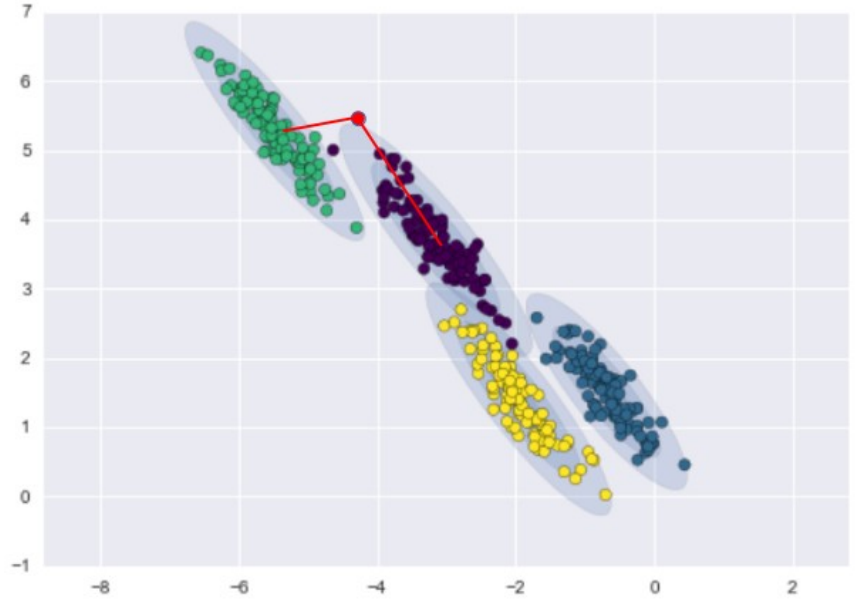
R Global Environment

Plots Connections Help Viewer Presentation

Zoom Export Publish



Gaussian Mixture Model (GMM)



Expectation-Maximization (EM) algorithm:

$$L(X|\mu_k,\sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(X-\mu_k)^2}{2\sigma_k^2}}$$

Once we computed the likelihood $L(X|\mu,\sigma)$, which we assume follows the Normal distribution, of observing the data given the parameters (means and variances), we can compute the posterior probability of data point assignment to each cluster using the Bayes rule.

$$p_k(\mu_k,\sigma_k|X) = \frac{L(X|\mu_k,\sigma_k)\phi_k(\mu_k,\sigma_k)}{\sum_k L(X|\mu_k,\sigma_k)\phi_k(\mu_k,\sigma_k)}$$

Next, differentiating the posterior probability with respect to the parameters, means and variances, i.e. by maximizing the posterior probability of the parameters given the data, we can derive an rule for updating the means and variances of the clusters.

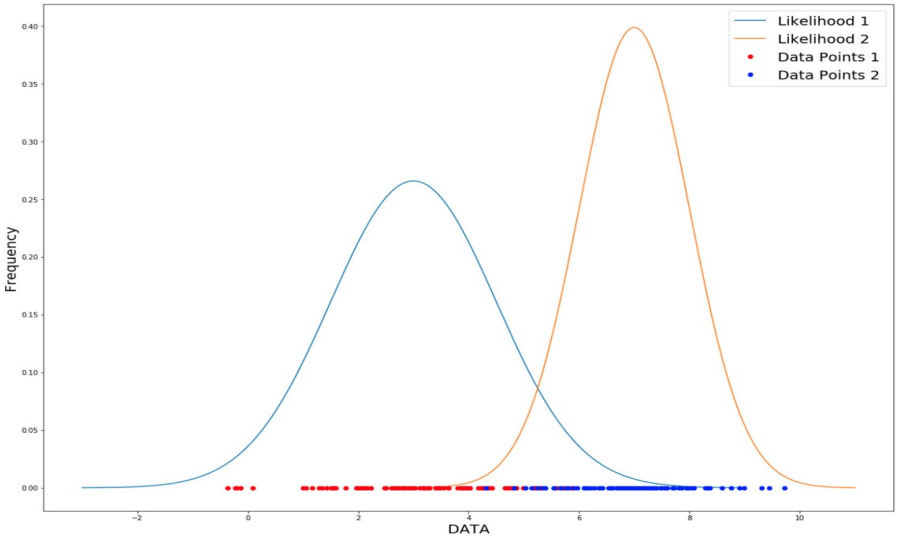
$$\mu_k = \frac{\sum_i X_i p_k(\mu_k,\sigma_k|X_i)}{\sum_i p_k(\mu_k,\sigma_k|X_i)}, \quad \sigma_k^2 = \frac{\sum_i (X_i-\mu_k)^2 p_k(\mu_k,\sigma_k|X_i)}{\sum_i p_k(\mu_k,\sigma_k|X_i)}, \quad \phi_k = \frac{1}{N}\sum_i p_k(\mu_k,\sigma_k|X_i)$$

GMM is a probabilistic clustering

Disadvantages of partitioning clustering (no matter type):

1) Assumes spherical / ellipsoidal symmetry of clusters
2) Number of clusters has to be defined a-priori

GMM from scratch in Python



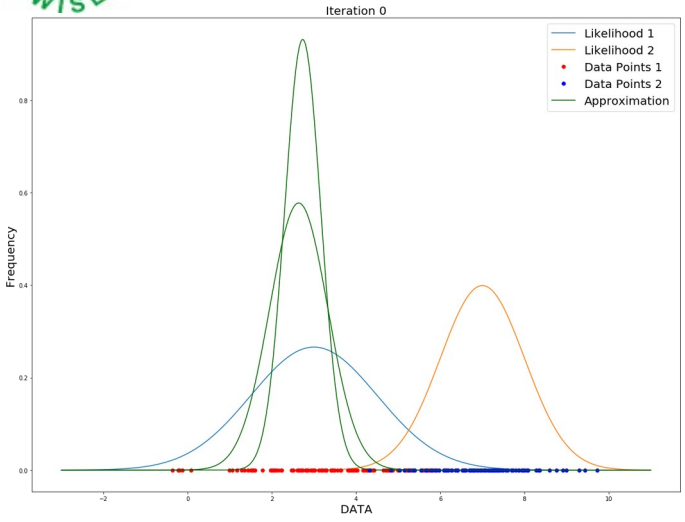

```
import random
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

n = 100
mu1 = 3; sigma1 = 1.5
mu2 = 7; sigma2 = 1

x1 = np.random.normal(mu1, sigma1, n)
x2 = np.random.normal(mu2, sigma2, n)
X = np.array(list(x1) + list(x2))
np.random.shuffle(X)
print("Dataset shape:", X.shape)

x1_approx = np.linspace(mu1 - 4*sigma1, mu1 + 4*sigma1, 10000)
x2_approx = np.linspace(mu2 - 4*sigma2, mu2 + 4*sigma2, 10000)
X_approx = np.array(list(x1_approx) + list(x2_approx))

def pdf(data, mean, sigma):
    return (1 / np.sqrt(2 * np.pi * sigma**2)) * np.exp( - (np.square(data - mean) / (2 * sigma**2)))

plt.figure(figsize = (20,15))
#plt.plot(x1_approx, norm.pdf(x1_approx, mu1, sigma1))
#plt.plot(x2_approx, norm.pdf(x2_approx, mu2, sigma2))
plt.plot(x1_approx, pdf(x1_approx, mu1, sigma1))
plt.plot(x2_approx, pdf(x2_approx, mu2, sigma2))

plt.plot(x1, list(0 for i in range(0, len(list(x1)))), 'ro')
plt.plot(x2, list(0 for i in range(0, len(list(x2)))), 'bo')

plt.ylabel('Frequency', fontsize = 20)
plt.xlabel('DATA', fontsize = 20)
plt.legend(['Likelihood 1', 'Likelihood 2','Data Points 1', 'Data Points 2'], loc = 'upper right', fontsize = 20)
plt.show()
```

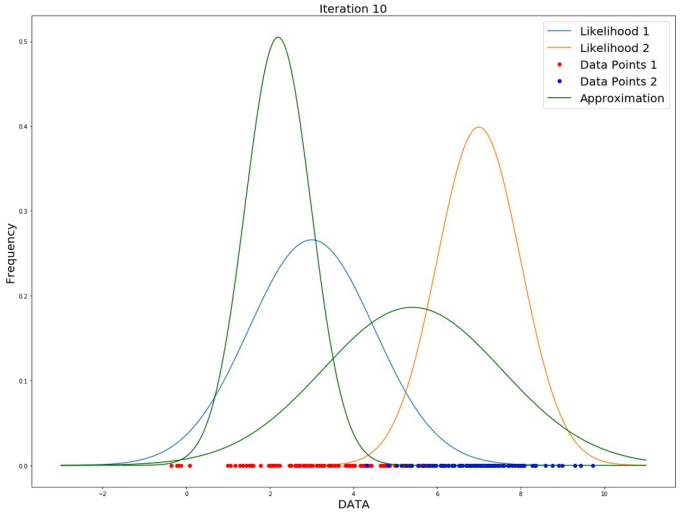
Coding Expectation-Maximization (EM) algorithm from scratch in Python:

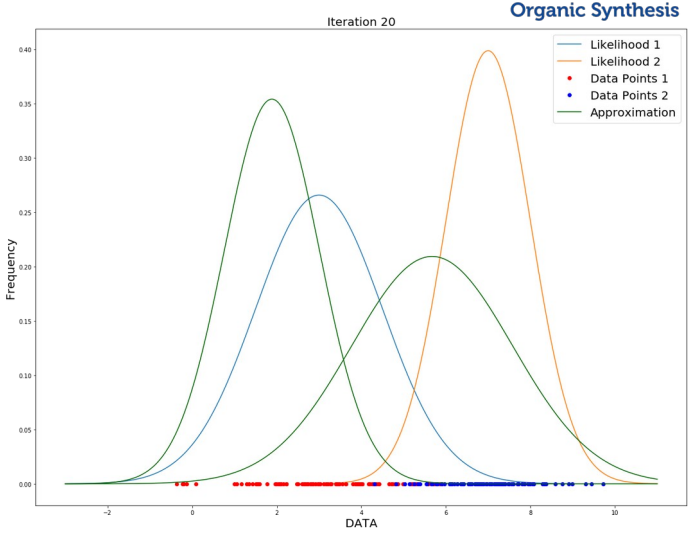
```
k = 2
np.random.seed(123)
means = np.random.choice(X, k)
sigmas = np.random.random_sample(size = k)
priors = np.ones((k)) / k
print(means, sigmas, priors)
```

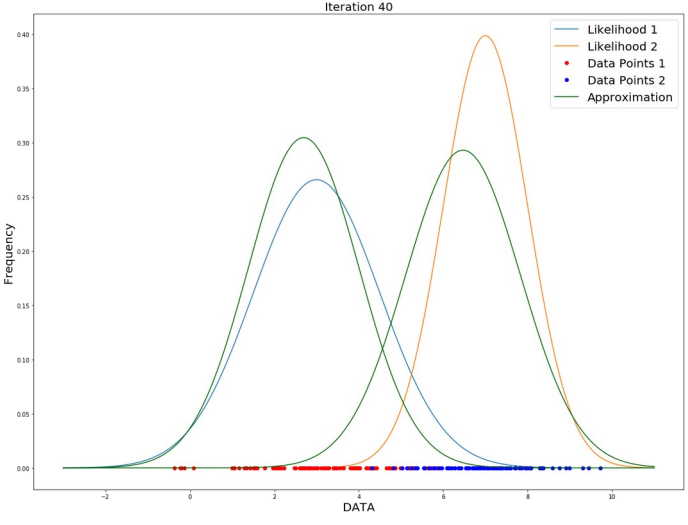
```
from sklearn import cluster, datasets, mixture
from scipy.stats import multivariate_normal

N_iter = 100
bins = np.linspace(np.min(X_approx),np.max(X_approx), 10000)
for step in range(N_iter + 1):
    if step % 10 == 0:

        plt.figure(figsize = (20, 15))
        plt.ylabel('Frequency', fontsize = 20)
        plt.xlabel('DATA', fontsize = 20)
        plt.title("Iteration {}".format(step), fontsize = 20)

        plt.plot(x1_approx, pdf(x1_approx, mu1, sigma1))
        plt.plot(x2_approx, pdf(x2_approx, mu2, sigma2))

        plt.plot(x1, list(0 for i in range(0, len(list(x1)))), 'ro')
        plt.plot(x2, list(0 for i in range(0, len(list(x2)))), 'bo')


        print(means[0], means[1])
        print(sigmas[0], sigmas[1])

        plt.plot(bins, pdf(bins, means[0], sigmas[0]), color = 'darkgreen')
        plt.plot(bins, pdf(bins, means[1], sigmas[1]), color = 'darkgreen')

        plt.legend(['Likelihood 1', 'Likelihood 2','Data Points 1', 'Data Points 2', 'Approximation'],
                   loc = 'upper right', fontsize = 20)
        plt.show()

    # Expectation step
    likelihood = []
    for j in range(k):
        likelihood.append(pdf(X, means[j], sigmas[j]))

    # Maximization step
    posterior = []
    for j in range(k):
        posterior.append((likelihood[j] * priors[j]) / (np.sum([likelihood[i] * priors[i] for i in range(k)],
                                                               axis = 0)))
        means[j] = np.sum(posterior[j] * X) / np.sum(posterior[j])
        sigmas[j] = np.sqrt(np.sum(posterior[j] * np.square(X - means[j])) / np.sum(posterior[j]))
        priors[j] = np.mean(priors[j])
```

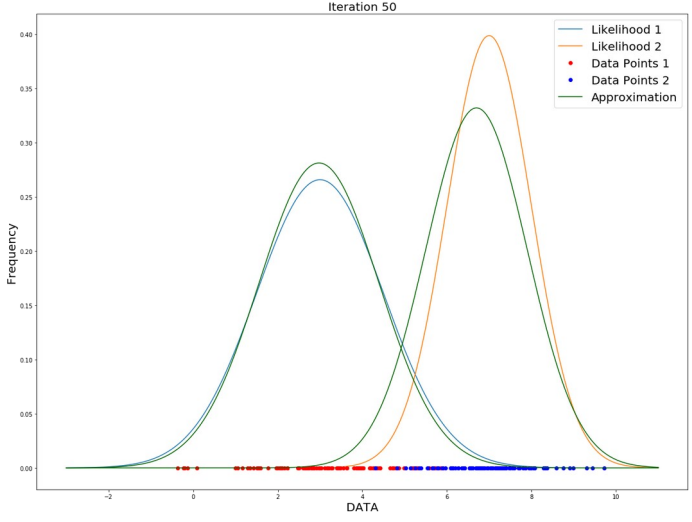
GMM: iteration by iteration

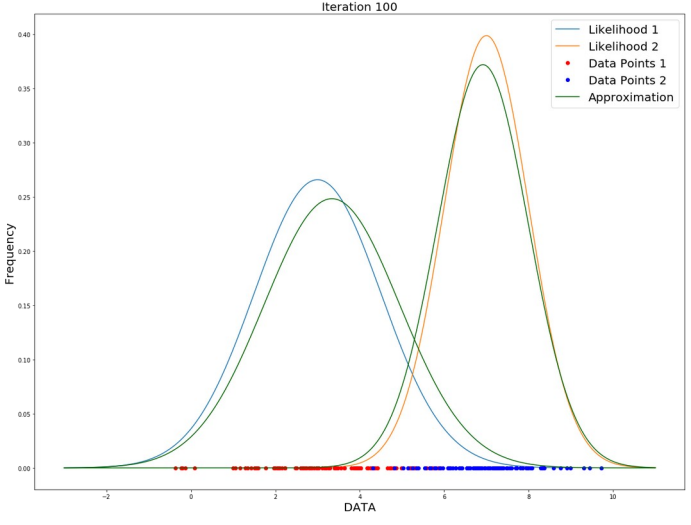












Session summary

Take home messages of the session:

1) Hierachical clustering involves merging pairs of most similar points and building a dendrogram of their relationships, it is sensitive to noise in the data

2) Partitioning clustering implies dividing manifold of data points into groups of most similar points

3) K-means and Gaussian Mixture Model (GMM) assume spherical / ellipsoidal symmetry of clusters and demand the number of clusters to be known a-priori

Acknowledgments: LIOS + TARGETWISE

Latvian Institute of Organic Synthesis

2027 National Development Plan

Funded by the European Union