CLUSTERING

E. Fersini

INTRODUCTION

- K-Means is a flat clustering technique that produces *k* partitions (clusters).
- During this lab we will demonstrate how to perform k-means clustering on a customer dataset.

```
customer= read.csv('customer.csv', header=TRUE)
```

```
customer = scale(customer[,-1])
```

K-MEANS

- The k-means algorithm:
  - The goal is to partition n objects into *k* clusters, where each object belongs to the cluster with the nearest mean.
  - The objective of the algorithm is to minimize the **within-cluster sum of squares** (**WCSS**). Assuming x is the given set of observations, S = {*S*1 , *S*2,..., *Sk* } denotes *k* partitions, and μi is the mean of *Si* , then we can formulate the WCSS function as follows:

$$\mathrm{f} = \sum_{i=1}^{k} \sum_{x \in S_i} \left\| x - \mu_i \right\|^2$$

K-MEANS

- The k-means process is as follows:
  1. Specify the number of *k* clusters.
  2. Randomly create k partitions.
  3. Calculate the center of the partitions.
  4. Associate objects closest to the cluster center.
  5. Repeat steps 2, 3, and 4 until the WCSS changes very.

- The k-means algorithm in practice:

```
set.seed(22)
fit = kmeans(customer, 4)
fit
```

SILHOUETTE

- During this lab we will demonstrate how to validate clusters.
  - To validate a clustering method, two criteria can be usually used: intercluster distance and intracluster distance.
    - The higher the intercluster distance, the better it is
    - The lower the intracluster distance, the better it is.

$$\text{Silhouette}(x) = \frac{b(x) - a(x)}{\max\left(\left[b(x), a(x)\right]\right)}$$

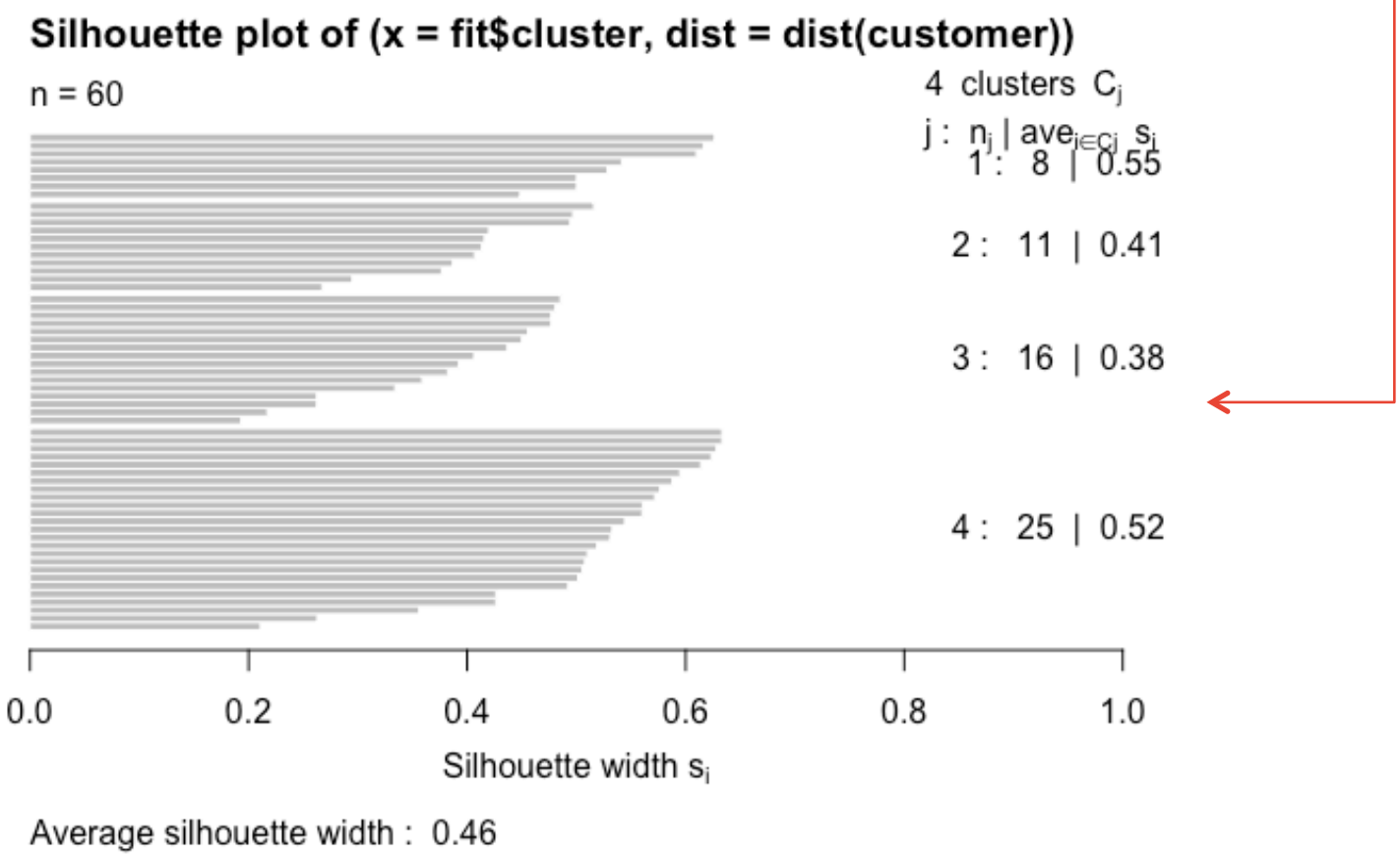
a(x) is the average distance between x and all other points within the cluster

b(x) is the minimum of the average distances between x and the points in the other clusters

```
library(cluster)
kms = silhouette(fit$cluster,dist(customer))
plot(kms)
```

SILHOUETTE

mean similarity of each own cluster minus the mean similarity of the next similar cluster

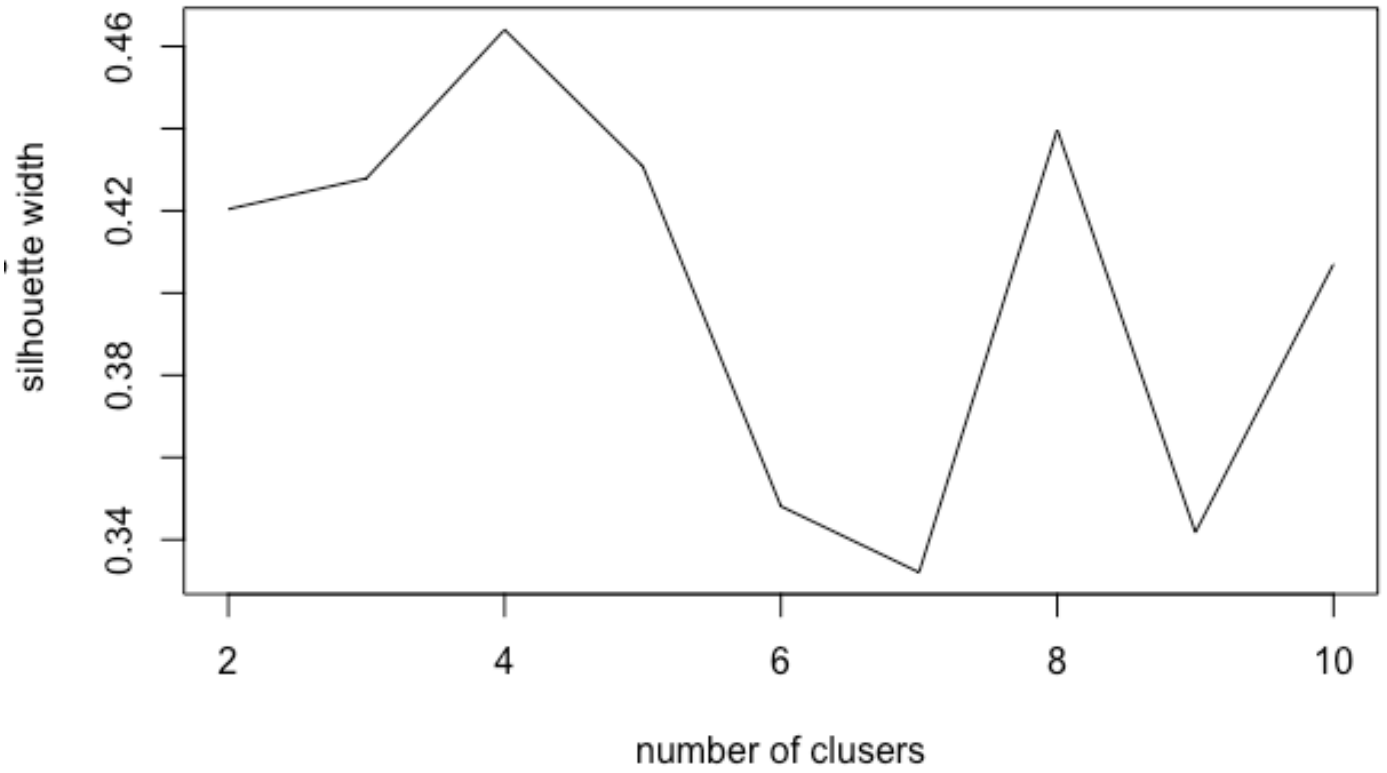


OPTIMAL NUMBER OF CLUSTERS

- While k-means clustering is fast and easy to use, it requires *k* to be the input at the beginning.
- Therefore, we can use the silhouette index to determine the optimum number of clusters for k-means.

```
library(fpc)
nk = 2:10
set.seed(22)
SW = sapply(nk, function(k) {cluster.stats(dist(customer), kmeans(customer,
centers=k)$cluster)$avg.silwidth})
SW
plot(nk, SW, type="l", xlab="number of clusers", ylab="average silhouette")
```

OPTIMAL NUMBER OF CLUSTERS



DISSIMILARITY MATRIX

- A dissimilarity matrix can be used as a measurement for the quality of a cluster. To visualize the matrix, we can use a heat map on a distance matrix.
  - Within the plot, entries with low dissimilarity (or high similarity) are plotted darker, which is helpful to identify hidden structures in the data.

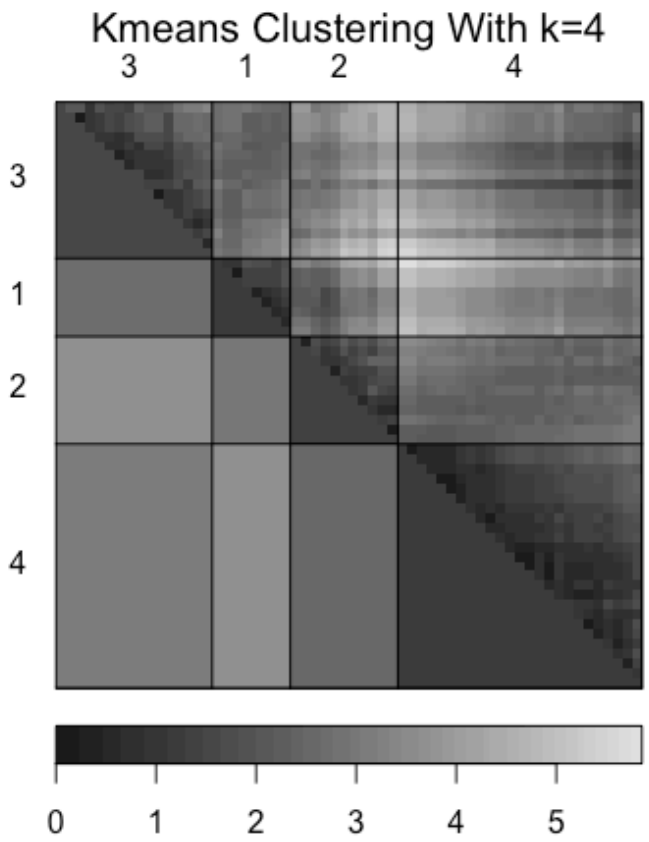
1. install and load the "seriation" package:

```
> install.packages("seriation")
> library(seriation)
```

2. Use dissplot() to visualize the dissimilarity matrix:

```
> dissplot(dist(customer), labels=fit$cluster,options=list(main="Kmeans
Clustering With k=4"))
```

DISSIMILARITY MATRIX



clusters similar to each other are plotted darker, and dissimilar combinations are plotted lighter.

ASSIGNMENT

- Make your cluster analysis on the iris dataset
  - Take care about the randomness of the KMEANS
    - Look at the parameter **nstart**
- Evaluate your results with respect to the ground truth
  - cluster.evaluation(G, S)
    - true_cluster <- c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
    - new_cluster <- c(2, 1, 2, 3, 3, 2, 2, 1, 3, 3, 3, 3)
    - cluster.evaluation(true_cluster, new_cluster)
- Plot your best cluster assignment (using PCA)
  - library(fviz_cluster)
  - fviz_cluster(k1, iris[, -5], frame.type = "norm")