Unsupervised Learning

Clustering

Maarten Cruyff

Program

Afternoon session

Clustering

Clustering is an unsupervised statistical learning method that aims at finding homogeneous groups of observations. The two basic clustering methods are k-means and hierarchical clustering. In k-means clustering the user specifies the number of clusters, and the algorithm optimally divides the observations. This technique is helpful in for example marketing to identifying different groups of customers. Hierarchical clustering is useful for making taxonomies like the subdivision of organisms in species, subspecies and varieties.

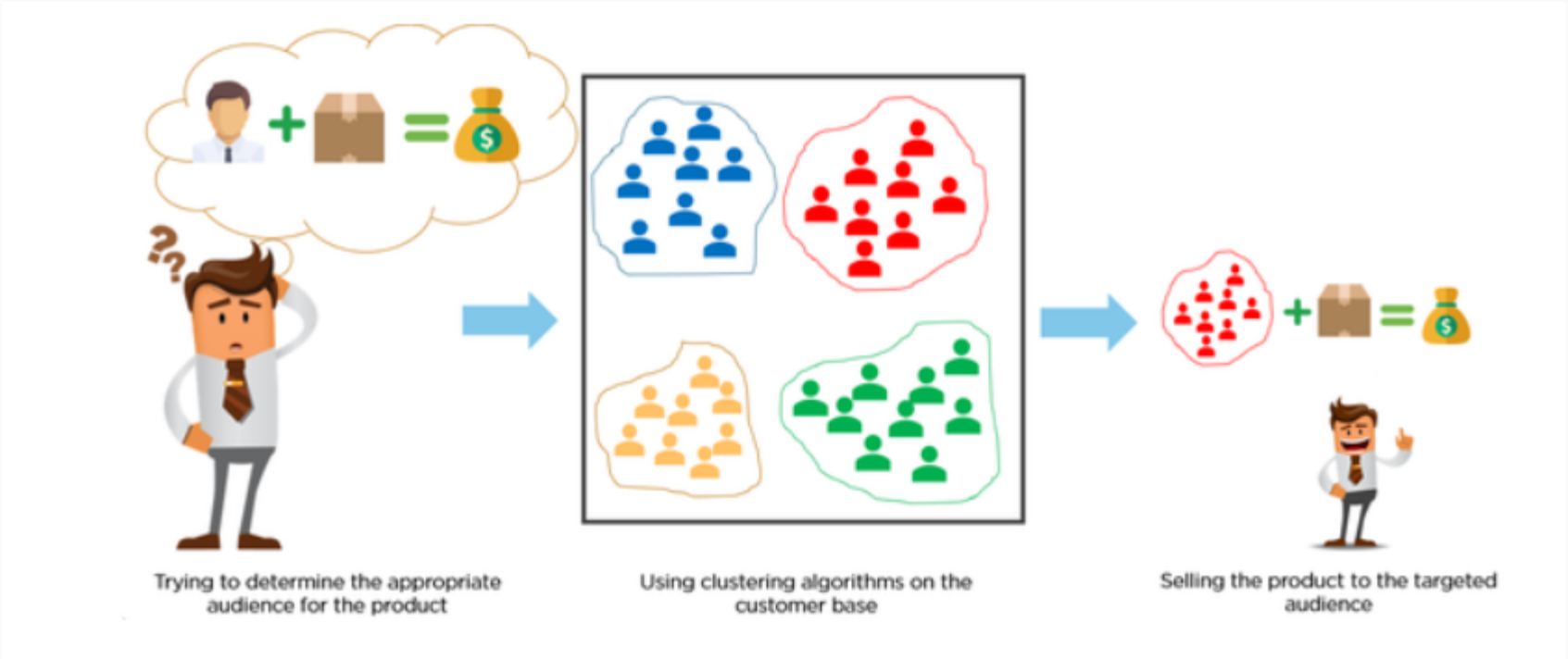
Course materials

- Lecture sheets: R and RStudio
- R lab
- R Markdown lab template

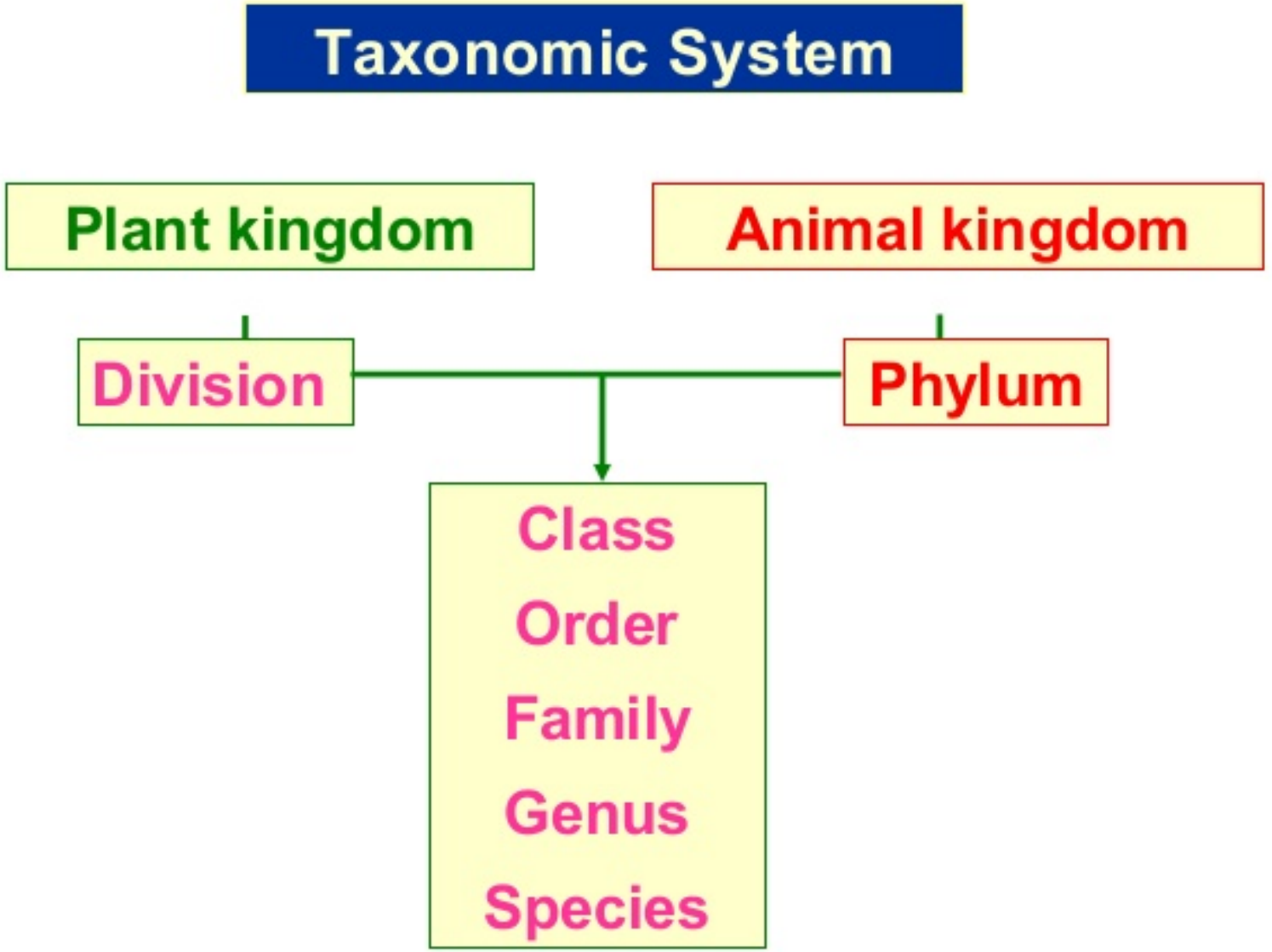
Content

1. K-means clustering
2. Hierarchical clustering

Customer segregation with k-means



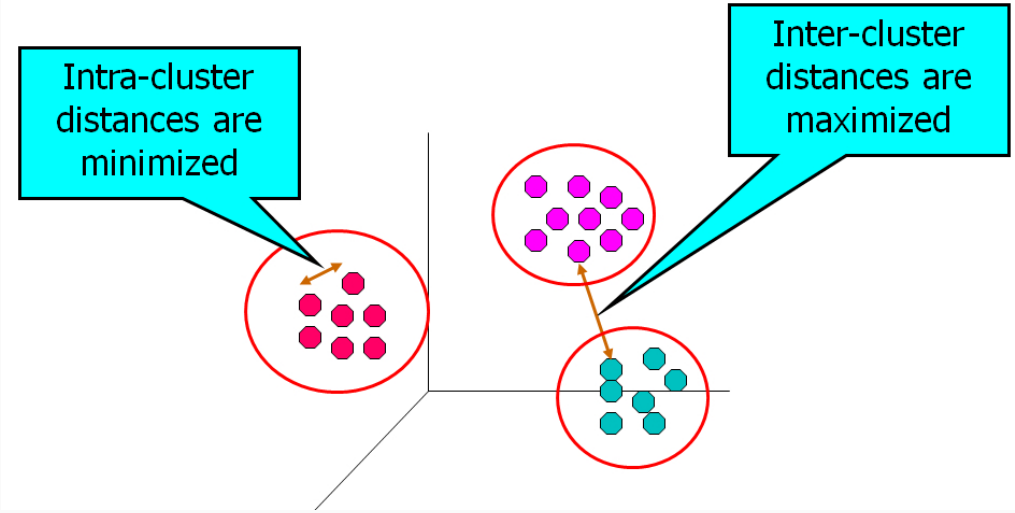
Taxonomis with hierarchical clustering



K-means clustering

K-means algorithm

1. Specify the total number of clusters K
2. Randomly assign each observation to a cluster
3. Compute cluster centroids
4. Reassign observations to cluster with closest centroid
5. Repeat 3 and 4 until convergence



Issues

Determination optimal number of clusters

- elbow criterion for within-cluster SS

Solution is local minimum of within-cluster SS

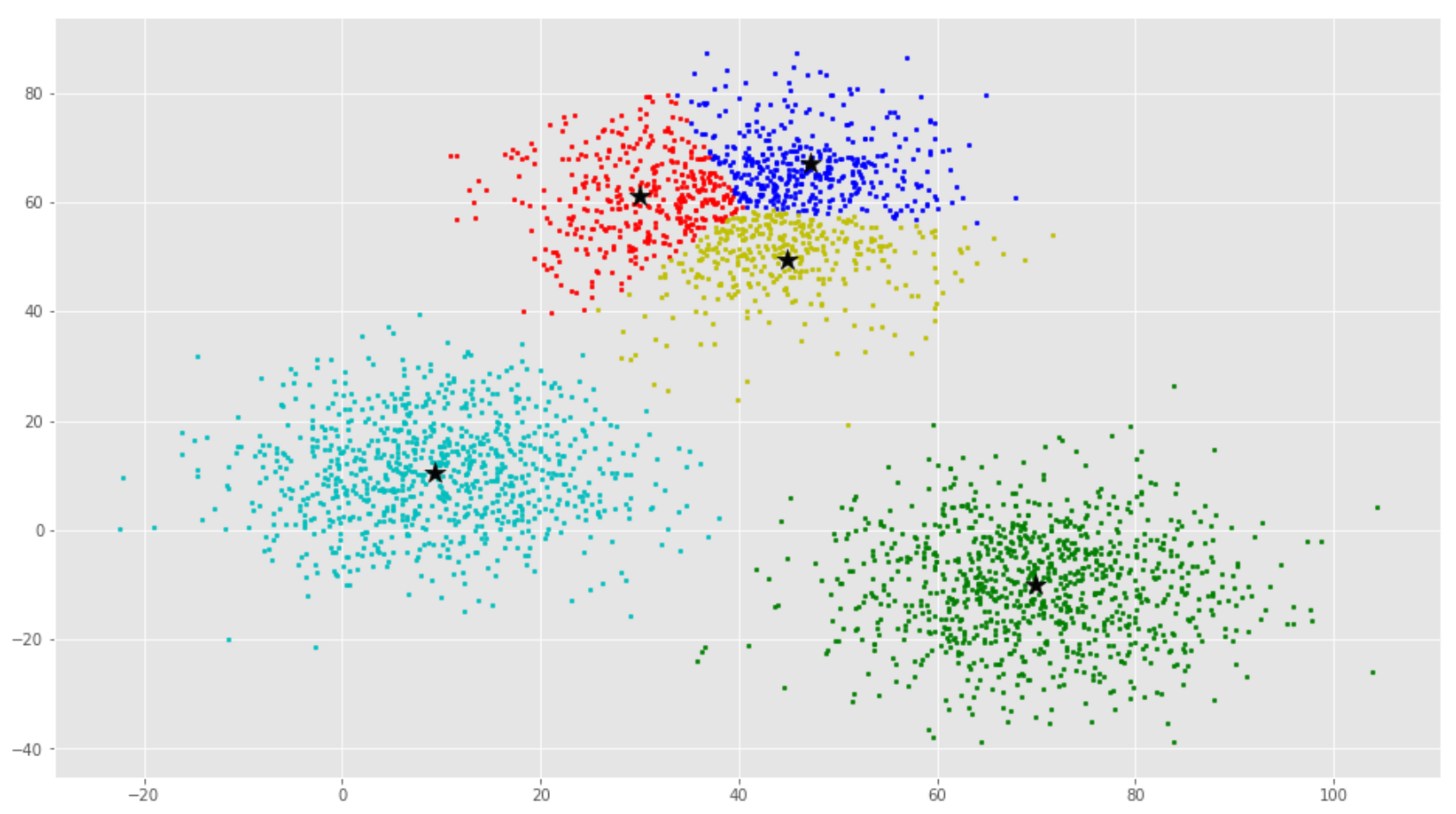
- try out multiple starting values, e.g.

Data

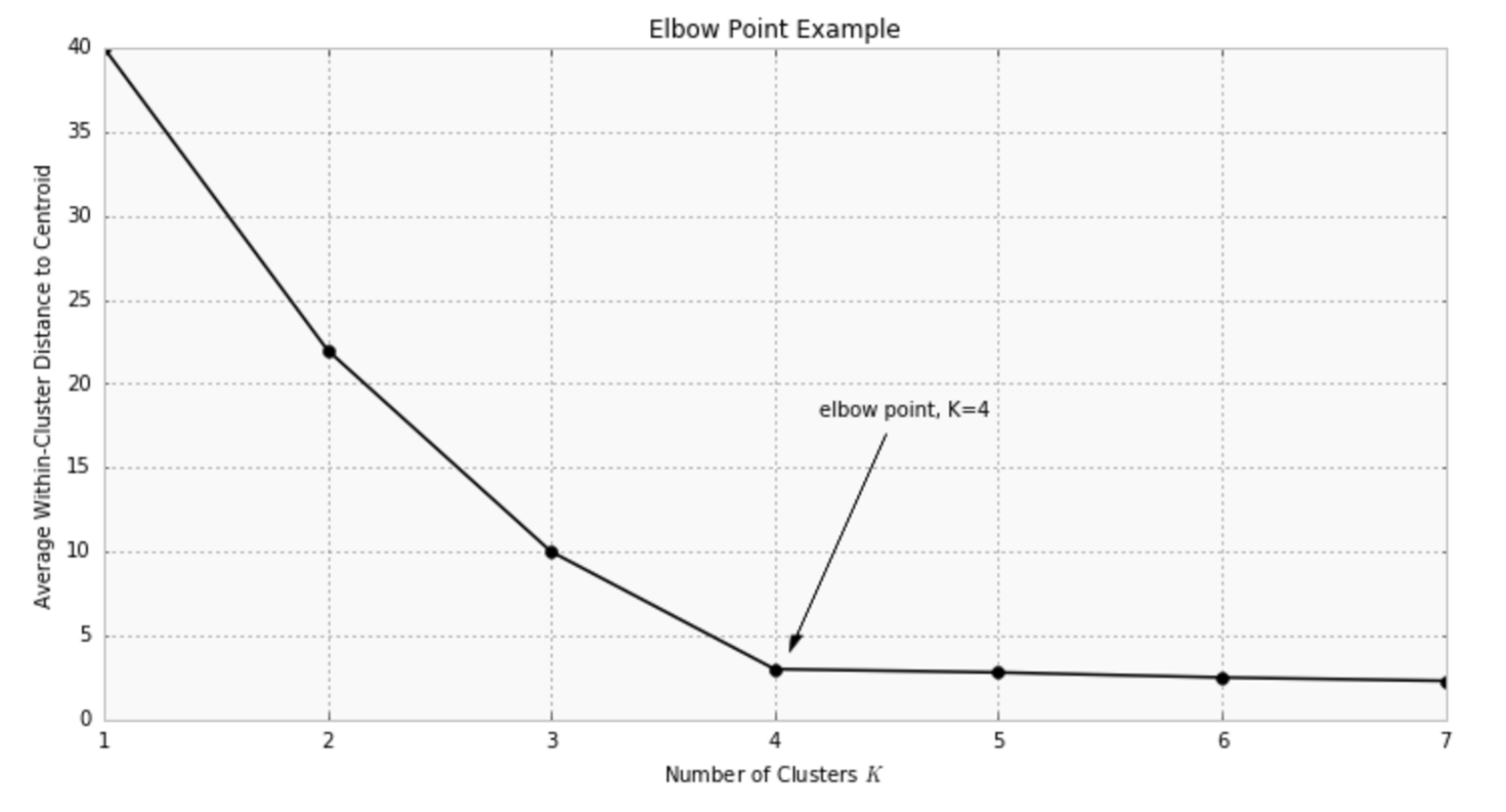
- standardization of features (as in PCA)
- clustering on principal components often works better

Optimal number of clusters

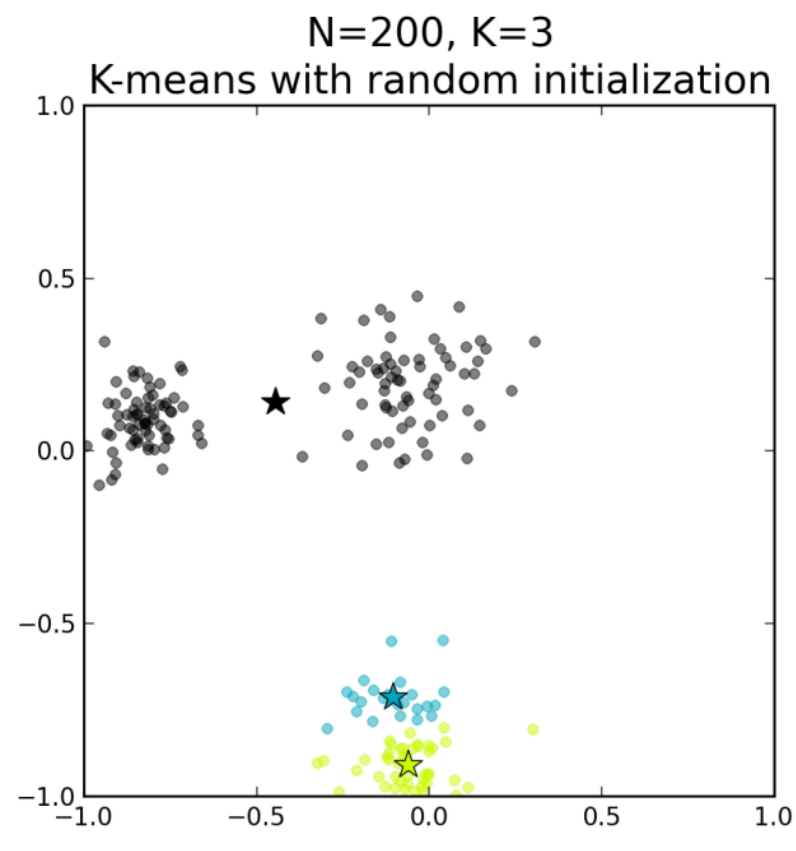
Would a solution with 3 clusters been better?

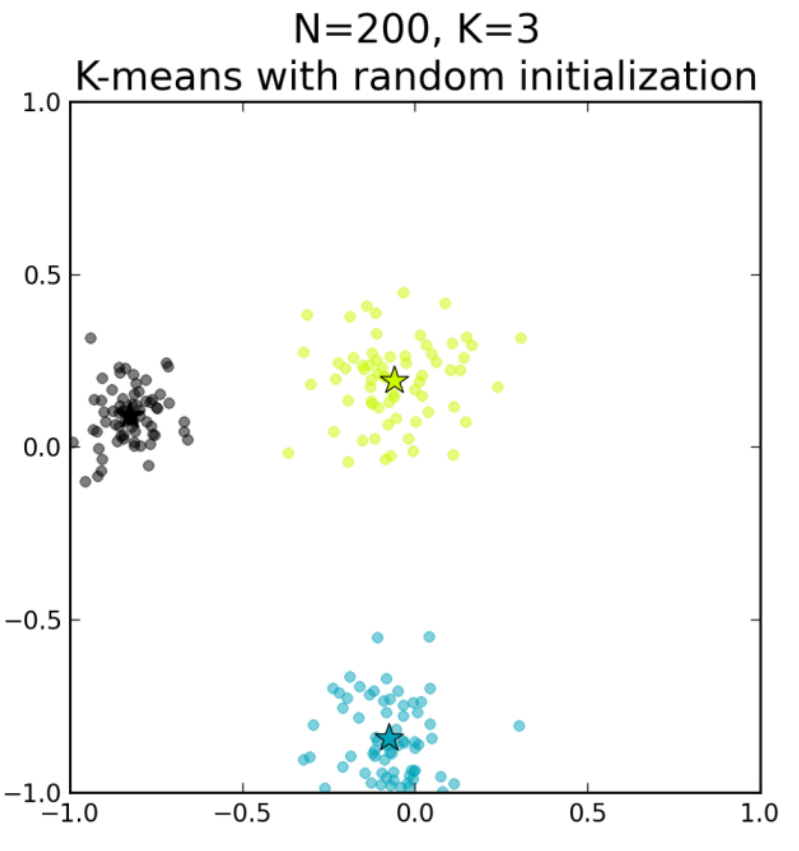


Elbow criterion

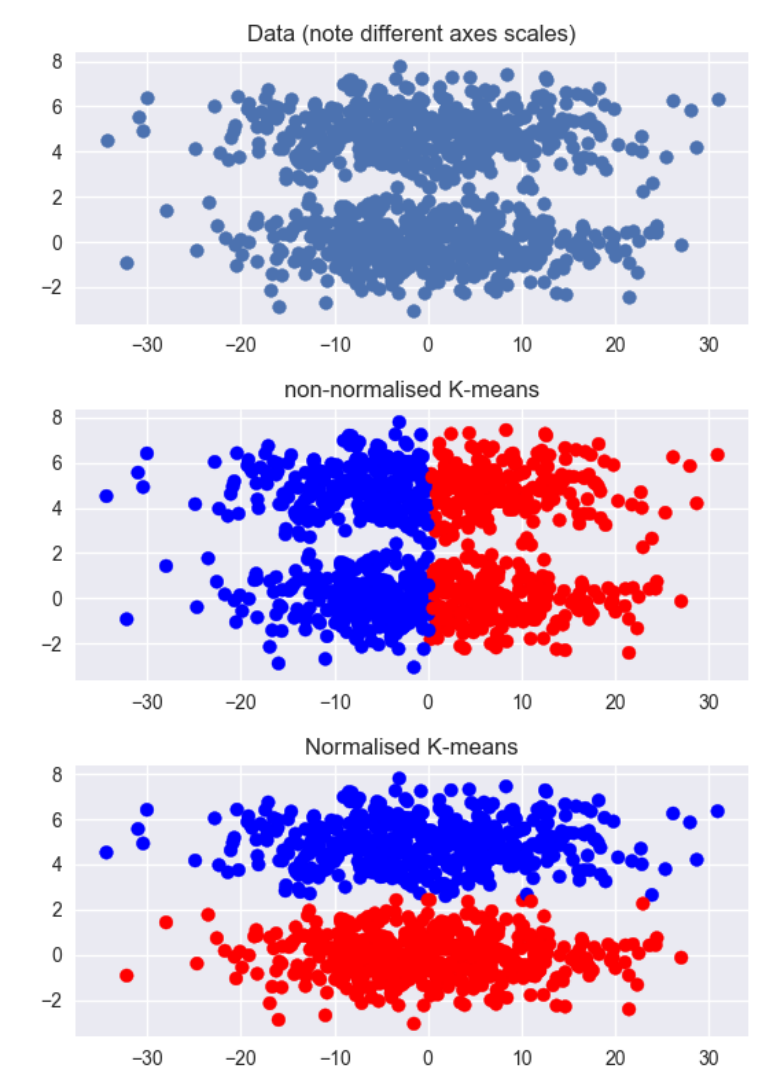


Local minimum





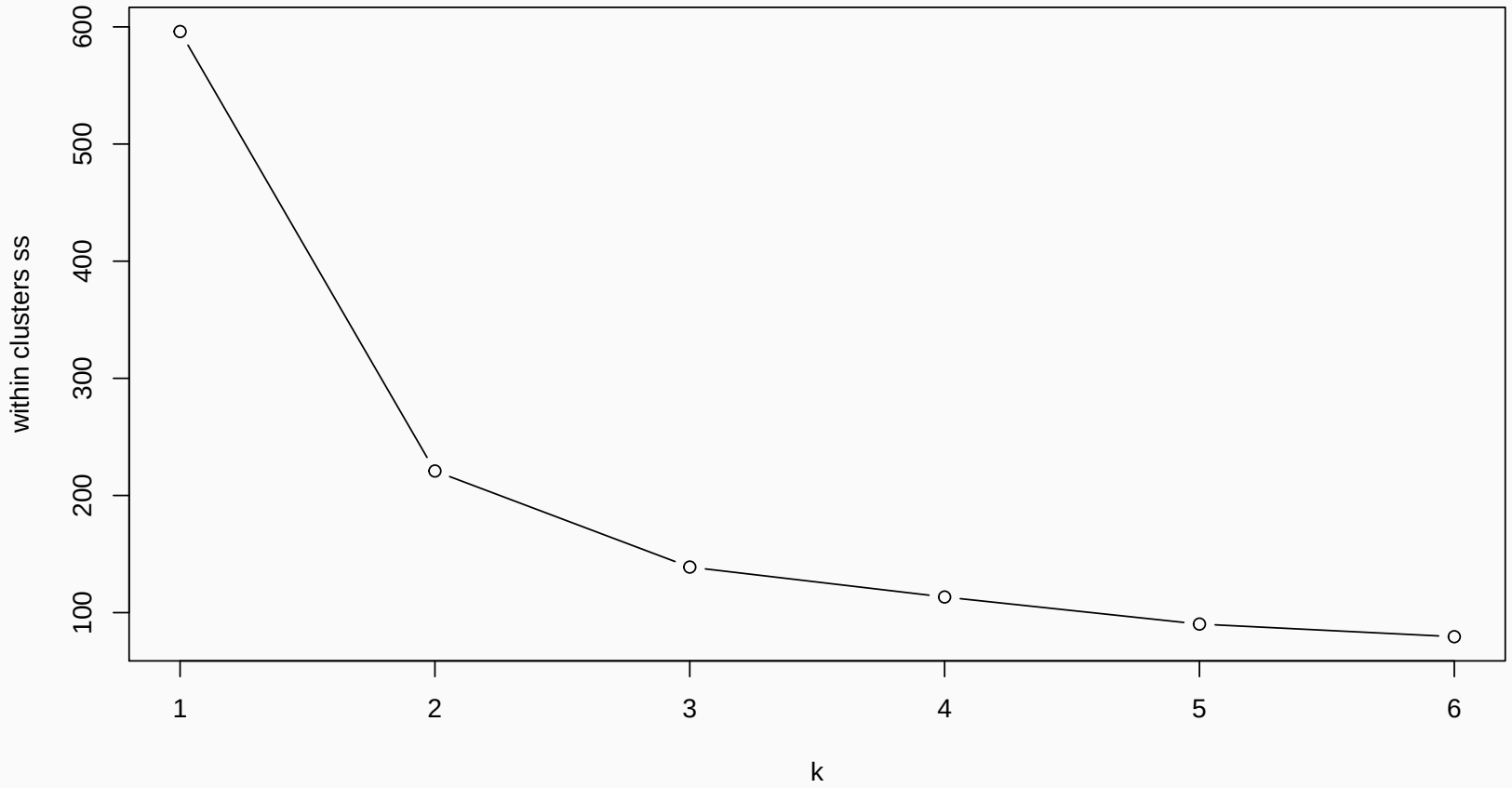
Scaling


Data (note different axes scales)
8
6
4
2
0
−2
−30
−20
−10
0
10
20
30
non-normalised K-means
8
6
4
2
0
−2
−30
−20
−10
0
10
20
30
Normalised K-means
8
6
4
2
0
−2
−30
−20
−10
0
10
20
30

K-means on iris data

Scree plot original iris data (scaled)

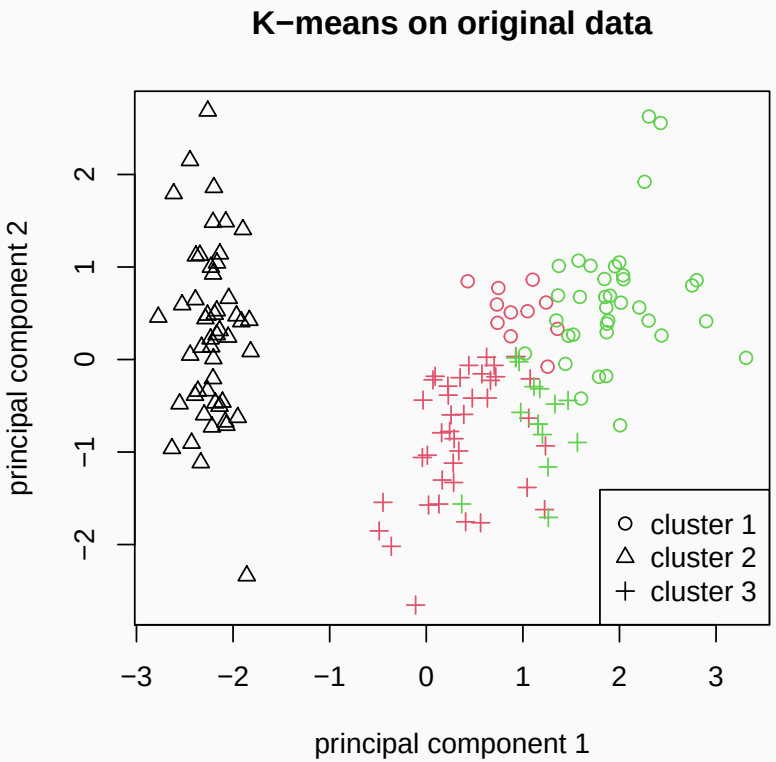
- 6 clusters
- 10 random starts

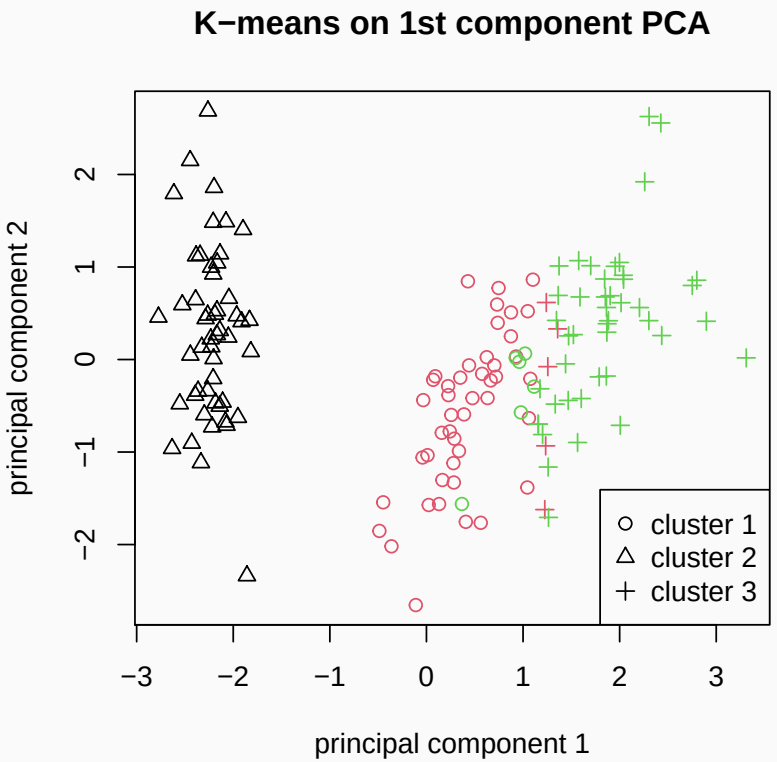


```
K-means clustering with 3 clusters of sizes 47, 50, 53

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   1.13217737  0.08812645    0.9928284   1.0141287
2  -1.01119138  0.85041372   -1.3006301  -1.2507035
3  -0.05005221 -0.88042696    0.3465767   0.2805873

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3
 [75] 3 1 1 1 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1
[112] 1 1 3 3 1 1 1 1 3 1 3 1 3 1 1 3 1 1 1 1 1 1 3 3 1 1 1 3 1 1 1 3 1 1 1 3 1
[149] 1 3

Within cluster sum of squares by cluster:
[1] 47.45019 47.35062 44.08754
 (between_SS / total_SS =  76.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Original data vs PC's

Sometimes clustering on less dimensions work best





Confusion matrices

- K-means on scaled `iris` data

```
            1  2  3
setosa      0 50  0
versicolor 11  0 39
virginica  36  0 14
```

- K-means on 1st principal component

```
            3  1  2
setosa      0  0 50
versicolor  5 45  0
virginica  44  6  0
```

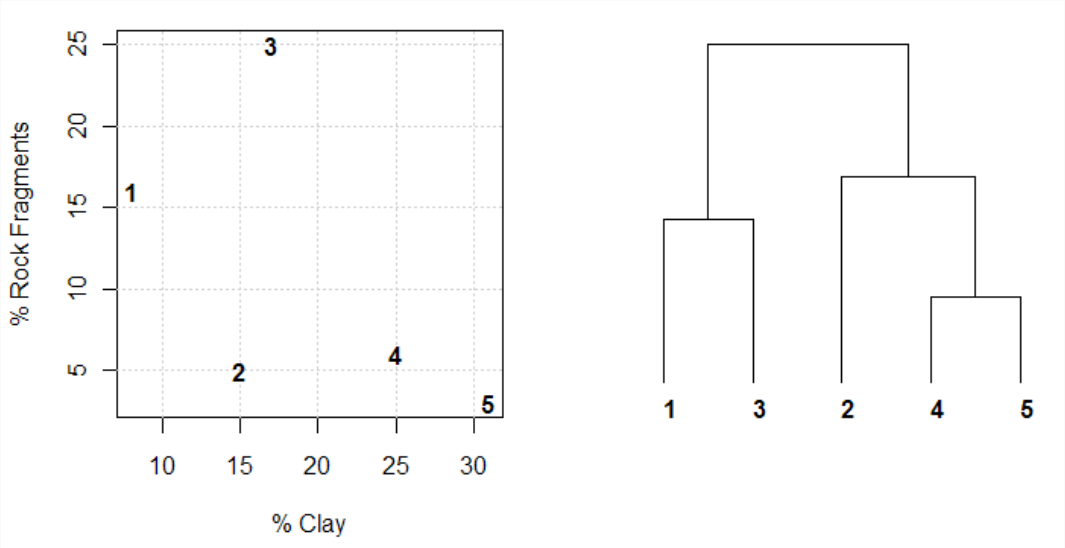
K-means in R

```r
k_fit <- kmeans(x, centers, nstart = 1)  # scale(x) for standardization

print(k_fit)   # print a summary

fitted(k_fit)  # centroids for each case

k_fit$withinsss # total within-cluster sum of squares
```

Hierarchical clustering

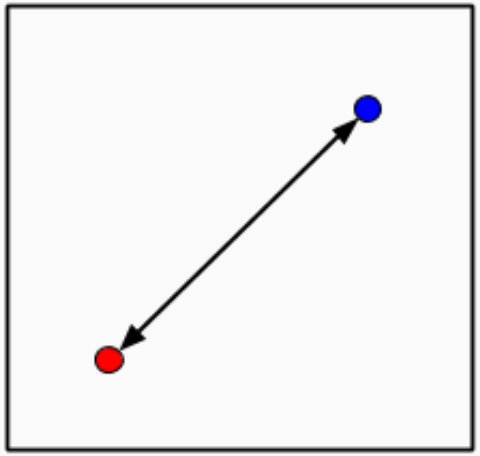
Algorithm (bottom-up and greedy)

0. Standardize all features
1. Treat each observation as a cluster
2. Compute *distances* between all $\binom{n}{2}$ cluster pairs
3. Link pair with smallest *distance* in new cluster
4. Repeat 2-3 until 2 clusters left
5. Plot dendogram (and optionally the clusters)

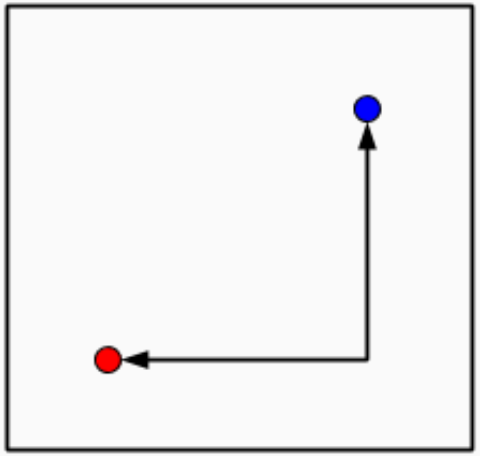


Compute distances

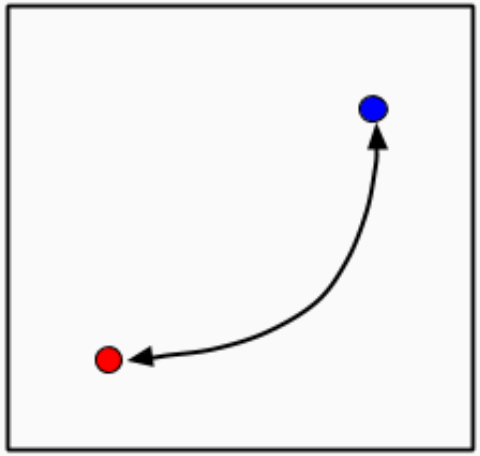
Euclidean



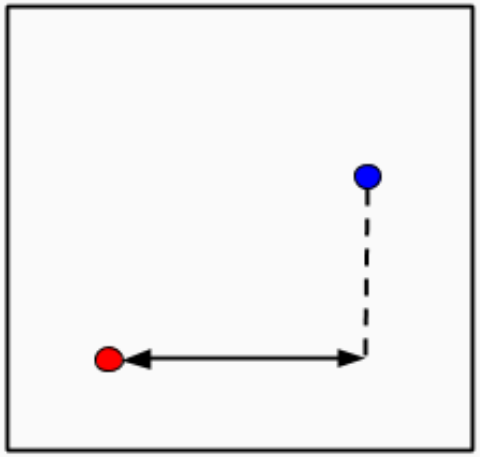
Manhattan



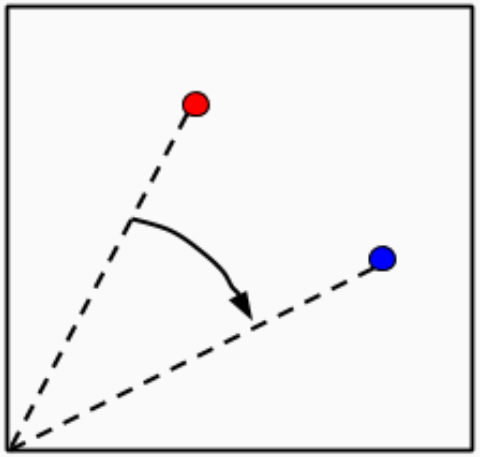
Minkowski



Chebychev



Cosine Similarity



Hamming

Function dist()

```
dist(scale(iris[1:3, -(5)]), "euclidean")
```

```
         1        2
2 2.568364
3 3.350831 2.485043
```

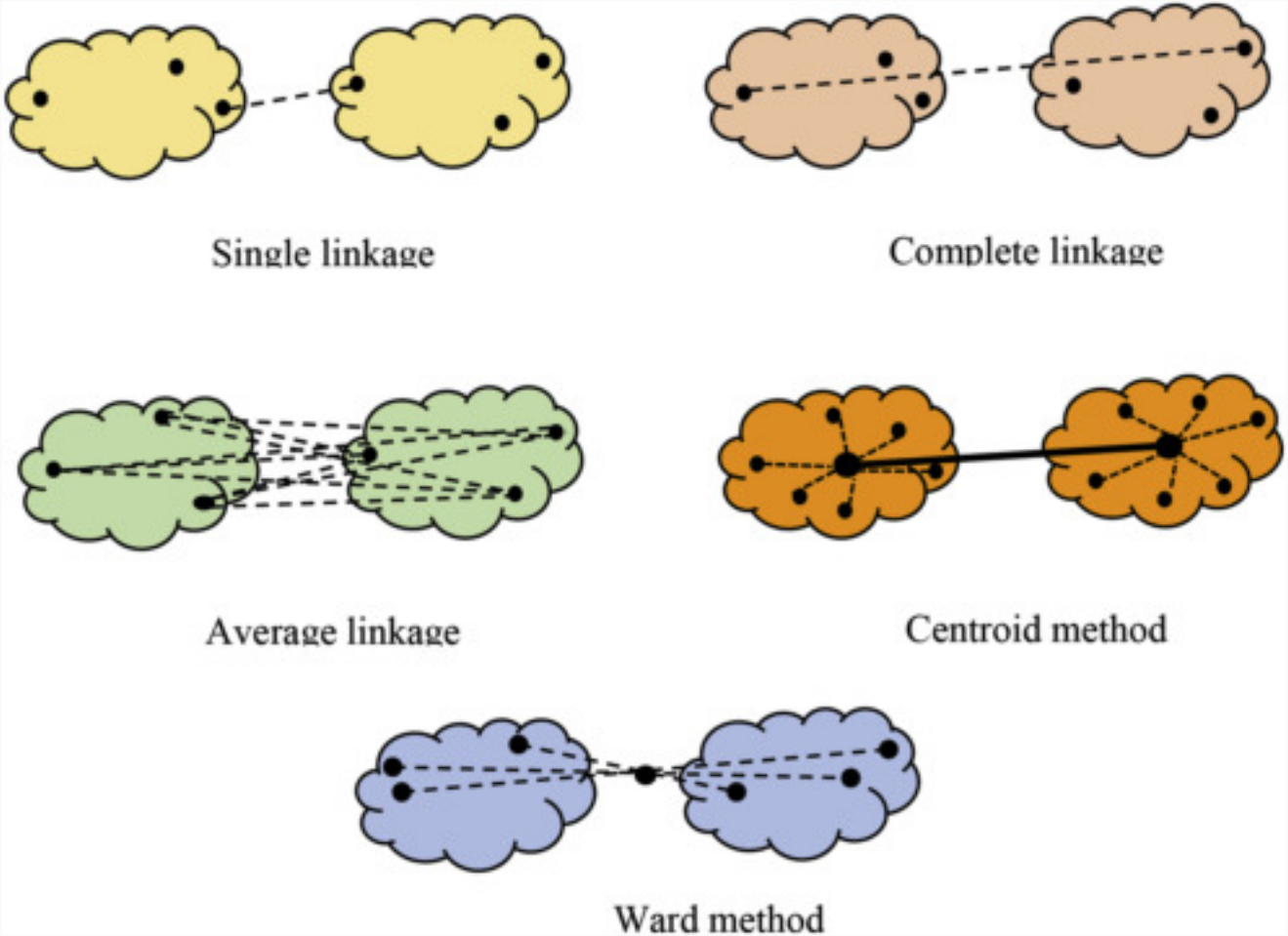
```
dist(scale(iris[1:3, -(5)]), "manhattan")
```

```
         1        2
2 3.982398
3 6.565507 4.702360
```

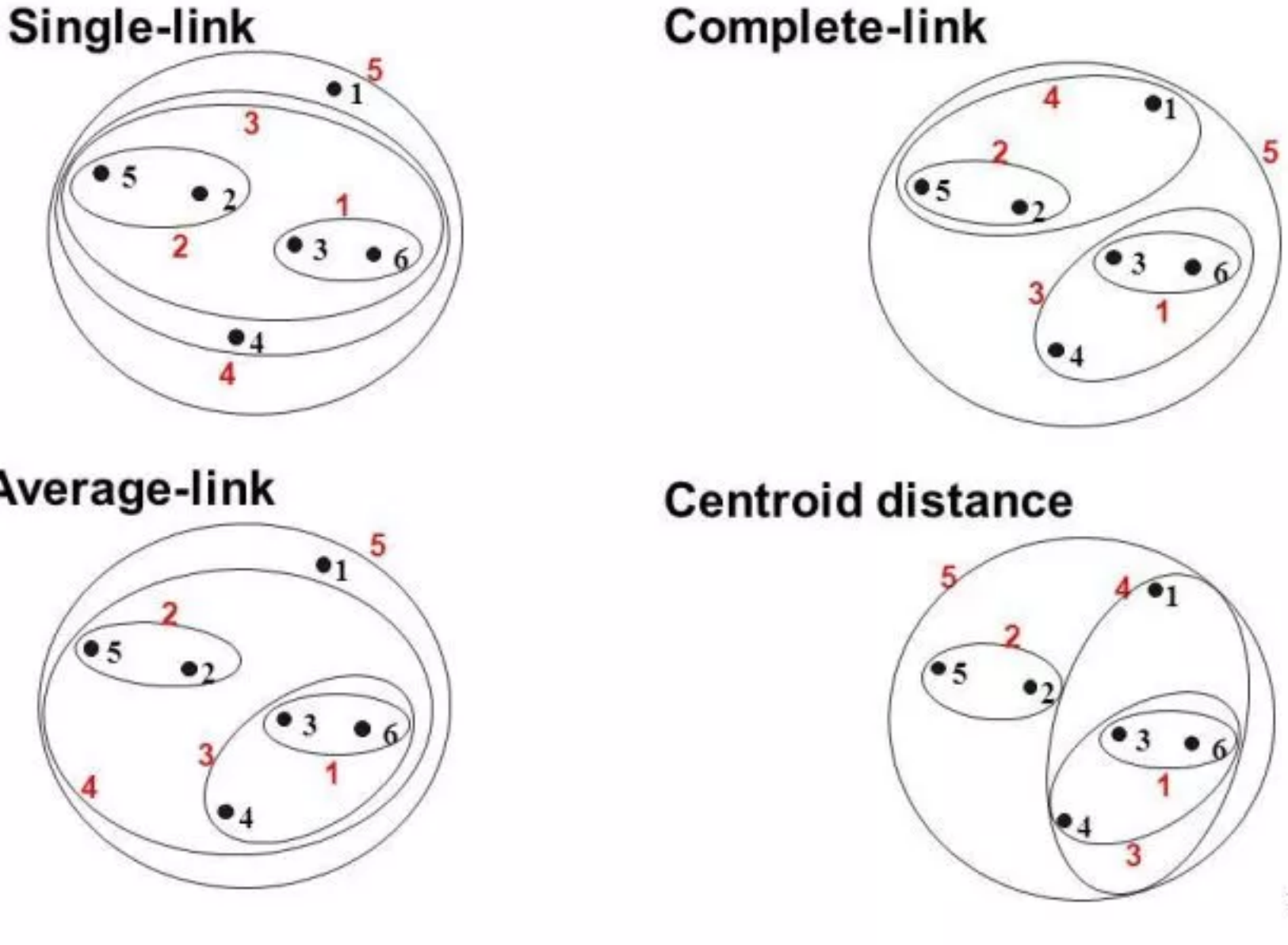
```
dist(scale(iris[1:3, -(5)]), "maximum")
```

```
         1        2
2 1.986799
3 2.000000 1.732051
```

Linkage methods



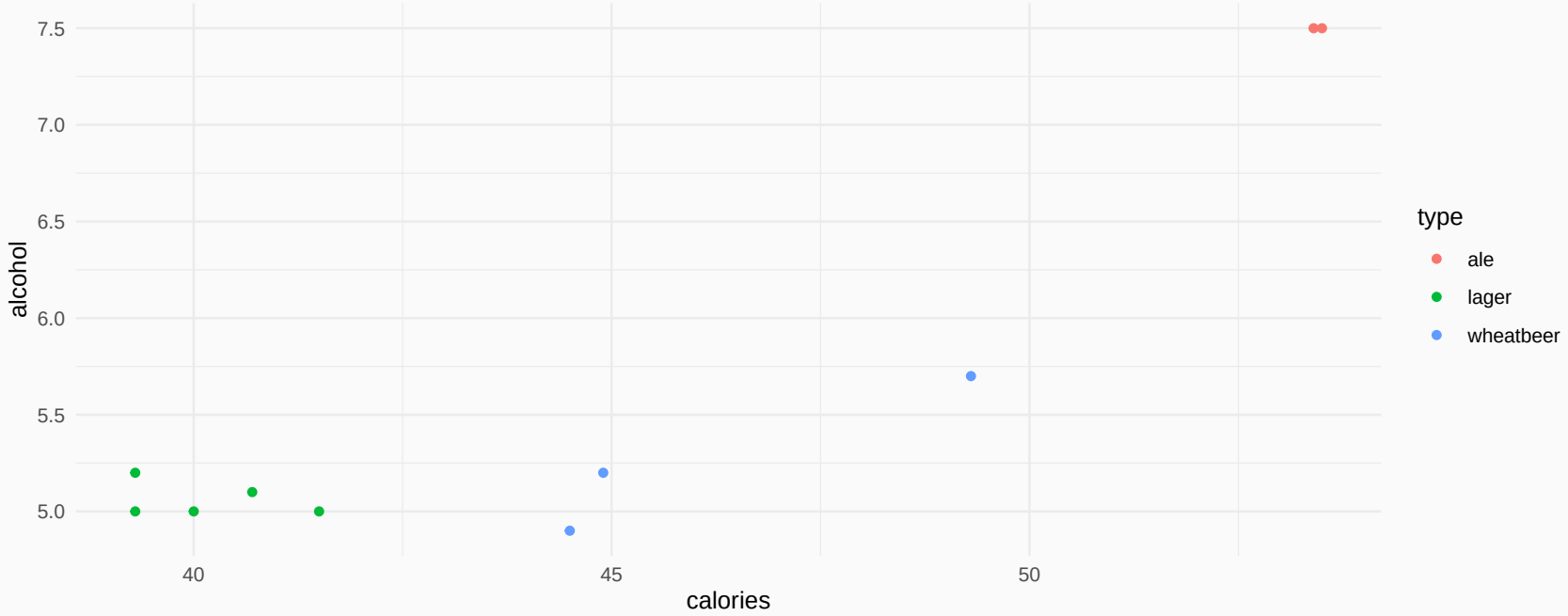
Effect linkage



Beer example

Cluster 10 beers types on calories and alcohol

- 3 types (ale, lager and wheatbeer)



Function `hclust()`

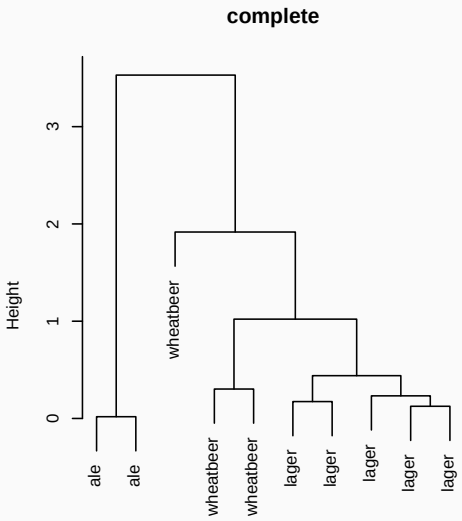
- compute clusters

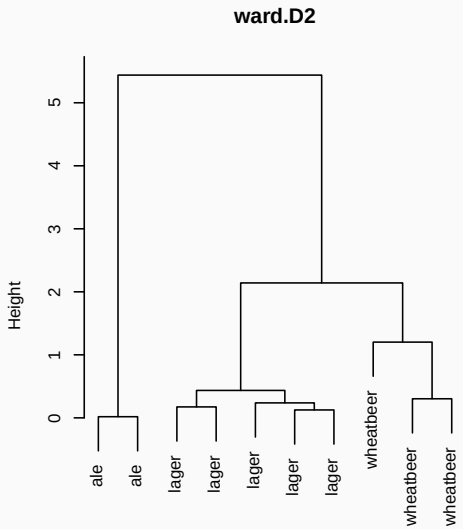
```
hc <- hclust(x = dist(scale(<data>),
                      method = "euclidean"),
             method = c("complete",
                        "ward.D2",
                        "single",
                        "average",
                        "centroid")
```

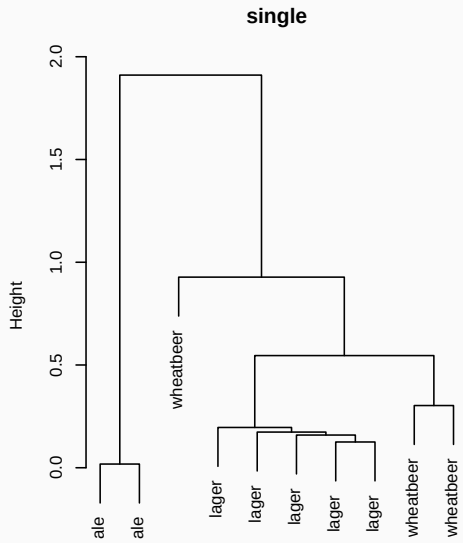
- plot clusters

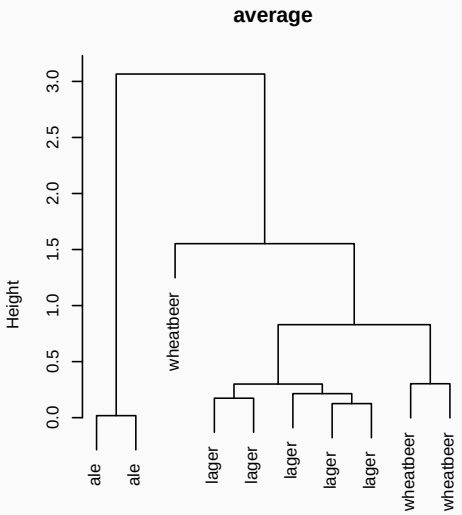
```
plot(hc, labels = <names observations>) # default is row names

rect.hclust(hc, k)                     # plot rectangles
```

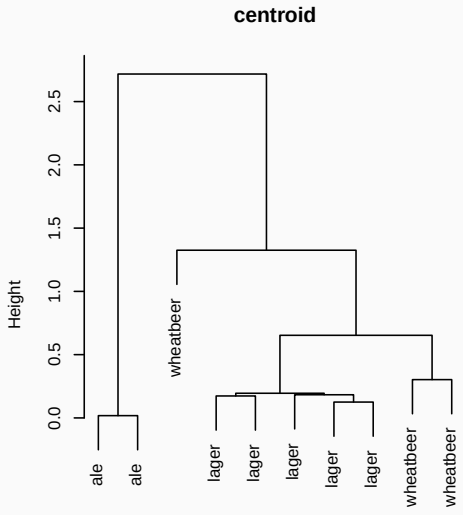
Comparisons











Number of clusters?