Statistical Computing with R: Masters in Data Sciences 503 (S26) Second Batch, SMS, TU, 2023

Shital Bhandary

Associate Professor

Statistics/Bio-statistics, Demography and Public Health Informatics

Patan Academy of Health Sciences, Lalitpur, Nepal

Faculty, Data Analysis and Decision Modeling, MBA, Pokhara University, Nepal

Faculty, FAIMER Fellowship in Health Professions Education, India/USA.

Review Preview: Unsupervised models

- Multidimensional scaling
- Clustering
  - K-means clustering
  - Hierarchical clustering

Multi-dimensional scaling (MDS)

https://www.statisticshowto.com/multidimensional-scaling/

- Multidimensional scaling is a **visual representation of distances or dissimilarities between sets of objects**.
- "Objects" can be colors, faces, map coordinates, political persuasion, or any kind of real or conceptual stimuli (Kruskal and Wish, 1978).
- Objects that are more **similar** (or have shorter distances) are closer together on the graph than objects that are less similar (or have longer distances).
- As well as interpreting dissimilarities as distances on a graph, MDS can also serve as a dimension reduction technique for high-dimensional data (Buja et. al, 2007).

Data Visualization with MDS paper – Buja et.al.

https://faculty.wharton.upenn.edu/wp-content/uploads/2012/04/Buja-Swayne-Littman-Dean-Hofman-Chen-JCGS-2008-06-Vol17-Data-Visualization-With-Multidimensional-Scaling.pdf

- There exist several types of MDS, and they differ mostly in the loss function they use.

- Kruskal-Shepard distance scaling

vs

- Classical Torgerson–Gowerinner-product scaling

Or,

- Metric scaling

vs

- Nonmetric scaling

Basic steps of classical MDS:

https://www.statisticshowto.com/multidimensional-scaling/

- **1. Assign a number of points to coordinates in n-dimensional space**.
- *N*-dimensional space could be **2-dimensional, 3-dimensional**, or higher spaces (at least, theoretically, because 4-dimensional spaces and above are difficult to model).
- The orientation of the coordinate axes is arbitrary and is mostly the researcher's choice.

- **2. Calculate Euclidean distances for all pairs of points**.
  The Euclidean distance is the "as the crow flies" straight-line distance between two points x and y in Euclidean space.
- It's calculated using the Pythagorean theorem ($c^2 = a^2 + b^2$), although it becomes somewhat more complicated for n-dimensional space (see "Euclidean Distance in n-dimensional space"). This results in the similarity matrix.

Remaining basic steps:

- **3. Compare the similarity matrix with the original input matrix** by evaluating the **stress** function.
- *Stress* is a goodness-of-fit measure, based on differences between predicted and actual distances.
- In his original 1964 MDS paper, **Kruskal wrote that fits close to zero are excellent**, while anything over 0.2 should be considered "poor".
- More recent authors suggest evaluating stress based on the **quality of the distance matrix and how many objects** are in that matrix.
- **4. Adjust coordinates, if necessary, to minimize stress**.

Let’s fit MDS in the USArrests data:

#Getting scaled data sans 3rd variable:

USArrests.1 <- scale(USArrests[,-3])

#Classical MDS using Kruskall’s stress

We first need to get the distance

#Distance calculation

- state.disimilarity <- dist(USArrests.1)

- #MDS fit
- mds.1 <- cmdscale(state.disimilarity)
- summary(mds.1)

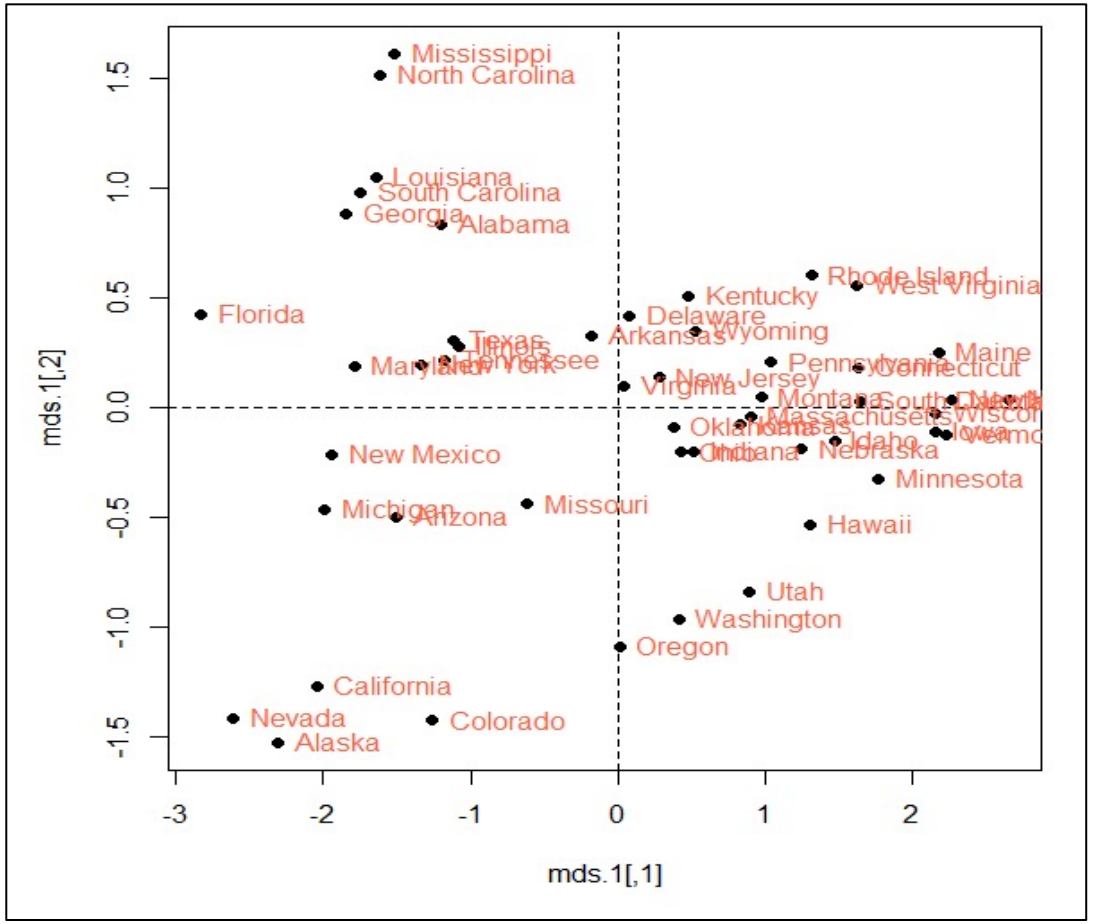
	V1	V2
•	Min. :-2.8307	Min. :-1.52396
•	1st Qu.:-1.4602	1st Qu.:-0.21091
•	Median : 0.3259	Median : 0.03608
•	Mean : 0.0000	Mean : 0.00000
•	3rd Qu.: 1.2870	3rd Qu.: 0.32033
•	Max. : 2.6545	Max. : 1.60646

Let’s plot the MDS:
This is a bi-plot so interpret as we did it!

#MDS plot

- plot(mds.1, pch = 19)
- abline(h=0, v=0, lty=2)
- text(mds.1, pos = 4, labels = rownames(USArrests.1), col = 'tomato’)

#What is the interpretation of this plot?

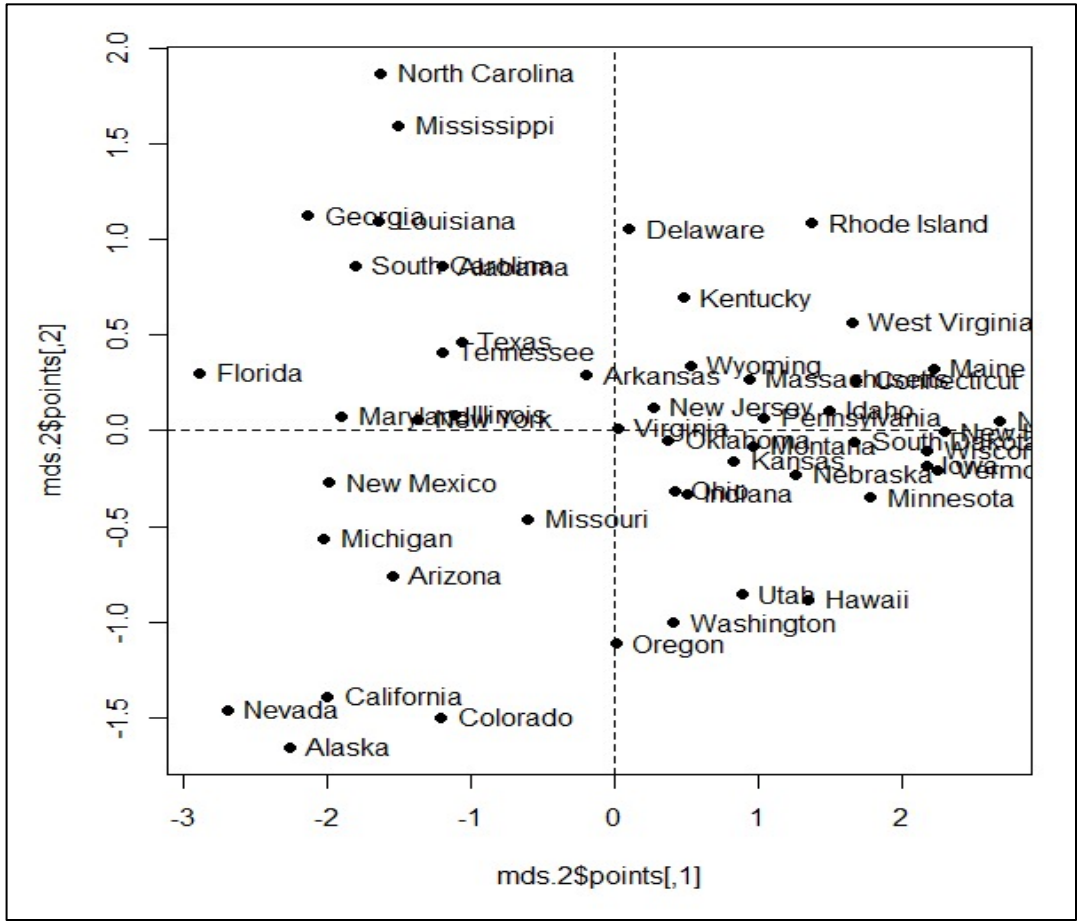


Can we improve it?

We can try using Sammon’t stress!

#Alternative approach with Sammon's stress

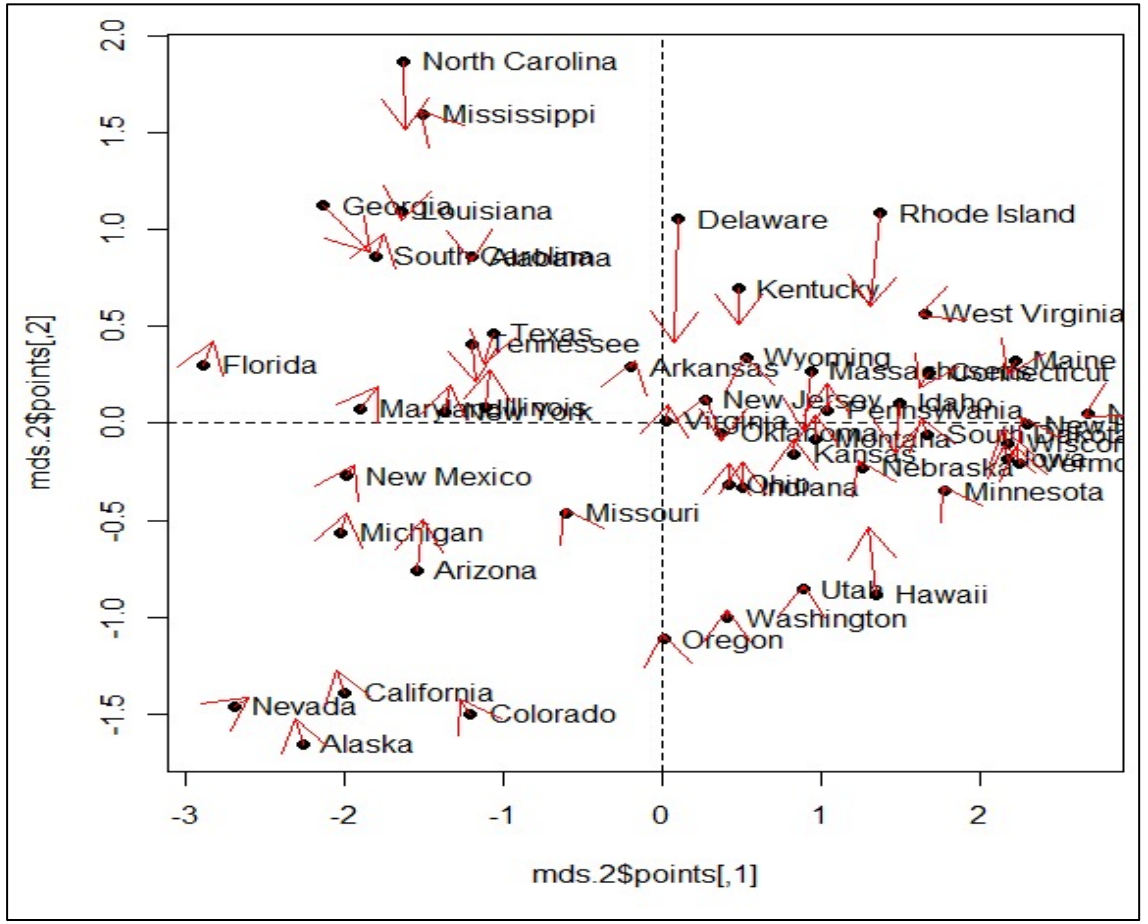
- mds.2 <- MASS::sammon(state.disimilarity, trace = FALSE)
- plot(mds.2$points, pch = 19)
- abline(h=0, v=0, lty=2)
- text(mds.2$points, pos = 4, labels = rownames(USArrests.1))



Comparing pca.1 (last class) and mds.2 models with annotations:

 # Compare with PCA (first two PCs):

- arrows(
- x0 = mds.2$points[,1], y0 = mds.2$points[,2],
- x1 = pca.1$x[,1], y1 = pca.1$x[,2],
- col='red', pch=19, cex=0.5)



Question/queries so far?

Cluster analysis (Clustering): Chapter 12 “An Introduction to Statistical Learning” book!

- Clustering refers to a very broad set of techniques for finding subgroups, or clustering clusters, in a data set.
- When we cluster the observations of a data set, we seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other.
- Both clustering and PCA seek to simplify the data via a small number of summaries, but their mechanisms are different:
- PCA looks to find a low-dimensional representation of the observations that explain a good fraction of the variance;
- Clustering looks to find homogeneous subgroups among the observations.

Cluster analysis (Clustering): Chapter 12 “An Introduction to Statistical Learning” book!

- Since clustering is popular in many fields, there exist a great number of clustering methods. In this section we focus on perhaps the two best-known clustering approaches:
-
  - **K-means clustering and**
  - **hierarchical clustering.**

- In K-means clustering, we seek to partition the observations into a pre-specified number of clusters.
- On the other hand, in hierarchical clustering, we do not know in advance how many clusters we want and we use “**dendogram**” to find the number of clusters for the data

k-means clustering: which “k” is the best?

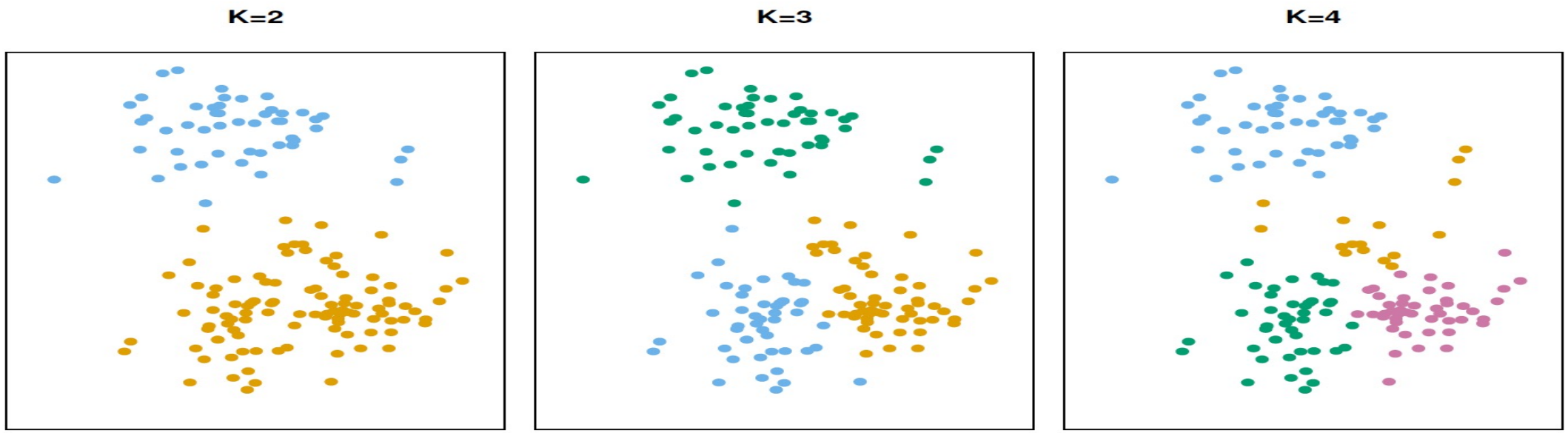


FIGURE 12.7. *A simulated data set with 150 observations in two-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.*

k-means clustering with random data: ISLR

#ISLR book

- set.seed (2)
- x <- matrix(rnorm (50 * 2), ncol = 2)
- x[1:25, 1] <- x[1:25, 1] + 3
- x[1:25, 2] <- x[1:25, 2] – 4

#We are creating two group!

#k-means clustering

- km.out <- kmeans(x, 2, nstart = 20)
- **km.out (check the variance explained!)**

#Checking the clusters

- km.out$cluster

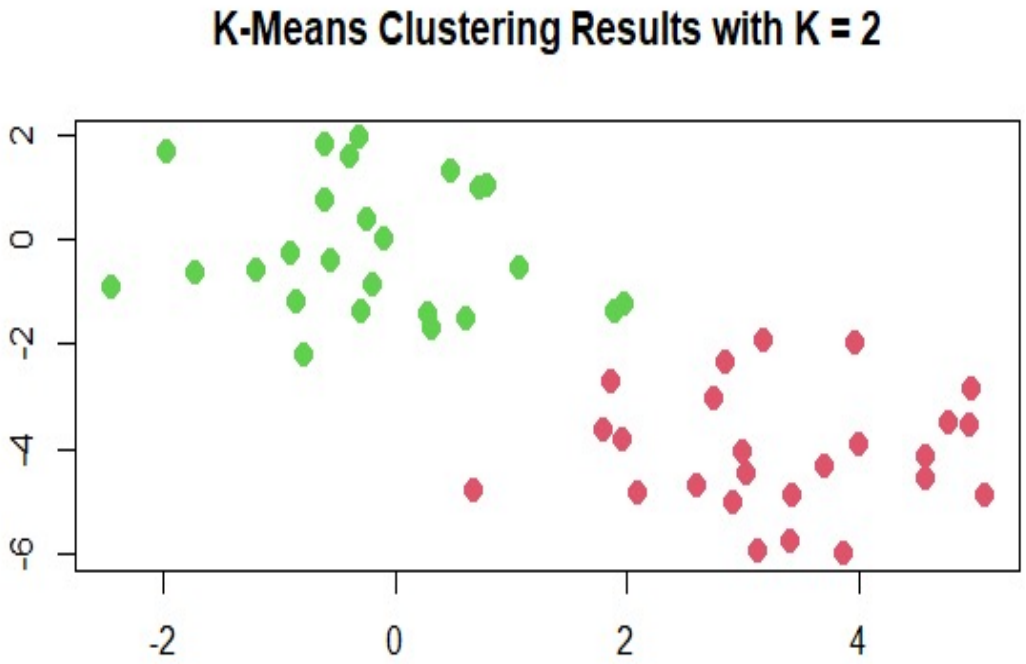
#We have used K=2 as we have created a random data with 2 groups

ISLR book: We strongly recommend always running K-means clustering with a large value of nstart, such as 20 or 50, since otherwise an undesirable local optimum may be obtained.

Plot the clusters:

#Plot

```
plot(x, col = (km.out$cluster + 1),
main = "K-Means Clustering
Results with K = 2",
xlab = "", ylab = "", pch = 20, cex =
2)
```



Let us use k=3 in the data and see what happens:

#Clustering with 3 clusters in the random data

- set.seed (4)
- km.out <- kmeans(x, 3, nstart = 20)
- km.out

K-means clustering with 3 clusters of sizes 17, 23, 10

- Cluster means:
- [,1] [,2]
- 1 3.7789567 -4.56200798
- 2 -0.3820397 -0.08740753
- 3 2.3001545 -2.69622023

Within cluster sum of squares by cluster:

- [1] 25.74089 52.67700 19.56137
- (between_SS / total_SS = 79.3 %)

Plot:

#Plot

plot(x, col = (km.out$cluster + 1),

main = "K-Means Clustering Results with K = 3",

xlab = "", ylab = "", pch = 20, cex = 2)

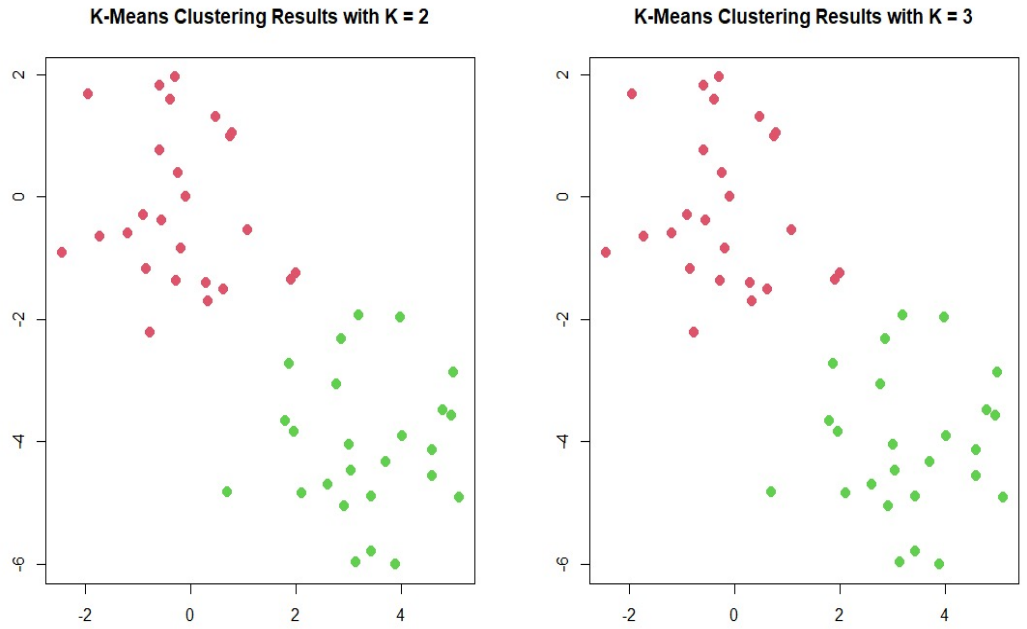
Comparing two plots: 2-cluster and 3-cluster

#Plot the clusters

- par(mfrow = c(1, 2))
- Code of plot with 2 clusters
- Code of plot with 3 clusters

How to decide: which k is best?

We need to use hierarchical clustering!



Comparing different nstarts:
Plot them and see the changes!

#nstart = 1

- set.seed (4)
- km.out <- kmeans(x, 3, nstart = 1)
- km.out$tot.withinss

#Km within ss for

- 97.97927
- (between_SS / total_SS = 79.3 %)

#nstart =20

- km.out <- kmeans(x, 3, nstart = 20)
- km.out$tot.withinss

#Km within ss

- 104.3319
- (between_SS / total_SS = 78.0 %)

Question:

- What is the difference when nstart = 1 and nstart = 20 used?
- Which one should we use?
- Why nstart=1 has more variance that nstart=20?
- https://datascience.stackexchange.com/questions/11485/k-means-in-r-usage-of-nstart-parameter

Let’s do k-means clustering with “iris” data:

#Load two packages for special plot

- library(ClusterR)
- **library(cluster)**

#Get, check and make data

- data(iris)
- str(iris)
- iris_1 <- iris[,-5]

Fitting K-Means clustering Model to training dataset

- set.seed(240)
- kmeans.res <- kmeans(iris_1, **centers = 3, nstart = 20**)
- kmeans.res

We have used k=3 as we know that there are 3 types of flowers!

k-means fit:

- K-means clustering with 3 clusters of sizes 50, 62, 38

- Cluster means:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.006000	3.428000	1.462000	0.246000
2	5.901613	2.748387	4.393548	1.433871
3	6.850000	3.073684	5.742105	2.071053

- Within cluster sum of squares by cluster:

- [1] 15.15100 39.82097 23.87947

- (between_SS / total_SS = **88.4 %**)

Confusion matrix: Possible here as we also have dependent variable to compare!

 # Confusion Matrix

- cm <- table(iris$Species, kmeans.res$cluster)
- cm

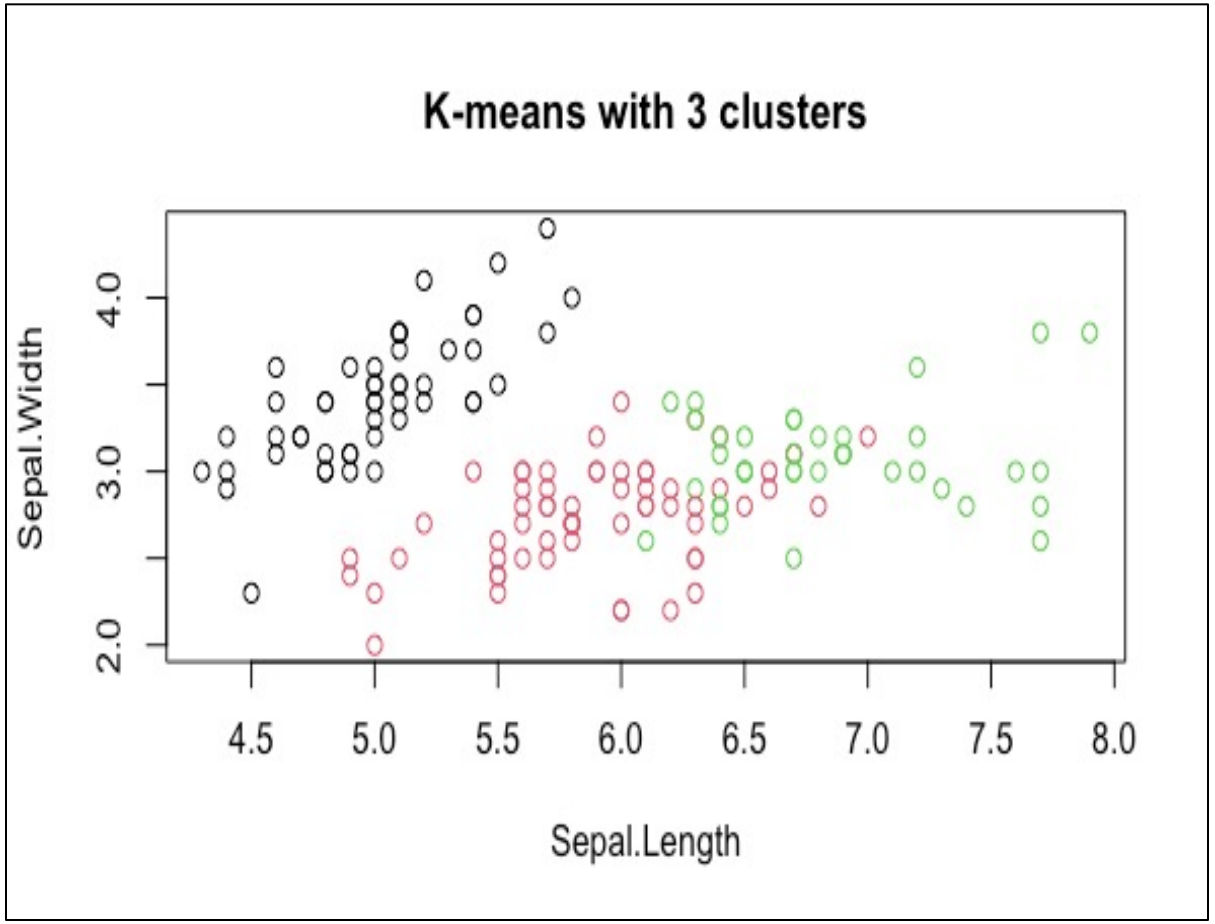
	1	2	3
• setosa	50	0	0
• versicolor	0	48	2
• virginica	0	14	36

- #Accuracy
- (accuracy <- sum(diag(cm))/sum(cm))
- [1] 0.8933333
- (mce <- 1 - accuracy)
- [1] 0.1066667

Model Evaluation and Visualization:

Model Evaluation and visualization

- plot(iris_1[c("Sepal.Length", "Sepal.Width")])
- plot(iris_1[c("Sepal.Length", "Sepal.Width")],
  col = kmeans.res$cluster)
- **plot(iris_1[c("Sepal.Length", "Sepal.Width")],**
  col = kmeans.res$cluster,
  main = "K-means with 3 clusters")



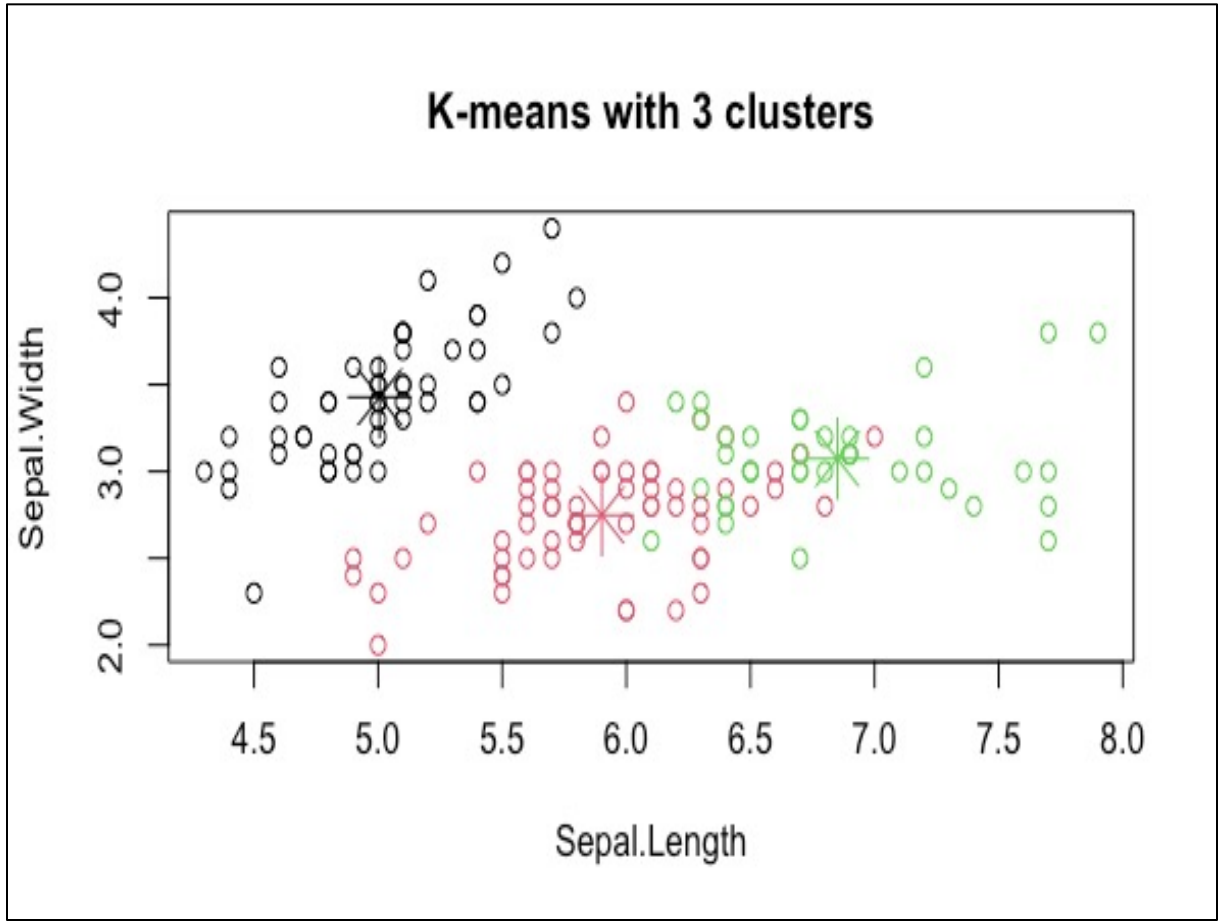
Adding cluster centers (use points after plot):

 # Getting cluster centers

- kmeans.res$centers
- kmeans.res$centers[, c("Sepal.Length",
- "Sepal.Width")]

 # Plotting cluster centers

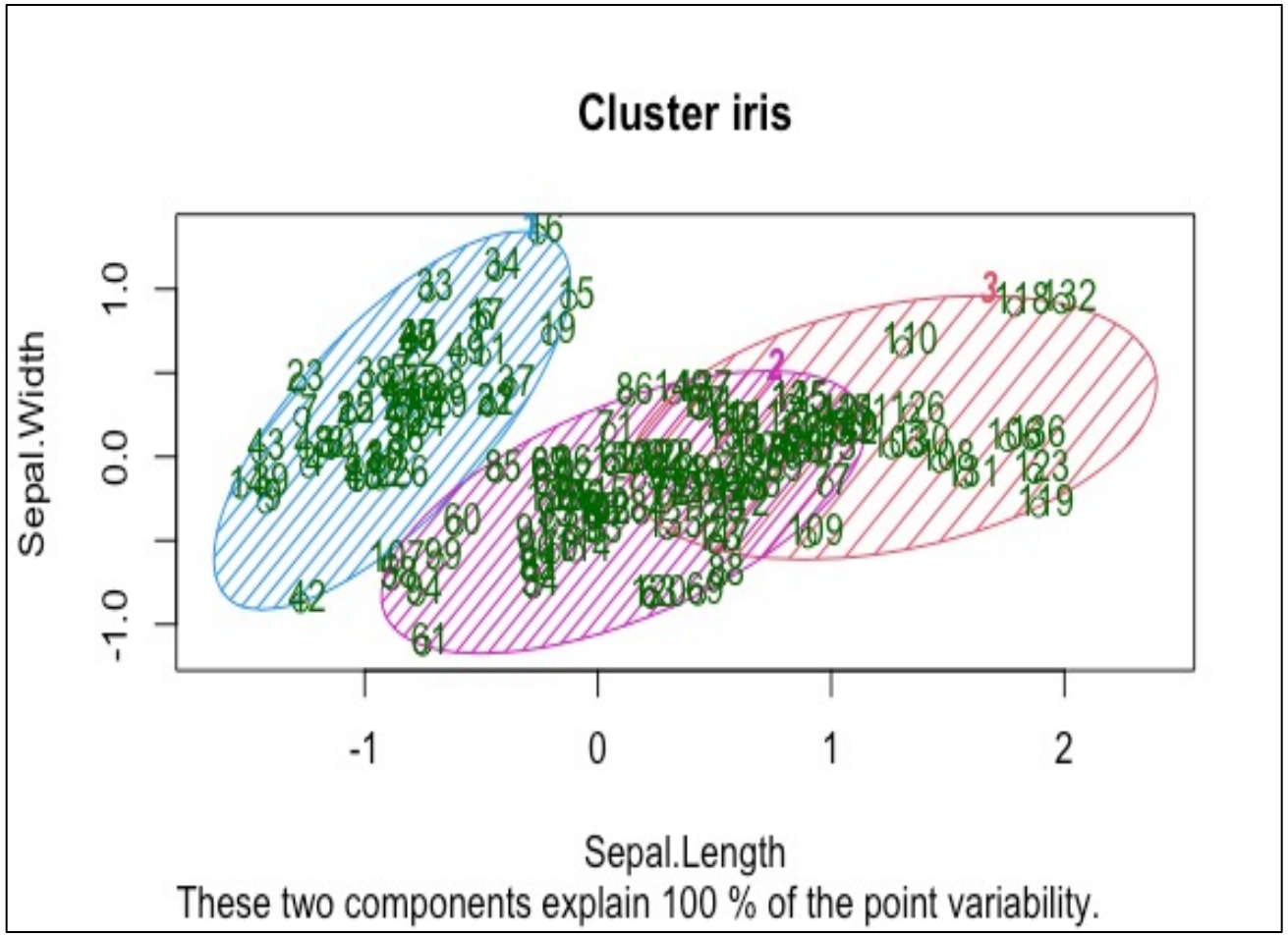
- points(kmeans.res$centers[, c("Sepal.Length",
- "Sepal.Width")],
- col = 1:3, pch = 8, cex = 3)



Visualizing clusters:

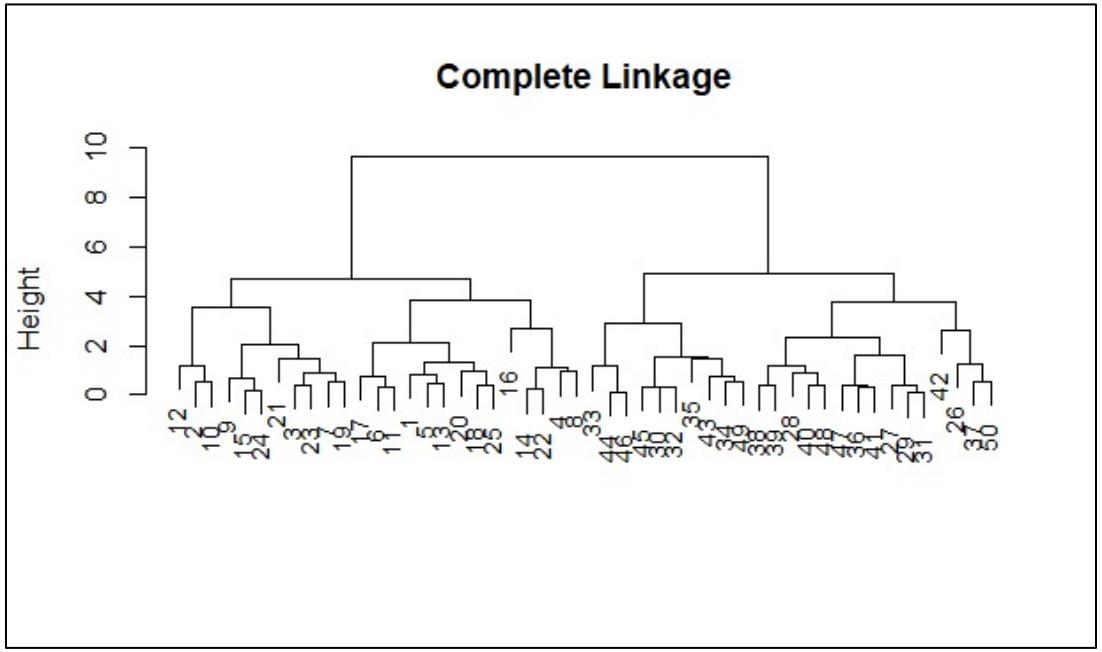
 # Visualizing clusters

- y_kmeans <- kmeans.res$cluster
- library(cluster)
- clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
- y_kmeans,
- lines = 0,
- shade = TRUE, color = TRUE,
- labels = 2,
- plotchar = FALSE, span = TRUE,
- main = paste("Cluster iris"),
- xlab = 'Sepal.Length',
- ylab = 'Sepal.Width')



Hierarchical cluster analysis (HCA):

- One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K.
- Hierarchical clustering is an alternative approach which **does not require that we commit to a particular choice of K**.
- Hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a **dendrogram**.



HCA algorithm:

- The hierarchical clustering dendrogram is obtained via an extremely simple algorithm.
- We begin by defining some sort of dissimilarity measure between each pair of observations. Most often, Euclidean distance is used.
- The algorithm proceeds iteratively.
- Starting out at the bottom of the dendrogram, each of the n observations is treated as its own cluster.

HCA algorithm:

- The two clusters that are most similar to each other are then fused so that there now are n−1 clusters.
- Next the two clusters that are most similar to each other are fused again, so that there now are n − 2 clusters.
- The algorithm proceeds in this fashion until all of the observations belong to one single cluster, and the dendrogram is complete.

Interpreting “dendogram”: Dendogram (1st figure) vs Bi-plot (2nd figure)

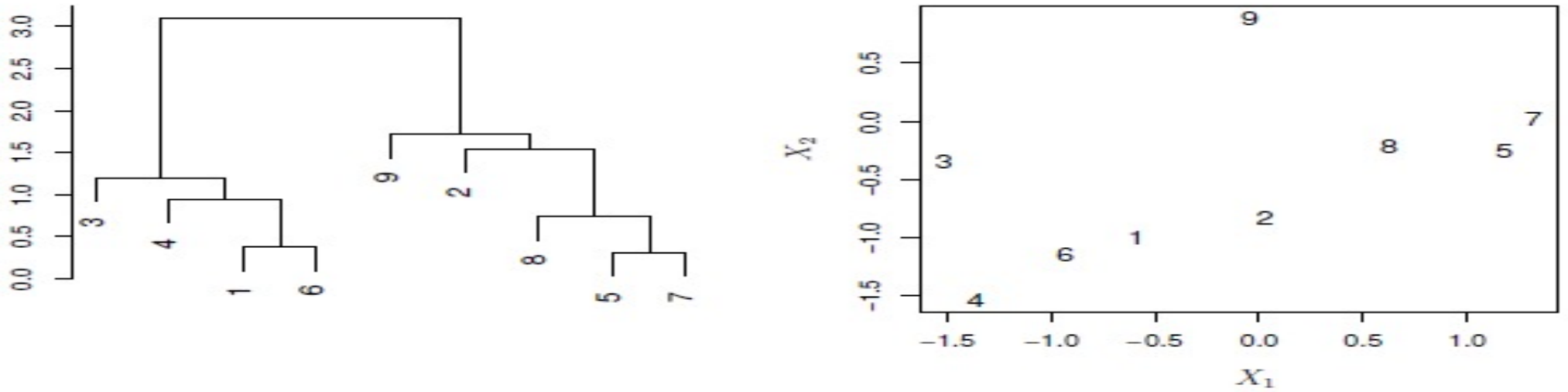


FIGURE 12.12. *An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space.* Left: *a dendrogram generated using Euclidean distance and complete linkage. Observations* 5 *and* 7 *are quite similar to each other, as are observations* 1 *and* 6. *However, observation* 9 *is* no more similar to *observation* 2 *than it is to observations* 8, 5, *and* 7, *even though observations* 9 *and* 2 *are close together in terms of horizontal distance. This is because observations* 2, 8, 5, *and* 7 *all fuse with observation* 9 *at the same height, approximately* 1.8. Right: *the raw data used to generate the dendrogram can be used to confirm that indeed, observation* 9 *is no more similar to observation* 2 *than it is to observations* 8, 5, *and* 7.

Hierarchical clustering: How many clusters?
Need to find k using a vertical “cut” line/s!

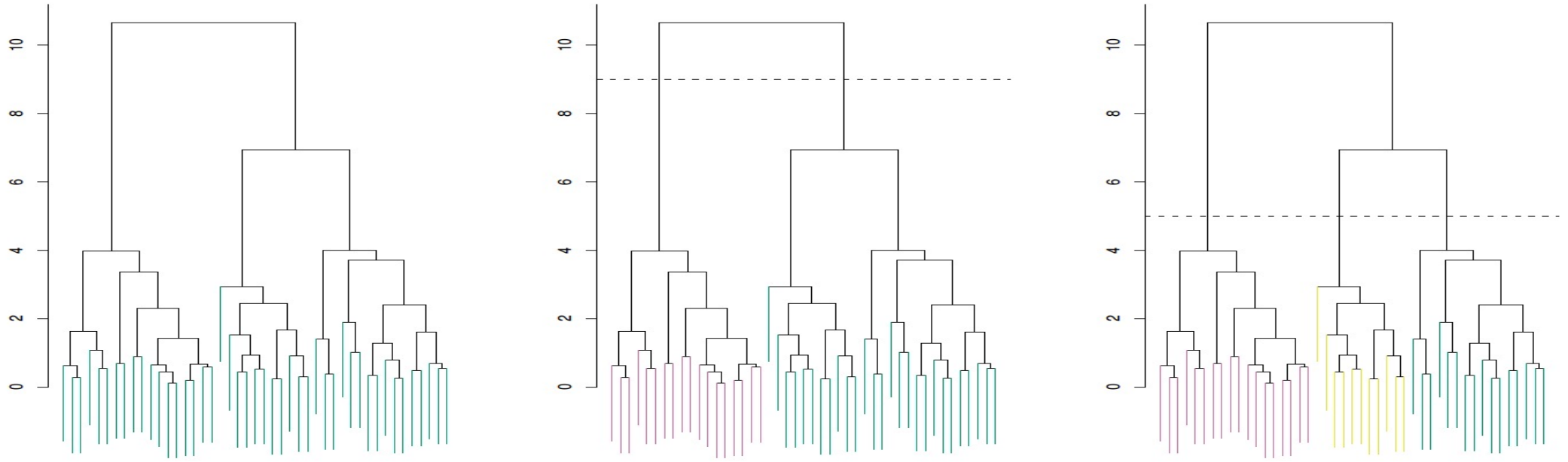


FIGURE 12.11. Left: *dendrogram obtained from hierarchically clustering the data from Figure 12.10 with complete linkage and Euclidean distance.* Center: *the dendrogram from the left-hand panel, cut at a height of nine (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.* Right: *the dendrogram from the left-hand panel, now cut at a height of five. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure.*

HCA: Linkage methods for choosing "dissimilarity" measure

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

TABLE 12.3. *A summary of the four most commonly-used types of linkage in hierarchical clustering.*

HCA with “single” linkage: USArrests.1 data:

#Hierarchical clustering with single linkage

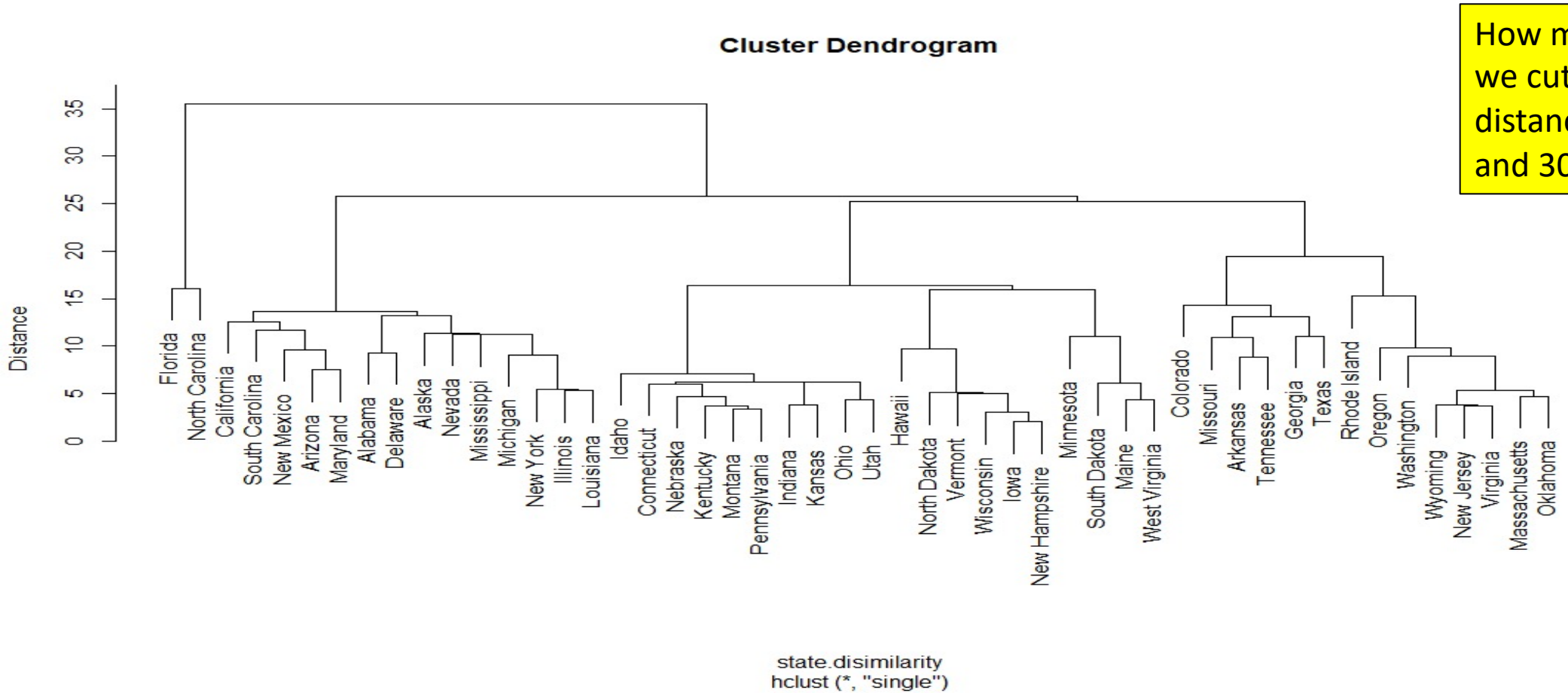
- #US Arrests data
- USArrests.1 <- USArrests[,-3]
- state.disimilarity <- dist(USArrests.1)
- hirar.1 <- hclust(state.disimilarity, method='single')
- **plot(hirar.1, labels=rownames(USArrests.1), ylab="Distance")**

- hirar.1

- Call:
- hclust(d = state.disimilarity, method = "single")

- Cluster method : single
- Distance : euclidean
- Number of objects: 50

HCA with “single” linkage in USArrests.1 data



How many k if we cut at distance of 20 and 30?

HCA with “single” linkage: USArrests.1 data:

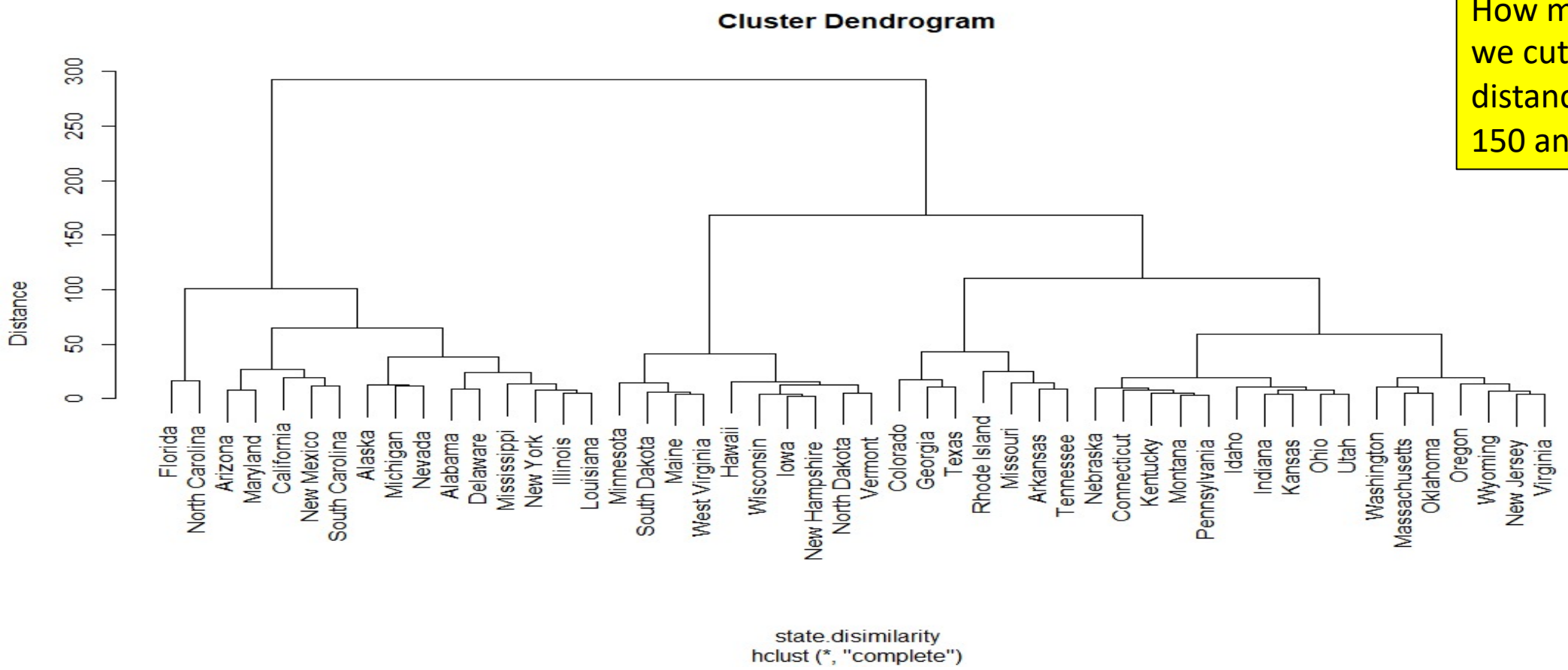
#Hierarchical clustering with complete linkage

- #US Arrests data
- hirar.2 <- hclust(state.disimilarity, method=‘complete')
- **plot(hirar.2, labels=rownames(USArrests.1), ylab="Distance")**

- Call:
- hclust(d = state.disimilarity, method = "complete")

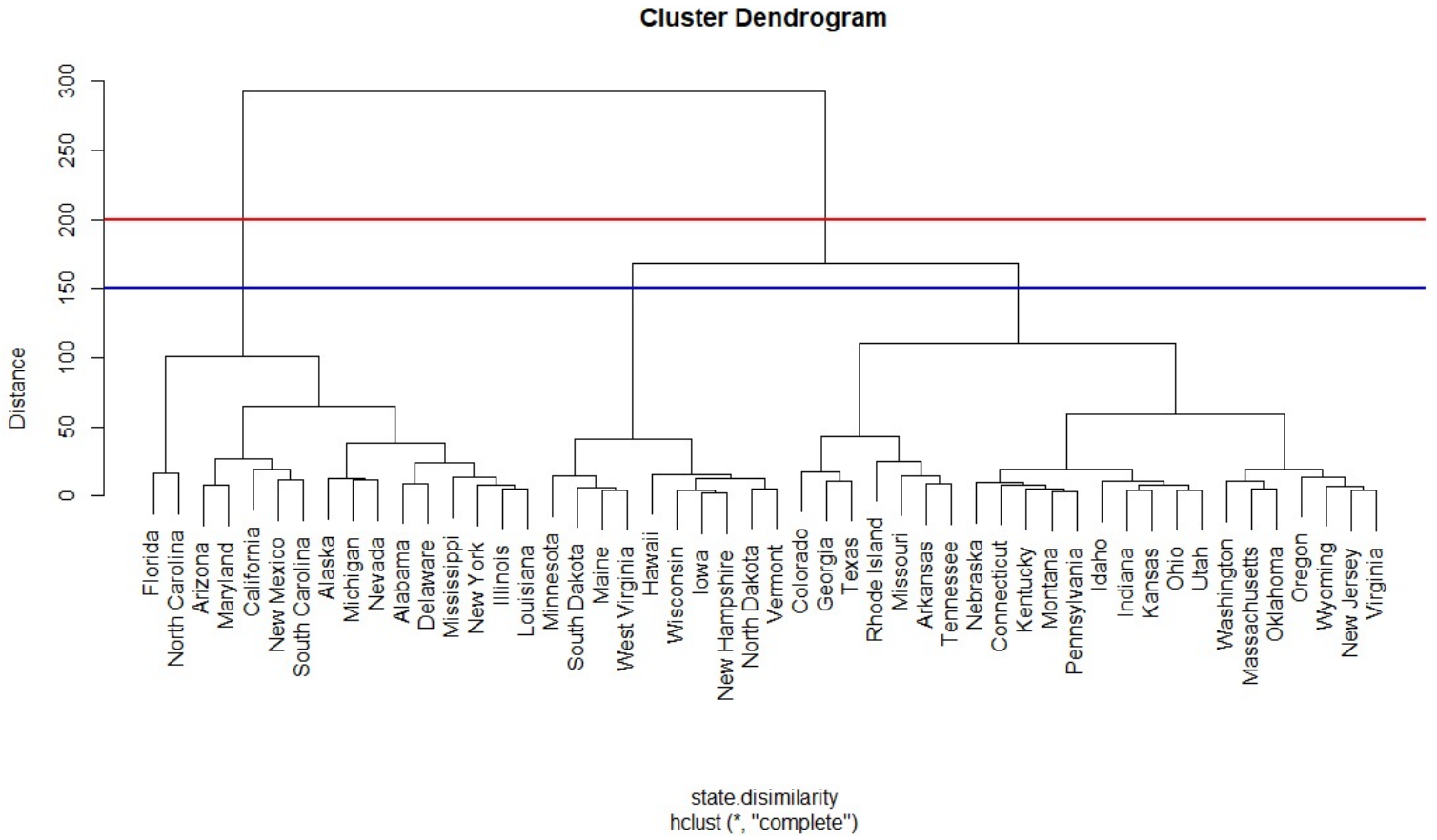
- Cluster method : complete
- Distance : euclidean
- Number of objects: 50

HCA with “complete” linkage in USArrests.1 data



How many k if we cut at distance of 75, 150 and 200?

HCA with “complete” linkage in USArrests.1 data with cut at distance of 200 and 150!



So, it is always better to use the HCA to determine the K and then use it to fit the k-means clustering.

In Data Science, we need to use all the four methods and find best k and the fit k-mean for each of the best K’s. Then select the best clustering model based on the highest R-square value.

Question/queries?

- Next class

- Association rules
- Monte Carlo Simulations

- Try to install “git” on Windows so that we can use “github” in R Studio while creating online projects

- We will need to use the offline projects in R Studio too

Thank you!

@shitalbhandary