INFS4203/7203 Data Mining
The University of Queensland, Australia
Semester 2, 2018

Tutorial Week 9: Clustering with R

Chandra Prasetyo Utomo

c.utomo@uq.edu.au

Objectives

1. To be able to implement K-Means Clustering algorithm in R with `kmeans` function
2. To be able to implement Hierarchical Clustering algorithm in R with `hclust` function
3. To gain familiarity with arguments (parameters) and output's components in `kmeans` and `hclust` functions.

Outline

1. K-Means Algorithm Implementation (*25 minutes*)
2. Hierarchical Clustering Implementation (*25 minutes*)

Part 1

K-Means Algorithm

~/Dropbox/DataMining2018/Tutorial01/session04 - RStudio

Go to file/function | Addins | session04

part3.r | part3b.r

Source on Save | Run | Source

```
1  ### R Tutorial Part 3: Clustering ###
2  ####      K-Means Clustering     ####
3
4  # Extract Data
5  iris <- read.table("./data/iris.data", sep = ',')
6
7  # Assign name to variables
8  names(iris) <- c("Sepal.Length", "Sepal.Width",
9                   "Petal.Length", "Petal.Width", "Species")
10
11 # Select first four vars (i.e remove class var)
12 iris2 <- iris[ , 1:4]
13
14 # For reproducible result
15 set.seed(2018)
16
17 # Cluster with K-means into nclust clusters
18 nclust = 3
19 (kmeans.result <- kmeans(iris2, nclust))
20
21 plot(iris[ , c("Sepal.Length", "Sepal.Width")],
22      col = kmeans.result$cluster)
23 title(paste("k = ", nclust, sep=""))
24 points(kmeans.result$centers[ , c("Sepal.Length", "Sepal.Width")],
25        col = 1:nclust, pch = 8, cex = 2)
26
```

26:1 | K-Means Clustering | R Script

Environment | History

Import Dataset | List

Global Environment

Data

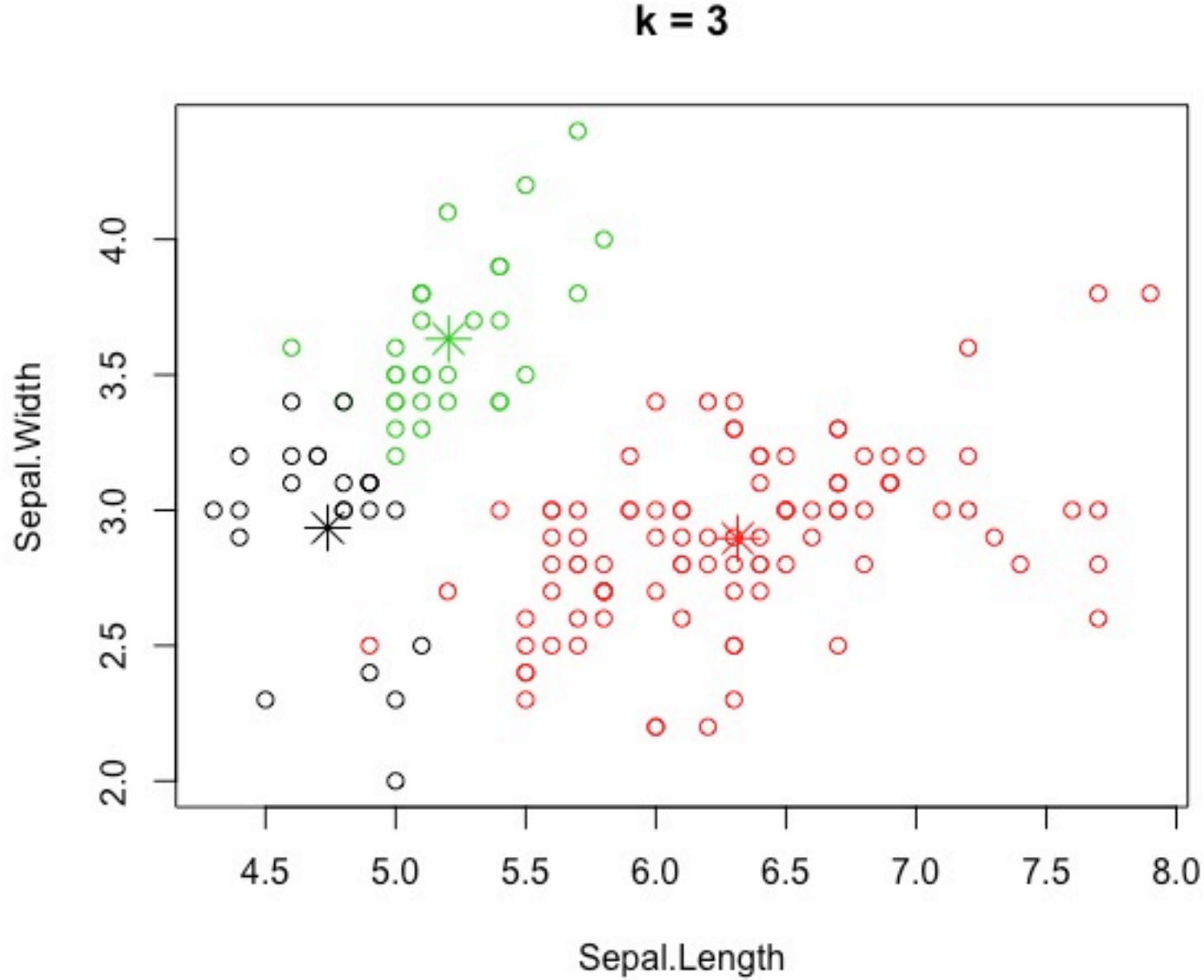
iris	150 obs. of 5 variables
iris2	150 obs. of 4 variables

Values

kmeans.result — List of 9

```
 cluster : int [1:150] 3 3 3 3 3 3 3 3 3 3 ...
 centers : num [1:3, 1:4] 6.85 5.9 5.01 3.07 2.75 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:3] "1" "2" "3"
 .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Len…
 totss : num 681
 withinss : num [1:3] 23.9 39.8 15.2
 tot.withinss: num 78.9
 betweenss : num 602
 size : int [1:3] 38 62 50
 iter : int 2
 ifault : int 0
 attr(*, "class")= chr "kmeans"
```

nclust — 3

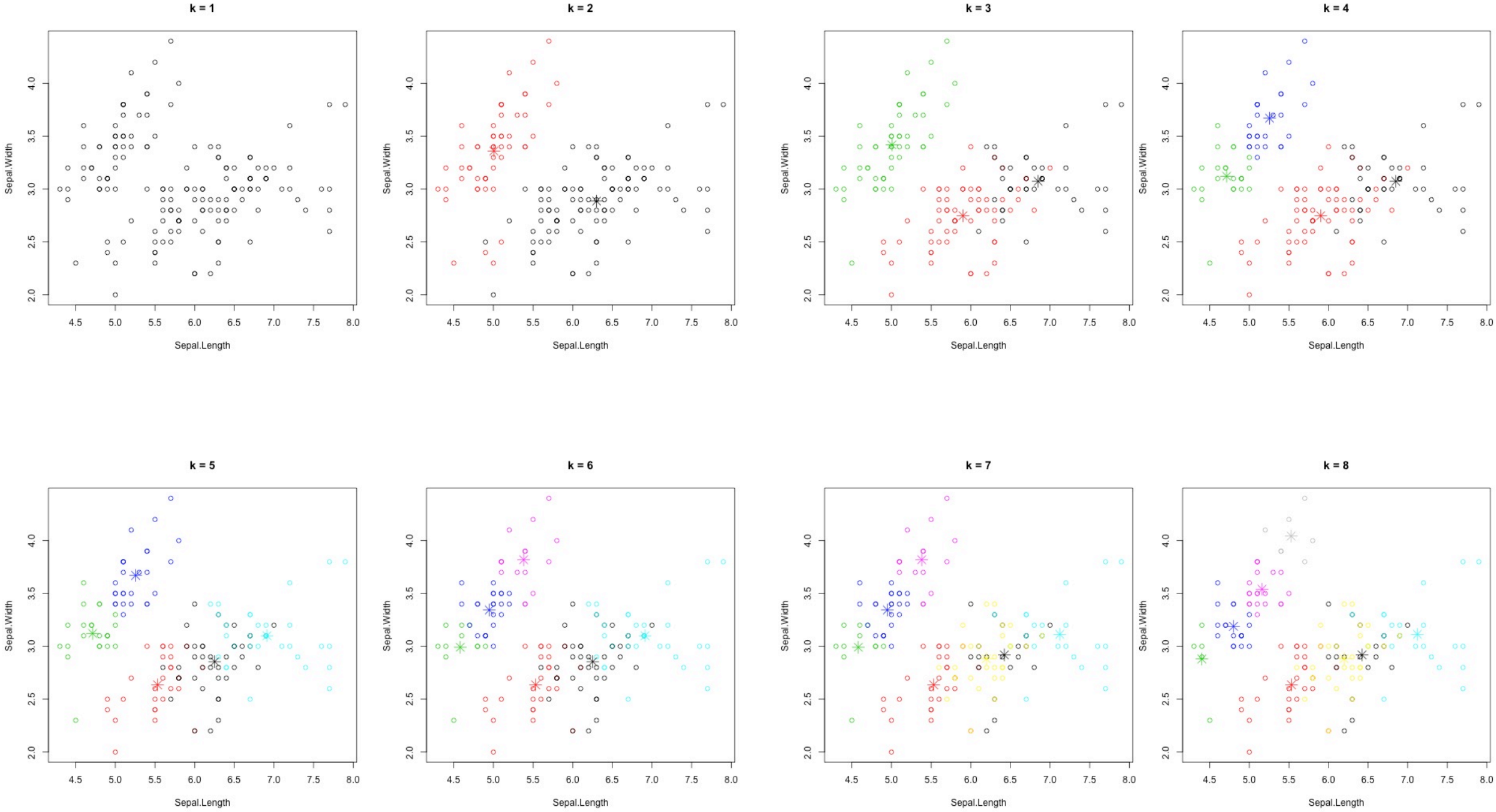

k = 3
Sepal.Width
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Sepal.Length

Arguments

`x`	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
`centers`	either the number of clusters, say *k*, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in x is chosen as the initial centres.
`iter.max`	the maximum number of iterations allowed.
`nstart`	if centers is a number, how many random sets should be chosen?
`algorithm`	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative names for one algorithm.
`object`	an **R** object of class "kmeans", typically the result ob of ob <- kmeans(..).
`method`	character: may be abbreviated. "centers" causes fitted to return cluster centers (one for each input point) and "classes" causes fitted to return a vector of class assignments.
`trace`	logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced. Higher values may produce more tracing information.
`...`	not used.

Outputs

`cluster`	A vector of integers (from 1:k) indicating the cluster to which each point is allocated.
`centers`	A matrix of cluster centres.
`totss`	The total sum of squares.
`withinss`	Vector of within-cluster sum of squares, one component per cluster.
`tot.withinss`	Total within-cluster sum of squares, i.e. sum(withinss).
`betweenss`	The between-cluster sum of squares, i.e. totss-tot.withinss.
`size`	The number of points in each cluster.
`iter`	The number of (outer) iterations.
`ifault`	integer: indicator of a possible algorithm problem – for experts.


k = 1
k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
Sepal.Width
Sepal.Length
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Discussion

- Why finding good number of clusters is useful in real-world applications and how to measure it?

Part 2

Hierarchical Clustering Algorithm

part3.r | part3b.r

Source on Save | Run | Source

```r
### R Tutorial Part 3: Clustering ###
####   Hierarchical Clustering   ####

# Extract Data
iris <- read.table("./data/iris.data", sep = ',')

# Assign name to variables
names(iris) <- c("Sepal.Length", "Sepal.Width",
                 "Petal.Length", "Petal.Width", "Species")

# Select first four vars (i.e remove class var)
iris2 <- iris[ , 1:4]

# For reproducible result
set.seed(2018)

n = nrow(iris)
idx <- sample(1:n, 40)
irisSample <- iris2[idx,]

hc <- hclust(dist(irisSample), method="ave")
plot(hc, hang = -1, labels=iris$Species[idx])

# Cut the dendrogram into nclust clusters
nclust = 3
rect.hclust(hc, k=nclust)
groups <- cutree(hc, k=nclust)
```

15:15 | Hierarchical Clustering | R Script

Environment | History

Import Dataset | List

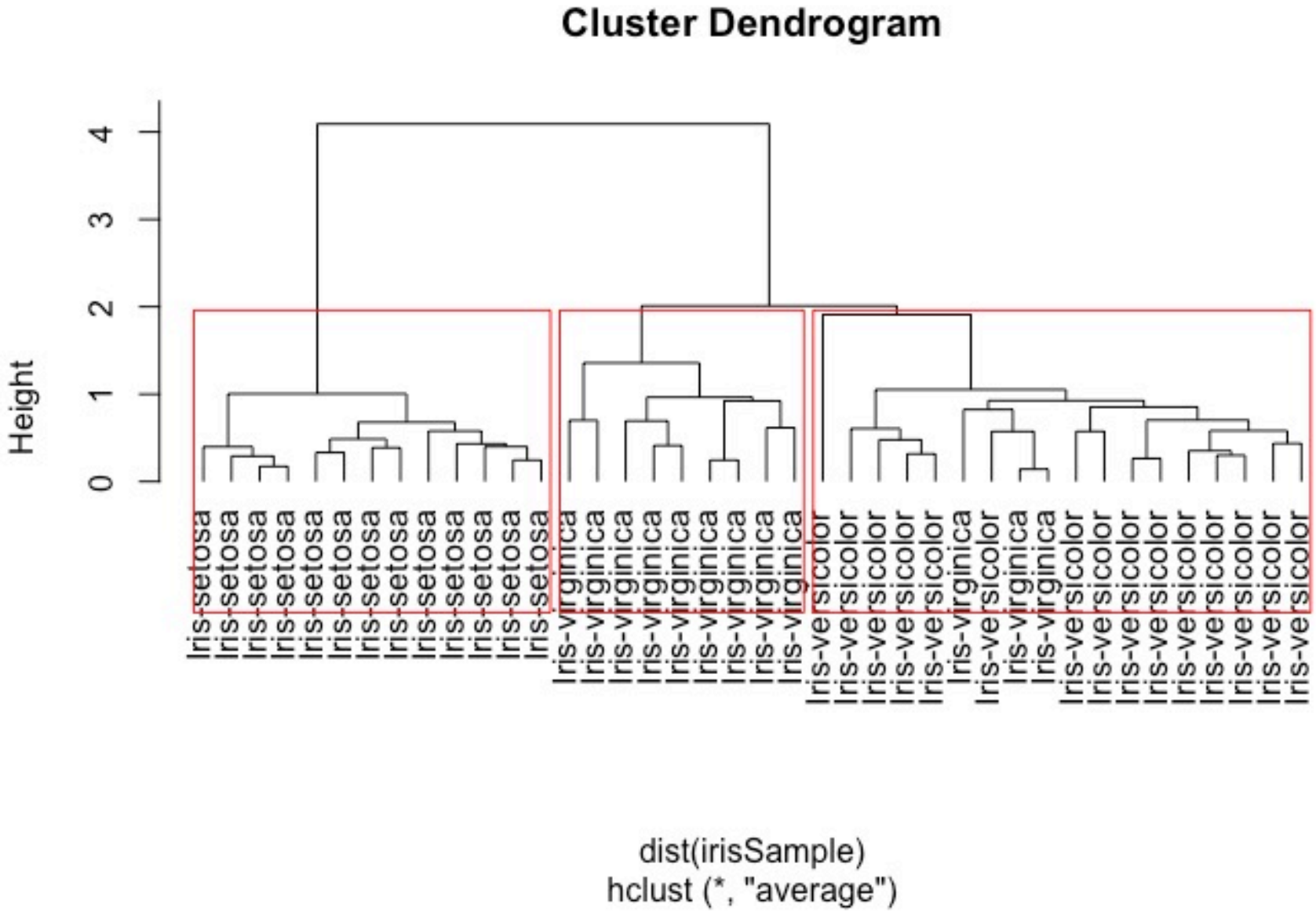
Global Environment

Data

iris	150 obs. of 5 variables
iris2	150 obs. of 4 variables
irisSample	40 obs. of 4 variables

Values

groups	Named int [1:40] 1 1 2 2 3 2 1 2 3 1 ...
hc	List of 7
merge :	int [1:39, 1:2] -14 -4 -5 -27 -2 -24 -11 -35 -8 -2…
height :	num [1:39] 0.141 0.173 0.245 0.245 0.265 ...
order :	int [1:40] 3 24 4 33 8 37 21 31 17 6 ...
labels :	chr [1:40] "51" "70" "9" "30" ...
method :	chr "average"
call :	language hclust(d = dist(irisSample), method = "ave…
dist.method:	chr "euclidean"
attr(*, "class")=	chr "hclust"
idx	int [1:40] 51 70 9 30 149 44 88 19 137 78 .…
n	150L
nclust	3


Cluster Dendrogram
Height
0
1
2
3
4
Iris-setosa
Iris-virginica
Iris-versicolor
dist(irisSample)
hclust (*, "average")

Arguments

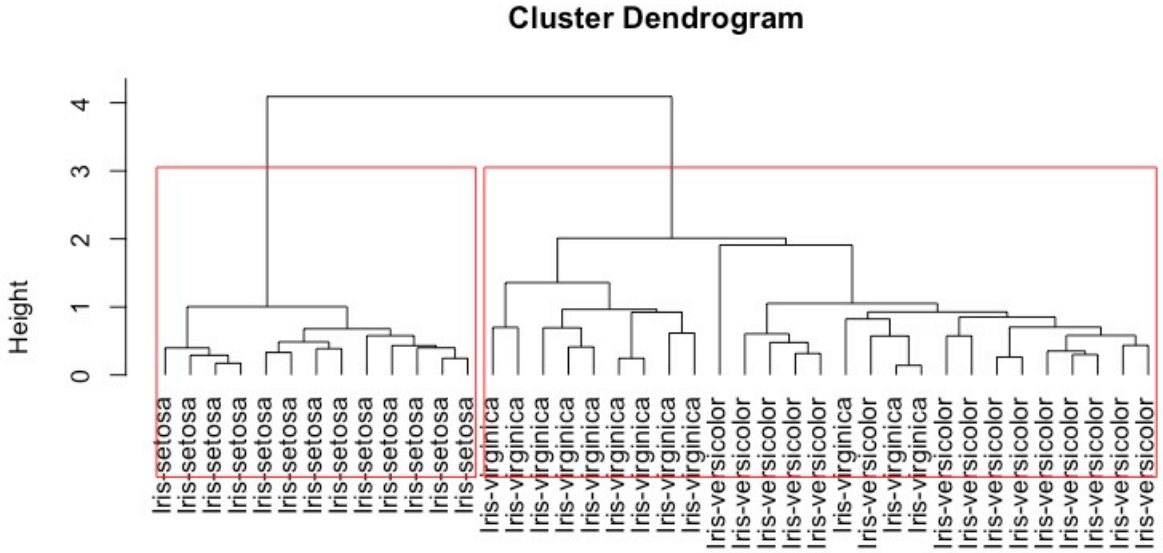
`d`	a dissimilarity structure as produced by dist.
`method`	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
`members`	NULL or a vector with length size of d. See the ‘Details’ section.
`x`	an object of the type produced by hclust.
`hang`	The fraction of the plot height by which labels should hang below the rest of the plot. A negative value will cause the labels to hang down from 0.

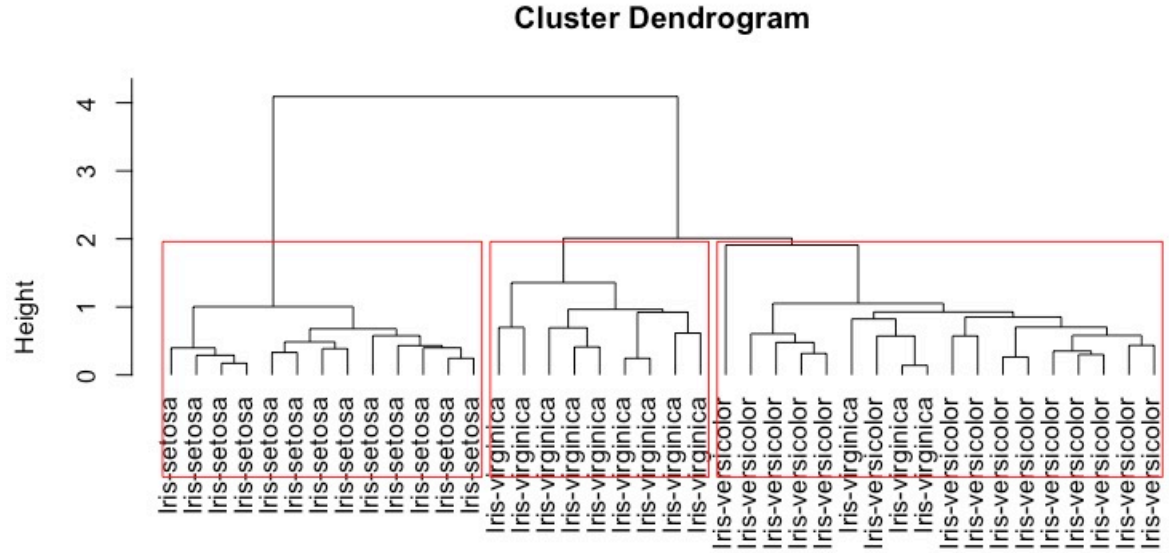
Arguments (cont'd)

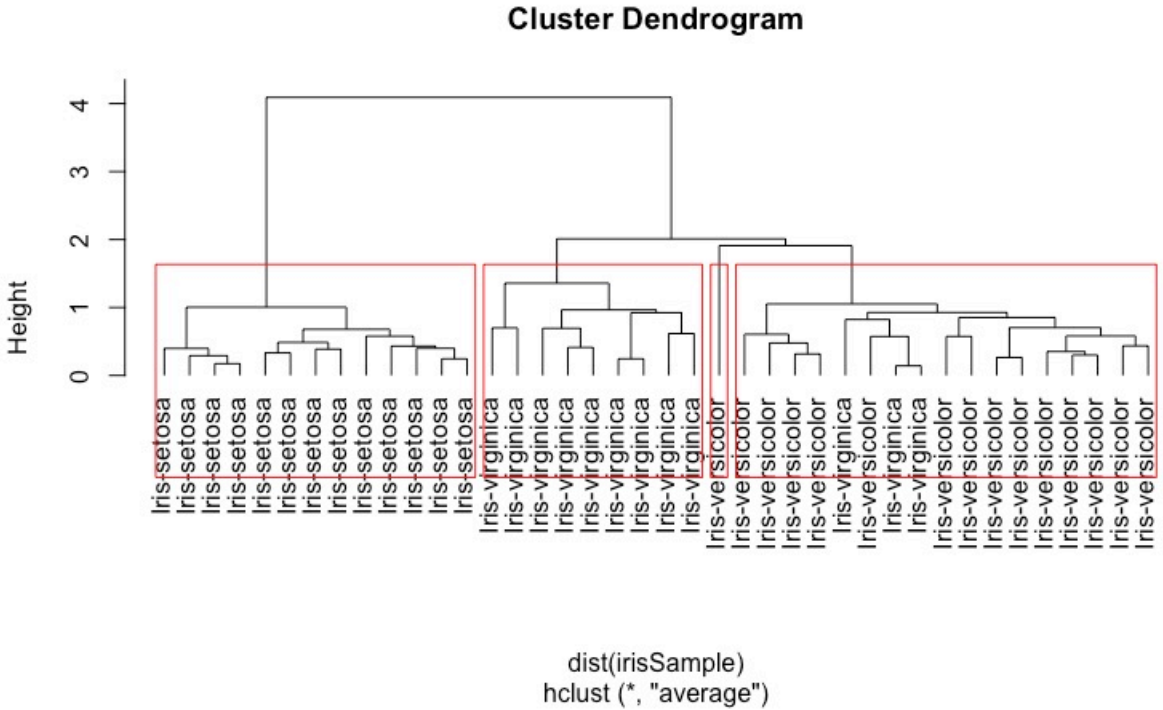
`check`	logical indicating if the x object should be checked for validity. This check is not necessary when x is known to be valid such as when it is the direct result of hclust(). The default is check=TRUE, as invalid inputs may crash **R** due to memory violation in the internal C plotting code.
`labels`	A character vector of labels for the leaves of the tree. By default the row names or row numbers of the original data are used. If labels = FALSE no labels at all are plotted.
`axes, frame.plot, ann`	logical flags as in plot.default.
`main, sub, xlab, ylab`	character strings for title. sub and xlab have a non-NULL default when there's a tree$call.
`...`	Further graphical arguments. E.g., cex controls the size of the labels (if plotted) in the same way as text.

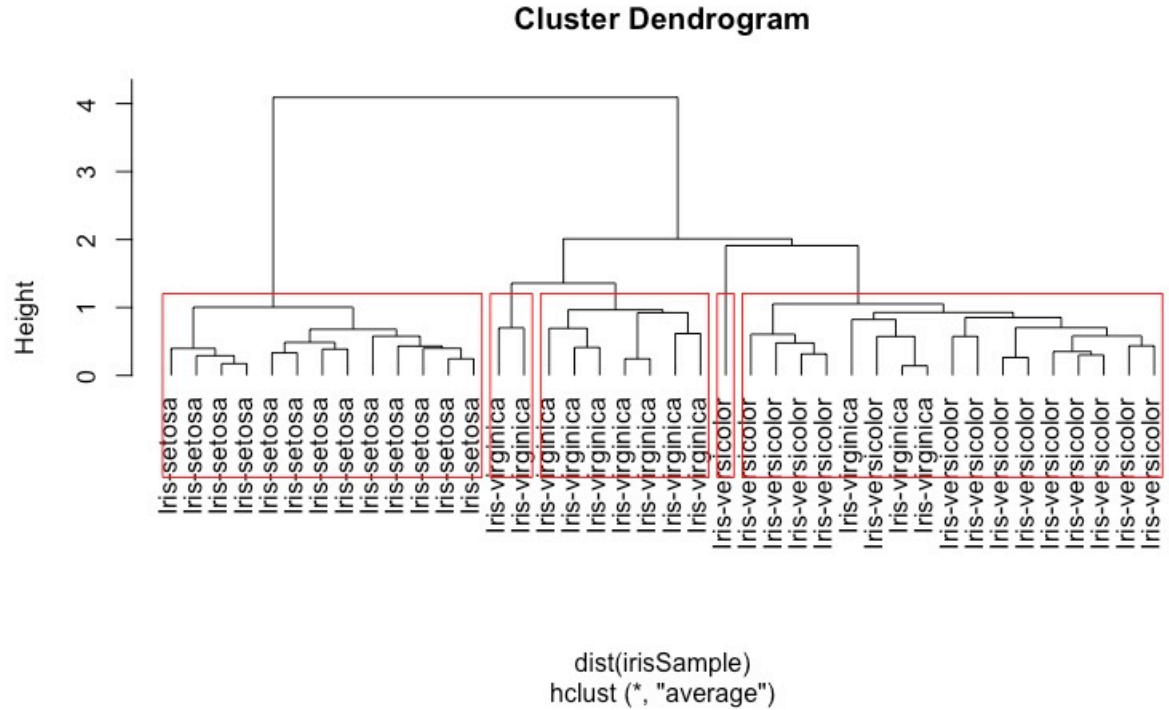
Outputs

`merge`	an *n-1* by 2 matrix. Row *i* of merge describes the merging of clusters at step *i* of the clustering. If an element *j* in the row is negative, then observation *-j* was merged at this stage. If *j* is positive then the merge was with the cluster formed at the (earlier) stage *j* of the algorithm. Thus negative entries in merge indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.
`height`	a set of *n-1* real values (non-decreasing for ultrametric trees). The clustering *height*: that is, the value of the criterion associated with the clustering method for the particular agglomeration.
`order`	a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches.
`labels`	labels for each of the objects being clustered.
`call`	the call which produced the result.
`method`	the cluster method that has been used.
`dist.method`	the distance that has been used to create d (only returned if the distance object has a "method" attribute).


Cluster Dendrogram
Height
0
1
2
3
4
Iris-setosa
Iris-virginica
Iris-versicolor


Cluster Dendrogram
Height
0
1
2
3
4
Iris-setosa
Iris-virginica
Iris-versicolor


Cluster Dendrogram
Height
0
1
2
3
4
Iris-setosa
Iris-virginica
Iris-versicolor
dist(irisSample)
hclust (*, "average")


Cluster Dendrogram
Height
0
1
2
3
4
Iris-setosa
Iris-virginica
Iris-versicolor
dist(irisSample)
hclust (*, "average")

Reference:

- R and Data Mining. Yangchan Zhao. Academic Press 2012. https://www.sciencedirect.com/book/9780123969637/r-and-data-mining