# Clustering.R

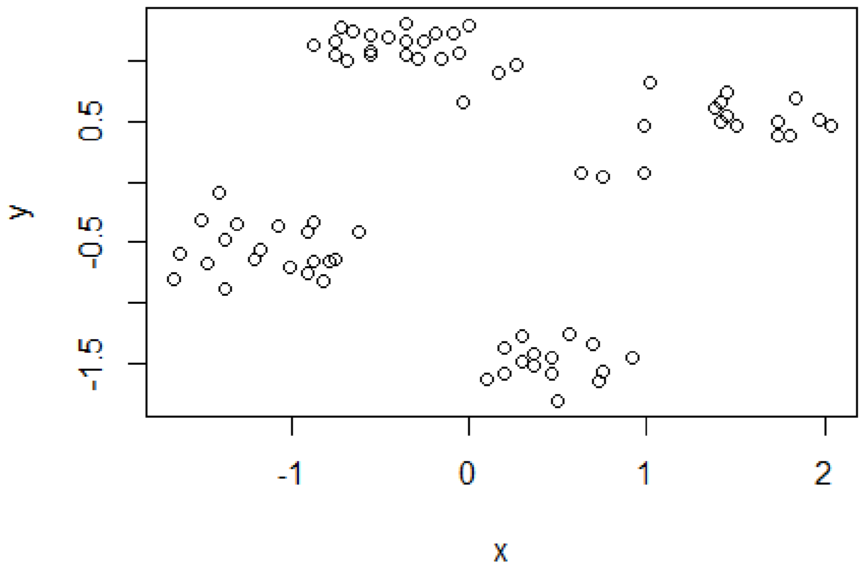
Wow

Wed Nov 21 08:45:12 2018

```
###################################################
#                   Clustering
###################################################
library(cluster)
# Ruspini data is in package cluster. It is a very simple data set with well
separated clusters
data(ruspini, package="cluster")
# Shuffle rows
ruspini <- ruspini[sample(1:nrow(ruspini)),]
head(ruspini)
```

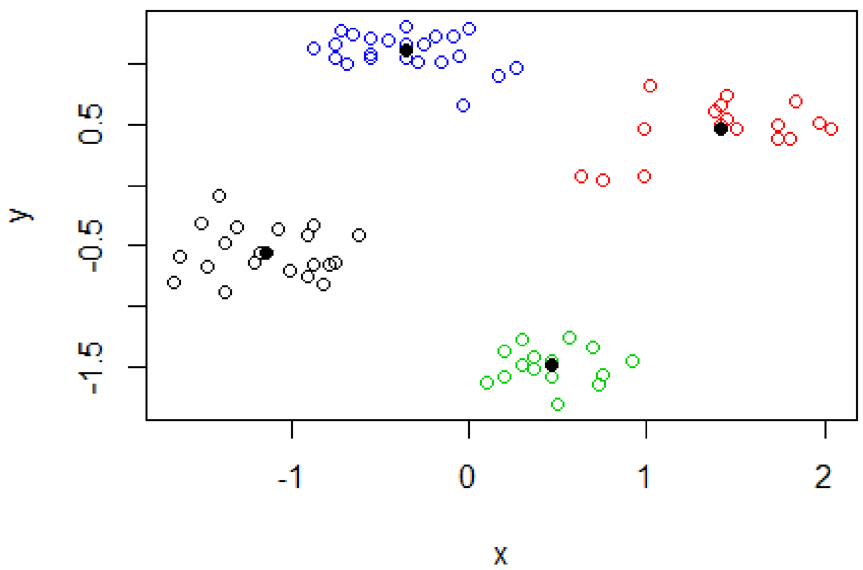
```
##      x   y
## 63  83  21
## 24  33 154
## 74  72  31
## 11  22  74
## 27  38 145
## 57 108 116
```

```
# Scale each column in the data to zero mean and unit standard deviation (z-
scores).
# This prevents one attribute with a large range to dominate the others for
the distance
# calculation.
ruspini <- scale(ruspini)
plot(ruspini)
```

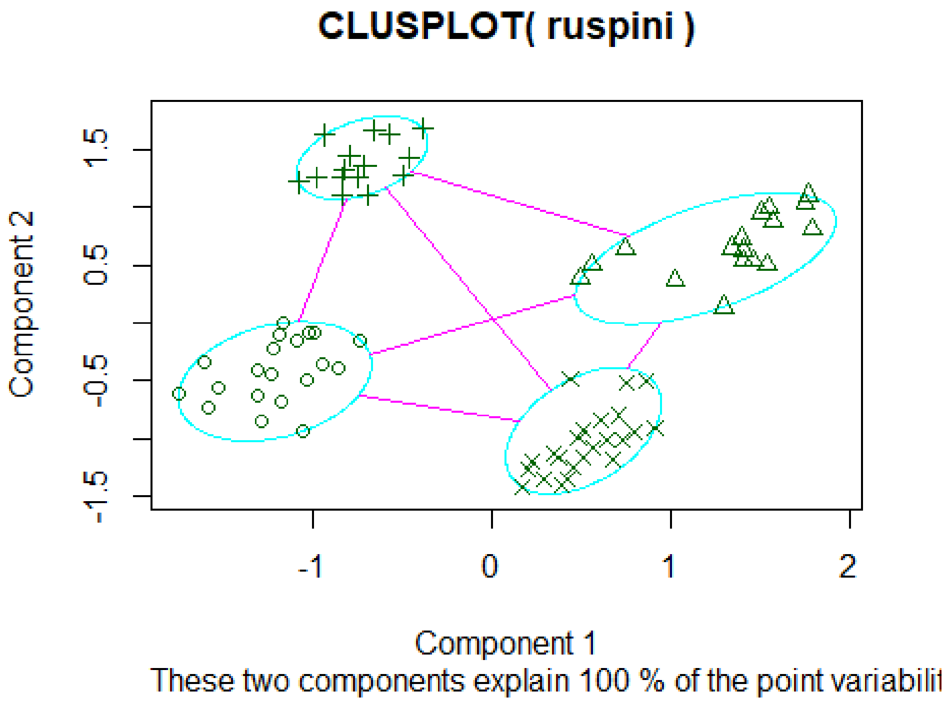



```
############### k-means clustering ###############
# We use k=4 clusters and run the algorithm 10 times with random initialized
centroids.
# The best result is returned
km <- kmeans(ruspini, centers=4, nstart=10)
km$cluster

## 63 24 74 11 27 57  1 62 68 39 36 38 25 45 54 13 69 56 53  7 46 10  9 59 19
##  3  4  3  1  4  2  1  3  3  4  4  4  4  2  2  1  3  2  2  1  2  1  1  2  1
## 44  6 15 73 41 51 66 43  8 33 64 20  3 47 67 65 49 40 22 60 26 71 70 75 34
##  2  1  1  3  4  2  3  4  1  4  3  1  1  2  3  3  2  4  4  2  4  3  3  3  4
## 18 28 35 52 29 72 50 16  4 48  5 23 30 61 37 31 21 14 32 17 12 42 55 58  2
##  1  4  4  2  4  3  2  1  1  2  1  4  4  3  4  4  4  1  4  1  1  4  2  2  1

km$centers #Centroids

##            x          y
## 1 -1.1385941 -0.5559591
## 2  1.4194387  0.4692907
## 3  0.4607268 -1.4912271
## 4 -0.3595425  1.1091151

km$withinss #The within cluster sum of squares.

## [1] 2.705477 3.641276 1.082373 2.658679

km$tot.withinss #= sum($withinss)
```

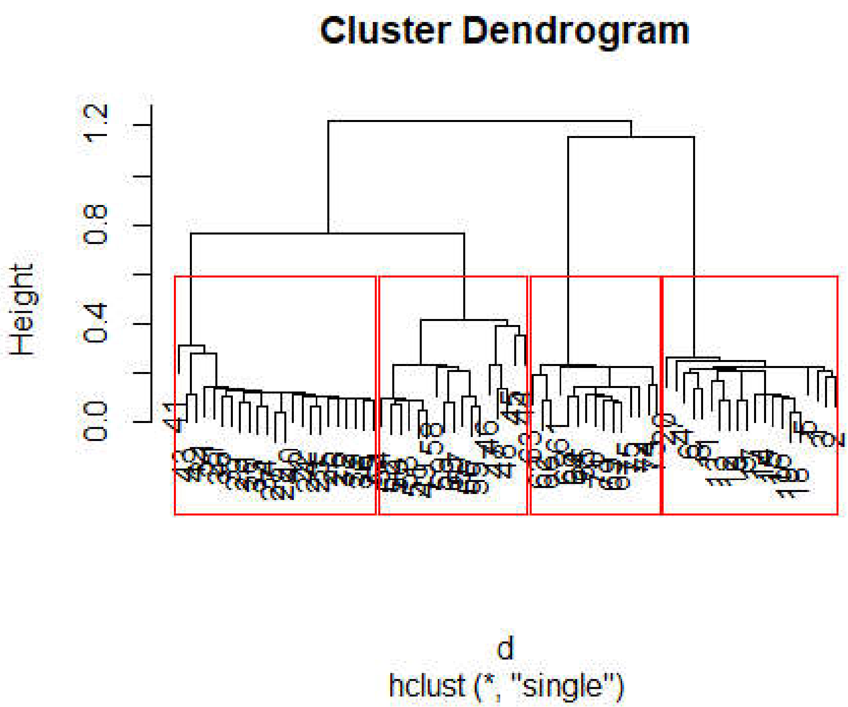
```
## [1] 10.08781
km$betweenss #The between clusters sum of squares.
## [1] 137.9122
km$totss #= $tot.withinss + $betweenss
## [1] 148
km$size
## [1] 20 17 15 23
km$iter
## [1] 2
plot(ruspini, col=km$cluster)
points(km$centers[,1],km$centers[,2],pch=16)
```



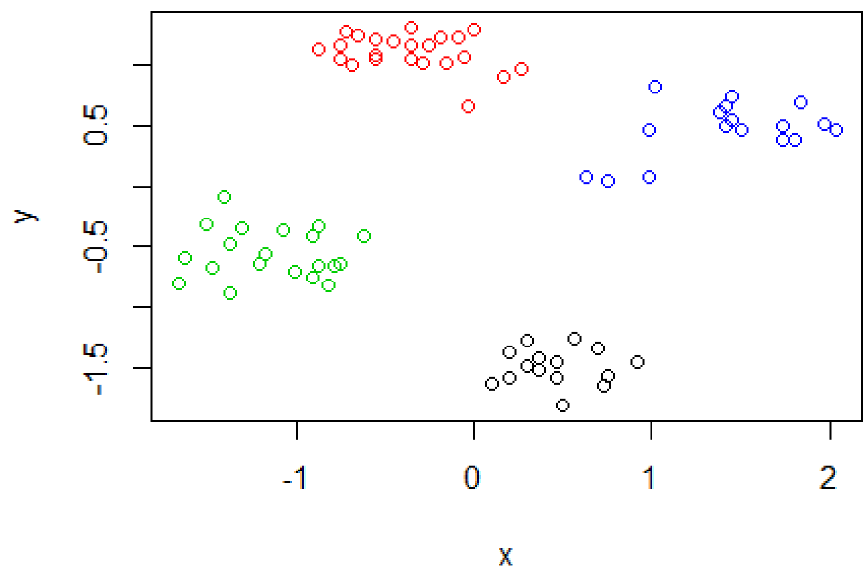

```
clusplot(ruspini, km$cluster) #from cluster package
```



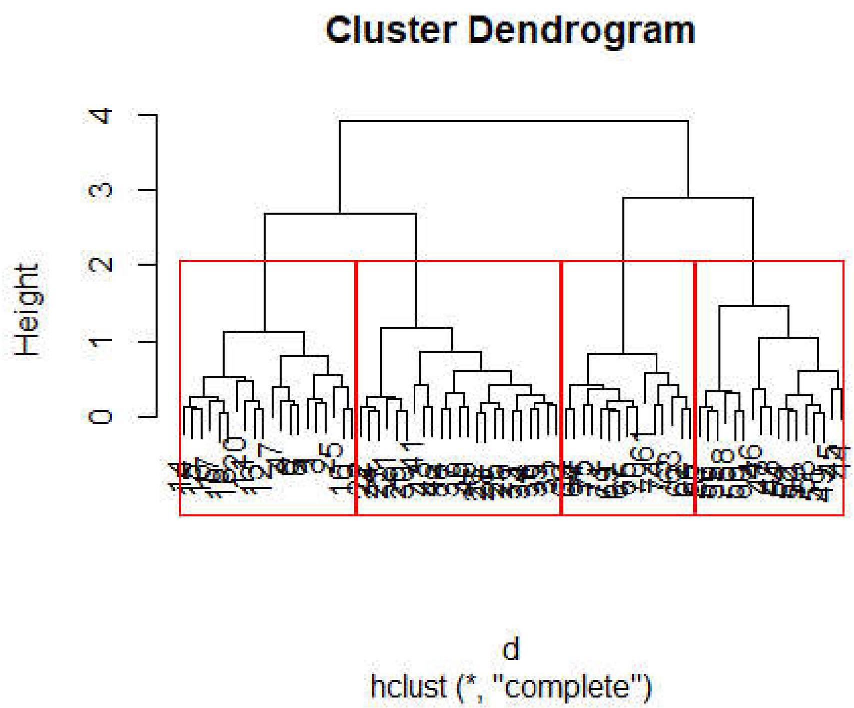

```
############## Hierarchical clustering ###############
# Distance measure default for "dist" function is Euclidean Distance
d <- dist(ruspini)
hc.sin <- hclust(d, method="single") #single link (min)
d.sin <- cophenetic(hc.sin)
c.sin <- cor(d,d.sin) #correlation
c.sin

## [1] 0.8475144

plot(hc.sin)
rect.hclust(hc.sin, k=4)
```



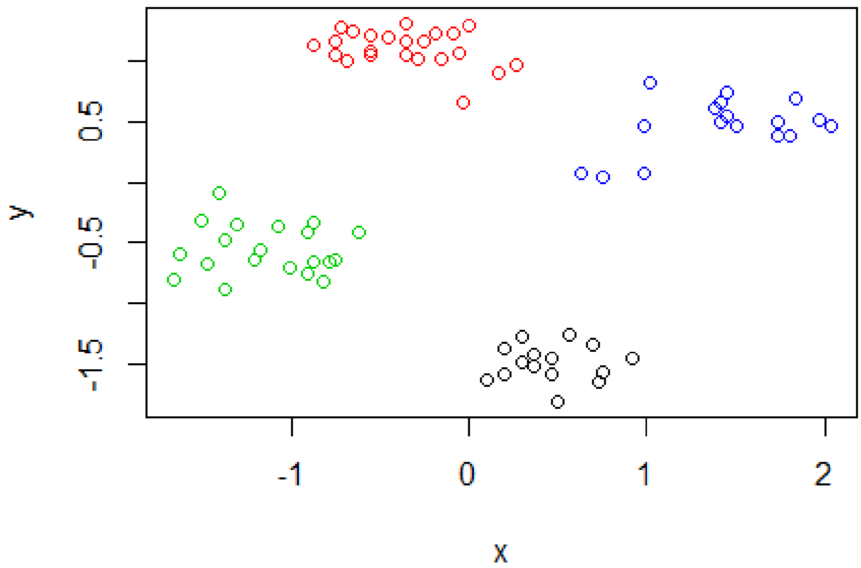

```
cluster.sin <- cutree(hc.sin, k=4)
plot(ruspini, col=cluster.sin)
```



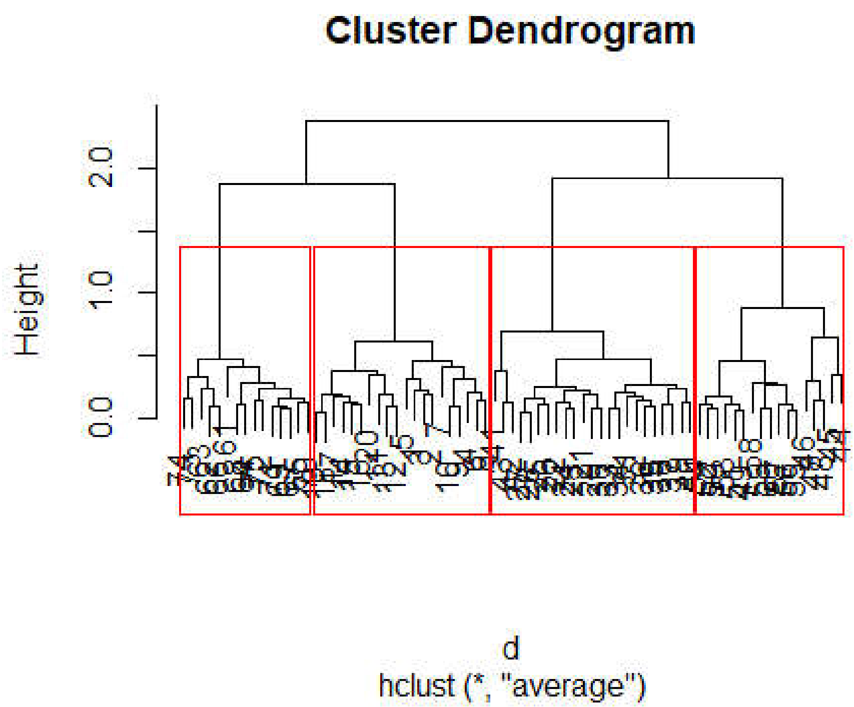
```
hc.com <- hclust(d, method="complete") #complete link (max)
d.com <- cophenetic(hc.com)
c.com <- cor(d,d.com) #correlation
c.com

## [1] 0.8432173

plot(hc.com)
rect.hclust(hc.com, k=4)
```



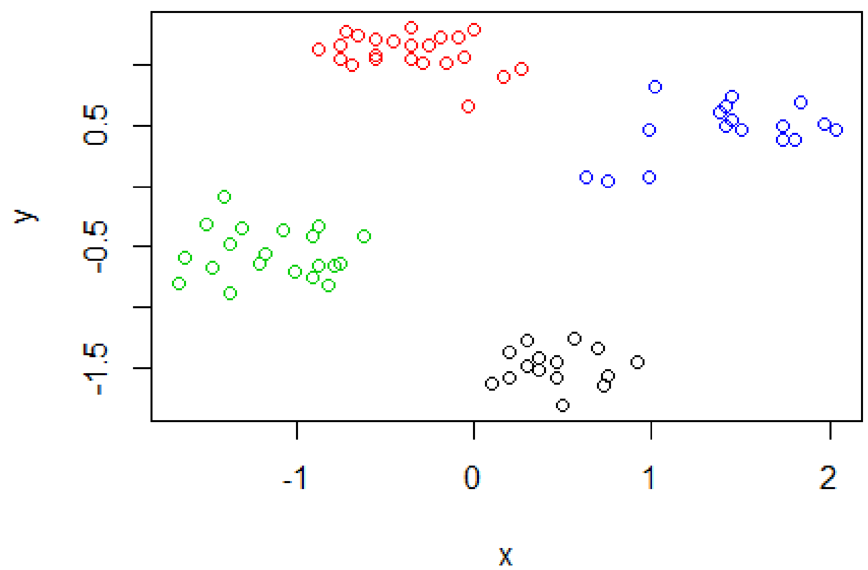

```
cluster.com <- cutree(hc.com, k=4)
plot(ruspini, col=cluster.com)
```



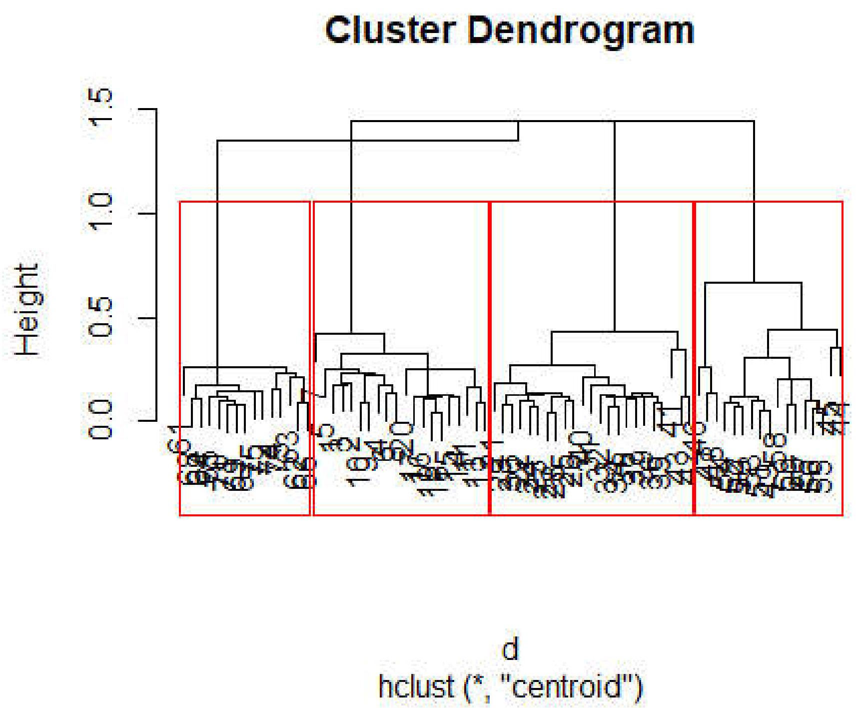

```
hc.avg <- hclust(d, method="average") #group average
d.avg <- cophenetic(hc.avg)
c.avg <- cor(d,d.avg) #correlation
c.avg

## [1] 0.8761993

plot(hc.avg)
rect.hclust(hc.avg, k=4)
```



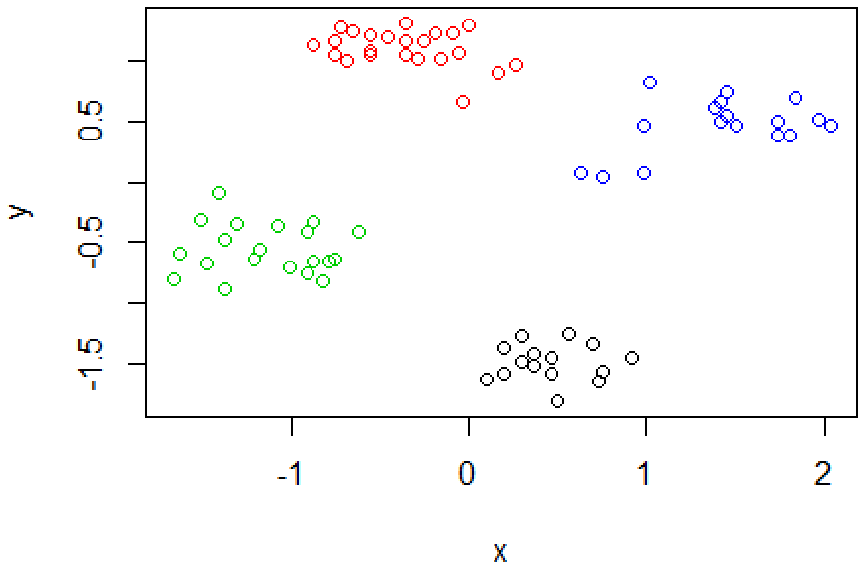

```
cluster.avg <- cutree(hc.avg, k=4)
plot(ruspini, col=cluster.avg)
```



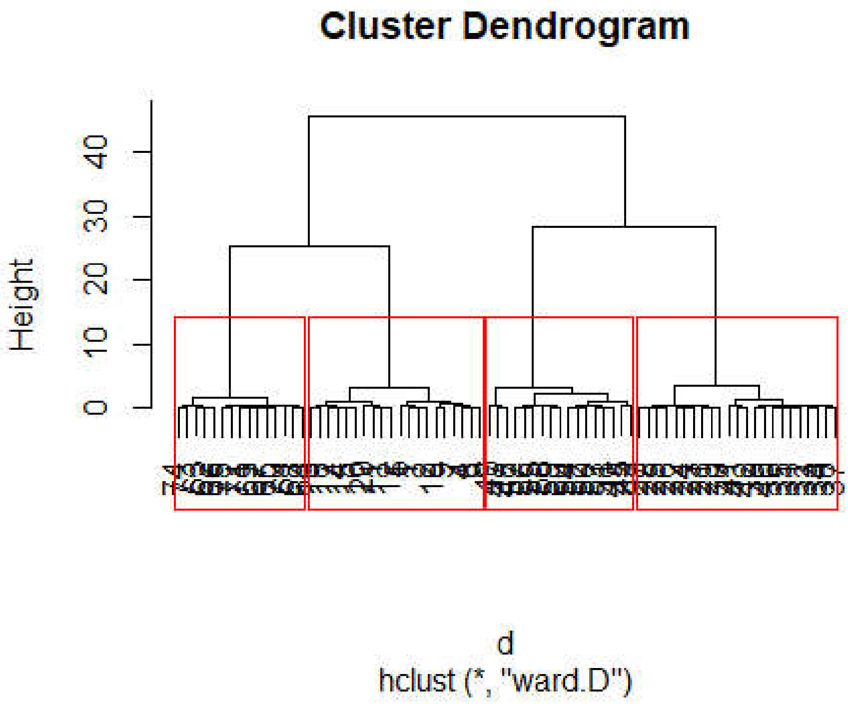
```
hc.cen <- hclust(d, method="centroid") #Centroid method
d.cen <- cophenetic(hc.cen)
c.cen <- cor(d,d.cen) #correlation
c.cen

## [1] 0.839942

plot(hc.cen)
rect.hclust(hc.cen, k=4)
```



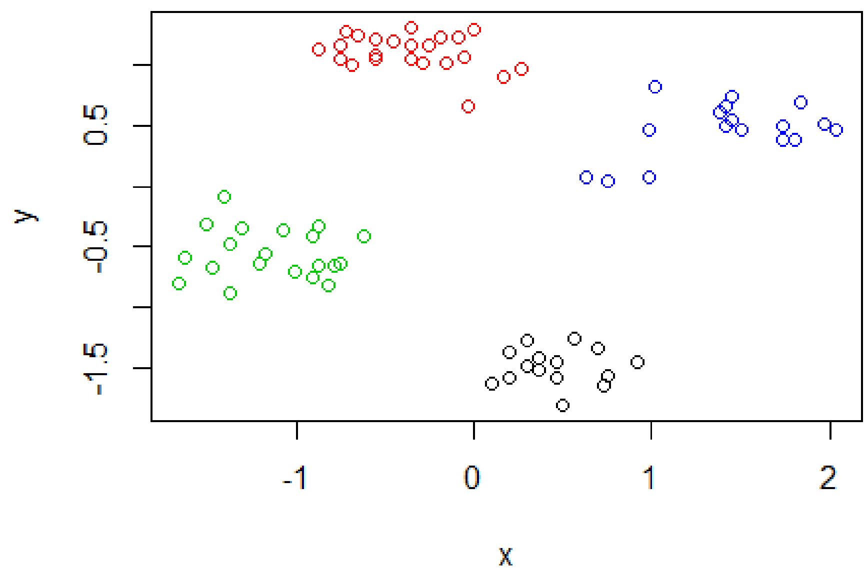

```
cluster.cen <- cutree(hc.cen, k=4)
plot(ruspini, col=cluster.cen)
```



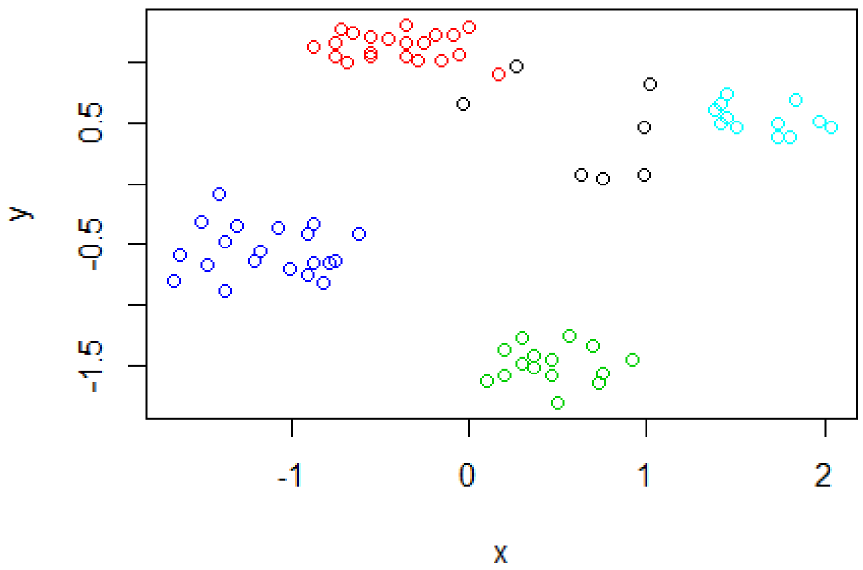

```r
hc.war <- hclust(d, method="ward.D") #Ward's method
d.war <- cophenetic(hc.war)
c.war <- cor(d,d.war) #correlation
c.war

## [1] 0.8555617

plot(hc.war)
rect.hclust(hc.war, k=4)
```



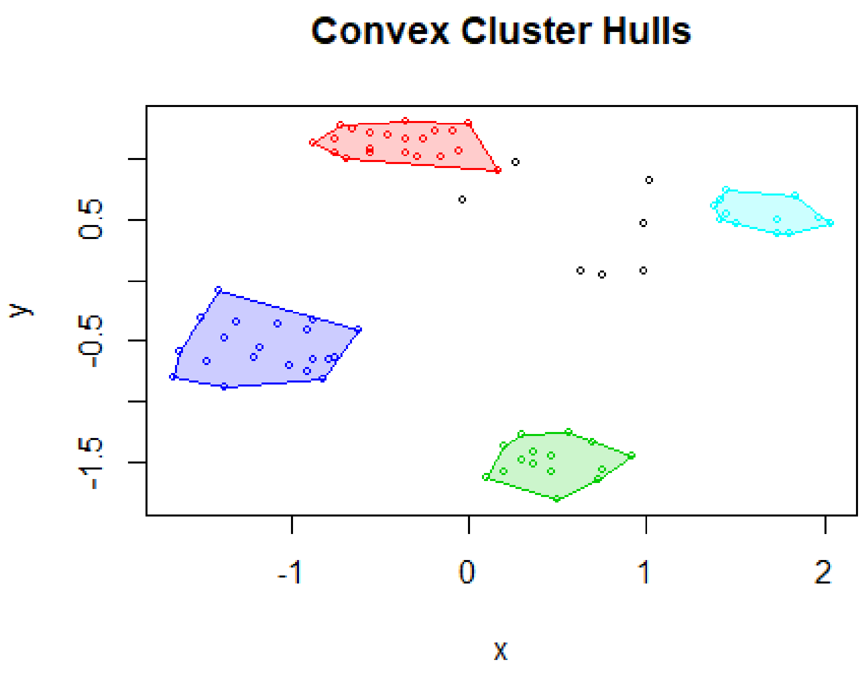

```
cluster.war <- cutree(hc.war, k=4)
plot(ruspini, col=cluster.war)
```



```
# The higher correlation, the better method
cbind(c("Single","Complete","Average","Centroid","Ward"),c(c.sin,c.com,c.avg,
c.cen,c.war))

##      [,1]       [,2]
## [1,] "Single"   "0.847514392831746"
## [2,] "Complete" "0.843217284138101"
## [3,] "Average"  "0.876199312423294"
## [4,] "Centroid" "0.83994198472377"
## [5,] "Ward"     "0.855561670394014"

################# DBSCAN ##################
library(dbscan)
db <- dbscan(ruspini, eps=0.3, minPts=5)
db

## DBSCAN clustering for 75 objects.
## Parameters: eps = 0.3, minPts = 5
## The clustering contains 4 cluster(s) and 7 noise points.
##
##  0  1  2  3  4
##  7 21 15 20 12
##
## Available fields: cluster, eps, minPts

plot(ruspini, col = db$cluster + 1) #Note: 0 is not a color so we add 1 to
cluster.
```



```
hullplot(ruspini, db)
predict(db, data = ruspini)

##  [1] 2 1 2 3 1 4 3 2 2 1 1 1 1 0 4 3 2 4 4 3 0 3 3 4 3 0 3 3 2 0 4 2 0 3 1
## [36] 2 3 3 0 2 2 4 1 1 4 1 2 2 2 1 3 1 1 4 1 2 4 3 3 0 3 1 1 2 1 1 1 3 1 3
## [71] 3 1 4 4 3

# Play with eps (neighborhood size) and MinPts (minimum of points needed for
core cluster) to find the best clusters.

####### Internal Cluster Validation #######
library(fpc)

##
## Attaching package: 'fpc'

## The following object is masked from 'package:dbscan':
##
##     dbscan
```




```
# within.cluster.ss = the within clusters sum of squares error (k-means
objective function)
# the avg.silwidth = average silhouette width
cluster.stats(d, km$cluster)

## $n
## [1] 75
##
```

```
## $cluster.number
## [1] 4
##
## $cluster.size
## [1] 20 17 15 23
##
## $min.cluster.size
## [1] 15
##
## $noisen
## [1] 0
##
## $diameter
## [1] 1.1192822 1.4627043 0.8359025 1.1591436
##
## $average.distance
## [1] 0.4824376 0.5805551 0.3564457 0.4286139
##
## $median.distance
## [1] 0.4492432 0.5023855 0.3379729 0.3934100
##
## $separation
## [1] 1.157682 0.767612 1.157682 0.767612
##
## $average.toother
## [1] 2.157193 2.293318 2.307969 2.148527
##
## $separation.matrix
##          [,1]     [,2]     [,3]     [,4]
## [1,] 0.000000 1.339721 1.157682 1.219930
## [2,] 1.339721 0.000000 1.308435 0.767612
## [3,] 1.157682 1.308435 0.000000 1.957726
## [4,] 1.219930 0.767612 1.957726 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]
## [1,] 0.000000 2.771960 1.874198 1.887363
## [2,] 2.771960 0.000000 2.220011 1.924915
## [3,] 1.874198 2.220011 0.000000 2.750174
## [4,] 1.887363 1.924915 2.750174 0.000000
##
## $average.between
## [1] 2.219257
##
## $average.within
## [1] 0.462697
##
## $n.between
## [1] 2091
##
```

```
## $n.within
## [1] 684
##
## $max.diameter
## [1] 1.462704
##
## $min.separation
## [1] 0.767612
##
## $within.cluster.ss
## [1] 10.08781
##
## $clus.avg.silwidths
##         1         2         3         4
## 0.7211353 0.6812849 0.8073733 0.7454551
##
## $avg.silwidth
## [1] 0.7368082
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.8415597
##
## $dunn
## [1] 0.5247896
##
## $dunn2
## [1] 3.228286
##
## $entropy
## [1] 1.37327
##
## $wb.ratio
## [1] 0.2084918
##
## $ch
## [1] 323.5512
##
## $cwidegap
## [1] 0.2611701 0.4149825 0.2351854 0.3152817
##
## $widestgap
## [1] 0.4149825
##
## $sindex
```

```
## [1] 0.8583457
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

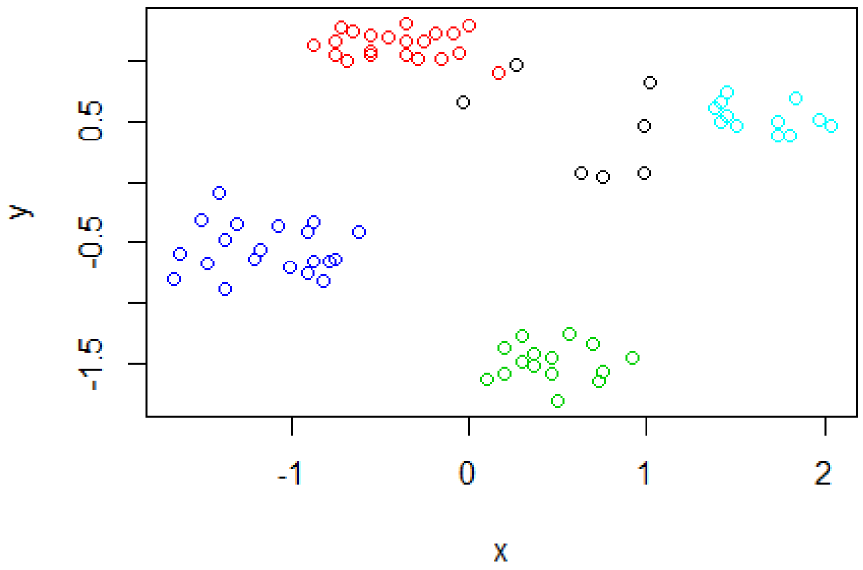
```r
sapply(list(
  km=km$cluster,
  hc_sing=cluster.sin,
  hc_comp=cluster.com,
  hc_aver=cluster.avg,
  hc_cent=cluster.cen,
  hc_ward=cluster.war),
  FUN=function(x)
    cluster.stats(d, x))[c("within.cluster.ss","avg.silwidth"),]
```

```
##                     km        hc_sing   hc_comp   hc_aver   hc_cent
## within.cluster.ss 10.08781  10.08781  10.08781  10.08781  10.08781
## avg.silwidth      0.7368082 0.7368082 0.7368082 0.7368082 0.7368082
##                     hc_ward
## within.cluster.ss 10.08781
## avg.silwidth      0.7368082
```

```r
# Internal validation for DBSCAN
# Remove outliers first and then apply cluster.stats function
db
```

```
## DBSCAN clustering for 75 objects.
## Parameters: eps = 0.3, minPts = 5
## The clustering contains 4 cluster(s) and 7 noise points.
##
##  0  1  2  3  4
##  7 21 15 20 12
##
## Available fields: cluster, eps, minPts
```

```r
plot(ruspini, col = db$cluster + 1) #Note: 0 is not a color so we add 1 to
cluster.
```

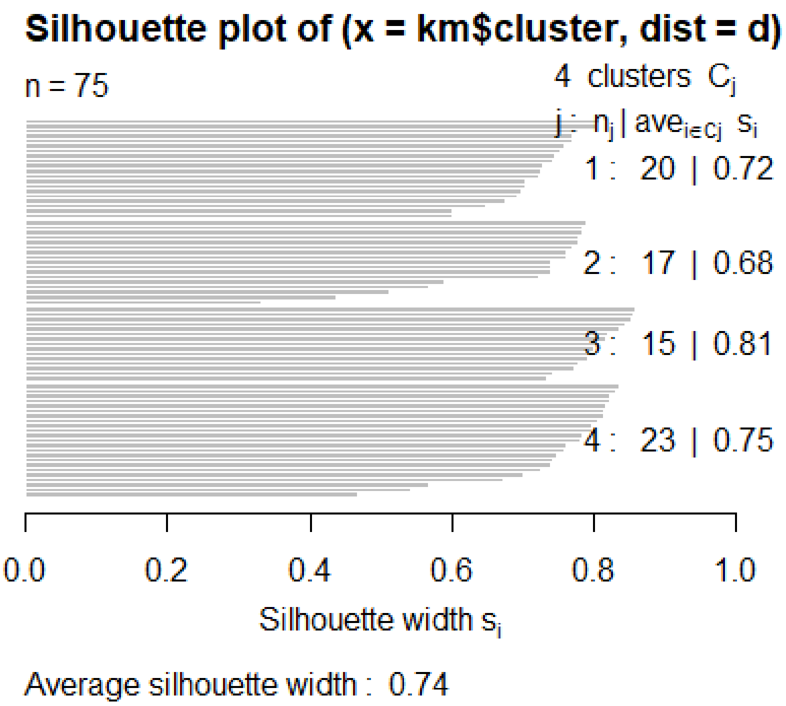



```
db.clus <- predict(db, data = ruspini)
db.clus

##  [1] 2 1 2 3 1 4 3 2 2 1 1 1 1 0 4 3 2 4 4 3 0 3 3 4 3 0 3 3 2 0 4 2 0 3 1
## [36] 2 3 3 0 2 2 4 1 1 4 1 2 2 2 1 3 1 1 4 1 2 4 3 3 0 3 1 1 2 1 1 1 3 1 3
## [71] 3 1 4 4 3

# From the plot and summary results outliers are cluster 0
outs <- which(predict(db, data = ruspini) == 0) #find location of points with
label 0
db.mod <- db.clus[-outs] #delete outliers
db.mod

##  [1] 2 1 2 3 1 4 3 2 2 1 1 1 1 4 3 2 4 4 3 3 3 4 3 3 3 2 4 2 3 1 2 3 3 2 2
## [36] 4 1 1 4 1 2 2 2 1 3 1 1 4 1 2 4 3 3 3 1 1 2 1 1 1 3 1 3 3 1 4 4 3

d.mat <- as.matrix(d) #change d into matrix form
d.mod <- d.mat[(1:nrow(ruspini))[-outs],(1:nrow(ruspini))[-outs]] #delete
rows and columns of points with label 0
d.mod <- as.dist(d.mod) #change d.mod into distance form

cluster.stats(d.mod, db.mod)

## $n
## [1] 68
##
## $cluster.number
## [1] 4
```

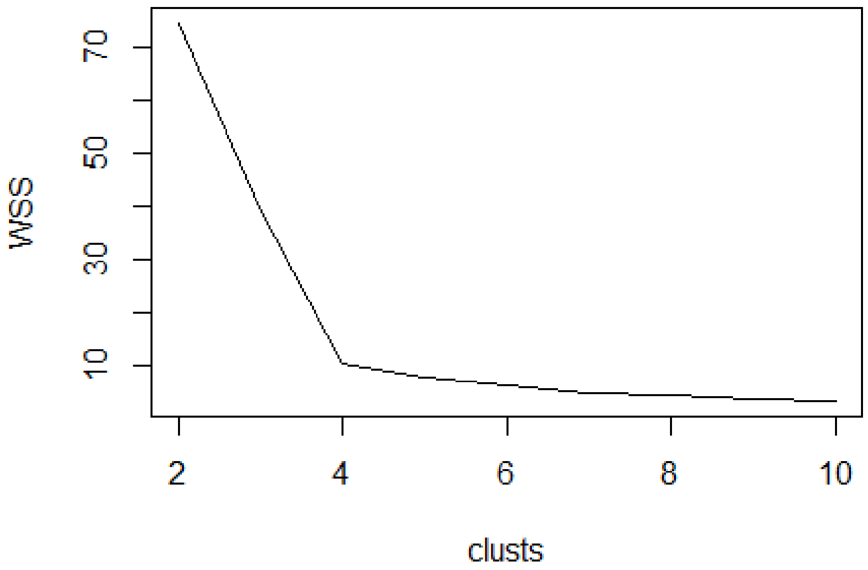
```
##
## $cluster.size
## [1] 21 15 20 12
##
## $min.cluster.size
## [1] 12
##
## $noisen
## [1] 0
##
## $diameter
## [1] 1.0731308 0.8359025 1.1192822 0.6712516
##
## $average.distance
## [1] 0.3780364 0.3564457 0.4824376 0.3307118
##
## $median.distance
## [1] 0.3309896 0.3379729 0.4492432 0.3165229
##
## $separation
## [1] 1.246610 1.157682 1.157682 1.246610
##
## $average.toother
## [1] 2.240535 2.349236 2.163048 2.511712
##
## $separation.matrix
##          [,1]     [,2]     [,3]     [,4]
## [1,] 0.000000 2.180421 1.305551 1.246610
## [2,] 2.180421 0.000000 1.157682 1.942315
## [3,] 1.305551 1.157682 0.000000 2.224353
## [4,] 1.246610 1.942315 2.224353 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]
## [1,] 0.000000 2.788872 1.888332 2.142121
## [2,] 2.788872 0.000000 1.874198 2.371602
## [3,] 1.888332 1.874198 0.000000 3.004864
## [4,] 2.142121 2.371602 3.004864 0.000000
##
## $average.between
## [1] 2.297436
##
## $average.within
## [1] 0.4033355
##
## $n.between
## [1] 1707
##
## $n.within
## [1] 571
```

```
##
## $max.diameter
## [1] 1.119282
##
## $min.separation
## [1] 1.157682
##
## $within.cluster.ss
## [1] 6.418682
##
## $clus.avg.silwidths
##         1         2         3         4
## 0.7897897 0.8083929 0.7207724 0.8448691
##
## $avg.silwidth
## [1] 0.783314
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.8681682
##
## $dunn
## [1] 1.034307
##
## $dunn2
## [1] 3.88485
##
## $entropy
## [1] 1.362313
##
## $wb.ratio
## [1] 0.175559
##
## $ch
## [1] 448.0998
##
## $cwidegap
## [1] 0.2822186 0.2351854 0.2611701 0.2304059
##
## $widestgap
## [1] 0.2822186
##
## $sindex
## [1] 1.193102
##
```

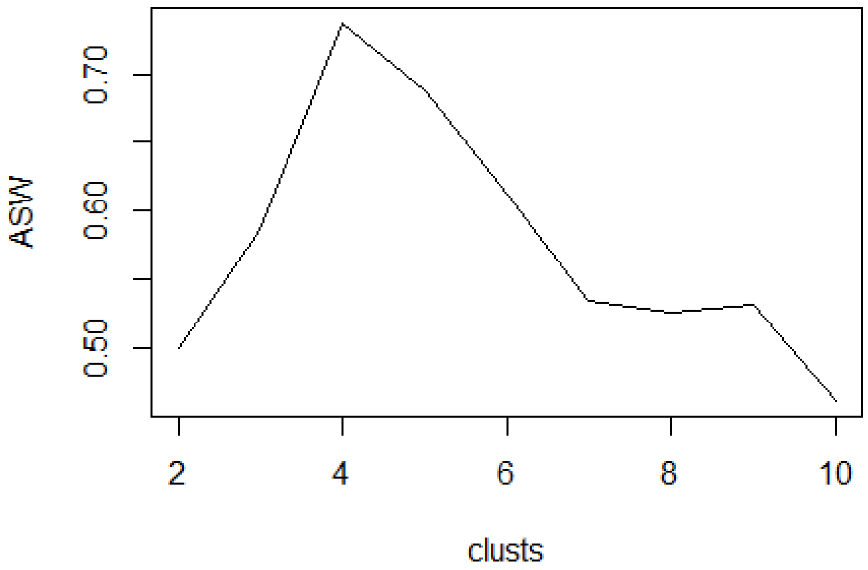
```
## $corrected.rand
## NULL
##
## $vi
## NULL

################## Silhouette Plot ####################
plot(silhouette(km$cluster, d))
```



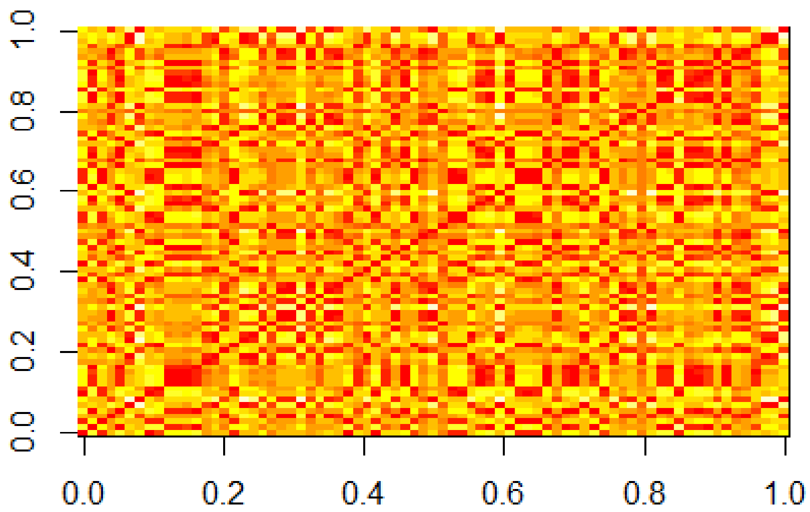

```
##### Find optimal number of clusters for k-means #####
clusts <- 2:10

# Use within sum of squares (look for the knee)
WSS <- sapply(clusts, FUN=function(k) {
  kmeans(ruspini, centers=k, nstart=5)$tot.withinss
})
plot(clusts, WSS, type="l")
```



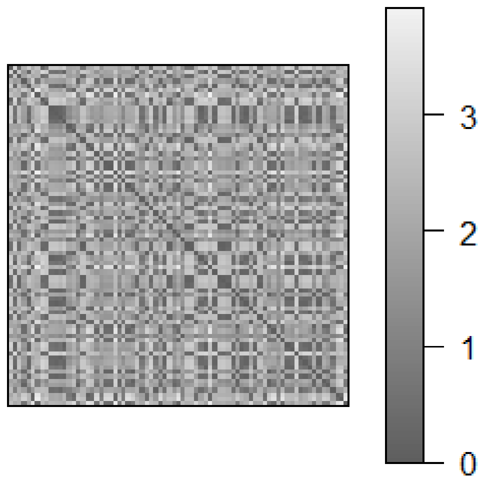

```r
# Use average silhouette width (look for the max)
ASW <- sapply(clusts, FUN=function(k) {
  cluster.stats(d, kmeans(ruspini, centers=k, nstart=5)$cluster)$avg.silwidth
})
plot(clusts, ASW, type="l")
```



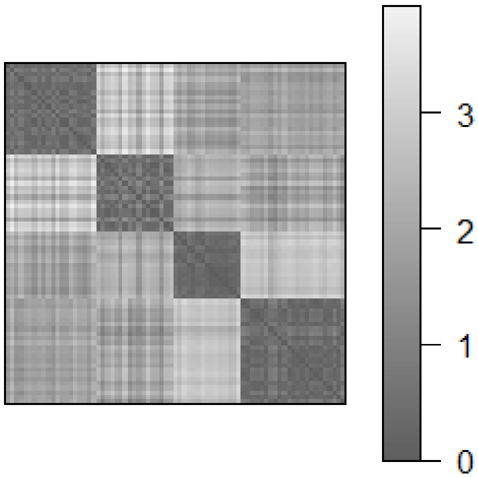

```
clusts[which.max(ASW)]

## [1] 4

########### Visualize the Distance Matrix #############
image(as.matrix(d))
```



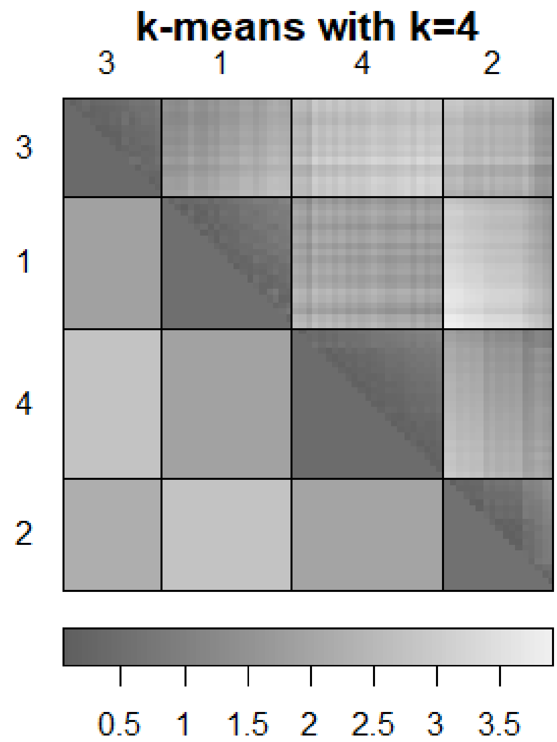

```
library(seriation)
pimage(d, colorkey=TRUE)
```



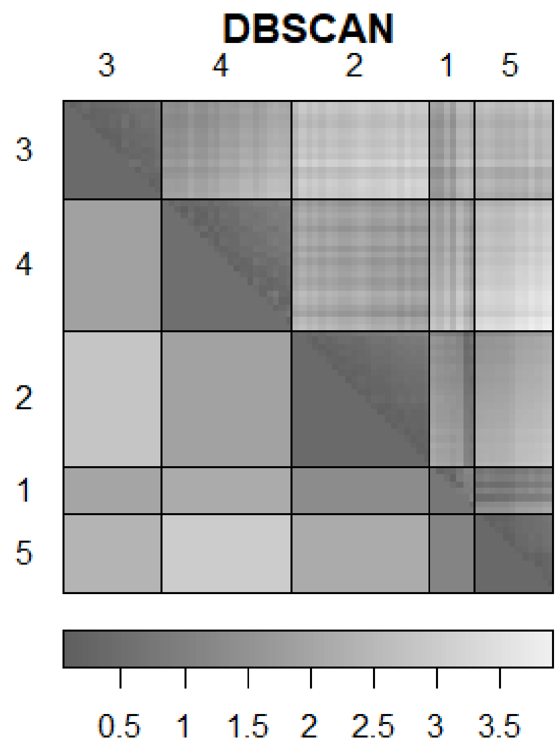
```r
# Reorder using cluster labels
pimage(d, order=order(km$cluster), colorkey=TRUE)
```



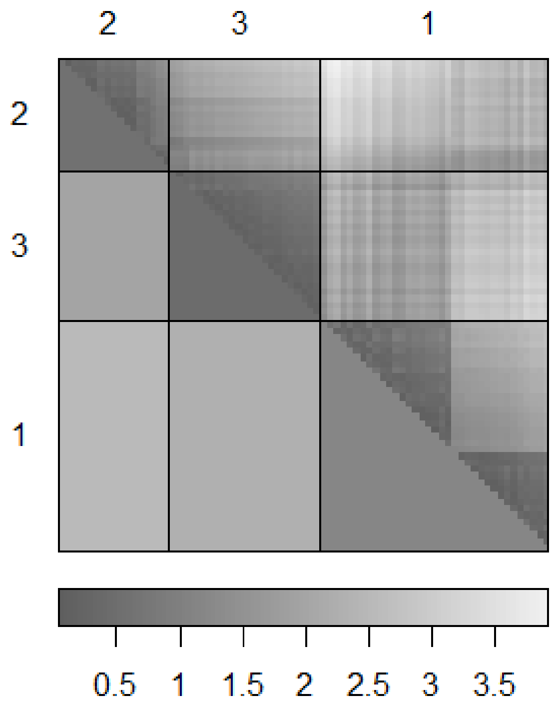

```r
# Use dissplot
dissplot(d, labels=km$cluster, options=list(main="k-means with k=4"))
```



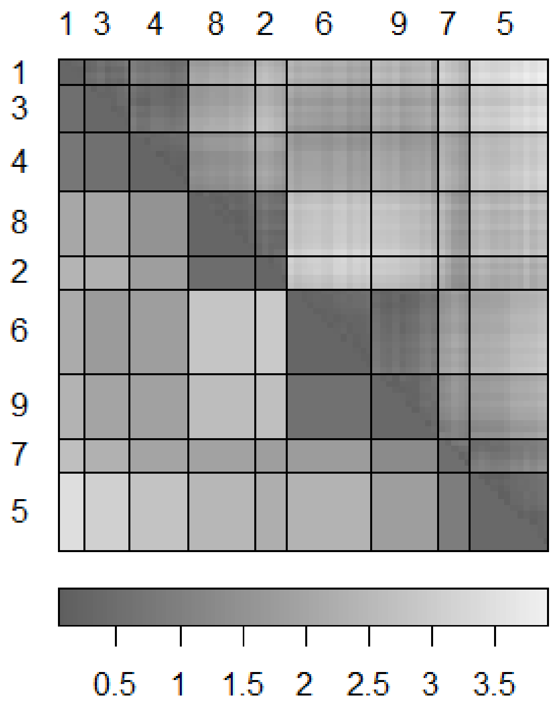

```
dissplot(d, labels=db$cluster+1L, options=list(main="DBSCAN"))
```



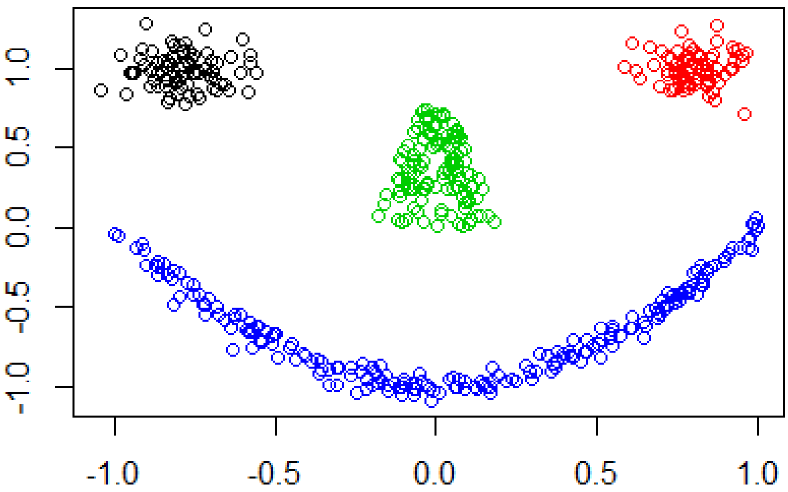
```r
# Misspecified k
dissplot(d, labels=kmeans(ruspini, centers=3)$cluster)
```



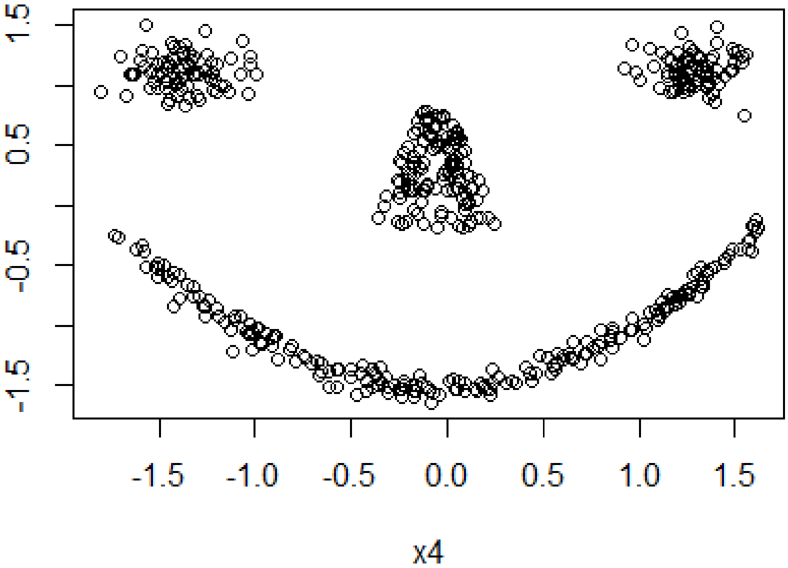

```r
dissplot(d, labels=kmeans(ruspini, centers=9)$cluster)
```



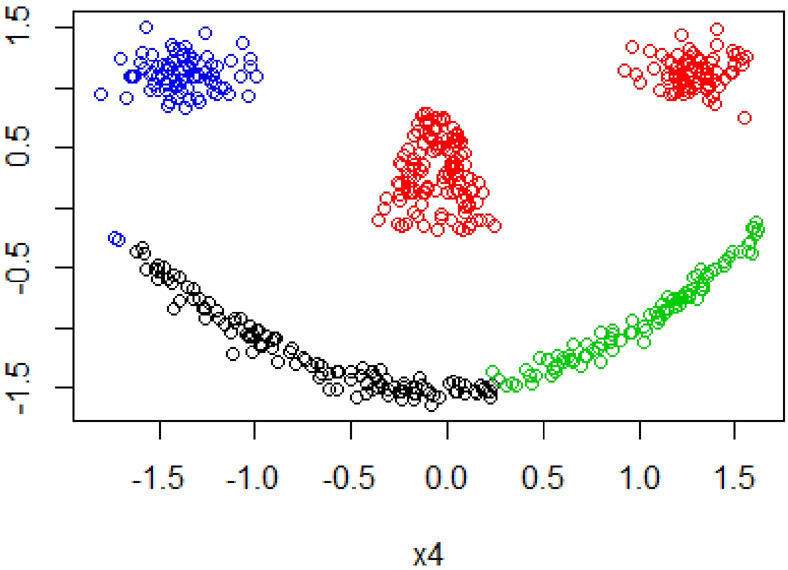

```
########### External cluster validation #############
# This uses the ground truth so we use an artifical data set with known
groups
library(mlbench)
shapes <- mlbench.smiley(n=500, sd1 = 0.1, sd2 = 0.05)
plot(shapes)
```



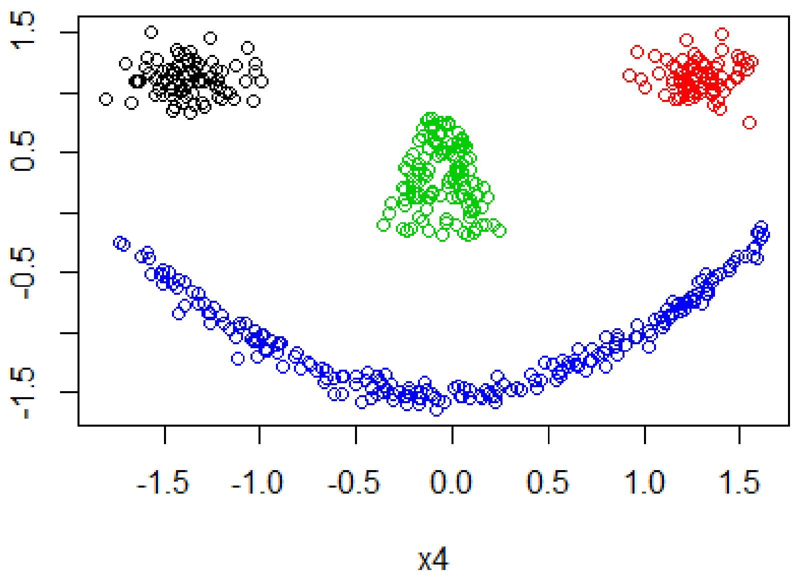

```r
# Prepare data
truth <- as.integer(shapes$class)
shapes <- scale(shapes$x)
plot(shapes)
```




```r
# k-means
km <- kmeans(shapes, centers=4)
plot(shapes, col=km$cluster)
```



```r
# Hierachical clustering
d <- dist(shapes)
hc <- cutree(hclust(d, method="single"), k=4)
plot(shapes, col=hc)
```




```r
# Compare with ground truth (look at corrected.rand)
cbind(
  cluster.stats(d, km$cluster, truth),
  cluster.stats(d, hc, truth)
)

##                      [,1]       [,2]
## n                    500        500
## cluster.number       4          4
## cluster.size         Numeric,4  Numeric,4
## min.cluster.size     85         83
## noisen               0          0
## diameter             Numeric,4  Numeric,4
## average.distance     Numeric,4  Numeric,4
## median.distance      Numeric,4  Numeric,4
## separation           Numeric,4  Numeric,4
## average.toother      Numeric,4  Numeric,4
## separation.matrix    Numeric,16 Numeric,16
## ave.between.matrix   Numeric,16 Numeric,16
## average.between      2.175435   2.154603
## average.within       0.8108392  0.8786488
## n.between            88911      88458
```

```
## n.within             35839      36292
## max.diameter         2.36325    3.350278
## min.separation       0.04843945 0.865681
## within.cluster.ss    219.7007   248.2025
## clus.avg.silwidths Numeric,4  Numeric,4
## avg.silwidth         0.5437689  0.5692807
## g2                   NULL       NULL
## g3                   NULL       NULL
## pearsongamma         0.6821088  0.6401812
## dunn                 0.02049696 0.2583908
## dunn2                1.8845     1.267372
## entropy              1.315598   1.307379
## wb.ratio             0.372725   0.4078008
## ch                   585.7007   499.4571
## cwidegap             Numeric,4  Numeric,4
## widestgap            1.129718   0.1969602
## sindex               0.2317201  0.9763586
## corrected.rand       0.5800392  1
## vi                   0.6057401  0

# Read ?cluster.stats for an explanation of all the available indices.
```