Clustering

Minería de datos: aprendizaje no supervisado y detección de anomalías

Francisco Luque Sánchez

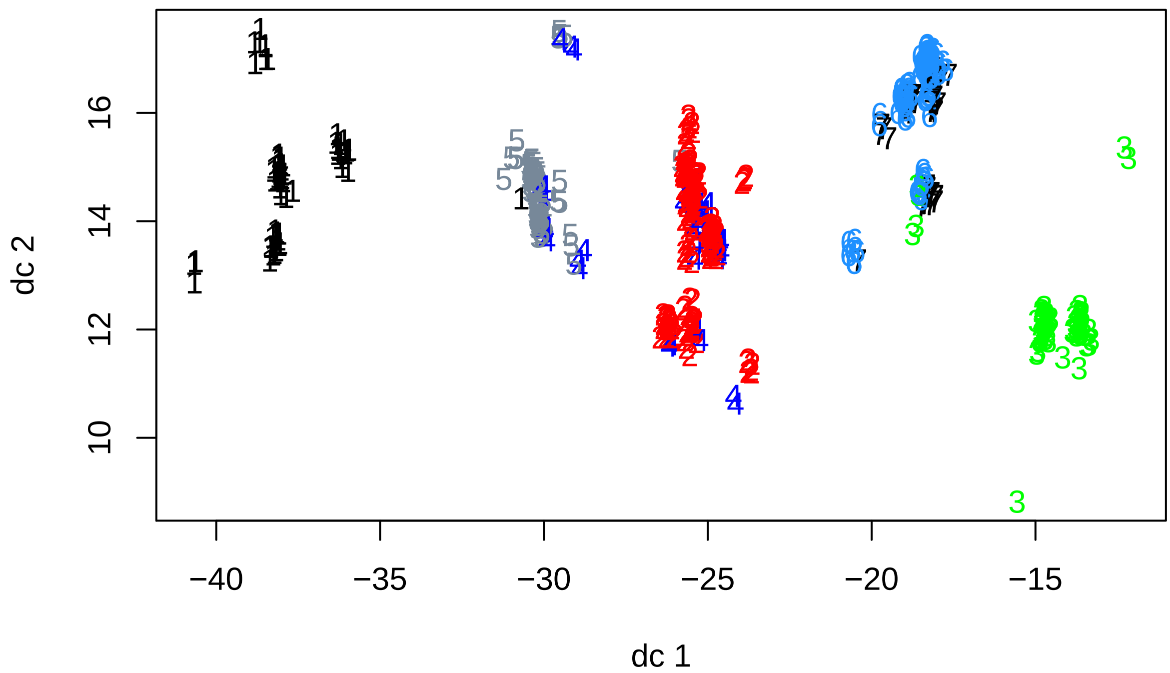
21/12/2019

```
read.dataset <- function(){
    dataset <- read.csv("dataset/Absenteeism_at_work.csv", sep = ";")

    ## Eliminamos columnas que no aportan información
    dataset$ID <- NULL
    dataset$Body.mass.index <- NULL # Este valor es función del peso y la
↪   altura
    dataset$Seasons <- NULL # Este valor codifica la misma información que
↪   el mes
    dataset$Reason.for.absence <- NULL # A priori no contemplaremos este
↪   valor

    ## Renombramos columnas con nombres largos
    dataset <- dataset %>%
        rename(
            Month = Month.of.absence,
            Day = Day.of.the.week,
            Transport.exp = Transportation.expense,
            Distance = Distance.from.Residence.to.Work,
            Workload = Work.load.Average.day,
            Absent.time = Absenteeism.time.in.hours
        )
    dataset
}

dataset <- read.dataset()

continuous.vars <- c(
    "Month", "Day", "Transport.exp", "Distance", "Service.time",
    "Age", "Workload", "Hit.target", "Education", "Son", "Pet",
    "Weight", "Height", "Absent.time"
)
binary.vars <- c("Disciplinary.failure", "Social.drinker", "Social.smoker")
```

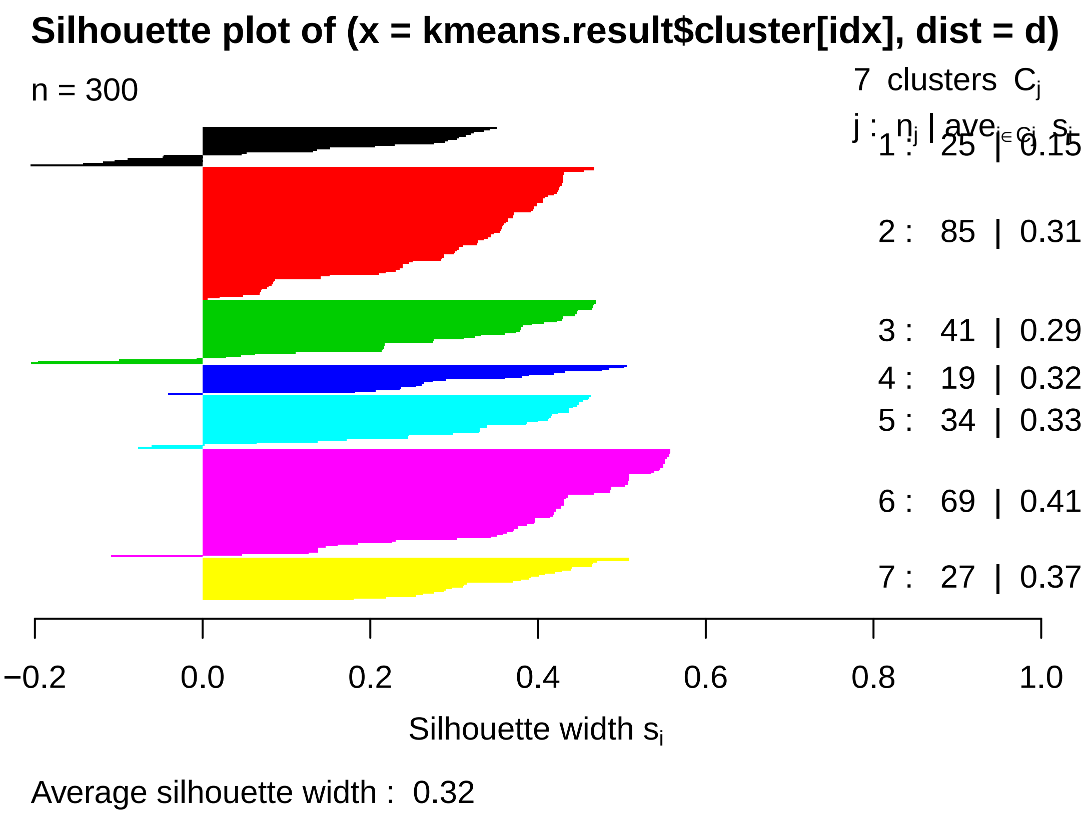
0.1 K-medias

```
nclust <- 7

dataset <- read.dataset()
continuous.data <- dataset %>% select(continuous.vars)
```

```r
binary.data <- dataset %>% select(binary.vars)

continuous.distances <- dist(continuous.data)
binary.distances <- dist(binary.data, method="binary")

final.distances <- (continuous.distances + binary.distances) / 2

kmeans.result <- kmeans(final.distances, nclust)

idx <- sample(1: dim(dataset)[1], 300)

plotcluster(dataset, kmeans.result$cluster)
```




```r
d1 <- dist(continuous.data[idx,])
d2 <- dist(binary.data[idx,], method="binary")

d <- (d1+d2)/2
sil <- silhouette(kmeans.result$cluster[idx], d)
plot(sil, col = 1:nclust)
```




```
cluster.stats(final.distances, kmeans.result$cluster)
```

```
## $n
## [1] 740
##
## $cluster.number
## [1] 7
##
## $cluster.size
## [1]  67 212  97  54  84 169  57
##
## $min.cluster.size
## [1] 54
##
## $noisen
## [1] 0
##
## $diameter
```

```
## [1] 107.98038  54.37913  91.19339 103.98421  57.79703  52.71997  50.84855
##
## $average.distance
## [1] 33.40221 22.63456 25.41323 32.05506 22.76055 20.61541 26.42731
##
## $median.distance
## [1] 28.19390 22.41651 20.00053 28.43870 21.51405 22.36457 27.26250
##
## $separation
## [1] 19.301596  8.589068 11.213960  6.650684  6.650684  5.918986  5.918986
##
## $average.toother
## [1] 83.26327 49.29113 68.86520 60.69396 55.09517 52.19018 57.57378
##
## $separation.matrix
##          [,1]      [,2]     [,3]      [,4]      [,5]      [,6]      [,7]
## [1,]  0.00000 41.557886 75.02502 24.073182 19.301596 65.689413 72.017598
## [2,] 41.55789  0.000000 37.64642  9.068405  8.589068 18.377976 18.553975
## [3,] 75.02502 37.646419  0.00000 44.659960 59.199474 11.213960 22.226111
## [4,] 24.07318  9.068405 44.65996  0.000000  6.650684 22.195910 21.594510
## [5,] 19.30160  8.589068 59.19947  6.650684  0.000000 37.772644 37.755872
## [6,] 65.68941 18.377976 11.21396 22.195910 37.772644  0.000000  5.918986
## [7,] 72.01760 18.553975 22.22611 21.594510 37.755872  5.918986  0.000000
##
## $ave.between.matrix
##           [,1]     [,2]      [,3]     [,4]     [,5]     [,6]      [,7]
## [1,]   0.00000 67.34230 124.31316 71.26770 46.07213 95.42966 102.72128
## [2,]  67.34230  0.00000  65.77676 49.89154 34.77757 38.38316  53.17939
## [3,] 124.31316 65.77676   0.00000 83.29683 89.51458 42.38401  49.58802
## [4,]  71.26770 49.89154  83.29683  0.00000 50.42012 65.31600  51.41452
## [5,]  46.07213 34.77757  89.51458 50.42012  0.00000 60.40074  71.39353
## [6,]  95.42966 38.38316  42.38401 65.31600 60.40074  0.00000  44.87016
## [7,] 102.72128 53.17939  49.58802 51.41452 71.39353 44.87016   0.00000
##
## $average.between
## [1] 58.45747
##
## $average.within
```

```
## [1] 24.50646
##
## $n.between
## [1] 223488
##
## $n.within
## [1] 49942
##
## $max.diameter
## [1] 107.9804
##
## $min.separation
## [1] 5.918986
##
## $within.cluster.ss
## [1] 286413.2
##
## $clus.avg.silwidths
##         1         2         3         4         5         6         7
## 0.2555981 0.2917613 0.3661109 0.2586263 0.3347376 0.4194012 0.3441650
##
## $avg.silwidth
## [1] 0.33388
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.5244844
##
## $dunn
## [1] 0.05481539
##
## $dunn2
## [1] 1.041176
```

```
##
## $entropy
## [1] 1.81468
##
## $wb.ratio
## [1] 0.4192186
##
## $ch
## [1] 410.6284
##
## $cwidegap
## [1] 48.99857 15.34082 31.70616 52.45713 18.38207 24.00521 18.72087
##
## $widestgap
## [1] 52.45713
##
## $sindex
## [1] 9.997771
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

0.1.1 Normalización de variables

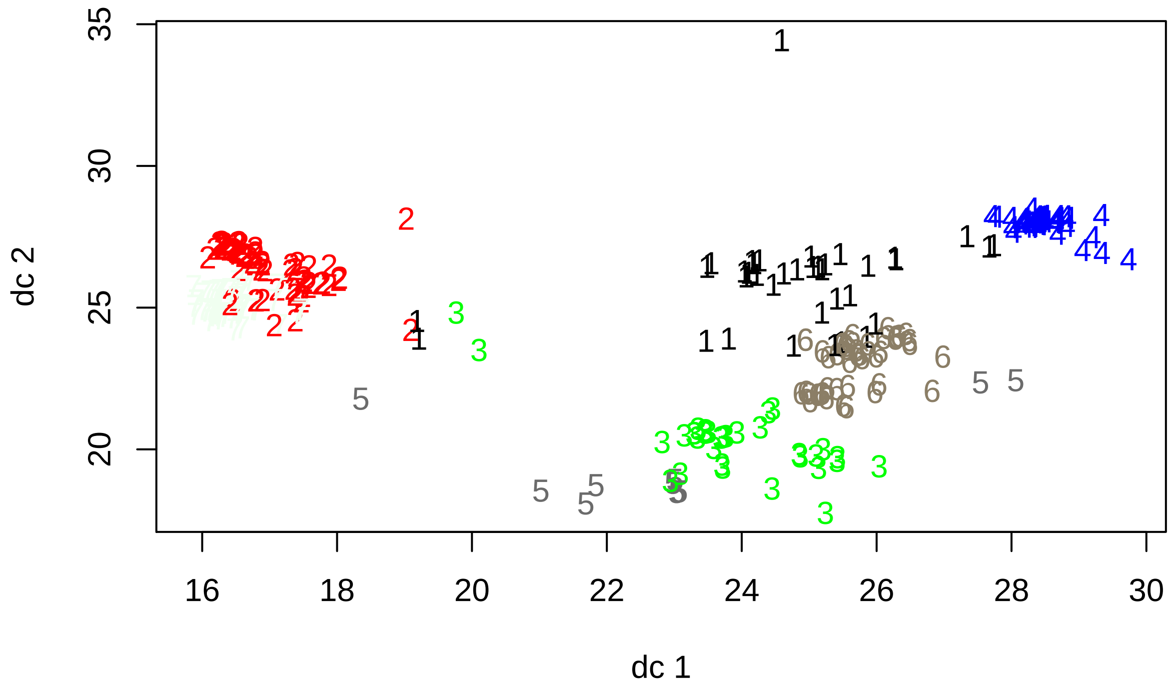
```r
dataset <- read.dataset()
continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

## Normalizamos los atributos continuos
## Esta función nos estandariza los valores del dataframe por columnas
scaled.data <- scale(continuous.data)

## Comprobamos que en efecto las columnas están normalizadas
apply(scaled.data, 2, mean) # Los valores son muy cercanos a cero
```

```
##          Month           Day Transport.exp      Distance  Service.time
## -6.606220e-17  5.330670e-17 -7.672840e-17 -7.982805e-18 -8.773401e-
17
##           Age      Workload    Hit.target     Education           Son
## -4.306392e-16 -3.965324e-16 -1.980806e-16  1.192995e-16 -1.864950e-
17
##           Pet        Weight        Height   Absent.time
## -1.271518e-17 -2.863676e-17 -8.874998e-16  8.386304e-18
```

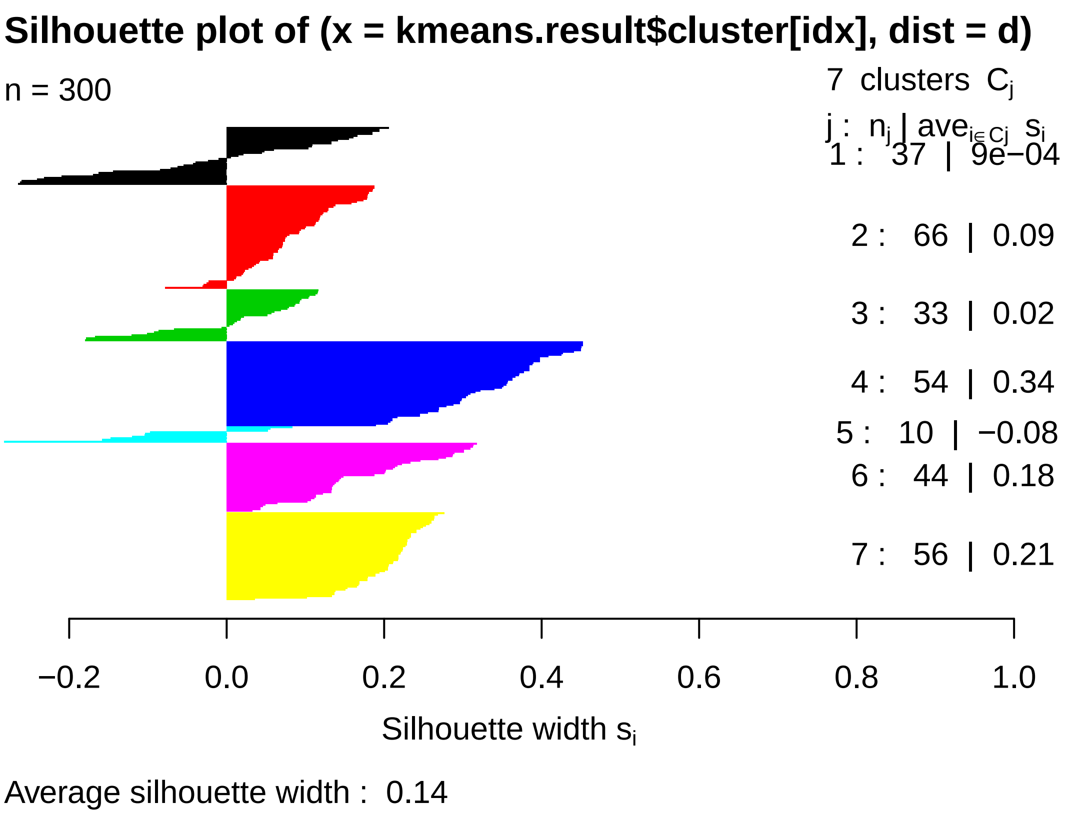
```r
apply(scaled.data, 2, sd)
```

```
##          Month           Day Transport.exp      Distance  Service.time
##             1             1             1             1             1
##           Age      Workload    Hit.target     Education           Son
##             1             1             1             1             1
##           Pet        Weight        Height   Absent.time
##             1             1             1             1
```

```r
scaled.distances <- dist(scaled.data)
binary.distances <- dist(binary.data, method="binary")

final.distances <- (scaled.distances + binary.distances) / 2

kmeans.result <- kmeans(final.distances, nclust)

idx <- sample(1: dim(dataset)[1], 300)

plotcluster(dataset[idx,], kmeans.result$cluster[idx])
```




```r
d1 <- dist(scaled.data[idx,])
d2 <- dist(binary.data[idx,], method="binary")

d <- (d1+d2)/2
sil <- silhouette(kmeans.result$cluster[idx], d)
plot(sil, col = 1:nclust)
```



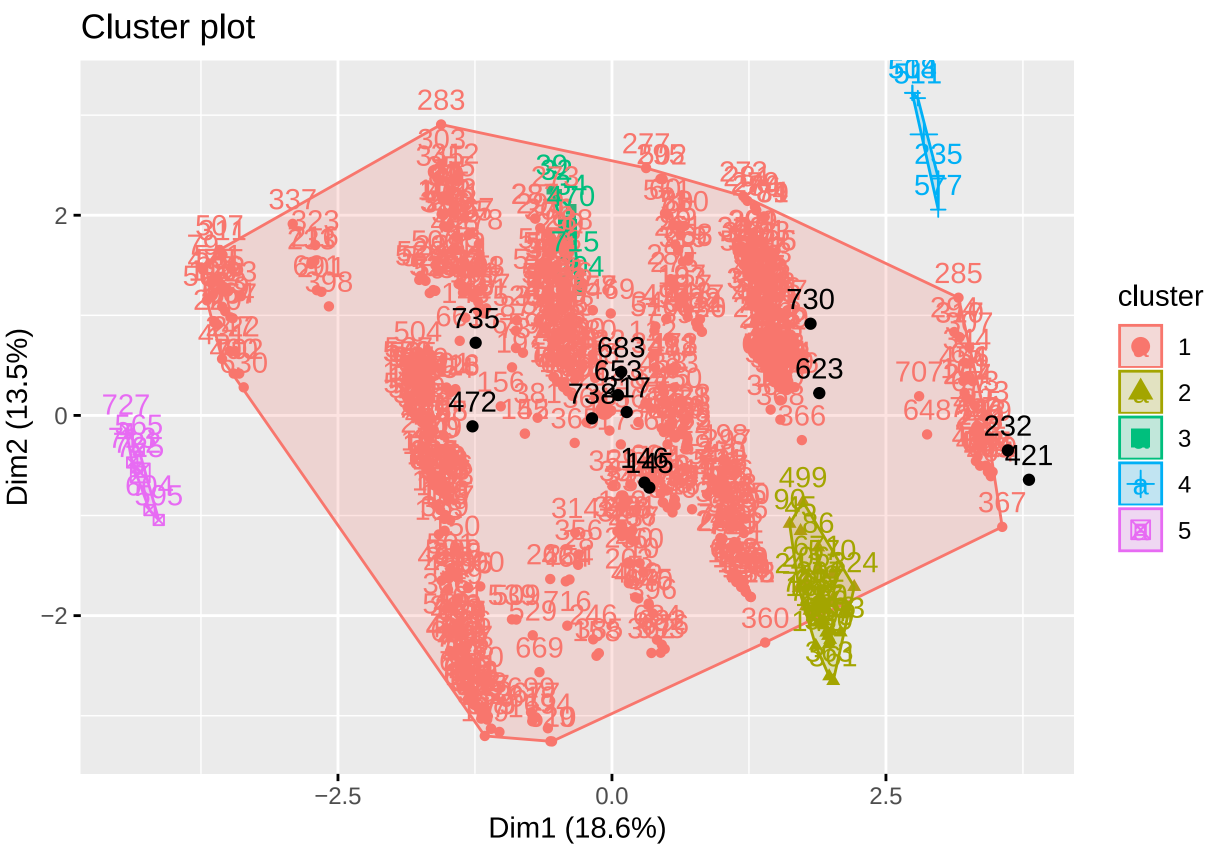

```
cluster.stats(final.distances, kmeans.result$cluster)$avg.silwidth
```

```
## [1] 0.1369902
```

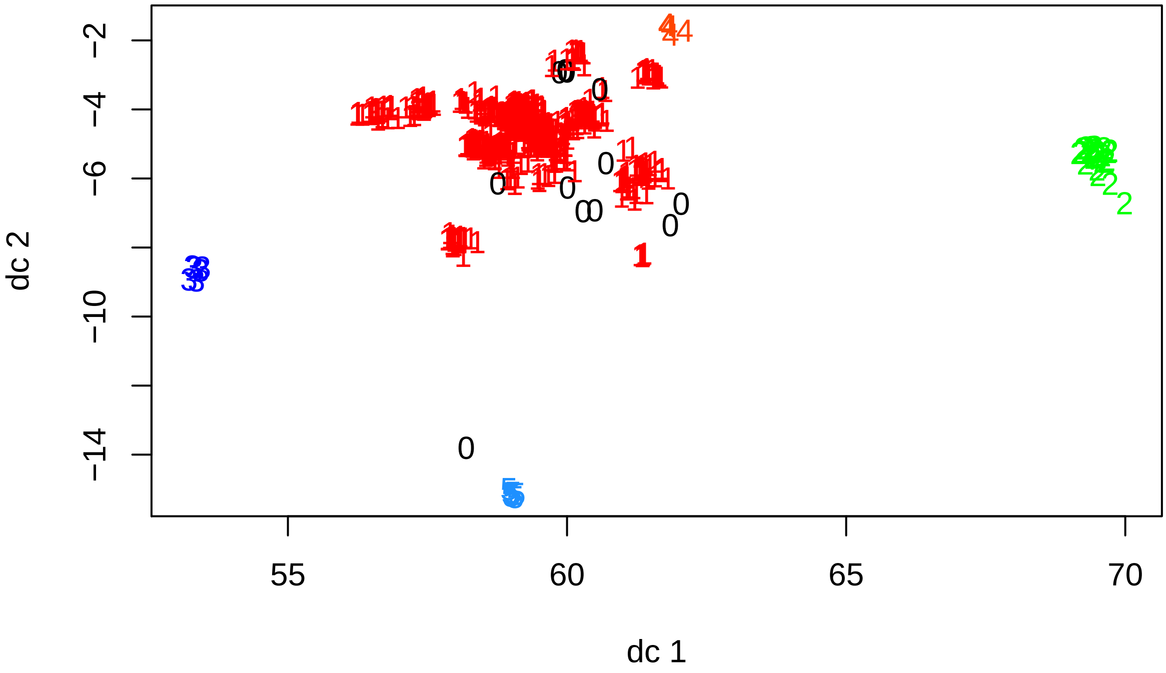
0.2 DBSCAN

```
dataset <- read.dataset()
continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

scaled.data <- scale(continuous.data)

scaled.distances <- dist(scaled.data)
binary.distances <- dist(binary.data, method="binary")

final.distances <- (scaled.distances + binary.distances) / 2
```

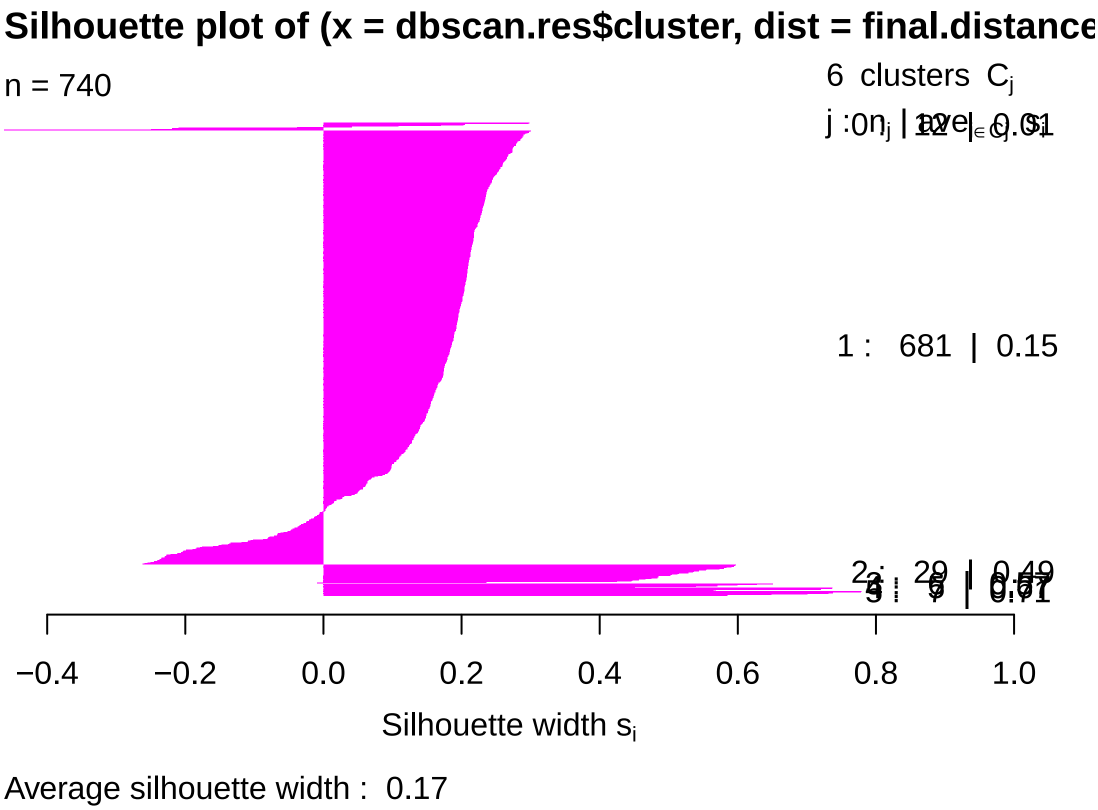
```r
## Aplicamos DBSCAN (con el parámetro dist este método acepta una matriz
## de distancias en lugar de los datos)
dbscan.res <- dbscan(final.distances, eps=2, MinPts=5, method="dist")

fviz_cluster(dbscan.res, scaled.data)
```




```r
plotcluster(dataset, dbscan.res$cluster)
```




```r
sil <- silhouette(dbscan.res$cluster, final.distances)

plot(sil, col=length(unique(dbscan.res$cluster)))
```



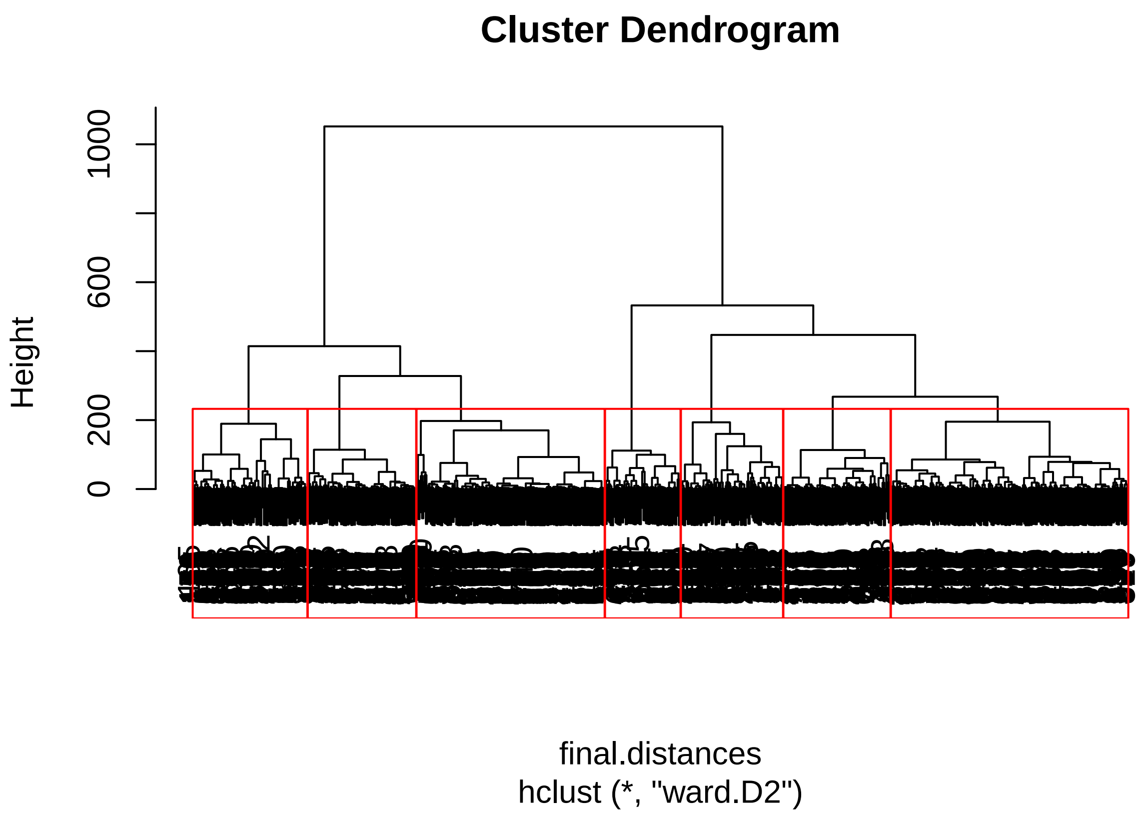
0.3 Clustering jerárquico

```
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

continuous.distances <- dist(continuous.data)
binary.distances <- dist(binary.data, method="binary")
final.distances <- (continuous.distances + binary.distances) / 2

hclust <- hclust(final.distances, method="ward.D2")

hclust
```

```
##
## Call:
## hclust(d = final.distances, method = "ward.D2")
```

```
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 740
```

```r
plot(hclust)

rect.hclust(hclust, k=nclust)
```

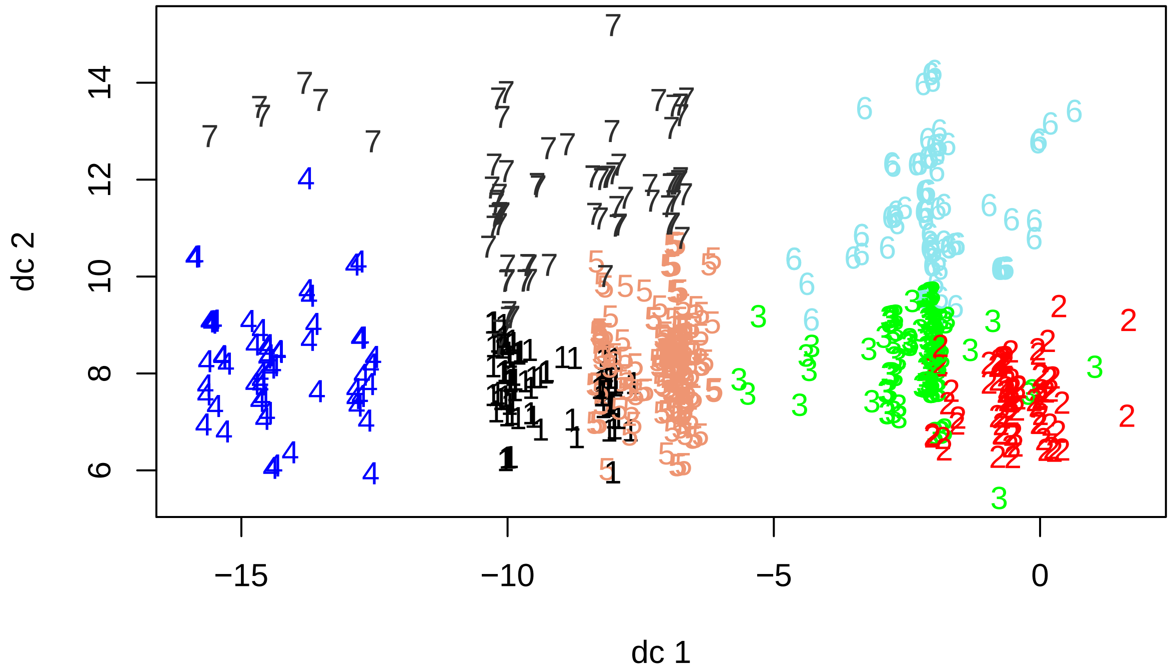



```r
groups <- cutree(hclust, k=nclust)

groups
```

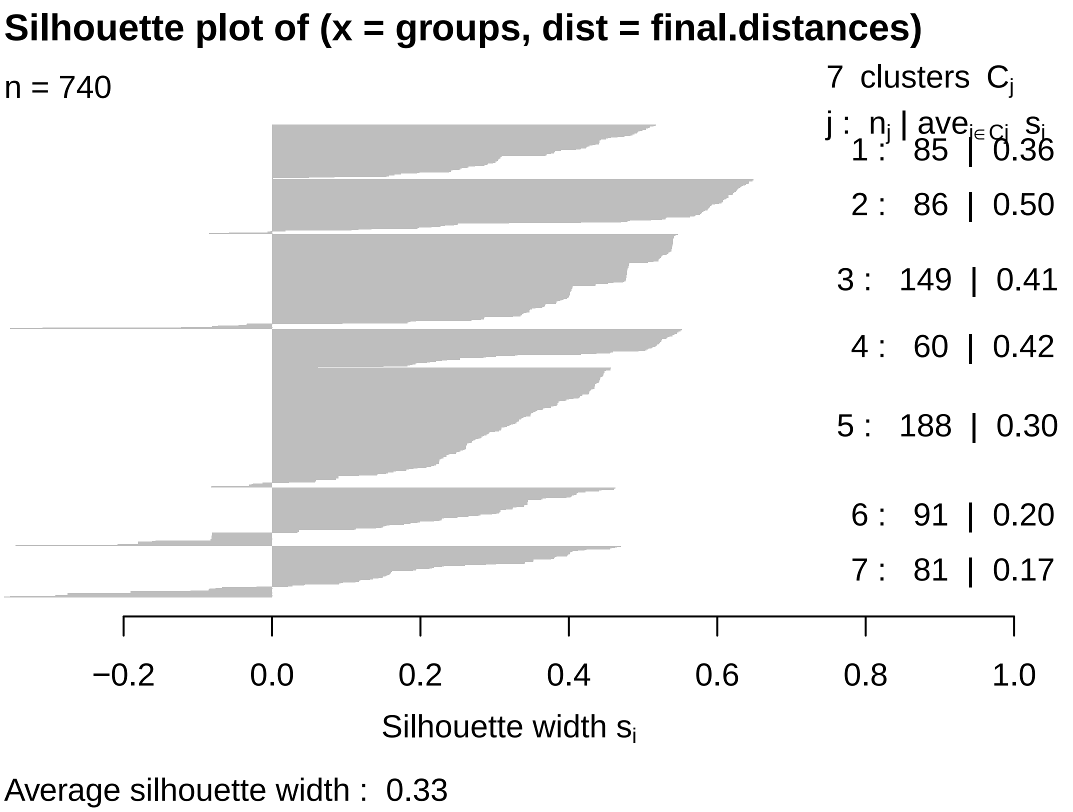
```
##   [1] 1 2 3 1 1 3 4 1 2 5 1 1 1 3 3 5 3 3 3 5 4 3 4 1 1 4 1 3 1 3 3 5 5 5 3 4
##  [38] 1 5 1 3 2 3 5 2 2 3 1 1 1 1 5 5 2 1 2 5 3 4 5 3 1 3 2 2 3 3 5 2 5 5 1 3
##  [75] 1 1 5 1 4 1 6 5 6 6 7 6 6 7 5 6 5 6 5 5 5 5 7 7 5 3 2 5 5 1 3 4 5 1 5 4
## [112] 5 5 5 6 6 6 5 6 5 5 6 5 6 6 6 6 6 6 6 7 6 6 6 6 5 7 6 5 6 6 6 5 7 4 4 5
## [149] 6 6 5 4 4 5 6 7 7 6 6 7 7 6 7 7 6 6 7 6 7 6 7 7 7 6 7 7 6 7 7 6 7 7 7 7
```

```
## [186] 7 6 7 6 7 7 6 6 7 7 7 7 7 7 6 4 7 7 7 7 6 6 6 7 6 6 6 7 7 7 7 7 6 6 7 7
## [223] 7 7 7 6 6 7 6 7 6 6 7 6 7 7 2 2 2 2 4 3 2 4 4 2 4 1 6 7 1 5 1 2 1 5 1 1
## [260] 1 5 1 3 2 5 5 2 5 4 3 6 6 7 5 5 7 5 7 6 5 6 7 5 5 2 5 7 7 6 1 5 1 1 2 5
## [297] 2 4 3 3 5 1 1 5 5 5 2 1 3 2 4 1 1 2 3 3 2 2 5 6 5 7 4 3 5 4 5 2 5 6 5 7
## [334] 6 5 5 4 1 6 5 3 3 5 2 1 5 3 3 3 2 5 3 5 5 1 2 3 1 7 6 6 7 6 7 7 7 6 6 6
## [371] 3 5 1 1 3 3 3 3 5 3 3 3 5 1 2 3 3 5 3 3 2 5 5 1 3 5 3 4 3 4 4 5 2 2 2 5
## [408] 2 3 1 1 5 3 1 4 4 5 4 3 2 3 1 1 3 2 5 5 4 3 3 4 1 3 3 5 5 4 3 2 5 3 2 3
## [445] 5 1 3 1 3 5 3 5 3 5 2 3 3 5 4 5 3 5 1 1 4 2 1 2 1 5 5 3 3 2 4 1 5 5 1 1
## [482] 3 5 2 1 5 1 3 3 4 5 5 2 1 1 5 5 2 2 1 3 5 3 5 3 4 4 7 5 5 7 7 2 7 7 5 4
## [519] 5 4 6 5 7 4 5 4 5 6 5 5 5 5 3 2 4 3 2 1 5 5 4 1 2 5 3 2 2 5 1 1 5 5 5 4
## [556] 6 7 5 3 5 5 4 3 5 5 3 5 5 5 3 6 6 6 6 6 6 7 6 6 6 6 6 7 6 6 3 5 5 3 5 3
## [593] 5 3 5 3 5 3 3 5 3 3 3 5 3 3 3 3 2 5 3 5 3 5 5 3 3 5 3 5 5 5 3 3 5 3 5 3
## [630] 4 3 3 3 3 3 5 3 5 3 1 5 3 5 3 3 3 3 2 2 3 3 3 1 4 5 2 4 5 3 3 4 3 2 4 2
## [667] 3 5 5 3 2 5 2 3 5 5 5 2 5 3 1 5 1 5 3 2 2 2 2 3 4 3 2 5 1 3 5 4 5 1 3 4
## [704] 3 2 5 2 3 2 2 4 2 5 3 5 1 2 3 1 3 2 5 2 2 5 5 5 5 2 3 3 2 4 5 4 1 5 2 5
```

```r
plotcluster(dataset, groups)
```




```r
sil <- silhouette(groups, final.distances)

plot(sil, col <- nclust)
```




```r
cluster.stats(final.distances, groups)
```

```
## $n
## [1] 740
##
## $cluster.number
## [1] 7
##
## $cluster.size
## [1]  85  86 149  60 188  91  81
##
## $min.cluster.size
## [1] 60
##
## $noisen
## [1] 0
##
## $diameter
```

```
## [1] 64.18388 44.86932 76.86780 68.53067 57.09457 72.64573 87.22242
##
## $average.distance
## [1] 20.63111 18.06314 22.38191 25.98852 22.01289 31.09974 34.26386
##
## $median.distance
## [1] 21.04867 17.13792 21.58206 25.61588 21.22675 31.46195 32.34045
##
## $separation
## [1]  4.237086 12.850949  5.855443 19.301596  5.907557  5.855443  4.237086
##
## $average.toother
## [1] 53.33169 64.81951 52.07720 81.50985 47.84407 59.67269 62.09922
##
## $separation.matrix
##           [,1]     [,2]      [,3]     [,4]      [,5]      [,6]      [,7]
## [1,]  0.000000 54.05776 35.444645 19.30160  5.907557 39.019802  4.237086
## [2,] 54.057761  0.00000 15.558173 87.34901 35.617659 12.850949 43.997987
## [3,] 35.444645 15.55817  0.000000 71.80881 18.377976  5.855443 33.459249
## [4,] 19.301596 87.34901 71.808810  0.00000 41.588320 71.805292 23.957408
## [5,]  5.907557 35.61766 18.377976 41.58832  0.000000 18.493242  9.068405
## [6,] 39.019802 12.85095  5.855443 71.80529 18.493242  0.000000 21.594510
## [7,]  4.237086 43.99799 33.459249 23.95741  9.068405 21.594510  0.000000
##
## $ave.between.matrix
##          [,1]      [,2]     [,3]      [,4]     [,5]      [,6]     [,7]
## [1,]  0.00000  81.24277 55.57420  46.16616 32.78187  73.81338 49.56595
## [2,] 81.24277   0.00000 38.46594 119.38343 59.56431  46.37241 88.56684
## [3,] 55.57420  38.46594  0.00000  93.77693 38.42292  46.38715 70.05421
## [4,] 46.16616 119.38343 93.77693   0.00000 66.55963 106.43749 62.51626
## [5,] 32.78187  59.56431 38.42292  66.55963  0.00000  53.31009 48.53242
## [6,] 73.81338  46.37241 46.38715 106.43749 53.31009   0.00000 63.52069
## [7,] 49.56595  88.56684 70.05421  62.51626 48.53242  63.52069  0.00000
##
## $average.between
## [1] 57.61696
##
## $average.within
```

```
## [1] 24.25022
##
## $n.between
## [1] 228496
##
## $n.within
## [1] 44934
##
## $max.diameter
## [1] 87.22242
##
## $min.separation
## [1] 4.237086
##
## $within.cluster.ss
## [1] 278657
##
## $clus.avg.silwidths
##         1         2         3         4         5         6         7
## 0.3611295 0.5049980 0.4098718 0.4204465 0.2988491 0.1953457 0.1656153
##
## $avg.silwidth
## [1] 0.3348628
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.4861356
##
## $dunn
## [1] 0.04857795
##
## $dunn2
## [1] 0.9567476
```

```
##
## $entropy
## [1] 1.873086
##
## $wb.ratio
## [1] 0.4208869
##
## $ch
## [1] 425.4585
##
## $cwidegap
## [1] 24.52968 22.23173 38.20329 29.51472 15.34082 28.03025 25.29919
##
## $widestgap
## [1] 38.20329
##
## $sindex
## [1] 8.263557
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

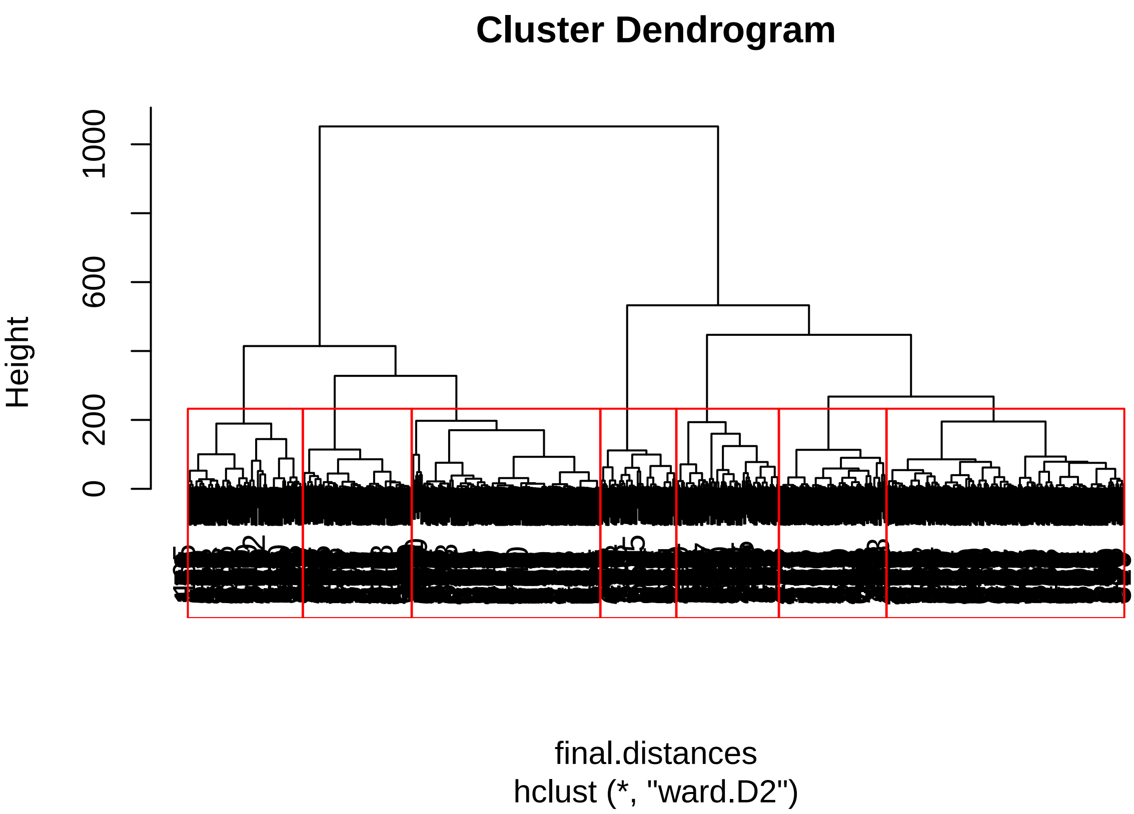
0.3.1 Normalización de variables

```
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

continuous.distances <- dist(continuous.data)
binary.distances <- dist(binary.data, method="binary")
final.distances <- (continuous.distances + binary.distances) / 2

hclust <- hclust(final.distances, method="ward.D2")
```

```r
hclust <- hclust(final.distances, method="ward.D2")

hclust
```

```
##
## Call:
## hclust(d = final.distances, method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 740
```

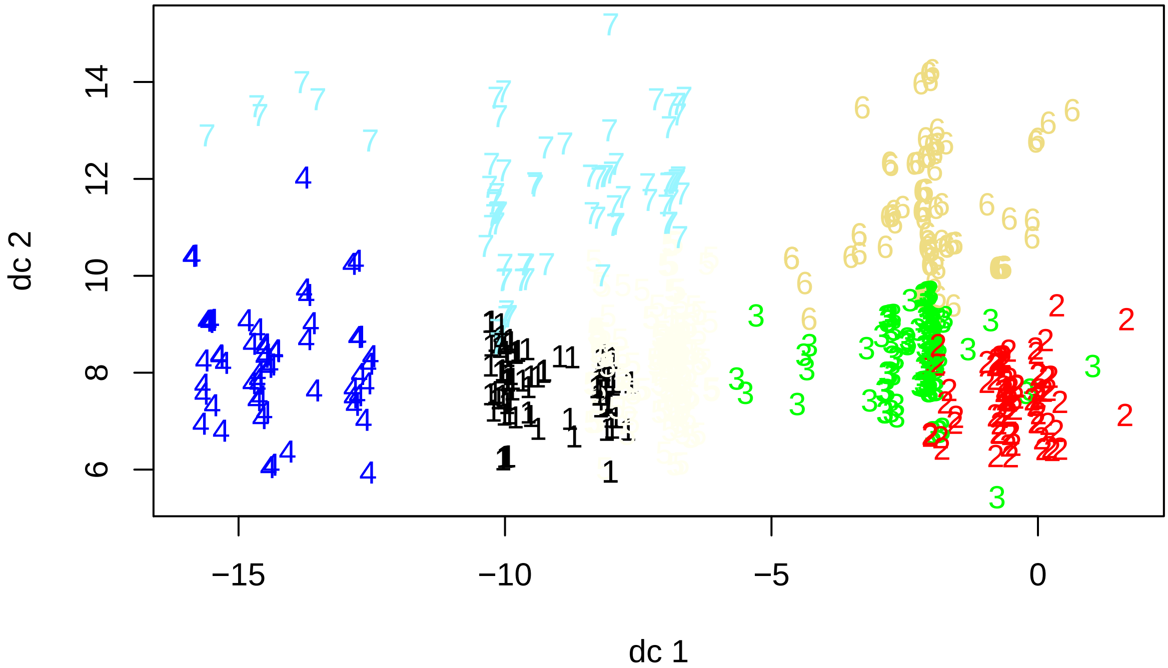
```r
plot(hclust)

rect.hclust(hclust, k=nclust)
```



```r
groups <- cutree(hclust, k=nclust)

groups
```

```
##   [1] 1 2 3 1 1 3 4 1 2 5 1 1 1 3 3 5 3 3 3 5 4 3 4 1 1 4 1 3 1 3 3 5 5 5 3 4
##  [38] 1 5 1 3 2 3 5 2 2 3 1 1 1 1 5 5 2 1 2 5 3 4 5 3 1 3 2 2 3 3 5 2 5 5 1 3
##  [75] 1 1 5 1 4 1 6 5 6 6 7 6 6 7 5 6 5 6 5 5 5 5 7 7 5 3 2 5 5 1 3 4 5 1 5 4
## [112] 5 5 5 6 6 6 5 6 5 5 6 5 6 6 6 6 6 6 6 7 6 6 6 6 5 7 6 5 6 6 6 5 7 4 4 5
## [149] 6 6 5 4 4 5 6 7 7 6 6 7 7 6 7 7 6 6 7 6 7 6 7 7 7 6 7 7 6 7 7 6 7 7 7 7
## [186] 7 6 7 6 7 7 6 6 7 7 7 7 7 7 6 4 7 7 7 7 6 6 6 7 6 6 6 7 7 7 7 7 6 6 7 7
## [223] 7 7 7 6 6 7 6 7 6 6 7 6 7 7 2 2 2 2 4 3 2 4 4 2 4 1 6 7 1 5 1 2 1 5 1 1
## [260] 1 5 1 3 2 5 5 2 5 4 3 6 6 7 5 5 7 5 7 6 5 6 7 5 5 2 5 7 7 6 1 5 1 1 2 5
## [297] 2 4 3 3 5 1 1 5 5 5 2 1 3 2 4 1 1 2 3 3 2 2 5 6 5 7 4 3 5 4 5 2 5 6 5 7
## [334] 6 5 5 4 1 6 5 3 3 5 2 1 5 3 3 3 2 5 3 5 5 1 2 3 1 7 6 6 7 6 7 7 7 6 6 6
## [371] 3 5 1 1 3 3 3 3 5 3 3 3 5 1 2 3 3 5 3 3 2 5 5 1 3 5 3 4 3 4 4 5 2 2 2 5
## [408] 2 3 1 1 5 3 1 4 4 5 4 3 2 3 1 1 3 2 5 5 4 3 3 4 1 3 3 5 5 4 3 2 5 3 2 3
## [445] 5 1 3 1 3 5 3 5 3 5 2 3 3 5 4 5 3 5 1 1 4 2 1 2 1 5 5 3 3 2 4 1 5 5 1 1
## [482] 3 5 2 1 5 1 3 3 4 5 5 2 1 1 5 5 2 2 1 3 5 3 5 3 4 4 7 5 5 7 7 2 7 7 5 4
## [519] 5 4 6 5 7 4 5 4 5 6 5 5 5 5 3 2 4 3 2 1 5 5 4 1 2 5 3 2 2 5 1 1 5 5 5 4
## [556] 6 7 5 3 5 5 4 3 5 5 3 5 5 5 3 6 6 6 6 6 6 7 6 6 6 6 6 7 6 6 3 5 5 3 5 3
## [593] 5 3 5 3 5 3 3 5 3 3 3 5 3 3 3 3 2 5 3 5 3 5 5 3 3 5 3 5 5 5 3 3 5 3 5 3
## [630] 4 3 3 3 3 3 5 3 5 3 1 5 3 5 3 3 3 3 2 2 3 3 3 1 4 5 2 4 5 3 3 4 3 2 4 2
## [667] 3 5 5 3 2 5 2 3 5 5 5 2 5 3 1 5 1 5 3 2 2 2 2 3 4 3 2 5 1 3 5 4 5 1 3 4
## [704] 3 2 5 2 3 2 2 4 2 5 3 5 1 2 3 1 3 2 5 2 2 5 5 5 5 2 3 3 2 4 5 4 1 5 2 5
```

```r
plotcluster(dataset, groups)
```




```r
sil <- silhouette(groups, final.distances)

plot(sil, col <- nclust)
```


```
cluster.stats(final.distances, groups)
```

```
## $n
## [1] 740
##
## $cluster.number
## [1] 7
##
## $cluster.size
## [1]  85  86 149  60 188  91  81
##
## $min.cluster.size
## [1] 60
##
## $noisen
## [1] 0
##
## $diameter
```

```
## [1] 64.18388 44.86932 76.86780 68.53067 57.09457 72.64573 87.22242
##
## $average.distance
## [1] 20.63111 18.06314 22.38191 25.98852 22.01289 31.09974 34.26386
##
## $median.distance
## [1] 21.04867 17.13792 21.58206 25.61588 21.22675 31.46195 32.34045
##
## $separation
## [1]  4.237086 12.850949  5.855443 19.301596  5.907557  5.855443  4.237086
##
## $average.toother
## [1] 53.33169 64.81951 52.07720 81.50985 47.84407 59.67269 62.09922
##
## $separation.matrix
##           [,1]     [,2]      [,3]     [,4]      [,5]      [,6]      [,7]
## [1,]  0.000000 54.05776 35.444645 19.30160  5.907557 39.019802  4.237086
## [2,] 54.057761  0.00000 15.558173 87.34901 35.617659 12.850949 43.997987
## [3,] 35.444645 15.55817  0.000000 71.80881 18.377976  5.855443 33.459249
## [4,] 19.301596 87.34901 71.808810  0.00000 41.588320 71.805292 23.957408
## [5,]  5.907557 35.61766 18.377976 41.58832  0.000000 18.493242  9.068405
## [6,] 39.019802 12.85095  5.855443 71.80529 18.493242  0.000000 21.594510
## [7,]  4.237086 43.99799 33.459249 23.95741  9.068405 21.594510  0.000000
##
## $ave.between.matrix
##          [,1]      [,2]     [,3]      [,4]     [,5]      [,6]     [,7]
## [1,]  0.00000  81.24277 55.57420  46.16616 32.78187  73.81338 49.56595
## [2,] 81.24277   0.00000 38.46594 119.38343 59.56431  46.37241 88.56684
## [3,] 55.57420  38.46594  0.00000  93.77693 38.42292  46.38715 70.05421
## [4,] 46.16616 119.38343 93.77693   0.00000 66.55963 106.43749 62.51626
## [5,] 32.78187  59.56431 38.42292  66.55963  0.00000  53.31009 48.53242
## [6,] 73.81338  46.37241 46.38715 106.43749 53.31009   0.00000 63.52069
## [7,] 49.56595  88.56684 70.05421  62.51626 48.53242  63.52069  0.00000
##
## $average.between
## [1] 57.61696
##
## $average.within
```

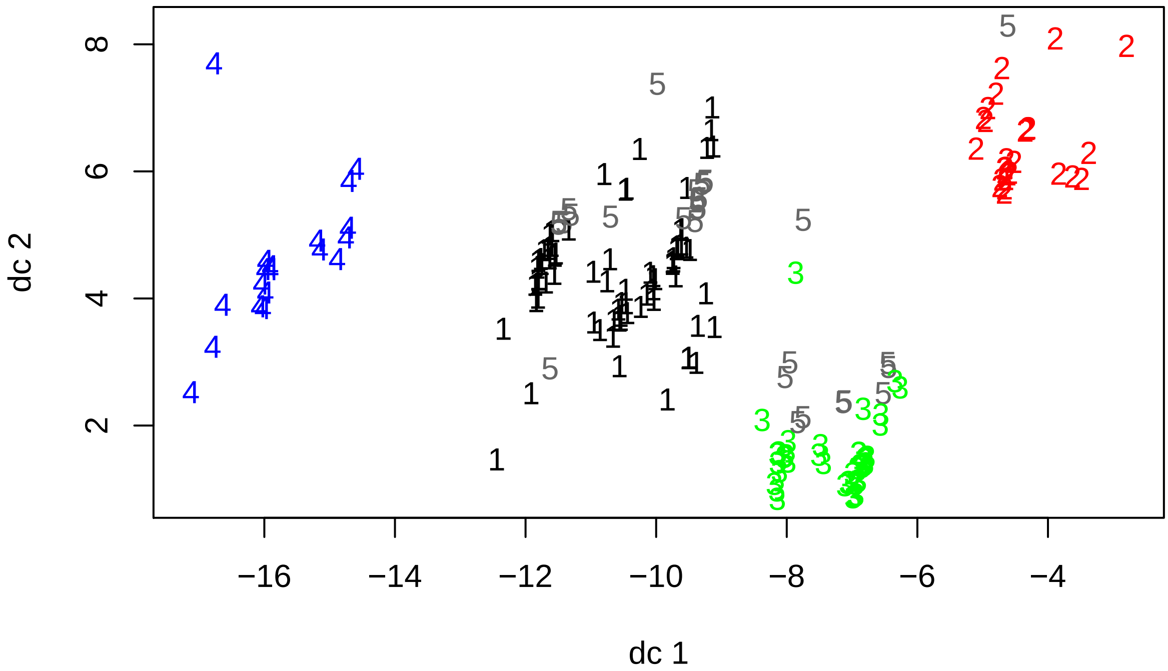
```
## [1] 24.25022
##
## $n.between
## [1] 228496
##
## $n.within
## [1] 44934
##
## $max.diameter
## [1] 87.22242
##
## $min.separation
## [1] 4.237086
##
## $within.cluster.ss
## [1] 278657
##
## $clus.avg.silwidths
##         1         2         3         4         5         6         7
## 0.3611295 0.5049980 0.4098718 0.4204465 0.2988491 0.1953457 0.1656153
##
## $avg.silwidth
## [1] 0.3348628
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.4861356
##
## $dunn
## [1] 0.04857795
##
## $dunn2
## [1] 0.9567476
```

```
##
## $entropy
## [1] 1.873086
##
## $wb.ratio
## [1] 0.4208869
##
## $ch
## [1] 425.4585
##
## $cwidegap
## [1] 24.52968 22.23173 38.20329 29.51472 15.34082 28.03025 25.29919
##
## $widestgap
## [1] 38.20329
##
## $sindex
## [1] 8.263557
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

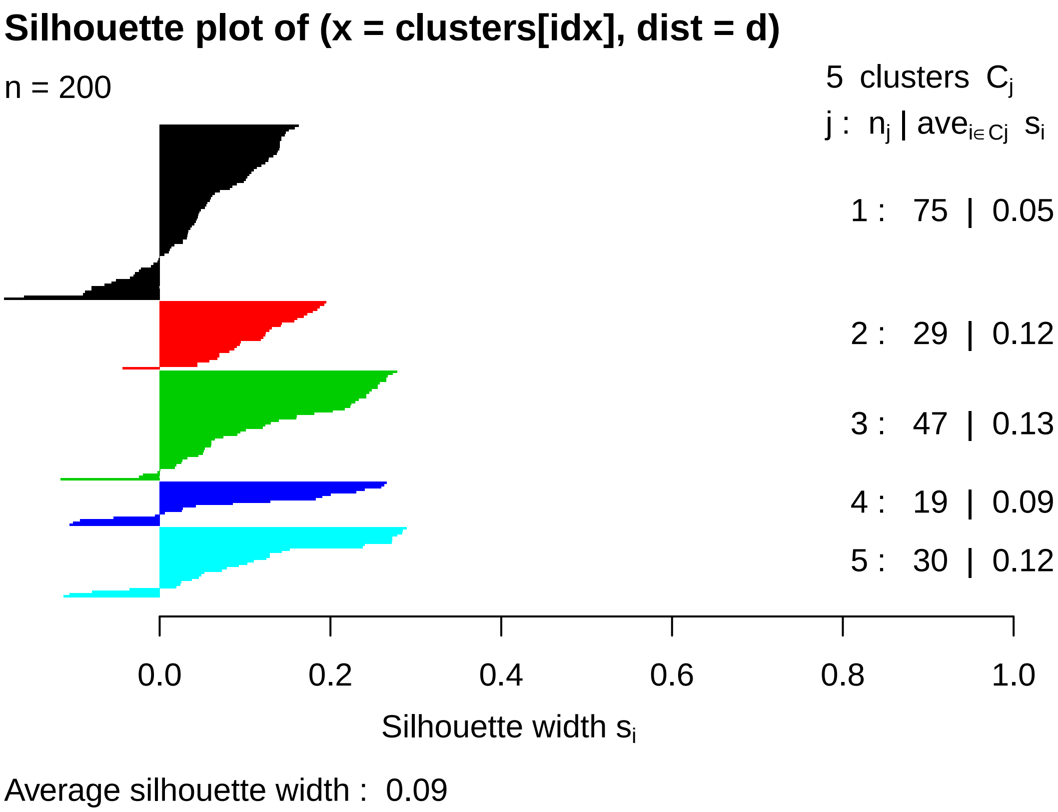
0.4 k-medioides

```
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

continuous.distances <- dist(continuous.data)
binary.distances <- dist(binary.data, method="binary")
final.distances <- (continuous.distances + binary.distances) / 2

pam.result <- pam(final.distances, 5)
```

```r
idx <- sample(1:dim(dataset)[1], 200)

clusters <- pam.result$cluster

plotcluster(dataset[idx,], clusters[idx])
```




```r
scaled.d <- dist(scaled.data[idx,])
binary.d <- dist(binary.data[idx,], method="binary")

d <- (scaled.d + binary.d) / 2
sil <- silhouette(clusters[idx], d)

plot(sil, col=1:5)
```



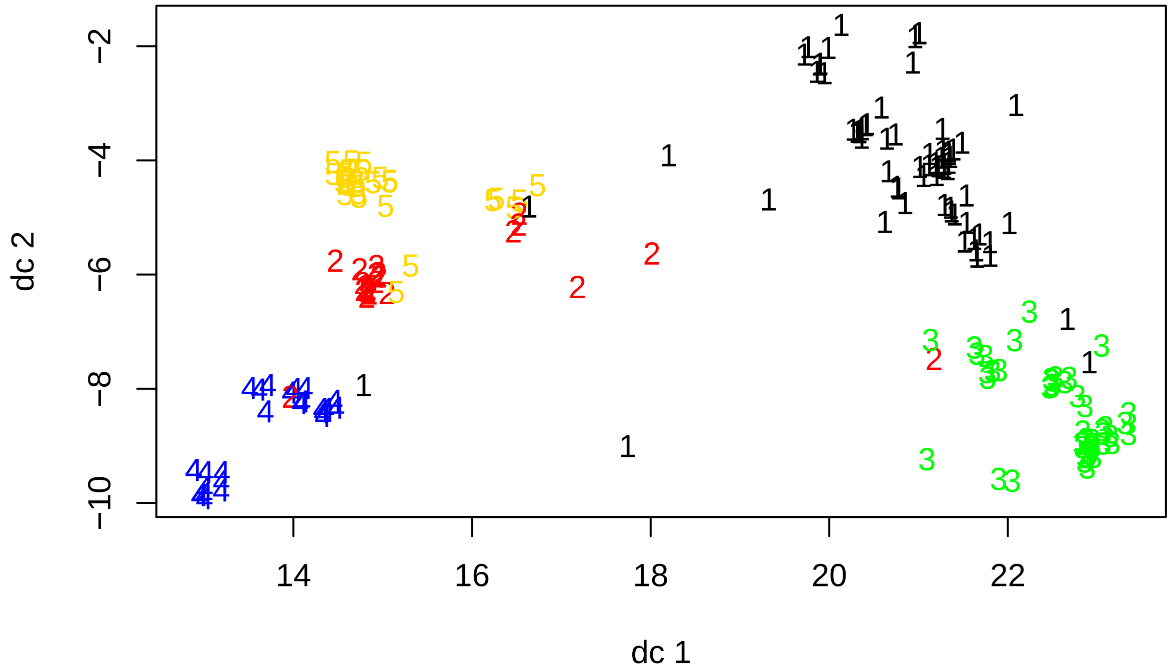

```
cluster.stats(final.distances, clusters)$avg.silwidth
```

```
## [1] 0.3501093
```

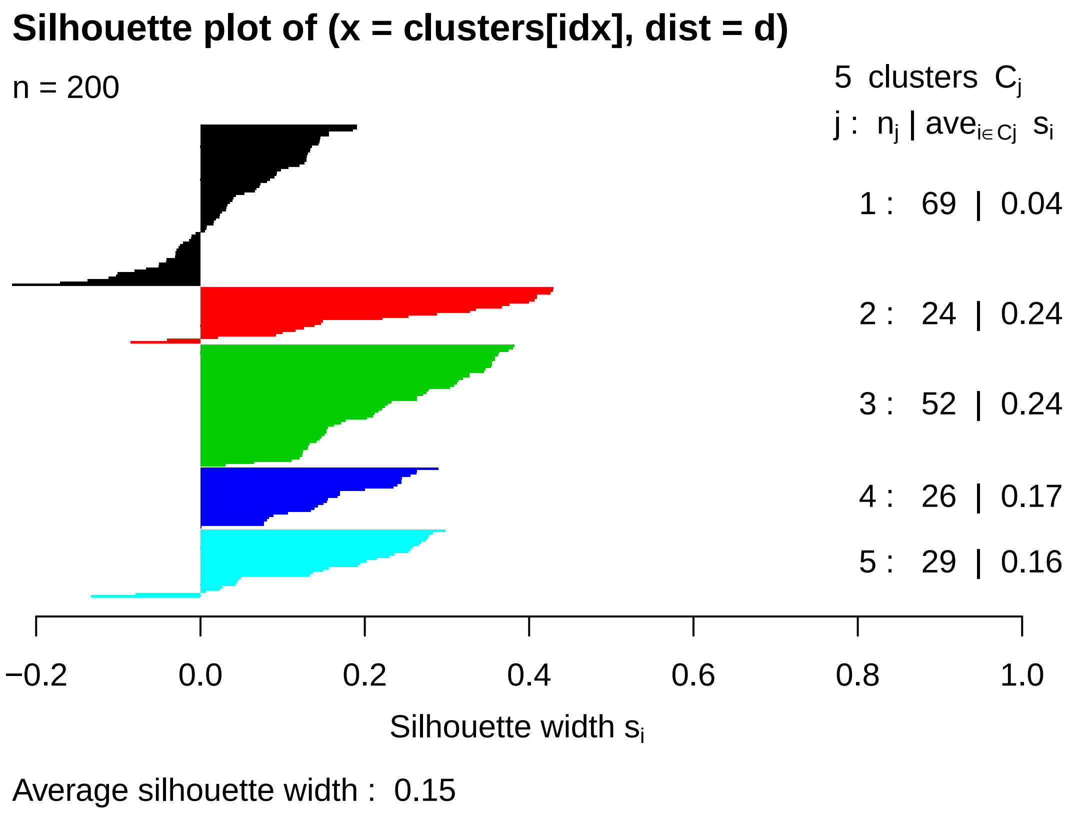
0.4.1 Resultados con normalización de variables

```
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

scaled.data <- scale(continuous.data)

scaled.distances <- dist(scaled.data)
binary.distances <- dist(binary.data, method="binary")
final.distances <- (scaled.distances + binary.distances) / 2

pam.result <- pam(final.distances, 5)
```

```
idx <- sample(1:dim(dataset)[1], 200)

clusters <- pam.result$cluster

plotcluster(dataset[idx,], clusters[idx])
```




```
scaled.d <- dist(scaled.data[idx,])
binary.d <- dist(binary.data[idx,], method="binary")

d <- (scaled.d + binary.d) / 2
sil <- silhouette(clusters[idx], d)

plot(sil, col=1:5)
```




```r
cluster.stats(final.distances, clusters)$avg.silwidth
```

```
## [1] 0.157716
```

0.4.2 Búsqueda del valor óptimo de k

```r
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

scaled.data <- scale(continuous.data)

scaled.distances <- dist(scaled.data)
binary.distances <- dist(binary.data, method="binary")

final.distances <- (scaled.distances + binary.distances) / 2
```

```r
pamk.result <- pamk(scaled.distances)

plot(pamk.result$pamobject)
```


```r
cluster.stats(final.distances, pamk.result$pamobject$clustering)
```

```
## $n
## [1] 740
##
## $cluster.number
## [1] 10
##
## $cluster.size
##  [1] 170  44 119  37  45  30  48 120  75  52
##
## $min.cluster.size
## [1] 30
##
```

```
## $noisen
## [1] 0
##
## $diameter
##  [1] 4.555276 5.834609 3.829699 3.606786 3.145392 5.237881 3.850114 5.737936
##  [9] 4.172827 3.143330
##
## $average.distance
##  [1] 2.256950 2.201947 1.505298 1.972553 1.482826 1.830242 2.110267 2.088655
##  [9] 1.866325 1.417789
##
## $median.distance
##  [1] 2.283216 1.850978 1.443048 2.143635 1.521093 1.666095 2.190573 1.931623
##  [9] 1.888124 1.412326
##
## $separation
##  [1] 0.3516979 0.3536922 0.7246569 0.8024480 0.7033958 1.1732103 0.4035716
##  [8] 0.3516979 0.3969082 0.7385177
##
## $average.toother
##  [1] 2.758539 3.184774 2.874486 3.465718 3.027615 3.540243 2.984616 2.849491
##  [9] 2.813014 3.136856
##
## $separation.matrix
##            [,1]      [,2]      [,3]     [,4]      [,5]     [,6]      [,7]
##  [1,] 0.0000000 0.3536922 1.2657973 0.802448 0.9988275 2.258271 0.4035716
##  [2,] 0.3536922 0.0000000 0.7246569 2.864052 2.2725584 2.062760 1.7588665
##  [3,] 1.2657973 0.7246569 0.0000000 2.833382 2.2886519 2.485909 2.3447520
##  [4,] 0.8024480 2.8640524 2.8333819 0.000000 2.4437086 2.973980 2.5630211
##  [5,] 0.9988275 2.2725584 2.2886519 2.443709 0.0000000 2.950699 2.2997456
##  [6,] 2.2582708 2.0627603 2.4859087 2.973980 2.9506994 0.000000 2.2728631
##  [7,] 0.4035716 1.7588665 2.3447520 2.563021 2.2997456 2.272863 0.0000000
##  [8,] 0.3516979 1.5334234 1.5510859 1.918735 0.7033958 2.235290 0.9016867
##  [9,] 0.7033958 0.3969082 1.0181614 2.762128 2.1348232 1.173210 1.5328948
## [10,] 1.0019379 2.9095792 2.9490063 3.289092 2.9251029 2.591850 0.7385177
##            [,8]      [,9]     [,10]
##  [1,] 0.3516979 0.7033958 1.0019379
##  [2,] 1.5334234 0.3969082 2.9095792
```

```
##  [3,] 1.5510859 1.0181614 2.9490063
##  [4,] 1.9187345 2.7621276 3.2890919
##  [5,] 0.7033958 2.1348232 2.9251029
##  [6,] 2.2352902 1.1732103 2.5918503
##  [7,] 0.9016867 1.5328948 0.7385177
##  [8,] 0.0000000 0.9246036 1.4324056
##  [9,] 0.9246036 0.0000000 1.8858558
## [10,] 1.4324056 1.8858558 0.0000000
##
## $ave.between.matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
##  [1,] 0.000000 2.908732 2.496834 3.189935 2.568054 3.425274 2.842888 2.598209
##  [2,] 2.908732 0.000000 2.682758 4.258318 3.455613 3.081774 3.325480 3.541729
##  [3,] 2.496834 2.682758 0.000000 3.462103 2.839123 3.329493 3.176894 2.965102
##  [4,] 3.189935 4.258318 3.462103 0.000000 3.155436 4.123244 3.715033 3.032488
##  [5,] 2.568054 3.455613 2.839123 3.155436 0.000000 3.673264 3.469562 2.883890
##  [6,] 3.425274 3.081774 3.329493 4.123244 3.673264 0.000000 3.562502 3.745092
##  [7,] 2.842888 3.325480 3.176894 3.715033 3.469562 3.562502 0.000000 2.781273
##  [8,] 2.598209 3.541729 2.965102 3.032488 2.883890 3.745092 2.781273 0.000000
##  [9,] 2.685423 2.811686 2.709408 3.820172 3.394047 3.318954 2.671974 2.542088
## [10,] 3.101177 3.881716 3.371173 3.852492 3.531000 4.082278 2.366912 2.650354
##           [,9]    [,10]
##  [1,] 2.685423 3.101177
##  [2,] 2.811686 3.881716
##  [3,] 2.709408 3.371173
##  [4,] 3.820172 3.852492
##  [5,] 3.394047 3.531000
##  [6,] 3.318954 4.082278
##  [7,] 2.671974 2.366912
##  [8,] 2.542088 2.650354
##  [9,] 0.000000 2.712424
## [10,] 2.712424 0.000000
##
## $average.between
## [1] 2.960555
##
## $average.within
## [1] 1.918848
```

```
##
## $n.between
## [1] 236638
##
## $n.within
## [1] 36792
##
## $max.diameter
## [1] 5.834609
##
## $min.separation
## [1] 0.3516979
##
## $within.cluster.ss
## [1] 1615.245
##
## $clus.avg.silwidths
##          1          2          3          4          5          6          7
## 0.02015947 0.16766254 0.40472562 0.34790604 0.42436504 0.40740249 0.09661172
##          8          9         10
## 0.12767454 0.23347759 0.39976107
##
## $avg.silwidth
## [1] 0.2181275
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.4241458
##
## $dunn
## [1] 0.06027788
##
## $dunn2
```

```
## [1] 1.048722
##
## $entropy
## [1] 2.140652
##
## $wb.ratio
## [1] 0.6481378
##
## $ch
## [1] 78.85568
##
## $cwidegap
##  [1] 2.174011 2.410118 2.531175 2.115734 1.551279 2.129829 2.041725 2.688481
##  [9] 2.020455 1.834346
##
## $widestgap
## [1] 2.688481
##
## $sindex
## [1] 0.7009609
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

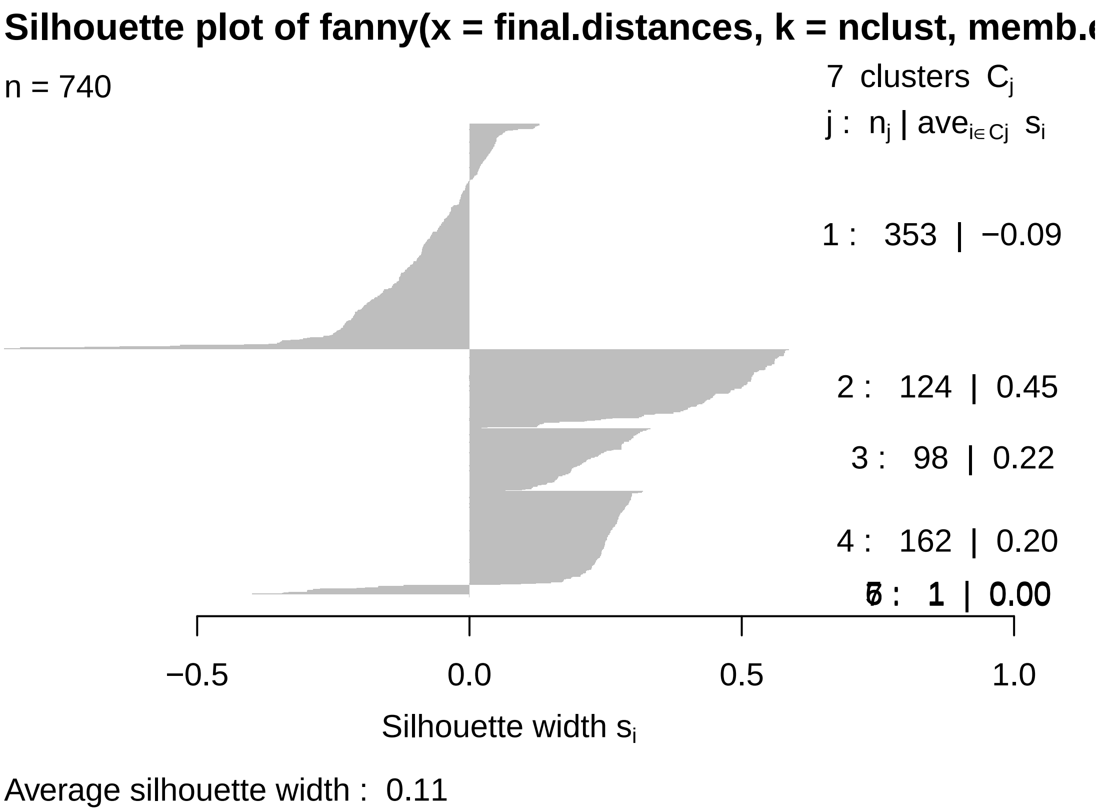
0.5 Fuzzy k-means

```
dataset <- read.dataset()

continuous.data <- dataset %>% select(continuous.vars)
binary.data <- dataset %>% select(binary.vars)

scaled.data <- scale(continuous.data)

scaled.distances <- dist(scaled.data)
```

```r
binary.distances <- dist(binary.data, method="binary")
final.distances <- (scaled.distances + binary.distances) / 2

fuzzy.result <- fanny(final.distances, nclust, memb.exp=1.3)
```

```
## Warning in fanny(final.distances, nclust, memb.exp = 1.3): FANNY algorithm has
## not converged in 'maxit' = 500 iterations
```

```r
plot(fuzzy.result)
```




```r
str(fuzzy.result)
```

```
## List of 10
##  $ membership : num [1:740, 1:7] 0.2084 0.1775 0.0324 0.1959 0.2075 ...
##  $ coeff      : Named num [1:2] 0.243 0.117
##   ..- attr(*, "names")= chr [1:2] "dunn_coeff" "normalized"
##  $ memb.exp   : num 1.3
```

```
##  $ clustering : int [1:740] 1 1 2 1 1 2 1 1 1 3 ...
##  $ k.crisp    : num 7
##  $ objective  : Named num [1:2] 5.78e+02 1.00e-15
##   ..- attr(*, "names")= chr [1:2] "objective" "tolerance"
##  $ convergence: Named int [1:3] -1 0 500
##   ..- attr(*, "names")= chr [1:3] "iterations" "converged" "maxit"
##  $ diss       : NULL
##  $ call       : language fanny(x = final.distances, k = nclust, memb.exp = 1.3
##  $ silinfo    :List of 3
##   ..$ widths         : num [1:740, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:740] "524" "526" "506" "554" ...
##   .. .. ..$ : chr [1:3] "cluster" "neighbor" "sil_width"
##   ..$ clus.avg.widths: num [1:7] -0.0894 0.4451 0.2223 0.2023 0 ...
##   ..$ avg.width      : num 0.106
##  - attr(*, "class")= chr [1:2] "fanny" "partition"
```

```
cluster.stats(final.distances, fuzzy.result$clustering)
```

```
## $n
## [1] 740
##
## $cluster.number
## [1] 7
##
## $cluster.size
## [1] 353 124  98 162   1   1   1
##
## $min.cluster.size
## [1] 1
##
## $noisen
## [1] 0
##
## $diameter
## [1] 6.449945 3.499390 4.104616 5.058020       NA       NA       NA
##
## $average.distance
```

```
## [1] 2.838793 1.471330 1.924823 1.872636      NaN      NaN      NaN
##
## $median.distance
## [1] 2.821326 1.479724 2.001979 1.911881       NA       NA       NA
##
## $separation
## [1] 0.3625197 0.3625197 1.5328948 0.4805208 0.4782953 1.1634500 1.1634500
##
## $average.toother
## [1] 3.009530 2.843380 3.132184 2.877534 3.040798 5.180663 5.336964
##
## $separation.matrix
##           [,1]      [,2]     [,3]      [,4]      [,5]     [,6]     [,7]
## [1,] 0.0000000 0.3625197 1.872921 0.4805208 0.4782953 2.410118 2.550229
## [2,] 0.3625197 0.0000000 2.293058 1.5905782 2.2537170 4.508649 4.730019
## [3,] 1.8729215 2.2930579 0.000000 1.5328948 2.4755152 3.966412 3.751693
## [4,] 0.4805208 1.5905782 1.532895 0.0000000 1.4084199 2.793853 2.626589
## [5,] 0.4782953 2.2537170 2.475515 1.4084199 0.0000000 3.939181 4.305919
## [6,] 2.4101176 4.5086493 3.966412 2.7938532 3.9391807 0.000000 1.163450
## [7,] 2.5502291 4.7300189 3.751693 2.6265889 4.3059191 1.163450 0.000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 0.000000 2.706449 3.370743 2.993093 3.167514 5.293705 5.413286
## [2,] 2.706449 0.000000 3.286886 2.839814 3.079662 5.335756 5.565537
## [3,] 3.370743 3.286886 0.000000 2.468406 3.111053 5.122351 5.300717
## [4,] 2.993093 2.839814 2.468406 0.000000 2.679077 4.883365 5.049753
## [5,] 3.167514 3.079662 3.111053 2.679077 0.000000 3.939181 4.305919
## [6,] 5.293705 5.335756 5.122351 4.883365 3.939181 0.000000 1.163450
## [7,] 5.413286 5.565537 5.300717 5.049753 4.305919 1.163450 0.000000
##
## $average.between
## [1] 2.971909
##
## $average.within
## [1] 2.274815
##
## $n.between
```

```
## [1] 185882
##
## $n.within
## [1] 87548
##
## $max.diameter
## [1] 6.449945
##
## $min.separation
## [1] 0.3625197
##
## $within.cluster.ss
## [1] 2215.586
##
## $clus.avg.silwidths
##           1           2           3           4           5           6
## -0.08937679  0.44505450  0.22225222  0.20234118  0.00000000  0.00000000
##           7
##  0.00000000
##
## $avg.silwidth
## [1] 0.1056713
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.2647548
##
## $dunn
## [1] 0.05620508
##
## $dunn2
## [1] 0.4098397
##
```

```
## $entropy
## [1] 1.279494
##
## $wb.ratio
## [1] 0.765439
##
## $ch
## [1] 53.48486
##
## $cwidegap
## [1] 2.650516 2.539814 1.834346 2.632485 0.000000 0.000000 0.000000
##
## $widestgap
## [1] 2.650516
##
## $sindex
## [1] 0.7853203
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```