CSDA1010SUMA18 - LAB EXERCISE 3: Classification Problem

```
library(readr)
library(dplyr)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(Amelia)
library(rattle)
library(RColorBrewer)
library(caret)
```

Nursery Data Set reference and short description

Source: http://archive.ics.uci.edu/ml/datasets/Nursery

```
| class values

not_recom, recommend, very_recom, priority, spec_prior

| attributes

parents:     usual, pretentious, great_pret.
has_nurs:    proper, less_proper, improper, critical, very_crit.
form:        complete, completed, incomplete, foster.
children:    1, 2, 3, more.
housing:     convenient, less_conv, critical.
finance:     convenient, inconv.
social:      nonprob, slightly_prob, problematic.
health:      recommended, priority, not_recom.
```

```
nursery_data <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data

# nursery_data <- read.csv(file = "../data/nursery_data.csv")
```

Data Set exploration and cleaning

```
set.seed(77)
dim(nursery_data)
```

```
## [1] 12960     9
```

```
#head(nursery_data)
#str(nursery_data)
```

Coding for categorical variables

Reorder factors

Very often, especially when plotting data, we need to reorder the levels of a factor because the default order is alphabetical. A direct way of reordering, using standard syntax is as follows.

Current levels, need to be corrected to correspont to the dataset description

```
print (levels(nursery_data$parents))
```

```
## [1] "great_pret"  "pretentious" "usual"
```

```
print (levels(nursery_data$has_nurs))
```

```
## [1] "critical"    "improper"    "less_proper" "proper"      "very_crit"
```

```
print (levels(nursery_data$form))
```

```
## [1] "complete"   "completed"  "foster"     "incomplete"
```

```
print (levels(nursery_data$children))
```

```
## [1] "1"    "2"    "3"    "more"
```

```
print (levels(nursery_data$housing))
```

```
## [1] "convenient" "critical"   "less_conv"
```

```
print (levels(nursery_data$finance))
```

```
## [1] "convenient" "inconv"
```

```
print (levels(nursery_data$social))
```

```
## [1] "nonprob"       "problematic"   "slightly_prob"
```

```
print (levels(nursery_data$health))
```

```
## [1] "not_recom"   "priority"    "recommended"
```

```
print (levels(nursery_data$class))
```

```
## [1] "not_recom"  "priority"   "recommend"  "spec_prior" "very_recom"
```

Correction:

```
nursery_data$parents <- factor(nursery_data$parents,levels(nursery_data$parents)[c(3,2,1)])
nursery_data$has_nurs <- factor(nursery_data$has_nurs,levels(nursery_data$has_nurs)[c(4,3,2,1,5)])
nursery_data$form <- factor(nursery_data$form,levels(nursery_data$form)[c(1,2,4,3)])
nursery_data$children <- factor(nursery_data$children,levels(nursery_data$children)[c(1,2,3,4)])
nursery_data$housing <- factor(nursery_data$housing,levels(nursery_data$housing)[c(1,3,2)])
nursery_data$finance <- factor(nursery_data$finance,levels(nursery_data$finance)[c(1,2)])
nursery_data$social <- factor(nursery_data$social,levels(nursery_data$social)[c(1,3,2)])
nursery_data$health <- factor(nursery_data$health,levels(nursery_data$health)[c(1,3,2)])
nursery_data$class <- factor(nursery_data$class,levels(nursery_data$class)[c(1,3,5,2,4)])
```

Corrected levels, now correspont to the dataset description

```
print (levels(nursery_data$parents))
```

```
## [1] "usual"       "pretentious" "great_pret"
```

```
print (levels(nursery_data$has_nurs))
```

```
## [1] "proper"      "less_proper" "improper"    "critical"    "very_crit"
```

```
print (levels(nursery_data$form))
```

```
## [1] "complete"   "completed"  "incomplete" "foster"
```

```
print (levels(nursery_data$children))
```

```
## [1] "1"    "2"    "3"    "more"
```

```
print (levels(nursery_data$housing))
```

```
## [1] "convenient" "less_conv"  "critical"
```

```
print (levels(nursery_data$finance))
```

```
## [1] "convenient" "inconv"
```

```
print (levels(nursery_data$social))
```

```
## [1] "nonprob"       "slightly_prob" "problematic"
```

```
print (levels(nursery_data$health))
```

```
## [1] "not_recom"   "recommended" "priority"
```

```
print (levels(nursery_data$class))
```

```
## [1] "not_recom"  "recommend"  "very_recom" "priority"   "spec_prior"
```

Convert to numbers in one step

[Ref] (https://stackoverflow.com/questions/47922184/convert-categorical-variables-to-numeric-in-r)

```
data <- data.matrix(nursery_data)
head(data)
```

```
##      parents has_nurs form children housing finance social health class
## [1,]       1        1    1        1       1       1      1      2     2
## [2,]       1        1    1        1       1       1      1      3     4
## [3,]       1        1    1        1       1       1      1      1     1
## [4,]       1        1    1        1       1       1      2      2     2
## [5,]       1        1    1        1       1       1      2      3     4
## [6,]       1        1    1        1       1       1      2      1     1
```

Preparing scaled dataand split into train and test

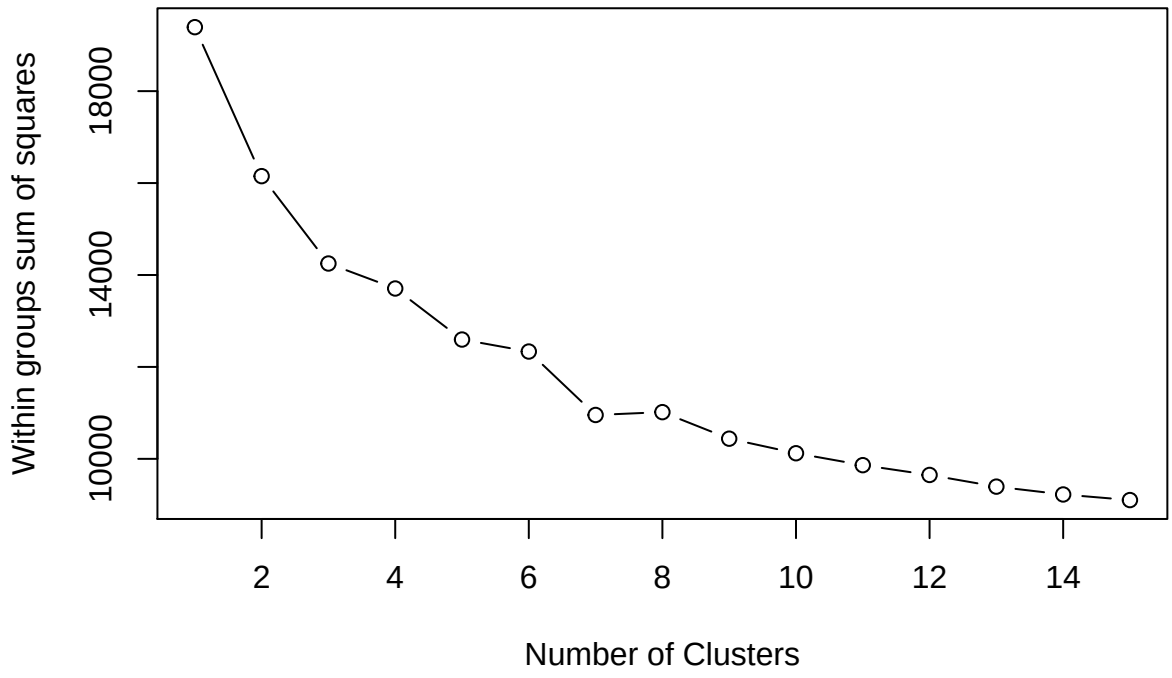
```
index <- sample(1:nrow(data),round(0.75*nrow(data)))
#index <- createDataPartition(y= data$QLT, p=0.5, list = FALSE)
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
scaled <- as.data.frame(scale(data, center = mins, scale = maxs - mins))
train_ <- scaled[index,]
test_ <- scaled[-index,]
```

Clustering

A fundamental question is how to determine the value of the parameter k. If we looks at the percentage of variance explained as a function of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the 'elbow criterion'.

```
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(scaled, nc=15)
```



Clustering using K-means method

```
set.seed(420)
clusters_num =5
k.means.fit <- kmeans(scaled, clusters_num,iter.max = 1000)
```

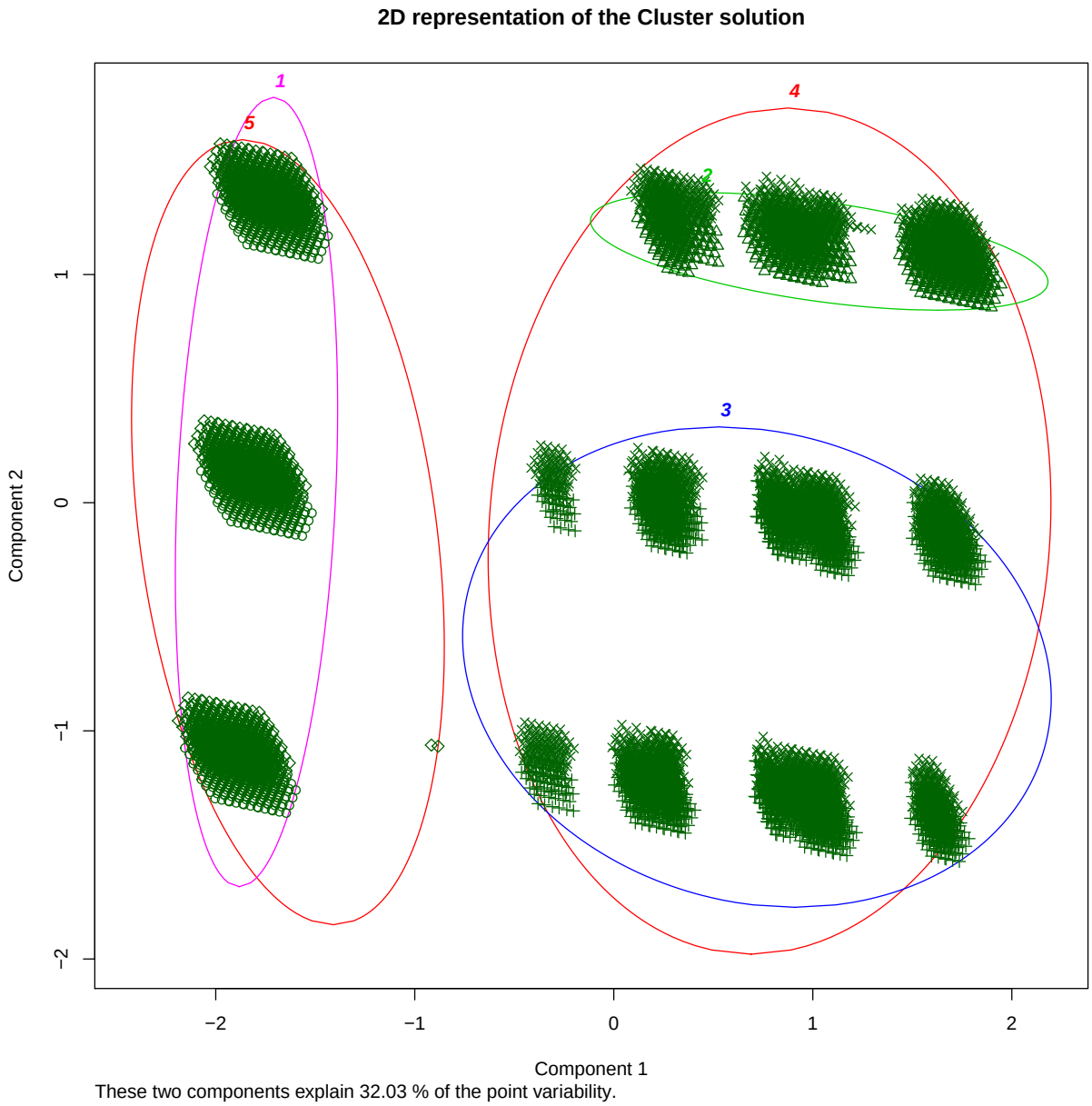
```r
# attributes(k.means.fit)
k.means.fit$centers
```

```
##     parents  has_nurs      form  children   housing finance    social
## 1 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 2 1.0000000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 3 0.2500000 0.5000000 0.5000000 0.5000000 0.5000000       1 0.5000000
## 4 0.5002316 0.5002316 0.5002316 0.5002316 0.5002316       0 0.5001158
## 5 0.4995375 0.4995375 0.4995375 0.4995375 0.4995375       0 0.4997687
##         health        class
## 1 0.0000000000 0.0000000000
## 2 0.7500000000 0.9399305556
## 3 0.7500000000 0.8354166667
## 4 0.7501157943 0.8448355720
## 5 0.0004625347 0.0002312673
```

```r
# plot(k.means.fit$centers[,c("RS","ALC")])
# k.means.fit$cluster
k.means.fit$size
```

```
## [1] 2160 1440 2880 4318 2162
```

```r
library(cluster)
clusplot(scaled, k.means.fit$cluster, main='2D representation of the Cluster solution',
         color=TRUE, shade=FALSE,
         labels=clusters_num, lines=0)
```



These two components explain 32.03 % of the point variability.

Explain clusters

Explain by 'class'

Let's try to explain clusters by the 'class'. Code below builds a matrix whe columns are cluster numbers and rows are target classes.

```
table(nursery_data$class,k.means.fit$cluster)
```

```
##
##                1    2    3    4    5
##   not_recom 2160    0    0    0 2160
##   recommend    0    0    0    0    2
```

```
##   very_recom    0    0  110  218    0
##   priority      0  346 1676 2244    0
##   spec_prior    0 1094 1094 1856    0
```

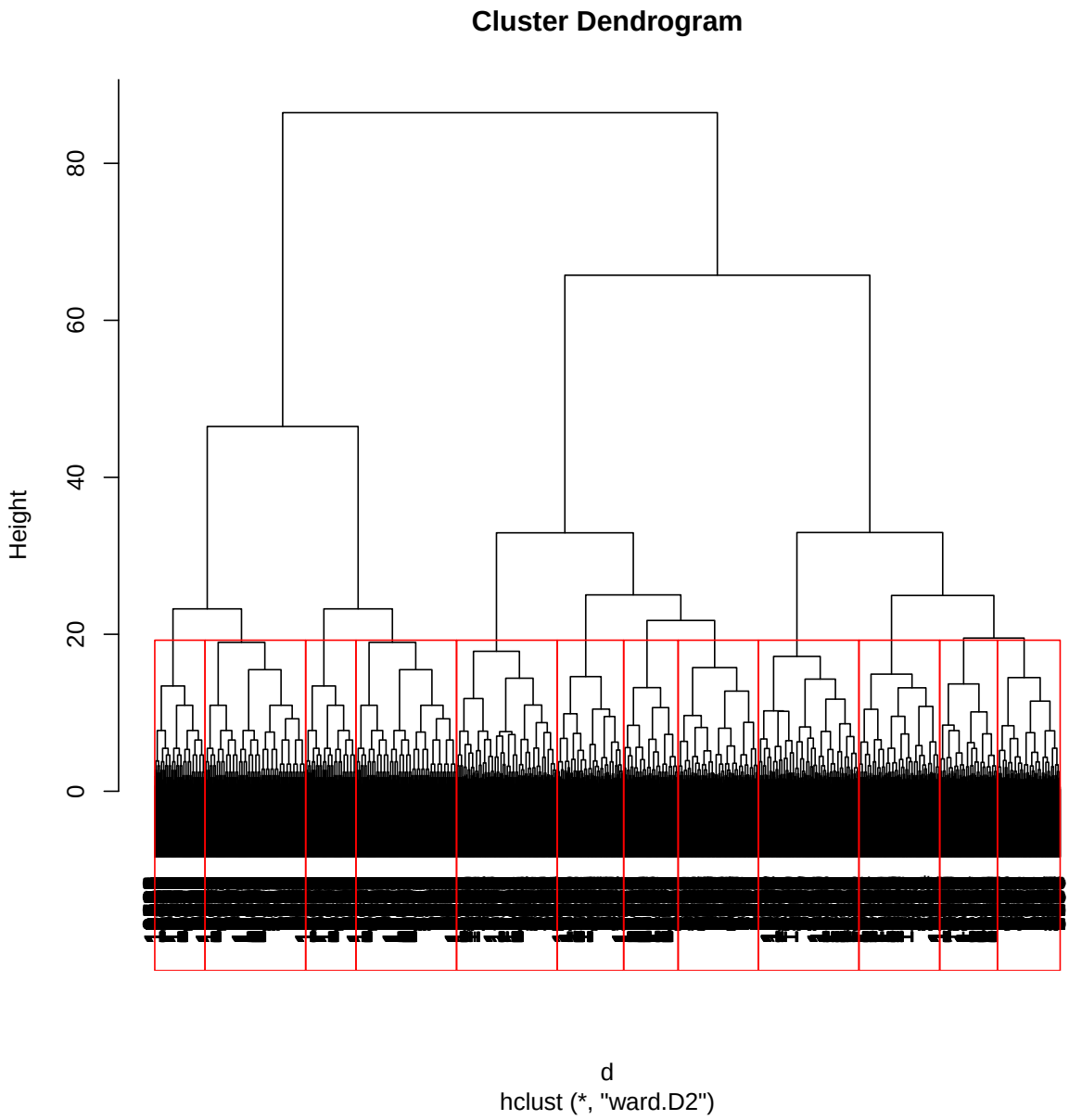
Hierarchical Clustering

Hierarchical methods uses a distance matrix as an input for the clustering algorithm. The choice of an appropriate metric will influence the shape of the clusters, as some element may be close to one another according to one distance and farther away according to another. We use the Euclidean distance as an input for the clustering algorithm ward.2D minimum variance criterion minimizes the total within-cluster variance:

```
d <- dist(scaled, method = "euclidean")
H.fit <- hclust(d, method="ward.D2")
```

The clustering output can be displayed in a dendrogram

```
clusters_num = 12
plot(H.fit)
groups <- cutree(H.fit, k=clusters_num)
rect.hclust(H.fit, k=clusters_num, border="red")
```



The clustering performance can be evaluated with the aid of a confusion matrix as follows:

```
table(nursery_data$class,groups)
```

```
##              groups
##                 1    2    3    4    5    6    7    8    9   10   11   12
##   not_recom     0 1440    0  720    0 1440    0  720    0    0    0    0
##   recommend     2    0    0    0    0    0    0    0    0    0    0    0
##   very_recom  174    0    0    0  100    0    0    0   44   10    0    0
##   priority    704    0  668    0  682    0  568    0  506  448  366  324
##   spec_prior  276    0  772    0  366    0  872    0  348  494  460  456
```