Untitled

2023-12-01

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
Cereals <- read.csv("C:\\Users\\13308\\OneDrive\\Documents\\64060_-spavliga\\Cereals.csv")

#1

data <- Cereals

data <- data[,-3]
data <- data[,-2]
data <- data[,-1]

sum(is.na(data))
```

```
## [1] 4
```

```
data <- na.omit(data)

sum(is.na(data))
```

```
## [1] 0
```

```
head(data)
```

```
##   calories protein fat sodium fiber carbo sugars potass vitamins shelf weight
## 1       70       4   1    130  10.0   5.0      6    280       25     3      1
## 2      120       3   5     15   2.0   8.0      8    135        0     3      1
## 3       70       4   1    260   9.0   7.0      5    320       25     3      1
## 4       50       4   0    140  14.0   8.0      0    330       25     3      1
## 6      110       2   2    180   1.5  10.5     10     70       25     1      1
## 7      110       2   0    125   1.0  11.0     14     30       25     2      1
##   cups   rating
## 1 0.33 68.40297
## 2 1.00 33.98368
## 3 0.33 59.42551
## 4 0.50 93.70491
## 6 0.75 29.50954
## 7 1.00 33.17409
```

```r
datanorm <- scale(data)
summary(datanorm)
```

```
##     calories          protein              fat              sodium       
##  Min.   :-2.8738   Min.   :-1.40687   Min.   :-0.9932   Min.   :-1.9616  
##  1st Qu.:-0.3541   1st Qu.:-0.47733   1st Qu.:-0.9932   1st Qu.:-0.3306  
##  Median : 0.1498   Median :-0.01256   Median : 0.0000   Median : 0.2131  
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.1498   3rd Qu.: 0.45221   3rd Qu.: 0.0000   3rd Qu.: 0.6661  
##  Max.   : 2.6695   Max.   : 3.24083   Max.   : 3.9729   Max.   : 1.9045  
##      fiber              carbo              sugars            potass       
##  Min.   :-0.89778   Min.   :-2.50014   Min.   :-1.6306   Min.   :-1.1783  
##  1st Qu.:-0.79462   1st Qu.:-0.70143   1st Qu.:-0.9424   1st Qu.:-0.8079  
##  Median :-0.07249   Median :-0.05903   Median :-0.0248   Median :-0.1201  
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.34015   3rd Qu.: 0.58337   3rd Qu.: 0.8928   3rd Qu.: 0.3031  
##  Max.   : 4.87925   Max.   : 2.12512   Max.   : 1.8104   Max.   : 3.2660  
##     vitamins           shelf              weight             cups        
##  Min.   :-1.3032   Min.   :-1.4617   Min.   :-3.4600   Min.   :-2.4251  
##  1st Qu.:-0.1818   1st Qu.:-1.1612   1st Qu.:-0.2008   1st Qu.:-0.6432  
##  Median :-0.1818   Median :-0.2599   Median :-0.2008   Median :-0.3038  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.:-0.1818   3rd Qu.: 0.9420   3rd Qu.:-0.2008   3rd Qu.: 0.7568  
##  Max.   : 3.1822   Max.   : 0.9420   Max.   : 3.0583   Max.   : 2.8780  
##      rating       
##  Min.   :-1.7336  
##  1st Qu.:-0.7071  
##  Median :-0.1510  
##  Mean   : 0.0000  
##  3rd Qu.: 0.5807  
##  Max.   : 3.6578
```

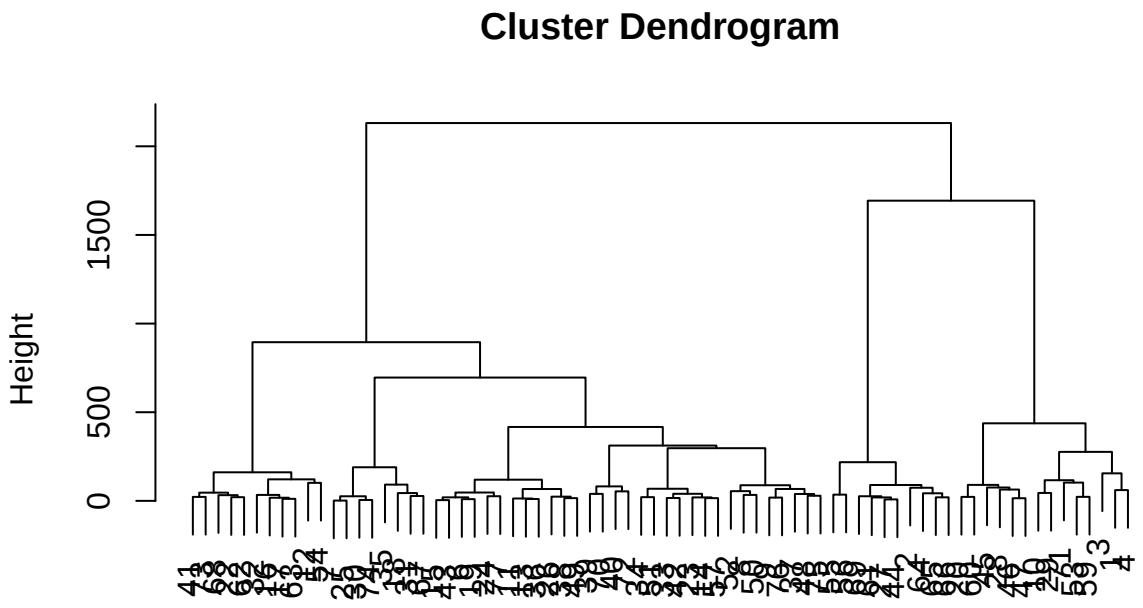
```r
str(data)
```

```
## 'data.frame':    74 obs. of  13 variables:
##  $ calories: int  70 120 70 50 110 110 130 90 90 120 ...
##  $ protein : int  4 3 4 4 2 2 3 2 3 1 ...
##  $ fat     : int  1 5 1 0 2 0 2 1 0 2 ...
##  $ sodium  : int  130 15 260 140 180 125 210 200 210 220 ...
##  $ fiber   : num  10 2 9 14 1.5 1 2 4 5 0 ...
##  $ carbo   : num  5 8 7 8 10.5 11 18 15 13 12 ...
##  $ sugars  : int  6 8 5 0 10 14 8 6 5 12 ...
##  $ potass  : int  280 135 320 330 70 30 100 125 190 35 ...
##  $ vitamins: int  25 0 25 25 25 25 25 25 25 25 ...
##  $ shelf   : int  3 3 3 3 1 2 3 1 3 2 ...
##  $ weight  : num  1 1 1 1 1 1 1.33 1 1 1 ...
##  $ cups    : num  0.33 1 0.33 0.5 0.75 1 0.75 0.67 0.67 0.75 ...
##  $ rating  : num  68.4 34 59.4 93.7 29.5 ...
##  - attr(*, "na.action")= 'omit' Named int [1:3] 5 21 58
##   ..- attr(*, "names")= chr [1:3] "5" "21" "58"
```

```r
datatest <- dist(data, method = "euclidean")
dataclust <- hclust(datatest, method = "ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```r
# plot(dataclust, cex = 0.6, hang = -1)
plot(dataclust)
```




```r
library(cluster)

datasingle <- agnes(data, method = "single")
datacomplete <- agnes(data, method= "complete")
dataaverage <- agnes(data, method = "average")
dataward <- agnes(data, method = "ward")

print(datasingle$ac)
```

```
## [1] 0.7311616
```

```r
print(datacomplete$ac)
```

```
## [1] 0.922957
```

```r
print(dataaverage$ac)
```

```
## [1] 0.8792621
```

```r
print(dataward$ac)
```

```
## [1] 0.9597071
```

```r
# We would choose ward as it results in the highest agglomerative coefficient.

datacut <- cutree(dataclust, k = 4)
datacut
```

```
##  1  2  3  4  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 25 26 27 28
##  1  2  1  1  3  3  3  3  1  3  4  3  3  3  4  4  3  3  1  4  3  3  3  3  2  1
## 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
##  1  3  3  4  3  3  3  3  3  3  3  3  4  3  3  2  1  1  1  3  3  3  3  3  1  4
## 55 56 57 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
##  2  2  3  1  1  2  4  4  2  2  2  3  4  2  3  1  3  4  3  3  3  3
```

```r
pltree(dataward, cex = 0.6, hang = -1, main = "Dendrogram")
rect.hclust(dataward, k = 4, border = 1:4)

# 2
# I am choosing 4 clusters, but noting that the front and back clusters stay
# roughly the same when k = 3, 4 or 5. The values inbetween the aforementioned
# clusters become 1-3 clusters depending on k being 3, 4 or 5.

# 3

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

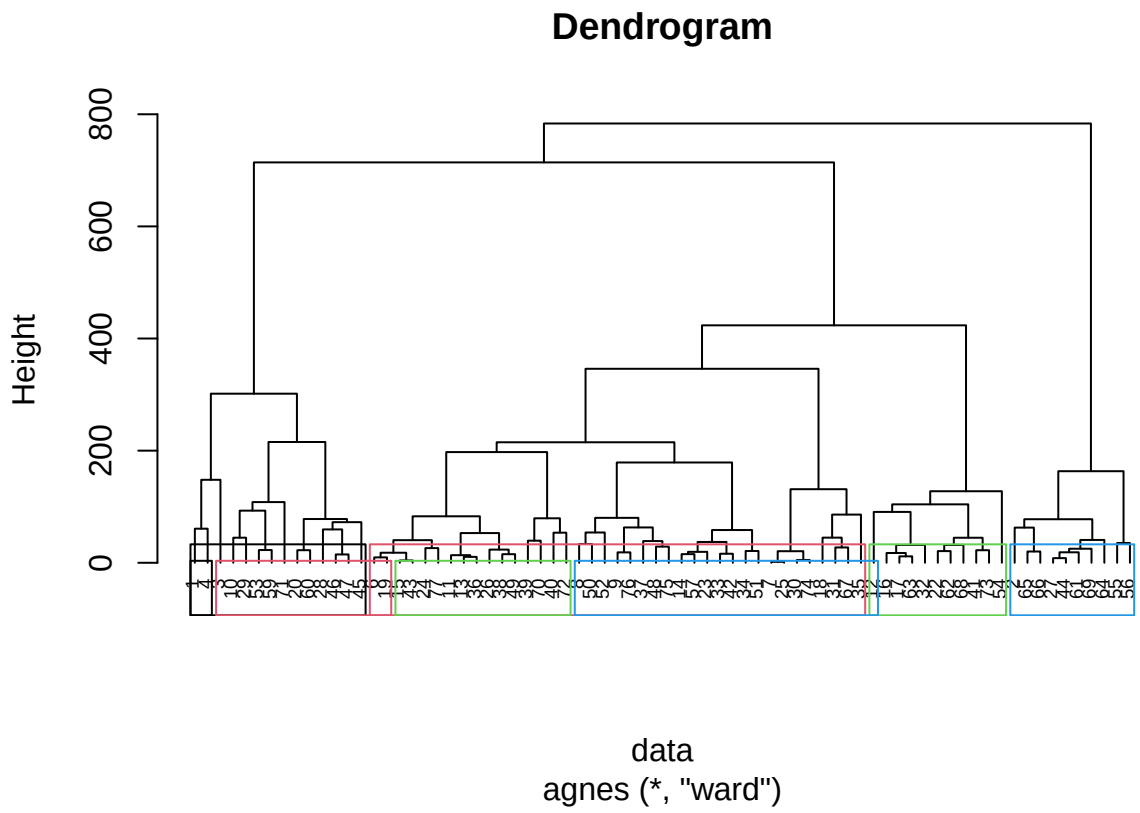
```
## Loading required package: lattice
```

```r
library(class)
```

```
## Warning: package 'class' was built under R version 4.2.3
```

```r
set.seed(123)

IndexData <- createDataPartition(data$rating, p=0.7, list=FALSE)
DataPart7 <- data[IndexData,]
DataPart3 <- data[-IndexData,]
```

```
DataPart7 <- as.data.frame(DataPart7)
DataPart3 <- as.data.frame(DataPart3)

DataPart7 <- scale(DataPart7)
DataPart3 <- scale(DataPart3)

dataward7 <- agnes(DataPart7, method = "ward")
rect.hclust(dataward7, k = 4, border = 1:4)
```




```
d<-dist(DataPart7)
fit.ward<-hclust(d, method = "ward")
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
clusters<-cutree(fit.ward, k=4)

clustercenter <- aggregate(DataPart7,list(cluster=clusters),mean)
clustercenter[,2:14]
```

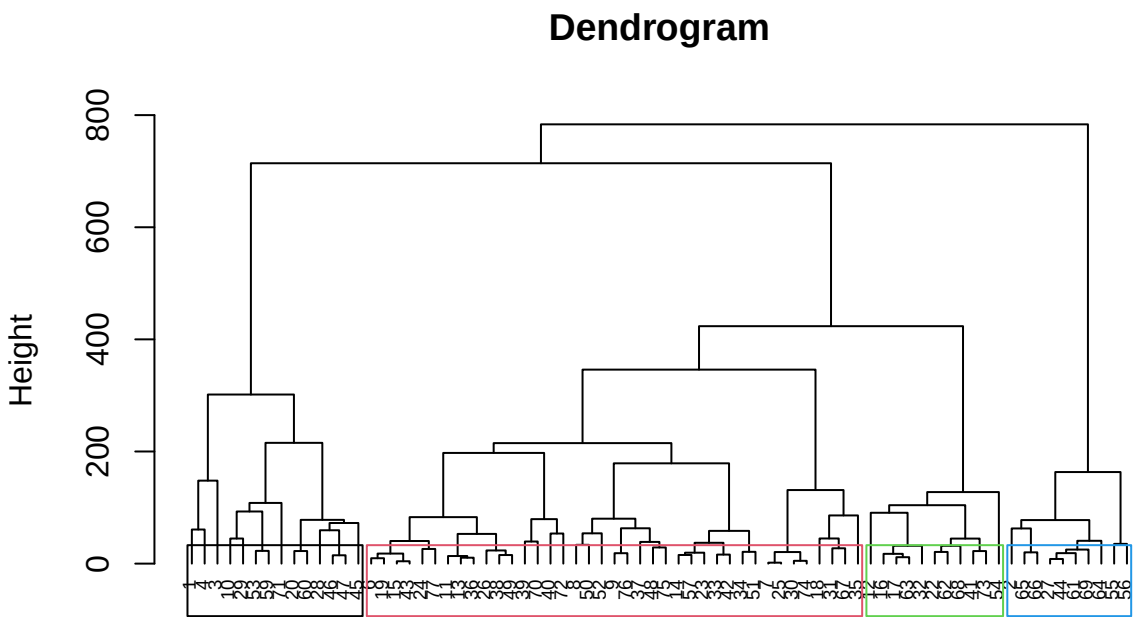
```
##       calories     protein         fat      sodium       fiber      carbo
## 1  0.676039939  0.62851038  0.88154309 -0.06670409  1.00166593 -0.3459942
## 2  0.159595635 -0.92202792  0.07259767  0.14473529 -0.67554214 -0.5957761
## 3 -0.006404319  0.39880101 -0.58522608  1.16687207 -0.67323828  1.3156502
```

```
## 4 -0.734688434  0.09477389 -0.60091345 -0.54932779 -0.01054052  0.2892355
##       sugars     potass   vitamins      shelf     weight       cups      rating
## 1  0.4727782  1.1922410  0.1380093  0.8041672  0.8665950 -0.6072387 -0.05335428
## 2  0.9817880 -0.6777884 -0.2168718 -0.5423453 -0.1797086  0.1876919 -1.06226961
## 3 -1.0951537 -0.7313607 -0.2168718 -1.4560502 -0.1797086  1.1510721  0.24871606
## 4 -0.8324951 -0.1527802  0.1588847  0.3685308 -0.5320785 -0.1037826  0.88196153
```

```
assignedclusterlabels <- character(nrow(DataPart3))

for (i in 1:nrow(DataPart3)) {
  distances <- apply(clustercenter[, -1], 1, function(c) sqrt(sum((unlist(DataPart3[i,]) - c)^2)))
  closestcluster <- which.min(distances)
  assignedclusterlabels[i] <- as.character(clustercenter$cluster[closestcluster])
}

for (i in 1:nrow(DataPart3)) {
  cat("Record", paste(DataPart3[i,], collapse = " "), "assigned to Cluster", assignedclusterlabels[i],
}
```

```
## Record -3.06488136113741 1.87747262960169 -1.04586074656009 -0.205582433755664 3.52752070562171 -1.67
## Record 0.437840194448202 -0.545072698916619 1.41498806887541 0.392475555351723 -0.391946745069079 -0
## Record 0.437840194448202 -0.545072698916619 -1.04586074656009 -0.429854179670935 -0.548725443096711 -
## Record -0.72973365741367 -0.545072698916619 0.184563661157662 0.691504549905417 0.391946745069079 0.1
## Record -0.72973365741367 0.666199965342534 -1.04586074656009 0.841019047182263 0.705504141124343 -0.1
## Record 0.437840194448202 0.666199965342534 1.41498806887541 -0.205582433755664 -0.235168047041448 -0
## Record 0.437840194448202 -0.545072698916619 -1.04586074656009 0.99053354445911 -0.548725443096711 1.9
## Record 0.437840194448202 -1.75634536317577 -1.04586074656009 0.691504549905417 -0.548725443096711 0.0
## Record 1.02162712037914 0.666199965342534 1.41498806887541 0.0934465607980293 0.705504141124343 -0.54
## Record 0.437840194448202 -1.75634536317577 0.184563661157662 -0.280339682394088 -0.862282839151974 -0
## Record -0.145946731482734 -0.545072698916619 -1.04586074656009 -1.62597015788571 -0.862282839151974 -
## Record -0.145946731482734 1.87747262960169 1.41498806887541 -0.0560679364788176 -0.235168047041448 -0
## Record 0.437840194448202 -0.545072698916619 0.184563661157662 0.392475555351723 -0.862282839151974 -0
## Record -0.145946731482734 -0.545072698916619 0.184563661157662 0.99053354445911 -0.235168047041448 0.
## Record 1.60541404631007 0.666199965342534 1.41498806887541 0.242961058074876 -0.391946745069079 -0.1
## Record 1.02162712037914 0.666199965342534 0.184563661157662 0.691504549905417 1.01906153717961 -0.83
## Record -1.31352058334461 -0.545072698916619 -1.04586074656009 -2.29878539563152 0.0783893490138158 0.
## Record -0.72973365741367 0.666199965342534 -1.04586074656009 -2.29878539563152 0.0783893490138158 1.7
## Record 0.437840194448202 -0.545072698916619 0.184563661157662 0.691504549905417 -0.862282839151974 1.
## Record -0.145946731482734 0.666199965342534 0.184563661157662 0.691504549905417 0.0783893490138158 0.
```

```
# Cluster Assessment

pltree(dataward, cex = 0.6, hang = -1, main = "Dendrogram")
rect.hclust(dataward, k = 4, border = 1:4)
```




```
# In review, only 6 out of 20 values were correctly assigned. With a 30%
# accuracy, this is not very consistent and would indicate a rather low
# stability.

#4

#I would be inclined to not normalize the data as we are focused on the
#real-world implications and the real measurements of the variables as opposed
#to the normalized ones, however it does open up the risk of our results not
#being entirely correct. Regarding the school lunch example, I am assuming that
#the cluster or dendrogram info will be given to the lunch staff. I wouldn't
#expect the lunch staff to know what to do with the normalized values. Being
#said, some options would be to perform calculations with the normalized data
#and simply provide the dendrogram results or just provide the cereals that
#make up the best cluster.

#It would also be possible to avoid clustering entirely and evaluate
#which cereals to choose based off rating, which does appear
#to be representative of the general health of the cereal.
```