Assignment_5

wliu16

2022-12-01

Data preprocess

```r
cereal <- read.csv("Cereals.csv")
cereal <- na.omit(cereal) # 3 NA records removed, 74 records in total
summary(cereal) #original data summary
```

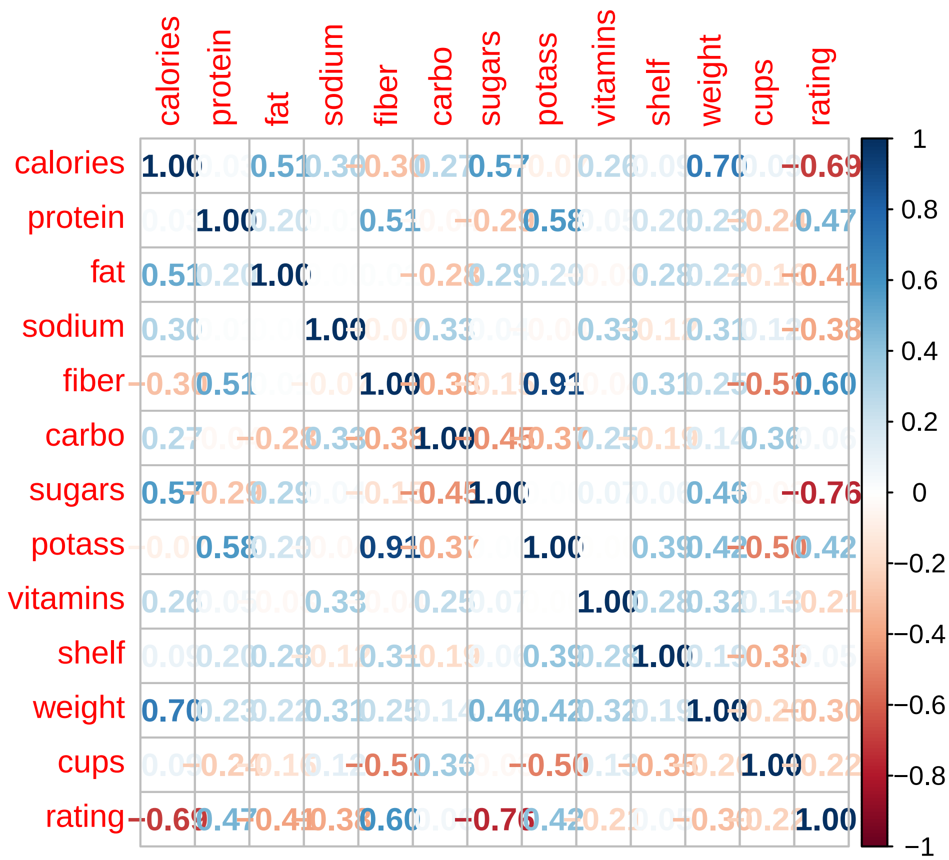
```
##      name               mfr                type              calories
##  Length:74          Length:74          Length:74          Min.   : 50
##  Class :character   Class :character   Class :character   1st Qu.:100
##  Mode  :character   Mode  :character   Mode  :character   Median :110
##                                                           Mean   :107
##                                                           3rd Qu.:110
##                                                           Max.   :160
##     protein           fat        sodium          fiber            carbo
##  Min.   :1.000   Min.   :0   Min.   :  0.0   Min.   : 0.000   Min.   : 5.00
##  1st Qu.:2.000   1st Qu.:0   1st Qu.:135.0   1st Qu.: 0.250   1st Qu.:12.00
##  Median :2.500   Median :1   Median :180.0   Median : 2.000   Median :14.50
##  Mean   :2.514   Mean   :1   Mean   :162.4   Mean   : 2.176   Mean   :14.73
##  3rd Qu.:3.000   3rd Qu.:1   3rd Qu.:217.5   3rd Qu.: 3.000   3rd Qu.:17.00
##  Max.   :6.000   Max.   :5   Max.   :320.0   Max.   :14.000   Max.   :23.00
##      sugars           potass          vitamins          shelf
##  Min.   : 0.000   Min.   : 15.00   Min.   :  0.00   Min.   :1.000
##  1st Qu.: 3.000   1st Qu.: 41.25   1st Qu.: 25.00   1st Qu.:1.250
##  Median : 7.000   Median : 90.00   Median : 25.00   Median :2.000
##  Mean   : 7.108   Mean   : 98.51   Mean   : 29.05   Mean   :2.216
##  3rd Qu.:11.000   3rd Qu.:120.00   3rd Qu.: 25.00   3rd Qu.:3.000
##  Max.   :15.000   Max.   :330.00   Max.   :100.00   Max.   :3.000
##      weight           cups            rating
##  Min.   :0.500   Min.   :0.2500   Min.   :18.04
##  1st Qu.:1.000   1st Qu.:0.6700   1st Qu.:32.45
##  Median :1.000   Median :0.7500   Median :40.25
##  Mean   :1.031   Mean   :0.8216   Mean   :42.37
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:50.52
##  Max.   :1.500   Max.   :1.5000   Max.   :93.70
```

```r
head(cereal)  #original data snapshot
```

```
##                        name mfr type calories protein fat sodium fiber carbo
## 1                 100%_Bran   N    C       70       4   1    130  10.0   5.0
## 2         100%_Natural_Bran   Q    C      120       3   5     15   2.0   8.0
## 3                  All-Bran   K    C       70       4   1    260   9.0   7.0
## 4 All-Bran_with_Extra_Fiber   K    C       50       4   0    140  14.0   8.0
## 6   Apple_Cinnamon_Cheerios   G    C      110       2   2    180   1.5  10.5
## 7                Apple_Jacks   K    C      110       2   0    125   1.0  11.0
```

```
##   sugars potass vitamins shelf weight cups   rating
## 1      6    280       25     3      1 0.33 68.40297
## 2      8    135        0     3      1 1.00 33.98368
## 3      5    320       25     3      1 0.33 59.42551
## 4      0    330       25     3      1 0.50 93.70491
## 6     10     70       25     1      1 0.75 29.50954
## 7     14     30       25     2      1 1.00 33.17409
```

```
corrmatrix <- cor(cereal[, 4:16])
corrplot(corrmatrix, method = 'number')
```



```
#data scaling
df <- cereal[, 4:16]
df_scaled<- scale(df)
rownames(df_scaled) <- cereal[, 1] #create new dataframe with only numerical data
```

Early observations on the univariate data: there might be outliers on high or low ends
protein: outliers on max
fat: outliers on max
sodium: outliers on min
fiber: outliers on max
potass: outliers on max
vitamins: outliers on max
rating: vitamins: outliers on max

Hierarchical clustering

```r
d <- dist(df_scaled, method = "euclidean")
hc1 <- agnes(d, method = "complete")
hc2 <- agnes(d, method = "single")
hc3 <- agnes(d, method = "average")
hc4 <- agnes(d, method = "ward")

print(hc1$ac)
```

```
## [1] 0.8353712
```

```r
print(hc2$ac)
```

```
## [1] 0.6067859
```

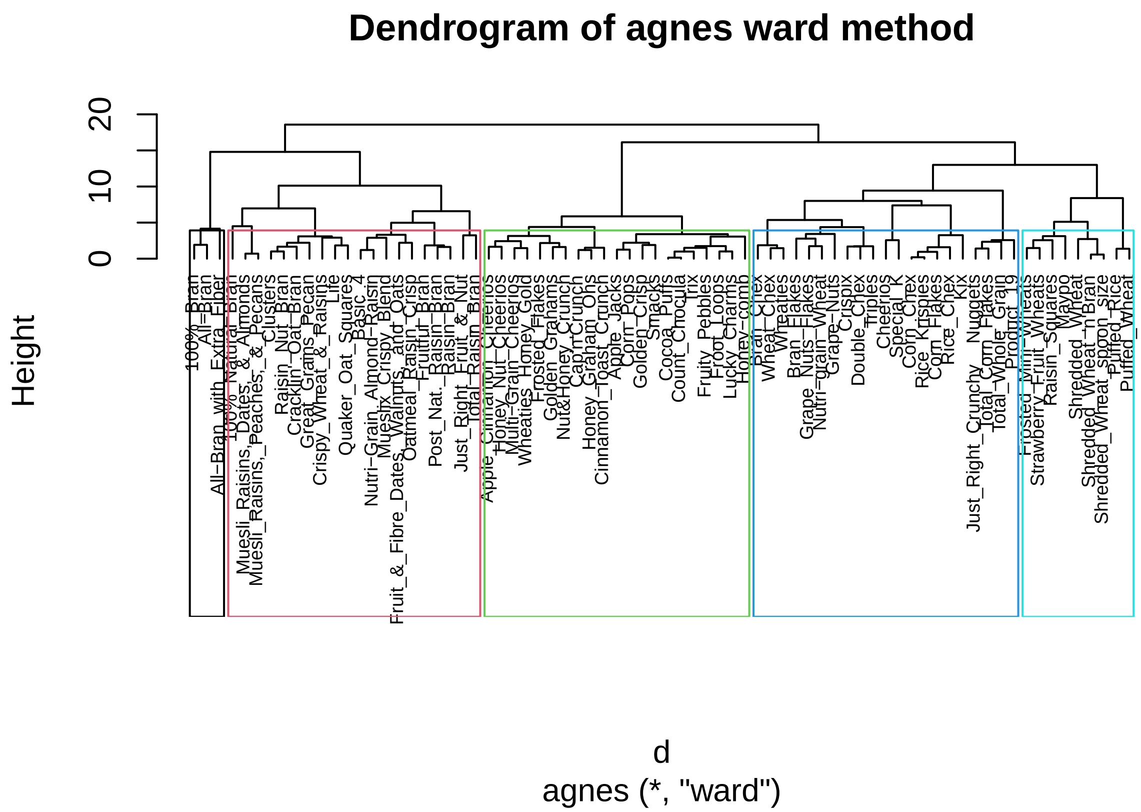
```r
print(hc3$ac)
```

```
## [1] 0.7766075
```

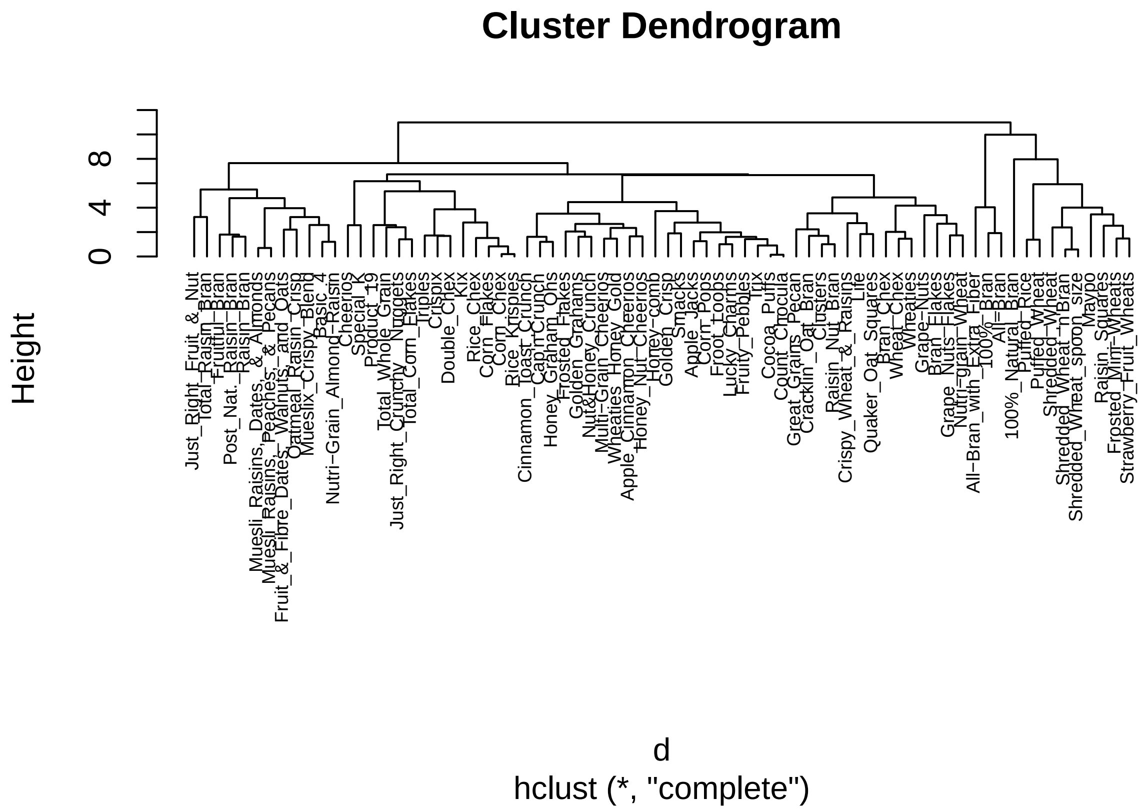
```r
print(hc4$ac)
```

```
## [1] 0.9046042
```

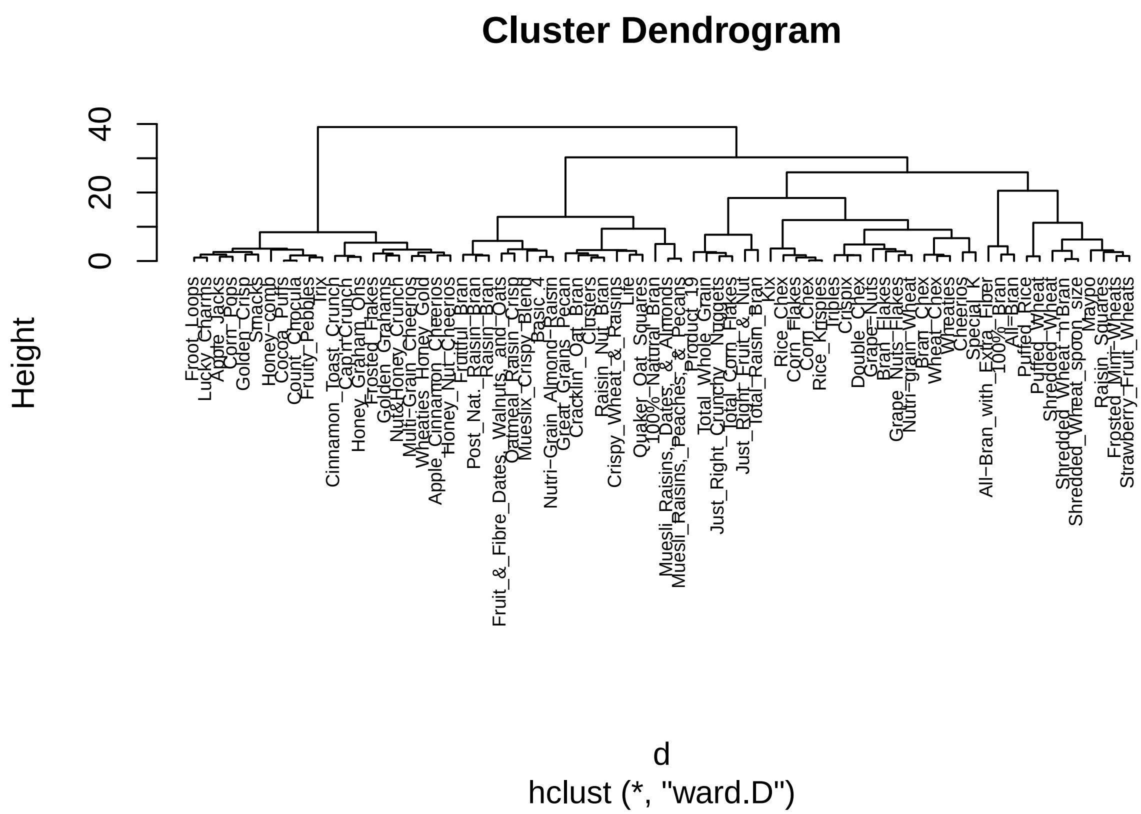
```r
pltree(hc4, cex = 0.6, hang = -1, main = "Dendrogram of agnes ward method")
rect.hclust(hc4, k = 5, border = 1:5)
```




```r
#comparison between complete and ward using hclust
hc_comp <- hclust(d, method = "complete")
hc_w <- hclust(d, method = "ward.D")
plot(hc_comp, cex = 0.6, hang = -1) #complete
```




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



Conclusion
(1) The best method is ward method as the agglomerative coefficient is max among the four methods. Complete method is also a good alternative.

If we use hclust to plot, complete and ward both show clean plots. Complete shows 5 clusters and ward shows 4. For this practice, we will go with ward method as the best method.

(2) I would choose 5 as the **number of clusters**. This is based on visual examination to the ward method and complete method.

```
#test plot of Euclidean distance vs no of clusters optimal no of clusters
```

Check on stability of clusters

```
set.seed(111)

train_index_c <- createDataPartition(df$rating, p= 0.6, list = FALSE)
validate_c <- df[- train_index_c, ]  # 40% as validation
train_c <- df[train_index_c, ] # 60% as training and testing

validate_c <- scale(validate_c)
train_c <- scale(train_c)
```

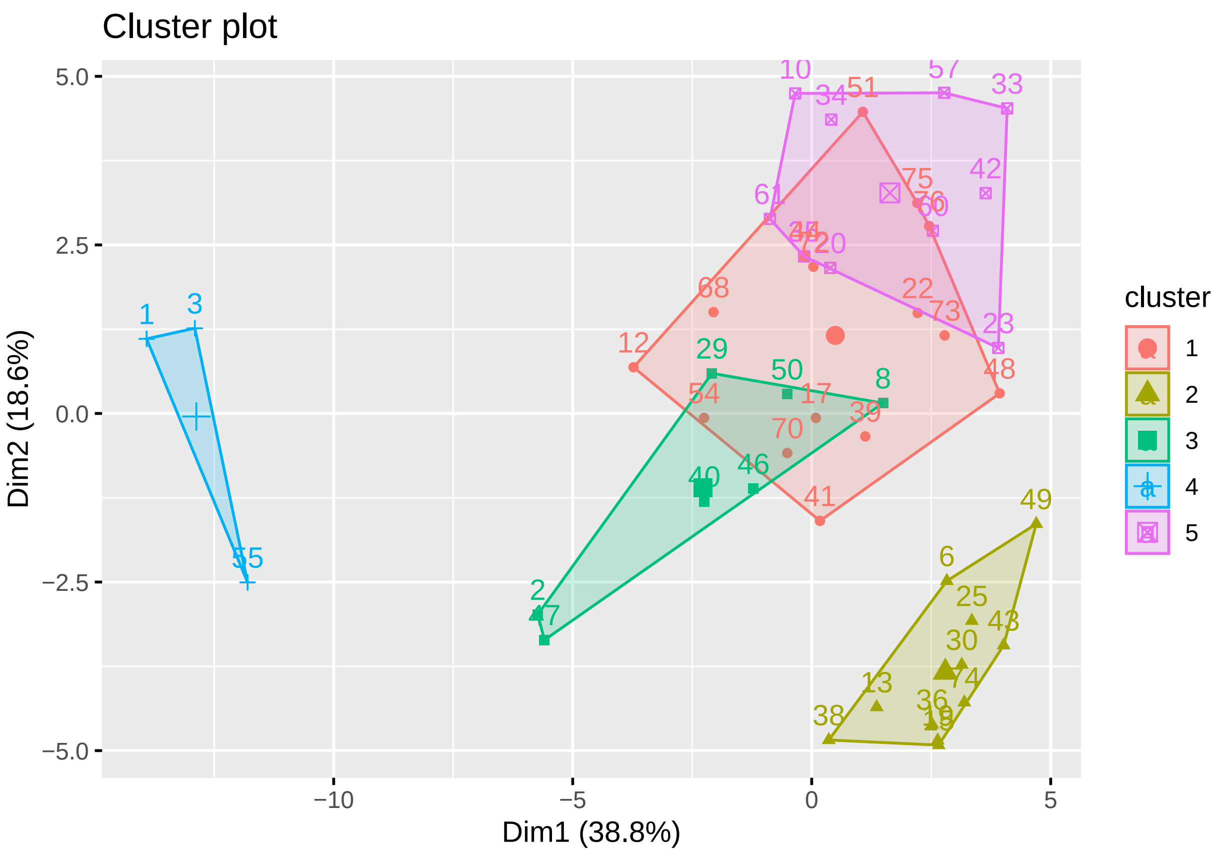
```
d2 <- dist(train_c, method = "euclidean")
d3 <- dist(validate_c, method = "euclidean")

hc_ward_train <- agnes(d2, method = "ward")
hc_ward_validate <- agnes(d3, method = "ward")
```

```
k_t <- kmeans (d2, centers = 5, nstart = 25)
k_t$centers
```

```
##          1        2        3        6        8       10       12       13
## 1 7.967800 6.906590 7.594565 4.704842 4.681772 4.525528 4.331638 5.182774
## 2 8.535030 5.479877 8.309637 2.189860 4.321893 5.338304 6.121088 2.373743
## 3 7.872511 5.146281 7.601316 5.073019 2.808911 5.307493 6.273949 5.033307
## 4 3.677442 8.063725 3.827698 7.578475 8.131495 5.022271 8.116483 8.528609
## 5 5.769457 5.114209 5.739222 4.079760 4.086324 2.963445 5.594050 4.713348
##         15       17       19       20       22       23       25       29
## 1 4.974599 3.441628 4.980762 5.287565 3.251136 4.489539 4.724198 5.569449
## 2 1.423779 4.633822 1.456496 4.582716 4.381968 2.943412 1.765088 5.194076
## 3 5.257400 6.331684 5.226176 4.380366 5.345016 4.471176 5.010991 3.760662
## 4 8.197772 8.081412 8.168678 6.055757 7.576554 6.315716 7.662901 7.171546
## 5 4.665498 5.229181 4.645698 2.669497 4.219611 2.604431 4.068141 4.396180
##         30       33       34       35       36       38       39       40
## 1 4.848704 3.520775 4.401905 5.253916 4.909800 4.801771 3.796492 5.133637
## 2 1.638846 4.117616 5.253468 4.961404 1.758105 2.514624 4.278343 5.368813
## 3 5.188961 4.598047 5.236725 4.651842 4.965305 6.134692 5.164181 3.903471
## 4 7.947479 5.971325 6.301575 6.651761 8.327016 8.654483 7.971297 9.039925
## 5 4.211122 2.382682 2.893132 2.822867 4.684861 5.560833 4.557340 5.274525
##         41       42       43       44       46       47       48       49
## 1 3.611015 3.966265 4.493673 4.227321 5.511264 6.800480 3.344115 4.087701
## 2 4.360610 3.834046 1.469902 5.052422 4.770842 5.772907 3.039714 1.998316
## 3 5.914702 4.627010 4.916140 5.727665 3.535555 3.402447 5.240474 4.611051
## 4 8.600125 6.242160 7.854626 6.935653 8.122302 9.490221 7.101365 7.810482
## 5 5.504342 2.488657 4.171300 3.921031 4.514548 5.803263 3.978618 3.746769
##         50       51       54       55       57       60       61       68
```

```
## 1 4.995514 3.472850 3.889907 7.102305 4.152854 4.776383 5.048985 4.017793
## 2 5.149799 5.231537 5.642161 7.378183 4.501119 3.929099 5.102551 5.612719
## 3 2.955339 5.600655 6.199002 9.531091 4.640665 4.405843 5.806489 6.281182
## 4 8.325595 6.150310 8.391676 6.048828 5.917482 5.930290 6.044087 7.819170
## 5 4.498909 3.522954 5.515583 6.627048 2.135463 2.196201 3.109669 5.061386
##          70        72        73        74        75        76
## 1 3.768592 3.704618 3.415195 4.742373 3.381285 3.155341
## 2 4.986953 5.248200 4.104648 1.478435 4.461652 4.368525
## 3 5.687825 5.359895 4.956150 5.245854 5.364756 5.407231
## 4 8.526692 7.058347 7.674280 8.086594 6.597121 6.722869
## 5 5.167939 4.269072 3.990893 4.472778 3.692208 3.911365
```

```
fviz_cluster(k_t, data = d2)
```

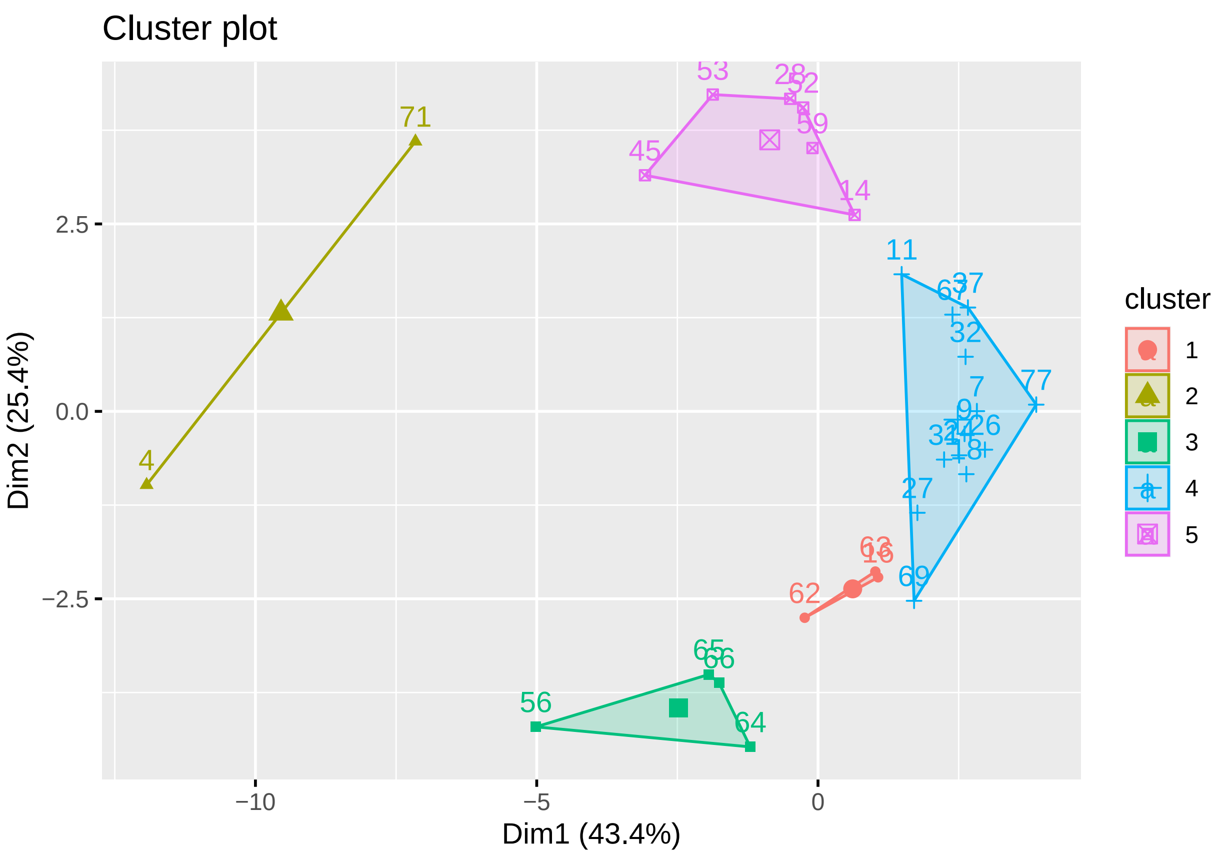



```
k_v <- kmeans (d3, centers = 5, nstart = 25)
k_v$centers
```

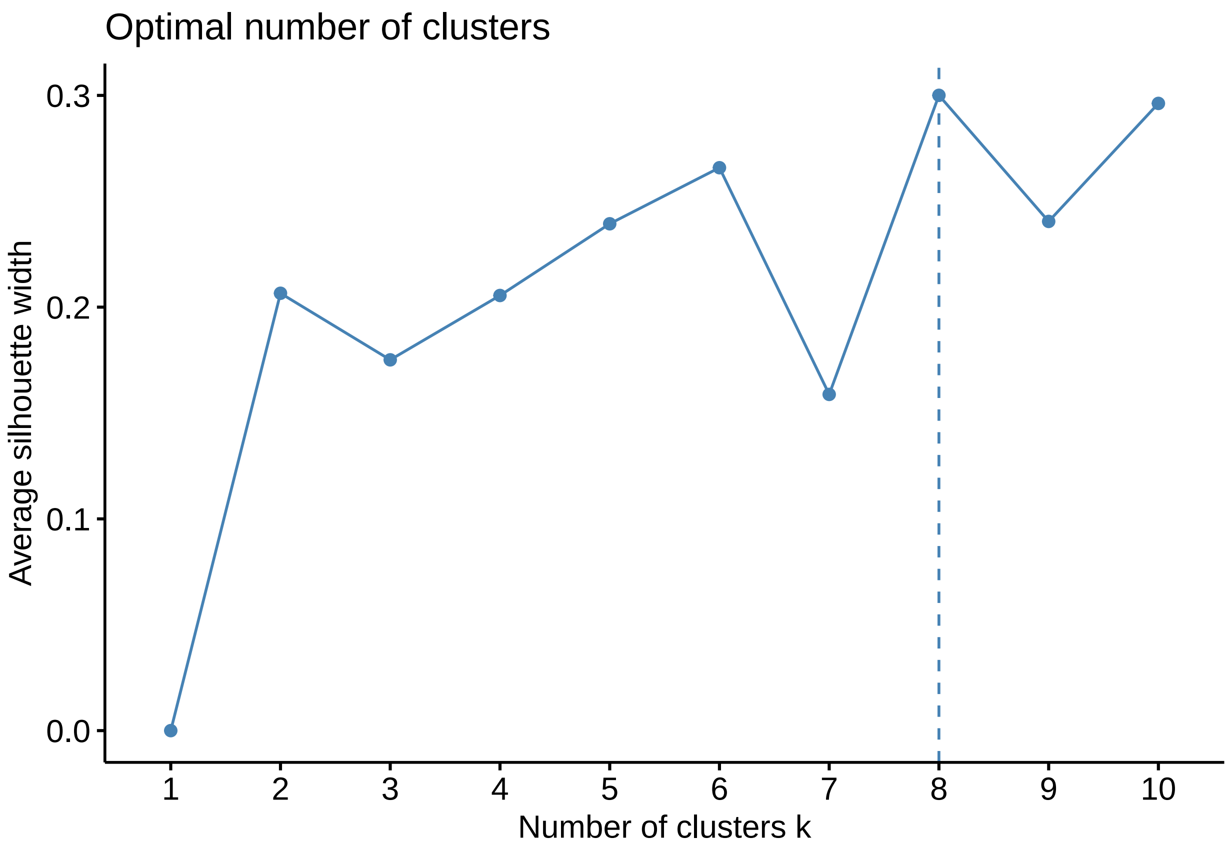
```
##         4        7        9       11       14        16       18       24
## 1 9.601893 4.237441 3.892908 4.809853 5.704819 0.5463308 3.910074 3.408813
## 2 4.833155 7.589936 6.868105 8.265568 6.924806 8.3872985 7.934525 7.132319
## 3 7.158142 5.351790 4.295151 6.601134 5.516432 5.2064151 5.192537 4.547096
## 4 8.303098 2.566362 2.961120 3.170624 3.904608 3.9381749 2.644556 3.063577
## 5 7.484726 4.657260 4.124080 4.398839 2.644996 5.8383736 5.127329 4.281198
##         26        27        28        31        32        37        45        52
## 1 3.460431 4.719206 6.159589 4.638341 3.550839 4.204707 6.432145 5.994403
## 2 8.079935 6.693864 6.083092 7.954386 8.016644 7.237645 7.460904 6.997815
## 3 5.421793 3.511469 6.224397 5.107198 5.971333 5.465410 6.944372 6.606051
## 4 2.574454 3.199756 4.388753 2.745396 2.815178 2.869911 5.416566 4.304330
```

```
## 5 5.099300 4.295649 1.959130 5.105719 4.564692 3.799829 3.391397 2.331039
##         53        56        59         62         63       64       65       66
## 1 6.617732 6.623279 5.455232 0.9922691 0.5579658 4.539289 5.152434 4.936438
## 2 5.898252 9.117837 5.993608 8.8657222 8.3649404 8.269973 7.645951 7.873446
## 3 6.814155 3.667565 6.146125 5.4699166 5.2622484 2.274254 2.156441 2.160669
## 4 4.839153 5.799647 4.219407 4.3864423 3.9610931 4.520096 4.908774 4.859670
## 5 2.455347 7.523936 2.442496 6.5225464 5.8210404 6.391993 5.758519 5.826574
##         67        69        71        77
## 1 5.318919 4.054697 7.476748 3.031851
## 2 7.580398 7.068308 4.833155 7.556155
## 3 5.592962 3.376229 9.295462 4.852509
## 4 2.721186 3.210505 6.770408 2.435456
## 5 4.050874 5.011296 5.634766 4.141704
```

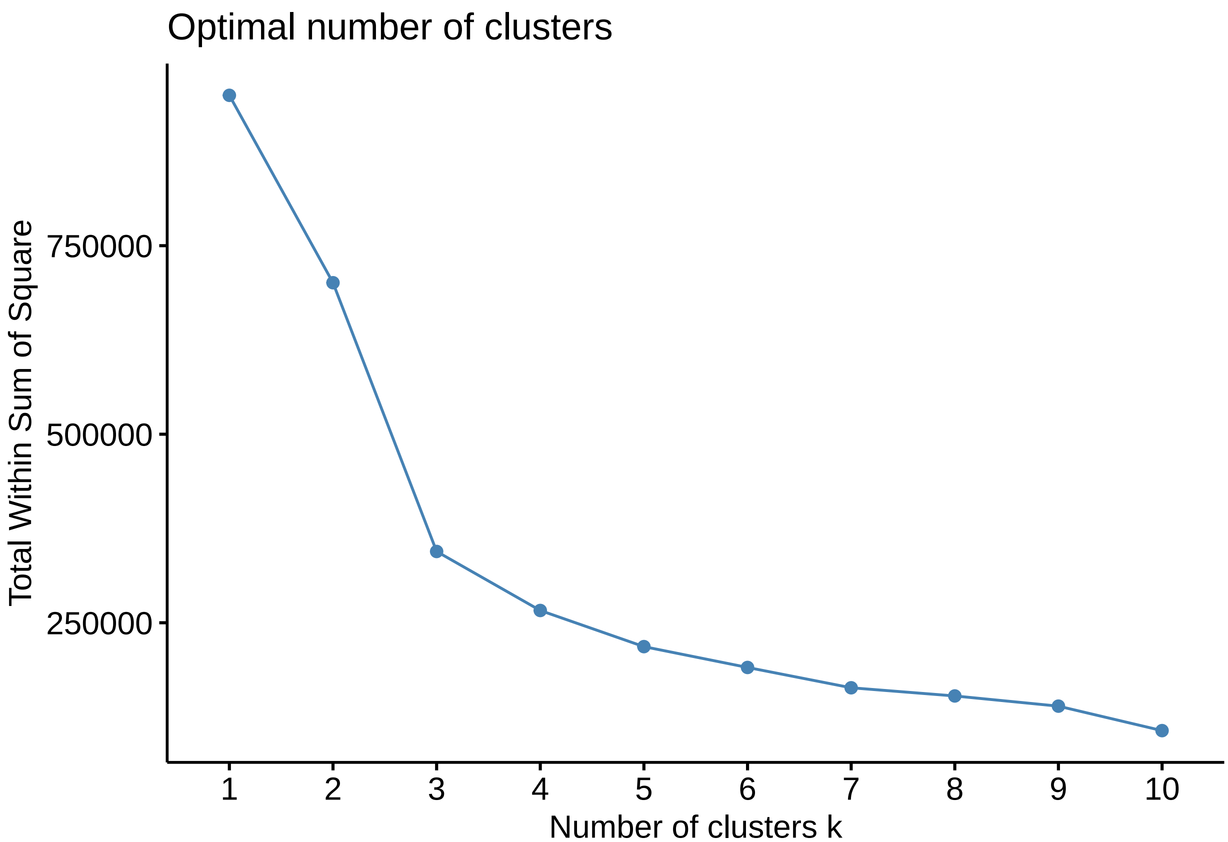
```
fviz_cluster(k_v, data = d3)
```




```
fviz_nbclust(df_scaled, kmeans, method='silhouette')
```




```
fviz_nbclust(df, kmeans, method = "wss")
```




```
# k = 3 and k = 8 are suggested
```

Recommendation to Elementary school Should the data be normalized?
The data should not be normalized as we need to use units to filter healthy cereal.

The standard for healthy cereal should be full of nutrients (fiber) compared to other cereals which are simply tasty. Vitamins and minerals are also something nice to have. According to the official guideline from FDA, cereals have to contain three-fourth ounces of whole grains and no more than 1 gram of saturated fat, 230 milligrams of sodium and 2.5 grams of added sugars in order to be considered as healthy.
Reference: https://www.cnbc.com/2022/10/11/fda-redefined-healthy-these-7-cereals-do-not-qualify.html