M4L2 Homework Assignment

Joshua Conte

October 8, 2017

1 M4L2 Homework Assignment

R studio was configured with the following parameters before beginning the project:

```r
# clears the console in RStudio
cat("\014")
```

```r
# clears environment
rm(list = ls())

# Load required packages
require(ggplot2)
require(cluster)
require(amap)
require(useful)
```

1.1 Load Data.

I opened the Wholesale customers Data Set using read.csv2 and downloaded it directly from the UC Irvine Machine Learning Repository.

To format the data, the data is separated by ‘,’, stringsAsFactors = FALSE so that the strings in a data frame will be treated as plain strings and not as factor variables. I set na strings for missing data. Once the data was loaded I added the column names and changed the data types to numeric and finally removed the text data type.

Below is my R code:

```r
# Some csv files are really big and take a while to open.  This command checks to
# see if it is already opened, if it is, it does not open it again.
# I also omitted the first column
if (!exists("dfWCD")) {
dfWCD <-
  read.csv2("Wholesale customers data.csv",
    sep = ",",
    stringsAsFactors = FALSE,
    na.strings=c("","NA")
  )
# Add a column so I know which study the data is refereing to
study <- sprintf("study_%s",seq(1:440))
dfWCD$study<-study
}

# Download directly from site (unreliable from Ecuador)
# if (!exists("dfWCD")) {
# dfWCD <-
#   read.csv2(
#     url(
#       "https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale customers data.csv"
#     ),
#     sep = ",",
#     stringsAsFactors = FALSE,
#     na.strings=c("","NA")
#   )
# # Add a column so I know which study the data is refereing to
# study <- sprintf("study_%s",seq(1:440))
# dfWCD$study<-study
# }

# change 2 to 24 to numeric
dfWCD[1:8] <- sapply(dfWCD[1:8], as.numeric)

# Print first lines
str(dfWCD)
```

```
## 'data.frame':    440 obs. of  9 variables:
##  $ Channel         : num  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num  12669 7057 6353 13265 22615 ...
##  $ Milk            : num  9656 9810 8808 1196 5410 ...
##  $ Grocery         : num  7561 9568 7684 4221 7198 ...
##  $ Frozen          : num  214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num  2674 3293 3516 507 1777 ...
##  $ Delicassen      : num  1338 1776 7844 1788 5185 ...
##  $ study           : chr  "study_1" "study_2" "study_3" "study_4" ...
```

```
# Select the first 8 lines for plotting
dfWCD2<-dfWCD[1:8]

# Print first lines
str(dfWCD2)
```

```
## 'data.frame':    440 obs. of  8 variables:
##  $ Channel         : num  2 2 2 1 2 2 2 2 1 2 ...
##  $ Region          : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ Fresh           : num  12669 7057 6353 13265 22615 ...
##  $ Milk            : num  9656 9810 8808 1196 5410 ...
##  $ Grocery         : num  7561 9568 7684 4221 7198 ...
##  $ Frozen          : num  214 1762 2405 6404 3915 ...
##  $ Detergents_Paper: num  2674 3293 3516 507 1777 ...
##  $ Delicassen      : num  1338 1776 7844 1788 5185 ...
```

1.2 Clustering

Clustering is grouping like with like such that:

1. Similar objects are close to one another within the same cluster.
2. Dissimilar to the objects in other clusters.

1.2.1 Understanding the data

The data set refers to clients of a wholesale distributor in Portugal. It includes the annual spending in monetary units (m.u.) on diverse product categories. The data has the following attribute information:

1. FRESH: annual spending (m.u.) on fresh products (Continuous);
2. MILK: annual spending (m.u.) on milk products (Continuous);
3. GROCERY: annual spending (m.u.)on grocery products (Continuous);
4. FROZEN: annual spending (m.u.)on frozen products (Continuous)
5. DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)

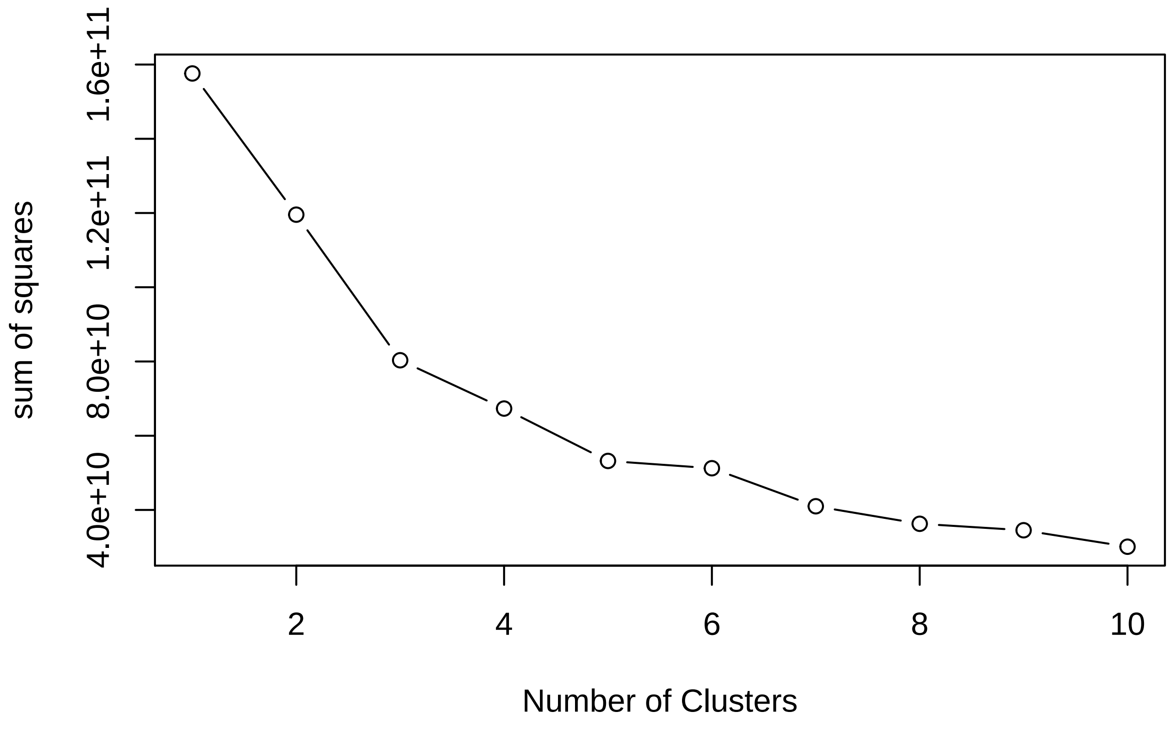
6. DELICATESSEN: annual spending (m.u.)on and delicatessen products (Continuous);
7. CHANNEL: customer channel - 1 = Horeca (Hotel/Restaurant/Cafe) or 2 = Retail channel (Nominal) 2 = Oporto or 3 = Other (Nominal)

1.2.2 Number of Clusters

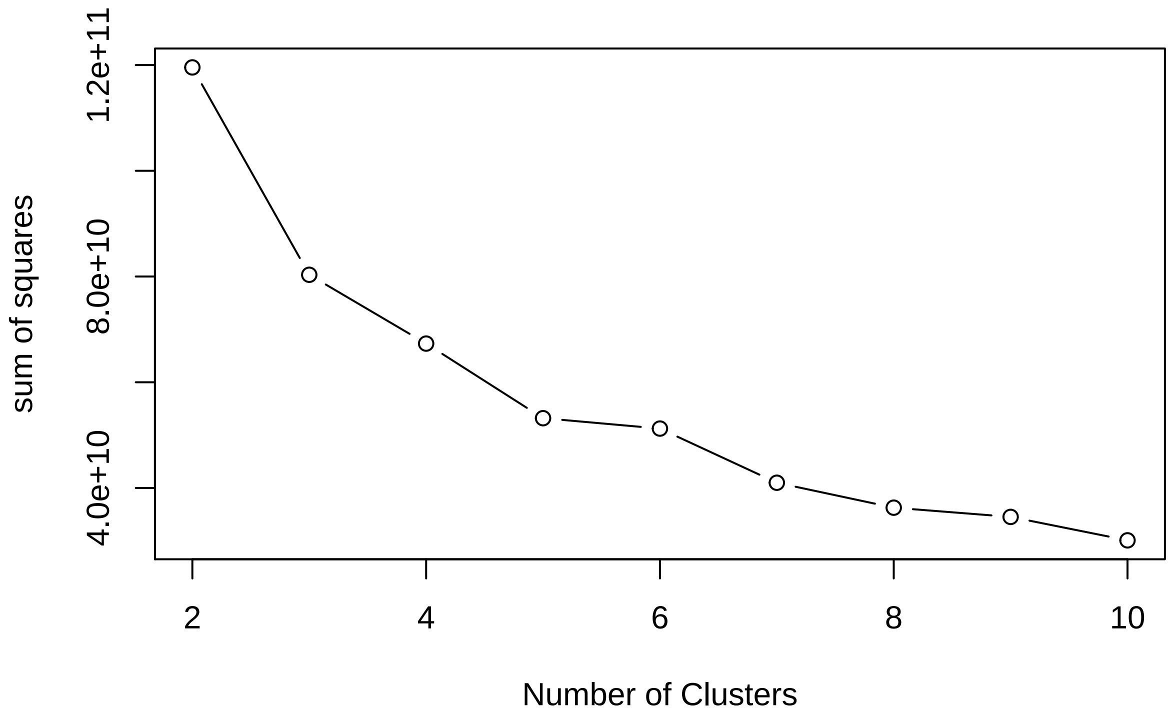
Before I can begin clustering analysis, I need to determine the number of clusters. For determining "the right number of clusters", for this analysis I will use the averaged Silhouette width and Gap statistic and Hartigan's rule.

1.2.2.1 Averaged Silhouette Width and Gap Statistic

```
# Determining number of clusters
sos <- (nrow(dfWCD2) - 1) * sum(apply(dfWCD2, 2, var))
for (i in 2:10)
  sos[i] <- sum(kmeans(dfWCD2, centers = i)$withinss)
plot(1:10,
     sos,
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```

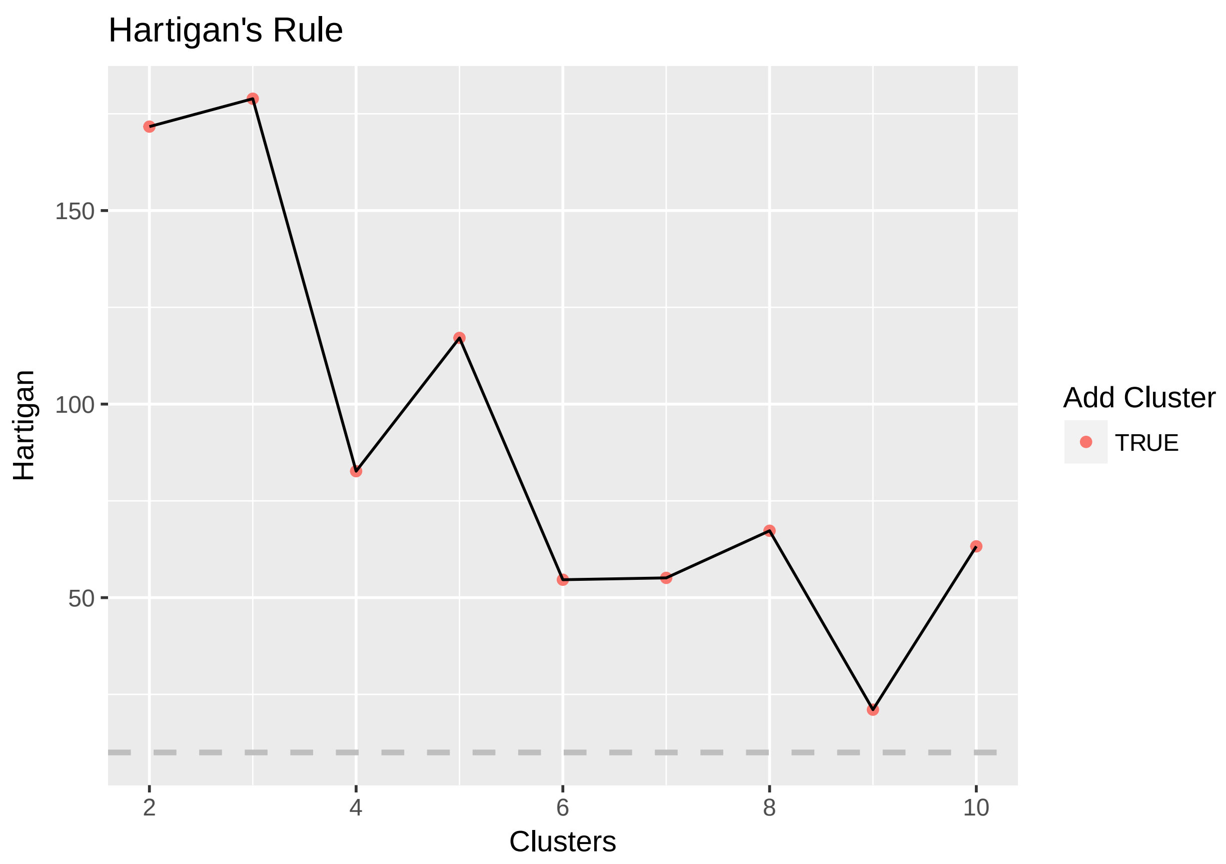


```
plot(2:10,
     sos[c(2:10)],
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```



1.2.2.2 Hartigan's rule

```
# Hartigans's rule FitKMean (similarity)
# require(useful)
best<-FitKMeans(dfWCD2,max.clusters=10, seed=111)
PlotHartigan(best)
```



1.2.2.3 Number of Clusters Results

Based off of the graphs above it looks like I should perform clustering with 3 to 5 clusters. The analysis looks better with 4 after some trial and error.

1.2.3 Partitioning-based clustering

Partitioning algorithms construct various partitions and then evaluate them by some criterion Hierarchy algorithms, two examples are k-means and k-mediods algorithms.

1.2.3.1 k-means

k-means clustering is a method of vector quantization, originally from signal processing. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

Below I apply the k-means algorithm:

```r
# Clustering with 4 clusters
k <- 4
dfWCD2.4.cluster <- kmeans(dfWCD2,k)
dfWCD2.4.cluster
```

```
## K-means clustering with 4 clusters of sizes 95, 11, 58, 276
##
```

```
## Cluster means:
##    Channel   Region     Fresh      Milk   Grocery   Frozen
## 1 1.863158 2.536842  4808.842 10525.011 16909.789 1462.589
## 2 1.909091 2.545455 19888.273 36142.364 45517.455 6328.909
## 3 1.137931 2.586207 36144.483  5471.466  6128.793 6298.655
## 4 1.152174 2.536232  9087.464  3027.428  3753.514 2817.986
##   Detergents_Paper Delicassen
## 1         7302.400   1650.884
## 2        21417.091   8414.000
## 3         1064.000   2316.724
## 4         1003.004   1040.525
##
## Clustering vector:
##   [1] 4 1 4 4 3 4 4 4 4 1 1 4 3 1 3 4 1 4 4 4 4 4 3 2 3 4 4 4 1 3 4 4 4 3 4
##  [36] 1 3 1 1 3 3 4 1 1 4 1 1 2 4 1 4 4 3 1 3 4 1 1 4 4 4 2 4 1 4 2 4 4 4 4
##  [71] 4 1 4 4 4 4 4 1 4 4 4 1 1 4 4 2 2 3 4 3 4 4 2 4 1 4 4 4 4 4 1 1 4 3 4
## [106] 4 1 1 4 1 4 1 4 4 4 4 4 4 4 4 4 4 4 4 3 3 4 4 4 3 4 4 4 4 4 4 4 4 4 4
## [141] 4 3 3 4 4 1 4 4 4 3 4 4 4 4 4 1 1 4 1 1 1 4 4 1 4 1 1 4 4 4 1 1 4 1 4
## [176] 1 3 4 4 4 4 3 1 2 4 4 4 1 1 1 4 4 4 1 4 4 3 1 4 4 1 1 3 4 4 1 4 4 4 1
## [211] 4 2 4 4 1 1 1 4 1 4 4 1 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 3 3 3 4 4 1
## [246] 1 4 4 4 4 4 2 4 3 4 3 4 4 3 3 4 4 4 4 1 1 1 4 1 4 4 4 4 3 4 4 3 4 4 4
## [281] 4 4 3 3 3 3 4 4 4 3 4 4 4 1 4 4 4 4 4 4 4 1 1 1 1 1 1 4 4 1 4 3 1 4 4
## [316] 1 4 4 4 1 4 4 4 4 3 3 4 4 4 4 4 1 4 2 4 3 4 4 4 4 1 1 4 1 4 4 1 3 4 1
## [351] 4 1 4 1 4 4 4 1 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 1 3 4 4 3 4 3 4 1
## [386] 4 4 4 4 4 4 4 4 3 4 4 1 4 4 4 4 3 3 3 4 4 3 1 4 4 4 4 1 4 4 4 1 1 1 4
## [421] 1 4 3 4 4 4 1 3 4 4 1 4 4 4 4 3 3 1 4 4
##
## Within cluster sum of squares by cluster:
## [1] 10434501446 14680550155 20922612384 18817881861
##  (between_SS / total_SS =  58.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

This calculation makes the k-means calculation more stable, it performs this analysis 1000 times and takes the ones with the least error:

```r
# Clustering with 4 clusters
k <- 4
trails<-1000
dfWCD2.4.cluster <- kmeans(dfWCD2,k, nstart = trails)
dfWCD2.4.cluster
```

```
## K-means clustering with 4 clusters of sizes 93, 281, 59, 7
##
## Cluster means:
##    Channel   Region     Fresh      Milk   Grocery   Frozen
## 1 1.881720 2.505376  5121.527 11293.570 17686.441 1545.806
## 2 1.160142 2.544484  9004.922  3103.815  3861.922 2804.562
## 3 1.135593 2.593220 36156.390  6123.644  6366.780 6811.119
## 4 2.000000 2.571429 20031.286 38084.000 56126.143 2564.571
##   Detergents_Paper Delicassen
## 1        7702.699   1886.527
## 2        1053.534   1051.053
## 3        1050.017   3090.051
## 4       27644.571   2548.143
##
## Clustering vector:
##   [1] 2 2 2 2 3 2 2 2 2 1 1 2 3 1 3 2 1 2 2 2 2 2 3 1 3 2 2 2 1 3 2 2 2 3 2
##  [36] 1 3 1 1 3 3 2 1 1 2 1 1 4 2 1 2 2 3 1 3 2 1 1 2 2 2 4 2 1 2 4 2 2 2 2
##  [71] 2 1 2 2 2 2 2 1 2 2 2 1 1 2 2 4 4 3 2 3 2 2 1 2 1 2 2 2 2 2 1 1 2 3 2
## [106] 2 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 2 2 2 3 2 2 2 2 2 2 2 2 2
## [141] 2 3 3 2 2 1 2 2 2 3 2 2 2 2 2 1 1 2 2 1 1 2 2 1 2 1 2 2 2 2 1 1 2 1 2
## [176] 1 3 2 2 2 2 3 1 3 2 2 2 2 1 1 2 2 2 1 2 2 3 1 2 2 1 1 3 2 2 1 2 2 2 1
## [211] 2 4 2 2 1 1 1 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 3 3 3 2 2 1
## [246] 1 2 2 2 2 2 1 2 3 2 3 2 2 3 3 2 2 2 2 1 1 1 2 1 2 2 2 2 3 2 2 3 2 2 2
## [281] 2 2 3 3 3 3 2 2 2 3 2 2 2 1 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 1 2 3 1 2 2
## [316] 1 2 2 2 1 2 2 2 2 3 3 2 2 2 2 2 1 2 4 2 3 2 2 2 2 1 1 2 1 2 2 1 3 2 1
## [351] 2 1 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 1 3 2 2 3 2 3 2 1
## [386] 2 2 2 2 2 2 2 2 3 2 2 1 2 2 2 2 3 3 3 2 2 3 1 2 2 2 2 1 2 2 2 1 2 1 2
## [421] 1 2 3 2 2 2 1 3 2 2 1 2 2 2 2 3 3 1 2 2
##
## Within cluster sum of squares by cluster:
## [1] 12568477117 19299314618 25518237851  7469349068
##  (between_SS / total_SS =  58.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

There is 1 cluster that I need to look at with a value 7. Below is a table looking at all of the information

```r
# Clustering with 5 clusters
cm<-table(dfWCD$study,dfWCD2.4.cluster$cluster)
head(cm)
```

```
##
##              1 2 3 4
##   study_1    0 1 0 0
##   study_10   1 0 0 0
##   study_100  0 1 0 0
##   study_101  1 0 0 0
##   study_102  1 0 0 0
##   study_103  0 1 0 0
```

```r
# Get cluster of 7
# clusterOf7<-cm[,4][cm[,4]>0]
# clusterOf7


# Look at the group of 7
dfSeven <- dfWCD[c(212, 334, 48, 62, 66, 86, 87), ]
dfSeven
```

```
##     Channel Region Fresh  Milk Grocery Frozen Detergents_Paper Delicassen
## 212       2      1 12119 28326   39694   4736            19410       2870
## 334       2      2  8565  4980   67298    131            38102       1215
## 48        2      3 44466 54259   55571   7782            24171       6465
## 62        2      3 35942 38369   59598   3254            26701       2017
## 66        2      3    85 20959   45828     36            24231       1423
## 86        2      3 16117 46197   92780   1026            40827       2944
## 87        2      3 22925 73498   32114    987            20070        903
##         study
## 212 study_212
## 334 study_334
## 48   study_48
## 62   study_62
## 66   study_66
## 86   study_86
## 87   study_87
```

I cannot make too much sense of this data and will review the centroid plot.

1.2.3.1.1 Evaluating model performance

```
# Evaluating model performance
# look at the size of the clusters
dfWCD2.4.cluster$size
```

```
## [1]  93 281  59   7
```

```
# look at the cluster centers
dfWCD2.4.cluster$centers
```

```
##    Channel   Region     Fresh      Milk   Grocery   Frozen
## 1 1.881720 2.505376  5121.527 11293.570 17686.441 1545.806
## 2 1.160142 2.544484  9004.922  3103.815  3861.922 2804.562
## 3 1.135593 2.593220 36156.390  6123.644  6366.780 6811.119
## 4 2.000000 2.571429 20031.286 38084.000 56126.143 2564.571
##   Detergents_Paper Delicassen
## 1         7702.699   1886.527
## 2         1053.534   1051.053
## 3         1050.017   3090.051
## 4        27644.571   2548.143
```

```
names(dfWCD2)
```

```
## [1] "Channel"          "Region"           "Fresh"
## [4] "Milk"             "Grocery"          "Frozen"
## [7] "Detergents_Paper" "Delicassen"
```

```
# mean of 'Fresh' by cluster
Fresh<-aggregate(data = dfWCD2, Fresh ~ dfWCD2.4.cluster$cluster, mean)
Fresh
```

```
##   dfWCD2.4.cluster$cluster     Fresh
## 1                        1  5121.527
## 2                        2  9004.922
## 3                        3 36156.390
## 4                        4 20031.286
```

```
# mean of 'Milk' by cluster
Milk<-aggregate(data = dfWCD2, Milk ~ dfWCD2.4.cluster$cluster, mean)
Milk
```

```
##   dfWCD2.4.cluster$cluster      Milk
## 1                        1 11293.570
## 2                        2  3103.815
## 3                        3  6123.644
## 4                        4 38084.000
```

```
# mean of 'Grocery' by cluster
Grocery<-aggregate(data = dfWCD2, Grocery ~ dfWCD2.4.cluster$cluster, mean)
Grocery
```

```
##   dfWCD2.4.cluster$cluster   Grocery
## 1                        1 17686.441
## 2                        2  3861.922
## 3                        3  6366.780
## 4                        4 56126.143
```

```
# mean 'Detergents_Paper'  by cluster
DP<-aggregate(data = dfWCD2, Detergents_Paper ~ dfWCD2.4.cluster$cluster, mean)
DP
```

```
##   dfWCD2.4.cluster$cluster Detergents_Paper
## 1                        1         7702.699
## 2                        2         1053.534
## 3                        3         1050.017
## 4                        4        27644.571
```

```
# mean 'Delicassen'  by cluster
Delicassen<-aggregate(data = dfWCD2, Delicassen ~ dfWCD2.4.cluster$cluster, mean)
Delicassen
```

```
##   dfWCD2.4.cluster$cluster Delicassen
## 1                        1   1886.527
## 2                        2   1051.053
## 3                        3   3090.051
## 4                        4   2548.143
```

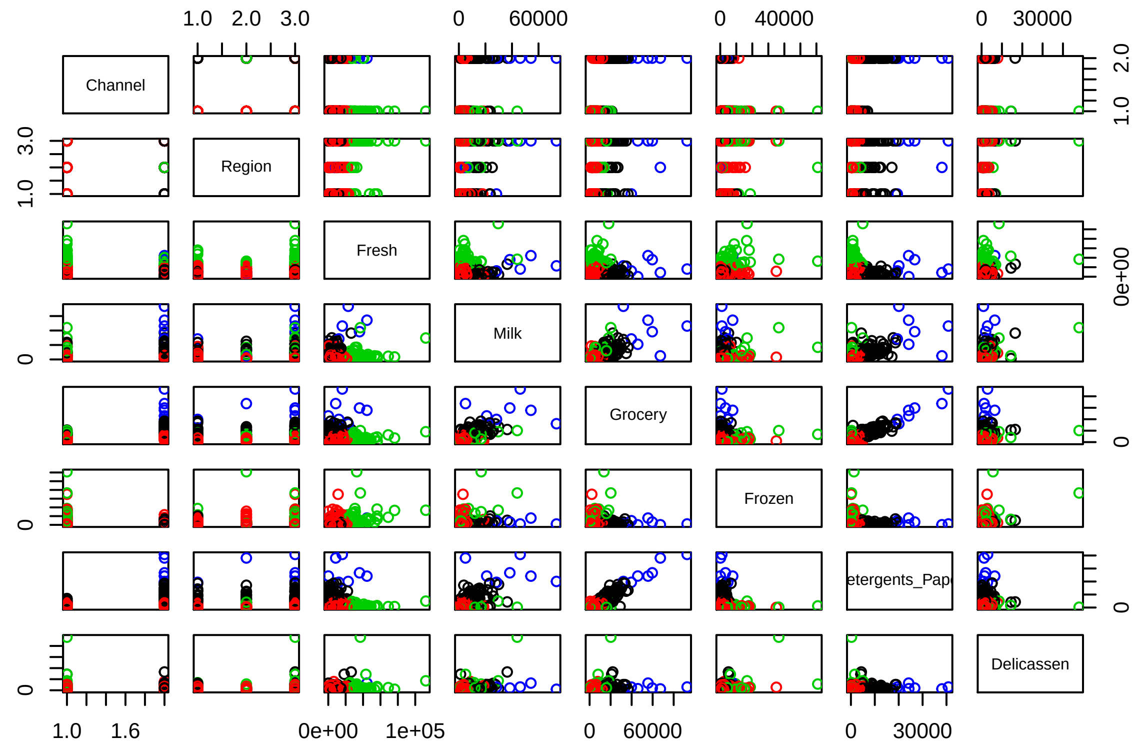
1.2.3.1.2 Multidimensional scaling (MDS)

Multidimensional scaling (MDS) is a means of visualizing the level of similarity of individual cases of a high-dimenional dataset.

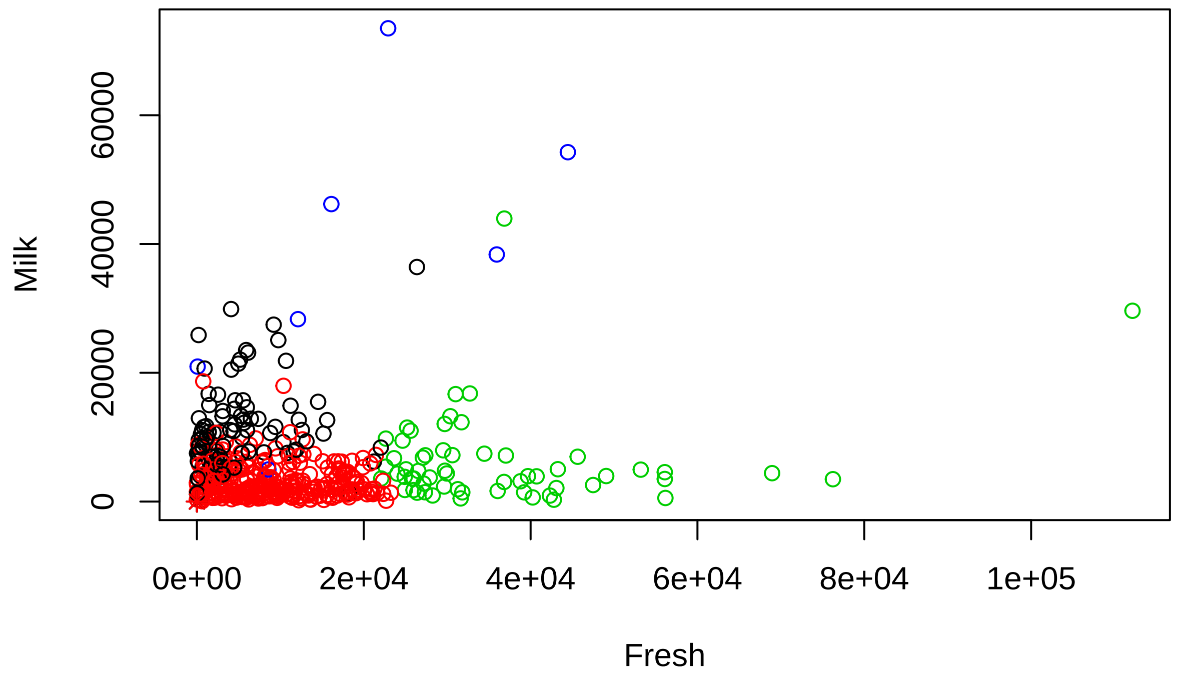
MDS attempts to find an embedding from the I objects into $\mathbb{R}^N$ such that distances are preserved.

Below are some MDS plots:

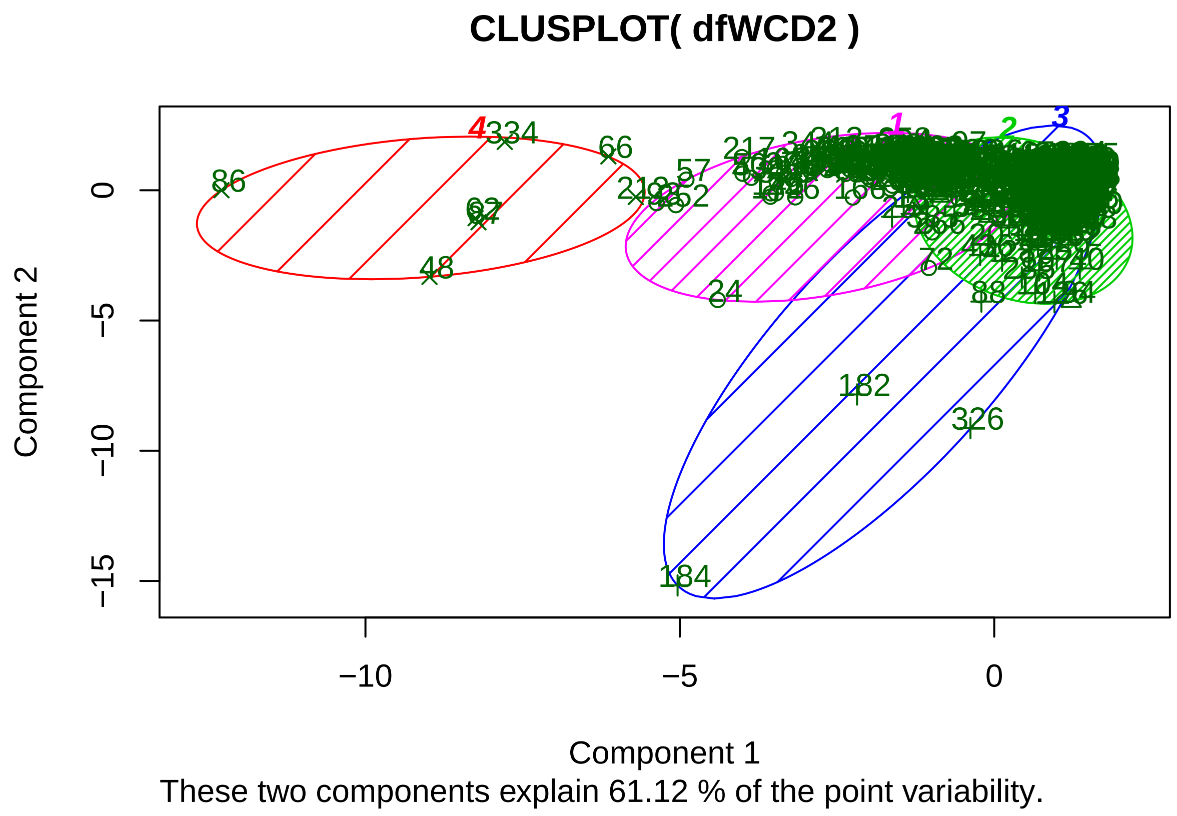
```
plot(dfWCD2,col=dfWCD2.4.cluster$cluster)     # Plot Clusters
```



```
# Plot drinks and mcv
 plot(dfWCD2[c("Fresh","Milk")],col=dfWCD2.4.cluster$cluster)
 points(dfWCD2.4.cluster$centers, col = 1:2, pch = 8)
```



```r
# Centroid Plot against 1st two discriminant functions
clusplot(dfWCD2, dfWCD2.4.cluster$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



```r
# library(fpc)
# plotcluster(dfWCD2,dfWCD2.4.cluster$cluster)
```

The plots look good, the data is clustered as expected. The centroid plot looks ok. I can see the small cluster is grouped ok, however, there is some overlaping with a couple of clusters in the corner, it seems like there is a lot going on in the upper right corner, but I do not think this is an issue.

1.2.3.2 K-medoids clustering in R

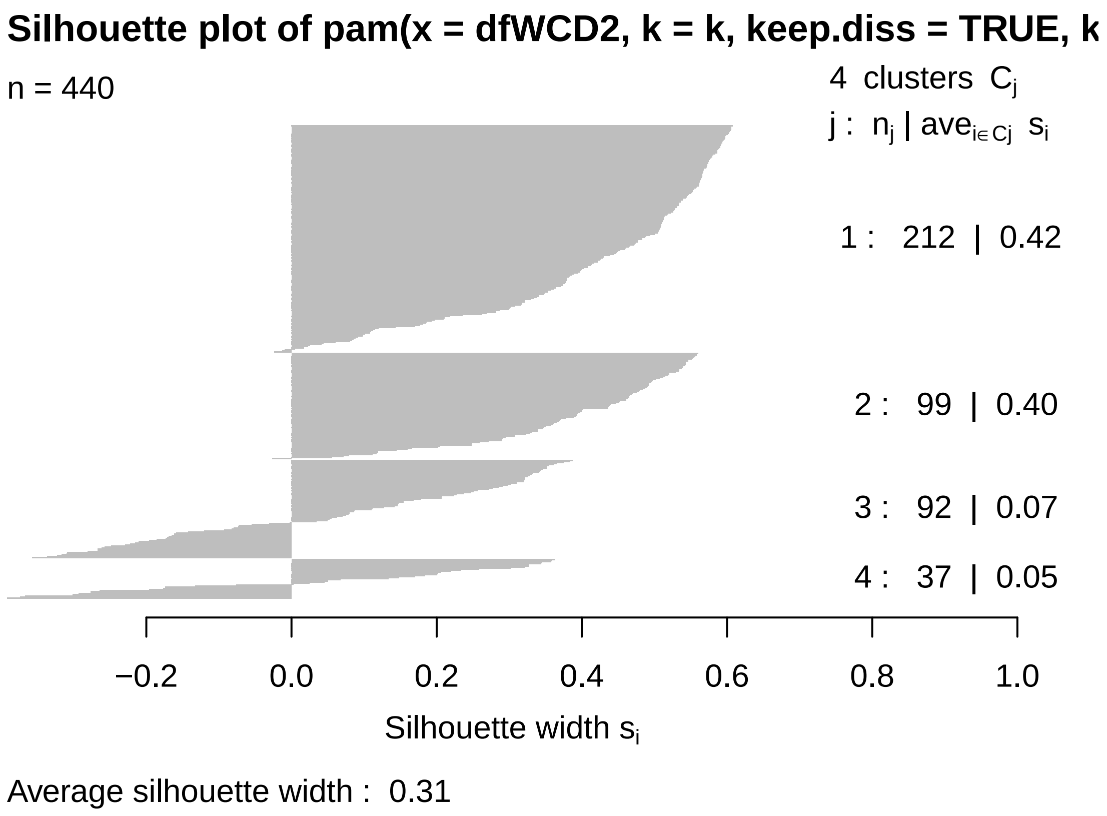
The K-medoids or Partitioning Around Medoids (PAM) algorithm is related to the k-means algorithm and but uses medoid shifts rather than reassigning points based on Euclidean distance. Each cluster is represented by one of the objects (i.e. points) in the cluster A medoid is a point in a cluster whose dissimilarity to all the points in the cluster is minimal. Medoids are similar in concept to means or centroids, but medoids are always members of the data set. That is, in 2D Cartesian space a centroid can be any valid x.y coordinate. Whereas a medoid must be one of the data points.

Below I use R to apply the k-medoids algorithm:

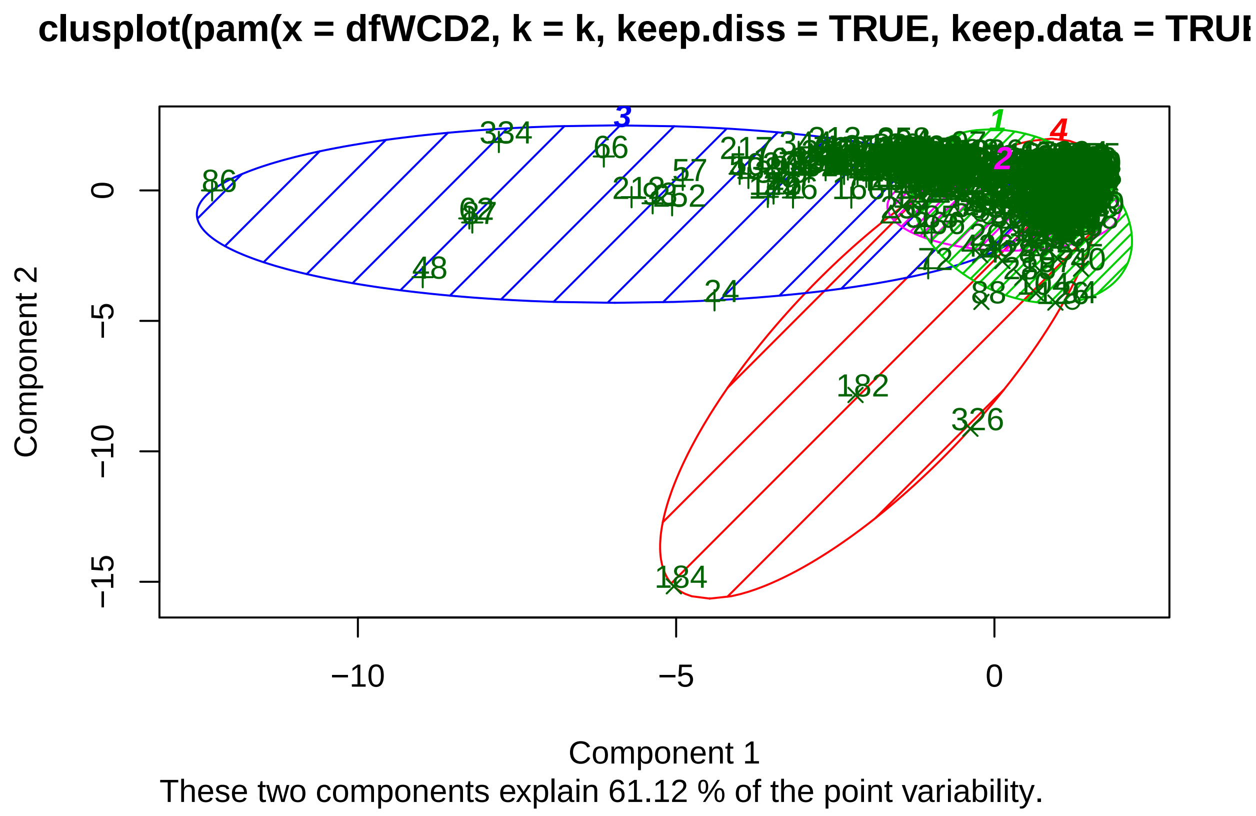
```r
# PAM
k<-4
dfWCD2.pam.4.clust<- pam(dfWCD2,k, keep.diss = TRUE, keep.data = TRUE)
dfWCD2.pam.4.clust
```

```
## Medoids:
##       ID Channel Region Fresh  Milk Grocery Frozen Detergents_Paper
## [1,] 292       1      3  6022  3354    3261   2507              212
## [2,] 297       1      2 19087  1304    3643   3045              710
## [3,]  10       2      3  6006 11093   18881   1159             7425
## [4,] 371       2      3 39679  3944    4955   1364             523
##      Delicassen
## [1,]        686
## [2,]        898
## [3,]       2098
## [4,]       2235
## Clustering vector:
##   [1] 1 1 1 2 2 1 1 1 1 3 3 2 4 2 2 1 3 1 2 1 2 1 4 3 2 2 1 2 3 4 2 1 2 4 1
##  [36] 1 4 3 3 4 2 2 3 3 1 3 3 3 1 3 1 1 4 3 2 1 3 3 2 1 1 3 1 3 1 3 1 2 1 1
##  [71] 2 3 1 2 1 2 1 3 1 1 1 3 3 2 1 3 3 4 1 2 1 2 3 1 3 1 1 1 1 1 3 3 1 4 2
## [106] 2 3 3 1 3 1 3 2 2 2 1 1 1 2 1 2 1 1 1 4 4 2 2 1 4 1 1 2 1 1 1 1 1 2 1
## [141] 2 4 4 1 2 3 1 1 1 4 2 1 2 1 1 3 3 2 1 3 3 2 2 3 1 3 1 1 1 1 3 3 1 3 1
## [176] 3 4 1 1 1 1 4 3 4 1 1 1 1 3 3 2 2 1 3 1 2 4 1 1 1 3 3 2 1 1 3 1 1 1 3
## [211] 2 3 1 1 3 3 3 2 3 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 2 1 4 2 2 2 1 1
## [246] 3 1 2 2 1 1 3 1 4 1 4 1 1 4 4 1 1 2 1 3 3 3 2 3 2 1 1 1 4 1 1 2 1 1 2
## [281] 1 1 4 2 4 4 1 2 2 4 1 1 1 3 2 1 2 1 1 1 2 3 1 3 3 3 3 2 1 3 1 4 3 1 1
## [316] 3 1 1 1 3 1 1 2 2 2 4 1 1 2 1 1 3 2 3 2 2 2 1 1 1 1 3 1 3 1 1 3 2 1 3
## [351] 1 3 1 3 2 1 2 3 1 1 2 1 1 1 1 1 1 1 2 1 4 2 1 2 1 1 3 4 1 1 2 2 4 1 3
## [386] 1 1 2 1 1 1 1 1 2 1 1 3 1 1 1 1 2 2 2 2 1 2 3 1 1 1 1 1 1 1 1 3 1 3 1
## [421] 3 2 2 2 2 1 3 4 1 1 3 1 2 1 2 4 4 3 1 1
## Objective function:
##    build     swap
## 9285.399 8863.968
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

```r
plot(dfWCD2.pam.4.clust, which.plots = 2)
```



```
# long lines good - means greater within cluster similarity

# Centroid Plot against 1st two discriminant functions
clusplot(dfWCD2.pam.4.clust, color=TRUE, shade=TRUE, labels=2, lines=0)
```



The silhouette plot dosn't loof too good, long lines are good, short ones are not. It appears that there are more short lines.

The centroid plot looks good, but there seems to be a little overlap with the clusters

1.3 Confusion Plots

confusion plot is a plot of the confusion matrix. A confusion matrix, also known as a contingency table or an error matrix. A confusion matrix is a 2x2 table with counts of the following in each of its quadrents. true positive (TP) eqv. with hit true negative (TN) eqv. with correct rejection false positive (FP) eqv. with false alarm, Type I error false negative (FN) eqv. with miss, Type II error

To make a confusion plot I am going to make data with 2 clusters and compare how many times milk was purchased above the mean (above 5796.27) and how many times grocery was purchased above the mean (above 7951.28) for both k-means and pam.

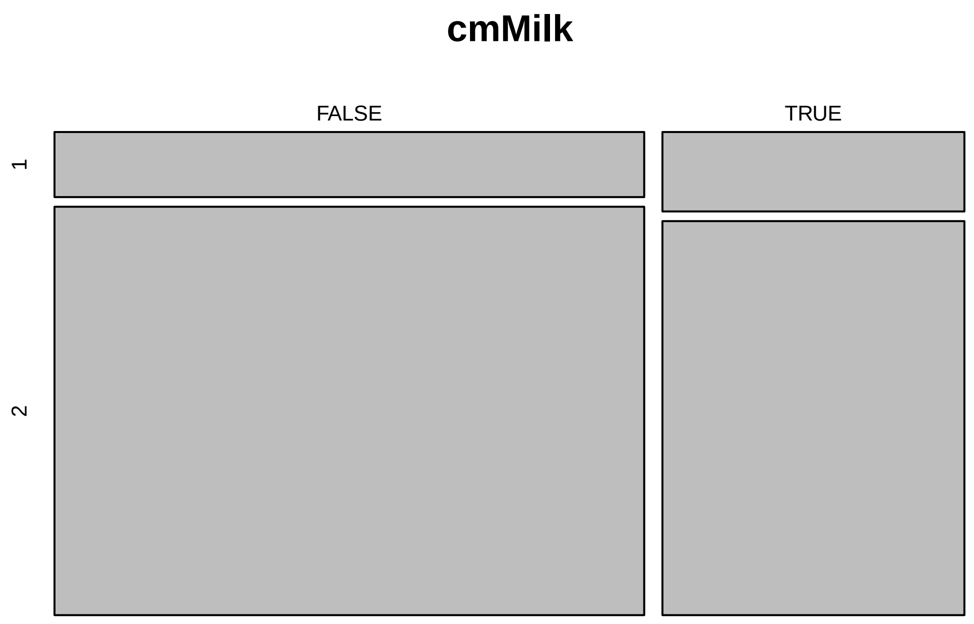
Below is the R code:

```
# Clustering with 2 clusters
k <- 2
trails<-1000

# k-means
dfWCD2.2.cluster <- kmeans(dfWCD2,k, nstart = trails)

# Confusion matrix of Milk
cmMilk<-table(dfWCD2$Milk>5796.27,dfWCD2.2.cluster$cluster)
```

```
cmMilk
```

```
##
##           1   2
##   FALSE  40 251
##   TRUE   25 124
```

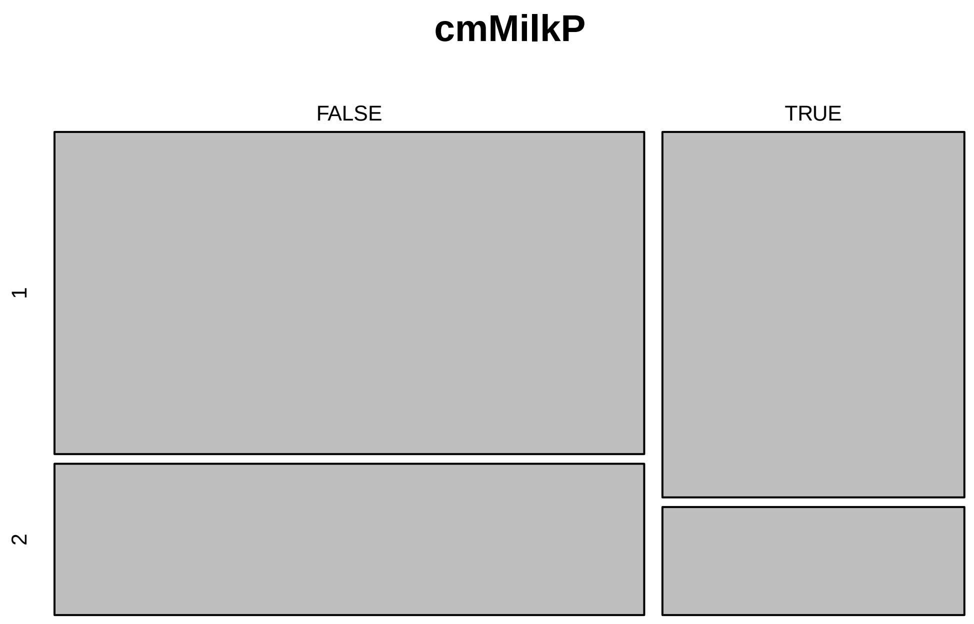
```
plot(cmMilk)
```



```
# pam
dfWCD2.pam.2.clust<- pam(dfWCD2,k, keep.diss = TRUE, keep.data = TRUE)

# Confusion matrix of Milk
cmMilkP<-table(dfWCD2$Milk>5796.27,dfWCD2.pam.2.clust$cluster)
cmMilkP
```

```
##
##           1   2
##   FALSE 198  93
##   TRUE  115  34
```

```
plot(cmMilkP)
```

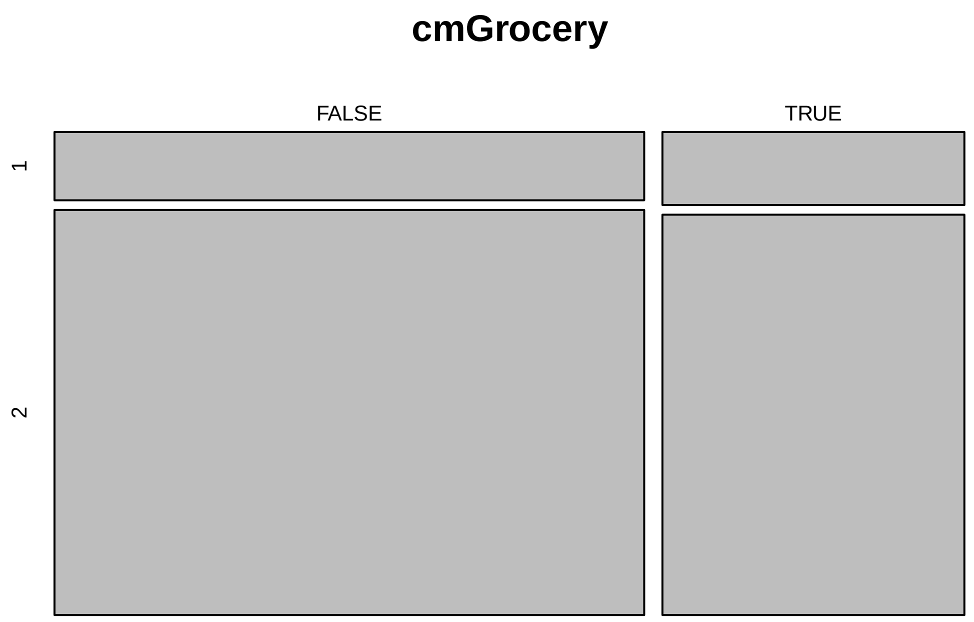


For the milk data, both the k-means and pam data show me that most of the data is about 50/50 for both groups, as I would expect.

```
# Clustering with 2 clusters
k <- 2
trails<-1000

# k-means
dfWCD2.2.cluster <- kmeans(dfWCD2,k, nstart = trails)

# Confusion matrix of grocery
cmGrocery<-table(dfWCD2$Grocery>7951.28,dfWCD2.2.cluster$cluster)
cmGrocery
```

```
##
##           1   2
##   FALSE  42 249
##   TRUE   23 126
```

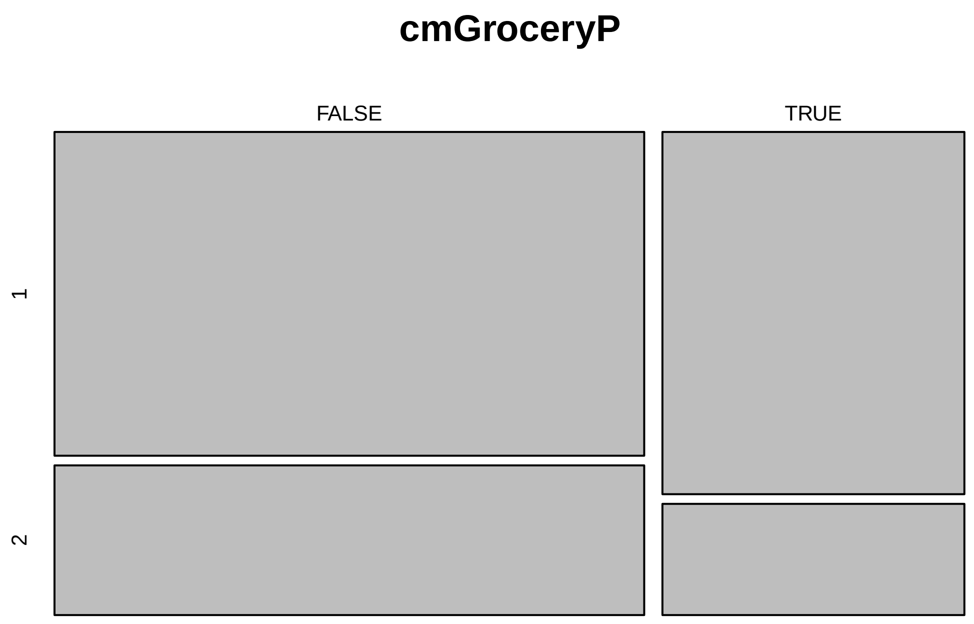
```
plot(cmGrocery)
```



```r
# pam
dfWCD2.pam.2.clust<- pam(dfWCD2,k, keep.diss = TRUE, keep.data = TRUE)

# Confusion matrix of Grocery
cmGroceryP<-table(dfWCD2$Grocery>7951.28,dfWCD2.pam.2.clust$cluster)
cmGroceryP
```

```
##
##           1   2
##   FALSE 199  92
##   TRUE  114  35
```

```r
plot(cmGroceryP)
```



The grocery data is a little different, the larger group is almost 60/40 (false/true), but the smaller group has 2/3 false to true.

1.4 Gap statistic

clusGap() calculates a goodness of clustering measure, the â gapâ statistic. For each number of clusters k, it compares $\log(W(k))$ with $E^*[\log(W(k))]$ where the latter is defined via bootstrapping, i.e. simulating from a reference distribution.

maxSE(f, SE.f) determines the location of the maximum of f, taking a â 1-SE ruleâ into account for the *SE* methods. The default method "firstSEmax" looks for the smallest k such that its value f(k) is not more than 1 standard error away from the first local maximum. This is similar but not the same as "Tibs2001SEmax", Tibshirani et al's recommendation of determining the number of clusters from the gap statistics and their standard deviations.

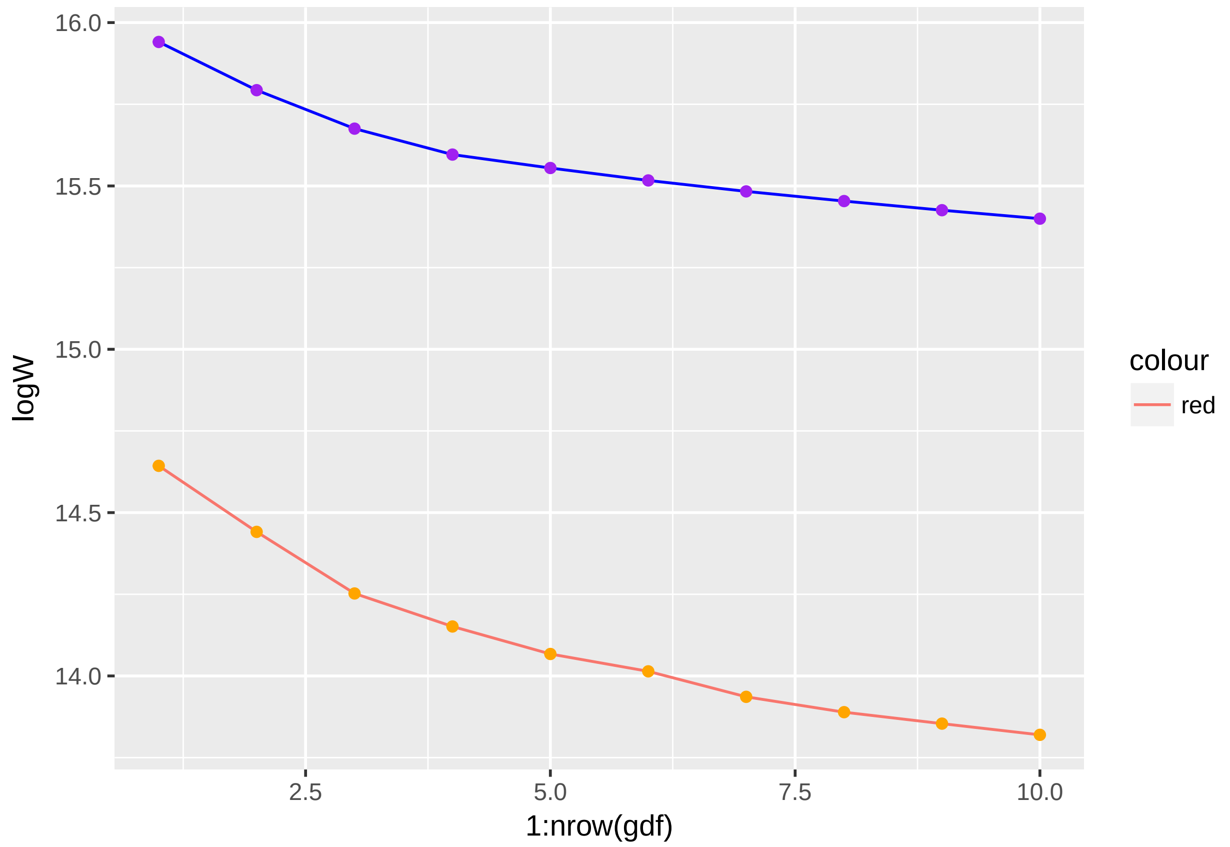
```
gap <- clusGap(dfWCD2, FUNcluster = pam, K.max = 10) # Bootstrapping
gap$Tab
```

```
##         logW   E.logW      gap     SE.sim
##  [1,] 14.64325 15.94079 1.297536 0.01114409
##  [2,] 14.44108 15.79317 1.352091 0.01498568
##  [3,] 14.25248 15.67539 1.422905 0.01117061
##  [4,] 14.15140 15.59598 1.444578 0.01520112
##  [5,] 14.06726 15.55488 1.487619 0.01185845
##  [6,] 14.01411 15.51682 1.502710 0.01137696
##  [7,] 13.93610 15.48332 1.547213 0.01091849
##  [8,] 13.88909 15.45366 1.564568 0.01053377
##  [9,] 13.85404 15.42574 1.571702 0.01005508
## [10,] 13.81975 15.39979 1.580041 0.01077285
```

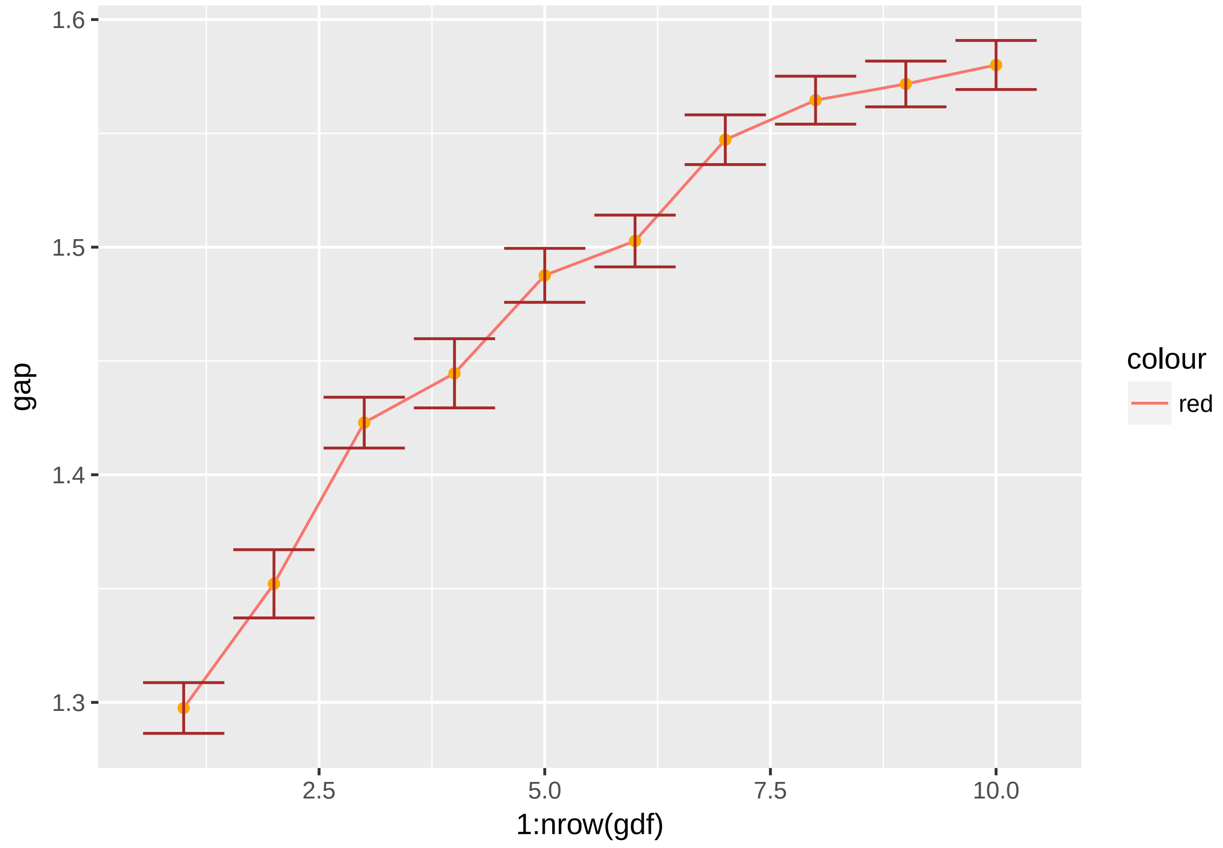
```r
gdf <- as.data.frame(gap$Tab)
head(gdf)
```

```
##       logW   E.logW      gap     SE.sim
## 1 14.64325 15.94079 1.297536 0.01114409
## 2 14.44108 15.79317 1.352091 0.01498568
## 3 14.25248 15.67539 1.422905 0.01117061
## 4 14.15140 15.59598 1.444578 0.01520112
## 5 14.06726 15.55488 1.487619 0.01185845
## 6 14.01411 15.51682 1.502710 0.01137696
```

```r
qplot(
  x = 1:nrow(gdf),
  y = logW,
  data = gdf,
  geom = "line",
  color = "red"
) +
  geom_point(aes(y = logW), color = "orange") +
  geom_line(aes(y = E.logW), color = "blue") +
  geom_point(aes(y = E.logW), color = "purple")
```



```r
# Gap statistic
qplot(
  x = 1:nrow(gdf),
  y = gap,
  data = gdf,
  geom = "line",
  color = "red"
) +
  geom_point(aes(y = gap), color = "orange") +
  geom_errorbar(aes(ymin = gap - SE.sim, ymax = gap + SE.sim), color = "brown")
```



It looks like around 3 or 4, the gap increases at a higher slope, so I think a k of 4 is good.

1.5 Hierarchical clustering in R

In hierarchical clustering the idea is to group data objects (i.e. points) into a tree of clusters. That is, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters.

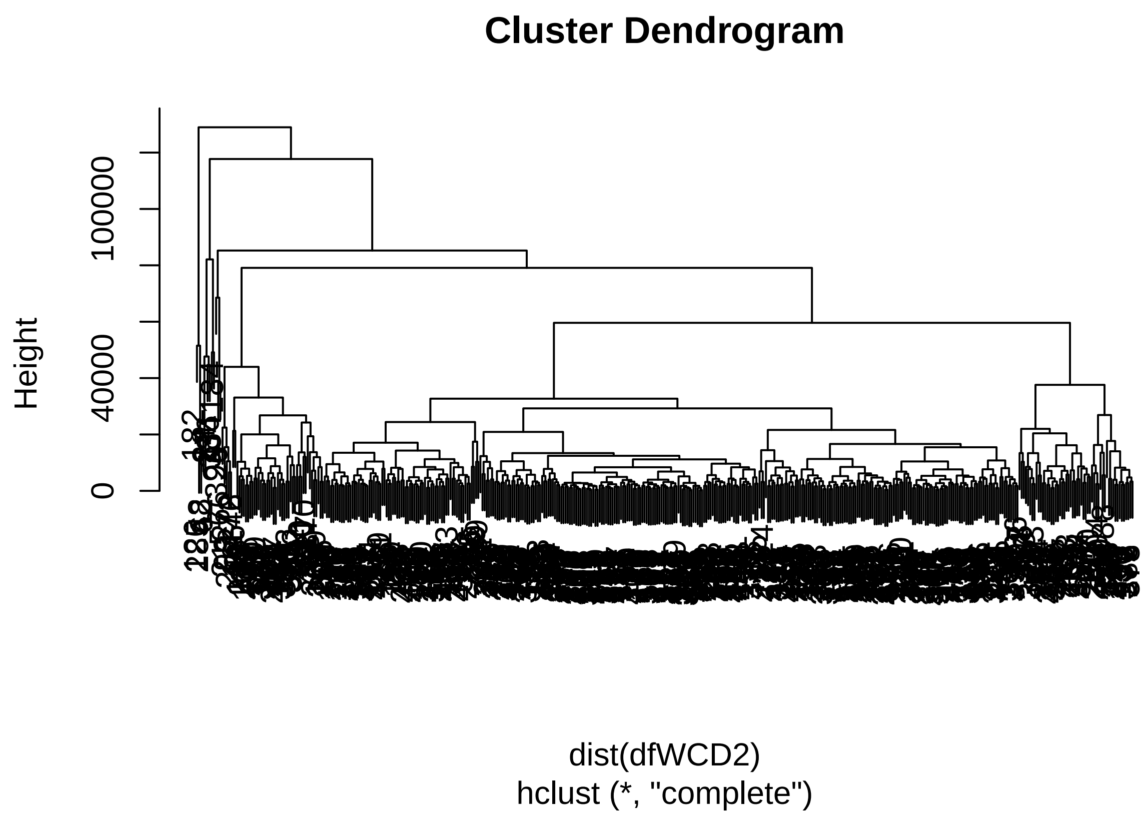
These trees (hierarchies) generally fall into two types:

1. Agglomerative hierarchical clustering: Initially each data object (i.e. point) in its own cluster. Iteratively the clusters are merged together from the “bottom-up.” The two most similar/closest objects are aggreated in to the same cluster/data object. Then the next two, until there is just one cluster/data object. This agglomerative approach result in “straggly” (long and thin) clusters due to a chaining effect. It is also sensitive to noise.
2. Divisive hierarchical clustering: in divisive hierarchical clustering all data objects (i.e. points) are initially in one cluster. These clusters are successively divided recursivley in a “top-down” manner. The cluster is broken in to two clusters that are most dissimilar. Then each of those clusters is broken in to two cluster that are most dissimilar. This continues until each clsuter is a single data object

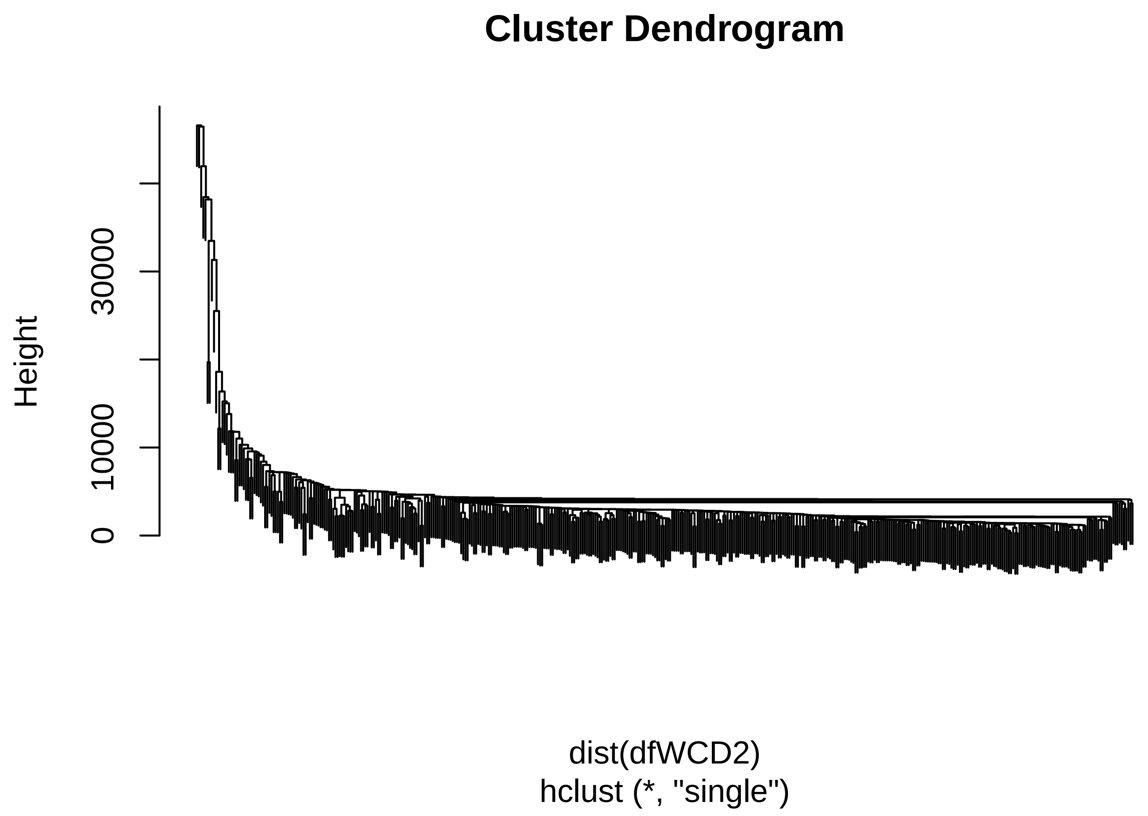
(i.e. point).

Below is my analysis using Hierarchical clustering:

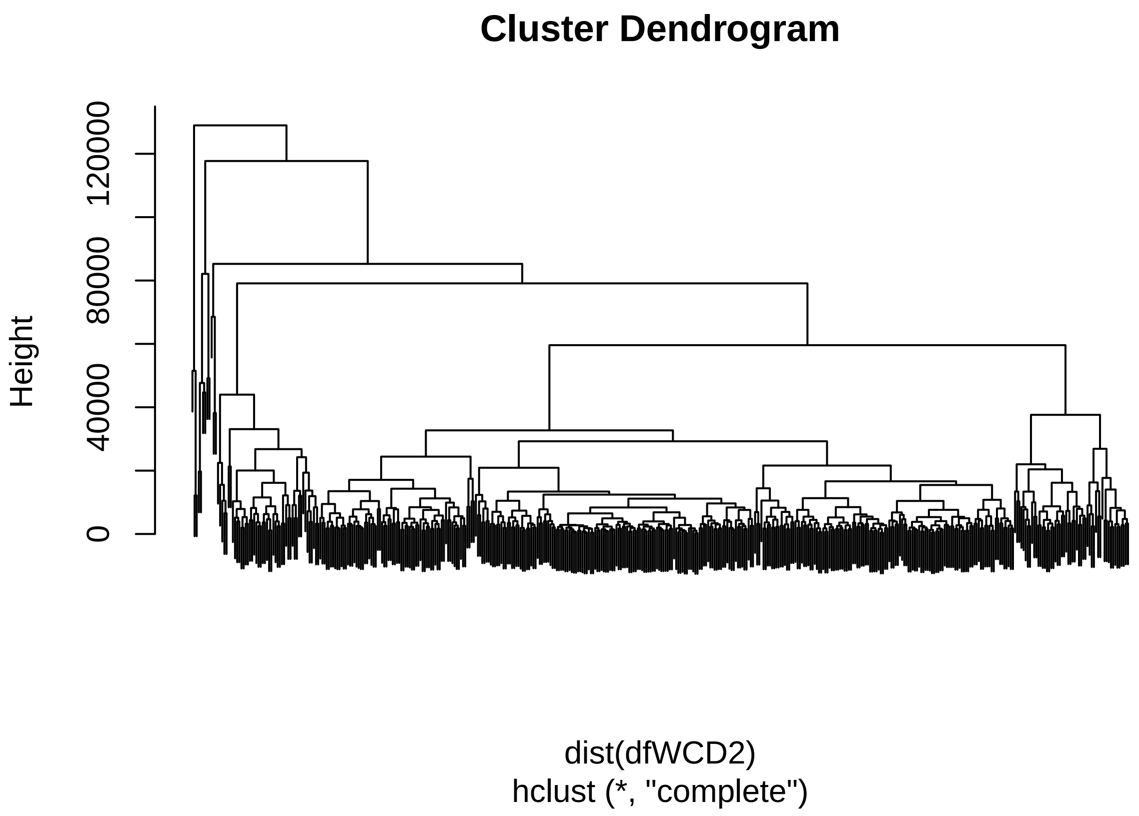
```r
dfWCD2.h.clust<- hclust(d=dist(dfWCD2))
plot(dfWCD2.h.clust)
```



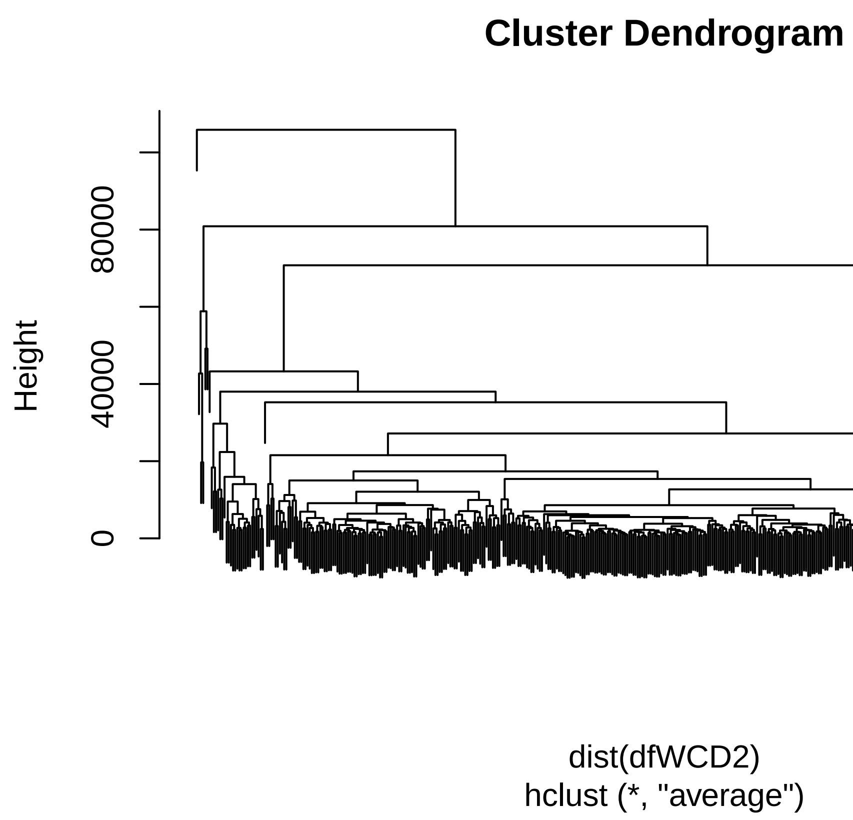
```r
dfWCD2.h.clust.si<- hclust(dist(dfWCD2), method = "single")
dfWCD2.h.clust.co<- hclust(dist(dfWCD2), method = "complete")
dfWCD2.h.clust.av<- hclust(dist(dfWCD2), method = "average")
dfWCD2.h.clust.ce<- hclust(dist(dfWCD2), method = "centroid")
dfWCD2.h.clust.me<- hclust(dist(dfWCD2), method = "ward.D")
plot(dfWCD2.h.clust.si, labels = FALSE)
```



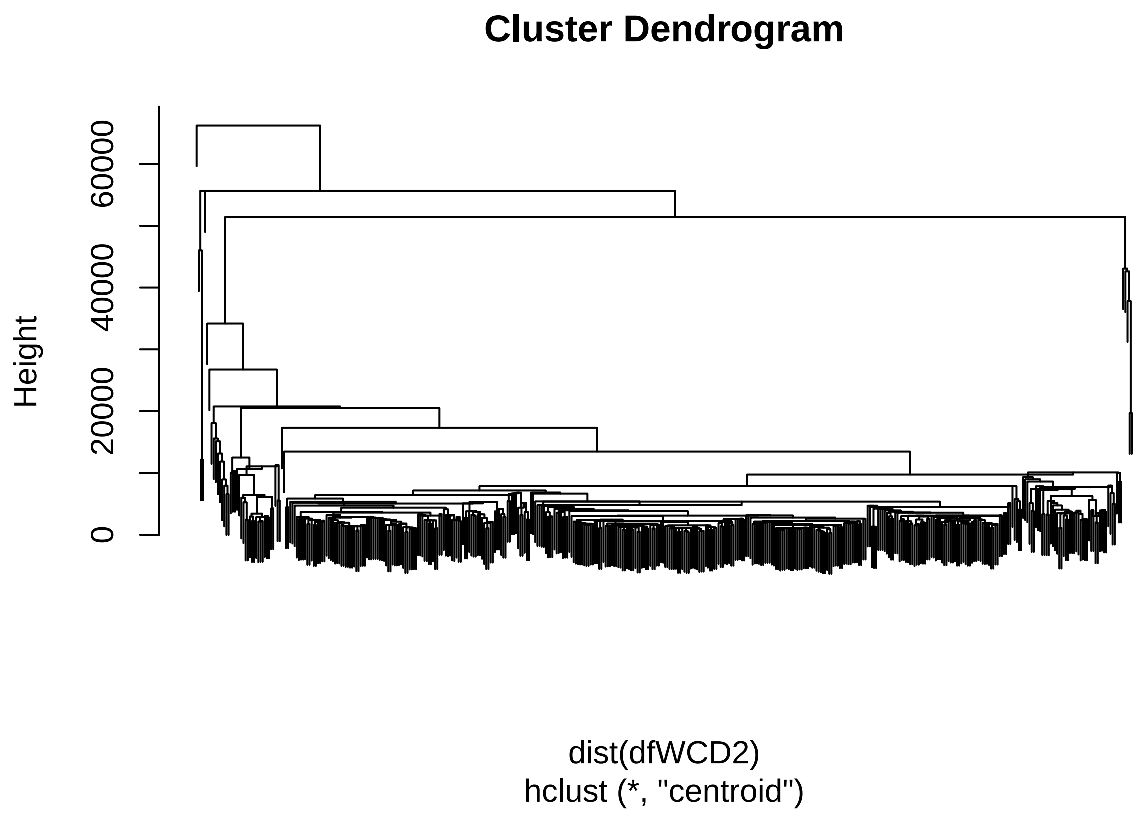
```r
plot(dfWCD2.h.clust.co, labels = FALSE)
```



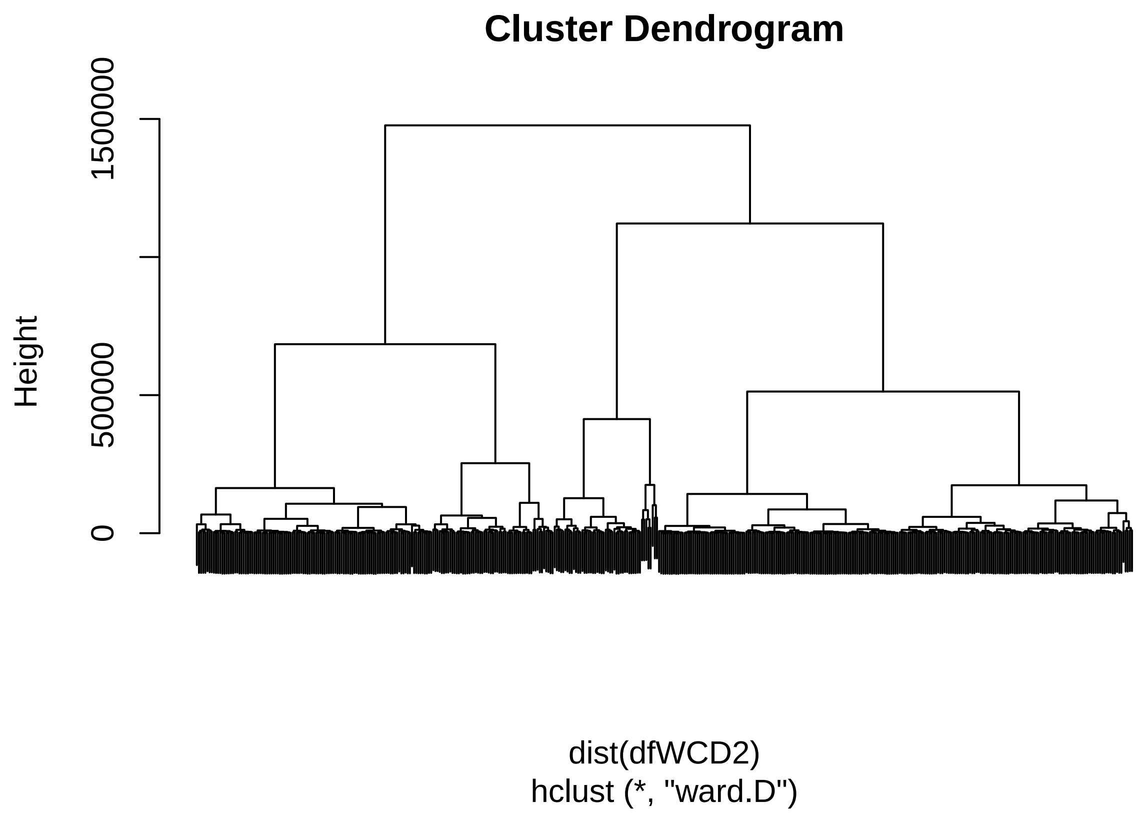
```r
plot(dfWCD2.h.clust.av, labels = FALSE)
```



```r
plot(dfWCD2.h.clust.ce, labels = FALSE)
```

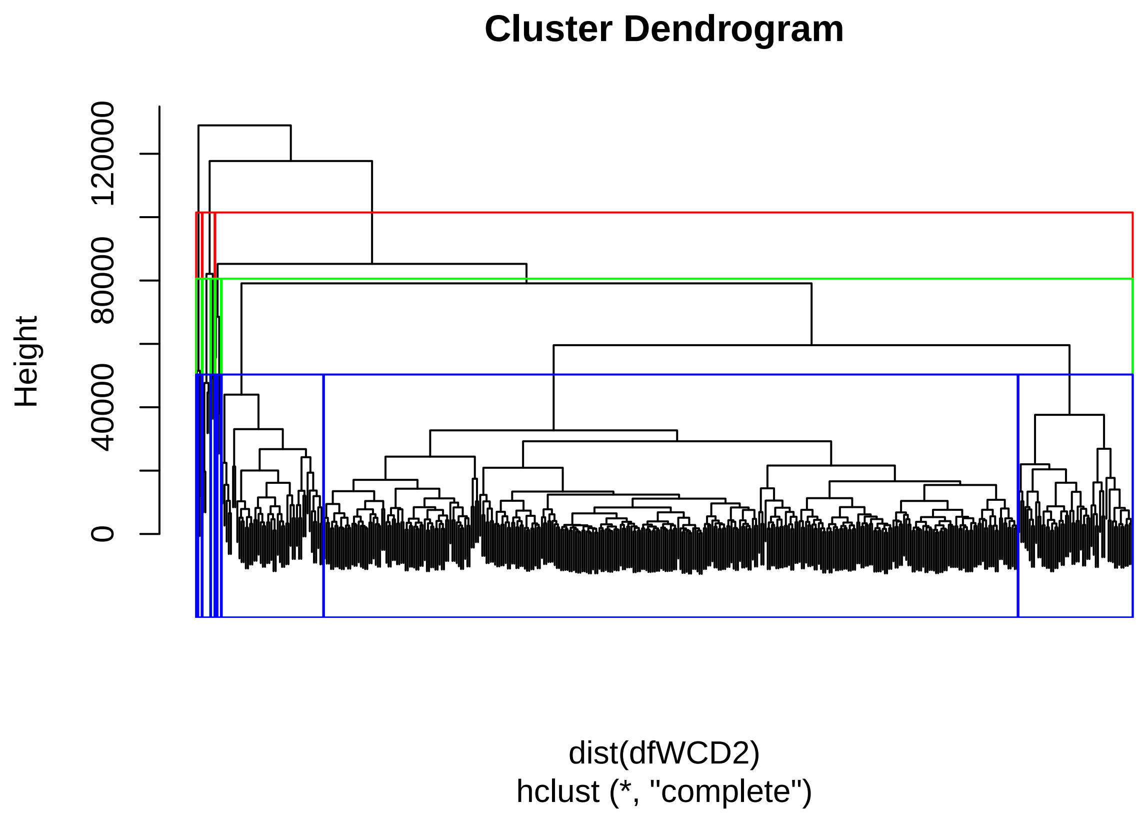


```r
plot(dfWCD2.h.clust.me, labels = FALSE)
```



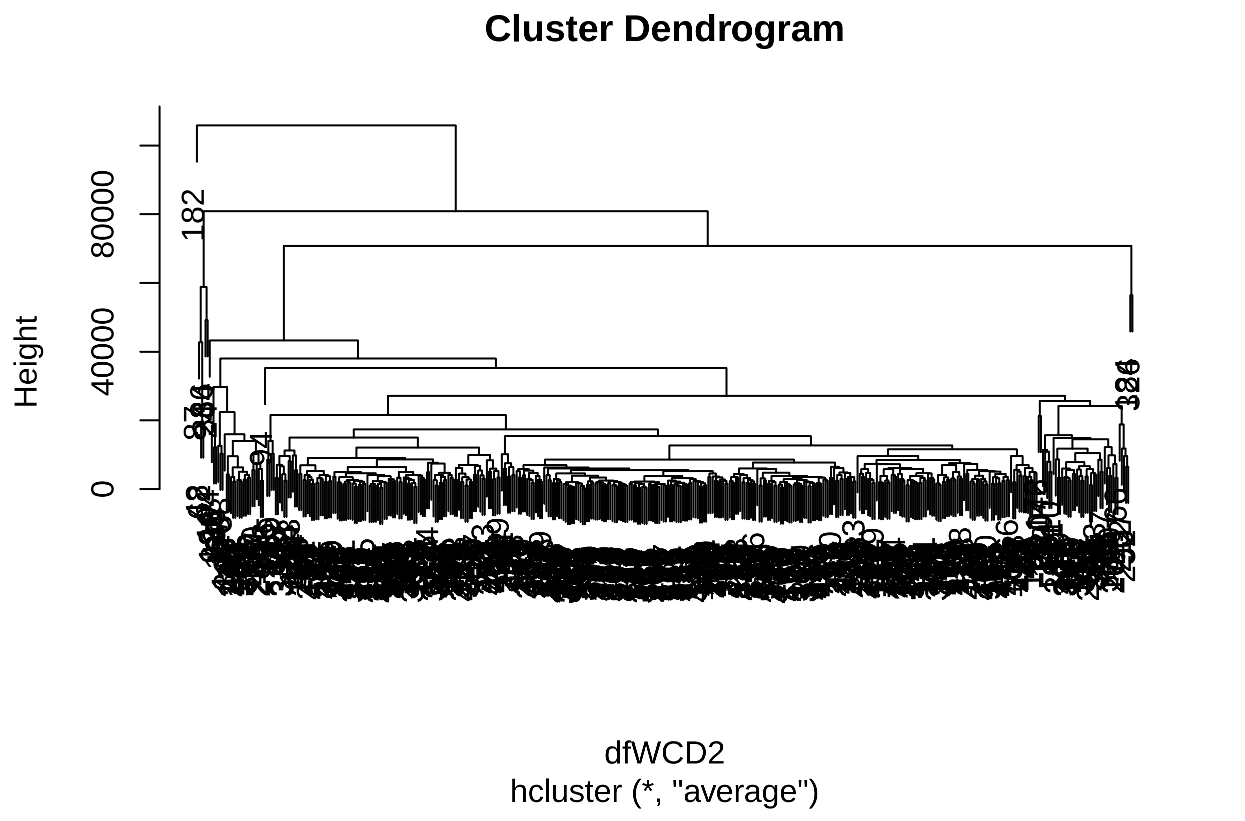
1.5.1 Plotting to deterimine the cluster level.

```r
plot(dfWCD2.h.clust, labels = FALSE)
rect.hclust(dfWCD2.h.clust, k=3, border="red")
rect.hclust(dfWCD2.h.clust, k=5, border="green")
rect.hclust(dfWCD2.h.clust, k=9, border="blue")
```

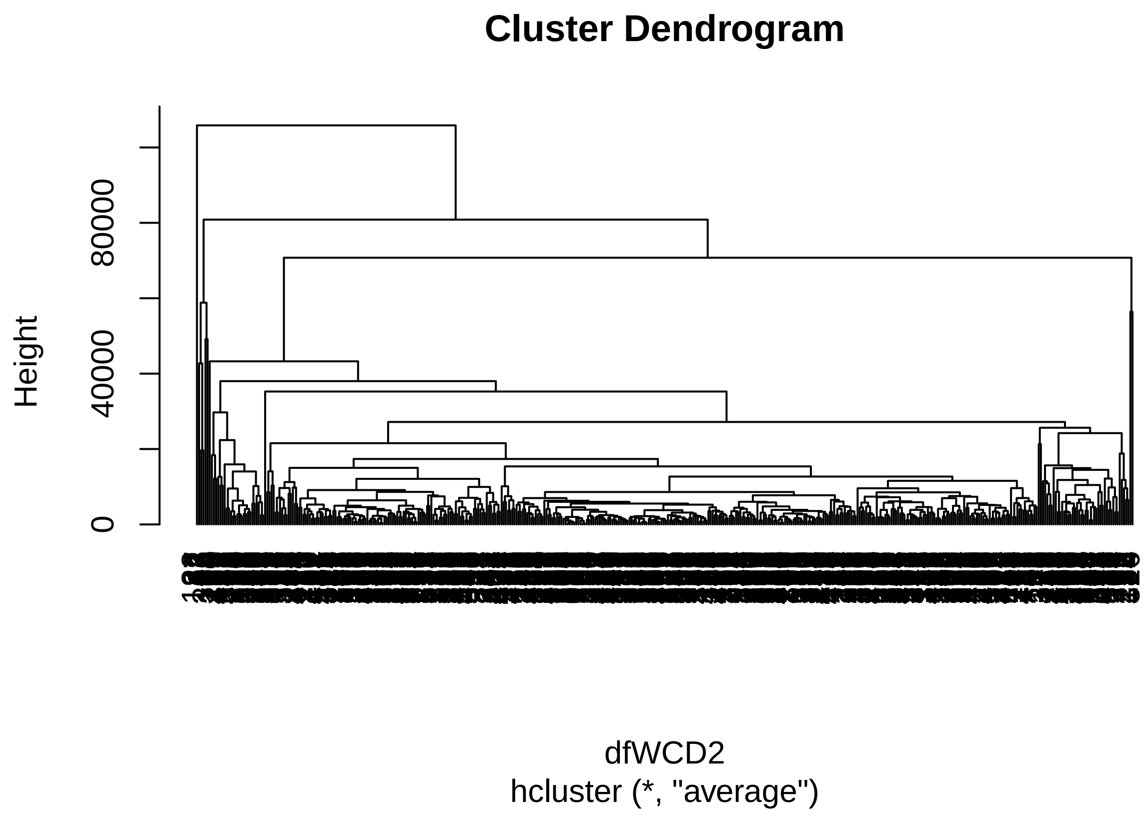


1.5.2 Hierarchical clustering using centroid clustering and squared Euclidean distance

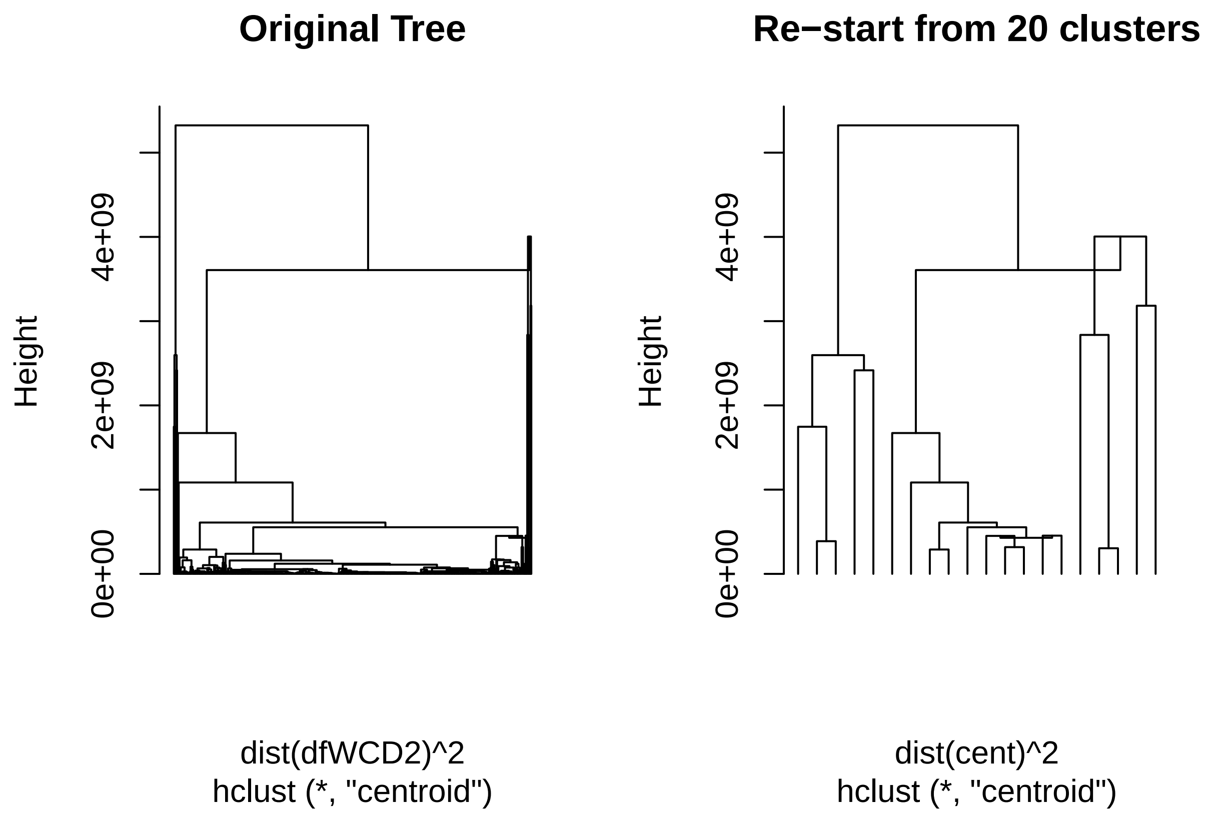
```
h_c <- hcluster(dfWCD2,link = "ave") # require(amap)
plot(h_c)
```



```r
plot(h_c, hang = -1)
```



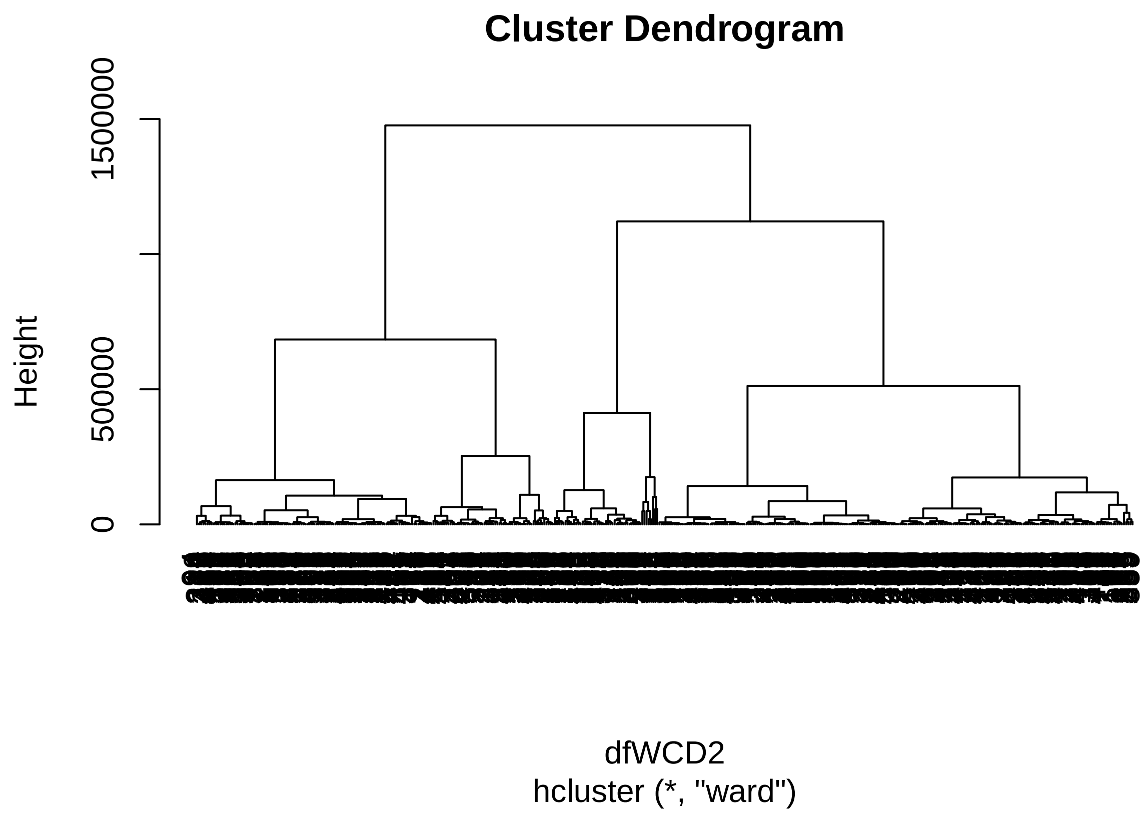
```
### centroid clustering and squared Euclidean distance
h_c<- hclust(dist(dfWCD2)^2, "cen")

### Cutting the tree into 20 clusters and reconstruct upper part of the tree from cluster center
memb <- cutree(h_c, k = 20)
cent <- NULL
for(k in 1:20){
  cent <- rbind(cent, colMeans(dfWCD2[,-1][memb == k, , drop = FALSE]))
}
h_c1 <- hclust(dist(cent)^2, method = "cen", members = table(memb))
opar <- par(mfrow = c(1, 2))
plot(h_c,  labels = FALSE, hang = -1, main = "Original Tree")
plot(h_c1, labels = FALSE, hang = -1, main = "Re-start from 20 clusters")
```



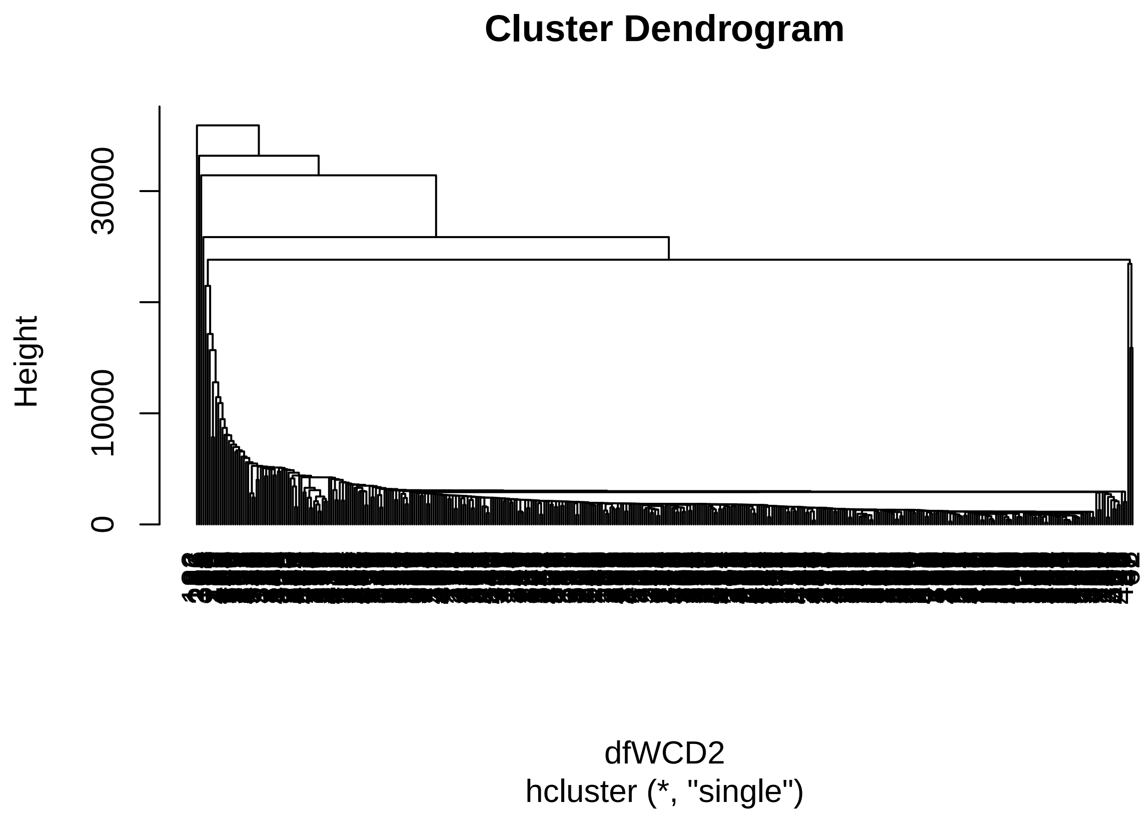
```
par(opar)
```

1.5.3 Other combinations

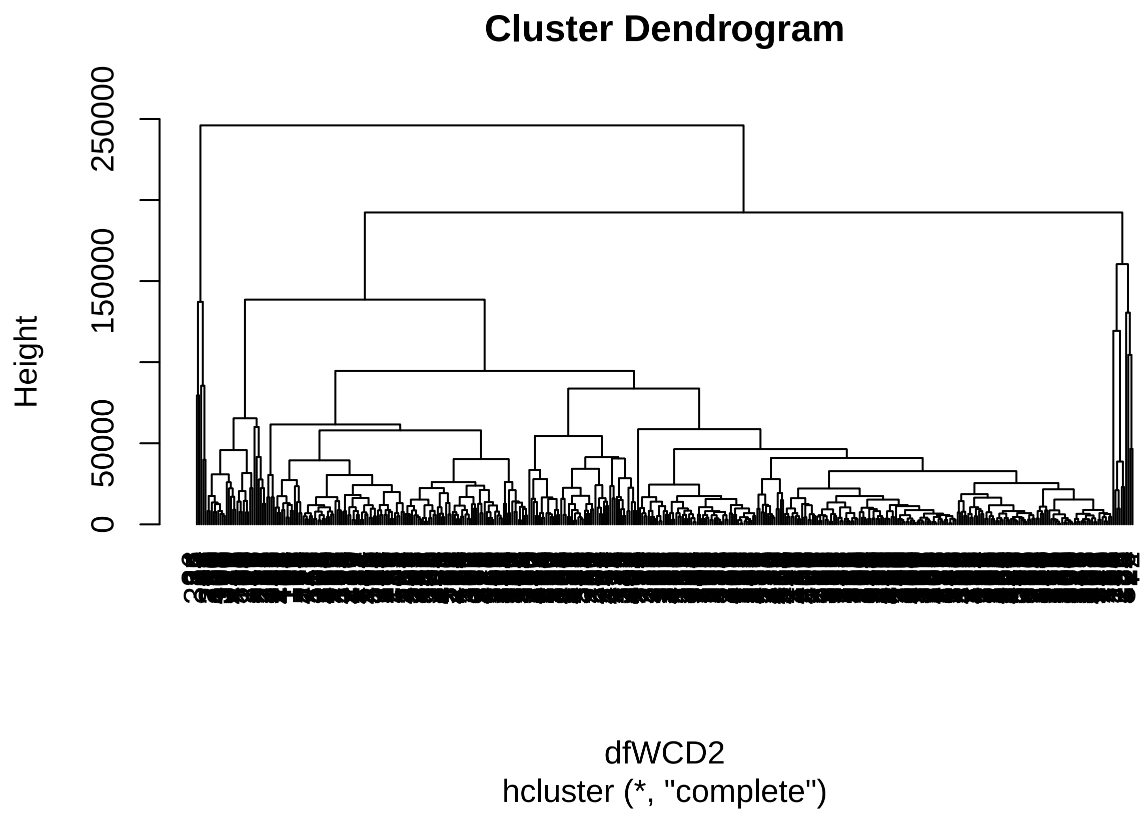
```
## other combinations

h_c <- hcluster(dfWCD2,method = "euc",link = "ward", nbproc= 1,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



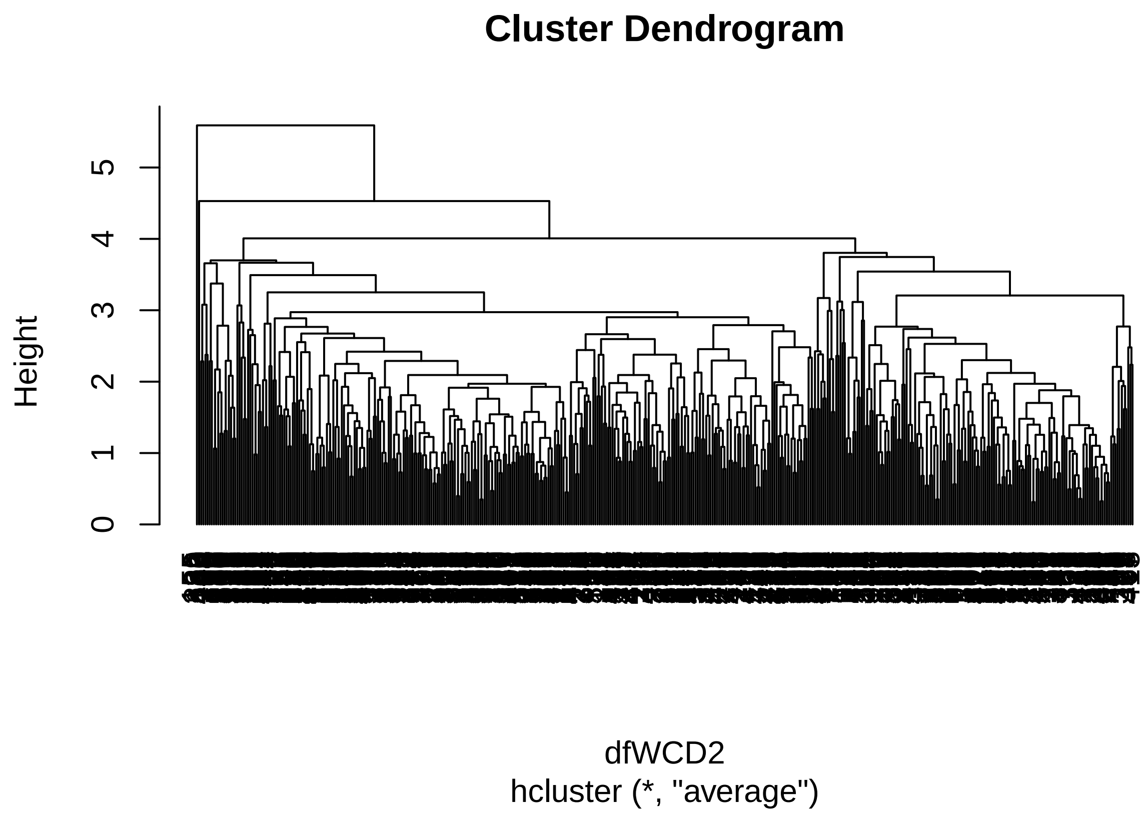
```r
h_c <- hcluster(dfWCD2,method = "max",link = "single", nbproc= 2,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



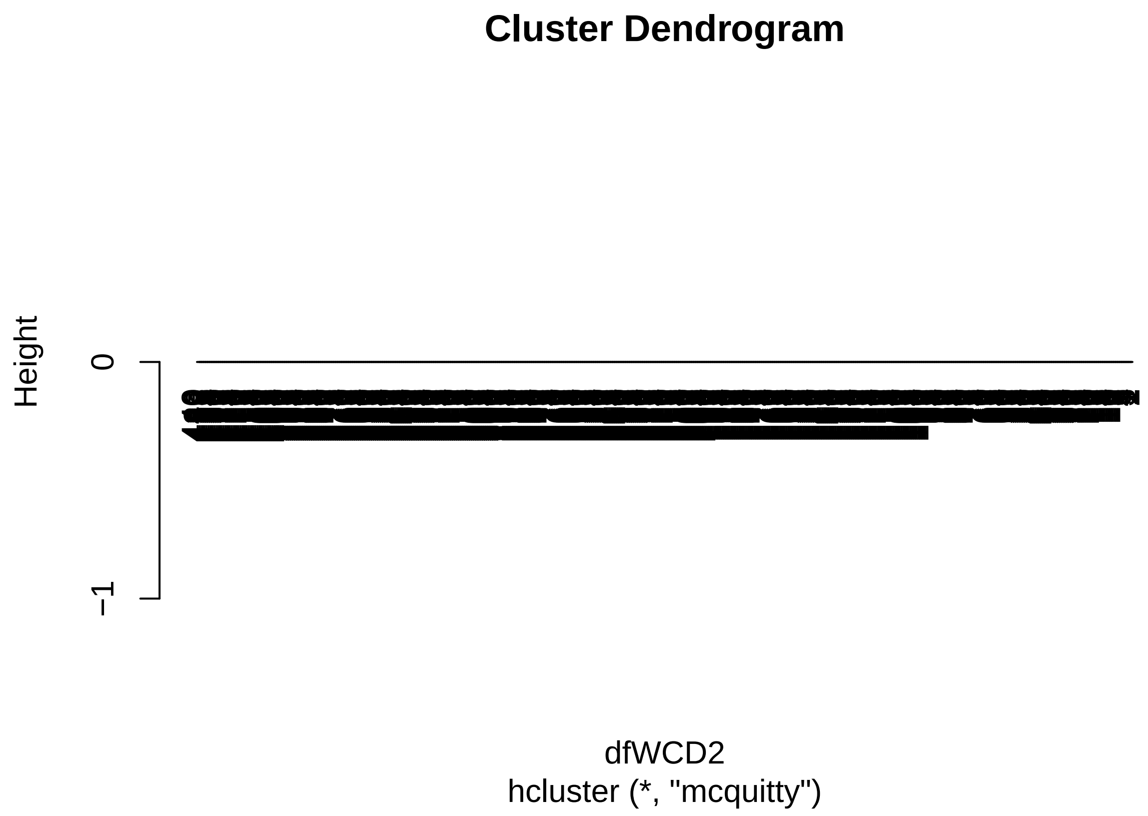
```r
h_c <- hcluster(dfWCD2,method = "man",link = "complete", nbproc= 1,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



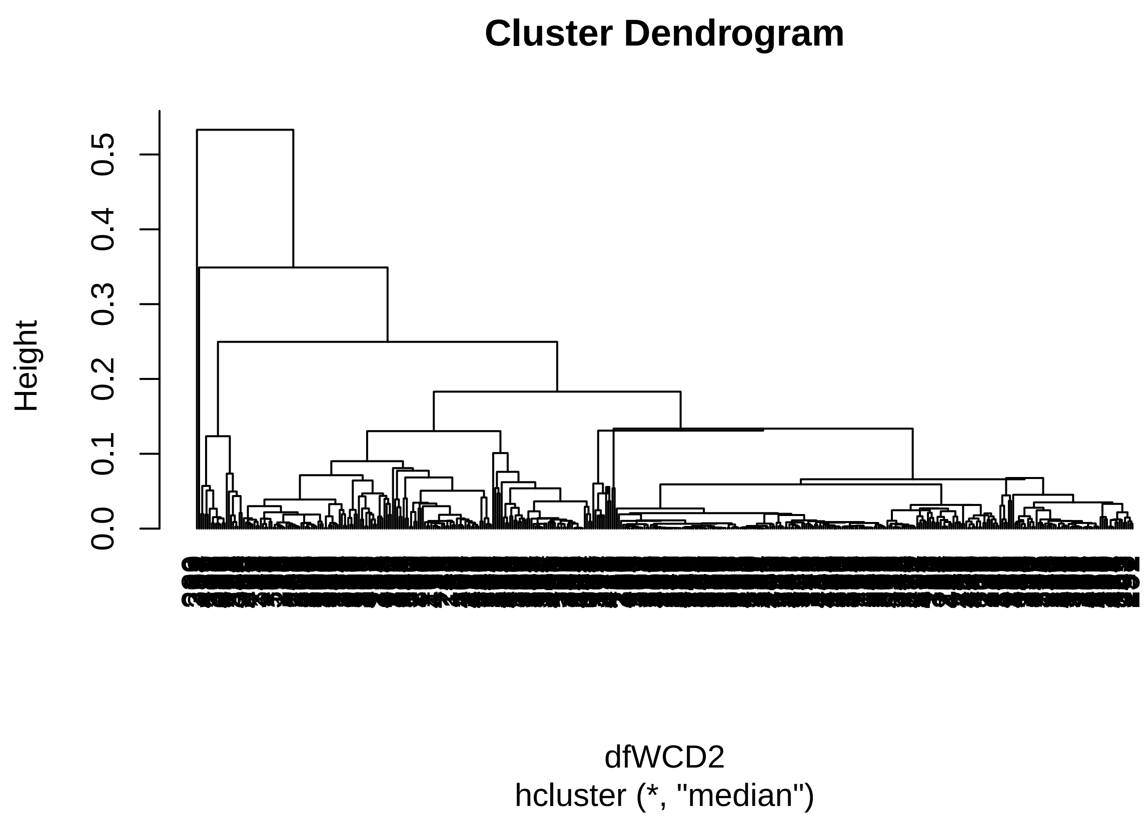
```r
h_c <- hcluster(dfWCD2,method = "can",link = "average", nbproc= 2,
                doubleprecision = TRUE)
plot(h_c, hang = -1)
```



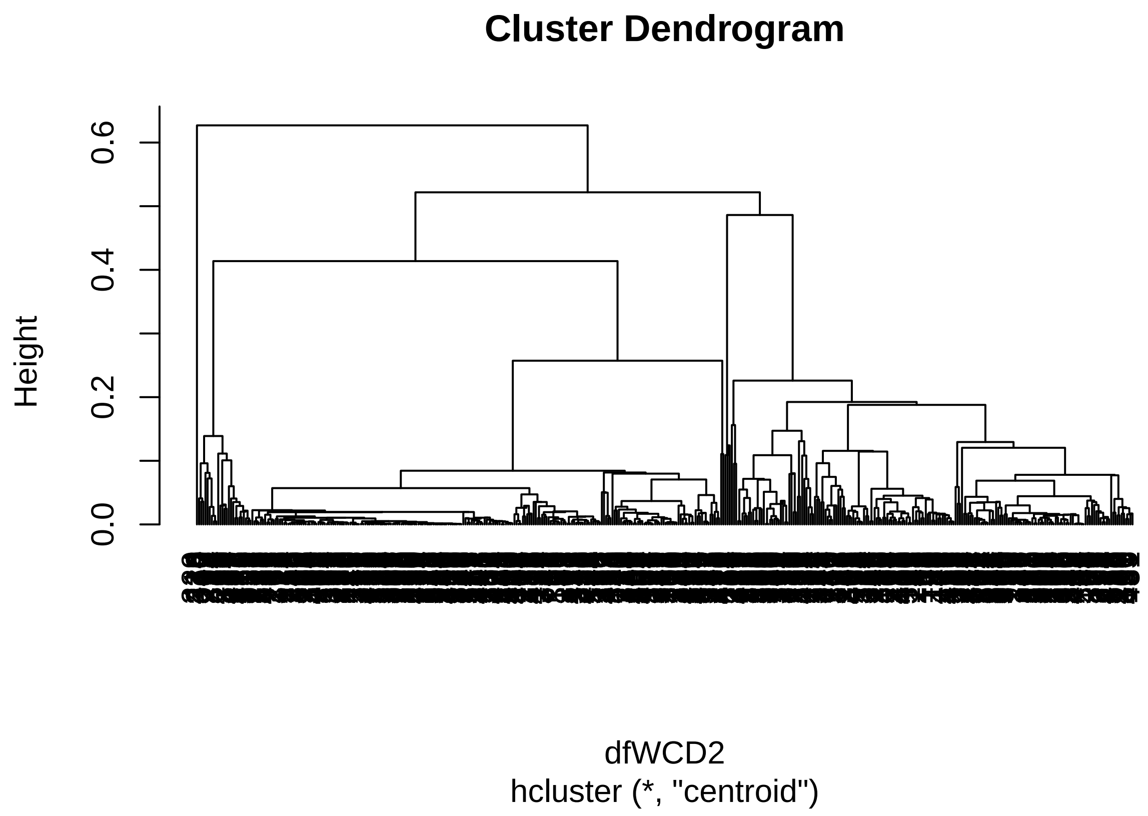
```
h_c <- hcluster(dfWCD2,method = "bin",link = "mcquitty", nbproc= 1,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```



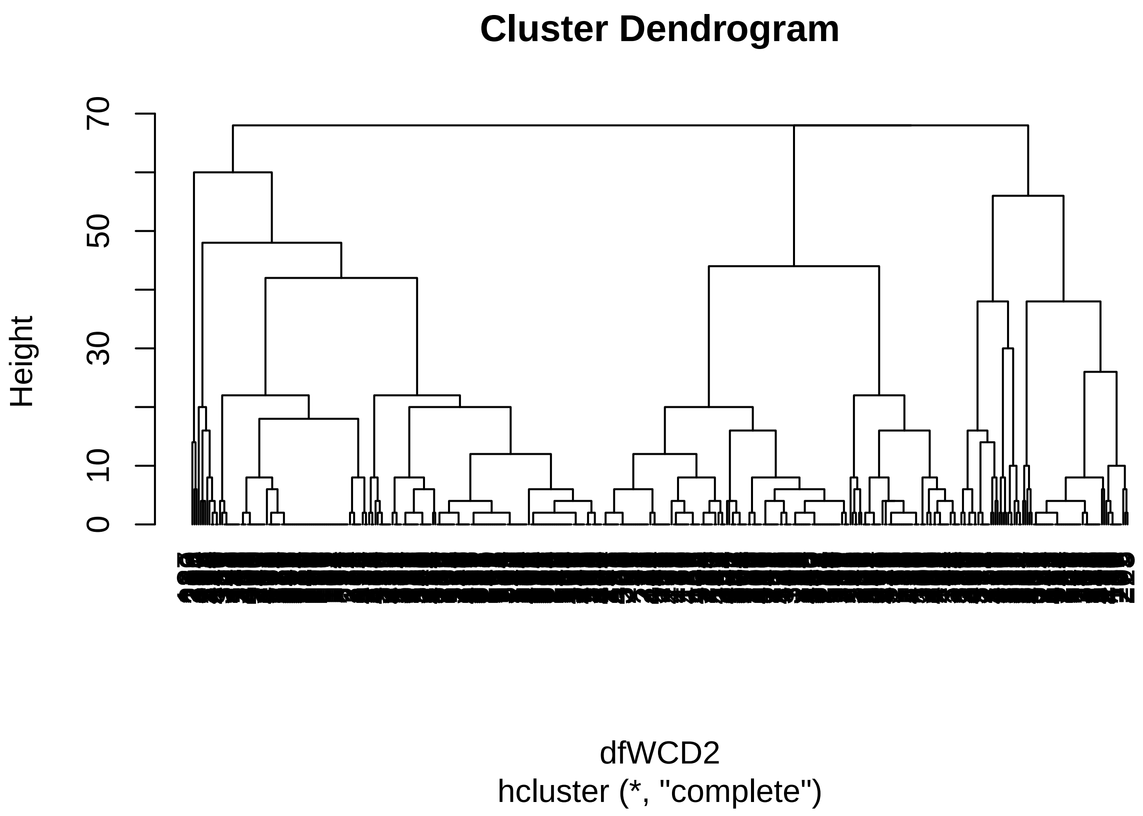
```r
h_c <- hcluster(dfWCD2,method = "pea",link = "median", nbproc= 2,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```



```r
h_c <- hcluster(dfWCD2,method = "cor",link = "centroid", nbproc= 1,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```



```r
h_c <- hcluster(dfWCD2,method = "spe",link = "complete", nbproc= 2,
                doubleprecision = FALSE)
plot(h_c, hang = -1)
```



1.6 Questions

1. How did you choose a k for k-means? I used a mixture of the averaged Silhouette width and Gap statistic, Hartigan's rule, gap analysis, and trial and error. I eventually settled with 4 after reviewing the gap analysis.

2. Evaluate the model performance. How do the clustering approaches compare on the same data? K-means clustering with 4 clusters of sizes 93, 281, 59, 7. The cluster of seven looked low, but after analysis it seemed ok. I started by analyzing the raw data, but I could not make much sense of the data until it was in the centroid plot, then it looked like it fit, the seven all clustered together nicely. Overall the plots look good too, in the individual plots I can see clustering and the clustering looks acceptable.

3. Generate and plot confusion matrices for the k-means and PAM. What do they tell you? I plotted four different confusion plots, two for milk and two for grocery for k-means and pam. For the milk data, both the k-means and pam data show me that most of the data is about 50/50 for both groups, as I would expect. The grocery data is a little different, the larger group is almost 60/40 (false/true), but the smaller group has 2/3 false to true.

4. Generate centroid plots against the 1st two discriminant functions for k-means and PAM. What do they tell you? For k-means, the centroid plot looks good. I can see that the small cluster is grouped ok, however, there is some overlapping with a couple of clusters, but I do not think this is an issue. The pam plot is similar, but there seems to be a little overlap with the clusters

5. Generate silhouette plots for PAM. What do they tell you? The silhouette plot doesn't look too good. A good plot should have a lot of long lines, it appears that there are more short lines with this data.

6. For the hierarchical clustering use all linkage methods (Single Link, Complete Link, Average

Link, Centroid and Minimum energy clustering) and generate dendograms. How do they compare on the same data? They all look different. The single is hard to read, the complete looks ok, the average and centroid have a couple clads going off to nowhere, and the ward plot looks the best, it is organized nicely.

7. For the hierarchical clustering use both agglomerative and divisive clustering with a linkage method of your choice and generate dendograms. How do they compare on the same data? In section 1.5.3 I ran many different combinations to generate dendograms and they all look different. The mcquitty plot doesn't look like anything.

8. For the hierarchical clustering use centroid clustering and squared Euclidean distance and generate dendograms. How do they compare on the same data? The original tree is hard to read, especially toward the bottom. The re-start from 20 clusters looks great and it is easy to read.