Cluster Analysis with R

Gabriel Martos

Clustering wines

K-Means

This first example is to learn to make cluster analysis with R. The library *rattle* is loaded in order to use the data set *wines*.

```
# install.packages('rattle')
data(wine, package='rattle')
head(wine)
```

```
##   Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
## 1    1   14.23  1.71 2.43       15.6       127    2.80       3.06
## 2    1   13.20  1.78 2.14       11.2       100    2.65       2.76
## 3    1   13.16  2.36 2.67       18.6       101    2.80       3.24
## 4    1   14.37  1.95 2.50       16.8       113    3.85       3.49
## 5    1   13.24  2.59 2.87       21.0       118    2.80       2.69
## 6    1   14.20  1.76 2.45       15.2       112    3.27       3.39
##   Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
## 1          0.28            2.29  5.64 1.04     3.92    1065
## 2          0.26            1.28  4.38 1.05     3.40    1050
## 3          0.30            2.81  5.68 1.03     3.17    1185
## 4          0.24            2.18  7.80 0.86     3.45    1480
## 5          0.39            1.82  4.32 1.04     2.93     735
## 6          0.34            1.97  6.75 1.05     2.85    1450
```

In this data set we observe the composition of different wines. Given a set of observations $(x_1, x_2, ., x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into ($k \leq n$) $S = \{S_1, S_2, ., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS). In other words, its objective is to find::

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

where μ_i is the mean of points in S_i. The clustering optimization problem is solved with the function *kmeans* in R.

```
wine.stand <- scale(wine[-1])  # To standarize the variables

# K-Means
k.means.fit <- kmeans(wine.stand, 3) # k = 3
```

In **k.means.fit** are contained all the elements of the cluster output:

```
attributes(k.means.fit)
```

```
## $names
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
##
## $class
## [1] "kmeans"
```

```
# Centroids:
k.means.fit$centers
```

```
##   Alcohol   Malic     Ash Alcalinity Magnesium  Phenols Flavanoids
## 1  0.1644  0.8691  0.1864     0.5229  -0.07526 -0.97658   -1.21183
## 2  0.8329 -0.3030  0.3637    -0.6085   0.57596  0.88275    0.97507
## 3 -0.9235 -0.3929 -0.4931     0.1701  -0.49033 -0.07577    0.02075
##   Nonflavanoids Proanthocyanins   Color     Hue Dilution Proline
## 1       0.72402         -0.7775  0.9389 -1.1615  -1.2888 -0.4059
## 2      -0.56051          0.5787  0.1706  0.4727   0.7771  1.1220
## 3      -0.03344          0.0581 -0.8994  0.4605   0.2700 -0.7517
```

```
# Clusters:
k.means.fit$cluster
```

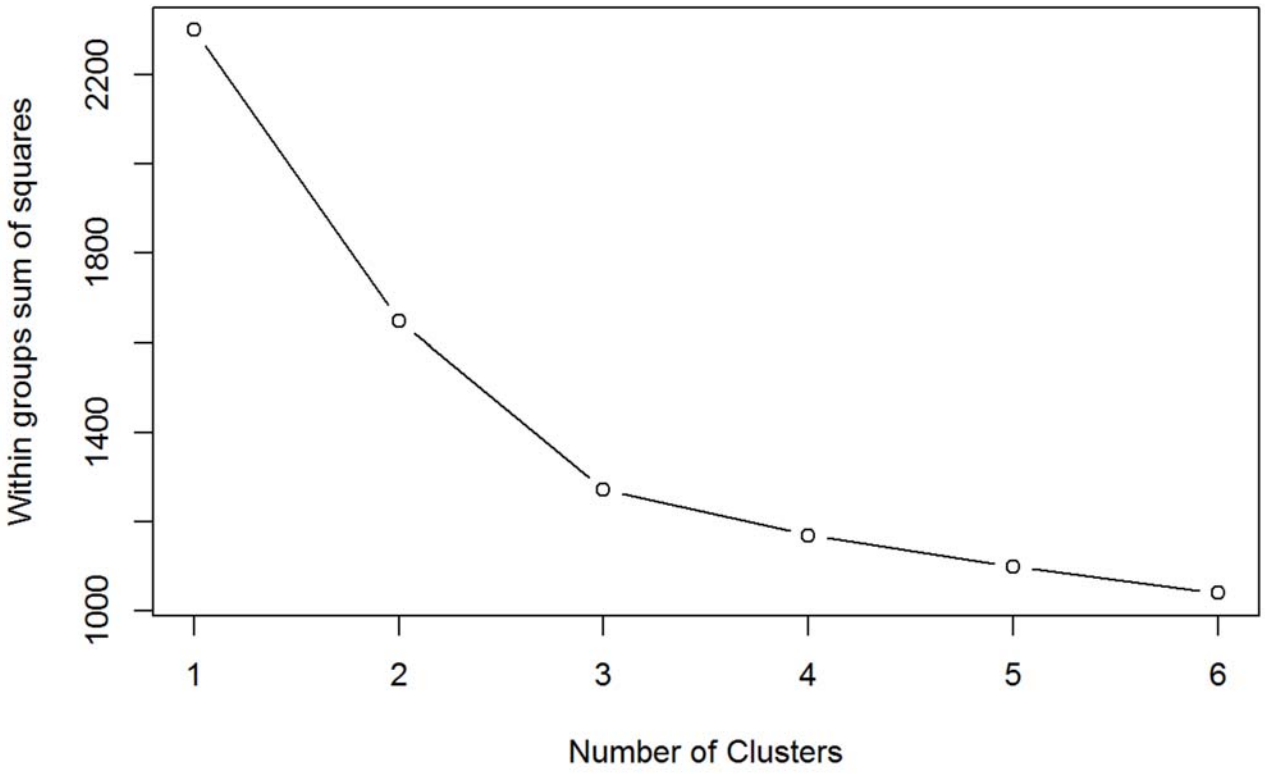
```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3
##  [71] 3 3 3 2 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 2 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1
## [141] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1
```

```
# Cluster size:
k.means.fit$size
```

```
## [1] 51 62 65
```

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".

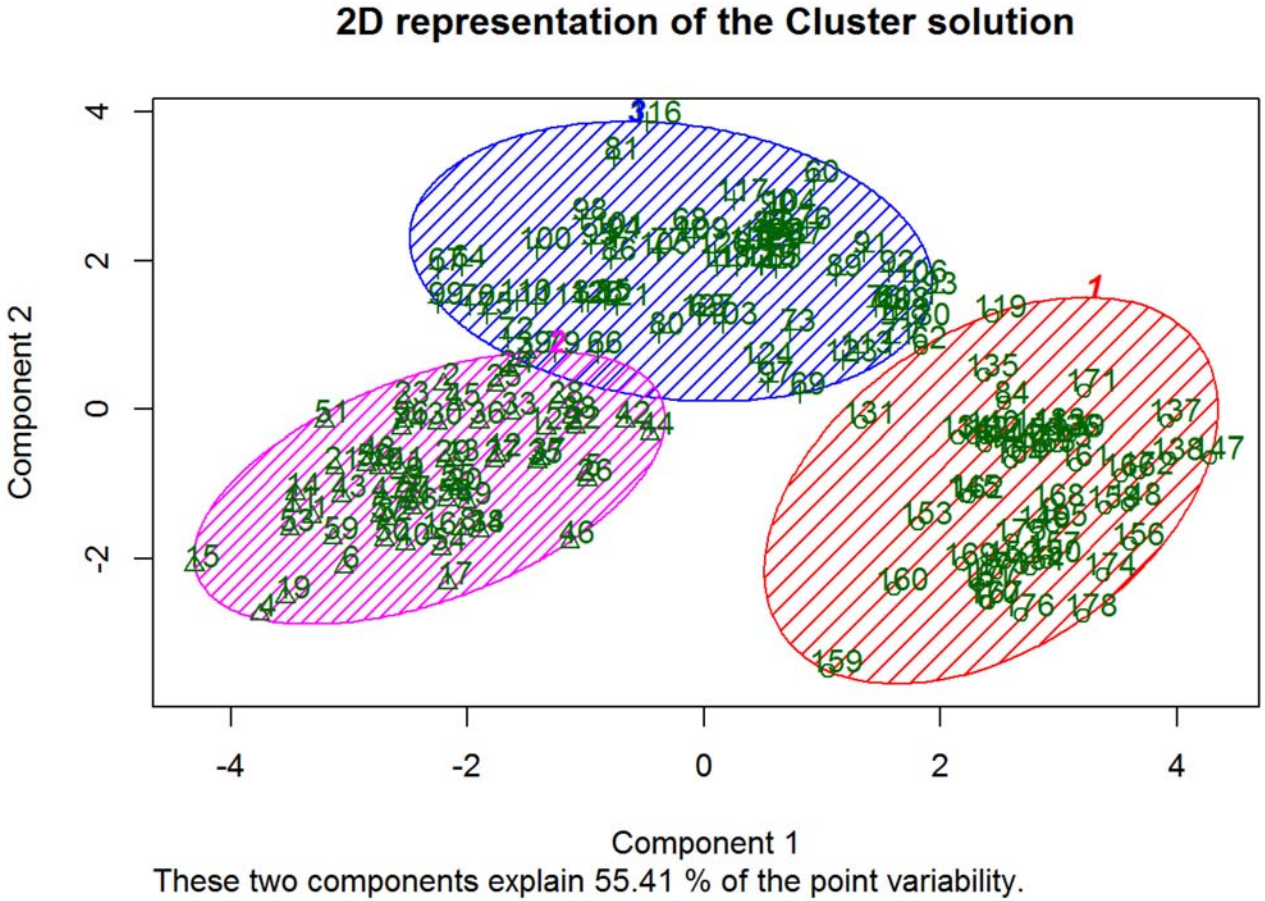
```
wssplot <- function(data, nc=15, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(wine.stand, nc=6)
```




```
Which is the optimal value for k in this case? Why?
```

Library **clusters** allow us to represent (with the aid of PCA) the cluster solution into 2 dimensions:

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



In order to evaluate the clustering performance we build a confusion matrix:

```
table(wine[,1],k.means.fit$cluster)
```

```
##
##      1  2  3
##   1  0 59  0
##   2  3  3 65
##   3 48  0  0
```

Hierarchical clustering:

Hierarchical methods use 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 elements may be close to one another according to one distance and farther away according to another.

```
d <- dist(wine.stand, method = "euclidean") # Euclidean distance matrix.
```

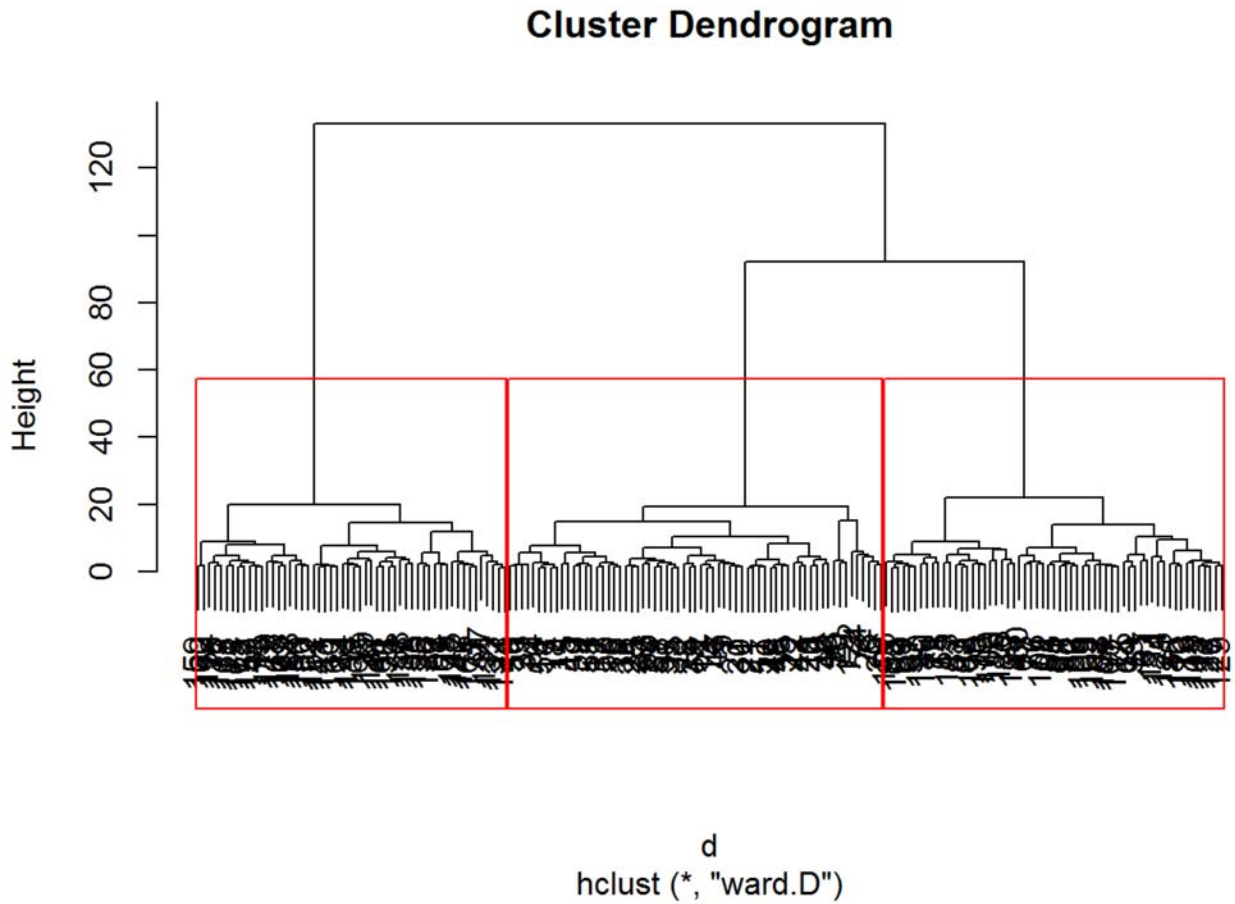
We use the Euclidean distance as an input for the clustering algorithm (Ward's minimum variance criterion minimizes the total within-cluster variance):

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

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

The clustering output can be displayed in a *dendrogram*

```
plot(H.fit) # display dendogram
groups <- cutree(H.fit, k=3) # cut tree into 5 clusters
# draw dendogram with red borders around the 5 clusters
rect.hclust(H.fit, k=3, border="red")
```



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

```
table(wine[,1],groups)
```

```
##    groups
##      1  2  3
##   1 58  1  0
##   2  7 58  6
##   3  0  0 48
```

Study case I: EUROPEAN PROTEIN CONSUMPTION

We consider 25 European countries (n = 25 units) and their protein intakes (in percent) from nine major food sources (p = 9). The data are listed below.

```
url = 'http://www.biz.uiowa.edu/faculty/jledolter/DataMining/protein.csv'
food <- read.csv(url)
head(food)
```

```
##          Country RedMeat WhiteMeat Eggs Milk Fish Cereals Starch Nuts
## 1        Albania    10.1       1.4  0.5  8.9  0.2    42.3    0.6  5.5
## 2        Austria     8.9      14.0  4.3 19.9  2.1    28.0    3.6  1.3
## 3        Belgium    13.5       9.3  4.1 17.5  4.5    26.6    5.7  2.1
## 4       Bulgaria     7.8       6.0  1.6  8.3  1.2    56.7    1.1  3.7
## 5 Czechoslovakia     9.7      11.4  2.8 12.5  2.0    34.3    5.0  1.1
## 6        Denmark    10.6      10.8  3.7 25.0  9.9    21.9    4.8  0.7
##   Fr.Veg
## 1    1.7
## 2    4.3
## 3    4.0
## 4    4.2
## 5    4.0
## 6    2.4
```

We start first, clustering on just *Red* and *White* meat (p=2) and k=3 clusters.

```
set.seed(123456789) ## to fix the random starting clusters
grpMeat <- kmeans(food[,c("WhiteMeat","RedMeat")], centers=3, nstart=10)
grpMeat
```

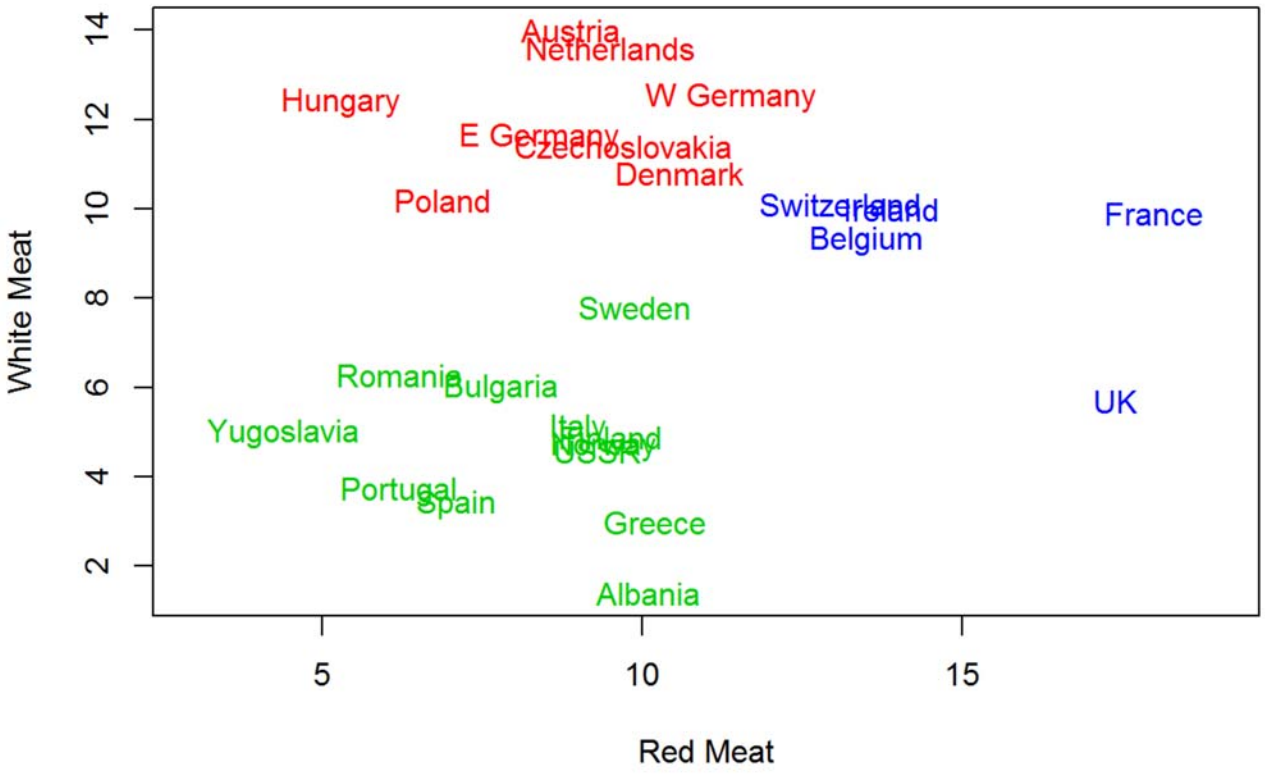
```
## K-means clustering with 3 clusters of sizes 8, 12, 5
##
## Cluster means:
##   WhiteMeat RedMeat
## 1    12.062   8.838
## 2     4.658   8.258
## 3     9.000  15.180
##
## Clustering vector:
##  [1] 2 1 3 2 1 1 1 2 3 2 1 3 2 1 2 1 2 2 2 2 3 3 2 1 2
##
## Within cluster sum of squares by cluster:
## [1] 39.46 69.86 35.67
##  (between_SS / total_SS =  75.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
## list of cluster assignments
o=order(grpMeat$cluster)
data.frame(food$Country[o],grpMeat$cluster[o])
```

```
##    food.Country.o. grpMeat.cluster.o.
## 1          Austria                  1
## 2   Czechoslovakia                  1
## 3          Denmark                  1
## 4        E Germany                  1
## 5          Hungary                  1
## 6      Netherlands                  1
## 7           Poland                  1
## 8        W Germany                  1
## 9          Albania                  2
## 10        Bulgaria                  2
## 11         Finland                  2
## 12          Greece                  2
## 13           Italy                  2
## 14          Norway                  2
## 15        Portugal                  2
## 16         Romania                  2
## 17           Spain                  2
## 18          Sweden                  2
## 19            USSR                  2
## 20      Yugoslavia                  2
## 21         Belgium                  3
## 22          France                  3
## 23         Ireland                  3
## 24     Switzerland                  3
## 25              UK                  3
```

To see a graphical representation of the clustering solution we plot cluster assignments on Red and White meat on a scatter plot:

```
plot(food$Red, food$White, type="n", xlim=c(3,19), xlab="Red Meat", ylab="White Meat")
text(x=food$Red, y=food$White, labels=food$Country,col=grpMeat$cluster+1)
```

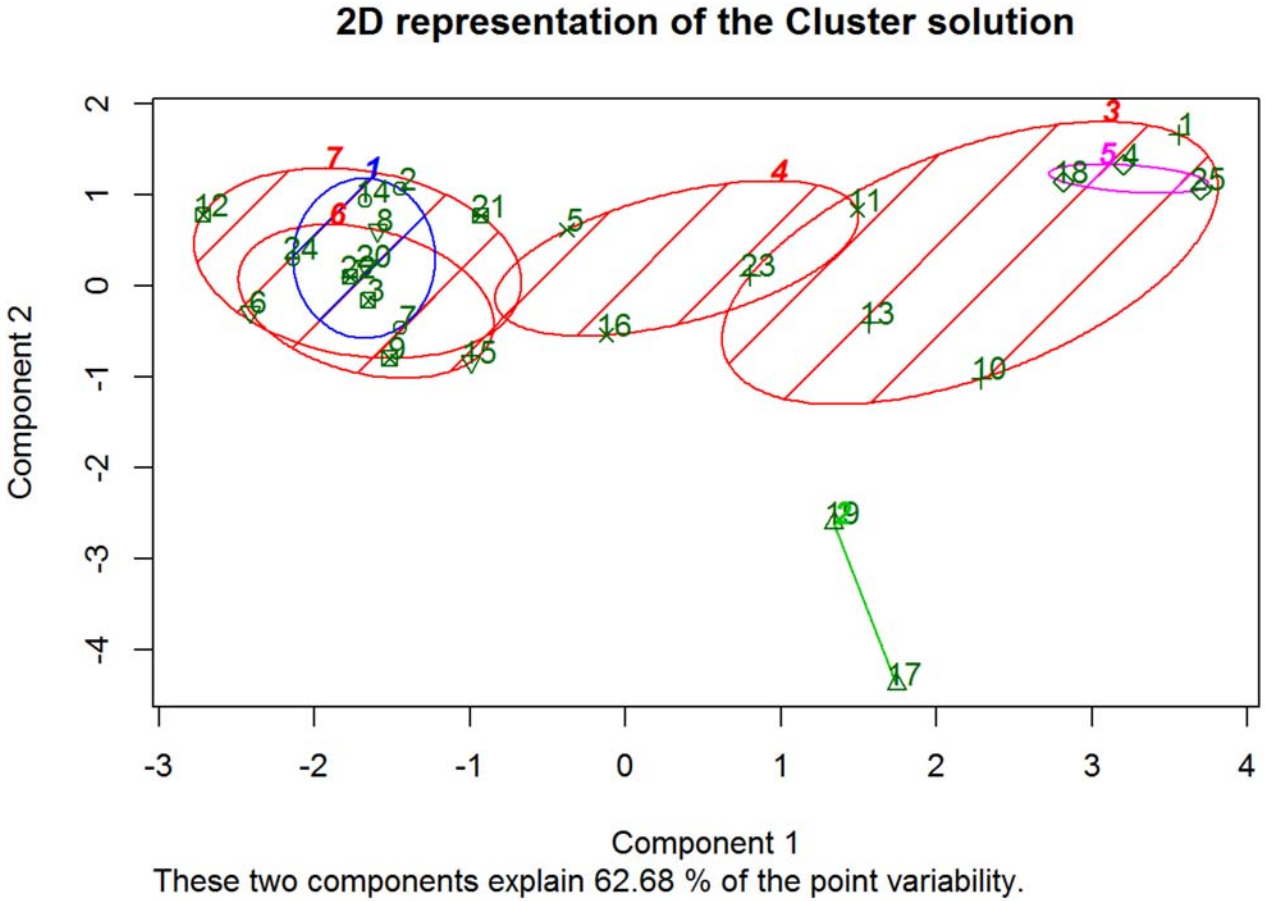


Next, we cluster on all nine protein groups and prepare the program to create seven clusters. The resulting clusters, shown in color on a scatter plot of white meat against red meat (any other pair of features could be selected), actually makes lot of sense. Countries in close geographic proximity tend to be clustered into the same group.

```
## same analysis, but now with clustering on all
## protein groups change the number of clusters to 7
set.seed(123456789)
grpProtein <- kmeans(food[,-1], centers=7, nstart=10)
o=order(grpProtein$cluster)
data.frame(food$Country[o],grpProtein$cluster[o])
```

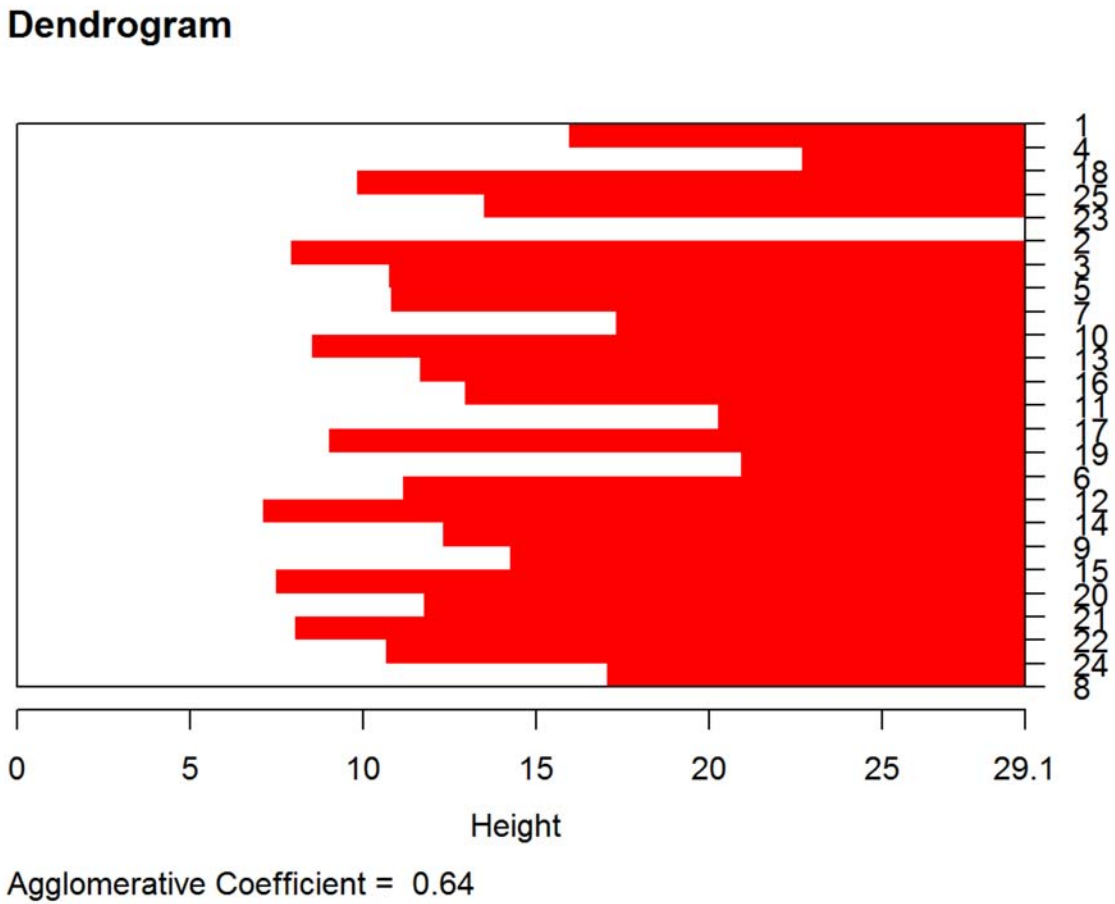
```
##    food.Country.o. grpProtein.cluster.o.
## 1          Austria                     1
## 2        E Germany                     1
## 3      Netherlands                     1
## 4        W Germany                     1
## 5         Portugal                     2
## 6            Spain                     2
## 7          Albania                     3
## 8           Greece                     3
## 9            Italy                     3
## 10            USSR                     3
## 11  Czechoslovakia                     4
## 12         Hungary                     4
## 13          Poland                     4
## 14        Bulgaria                     5
## 15         Romania                     5
## 16      Yugoslavia                     5
## 17         Denmark                     6
## 18         Finland                     6
## 19          Norway                     6
## 20          Sweden                     6
## 21         Belgium                     7
## 22          France                     7
## 23         Ireland                     7
## 24     Switzerland                     7
## 25              UK                     7
```

```
library(cluster)
clusplot(food[,-1], grpProtein$cluster, main='2D representation of the Cluster solution', color=TRUE, shade=TRUE, labels=2, lines=0)
```

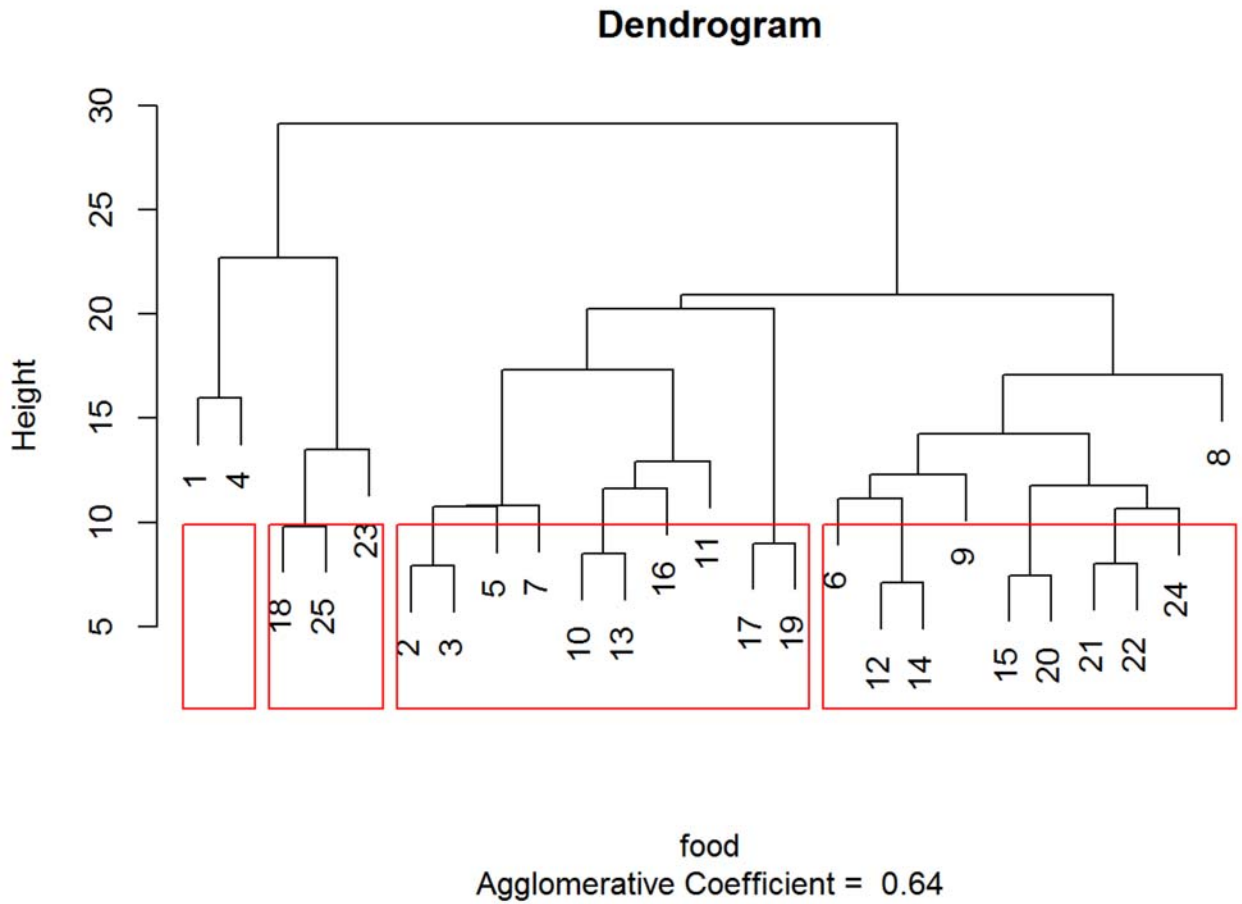


Alternatively we can implement a Hierarchical approach. We use the *agnes* function in the package *cluster*. Argument diss=FALSE indicates that we use the dissimilarity matrix that is being calculated from raw data. Argument metric="euclidian" indicates that we use Euclidean distance. No standardization is used and the link function is the "average" linkage.

```
foodagg=agnes(food,diss=FALSE,metric="euclidian")
plot(foodagg, main='Dendrogram') ## dendrogram
```




```
groups <- cutree(foodagg, k=4) # cut tree into 3 clusters
rect.hclust(foodagg, k=4, border="red")
```



Study case II: Customer Segmentation

Customer segmentation is as simple as it sounds: grouping customers by their characteristics - and why would you want to do that? To better serve their needs!

Our example is to do with e-mail marketing. We use the dataset from this link (http://www.salemmarafi.com/wp-content/uploads/2014/04/clustering-vanilla.xls)

```
offers<-read.table('offers.csv', sep = ';', header=T)
head(offers)
```

```
##   OfferID Campaign           Varietal MinimumQt Discount      Origin
## 1       1  January             Malbec        72       56      France
## 2       2  January          Pinot Noir        72       17      France
## 3       3 February           Espumante       144       32      Oregon
## 4       4 February           Champagne        72       48      France
## 5       5 February Cabernet Sauvignon       144       44 New Zealand
## 6       6    March            Prosecco       144       86       Chile
##   PastPeak
## 1    FALSE
## 2    FALSE
## 3     TRUE
## 4     TRUE
## 5     TRUE
## 6    FALSE
```

```
transactions<-read.table('transactions.csv', sep = ';', header=T)
head(transactions)
```

```
##   CustomerLastName OfferID
## 1            Smith       2
## 2            Smith      24
## 3          Johnson      17
## 4          Johnson      24
## 5          Johnson      26
## 6         Williams      18
```

Step 1: Organizing the information

We have two data sets: one for the offers and the other for the transactions. First what we need to do is create a transaction matrix. That means, we need to put the offers we mailed out next to the transaction history of each customer. This is easily achieved with a pivot table.

```
# Create transaction matrix (a pivot table like in Excel way!)
library(reshape)
pivot<-melt(transactions[1:2])
```

```
## Using CustomerLastName as id variables
```

```
pivot<-(cast(pivot,value~CustomerLastName,fill=0,fun.aggregate=function(x) length(x)))
pivot<-cbind(offers,pivot[-1])

# write.csv(file="pivot.csv",pivot) # to save your data

cluster.data<-pivot[,8:length(pivot)]
cluster.data<-t(cluster.data)
head(cluster.data)
```

```
##          1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## Adams    0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
## Allen    0 0 0 0 0 0 0 0 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## Anderson 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
## Bailey   0 0 0 0 0 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## Baker    0 0 0 0 0 0 1 0 0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
## Barnes   0 0 0 0 0 0 0 0 0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0
##          26 27 28 29 30 31 32
## Adams     0  0  0  1  1  0  0
## Allen     0  1  0  0  0  0  0
## Anderson  1  0  0  0  0  0  0
## Bailey    0  0  0  0  1  0  0
## Baker     0  0  0  0  0  1  0
## Barnes    0  0  0  0  0  1  0
```

In the clustering data set, rows represents costumers and columns are different wine brands/types.

Step 2: Distances and Clusters

We will use $k = 4$ indicating that we will use 4 clusters. This is somewhat arbitrary, but the number you pick should be representative of the number of segments you can handle as a business. So 100 segments does not make sense for an e-mail marketing campaign.

We need to calculate how far away each customer is from the cluster's mean. To do this we could use many distances/dissimilarity index, one of which is the Gower dissimilarity.

```
library(cluster)
D=daisy(cluster.data, metric='gower')
```

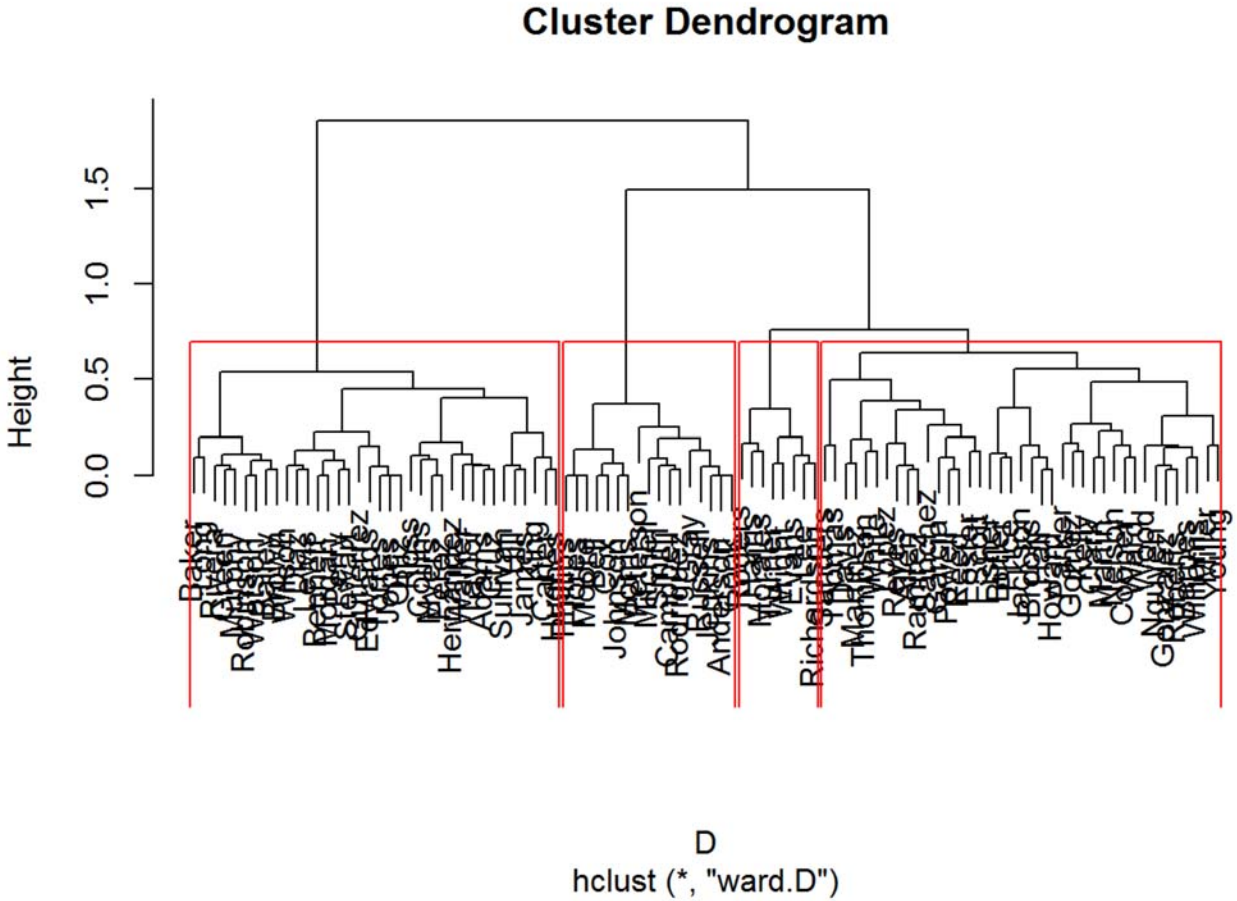
```
## Warning: binary variable(s) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
## 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
## treated as interval scaled
```

After the creation of a distance matrix, we implement a Ward's hierarchical clustering procedure:

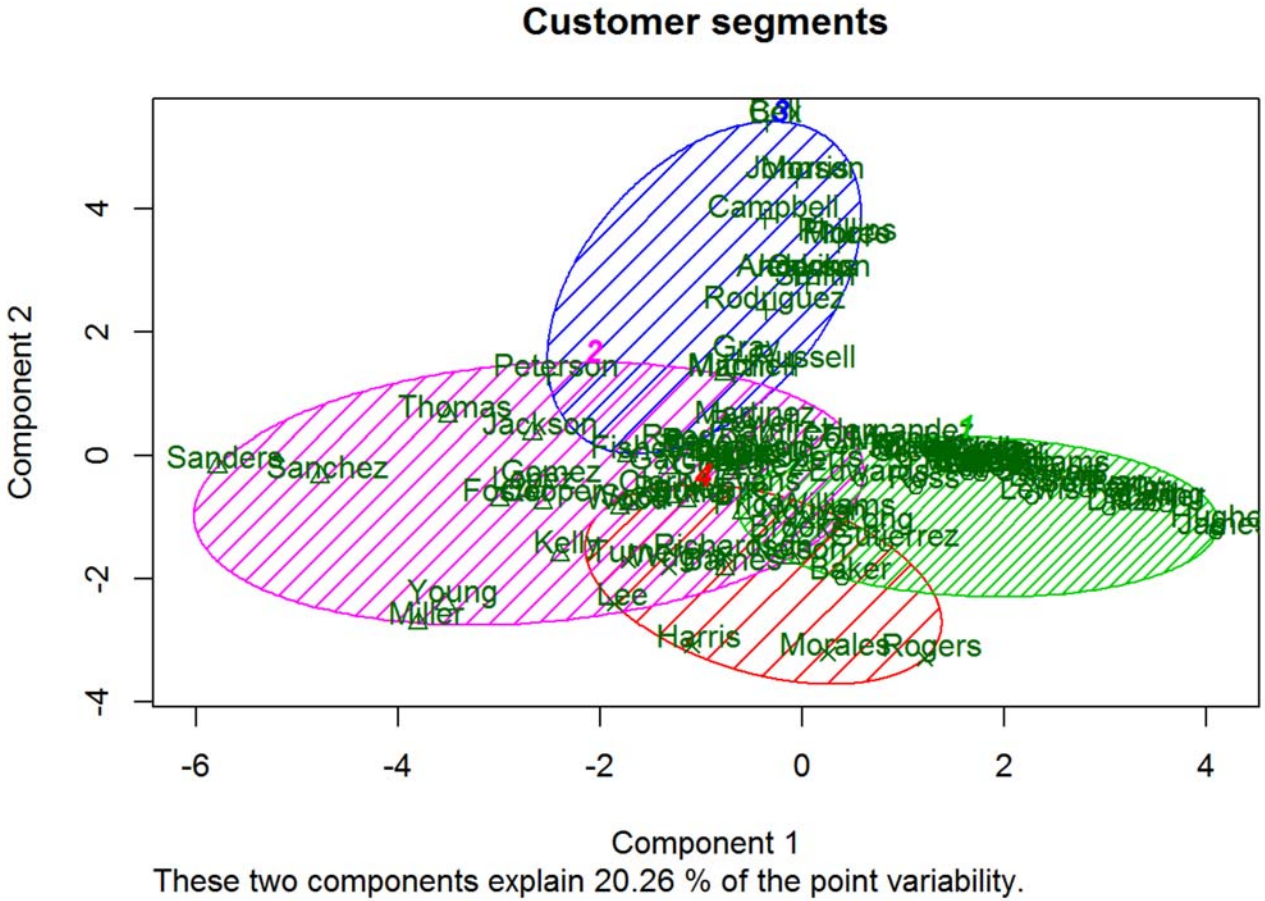
```
H.fit <- hclust(D, method="ward")
```

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

```
plot(H.fit) # display dendrogram

groups <- cutree(H.fit, k=4) # cut tree into 4 clusters

# draw dendogram with red borders around the 4 clusters
rect.hclust(H.fit, k=4, border="red")
```




```
# 2D representation of the Segmentation:
clusplot(cluster.data, groups, color=TRUE, shade=TRUE,
         labels=2, lines=0, main= 'Customer segments')
```



Top get the top deals we will have to do a little bit of data manipulation. First we need to combine our clusters and transactions. Notably the lengths of the 'tables' holding transactions and clusters are different. So we need a way to merge the data . so we use the *merge()* function and give our columns sensible names:

```
# Merge Data

cluster.deals<-merge(transactions[1:2],groups,by.x = "CustomerLastName", by.y = "row.names")

colnames(cluster.deals)<-c("Name","Offer","Cluster")
head(cluster.deals)
```

```
##       Name Offer Cluster
## 1    Adams    18       1
## 2    Adams    29       1
## 3    Adams    30       1
## 4    Allen     9       2
## 5    Allen    27       2
## 6 Anderson    24       3
```

We then want to repeat the pivoting process to get Offers in rows and clusters in columns counting the total number of transactions for each cluster. Once we have our pivot table we will merge it with the offers data table like we did before:

```
# Get top deals by cluster
cluster.pivot<-melt(cluster.deals,id=c("Offer","Cluster"))
cluster.pivot<-cast(cluster.pivot,Offer~Cluster,fun.aggregate=length)
cluster.topDeals<-cbind(offers,cluster.pivot[-1])
head(cluster.topDeals)
```

```
##   OfferID Campaign           Varietal MinimumQt Discount      Origin
## 1       1  January             Malbec        72       56      France
## 2       2  January         Pinot Noir        72       17      France
## 3       3 February           Espumante       144       32      Oregon
## 4       4 February           Champagne        72       48      France
## 5       5 February Cabernet Sauvignon       144       44 New Zealand
## 6       6    March            Prosecco       144       86       Chile
##   PastPeak 1 2 3 4
## 1    FALSE 0 8 2 0
## 2    FALSE 0 3 7 0
## 3     TRUE 1 2 0 3
## 4     TRUE 0 8 0 4
## 5     TRUE 0 4 0 0
## 6    FALSE 1 5 0 6
```

```
#### And finally we can export the data in excel format with the command:
#### write.csv(file="topdeals.csv",cluster.topDeals,row.names=F)
```

Study case III: Social Network Clustering Analysis

For this analysis, we will be using a dataset (https://raw.githubusercontent.com/brenden17/sklearnlab/master/facebook/snsdata.csv) representing a random sample of 30.000 U.S. high school students who had profiles on a well-known Social Network in from 2006 to 2009.

From the top 500 words appearing across all pages, 36 words were chosen to represent five categories of interests, namely extracurricular activities, fashion, religion, romance, and antisocial behavior. The 36 words include terms such as football, sexy, kissed, bible, shopping, death, and drugs. The final dataset indicates, for each person, how many times each word appeared in the person's SNS profile.

```
teens <- read.csv("snsdata.csv")
head(teens,3)
```

```
##   gradyear gender   age friends basketball football soccer softball
## 1     2006      M 18.98       7          0        0      0        0
## 2     2006      F 18.80       0          0        1      0        0
## 3     2006      M 18.34      69          0        1      0        0
##   volleyball swimming cheerleading baseball tennis sports cute sex sexy
## 1          0        0            0        0      0      0    0   0    0
## 2          0        0            0        0      0      0    1   0    0
## 3          0        0            0        0      0      0    0   0    0
##   hot kissed dance band marching music rock god church jesus bible hair
## 1   0      0     1    0        0     0    0   0      0     0     0    0
## 2   0      0     0    0        0     2    2   1      0     0     0    6
## 3   0      0     0    2        0     1    0   0      0     0     0    0
##   dress blonde mall shopping clothes hollister abercrombie die death drunk
## 1     0      0    0        0       0         0           0   0     0     0
## 2     4      0    1        0       0         0           0   0     0     0
## 3     0      0    0        0       0         0           0   0     1     0
##   drugs
## 1     0
## 2     0
## 3     0
```

```
dim(teens)
```

```
## [1] 30000    40
```

Let's also take a quick look at the specifics of the data. The first several lines of the str() output are as follows:

```
str(teens)
```

```
## 'data.frame':    30000 obs. of  40 variables:
##  $ gradyear    : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
##  $ gender      : Factor w/ 2 levels "F","M": 2 1 2 1 NA 1 1 2 1 1 ...
##  $ age         : num  19 18.8 18.3 18.9 19 ...
##  $ friends     : int  7 0 69 0 10 142 72 17 52 39 ...
##  $ basketball  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ football    : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ soccer      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ softball    : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ volleyball  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ swimming    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cheerleading: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ baseball    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ tennis      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sports      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ cute        : int  0 1 0 1 0 0 0 0 0 1 ...
##  $ sex         : int  0 0 0 0 1 1 0 2 0 0 ...
##  $ sexy        : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ hot         : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ kissed      : int  0 0 0 0 5 0 0 0 0 0 ...
##  $ dance       : int  1 0 0 0 1 0 0 0 0 0 ...
##  $ band        : int  0 0 2 0 1 0 1 0 0 0 ...
##  $ marching    : int  0 0 0 0 0 1 1 0 0 0 ...
##  $ music       : int  0 2 1 0 3 2 0 1 0 1 ...
##  $ rock        : int  0 2 0 1 0 0 0 1 0 1 ...
##  $ god         : int  0 1 0 0 1 0 0 0 0 6 ...
##  $ church      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ jesus       : int  0 0 0 0 0 0 0 0 0 2 ...
##  $ bible       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hair        : int  0 6 0 0 1 0 0 0 0 1 ...
##  $ dress       : int  0 4 0 0 0 1 0 0 0 0 ...
##  $ blonde      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ mall        : int  0 1 0 0 0 0 2 0 0 0 ...
##  $ shopping    : int  0 0 0 0 2 1 0 0 0 1 ...
##  $ clothes     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hollister   : int  0 0 0 0 0 0 2 0 0 0 ...
##  $ abercrombie : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ die         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ death       : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ drunk       : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ drugs       : int  0 0 0 0 1 0 0 0 0 0 ...
```

As we had expected, the data include 30,000 teenagers with four variables indicating personal characteristics and 36 words indicating interests. Note that there are some NA's in the variable *gender*.

```
summary(teens$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       3      16      17      18      18     107    5086
```

We will skip all the data with missing values:

```
teens = na.omit(teens)
dim(teens)
```

```
## [1] 24005    40
```

We'll start our cluster analysis by considering only the 36 features that represent the number of times various interests appeared on the SNS profiles of teens. For convenience, let's make a data frame containing only these features:

```
interests <- teens[5:40]
```

To apply z-score standardization to the interests data frame, we can use the scale() function with lapply(), as follows:

```
interests_z <- as.data.frame(lapply(interests, scale))
```

To divide teens into five clusters, we can use the following command:

```
teen_clusters <- kmeans(interests_z, 5)
```

number of examples falling in each of the groups. If the groups are too large or too small, then they are not likely to be very useful. To obtain the size of the kmeans() clusters, use the teen_clusters$size component as follows:

```
teen_clusters$size
```

```
## [1]   403 17255  4783   717   847
```

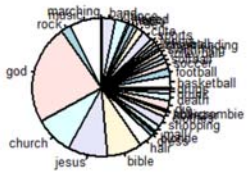
For a more in-depth look at the clusters, we can examine the coordinates of the cluster centroids using the teen_clusters$centers component, which is as follows for the first eight features:

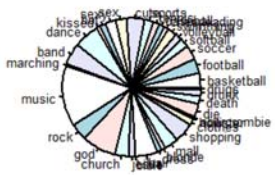
```
teen_clusters$centers
```

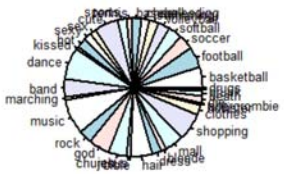
```
##   basketball  football   soccer softball volleyball swimming cheerleading
## 1     0.1510 -0.004452  0.01377 -0.03832   0.004122  0.03706     0.001678
## 2    -0.1636 -0.171249 -0.09280 -0.11707  -0.116837 -0.09624    -0.116317
## 3     0.4979  0.521693  0.29185  0.38495   0.378986  0.27075     0.336997
## 4     0.1605  0.249815  0.12107  0.04462   0.200136  0.21498     0.380099
## 5     0.3143  0.333318  0.13348  0.19162   0.068687  0.23196     0.144012
##    baseball   tennis   sports     cute       sex     sexy      hot
## 1  0.029672  0.04294  0.01238  0.02468  0.026179 -0.04248  0.07039
## 2 -0.109056 -0.05172 -0.12781 -0.18574 -0.096249 -0.08776 -0.13528
## 3  0.344627  0.14798  0.31350  0.52770 -0.008816  0.21064  0.36860
## 4  0.008058  0.09947  0.08630  0.40237  0.015712  0.13078  0.41142
## 5  0.254633  0.11345  0.75449  0.45156  1.984811  0.50795  0.29264
##     kissed    dance     band marching   music     rock     god   church
## 1 -0.02317 -0.01394  0.14685  0.08007  0.2349  0.12825  2.2406  1.24023
## 2 -0.13438 -0.16607 -0.09301 -0.05726 -0.1539 -0.12691 -0.1063 -0.14412
## 3 -0.04044  0.49912  0.25190  0.19532  0.3165  0.23156  0.1348  0.39146
## 4  0.03950  0.20828 -0.10137 -0.09403  0.1072  0.05519 -0.0184 -0.02148
## 5  2.94352  0.39499  0.48827  0.10495  1.1448  1.17016  0.3546  0.15357
##       jesus     bible     hair    dress    blonde     mall shopping
## 1  2.332629  6.048477  0.05815  0.02987 -0.003915 -0.06875 -0.01053
## 2 -0.074454 -0.109423 -0.20514 -0.15210 -0.027665 -0.18923 -0.23212
## 3  0.059948 -0.105090  0.22836  0.43163  0.028245  0.49390  0.68176
## 4  0.006561 -0.073294  0.41466  0.12938  0.058491  0.63797  0.76795
## 5  0.062839  0.006795  2.51095  0.53741  0.356443  0.55865  0.23374
##    clothes hollister abercrombie      die    death     drunk    drugs
## 1  0.04932  -0.09458    -0.08999  0.22400  0.28806  0.065740  0.08217
## 2 -0.19026  -0.15715    -0.15109 -0.09936 -0.08237 -0.088794 -0.11443
## 3  0.38665  -0.05650    -0.07429  0.02297  0.09233 -0.008421 -0.07876
## 4  0.54243   4.06769     3.90321  0.04736  0.08796  0.037217  0.02999
## 5  1.20992   0.12212     0.23620  1.74789  0.94516  1.793662  2.71150
```

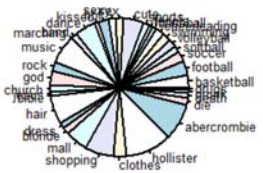
The cluster characterization can be obtained with *pie* charts:

```
par(mfrow=c(2,2))
pie(colSums(interests[teen_clusters$cluster==1,]),cex=0.5)

pie(colSums(interests[teen_clusters$cluster==2,]),cex=0.5)

pie(colSums(interests[teen_clusters$cluster==3,]),cex=0.5)

pie(colSums(interests[teen_clusters$cluster==4,]),cex=0.5)
```









Can you give a name to every cluster? Why this clustering information it could be useful?