KMD

Kritika Bhowmik

6/1/2019

First we would load all the necessary libraries

#If any errors are thrown while loading libraries please install the packages commented below  
#install.packages("tidyverse")  
#install.packages("caret")  
#install.packages("rpart.plot")  
#install.packages("e1071")  
library(tidyverse) # data manipulation

## Registered S3 methods overwritten by 'ggplot2':  
## method from   
## [.quosures rlang  
## c.quosures rlang  
## print.quosures rlang

## Registered S3 method overwritten by 'rvest':  
## method from  
## read\_xml.response xml2

## -- Attaching packages --------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1 v purrr 0.3.2   
## v tibble 2.1.1 v dplyr 0.8.0.1  
## v tidyr 0.8.3 v stringr 1.4.0   
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(cluster) # clustering algorithms  
library(factoextra) # clustering visualization

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ

library(dendextend) # for comparing two dendrograms

##   
## ---------------------  
## Welcome to dendextend version 1.12.0  
## Type citation('dendextend') for how to cite the package.  
##   
## Type browseVignettes(package = 'dendextend') for the package vignette.  
## The github page is: https://github.com/talgalili/dendextend/  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues  
## Or contact: <tal.galili@gmail.com>  
##   
## To suppress this message use: suppressPackageStartupMessages(library(dendextend))  
## ---------------------

##   
## Attaching package: 'dendextend'

## The following object is masked from 'package:stats':  
##   
## cutree

library(r2d3)

## r2d3 should be run under RStudio v1.2 or higher. Please update at:  
## https://www.rstudio.com/rstudio/download/preview/

We now need to preprocess the data that we will use for creating hierarchical clusters

#Data preprocessing  
df <- read\_rds('C:\\Users\\Kritika\\Documents\\SummerSem2019\\KMD Project\\KMD\_Tinnitus\\190426\_charite\_tinnitus.rds') %>%  
 arrange(.testdatum) %>%  
 group\_by(.jour\_nr) %>%  
 slice(1) %>%  
 ungroup() %>%  
 filter(.phase == "A") %>%  
 mutate(phqk\_paniksyndrom = if\_else(phqk\_phqk\_2a +  
 phqk\_phqk\_2b +  
 phqk\_phqk\_2c +  
 phqk\_phqk\_2d +  
 phqk\_phqk\_2e == 5, 1, 0)) %>%  
 select(.jour\_nr,  
 .age,  
 acsa\_acsa,  
 adsl\_adsl\_sum,  
 bi\_erschoepfung, bi\_magen, bi\_glieder, bi\_herz, bi\_beschwerden,  
 bsf\_geh, bsf\_eng, bsf\_aerg, bsf\_an\_de, bsf\_mued, bsf\_tnl,  
 isr\_deprsyn, isr\_angstsyn, isr\_zwasyn, isr\_somasyn, isr\_essstsyn,  
 isr\_zusatz, isr\_isr\_ges,  
 phqk\_depressivitaet, phqk\_paniksyndrom,  
 psq\_anford, psq\_anspan, psq\_freude, psq\_sorgen, psq\_psq\_sum,  
 schmerzskal\_beein10, schmerzskal\_haeuf10, schmerzskal\_staerke10,  
 ses\_ses\_affektiv, ses\_ses\_sensorisch,  
 sf8\_bp\_sf36ks, sf8\_gh\_sf36ag, sf8\_mcs8, sf8\_mh\_sf36pw, sf8\_pcs8,  
 sf8\_pf\_sf36kf, sf8\_re\_sf36er, sf8\_rp\_sf36kr, sf8\_sf\_sf36sf, sf8\_vt\_sf36vit,  
 sozk\_soz01\_male, sozk\_soz02\_german, sozk\_soz05\_partner, sozk\_soz06\_married,  
 sozk\_soz09\_abitur, sozk\_soz10\_keinAbschl, sozk\_soz11\_job, sozk\_soz18\_selbstst,   
 sozk\_soz1920\_krank, sozk\_soz21\_tindauer, sozk\_soz2224\_psycho, sozk\_soz25\_numdoc,  
 swop\_sw, swop\_opt, swop\_pes,  
 tq\_aku, tq\_co, tq\_em, tq\_inti, tq\_pb, tq\_sl, tq\_som, tq\_tf,  
 tinskal\_beein10, tinskal\_haeuf10, tinskal\_laut10,  
 starts\_with("tlq"), -tlq\_timestamp  
 ) %>%  
 drop\_na()  
 df\_scaled<-df%>%  
 scale() #for scaling/normalising the data

Hierarchical clustering can be done in two ways : 1) Top Down : Start with the entire data as a cluster and break them down until each point is a cluster in itself. 2) Bottom up : Start from individual points as clusters and merge them based on similarities until one cluster if formed.

There are different ways to calculate the distance/ similarity between individual points and in between the clusters which can be a very important factor in deciding the final cluster results.

So we first create a list of these methods and then create clusters for each combination.

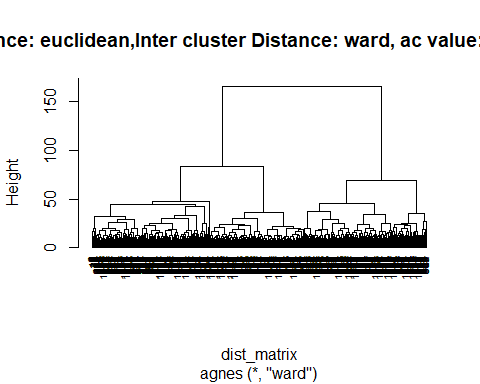
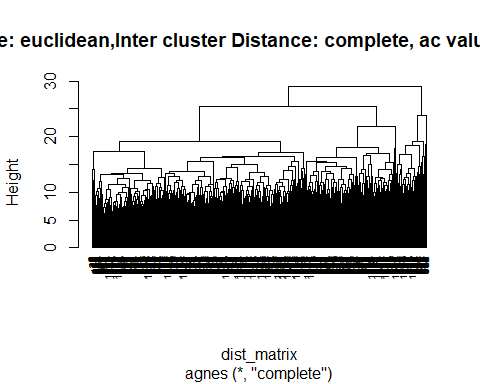
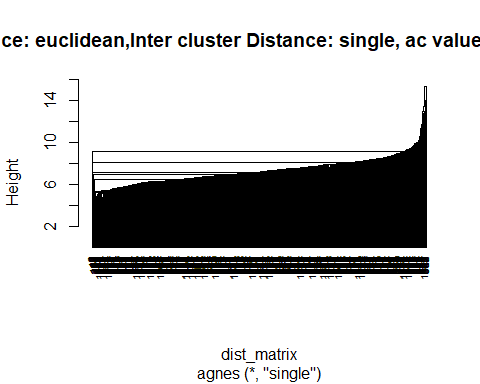
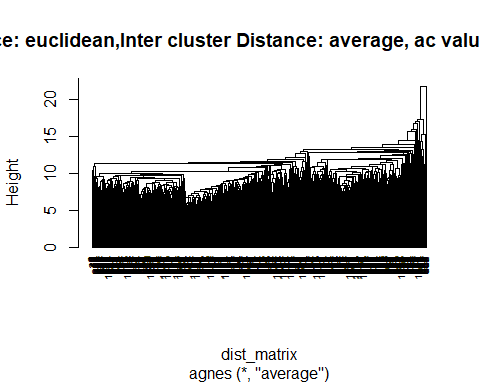
# methods to assess inter cluster similarity  
m <- c( "average", "single", "complete", "ward")  
names(m) <- c( "average", "single", "complete", "ward")  
  
#distance measures  
dist\_methods <- c("euclidean","manhattan","maximum", "canberra","binary","minkowski")  
names(dist\_methods)<- c("euclidean","manhattan","maximum", "canberra","binary","minkowski")

Now we can calculate the clusters. The ac value of each cluster gives the quality/ tightness of each cluster. The closer the value is to 1 the better are the results.

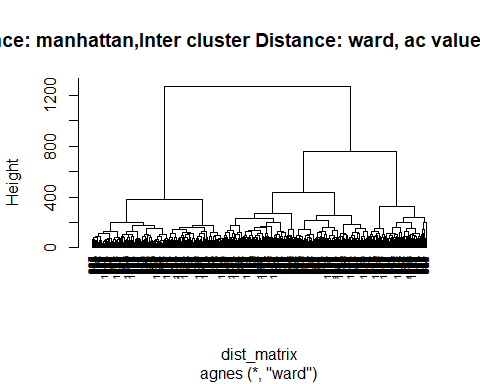
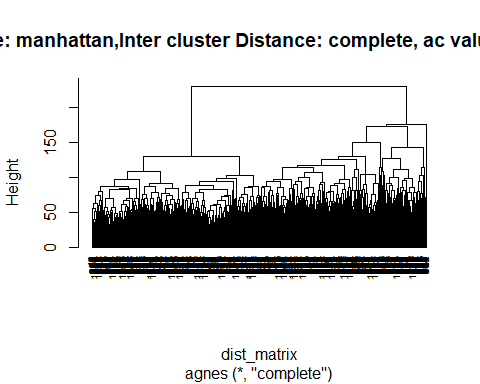
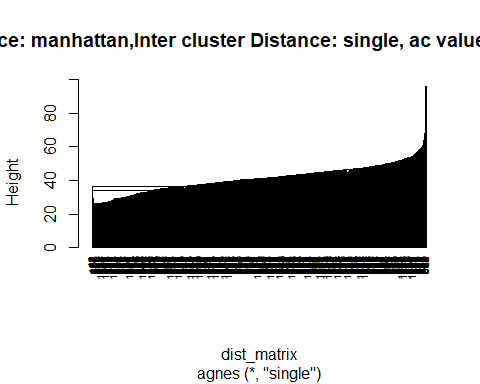
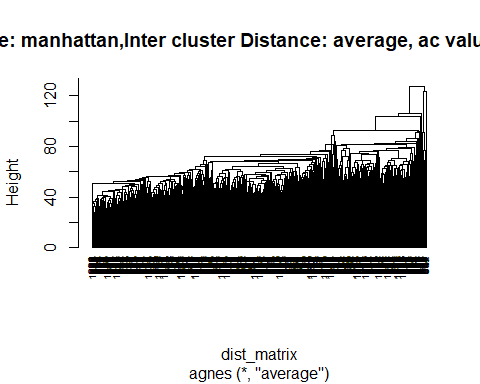
\*\*the agglomerative coefficient, measuring the clustering structure of the dataset.

For each observation i, denote by m(i) its dissimilarity to the first cluster it is merged with, divided by the dissimilarity of the merger in the final step of the algorithm. The ac is the average of all 1 - m(i). It can also be seen as the average width (or the percentage filled) of the banner plot. Because ac grows with the number of observations, this measure should not be used to compare datasets of very different sizes.\*\*

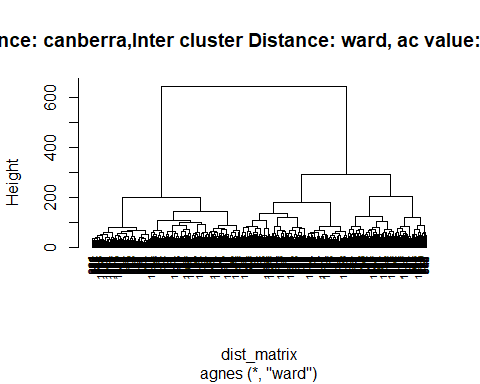
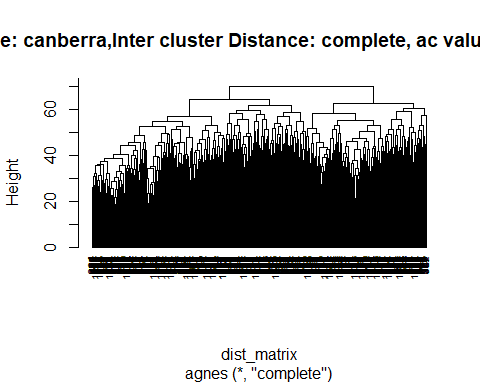
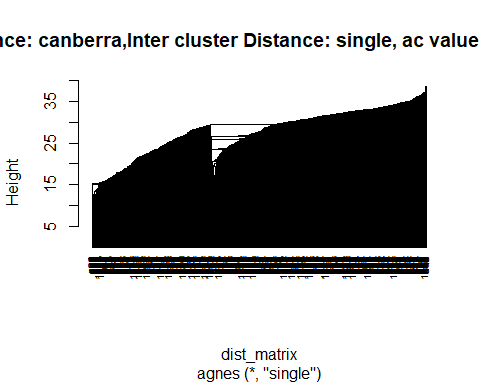
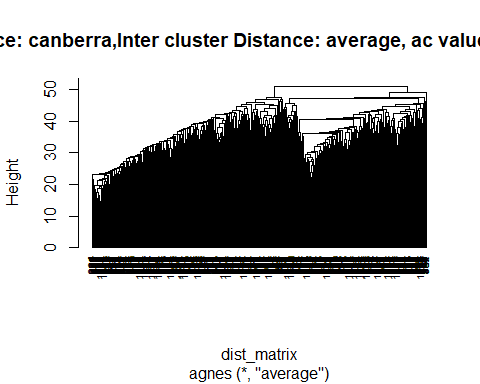
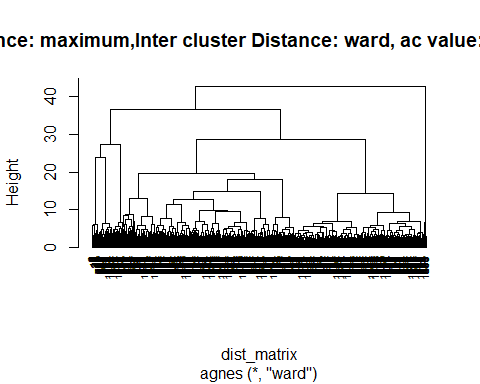
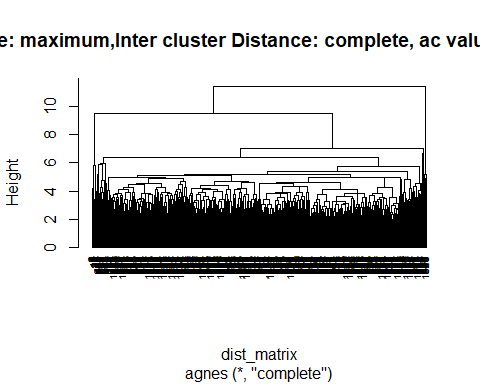
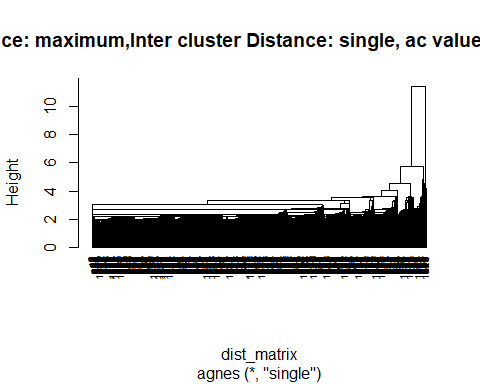
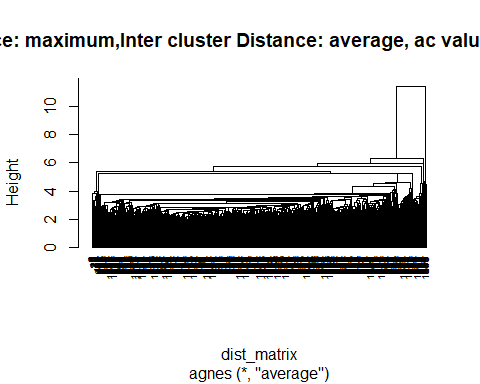
# function to compute coefficient  
final\_dist\_m<-list()  
final\_m<-list()  
for(dist\_m in dist\_methods)  
{  
 dist\_matrix<-dist(df\_scaled, method = dist\_m)  
 for(x in m)  
 {  
 hc <-agnes(dist\_matrix, method = x)  
 ac<-hc$ac  
 title<-sprintf("Distance: %s,Inter cluster Distance: %s, ac value: %f",dist\_m,x,ac)  
 pltree(hc, cex = 0.6, hang = -1,main = title)  
 if(!is.nan(ac) && ac>0.95)  
 {  
 print(title)  
 final\_dist\_m<-c(final\_dist\_m,dist\_m)  
 final\_m<-c(final\_m,x)  
 }  
 }  
}



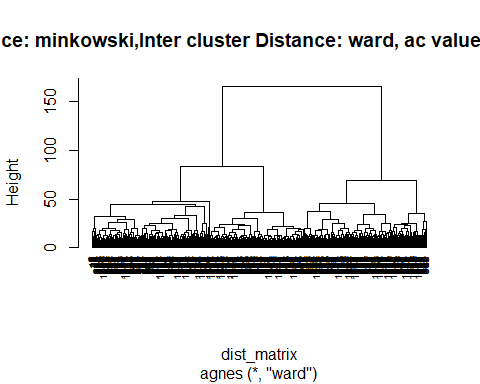
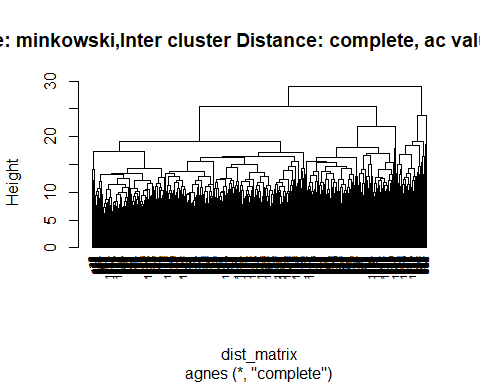
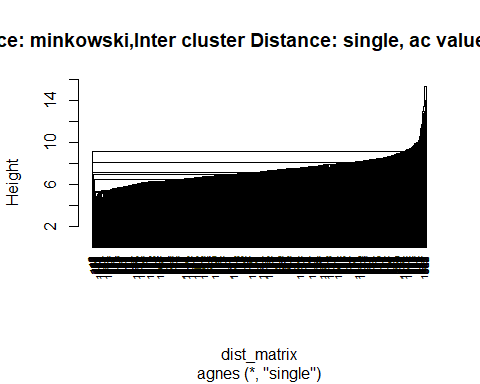
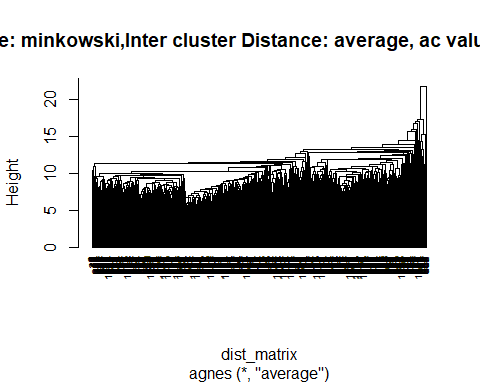
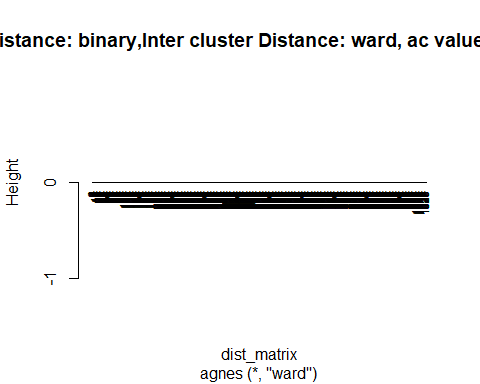
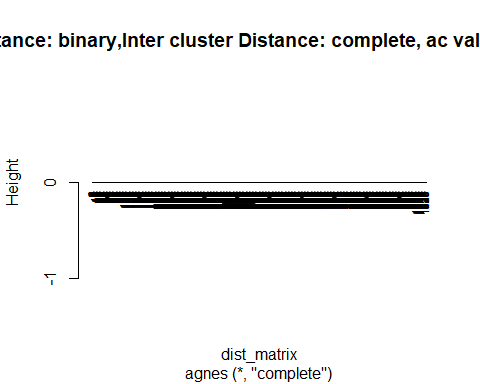
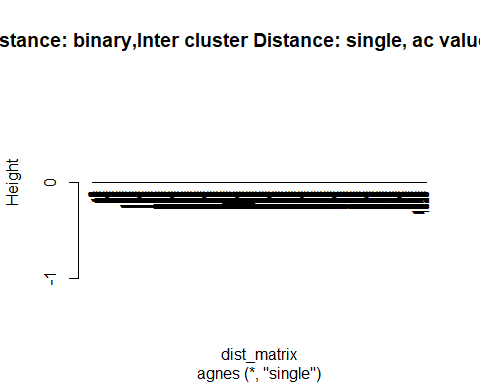
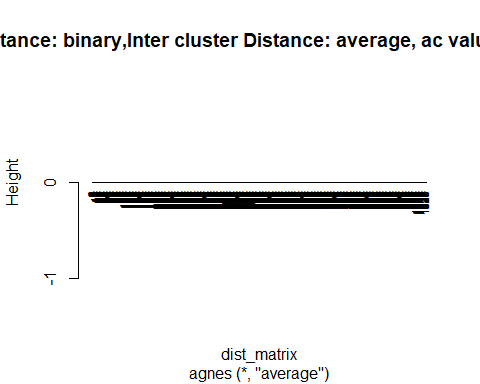
## [1] "Distance: euclidean,Inter cluster Distance: ward, ac value: 0.954031"



## [1] "Distance: manhattan,Inter cluster Distance: ward, ac value: 0.965808"



## [1] "Distance: canberra,Inter cluster Distance: ward, ac value: 0.954687"



## [1] "Distance: minkowski,Inter cluster Distance: ward, ac value: 0.954031"

Now we can choose the desired combination, for example: Euclidean and Ward method and then we can cut the tree to get the final clusters.

dist\_matrix<-dist(df\_scaled, method = "euclidean")  
hc <-agnes(dist\_matrix, method = "ward")  
sub\_group<-cutree(hc,k=4)  
table(sub\_group)

## sub\_group  
## 1 2 3 4   
## 365 270 287 108

We can see above the number of records in each cluster. Let us now label all our records with the respective cluster number.

df\_labeled<-df%>%  
 mutate(label = sub\_group)  
df\_labeled%>%head

## # A tibble: 6 x 79  
## .jour\_nr .age acsa\_acsa adsl\_adsl\_sum bi\_erschoepfung bi\_magen  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 11117 44 5.1 2 6 0  
## 2 11657 55 5.6 10 2 2  
## 3 12194 68 7 41 5 1  
## 4 22407 75 1 44 10 5  
## 5 22541 60 7.7 6 4 0  
## 6 23418 64 2.2 21 8 4  
## # ... with 73 more variables: bi\_glieder <dbl>, bi\_herz <dbl>,  
## # bi\_beschwerden <dbl>, bsf\_geh <dbl>, bsf\_eng <dbl>, bsf\_aerg <dbl>,  
## # bsf\_an\_de <dbl>, bsf\_mued <dbl>, bsf\_tnl <dbl>, isr\_deprsyn <dbl>,  
## # isr\_angstsyn <dbl>, isr\_zwasyn <dbl>, isr\_somasyn <dbl>,  
## # isr\_essstsyn <dbl>, isr\_zusatz <dbl>, isr\_isr\_ges <dbl>,  
## # phqk\_depressivitaet <dbl>, phqk\_paniksyndrom <dbl>, psq\_anford <dbl>,  
## # psq\_anspan <dbl>, psq\_freude <dbl>, psq\_sorgen <dbl>,  
## # psq\_psq\_sum <dbl>, schmerzskal\_beein10 <dbl>,  
## # schmerzskal\_haeuf10 <dbl>, schmerzskal\_staerke10 <dbl>,  
## # ses\_ses\_affektiv <dbl>, ses\_ses\_sensorisch <dbl>, sf8\_bp\_sf36ks <dbl>,  
## # sf8\_gh\_sf36ag <dbl>, sf8\_mcs8 <dbl>, sf8\_mh\_sf36pw <dbl>,  
## # sf8\_pcs8 <dbl>, sf8\_pf\_sf36kf <dbl>, sf8\_re\_sf36er <dbl>,  
## # sf8\_rp\_sf36kr <dbl>, sf8\_sf\_sf36sf <dbl>, sf8\_vt\_sf36vit <dbl>,  
## # sozk\_soz01\_male <int>, sozk\_soz02\_german <int>,  
## # sozk\_soz05\_partner <int>, sozk\_soz06\_married <int>,  
## # sozk\_soz09\_abitur <int>, sozk\_soz10\_keinAbschl <int>,  
## # sozk\_soz11\_job <int>, sozk\_soz18\_selbstst <dbl>,  
## # sozk\_soz1920\_krank <dbl>, sozk\_soz21\_tindauer <dbl>,  
## # sozk\_soz2224\_psycho <dbl>, sozk\_soz25\_numdoc <dbl>, swop\_sw <dbl>,  
## # swop\_opt <dbl>, swop\_pes <dbl>, tq\_aku <dbl>, tq\_co <dbl>,  
## # tq\_em <dbl>, tq\_inti <dbl>, tq\_pb <dbl>, tq\_sl <dbl>, tq\_som <dbl>,  
## # tq\_tf <dbl>, tinskal\_beein10 <dbl>, tinskal\_haeuf10 <dbl>,  
## # tinskal\_laut10 <dbl>, tlq\_tlq01\_1 <int>, tlq\_tlq01\_2 <int>,  
## # tlq\_tlq01\_3 <int>, tlq\_tlq01\_4 <int>, tlq\_tlq02\_1 <int>,  
## # tlq\_tlq02\_2 <int>, tlq\_tlq02\_3 <int>, tlq\_tlq02\_4 <int>, label <int>

Grouping the data based on the assigned labels and splitting into one data frame for each group.(extra step)

df\_grouped<-df\_labeled%>%  
 group\_by(label)  
df\_split<-group\_split(df\_grouped)

The .journ\_nr column is like an identifier for each record and is not significant for the cluster discription. So dropping that column. and then getting a summary of each cluster(group) as “cluster features” Also calculating the general population summary for each feature as “feature summary”.

cluster\_features<-data.frame(df\_labeled%>%  
 select(-.jour\_nr)%>%  
 group\_by(label)%>%  
 summarise\_each(mean))  
feature\_summary<-data.frame(df%>%  
 select(-.jour\_nr)%>%  
 summarise\_each(mean))

Finally creating a data frame which can be used for the visualization. The difference between “feature\_summary” and “cluster\_features” will tell us how different the feature of a cluster is from the mean of the population. Which is finally plotted in the graph.

options(scipen = 999)  
  
f<-function(cluster)  
 { cluster\_plot\_data<-rbind(feature\_summary,cluster[-1])  
 cluster\_name<-paste("cluster\_",cluster[1])  
 rownames(cluster\_plot\_data)<-c("general\_mean",cluster\_name)  
 cluster\_plot\_data<-t(cluster\_plot\_data)%>%data.frame()  
 cluster\_plot\_data<-mutate(cluster\_plot\_data,mean\_difference = cluster\_plot\_data[,1]-cluster\_plot\_data[,2],feature=rownames(cluster\_plot\_data))  
 r2d3(data = cluster\_plot\_data, script = "cluster\_chart.js",viewer ="internal")  
}  
apply(cluster\_features,1,f)

## [[1]]  
##   
## [[2]]  
##   
## [[3]]  
##   
## [[4]]

Now we have plotted all the feature values for each cluster. However for each cluster there are only a few features which clearly distinguish it from other clusters. To find out which features and their respective values are responsible to identify a cluster we can use supervised techniques. Let us now use a Decision Tree algorithm to identify which features are the most important to lead us to get our cluster labels as leaf nodes.

Now to get the decision tree as a classifier intead of a regressor we need to make the predicting variable as non numeric.

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart.plot)

## Loading required package: rpart

##   
## Attaching package: 'rpart'

## The following object is masked from 'package:dendextend':  
##   
## prune

library(e1071)  
df\_string\_labeled<-df\_labeled  
df\_string\_labeled$label<-sub("^","Type ",df\_labeled$label) #Adding "Type" to each cluster number to make it #non numeric  
trctrl <- trainControl(method = "boot", number = 10)  
dtree\_fit <- train(label ~., data = df\_string\_labeled, method = "rpart",  
 parms = list(split = "information"),  
 trControl=trctrl,  
 tuneLength = 10)  
  
prp(dtree\_fit$finalModel, box.palette = "Reds", tweak = 1.2)

