HW3 Multi

Benjan

11/24/2022

# Problem 1.

Lets create the correlation matrix first.

r <- matrix(c(1.00,0.44,0.41,0.29,0.33,0.25,  
 0.44,1.00,0.35,0.35,0.32,0.33,  
 0.41,0.35,1.00,0.16,0.19,0.18,  
 0.29,0.35,0.16,1.00,0.59,0.47,  
 0.33,0.32,0.19,0.59,1.00,0.46,  
 0.25,0.33,0.18,0.47,0.46,1.00), nrow = 6, byrow = TRUE)  
row.names(r) <- c('French','English','History','Arithmetic','Algebra','Geometry')  
colnames(r) <- c('French','English','History','Arithmetic','Algebra','Geometry')

## a.

Perfoming EFA on the matrix:

efa <- factanal(covmat = r, factors = 2)  
print(efa$loadings,cut=0.5)

##   
## Loadings:  
## Factor1 Factor2  
## French 0.661   
## English 0.551   
## History 0.591   
## Arithmetic 0.770   
## Algebra 0.715   
## Geometry 0.570   
##   
## Factor1 Factor2  
## SS loadings 1.593 1.215  
## Proportion Var 0.265 0.202  
## Cumulative Var 0.265 0.468

We can see that the Math courses are related to each other and remaining variables form another group.

## b.

Now, let perform EFA for 3 factors.

efa\_3 <- factanal(covmat = r, factors = 3)  
print(efa\_3$loadings,cut=0.5)

##   
## Loadings:  
## Factor1 Factor2 Factor3  
## French 0.706   
## English 0.538   
## History 0.554   
## Arithmetic 0.741   
## Algebra 0.756   
## Geometry 0.567   
##   
## Factor1 Factor2 Factor3  
## SS loadings 1.588 1.207 0.181  
## Proportion Var 0.265 0.201 0.030  
## Cumulative Var 0.265 0.466 0.496

The first factor is Math and the second factor is language and other non-Math courses.

## c.

‘Varimax rotation’ which is a orthogonal rotation is used as default. Varimax rotation simplifies the interpretation from factor analysis. It gives us extreme values, few very large and many near-zero and so, few factors are needed to explain each variable. There is less overlap between factors and it is easy to separate them.

## d.

Lets do the part without rotation.

efa\_nr <- factanal(covmat = r, factors = 2,rotation='none')  
print(efa\_nr$loadings)

##   
## Loadings:  
## Factor1 Factor2  
## French 0.558 0.425   
## English 0.569 0.286   
## History 0.392 0.450   
## Arithmetic 0.738 -0.279   
## Algebra 0.718 -0.209   
## Geometry 0.595 -0.133   
##   
## Factor1 Factor2  
## SS loadings 2.204 0.603  
## Proportion Var 0.367 0.101  
## Cumulative Var 0.367 0.468

Because we did not apply varimax rotation, all the greater loading values are in factor 1. It is difficult to explain the relationship among factors as the separations are not clear.

# Problem 2.

Lets read the data first.

genes <- read.csv("https://raw.githubusercontent.com/EricBrownTTU/ISQS6350/main/geneexpression.csv", header = FALSE)  
head(genes)

## V1 V2 V3 V4 V5 V6  
## 1 -0.96193340 0.4418028 -0.9750051 1.4175040 0.8188148 0.3162937  
## 2 -0.29252570 -1.1392670 0.1958370 -1.2811210 -0.2514393 2.5119970  
## 3 0.25878820 -0.9728448 0.5884858 -0.8002581 -1.8203980 -2.0589240  
## 4 -1.15213200 -2.2131680 -0.8615249 0.6309253 0.9517719 -1.1657240  
## 5 0.19578280 0.5933059 0.2829921 0.2471472 1.9786680 -0.8710180  
## 6 0.03012394 -0.6910143 -0.4034258 -0.7298590 -0.3640986 1.1253490  
## V7 V8 V9 V10 V11 V12  
## 1 -0.02496682 -0.06396600 0.03149702 -0.3503106 -0.7227299 -0.2819547  
## 2 -0.92220620 0.05954277 -1.40964500 -0.6567122 -0.1157652 0.8259783  
## 3 -0.06476437 1.59212400 -0.17311700 -0.1210874 -0.1875790 -1.5001630  
## 4 -0.39155860 1.06361900 -0.35000900 -1.4890580 -0.2432189 -0.4330340  
## 5 -0.98971500 -1.03225300 -1.10965400 -0.3851423 1.6509570 -1.7449090  
## 6 -1.40404100 -0.80613040 -1.23792400 0.5776018 -0.2720642 2.1765620  
## V13 V14 V15 V16 V17 V18  
## 1 1.33751500 0.70197980 1.0076160 -0.4653828 0.6385951 0.2867807  
## 2 0.34644960 -0.56954860 -0.1315365 0.6902290 -0.9090382 1.3026420  
## 3 -1.22873700 0.85598900 1.2498550 -0.8980815 0.8702058 -0.2252529  
## 4 -0.03879128 -0.05789677 -1.3977620 -0.1561871 -2.7359820 0.7756169  
## 5 -0.37888530 -0.67982610 -2.1315840 -0.2301718 0.4661243 -1.8004490  
## 6 1.43640700 -1.02578100 0.2981582 -0.5559659 0.2046529 -1.1916480  
## V19 V20 V21 V22 V23 V24  
## 1 -0.2270782 -0.22004520 -1.2425730 -0.1085056 -1.8642620 -0.5005122  
## 2 -1.6726950 -0.52550400 0.7979700 -0.6897930 0.8995305 0.4285812  
## 3 0.4502892 0.55144040 0.1462943 0.1297400 1.3042290 -1.6619080  
## 4 0.6141562 2.01919400 1.0811390 -1.0766180 -0.2434181 0.5134822  
## 5 0.6262904 -0.09772305 -0.2997108 -0.5295591 -2.0235670 -0.5108402  
## 6 0.2350916 0.67096470 0.1307988 1.0689940 1.2309870 1.1344690  
## V25 V26 V27 V28 V29 V30  
## 1 -1.32500800 1.06341100 -0.2963712 -0.1216457 0.08516605 0.62417640  
## 2 -0.67611410 -0.53409490 -1.7325070 -1.6034470 -1.08362000 0.03342185  
## 3 -1.63037600 -0.07742528 1.3061820 0.7926002 1.55946500 -0.68851160  
## 4 -0.51285780 2.55167600 -2.3143010 -1.2764700 -1.22927100 1.43439600  
## 5 0.04600274 1.26803000 -0.7439868 0.2231319 0.85846280 0.27472610  
## 6 0.55636800 -0.35876640 1.0798650 -0.2064905 -0.00616453 0.16425470  
## V31 V32 V33 V34 V35 V36  
## 1 -0.5095915 -0.216725500 -0.05550597 -0.4844491 -0.5215811 1.9491350  
## 2 1.7007080 0.007289556 0.09906234 0.5638533 -0.2572752 -0.5817805  
## 3 -0.6154720 0.009999363 0.94581000 -0.3185212 -0.1178895 0.6213662  
## 4 -0.2842774 0.198945600 -0.09183320 0.3496279 -0.2989097 1.5136960  
## 5 -0.6929984 -0.845707200 -0.17749680 -0.1664908 1.4831550 -1.6879460  
## 6 1.1567370 0.241774500 0.08863952 0.1829540 0.9426771 -0.2096004  
## V37 V38 V39 V40  
## 1 1.32433500 0.4681471 1.06110000 1.6559700  
## 2 -0.16988710 -0.5423036 0.31293890 -1.2843770  
## 3 -0.07076396 0.4016818 -0.01622713 -0.5265532  
## 4 0.67118470 0.0108553 -1.04368900 1.6252750  
## 5 -0.14142960 0.2007785 -0.67594210 2.2206110  
## 6 0.53626210 -1.1852260 -0.42274760 0.6243603

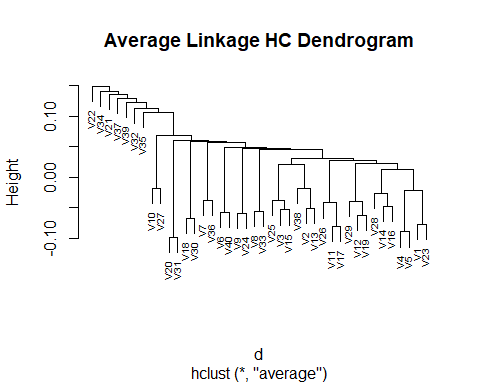
## a.

The correlation matrix and the distance matrix of it are:

genes\_r <- cor(genes)  
d <- as.dist(genes\_r)

Lets conduct average linkage clustering.

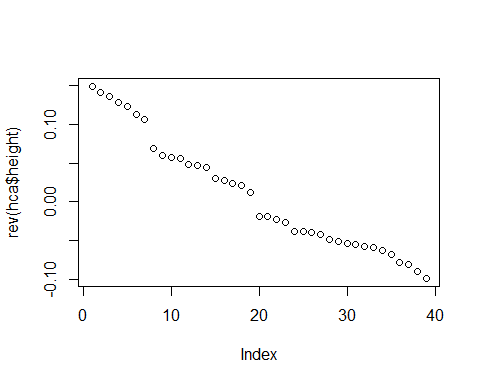
hca <- hclust(d, "average")  
plot(hca, main = "Average Linkage HC Dendrogram",cex=0.6)



## b.

From the dendogram, it looks like there are two clusters.Lets look at the scree plot.

plot(rev(hca$height))



From the scree plot , theres seems to be three clusters rather than two. Lets force them into two groups and see if we get 20 in each.

cta <- cutree(hca, 2)  
table(cta)

## cta  
## 1 2   
## 39 1

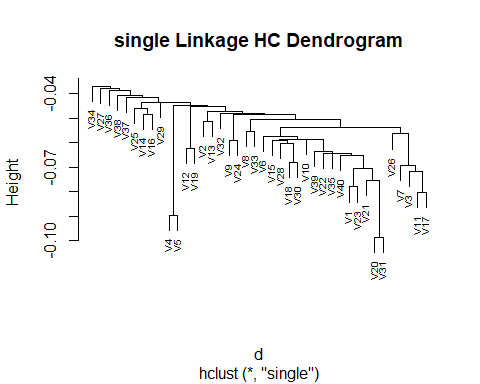
Almost all of them fall in the same group. We can see that average clustering does not work.

## c.

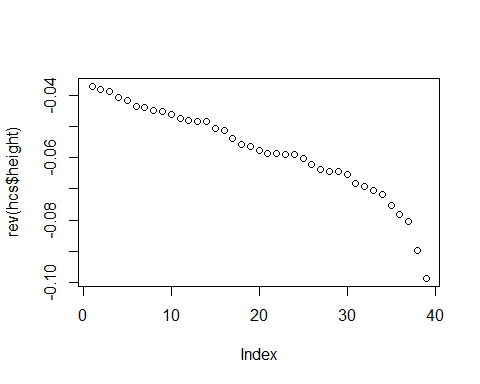
Now, lets try single and complete clustering.

**Single**

hcs <- hclust(d, "single")  
plot(hcs, main = "single Linkage HC Dendrogram",cex=0.6)



plot(rev(hcs$height))



It looks like almost all of them fall in one cluster and only a few in another. Lets put them in two clusters and check.

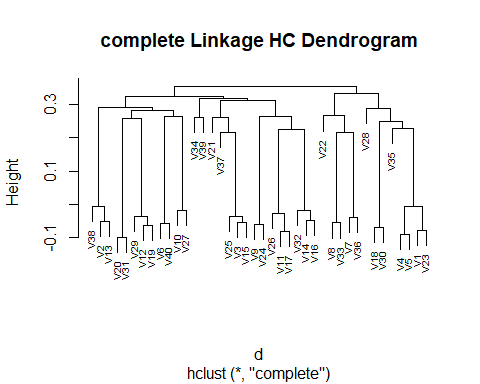
cts <- cutree(hcs, 2)  
table(cts)

## cts  
## 1 2   
## 39 1

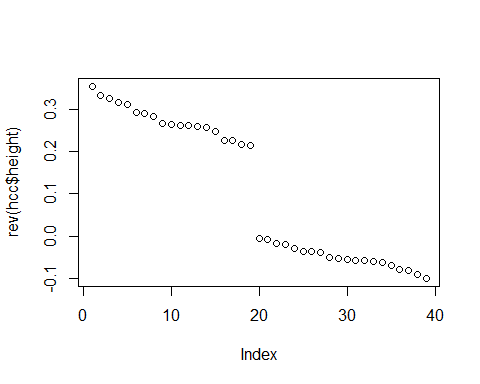
Single clustering fails too.

**Complete**

hcc <- hclust(d, "complete")  
plot(hcc, main = "complete Linkage HC Dendrogram",cex=0.6)



plot(rev(hcc$height))



There seems to be two clusters. Lets put them in two clusters and check.

ctc <- cutree(hcc, 2)  
table(ctc)

## ctc  
## 1 2   
## 13 27

We can see that there are two clusters and although they are not equally divided, complete clustering seems to work the best.

## d.

For this question , we will use complete clustering. Lets check each groups.

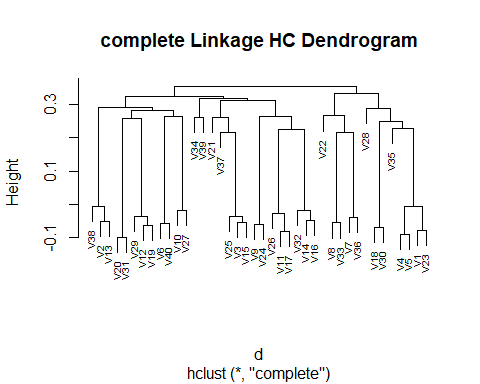
c1 <- subset(colnames(genes), ctc == 1)  
c1

## [1] "V1" "V4" "V5" "V7" "V8" "V18" "V22" "V23" "V28" "V30" "V33" "V35"  
## [13] "V36"

c2 <- subset(colnames(genes),ctc==2)

We will use this information and combine it with the dendogram to visualize the groups.

plot(hcc, main = "complete Linkage HC Dendrogram",cex=0.6)



As we can see, v22 is the lefmost point of the first group and anything to its right is group1. v22 is closer to group2 than any other points in group1 , so the gene v22 differs the most in group 1. Similarly, in group2 v34 and v39 are among the last to join the group as seen in the dendogram and thus these genes differ the most in group2.

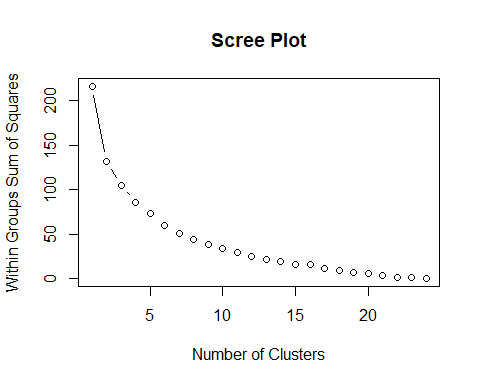
# Problem 3.

Lets read and scale the data.

protein <- read.csv("https://raw.githubusercontent.com/EricBrownTTU/ISQS6350/main/protein.csv",  
row.names = "Country")  
protein <- protein[,1:9]  
protein\_s <- scale(protein)

Now, we will create the scree plot.

plot.wgss <- function(mydata, maxc){  
wss <- numeric(maxc)  
for (i in 1:maxc){  
wss[i] <- kmeans(mydata, iter.max = 100,  
centers = i, nstart = 10)$tot.withinss  
}  
plot(1:maxc, wss, type = "b",  
xlab = "Number of Clusters",  
ylab = "Within Groups Sum of Squares",  
main = "Scree Plot")  
}  
  
plot.wgss(protein\_s,24)



I think 4 will be a good choice for number of clusters as the rate of decrease of wgss is significant until there.

# b.

Lets do the kmeans clustering with 4 clusters.

km <- kmeans(protein\_s, centers = 4, nstart = 10)

Lets see how many observations are in each cluster.

table(km$cluster)

##   
## 1 2 3 4   
## 11 2 8 4

Lets see which countries belong in each cluster.

km$cluster

## Albania Austria Belgium Bulgaria Czechoslovakia   
## 3 1 1 3 1   
## Denmark East\_Germany Finland France Greece   
## 4 1 4 1 3   
## Hungary Ireland Italy The\_Netherlands Norway   
## 3 1 3 1 4   
## Poland Portugal Romania Spain Sweden   
## 1 2 3 2 4   
## Switzerland United\_Kingdom USSR West\_Germany Yugoslavia   
## 1 1 3 1 3

The WGSS is:

km$tot.withinss

## [1] 86.13988

## c.

The means of variables for each cluster is:

round(km$centers,2)

## Red\_Meat White\_Meat Eggs Milk Fish Cereals Starchy\_Foods  
## 1 0.62 0.77 0.66 0.31 -0.26 -0.52 0.44  
## 2 -0.97 -1.18 -0.97 -1.45 1.79 -0.39 0.99  
## 3 -0.64 -0.68 -0.86 -0.73 -0.69 1.24 -0.97  
## 4 0.06 -0.18 0.38 1.33 1.22 -0.87 0.24  
## Pulses\_nuts\_oilseeds Fruits\_Vegetables  
## 1 -0.62 0.13  
## 2 1.20 1.72  
## 3 1.01 -0.04  
## 4 -0.91 -1.15

Cluster 2 contains with highest consumption of proteins. Cluster 3 is countries that depend on fish for majority of their protein supply . Cluster 4 is countries with lowest protein consumption and depend on carbs like cereals for their calorie supply. Cluster 1 is countries that heavily depend on fish and dairy for their food supply.

## d.

Lets install the library first.

library(mclust)

## Package 'mclust' version 5.4.10  
## Type 'citation("mclust")' for citing this R package in publications.

Now, we will do the model based clustering.

mbc.p <- Mclust(protein\_s)  
table(mbc.p$classification)

##   
## 1 2 3   
## 10 3 12

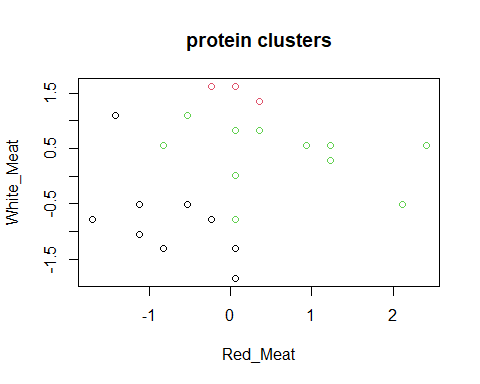
The model choose 3 clusters. The countries in each group are:

mbc.p$classification

## Albania Austria Belgium Bulgaria Czechoslovakia   
## 1 2 3 1 3   
## Denmark East\_Germany Finland France Greece   
## 3 3 3 3 1   
## Hungary Ireland Italy The\_Netherlands Norway   
## 1 3 1 2 3   
## Poland Portugal Romania Spain Sweden   
## 3 1 1 1 3   
## Switzerland United\_Kingdom USSR West\_Germany Yugoslavia   
## 3 3 1 2 1

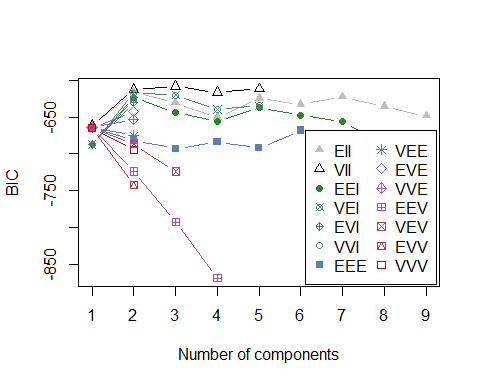
The plot is:

plot(protein\_s,col=mbc.p$classification,main='protein clusters')



The black, green and red denote the three different clusters. Now, we will make a BIC plot to see which combination of model and number of clusters gives us maximum BIC.

plot(mbc.p, what = "BIC")



As we can see from the BIC plot, VII model and 3 clusters gives us maximum BIC.

We can check the model name.

mbc.p$modelName

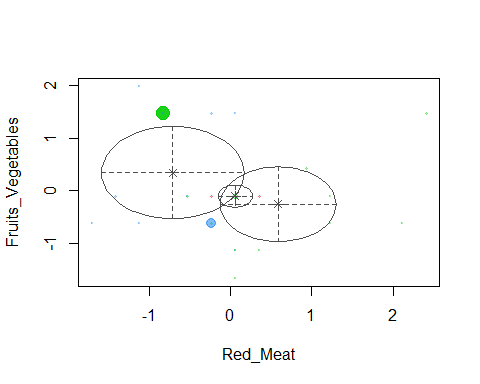
## [1] "VII"

So, the mixture distribution is spherical in shape with unequal volumes.

## e.

Lets plot the uncertainty.

plot(mbc.p, what = "uncertainty", dimens = c(1,9))



There are two major points. We can check this numerically.

protein.clust.data <- cbind(rownames(protein), mbc.p$classification,  
mbc.p$uncertainty)  
protein.clust.data[order(mbc.p$uncertainty),]

## [,1] [,2] [,3]   
## Yugoslavia "Yugoslavia" "1" "3.42037509426518e-11"  
## Albania "Albania" "1" "3.81732645493571e-10"  
## Bulgaria "Bulgaria" "1" "6.09161077136378e-09"  
## Portugal "Portugal" "1" "1.12820238595823e-07"  
## Greece "Greece" "1" "2.83345470086793e-07"  
## Romania "Romania" "1" "3.31861341740236e-07"  
## Ireland "Ireland" "3" "3.7184271273194e-07"   
## Denmark "Denmark" "3" "6.17927211732372e-07"  
## Sweden "Sweden" "3" "5.40233909640531e-06"  
## United\_Kingdom "United\_Kingdom" "3" "1.01241255613393e-05"  
## Belgium "Belgium" "3" "1.37702157101405e-05"  
## The\_Netherlands "The\_Netherlands" "2" "2.32908576307533e-05"  
## Finland "Finland" "3" "3.13618104667013e-05"  
## Spain "Spain" "1" "0.000110627808004748"  
## Hungary "Hungary" "1" "0.000165149878018411"  
## France "France" "3" "0.000215052101973368"  
## Austria "Austria" "2" "0.000279077230874392"  
## East\_Germany "East\_Germany" "3" "0.000296121951420258"  
## Italy "Italy" "1" "0.000320544395165934"  
## Norway "Norway" "3" "0.00078523181235246"   
## Switzerland "Switzerland" "3" "0.00100679853994856"   
## West\_Germany "West\_Germany" "2" "0.00186708582422601"   
## Czechoslovakia "Czechoslovakia" "3" "0.0106407112009743"   
## USSR "USSR" "1" "0.0696205251454159"   
## Poland "Poland" "3" "0.170355308071727"

Poland has the highest uncertainty of 17 percent that it falls in class 3 and USSR with 7 percent uncertainty of being in class 1. Lets compare the values of Poland with the mean values of group 3.

protein\_s['Poland',]

## Red\_Meat White\_Meat Eggs   
## -0.82269954 0.55603579 -0.07174575   
## Milk Fish Cereals   
## 0.24233366 -0.36883881 0.33423255   
## Starchy\_Foods Pulses\_nuts\_oilseeds Fruits\_Vegetables   
## 0.99076023 -0.53489822 1.46225231

colMeans(protein\_s[mbc.p$classification==3,])

## Red\_Meat White\_Meat Eggs   
## 0.5729515 0.2664338 0.5261355   
## Milk Fish Cereals   
## 0.6063038 0.2795107 -0.5513021   
## Starchy\_Foods Pulses\_nuts\_oilseeds Fruits\_Vegetables   
## 0.4873252 -0.6999903 -0.2350048

From the values above, the uncertainty of Poland lying in group 3 makes sense.

# Problem 4.

## a.

Lets read the data first.

courses <- read.csv("https://raw.githubusercontent.com/EricBrownTTU/ISQS6350/main/Coursetopics.csv",  
 header = FALSE)  
head(courses)

## V1 V2 V3 V4 V5 V6 V7 V8  
## 1 Intro DataMining Survey Cat Data Regression Forecast DOE SW  
## 2 1 1 0 0 0 0 0 0  
## 3 0 0 1 0 0 0 0 0  
## 4 0 1 0 1 1 0 0 1  
## 5 1 0 0 0 0 0 0 0  
## 6 1 1 0 0 0 0 0 0

The column names is the first row. So, we need to do some manipulation.

names(courses) <- courses[1,]  
courses <- courses[-1,]

Now, we will find the rules:

library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

myrules <- apriori(courses, parameter = list(support = 0.01, confidence =  
 0.25, minlen = 1))

## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7, 8 not logical or factor. Applying  
## default discretization (see '? discretizeDF').

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.25 0.1 1 none FALSE TRUE 5 0.01 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[16 item(s), 365 transaction(s)] done [0.00s].  
## sorting and recoding items ... [16 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.00s].  
## writing ... [10454 rule(s)] done [0.01s].  
## creating S4 object ... done [0.01s].

It gave us 10454 rules.

## b.

Lets sort the rules by decreasing lift.

inspect(sort(myrules, by = "lift")[1:5])

## lhs rhs support confidence coverage lift count  
## [1] {DataMining=1,   
## Cat Data=0,   
## Regression=1} => {Forecast=1} 0.01643836 1 0.01643836 7.156863 6  
## [2] {Intro=1,   
## DataMining=1,   
## Cat Data=0,   
## Regression=1} => {Forecast=1} 0.01095890 1 0.01095890 7.156863 4  
## [3] {DataMining=1,   
## Cat Data=0,   
## Regression=1,   
## SW=0} => {Forecast=1} 0.01095890 1 0.01095890 7.156863 4  
## [4] {DataMining=1,   
## Survey=0,   
## Cat Data=0,   
## Regression=1} => {Forecast=1} 0.01369863 1 0.01369863 7.156863 5  
## [5] {DataMining=1,   
## Cat Data=0,   
## Regression=1,   
## DOE=0} => {Forecast=1} 0.01643836 1 0.01643836 7.156863 6

The lift for the first relation is 7.15. This means that when someone takes Data Mining and regression courses but not Cat Data they take Forecasting 7.15 times more than when there is no relation. The support is 0.016, which means that the rule is applied in 1.6 percent of transactions. The confidence of 1 means that rule has 100 percent predictive accuracy or everytime the left side is satisfied , the right side is satisfied as well.

## c.

Lets see the rules that contain both Intro and Regression courses.

courses.rules <- subset(myrules, items %in% "Intro=1" & items %in% "Regression=1")

Lets sort the rules by decreasing lift.

inspect(sort(courses.rules, by = "lift")[1:5])

## lhs rhs support confidence coverage lift count  
## [1] {Intro=1,   
## DataMining=1,   
## Cat Data=0,   
## Regression=1} => {Forecast=1} 0.01095890 1 0.01095890 7.156863 4  
## [2] {Intro=1,   
## DataMining=1,   
## Cat Data=0,   
## Regression=1,   
## DOE=0} => {Forecast=1} 0.01095890 1 0.01095890 7.156863 4  
## [3] {Intro=1,   
## Cat Data=0,   
## Regression=1,   
## Forecast=1} => {DataMining=1} 0.01095890 1 0.01095890 5.615385 4  
## [4] {Intro=1,   
## Cat Data=0,   
## Regression=1,   
## Forecast=1,   
## DOE=0} => {DataMining=1} 0.01095890 1 0.01095890 5.615385 4  
## [5] {Intro=1,   
## DataMining=1,   
## Survey=0,   
## Cat Data=1} => {Regression=1} 0.01369863 1 0.01369863 4.802632 5

As we can see, the relation with forecast has the highest lift. This means that someone who has taken Intro and Regression is more likely to take Forecast than without the relation. So, I would recommend the student to take Forecast.