codigoTFG

Jorge de Andrés

12 de enero de 2019

Lo primero que tengo que hacer es importar el dataset que he creado

dataset <- read.csv("C:/Users/jorge/Desktop/Documentos Clase/Universidad/4ºCarrera/2º Cuatrimestre/TFG/Proyecto/datos.txt", header = TRUE)

Ahora lo que hago es pasarlo a una matriz, quitando tanto el nombre (que no me interesa) como la etiqueta (que no la necesito por ahora)

matriz.pacientes.etiquetas <- dataset[, -1]  
matriz.pacientes.datos <- matriz.pacientes.etiquetas[, -25]

# Análisis Exploratorio

Primero compruebo que todos los datos tienen un tipo correcto.

sapply(matriz.pacientes.datos, class)

## edad sex rel\_ctxo\_rel\_mala rel\_ctxo\_trauma   
## "integer" "integer" "integer" "integer"   
## rel\_ctxo\_buena ed\_perm ed\_norm ed\_estr   
## "integer" "integer" "integer" "integer"   
## resil\_ba resil\_me resil\_al pen\_dic   
## "integer" "integer" "integer" "integer"   
## gen\_ex etiq fil\_men max\_min   
## "integer" "integer" "integer" "integer"   
## conc\_arb pseu\_res deb raz\_emo   
## "integer" "integer" "integer" "integer"   
## inhib asert agres impuls   
## "integer" "integer" "integer" "integer"

Veo la media de la edad de los pacientes y el rango en el que se mueve

mean(matriz.pacientes.datos[, 1])

## [1] 26.46269

range(matriz.pacientes.datos[, 1])

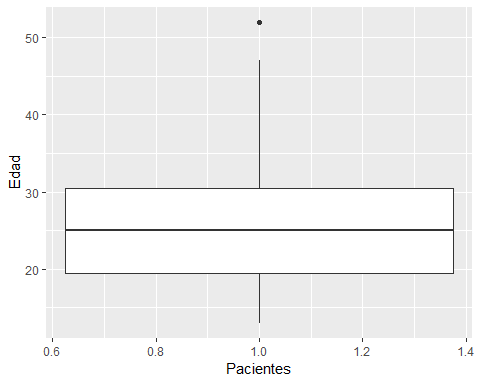
## [1] 13 52

Voy a ver estos datos gráficamente:

#install.packages("ggplot2")  
library("ggplot2")

## Warning: package 'ggplot2' was built under R version 3.5.2

qplot(1, matriz.pacientes.datos[, 1], xlab = "Pacientes", ylab = "Edad", geom="boxplot")



Finalmente, veo un resúmen de cada columna

summary(matriz.pacientes.datos)

## edad sex rel\_ctxo\_rel\_mala rel\_ctxo\_trauma   
## Min. :13.00 Min. :0.000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:19.50 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :25.00 Median :0.000 Median :0.0000 Median :0.0000   
## Mean :26.46 Mean :0.209 Mean :0.1343 Mean :0.3582   
## 3rd Qu.:30.50 3rd Qu.:0.000 3rd Qu.:0.0000 3rd Qu.:1.0000   
## Max. :52.00 Max. :1.000 Max. :1.0000 Max. :1.0000   
## rel\_ctxo\_buena ed\_perm ed\_norm ed\_estr   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.0000 Median :0.0000   
## Mean :0.5075 Mean :0.2836 Mean :0.4925 Mean :0.2239   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## resil\_ba resil\_me resil\_al pen\_dic   
## Min. :0.0000 Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000 1st Qu.:1.0000   
## Median :1.0000 Median :0.0000 Median :0.00000 Median :1.0000   
## Mean :0.5672 Mean :0.4179 Mean :0.01493 Mean :0.8955   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.00000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.00000 Max. :1.0000   
## gen\_ex etiq fil\_men max\_min   
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.0000   
## 1st Qu.:1.0000 1st Qu.:0.5000 1st Qu.:1.000 1st Qu.:1.0000   
## Median :1.0000 Median :1.0000 Median :1.000 Median :1.0000   
## Mean :0.9552 Mean :0.7463 Mean :0.791 Mean :0.9701   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.0000   
## conc\_arb pseu\_res deb raz\_emo   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:1.0000 1st Qu.:0.0000 1st Qu.:1.0000 1st Qu.:1.000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :1.000   
## Mean :0.9851 Mean :0.5075 Mean :0.9403 Mean :0.791   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000   
## inhib asert agres impuls   
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.0000   
## Median :1.0000 Median :0.0000 Median :0.000 Median :1.0000   
## Mean :0.6567 Mean :0.1343 Mean :0.209 Mean :0.6119   
## 3rd Qu.:1.0000 3rd Qu.:0.0000 3rd Qu.:0.000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.0000

Como se puede ver, los datos de los pacientes están muy distanciados, y además su media es muy alta. Así, la media de la edad difiere enormemente del resto de valores de la matriz. Debido a ello, debemos de hacer un preprocesado de los datos del problema.

# Preparación de los datos

Como he comentado antes, Lo que voy a hacer ahora es un centrado y escalado de los datos de la matriz. De esta manera, la red neuronal no tendrá ningún valor que destaque especialmente y con ello no dará de inicio más peso a unos valores que a otros, ya que no lo buscamos.

Lo primero que hacemos es importar la librería caret

#install.packages("caret")  
library(caret)

## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

Ahora hacemos un centrado y escalado de los datos, ya que la edad no sigue el rango del resto de valores, y distorsionaría la predicción

preObjeto <- preProcess(matriz.pacientes.datos, method=c("center", "scale")) # Quiero hacer un centrado y escalado  
matriz.pacientes.datos.centscal <- predict(preObjeto, matriz.pacientes.datos) # Obtengo los valores en la matriz centscal

Después del preprocesado, aunque con los datos no preprocesados, voy a hacer la visualización de algunas relaciones entre variables, de tal manera que podamos ver gráficamente algunos aspectos interesantes:

## Visualización de Datos

Para empezar voy a sacar una nube de palabras para mostrar los nombres más comúnes en los datos facilitados:

# Lo primero que tengo que hacer es contar la frecuencia de los nombres  
  
#install.packages("plyr")  
library(plyr)  
#install.packages("wordcloud")  
library(wordcloud)

## Warning: package 'wordcloud' was built under R version 3.5.2

## Loading required package: RColorBrewer

dataNombres <- ddply(dataset,.(nom),nrow)  
dataNombres <- dataNombres[order(dataNombres$V1, decreasing = TRUE), ]

Una vez que tengo los nombres contados y ordenados, es el momento de crear la WordCloud

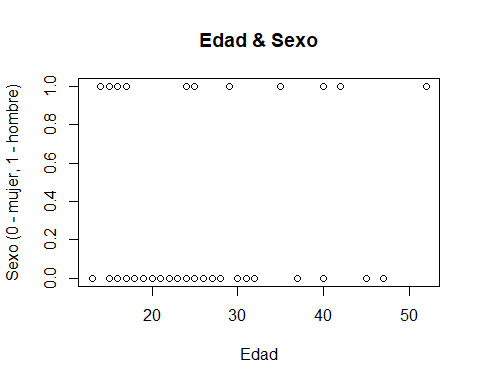
set.seed(9999) #Para el mantenimiento del mismo patrón  
  
wordcloud(words = dataNombres$nom, freq = dataNombres$V1, min.freq = 1, random.order=FALSE, rot.per=0.5, colors=c("Orange","Purple","Pink", "Red", "Yellow", "Green", "Blue", "Black"))

## Warning in wordcloud(words = dataNombres$nom, freq = dataNombres$V1,  
## min.freq = 1, : Yolanda could not be fit on page. It will not be plotted.



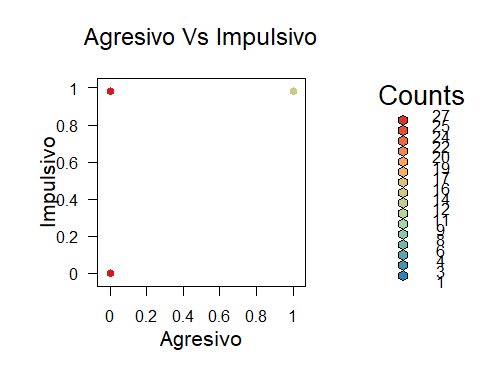
Ahora voy a sacar un plot para ver la relación entre la edad y el sexo de las personas que están en consulta

plot(matriz.pacientes.datos[,1], matriz.pacientes.datos[,2], xlab="Edad", ylab="Sexo (0 - mujer, 1 - hombre)", main="Edad & Sexo")



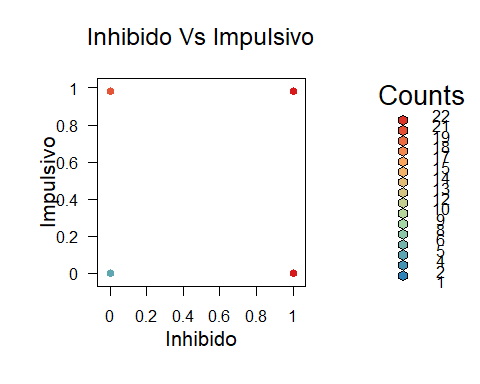
Otro plot para ver la correlación entre ser agresivo y ser impulsivo

#install.packages("hexbin")  
#install.packages("RColorBrewer")  
  
library(hexbin)  
library(RColorBrewer)  
  
rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))  
df <- data.frame(matriz.pacientes.datos[, 23], matriz.pacientes.datos[, 24])  
h <- hexbin(df)  
plot(h, colramp=rf, xlab="Agresivo", ylab="Impulsivo", main="Agresivo Vs Impulsivo")



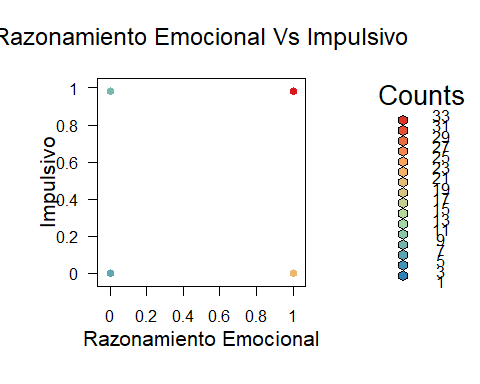
Otro plot similar para ver la relación de ser inhibido e impulsivo

df <- data.frame(matriz.pacientes.datos[, 21], matriz.pacientes.datos[, 24])  
h <- hexbin(df)  
plot(h, colramp=rf, xlab="Inhibido", ylab="Impulsivo", main="Inhibido Vs Impulsivo")



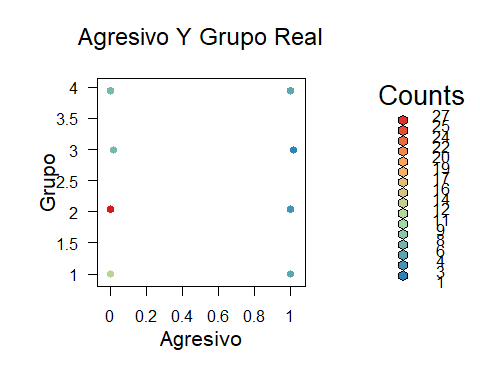
Voy a ver la relación entre el razonamiento emocional (actuar según tus sentimientos) y la impulsividad

df <- data.frame(matriz.pacientes.datos[, 20], matriz.pacientes.datos[, 24])  
h <- hexbin(df)  
plot(h, colramp=rf, xlab="Razonamiento Emocional", ylab="Impulsivo", main="Razonamiento Emocional Vs Impulsivo")



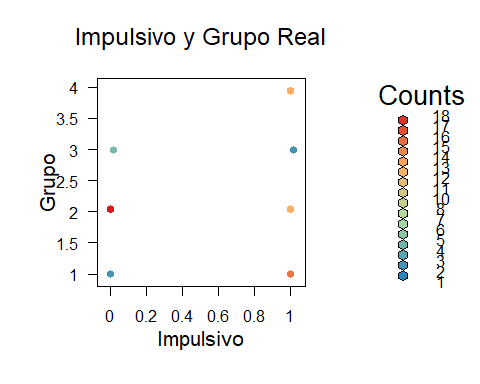
Ahora quiero sacar una relación entre ser agresivo y ver el grupo en el que están

rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))  
df <- data.frame(matriz.pacientes.datos[, 23], matriz.pacientes.etiquetas[, 25])  
h <- hexbin(df)  
plot(h, colramp=rf, xlab="Agresivo", ylab="Grupo", main="Agresivo Y Grupo Real")



Voy a hacer lo mismo con la impulsividad

rf <- colorRampPalette(rev(brewer.pal(4,'Spectral')))  
df <- data.frame(matriz.pacientes.datos[, 24], matriz.pacientes.etiquetas[, 25])  
h <- hexbin(df)  
plot(h, colramp=rf, xlab="Impulsivo", ylab="Grupo", main="Impulsivo y Grupo Real")



De estas gráficas estamos obteniendo información realmente interesante antes de la predicción de los datos. He preferido hacer gráficas en 2D porque las gráficas en 3D son mucho más difíciles de interpretar que estas bonitas gráficas en 2D

Vamos a ver la correlación que tienen mis variables

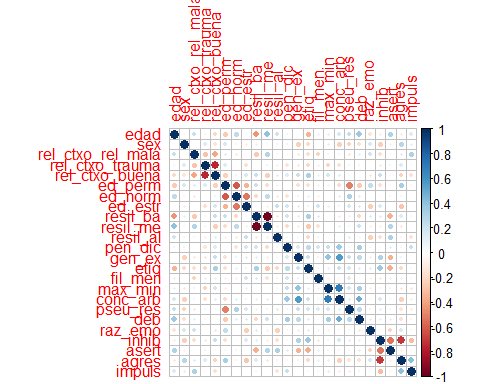
res <- cor(matriz.pacientes.datos[, 1:24], method = "spearman") # Por mi tipo de datos, hacemos la correlación por spearman  
options(width = 100)  
res.round <- round(res, 2)

Como saca una tabla enorme, lo que voy a hacer es usar una librería que me da una función para sacar de una forma bonita las correlaciones entre las variables.

#install.packages("corrplot")  
library(corrplot)

## corrplot 0.84 loaded

corrplot(res.round, method="circle")



Como podemos ver, por ejemplo, resiliencia baja y media tienen una correlación de -1, ya que si hay una no hay la otra y viceversa. Esto pasa igual con las relaciones entre contexto, ya que buena - trauma, trauma - mala, mala - buena tienen que ser inversas.

Ahora voy a sacar un PCA para ver la importancia de las variables:

#install.packages("FactoMineR")  
library("FactoMineR")

## Warning: package 'FactoMineR' was built under R version 3.5.2

#devtools::install\_github("kassambara/factoextra")

Para los cálculos, uso la matriz con el centrado y escalado ya hechos

resultado.pca <- PCA(matriz.pacientes.datos.centscal, graph = FALSE)  
#Con la siguiente línea podemos ver que podemos hacer con esto calculado  
print(resultado.pca)

## \*\*Results for the Principal Component Analysis (PCA)\*\*  
## The analysis was performed on 67 individuals, described by 24 variables  
## \*The results are available in the following objects:  
##   
## name description   
## 1 "$eig" "eigenvalues"   
## 2 "$var" "results for the variables"   
## 3 "$var$coord" "coord. for the variables"   
## 4 "$var$cor" "correlations variables - dimensions"  
## 5 "$var$cos2" "cos2 for the variables"   
## 6 "$var$contrib" "contributions of the variables"   
## 7 "$ind" "results for the individuals"   
## 8 "$ind$coord" "coord. for the individuals"   
## 9 "$ind$cos2" "cos2 for the individuals"   
## 10 "$ind$contrib" "contributions of the individuals"   
## 11 "$call" "summary statistics"   
## 12 "$call$centre" "mean of the variables"   
## 13 "$call$ecart.type" "standard error of the variables"   
## 14 "$call$row.w" "weights for the individuals"   
## 15 "$call$col.w" "weights for the variables"

Nos interesa ver los eigenvalues, que son los que presentarán la cantidad de varianza que aportan las variables:

eigenvalues.PCA <- resultado.pca$eig  
eigenvalues.PCA

## eigenvalue percentage of variance cumulative percentage of variance  
## comp 1 3.896946e+00 1.623727e+01 16.23727  
## comp 2 3.348839e+00 1.395349e+01 30.19077  
## comp 3 2.189584e+00 9.123265e+00 39.31403  
## comp 4 2.044520e+00 8.518834e+00 47.83287  
## comp 5 1.737900e+00 7.241252e+00 55.07412  
## comp 6 1.521215e+00 6.338397e+00 61.41252  
## comp 7 1.374042e+00 5.725176e+00 67.13769  
## comp 8 1.079722e+00 4.498843e+00 71.63653  
## comp 9 9.591848e-01 3.996603e+00 75.63314  
## comp 10 9.311536e-01 3.879807e+00 79.51294  
## comp 11 8.644377e-01 3.601824e+00 83.11477  
## comp 12 8.099267e-01 3.374695e+00 86.48946  
## comp 13 6.658121e-01 2.774217e+00 89.26368  
## comp 14 5.935233e-01 2.473014e+00 91.73669  
## comp 15 4.698651e-01 1.957771e+00 93.69447  
## comp 16 4.632196e-01 1.930082e+00 95.62455  
## comp 17 3.922638e-01 1.634433e+00 97.25898  
## comp 18 2.445767e-01 1.019069e+00 98.27805  
## comp 19 2.251255e-01 9.380229e-01 99.21607  
## comp 20 1.497768e-01 6.240699e-01 99.84014  
## comp 21 3.836592e-02 1.598580e-01 100.00000  
## comp 22 1.475982e-31 6.149926e-31 100.00000  
## comp 23 3.435901e-32 1.431625e-31 100.00000  
## comp 24 4.473592e-33 1.863997e-32 100.00000

Como se puede comprobar, de las 24 variables (componentes) que tenemos, la mitad de la varianza la conseguimos con aproximadamente 5 variables. También se puede ver que a parti de las 17 variables prácticamente no hay un aumento de la varianza. En el caso de un problema grande, sería interesante la eliminación de algunas de las variables, para dejar un dataset más pequeño con el que poder trabajar. En nuestro caso, nuestro problema es pequeño, y además las variables están escogidas a mano, por lo que no haré una reducción del dataset.

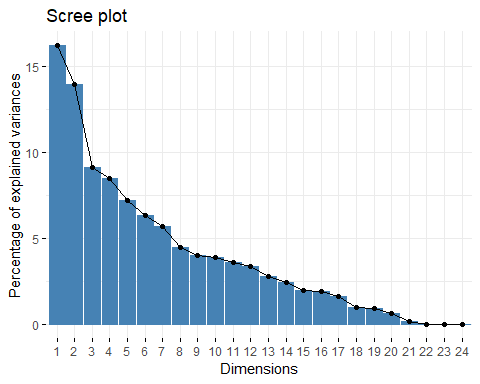
Ahora, para completar este apartado de PCA, lo que voy a hacer es sacar la gráfica de la varianza acumulada con los valores anteriores:

#install.packages("factoextra")  
library("factoextra")

## Warning: package 'factoextra' was built under R version 3.5.2

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

plotPCA <- fviz\_screeplot(resultado.pca, ncp=24)  
plot(plotPCA)



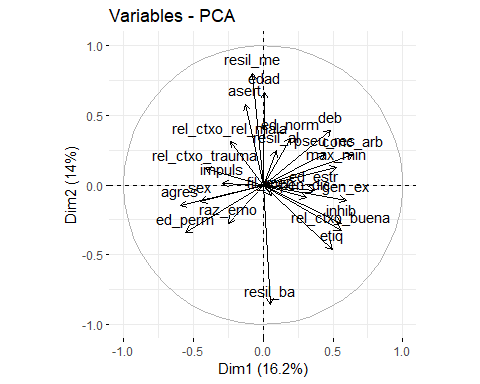
Ahora voy a sacar un “Factor Map” de las variables. Esto lo puedo hacer gracias a las coordenadas que me da una de las variables tras hacer el PCA. Así, voy primero a ver la tabla y luego voy a sacar el mapa:

resultado.pca$var$coord

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## edad 0.007991017 0.66451493 0.19586260 0.007841951 0.02805228  
## sex -0.447913174 -0.11533971 0.10245651 0.057192465 0.11492883  
## rel\_ctxo\_rel\_mala -0.234645431 0.31242667 0.16869825 0.565844840 -0.35019764  
## rel\_ctxo\_trauma -0.417381244 0.11742230 -0.21001522 -0.136376092 0.34458474  
## rel\_ctxo\_buena 0.560339731 -0.32571613 0.08634947 -0.255162421 -0.09161091  
## ed\_perm -0.553915390 -0.34063941 0.23874352 0.241008740 -0.33289199  
## ed\_norm 0.195717913 0.33691768 -0.54551596 0.069259334 0.57202966  
## ed\_estr 0.364218109 -0.03574847 0.39611368 -0.343671891 -0.32610946  
## resil\_ba 0.051112365 -0.85796492 0.20216746 -0.039061871 0.21241196  
## resil\_me -0.074743641 0.80191325 -0.29969255 0.153867444 -0.18199709  
## resil\_al 0.095173049 0.24393919 0.39293737 -0.466258490 -0.12766293  
## pen\_dic 0.311964031 -0.08886036 0.58511514 0.186141485 0.18431008  
## gen\_ex 0.595148670 -0.10912103 0.08286185 0.282531851 0.16255400  
## etiq 0.499365039 -0.45912251 -0.17281954 0.173075672 0.14741419  
## fil\_men 0.059354773 -0.07552104 0.31916448 -0.355626217 0.39559535  
## max\_min 0.524773891 0.12325520 0.31096225 0.378637566 -0.01643986  
## conc\_arb 0.645068936 0.21765964 0.22093906 0.466271365 0.04542650  
## pseu\_res 0.443972323 0.22314014 -0.11950533 -0.176808668 0.43526671  
## deb 0.484524206 0.38834362 0.18502988 0.268386661 0.11400205  
## raz\_emo -0.251049993 -0.27394959 -0.20413968 0.433531845 0.02071493  
## inhib 0.563528317 -0.27772651 -0.48974596 0.018053816 -0.35529325  
## asert -0.126327074 0.57800397 0.34437650 -0.343315766 0.04021108  
## agres -0.591302668 -0.14500954 0.29219301 0.239210208 0.43081535  
## impuls -0.289816690 0.01403726 0.32193103 0.339678159 0.27914883

Como se puede ver, me está poniendo mis 24 variables en 5 dimensiones, con unas coordenadas concretas. Ahora, lo que voy a hacer, es representarlo. Con esta representación podré sacar algunas conclusiones:

fviz\_pca\_var(resultado.pca)



Con esto puedo sacar conclusiones al igual que con el gran gráfico de correlaciones de variables, solo que esta representación está intencionada para más de 2 dimensiones.

Puedo ver algunas de las conclusiones fáciles que saqué anteriormente, como que resiliencia media es contraria a baja, o que la relación con el contexto de trauma y mala son contrarias a buena.

Otras relaciones también puedo ver, como que los deberías y el razonamiento emocional parecen ser ciertamente contrarios, o que el filtro mental no depende de prácticamente nada ya que está en todo el centro.

También es importante ver como, mediante dos componentes principales (dos dimensiones), solo estoy explicando un del 30,2% del total, lo que es muy poco. Por unirlo con los gráficos anteriores, estas dos componentes que se han elegido como x e y son las dos variables que más varianza (y por lo tanto, explicación) tenían en el gráfico de barras anterior.

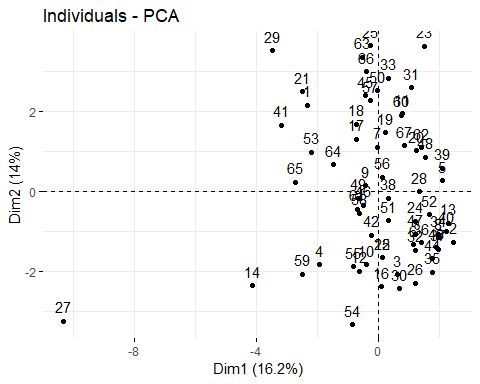
Ahora mi siguiente paso es sacar un gráfico de los individuos, para ver donde están colocados en este sistema:

head(resultado.pca$ind$coord) # Solo saco los primeros para no ocupar demasiado espacio

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## 1 -2.3243690 2.147815 -1.1849618 2.4481512 -0.7586328  
## 2 2.4647257 -1.262473 0.2217190 -1.1784100 -1.4473760  
## 3 0.6387125 -2.080331 -0.1818521 0.7676582 -2.0265412  
## 4 -1.9384395 -1.832160 1.4628618 1.1820858 1.1852182  
## 5 2.0986406 0.262897 -0.2150152 -0.7686587 -1.2663434  
## 6 1.1578332 -1.323444 -0.8453683 0.9774806 -0.1661987

Ahora, tras ver que todos mis individuos tienen unas ciertas coordenadas, vamos a representarlos gráficamente:

fviz\_pca\_ind(resultado.pca)

 Se puede ver que la mayoría de los pacientes están en torno al centro, mientras que tenemos un outlayer, que es el número 27.

# Modelos de Inteligencia Artificial supervisados

Ahora vamos a importar la librería nnet, que nos sirve para hacer perceptrones

#install.packages("nnet")  
library(nnet)

Ahora lo que hago es coger un conjunto muy grande de los datos para hacer el entrenamiento

conjuntoEntrenamiento <- sample(1:67, 55)

1 NEURONA

Lo que voy a hacer ahora es entrenar la red neuronal con diferente cantidad de neuronas,y voy a ir comparando el resultado…

SIN SOFTMAX

pacientes.1neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=1 )

## # weights: 33  
## initial value 51.862628   
## iter 10 value 40.017741  
## iter 20 value 37.710606  
## iter 30 value 37.505552  
## iter 40 value 33.503243  
## iter 50 value 32.995063  
## iter 60 value 27.307868  
## iter 70 value 26.685843  
## iter 80 value 26.503322  
## iter 90 value 26.117161  
## iter 100 value 25.725855  
## final value 25.725855   
## stopped after 100 iterations

#Una vez que lo tengo entrenado, lo que voy a hacer es calcular el error tanto en el entrenamiento como en el test de cada uno  
  
pacientes.prediccion.1neu <- predict( pacientes.1neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.1neu) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.02471211 0.8241317 0.006846030 0.0000000  
## 7 0.48642792 0.0000000 0.065734152 0.4517097  
## 13 0.02423317 0.8715470 0.006759076 0.0000000  
## 8 0.02423317 0.8715470 0.006759076 0.0000000  
## 29 0.02423324 0.8715412 0.006759088 0.0000000  
## 64 0.02423317 0.8715470 0.006759076 0.0000000

# Ahora que los tengo todos entrenados, Determinamos cual es la máxima, es decir, la clase a la que hay que asignar los objetos  
  
pacientes.prediccion.1neu.class <- apply( pacientes.prediccion.1neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.1neu.class

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 1 2 2 2 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1 2 2 1 2 1 2 2 1 1 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 2 1 1 1 2 1 1 2 2 2 2 1 2 2 2 1 1 1 1 1 2 1

# Lo visualizo en forma de tabla para ir viendo el error  
  
table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.1neu.class 1 2 3 4  
## 1 14 2 2 11  
## 2 1 20 5 0

#Calculo el acierto  
  
sum( diag( table( pacientes.prediccion.1neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.6181818

TEST

pacientes.prediccion.test.1neu <- predict( pacientes.1neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.1neu

## 1 2 3 4  
## 1 0.02423317 0.871547 0.006759076 0.0000000  
## 2 0.02423317 0.871547 0.006759076 0.0000000  
## 9 0.48642763 0.000000 0.065734106 0.4516981  
## 16 0.48642792 0.000000 0.065734152 0.4517097  
## 19 0.02423317 0.871547 0.006759076 0.0000000  
## 21 0.02423317 0.871547 0.006759076 0.0000000  
## 23 0.02423317 0.871547 0.006759076 0.0000000  
## 31 0.02423317 0.871547 0.006759076 0.0000000  
## 38 0.48642792 0.000000 0.065734152 0.4517097  
## 44 0.48642792 0.000000 0.065734152 0.4517097  
## 53 0.48642792 0.000000 0.065734152 0.4517097  
## 60 0.02423317 0.871547 0.006759076 0.0000000

pacientes.prediccion.test.1neu.class <- apply( pacientes.prediccion.test.1neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.1neu.class

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 1 1 2 2 2 2 1 1 1 2

table( pacientes.prediccion.test.1neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.1neu.class 1 2 3 4  
## 1 0 3 1 1  
## 2 2 5 0 0

sum( diag( table( pacientes.prediccion.test.1neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.4166667

Lo voy a entrenar también con el SOFTMAX = true. Esto optimiza la verosimilitud, no el error cuadrático medio… ###################### CON SOFTMAX ##############################

pacientes.1neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=1, softmax = T )

## # weights: 33  
## initial value 74.527157   
## iter 10 value 54.298664  
## iter 20 value 51.835971  
## iter 30 value 51.614314  
## iter 40 value 51.575723  
## iter 50 value 51.450384  
## iter 60 value 51.447976  
## final value 51.447972   
## converged

#Una vez que lo tengo entrenado, lo que voy a hacer es calcular el error tanto en el entrenamiento como en el test de cada uno  
  
pacientes.prediccion.1neu.softmax <- predict( pacientes.1neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.1neu.softmax) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.1330273 0.67877714 0.17842780 0.009767797  
## 7 0.4129254 0.01955695 0.03692291 0.530594707  
## 13 0.1330273 0.67877714 0.17842780 0.009767797  
## 8 0.1330273 0.67877714 0.17842780 0.009767797  
## 29 0.1330273 0.67877714 0.17842780 0.009767797  
## 64 0.1330273 0.67877714 0.17842780 0.009767797

# Ahora que los tengo todos entrenados, Determinamos cual es la máxima, es decir, la clase a la que hay que asignar los objetos  
  
pacientes.prediccion.1neu.class.softmax <- apply( pacientes.prediccion.1neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.1neu.class.softmax

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 2 1 2 4 2 1 2 4 2 4 2 4 2 4 2 2 4 2 4 2 2 2 2 4 1 4 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 2 4 2 1 2 4 4 2 2 2 2 1 2 2 2 2 1 4 4 1 2 4

# Lo visualizo en forma de tabla para ir viendo el error  
  
table( pacientes.prediccion.1neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.1neu.class.softmax 1 2 3 4  
## 1 8 0 0 0  
## 2 3 21 6 1  
## 4 4 1 1 10

#Calculo el acierto  
  
sum( diag( table( pacientes.prediccion.1neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.5454545

TEST

pacientes.prediccion.test.1neu.softmax <- predict( pacientes.1neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.1neu.softmax

## 1 2 3 4  
## 1 0.1330273 0.67877714 0.17842780 0.009767797  
## 2 0.1330273 0.67877714 0.17842780 0.009767797  
## 9 0.1330273 0.67877714 0.17842780 0.009767797  
## 16 0.4129254 0.01955695 0.03692291 0.530594707  
## 19 0.1330273 0.67877714 0.17842780 0.009767797  
## 21 0.4129254 0.01955695 0.03692291 0.530594707  
## 23 0.1330273 0.67877714 0.17842780 0.009767797  
## 31 0.1330273 0.67877714 0.17842780 0.009767797  
## 38 0.4129254 0.01955695 0.03692291 0.530594707  
## 44 0.4129254 0.01955695 0.03692291 0.530594707  
## 53 0.4129254 0.01955695 0.03692291 0.530594707  
## 60 0.1330273 0.67877714 0.17842780 0.009767797

pacientes.prediccion.test.1neu.class.softmax <- apply( pacientes.prediccion.test.1neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.1neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 4 2 2 4 4 4 2

table( pacientes.prediccion.test.1neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.1neu.class.softmax 1 2 3 4  
## 2 2 5 0 0  
## 4 0 3 1 1

sum( diag( table( pacientes.prediccion.test.1neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.4166667

2 NEURONAS

A partir de ahora voy a hacer exactamente lo mismo, por lo que haré chunks más grandes para evitar una sobrecarga de chunks, y reduciré la cantidad de comentarios, ya que serán redundantes

SIN SOFTMAX

pacientes.2neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=2 )

## # weights: 62  
## initial value 49.538983   
## iter 10 value 28.265769  
## iter 20 value 21.648355  
## iter 30 value 20.729675  
## iter 40 value 19.925328  
## iter 50 value 19.732282  
## iter 60 value 19.592685  
## iter 70 value 19.539035  
## iter 80 value 19.474595  
## iter 90 value 19.435052  
## iter 100 value 19.389785  
## final value 19.389785   
## stopped after 100 iterations

pacientes.prediccion.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.2neu) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.04399746 0.9068493 0.0000000000 0.0000000  
## 7 0.58184743 0.0000000 0.0008188038 0.3709968  
## 13 0.04399735 0.9068503 0.0000000000 0.0000000  
## 8 0.04399735 0.9068503 0.0000000000 0.0000000  
## 29 0.04399735 0.9068503 0.0000000000 0.0000000  
## 64 0.04399735 0.9068503 0.0000000000 0.0000000

pacientes.prediccion.2neu.class <- apply( pacientes.prediccion.2neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.2neu.class

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 1 2 2 2 2 2 1 3 4 2 1 2 1 2 1 2 1 2 1 2 2 4 2 1 2 3 1 3 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 1 1 2 1 2 1 1 2 2 2 3 1 2 2 1 3 1 4 1 1 2 4

table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.2neu.class 1 2 3 4  
## 1 13 1 3 6  
## 2 1 21 0 1  
## 3 1 0 4 0  
## 4 0 0 0 4

sum( diag( table( pacientes.prediccion.2neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.7636364

TEST

pacientes.prediccion.test.2neu <- predict( pacientes.2neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.2neu

## 1 2 3 4  
## 1 0.04527193 8.942445e-01 0.0000000000 0.0000000  
## 2 0.04399743 9.068496e-01 0.0000000000 0.0000000  
## 9 0.04399735 9.068503e-01 0.0000000000 0.0000000  
## 16 0.10697593 1.319347e-06 0.0000000000 0.9159080  
## 19 0.04399735 9.068503e-01 0.0000000000 0.0000000  
## 21 0.04399735 9.068503e-01 0.0000000000 0.0000000  
## 23 0.34685167 9.570256e-05 0.7833317375 0.0000000  
## 31 0.34686283 9.566449e-05 0.7833681779 0.0000000  
## 38 0.58184926 0.000000e+00 0.0008187502 0.3710486  
## 44 0.58184926 0.000000e+00 0.0008187502 0.3710486  
## 53 0.57008106 0.000000e+00 0.0005168065 0.3856317  
## 60 0.34685980 9.568566e-05 0.7833879588 0.0000000

pacientes.prediccion.test.2neu.class <- apply( pacientes.prediccion.test.2neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.2neu.class

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 2 3 3 1 1 1 3

table( pacientes.prediccion.test.2neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.2neu.class 1 2 3 4  
## 1 0 2 1 0  
## 2 1 4 0 0  
## 3 1 2 0 0  
## 4 0 0 0 1

sum( diag( table( pacientes.prediccion.test.2neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.4166667

CON SOFTMAX

pacientes.2neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=2, softmax = T )

## # weights: 62  
## initial value 82.062671   
## iter 10 value 44.878111  
## iter 20 value 39.410574  
## iter 30 value 38.715237  
## iter 40 value 38.080741  
## iter 50 value 37.984463  
## iter 60 value 37.960784  
## iter 70 value 37.957038  
## iter 80 value 37.956238  
## final value 37.956160   
## converged

pacientes.prediccion.test.2neu.softmax <- predict( pacientes.2neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.test.2neu.softmax)

## 1 2 3 4  
## 1 0.04175584 0.56988046 0.27725286 0.1111108  
## 2 0.11225648 0.82475938 0.06298414 0.0000000  
## 9 0.04175584 0.56988046 0.27725286 0.1111108  
## 16 0.11894351 0.01061556 0.10120990 0.7692310  
## 19 0.04175584 0.56988046 0.27725286 0.1111108  
## 21 0.11225648 0.82475938 0.06298414 0.0000000

pacientes.prediccion.test.2neu.class.softmax <- apply( pacientes.prediccion.test.2neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.2neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 2 2 2 4 1 1 1

table( pacientes.prediccion.test.2neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.2neu.class.softmax 1 2 3 4  
## 1 0 3 0 0  
## 2 2 5 0 0  
## 4 0 0 1 1

sum( diag( table( pacientes.prediccion.test.2neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.5

TEST

pacientes.prediccion.test.2neu.softmax <- predict( pacientes.2neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.2neu.softmax

## 1 2 3 4  
## 1 0.04175584 0.56988046 0.27725286 0.1111108  
## 2 0.11225648 0.82475938 0.06298414 0.0000000  
## 9 0.04175584 0.56988046 0.27725286 0.1111108  
## 16 0.11894351 0.01061556 0.10120990 0.7692310  
## 19 0.04175584 0.56988046 0.27725286 0.1111108  
## 21 0.11225648 0.82475938 0.06298414 0.0000000  
## 23 0.11225648 0.82475938 0.06298414 0.0000000  
## 31 0.11225648 0.82475938 0.06298414 0.0000000  
## 38 0.11894351 0.01061556 0.10120990 0.7692310  
## 44 0.89289881 0.04289966 0.06420153 0.0000000  
## 53 0.89289881 0.04289966 0.06420153 0.0000000  
## 60 0.89289881 0.04289966 0.06420153 0.0000000

pacientes.prediccion.test.2neu.class.softmax <- apply( pacientes.prediccion.test.2neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.2neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 2 2 2 4 1 1 1

table( pacientes.prediccion.test.2neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.2neu.class.softmax 1 2 3 4  
## 1 0 3 0 0  
## 2 2 5 0 0  
## 4 0 0 1 1

sum(diag(table(pacientes.prediccion.test.2neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.5

3 NEURONAS

SIN SOFTMAX

pacientes.3neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3)

## # weights: 91  
## initial value 61.871178   
## iter 10 value 47.817664  
## iter 20 value 35.301007  
## iter 30 value 33.397692  
## iter 40 value 27.491537  
## iter 50 value 26.718529  
## iter 60 value 25.237718  
## iter 70 value 24.603305  
## iter 80 value 23.897311  
## iter 90 value 23.835162  
## iter 100 value 23.728130  
## final value 23.728130   
## stopped after 100 iterations

pacientes.prediccion.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.3neu) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.0790717188 0.86307818 1.011850e-06 0.000000e+00  
## 7 0.0514413954 0.04334407 0.000000e+00 0.000000e+00  
## 13 0.0085650607 0.99865510 6.622849e-06 1.774781e-06  
## 8 0.0806728681 0.85810479 9.960454e-07 0.000000e+00  
## 29 0.0028626209 0.95274697 6.398238e-07 4.053717e-07  
## 64 0.0005845275 0.99999496 5.980341e-05 2.263103e-05

pacientes.prediccion.3neu.class <- apply( pacientes.prediccion.3neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.3neu.class

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 1 2 2 2 2 2 1 1 1 2 1 2 1 2 2 2 1 2 1 2 2 1 2 1 2 1 1 1 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 1 1 1 1 2 1 1 2 2 2 1 1 2 2 1 1 1 1 1 1 2 1

table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.3neu.class 1 2 3 4  
## 1 14 0 7 11  
## 2 1 22 0 0

sum( diag( table( pacientes.prediccion.3neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.6545455

TEST

pacientes.prediccion.test.3neu <- predict( pacientes.3neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.3neu

## 1 2 3 4  
## 1 0.0005887647 0.999994745 5.861290e-05 2.223046e-05  
## 2 0.0248025521 0.002217553 0.000000e+00 0.000000e+00  
## 9 0.0077799158 0.714804490 0.000000e+00 0.000000e+00  
## 16 0.7404113928 0.000000000 0.000000e+00 0.000000e+00  
## 19 0.0028524065 0.953592695 6.457713e-07 4.087532e-07  
## 21 0.0028612552 0.952791652 6.400741e-07 4.055552e-07  
## 23 0.9390174608 0.000000000 0.000000e+00 0.000000e+00  
## 31 0.1073746767 0.000000000 0.000000e+00 0.000000e+00  
## 38 0.2838860861 0.001122148 0.000000e+00 0.000000e+00  
## 44 0.7586529062 0.000000000 0.000000e+00 0.000000e+00  
## 53 0.1784726285 0.000000000 0.000000e+00 0.000000e+00  
## 60 0.0058818925 0.816656446 3.534651e-07 0.000000e+00

pacientes.prediccion.test.3neu.class <- apply( pacientes.prediccion.test.3neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.3neu.class

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 1 2 1 2 2 1 1 1 1 1 2

table( pacientes.prediccion.test.3neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.3neu.class 1 2 3 4  
## 1 1 4 1 1  
## 2 1 4 0 0

sum( diag( table( pacientes.prediccion.test.3neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.4166667

CON SOFTMAX

pacientes.3neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, softmax = T)

## # weights: 91  
## initial value 76.645794   
## iter 10 value 39.722801  
## iter 20 value 23.909792  
## iter 30 value 21.739006  
## iter 40 value 20.636657  
## iter 50 value 19.463426  
## iter 60 value 18.636896  
## iter 70 value 16.789124  
## iter 80 value 15.476038  
## iter 90 value 14.292593  
## iter 100 value 13.917275  
## final value 13.917275   
## stopped after 100 iterations

pacientes.prediccion.3neu.softmax <- predict( pacientes.3neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.3neu.softmax) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 5.867156e-04 9.994133e-01 2.025784e-10 3.919818e-59  
## 7 9.287588e-07 1.192536e-09 2.175601e-03 9.978235e-01  
## 13 2.023741e-03 9.975806e-01 3.956480e-04 6.499652e-42  
## 8 9.848131e-01 1.518687e-02 5.037061e-10 7.194121e-79  
## 29 6.528380e-08 9.999967e-01 3.189913e-06 1.940646e-26  
## 64 2.634249e-11 4.948343e-01 3.205757e-03 5.019600e-01

pacientes.prediccion.3neu.class.softmax <- apply(pacientes.prediccion.3neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.3neu.class.softmax

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 1 2 4 2 1 3 4 2 1 2 4 2 4 2 4 2 1 2 2 4 2 4 2 3 1 1 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 3 4 4 2 2 3 4 2 2 2 3 1 2 2 3 1 1 4 4 1 2 4

table( pacientes.prediccion.3neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.3neu.class.softmax 1 2 3 4  
## 1 13 0 1 0  
## 2 1 20 0 0  
## 3 0 0 6 0  
## 4 1 2 0 11

sum( diag( table( pacientes.prediccion.3neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.9090909

TEST

pacientes.prediccion.test.3neu.softmax <- predict( pacientes.3neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.3neu.softmax

## 1 2 3 4  
## 1 1.468009e-06 1.943408e-09 3.269218e-03 9.967293e-01  
## 2 6.356359e-04 9.993644e-01 1.724356e-10 1.229489e-59  
## 9 2.544318e-11 4.823744e-01 3.129657e-03 5.144959e-01  
## 16 9.287588e-07 1.192536e-09 2.175601e-03 9.978235e-01  
## 19 1.217431e-01 8.780797e-01 1.771480e-04 6.144726e-52  
## 21 2.544318e-11 4.823744e-01 3.129657e-03 5.144959e-01  
## 23 6.382309e-04 9.993618e-01 1.716794e-10 1.188347e-59  
## 31 1.220686e-01 8.777549e-01 1.765075e-04 5.988973e-52  
## 38 1.000000e+00 1.066274e-10 5.155116e-12 1.024574e-60  
## 44 1.000000e+00 8.879413e-09 8.722955e-10 2.334808e-48  
## 53 1.000000e+00 1.063948e-10 5.147360e-12 1.013220e-60  
## 60 9.972831e-01 2.716763e-03 1.445202e-07 8.660553e-63

pacientes.prediccion.test.3neu.class.softmax <- apply( pacientes.prediccion.test.3neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.3neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 4 2 4 4 2 4 2 2 1 1 1 1

table( pacientes.prediccion.test.3neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.3neu.class.softmax 1 2 3 4  
## 1 0 3 1 0  
## 2 1 3 0 0  
## 4 1 2 0 1

sum( diag( table( pacientes.prediccion.test.3neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.25

3 NEURONAS

Con Decay

SIN SOFTMAX

pacientes.3neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, decay = 0.2)

## # weights: 91  
## initial value 53.360776   
## iter 10 value 34.587550  
## iter 20 value 32.461109  
## iter 30 value 32.308555  
## iter 40 value 32.305073  
## final value 32.305070   
## converged

pacientes.prediccion.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.3neu.decay) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.19143783 0.5294518 0.1972312 0.17498425  
## 7 0.32932890 0.2493516 0.2306942 0.34437529  
## 13 0.26574009 0.7187337 0.0938256 0.05024108  
## 8 0.19302034 0.6920074 0.1388143 0.07768974  
## 29 0.10848820 0.6973226 0.1628088 0.20100781  
## 64 0.07680703 0.7351412 0.1844644 0.22525715

pacientes.prediccion.3neu.class.decay <- apply( pacientes.prediccion.3neu.decay, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.3neu.class.decay

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 2 1 2 2 2 1 2 1 2 1 2 1 2 1 2 2 4 2 1 2 2 1 2 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 2 4 2 2 2 4 4 2 2 2 2 1 2 2 2 2 1 4 4 1 2 4

table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.3neu.class.decay 1 2 3 4  
## 1 12 1 0 2  
## 2 3 21 6 2  
## 4 0 0 1 7

sum( diag( table( pacientes.prediccion.3neu.class.decay, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.6181818

TEST

pacientes.prediccion.test.3neu.decay <- predict( pacientes.3neu.decay, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.3neu.decay

## 1 2 3 4  
## 1 0.11479502 0.4769102 0.36453329 0.23937238  
## 2 0.30177983 0.6681470 0.09433532 0.05821056  
## 9 0.07480884 0.7246551 0.21908156 0.19714766  
## 16 0.38841079 0.2635217 0.17626687 0.30631392  
## 19 0.07169629 0.7046126 0.27075163 0.17680643  
## 21 0.05429864 0.8081079 0.21238206 0.15679923  
## 23 0.36056764 0.5428384 0.12297850 0.06725794  
## 31 0.38477687 0.4682036 0.09848236 0.14737872  
## 38 0.67360778 0.1362528 0.15015957 0.23809737  
## 44 0.63247940 0.2045096 0.10406588 0.23010604  
## 53 0.38422086 0.2011038 0.21287287 0.41256789  
## 60 0.20536251 0.3636631 0.26524601 0.30770728

pacientes.prediccion.test.3neu.class.decay <- apply( pacientes.prediccion.test.3neu.decay, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.3neu.class.decay

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 1 2 2 2 2 1 1 4 2

table( pacientes.prediccion.test.3neu.class.decay , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.3neu.class.decay 1 2 3 4  
## 1 0 1 1 1  
## 2 2 6 0 0  
## 4 0 1 0 0

sum( diag( table( pacientes.prediccion.test.3neu.class.decay, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.5

CON SOFTMAX

pacientes.3neu.decay.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=3, softmax = T, decay = 0.03)

## # weights: 91  
## initial value 73.613757   
## iter 10 value 46.287694  
## iter 20 value 34.545127  
## iter 30 value 31.877008  
## iter 40 value 30.319204  
## iter 50 value 30.067493  
## iter 60 value 29.983304  
## iter 70 value 29.552233  
## iter 80 value 29.346772  
## iter 90 value 28.649292  
## iter 100 value 27.880725  
## final value 27.880725   
## stopped after 100 iterations

pacientes.prediccion.3neu.decay.softmax <- predict( pacientes.3neu.decay.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.3neu.decay.softmax) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 3.531878e-03 0.95404045 0.0400578740 0.002369794  
## 7 2.463093e-02 0.09176792 0.0165621392 0.867039018  
## 13 3.333570e-03 0.95382620 0.0411025750 0.001737658  
## 8 1.020107e-01 0.53191629 0.3637301557 0.002342877  
## 29 6.477595e-02 0.73383341 0.1148460321 0.086544608  
## 64 2.383382e-05 0.97395884 0.0005018983 0.025515424

pacientes.prediccion.3neu.class.decay.softmax <- apply( pacientes.prediccion.3neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.3neu.class.decay.softmax

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 2 1 3 4 2 1 2 4 2 2 2 4 2 1 2 2 4 2 1 2 3 3 3 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 1 4 2 1 2 1 4 2 2 2 3 1 3 2 3 3 1 4 4 1 2 4

table( pacientes.prediccion.3neu.class.decay.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.3neu.class.decay.softmax 1 2 3 4  
## 1 12 0 2 0  
## 2 1 21 0 1  
## 3 2 1 5 0  
## 4 0 0 0 10

sum( diag( table( pacientes.prediccion.3neu.class.decay.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.8727273

TEST

pacientes.prediccion.test.3neu.decay.softmax <- predict( pacientes.3neu.decay.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.3neu.decay.softmax

## 1 2 3 4  
## 1 1.444490e-04 9.788053e-01 0.0020601530 1.899008e-02  
## 2 3.055655e-03 9.600707e-01 0.0340823332 2.791297e-03  
## 9 3.719402e-05 9.836348e-01 0.0007887197 1.553926e-02  
## 16 7.380398e-01 2.524370e-03 0.0498200381 2.096158e-01  
## 19 8.638381e-05 9.723937e-01 0.0012782175 2.624172e-02  
## 21 3.331787e-05 9.662755e-01 0.0006114489 3.307972e-02  
## 23 9.837426e-01 2.580638e-06 0.0161835877 7.122648e-05  
## 31 7.743745e-01 9.792208e-03 0.1809706032 3.486267e-02  
## 38 7.191849e-01 8.354252e-03 0.0820715994 1.903892e-01  
## 44 8.248876e-01 4.638339e-03 0.1364162027 3.405788e-02  
## 53 4.636544e-02 1.089406e-02 0.0100593272 9.326812e-01  
## 60 2.545860e-01 3.361961e-01 0.3728025147 3.641533e-02

pacientes.prediccion.test.3neu.class.decay.softmax <- apply( pacientes.prediccion.test.3neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.3neu.class.decay.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 1 2 2 1 1 1 1 4 3

table( pacientes.prediccion.test.3neu.class.decay.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.3neu.class.decay.softmax 1 2 3 4  
## 1 1 2 1 1  
## 2 1 4 0 0  
## 3 0 1 0 0  
## 4 0 1 0 0

sum( diag( table( pacientes.prediccion.test.3neu.class.decay.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.4166667

5 NEURONAS

SIN SOFTMAX

pacientes.5neu <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5 )

## # weights: 149  
## initial value 77.393461   
## iter 10 value 41.294234  
## iter 20 value 34.752125  
## iter 30 value 34.008274  
## final value 34.000141   
## converged

pacientes.prediccion.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.5neu) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.000000e+00 0.9999839 0.000000e+00 0.000000e+00  
## 7 0.000000e+00 0.0000000 0.000000e+00 0.000000e+00  
## 13 1.291181e-06 1.0000000 0.000000e+00 0.000000e+00  
## 8 1.291229e-06 1.0000000 0.000000e+00 0.000000e+00  
## 29 1.149782e-04 1.0000000 3.475805e-06 1.486962e-05  
## 64 6.489409e-05 1.0000000 1.355004e-06 7.260470e-06

pacientes.prediccion.5neu.class <- apply( pacientes.prediccion.5neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.5neu.class

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 3 2 2 2 2 2 3 1 2 2 1 2 2 2 2 2 1 2 2 2 2 4 2 2 2 2 4 2 1 2 2 2   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 1 1 2 4 2 2 2 2 2 2 1 4 2 2 4 2 4 2 1 2 2 2

table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.5neu.class 1 2 3 4  
## 1 2 0 3 3  
## 2 8 22 3 6  
## 3 1 0 0 1  
## 4 4 0 1 1

sum( diag( table( pacientes.prediccion.5neu.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.4545455

TEST

pacientes.prediccion.test.5neu <- predict( pacientes.5neu, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.5neu

## 1 2 3 4  
## 1 0.000000e+00 4.847313e-07 0 0  
## 2 0.000000e+00 9.989812e-01 0 0  
## 9 1.191935e-06 3.561018e-05 0 0  
## 16 0.000000e+00 0.000000e+00 0 0  
## 19 1.993962e-06 9.962742e-01 0 0  
## 21 0.000000e+00 9.989728e-01 0 0  
## 23 0.000000e+00 0.000000e+00 0 0  
## 31 0.000000e+00 2.571683e-05 0 0  
## 38 0.000000e+00 0.000000e+00 0 0  
## 44 0.000000e+00 0.000000e+00 0 0  
## 53 0.000000e+00 0.000000e+00 0 0  
## 60 0.000000e+00 7.627999e-04 0 0

pacientes.prediccion.test.5neu.class <- apply( pacientes.prediccion.test.5neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.5neu.class

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 2 1 2 2 3 3 2

table( pacientes.prediccion.test.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.5neu.class 1 2 3 4  
## 1 1 0 0 0  
## 2 1 6 1 0  
## 3 0 2 0 0  
## 4 0 0 0 1

sum( diag( table( pacientes.prediccion.test.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.6666667

CON SOFTMAX

pacientes.5neu.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, softmax = T )

## # weights: 149  
## initial value 74.417415   
## iter 10 value 23.034961  
## iter 20 value 8.300445  
## iter 30 value 7.489889  
## iter 40 value 6.980184  
## iter 50 value 6.953290  
## iter 60 value 6.939518  
## iter 70 value 6.937036  
## iter 80 value 6.898074  
## iter 90 value 6.829923  
## iter 100 value 6.757621  
## final value 6.757621   
## stopped after 100 iterations

pacientes.prediccion.5neu.softmax <- predict( pacientes.5neu.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.5neu.softmax) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 1.512695e-09 1.000000e+00 3.360357e-10 2.597440e-20  
## 7 9.148869e-08 2.661221e-20 5.255573e-06 9.999947e-01  
## 13 3.528086e-08 1.000000e+00 7.282038e-13 5.292564e-22  
## 8 2.975084e-01 7.024910e-01 6.507887e-07 7.374263e-23  
## 29 1.696553e-14 9.999849e-01 1.508363e-05 3.289168e-16  
## 64 2.168360e-19 9.999511e-01 4.889323e-05 1.732838e-08

pacientes.prediccion.5neu.class.softmax <- apply( pacientes.prediccion.5neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.5neu.class.softmax

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 2 1 3 4 2 1 2 2 2 2 2 4 2 1 2 2 4 2 2 2 3 1 3 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 3 4 4 1 2 3 4 2 2 2 3 1 2 2 3 3 1 4 4 1 2 4

table( pacientes.prediccion.5neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.5neu.class.softmax 1 2 3 4  
## 1 12 0 0 0  
## 2 2 22 0 1  
## 3 1 0 7 0  
## 4 0 0 0 10

sum( diag( table( pacientes.prediccion.5neu.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.9272727

TEST

pacientes.prediccion.test.5neu.softmax <- predict( pacientes.5neu.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.5neu.softmax

## 1 2 3 4  
## 1 3.828020e-19 9.999301e-01 6.990122e-05 3.452630e-08  
## 2 3.528086e-08 1.000000e+00 7.282038e-13 5.292564e-22  
## 9 2.218978e-05 2.421065e-03 9.971538e-01 4.029245e-04  
## 16 1.696656e-01 3.425064e-07 8.303341e-01 7.468970e-10  
## 19 2.506382e-14 4.178556e-07 3.238713e-04 9.996757e-01  
## 21 2.186451e-19 9.999512e-01 4.879881e-05 1.700813e-08  
## 23 9.999552e-01 4.483995e-05 8.702427e-10 9.094930e-12  
## 31 9.999868e-01 1.317458e-05 1.093286e-12 2.364499e-13  
## 38 9.118103e-08 2.251733e-20 4.709600e-06 9.999952e-01  
## 44 9.999869e-01 1.314015e-05 1.092929e-12 2.357253e-13  
## 53 9.684551e-03 1.741833e-10 6.738615e-04 9.896416e-01  
## 60 1.000000e+00 3.867076e-13 8.171475e-14 4.326095e-21

pacientes.prediccion.test.5neu.class.softmax <- apply( pacientes.prediccion.test.5neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.5neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 3 3 4 2 1 1 4 1 4 1

table( pacientes.prediccion.test.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.5neu.class.softmax 1 2 3 4  
## 1 1 3 0 0  
## 2 1 2 0 0  
## 3 0 1 0 1  
## 4 0 2 1 0

sum( diag( table( pacientes.prediccion.test.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.25

5 NEURONAS

CON DECAY

SIN SOFTMAX

pacientes.5neu.decay <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, decay=0.1)

## # weights: 149  
## initial value 52.375807   
## iter 10 value 29.383238  
## iter 20 value 25.744753  
## iter 30 value 24.933954  
## iter 40 value 24.840943  
## iter 50 value 24.767431  
## iter 60 value 24.679744  
## iter 70 value 24.654604  
## iter 80 value 24.652131  
## iter 90 value 24.652048  
## iter 90 value 24.652048  
## iter 90 value 24.652048  
## final value 24.652048   
## converged

pacientes.prediccion.5neu.decay <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.5neu.decay) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.10264898 0.6821730 0.23044727 0.14295809  
## 7 0.17288169 0.1391311 0.16026890 0.48814966  
## 13 0.29010585 0.8952882 0.04107492 0.02098322  
## 8 0.21812660 0.8089657 0.14857651 0.01757076  
## 29 0.04689942 0.8031595 0.13134707 0.12902323  
## 64 0.05495244 0.7977479 0.15939985 0.19419123

pacientes.prediccion.5neu.decay.class <- apply( pacientes.prediccion.5neu.decay, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.5neu.decay.class

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 1 1 3 4 2 1 2 1 2 2 2 1 2 1 2 2 4 2 1 2 3 1 2 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 3 4 4 1 2 4 4 2 2 2 3 1 2 2 3 2 1 4 4 1 2 4

table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.5neu.decay.class 1 2 3 4  
## 1 13 1 0 2  
## 2 2 21 1 0  
## 3 0 0 5 0  
## 4 0 0 1 9

sum( diag( table( pacientes.prediccion.5neu.decay.class, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.8727273

TEST

pacientes.prediccion.test.decay.5neu <- predict( pacientes.5neu.decay, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.decay.5neu

## 1 2 3 4  
## 1 0.04994066 0.77983599 0.14900039 0.12781777  
## 2 0.47083319 0.68918317 0.03054322 0.03488642  
## 9 0.01646834 0.76080682 0.25290340 0.32641544  
## 16 0.37940097 0.15938666 0.08542637 0.52810383  
## 19 0.01777844 0.57967573 0.35629919 0.32042600  
## 21 0.01083372 0.73015435 0.25885963 0.42311760  
## 23 0.62811729 0.41189732 0.20674800 0.01310582  
## 31 0.37493099 0.49651022 0.02183679 0.19284275  
## 38 0.73392890 0.03793791 0.22571329 0.20228526  
## 44 0.62160693 0.17373931 0.04357842 0.46016291  
## 53 0.29389273 0.15332482 0.24267419 0.25578093  
## 60 0.08731519 0.46844844 0.06293776 0.29285021

pacientes.prediccion.test.decay.5neu.class <- apply( pacientes.prediccion.test.decay.5neu, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.decay.5neu.class

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 2 2 2 4 2 2 1 2 1 1 1 2

table( pacientes.prediccion.test.decay.5neu.class , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.decay.5neu.class 1 2 3 4  
## 1 1 2 1 0  
## 2 1 6 0 0  
## 4 0 0 0 1

sum( diag( table( pacientes.prediccion.test.decay.5neu.class, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.5833333

CON SOFTMAX

pacientes.5neu.decay.softmax <- nnet( matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], class.ind( matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) , size=5, softmax = T, decay = 0.05)

## # weights: 149  
## initial value 94.853354   
## iter 10 value 40.630240  
## iter 20 value 25.041876  
## iter 30 value 22.526756  
## iter 40 value 21.682599  
## iter 50 value 21.373516  
## iter 60 value 21.341268  
## iter 70 value 21.340207  
## final value 21.340190   
## converged

pacientes.prediccion.5neu.decay.softmax <- predict( pacientes.5neu.decay.softmax, matriz.pacientes.datos.centscal[conjuntoEntrenamiento, 1:24], type="raw" )  
head(pacientes.prediccion.5neu.decay.softmax) # Vemos las probabilidades de pertenencia de cada valor

## 1 2 3 4  
## 35 0.050224577 0.88892492 0.0572271127 0.0036233896  
## 7 0.094578278 0.02385735 0.0213395925 0.8602247787  
## 13 0.072960474 0.92632537 0.0006006058 0.0001135457  
## 8 0.314176778 0.63525652 0.0503871346 0.0001795640  
## 29 0.010075479 0.97792058 0.0079026846 0.0041012602  
## 64 0.001332565 0.97354150 0.0024318281 0.0226941050

pacientes.prediccion.5neu.decay.class.softmax <- apply( pacientes.prediccion.5neu.decay.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.5neu.decay.class.softmax

## 35 7 13 8 29 64 46 25 63 6 50 55 66 51 30 61 33 24 3 56 10 39 27 45 49 20 54 67 22 14 12 42 52   
## 2 4 2 2 2 2 2 1 3 4 2 1 2 4 2 2 2 4 2 1 2 2 4 2 1 2 3 1 3 1 1 1 1   
## 41 59 17 62 11 32 65 36 37 18 57 58 34 48 26 15 43 47 4 40 5 28   
## 3 4 4 1 2 3 4 2 2 2 3 1 2 2 3 3 1 4 4 1 2 4

table( pacientes.prediccion.5neu.decay.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) # Lo vemos en forma de tabla.

##   
## pacientes.prediccion.5neu.decay.class.softmax 1 2 3 4  
## 1 13 0 0 0  
## 2 1 22 0 0  
## 3 1 0 7 0  
## 4 0 0 0 11

sum( diag( table( pacientes.prediccion.5neu.decay.class.softmax, matriz.pacientes.etiquetas[conjuntoEntrenamiento, 25] ) ) )/55 # Esta cuenta nos da el índice de acierto

## [1] 0.9636364

TEST

pacientes.prediccion.test.decay.5neu.softmax <- predict( pacientes.5neu.decay.softmax, matriz.pacientes.datos.centscal[-conjuntoEntrenamiento, 1:24], type="raw" )  
pacientes.prediccion.test.decay.5neu.softmax

## 1 2 3 4  
## 1 0.0045613455 0.004778606 0.4499968157 5.406632e-01  
## 2 0.6265411775 0.371774855 0.0002963616 1.387606e-03  
## 9 0.0017411664 0.967016717 0.0084754757 2.276664e-02  
## 16 0.5313489055 0.041115487 0.1579253190 2.696103e-01  
## 19 0.0193597065 0.906991842 0.0395072178 3.414123e-02  
## 21 0.0006523018 0.982613979 0.0016215893 1.511213e-02  
## 23 0.9816112728 0.018275460 0.0001038186 9.448468e-06  
## 31 0.7851785092 0.208958895 0.0007354453 5.127151e-03  
## 38 0.9546200279 0.003824067 0.0148988018 2.665710e-02  
## 44 0.9645622736 0.009482156 0.0069392853 1.901628e-02  
## 53 0.3222326345 0.017548702 0.4385367310 2.216819e-01  
## 60 0.4918261895 0.237846637 0.2284746848 4.185249e-02

pacientes.prediccion.test.decay.5neu.class.softmax <- apply( pacientes.prediccion.test.decay.5neu.softmax, MARGIN=1, FUN='which.is.max')  
pacientes.prediccion.test.decay.5neu.class.softmax

## 1 2 9 16 19 21 23 31 38 44 53 60   
## 4 1 2 1 2 2 1 1 1 1 3 1

table( pacientes.prediccion.test.decay.5neu.class.softmax , matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] )

##   
## pacientes.prediccion.test.decay.5neu.class.softmax 1 2 3 4  
## 1 1 4 1 1  
## 2 0 3 0 0  
## 3 0 1 0 0  
## 4 1 0 0 0

sum( diag( table( pacientes.prediccion.test.decay.5neu.class.softmax, matriz.pacientes.etiquetas[-conjuntoEntrenamiento, 25] ) ) )/12

## [1] 0.3333333

# Obtención de Resultados de Perceptrón

Importo los datos:

dataset.resultados <- read.csv2("C:/Users/jorge/Desktop/Documentos Clase/Universidad/4ºCarrera/1er Cuatrimestre/Inteligencia Artificial/Trabajo Fin de Asignatura/Resultados.txt")

Ahora voy a sacar un gráfico interactivo donde comparo los resultados.

#install.packages("plotly")  
library("plotly")

##   
## Attaching package: 'plotly'

## The following objects are masked from 'package:plyr':  
##   
## arrange, mutate, rename, summarise

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

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

## The following object is masked from 'package:graphics':  
##   
## layout

tipos = dataset.resultados[, 1]  
real = dataset.resultados[, 2]  
practico = dataset.resultados[, 3]  
  
p<- plot\_ly(dataset.resultados, x = ~tipos, y = ~real, type = 'bar', name = 'Real') %>% add\_trace(y = ~practico, name = 'Práctico') %>% layout(yaxis = list(title = 'Porcentaje'), barmode = 'group')  
  
p

## PhantomJS not found. You can install it with webshot::install\_phantomjs(). If it is installed, please make sure the phantomjs executable can be found via the PATH variable.

#Mostramos el gráfico interactivo

Ahora que hemos sacado los resultados obtenidos con el perceptrón multicapa, vamos con otras técnicas supervisadas:

## KNN

Instalamos la librería class:

#install.packages("class")  
library("class")

## Warning: package 'class' was built under R version 3.5.2

Hacemos nuevos conjuntos:

# Para hacer la predicción con knn, voy a coger los grupos de una manera distinta:  
  
conjuntoEntrenamiento = matriz.pacientes.datos.centscal[1:55, 1:24]  
conjuntoTest = matriz.pacientes.datos.centscal[56:67, 1:24]  
# Utilizo por supuesto la matriz de centrado y escalado  
  
etiquetasEntrenamiento = matriz.pacientes.etiquetas[1:55, 25]  
etiquetasTest = matriz.pacientes.etiquetas[56:67, 25]

Si quisiéramos mostrar los conjuntos de entrenamiento y de test…

conjuntoEntrenamiento  
conjuntoTest  
etiquetasEntrenamiento  
etiquetasTest

Para sacar los resultados de una manera visual, importamos lo siguiente:

#install.packages("gmodels")  
library("gmodels")

Comenzamos las pruebas. Como sabemos, normalmente el mejor valor de K para KNN suele ser el valor que más se acerque a la raíz cuadrada del total de los valores. Por eso, empezaremos por K = 8:

### Para K = 8…

prediccion.knn.8 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 8)  
prediccion.knn.8

## [1] 1 2 2 1 2 1 2 2 2 1 2 2  
## Levels: 1 2 3 4

Sacamos crosstable:

CrossTable(x = etiquetasTest , y = prediccion.knn.8, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 12   
##   
##   
## | prediccion.knn.8   
## etiquetasTest | 1 | 2 | Row Total |   
## --------------|-----------|-----------|-----------|  
## 1 | 1 | 3 | 4 |   
## | 0.250 | 0.750 | 0.333 |   
## | 0.250 | 0.375 | |   
## | 0.083 | 0.250 | |   
## --------------|-----------|-----------|-----------|  
## 2 | 1 | 3 | 4 |   
## | 0.250 | 0.750 | 0.333 |   
## | 0.250 | 0.375 | |   
## | 0.083 | 0.250 | |   
## --------------|-----------|-----------|-----------|  
## 3 | 0 | 2 | 2 |   
## | 0.000 | 1.000 | 0.167 |   
## | 0.000 | 0.250 | |   
## | 0.000 | 0.167 | |   
## --------------|-----------|-----------|-----------|  
## 4 | 2 | 0 | 2 |   
## | 1.000 | 0.000 | 0.167 |   
## | 0.500 | 0.000 | |   
## | 0.167 | 0.000 | |   
## --------------|-----------|-----------|-----------|  
## Column Total | 4 | 8 | 12 |   
## | 0.333 | 0.667 | |   
## --------------|-----------|-----------|-----------|  
##   
##

### Para K = 6

prediccion.knn.6 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 6)  
prediccion.knn.6

## [1] 1 2 2 1 2 1 2 2 2 1 2 2  
## Levels: 1 2 3 4

Obtenemos la crosstable:

CrossTable(x = etiquetasTest , y = prediccion.knn.6, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 12   
##   
##   
## | prediccion.knn.6   
## etiquetasTest | 1 | 2 | Row Total |   
## --------------|-----------|-----------|-----------|  
## 1 | 1 | 3 | 4 |   
## | 0.250 | 0.750 | 0.333 |   
## | 0.250 | 0.375 | |   
## | 0.083 | 0.250 | |   
## --------------|-----------|-----------|-----------|  
## 2 | 1 | 3 | 4 |   
## | 0.250 | 0.750 | 0.333 |   
## | 0.250 | 0.375 | |   
## | 0.083 | 0.250 | |   
## --------------|-----------|-----------|-----------|  
## 3 | 0 | 2 | 2 |   
## | 0.000 | 1.000 | 0.167 |   
## | 0.000 | 0.250 | |   
## | 0.000 | 0.167 | |   
## --------------|-----------|-----------|-----------|  
## 4 | 2 | 0 | 2 |   
## | 1.000 | 0.000 | 0.167 |   
## | 0.500 | 0.000 | |   
## | 0.167 | 0.000 | |   
## --------------|-----------|-----------|-----------|  
## Column Total | 4 | 8 | 12 |   
## | 0.333 | 0.667 | |   
## --------------|-----------|-----------|-----------|  
##   
##

### Para k = 10

prediccion.knn.10 <- knn(train = conjuntoEntrenamiento, test = conjuntoTest, cl = etiquetasEntrenamiento, k = 10)  
prediccion.knn.10

## [1] 1 2 2 1 2 1 2 2 4 1 2 2  
## Levels: 1 2 3 4

Obtenemos la crosstable:

CrossTable(x = etiquetasTest , y = prediccion.knn.10, prop.chisq = FALSE)

##   
##   
## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Row Total |  
## | N / Col Total |  
## | N / Table Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 12   
##   
##   
## | prediccion.knn.10   
## etiquetasTest | 1 | 2 | 4 | Row Total |   
## --------------|-----------|-----------|-----------|-----------|  
## 1 | 1 | 3 | 0 | 4 |   
## | 0.250 | 0.750 | 0.000 | 0.333 |   
## | 0.250 | 0.429 | 0.000 | |   
## | 0.083 | 0.250 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 2 | 1 | 2 | 1 | 4 |   
## | 0.250 | 0.500 | 0.250 | 0.333 |   
## | 0.250 | 0.286 | 1.000 | |   
## | 0.083 | 0.167 | 0.083 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 3 | 0 | 2 | 0 | 2 |   
## | 0.000 | 1.000 | 0.000 | 0.167 |   
## | 0.000 | 0.286 | 0.000 | |   
## | 0.000 | 0.167 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## 4 | 2 | 0 | 0 | 2 |   
## | 1.000 | 0.000 | 0.000 | 0.167 |   
## | 0.500 | 0.000 | 0.000 | |   
## | 0.167 | 0.000 | 0.000 | |   
## --------------|-----------|-----------|-----------|-----------|  
## Column Total | 4 | 7 | 1 | 12 |   
## | 0.333 | 0.583 | 0.083 | |   
## --------------|-----------|-----------|-----------|-----------|  
##   
##

### Como se puede observar, la mejor predicción la hemos hecho con K = 8

## Random Forest

Ahora voy a implementar una solución mediante Random Forest:

Lo primero que hacemos es importar el paquete de Random Forest

#install.packages("randomForest")  
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.2

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

set.seed(1000) #Pongo una seed para reproducibilidad

Una vez instalado e importado, lo que tengo que hacer es crear el Random Forest, y ejecutarlo…

model <- randomForest(grupo ~ ., data = dataset[, 2:26], importance = TRUE, ntree = 450)

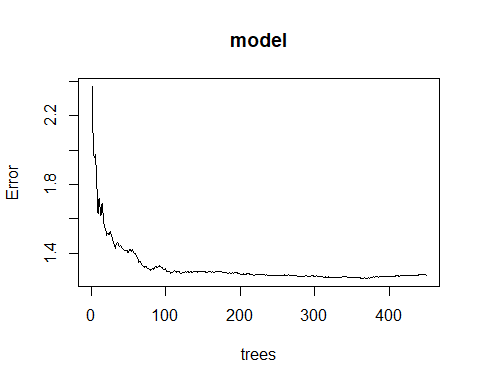
## Warning in randomForest.default(m, y, ...): The response has five or fewer unique values. Are you  
## sure you want to do regression?

model

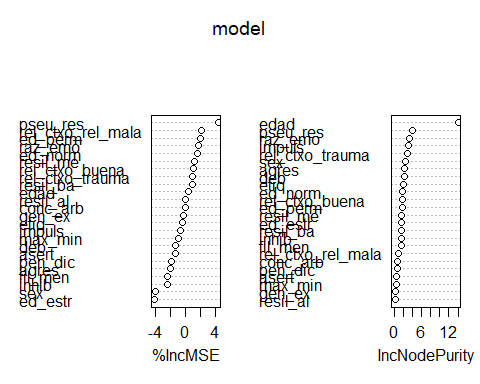
##   
## Call:  
## randomForest(formula = grupo ~ ., data = dataset[, 2:26], importance = TRUE, ntree = 450)   
## Type of random forest: regression  
## Number of trees: 450  
## No. of variables tried at each split: 8  
##   
## Mean of squared residuals: 1.274016  
## % Var explained: -22.57

Ahora obtenemos el número de árboles que necesitamos realmente, y la importancia de las variables en este modelo:

plot(model)



varImpPlot(model) # Gracias a importance = true

 Ahora voy a buscar los valores óptimos:

which.min(model$mse) # Esta función me dice con qué cantidad de árboles tengo el mínimo MSE

## [1] 364

Ahora lo voy a hacer con 10 fold X Validation:

result <- rfcv(dataset[, 2:26], dataset$grupo, cv.fold=10)

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, , drop = FALSE], trainy[idx != : The response has  
## five or fewer unique values. Are you sure you want to do regression?

## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?  
  
## Warning in randomForest.default(trainx[idx != i, imp.idx, drop = FALSE], : The response has five or  
## fewer unique values. Are you sure you want to do regression?

head(result)

## $n.var  
## [1] 25 12 6 3 1  
##   
## $error.cv  
## 25 12 6 3 1   
## 3.319396e-01 2.716757e-01 1.700439e-01 2.837366e-01 5.970149e-08   
##   
## $predicted  
## $predicted$`25`  
## [1] 1.993913 1.992933 2.020184 3.363310 1.905232 3.343434 3.054079 1.612227 2.073325 2.177215  
## [11] 2.032144 1.783364 1.761988 2.532818 1.903002 3.342102 2.924512 1.821971 2.109888 1.999068  
## [21] 2.091529 2.695563 1.596475 2.879952 1.739348 2.519102 2.788335 2.971437 2.126533 1.953144  
## [31] 1.808795 3.252565 1.878313 1.772235 2.001307 1.712775 2.013070 2.956271 1.898763 1.310238  
## [41] 2.780433 1.665579 1.358530 1.914749 2.018933 2.115638 3.265650 1.885177 1.861808 2.078862  
## [51] 2.823874 1.538091 2.048812 3.022628 1.908127 1.545318 2.798718 1.635731 2.943768 2.112265  
## [61] 1.800125 1.822014 2.730315 2.163588 3.147196 2.011407 1.619086  
##   
## $predicted$`12`  
## [1] 1.709995 1.956950 2.015193 3.440359 1.958112 3.473471 3.032968 1.602529 2.073045 2.128853  
## [11] 1.926740 1.749157 1.824326 2.409255 1.731112 3.505878 2.940621 1.803618 2.086426 2.003724  
## [21] 2.011963 2.655634 1.640717 2.991330 1.659466 2.565823 2.773655 3.148435 2.008919 2.020365  
## [31] 1.836467 3.258050 1.878895 1.844532 2.014800 1.758636 1.996487 3.043573 2.004080 1.309799  
## [41] 2.799811 1.445589 1.420034 1.838659 1.878004 2.101720 3.411599 1.949641 1.582048 2.017598  
## [51] 2.989975 1.305952 1.919479 3.028261 1.743228 1.529141 2.798607 1.775220 2.856588 2.061387  
## [61] 1.666717 1.666561 2.894403 2.326226 3.257801 1.863595 1.512275  
##   
## $predicted$`6`  
## [1] 1.581944 1.947100 2.068890 3.557709 1.868991 3.548561 3.333730 1.592866 2.117311 2.022979  
## [11] 1.833490 1.518219 1.815977 2.201090 1.705856 3.540774 3.296121 1.909515 1.975406 2.052500  
## [21] 2.003345 2.676291 1.534018 2.848940 1.474212 2.832342 3.524832 3.326607 2.127145 1.975406  
## [31] 1.849392 3.209098 1.898669 1.877851 1.950933 1.912061 1.950933 3.125555 2.019295 1.318627  
## [41] 2.923891 1.371492 1.371492 1.680691 1.906288 1.916684 3.677957 2.001872 1.387722 1.902593  
## [51] 3.103785 1.385533 2.001872 2.824331 1.562654 1.487184 2.770162 1.487184 3.319743 2.052829  
## [61] 1.743882 1.547959 2.813965 1.980946 3.317444 1.835531 1.311050  
##   
## $predicted$`3`  
## [1] 1.761616 1.945078 2.089546 3.464108 1.913485 3.290192 3.118778 1.737515 2.027963 1.992519  
## [11] 1.843369 1.700665 1.836772 2.608658 1.737905 3.341112 2.817290 1.869232 2.012031 2.045992  
## [21] 2.007803 2.816391 1.737515 2.679074 1.461020 2.820341 3.312161 3.123645 2.056795 2.012031  
## [31] 1.765664 3.080258 1.848597 1.920581 1.996053 1.913032 1.996053 2.922194 2.027963 1.461020  
## [41] 2.954493 1.652053 1.652053 1.765664 1.874456 1.931836 3.276952 1.972680 1.691276 1.913485  
## [51] 3.055519 1.694666 1.972680 2.724928 1.698051 1.625836 2.816391 1.625836 3.140082 1.869232  
## [61] 1.920581 1.673087 2.885527 2.089546 3.413341 1.893312 1.598987  
##   
## $predicted$`1`  
## [1] 1.000 2.000 2.000 4.000 2.000 4.000 4.000 1.000 2.000 2.000 2.000 1.000 2.000 1.000 1.000 4.000  
## [17] 4.000 2.000 2.000 2.000 2.000 3.000 1.000 4.000 1.000 3.000 4.000 4.000 2.000 2.000 2.000 3.000  
## [33] 2.000 2.000 2.000 2.000 2.000 3.000 2.000 1.000 3.000 1.000 1.000 2.000 2.000 2.000 4.000 2.000  
## [49] 1.000 2.000 4.000 1.000 2.000 2.998 1.000 1.000 3.000 1.000 4.000 2.000 2.000 1.000 3.000 2.000  
## [65] 4.000 2.000 1.000

Podemos ver el error, bajo la variable $error.cv, y podemos ver las predicciones que se han hecho para cada una de las n.var.

## SVM de Kernel Lineal

Lo bueno que tiene SVM es que es muy robusto frente a la dimensión, por lo que deberíamos de obtener a priori buenos resultados con este método.

Lo primero que hay que hacer es importar la librería…

#install.packages("e1071")  
library("e1071")

## Warning: package 'e1071' was built under R version 3.5.2

Con este método no necesito tener un conjunto de entrenaminento y otro de test, por lo que sigo adelante.

Ahora que hemos instalado la librería, vamos a crear el SVM:

modelo.svm <- svm(matriz.pacientes.datos.centscal, dataset[, 26])  
summary(modelo.svm)

##   
## Call:  
## svm.default(x = matriz.pacientes.datos.centscal, y = dataset[, 26])  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.04166667   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 60

Ahora que tenemos creado este primer modelo, toca predecir:

prediccion <- predict(modelo.svm, matriz.pacientes.datos.centscal)  
prediccion

## 1 2 3 4 5 6 7 8 9 10 11   
## 1.549973 1.897491 2.102992 2.931869 1.958610 2.933492 2.460079 1.968849 2.102839 2.102609 2.102858   
## 12 13 14 15 16 17 18 19 20 21 22   
## 1.834806 1.897158 1.792309 2.081453 3.345387 2.644334 1.896957 2.102818 2.101751 2.102659 2.081453   
## 23 24 25 26 27 28 29 30 31 32 33   
## 1.102789 2.171444 1.688974 2.247323 3.098435 2.310026 2.102645 2.090946 1.897066 2.757770 2.102934   
## 34 35 36 37 38 39 40 41 42 43 44   
## 1.935061 2.103190 1.897447 2.078152 2.427070 1.897235 1.396473 2.724358 1.577114 1.558528 1.911562   
## 45 46 47 48 49 50 51 52 53 54 55   
## 2.102722 2.093837 2.558446 2.102649 1.964595 2.102263 1.784159 1.755280 2.102747 2.897155 1.865630   
## 56 57 58 59 60 61 62 63 64 65 66   
## 1.404453 2.647560 1.341993 2.806776 2.103103 2.102609 1.945673 2.749015 2.102654 3.127086 2.102353   
## 67   
## 1.482761

El problema que tenemos con estas predicciones es que están siendo contínuas, y no discretas, por lo que las voy a discretizar redondeando:

prediccion <- round(prediccion, digits = 0) #No quiero dígitos decimales  
prediccion

## 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 33   
## 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 1 2 2 2 3 2 2 2 2 3 2   
## 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66   
## 2 2 2 2 2 2 1 3 2 2 2 2 2 3 2 2 2 2 2 2 3 2 1 3 1 3 2 2 2 3 2 3 2   
## 67   
## 1

Ahora que hemos predicho, tenemos que sacar la matriz de confusión:

matriz.conf <- table(prediccion, dataset[ ,26])  
matriz.conf

##   
## prediccion 1 2 3 4  
## 1 5 0 0 0  
## 2 12 30 3 4  
## 3 0 0 5 8

sum(diag(matriz.conf))/67

## [1] 0.5970149

Obtenemos un porcentaje de acierto medio, pero esto es sin tener en cuenta que los pacientes del grupo 2 pueden pertenecer al 1, lo cual suma alrededor de 10 pacientes más, por lo que obtendríamos valores mucho más altos que rondarían el 65-70% de acierto.

## SVM de Kernel RBF

modelo\_svm.radial <- svm(grupo ~ ., data=dataset[2:26], kernel="radial")  
summary(modelo\_svm.radial)

##   
## Call:  
## svm(formula = grupo ~ ., data = dataset[2:26], kernel = "radial")  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.04166667   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 60

Ahora que tenemos creado este primer modelo, toca predecir:

prediccion.radial <- predict(modelo\_svm.radial,dataset[,2:25])  
prediccion.radial

## 1 2 3 4 5 6 7 8 9 10 11   
## 1.549973 1.897491 2.102992 2.931869 1.958610 2.933492 2.460079 1.968849 2.102839 2.102609 2.102858   
## 12 13 14 15 16 17 18 19 20 21 22   
## 1.834806 1.897158 1.792309 2.081453 3.345387 2.644334 1.896957 2.102818 2.101751 2.102659 2.081453   
## 23 24 25 26 27 28 29 30 31 32 33   
## 1.102789 2.171444 1.688974 2.247323 3.098435 2.310026 2.102645 2.090946 1.897066 2.757770 2.102934   
## 34 35 36 37 38 39 40 41 42 43 44   
## 1.935061 2.103190 1.897447 2.078152 2.427070 1.897235 1.396473 2.724358 1.577114 1.558528 1.911562   
## 45 46 47 48 49 50 51 52 53 54 55   
## 2.102722 2.093837 2.558446 2.102649 1.964595 2.102263 1.784159 1.755280 2.102747 2.897155 1.865630   
## 56 57 58 59 60 61 62 63 64 65 66   
## 1.404453 2.647560 1.341993 2.806776 2.103103 2.102609 1.945673 2.749015 2.102654 3.127086 2.102353   
## 67   
## 1.482761

El problema que tenemos con estas predicciones es que están siendo contínuas, y no discretas, por lo que las voy a discretizar redondeando:

prediccion.radial <- round(prediccion.radial, digits = 0)  
prediccion.radial

## 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 33   
## 2 2 2 3 2 3 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 1 2 2 2 3 2 2 2 2 3 2   
## 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66   
## 2 2 2 2 2 2 1 3 2 2 2 2 2 3 2 2 2 2 2 2 3 2 1 3 1 3 2 2 2 3 2 3 2   
## 67   
## 1

Ahora que hemos predicho, tenemos que sacar la matriz de confusión:

matriz.conf.radial <- table(prediccion.radial, dataset[,26])  
matriz.conf.radial

##   
## prediccion.radial 1 2 3 4  
## 1 5 0 0 0  
## 2 12 30 3 4  
## 3 0 0 5 8

sum(diag(matriz.conf.radial))/67

## [1] 0.5970149

Obtenemos un acierto del 60%, al que hay que sumar otros 12 pacientes. ### Si añadimos estos pacientes, nos encontramos con un acierto del 77% Por lo tanto, SVM de Kernel radial es una buena técnica para la predicción en este problema.

Ahora pasamos a los modelos de inteligencia artificial no supervisados:

# Modelos de inteligencia artificial no supervisados

El primer modelo de inteligencia artificial no supervisado que voy a usar es un modelo de clustering llamado Dendrograma.

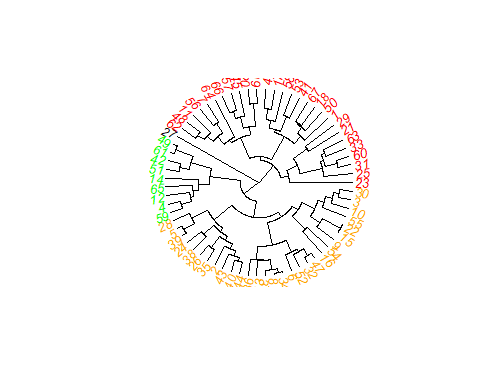
## Dendrograma

Para esto, lo que voy a hacer es dividirlo en 4 clusters, coincidiendo con los 4 grupos de trastornos que tengo.

#install.packages("ape")  
library(ape)

## Warning: package 'ape' was built under R version 3.5.2

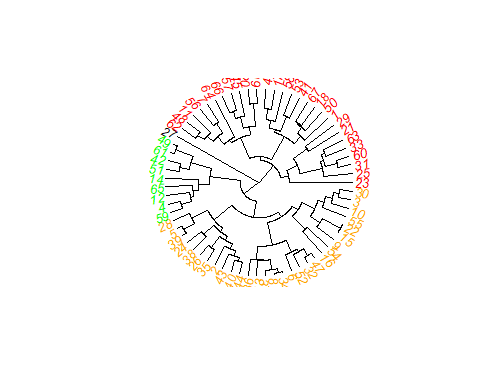
dd <- dist(scale(dataset[,2:25]), method = "euclidean") #Nos basamos en la distancia euclídea  
hier.clust <- hclust(dd, method = "ward.D2")  
colores.dendrograma <- c("red", "orange", "green", "black") # Creamos los colores con los que queremos el cluster  
cluster.4 <- cutree(hier.clust, 4) # Cluster jerárquico de 4...  
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3, cex = 0.8) #Lo pintamos



Como vemos, estamos obteniendo el indentificador de cada paciente en el dendrograma, donde los pacientes que mas se parecen estarán más juntos, mientras que los que menos se parecen estarán más separados. Es interesante analizar como los pacientes verdes y los naranjas surgen de la misma salida del centro, cosa que no ocurre con los rojos y los negros, lo cual quiere decir que algo tienen en común estos dos tipos de casos.

Ahora voy a hacer el mismo dendrograma pero con el DataSet de centrado y escalado, de tal manera que veamos a ver si hay diferencias:

dd <- dist(scale(matriz.pacientes.datos.centscal), method = "euclidean")  
hier.clust <- hclust(dd, method = "ward.D2")  
colores.dendrograma <- c("red", "orange", "green", "black")  
cluster.4 <- cutree(hier.clust, 4)  
plot(as.phylo(hier.clust), type = "fan", tip.color = colores.dendrograma[cluster.4], label.offset = 0.3, cex = 0.8)



Si lo comparamos, vemos que hemos obtenido exactamente el mismo resultado, por lo que en este caso el centrado y escalado no es necesario.

Vamos a analizar algunos pacientes aleatoriamente para ver si ha acertado, o al menos si se ha aproximado:

* Paciente 1: Analizado - Rojo ——— Real: 1
* Paciente 2: Analizado - Naranja —— Real: 2
* Paciente 3: Analizado - Naranja —— Real: 2
* Paciente 4: Analizado - Verde ——– Real: 4
* Paciente 5: Analizado - Naranja —— Real: 2
* Paciente 27: Analizado - Negro ——- Real: 4

Es decir, a la vista de estos resultados, podemos concluir que el grupo 1 es el de los pacientes en rojo, el grupo 2 es el de los pacientes en naranja, y luego entre el grupo 3 y el grupo 4 hay dudas, pero teniendo varios pacientes tanto del grupo 3 como del grupo 4 en nuestro DataSet parece ser que algo de error ha cometido.

## K-Means

El algoritmo KMeans en principio no es el algoritmo más adecuado para este trabajo, ya que se basa en círculos para la clasificación de los individuos, cuando en principio en mis datos esto no es así. De todas formas, voy a clasificar a los pacientes siguiendo este algoritmo para comprobar la eficacia que tiene sobre mi problema:

#install.packages("cluster")  
#install.packages("fpc")  
  
library(cluster)  
library(fpc)

## Warning: package 'fpc' was built under R version 3.5.2

Hacemos el clustering y vemos algunos resultados:

datos.kmeans <- matriz.pacientes.datos # Sin la clasificación dentro del dataset  
  
clusters <- kmeans(datos.kmeans, centers=4)  
clusters$centers

## edad sex rel\_ctxo\_rel\_mala rel\_ctxo\_trauma rel\_ctxo\_buena ed\_perm ed\_norm  
## 1 18.96667 0.1666667 0.10000000 0.3000000 0.6000000 0.43333333 0.3000000  
## 2 39.00000 0.5000000 0.33333333 0.3333333 0.3333333 0.50000000 0.3333333  
## 3 28.15385 0.1923077 0.07692308 0.4230769 0.5000000 0.03846154 0.7692308  
## 4 47.60000 0.2000000 0.40000000 0.4000000 0.2000000 0.40000000 0.4000000  
## ed\_estr resil\_ba resil\_me resil\_al pen\_dic gen\_ex etiq fil\_men max\_min  
## 1 0.2666667 0.7666667 0.2333333 0.0 0.9000000 0.9666667 0.8666667 0.7666667 0.9666667  
## 2 0.1666667 0.1666667 0.8333333 0.0 1.0000000 1.0000000 0.3333333 0.6666667 1.0000000  
## 3 0.1923077 0.4615385 0.5384615 0.0 0.8846154 0.9615385 0.7692308 0.8076923 0.9615385  
## 4 0.2000000 0.4000000 0.4000000 0.2 0.8000000 0.8000000 0.4000000 1.0000000 1.0000000  
## conc\_arb pseu\_res deb raz\_emo inhib asert agres impuls  
## 1 0.9666667 0.3666667 0.8666667 0.8333333 0.7333333 0.03333333 0.2333333 0.5333333  
## 2 1.0000000 0.3333333 1.0000000 0.6666667 0.5000000 0.16666667 0.3333333 0.8333333  
## 3 1.0000000 0.7307692 1.0000000 0.8076923 0.6153846 0.19230769 0.1923077 0.6538462  
## 4 1.0000000 0.4000000 1.0000000 0.6000000 0.6000000 0.40000000 0.0000000 0.6000000

clusters$cluster

## [1] 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 1 2 1 3 3 3 1 4 1 2 1 1 1 4 1 3 1 3 3 1 3 1 3 3 3 2 3 3 1 2 4 3 3  
## [49] 3 3 3 2 2 1 1 4 3 3 1 1 3 3 3 3 3 4 1

Interpretando estos resultados, obtenemos:

* El cluster 1 destaca por sexo más hacia masculino que otros, una relación contexto ciertamente buena, una educación permisiva, una resiliencia baja, maximización y minimización, razonamiento emocional, cierta inhibición y poca agresividad.
* El cluster 2 destaca por una edad mayor, es el cluster con mejor relación con el contexto, y suelen tener las personas de este cluster una educación normal. Destaca por una resiliencia media, pensamiento dicotómico, generalización excesiva, etiquetado, conclusiones arbitrarias, deberías, razonamiento emocional e inhibición.
* El cluster número 3 destaca por tener una edad aún más elevada, más ratio de personas del sexo femenino que ningún otro cluster, y tienen una relación con el contexto bastante variable. La educación de estas personas es principalmente normal, con una resiliencia que puede ser tanto baja como media. Destacan por el pensamiento dicotómico, generalización excesiva, poco etiquetado, maximización y minimización, filtro mental, conclusiones arbitrarias, pseudoresponsabilidad, deberías, y suelen ser bastante inhibidos e impulsivos.
* Finalmente, el cluster 4 destaca por ser el que tiene la edad más elevada y el ratio de sexo más masculino. La relación con el contexto de estos individuos clasificados en este grupo es principalmente de trauma, aunque también hay buenas y malas. La educación de estos individuos es principalmente permisiva, y la resiliciencia tiende a media. Destacan por la poca etiquetación que hacen, pero un gran fitro mental, conclusiones arbitrarias, poca pseudo-responsabilidad, muchos deberías, poco razonamiento emocional, y son principalmente inhibidos e impulsivos.

Ahora sacamos la gráfica para poder ver como los ha clasificado sobre dos componentes principales artificiales:

clusplot(datos.kmeans, clusters$cluster, color = TRUE, main = "Representación 2D con Clusplot", labels = 4, xlab = "Componente 1", ylab = "Componente 2") # Con las dos componentes principales que más explicación nos dan

