Métodos de clasificación y Modelos de regresión

María Isabel Chuya - Nataly Quintanilla

Métodos de clasificación

Veremos un resumen de todos los métodos que hemos visto incluyendo Knn y Naive Bayes. Tened en cuenta que es un método de clasificación multiclase con más de 2 niveles.

Cargamos librerías

```
library(ggplot2)
library(ggpubr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(glmnet) ## regresiones logisitcas
```

```
Loading required package: Matrix
```

```
Loaded glmnet 4.1-7
```

```r
library(caret) ### bayes y knn
```

```
Loading required package: lattice
```

```r
library(e1071) ## bayes
```

Cargamos datos

```r
# quitamos la primera columna
datos <- read.table("./yeast.data",header = F)[,-1]
```

Creamos las funciones que vamos a necesitar, es decir las funciones de transformación

```r
min.max.mean <- function(X) apply(X,2,function(x) (x-mean(x))/(max(x)-min(x)))
min.max.median <- function(X) apply(X,2,function(x) (x-median(x))/(max(x)-min(x)))
min.max <- function(X) apply(X,2,function(x) (x-min(x))/(max(x)-min(x)))
zscore <- function(X) apply(X,2,function(x) (x-mean(x))/sd(x))
l2 <- function(X) apply(X,2,function(x) x/sqrt(sum(x^2)))
```

Para hacer las transformaciones, solamente necesitamos las variables numéricas.

```r
datos <- as.data.frame(datos)
datos.numericos <- datos[, which(unlist(lapply(datos, is.numeric)))]
clase <- datos$V10 <- as.factor(datos$V10)
colnames(datos.numericos) <- paste0("Var", rep(1:8))
### procedemos a crear una lista con todas las transformaciones

datos.lista <- list(
  raw = bind_cols(datos.numericos,clase=clase),
  zscore = bind_cols(zscore(datos.numericos), clase = clase),
  l2 = bind_cols(l2(datos.numericos), clase = clase),
  media = bind_cols(min.max.mean(datos.numericos), clase = clase),
  mediana = bind_cols(min.max.median(datos.numericos), clase = clase),
  min_max = bind_cols(min.max(datos.numericos), clase = clase))
```

Descriptiva Gráfica

Al ser demasiadas variables, podemos realizar un `melt`

```
lista_graficos <- vector("list",length=length(datos.lista))
datos.melt <- lapply(datos.lista,reshape2::melt)
```

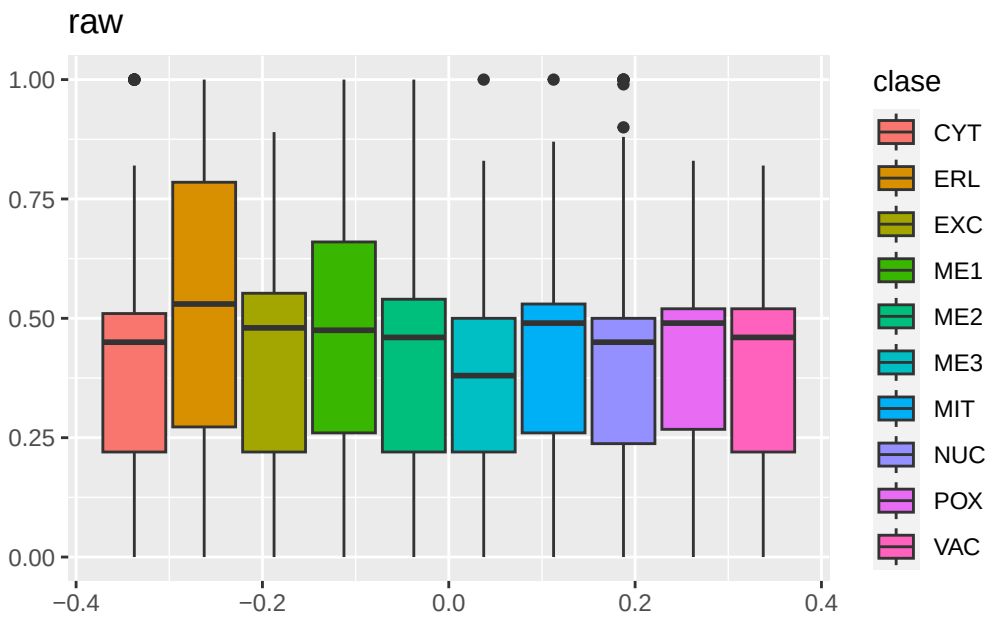
```
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
```

Podemos ver la cabecera de alguna transfomacion para ver el nombre nuevo de las variables

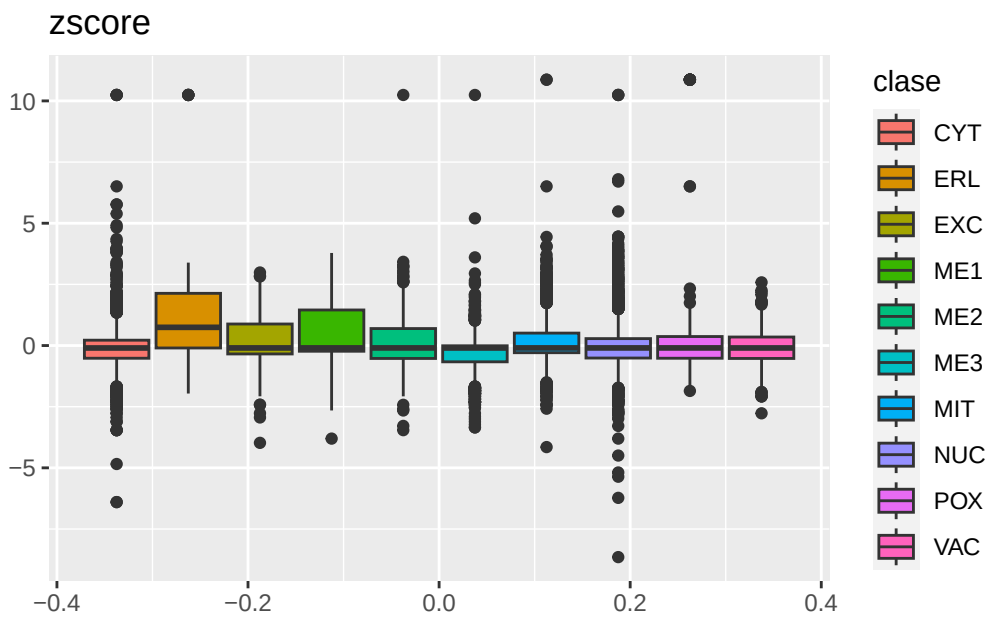
```
head(datos.melt$zscore)
```

```
  clase variable       value
1   MIT     Var1  0.58178524
2   MIT     Var1 -0.51071851
3   MIT     Var1  1.01878674
4   NUC     Var1  0.58178524
5   MIT     Var1 -0.58355209
6   CYT     Var1  0.07195016
```

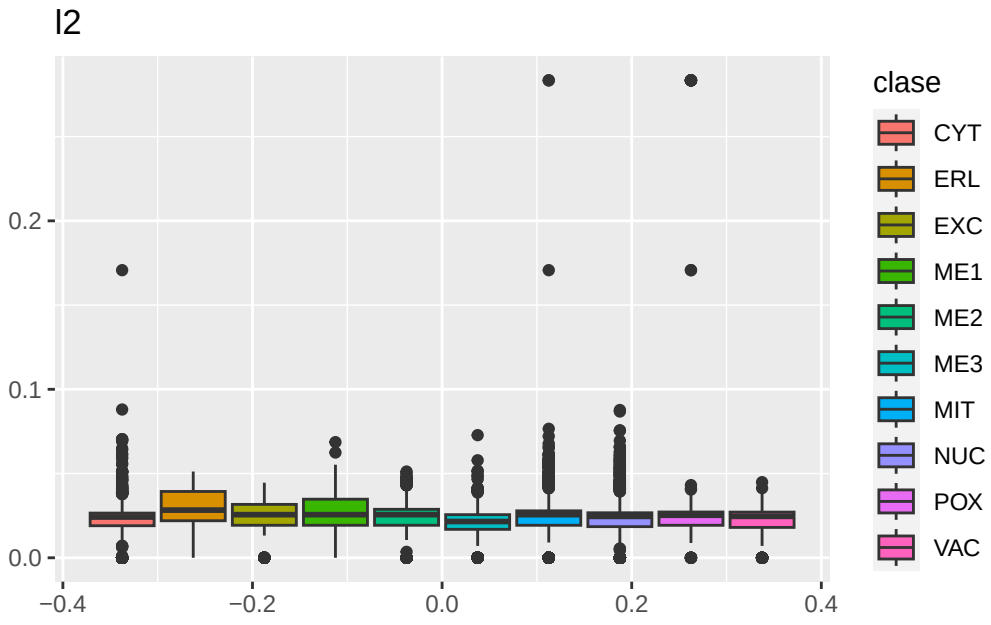
```
for(l in 1:length(datos.melt)){

  X <- datos.melt[[l]]
  nombre <- names(datos.melt)[l]
  lista_graficos[[l]] <- ggplot(X,aes(y=value,fill=clase))+geom_boxplot()+ggtitle(nombre)+


}

names(lista_graficos) <- paste0("plot",1:length(datos.lista))

lista_graficos$plot1
```



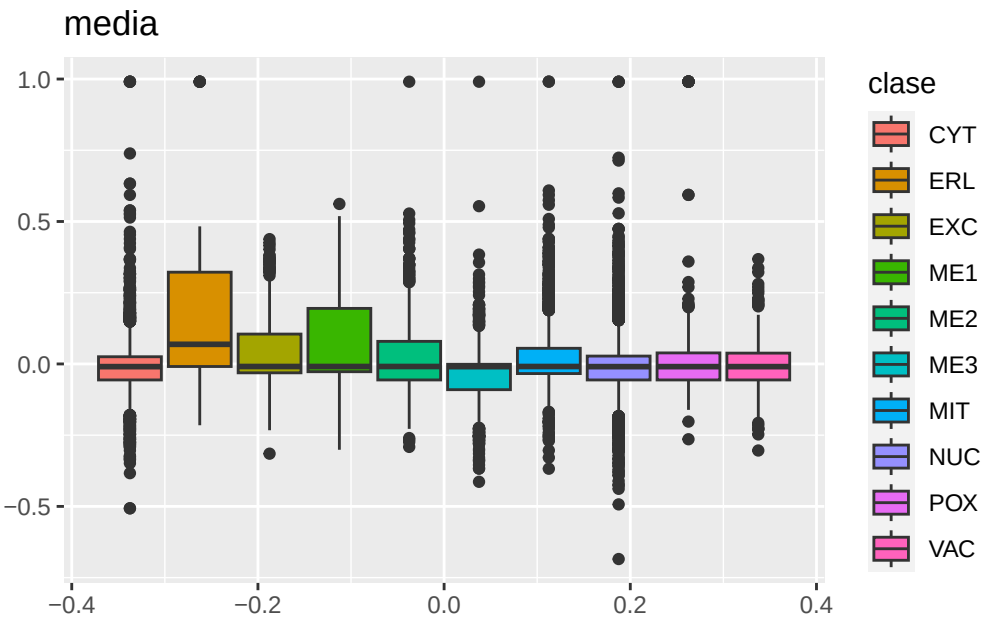

```
lista_graficos$plot2
```



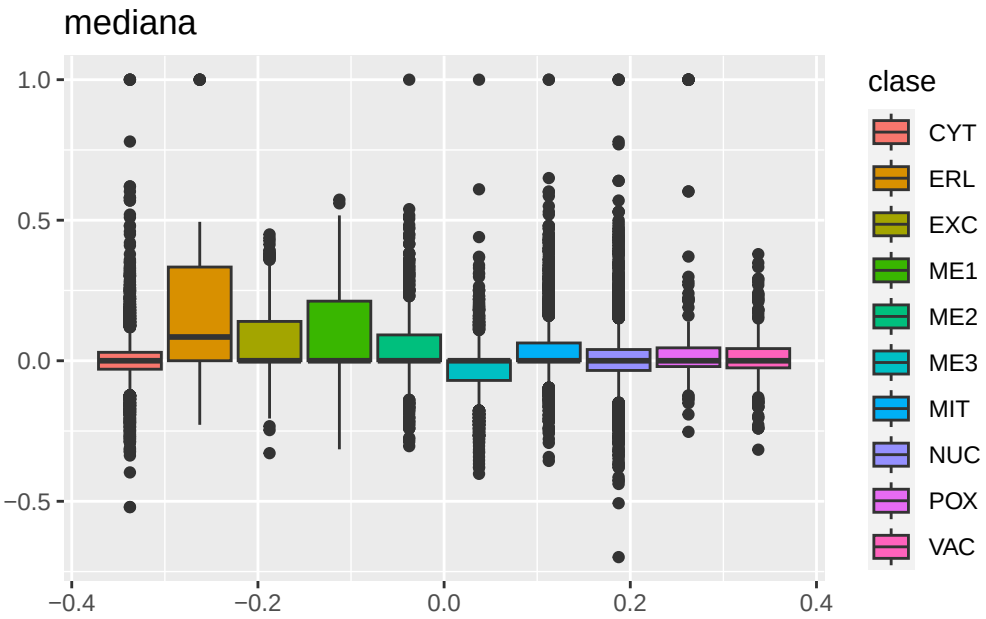
```
lista_graficos$plot3
```



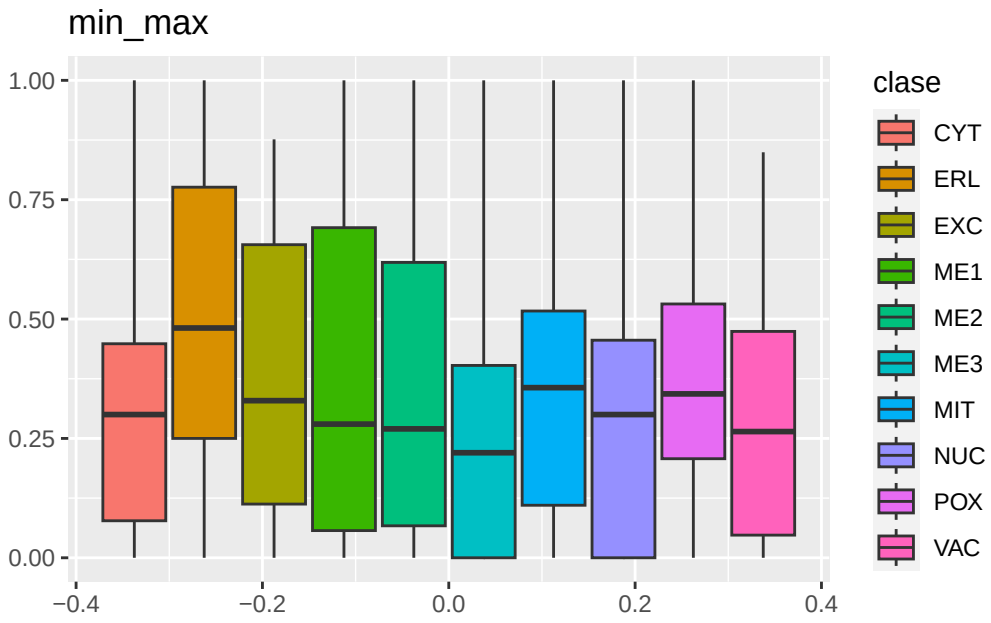

```
lista_graficos$plot4
```




```
lista_graficos$plot5
```



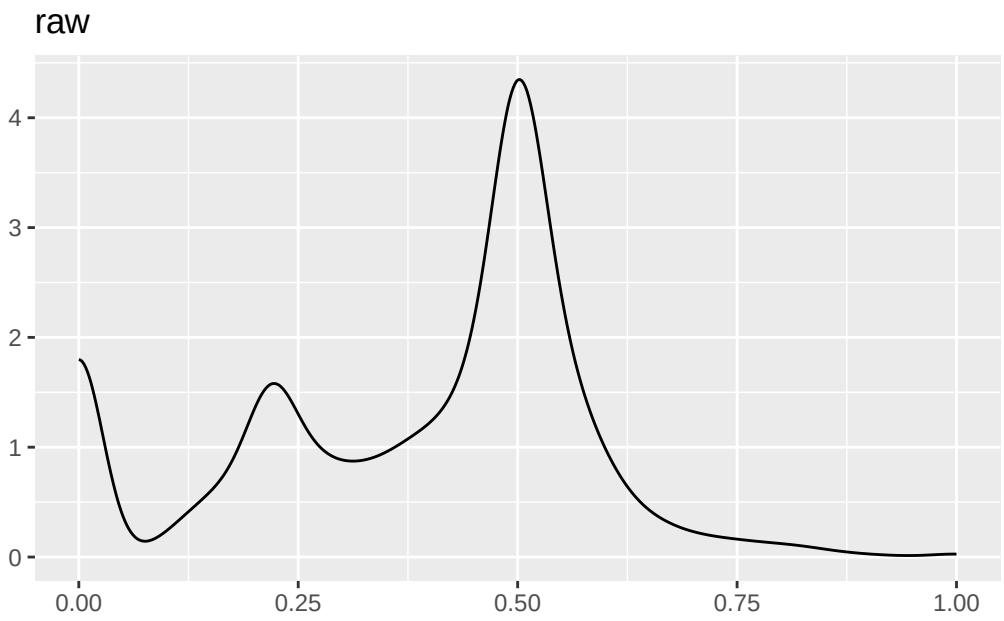
```
lista_graficos$plot6
```



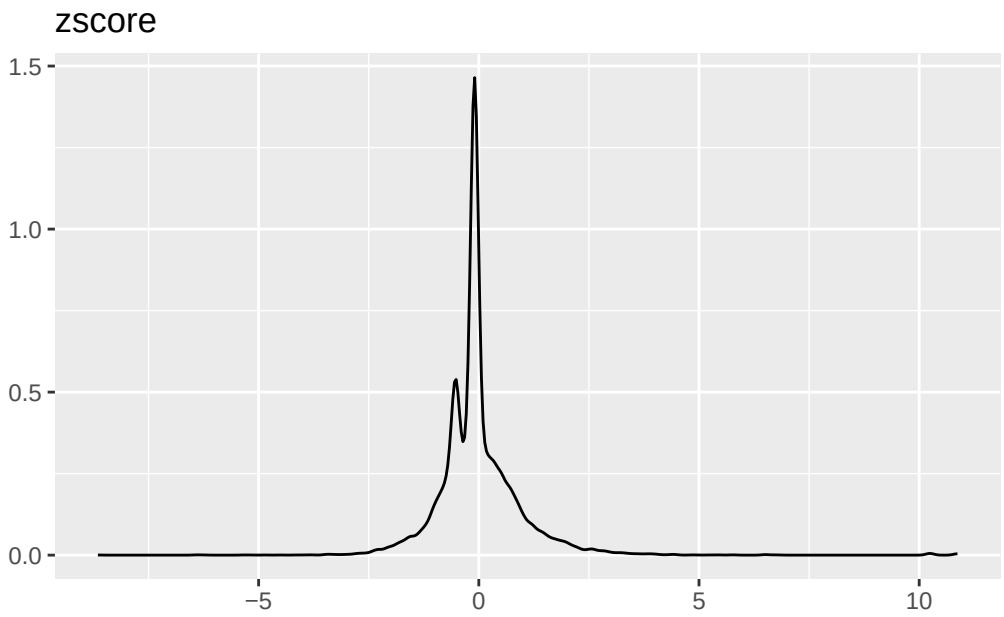
Así por ejemplo la normalización min-max es la mejor, puesto que no tenemos outliers

Otra forma de ver la transfomración es mediante gráficos de densidad

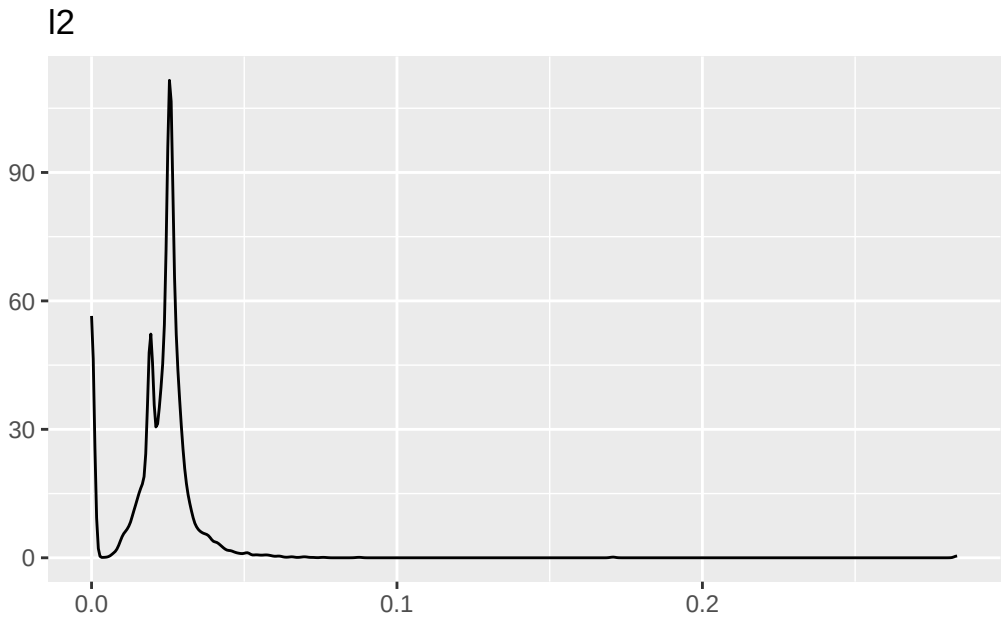
```
for(l in 1:length(datos.melt)){

  X <- datos.melt[[l]]
  nombre <- names(datos.melt)[l]
  lista_graficos[[l]] <- ggplot(X,aes(x=value))+geom_density()+ggtitle(nombre)+xlab("")+yl


}

names(lista_graficos) <- paste0("plot",1:length(datos.lista))

lista_graficos$plot1
```



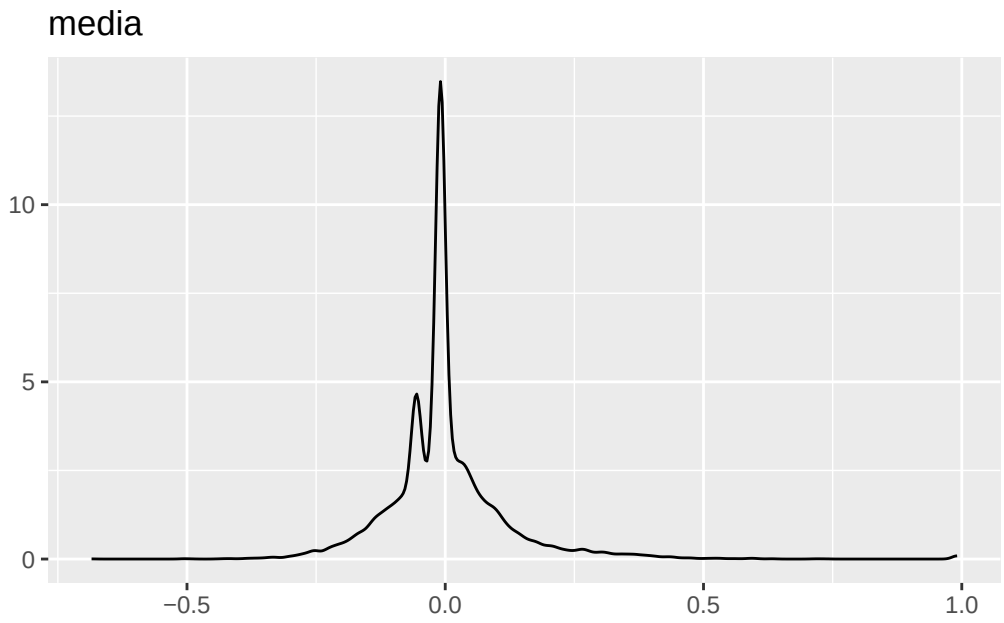

```
lista_graficos$plot2
```



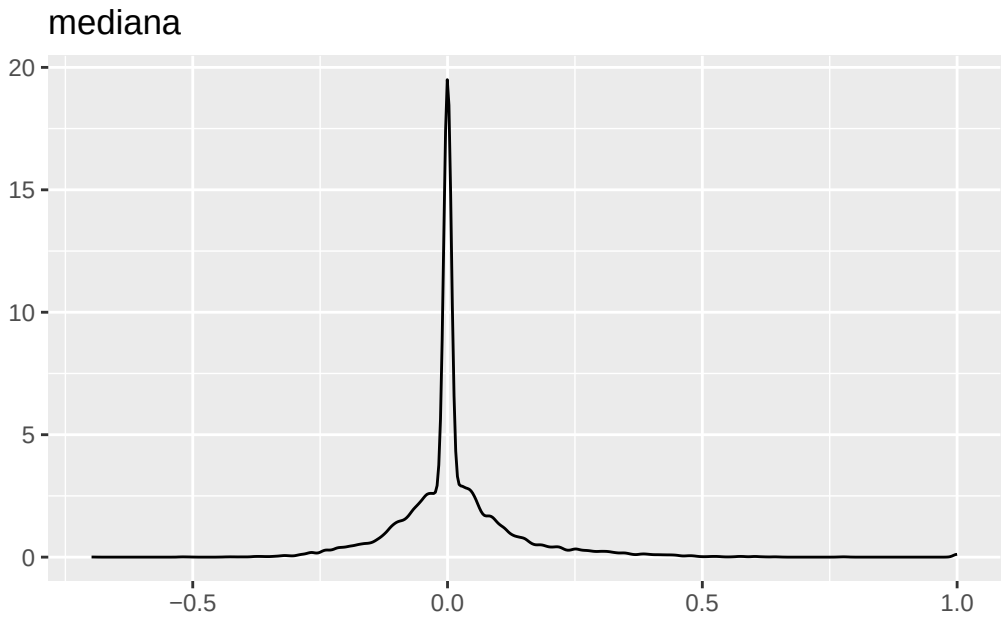
```
lista_graficos$plot3
```



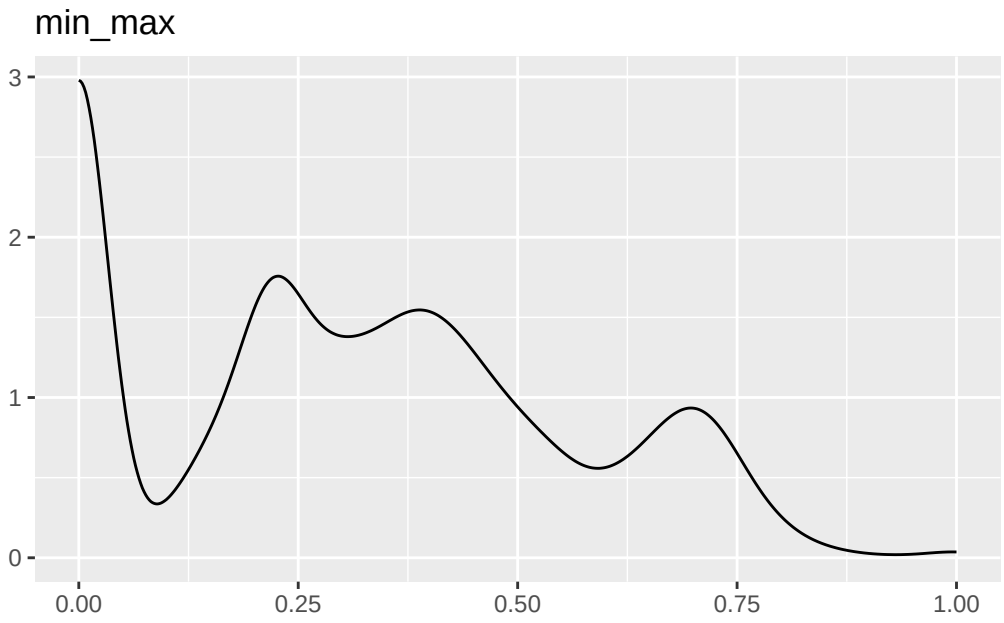

```
lista_graficos$plot4
```




```
lista_graficos$plot5
```

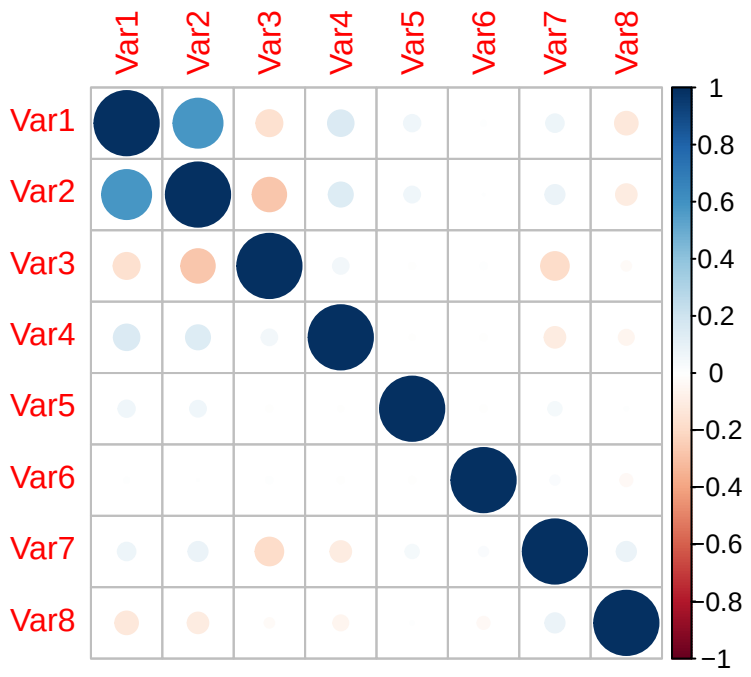


```
lista_graficos$plot6
```

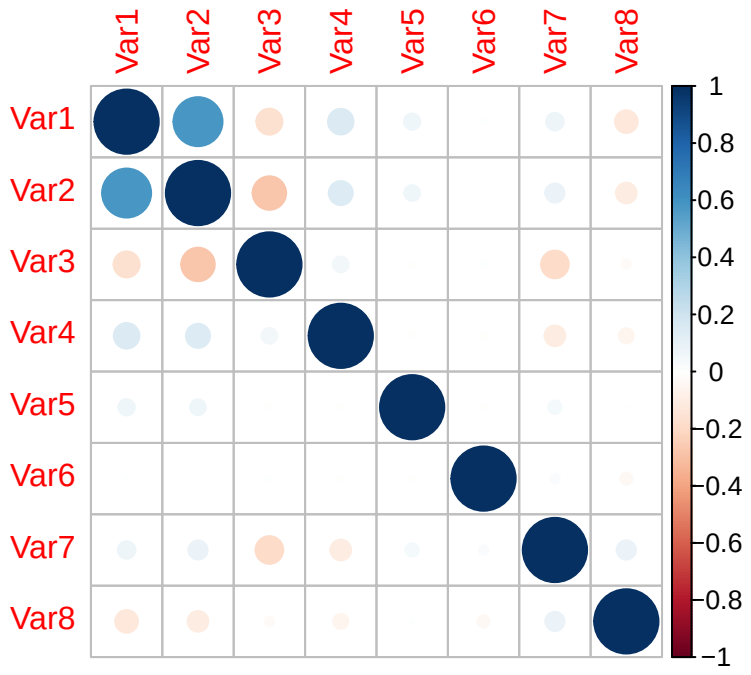


Sin embargo, al ver la densidad, no tenemos una transformacion uniforme.

```
corrplot::corrplot(cor(datos.numericos))
```




```
corrplot::corrplot(cor(datos.lista$media[,-ncol(datos)]))
```



Partición de datos

NOTA: PODEMOS CREAR LA PARTICIÓN CON `caret` o a mano, el 70 porciento de los datos. A mano sería

```
set.seed(123456789)
n  <- nrow(datos)
idx <- sample(1:n,n*0.7)
### para conjunto de datos podemos realizar el split
datos.train.lista <- lapply(datos.lista, function(x) x[idx,])
datos.test.lista <- lapply(datos.lista, function(x) x[-idx,])
```

Ejemplo regresión logística

https://rstudio-pubs-static.s3.amazonaws.com/38437_18a39a6487134d67b5f5e0d47221ec8d.html

https://rpubs.com/jkylearmstrong/logit_w_caret

alpha=1 es lasso y 0 es ridge

```
#Fijamos la semilla y la muestra
set.seed(123456789)
trControl <- trainControl(method = 'cv', number = 10)
n  <- nrow(datos)
idx <- sample(1:n,size=n*0.7,replace=F)
lambda_seq <- seq(0.01, 1, by = 0.01)

#Para conjunto de datos podemos realizar el split
entrenamiento <- lapply(datos.lista, function(x) x[idx,])
test <- lapply(datos.lista, function(x) x[-idx,])

#Regresion logistica Lineal
set.seed(123456789)
myfnlog <- function(x) train(clase ~ ., data = x, method = "multinom", trControl = trContr
logistica.lista <- lapply(entrenamiento,myfnlog)
logisita.pred <- vector("list",length = length(datos.lista))
for(l in 1:length(datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))
for(l in 1:length(datos.lista)){
```

```
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
accuracy_logis<-accuracy

#Ridge
set.seed(123456789)
myfnridge <- function(x) train(clase ~ ., data = x, method = "glmnet", trControl = trContr

logistica.lista <- lapply(entrenamiento,myfnridge)
logisita.pred <- vector("list",length = length(datos.lista))
for(l in 1:length( datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
ridgeaccuracy <- accuracy

# Regresion logistica de lasso
set.seed(123456789)
Dlasso <- function(x) train(clase ~ ., data = x, method = "glmnet", trControl = trControl,

logistica.lista <- lapply(entrenamiento,Dlasso)
logisita.pred <- vector("list",length = length(datos.lista))
for(l in 1:length( datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
lassoaccuracy <- accuracy
print(lassoaccuracy)
```

```
      raw    zscore        l2     media   mediana   min_max
0.5919283 0.5919283 0.5919283 0.5919283 0.5919283 0.5919283
```

```r
#Knn
set.seed(123456789)
myfnknn <- function(x) train(clase ~ ., data = x, method = "knn", trControl = trControl, t

logistica.lista <- lapply(entrenamiento,myfnknn)
logisita.pred <- vector("list",length = length(datos.lista))
for(l in 1:length( datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
knnaccuracy <- accuracy


#Naive Bayes
set.seed(123456789)
myfnknn <- function(x) train(clase ~ ., data = x, method = "naive_bayes", trControl = trCo

logistica.lista <- lapply(entrenamiento,myfnknn)
logisita.pred <- vector("list",length = length(datos.lista))
for(l in 1:length( datos.lista)){
  logisita.pred[[l]] <- predict(logistica.lista[[l]],test[[l]])
}
names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))
for(l in 1:length(datos.lista)){
  accuracy[l] <- confusionMatrix(test$raw$clase,logisita.pred[[l]])$overall[1]
}
names(accuracy) <- names(datos.lista)
bayesaccuracy <- accuracy


#Crearemos una matriz de 5x6
matriz <- matrix(nrow = 5, ncol = 6)

# Asignaremos los vectores a las filas de la matriz
matriz[1, ] <- accuracy
```

```r
matriz[2, ] <- ridgeaccuracy
matriz[3, ] <- lassoaccuracy
matriz[4, ] <- knnaccuracy
matriz[5, ] <- bayesaccuracy

# Vamos a dar nombres a las columnas y filas de nuestra matriz
Fila <- c("Regresion Logis", "Regresion Ridge", "Regresion Lassso", "Knn", "Naive Bayes")
Columna <- c("raw", "zscore", "l2", "media", "mediana", "min_max")

# Asignar nombres a las filas y columnas de la matriz
rownames(matriz) <- Fila
colnames(matriz) <- Columna

# Vamos a omprimir la matriz
print(matriz)
```

```
                     raw    zscore        l2     media   mediana   min_max
Regresion Logis  0.4080717 0.5336323 0.4596413 0.5336323 0.4125561 0.4125561
Regresion Ridge  0.5919283 0.5941704 0.5941704 0.5941704 0.5941704 0.5941704
Regresion Lassso 0.5919283 0.5919283 0.5919283 0.5919283 0.5919283 0.5919283
Knn              0.5852018 0.5717489 0.5986547 0.5941704 0.5874439 0.6031390
Naive Bayes      0.4080717 0.5336323 0.4596413 0.5336323 0.4125561 0.4125561
```