Regresión logística de Bridge y Lasso

```
set.seed(123456789)
```

Primero se cargan las 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
```

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

```
Loading required package: lattice
```

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

Cargamos los datos

```
datos <- read.table("./yeast.data",header = F)[,-1]
```

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

```
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.

```
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)
```

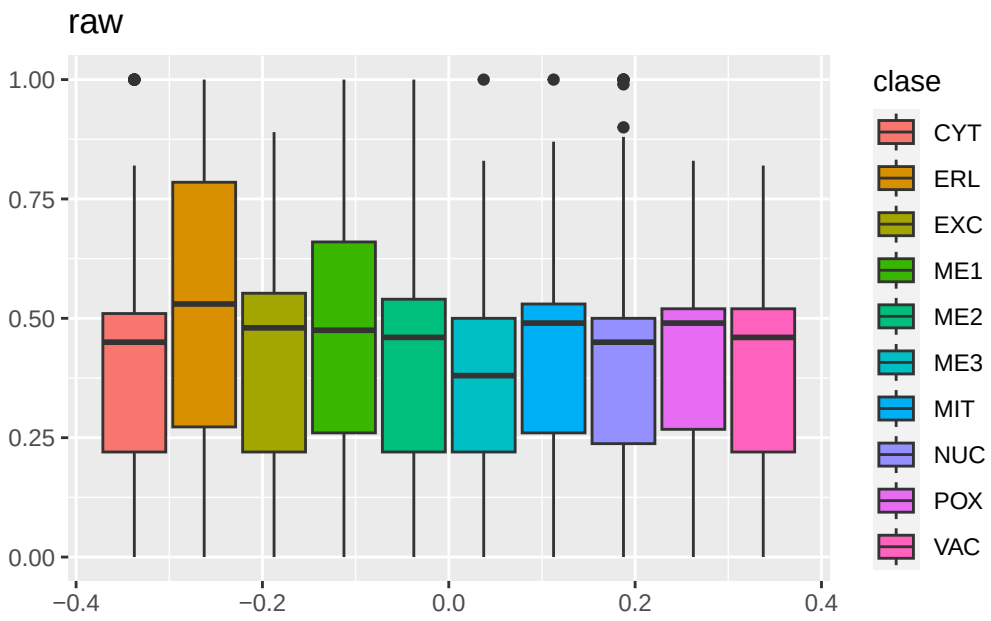
```
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
Using clase as id variables
```

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

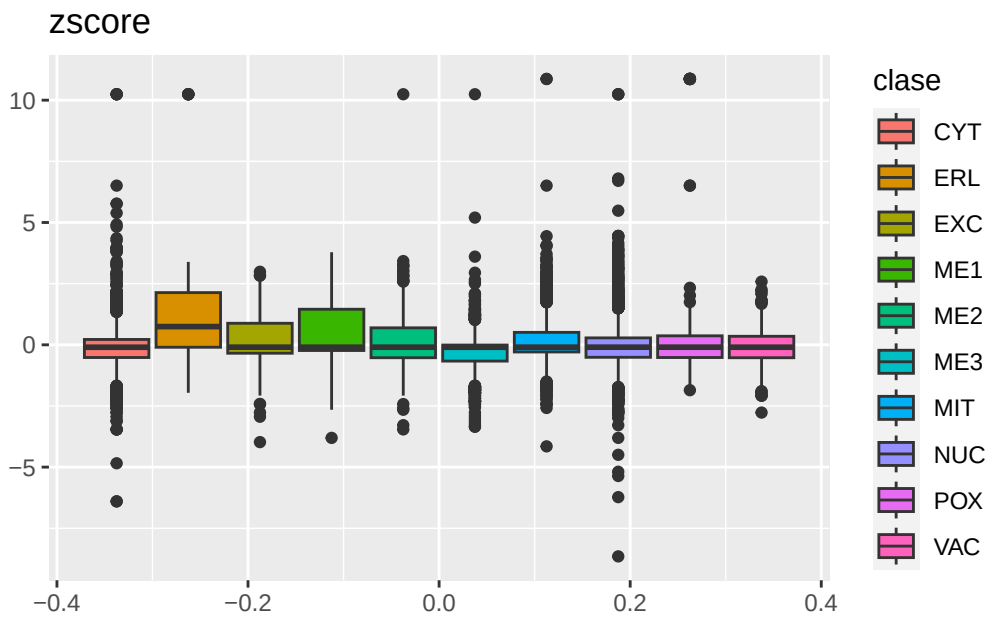
```
head(datos.melt$zscore)
```

```
  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
```

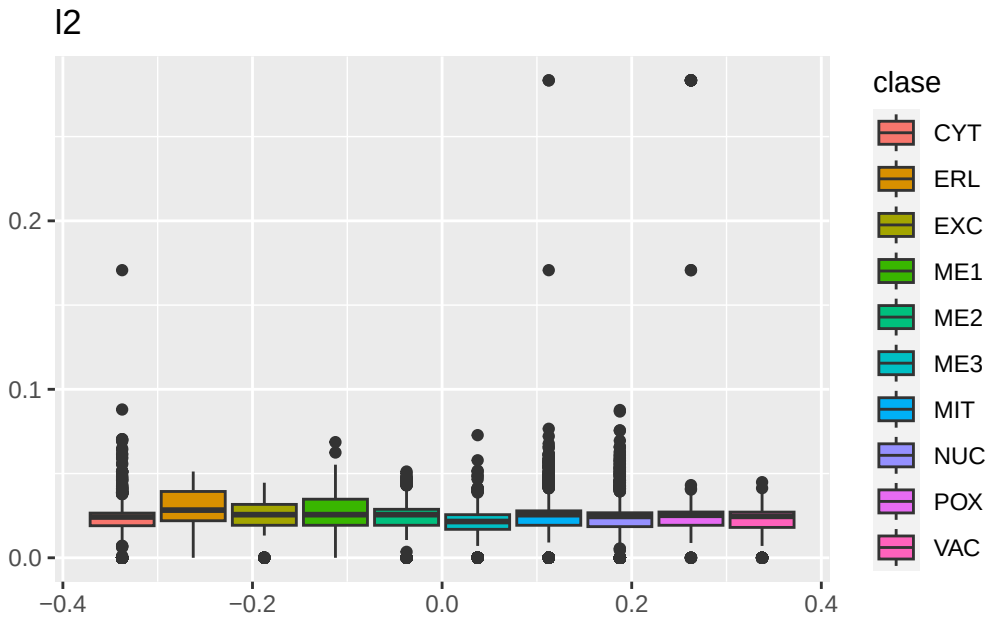
```
for(l in 1:length(datos.melt)){

  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
```



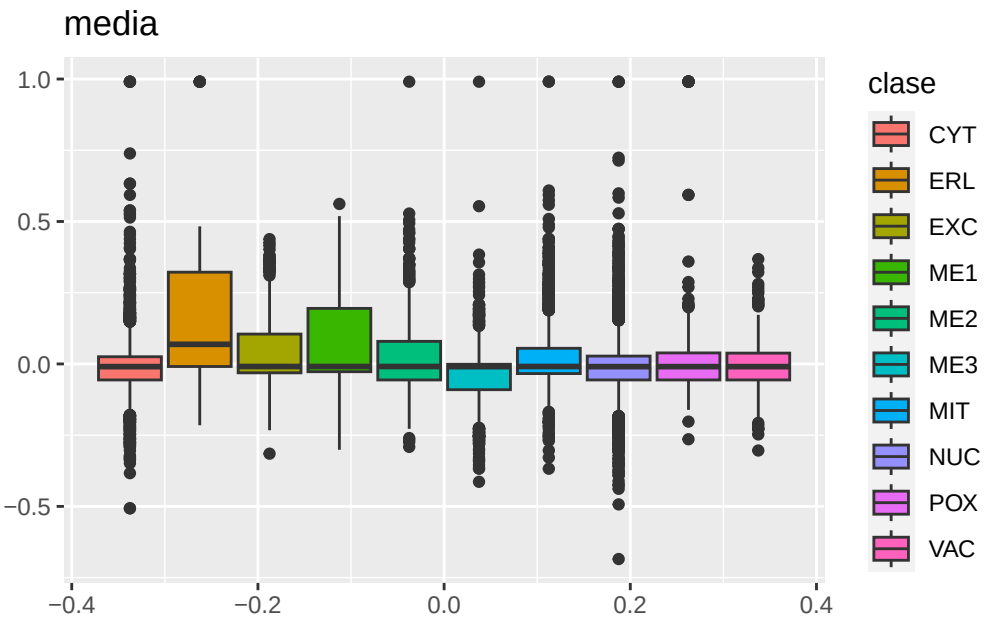

```
lista_graficos$plot2
```



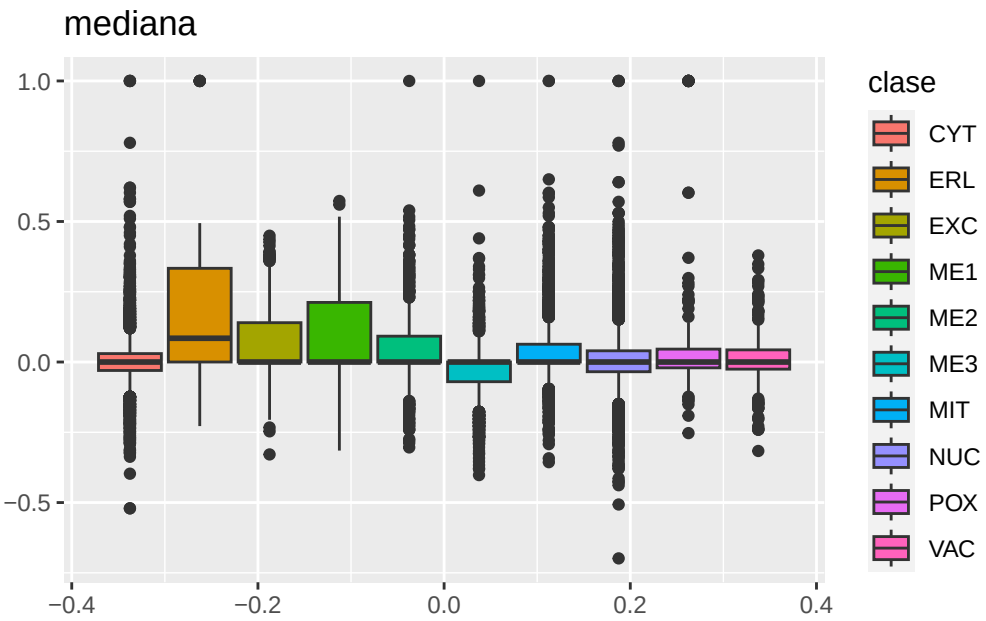
```
lista_graficos$plot3
```



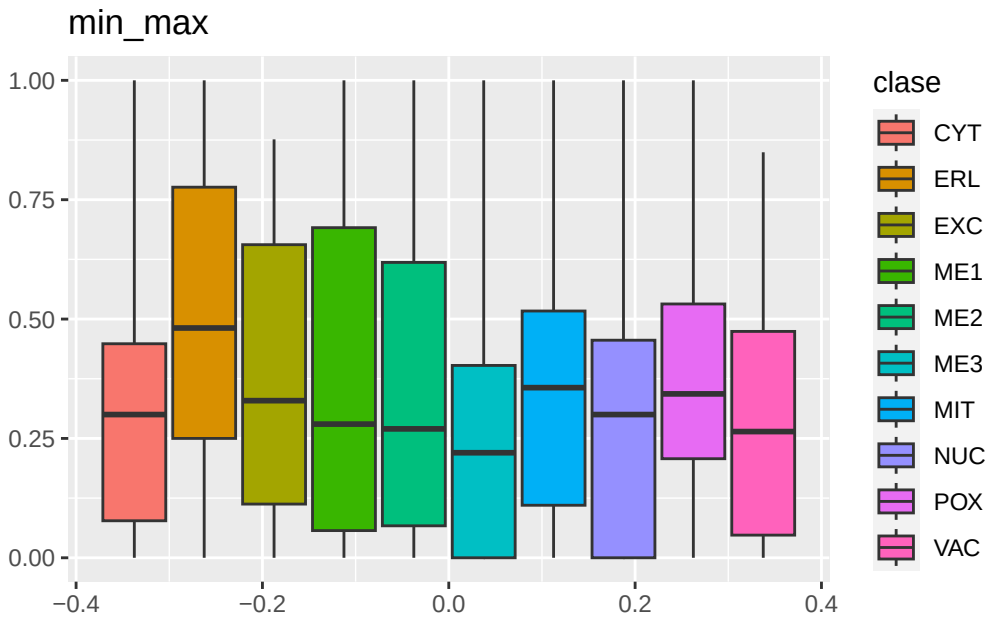

```
lista_graficos$plot4
```




```
lista_graficos$plot5
```



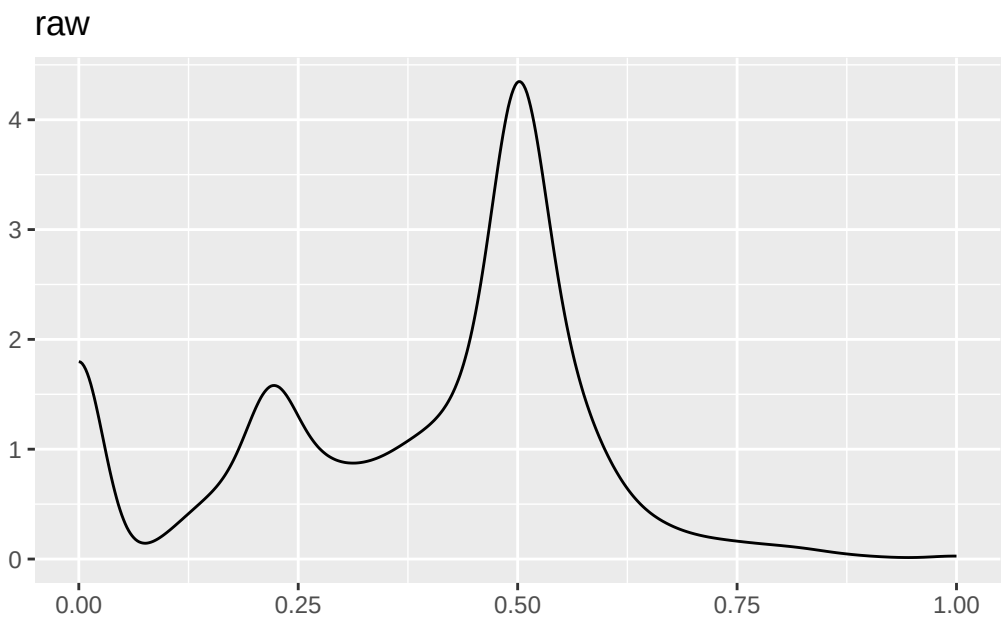
```
lista_graficos$plot6
```



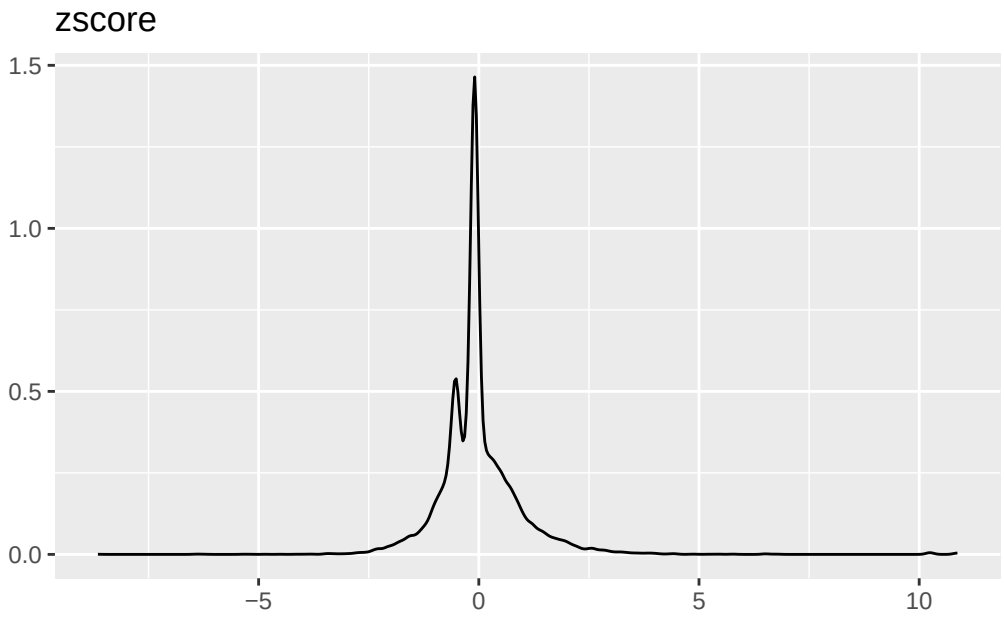
Así por ejemplo la normalización min-max es la mejor, puesto que no tenemos outliers

Otra forma de ver la transfomración es mediante gráficos de densidad

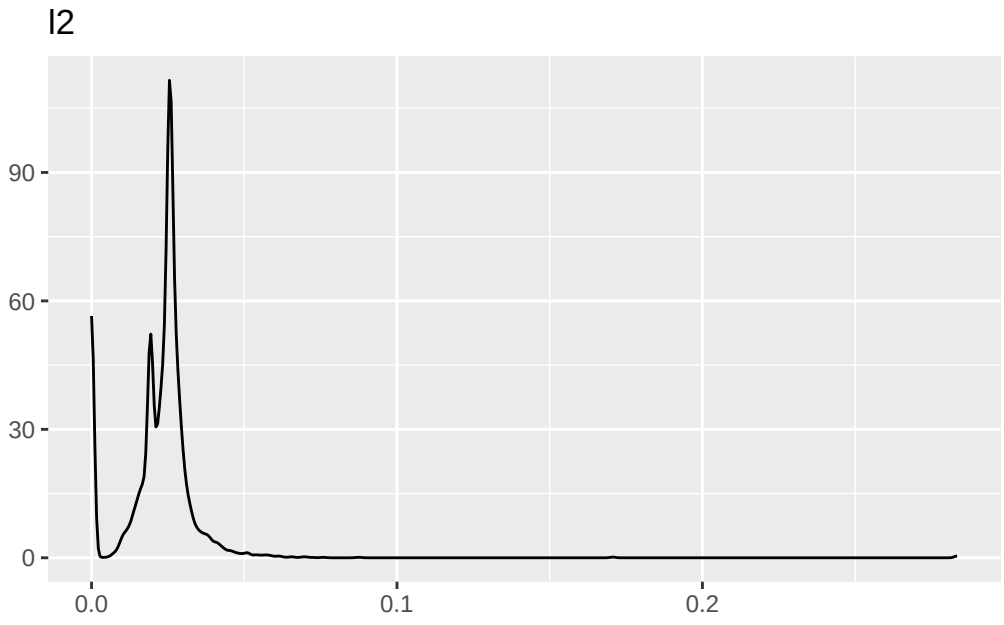
```
for(l in 1:length(datos.melt)){

  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
```



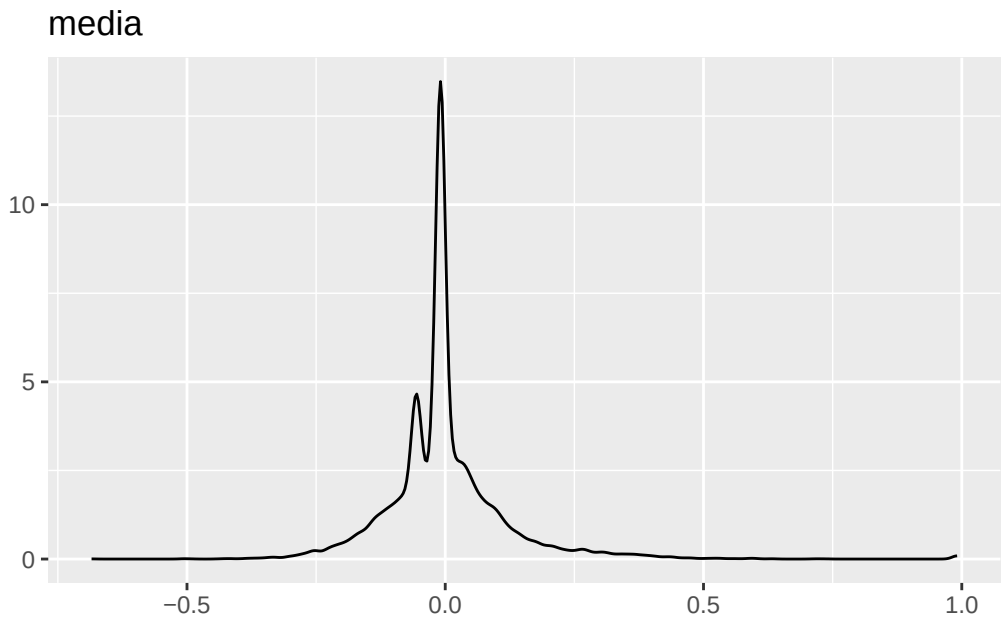

```
lista_graficos$plot2
```



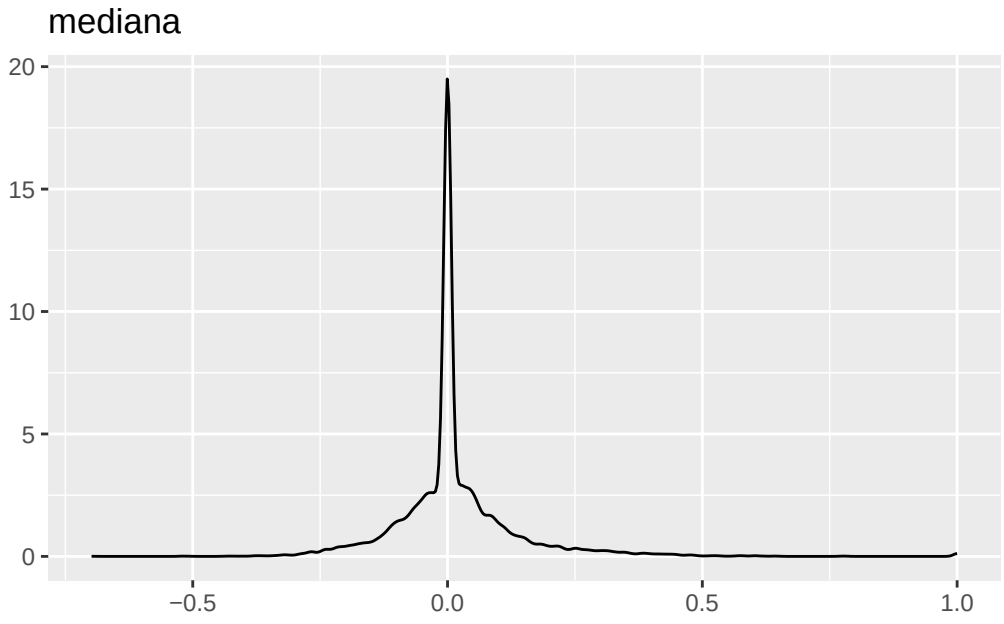
```
lista_graficos$plot3
```



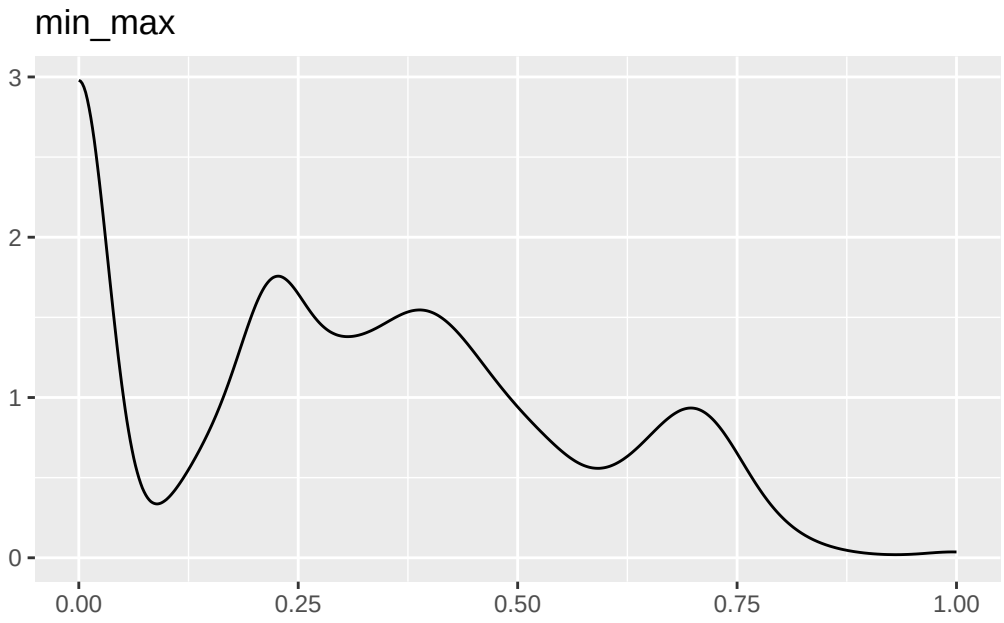

```
lista_graficos$plot4
```




```
lista_graficos$plot5
```

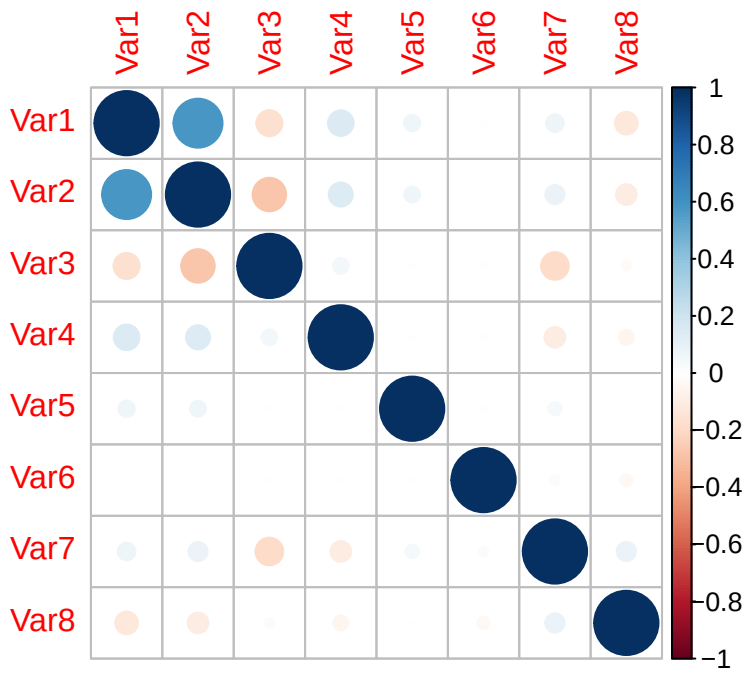


```
lista_graficos$plot6
```

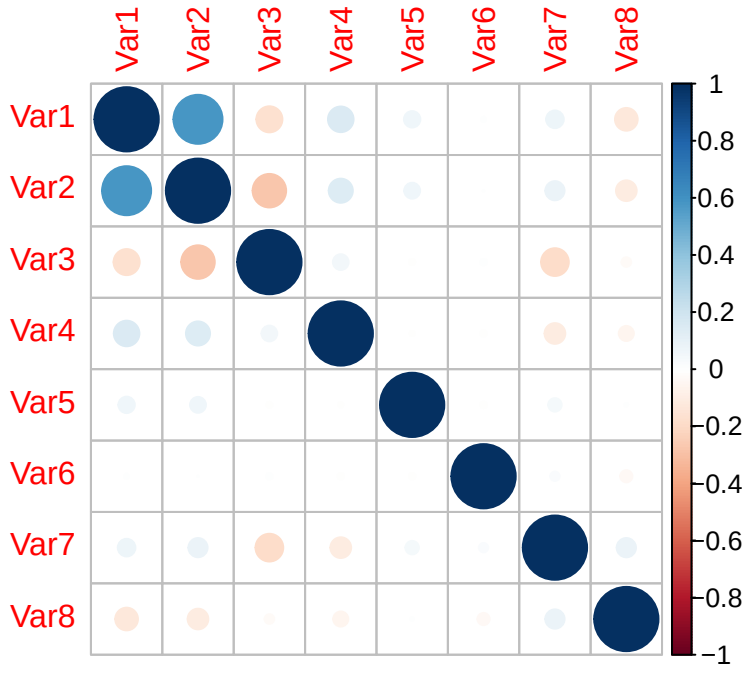


Sin embargo, al ver la densidad, no tenemos una transformacion uniforme.

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




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



Partición de datos

para conjunto de datos podemos realizar el split

```
set.seed(123456789)
n  <- nrow(datos)
idx <- sample(1:n,n*0.7)
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

```
set.seed(123456789)
trControl <- trainControl(method = 'cv',
                          number = 5)
myfnlog <- function(x) train(clase ~ ., data = x, method = "multinom", trControl = trContr

logistica.lista <- lapply(datos.train.lista,myfnlog)

logisita.pred <- vector("list",length = length(datos.lista))

for(l in 1:length(datos.lista)){

  logisita.pred[[l]] <- predict(logistica.lista[[l]],datos.test.lista[[l]])


}

names(logisita.pred) <- names(datos.lista)
accuracy <- vector("numeric",length = length(datos.lista))

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

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


}

names(accuracy) <- names(datos.lista)

### Este valor lo tienen que guardar solamente haremos por accuracy y kappa
### tenemos que mirar el objeto matconf
```

```
set.seed(123456789)

### para conjunto de datos podemos realizar el split
### lasso 1 ridge 0
datos.train.lista <- lapply(datos.lista, function(x) x[idx,])
datos.test.lista <- lapply(datos.lista, function(x) x[-idx,])

cvfit_lasso <- cv.glmnet(as.matrix(datos.train.lista$raw[,-ncol(datos)]),as.numeric(datos.
```

Regresión logística de Ridge

```
set.seed(123456789)
trControl <- trainControl(method = 'cv',
                          number = 5)
myfnlog <- function(x) train(clase ~ ., data = x, method = "glmnet", trControl = trControl

ridge.lista <- lapply(datos.train.lista,myfnlog)
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
```

```
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
ridge.pred <- vector("list",length = length(ridge.lista))

for(l in 1:length(ridge.lista)){

  ridge.pred[[l]] <- predict(ridge.lista[[l]],datos.test.lista[[l]])


}

names(ridge.pred) <- names(ridge.lista)
ridge.accuracy <- vector("numeric",length = length(datos.lista))

for(l in 1:length(ridge.lista)){

  ridge.accuracy[l] <- confusionMatrix(datos.test.lista$raw$clase,ridge.pred[[l]])$overall


}

names(ridge.accuracy) <- names(datos.lista)

### Este valor lo tienen que guardar solamente haremos por accuracy y kappa
### tenemos que mirar el objeto matconf
```

Regresión de Lasso

```
set.seed(123456789)
trControl <- trainControl(method = 'cv',
                          number = 5)
myfnlasso <- function(x) train(clase ~ ., data = x, method = "glmnet", trControl = trContr

lasso.lista <- lapply(datos.train.lista,myfnlasso)
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
Warning: from glmnet C++ code (error code -22); Convergence for 22th lambda
value not reached after maxit=100000 iterations; solutions for larger lambdas
returned
```

```
Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground

Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```r
lasso.pred <- vector("list",length = length(lasso.lista))

for(l in 1:length(lasso.lista)){
```

```
  lasso.pred[[l]] <- predict(lasso.lista[[l]],datos.test.lista[[l]])


}

names(lasso.pred) <- names(lasso.lista)
lasso.accuracy <- vector("numeric",length = length(datos.lista))

for(l in 1:length(lasso.lista)){

  lasso.accuracy[l] <- confusionMatrix(datos.test.lista$raw$clase,lasso.pred[[l]])$overall


}

names(lasso.accuracy) <- names(datos.lista)

### Este valor lo tienen que guardar solamente haremos por accuracy y kappa
### tenemos que mirar el objeto matconf
```

Knn

```
set.seed(123456789)
k_value=c(1:20)
kControl<- trainControl(method = 'repeatedcv',
                        number = 5,repeats = 15)
myfnknn <- function(x) train(clase ~ ., data = x, method = "knn", trControl = kControl,tun

knn.lista <- lapply(datos.train.lista,myfnknn)

knn.pred <- vector("list",length = length(knn.lista))

for(l in 1:length(knn.lista)){

  knn.pred[[l]] <- predict(knn.lista[[l]],datos.test.lista[[l]])


}

names(knn.pred) <- names(knn.lista)
```

```r
knn.accuracy <- vector("numeric",length = length(datos.lista))

for(l in 1:length(knn.lista)){

  knn.accuracy[l] <- confusionMatrix(datos.test.lista$raw$clase,knn.pred[[l]])$overall[1]


}

names(knn.accuracy) <- names(datos.lista)

### Este valor lo tienen que guardar solamente haremos por accuracy y kappa
### tenemos que mirar el objeto matconf
```

Bayes

```r
set.seed(123456789)
trControl <- trainControl(method = 'cv',
                          number = 5)
myfnbayes <- function(x) train(clase ~ ., data = x, method = "naive_bayes", trControl = tr

bayes.lista <- lapply(datos.train.lista,myfnbayes)

bayes.pred <- vector("list",length = length(bayes.lista))

for(l in 1:length(bayes.lista)){

  bayes.pred[[l]] <- predict(bayes.lista[[l]],datos.test.lista[[l]])


}

names(bayes.pred) <- names(bayes.lista)
bayes.accuracy <- vector("numeric",length = length(datos.lista))

for(l in 1:length(bayes.lista)){

  bayes.accuracy[l] <- confusionMatrix(datos.test.lista$raw$clase,bayes.pred[[l]])$overall
```

```
}

names(bayes.accuracy) <- names(datos.lista)

### Este valor lo tienen que guardar solamente haremos por accuracy y kappa
### tenemos que mirar el objeto matconf
```

Matriz

```
m <- cbind(accuracy,ridge.accuracy,lasso.accuracy,knn.accuracy,bayes.accuracy )
print(m)
```

```
         accuracy ridge.accuracy lasso.accuracy knn.accuracy bayes.accuracy
raw     0.5919283      0.5852018      0.5986547    0.5941704      0.4080717
zscore  0.5919283      0.5919283      0.5986547    0.5919283      0.5336323
l2      0.5964126      0.5919283      0.5964126    0.5896861      0.4596413
media   0.5941704      0.5941704      0.5941704    0.5964126      0.5336323
mediana 0.5941704      0.5896861      0.5964126    0.5784753      0.4125561
min_max 0.5941704      0.5919283      0.5986547    0.5964126      0.4125561
```