ANEXO: código R

ÍNDICE

1. CLASIFICACIÓN MUTACIÓN 2
2. CLASIFICACIÓN DE ESTADIO DIAGNÓSTICO 10
3. CLASIFICACIÓN DE FENOTIPO DE MOTONEURONA 18
4. DIFERENCIAS DE DISTRIBUCION POR CLASES: ESTADIO DIAGNOSTICO 23
5. DIFERENCIAS EN DISTRIBUCIÓN DE MARCADORES POR CLASE, ESTADIO DIAGNÓSTICO Y DETERIORO COGNITIVO COMBINADOS 32
6. DIFERENCIAS EN DISTRIBUCIÓN DE MARCADORES POR CLASES: MUTACIÓN C9ORF72 41
7. MODELIZACIÓN DE VARIABLES PRONÓSTICO 48
8. DISCUSIÓN 55

1. CLASIFICACIÓN MUTACIÓN

```
library("readxl")
library('MASS')
library(caret)
library(MASS)
library(pROC)

#Cargar datos

ela<- read_excel("BD_solopacientesELA.xlsx")


Genotipo.bin<- c()
Genotipo.bin[ela$Genotipo == 'C9ORF72']<- 'C9ORF72'
Genotipo.bin[!ela$Genotipo == 'C9ORF72']<- 'No.C9ORF72'
Genotipo.bin<- as.factor(Genotipo.bin)
relevel(Genotipo.bin, ref = 'No.C9ORF72')
ela$Genotipo.bin<- Genotipo.bin



volum<- ela[,c(52:239,
               335,336,
               371,372)]; names(volum)

index_volum.porc<- grep("Percentage|NotVent", names(volum))
index.global_volum.porc<- grep("Percentage|NotVent", names(ela))
volum.porc<- ela[,index.global_volum.porc]

Iron<- ela[,240: 299]; names(Iron)
iron<- Iron[,grep('Median',names(Iron))]; names(iron) ##este es el que se
usará
index.global_iron<- grep('Median',names(ela))

thickness_left<- ela[,301:334]
index.global_th.left<- 301:334; names(thickness_left)

thickness_right<- ela[,337:370]; names(thickness_right)
index.global_th.right<- 337:370
```

Método 1: ***Elastic Net***

```
## funciones para aplicar Elastic Net
##### especificar familia, nfolds y ncv

library(glmnet)
penalize<- function(y,x,a,maxit=10^5,familia='gaussian'){
  set.seed(1)
```

```r
  #index<- rowSums(!is.na(x))>=2
  index<- complete.cases(x)
  y <- data.matrix(y)
  x <- as.matrix(x)
  cv_model <- cv.glmnet(x= x[index, ], y= y[index], alpha = a,
                        family=familia, maxit=maxit, nfolds = 5, ncv=3)

  best_lambda <- cv_model$lambda.min
  #best_lambda
  best_model <- glmnet(x[index, ], y[index], alpha = a,
                       family=familia,lambda = best_lambda)
  mod.lasso<- coef(best_model)
  return(list(mod.lasso,
              #min(cv_model$cvm)
              sqrt(min(cv_model$cvm))
              ))
}


alpha<- seq(0,1, by=0.001)

EN<- function(y,x){

L<-list()
for (i in 1:length(alpha)) {
  L[[i]]<-penalize(y, x,a=alpha[i])
}

errorCV<- unlist(lapply(1:length(L), function(i) L[[i]][[2]]))
return(list(mod.final= L[[which.min(errorCV)]],
            alpha=alpha[which.min(errorCV)],
            errorCV))
}

# EN por bloque
th.left_EN<- EN(y=Genotipo.bin, x=thickness_left)
th.right_EN<- EN(y=Genotipo.bin, x=thickness_right)
volum_EN<- EN(y=Genotipo.bin, x=volum.porc)
iron_EN<- EN(y=Genotipo.bin, x=iron)

# Union de cada bloque
marcadores.finales<- function(en, indice){
  indice[which(en !=0)]
}

th.left_MF<- marcadores.finales(th.left_EN$mod.final[[1]][-1],
                                index.global_th.left)
th.rigth_MF<- marcadores.finales(th.right_EN$mod.final[[1]][-1],
                                 index.global_th.right)
volum_MF<- marcadores.finales(volum_EN$mod.final[[1]][-1],
```

```
                                index.global_volum.porc)
iron_MF<- marcadores.finales(iron_EN$mod.final[[1]][-1],
index.global_iron)

MF_index<- sort(c(volum_MF, th.left_MF, th.rigth_MF, iron_MF))


# EN sobre la union
union_EN<- EN(y=Genotipo.bin, x=ela[,MF_index])


# Step
MFunion_index<- which(names(ela) %in%
        rownames(union_EN$mod.final[[1]])[-1][union_EN$mod.final[[1]][-
1]!=0])

summary(glm(Genotipo.bin~., data=ela[,MFunion_index[-12]],
            family=binomial(link = 'logit')))
#which(cor(ela[,MFunion_index])==1, arr.ind=TRUE)

fullmodel_gen<-glm(relevel(Genotipo.bin, ref = 'No.C9ORF72')~.,
                   data = ela[,MFunion_index],
                   family = binomial(link = 'logit'))
nullmodel_gen<- glm(relevel(Genotipo.bin, ref = 'No.C9ORF72')~1,
                    data = ela[,MFunion_index],
                    family = binomial(link = 'logit'))

stepforward_gen<-stepAIC(nullmodel_gen,
                     direction = 'forward',
                     scope = list(upper = fullmodel_gen,
                                  lower = nullmodel_gen),
                     trace = 0)

summary(stepforward_gen)

stepbackward_gen<-stepAIC(fullmodel_gen,
                     direction = 'backward',
                     scope = list(upper = fullmodel_gen,
                                  lower = nullmodel_gen),
                     trace = 0)
summary(stepbackward_gen)


# CV
modfinal.gen_df<- data.frame(ela$Genotipo.bin, ela$lh_BrainSegVolNotVent,
                             ela$lh_lingual_thickness,
                             ela$lh_precuneus_thickness,
                             ela$rh_paracentral_thickness,
                             ela$rh_parahippocampal_thickness)
```

```
set.seed(1)
CV_gen1<- train(ela.Genotipo.bin~.,data = modfinal.gen_df,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='binomial')

set.seed(1)
CV_gen2<- train(ela.Genotipo.bin~.,data = modfinal.gen_df,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3,
                               summaryFunction = twoClassSummary,
                               savePredictions = "all",
                               classProbs = TRUE),
      method='glm', family='binomial')



# Ajuste
clasif_gen<- c()
clasif_gen[predict(stepforward_gen, type = 'response')>=0.5]<-
'No.C9ORF72'
clasif_gen[!predict(stepforward_gen, type = 'response')>=0.5]<- 'C9ORF72'
clasif_gen<- as.factor(clasif_gen)

library(caret)
confusionMatrix(Genotipo.bin,clasif_gen)

library(pROC)
roc.gen<-roc(Genotipo.bin, predict(stepforward_gen, type = 'response'))
plot(roc.gen)
auc(roc.gen)
```

#Método 2: análisis discriminante

```
allmarkers<- cbind(thickness_left,thickness_right, iron, volum.porc)

# funcion para realizar PCA
PCA<- function(Y, X, acum.var){

casoscomp<- complete.cases(cbind(Y,X))

y<- as.factor(Y[casoscomp])
x<- X[casoscomp,]

pca <- prcomp(x, scale = TRUE)
prop.acumulada_var<- summary(pca)$importance[3,]
num.PC<- which(prop.acumulada_var >= acum.var)[1]

# Obtener las covariables transformadas por PCA
covariables_pca <- data.frame(as.data.frame(pca$x)[,1:num.PC])
```

```r
colnames(covariables_pca)<- sapply(1:num.PC, function(x) paste('PC',x,sep
= ''))

var.exp<- summary(pca)$importance[3,num.PC] # varianza explicada

return(list(PC=covariables_pca,
            coefPC=pca$rotation[,1:num.PC],
            var.exp= var.exp,
            numPC= num.PC,
            obs=sum(casoscomp),
            casoscomp=casoscomp,
            prcomp=pca
       ))
}

## funciones para validacion cruzada de analisis discriminante

cv<- function(factor, nfolds, PCA, AD='lda'){

y<- as.factor(factor[PCA$casoscomp])

sepFolds<-lapply(1:length(levels(y)), function(x)
  {createFolds(y[y==levels(y)[x]], k=nfolds, returnTrain = FALSE)} )

Folds<- list()
for (k in 1:nfolds) {
  Folds[[k]]<- c(which(y==levels(y)[1])[unlist(sepFolds[[1]][k])],
  which(y==levels(y)[2])[unlist(sepFolds[[2]][k])])
}

Accuracy<- c()

AUC<- c()

X<- data.frame(PCA$PC)

for (i in 1:nfolds) {

  test_indices <- Folds[[i]]

  X_train <- data.frame(X[-test_indices, ])
  colnames(X_train)<- colnames(PCA$PC)
  y_train <- y[-test_indices]

  X_test <- data.frame(X[test_indices, ])
  colnames(X_test)<- colnames(PCA$PC)
  y_test <- y[test_indices]

  # Ajustar el modelo
  if (AD=='lda'){
```

```
      mod<- lda(y_train ~., data= X_train)
  }

  if (AD=='qda'){
      mod<- qda(y_train ~., data= X_train)
  }

  # Hacer predicciones en test usando el modelo ajustado
  y_pred <- predict(mod, newdata = X_test)

  # Calcular errores
  cm <- confusionMatrix(y_pred$class, y_test)
  accuracy<- sum(diag(cm$table))/sum(cm$table)

  sensitivity <- cm$byClass["Sensitivity"]
  specificity <- cm$byClass["Specificity"]
  auc<- suppressMessages(auc(roc(y_test, y_pred$posterior[,1])))

  Accuracy<- c(Accuracy, accuracy)
  Sensitivity<- c(Sensitivity, sensitivity)
  Specificity<- c(Specificity, specificity)
  AUC<- c(AUC, auc)

}

# Media de cada error
Accuracy_CV<- mean(Accuracy, na.rm=T)
Sensitivity_CV<- mean(Sensitivity, na.rm=T)
Specificity_CV<- mean(Specificity, na.rm=T)
AUC_CV<- mean(AUC, na.rm=T)

Error_average<- rbind(Accuracy_CV, Sensitivity_CV, Specificity_CV,
AUC_CV)

return(Error_average)
}


# cv(Genotipo.bin, 5, PCA_gen, AD='lda')


# funcion para validacion cruzada con repeticion
repeated_CV<- function(factor, nfolds, ntimes, PCA, AD){

CV<- list()
seed<- rnorm(ntimes)

for (i in 1:ntimes) {
set.seed(i)
CV[[i]]<- cv(factor, nfolds, PCA, AD)
```

```
}

CV_average<- Reduce('+', CV)/ntimes
return(CV_average)
}

#repeated_CV(Genotipo.bin, 5, 3, PCA(Genotipo.bin,allmarkers,0.6),
AD='lda')
```

#Resultados, comparacion capacidad predictiva

```
## error de precision en CV según varianza explicada (num. PC)


Var<- summary(PCA_gen$prcomp)$importance[3, ]

CVlda_var<- sapply(Var,function(v) {repeated_CV(factor = Genotipo.bin,
            nfolds = 5, ntimes = 3,
            PCA(Genotipo.bin,allmarkers,v), AD='lda')})

CVqda_var<- sapply(Var, function(v) {
  tryCatch({
    repeated_CV(factor = Genotipo.bin, nfolds = 5, ntimes = 3,
                PCA(Genotipo.bin, allmarkers, v), AD = 'qda')
  }, error = function(e) {
    NA  # Valor que se asignará si ocurre un error
  })
})
CVqda_var<- sapply(1:sum(!is.na(CVqda_var)), function(n) CVqda_var[[n]])

PC.optim_lda<- which.max(CVlda_var[4,])
PCA.best_MUT<- PCA(Genotipo.bin, allmarkers, Var[PC.optim_lda])
lda.best_MUT<- lda(Genotipo.bin[PCA.best_MUT$casoscomp]~., data =
PCA.best_MUT$PC)

PC.optim_qda<- which.max(CVqda_var[4,])

round(CVlda_var[,PC.optim_lda],2 )
round(CVqda_var[,PC.optim_qda],2)


### LDA similar a QDA
##############################

PCA_gen$var.exp

CV_gen1;CV_gen2

### mejor reg.log. que análisis discriminante
```

Comparación ajuste

```
# A.D.

PC_lda<- PCA(Genotipo.bin, allmarkers, Var[PC.optim_lda]); PC_lda$var.exp
PC_qda<- PCA(Genotipo.bin, allmarkers, Var[PC.optim_qda])

lda_best<- lda(Genotipo.bin[PC_lda$casoscomp]~., data = PC_lda$PC)
qda_best<- qda(Genotipo.bin[PC_qda$casoscomp]~., data = PC_qda$PC)

confusionMatrix(predict(lda_best)$class, Genotipo.bin[PC_lda$casoscomp])
# LDA
confusionMatrix(predict(qda_best)$class, Genotipo.bin[PC_qda$casoscomp])
# QDA

roc.gen_LDA<-roc(Genotipo.bin[PC_lda$casoscomp],
                 predict(lda_best)$posterior[,1])
auc(roc.gen_LDA)

roc.gen_QDA<-roc(Genotipo.bin[PC_qda$casoscomp],
                 predict(qda_best)$posterior[,1])
auc(roc.gen_QDA)



# Reg. log.
confusionMatrix(Genotipo.bin,clasif_gen)
roc.gen<-roc(Genotipo.bin, predict(stepforward_gen, type = 'response'))
#plot(roc.gen)
auc(roc.gen)

library(knitr)
library(kableExtra)

# Crear el dataframe bestmod_mut
bestmod_mut <-
as.data.frame(round(summary(stepforward_gen)$coefficients[,-3],3))
colnames(bestmod_mut) <- c('Coeficiente', 'Error estándar', 'p-valor')
rownames(bestmod_mut) <- c('(Intercepto)', rownames(bestmod_mut)[2:6])

# Generar la tabla con kable y aplicar estilos y formato
tabla_modmut <- kable(bestmod_mut) %>%
  kable_styling("striped", full_width = FALSE) %>%
  row_spec(3, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(4, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(5, bold = TRUE, extra_css = "font-weight: bold;")

tabla_modmut
```

2. CLASIFICACIÓN DE ESTADIO DIAGNÓSTICO

```
library(caret)
library(MASS)
library(pROC)
library(pracma)
install.packages("officer")
install.packages("flextable")
library(officer)
library(flextable)
install.packages("kableExtra")
library(kableExtra)

PCA_ED<- PCA(ela$`Estadio diagnóstico`, allmarkers, 0.9)

## Medidas de capacidad predictiva


cv3<- function(factor, nfolds, PCA, AD='lda'){

y<- as.factor(factor[PCA$casoscomp])

sepFolds<-lapply(1:length(levels(y)), function(x)
  {createFolds(y[y==levels(y)[x]], k=nfolds, returnTrain = FALSE)} )

Folds<- list()
for (k in 1:nfolds) {
  Folds[[k]]<- c(which(y==levels(y)[1])[unlist(sepFolds[[1]][k])],
  which(y==levels(y)[2])[unlist(sepFolds[[2]][k])],
  which(y==levels(y)[3])[unlist(sepFolds[[3]][k])])
}

Accuracy<- c()
Sensitivity<- matrix(ncol=3, nrow =0 )
Specificity<- matrix(ncol=3, nrow =0 )
AUC<- c()

X<- data.frame(PCA$PC)

for (i in 1:nfolds) {

  test_indices <- Folds[[i]]

  X_train <- data.frame(X[-test_indices, ])
  colnames(X_train)<- colnames(PCA$PC)
  y_train <- y[-test_indices]
```

```
  X_test <- data.frame(X[test_indices, ])
  colnames(X_test)<- colnames(PCA$PC)
  y_test <- y[test_indices]

  # Ajustar el modelo
  if (AD=='lda'){
      mod<- lda(y_train ~., data= X_train)
  }

  if (AD=='qda'){
      mod<- qda(y_train ~., data= X_train)
  }

  # Hacer predicciones en test usando el modelo ajustado
  y_pred <- predict(mod, newdata = X_test)

  # Calcular errores
  cm <- confusionMatrix(y_pred$class, y_test)
  accuracy<- sum(diag(cm$table))/sum(cm$table)

  sensitivity <- cm$byClass[, "Sensitivity"]
  specificity <- cm$byClass[, "Specificity"]

  auc<- multiclass.roc(y_test, y_pred$posterior)
  auc_value <- sub(".*: ", "", auc$auc)
  auc<- as.numeric(auc_value)

  Accuracy<- c(Accuracy, accuracy)
  Sensitivity<- rbind(Sensitivity, sensitivity)
  Specificity<- rbind(Specificity, specificity)
  AUC<- c(AUC, auc)

}

# Media de cada error
Accuracy_CV<- mean(Accuracy, na.rm=T)
Sensitivity_CV<- sapply(1:3, function(x) mean(Sensitivity[,x], na.rm=T))
Specificity_CV<- sapply(1:3, function(x) mean(Specificity[,x], na.rm=T))
AUC_CV<- mean(AUC, na.rm=T)

Error_average<-
list(Accuracy_CV=Accuracy_CV,AUC_CV=AUC_CV,rbind(Sensitivity_CV,
Specificity_CV))

return(Error_average)
}

cv3(ela$`Estadio diagnóstico`, 5, PCA_ED, AD="lda")
```

```r
repeated_CV3<- function(factor, nfolds, ntimes, PCA, AD){

M<- list(c(0), c(0), matrix(rep(0,6),nrow = 2, ncol = 3))

for (i in 1:ntimes) {

set.seed(i)
CV<- cv3(factor, nfolds, PCA, AD)

M<- lapply(1:3, function(x) M[[x]] + CV[[x]])

}

M<- lapply(1:3, function(x) M[[x]]/ntimes)


return(M)
}

repeated_CV3(ela$`Estadio diagnóstico`, 5,3, PCA_ED, AD="lda")

## error de precisión en CV según varianza explicada (num. PC)



Var_ED<- summary(PCA_ED$prcomp)$importance[3, ]

CVlda_var_ED<- lapply(Var_ED,function(v) {repeated_CV3(factor =
ela$`Estadio diagnóstico`,
            nfolds = 5, ntimes = 3,
            PCA(ela$`Estadio diagnóstico`,allmarkers,v), AD='lda')})

CVqda_var_ED<- lapply(Var_ED, function(v) {
  tryCatch({
    repeated_CV3(factor = ela$`Estadio diagnóstico`,
                 nfolds = 5, ntimes = 3,
                 PCA(ela$`Estadio diagnóstico`,allmarkers,v), AD = 'qda')
  }, error = function(e) {
    NA  # Valor que se asignará si ocurre un error
  })
})
CVqda_var_ED<- lapply(1:sum(!is.na(CVqda_var_ED)), function(n)
CVqda_var_ED[[n]])

PC.optim_lda_ED<- which.max(sapply(1:length(CVlda_var_ED), function(x)
CVlda_var_ED[[x]][[2]]))
PC.optim_qda_ED<- which.max(sapply(1:length(CVqda_var_ED), function(x)
CVqda_var_ED[[x]][[2]]))
```

```
CVlda_var_ED[[PC.optim_lda_ED]]
CVqda_var_ED[[PC.optim_qda_ED]]

## LDA mejor que QDA
```

Cálculo de derivada parcial de log-odds en función de cada marcador

```
v.exp<- Var_ED[PC.optim_lda_ED]
PCA.best_ED<-PCA(ela$`Estadio diagnóstico`,allmarkers,v.exp)

ED<- as.factor(ela$`Estadio diagnóstico`[PCA.best_ED$casoscomp])
lda.best_ED<- lda(ED~.,data = PCA.best_ED$PC)
coef.FDL<- LDA(ED~.,data = cbind(ED,PCA.best_ED$PC),
               output = 'Discriminant
Functions')$original$discriminant.functions[-1,]


## E3 / E1
der.logodds_ED<- PCA.best_ED$coefPC %*% data.matrix(coef.FDL[,3]-
coef.FDL[,1])
order_der.logodds_ED<- data.matrix(der.logodds_ED[order(-
abs(der.logodds_ED)),])

# E2 / E1
der.logodds_ED2<- PCA.best_ED$coefPC %*% data.matrix(coef.FDL[,2]-
coef.FDL[,1])
order_der.logodds_ED2<- data.matrix(der.logodds_ED2[order(-
abs(der.logodds_ED2)),])

# E3 / E2
der.logodds_ED3<- PCA.best_ED$coefPC %*% data.matrix(coef.FDL[,3]-
coef.FDL[,2])
order_der.logodds_ED3<- data.matrix(der.logodds_ED3[order(-
abs(der.logodds_ED3)),])

# Gráfico E3 / E1
v.influyentes_ED<- which(abs(order_der.logodds_ED)>=
  quantile(abs(order_der.logodds_ED), 0.8))

df<- as.data.frame(data.matrix(order_der.logodds_ED[v.influyentes_ED,]))
row_names<- rownames(df)

library(ggplot2)
library(dplyr)

df_long <- reshape2::melt(df)
rownames(df_long)<- row_names


# Determine color based on sign
```

```r
df_long$color <- ifelse(df_long$value <= 0, "red", "blue")

# Get absolute values
df_long$value <- abs(df_long$value)

# Sort the data frame by absolute value in descending order
df_long <- df_long[order(df_long$value, decreasing = TRUE), ]

ggplot_ED<- ggplot(df_long, aes(x = reorder(row_names, -value), y =
value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds", bold("(Estadio
3/Estadio 1)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))

ggsave("grafico.png", plot = ggplot_ED, width = 8, height = 6)

# Gráfico E2 / E1

v.influyentes_ED2<- which(abs(order_der.logodds_ED2)>=
  quantile(abs(order_der.logodds_ED2), 0.8))

df_ED2<-
as.data.frame(data.matrix(order_der.logodds_ED2[v.influyentes_ED2,]))
row_names_ED2<- rownames(df_ED2)

library(ggplot2)
library(dplyr)

df_ED2_long <- reshape2::melt(df_ED2)
rownames(df_ED2_long)<- row_names_ED2


# Determine color based on sign
df_ED2_long$color <- ifelse(df_ED2_long$value <= 0, "red", "blue")

# Get absolute values
df_ED2_long$value <- abs(df_ED2_long$value)
```

```r
# Sort the data frame by absolute value in descending order
df_ED2_long <- df_ED2_long[order(df_ED2_long$value, decreasing = TRUE), ]

ggplot_ED2<- ggplot(df_ED2_long, aes(x = reorder(row_names_ED2, -value),
y = value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds", bold("(Estadio
2 / Estadio 1)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))

ggsave("graficoE2vsE1.png", plot = ggplot_ED2, width = 8, height = 6)

# Gráfico E3 / E2


v.influyentes_ED3<- which(abs(order_der.logodds_ED3)>=
  quantile(abs(order_der.logodds_ED3), 0.8))

df_ED3<-
as.data.frame(data.matrix(order_der.logodds_ED3[v.influyentes_ED3,]))
row_names_ED3<- rownames(df_ED3)

library(ggplot2)
library(dplyr)

df_ED3_long <- reshape2::melt(df_ED3)
rownames(df_ED3_long)<- row_names_ED3


df_ED3_long$color <- ifelse(df_ED3_long$value <= 0, "red", "blue")
df_ED3_long$value <- abs(df_ED3_long$value)
df_ED3_long <- df_ED3_long[order(df_ED3_long$value, decreasing = TRUE), ]

ggplot_ED3<- ggplot(df_ED3_long, aes(x = reorder(row_names_ED3, -value),
y = value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
```

```
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds", bold("(Estadio
3 / Estadio 2)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))

ggsave("graficoE3vsE2.png", plot = ggplot_ED3, width = 8, height = 6)

## marcadores que influyen de 1 a 2, pero no de 1 a 3
m12<- data.matrix(df_ED2[which(! rownames(df_ED2) %in% rownames(df)),])
rownames(m12)<-rownames(df_ED2)[which(! rownames(df_ED2) %in%
rownames(df))]

m13_12<- der.logodds_ED[rownames(m12),]
dif12vs13<- cbind(m12, m13_12)
colnames(dif12vs13)<- c('Estadio 2 / Estadio 1', 'Estadio 3 / Estadio 1')

## marcadores que influyen de 2 a 3, pero no de 1 a 3
m23<- data.matrix(df_ED3[which(! rownames(df_ED3) %in% rownames(df)),])
rownames(m23)<- rownames(df_ED3)[which(! rownames(df_ED3) %in%
rownames(df))]

##
m23<- data.matrix(df_ED3[which(! rownames(df_ED3) %in% rownames(df)),])
rownames(m23)<-rownames(df_ED3)[which(! rownames(df_ED3) %in%
rownames(df))]

m13_23<- der.logodds_ED[rownames(m23),]
dif23vs13<- cbind(m23, m13_23)
colnames(dif23vs13)<- c('Estadio 3 / Estadio 2', 'Estadio 3 / Estadio 1')

my_table <- kable(as.data.frame(round(dif12vs13,2)))
my_table <- kable_styling(my_table, "striped", full_width = FALSE)

my_table2 <- kable(as.data.frame(round(dif23vs13,2)))
my_table2 <- kable_styling(my_table2, "striped", full_width = FALSE)

# boxplots

datosboxplotED <- list(
  order_der.logodds_ED = order_der.logodds_ED,
  order_der.logodds_ED2 = order_der.logodds_ED2,
  order_der.logodds_ED3 = order_der.logodds_ED3
)
```

```r
df_boxplotED <- data.frame(
  Grupo = rep(names(datosboxplot), lengths(datosboxplot)),
  Valores = unlist(datosboxplot)
)

ggplot(df_boxplotED, aes(x = Grupo, y = Valores)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("Estadio 3 / Estadio 1",
                              "Estadio 2 / Estadio 1",
                              "Estadio 3 / Estadio 2")) +
  xlab("") +
  ylab("")
```

3. CLASIFICACIÓN DE FENOTIPO DE MOTONEURONA

```
library(caret)
library(MASS)
library(pROC)
library(flipMultivariates)

FM<- as.factor(ela$`Fenotipo motoneurona`)
table(FM)

PCA_FM<- PCA(FM, allmarkers, 0.9)

## error de precisión en CV segÚn varianza explicada (num. PC)

v_inicial_FM<- (floor(100*summary(PCA_FM$prcomp)$importance[3,1])+1)/100
v_resto_FM<- summary(PCA_FM$prcomp)$importance[3,-1]
Var_FM<- c(v_inicial_FM,v_resto_FM)

CVlda_var_FM<- lapply(Var_FM,function(v) {repeated_CV3(factor = FM,
            nfolds = 5, ntimes = 3,
            PCA(FM,allmarkers,v), AD='lda')})

CVqda_var_FM<- lapply(Var_FM, function(v) {
  tryCatch({
    repeated_CV3(factor = FM,
                 nfolds = 5, ntimes = 3,
                 PCA(FM,allmarkers,v), AD = 'qda')
  }, error = function(e) {
    NA  # Valor que se asignarÁ si ocurre un error
  })
})
CVqda_var_FM<- lapply(1:sum(!is.na(CVqda_var_FM)), function(n)
CVqda_var_FM[[n]])

PC.optim_lda_FM<- which.max(sapply(1:length(CVlda_var_FM), function(x)
CVlda_var_FM[[x]][[2]]))
PC.optim_qda_FM<- which.max(sapply(1:length(CVqda_var_FM), function(x)
CVqda_var_FM[[x]][[2]]))

CVlda_var_FM[[PC.optim_lda_FM]]
CVqda_var_FM[[PC.optim_qda_FM]]

## LDA mejor que QDA

v.exp_FM<- Var_FM[PC.optim_lda_FM]
PCA.best_FM<-PCA(FM,allmarkers,v.exp_FM)

FMcc<- FM[PCA.best_FM$casoscomp]
```

```r
lda.best_FM<- lda(FMcc~.,data = PCA.best_FM$PC)
coef.FDL_FM<- LDA(FMcc~.,data = cbind(FMcc,PCA.best_FM$PC),
                  output = 'Discriminant
Functions')$original$discriminant.functions[-1,]


## ELAc / MNI
der.logodds_FM1<- PCA.best_FM$coefPC%*%data.matrix(coef.FDL_FM[,1]-
coef.FDL_FM[,2])
order_der.logodds_FM1<- data.matrix(der.logodds_FM1[order(-
abs(der.logodds_FM1)),])

## ELAc / MNS
der.logodds_FM2<- PCA.best_FM$coefPC%*%data.matrix(coef.FDL_FM[,1]-
coef.FDL_FM[,3])
order_der.logodds_FM2<- data.matrix(der.logodds_FM2[order(-
abs(der.logodds_FM2)),])

## MNI / MNS
der.logodds_FM3<- PCA.best_FM$coefPC%*%data.matrix(coef.FDL_FM[,2]-
coef.FDL_FM[,3])
order_der.logodds_FM3<- data.matrix(der.logodds_FM3[order(-
abs(der.logodds_FM3)),])

v.influyentes_FM1<- which(abs(order_der.logodds_FM1)>=
  quantile(abs(order_der.logodds_FM1), 0.8))

df_FM1<-
as.data.frame(data.matrix(order_der.logodds_FM1[v.influyentes_FM1,]))
row_names_FM1<- rownames(df_FM1)

library(ggplot2)
library(dplyr)

df_FM1_long <- reshape2::melt(df_FM1)
rownames(df_FM1_long)<- row_names_FM1


# Determine color based on sign
df_FM1_long$color <- ifelse(df_FM1_long$value <= 0, "red", "blue")

# Get absolute values
df_FM1_long$value <- abs(df_FM1_long$value)

# Sort the data frame by absolute value in descending order
df_FM1_long <- df_FM1_long[order(df_FM1_long$value, decreasing = TRUE), ]

ggplot_FM1<- ggplot(df_FM1_long, aes(x = reorder(row_names_FM1, -value),
y = value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
```

```
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 75, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds",
bold("(ELAc/MNI)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))

ggsave("grafico2.png", plot = ggplot_FM1, width = 8, height = 6)

v.influyentes_FM2<- which(abs(order_der.logodds_FM2)>=
  quantile(abs(order_der.logodds_FM2), 0.8))

df_FM2<-
as.data.frame(data.matrix(order_der.logodds_FM2[v.influyentes_FM2,]))
row_names_FM2<- rownames(df_FM2)

library(ggplot2)
library(dplyr)

df_FM2_long <- reshape2::melt(df_FM2)
rownames(df_FM2_long)<- row_names_FM2


# Determine color based on sign
df_FM2_long$color <- ifelse(df_FM2_long$value <= 0, "red", "blue")

# Get absolute values
df_FM2_long$value <- abs(df_FM2_long$value)

# Sort the data frame by absolute value in descending order
df_FM2_long <- df_FM2_long[order(df_FM2_long$value, decreasing = TRUE), ]

ggplot_FM2<- ggplot(df_FM2_long, aes(x = reorder(row_names_FM2, -value),
y = value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
```

```
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds",
bold("(ELAc/MNS)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))

ggsave("grafico3.png", plot = ggplot_FM2, width = 8, height = 6)

v.influyentes_FM3<- which(abs(order_der.logodds_FM3)>=
  quantile(abs(order_der.logodds_FM3), 0.8))

df_FM3<-
as.data.frame(data.matrix(order_der.logodds_FM3[v.influyentes_FM3,]))
row_names_FM3<- rownames(df_FM3)

library(ggplot2)
library(dplyr)

df_FM3_long <- reshape2::melt(df_FM3)
rownames(df_FM3_long)<- row_names_FM3


# Determine color based on sign
df_FM3_long$color <- ifelse(df_FM3_long$value <= 0, "red", "blue")

# Get absolute values
df_FM3_long$value <- abs(df_FM3_long$value)

# Sort the data frame by absolute value in descending order
df_FM3_long <- df_FM3_long[order(df_FM3_long$value, decreasing = TRUE), ]

ggplot_FM3<- ggplot(df_FM3_long, aes(x = reorder(row_names_FM3, -value),
y = value, fill = color)) +
  geom_bar(stat = "identity", width = 0.75) +
  scale_fill_manual(values = c("coral2", "cadetblue3"), labels =
c("positiva", "negativa"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.55, 1),
        legend.justification = c(1, 1),
        legend.box.just = "right") +
  labs(x = " ",
       y = "valor absoluto") +
  ggtitle(expression(atop("Derivada parcial de log-odds", bold("(MNI /
MNS)")))) +
  theme(plot.title = element_text(size = 11, hjust = 0.5, face =
"plain"))
```

```r
ggsave("graficoMNIvsMNS.png", plot = ggplot_FM3, width = 8, height = 6)

# marcadores que, excepcionalmente, son solo influyentes para MNI / MNS

E<- data.matrix(df_FM3[!rownames(df_FM3) %in% c(rownames(df_FM1),
rownames(df_FM2)),])

rownames(E)<- rownames(df_FM3)[!rownames(df_FM3) %in% c(rownames(df_FM1),
rownames(df_FM2))]

Ecomp<- cbind(E, der.logodds_FM1[rownames(E), ],
der.logodds_FM2[rownames(E), ])
colnames(Ecomp)<- c('MNI / MNS', 'ELAc / MNI', 'ELAc / MNS')

Ecomp <- kable(as.data.frame(round(Ecomp,2)))
Ecomp <- kable_styling(Ecomp, "striped", full_width = FALSE)

# boxplots de coeficientes de clasificaciÓn
library(ggplot2)

datosboxplotFM <- list(
  order_der.logodds_FM1 = order_der.logodds_FM1,
  order_der.logodds_FM2 = order_der.logodds_FM2,
  order_der.logodds_FM3 = order_der.logodds_FM3
)

df_boxplotFM <- data.frame(
  Grupo = rep(names(datosboxplotFM), lengths(datosboxplotFM)),
  Valores = unlist(datosboxplotFM)
)

ggplot(df_boxplotFM, aes(x = Grupo, y = Valores)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("ELAc / MNI", "ELAc / MNS", "MNI / MNS")) +
  xlab("") +
  ylab("")
```

4. DIFERENCIAS DE DISTRIBUCION POR CLASES: ESTADIO DIAGNOSTICO

```
## Previamente se habia mostrado que:

PCA.best_ED<-PCA(ela$`Estadio diagnostico`,allmarkers,v.exp)
lda.best_ED<- lda(ED~., data = PCA.best_ED$PC)

# matriz covarianzas estimada
covarianzas_nivel_ED <- lapply(1:3, function(k)
cov(PCA.best_ED$PC[ED==levels(ED)[k], ]))
M_ED<- lapply(1:3, function(k) covarianzas_nivel_ED[[k]]*(table(ED)[k]-
1))
S_ED<- Reduce('+', M_ED)/(sum(table(ED))-3)

# proyeccion de observaciones en el plano
X.proj_ED<-data.matrix(PCA.best_ED$PC)%*%lda.best_ED$scaling

# funciones para expresar como recta la frontera de decision entre clase
i vs. j
## especificar matriz varianza-covarianza y modelo LDA

P<- function(i,j){
  t(solve(S_ED)%*%(lda.best_ED$means[i,] - lda.best_ED$means[j,]))
}


d<- function(i,j,alpha){

0.5*(lda.best_ED$means[i,]%*%solve(S_ED)%*%data.matrix(lda.best_ED$means[
i,])-
lda.best_ED$means[j,]%*%solve(S_ED)%*%data.matrix(lda.best_ED$means[j,]))
  +log(lda.best_ED$prior[j]/lda.best_ED$prior[i]) + log(alpha)
}


round(
  (lda.best_ED$scaling%*%
     t(lda.best_ED$scaling)%*%solve(S_ED)%*%
     (lda.best_ED$means[1,] - lda.best_ED$means[3,]))
  - (solve(S_ED)%*%(lda.best_ED$means[1,] - lda.best_ED$means[3,])) ,2)

# Reduccion de dimensionalidad de los marcadores para ED

library(ggplot2)

data <- data.frame(X = X.proj_ED[, 1], Y = X.proj_ED[, 2], Class =
as.factor(ED))
```

```r
# Dibuja la grafica
ggplot(data, aes(x = X, y = Y, color = Class)) +
  geom_point(size = 1.5) +
  labs(x = "1 discriminante lineal (62%)", y = "2 discriminante lineal
(38%)",
       title = expression(atop("Reduccion de dimensionalidad de los
marcadores segun",
                               italic("Estadio Diagnostico")))) +
  scale_color_manual(values = c("#00AFBB", "#E7B800", "#FC4E07"),
                     name = "Estadio diagnostico") +
  theme(legend.title = element_text(face = "italic"),
        legend.position = "bottom") +
  geom_point(aes(x = ED1.media_proj[1], y = ED1.media_proj[2]), shape =
13,
             color = "#00AFBB", size = 6.5) +
  geom_point(aes(x = ED2.media_proj[1], y = ED2.media_proj[2]), shape =
13,
             color = "#E7B800", size = 6.5) +
  geom_point(aes(x = ED3.media_proj[1], y = ED3.media_proj[2]), shape =
13,
             color = "#FC4E07", size = 6.5)

# funcion distancia a frontera de decision (Ax=b) para X0
D<- function(A, b, X0){
  sapply(1:nrow(X0), function(i) (sum(A*X0[i,])-b)/sqrt(sum(A^2)) )
}

dist_ED<- D(P(1,3),d(1,3,1),PCA.best_ED$PC)


Cor_ED<- data.matrix(sapply(1:ncol(allmarkers), function(j)
  cor(dist_ED, allmarkers[PCA.best_ED$casoscomp,j])))
rownames(Cor_ED)<- colnames(allmarkers)

cbind(Cor_ED, data.matrix(sapply(1:ncol(allmarkers),
                                 function(j)
                                   cor(new_x,
allmarkers[PCA.best_ED$casoscomp,j]))))



order.Cor_ED<- data.matrix(Cor_ED[order(-abs(Cor_ED)),])

m.influyentes_ED<- abs(order.Cor_ED) >= quantile(abs(order.Cor_ED), 0.9)
sum(m.influyentes_ED)

# calculo distancia correlacion (entre 0 y 1; 0 --> independencia)
library(energy)
dCor_ED<- sapply(1:ncol(allmarkers), function(j)
```

```
  dcor(new_x, allmarkers[PCA.best_ED$casoscomp,j], index = 1.0))

dCor_ED<- data.matrix(dCor)
rownames(dCor_ED)<- colnames(allmarkers)
order.dCor_ED<- data.matrix(dCor_ED[order(-abs(dCor_ED)) , ])

# boxplot para abs(rho_Pearson) y dist correlacion

boxplot(list(abs(order.Cor_ED), order.dCor_ED))

dataCoefs_ED <- data.frame(Abs_Cor = abs(order.Cor_ED), dCor =
order.dCor_ED)

df_dataCoefs_ED <- data.frame(
  Grupo = rep(names(dataCoefs_ED), lengths(dataCoefs_ED)),
  Valores = unlist(dataCoefs_ED)
)

ggplot(df_dataCoefs_ED, aes(x = Grupo, y = Valores)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("Coeficiente correlacion de Pearson (valor
absoluto)",
                              "Distancia de correlacion")) +
  xlab("") +
  ylab("")

minCor_ED<- abs(order.Cor_ED)<quantile(abs(order.Cor_ED), 0.1)

minCor.dCor<- cbind(abs(order.Cor_ED)[minCor_ED,],
order.dCor_ED[minCor_ED,])
colnames(minCor.dCor)<- c('Rho de Pearson (valor absoluto)',
                          'Distancia de correlacion')

minCor.dCor <- kable(as.data.frame(round(minCor.dCor,5)))
minCor.dCor <- kable_styling(minCor.dCor, "striped", full_width = FALSE)

library(ggplot2)
library(dplyr)
library(tidyr)

df_minCor <- as.data.frame(minCor.dCor)

# Agregar una columna con los nombres de fila
df_minCor$Marcador <- rownames(df_minCor)

# Ordenar el data frame por Rho de Pearson en orden descendente
df_minCor_long <- df_minCor %>%
  tidyr::gather(key = "Tipo de correlacion", value = "Valor", -Marcador)
%>%
  arrange(desc(abs(Valor)), `Tipo de correlacion`)
```

```r
ggplot(df_minCor_long, aes(x = reorder(Marcador, -Valor), y = abs(Valor),
                           fill = `Tipo de correlacion`)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.75) +
  scale_fill_manual(values = c("burlywood3", "cadetblue3"),
                    labels = c("Distancia de correlacion","Rho de
Pearson"),
                    guide = guide_legend(title = NULL)) +
  theme(axis.text.x = element_text(angle = 68, hjust = 1, size = 8),
        axis.title.y = element_text(size = 9),
        legend.position = c(0.5, 0.5),
        legend.justification = c(0.5, 0.5),
        legend.box.just = "center",
        legend.text = element_text(size = 8),
        legend.background = element_rect(fill = "transparent", colour =
NA),
        legend.box.background = element_rect(fill = "white", colour =
"black")) +
  labs(x = " ",
       y = " ")

# plots de posibles FN (valores proximos a 0 para rho Pearson)

posibles_FN_ED <- rownames(order.Cor_ED)[abs(order.Cor_ED) < 0.01]

library(ggplot2)
library(gridExtra)

n_plots_ED <- length(posibles_FN_ED)
n_cols_ED <- 2
n_rows_ED <- ceiling(n_plots_ED / n_cols_ED)

# Crear una lista para almacenar los graficos
plots_ED <- list()

for (n_ED in 1:n_plots_ED) {
  # Crear un nuevo dataframe con los datos necesarios para el grafico
  plot_data_ED <- data.frame(x_ED = new_x,
                             y_ED = allmarkers[PCA.best_ED$casoscomp,
                                               posibles_FN_ED[n_ED]])

  # Crear el grafico utilizando ggplot2
  p_ED <- ggplot(plot_data_ED, aes(x = x_ED, y = y_ED)) +
    geom_point() +
    labs(x = NULL, y = '') +
    theme_minimal() +
    theme(axis.title.y = element_text(face = "bold")) +
    ggtitle(label = posibles_FN_ED[n_ED], subtitle = "") +
    theme(plot.title = element_text(face = "bold"))
```

```r
  # Almacenar el grafico en la lista
  plots_ED[[n_ED]] <- p_ED
}

grid_plot_ED <- grid.arrange(grobs = plots_ED, ncol = n_cols_ED)

grid_plot_ED$top <- NULL

ggsave("grafico_combinado_ED.png", grid_plot_ED, width = 12, height = 8)

# boxplots de marcadores con mayor correlacion con distancia entre
estadio 3 y 1, por estadios

ggplotCorED <- function(a) { ## a=percentil considerado
  df_m.influyentes_ED <- allmarkers[PCA.best_ED$casoscomp,
                           rownames(abs(order.Cor_ED))[abs(order.Cor_ED) >

quantile(abs(order.Cor_ED),a)]]

  lista_df <- list(df_m.influyentes_ED[ED == '1', ],
df_m.influyentes_ED[ED == '3', ])

  library(ggplot2)
  library(tidyr)

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.numeric(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  ggplot(df_largo, aes(x = as.factor(Clase), y = Valor, fill =
as.factor(Clase))) +
    geom_boxplot() +
    facet_wrap(~ Variable, ncol = 2, scales = "free_y") +
    xlab(NULL) +
    ylab(NULL) +
    ggtitle(" ") +
    scale_x_discrete(labels = c("Estadio 1", "Estadio 3")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01),
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
```

```
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "none")
}


ggplotCorED(0.92)

ggplotQQ_ED <- function(a) {
  df_m.influyentes_ED <- allmarkers[PCA.best_ED$casoscomp,
                           rownames(abs(order.Cor_ED))[abs(order.Cor_ED) >

quantile(abs(order.Cor_ED),a)]]

  lista_df <- list(df_m.influyentes_ED[ED == '1', ],
df_m.influyentes_ED[ED == '3', ])

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.factor(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  percentiles <- seq(5, 95, by = 5)

  df_largo <- df_largo %>%
    group_by(Clase, Variable) %>%
    mutate(Quantiles = list(map_dfr(percentiles,
                                    ~ data.frame(Quantile = .x,
                                                 Value = quantile(Valor,
                                                                  probs =
.x/100))))) %>%
    unnest(Quantiles)

 ggplot(df_largo, aes(x = Quantile, y = Value, colour = Clase)) +
    geom_point() +
    facet_wrap(~ Variable, ncol = 2, scales = "free_y") +
    xlab("Percentil") +
    ylab(" ") +
    ggtitle(" ") +
    scale_colour_discrete(name = "Estadio", labels = c("1", "3")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01, size = 8),
          axis.text.y = element_text(size = 5),
```

```
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "top",   # Posicion de la leyenda
          legend.direction = "horizontal")
}

ggplotQQ_ED(0.92)

ggplotDensity <- function(a) { ## a=percentil considerado
  df_m.influyentes_ED <- allmarkers[PCA.best_ED$casoscomp,
                           rownames(abs(order.Cor_ED))[abs(order.Cor_ED) >

quantile(abs(order.Cor_ED),a)]]

  lista_df <- list(df_m.influyentes_ED[ED == '1', ],
df_m.influyentes_ED[ED == '3', ])

  library(ggplot2)
  library(tidyr)

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.numeric(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  ggplot(df_largo, aes(x = Valor, fill = as.factor(Clase))) +
    geom_density(alpha = 0.5) +
    facet_wrap(~ Variable, ncol = 2, scales = "free") +
    xlab(NULL) +
    ylab(NULL) +
    ggtitle(" ") +
    scale_x_discrete(labels = c("Estadio 1", "Estadio 3")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01),
                    axis.text.y = element_text(size = 5),

          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
```

```
"lines"),
          legend.position = "topright")
}

ggplotDensity(0.92)

##### Wilcoxon

WT_ED<- lapply(rownames(order.Cor_ED), function(j)
  wilcox.test(allmarkers[PCA.best_ED$casoscomp,j][ED=='3'],
       allmarkers[PCA.best_ED$casoscomp,j][ED=='1']))


pED<- sapply(rownames(order.Cor_ED), function(j)
  wilcox.test(allmarkers[PCA.best_ED$casoscomp,j][ED=='3'],
       allmarkers[PCA.best_ED$casoscomp,j][ED=='1'])$p.value)

WED<- sapply(rownames(order.Cor_ED), function(j)
  wilcox.test(allmarkers[PCA.best_ED$casoscomp,j][ED=='3'],
       allmarkers[PCA.best_ED$casoscomp,j][ED=='1'])[[1]])



library(ggplot2)
library(dplyr)

df_pED<- data.frame(pvalor=pED, Rho=abs(order.Cor_ED))


ggplot(df_pED, aes(x = Rho, y = pvalor)) +
  geom_point() +
  geom_smooth(method = "loess", se=F) +
  labs(x = "Rho de Pearson (valor absoluto)", y = "p-valor")


# Effectsize

#install.packages('coin')

Z_ED<- sapply(rownames(order.Cor_ED),function(j)

abs(statistic(coin::wilcox_test(allmarkers[PCA.best_ED$casoscomp,j][!ED==
'2']~
                                    ED[!ED=='2']))))

effsizeED<- Z_ED/sqrt(sum(ED=='3')+sum(ED=='1'))

df_effsizeED<- data.frame(effsizeED= effsizeED, Rho=abs(order.Cor_ED))

ggplot(df_effsizeED, aes(x=Rho, y=effsizeED)) +
```

```
  geom_point() +
  geom_smooth(method="loess",se=F) +
  labs(x='Rho de Pearson (valor absoluto)', y='Tamano de efecto')

# modelos de prediccion del pronostico

modED.d<- lm(ela$`ALSFRS-R`[PCA.best_ED$casoscomp]~ new_x)
summary(modED.d)
modED.d_df<- data.frame(cbind(ela$`ALSFRS-R`[PCA.best_ED$casoscomp],
new_x))
colnames(modED.d_df)<- c('ALSFRS', 'distancia')
train(ALSFRS~distancia, data = modED.d_df,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian') # rmse 5.55
```

5. DIFERENCIAS EN DISTRIBUCIÓN DE MARCADORES POR CLASE, ESTADIO DIAGNÓSTICO Y DETERIORO COGNITIVO COMBINADOS

```
ela$`Deterioro cognitivo`[ela$`Deterioro cognitivo`=='NA']<- NA

detcog<- as.factor(ela$`Deterioro cognitivo`)
detcog[detcog=='DFT']<-NA

DCED<-data.frame(index= 1:125, int=interaction(ela$`Estadio diagnÓstico`,
detcog))

DCED<- droplevels.data.frame(DCED)

## Medidas de capacidad predictiva


cv6<- function(factor, nfolds, PCA, AD='lda'){

y<- as.factor(factor[PCA$casoscomp])

sepFolds<-lapply(1:length(levels(y)), function(x)
  {createFolds(y[y==levels(y)[x]], k=nfolds, returnTrain = FALSE)} )

Folds<- list()
for (k in 1:nfolds) {
  Folds[[k]]<- c(which(y==levels(y)[1])[unlist(sepFolds[[1]][k])],
  which(y==levels(y)[2])[unlist(sepFolds[[2]][k])],
  which(y==levels(y)[3])[unlist(sepFolds[[3]][k])],
  which(y==levels(y)[4])[unlist(sepFolds[[4]][k])],
  which(y==levels(y)[5])[unlist(sepFolds[[5]][k])],
  which(y==levels(y)[6])[unlist(sepFolds[[6]][k])]
)
}

Accuracy<- c()
Sensitivity<- matrix(ncol=6, nrow =0 )
Specificity<- matrix(ncol=6, nrow =0 )
AUC<- c()

X<- data.frame(PCA$PC)

for (i in 1:nfolds) {

  test_indices <- Folds[[i]]

  X_train <- data.frame(X[-test_indices, ])
  colnames(X_train)<- colnames(PCA$PC)
```

```r
  y_train <- y[-test_indices]

  X_test <- data.frame(X[test_indices, ])
  colnames(X_test)<- colnames(PCA$PC)
  y_test <- y[test_indices]

  # Ajustar el modelo
  if (AD=='lda'){
      mod<- lda(y_train ~., data= X_train)
  }

  if (AD=='qda'){
      mod<- qda(y_train ~., data= X_train)
  }

  # Hacer predicciones en test usando el modelo ajustado
  y_pred <- predict(mod, newdata = X_test)

  # Calcular errores
  cm <- confusionMatrix(y_pred$class, y_test)
  accuracy<- sum(diag(cm$table))/sum(cm$table)

  sensitivity <- cm$byClass[, "Sensitivity"]
  specificity <- cm$byClass[, "Specificity"]

  auc<- multiclass.roc(y_test, y_pred$posterior)
  auc_value <- sub(".*: ", "", auc$auc)
  auc<- as.numeric(auc_value)

  Accuracy<- c(Accuracy, accuracy)
  Sensitivity<- rbind(Sensitivity, sensitivity)
  Specificity<- rbind(Specificity, specificity)
  AUC<- c(AUC, auc)

}

# Media de cada error
Accuracy_CV<- mean(Accuracy, na.rm=T)
Sensitivity_CV<- sapply(1:6, function(x) mean(Sensitivity[,x], na.rm=T))
Specificity_CV<- sapply(1:6, function(x) mean(Specificity[,x], na.rm=T))
AUC_CV<- mean(AUC, na.rm=T)

Error_average<-
list(Accuracy_CV=Accuracy_CV,AUC_CV=AUC_CV,rbind(Sensitivity_CV,
Specificity_CV))

return(Error_average)
}

#cv6(DCED$int, 5,PCA(DCED$int,allmarkers,0.9)  , AD="lda")
```

```r
repeated_CV6<- function(factor, nfolds, ntimes, PCA, AD){

M<- list(c(0), c(0), matrix(rep(0,12),nrow = 2, ncol = 6))

for (i in 1:ntimes) {

set.seed(i)
CV<- cv6(factor, nfolds, PCA, AD)

M<- lapply(1:3, function(x) M[[x]] + CV[[x]])

}

M<- lapply(1:3, function(x) M[[x]]/ntimes)


return(M)
}

#repeated_CV6(DCED$int, 5,3,PCA(DCED$int,allmarkers,0.9)  , AD="lda")

## error de precisiÓn en CV segÚn varianza explicada (num. PC)

PCA_DCED<- PCA(DCED$int, allmarkers, 0.9)

Var_DCED<- summary(PCA_DCED$prcomp)$importance[3, ]


CVlda_var_DCED<- lapply(Var_DCED,function(v) {repeated_CV6(factor =
DCED$int,
            nfolds = 5, ntimes = 3,
            PCA(DCED$int,allmarkers,v), AD='lda')})


PC.optim_lda_DCED<- which.max(unlist(lapply(1:length(CVlda_var_DCED),
function(i) CVlda_var_DCED[[i]][2]))) # 1 PC da mejor resultados
predictivos

CVlda_var_DCED[[PC.optim_lda_DCED]]


## Mejor modelo
v.exp_DCED<- Var_DCED[PC.optim_lda_DCED]
PCA.best_DCED<-PCA(DCED$int,allmarkers,v.exp_DCED) # mejor PCA

Dced<- as.factor(DCED$int[PCA.best_DCED$casoscomp])
lda.best_DCED<- lda(Dced~.,data = PCA.best_DCED$PC) # mejor modelo lda

## CÁlculo frontera (punto) de decisiÓn
```

```r
DF<-cbind(Dced,  PCA.best_DCED$PC)
LDA_DCED<- LDA(Dced~., data=DF, output = 'Discriminant Functions')


front_DCED<- (LDA_DCED$original$discriminant.functions[1 ,4] -
LDA_DCED$original$discriminant.functions[1 ,3])/
(LDA_DCED$original$discriminant.functions[2 ,3] -
LDA_DCED$original$discriminant.functions[2 ,4])

dist_DCED<- PCA.best_DCED$PC- as.numeric(front_DCED)


##################

df_DCED<- cbind(Dced, PCA.best_DCED$PC)
colnames(df_DCED)<- c('Estadio.Deterioro', 'PC')

library(ggplot2)

# GrÁfico de dispersiÓn con línea horizontal y color por clase
ggplot(df_DCED, aes(x = Estadio.Deterioro, y = PC, color = Dced)) +
  geom_point() +
  geom_hline(aes(yintercept = front_DCED),
             linetype = "dashed", color = "red") +
  labs(x = " ", y = "1º componente principal", title = "") +
  theme_minimal()+
    theme(legend.position = "none")

Cor_DCED<- data.matrix(sapply(1:ncol(allmarkers), function(j)
cor(dist_DCED, allmarkers[PCA.best_DCED$casoscomp,j])))
rownames(Cor_DCED)<- colnames(allmarkers)

order.Cor_DCED<- data.matrix(Cor_DCED[order(-abs(Cor_DCED)),])

m.influyentes_DCED<- abs(order.Cor_DCED) >= quantile(abs(order.Cor_DCED),
0.9)
sum(m.influyentes_DCED)


# dist correlaciÓn

dCor_DCED<- sapply(1:ncol(allmarkers), function(j)
  dcor(dist_DCED, allmarkers[PCA.best_DCED$casoscomp,j], index = 1.0))

dCor_DCED<- data.matrix(dCor_DCED)
rownames(dCor_DCED)<- colnames(allmarkers)
order.dCor_DCED<- data.matrix(dCor_DCED[order(-abs(dCor_DCED)) , ])

ggplot(data.frame(abs(order.dCor_DCED)), aes(x = "", y =
abs(order.dCor_DCED)))+
```

```r
  geom_boxplot() +
  labs(x = "", y = "Distancia de correlaciÓn")

# boxplot para abs(rho_Pearson): estadio diagnÓstico vs. combinaciÓn
estadio diagnÓstico + DC


dataCoefs_ED <- data.frame(Abs_Cor_ED = abs(order.Cor_ED), Abs_Cor_DCED =
abs(order.Cor_DCED))

df_dataCoefs_ED <- data.frame(
  Grupo = rep(names(dataCoefs_ED), lengths(dataCoefs_ED)),
  Valores = unlist(dataCoefs_ED)
)

ggplot(df_dataCoefs_ED, aes(x = Grupo, y = Valores)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("CombinaciÓn Estadio DiagnÓstico y
Deterioro Cognitivo ",
                              "Estadio DiagnÓstico")) +
  xlab("") +
  ylab("Rho de Pearson (valor absoluto)")



ggplotQQ_DCED <- function(a) {
  df_m.influyentes_DCED <- allmarkers[PCA.best_DCED$casoscomp,

rownames(abs(order.Cor_DCED))[abs(order.Cor_DCED) >
quantile(abs(order.Cor_DCED),a)]]

  lista_df <- list(df_m.influyentes_DCED[Dced == '1.NO', ],
df_m.influyentes_DCED[Dced == '3.LEVE', ])

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.factor(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  percentiles <- seq(5, 95, by = 5)

  df_largo <- df_largo %>%
    group_by(Clase, Variable) %>%
```

```
    mutate(Quantiles = list(map_dfr(percentiles, ~ data.frame(Quantile =
.x, Value = quantile(Valor, probs = .x/100))))) %>%
    unnest(Quantiles)

 ggplot(df_largo, aes(x = Quantile, y = Value, colour = Clase)) +
    geom_point() +
    facet_wrap(~ Variable, ncol = 2, scales = "free_y") +
    xlab("Percentil") +
    ylab(" ") +
    ggtitle(" ") +
    scale_colour_discrete(name = "Estadio", labels = c("Inicial",
"Final")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01, size = 8),
          axis.text.y = element_text(size = 5),
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "none",   # PosiciÓn de la leyenda
          legend.direction = "horizontal")
}

ggplotQQ_DCED(0.92)

ggplotDensity_DCED <- function(a) { ## a=percentil considerado
  df_m.influyentes_DCED <- allmarkers[PCA.best_DCED$casoscomp,

rownames(abs(order.Cor_DCED))[abs(order.Cor_DCED) >
quantile(abs(order.Cor_DCED),a)]]

  lista_df <- list(df_m.influyentes_DCED[Dced == '1.NO', ],
df_m.influyentes_DCED[Dced == '3.LEVE', ])

  library(ggplot2)
  library(tidyr)

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.numeric(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)
```

```
  ggplot(df_largo, aes(x = Valor, fill = as.factor(Clase))) +
    geom_density(alpha = 0.5) +
    facet_wrap(~ Variable, ncol = 2, scales = "free") +
    xlab(NULL) +
    ylab(NULL) +
    ggtitle(" ") +
    scale_x_discrete(labels = c("Estadio 1", "Estadio 3")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01),
          axis.text.y = element_text(size = 5),
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "topright")
}

ggplotDensity_DCED(0.92)

WT_DCED<- lapply(rownames(order.Cor_DCED), function(j)
wilcox.test(allmarkers[PCA.best_DCED$casoscomp,j][Dced=='3.LEVE'],
       allmarkers[PCA.best_DCED$casoscomp,j][Dced=='1.NO']))

pDCED<- sapply(rownames(order.Cor_DCED), function(j)
wilcox.test(allmarkers[PCA.best_DCED$casoscomp,j][Dced=='3.LEVE'],
       allmarkers[PCA.best_DCED$casoscomp,j][Dced=='1.NO'])$p.value)
# pDCED[1:18][pDCED[1:18]<0.05/18]

W_DCED<- sapply(rownames(order.Cor_DCED), function(j)
wilcox.test(allmarkers[PCA.best_DCED$casoscomp,j][Dced=='3.LEVE'],
       allmarkers[PCA.best_DCED$casoscomp,j][Dced=='1.NO'])[[1]])



library(ggplot2)
library(dplyr)

df_pDCED<- data.frame(pvalor=pDCED, Rho=abs(order.Cor_DCED))


ggplot(df_pDCED, aes(x = Rho, y = pvalor)) +
  geom_point() +
  geom_smooth(method = "loess", se=F) +
  labs(x = "Rho de Pearson (valor absoluto)", y = "p-valor")


# tamaÑo de efecto

#install.packages('coin')
```

```r
Z_DCED<- sapply(rownames(order.Cor_DCED),function(j)

abs(statistic(coin::wilcox_test(allmarkers[PCA.best_DCED$casoscomp,j][Dce
d=='3.LEVE' | Dced=='1.NO']~Dced[Dced=='3.LEVE' | Dced=="1.NO"]))))

effsize_DCED<- Z_DCED/sqrt(sum(Dced=="3.LEVE")+sum(Dced=="1.NO"))

df_effsize_DCED<- data.frame(effsize_DCED= effsize_DCED,
Rho=abs(order.Cor_DCED))

ggplot(df_effsize_DCED, aes(x=Rho, y=effsize_DCED)) +
  geom_point() +
  geom_smooth(method="loess",se=F) +
  labs(x='Rho de Pearson (valor absoluto)', y='TamaÑo de efecto')

list(data.frame(effsize_DCED), data.frame(data.frame(effsizeED)))

library(ggplot2)
library(dplyr)

# Convertir a dataframes
df1 <- data.frame(effsize_DCED)
df2 <- data.frame(effsizeED[rownames(df1)])


# Calcula las diferencias
df_diff <- df1 - df2

# AÑade una columna con los nombres de las filas
df_diff$RowNames <- rownames(df_diff)

# Convierte el dataframe a formato largo
df_diff_long <- df_diff %>%
    tidyr::pivot_longer(cols = -RowNames, names_to = "Variable",
values_to = "Difference")

ggplot(df_diff_long, aes(x = RowNames, y = Difference, fill = Variable))
+
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(x = NULL, y = " ") +
  theme(
    axis.text.x = element_blank(), # Elimina las etiquetas del eje X
    axis.ticks.x = element_blank(), # Elimina las marcas del eje X
    legend.position = "none" # Elimina la leyenda
  )

# Convierte a dataframes
df1 <- data.frame(effsize_DCED)
df2 <- data.frame(effsizeED)
```

```r
# Asegurar que los dos dataframes tienen el mismo nÚmero de filas
if(nrow(df1) != nrow(df2)) {
  stop("Los dataframes no tienen el mismo nÚmero de filas")
}

# Cambiar los dataframes a formato largo
df1_long <- df1 %>%
  tidyr::pivot_longer(everything(), names_to = "Variable", values_to =
"Value") %>%
  mutate(Source = "Estadio diagnÓstico + Deterioro Cognitivo")

df2_long <- df2 %>%
  tidyr::pivot_longer(everything(), names_to = "Variable", values_to =
"Value") %>%
  mutate(Source = "Estadio diagnÓstico")

# Combina los dataframes
df_combined <- rbind(df1_long, df2_long)

# Crea los boxplots
ggplot(df_combined, aes(x = Source, y = Value, fill = Source)) +
  geom_boxplot() +
  labs(x = NULL, y = "TamaÑo de efecto", fill = " ") +
  theme(axis.text.x = element_text(angle = 0, vjust = 0.5, hjust = 1),
                  legend.position = "top",   # PosiciÓn de la leyenda
          legend.direction = "horizontal")
```

6. DIFERENCIAS EN DISTRIBUCIÓN DE MARCADORES POR CLASES: MUTACIÓN C9ORF72

```
## Modelo LDA con mejor rendimiento predictivo

PCA.best_MUT<- PCA(Genotipo.bin, allmarkers, Var[PC.optim_lda])
Gbin<- Genotipo.bin[PCA.best_MUT$casoscomp]

lda.best_MUT<- lda(Gbin~., data = PCA.best_MUT$PC)

# matriz covarianzas estimada
covarianzas_nivel_MUT<- lapply(1:2, function(k)
cov(PCA.best_MUT$PC[Gbin==levels(Gbin)[k], ]))

M_MUT<- lapply(1:2, function(k)
covarianzas_nivel_MUT[[k]]*(table(Gbin)[k]-1))
S_MUT<- Reduce('+', M_MUT)/(sum(table(Gbin))-2)

# proyección de observaciones en el plano
X.proj_MUT<-data.matrix(PCA.best_MUT$PC)%*%lda.best_MUT$scaling

# funciones para calcular frontera de decisión entre clase i vs. j
(punto)
## especificar matriz varianza-covarianza y modelo LDA


P_MUT<- function(l,k){
  t(solve(S_MUT)%*%(lda.best_MUT$means[k,] - lda.best_MUT$means[l,]))
}

d_MUT<- function(l,k,alpha){
  0.5*(lda.best_MUT$means[k,]+ lda.best_MUT$means[l,])%*% solve(S_MUT)%*%
(lda.best_MUT$means[k,]- lda.best_MUT$means[l,])-
log(lda.best_MUT$prior[k]/lda.best_MUT$prior[l]) + log(alpha)
  }


d_MUT(1,2,1)/P_MUT(1,2) %*% lda.best_MUT$scaling

num.LD<-1
L<- matrix(lda.best_MUT$scaling, num.LD, PCA.best_MUT$numPC)
A<- matrix(P_MUT(1,2), PCA.best_MUT$numPC, 1)
c<- as.numeric(d_MUT(1,2,1))


front_aproxMUT<- c*L%*%t(L) / L%*%A
```

```r
t(lda.best_ED$scaling[,2]) %*%lda.best_ED$scaling[,1]

df_MUT<- data.frame(Gbin, X.proj_MUT)
colnames(df_MUT)<- c('clase', 'LD1')

library(ggplot2)

# Gráfico de dispersión con línea horizontal y color por clase
ggplot(df_MUT, aes(x = clase, y = LD1, color = Gbin)) +
  geom_point() +
  geom_hline(aes(yintercept = front_aproxMUT,
             linetype = "dashed", color = "red")) +
  labs(x = " ", y = "1ª discriminante lineal", title = "") +
  theme_minimal()+
    theme(legend.position = "none")

# función distancia a frontera de decisión (Ax=b) para X0
D<- function(A, b, X0){
  sapply(1:nrow(X0), function(i) (sum(A*X0[i,])-b)/sqrt(sum(A^2)) )
}

dist_MUT<- D(P(1,2),d(1,2,1),PCA.best_MUT$PC)


Cor_MUT<- data.matrix(sapply(1:ncol(allmarkers), function(j)
cor(dist_MUT, allmarkers[PCA.best_MUT$casoscomp,j])))
rownames(Cor_MUT)<- colnames(allmarkers)

order.Cor_MUT<- data.matrix(Cor_MUT[order(-abs(Cor_MUT)),])

m.influyentes_MUT<- abs(order.Cor_MUT) >= quantile(abs(order.Cor_MUT),
0.9)
sum(m.influyentes_MUT)

# boxplot para abs(rho_Pearson): estadio diagnóstico vs. combinación
estadio diagnóstico + DC


dataCoefs_MUT <- data.frame(Abs_Cor_MUT = abs(order.Cor_MUT))

ggplot(dataCoefs_MUT, aes(y = Abs_Cor_MUT)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("Mutación C9Orf72")) +
  xlab("Mutación C9Orf72") +
  ylab("Rho de Pearson (valor absoluto)")

ggplotQQ_MUT <- function(a) {
  df_m.influyentes_MUT <- allmarkers[PCA.best_MUT$casoscomp,
                          rownames(abs(order.Cor_MUT))[abs(order.Cor_MUT)
> quantile(abs(order.Cor_MUT),a)]]
```

```
  lista_df <- list(df_m.influyentes_MUT[Gbin == levels(Gbin)[1], ],
df_m.influyentes_MUT[Gbin == levels(Gbin)[2], ])

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.factor(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  percentiles <- seq(5, 95, by = 5)

  df_largo <- df_largo %>%
    group_by(Clase, Variable) %>%
    mutate(Quantiles = list(map_dfr(percentiles, ~ data.frame(Quantile =
.x, Value = quantile(Valor, probs = .x/100))))) %>%
    unnest(Quantiles)

 ggplot(df_largo, aes(x = Quantile, y = Value, colour = Clase)) +
    geom_point() +
    facet_wrap(~ Variable, ncol = 2, scales = "free_y") +
    xlab("Percentil") +
    ylab(" ") +
    ggtitle(" ") +
    scale_colour_discrete(name = "Mutación", labels = c("C9orf72", "No
C9orf72")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01, size = 8),
          axis.text.y = element_text(size = 5),
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "top",   # Posición de la leyenda
          legend.direction = "horizontal")
}

ggplotQQ_MUT(0.92)


summary(allmarkers[PCA.best_MUT$casoscomp,rownames(order.Cor_MUT)[1]][Gbi
n=='C9ORF72'])
```

```r
summary(allmarkers[PCA.best_MUT$casoscomp,rownames(order.Cor_MUT)[1]][Gbi
n=='No.C9ORF72'])

ggplotDensity_MUT <- function(a) { ## a=percentil considerado
  df_m.influyentes_MUT <- allmarkers[PCA.best_MUT$casoscomp,
                          rownames(abs(order.Cor_MUT))[abs(order.Cor_MUT)
> quantile(abs(order.Cor_MUT),a)]]

  lista_df <- list(df_m.influyentes_MUT[Gbin == levels(Gbin)[1], ],
df_m.influyentes_MUT[Gbin == levels(Gbin)[2], ])

  library(ggplot2)
  library(tidyr)

  df_combinado <- bind_rows(lista_df, .id = "Clase")

  df_combinado$Clase <- as.numeric(df_combinado$Clase)

  df_largo <- df_combinado %>%
    pivot_longer(cols = -c(Clase), names_to = "Variable", values_to =
"Valor")

  orden_variables <- unique(df_largo$Variable)

  df_largo$Variable <- factor(df_largo$Variable, levels =
orden_variables)

  ggplot(df_largo, aes(x = Valor, fill = as.factor(Clase))) +
    geom_density(alpha = 0.5) +
    facet_wrap(~ Variable, ncol = 2, scales = "free") +
    xlab(NULL) +
    ylab(NULL) +
    ggtitle(" ") +
    scale_x_discrete(labels = c("Estadio 1", "Estadio 3")) +
    theme(axis.text.x = element_text(angle = 0, hjust = 1.01),
          axis.text.y = element_text(size = 5),
          axis.ticks.x = element_blank(),
          strip.text = element_text(size = 8.5, hjust = 0),
          strip.placement = "outside",
          plot.margin = margin(t = 1.5, r = 1.5, b = 1.5, l = 1.5, unit =
"lines"),
          legend.position = "topright")
}

ggplotDensity_MUT(0.92)

WT_MUT<- lapply(rownames(order.Cor_MUT), function(j)
wilcox.test(allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[2]],
       allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[1]]))
```

```r
pMUT<- sapply(rownames(order.Cor_MUT), function(j)
wilcox.test(allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[2]],

allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[1]])$p.value)


W_MUT<- sapply(rownames(order.Cor_MUT), function(j)
wilcox.test(allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[2]],
       allmarkers[PCA.best_MUT$casoscomp,j][Gbin==levels(Gbin)[1]])[[1]])



library(ggplot2)
library(dplyr)

df_pMUT<- data.frame(pvalor=pMUT, Rho=abs(order.Cor_MUT))


ggplot(df_pMUT, aes(x = Rho, y = pvalor)) +
  geom_point() +
  geom_smooth(method = "loess", se=F) +
  labs(x = "Rho de Pearson (valor absoluto)", y = "p-valor")


# tamaño de efecto

#install.packages('coin')

Z_MUT<- sapply(rownames(order.Cor_MUT),function(j)

abs(statistic(coin::wilcox_test(allmarkers[PCA.best_MUT$casoscomp,j]~Gbin
))))

effsize_MUT<-
Z_MUT/sqrt(sum(Gbin==levels(Gbin)[2])+sum(Gbin==levels(Gbin)[1]))

df_effsize_MUT<- data.frame(effsize_MUT= effsize_MUT,
Rho=abs(order.Cor_MUT))

ggplot(df_effsize_MUT, aes(x=Rho, y=effsize_MUT)) +
  geom_point() +
  geom_smooth(method="loess",se=F) +
  labs(x='Rho de Pearson (valor absoluto)', y='Tamaño de efecto')

df_effsize_MUT

Z_MUT2<-sapply(1:ncol(allmarkers),function(j)
  abs(statistic(coin::wilcox_test(allmarkers[PCA.best_MUT$casoscomp
,j]~Gbin))))
```

```
effsize_MUT2<-Z_MUT2/sqrt(length(Gbin))

names(effsize_MUT2)<- colnames(allmarkers)

cbind(data.matrix(effsize_MUT2),
data.matrix(data.matrix(effsize_MUT)[rownames(data.matrix(effsize_MUT2)),
]))

order.Cor_MUT
```

Marcadores con menor solapamiento entre clases (resumen para todas las variables categóricas estudiadas)

```
TEbig_ED<- data.matrix(effsizeED[effsizeED>quantile(effsizeED,0.9)])
TEbig_ED<- as.data.frame(TEbig_ED[order(-TEbig_ED),])
colnames(TEbig_ED)<- 'Tamaño de Efecto'
TEbig_ED <- kable(TEbig_ED,
                  caption = cell_spec('Estadio Diagnóstico (1 vs. 3)',
"html", color = "coral", bold = TRUE,
                                      font_size = 16)
                    )
TEbig_ED<- kable_styling(TEbig_ED, "striped", full_width = FALSE)
TEbig_ED


TEbig_DCED<-
data.matrix(effsize_DCED[effsize_DCED>quantile(effsize_DCED,0.9)])
TEbig_DCED<- as.data.frame(TEbig_DCED[order(-TEbig_DCED),])
colnames(TEbig_DCED)<- 'Tamaño de Efecto'
TEbig_DCED <- kable(TEbig_DCED,
                  caption = cell_spec('Estadio Diagnóstico + Deterioro
Cognitivo (inicial vs. final)', "html", color = "coral", bold = TRUE,
                                      font_size = 16)
                    )
TEbig_DCED<- kable_styling(TEbig_DCED, "striped", full_width = FALSE)
TEbig_DCED


TEbig_MUT<-
data.matrix(effsize_MUT[effsize_MUT>quantile(effsize_MUT,0.9)])
TEbig_MUT<- as.data.frame(TEbig_MUT[order(-TEbig_MUT),])
colnames(TEbig_MUT)<- 'Tamaño de Efecto'
TEbig_MUT <- kable(TEbig_MUT,
                  caption = cell_spec('Mutación (C9orf72 vs. No
C9orf72)', "html", color = "coral", bold = TRUE,
                                      font_size = 16)
                    )
TEbig_MUT<- kable_styling(TEbig_MUT, "striped", full_width = FALSE)
TEbig_MUT
```

```
data.matrix(effsizeED)
data.matrix(effsize_DCED)

intersect(c(rownames(data.matrix(effsizeED[effsizeED>quantile(effsizeED,0
.9)])),

rownames(data.matrix(effsize_DCED[effsize_DCED>quantile(effsize_DCED,0.9)
]))),

rownames(data.matrix(effsize_MUT[effsize_MUT>quantile(effsize_MUT,0.9)]))
)

TEvsRho_ED<- summary(lm(effsizeED~ abs(Cor_ED)[names(effsizeED),]))
TEvsRho_DCED<- summary(lm(effsize_DCED~
abs(Cor_DCED)[names(effsize_DCED),]))
TEvsRho_MUT<- summary(lm(effsize_MUT~ abs(Cor_MUT)[names(effsize_MUT),]))
```

7. MODELIZACIÓN DE VARIABLES PRONÓSTICO

```
px_ED<- cbind(X.proj_ED, dist_ED,
              D(P(1,2),d(1,2,1), PCA.best_ED$PC),
              D(P(2,3),d(2,3,1), PCA.best_ED$PC)
              )
colnames(px_ED)<- c('LD1_ED', 'LD2_ED', 'dist_ED13', 'dist_ED12',
'dist_ED23')


px_DCED<- cbind(PCA.best_DCED$PC, dist_DCED); colnames(px_DCED)<-
c('PC1_DCED', 'dist_DCED')
px_MUT<- data.frame(X.proj_MUT, dist_MUT, ); colnames(px_MUT)<-
c('LD1_MUT', 'dist_MUT')


addNA<- function(original,casoscomp){
 original<- as.matrix(original)
  matrixNA<- matrix(NA, nrow(allmarkers)-nrow(original), ncol(original));
  colnames(matrixNA)<- colnames(original)

originalc<- rbind(original,matrixNA)
rownames(originalc)[complete.cases(originalc)]<- which(casoscomp)
rownames(originalc)[!complete.cases(originalc)]<- which(!casoscomp)
return(originalc)
}

addNA(px_ED, PCA.best_ED$casoscomp)
addNA(px_DCED, PCA.best_DCED$casoscomp)
addNA(px_MUT, PCA.best_MUT$casoscomp)


predpx<- merge(
  merge(addNA(px_ED, PCA.best_ED$casoscomp),
      addNA(px_DCED, PCA.best_DCED$casoscomp),
          by = "row.names", all = F)[,-1],

  addNA(px_MUT, PCA.best_MUT$casoscomp),
  by = "row.names", all = F
)[,-1]
#predpx<- predpx[, -c(6,7)]


sexo<- as.factor(ela$PatientSex)
edad<- ela$age
```

```
################ ALSFR

EN(y=ela$`ALSFRS-R`,x=predpx)
penalize(ela$`ALSFRS-R`, predpx, 1)

mod.px1<- lm(ela$`ALSFRS-R`~., data = cbind(predpx[,c(3,5,9)], sexo,
edad))
summary(mod.px1)

# CV

CV_px<- data.frame(ALSFRS=ela$`ALSFRS-R`, cbind(predpx[,c(3,5,9)], sexo,
edad))
CV_px<- CV_px[complete.cases(CV_px), ]

set.seed(1)
train(ALSFRS~.,data = CV_px,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')


CV_pxcat<- data.frame(ALSFRS=ela$`ALSFRS-R`, Genotipo.bin,
as.factor(ela$`Estadio diagnÓstico`),
sexo, edad); colnames(CV_pxcat)[3]<- 'EstadioDx'
CV_pxcat<- CV_pxcat[rownames(CV_px),]

set.seed(1)
train(ALSFRS~.,data = CV_pxcat,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')

################ TP

EN(y=ela$`Tasa progresiÓn`,x=predpx)
penalize(ela$`Tasa progresiÓn`, predpx, 1)

mod.px2<- lm(ela$`Tasa progresiÓn`~., data = cbind(predpx[,c(3,5,8)],
sexo, edad))
summary(mod.px2)

# CV

CV_px2<- data.frame(TP=ela$`Tasa progresiÓn`, cbind(predpx[,c(3,5,8)],
sexo, edad))
CV_px2<- CV_px2[complete.cases(CV_px2), ]

set.seed(1)
train(TP~.,data = CV_px2,
```

```
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')

#efectos mixtos, CV

CV_pxcat2<- data.frame(ALSFRS=ela$`Tasa progresiÓn`, Genotipo.bin,
as.factor(ela$`Estadio diagnÓstico`),
sexo, edad); colnames(CV_pxcat)[3]<- 'EstadioDx'
CV_pxcat2<- CV_pxcat2[rownames(CV_px2),]

set.seed(1)
train(ALSFRS~.,data = CV_pxcat2,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')

library(knitr)
library(kableExtra)


bestmod_px1 <- as.data.frame(round(summary(mod.px1)$coefficients[,-3],3))
colnames(bestmod_px1) <- c('Coeficiente', 'Error estÁndar', 'p-valor')
rownames(bestmod_px1) <- c('(Intercepto)', rownames(bestmod_px1)[2:6])

# Generar la tabla con kable y aplicar estilos y formato
tabla_bestmod_px1 <- kable(bestmod_px1) %>%
  kable_styling("striped", full_width = FALSE) %>%
  row_spec(3, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(4, bold = TRUE, extra_css = "font-weight: bold;")

tabla_bestmod_px1


####


bestmod_px2 <- as.data.frame(round(summary(mod.px2)$coefficients[,-3],3))
colnames(bestmod_px2) <- c('Coeficiente', 'Error estÁndar', 'p-valor')
rownames(bestmod_px2) <- c('(Intercepto)', rownames(bestmod_px2)[2:6])

# Generar la tabla con kable y aplicar estilos y formato
tabla_bestmod_px2 <- kable(bestmod_px2) %>%
  kable_styling("striped", full_width = FALSE) %>%
  row_spec(3, bold = TRUE, extra_css = "font-weight: bold;")

tabla_bestmod_px2

#######
```

```r
bestmod_px1.all <- as.data.frame(round(summary(modEN_px1)$coefficients[,-3],3))
colnames(bestmod_px1.all) <- c('Coeficiente', 'Error estÁndar', 'p-valor')
rownames(bestmod_px1.all)[1] <- c('(Intercepto)')

tabla_bestmod_px1.all <- kable(bestmod_px1.all) %>%
  kable_styling("striped", full_width = FALSE) %>%
row_spec(3, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(5, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(9, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(11, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(13, bold = TRUE, extra_css = "font-weight: bold;")

tabla_bestmod_px1.all
#######



bestmod_px2.all <- as.data.frame(round(summary(modEN_px2)$coefficients[,-3],3))
colnames(bestmod_px2.all) <- c('Coeficiente', 'Error estÁndar', 'p-valor')
rownames(bestmod_px2.all)[1] <- c('(Intercepto)')

# Generar la tabla con kable y aplicar estilos y formato
tabla_bestmod_px2.all <- kable(bestmod_px2.all) %>%
  kable_styling("striped", full_width = FALSE) %>%
  row_spec(3, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(6, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(9, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(10, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(12, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(13, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(14, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(15, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(16, bold = TRUE, extra_css = "font-weight: bold;") %>%
     row_spec(17, bold = TRUE, extra_css = "font-weight: bold;") %>%
  row_spec(18, bold = TRUE, extra_css = "font-weight: bold;")

tabla_bestmod_px2.all

# EN por bloque
th.left_EN<- EN(y=ela$`ALSFRS-R`, x=thickness_left)
th.right_EN<- EN(y=ela$`ALSFRS-R`, x=thickness_right)
volum_EN<- EN(y=ela$`ALSFRS-R`, x=volum.porc)
iron_EN<- EN(y=ela$`ALSFRS-R`, x=iron)

# Union de cada bloque
```

```r
marcadores.finales<- function(en, indice){
  indice[which(en !=0)]
}

th.left_MF<- marcadores.finales(th.left_EN$mod.final[[1]][-1],
                                index.global_th.left)
th.rigth_MF<- marcadores.finales(th.right_EN$mod.final[[1]][-1],
                                 index.global_th.right)
volum_MF<- marcadores.finales(volum_EN$mod.final[[1]][-1],
                              index.global_volum.porc)
iron_MF<- marcadores.finales(iron_EN$mod.final[[1]][-1],
index.global_iron)

MF_index<- sort(c(volum_MF, th.left_MF, th.rigth_MF, iron_MF))


# EN sobre la union
union_EN<- EN(y=ela$`ALSFRS-R`, x=ela[ ,MF_index])



# Step
MFunion_index<- which(names(ela) %in%
        rownames(union_EN$mod.final[[1]])[-1][union_EN$mod.final[[1]][-
1]!=0])

fullmodel<-glm(ela$`ALSFRS-R`~.,
                   data = ela[,MFunion_index])

nullmodel<- glm(ela$`ALSFRS-R`~1,
                    data = ela[,MFunion_index])

stepforward<-stepAIC(nullmodel,
                     direction = 'forward',
                     scope = list(upper = fullmodel,
                                  lower = nullmodel),
                     trace = 0)

stepbackward<-stepAIC(fullmodel,
                     direction = 'backward',
                     scope = list(upper = fullmodel,
                                  lower = nullmodel),
                     trace = 0)

summary(stepbackward)

pred.modEN_px1<-
gsub('`','',rownames(summary(stepbackward)$coefficients)[-1] )
dfmodEN_px1<- cbind(allmarkers[,pred.modEN_px1],sexo, edad) #aÑade sexo y
edad
```

```r
modEN_px1<- lm(ela$`ALSFRS-R`~., data = dfmodEN_px1) #modelo ajustado

## CV
dfCVEN_px1<- data.frame(ALSFRS=ela$`ALSFRS-R`,
                        dfmodEN_px1)

dfCVEN_px1<- dfCVEN_px1[complete.cases(dfCVEN_px1),]



set.seed(1)
train(ALSFRS~.,data = dfCVEN_px1,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')

# EN por bloque
th.left_EN<- EN(y=ela$`Tasa progresiÓn`, x=thickness_left)
th.right_EN<- EN(y=ela$`Tasa progresiÓn`, x=thickness_right)
volum_EN<- EN(y=ela$`Tasa progresiÓn`, x=volum.porc)
iron_EN<- EN(y=ela$`Tasa progresiÓn`, x=iron)

# Union de cada bloque
marcadores.finales<- function(en, indice){
  indice[which(en !=0)]
}

th.left_MF<- marcadores.finales(th.left_EN$mod.final[[1]][-1],
                                index.global_th.left)
th.rigth_MF<- marcadores.finales(th.right_EN$mod.final[[1]][-1],
                                 index.global_th.right)
volum_MF<- marcadores.finales(volum_EN$mod.final[[1]][-1],
                              index.global_volum.porc)
iron_MF<- marcadores.finales(iron_EN$mod.final[[1]][-1],
index.global_iron)

MF_index<- sort(c(volum_MF, th.left_MF, th.rigth_MF, iron_MF))


# EN sobre la union
union_EN<- EN(y=ela$`Tasa progresiÓn`, x=ela[ ,MF_index])



# Step
MFunion_index<- which(names(ela) %in%
        rownames(union_EN$mod.final[[1]])[-1][union_EN$mod.final[[1]][-
1]!=0])
```

```r
fullmodel<-glm(ela$`Tasa progresiÓn`~.,
                   data = ela[,MFunion_index])

nullmodel<- glm(ela$`Tasa progresiÓn`~1,
                    data = ela[,MFunion_index])

stepforward<-stepAIC(nullmodel,
                     direction = 'forward',
                     scope = list(upper = fullmodel,
                                  lower = nullmodel),
                     trace = 0)

stepbackward<-stepAIC(fullmodel,
                     direction = 'backward',
                     scope = list(upper = fullmodel,
                                  lower = nullmodel),
                     trace = 0)



pred.modEN_px2<-
gsub(``,'',rownames(summary(stepbackward)$coefficients)[-1] )
dfmodEN_px2<- cbind(allmarkers[,pred.modEN_px2],sexo, edad) #aÑade sexo y
edad

modEN_px2<- lm(ela$`Tasa progresiÓn`~., data = dfmodEN_px2) #modelo
ajustado




## CV
dfCVEN_px2<- data.frame(TP=ela$`Tasa progresiÓn`,
                        dfmodEN_px2)

dfCVEN_px2<- dfCVEN_px2[complete.cases(dfCVEN_px2),]



set.seed(1)
train(TP~.,data = dfCVEN_px2,
      trControl = trainControl(method = "repeatedcv", number = 5, repeats
= 3)
      ,method='glm', family='gaussian')
```

8. DISCUSIÓN

```
lda_cv<-function(N){

ED_1<- PCA.best_ED$PC[ED=='1',]
set.seed(1)
newED_1<- data.frame(ED= rep(1, N), mvrnorm(n=N, mu=apply(ED_1,2,mean),
Sigma =cov(ED_1)))


ED_2<- PCA.best_ED$PC[ED=='2',]
set.seed(1)
newED_2<- data.frame(ED= rep(2, N), mvrnorm(n=N, mu=apply(ED_2,2,mean),
Sigma =cov(ED_2)))

ED_3<- PCA.best_ED$PC[ED=='3',]
set.seed(1)
newED_3<- data.frame(ED= rep(3, N), mvrnorm(n=N, mu=apply(ED_3,2,mean),
Sigma =cov(ED_3)))



newdata_ED<- rbind(cbind(ED,PCA.best_ED$PC),
      rbind(newED_1,newED_2,newED_3))


newcv3(newdata_ED$ED, nfolds= 3, DATA = newdata_ED[,-1], AD='lda')

}


qda_cv<-function(N){

ED_1<- PCA.best_ED$PC[ED=='1',]
set.seed(1)
newED_1<- data.frame(ED= rep(1, N), mvrnorm(n=N, mu=apply(ED_1,2,mean),
Sigma =cov(ED_1)))


ED_2<- PCA.best_ED$PC[ED=='2',]
set.seed(1)
newED_2<- data.frame(ED= rep(2, N), mvrnorm(n=N, mu=apply(ED_2,2,mean),
Sigma =cov(ED_2)))

ED_3<- PCA.best_ED$PC[ED=='3',]
set.seed(1)
newED_3<- data.frame(ED= rep(3, N), mvrnorm(n=N, mu=apply(ED_3,2,mean),
```

```r
Sigma =cov(ED_3)))



newdata_ED<- rbind(cbind(ED,PCA.best_ED$PC),
      rbind(newED_1,newED_2,newED_3))


newcv3(newdata_ED$ED, nfolds= 3, DATA = newdata_ED[,-1], AD='qda')

}



lda_result<-t(sapply(2:500, function(n) lda_cv(N=n)))
qda_result<-t(sapply(2:500, function(n) qda_cv(N=n)))

lda_df <- data.frame(lda_result)

# Añadir columna para el número de fila multiplicado por 5
lda_df$X <- seq_len(nrow(lda_df)) * 3

# Reorganizar datos en formato largo
lda_df_long <- tidyr::gather(lda_df, variable, value, -X)

# Graficar líneas utilizando ggplot2
ggplot(lda_df_long, aes(x = X, y = value, color = variable)) +
  geom_line() +
  labs(title = "LDA") +
  guides(color = guide_legend(title = NULL)) +
  labs(x = "Tamaño muestral adicional", y = " ")+
  scale_color_manual(values = c("#F8766D", "#00BFC4"), labels =
c("Precisión", "AUC")) +
  theme(plot.title = element_text(face = "bold", hjust = 0.5)) +
  ylim(0, 1)



qda_df <- data.frame(qda_result)

# Añadir columna para el número de fila multiplicado por 5
qda_df$X <- seq_len(nrow(qda_df)) * 3

# Reorganizar datos en formato largo
qda_df_long <- tidyr::gather(qda_df, variable, value, -X)

# Graficar líneas utilizando ggplot2
ggplot(qda_df_long, aes(x = X, y = value, color = variable)) +
  geom_line() +
  labs(title = "QDA") +
```

```r
  guides(color = guide_legend(title = NULL)) +
  labs(x = "Tamaño muestral adicional", y = " ")+
  scale_color_manual(values = c("#F8766D", "#00BFC4"), labels =
c("Precisión", "AUC")) +
  theme(plot.title = element_text(face = "bold", hjust = 0.5))

newcv3<- function(factor, nfolds, DATA, AD='lda'){

y<- as.factor(factor )

sepFolds<-lapply(1:length(levels(y)), function(x)
  {createFolds(y[y==levels(y)[x]], k=nfolds, returnTrain = FALSE)} )

Folds<- list()
for (k in 1:nfolds) {
  Folds[[k]]<- c(which(y==levels(y)[1])[unlist(sepFolds[[1]][k])],
  which(y==levels(y)[2])[unlist(sepFolds[[2]][k])],
  which(y==levels(y)[3])[unlist(sepFolds[[3]][k])])
}

Accuracy<- c()
Sensitivity<- matrix(ncol=3, nrow =0 )
Specificity<- matrix(ncol=3, nrow =0 )
AUC<- c()

X<- data.frame(DATA)

for (i in 1:nfolds) {

  test_indices <- Folds[[i]]

  X_train <- data.frame(X[-test_indices, ])
  colnames(X_train)<- colnames(DATA)
  y_train <- y[-test_indices]

  X_test <- data.frame(X[test_indices, ])
  colnames(X_test)<- colnames(DATA)
  y_test <- y[test_indices]

  # Ajustar el modelo
  if (AD=='lda'){
      mod<- lda(y_train ~., data= X_train)
  }

  if (AD=='qda'){
      mod<- qda(y_train ~., data= X_train)
  }

  # Hacer predicciones en test usando el modelo ajustado
  y_pred <- predict(mod, newdata = X_test)
```

```r
  # Calcular errores
  cm <- confusionMatrix(y_pred$class, y_test)
  accuracy<- sum(diag(cm$table))/sum(cm$table)

  sensitivity <- cm$byClass[, "Sensitivity"]
  specificity <- cm$byClass[, "Specificity"]

  auc<- multiclass.roc(y_test, y_pred$posterior)
  auc_value <- sub(".*: ", "", auc$auc)
  auc<- as.numeric(auc_value)

  Accuracy<- c(Accuracy, accuracy)
  Sensitivity<- rbind(Sensitivity, sensitivity)
  Specificity<- rbind(Specificity, specificity)
  AUC<- c(AUC, auc)

}

# Media de cada error
Accuracy_CV<- mean(Accuracy, na.rm=T)
Sensitivity_CV<- sapply(1:3, function(x) mean(Sensitivity[,x], na.rm=T))
Specificity_CV<- sapply(1:3, function(x) mean(Specificity[,x], na.rm=T))
AUC_CV<- mean(AUC, na.rm=T)

Error_average<- c(Accuracy=Accuracy_CV,
                  AUC=AUC_CV)

return(Error_average)
}
```