M2.851 - Tipología y ciclo de vida de los datos. PRÁCTICA2

Autor: Antonio Fernández Serra

Diciembre 2021

Contents

1.- Descripción del dataset

El cáncer de mama (BC por sus siglas en inglés), es el cáncer femenino de mayor incidencia, en el año 2020, se diagnosticaron aproximadamente 2.3 millones de casos y 685000 pacientes murieron por la enfermedad. Pero gracias, entre otros muchos factores, al intensivo esfuerzo en investigación, también se trata de una enfermedad muy prevalente, consiguiéndose unas tasas de curación y cronificación, inéditas en el campo de la oncología. En este sentido, 7.5 millones de mujeres vivas han sido diagnosticadas de BC en los últimos 5 años. Estos tumores también se dan en hombres aunque en mucha menor medida (0.5-1% de los casos), esto se debe a que tanto el BC como los tumores prostáticos tienen un importante componente hormonal, y comparando las mamas entre sexos, es la mama femenina la que lleva a cabo en mayor medida esta acción hormonal [1,2].

BC es una enfermedad sumamente heterogénea con un comportamiento clínico, biológico, de supervivencia e histológicamente muy distinto. Constituyendo este tipo de tumor, actualmente, un paradigma de la medicina personalizada, por lo que la investigación de cualquier tumor, pero especialmente de este, requiere de una cuidadosa delimitación previa del subtipo y las características a estudiar Este tipo de tumores ha sido objeto de una intensa investigación en los últimos años poniéndose a la vanguardia de los tipos de cáncer más conocidos y a la cola en cuanto a morbilidad y mortalidad [3]. Los marcadores inmunohistoquímicos clásicos como ER (receptor de estrógenos), PR (receptor de prostágenos) y HER2 junto con las variables

clinicopatológicas clásicas como: el tamaño, grado tumoral y afectación tumoral se han usado para establecer el pronóstico y el manejo clínico de las pacientes. Un paso más allá se llegó con los trabajos de Sorlie y cols en los que se delimitan los 5 subtipos intrínsecos (Luminal A y B, Her-2, Basal y normal-like) a partir de la expresión de 456 genes extraídos de experimentos realizados con microarrays [4]. Estos subtipos tienen distinto pronóstico, por lo que su establecimiento es fundamental en el manejo terapéutico del BC.

El tratamiento de elección en BC es la cirugía, con la intención de eliminar totalmente el tumor, cuando esto se consigue, en ausencia de otras condicionantes genéticas, se equipara el riesgo de cáncer con el resto de población. No obstante, no siempre nos movemos en este escenario clínico ideal, tumores avanzados al diagnóstico pueden necesitar de una terapia sistémica de inicio llamada neoadyuvante, y cirugías subóptimas ya sea por las características iniciales del tumor primario, o bien por un infraestadiaje del tumor previo a la cirugía (radiológicamente o en biopsia), pueden requerir de una terapia adyuvante después de la cirugía. Dentro de estos escenarios clínicos extremos, se extienden una serie de grises que hace necesario adoptar distintos enfoques terapéuticos derivados de estos [5]. En este entorno tan complejo, es evidente que cualquier guía para moverse entre estas opciones es muy necesaria, constituyendo uno de los campos más activos en investigación biomédica oncológica. Para este fin, multitud de factores clínicos, biológicos en un sentido amplio (genéticos, genómicos, transcriptómicos, proteómicos, etc.) son intensivamente investigados en una plétora de proyectos de investigación y ensayos traslacionales asociados a ensayos clínicos. Una de las estrategias más usadas, es relacionar estos factores a estudio con la supervivencia cáncer-específica, o bien el intervalo libre de enfermedad o recaída, o bien relacionarlos con un diagnóstico más temprano de la enfermedad, obteniendo biomarcadores con potencial diagnóstico o pronóstico.

Dentro de estos esfuerzos a nivel de investigación, existen iniciativas a gran escala, a fin de obtener datos multiómicos (datos provenientes de técnicas de alto rendimiento para medir grandes cantidades de información a nivel de mutaciones, expresión de genes, proteínas, metabolitos, etc.), con una rica anotación clínica. La investigación en biología molecular tiene una rica tradición en iniciativas coordinadas multilaboratorios para abordar problemas o retos que serían difícilmente abordables de otra forma. Así, el proyecto genoma humano dio el pistoletazo de salida con su objetivo de secuenciar completamente el genoma de nuestra especie, con el resultado de la publicación del borrador en el año 2000, y la secuencia definitiva en 2003 [6]. Otro hito en este sentido es el proyecto ENCODE, centrado en descifrar elementos reguladores del genoma, las regiones anteriormente (mal) conocidas como ADN basura. Este conocimiento supuso otro hito en la investigación en biología molecular [7]. El más reciente de estos proyectos es el llamado The Cancer Genome Atlas (TCGA) que busca identificar globalmente los cambios en el ADN ya sea a nivel transcriptómico, de mutaciones, o de grandes alteraciones cromosómicas, CNV por sus siglas en inglés (Copy Number Variation). En el proyecto se analizan más de 20000 muestras incluyendo tumores primarios y tejido sano emparejado de 33 tipos tumorales distintos. Siendo su objetivo el descubrimiento más profundo del origen y distintos aspectos del cáncer que en definitiva puede llevarnos al conocimiento de nuevas maneras de diagnosticar y tratar el cáncer. El Instituto Nacional del Cáncer (NCI) y el Instituto Nacional para la Investigación del Genoma Humano (NHGRI) de los E.E.U.U. que forman parte de los Institutos Nacionales de Salud (NIH) dirigen el proyecto, en el que participan multitud de laboratorios a lo largo de todo el globo [8].

Con este complicado transfondo en mente, se marca como objetivo del siguiente trabajo la obtención, integración, limpieza y pretratamiento de datos clínicos, patológica, de expresión de proteínas y transcriptómicos de 2 fuentes, a fin de poder aplicar algoritmos de Machine Learning para establecer el pronóstico en esta enfermedad.

La fuente de los datos usados en la presente práctica se obtuvieron de esta fuente primaria pero descargados de Kaggle [1], donde se encuentra el dataset depurado con datos de BC, por otro lado, para la integración se descargaron los datos transcriptómicos del portal http://firebrowse.org/, específico para estos datos.

2.- Integración y selección de los datos

En este apartado, partiendo de las 2 fuentes de datos. El primer archivo proveniente de kaggle que contiene las principales características clínicas y patológicas, junto a la expresión de 4 proteínas provenientes de 334 casos (BRCA.csv). El segundo archivo, provenientes de firebrowse (BRCA.mRNAseq_raw_counts.txt), contiene información en crudo, obtenida con la técnica RNAseq de la expresión de 20532 genes de 879 casos de BC. Uno de los primeros pasos del análisis, será obtener los casos comunes entre ambos datasets, sobre los que integraremos la información.

```
# librerías
library(gdata)
library(tidyr)
library(Boruta)
library(caret)
library(lubridate)
library(car)
library(missForest)
library(VIM)
library(purrr)
library(tibble)
library(PerformanceAnalytics)
library(knitr)
library(dplyr)
```

```
# Directorio de trabajo

setwd("/home/flatline/Documentos/Master_Data_Science/Ciclo_de_Vida_Dato/Practica_2/gdac.broadinstitute.



# Cargamos las dos fuentes de datos

## Datos transcriptómicos obtenidos de http://firebrowse.org/
tcga_data <- read.table("BRCA.mRNAseq_raw_counts.txt", header = T, sep = "\t",
                        check.names=FALSE)

rownames(tcga_data) <- tcga_data[,1]
tcga_data <- tcga_data[,-1]

tcga_data <- t(tcga_data)


setwd("/home/flatline/Documentos/Master_Data_Science/Ciclo_de_Vida_Dato/Practica_2")

## Datos clinicopatológicos y expresión de 4 proteínas obtenidas de kaggle

clin_data <- read.csv("BRCA.csv",header=T, fill=T)
rownames(clin_data) <- clin_data[,1]
clin_data <- clin_data[,-1]

## Como tenemos más casos en el dataset de expresión génica (tcga_data), aunque no
## son todos comunes, los usamos para la integración

selection <- rownames(clin_data)
```

2.1.- Integración y selección de los datos de interés a analizar.

Nos quedamos con 239 muestras comunes entre el dataset clínico y de proteínas y el dataset transcriptómico

---

```
tcga <- tcga_data[rownames(tcga_data) %in% selection,]


## Creación de la variable respuesta categórica dicotómica
## Cálculo de la supervivencia en días con la función dmy
## de la librería lubridate

clin_data$OS <- as.numeric(dmy(clin_data$Date_of_Last_Visit) -
  dmy(clin_data$Date_of_Surgery),units="days")


# Codifico la Supervivencia Global en días según sea mayor o igual o menor a la mediana
# Esta será la variable respuesta con etiquetas "High OS" y "Low_OS"

clin_data$c_OS <- ifelse(clin_data$OS < median(clin_data$OS, na.rm=T), "Low OS",
                         "High OS")

# Contamos el número de NAs

prop.table(table(is.na(clin_data$OS)))
```

```
##
##     FALSE      TRUE
## 0.9491018 0.0508982
```

```
# Como se trata de un pequeño porcentaje (5.08%) opto por eliminar
## las instancias sin fecha de última revisión del dataset clin_data
## antes de la integración, luego de la integración volverá a limpiarse
## el dataset

clin_data <- clin_data %>% drop_na()

## Quedan 317 muestras


# Integración de los datos de expresión génica con los datos clínicos

data <- merge(clin_data, tcga_data, by=0, all=TRUE)
data <- data %>% drop_na()

## Quedan 227 muestras comunes a ambos datasets

# Selección de genes más informativos

## Creamos dataset temp_data con la expresión de los 20532 genes
## Como es una cantidad muy alta, habrá que hacer una limpieza muy
## drástica


temp_data <- data[,18:length(data)]
```

```
rownames(temp_data) <- data$Row.names
```

El dataset de expresión génica tiene una dimensionalidad demasiado alta, lo que dificulta el ajuste de modelos de minería de datos, y por otro lado, hará que los cálculos sean computacionalmente demasiado altos, y además el modelo sea difícilmente generalizable.

```
### LIMPIEZA DEL DATASET TRANSCRIPTÓMICO ##########################

# En primer lugar eliminamos todos los genes que tengan en total menos de 227 lecturas,
# lo que indicará que la información de expresión de ese gen es poco informativa. Este
# cut-off corresponde a una lectura promedio por caso

temp_clean <- temp_data[2:length(temp_data)][,-which(colSums(temp_data[sapply(temp_data, is.numeric)])
                                                     < 227)]

# Eliminamos 1365 genes que tienen menos de 250 lecturas

#### Feature selection near to zero variance (by rows)

# nearZeroVar() del paquete caret es una función que identifica aquellas variables
# con una varianza cercana a 0 y que por tanto tampoco tienen poder discriminante
# entre los 2 grupos de la variable respuesta

nzv_metrics <- nearZeroVar(temp_clean,freqCut = 95/5, uniqueCut = 10, saveMetrics = TRUE,
                           names = FALSE, foreach = FALSE, allowParallel = TRUE)

f <- names(temp_clean)[nearZeroVar(temp_clean)]

temp_clean <- temp_clean[,!names(temp_clean) %in% f]
rownames(temp_clean) <- data$Row.names


## 22 genes con Varianza cercana a 0 eliminados
```

El siguiente paso es eliminar los atributos con alta correlación, porque no aportan información al modelo, crean ruido, aumentan el coste computacional y empeoran considerablemente la interpretabilidad y generabilidad del modelo.

```
# En el primer filtro, vamos a eliminar todas las variables con un coeficiente de correlación
# mayor de 0.85

cor_temp_clean <- cor(temp_clean)
highlyCorrelated <- findCorrelation(cor_temp_clean, cutoff=0.85)

# Tenemos 3724 atributos con R > 0.85 que vamos a limpiar del dataset

temp_clean <- temp_clean[, -highlyCorrelated]

# El siguiente criterio es eliminar todos los atributos que tengan una baja desviación std
## Eliminación de la desviación estandár hasta el primer cuartil (227.4476)

sd_vector <- apply(temp_clean, 2, sd)
```

```
length(sd_vector)
```

```
## [1] 15463
```

```
## Son 3866 variables a eliminar

rm_sd <- sd_vector > quantile(sd_vector)[2]
tmp_clean <- temp_clean[,rm_sd]  # 3866 genes eliminados
temp_clean <- tmp_clean

c_OS <- temp_data$c_OS
temp_clean$c_Os <- c_OS
temp_clean$c_Os <-as.factor(temp_clean$c_Os)
```

A continuación, llevaremos a cabo una selección de atributos más profunda usando el algoritmo BORUTA, este algoritmo basado en Random Forest, fue desarrollado en 2010 y es considerado como muy poderoso y eficiente, consiste en enfrentar las variables a permutaciones al azar de esas mismas variables, de modo que sólo se retienen aquellas que aportan más información que un cierto umbral establecido por las variables permutadas (o shadow features), luego la probabilidad de estas variables de modela con una distribución binomial negativa que les asignará tres etiquetas: atributos que deben conservarse, otros que deben eliminarse y dudosas, repitiendo el proceso iterativamente, aunque aumenta el tiempo de procesado computacional, se irán asignando aributos dudosos a uno de los 2 grupos [9].

```
# Selección de atributos informativos con el algoritmo BORUTA
# basado en Random Forest

set.seed(111)
boruta.train <- Boruta(c_Os~., data = temp_clean, doTrace = 2)
print(boruta.train)
```

```
## Boruta performed 99 iterations in 5.116743 mins.
##  7 attributes confirmed important: `ALPL|249`, `CEBPD|1052`,
## `CMTM7|112616`, `IGDCC3|9543`, `PHOSPHO2|493911` and 2 more;
##  11565 attributes confirmed unimportant: `?|10431`, `?|155060`,
## `?|57714`, `?|8225`, `?|90288` and 11560 more;
##  25 tentative attributes left: `C20orf160|140706`, `CDC42EP1|11135`,
## `CNNM1|26507`, `COQ3|51805`, `ELF3|1999` and 20 more;
```

```
## Como quedan 25 atributos en los que no está clara
## Su mayor importancia que su shadow feature, repito
## el proceso con más iteraciones

set.seed(123)
boruta.train2 <- Boruta(c_Os~., data = temp_clean,
                        doTrace = 1, maxRuns=6000)

print(boruta.train2)
```

```
## Boruta performed 5999 iterations in 13.26194 mins.
##  30 attributes confirmed important: `AFP|174`, `ALPL|249`,
## `BEND3|57673`, `C1orf25|81627`, `C20orf160|140706` and 25 more;
##  11566 attributes confirmed unimportant: `?|10431`, `?|155060`,
## `?|57714`, `?|8225`, `?|90288` and 11561 more;
##  1 tentative attributes left: `KRT15|3866`;
```

```
select_boruta <- getSelectedAttributes(boruta.train2, withTentative = F)
select_boruta <- gsub("`","",select_boruta)




set.seed(123)
boruta.train3 <- Boruta(c_Os~., data = temp_clean[,c("c_Os",select_boruta)],
                        doTrace = 1, maxRuns=2000)

print(boruta.train3)
```

```
## Boruta performed 946 iterations in 1.180503 mins.
##  30 attributes confirmed important: `AFP|174`, `ALPL|249`,
## `BEND3|57673`, `C1orf25|81627`, `C20orf160|140706` and 25 more;
##  No attributes deemed unimportant.
```

```
## Con 6000 iteraciones retenemos 30 variables, como es un número manejable, seguimos
## con ellas el análisis.
```

Ahora creamos el dataframe defiinitivo de trabajo, que incluirá las variables clínicas,la expresión de las 4 proteínas, y la expresión de los 30 genes que más poder discriminante respecto a la variable c_Os (supervivencia global), este dataset ya es manejable.

```
gene_exp <- temp_clean[,select_boruta]

# Creación del dataframe integrado

def_data <- cbind(data[,1:18],gene_exp)
def_data <- def_data[,-1]

# Antes de seguir limpiamos el workspace de todos los dataframes y variables
# auxiliares que hemos ido creando

# rm(selection, f, highlyCorrelated, nzv_metrics, sd_vector,
#    cor_temp_clean, temp_data, tmp_clean, select_boruta,
#    temp_clean, gene_exp, tcga, tcga_data, boruta.train,
#    boruta.train2)


## Estandariación de los datos de expresión génica

num_var <- unlist(lapply(def_data, is.numeric))
def_data_num <- def_data[,num_var]

process <- preProcess(def_data_num,method=c("center","scale"))
def_data_norm <- predict(process, def_data_num)

def_data_temp <- cbind(def_data[,-which(names(def_data) %in% colnames(def_data_norm))], def_data_norm)
def_data <- def_data_temp

# Eliminamos los campos de fechas que hemos usado para calcular la supervivencia global (OS), porque ya
# son informativos.
def_data <- def_data[,-c(8,9)]
```

```
# rm(def_data_norm, def_data_num, def_data_temp)
```

3.- Limpieza de los datos

3.1 Comprobación de 0 y NAs

```
sum(is.na(def_data)) / length(def_data)
```

```
## [1] 0
```

```
sum(def_data==0) / length(def_data)
```

```
## [1] 0
```

No tenemos valores NA, porque ya hemos limpiado el 5.04% que teníamos antes de la integración de los 2 datasets usados en esta práctica. Tampoco tenemos ningún valor 0, aunque es de resaltar que en este caso no trataríamos los 0, porque en expresión génica y proteíca, un valor de 0 es informativo indicando que las células que componen esa muestra tumoral, no expresan ese gen o proteína. Vamos a introducir otro 2.1% de Nas con la función prodNA.

Vamos a seguir 2 alternativas en paralelo, por un lado crearemos un dataframe eliminando esos casos (def_data1) en los que se han introudcido NAs (en total un 4.75%), y crearemos otro dataset, asignando los valores perdidos con un algoritmo basado en K-Nearest Neighbors, implementado en la librería VIM (def_data2), para luego comprobar cual de los 2 enfoques nos permite construir un clasificador más exacto.

```
def_data_temp <- prodNA(def_data, noNA=0.005)
sum(is.na(def_data_temp)) / length(def_data_temp)
```

```
## [1] 1.133333
```

```
# # Tenemos un 4.75% de datos perdidos
# A continuación crearemos 2 dataframes, uno con los NA
# eliminados (def_data1) y el 2 con los valores imputados
# con un algoritmo k-nearest neighbors usando la librería VIM y la función KNN

def_data1 <- def_data_temp %>% drop_na()
sum(is.na(def_data1)) / length(def_data1)
```

```
## [1] 0
```

```
## Quedan 182 casos, aún reduciendo hasta un 0.5%
# el número de casos que quedan son 180, con lo
# que perdemos 47 que en
# en una variable u otra tengan NAs

df <- data.frame(apply(def_data_temp, 2, as.factor))
def_data2 <- kNN(def_data_temp, k=3)
sum(is.na(def_data2)) / length(def_data2)
```

```
## [1] 0
```

```
def_data2 <- def_data2[,1:45]

# colnames(def_data1) == colnames(def_data2)
```

3.2 Identificación y tratamiento de valores extremos

```
# Identificación outliers


outliers <- function(dataframe){
  dataframe %>%
    dplyr::select_if(is.numeric) %>%
    map(~ boxplot.stats(.x)$out)

  }

# outliers(def_data_temp)

## DataFrame con los outliers eliminados (def_data3)


outlierreplacement <- function(dataframe){
  dataframe %>%
    map_if(is.numeric, ~ replace(.x, .x %in% boxplot.stats(.x)$out, NA)) %>%
    dplyr::bind_cols()
}


def_data3 <- outlierreplacement(def_data) %>% drop_na()

# Si eliminamos todos los outliers, nos quedamos con sólo 53 casos, no es viable
# luego no seguimos con el dataset
```

4.- Análisis de los datos

4.1.- Selección de los grupos de datos que se quieren analizar/comparar

Una vez que tenemos el dataset limpio, el objetivo es estudiar la influencia de las variables clínicas, patológicas y de expresión de DNA y proteínas en el pronóstico del Cáncer de Mama, para ello, en primer lugar, estandarizaremos y comprobaremos la homocedasticidad de los datos usando con los tests de contraste de hipótesis de Shapiro-Wilks y Levene respectivamente. Posteriormente comprobaremos su correlación (con las gráficas en el apartado 5). Al final usaremos 2 algoritmos de Machine Learning que trabajan sobre variables tanto categóricas como continúas (random forest y extreme gradient boosting) con el paquete caret. En el apartado de limpieza de datos hemos obtenido 3 datasets: el primero (def_data1) es el que tiene Nas obtenidos al azar con la función prodNAs y eliminados, con lo que quedan 180 muestras, en def_data2 se han imputado esos NAs por un algoritmo basado en k-nn y en def_data3, se han eliminado los outliers, con lo que nos queda un dataset de 53 muestras, que no es viable para el análisis, y por tanto no se seguirá esta estrategia. Los datasets 1 y 2 los dividiremos en serie de entrenamiento y validación y haremos el tuning de 3 modelos de machine learning anteriormente citados. Aunque soy consciente de que el enunciado no pide

explícitamente que se desarrollen este tipo de algoritmos, dada que la naturaleza de los datos y el problema considero adecuado este enfoque tanto para obtener conclusiones, como para llevar a cabo el benchmark de todos los procesos de integración, limpieza y procesado de los datos.

4.2.- Comprobación de la normalidad y homogeneidad de la varianza

```
# Test de Shapiro-Wilks para medir normalidad


res_shapiro <- def_data%>%
  dplyr::select_if(is.numeric)%>%
  sapply(shapiro.test)%>%
  t()%>%
  data.frame()%>%
  dplyr::select(p.value)%>%
  dplyr::mutate(Is_normally_distributed=p.value>=.05)


# Test de Levene

responses <- as.matrix(def_data[,10:length(def_data)])
res_levene <- data.frame(var = colnames(responses), p_levene = rep(NA,dim(responses)[2]))
res_levene$p_levene <- apply(responses,2,function(x) {leveneTest(x ~ as.factor(def_data$c_OS))[1,3]})
res_levene$homocedasticity <- ifelse(res_levene$p_levene < 0.05, "heterocedastico",
                                     "homocedastico")

df_shapiro_levene <- as.data.frame(cbind(res_shapiro[,c(1,2)], res_levene[,c(2,3)]))
colnames(df_shapiro_levene)[1] <- "p_shapiro"
```

4.3.- Aplicación de pruebas estadísticas para comparar los grupos de datos

```
res_kw <- def_data%>%
  dplyr::select_if(is.numeric)%>%
  sapply(wilcox.test)%>%
  t()%>%
  data.frame()%>%
  dplyr::select(p.value)%>%
  dplyr::mutate(Different_OS=p.value<.05)


res_chi <- def_data[,-c(1,4,5)]%>%
  dplyr::select(where(is.character)) %>%
  dplyr::summarise_all(dplyr::funs(chisq.test(.,def_data$c_OS)$p.value))


res_cor <- def_data[,-c(1,4,5)]%>%
  dplyr::select(where(is.numeric)) %>%
  dplyr::summarise_all(dplyr::funs(cor.test(.,def_data$OS)$p.value))
```

```r
# división en series de entrenamiento y validación

# Eliminamos el género que al dividirlo queda como 1 solo factor y nos da error
def_data1 <- def_data1[,-1]
def_data2 <- def_data2[,-1]

def_data1 <- def_data1[,-c(3,4)]
def_data2 <- def_data2[,-c(3,4)]

# Convertimos chr a factores
def_data1[sapply(def_data1, is.character)] <- lapply(def_data1[sapply(def_data1, is.character)],
                                       as.factor)


n <- nrow(def_data1)
train_rows <- sample(seq(n), size = .8 * n)
def_train1 <- def_data1[ train_rows, ]
def_test1  <- def_data1[-train_rows, ]


n <- nrow(def_data2)
train_rows <- sample(seq(n), size = .8 * n)
def_train2 <- def_data2[ train_rows, ]
def_test2  <- def_data2[-train_rows, ]

# Eliminamos columnas date of surgery y date of last visit que el algoritmo toma
# como factor de distintos niveles entre train y test
# def_train2 <- def_train2[,-c(6,7)]
# def_test2 <- def_test2[,-c(6,7)]

def_train2<-def_train2[complete.cases(def_train2),]
def_test2<-def_test2[complete.cases(def_test2),]


## Random Forest

hiperparam <- expand.grid(mtry = c(2, 5, 10),
               min.node.size = c(2, 3, 4, 5, 10),
               splitrule = "gini")


set.seed(123)
seeds <- vector(mode = "list", length = 21)
for (i in 1:20) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparam))
}
seeds[[21]] <- sample.int(1000, 1)


# Definición del entrenamiento
control_train <- trainControl(method = "cv", number = 5,
                              seeds = seeds, returnResamp = "final",
```

```
                              verboseIter = FALSE, allowParallel = TRUE)



set.seed(123)
rf_1 <- train(
  form = c_OS ~ .,
  data = def_train1,
  method = "ranger",
  tuneGrid = hiperparam,
  metric = "Accuracy",
  trControl = control_train,
  # Número de árboles ajustados
  num.trees = 500)


confusionMatrix(predict(rf_1, def_test1),
                as.factor(def_test1$c_OS))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High OS Low OS
##    High OS      21      0
##    Low OS        0     16
##
##                Accuracy : 1
##                  95% CI : (0.9051, 1)
##     No Information Rate : 0.5676
##     P-Value [Acc > NIR] : 7.919e-10
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5676
##          Detection Rate : 0.5676
##    Detection Prevalence : 0.5676
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : High OS
##
```

```
set.seed(123)
rf_2 <- train(
  form = c_OS ~ .,
  data = def_train2,
  method = "ranger",
  tuneGrid = hiperparam,
  metric = "Accuracy",
```

```
  trControl = control_train,
  # Número de árboles ajustados
  num.trees = 500)

confusionMatrix(predict(rf_2, def_test2),
                as.factor(def_test2$c_OS))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High OS Low OS
##    High OS      25      0
##    Low OS        1     20
##
##                Accuracy : 0.9783
##                  95% CI : (0.8847, 0.9994)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 1.455e-10
##
##                   Kappa : 0.956
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9615
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9524
##              Prevalence : 0.5652
##          Detection Rate : 0.5435
##    Detection Prevalence : 0.5435
##       Balanced Accuracy : 0.9808
##
##        'Positive' Class : High OS
##
```

```
## Extreme gradient boosting

hiperparam <- expand.grid(nrounds = c(100,200),
                        max_depth = c(10, 15, 20, 25),
                        colsample_bytree = seq(0.5, 0.9, length.out = 5),
                        eta = 0.1,
                        gamma=0,
                        min_child_weight = 1,
                        subsample = 1)

set.seed(123)
seeds <- vector(mode = "list", length = 6)
for (i in 1:5) {
  seeds[[i]] <- sample.int(1000, 20)
}
seeds[[6]] <- sample.int(1000, 1)

control_train <- trainControl(method = "cv", number = 5,
                              seeds = seeds, returnResamp = "final",
```

```
                              verboseIter = FALSE, allowParallel = TRUE)


set.seed(453)

xgb_1 <- capture.output(train(
  form = c_OS ~ .,
  data = def_train1,
  trControl = control_train,
  tuneGrid = hiperparam,
  method = "xgbTree"
))

load("xgb_1.rda")


confusionMatrix(predict(xgb_1, newdata=def_test1),
                as.factor(def_test1$c_OS))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High OS Low OS
##    High OS      21       0
##    Low OS        0      16
##
##                Accuracy : 1
##                  95% CI : (0.9051, 1)
##     No Information Rate : 0.5676
##     P-Value [Acc > NIR] : 7.919e-10
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5676
##          Detection Rate : 0.5676
##    Detection Prevalence : 0.5676
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : High OS
##
```

```
set.seed(453)
xgb_2 <- capture.output(train(
  form = c_OS ~ .,
  data = def_train2,
  trControl = control_train,
  tuneGrid = hiperparam,
  method = "xgbTree",
```

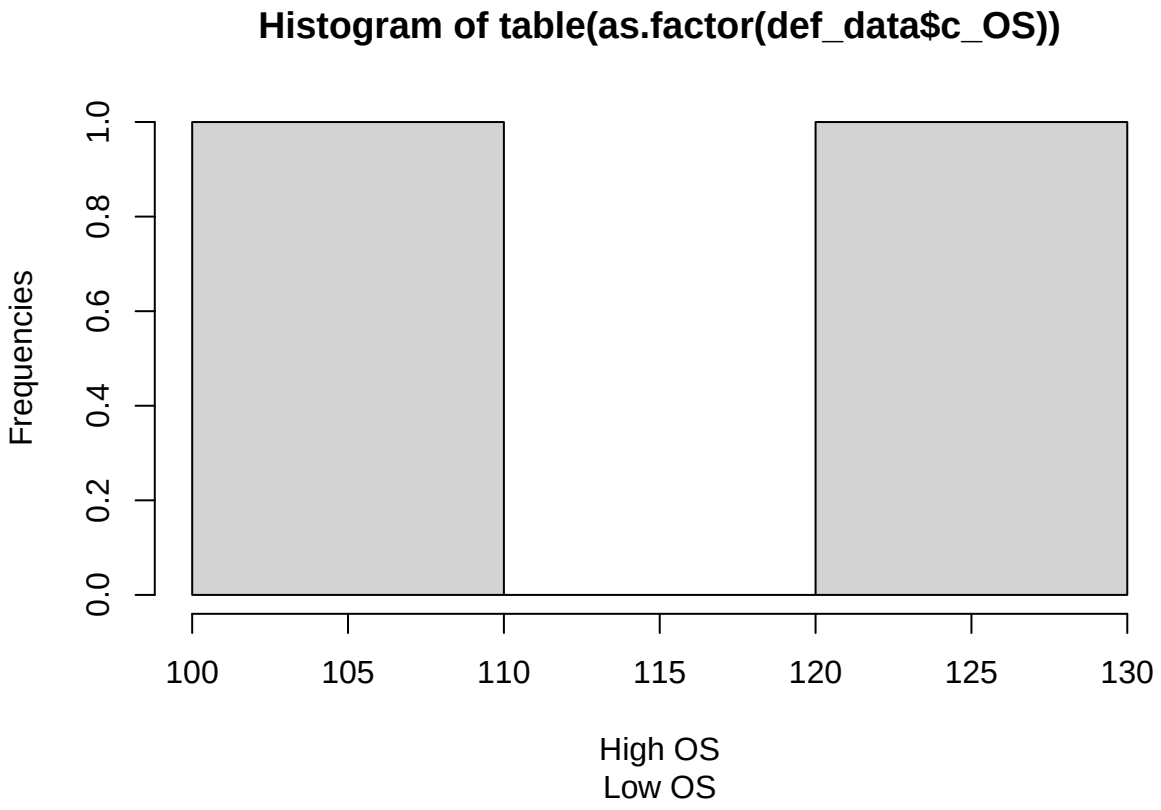
```
  verbose=F
))

load("xgb_2.rda")

confusionMatrix(predict(xgb_2, newdata=def_test2),
                as.factor(def_test2$c_OS))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High OS Low OS
##    High OS      26      0
##    Low OS        0     20
##
##                Accuracy : 1
##                  95% CI : (0.9229, 1)
##     No Information Rate : 0.5652
##     P-Value [Acc > NIR] : 3.999e-12
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5652
##          Detection Rate : 0.5652
##    Detection Prevalence : 0.5652
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : High OS
##
```

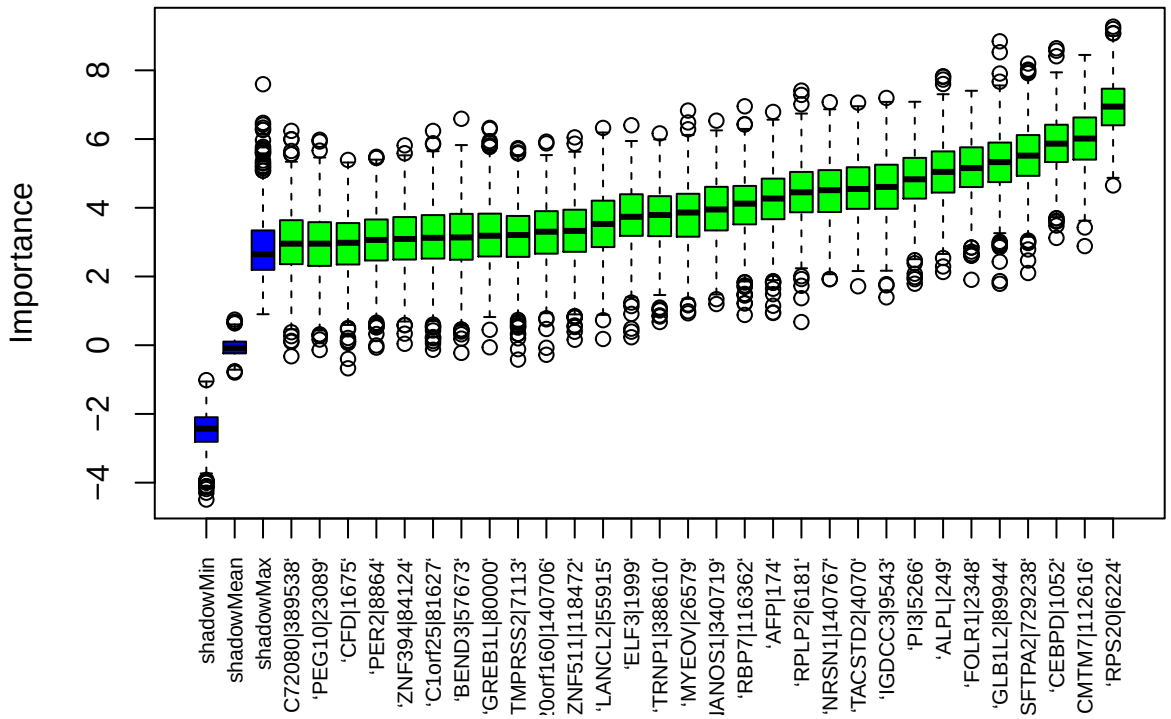
5.- Representación de los resultados a partir de tablas y gráficas

---

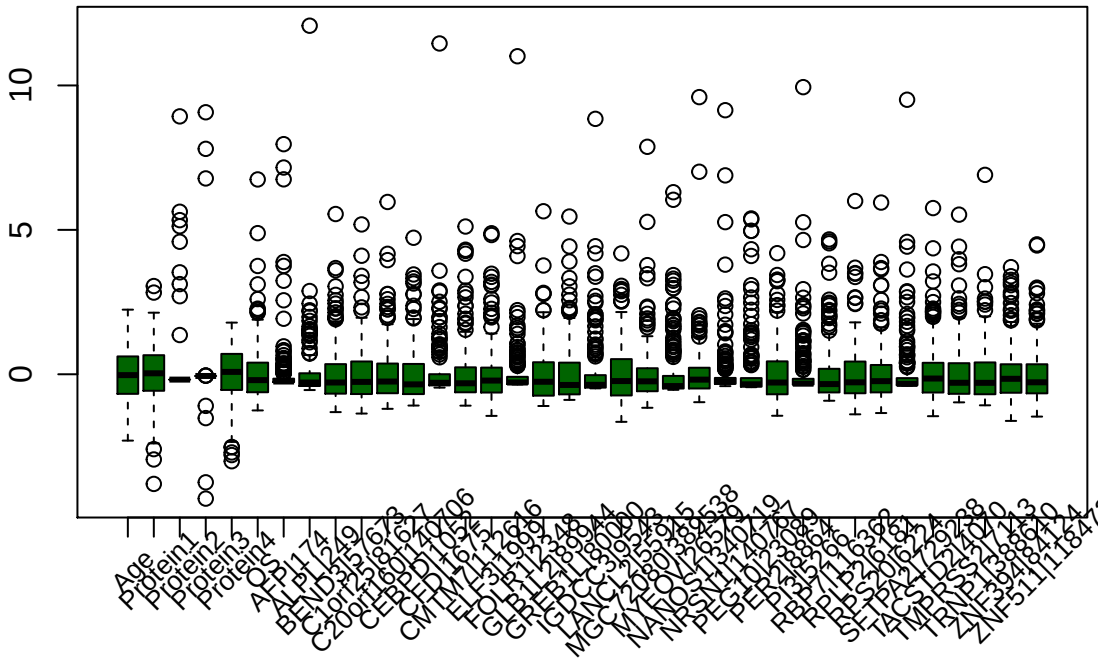
```
## Histograma de la variable respuesta

hist(table(as.factor(def_data$c_OS)), freq=T,
     xlab = levels(as.factor(def_data$c_OS)),
     ylab = "Frequencies")
```




```
## Gráfica de los 30 atributos seleccionados por el algoritmo Boruta

plot(boruta.train3, xlab = "", xaxt = "n")
lz<-lapply(1:ncol(boruta.train3$ImpHistory),function(i)
  boruta.train3$ImpHistory[is.finite(boruta.train3$ImpHistory[,i]),i])
names(lz) <- colnames(boruta.train3$ImpHistory)
Labels <- sort(sapply(lz,median))
axis(side = 1,las=2,labels = names(Labels),
     at = 1:ncol(boruta.train3$ImpHistory), cex.axis = 0.7)
```




```
## Boxplot de los datos normalizados

boxplot(def_data[,10:length(def_data)], pars  =  list(xaxt = "n"), col="darkgreen")
axis(1, at=(10:length(def_data))-9, labels = FALSE)
text((10:length(def_data))-9, par("usr")[3] - 1,
     labels = colnames(def_data[,10:length(def_data)]), srt = 45, pos = 1, xpd = TRUE, cex=0.8)
```

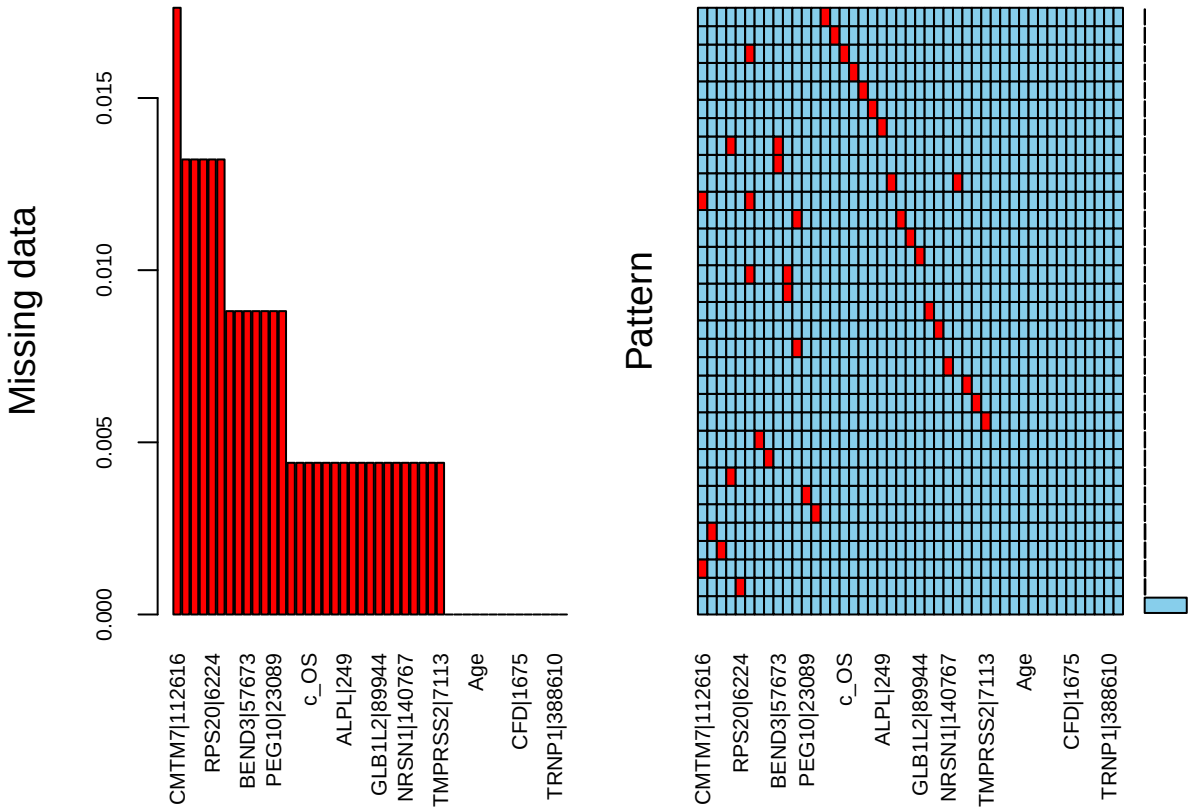



```
# Resumen de los datos limpios

summary(def_data_temp)
```

```
##     Gender           Tumour_Stage        Histology          ER.status
##  Length:227         Length:227         Length:227         Length:227
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   PR.status         HER2.status        Surgery_type       Patient_Status
##  Length:227         Length:227         Length:227         Length:227
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      c_OS                Age               Protein1            Protein2
##  Length:227         Min.   :-2.29714   Min.   :-3.796937   Min.   :-0.1912
##  Class :character   1st Qu.:-0.67773   1st Qu.:-0.571413   1st Qu.:-0.1867
##  Mode  :character   Median :-0.02996   Median : 0.029730   Median :-0.1847
##                     Mean   : 0.00000   Mean   :-0.004761   Mean   : 0.0000
##                     3rd Qu.: 0.61780   3rd Qu.: 0.656448   3rd Qu.:-0.1825
##                     Max.   : 2.23722   Max.   : 3.054022   Max.   : 8.9292
##                                        NA's   :1
##     Protein3            Protein4              OS               AFP|174
##  Min.   :-4.305365   Min.   :-3.01144   Min.   :-1.252133   Min.   :-0.284310
##  1st Qu.:-0.061882   1st Qu.:-0.54278   1st Qu.:-0.624931   1st Qu.:-0.274845
##  Median :-0.059570   Median : 0.08183   Median :-0.201720   Median :-0.250395
##  Mean   : 0.000488   Mean   : 0.00000   Mean   : 0.000945   Mean   : 0.001237
##  3rd Qu.:-0.056670   3rd Qu.: 0.70509   3rd Qu.: 0.408262   3rd Qu.:-0.150624
##  Max.   : 9.071134   Max.   : 1.78831   Max.   : 6.745844   Max.   : 7.967143
##  NA's   :2                              NA's   :1           NA's   :1
##     ALPL|249          BEND3|57673        C1orf25|81627     C20orf160|140706
##  Min.   :-0.547027   Min.   :-1.310806   Min.   :-1.3598   Min.   :-1.1972
##  1st Qu.:-0.406269   1st Qu.:-0.673822   1st Qu.:-0.6844   1st Qu.:-0.6567
##  Median :-0.287012   Median :-0.281337   Median :-0.2643   Median :-0.2524
##  Mean   :-0.002505   Mean   : 0.004131   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.028689   3rd Qu.: 0.359936   3rd Qu.: 0.4434   3rd Qu.: 0.3706
##  Max.   :12.068296   Max.   : 5.548031   Max.   : 5.1884   Max.   : 5.9654
##  NA's   :1           NA's   :2
##    CEBPD|1052          CFD|1675          CMTM7|112616
##  Min.   :-1.083079   Min.   :-0.462557   Min.   :-1.087057
##  1st Qu.:-0.687529   1st Qu.:-0.385792   1st Qu.:-0.624524
##  Median :-0.346241   Median :-0.295525   Median :-0.317798
##  Mean   : 0.001924   Mean   : 0.000000   Mean   :-0.009375
##  3rd Qu.: 0.361783   3rd Qu.: 0.001332   3rd Qu.: 0.211882
##  Max.   : 4.726307   Max.   :11.454115   Max.   : 5.115256
##  NA's   :1                               NA's   :4
##    ELF3|1999           FOLR1|2348         GLB1L2|89944
##  Min.   :-1.441554   Min.   :-0.354123   Min.   :-1.099397
##  1st Qu.:-0.633441   1st Qu.:-0.338950   1st Qu.:-0.741613
##  Median :-0.218502   Median :-0.284453   Median :-0.262137
##  Mean   :-0.001614   Mean   : 0.001561   Mean   :-0.001927
##  3rd Qu.: 0.221019   3rd Qu.:-0.084424   3rd Qu.: 0.396619
```

```
##  Max.   : 4.882205   Max.   :11.011026   Max.   : 5.644007
##  NA's   :1           NA's   :1           NA's   :1
##   GREB1L|80000        IGDCC3|9543         LANCL2|55915
##  Min.   :-0.887600   Min.   :-0.488096   Min.   :-1.647428
##  1st Qu.:-0.695698   1st Qu.:-0.462958   1st Qu.:-0.738457
##  Median :-0.365639   Median :-0.353607   Median :-0.245192
##  Mean   : 0.007401   Mean   : 0.003847   Mean   :-0.003021
##  3rd Qu.: 0.404548   3rd Qu.:-0.028067   3rd Qu.: 0.499579
##  Max.   : 5.461549   Max.   : 8.843201   Max.   : 4.183994
##  NA's   :3           NA's   :2           NA's   :1
##  MGC72080|389538      MYEOV|26579       NANOS1|340719        NRSN1|140767
##  Min.   :-1.160431   Min.   :-0.53943   Min.   :-0.966805   Min.   :-0.411340
##  1st Qu.:-0.588695   1st Qu.:-0.49293   1st Qu.:-0.493788   1st Qu.:-0.328997
##  Median :-0.247934   Median :-0.40833   Median :-0.185555   Median :-0.269072
##  Mean   :-0.001931   Mean   : 0.00000   Mean   : 0.002197   Mean   : 0.000874
##  3rd Qu.: 0.204750   3rd Qu.:-0.05771   3rd Qu.: 0.230558   3rd Qu.:-0.123356
##  Max.   : 7.876135   Max.   : 6.30377   Max.   : 9.596070   Max.   : 9.142145
##  NA's   :1                              NA's   :2           NA's   :1
##   PEG10|23089          PER2|8864           PI3|5266          RBP7|116362
##  Min.   :-0.447249   Min.   :-1.43694   Min.   :-0.369071   Min.   :-0.916977
##  1st Qu.:-0.428746   1st Qu.:-0.69379   1st Qu.:-0.354610   1st Qu.:-0.629407
##  Median :-0.347283   Median :-0.28997   Median :-0.325689   Median :-0.339669
##  Mean   : 0.003535   Mean   :-0.00201   Mean   : 0.001569   Mean   :-0.001984
##  3rd Qu.:-0.132306   3rd Qu.: 0.43148   3rd Qu.:-0.156677   3rd Qu.: 0.176945
##  Max.   : 5.394616   Max.   : 4.19267   Max.   : 9.941546   Max.   : 4.668243
##  NA's   :2           NA's   :1          NA's   :1           NA's   :2
##    RPLP2|6181        RPS20|6224        SFTPA2|729238      TACSTD2|4070
##  Min.   :-1.3869   Min.   :-1.342988   Min.   :-0.3945   Min.   :-1.451991
##  1st Qu.:-0.6574   1st Qu.:-0.605396   1st Qu.:-0.3761   1st Qu.:-0.637656
##  Median :-0.2795   Median :-0.235413   Median :-0.3338   Median :-0.148425
##  Mean   : 0.0000   Mean   : 0.007805   Mean   : 0.0000   Mean   :-0.003193
##  3rd Qu.: 0.4398   3rd Qu.: 0.333691   3rd Qu.:-0.1331   3rd Qu.: 0.386918
##  Max.   : 5.9988   Max.   : 5.945397   Max.   : 9.5054   Max.   : 5.754379
##                    NA's   :3                             NA's   :1
##   TMPRSS2|7113       TRNP1|388610      ZNF394|84124      ZNF511|118472
##  Min.   :-0.97290   Min.   :-1.0747   Min.   :-1.6151   Min.   :-1.465808
##  1st Qu.:-0.67936   1st Qu.:-0.6916   1st Qu.:-0.6404   1st Qu.:-0.659086
##  Median :-0.29218   Median :-0.2968   Median :-0.1573   Median :-0.258971
##  Mean   : 0.00254   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.003377
##  3rd Qu.: 0.38883   3rd Qu.: 0.4099   3rd Qu.: 0.3459   3rd Qu.: 0.331805
##  Max.   : 5.52652   Max.   : 6.9039   Max.   : 3.7089   Max.   : 4.497304
##  NA's   :1                                              NA's   :3
```

```
# glimpse(def_data_temp)


## Gráfico de los datos perdidos creados con prodNA

aggr(def_data_temp, numbers=T, sortVars=T, labels=names(def_data_temp),
     cex.axis=.7, gap=3, ylab=c("Missing data", "Pattern"))
```




```
##
##  Variables sorted by number of missings:
##          Variable       Count
##      CMTM7|112616 0.017621145
##         ER.status 0.013215859
##      Surgery_type 0.013215859
##      GREB1L|80000 0.013215859
##        RPS20|6224 0.013215859
##     ZNF511|118472 0.013215859
##            Gender 0.008810573
##          Protein3 0.008810573
##       BEND3|57673 0.008810573
##       IGDCC3|9543 0.008810573
##     NANOS1|340719 0.008810573
##       PEG10|23089 0.008810573
##       RBP7|116362 0.008810573
##         Histology 0.004405286
##         PR.status 0.004405286
##              c_OS 0.004405286
##          Protein1 0.004405286
##                OS 0.004405286
##           AFP|174 0.004405286
##          ALPL|249 0.004405286
##        CEBPD|1052 0.004405286
##         ELF3|1999 0.004405286
##        FOLR1|2348 0.004405286
##      GLB1L2|89944 0.004405286
##      LANCL2|55915 0.004405286
```

```
##   MGC72080|389538 0.004405286
##      NRSN1|140767 0.004405286
##         PER2|8864 0.004405286
##          PI3|5266 0.004405286
##      TACSTD2|4070 0.004405286
##      TMPRSS2|7113 0.004405286
##      Tumour_Stage 0.000000000
##       HER2.status 0.000000000
##    Patient_Status 0.000000000
##               Age 0.000000000
##          Protein2 0.000000000
##          Protein4 0.000000000
##     C1orf25|81627 0.000000000
##  C20orf160|140706 0.000000000
##          CFD|1675 0.000000000
##       MYEOV|26579 0.000000000
##        RPLP2|6181 0.000000000
##     SFTPA2|729238 0.000000000
##      TRNP1|388610 0.000000000
##      ZNF394|84124 0.000000000
```

```
## Número de outliers en cada una de las variables
num_outliers <- as.data.frame(lengths(outliers(def_data_temp), use.names = T))
num_outliers <- tibble::rownames_to_column(num_outliers, "variable")
names(num_outliers) <- c("variable", "num outliers")

print(num_outliers)
```

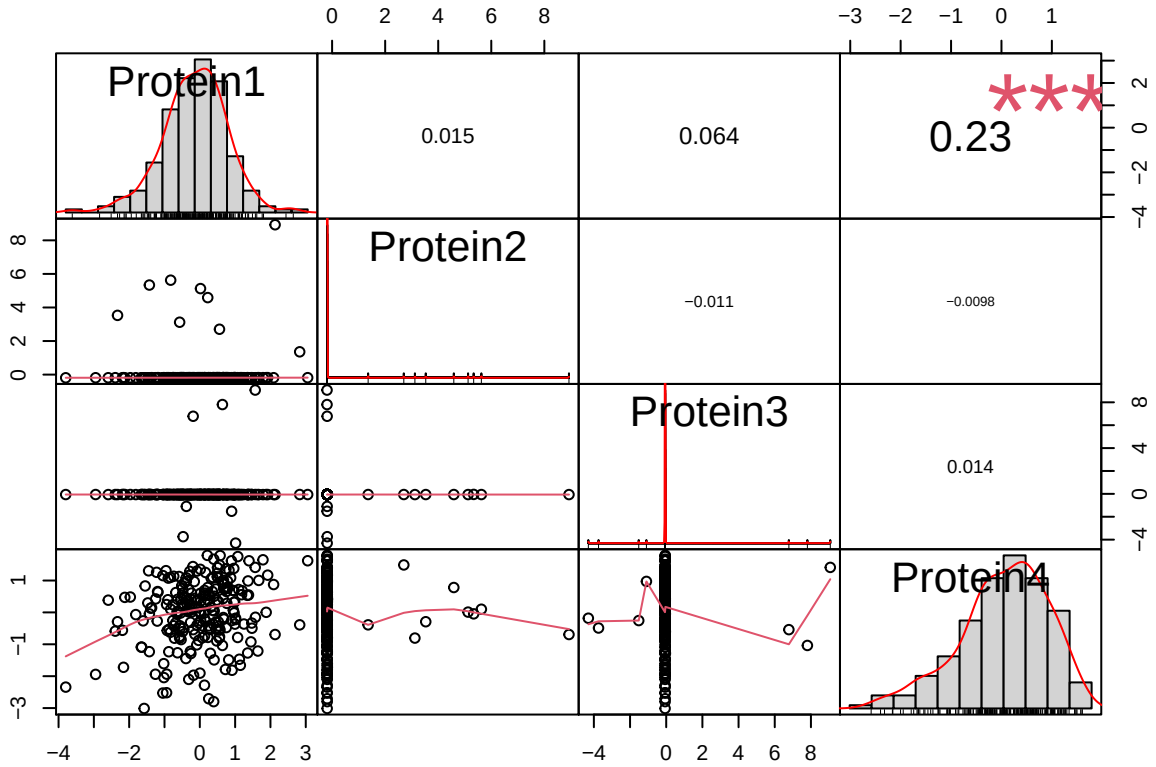
```
##            variable num outliers
## 1               Age            0
## 2          Protein1            5
## 3          Protein2            9
## 4          Protein3           10
## 5          Protein4            5
## 6                OS            9
## 7           AFP|174           30
## 8          ALPL|249           21
## 9       BEND3|57673           13
## 10    C1orf25|81627            7
## 11 C20orf160|140706           13
## 12       CEBPD|1052           10
## 13         CFD|1675           24
## 14     CMTM7|112616           16
## 15        ELF3|1999           15
## 16       FOLR1|2348           29
## 17     GLB1L2|89944            8
## 18     GREB1L|80000           11
## 19      IGDCC3|9543           25
## 20     LANCL2|55915            9
## 21  MGC72080|389538           13
## 22      MYEOV|26579           33
## 23    NANOS1|340719           12
## 24     NRSN1|140767           31
## 25      PEG10|23089           31
## 26        PER2|8864           11
```

```
## 27         PI3|5266           37
## 28      RBP7|116362           19
## 29       RPLP2|6181            9
## 30       RPS20|6224           13
## 31    SFTPA2|729238           35
## 32     TACSTD2|4070           13
## 33     TMPRSS2|7113           10
## 34     TRNP1|388610            8
## 35     ZNF394|84124           14
## 36    ZNF511|118472           15
```

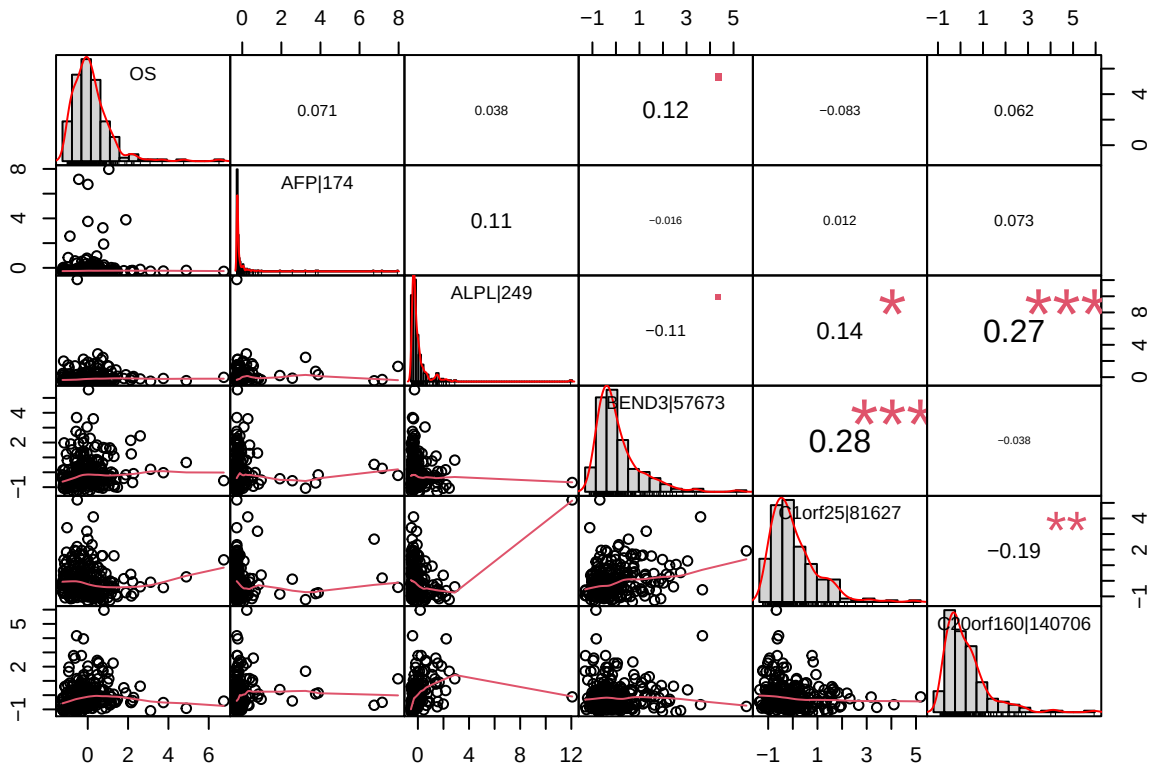
```
## Dataframe con los resultados de los tests de normalidad y homocedasticidad

kable(df_shapiro_levene, "simple")
```

	p_shapiro	Is_normally_distributed	p_levene	homocedasticity
Age	0.01800701	FALSE	0.1635941	homocedastico
Protein1	0.06897328	TRUE	0.1425891	homocedastico
Protein2	3.004165e-30	FALSE	0.2559155	homocedastico
Protein3	1.300341e-30	FALSE	0.1560008	homocedastico
Protein4	9.706782e-05	FALSE	0.5979542	homocedastico
OS	2.191928e-15	FALSE	0.0002861	heterocedastico
AFP|174	8.118072e-29	FALSE	0.1992899	homocedastico
ALPL|249	9.971425e-27	FALSE	0.6602906	homocedastico
BEND3|57673	2.834525e-14	FALSE	0.1890256	homocedastico
C1orf25|81627	1.419783e-12	FALSE	0.7033401	homocedastico
C20orf160|140706	3.46715e-15	FALSE	0.7606778	homocedastico
CEBPD|1052	7.102418e-15	FALSE	0.0136442	heterocedastico
CFD|1675	8.279016e-27	FALSE	0.7567185	homocedastico
CMTM7|112616	7.401108e-18	FALSE	0.0000011	heterocedastico
ELF3|1999	3.979074e-15	FALSE	0.0021043	heterocedastico
FOLR1|2348	8.157645e-28	FALSE	0.0022690	heterocedastico
GLB1L2|89944	3.444176e-13	FALSE	0.5910298	homocedastico
GREB1L|80000	2.615837e-17	FALSE	0.0045196	heterocedastico
IGDCC3|9543	8.105034e-25	FALSE	0.0094854	heterocedastico
LANCL2|55915	3.213574e-10	FALSE	0.3965358	homocedastico
MGC72080|389538	2.794284e-20	FALSE	0.1743462	homocedastico
MYEOV|26579	1.43382e-23	FALSE	0.0003884	heterocedastico
NANOS1|340719	3.331553e-23	FALSE	0.0846398	homocedastico
NRSN1|140767	1.25033e-27	FALSE	0.3448318	homocedastico
PEG10|23089	1.76877e-25	FALSE	0.8900544	homocedastico
PER2|8864	1.224014e-12	FALSE	0.0000295	heterocedastico
PI3|5266	5.038916e-27	FALSE	0.0001446	heterocedastico
RBP7|116362	2.132067e-18	FALSE	0.0105717	heterocedastico
RPLP2|6181	5.593193e-15	FALSE	0.1712887	homocedastico
RPS20|6224	1.961008e-15	FALSE	0.1601191	homocedastico
SFTPA2|729238	1.456948e-26	FALSE	0.0650492	homocedastico
TACSTD2|4070	5.190384e-13	FALSE	0.0332368	heterocedastico
TMPRSS2|7113	2.189011e-16	FALSE	0.0005645	heterocedastico
TRNP1|388610	6.531814e-16	FALSE	0.1695275	homocedastico
ZNF394|84124	9.369101e-11	FALSE	0.5217023	homocedastico
ZNF511|118472	8.605966e-13	FALSE	0.3824289	homocedastico

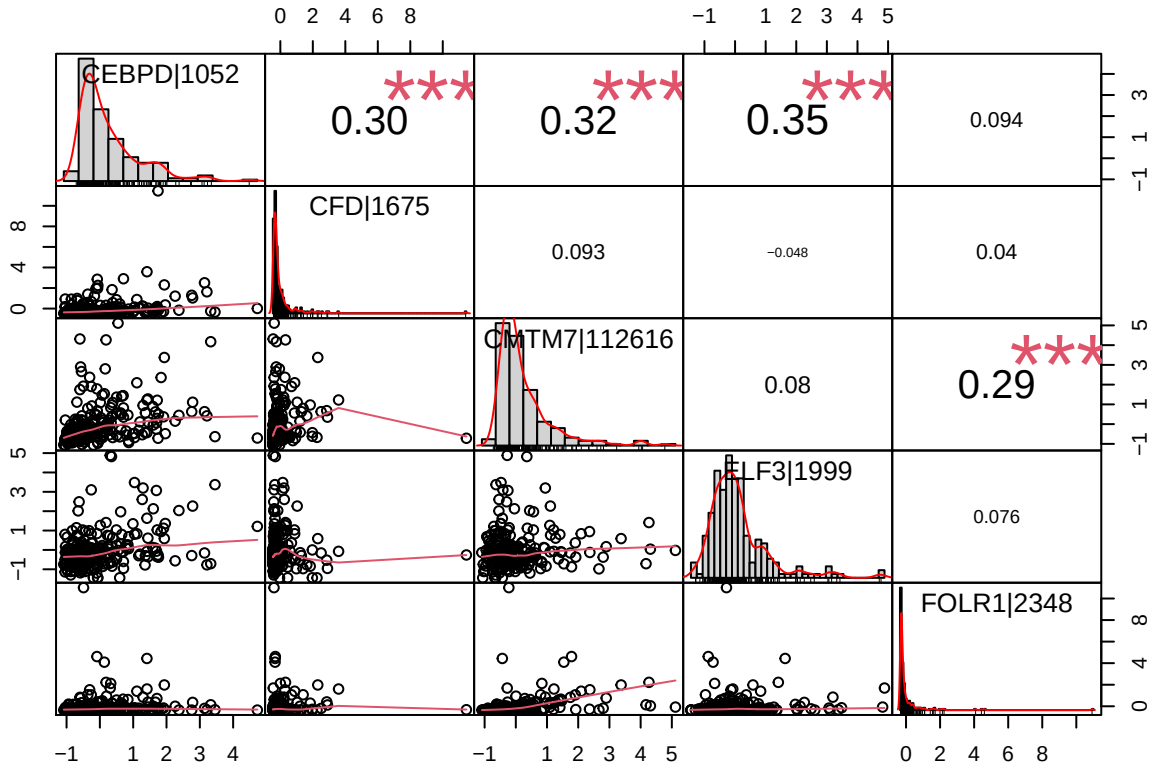
```
## Correlaciones entre la expresión de las proteínas

chart.Correlation(def_data[,11:14])
```



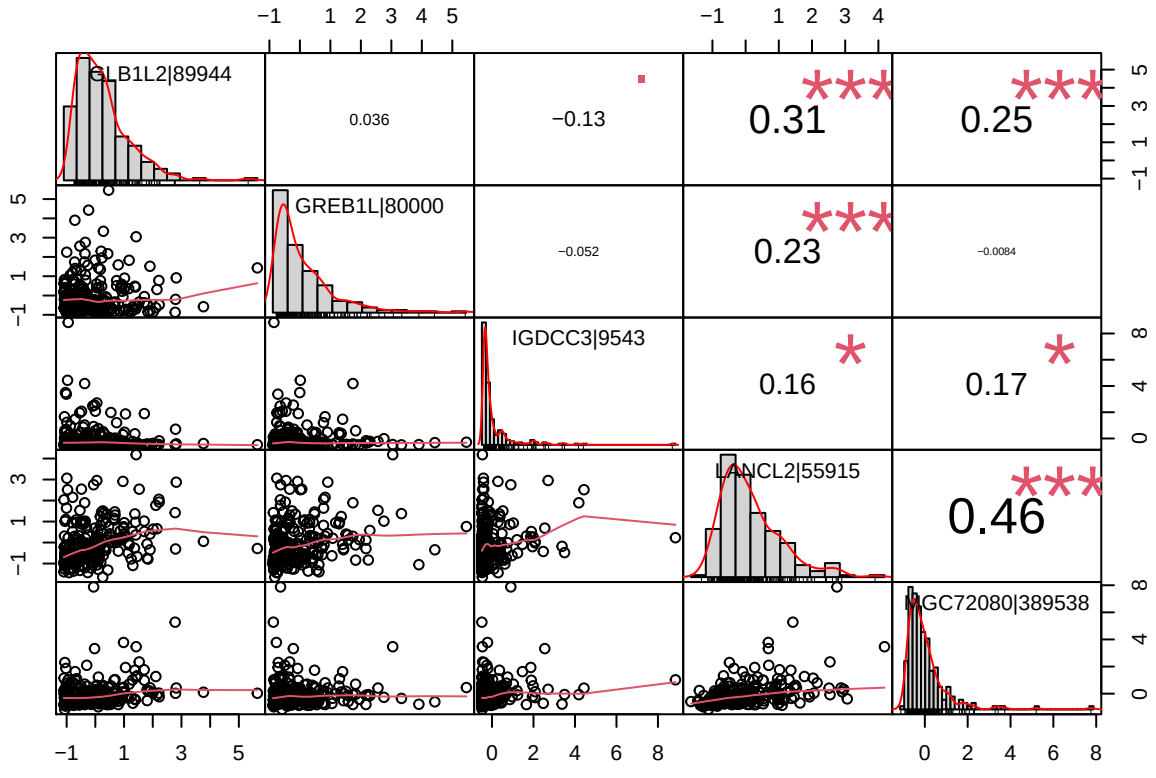

```
## Correlaciones entre la expresión génica

chart.Correlation(def_data[,15:20])
```



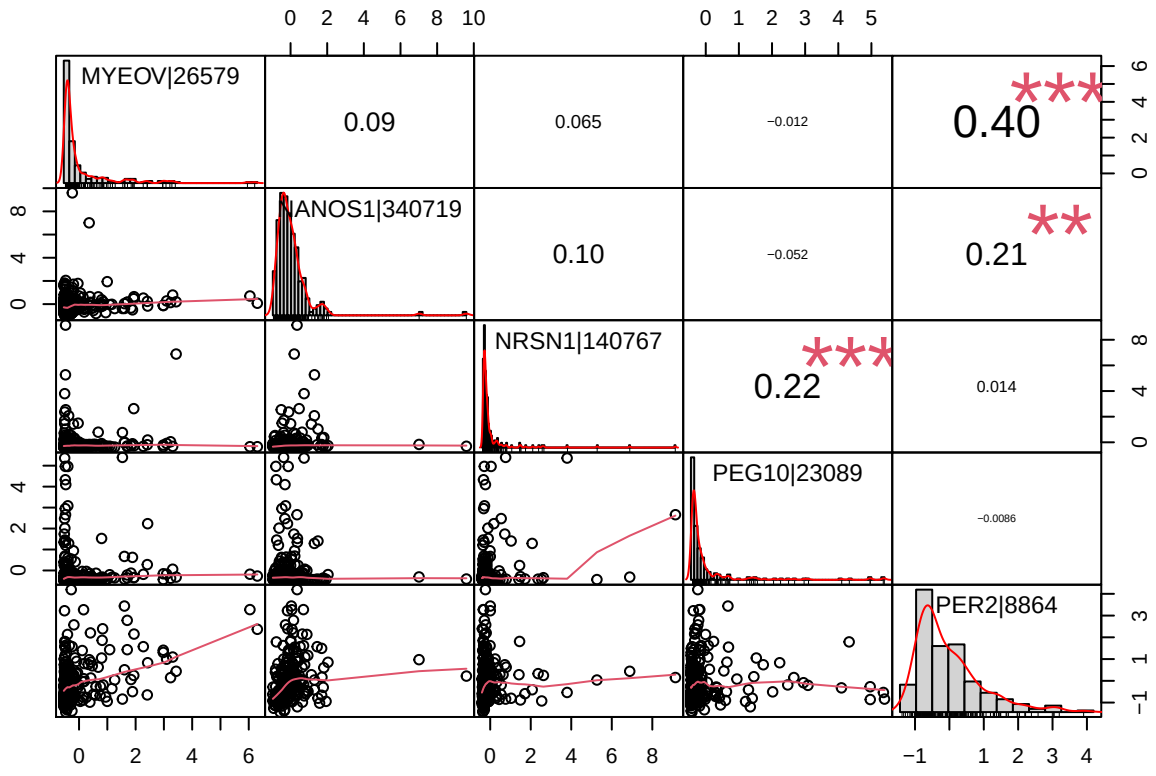

```
chart.Correlation(def_data[,21:25])
```



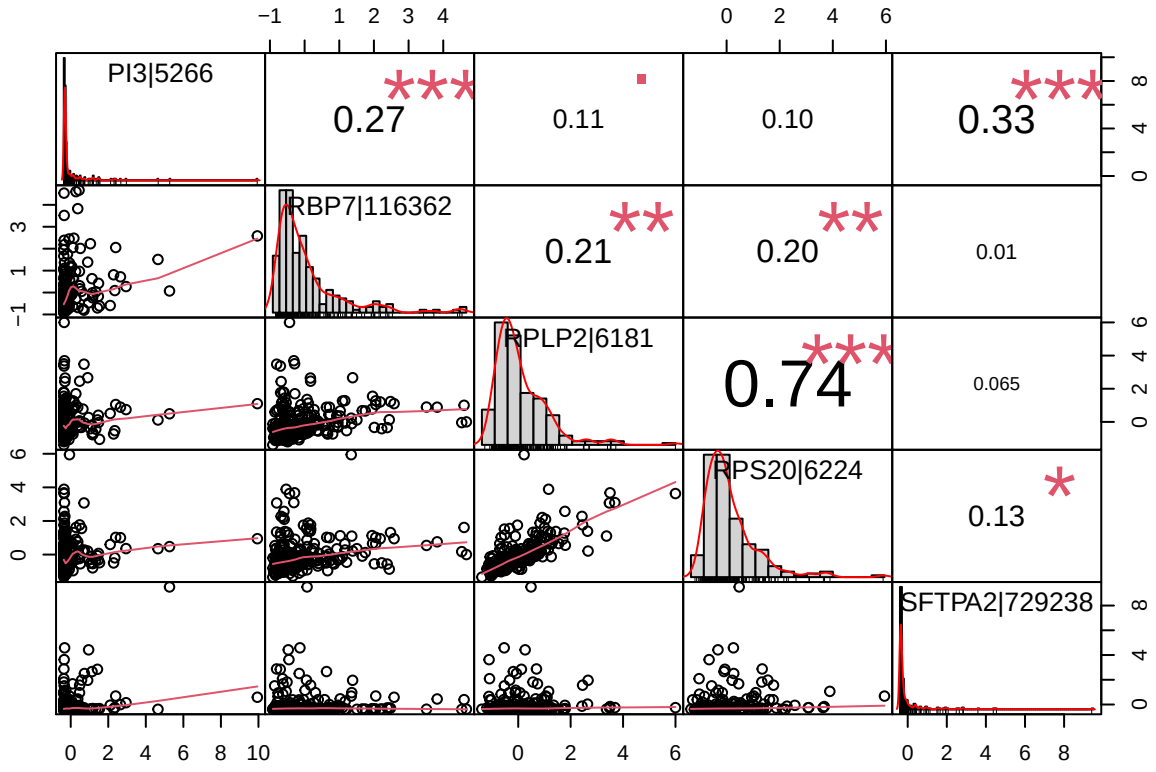

```
chart.Correlation(def_data[,26:30])
```



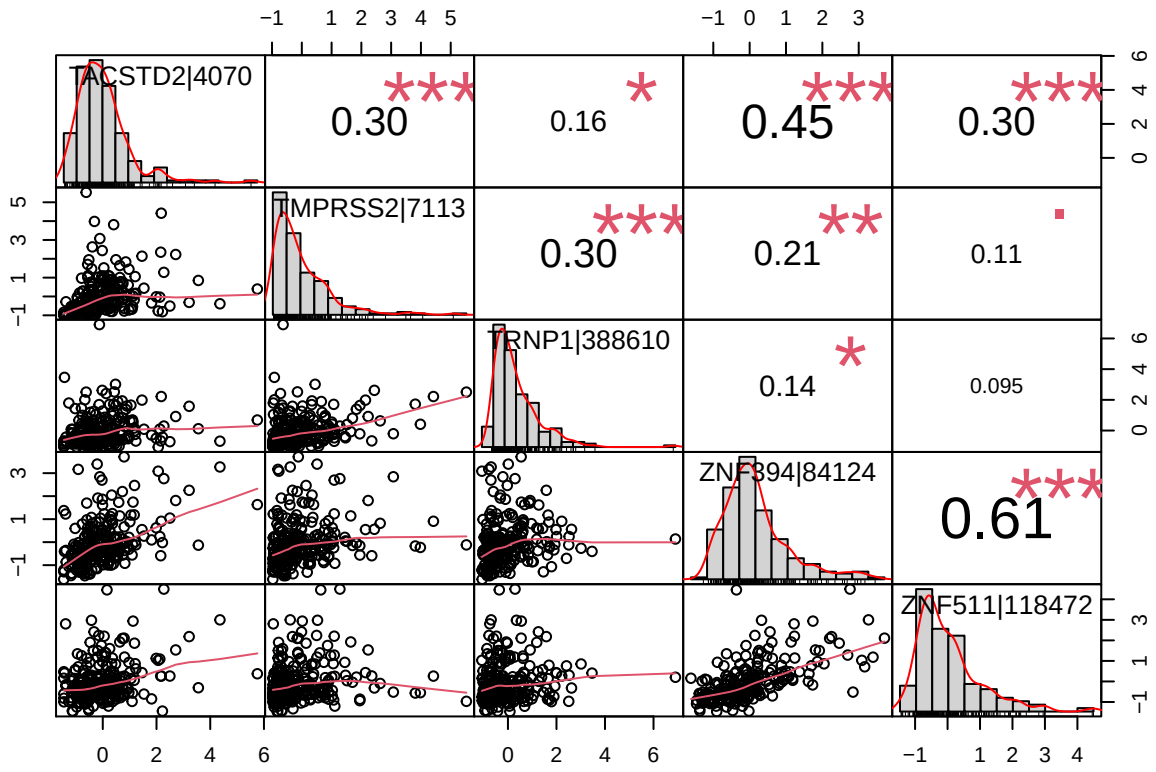

```
chart.Correlation(def_data[,31:35])
```




```
chart.Correlation(def_data[,36:40])
```




```
chart.Correlation(def_data[,41:length(def_data)])
```




```
# Se ve claramente, que sólo han quedado variables poco correlaciondas,
# la mayor R es de menos de 0.75
```

```
# Resultados de los test de inferencia (U-Mann Whitney)

kable(res_kw, "simple")
```

	p.value	Different_OS
Age	0.8633247	FALSE
Protein1	0.6719318	FALSE
Protein2	2.570825e-28	TRUE
Protein3	3.46589e-35	TRUE
Protein4	0.3778713	FALSE
OS	0.03901807	TRUE
AFP|174	1.033558e-16	TRUE
ALPL|249	2.533983e-07	TRUE
BEND3|57673	0.01210074	TRUE
C1orf25|81627	0.05277158	FALSE
C20orf160|140706	0.009570528	TRUE
CEBPD|1052	0.01585334	TRUE
CFD|1675	7.377862e-09	TRUE
CMTM7|112616	0.0009680427	TRUE
ELF3|1999	0.005095346	TRUE
FOLR1|2348	2.351847e-10	TRUE
GLB1L2|89944	0.06259174	FALSE
GREB1L|80000	0.004283811	TRUE
IGDCC3|9543	2.096438e-07	TRUE
LANCL2|55915	0.09934041	FALSE
MGC72080|389538	0.0007184457	TRUE
MYEOV|26579	6.274239e-07	TRUE
NANOS1|340719	0.001153291	TRUE
NRSN1|140767	7.163483e-13	TRUE
PEG10|23089	9.56845e-11	TRUE
PER2|8864	0.03421817	TRUE
PI3|5266	2.467893e-09	TRUE
RBP7|116362	0.0006267007	TRUE
RPLP2|6181	0.03623579	TRUE
RPS20|6224	0.0107544	TRUE
SFTPA2|729238	1.979468e-09	TRUE
TACSTD2|4070	0.0479089	TRUE
TMPRSS2|7113	0.01512228	TRUE
TRNP1|388610	0.01818641	TRUE
ZNF394|84124	0.03797801	TRUE
ZNF511|118472	0.02873148	TRUE

```
# Resultados del test de Chi-cuadrado sobre las variables no numéricas

kable(res_chi, "simple")
```

Tumour_Stage	Histology	HER2.status	Surgery_type	Patient_Status	c_OS
0.5425755	0.3204715	0.6138603	0.0269971	1	0

```r
# Resultados de la correlación entre la supervivencia global y las variables numéricas de edad y expresi

kable(res_cor, "simple")
```

Age	Protein1	Protein2	Protein3	Protein4	OS	AFP|174	ALPL|249	BEND3|57673	C1orf25|8162
0.7056768	0.7989389	0.6007501	0.4437977	0.4945913	0	0.2867525	0.5652249	0.0741947	0.215061

```r
# Una conclusión que sacamos, es que la limpieza del dataframe ha sido efectiva, casi todos los atribut
# correlacionan bien con la variable respuesta.


# Accuaracy de los 4 modelos de Machine Learning
a <- c(1,1)
b <- c(0.9783, 1)
res_model <- cbind(a,b)
rownames(res_model) <- c("RF", "XGB")
colnames(res_model) <- c("NAs removed", "NAs imputed")

kable(res_model, "simple")
```

	NAs removed	NAs imputed
RF	1	0.9783
XGB	1	1.0000

6.- Resolución del problema

En primer lugar, la comparación de las variables numéricas mediante el test no paramétrico U-Mann Whitney (recordemos que sólo el atributo de la expresión de la proteína 1 está normalmente distribuido) nos indican que más del 83% de las variables tienen una diferencia estadísticamente significativa entre los grupos de alta y baja supervivencia global. Esto es indicativo de la bondad de todos los procesos aplicados a la limpieza del dataframe, seleccionando aquellos atributos más informativos.

La gestión de elementos vacíos se llevan a cabo dos estrategias; la eliminación y la asignación con un algoritmo basado en k-nearest neighbors, estos dos datasets se usan en 2 algoritmos de machine learning (Random Forest y Extreme Gradient Boosting), a fin de seleccionar el que mejor prediga la variante respuesta, ese dataset será el más adecuado, y por tanto el que se devolverá después de la limpieza, integración y preparación de los datos. A su vez, este enfoque nos responde a la idoneidad de estos datos para predecir la variable respuesta, en este caso la supervivencia global tumorespecífica, mostrando un gran poder predictor usando ambos algoritmos, por tanto llegamos a la conclusión de la idoneidad de los datos.

7.- Bibliografía

- [1] https://www.kaggle.com/sarwat182/breast-cancer-analysis/
- [2] Manni A. Endocrine therapy of breast and prostate cancer. Endocrinol Metab Clin North Am. 1989 Jun;18(2):569-92. PMID: 2663486.
- [3] Perou CM, Sørlie T, Eisen MB, van de Rijn M, Jeffrey SS, Rees CA, Pollack JR, Ross DT, Johnsen H, Akslen LA, Fluge O, Pergamenschikov A, Williams C, Zhu SX, Lønning PE, Børresen-Dale AL, Brown PO, Botstein D. Molecular portraits of human breast tumours. Nature. 2000 Aug 17;406(6797):747-52. doi: 10.1038/35021093. PMID: 10963602.
- [4] Dai X, Li T, Bai Z, et al. Breast cancer intrinsic subtype classification, clinical use and future trends. Am J Cancer Res. 2015;5(10):2929-2943. Published 2015 Sep 15.
- [5] https://www.cancer.net/es/tipos-de-c%C3%A1ncer/c%C3%A1ncer-de-mama/tipos-de-tratamiento
- [6] Collins FS, Morgan M, Patrinos A. The Human Genome Project: lessons from large-scale biology. Science. 2003 Apr 11;300(5617):286-90. doi: 10.1126/science.1084564. PMID: 12690187
- [7] ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) Project. Science. 2004 Oct 22;306(5696):636-40. doi: 10.1126/science.1105136. PMID: 15499007.
- [8] Cancer Genome Atlas Network. Comprehensive molecular portraits of human breast tumours. Nature. 2012 Oct 4;490(7418):61-70. doi: 10.1038/nature11412. Epub 2012 Sep 23. PMID: 23000897; PMCID: PMC3465532.
- [9] https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a