Algoritmos de Machine Learning aplicados a la predicción de recaída y supervivencia de cáncer de mama utilizando datos de expresión genómica

Gloria Vidaña Bedera

Preparaciones previas

Instalación de paquetes

```
library(affy)
library(genefilter)
library(samr)
library(siggenes)
library(e1071)
library(nnet)
library(kernlab)
library(pander)
library(randomForest)
library(ROCR)
library(tidyverse)
library(ISLR)
library(glmnet)
library(ggplot2)
library(aod)
library(caret)
library(Matrix)
library(annotate)
library(hgu133a.db)
library(survival)
library(survminer)
library(KMsurv)
library(ggfortify)
```

Obtención de la semilla

```
semilla <- 224
```

Lectura de datos

Obtención de las muestras

1. Descargar el dataset GSE2034 de la página https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE2034
2. Descomprimir los .CEL y dejarlos en la carpeta GSE2034

```
numeros <- 36777:37062
ficheros <- paste("GSE2034/GSM", numeros, ".CEL", sep = "")
breast_cancer_2034 <- justRMA(filenames = ficheros)
```

Obtención de la matriz de expresión

```
matriz_exprs_2034 <- exprs( breast_cancer_2034)
dim(matriz_exprs_2034)
#head(matriz_exprs_2034)
```

- Tras la lectura de datos, se aprecia que hay 286 muestras y 22283 genes.

Obtención de los phenotipos

- Descargamos la tabla de phenotipos de la página WEB https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE2034.

```
phenotipos <- read.csv("./GEO Accession viewer.csv", header=T)
dim(phenotipos)
pander(head(phenotipos[1:10,]))
#head(phenotipos[1:10,])
#Al mostrar los datos, se ve que la primera columna es un índice. Se elimina.
phenotipos <- phenotipos[,2:7]
dim(phenotipos)
grep( "relapse", colnames( phenotipos) )
#[1] 4 5 7
# Las columna 4 es la interesante, ya que indica las recaídas.
labels_relapse <- as.factor( phenotipos[,4])
pander(table(labels_relapse))
```

Preprocesado de la matriz_exprs_2034

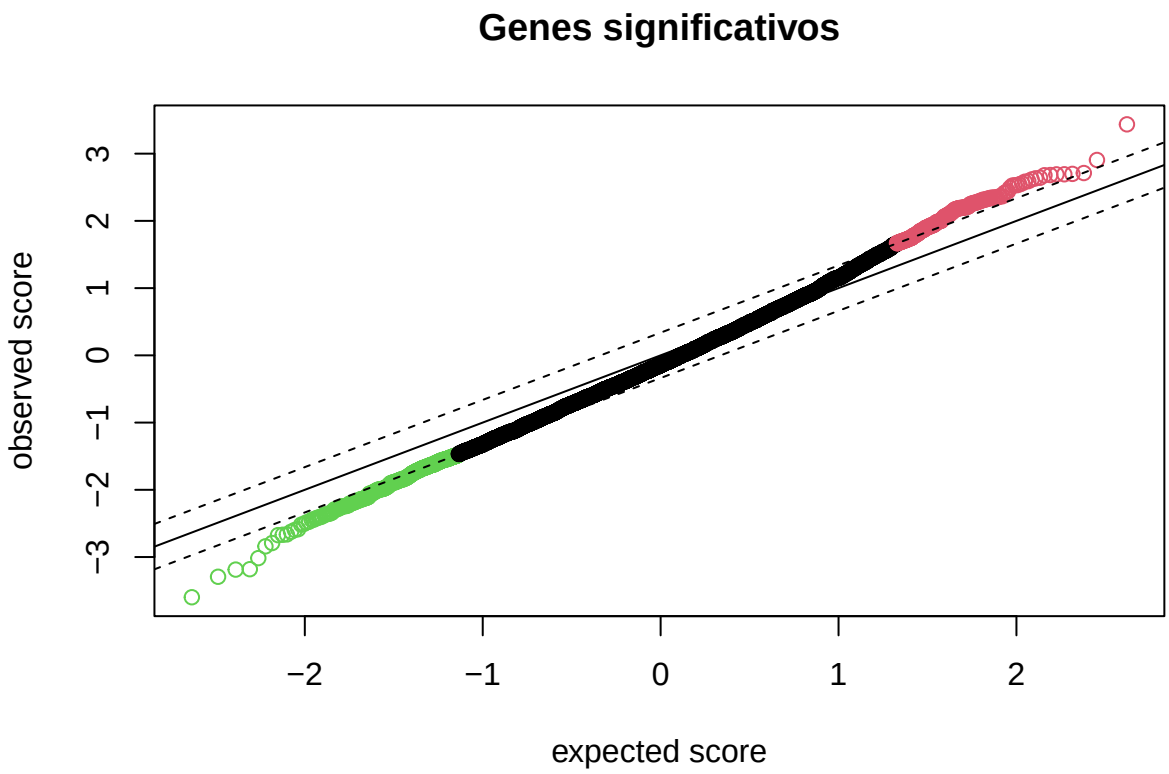
Filtrado y selección de los genes relevantes

```
#filtros
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.25)
f3 <- function(x) (median(2^x) > 150)
ff <- filterfun(f1, f2, f3)

#Filtrado
seleccion <- genefilter(matriz_exprs_2034, ff)
pander(summary(seleccion))
filtrado <- matriz_exprs_2034[seleccion,]
```

- Para el primer procesado se tomarán 7383 genes.

Reducción mediante test estadístico de análisis diferencial

```
set.seed(semilla)
genenames <- as.list(rownames(filtrado))
segundo_filtrado <- SAM(filtrado, labels_relapse, resp.type = "Two class unpaired",
                        geneid = genenames)
pander(summary(segundo_filtrado))
plot(segundo_filtrado)
title("Genes significativos")
```



#Generar conjuntos de entrenamiento y test

```
genes_up <- segundo_filtrado$siggenes.table$genes.up
genes_up <- as.data.frame(genes_up)

genes_low <- segundo_filtrado$siggenes.table$genes.lo
genes_low <- as.data.frame(genes_low)

total <- rbind(genes_up, genes_low)

keygenes <- total$`Gene Name`
keygenes <- unlist(keygenes)

filtrado <- filtrado[keygenes,]
#Número de genes con los que finalmente se trabajará.
pander(nrow(filtrado))
```

Clasificación mediante red neuronal

Programación de la red neuronal tipo RBF

```
set.seed(semilla)

muestra <- sample(1:286, 0.8*286)
a <- 1:2387
filtrado <- t(filtrado)

red_neuronal <- nnet(x = filtrado[muestra], y = class.ind(labels_relapse[muestra]),
                     size = 100, linout = F, skip = T, maxit = 1000)
predicciones_red <- predict(red_neuronal, newdata = filtrado[-muestra, ], type = "raw")
clases_predicciones <- apply(predicciones_red, MARGIN = 1, FUN = 'which.is.max')
clases_predicciones <- as.factor(clases_predicciones)
levels(clases_predicciones)<- c(0,1)

#matriz de confusión para comparar el valor predicho con el real
rbf_confusion <- table(clases_predicciones,  labels_relapse[-muestra])
rbf_exactitud <- sum(diag(rbf_confusion))/sum(rbf_confusion)

pander(rbf_confusion)
pander(rbf_exactitud)

confusionMatrix(table(clases_predicciones,  labels_relapse[-muestra]))
```

Programación de la red neuronal tipo SVM

- Para contrastar los resultados, se realizará sobre la misma matriz una predicción con una red neuronal tipo SVM, que clasifica y predice sin mínimos locales.

```
set.seed(semilla)

svm_classifier <- ksvm(labels_relapse[muestra] ~ ., data = filtrado[muestra,],
                       kernel = "rbfdot")

svm_classifier

#predicciones sobre el conjunto de test del dataset
svm_predictions <- predict(svm_classifier, filtrado[-muestra,])

head(svm_predictions)

svm_confusion <- table(svm_predictions,  labels_relapse[-muestra])
pander(svm_confusion)

svm_exactitud <- sum(diag(svm_confusion))/sum(svm_confusion)
pander(svm_exactitud)

confusionMatrix(table(svm_predictions,  labels_relapse[-muestra]))
```

Clasificación mediante perceptrón monocapa

```
set.seed(semilla)

pmonocapa <- nnet( filtrado[muestra], class.ind( labels_relapse[muestra]),
                   size=1, MaxNWts=1000, decay=5e-3, entropy=T)

predicciones_monocapa<- predict(pmonocapa, newdata = filtrado[-muestra, ], type = "raw")
clases_predicciones2 <- apply(predicciones_monocapa, MARGIN = 1, FUN = 'which.is.max')
clases_predicciones2 <- as.factor(clases_predicciones2)
levels(clases_predicciones2)<- c(0,1)

#matriz de confusión para comparar el valor predicho con el real
pmn_confusion <- table(clases_predicciones2,  labels_relapse[-muestra])
pander(pmn_confusion)

pmn_exactitud <- sum(diag(pmn_confusion))/sum(pmn_confusion)
pander(pmn_exactitud)

confusionMatrix(table(clases_predicciones2,  labels_relapse[-muestra]))
```

Clasificación mediante perceptrón multicapa

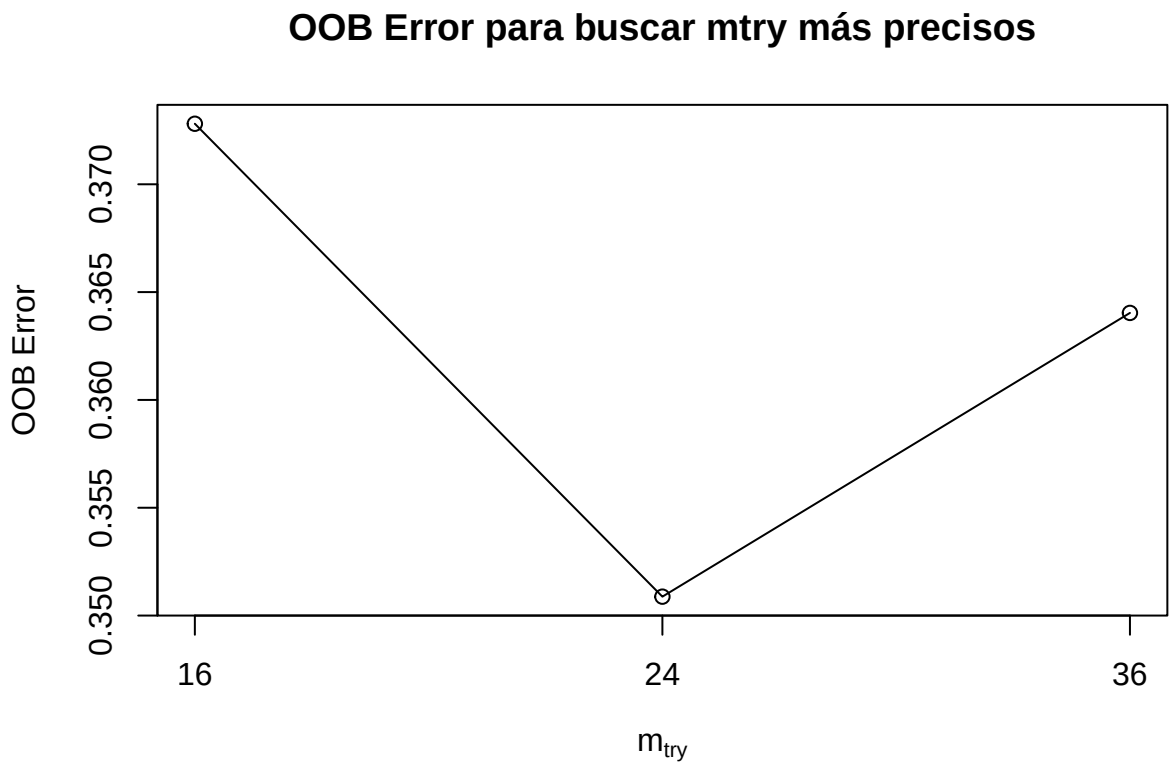
```
set.seed(semilla)

pmulticapa <- nnet( filtrado[muestra], class.ind( labels_relapse[muestra]) ,
                    size=100, MaxNWts=1000, decay=5e-3, entropy=T)

predicciones_multicapa <- predict(pmulticapa, newdata = filtrado[-muestra,], type = "raw")
clases_predicciones3 <- apply(predicciones_multicapa, MARGIN = 1, FUN = 'which.is.max')
clases_predicciones3 <- as.factor(clases_predicciones3)
levels(clases_predicciones3)<- c(0,1)

#matriz de confusión para comparar el valor predicho con el real
pml_confusion <- table(clases_predicciones3,  labels_relapse[-muestra])
pml_exactitud <- sum(diag(pml_confusion))/sum(pml_confusion)

pander(pml_confusion)
pander(pml_exactitud)

confusionMatrix(table(clases_predicciones3,  labels_relapse[-muestra]))
```

```
neuronas <- seq(from=10, to=100, length=3)
regulizacion <- seq(from=10^(-3), to=10^1, length=3)

error_pm = NULL
n_neuronas = NULL
n_regularizacion = NULL
for(i in neuronas){
```

```
  for(j in regulizacion){
    pmulticapa <- nnet( filtrado[muestra], class.ind( labels_relapse[muestra]) ,
                        size=i, MaxNWts=1000, decay=j, entropy=T)

    predicciones_multicapa <- predict(pmulticapa, newdata=filtrado[-muestra, ], type="raw")
    clases_predicciones3 <- apply(predicciones_multicapa, MARGIN = 1, FUN='which.is.max')
    clases_predicciones3 <- as.factor(clases_predicciones3)
    levels(clases_predicciones3)<- c(0,1)

    pml_confusion <- table(clases_predicciones3,  labels_relapse[-muestra])
    pml_exactitud <- sum(diag(pml_confusion))/sum(pml_confusion)

    error_pm <- c(pml_exactitud, error_pm)
    n_neuronas <- c(n_neuronas, i)
    n_regularizacion <- c(n_regularizacion, j)
  }

}
error_pm

error_ij <- list(error_pm, n_neuronas, n_regularizacion)
error_ij

#Buscar valor minimo y sus parametros
order(error_pm)
ordenado <- order(error_pm)
ordenado
error_ij[[1]][ordenado]
error_ij[[2]][ordenado]
error_ij[[3]][ordenado]

#Error: 0.6896552 - Neuronas: 55 - Regularización: 10.0000 (10)
#(buscamos que tenga menos neuronas y mayor regularizacion)
#Acierto maximo: 0.6896552 -> Error minimo = 1 - 0.6896552 = 0.3103448
```

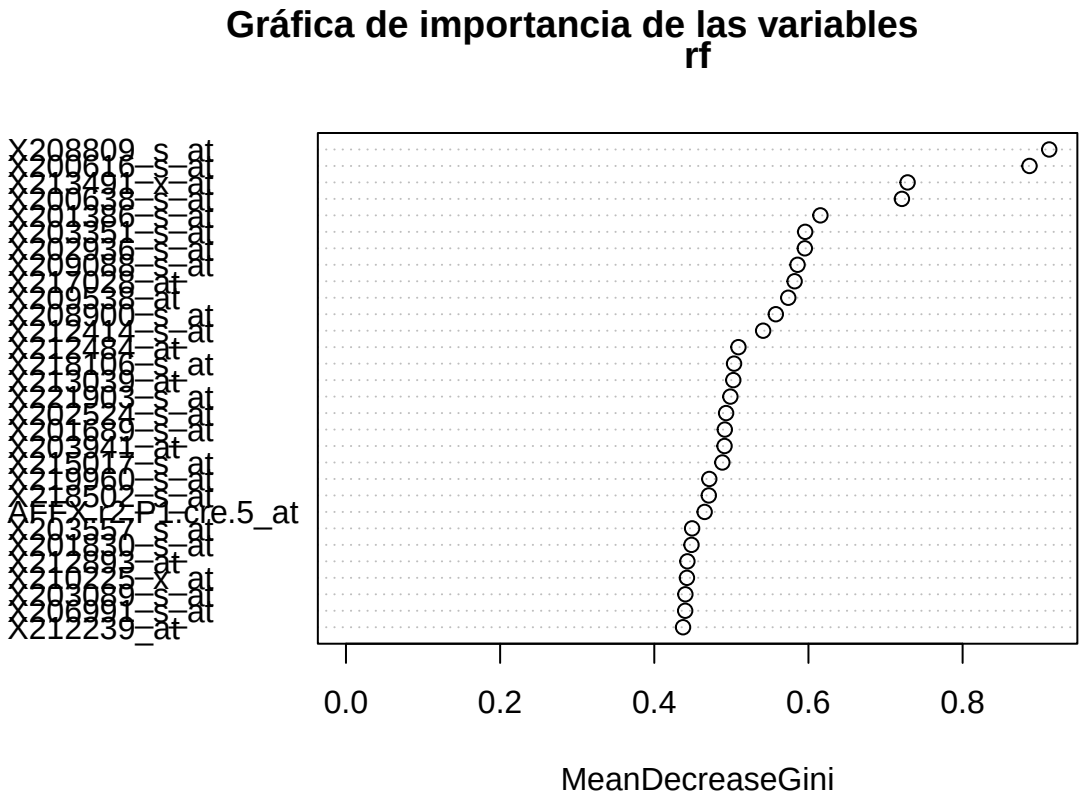
Clasificación mediante Random Forest

```
datos1 <- data.frame(labels_relapse)
datos2 <- data.frame(filtrado)

set.seed(semilla)
arboles <- 100

rf <- randomForest(datos1[muestra,] ~ . , data = datos2[muestra,], ntree = arboles,
                   xtest = as.data.frame(filtrado[-muestra, ]),
                   ytest = labels_relapse[-muestra])
pander(rf)

rf1_exactitud <- (33+6)/58
pander(rf1_exactitud)

floor(sqrt(ncol(datos2) - 1))
mtry <- tuneRF(datos2[muestra,], datos1[muestra,], ntreeTry = arboles, stepFactor = 1.5,
               improve = 0.01, trace = TRUE, plot = TRUE)
title("OOB Error para buscar mtry más precisos")
```



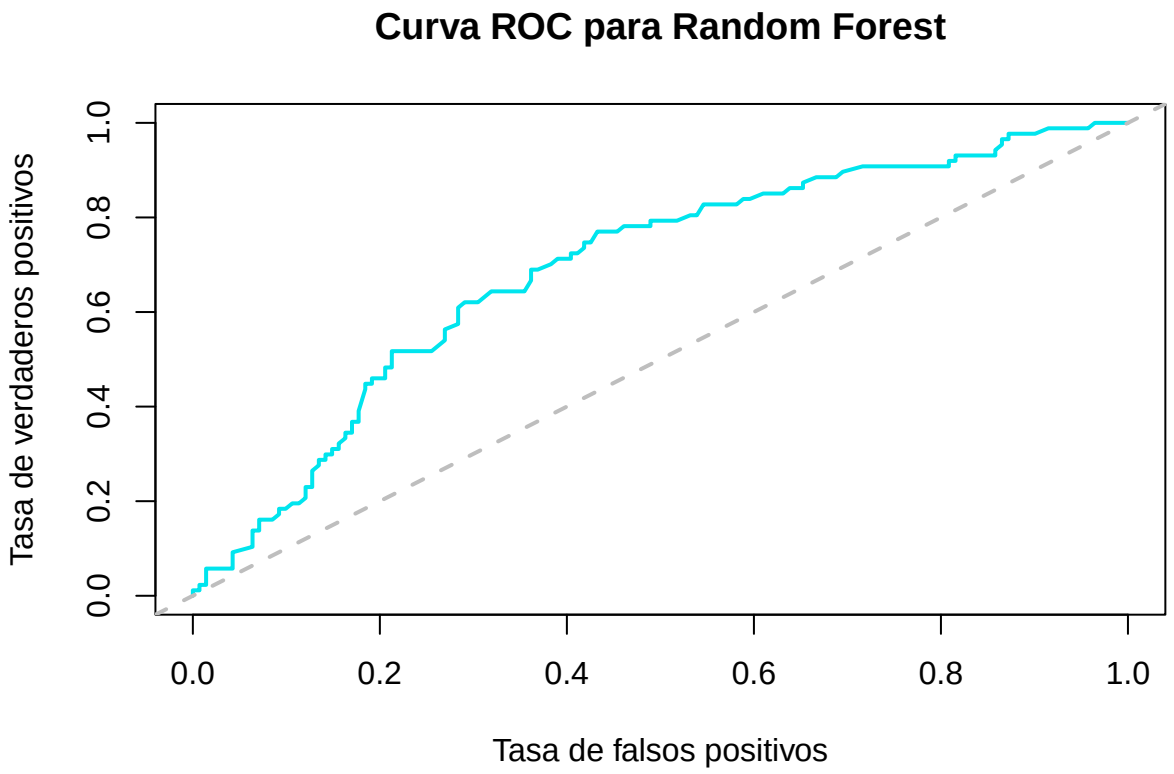

```
best.m <- mtry[mtry[,2] == min(mtry[,2]),1]
pander(mtry)
pander(best.m)
```

```
#Repetimos rf pero ahora con mtry = best.m
rf2 <- randomForest(datos1[muestra,] ~ . , data = datos2[muestra,], mtry = best.m,
                    importance = TRUE, ntree = arboles,
                    xtest = as.data.frame(filtrado[-muestra, ]),
                    ytest = labels_relapse[-muestra])
pander(rf2)

rf2_exactitud <- (35+5)/58
pander(rf2_exactitud)

rf <- randomForest(datos1[muestra,] ~ . , data = datos2[muestra,], ntree = arboles)

importance(rf)
varImpPlot(rf)
title("Gráfica de importancia de las variables", line = 2.5)
```



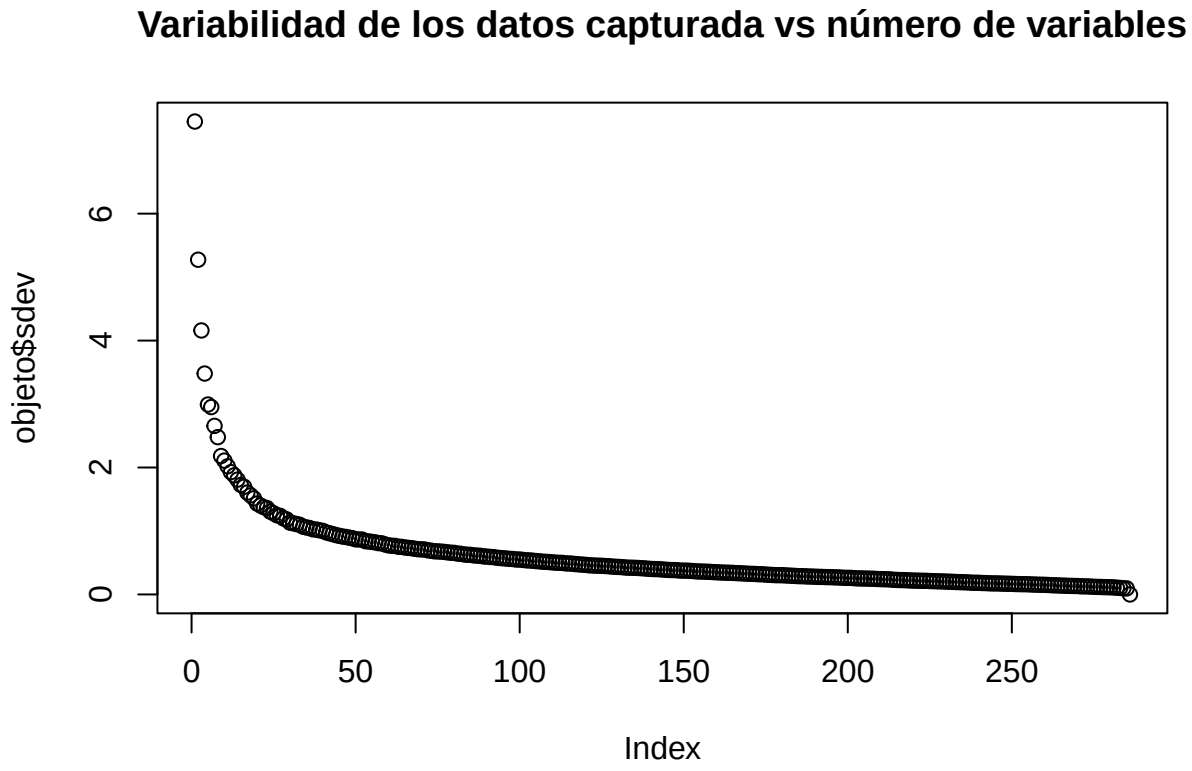

```
#Predecimos
pred1 <- predict(rf, type = "prob")
perf <- prediction(pred1[,2], datos1[muestra,])

# 1. Area bajo la curva
auc <- performance(perf, "auc")
str(auc)
auc@y.values
```

```r
# 2. Tasa de verdaderos y falsos positivos
pred3 <- performance(perf, "tpr","fpr")

# 3. Trazar la curva ROC
plot(pred3, main="Curva ROC para Random Forest",xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos", col="turquoise2",lwd=2)
abline(a=0,b=1,lwd=2,lty=2,col="gray")
```

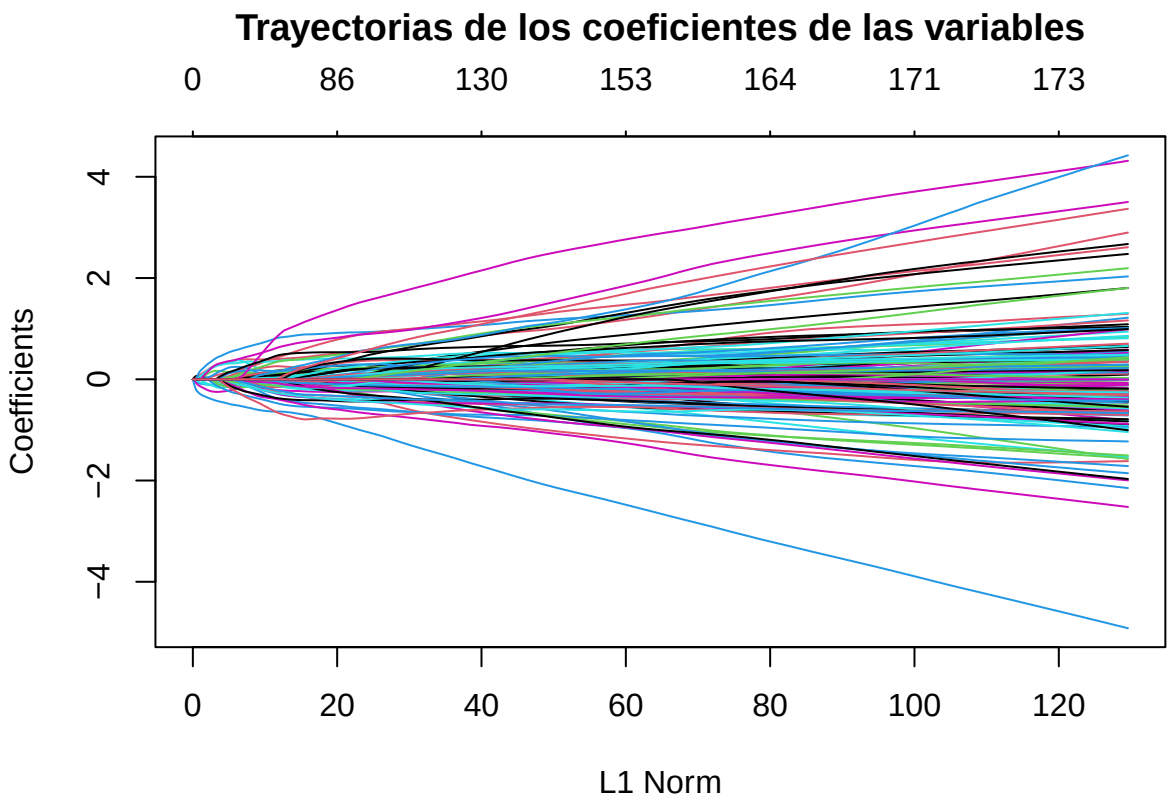


Clasificación mediante regresión logística

```
datos <-data.frame(labels_relapse, filtrado)

#colnames(datos)

table(datos$labels_relapse)

objeto <- prcomp(filtrado)
plot(objeto$sdev, col=1)
title("Variabilidad de los datos capturada vs número de variables")
```




```
filtrado.prcomp <- objeto$x[, 1:50]

datos_modelo <-data.frame(labels_relapse, filtrado.prcomp)

modelo <- glm(labels_relapse ~ ., data = datos_modelo, family = "binomial")
summary(modelo)

prl <- glmnet(x=filtrado, y=labels_relapse, family = "binomial", alpha = 1, lambda = NULL)
summary(prl)
plot(prl)
title("Trayectorias de los coeficientes de las variables", line = 2.5)
```




```
predicciones_rl<- predict(prl, newx = filtrado[-muestra, ], type = "class")
summary(predicciones_rl)

#matriz de confusión para comparar el valor predicho con el real
rl_confusion <- table(predicciones_rl[, 35],  labels_relapse[-muestra])
rl_exactitud <- sum(diag(rl_confusion))/sum(rl_confusion)

pander(rl_confusion)
pander(rl_exactitud)

confusionMatrix(table(predicciones_rl[, 35],  labels_relapse[-muestra]))
```

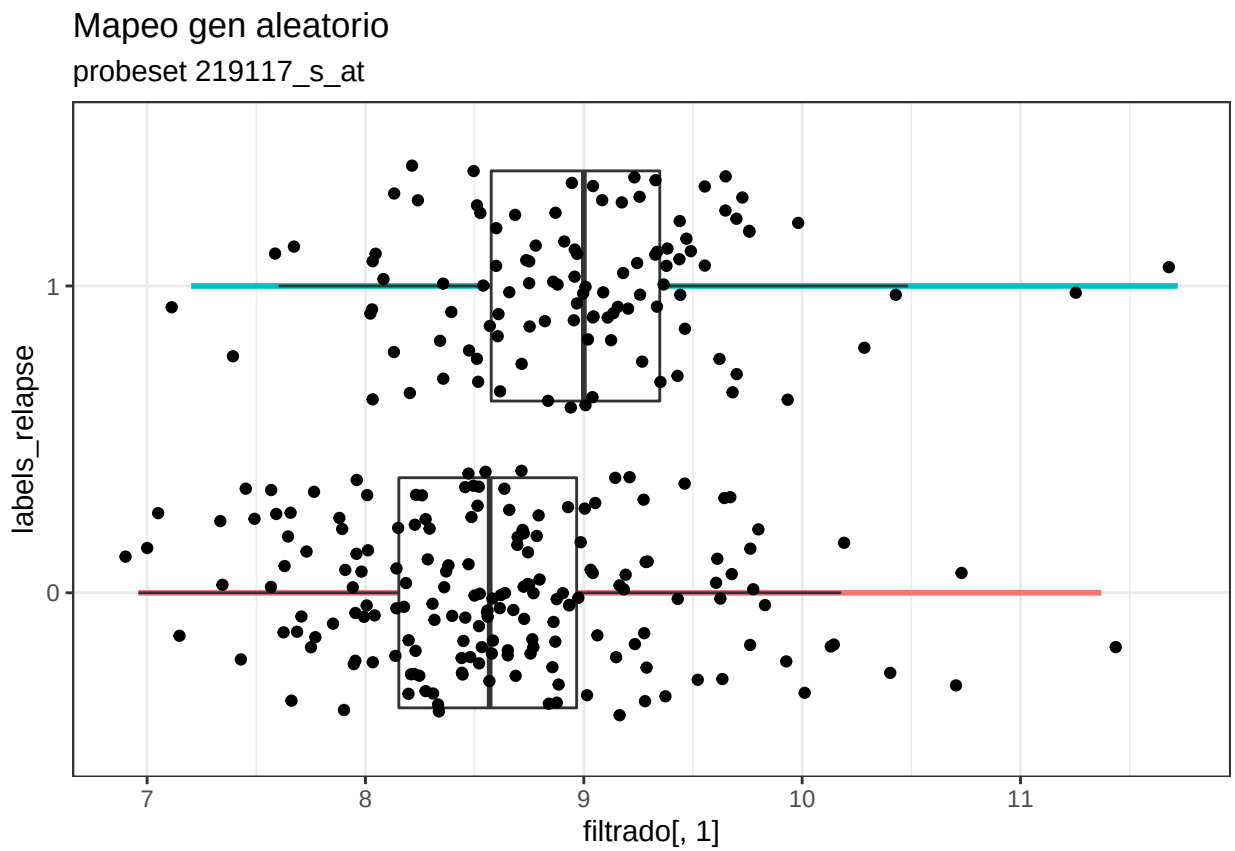
```
## Total positivos = 38
## Total clasificados positivos = 44
## Total negativos = 20
## Total clasificados negativos = 14
## Total = 58
exactitud <- (35+11)/58
cat("\nExactitud (Accuracy): \n", exactitud)
precision <- 11/14
cat("\nPrecision (Neg Pred Value): \n", precision)
sensibilidad <- 11/20
cat("\nSensibilidad (Specificity): \n", sensibilidad)
especifidad <- 35/38
cat("\nEspecifidad (Sensitivity): \n", especifidad)
tasa_error <- (3+9)/58
```

```
cat("\nTasa de error: \n", tasa_error)
tasa_fp <- 3/38
cat("\nTasa de falsos positivos: \n", tasa_fp)
tasa_fn <- 9/20
cat("\nTasa de falsos negativos: \n", tasa_fn)
```

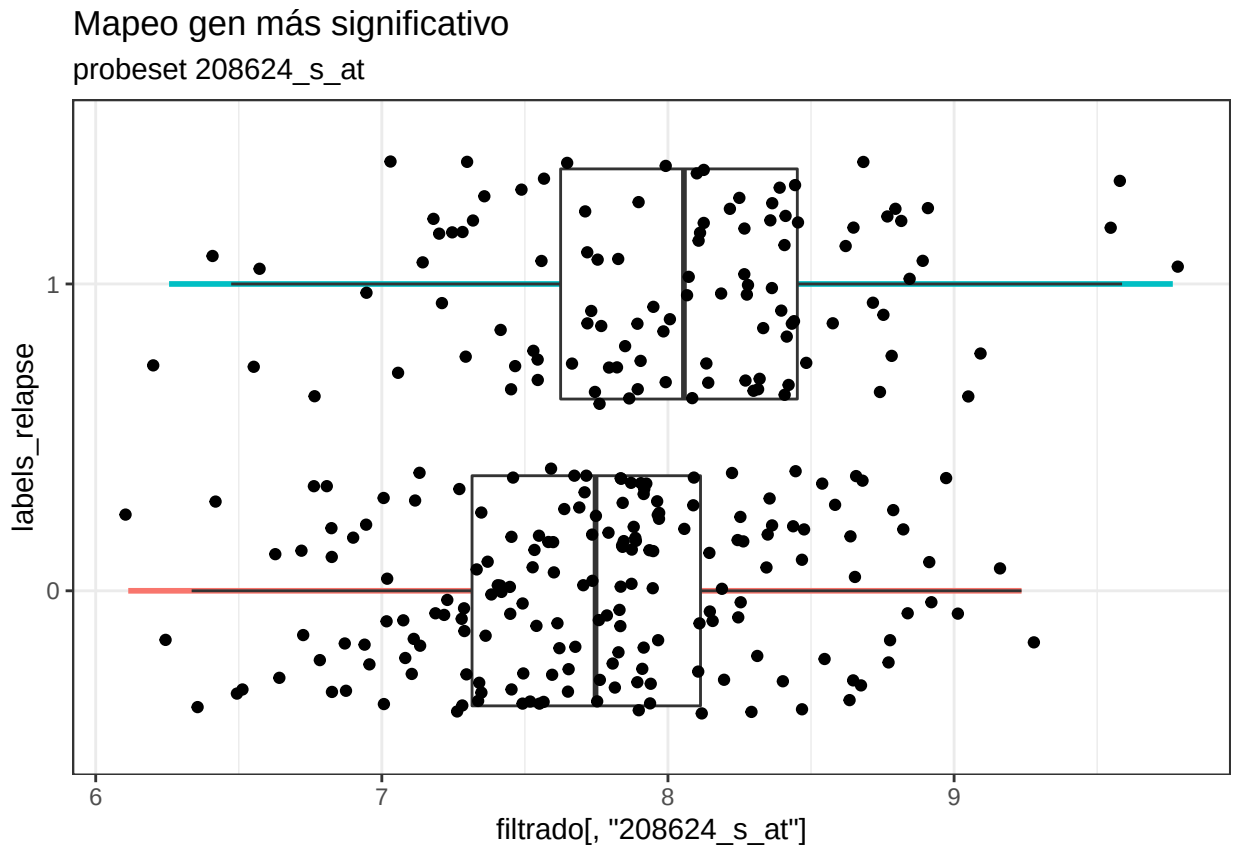
```
error_rl = NULL
for(i in 1:100){

  predicciones_rl<- predict(prl, newx = filtrado[-muestra, ], type = "class")

  rl_confusion <- table(predicciones_rl[, i],  labels_relapse[-muestra])
  rl_exactitud <- sum(diag(rl_confusion))/sum(rl_confusion)

  error_rl <- c(rl_exactitud, error_rl)
}
error_rl

ordenando <- order(error_rl)
ordenando
#El modelo 43 (modelo optimo) -> Cualquier modelo del 1 al 43 (son iguales)
#Error: 1.0000000
#Estoy viendo el acierto, no el error
#Acierto maximo: 1.0000000 -> Error minimo = 1 - 1.0000000 = 0

rl_confusion2 <- table(predicciones_rl[, 43],  labels_relapse[-muestra])
rl_exactitud2 <- sum(diag(rl_confusion2))/sum(rl_confusion2)

pander(rl_confusion2)
pander(rl_exactitud2)
```

```
#library(datasets)
coeficiente <- coef(prl, s=0.017080)
coeficiente
print(coeficiente[, 1] > 1)

row.names(coeficiente)[abs(coeficiente[,1])>0] #Con coeficiente + y -
row.names(coeficiente)[coeficiente[,1]>0] #Con coeficiente +
row.names(coeficiente)[coeficiente[,1]>1]
match("215778_x_at", row.names(coeficiente))
#más significativo: 215778_x_at  1.190855e+00 -> probeset [179]

#Ordenados de menor a mayor
order(coeficiente)

#Los 15 con mayor beta: 42 176 152 275 154 588 189  61 120  3 197  53  62  20 179
coeficiente[179, ]
print("\nProbeset 215778_x_at\n")
getSYMBOL("215778_x_at", "hgu133a")
coeficiente[20, ]
print("\nProbeset 207071_s_at\n")
getSYMBOL("207071_s_at", "hgu133a")
coeficiente[62, ]
```

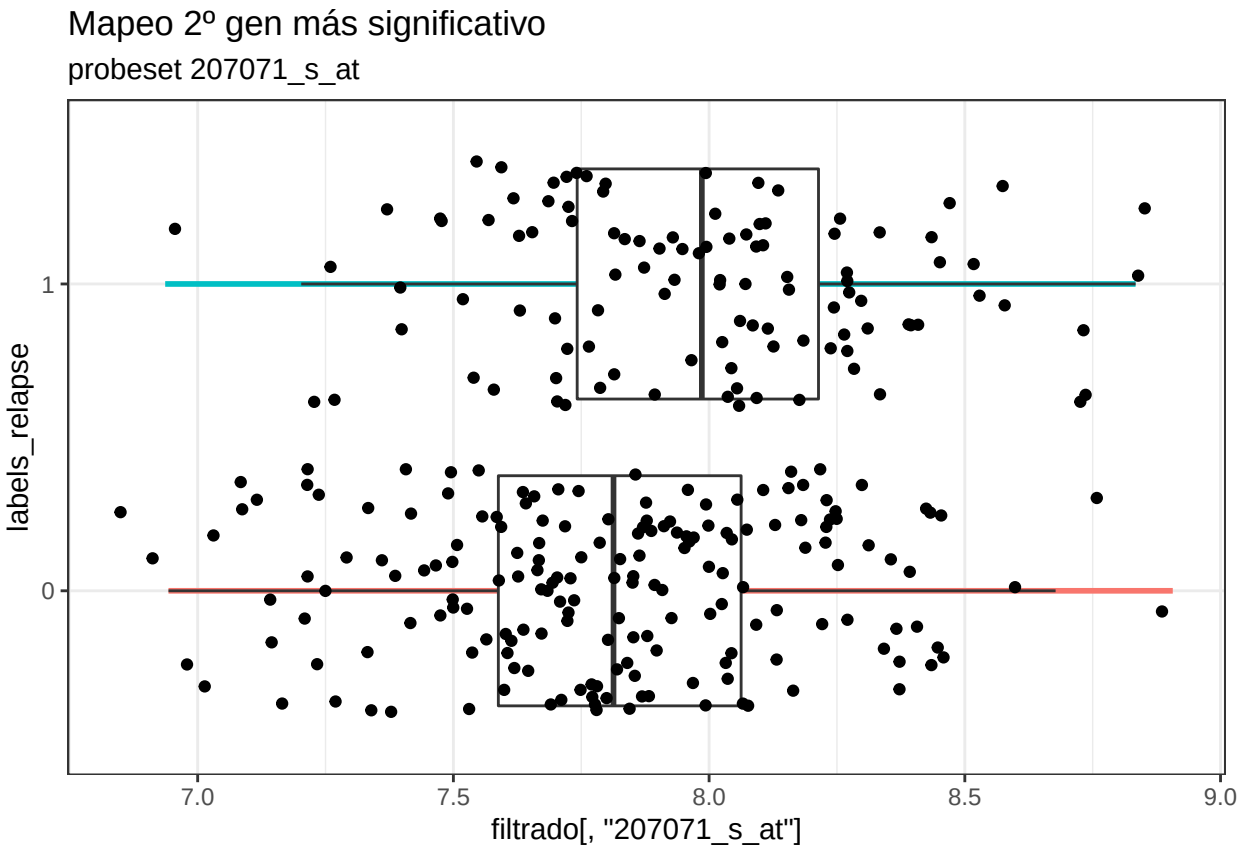
```
print("\nProbeset 218606_at \n")
getSYMBOL("218606_at", "hgu133a")
coeficiente[53, ]
print("\nProbeset 201186_at\n")
getSYMBOL("201186_at", "hgu133a")
coeficiente[197, ]
print("\nProbeset 200710_at\n")
getSYMBOL("200710_at", "hgu133a")
coeficiente[3, ]
print("\nProbeset 208624_s_at\n")
getSYMBOL("208624_s_at", "hgu133a")
coeficiente[120, ]
print("\nProbeset 203941_at\n")
getSYMBOL("203941_at", "hgu133a")
coeficiente[61, ]
print("\nProbeset 221979_at\n")
getSYMBOL("221979_at", "hgu133a")
coeficiente[189, ]
print("\nProbeset 78383_at\n")
getSYMBOL("78383_at", "hgu133a")
coeficiente[588, ]
print("\nProbeset 212765_at\n")
getSYMBOL("212765_at", "hgu133a")
coeficiente[154, ]
print("\nProbeset 209201_x_at\n")
getSYMBOL("209201_x_at", "hgu133a")
coeficiente[275, ]
print("\nProbeset 201773_at\n")
getSYMBOL("201773_at", "hgu133a")
coeficiente[152, ]
print("\nProbeset 212239_at\n")
getSYMBOL("212239_at", "hgu133a")
coeficiente[176, ]
print("\nProbeset 204193_at\n")
getSYMBOL("204193_at", "hgu133a")
coeficiente[42, ]
print("\nProbeset 204546_at\n")
getSYMBOL("204546_at", "hgu133a")

genes_relevantes <- c(42, 176, 152, 275, 154, 588, 189, 61, 120,  3, 197, 53, 62, 20, 179)
modelo2 <- glm(labels_relapse ~ ., data = datos[, genes_relevantes], family = "binomial")
summary(modelo2)

#Con un gen aleatorio (el gen con probeset "219117_s_at")
ggplot(data = datos, aes(x = filtrado[, 1], y = labels_relapse)) +
  ggtitle("Mapeo gen aleatorio", "probeset 219117_s_at") +
  geom_line(aes(colour = labels_relapse), size = 1) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```




```
#Repetir ggplot con el gen más significativo (buen marcador)
ggplot(data = datos, aes(x = filtrado[, "208624_s_at"], y = labels_relapse)) +
  ggtitle("Mapeo gen más significativo", "probeset 208624_s_at") +
  geom_line(aes(colour = labels_relapse), size = 1) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```




```
ggplot(data = datos, aes(x = filtrado[, "207071_s_at"], y = labels_relapse)) +
  ggtitle("Mapeo 2º gen más significativo", "probeset 207071_s_at") +
  geom_line(aes(colour = labels_relapse), size = 1) +
  geom_boxplot(outlier.shape = NA) +
  geom_jitter(width = 0.1) +
  theme_bw() +
  theme(legend.position = "null")
```


Mapeo 2º gen más significativo
probeset 207071_s_at
labels_relapse
1
0
7.0
7.5
8.0
8.5
9.0
filtrado[, "207071_s_at"]

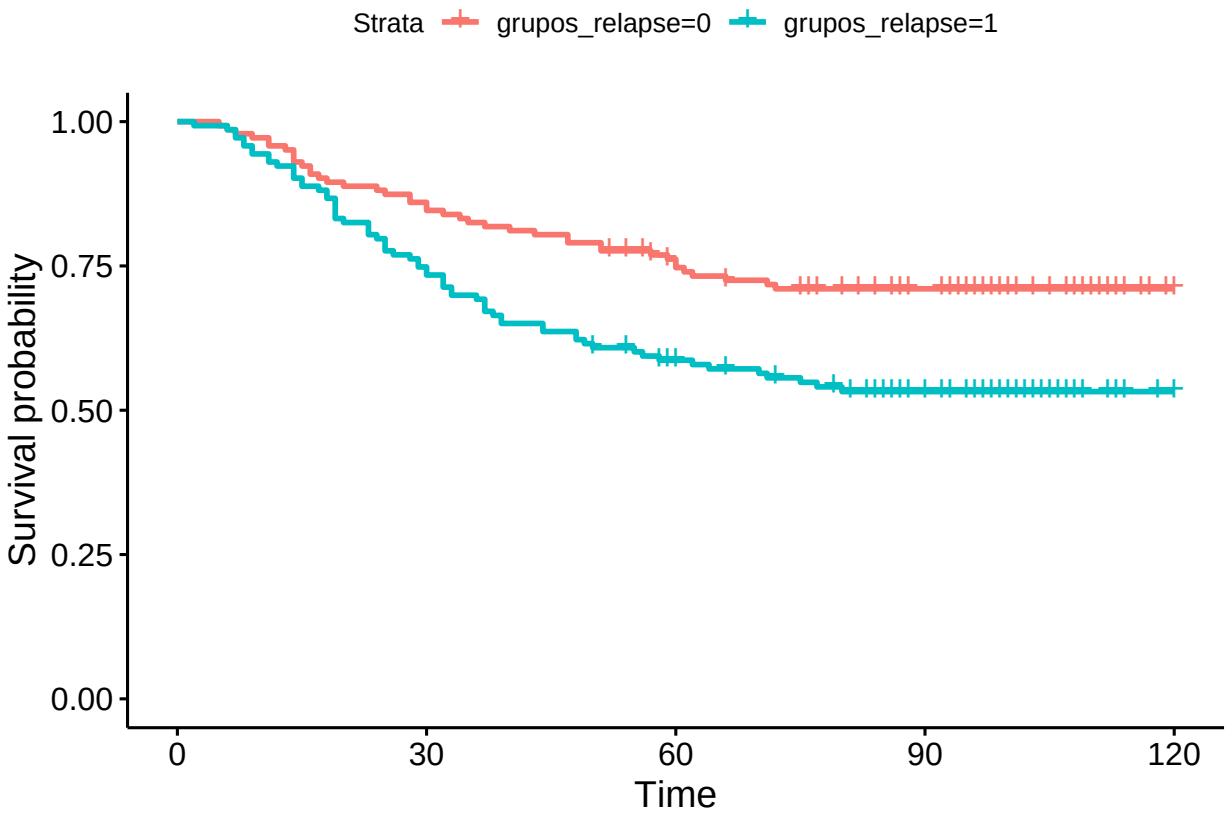
Estudio de supervivencia

```
#Para supervivencia, además de la columna 4, necesitaremos la columna 3
#La columna 3 undica el tiempo hasta la recaída o último seguimiento del paciente (meses)
time_relapse <- as.numeric( phenotipos[,3])
pander(table(time_relapse))
```

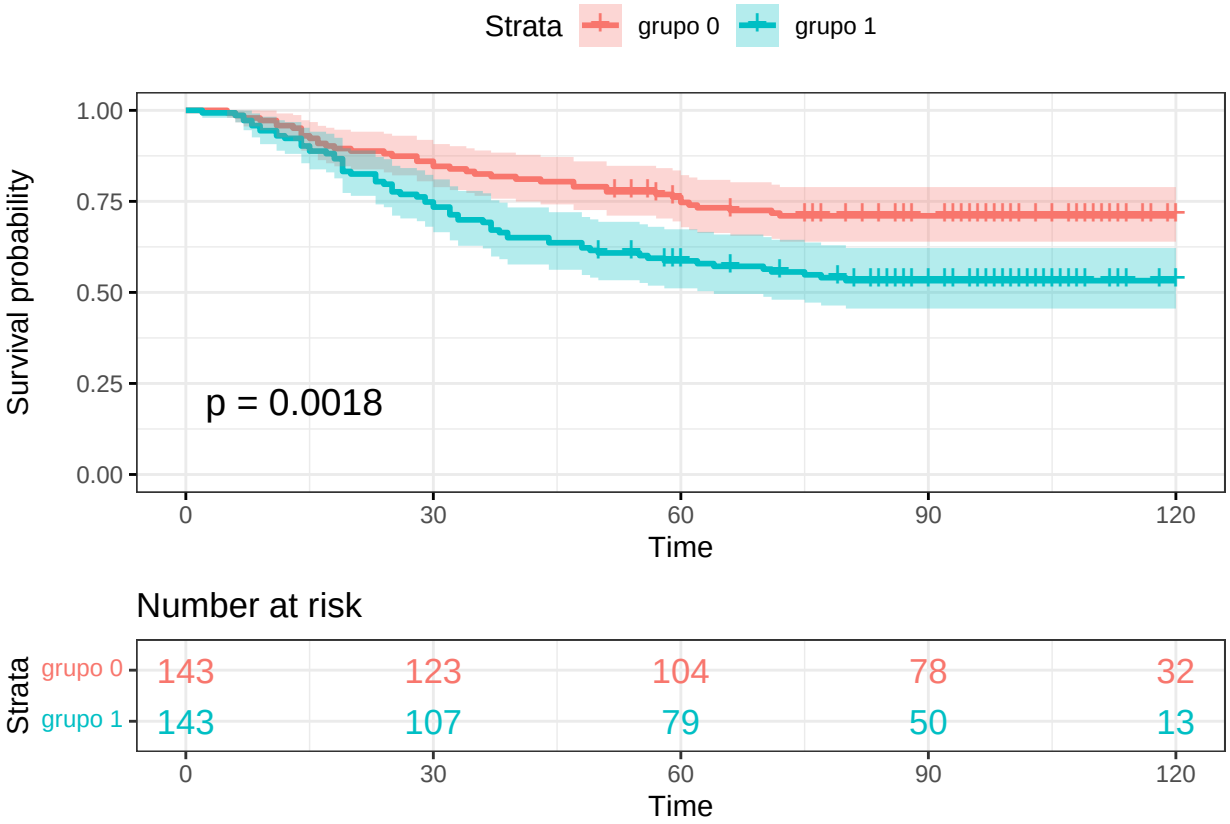
```
order(time_relapse)
#El tiempo máximo de estudio es de 171 meses (14 años y 3 meses)
#Cortaremos el estudio a los 10 años (120 meses)

labels_relapse[time_relapse > 120] = 0

time_relapse[time_relapse > 120] = 120

censurados <- rep(0, 286)
censurados[time_relapse < 120] = 1
pander(table(censurados))

#datos[, "X208624_s_at"]
median(datos[, "X208624_s_at"])

vec_grupo <- rep(1, 286)
vec_grupo[datos[, "X208624_s_at"] < 7.831116] = 0
```

```
#Añadir la columna grupo a la matriz phenotipos
phenotipos <- cbind(phenotipos, vec_grupo)

grupos_relapse <- as.factor( phenotipos[,7])
pander(table(grupos_relapse))
```

```
phenotipos_2 <- data.frame(time_relapse, labels_relapse, grupos_relapse)

write.table( file="phenotipos_2.txt", phenotipos_2)
phenotipos2 <- read.table("phenotipos_2.txt")

phenotipos2$grupos_relapse <- as.factor(phenotipos2$grupos_relapse)

phenotipos.k_m <- survfit(Surv(time_relapse, labels_relapse) ~ grupos_relapse,
                          data = phenotipos2)

ggsurvplot(phenotipos.k_m)
```




```
ggsurvplot(phenotipos.k_m, data = phenotipos2, size = 1, conf.int = TRUE, pval = TRUE,
           risk.table = TRUE, risk.table.col = "grupos_relapse",
           legend.labs = c("grupo 0", "grupo 1"), risk.table.height = 0.3, ggtheme = theme_bw ())
```


Strata
grupo 0
grupo 1
Survival probability
1.00
0.75
0.50
0.25
0.00
p = 0.0018
0
30
60
90
120
Time

Number at risk
Strata
grupo 0
grupo 1
143
123
104
78
32
143
107
79
50
13
0
30
60
90
120
Time