Predicción de la evolución clínica en pacientes HNSC a partir de datos multi-ómicos

Agustín Vera, Ángel Sayáns, Miriam Valdayo

2025-03-02

1. Introducción

El conjunto de datos proporcionado corresponde a un estudio realizado en 443 pacientes de Carcinoma Escamoso de Cabeza y Cuello (Head and Neck Squamous Cell Carcinoma, HNSC), incluyendo su información clínica y datos ómicos de las muestras correspondientes. Estas variables incluyen:

- **RNA-Seq:** perfil de expresión de 21520 genes, incluyendo una lista con 121 genes diferencialmente expresados entre los pacientes que fallecieron durante el estudio y los que no.
- **Factores de transcripción (FTs)**: perfil de expresión de 927 factores de transcripción.
- **Mutaciones:** matriz binaria de 106 genes estudiados.
- **Asociaciones:** lista de interacciones entre los FTs y sus genes diana recogidas en bases de datos, incluyendo sus códigos ENSEMBL.
- **Matriz de supervivencia:** evolución clínica de los pacientes, recogiendo el tiempo que el sujeto ha estado en el estudio hasta que ha ocurrido el evento de interés (TRUE, fallecimiento), o bien desde el inicio hasta el fin del estudio o pérdida de seguimiento (FALSE). Para este informe, **asumimos que todos los pacientes TRUE fallecieron durante el estudio (Exitus) y los pacientes FALSE se recuperaron (Alta)** para facilitar la realización del análisis posterior, aunque no se ajuste a la realidad del estudio.
- **Variables clínicas:** edad, género e historial de tabaquismo y alcoholismo de los pacientes, así como el estadío HNSC con el que entraron al estudio y su información histopatológica.

Para este informe, pretendemos generar un modelo que permita predecir la evolución clínica de los pacientes a partir de su firma ómica, entendiendo este *outcome* como una variable categórica binaria (Exitus/Alta). Para ello, se realizará un análisis integrativo múlti-ómico (RNA-Seq, FTs y mutaciones) supervisado (MultiBlock-sPLS-DA) y así identificar variables correlacionadas que expliquen el *outcome* de interés con las que construir el modelo predictor.

Adicionalmente, queremos explorar la regulación de los genes expresados diferencialmente en los pacientes Exitus, y si este *outcome* puede afectar a la regulación. Para esto, emplearemos el paquete MORE (Multi-Omics REgulation).

2. Exploración inicial de los datos por PCA

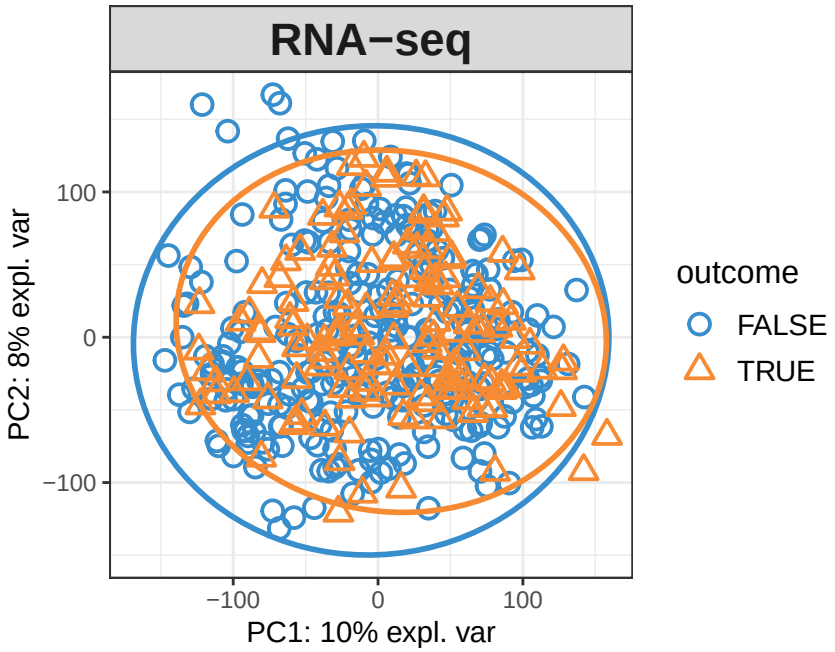
Antes de integrar los datos ómicos y clínicos en un modelo predictor, realizamos un análisis de componentes principales (PCA) de las ómicas cuantitativas (expresión de genes y de FTs) por separado para comprobar si las observaciones se agrupan en función del outcome sin supervisar el análisis.

```r
# Carga de datos y paquetes a usar
load("data.RData"); source(file ="all_sparse_functions.R");library(mixOmics);
library(dplyr);library(MORE);library(gridExtra);library(biomaRt);
library(clusterProfiler);library(org.Hs.eg.db);library(enrichplot);
library(FDRsampsize);library(lpSolve);library(MultiPower)
```

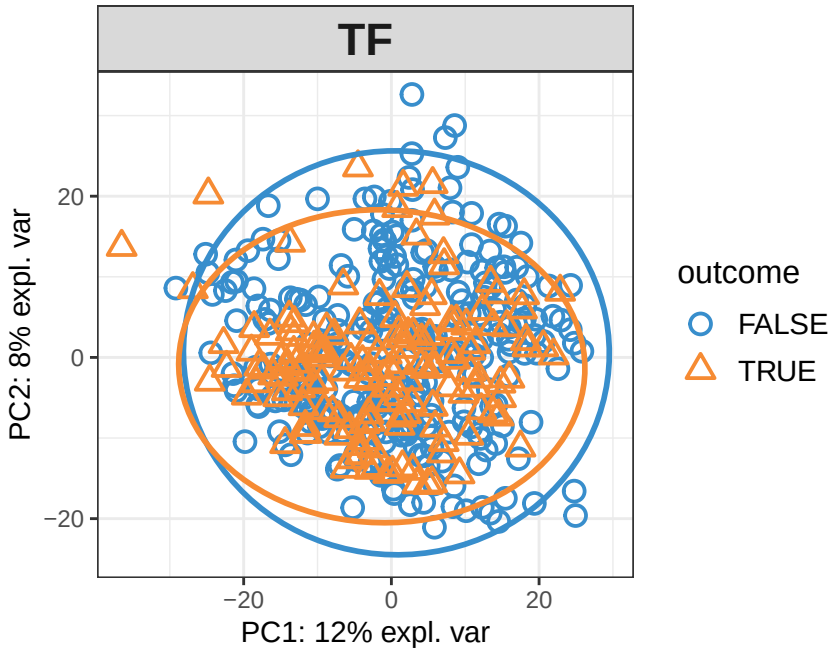
```r
# Guardamos los genes incluidos en cada matriz como vectores
genes.list <- colnames(genes.rna); degs.list <- degs$Genes;
 tf.list <- colnames(genes.tf); mutations.list <- colnames(genes.mutation);

# Centramos las matrices de expresión
gene.expr = scale(genes.rna, center = TRUE, scale = FALSE)
tf.expr= scale(genes.tf, center = TRUE, scale = FALSE)
```

Previamente a la realización del PCA final, hemos ejecutado un PCA con un número elevado de componentes (disponible en el **Anexo I**). En base a esos resultados, elegimos 7 componentes para RNA-Seq y 6 para los FTs como el número óptimo de componentes informativas. Aún así, estas componentes representan un porcentaje relativamente bajo de la variabilidad total de los datos, en especial de los 21520 genes recogidos en RNA-Seq.

```r
# RNA-Seq
mypca_final = mixOmics::pca(gene.expr, ncomp = 7, center = FALSE, scale = FALSE)

plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,group = as.factor(
  outcome$event),style = "ggplot2",legend = TRUE, legend.position = "right",
  legend.title = "outcome", ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE,
  title ="RNA-seq")
```



```r
# TF
pca_tf_final = mixOmics::pca(tf.expr, ncomp = 6, center = FALSE, scale = FALSE)
plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL,group = as.factor(
  outcome$event),style = "ggplot2",legend = TRUE, legend.position = "right",
  legend.title = "outcome", ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE,
  title = "TF")
```



Como podemos observar, no es posible discernir el outcome de los pacientes en base únicamente al perfil de expresión de las ómicas individuales, ni a partir de las dos primeras PC o las PC sucesivas (no representadas). Adicionalmente, también se han etiquetado las observaciones en el PCA en base a las otras variables clínicas en el **Anexo I.** Más allá de pequeñas variaciones en la PC1 que permiten intuir diferencias según el género y el grado histológico en ambas ómicas, tampoco es posible agrupar pacientes según su firma ómica individual. Esto puede deberse a que las variables clínicas no son capaces de explicar la variabilidad de los datos transcriptómicos, o a que la variación en los mismos datos no es suficiente para diferenciar los grupos.

Así, comprobamos que resulta muy difícil segregar los pacientes Exitus/Alta mediante análisis no supervisado de las ómicas individuales, por lo que se procedemos a la integración multiómica supervisada.

3. Análisis de Potencia Estadítica (MultiPower)

Previo a la integración de las ómicas, se ha realizado un análisis de potencia estadística para comprobar si los datos realizar un análisis de integración con la suficiente potencia.

```r
# Matriz de datos
statdata = statdesign = vector("list")
statdata$mrna <- as.data.frame(t(genes.rna))
statdata$TF<-as.data.frame(t(genes.tf))
statdata$mutations<-as.data.frame(t(genes.mutation))
statdata$mrna = statdata$mrna - min(statdata$mrna)
statdata$TF = statdata$TF - min(statdata$TF)
```

```
statdata$mutations = statdata$mutations - min(statdata$mutations)

# Matriz de diseño
conditions <- ifelse(outcome$event, "Exitus", "Alta")
names(conditions) <- rownames(outcome); statdesign <- list(
  mrna = conditions,TF = conditions, mutations = conditions)

# Multipower
statResultsEQ = MultiPower(
  data = statdata, groups = statdesign, type = type1, omicPower = 0.6,
  averagePower = 0.8, fdr = 0.05, equalSize = TRUE,max.size = 10000,
  omicCol = miscolores)
```

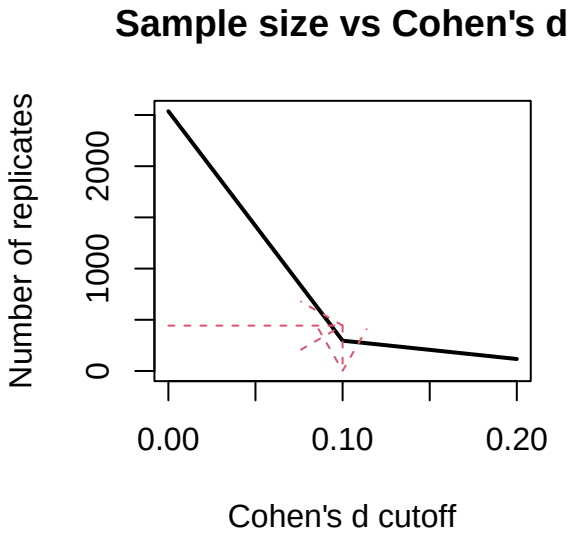
```
load("statResultsEQ.Rdata");
statResultsEQ$summary[c(2,3,5,9,10,11)]
```

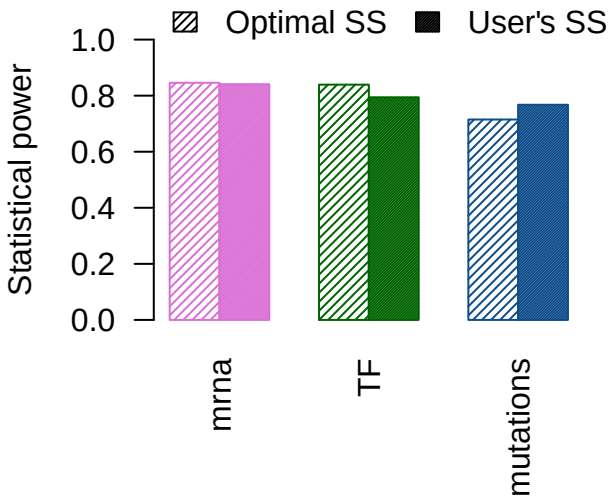
```
##           type numFeat maxCohenD minSampleSize optSampleSize  power
## mrna         2   21520      0.63           697          2537 0.8459
## TF           2     927      0.40           762          2537 0.8392
## mutations    3     106      0.28          1532          2537 0.7149
```

Como vemos, el factor limitante de este estudio es la baja potencia estadística de las 106 mutaciones contempladas. Para obtener una potencia estadística global media de 0.8 en un estudio multiómico, necesitaríamos un tamaño de muestra de 2537, lejos de las 443 observaciones con las que contamos. Por lo tanto, se procederá a realizar un análisis de integración con los datos disponibles, pero teniendo en cuenta que los resultados obtenidos podrían tener un alto número de falsos negativos.

Si quisieramos filtrar nuestros datos para obtener la potencia estadística deseada, tendríamos que filtrar features que tengan una d de Cohen inferior a 0.1, en base a los resultados postMultiPower indicando que nuestro número máximo de muestras es 443. Sin embargo, dado nuestro objetivo es realizar un modelo predictivo con el máximo número de variables y que las herramientas sparsePLS ya seleccionan las *features* de cada ómica que explican mayor variabilidad, procedemos con todas las variables sin filtrar.

```
type1 = c(2,2,3); miscolores = c("orchid", "darkgreen", "dodgerblue4")
load("statdata.RData")
names(miscolores) = names(statdata)
STATpostEQ = postMultiPower(optResults = statResultsEQ, max.size = 443,
                            omicCol = miscolores)
```





4. Integración multiómica para construir un modelo predictor (DIABLO)

Para tratar de predecir la evolución clínica de los pacientes a partir de la integración de datos ómicos y clínicos, realizamos un análisis MultiBlock sparsePLS-DA o DIABLO (*Data Integration Analysis for Biomarker Discovery using Latent Variable Approaches for Omics Studies*).

4.1. Preparación de los datos

En primer lugar, seleccionamos aleatoriamente a 43 de los 443 pacientes para formar un grupo *test*. Estos datos no se emplearán para generar el modelo, si no que se emplearan más adelante para validar la capacidad predictora del modelo y comprobar que no está sobre-ajustado para predecir solo los datos con los que se ha entrenado.

Los datos de los 400 pacientes restantes se emplearán para entrenar el modelo.

```
# Dividimos los pacientes en un grupo training (400) y en uno de test (43)
patients <- rownames(outcome)
set.seed(1)
test.ids <- sample(patients, 43, replace = FALSE )
outcome.test <- outcome[test.ids,]
train.ids <- setdiff(patients, test.ids)
outcome.train <- outcome[train.ids,]

# Los transformamos en factores
outcome.test.factor <- factor(factor(outcome.test[,2]), levels = c(TRUE, FALSE),
                              labels = c("Exitus", "Alta"))
outcome.train.factor <- factor(factor(outcome.train[,2]), levels = c(TRUE, FALSE),
                               labels = c("Exitus", "Alta"))
```

```r
# Valoramos si los pacientes de cada grupo son proporcionales
summary(outcome.test.factor)[1]/summary(outcome.test.factor)[2]
```

```
##    Exitus
## 0.4827586
```

```r
summary(outcome.train.factor)[1]/summary(outcome.train.factor)[2]
```

```
##    Exitus
## 0.5267176
```

El grupo *test* es bastante similar al de *training*, donde aproximadamente se da un Exitus por cada dos Altas.

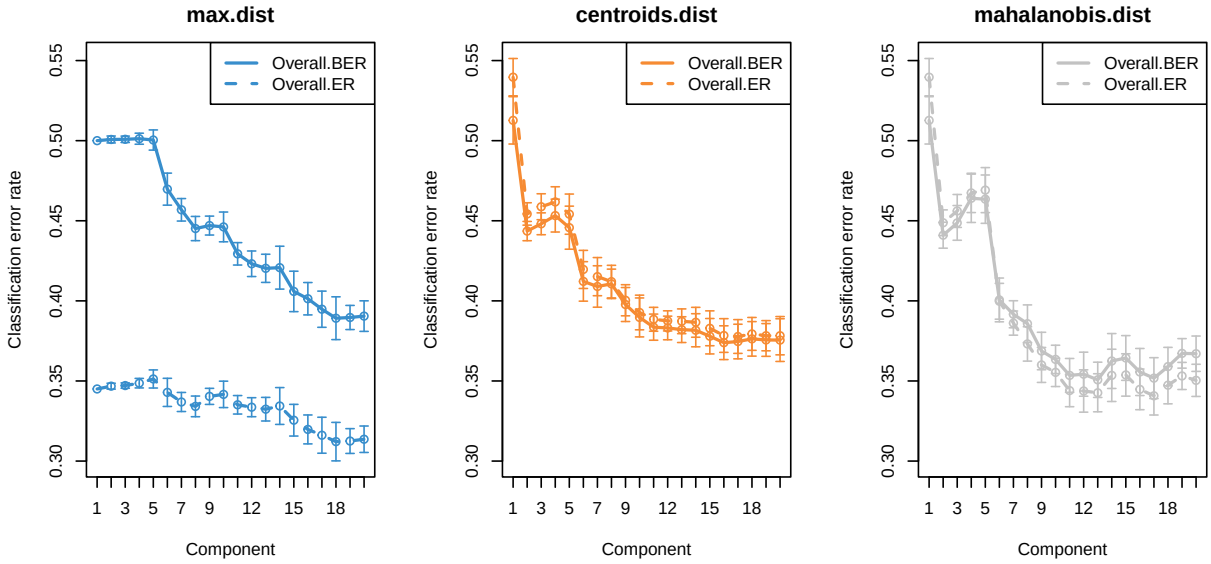
```r
# Creamos listas con los datos de cada grupo
data.test <- list(outcome = outcome.test.factor, genes.rna = genes.rna[test.ids,],
                  genes.tf = genes.tf[test.ids,],
                  genes.mutation= genes.mutation[test.ids,])

data.train <- list(outcome = outcome.train.factor, genes.rna = genes.rna[train.ids,],
                   genes.tf = genes.tf[train.ids,],
                   genes.mutation = genes.mutation[train.ids,])

# Centramos los datos ómicos antes de agruparlos
data.train$genes.rna <- scale(data.train$genes.rna, center = T, scale = F)
data.train$genes.tf <- scale(data.train$genes.tf, center = T, scale = F)
data.train$genes.mutation <- scale(data.train$genes.mutation, center = T, scale = F)
data.X <- list(genes.rna = data.train$genes.rna, genes.tf = data.train$genes.tf,
               genes.mutation = data.train$genes.mutation)

# Diseño. Consideramos el valor máximo de relación entre cada bloque (1)
design = matrix(1, ncol = length(data.X), nrow = length(data.X),
                dimnames = list(names(data.X), names(data.X)))
diag(design) = 0
```

4.2. Optimización de parámetros

Realizamos un DIABLO inicial con 20 componentes para seleccionar el número óptimo.

```r
set.seed(1)
pre.diablo <- mixOmics::block.plsda(X =data.X, Y = data.train$outcome, ncomp=20,
                                    design = design,scale = FALSE)

# Evaluamos performance por cross-validation
perf.diablo <- mixOmics::perf(pre.diablo, validation = 'Mfold',
                              folds = 5, nrepeat = 10, progressbar = TRUE)
```

```r
load("perf_diablo.RData")
plot(perf.diablo, overlay = 'measure', sd=TRUE)
```



En base a lo observado, elegimos 11 componentes como compromiso entre el óptimo observado en el error de la distancia por centroides y mahalanobis. A continuación optimizamos el número de variables de cada bloque que seleccionar para cada componente.

```
# 6 números de variables posibles por bloque, incluyendo el número máximo de
# features por bloque
list.keepX <- list(genes.rna = c(1,10,100,1000,10000,21520),
                   genes.tf = c(1,2,5,10,100,927),
                   genes.mutation = c(1,2,5,10,50,106))

BPPARAM <- BiocParallel::SnowParam(workers = parallel::detectCores()-1)
tune.diablo <- tune.block.splsda(X = data.X, Y = data.train$outcome, design = design,
                                 ncomp = 11, validation = 'Mfold', nrepeat = 5,
                                 folds = 5, dist = 'max.dist', test.keepX =list.keepX,
                                 scale = FALSE, BPPARAM = BPPARAM)
```

```
load("tune.diablo.RData")
num.var.diablo <- tune.diablo$choice.keepX
num.var.diablo
```

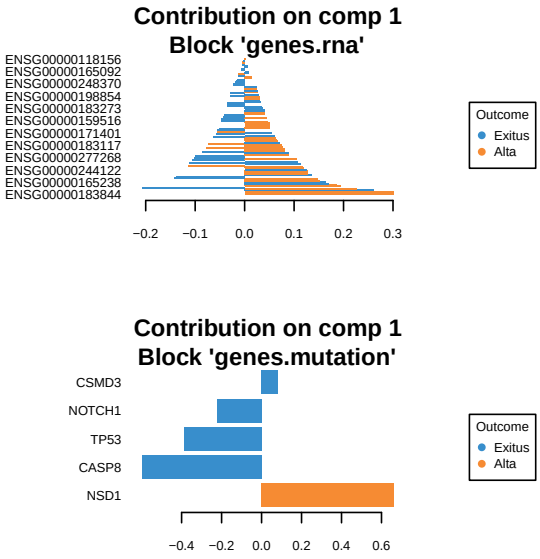
```
## $genes.rna
##  [1]   100  1000 21520 10000 10000 21520     1     1 10000 10000 10000
##
## $genes.tf
##  [1] 100 927 927   1   1   1 927 927 100   1   2
##
## $genes.mutation
##  [1]   5   1   1   1  50   1   1  10 106   1  50
```

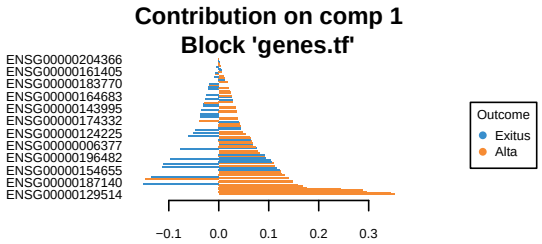
4.3. Generación del modelo final con los datos de entrenamiento

Con este número de variables óptimas seleccionadas por bloque para cada una de las 11 componentes, realizamos el MultiBlock sPLS-DA final.

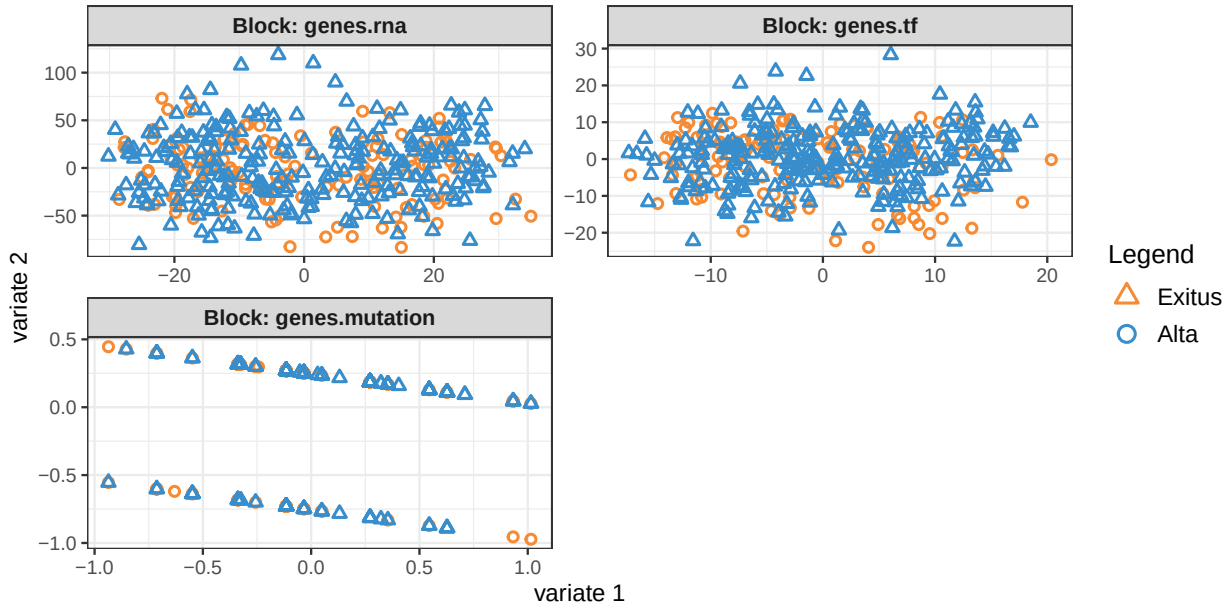
```
diablo <- block.splsda(X =data.X, Y = data.train$outcome, design = design,
                       ncomp=11, keepX = num.var.diablo, scale = FALSE)
```

```
load("diablo.RData")
plotLoadings(diablo, comp = 1, contrib = 'max')
```






```
plotIndiv(diablo, ind.names = FALSE, legend = TRUE, cex = 1.5, size.subtitle = 10)
```



El modelo óptimo generado ha elegido la expresión de 100 genes normales, 100 FTs y 5 mutaciones como las *features* que explican la mayor cantidad de variabilidad en la PC1 para predecir el outcome de los pacientes. No se ha seleccionado ningún gen como gen RNA-Seq y como FT para los dos bloques simultáneamente. Por otro lado, al representar los *score* de las observaciones según las dos primeras componentes, no parecen agruparse claramente según los grupos a predecir.

4.4. Capacidad predictora del modelo

Por último, evaluamos la capacidad robustez predictora del modelo tratando de predecir el *outcome* de los pacientes del grupo *test* a partir de sus datos ómicos.

```
# Prediccion de datos test
testPrediction = predict(object = diablo, newdata = list(genes.rna = data.test$genes.rna,
                                                  genes.tf = data.test$genes.tf,
                                                  genes.mutation = data.test$genes.mutation))

table(data.frame("prediction" = testPrediction$AveragedPredict.class$max.dist[,1],
                 "trueType" = data.test$outcome), useNA = "ifany")
```

```
##           trueType
## prediction Exitus Alta
##       Alta     14   29
```

```
table(data.frame("prediction" = testPrediction$AveragedPredict.class$max.dist[,5],
                 "trueType" = data.test$outcome), useNA = "ifany")
```

```
##           trueType
## prediction Exitus Alta
##     Alta        6   11
##     Exitus      8   18
```

Al tratar de predecir el outcome de los pacientes con la primera componente del modelo, todas las observaciones con designadas como Alta. Parece que este modelo minimiza los pacientes mal clasificados a costa de predecir pésimamente el pronóstico menos frecuente Exitus. Contemplando la componente 5 se predicen algo mejor los casos Exitus, pero un gran número de pacientes quedan mal clasificados.

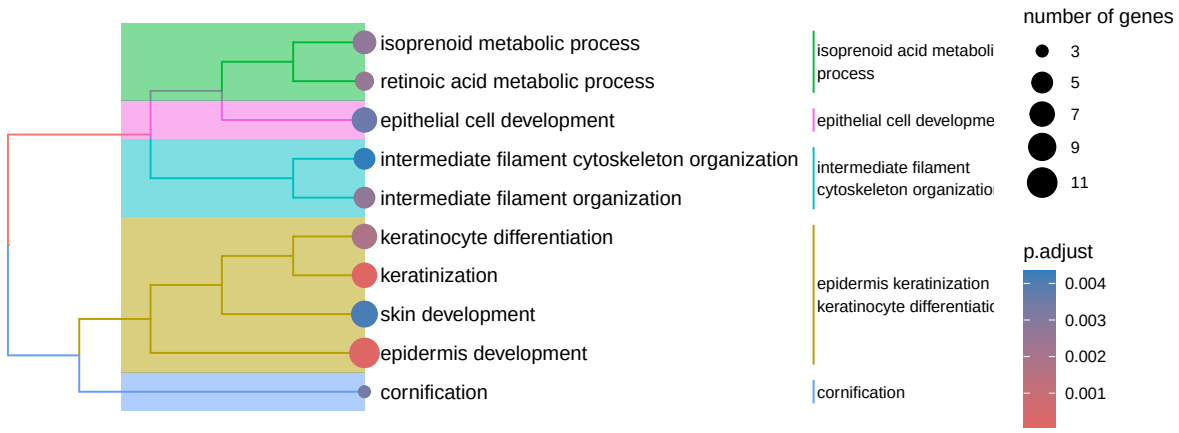
5. Interpretación del modelo de integración

En conjunto, parece que el modelo generado a partir de los datos ómicos de las observaciones de 400 pacientes no permite predecir si su enfermedad termina en Exitus o Alta. Esto puede deberse a la dificultad de predecir un pronóstico futuro a partir de una firma transcripcional ya de por si altamente variable entre pacientes; o posiblemente se deba a la imprecisión de la asunción realizada sobre el outcome "Alta" de los pacientes: el no haber registrado el fallecimiento de un paciente durante el transcurso del estudio no significa que ese paciente no pudiera haber fallecido más adelante.

A pesar de la limitada capacidad predictora del modelo, resulta interesante estudiar el papel que pueden tener las *features* seleccionadas como marcadores clave en las primeras componentes. Para ello, realizamos un análisis enriquecimiento funcional en términos de Ontología Génica para determinar los principales procesos biológicos relacionados con los 100 genes y 100 FTs seleccionados para la PC1.

```
comp1.rna <- diablo$loadings$genes.rna[,1]
comp1.rna.genes <- names(comp1.rna[comp1.rna != 0])
pc1.rna.enrichgo <- enrichGO(gene = comp1.rna.genes, OrgDb= org.Hs.eg.db, ont="BP",
                             pAdjustMethod="BH",pvalueCutoff=0.05,
                             readable = FALSE, keyType = "ENSEMBL")
```

```
load("PC1.rna.RData")
treeplot(pairwise_termsim(pc1.rna.enrichgo),showCategory = 10, fontsize= 3, cex = 0.9)
```



Destacamos los genes relacionados con el desarrollo epidérmico y con la diferenciación queratinocítica. Los queratinocitos han sido identificados como un tipo celular clave en el desarrollo tumoral de HNSC por estudios transcriptómicos (1), por lo que es coherente que los genes implicados en su regulación estén enriquecidos como marcadores predictores.

```
comp1.tf <- diablo$loadings$genes.tf[,1]
comp1.tf.genes <- names(comp1.tf[comp1.tf != 0])
pc1.tf.enrichgo <- enrichGO(gene = comp1.tf.genes, OrgDb= org.Hs.eg.db, ont="BP",
                            pAdjustMethod="BH",pvalueCutoff=0.05,
                            readable = FALSE, keyType = "ENSEMBL")
```

```
load("PC1.tf.RData")
treeplot(pairwise_termsim(pc1.tf.enrichgo),showCategory = 10, fontsize= 3, cex = 0.9)
```

En el caso de los procesos enriquecidos en los factores de transcripción seleccionados, destacamos rutas relacionadas con la proliferación celular durante desarrollo embrionario y regulación *stemness*. Las CSC (*Cancer stem cells*) han sido identificadas como una subpoblación muy plástica y resistente en los tumores HNSC que promueve y mantiene la tumorogénesis y metástasis (2), por lo que su regulación por FTs globales puede suponer un importante *driver* de progresión tumoral y un potencial factor pronóstico.

Por otra parte,la primera componente del modelo también contempla 5 mutaciones clave para predecir el *outcome* de los pacientes. Entre estas, destacamos NSD1, una histona metiltransferasa que regula el crecimiento celular y la autofagia en HNSC, y cuya mutación ha sido identificada por nuestro modelo como un factor de buen pronóstico. Notoriamente, Topchu et al. (2024) (3) también reportan "mutations inactivating NSD1 have been linked to improved outcomes in HNSCC". Además, muchas de las otras mutaciones seleccionadas están también comúnmente asociadas al HNSC (y especialmente a su subtipo oral), como la pérdida de función de CASP8, NOTCH1 o TP53 (4).

6. Regulación de DEGs: análisis complementario MORE

Adicionalmente, hemos explorado cómo se relacionan los reguladores con los genes diferencialmente expresados (*DEGs*) del *outcome*. Para ello hemos realizado dos modelos MORE, uno donde tenemos en cuenta las asociaciones previas establecidas entre los FT y los DEGs (PLS1) y otro sin las asociaciones previas para estudiar posibles relaciones no establecidas previamente (PLS2). Además, en ambos podemos explorar la relación que se establece entre las variables clínicas y los DEGs.

Por limitaciones de espacio cargaremos los datos para representar los resultados directamente. El código para la preparación de los datos y la generación de los modelos se puede consultar en el **Anexo II**.

6.1 Resultados de MORE

Aplicando la función "RegulationPerCondition" al modelo PLS1 obtenemos un resumen de los reguladores que afectan a cada gen, así como los coeficientes de regresión que relacionan el gen con el regulador para cada condición experimental. En él se nos indica si actúan en ambos grupos, uno o ninguno, así como la naturaleza de estos, es decir, cuánto ponderan en cada grupo (a mayor valor absoluto mayor peso) y la naturaleza represora o activadora de estos. Valores negativos y positivos respectivamente.

```
regpcond = RegulationPerCondition(moreResults)
```

```
##   |                                                                      |
```

```
length(unique(regpcond$targetF))
```

```
## [1] 73
```

```
length(unique(regpcond$regulator))
```

```
## [1] 22
```

```
head(regpcond)
```

```
##           targetF                          regulator   omic area  Group_1
## 1 ENSG00000127241      lymphnode_neck_dissectionYes clinic      -0.01241
## 2 ENSG00000224660                        genderMale clinic      -0.02787
## 3 ENSG00000224660      lymphnode_neck_dissectionYes clinic      -0.02994
## 4 ENSG00000066923                        genderMale clinic       0.03517
## 5 ENSG00000066923      lymphnode_neck_dissectionYes clinic      -0.04880
## 6 ENSG00000066923 neoplasm_histologic_gradeStage_II clinic      -0.03617
##    Group_0
```

```
## 1  0.00000
## 2  0.00000
## 3  0.00000
## 4  0.06555
## 5 -0.07176
## 6 -0.06099
```

En este caso, hallamos 19 reguladores significativos (tanto clínicos como FTs) de expresión génica para 117 genes. La mayoría de estos reguladores son variables clínicas.

Por otra parte, no nos ha sido posible ejecutar esta función en los resultados generados con PLS2. Parece que existen valores nulos en el objeto *moreResults2* que impiden que se ejecute correctamente. Las únicas diferencias respecto al objeto *moreResults* del PLS1 son la lista de asociaciones dentro de arguments (vacío por el propio diseño del PLS2) y que 10 de los 121 genes en ResultsPerTargetF carecen de reguladores significativos y sus coeficientes están vacíos. No obstante, ni incluyendo valores "None" artificialmente ni filtrando estos genes de la lista se ha conseguido ejecutar esta función con éxito.

A continuación evaluamos la calidad del ajuste de cada modelo.

```
# PLS1
mean(moreResults$GlobalSummary$GoodnessOfFit[,1])
```

```
## [1] 0.08306575
```

```
# PLS2
mean(moreResults2$GlobalSummary$GoodnessOfFit[,1])
```

```
## [1] 0.608
```

Como podemos ver, PLS2 ofrece un mejor valor R2 y un mejor ajuste.

A continuación estudiamos los reguladores globales de cada modelo.

```
# PLS1
moreResults$GlobalSummary$GlobalRegulators
```

```
## [1] "genderMale"                          "lymphnode_neck_dissectionYes"
## [3] "neoplasm_histologic_gradeStage_IV"   "alcohol_history_documentedYes"
## [5] "neoplasm_histologic_gradeStage_III" "neoplasm_histologic_gradeStage_II"
```

```
# PLS2
moreResults2$GlobalSummary$GlobalRegulators
```

```
##  [1] "TF-ENSG00000105717" "TF-ENSG00000110777" "TF-ENSG00000137265"
##  [4] "TF-ENSG00000140968" "TF-ENSG00000156127" "TF-ENSG00000161405"
##  [7] "TF-ENSG00000171163" "TF-ENSG00000180535" "TF-ENSG00000184677"
## [10] "TF-ENSG00000185155" "TF-ENSG00000185811" "TF-ENSG00000196419"
## [13] "TF-ENSG00000213347" "TF-ENSG00000213928"
```

Resulta llamtivo que cada uno de los modelos toma como reguladores globales un tipo de dato diferente: PLS1 toma variables clínicas y PLS2 FTs.

Para explorar este fenómeno, comparamos los reguladores significativos de algunos genes según el modelo PLS1 o PLS2.

```
# PLS1
head(moreResults$ResultsPerTargetF$ENSG00000127241$significantRegulators)
```

```
## [1] "lymphnode_neck_dissectionYes"
```

```
head(moreResults$ResultsPerTargetF$ENSG00000224660$significantRegulators)
```

```
## [1] "genderMale"                    "lymphnode_neck_dissectionYes"
```

```
# PLS2
head(moreResults2$ResultsPerTargetF$ENSG00000127241$significantRegulators)
```

```
## [1] "TF-ENSG00000129152" "TF-ENSG00000185155" "TF-ENSG00000144802"
## [4] "TF-ENSG00000137265" "TF-ENSG00000156127" "TF-ENSG00000185811"
```

```
head(moreResults2$ResultsPerTargetF$ENSG00000224660$significantRegulators)
```

```
## [1] "TF-ENSG00000171163" "TF-ENSG00000213928"
```

Una vez más, observamos que el modelo PLS2 da mucha importancia los FTs como reguladores significativos frente a las variables clínicas cualitativas de PLS1.

En conjunto, PLS2 parece ser un modelo con mejor ajuste a los datos donde sería interesante estudiar en mayor detalle la correlación entre la expresión de algunos genes de interés y los FTs que los regulan. No obstante, nuestra capacidad de análisis de los resultados de este modelo está limitada, dado que solo nos ha sido posible extraer la matriz "RegulationPerCondition" para el modelo PLS1

6.2 Resultados globales de MORE

Red Cytoscape

Generamos una red de interacciones del modelo PLS1 y la visualizamos mediante el *softaware* Cytoscape, diferenciando los reguladores según el outcome (Figura 1). En ambos grupos podemos ver cómo dos genes que no se integran con el resto de la red, justo aquellos que están regulados por FTs. Por otro lado se puede observar como la categoría clínica de género masculino regula exclusivamente a varios genes (positivamente en el grupo Alta y negativamente en Exitus). Genes como el ENSG00000180411 se encuentran en esta situación, indicando que esta variable clínica puede tener un efecto significativo en la regulación de los DEGs. La variable "lymphnode_neck_dissection" también actúa como regulador central de varios genes, aunque está más integrado con el resto de *features* reguladoras.

Over Representation Analysis (ORA)

Se ha realizado un análisis funcional de enriquecimiento de genes hub a partir del modelo PLS1. En éste análisis hemos obtenido una tabla donde aparecen 33 términos GO. Entre estos destacamos términos de unión a DNA y RNA, así como motivos que controlan la frecuencia, activación y cese de transcripción génica (como los genes reguladores de marcas epigéneticas), por lo que pueden tratarse de genes directamente implicados en la regulación de múltiples procesos celulares y con un gran impacto en la progresión tumoral. También se encuentran varios términos relacionados con la división celular. Tanto la reprogramación epigenética y transcripcional como la desregulación de la división celular son reconocidos *hallmarks* del cáncer (5).

Los resultados completos están disponibles en el **Anexo II**.

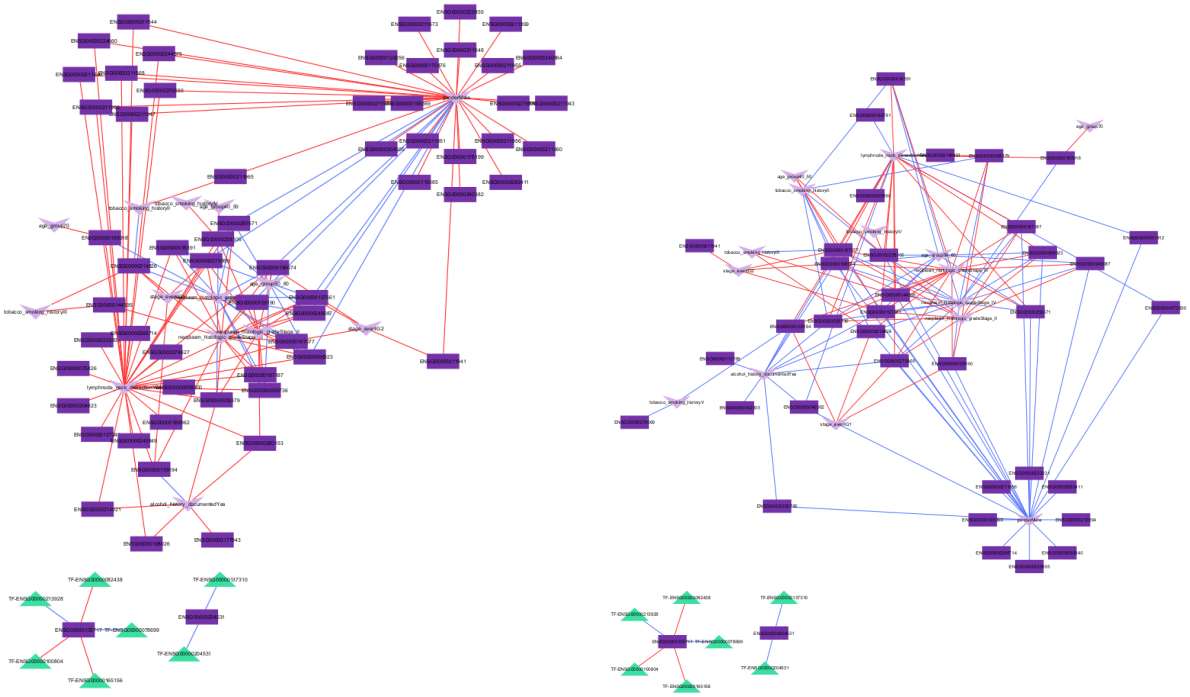


Figure 1: Group_1: "Exitus" a la izquierda y Group_0: "Alta" a la derecha

7. Conclusiones

En conjunto, el modelo predictivo generado mediante MultiBlock-sPLS-DA resulta pobre, pues tiene un sesgo considerable hacia clasificar los pacientes como Alta. Pensamos que esto se debe principalmente a la aproximación inicial que hemos seguido al asumir que los pacientes clasificados como FALSE en la matriz de outcome corresponden a Altas, aunque la realidad sea diferente. Teniendo en cuenta esto, sería recomendable implementar un modelo basado en el tiempo de supervivencia para predecir la evolución clínica en lugar usar el outcome final de forma categórica, usando por ejemplo el paquete Coxmos.

A su vez, el análisis MORE PLS1 muestra escasa relación entre las ómicas reguladoras y el outcome, coherente con lo observado en el MultiBlock-sPLS-DA. En su lugar, este muestra una red en la que la clínica se integra mejor que los factores de transcripción, lo que indicaría que este tipo de datos tendrían más peso en la diferenciación entre grupos del outcome, en especial el género de los pacientes. No obstante, el modelo MORE PLS2 parece integrar mejor los factores de transcripción y lograr un mejor ajuste, aunque las limitaciones técnicas encontradas nos impiden extraer su matriz de reguladores por condición y profundizar este análisis.

Sin embargo, los modelos generados han seleccionado varios genes como *features* clave relacionadas con el outcome de los pacientes que están implicados en la regulación del desarrollo queratinocítico, el perfil transcripcional *stemness* y otros procesos moleculares identificados en la literatura con el desarrollo tumoral de HNSC. Estos genes y procesos pueden ser un punto de partida prometedor para investigaciones futuras que permitan comprender mejor el desarrollo patológico de la enfermedad o identificar biomarcadores y dianas moleculares que mejoren el pronóstico y tratamiento actual respectivamente.

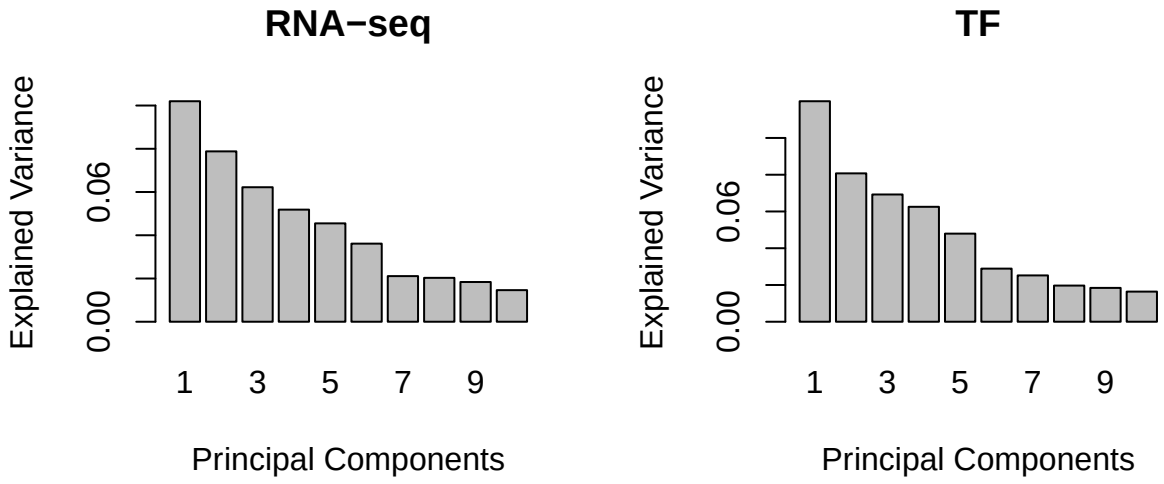
8. Bibliografía

- (1) Liu, S., Lian, M., Han, B., Fang, J., & Wang, Z. (2024). Single-cell integrated transcriptomics reveals the role of keratinocytes in head and neck squamous cell carcinoma. Journal of applied genetics, 65(4), 727–745.

- (2) Siqueira, J. M., Heguedusch, D., Rodini, C. O., Nunes, F. D., & Rodrigues, M. F. S. D. (2023). Mechanisms involved in cancer stem cell resistance in head and neck squamous cell carcinoma. Cancer drug resistance, 6(1), 116–137.
- (3) Topchu, I., Bychkov, I., Gursel, D., Makhov, P., & Boumber, Y. (2024). NSD1 supports cell growth and regulates autophagy in HPV-negative head and neck squamous cell carcinoma. Cell death discovery, 10(1), 75.
- (4) Chai, A. W. Y., Lim, K. P., & Cheong, S. C. (2020). Translational genomics and recent advances in oral squamous cell carcinoma. Seminars in cancer biology, 61, 71–83.
- (5) Hanahan D. (2022). Hallmarks of Cancer: New Dimensions. Cancer discovery, 12(1), 31–46.

Anexo I: PCA

```
mypca = mixOmics::pca(gene.expr, ncomp = 10, center = FALSE, scale = FALSE, )
plot(mypca, main = "RNA-seq")
pca_tf = mixOmics::pca(tf.expr, ncomp = 10, center = FALSE, scale = FALSE)
plot(pca_tf, main = "TF")
```



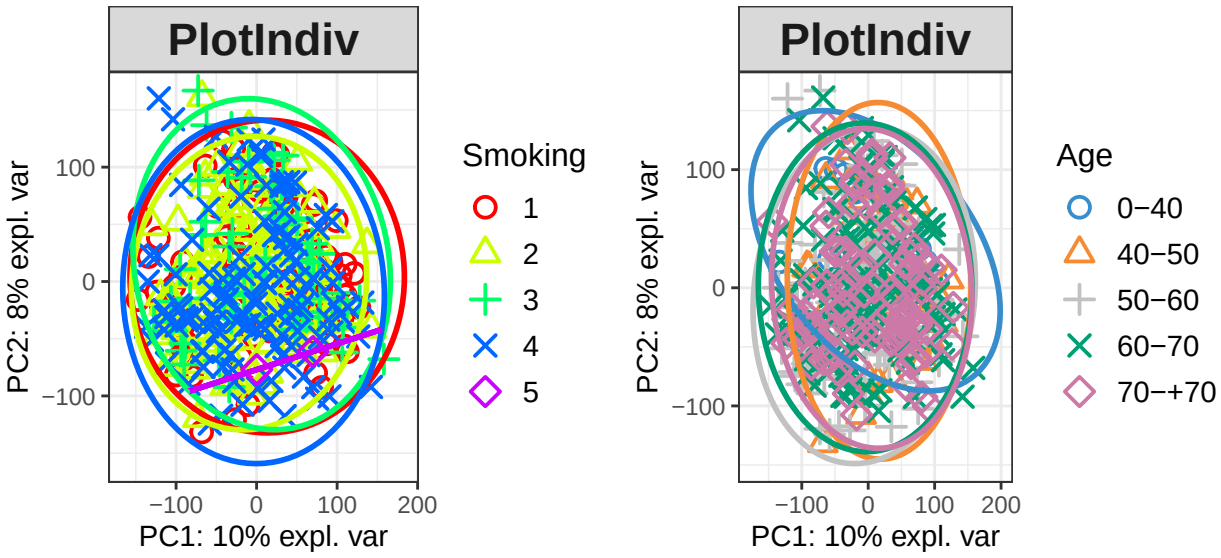
Como podemos observar, la varianza explicada empieza a no aumentar significativamente a partir de la componente 7 para RNA-Seq y 6 para FTs, por lo que se elige este número de componentes para continuar con el PCA.

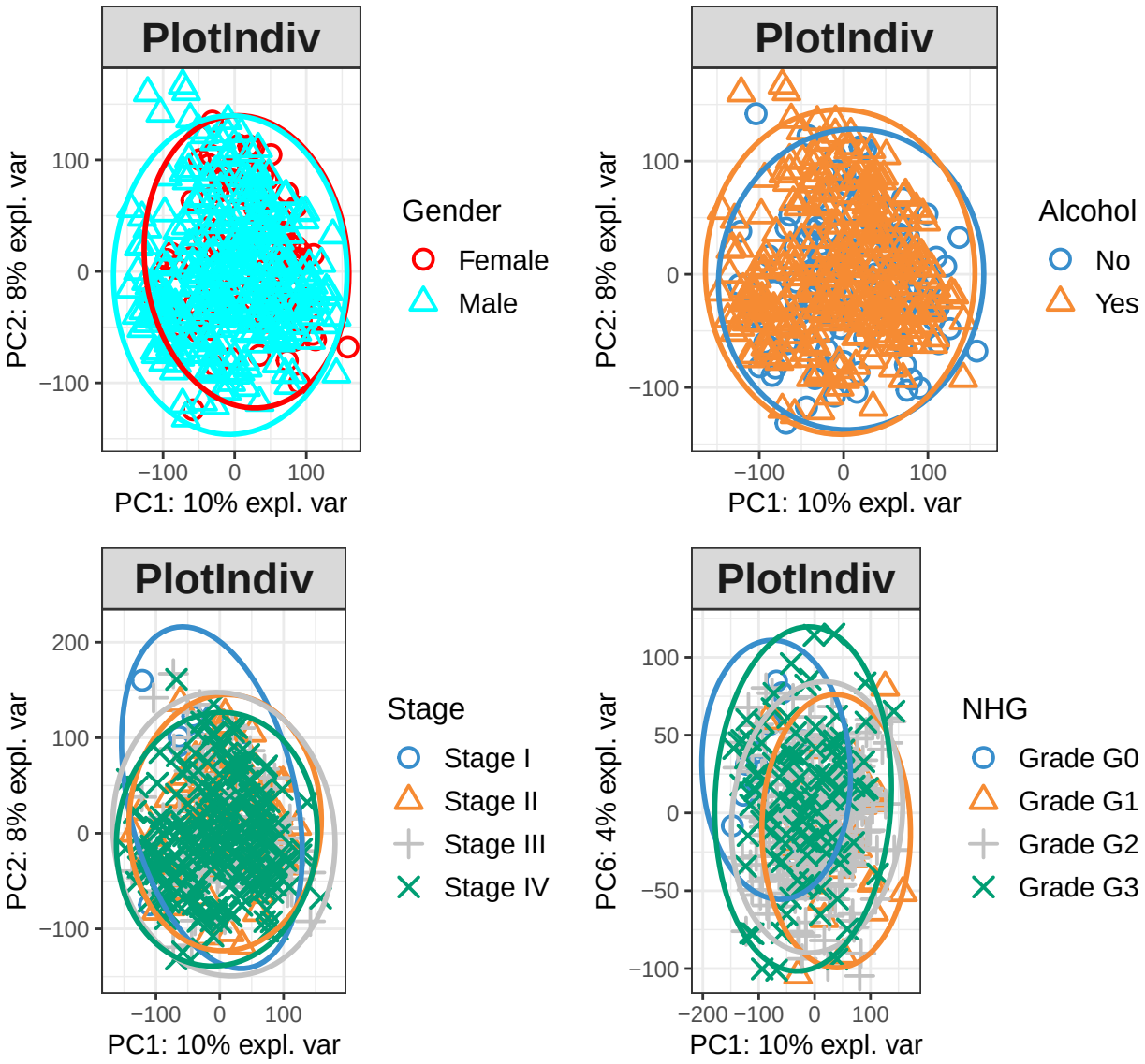
PCA RNA-Seq: agrupación según variables clínicas

```
# Agrupamos las edades por franjas para su representación
clinical <- clinical %>% mutate(age_group = cut(
age, breaks = c(0, 40, 50, 60, 70, Inf), labels = c("0-40", "40-50", "50-60",
                                                    "60-70","70-+70"),right = FALSE))
```

```
plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,
          group = clinical$tobacco_smoking_history,col = rainbow(5),
          style = "ggplot2", legend = TRUE, legend.position = "right",
          legend.title = "Smoking", ellipse = TRUE, ellipse.level = 0.95,
          centroid = FALSE)

plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,
          group = clinical$age_group,style = "ggplot2", legend = TRUE,
          legend.position = "right",  legend.title="Age", ellipse = TRUE,
          ellipse.level = 0.95, centroid = FALSE)

plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,group = clinical$gender,
          col = rainbow(2), style = "ggplot2", legend = TRUE,
```

```
          legend.position = "right", legend.title = "Gender", ellipse = TRUE,
          ellipse.level = 0.95, centroid = FALSE)


plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,
          group = clinical$alcohol_history_documented,style = "ggplot2",
          legend = TRUE, legend.position = "right", legend.title = "Alcohol",
          ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE)

plotIndiv(mypca_final, comp = 1:2, ind.names = NULL,
          group = clinical$stage_event,style = "ggplot2", color=rainbow(4),
          legend = TRUE, legend.position = "right",legend.title = "Stage",
          ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE)

plotIndiv(mypca_final, comp = c(1,6), ind.names = NULL,
          group = clinical$neoplasm_histologic_grade, style = "ggplot2",
          color=rainbow(3), legend = TRUE, legend.position = "right",
          legend.title = "NHG", ellipse = TRUE, ellipse.level = 0.95,
          centroid = FALSE)
```

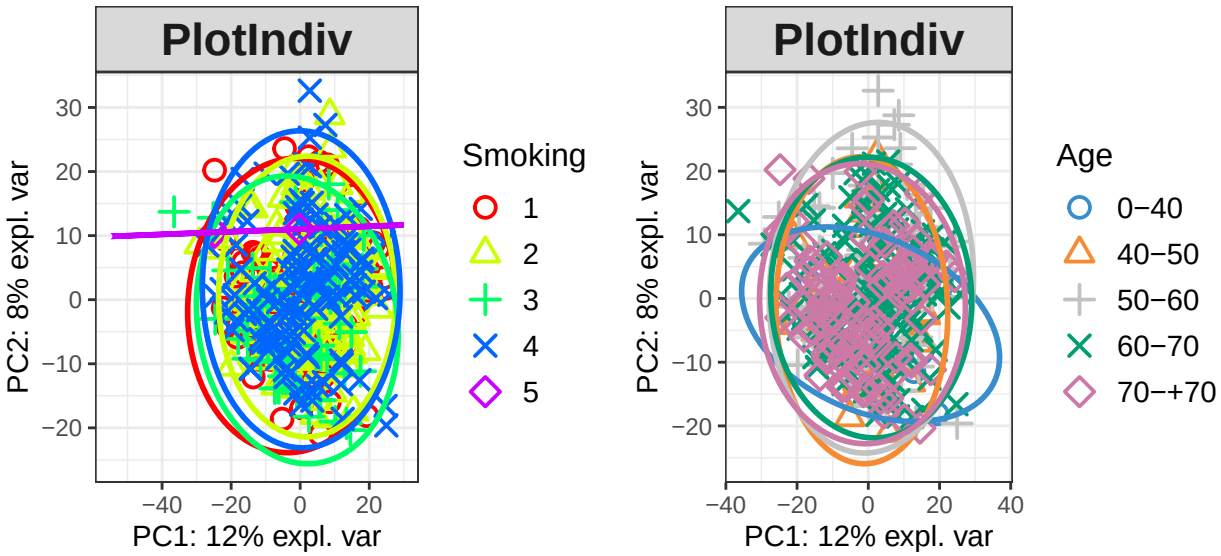


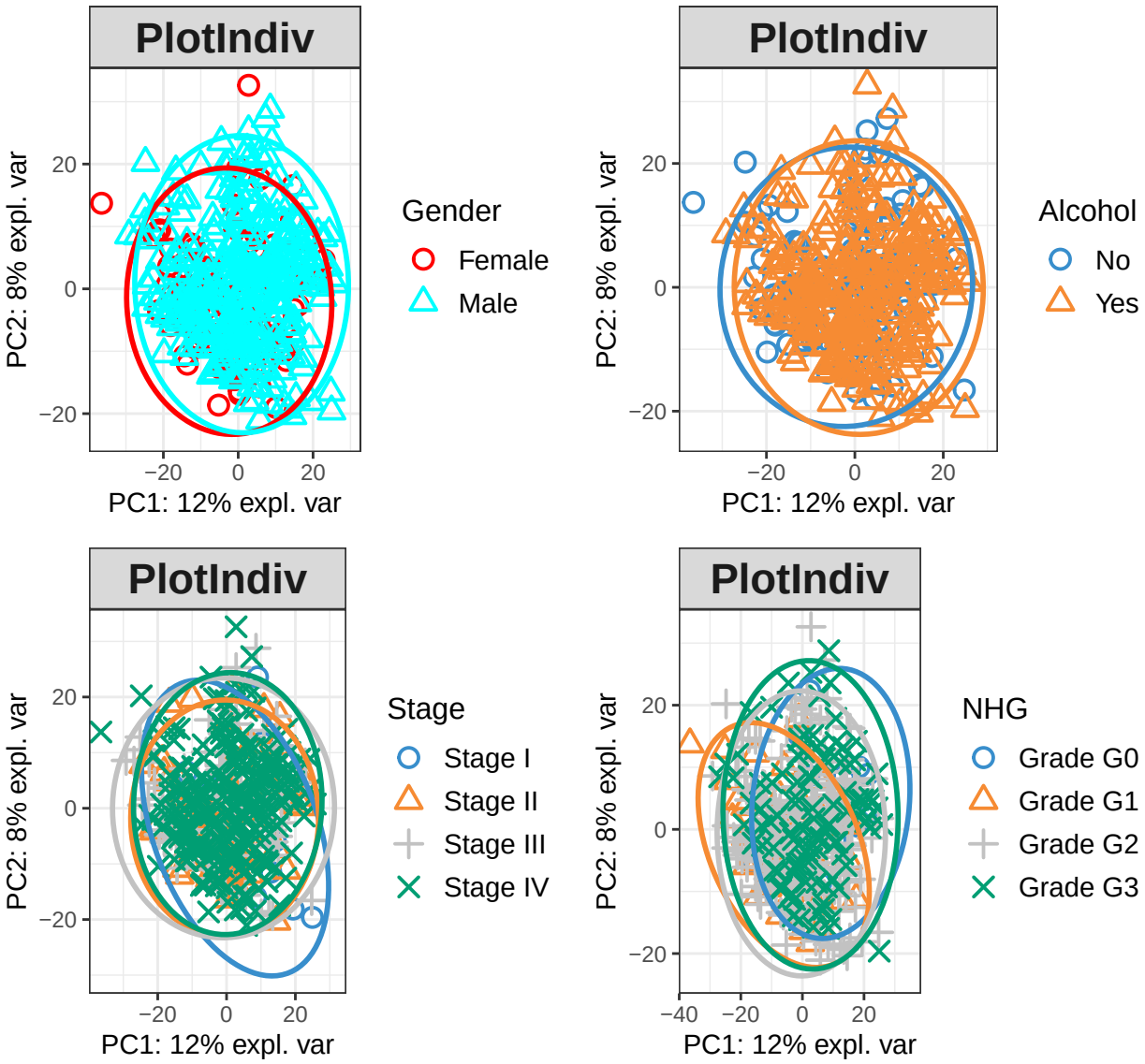


PCA TF: agrupación según variables clínicas

```
plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL,
          group = clinical$tobacco_smoking_history,col = rainbow(5),
          style = "ggplot2", legend = TRUE, legend.position = "right",
          legend.title = "Smoking", ellipse = TRUE, ellipse.level = 0.95,
          centroid = FALSE)

plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL,
          group = clinical$age_group,style = "ggplot2", legend = TRUE,
          legend.position = "right", legend.title="Age", ellipse = TRUE,
          ellipse.level = 0.95, centroid = FALSE)

plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL, group = clinical$gender,
          col = rainbow(2), style = "ggplot2", legend = TRUE,
```

```
          legend.position = "right", legend.title = "Gender", ellipse = TRUE,
          ellipse.level = 0.95, centroid = FALSE)

plotIndiv(pca_tf_final, comp = 1:2,ind.names = NULL,
          group = clinical$alcohol_history_documented, style = "ggplot2",
          legend = TRUE, legend.position = "right", legend.title = "Alcohol",
          ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE)

plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL,
          group = clinical$stage_event,style = "ggplot2", color=rainbow(4),
          legend = TRUE, legend.position = "right",legend.title = "Stage",
          ellipse = TRUE, ellipse.level = 0.95, centroid = FALSE)

plotIndiv(pca_tf_final, comp = 1:2, ind.names = NULL,
          group = clinical$neoplasm_histologic_grade,style = "ggplot2",
          color=rainbow(3),legend = TRUE, legend.position = "right",
          legend.title = "NHG", ellipse = TRUE, ellipse.level = 0.95,
          centroid = FALSE)
```





Anexo II: More de MORE

Preparación de datos

```
# Diseño según outcome
design = outcome[, -1, drop = FALSE]
design$event = design$event * 1

# DEGS y su matriz de expresión
DE.rna <- as.data.frame(t(genes.rna[, DE$...1]))

# FT
TF.omicas <- list(TF = as.data.frame(t(genes.tf)))
```

```r
# Convertimos en lista los datos de asociaciones
asociaciones_TF = list(TF = asociaciones[, c(5,4)])

# Adaptación de datos clínicos para ser legibles por la función more
grade_map <- c("Grade G3" = "G3", "Grade G2" = "G2", "Grade G1" = "G1",
               "Grade 0" = "G0")
grade_map2 = c("Stage IV" = "Stage_IV", "Stage III" = "Stage_III",
               "Stage II" = "Stage_II", "Stage I" = "Stage_I")
grade_map3 = c("5" = "V", "4" = "IV", "3" = "III", "2" = "II", "1" = "I" )
clinical[, 6] <- as.data.frame(lapply(clinical[6], function(x) {
  sapply(x, function(y) grade_map[y])}))
clinical[, 7] = as.data.frame(lapply(clinical[7], function(x) {
  sapply(x, function(y) grade_map2[y])}))
clinical[, 1] = as.data.frame(lapply(clinical[1], function(x) {
  sapply(x, function(y) grade_map3[y])}))
```

Sería interesante explorar el papel de las mutaciones como ómica reguladora de expresión. No obstante, ningún gen mutado se encuentra entre la lista de DEGs para el outcome, por lo que no las hemos tenido en cuenta para el MORE. El siguiente código compara la lista de genes mutados que aparecen en la lista de expresión génica (84) con la lista de genes diferencialmente expresados.

```r
# Mutaciones
# Traducimos entre IDs y ENSEMBL
ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
genes_ensembl <- getBM(attributes = c('ensembl_gene_id', 'hgnc_symbol'),
                  values = colnames(genes.mutation),
                  filters = 'hgnc_symbol',
                  mart = ensembl)
mapping_vector <- setNames(genes_ensembl$ensembl_gene_id,
                           genes_ensembl$hgnc_symbol)
colnames(genes.mutation) <- mapping_vector[colnames(genes.mutation)]
mutaciones_ENSEMBL = colnames(genes.mutation)[
  colnames(genes.mutation) %in% colnames(genes.rna)]
```

```r
load("mutaciones_ENSEMBL.RData")
mutaciones_ENSEMBL %in% rownames(DE.rna)
```

PLS1

```r
moreResults <- more(
  targetData = DE.rna,              # Datos de expresión diferencial entre "Exitus"
                                    # y "Alta"
  regulatoryData = TF.omicas,       # Datos de expresión de factores de transcripción
                                    # como ómica reguladora
  omicType = 0,                     # Datos numéricos
  method = "PLS1",                  # PLS1
  scaleType = 'auto',               # Escalado automático
  interactions = TRUE,              # Considerar interacciones
  associations = asociaciones_TF,   # Matriz de asociaciones entre TF y genes
  minVariation = 0.1,               # Filtrado por variación mínima
  clinic = clinical,                # Datos clínicos
```

```
  clinicType = c(1,0,1,1,1,1,1),    # Vector con el tipo de datos que constituyen
                                    # a cada variable clínica
  varSel = 'Perm',                  # Selección de variables por permutación
  condition = design,               # Diseño experimental
  alfa = 0.05,                      # Valor alfa para la selección de variables
  parallel = TRUE)                  # Paralelización del procesado
```

PLS2

El input es el mismo que el del PLS1, salvo por excluir la matriz de asociaciones.

```
moreResults2 <- more( targetData = DE.rna, regulatoryData = TF.omicas, omicType = 0,
  method = "PLS2", scaleType = 'auto', interactions = TRUE, minVariation = 0.1,
  clinic = clinical, clinicType = c(1,0,1,1,1,1,1), varSel = 'Perm',
  condition = design, alfa = 0.05, parallel = TRUE)
```

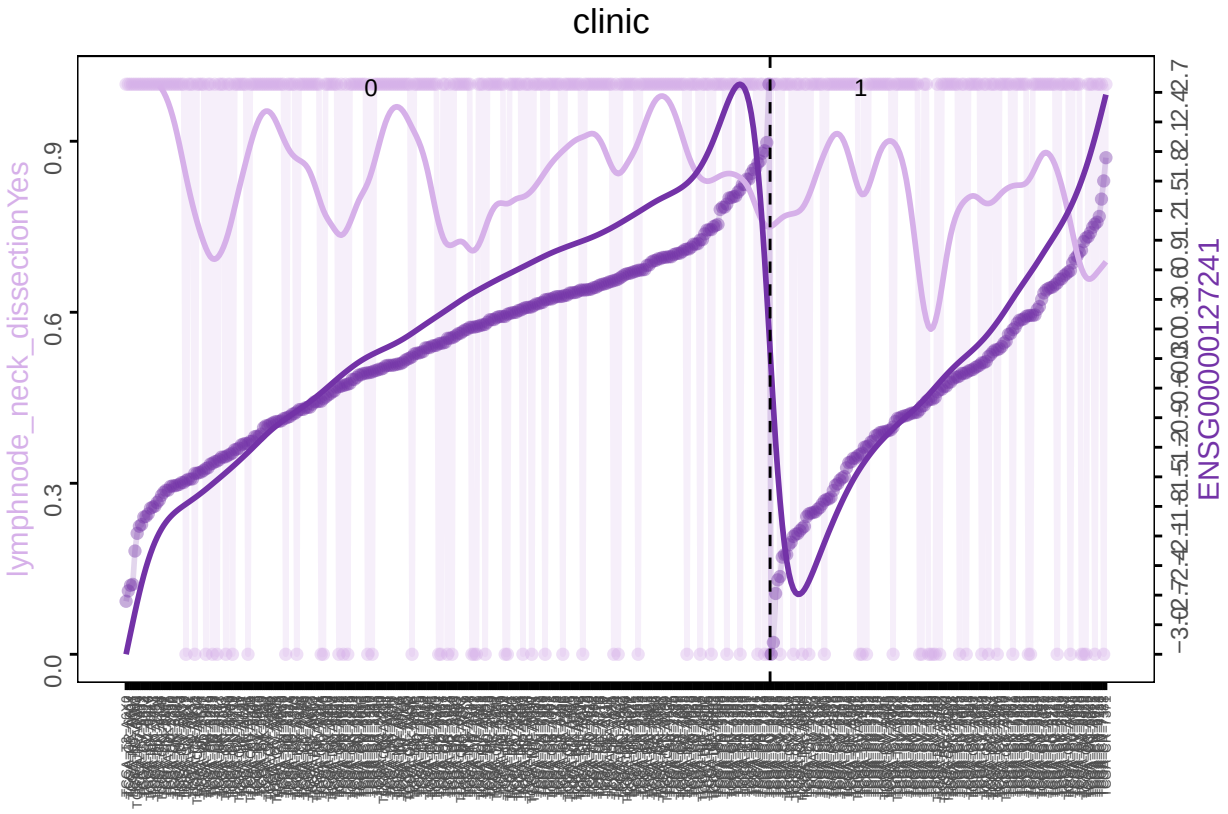
Aplicamos la función "RegulationPerCondition" al PLS2 insatisfactoriamente, de haber resultado efectivo, aplicaríamos el mismo análisis de PLS1 a PLS2 para obtener la información pertinente acerca de los reguladores del modelo. Parece que existen valores nulos en el objeto *moreResults2* que impiden que se ejecute correctamente. Las únicas diferencias respecto al objeto *moreResults* del PLS1 son la lista de asociaciones dentro de arguments (vacío por el propio diseño del PLS2) y que 10 de los 121 genes en ResultsPerTargetF carecen de reguladores significativos y sus coeficientes están vacíos. No obstante, ni incluyendo valores "None" artificialmente ni filtrando estos genes de la lista se ha conseguido ejecutar esta función con éxito.

```
regpcond2 = RegulationPerCondition(moreResults2)
```

Análisis gen a gen

A la hora de representar la corelación entre las variables reguladoras y la expresión de los genes de interés para cada observación, resulta poco interesante evaluar las gráficas de las variables clínicas categóricas predominantes en PLS1. No obstante, no nos ha sido posible representar la relación entre la expresión de genes y la expresión de sus FTs reguladores según el modelo PLS2. A pesar de haber sido identificados previamente como significativos (como TF-ENSG00000129152 para ENSG00000127241), en esta ocasión no son reconocidos como tales.

```
# PLS1
p1 = plotMORE(moreResults, targetF = "ENSG00000127241",
              regulator = "lymphnode_neck_dissectionYes", order = TRUE,size = 5)
```



Análisis de enriquecimiento (ORA)

Si bien lo ideal hubiese sido aplicar un ORA para el PLS2, no ha sido posible por no poder aplicar la función "RegulationPerConditon". Por otro lado, el GSDA (gene set enrichment analysis) no nos muestra resultados significativos, por lo que analizamos exclusivamente los resultados del ORA.

```
regpcond_filtered = RegulationPerCondition(moreResults, filterR2 = 0.8)
```

| | | 0% | |= | 1%

```
annotation = read.csv("annotation.txt")
regincond = RegulationInCondition(regpcond_filtered, cond = 'Group_1')
ORA=oraMORE(moreResults,regincond, byHubs = TRUE, annotation = annotation)

head(ORA$termDescr)
```

[1] "Binding to an RNA molecule or a portion thereof."
[2] "Binding to a specific upstream regulatory DNA sequence (transcription factor recognition sequence or binding site) located in cis relative to the transcription start site (i.e., on the same strand of DNA) of a gene transcribed by RNA polymerase II."
[3] "A DNA-binding transcription factor activity that modulates the transcription of specific gene sets transcribed by RNA polymerase II."
[4] "Any process that modulates the frequency, rate or extent of transcription mediated by RNA polymerase II."

[5] “Binding to DNA of a specific nucleotide composition, e.g. GC-rich DNA binding, or with a specific sequence motif or type of DNA e.g. promotor binding or rDNA binding.”
[6] “A transcription regulator activity that modulates transcription of gene sets via selective and non-covalent binding to a specific double-stranded genomic DNA sequence (sometimes referred to as a motif) within a cis-regulatory region. Regulatory regions include promoters (proximal and distal) and enhancers. Genes are transcriptional units, and include bacterial operons.”

GSEA

Por otro lado, el GSDA (gene set enrichment analysis) no nos muestra resultados significativos, por lo que analizamos exclusivamente los resultados del ORA.

```
regincond2 = RegulationInCondition(regpcond_filtered, cond = 'Group_0')
gsea_out = gseaMORE(outputRegincond =  regincond, outputRegincond2 = regincond2,
                    annotation = annotation)
```