TFM - Análisis de expresión diferencial

Uxue Alvarez Huesa

16 de enero, 2024

Índice

Análisis diferencial genómico de una base de datos **1**

Carga de librerías 1

Selección del dataset 2

Análisis de expresión diferencial genómica entre genotipos de Síndrome de Turner Xm y Xp 3

Análisis de expresión diferencial genómica entre genotipos de Síndrome de Turner X (X = Xm y Xp) y las muestras control (XX) 20

Análisis diferencial genómico de dos bases de datos **36**

Selección del dataset 36

Creación del objeto "targets" 37

Carga y lectura de los datos 37

Preprocesado de los datos 40

Análisis 46

Análisis de genes diferencialmente expresados utilizando p = 0.1 48

Referencias **54**

Análisis diferencial genómico de una base de datos

En este informe se va a realizar un análisis de microarrays a partir de datos de un estudio publicado y depositados en "*Gene Expression Omnibus*". El estudio que se ha seleccionado es el codificado con el identificador GEO **GSE46687** el cuál analiza la expresión génica diferencial en células mononucleares de sangre periférica (PBMC) de 45Xm y 45Xp mediante microarrays. También se analiza en paralelo la expresión génica de las mujeres control 46XX para investigar los cambios en la expresión génica de todo el genoma entre los pacientes con Síndrome de Turner con monosomía X y las mujeres sin monosomía (sin tener en cuenta el tipo de herencia del cromosoma X).

Carga de librerías

Antes de empezar con el análisis se cargan los paquetes necesarios, instalados previamente.

```
library(affycoretools)
library(annotate)
library(AnnotationDbi)
library(arrayQualityMetrics)
library(Biobase)
library(cluster)
library(clusterProfiler)
library(dplyr)
library(enrichplot)
library(genefilter)
library(ggplot2)
library(GOstats)
library(gplots)
library(limma)
library(oligo)
library(oligoClasses)
library(openxlsx)
library(org.Hs.eg.db)
library(pvca)
library(reactome.db)
library(ReactomePA)
library(readxl)
library(xtable)
```

Selección del dataset

Se ha seleccionado el dataset GSE46687, el cuál corresponde al trabajo de Cheng CM *et al.* llamado Gene Expression Profiling in 45X Turner Syndrome patients.

El estudio se compone de 36 muestras y las anotaciones de los genes se encuentran en *Affymetrix Human Genome U133 Plus 2.0 Array*. Estas 36 muestras se dividen en:

- **XX**: 10 réplicas de muestras de control (*wild type*).
- **Xm**: 16 réplicas de muestras con Síndrome de Turner con el cromosoma X heredado de la madre
- **Xp**: 10 réplicas de muestras con Síndrome de Turner con el cromosoma X heredado del padre

Por lo tanto, se tienen 3 tipos de muestras, una que pertenece al control sin monosomía del cromosoma X y, por lo tanto, Síndrome de Turner (XX) y dos tipos de genotipos de Síndrome de Turner (monosomía del cromosoma X); estos dos tipos de monosomía se dividen en si el cromosoma X ha sido heredado por parte de la madre (Xm) o del padre (Xp).

Teniendo esto en cuenta, para facilitar el análisis primero se va a analizar la diferencia de expresión genómica entre los individuos con Síndrome de Turner (Xm y Xp), para poder concluir si estas muestras se pueden unir y compararlas conjuntamente con las muestras de control o si, por el contrario, se debe realizar el estudio comparándolos individualmente con el control. Esto hará que resulte más fácil crear la matríz de correlación.

Análisis de expresión diferencial genómica entre genotipos de Síndrome de Turner Xm y Xp

Creación del objeto "targets"

En la carpeta `Data` donde se encuentran los archivos .CEL, se ha creado un archivo .csv llamado `targets1`. Este archivo contiene la información de las diferentes muestras del experimento para poder crear un objeto *AnnotatedDataFrame*.

fileName	grupos	ShortName	Colors
GSM1134026	Xm	26_Xm	red
GSM1134027	Xm	27_Xm	red
GSM1134028	Xm	28_Xm	red
GSM1134029	Xm	29_Xm	red
GSM1134030	Xm	30_Xm	red
GSM1134031	Xm	31_Xm	red
GSM1134032	Xm	32_Xm	red
GSM1134033	Xm	33_Xm	red
GSM1134034	Xm	34_Xm	red
GSM1134035	Xm	35_Xm	red
GSM1134036	Xm	36_Xm	red
GSM1134037	Xm	37_Xm	red
GSM1134038	Xm	38_Xm	red
GSM1134039	Xm	39_Xm	red
GSM1134040	Xm	40_Xm	red
GSM1134041	Xm	41_Xm	red
GSM1134042	Xp	42_Xp	blue
GSM1134043	Xp	43_Xp	blue
GSM1134044	Xp	44_Xp	blue
GSM1134045	Xp	45_Xp	blue
GSM1134046	Xp	46_Xp	blue
GSM1134047	Xp	47_Xp	blue
GSM1134048	Xp	48_Xp	blue
GSM1134049	Xp	49_Xp	blue
GSM1134050	Xp	50_Xp	blue
GSM1134051	Xp	51_Xp	blue

Figura 1: Contenido del inicio del archivo targets1.csv

Para organizar el estudio se han creado algunos directorios:

```
workingDir <- getwd()
dataDir <- file.path(workingDir, "Data")
```

`dataDir` será el directorio donde están guardados los datos del análisis.

Carga y lectura de datos

Para poder comenzar con el preprocesado de los datos, primero se lee el archivo `targets1.csv` y se almacenan las columnas `ShortName` y `Colors` de este archivo en dos nuevas variables (`sampleNames1` y `sampleColor1`) para poder crear los gráficos posteriormente. Como se puede observar, el objeto `targetsDF1` contiene el dataframe que se ha creado anteriormente con la información de las muestras y algunos campos importantes para el análisis.

```
# Carga del archivo CSV que contiene la información de las muestras a analizar (targets)
targetsDF1 <- read.csv2(file = file.path(dataDir, "targets1.csv"), header = TRUE, sep = ";")

# Extracción de las abreviaturas de los nombres y colores de las muestras del data frame 'targetsDF1'
```

```
sampleNames1 <- as.character(targetsDF1$ShortName)
sampleColor1 <- as.character(targetsDF1$Colors)

# Creación de un objeto AnnotatedDataFrame a partir de 'targetsDF1' para asociar información adicional
targets1 <- AnnotatedDataFrame(targetsDF1)

# Muestra del contenido del dataframe targetsDF1
targetsDF1
```

```
##          fileName Grupos ShortName Colors
## 1  GSM1134026.CEL     Xm     01_Xm    red
## 2  GSM1134027.CEL     Xm     02_Xm    red
## 3  GSM1134028.CEL     Xm     03_Xm    red
## 4  GSM1134029.CEL     Xm     04_Xm    red
## 5  GSM1134030.CEL     Xm     05_Xm    red
## 6  GSM1134031.CEL     Xm     06_Xm    red
## 7  GSM1134032.CEL     Xm     07_Xm    red
## 8  GSM1134033.CEL     Xm     08_Xm    red
## 9  GSM1134034.CEL     Xm     09_Xm    red
## 10 GSM1134035.CEL     Xm     10_Xm    red
## 11 GSM1134036.CEL     Xm     11_Xm    red
## 12 GSM1134037.CEL     Xm     12_Xm    red
## 13 GSM1134038.CEL     Xm     13_Xm    red
## 14 GSM1134039.CEL     Xm     14_Xm    red
## 15 GSM1134040.CEL     Xm     15_Xm    red
## 16 GSM1134041.CEL     Xm     16_Xm    red
## 17 GSM1134042.CEL     Xp     17_Xp   blue
## 18 GSM1134043.CEL     Xp     18_Xp   blue
## 19 GSM1134044.CEL     Xp     19_Xp   blue
## 20 GSM1134045.CEL     Xp     20_Xp   blue
## 21 GSM1134046.CEL     Xp     21_Xp   blue
## 22 GSM1134047.CEL     Xp     22_Xp   blue
## 23 GSM1134048.CEL     Xp     23_Xp   blue
## 24 GSM1134049.CEL     Xp     24_Xp   blue
## 25 GSM1134050.CEL     Xp     25_Xp   blue
## 26 GSM1134051.CEL     Xp     26_Xp   blue
```

A continuación, se guardan los nombres de los archivos .CEL guardados en el dataframe `targetsDF1` en un nuevo objeto llamado `CELfiles1` y se cargan los archivos del directorio que tengan el mismo nombre que los guardados en ese objeto. Estos archivos se almacenan en un objeto ExpressionFeatureSet llamado `rawData1`.

```
# Extracción de los nombres de los archivos CEL desde la columna 'fileName' del data frame 'targetsDF1'
CELfiles1 <- targetsDF1$fileName

# Lee los archivos CEL ubicados en 'dataDir', utilizando los nombres extraídos en CELfiles1
# Mediante read.celfiles() se leen los archivos CEL y se devuelve un objeto ExpressionFeatureSet (rawDa
# phenoData se asocia con el objeto AnnotatedDataFrame creado previamente ('targets1'), donde se encuen
rawData1 <- read.celfiles(file.path(dataDir, CELfiles1), phenoData = targets1)
```

```
## Reading in : E:/TFM/Data/GSM1134026.CEL
## Reading in : E:/TFM/Data/GSM1134027.CEL
## Reading in : E:/TFM/Data/GSM1134028.CEL
```

```
## Reading in : E:/TFM/Data/GSM1134029.CEL
## Reading in : E:/TFM/Data/GSM1134030.CEL
## Reading in : E:/TFM/Data/GSM1134031.CEL
## Reading in : E:/TFM/Data/GSM1134032.CEL
## Reading in : E:/TFM/Data/GSM1134033.CEL
## Reading in : E:/TFM/Data/GSM1134034.CEL
## Reading in : E:/TFM/Data/GSM1134035.CEL
## Reading in : E:/TFM/Data/GSM1134036.CEL
## Reading in : E:/TFM/Data/GSM1134037.CEL
## Reading in : E:/TFM/Data/GSM1134038.CEL
## Reading in : E:/TFM/Data/GSM1134039.CEL
## Reading in : E:/TFM/Data/GSM1134040.CEL
## Reading in : E:/TFM/Data/GSM1134041.CEL
## Reading in : E:/TFM/Data/GSM1134042.CEL
## Reading in : E:/TFM/Data/GSM1134043.CEL
## Reading in : E:/TFM/Data/GSM1134044.CEL
## Reading in : E:/TFM/Data/GSM1134045.CEL
## Reading in : E:/TFM/Data/GSM1134046.CEL
## Reading in : E:/TFM/Data/GSM1134047.CEL
## Reading in : E:/TFM/Data/GSM1134048.CEL
## Reading in : E:/TFM/Data/GSM1134049.CEL
## Reading in : E:/TFM/Data/GSM1134050.CEL
## Reading in : E:/TFM/Data/GSM1134051.CEL
```

Una vez cargados todos los archivos, se analiza el objeto ExpressionFeatureSet `rawData1`.

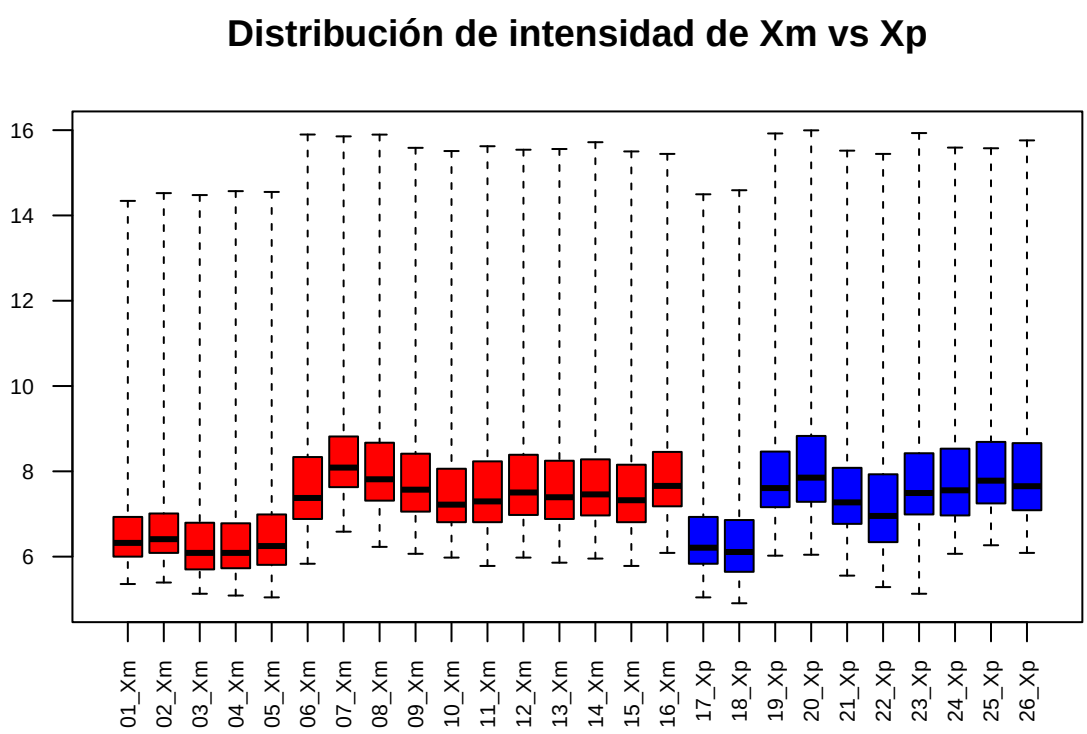
```
rawData1
```

```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 1354896 features, 26 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 26 (26 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 26 (26 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

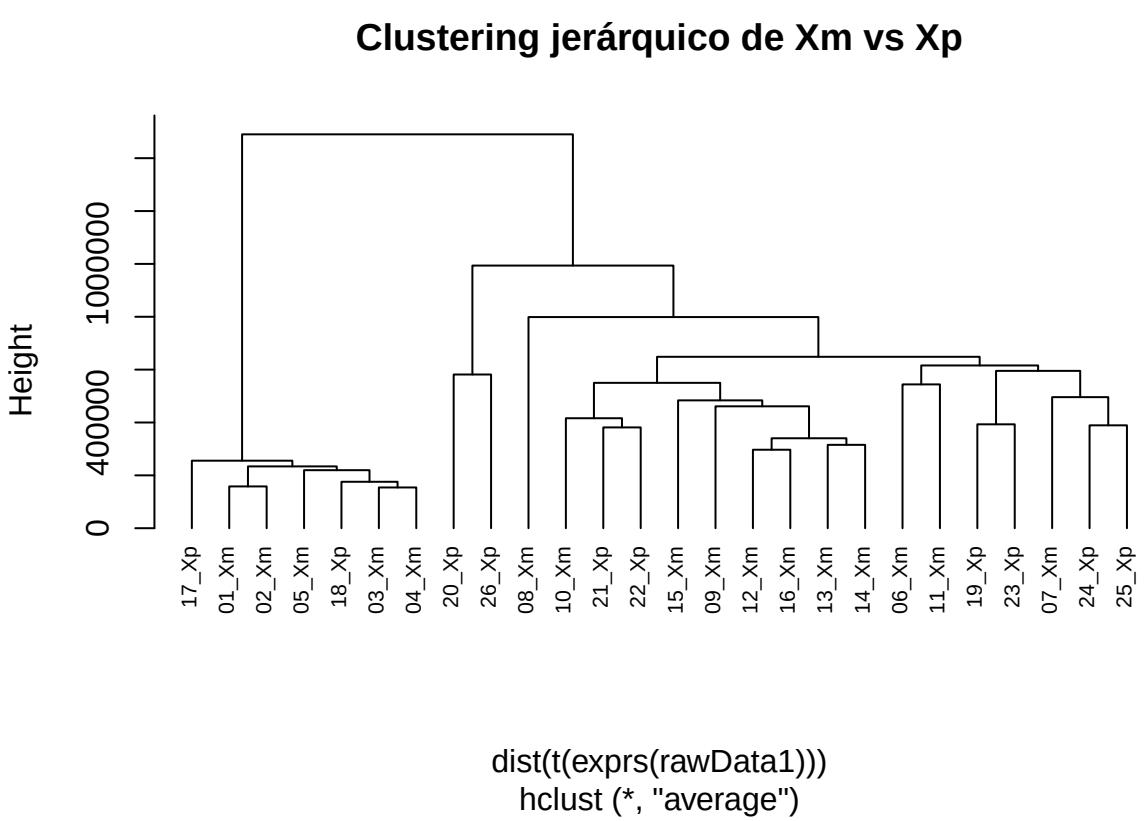
Como se puede observar, el archivo contiene 26 muestras, ya que se han eliminado las muestras que pertenecen al control, las cuáles hacen una suma de 1354896 sondas en total. Además, `Annotation` indica el paquete de anotaciones necesario para poder realizar este análisis, en este caso **hgu133plus2 .db**.

Preprocesado de los datos

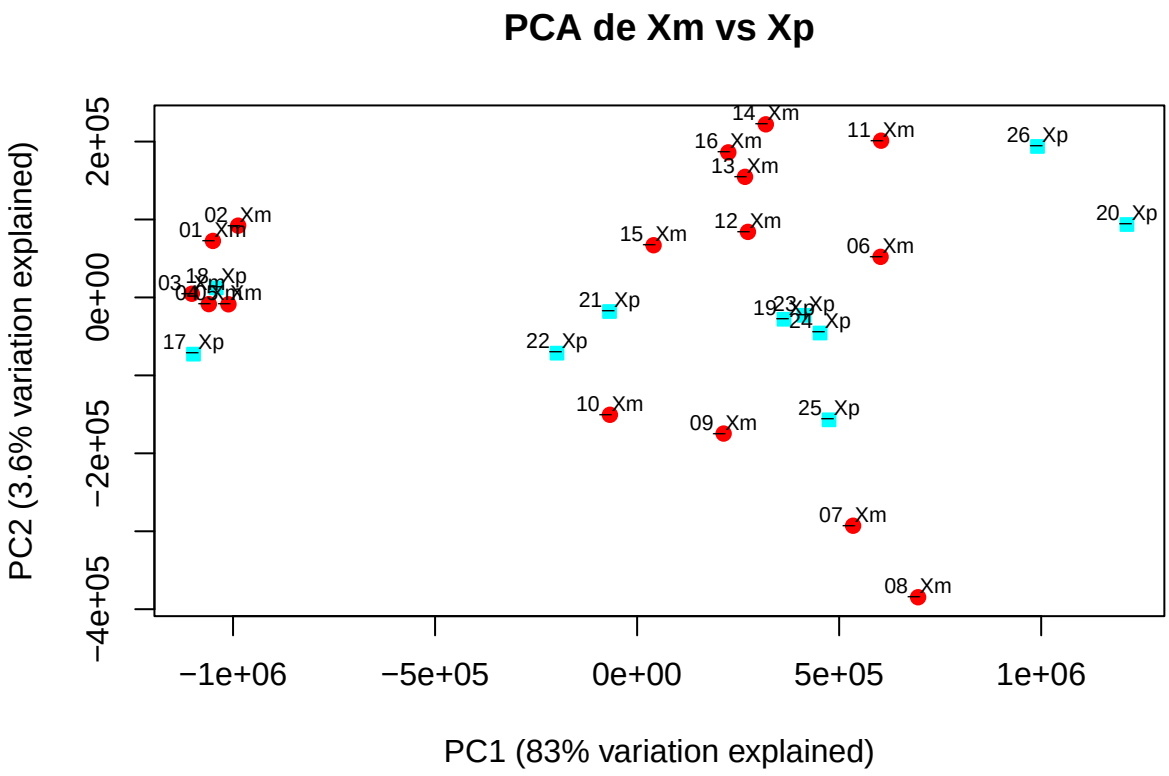
Exploración y control de calidad Mediante un **boxplot** se muestra como es la distribución de los valores.



Mediante un **clustering jerárquico** se muestra como se agrupan estas muestras. Para realizar esta gráfica, primero se calcula la distancia euclidiana entre las muestras de la matríz de expresión de `rawData1`, mediante `t()` se transpone la matriz para que las distancias se calculen entre las muestras en lugar de los genes. Mediante `hclust()` se realiza un clustering jerárquico con los datos modificados anteriormente utilizando el método "average". Para finalizar, se muestra el dendrograma resultante del clustering jerárquico.



En el **análisis de componentes principales**, se buscan las principales fuentes de variabilidad en los datos reduciendo las dimensiones. Esta gráfica se realiza empleando la función `plotPCA()` del paquete de Bioconductor `affycoretools`



Control de calidad con el paquete arrayQualityMetrics Con este paquete se obtienen los mismos gráficos que los anteriores pero todos a la vez en un archivo.

```
arrayQualityMetrics(rawData1, reporttitle = "QC_RawData1", force = TRUE)
```

Normalización Antes de comenzar con el análisis de expresión diferencial, es necesario hacer que los arrays sean comparables entre sí y tratar de reducir, y si es posible eliminar, toda la variabilidad en las muestras que no se deba a razones biológicas. El proceso de normalización trata de asegurar que las diferencias de intensidad presentes en el array reflejen la expresión diferencial de los genes, en lugar de sesgos artificiales debidos a cuestiones técnicas. El método más utilizado para la normalización de arrays es el método RMA.

```
normData1 <- rma(rawData1)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

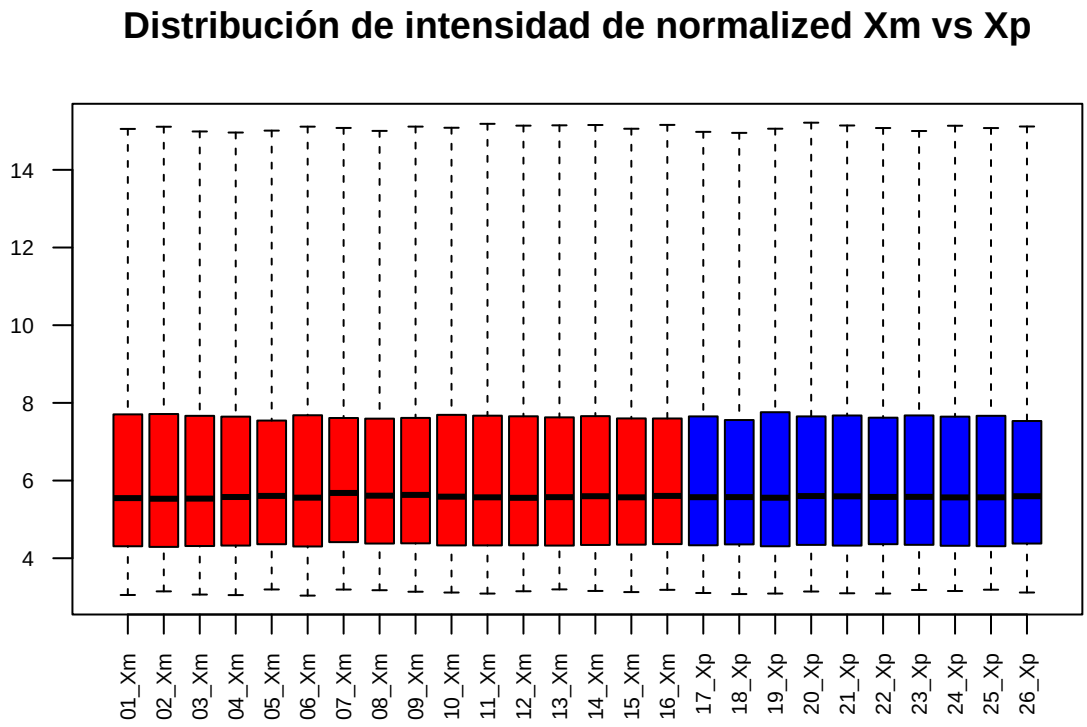
```
normData1
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 26 samples
##   element names: exprs
## protocolData
```

```
##   rowNames: 1 2 ... 26 (26 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 26 (26 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

Tras la normalización de los datos se obtienen menos sondas en el ExpressionSet `normData1` que en el archivo `rawData1`. Esto se debe a que en `rawData1` este número hace referencia a las sondas individuales, mientras que el objeto `normData1` hace referencia a los grupos de sondas.

Analizando la calidad de los datos tras la normalización:



Además, se crea un nuevo archivo QC mediante el paquete `arrayQualityMetrics` para los datos normalizados.

```
arrayQualityMetrics(normData1, reporttitle = "QC_NormData1", force = TRUE)
```

Se elimina la muestra que tiene porblemas de valores atípicos y se vuelven a cargar y normalizar lo datos.

Carga y lectura de los datos

```
targetsDF1 <- read.csv2(file = file.path(dataDir, "targets1.1.csv"), header = TRUE, sep = ";")

sampleNames1 <- as.character(targetsDF1$ShortName)
sampleColor1 <- as.character(targetsDF1$Colors)

targets1 <- AnnotatedDataFrame(targetsDF1)

targetsDF1
```

```
##          fileName Grupos ShortName Colors
## 1  GSM1134026.CEL     Xm     01_Xm    red
## 2  GSM1134027.CEL     Xm     02_Xm    red
## 3  GSM1134028.CEL     Xm     03_Xm    red
## 4  GSM1134029.CEL     Xm     04_Xm    red
## 5  GSM1134030.CEL     Xm     05_Xm    red
## 6  GSM1134031.CEL     Xm     06_Xm    red
## 7  GSM1134033.CEL     Xm     08_Xm    red
## 8  GSM1134034.CEL     Xm     09_Xm    red
## 9  GSM1134035.CEL     Xm     10_Xm    red
## 10 GSM1134036.CEL     Xm     11_Xm    red
## 11 GSM1134037.CEL     Xm     12_Xm    red
## 12 GSM1134038.CEL     Xm     13_Xm    red
## 13 GSM1134039.CEL     Xm     14_Xm    red
## 14 GSM1134040.CEL     Xm     15_Xm    red
## 15 GSM1134041.CEL     Xm     16_Xm    red
## 16 GSM1134042.CEL     Xp     17_Xp   blue
## 17 GSM1134043.CEL     Xp     18_Xp   blue
## 18 GSM1134044.CEL     Xp     19_Xp   blue
## 19 GSM1134045.CEL     Xp     20_Xp   blue
## 20 GSM1134046.CEL     Xp     21_Xp   blue
## 21 GSM1134047.CEL     Xp     22_Xp   blue
## 22 GSM1134048.CEL     Xp     23_Xp   blue
## 23 GSM1134049.CEL     Xp     24_Xp   blue
## 24 GSM1134050.CEL     Xp     25_Xp   blue
## 25 GSM1134051.CEL     Xp     26_Xp   blue
```

A continuación, se guardan los nombres de los archivos .CEL guardados en el dataframe `targetsDF2` en un nuevo objeto llamado `CELfiles2` y se cargan los archivos del directorio que tengan el mismo nombre que los guardados en ese objeto. Estos archivos se almacenan en un objeto ExpressionFeatureSet llamado `rawData2`.

```
CELfiles1 <- targetsDF1$fileName
rawData1 <- read.celfiles(file.path(dataDir, CELfiles1), phenoData = targets1)
```

```
## Reading in : E:/TFM/Data/GSM1134026.CEL
## Reading in : E:/TFM/Data/GSM1134027.CEL
## Reading in : E:/TFM/Data/GSM1134028.CEL
## Reading in : E:/TFM/Data/GSM1134029.CEL
## Reading in : E:/TFM/Data/GSM1134030.CEL
## Reading in : E:/TFM/Data/GSM1134031.CEL
## Reading in : E:/TFM/Data/GSM1134033.CEL
```

```
## Reading in : E:/TFM/Data/GSM1134034.CEL
## Reading in : E:/TFM/Data/GSM1134035.CEL
## Reading in : E:/TFM/Data/GSM1134036.CEL
## Reading in : E:/TFM/Data/GSM1134037.CEL
## Reading in : E:/TFM/Data/GSM1134038.CEL
## Reading in : E:/TFM/Data/GSM1134039.CEL
## Reading in : E:/TFM/Data/GSM1134040.CEL
## Reading in : E:/TFM/Data/GSM1134041.CEL
## Reading in : E:/TFM/Data/GSM1134042.CEL
## Reading in : E:/TFM/Data/GSM1134043.CEL
## Reading in : E:/TFM/Data/GSM1134044.CEL
## Reading in : E:/TFM/Data/GSM1134045.CEL
## Reading in : E:/TFM/Data/GSM1134046.CEL
## Reading in : E:/TFM/Data/GSM1134047.CEL
## Reading in : E:/TFM/Data/GSM1134048.CEL
## Reading in : E:/TFM/Data/GSM1134049.CEL
## Reading in : E:/TFM/Data/GSM1134050.CEL
## Reading in : E:/TFM/Data/GSM1134051.CEL
```

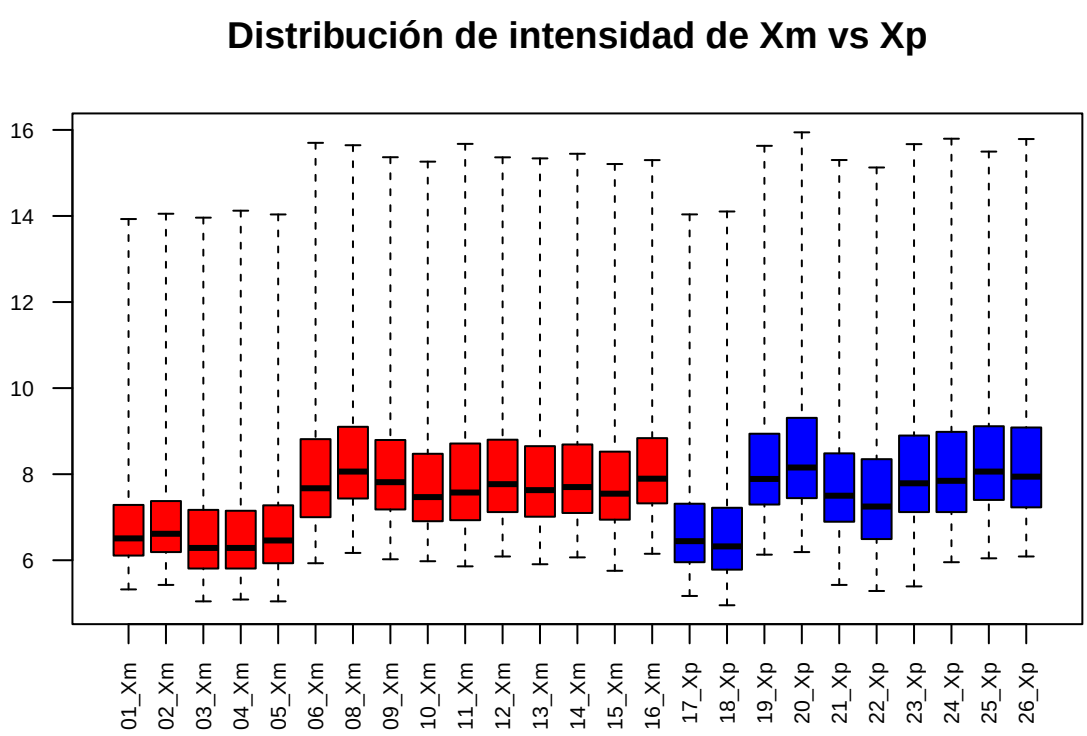
```
rawData1
```

```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 1354896 features, 25 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

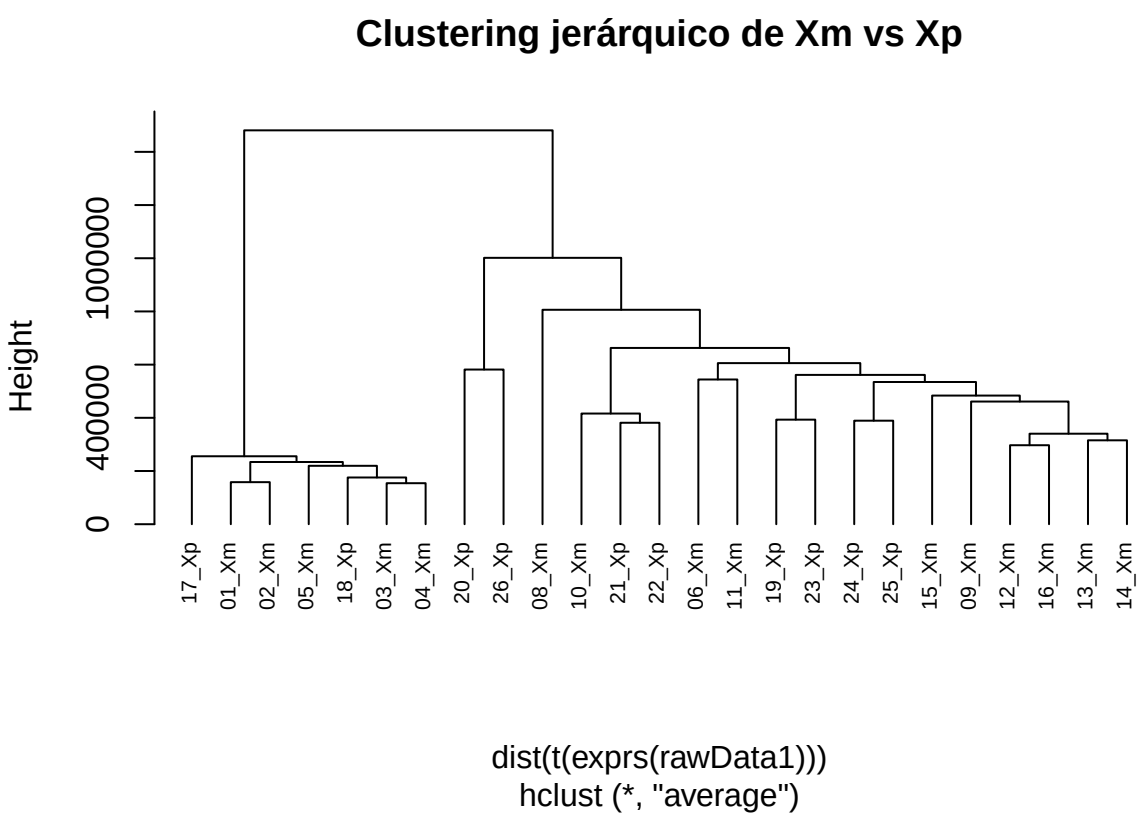
Como se puede observar, el archivo contiene 25 muestras, ya que se ha eliminado la muestra con problemas de outliers. Además, `Annotation` indica el paquete de anotaciones necesario para poder realizar este análisis, el mismo que se ha utilizado anteriormente.

Preprocesado de los datos

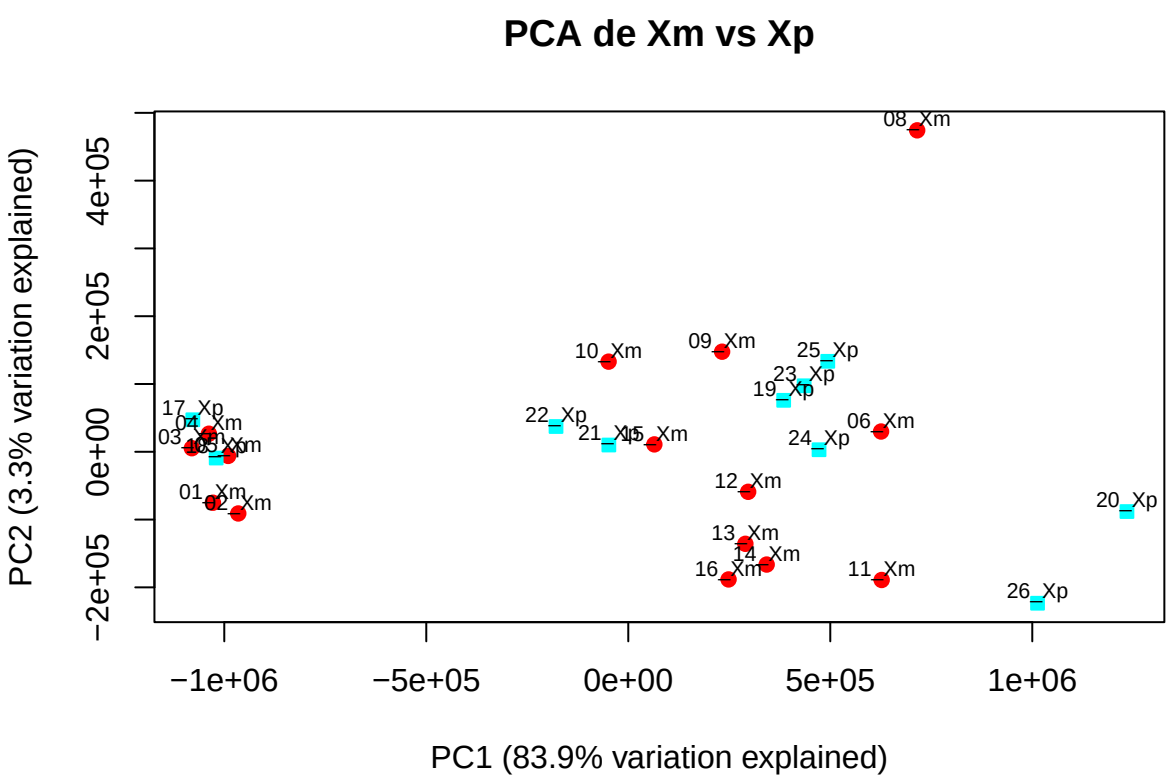
Exploración y control de calidad Mediante un **boxplot** se muestra como es la distribución de los valores.



Mediante un **clustering jerárquico** se muestra como se agrupan estas muestras.



En el **análisis de componentes principales**, se buscan las principales fuentes de variabilidad en los datos reduciendo las dimensiones.



Normalización

Antes de comenzar con el análisis de expresión diferencial y vistos los problemas que muestran algunas de las muestras, es necesario hacer normalizar los arrays con el método RMA.

```
normData1 <- rma(rawData1)
```

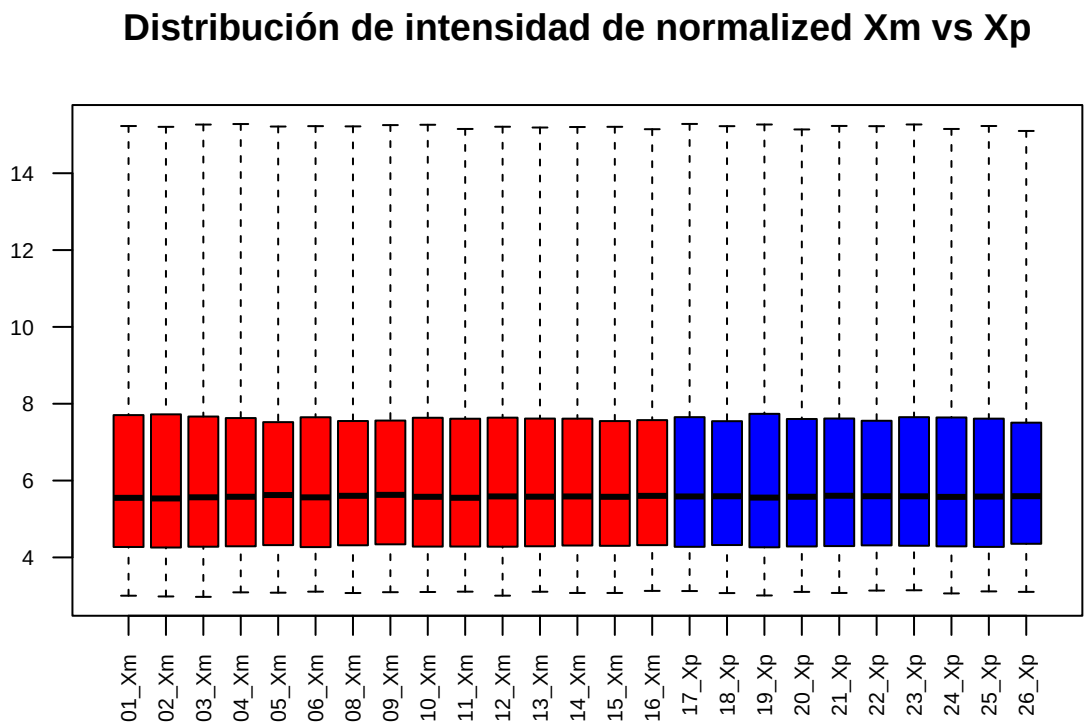
```
## Background correcting
## Normalizing
## Calculating Expression
```

```
normData1
```

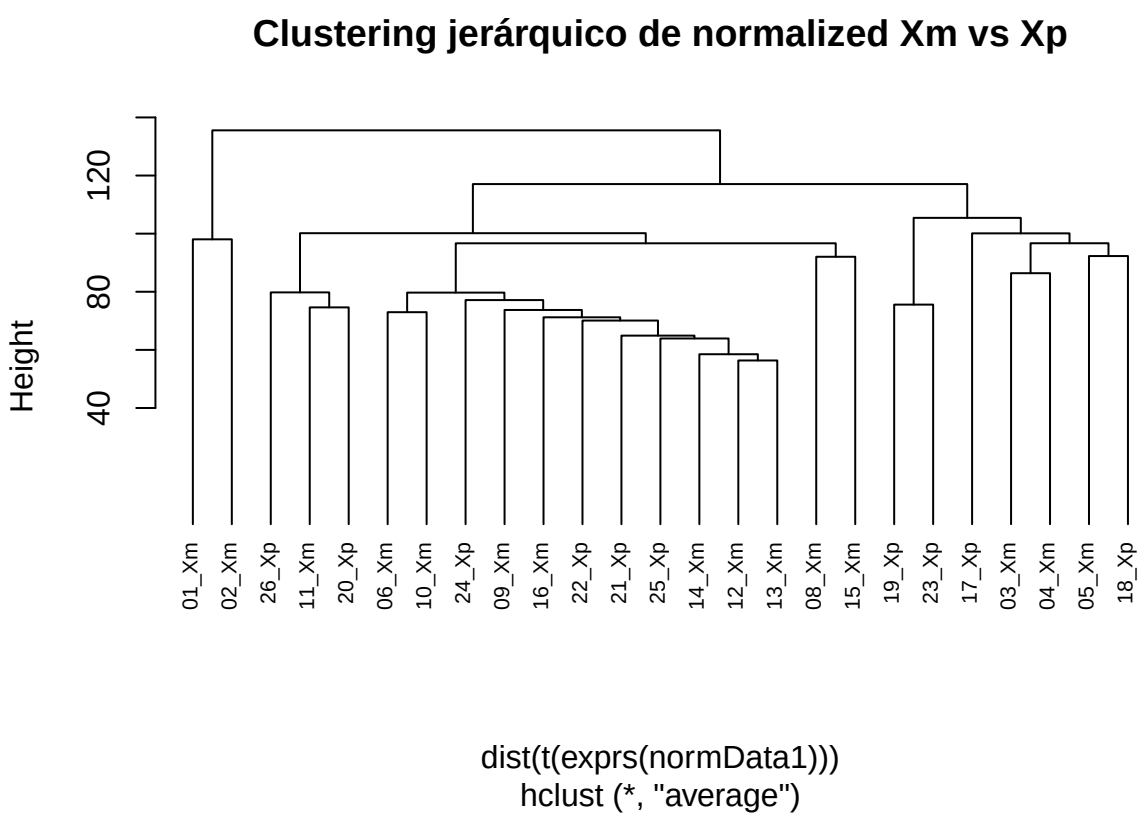
```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 25 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
```

```
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

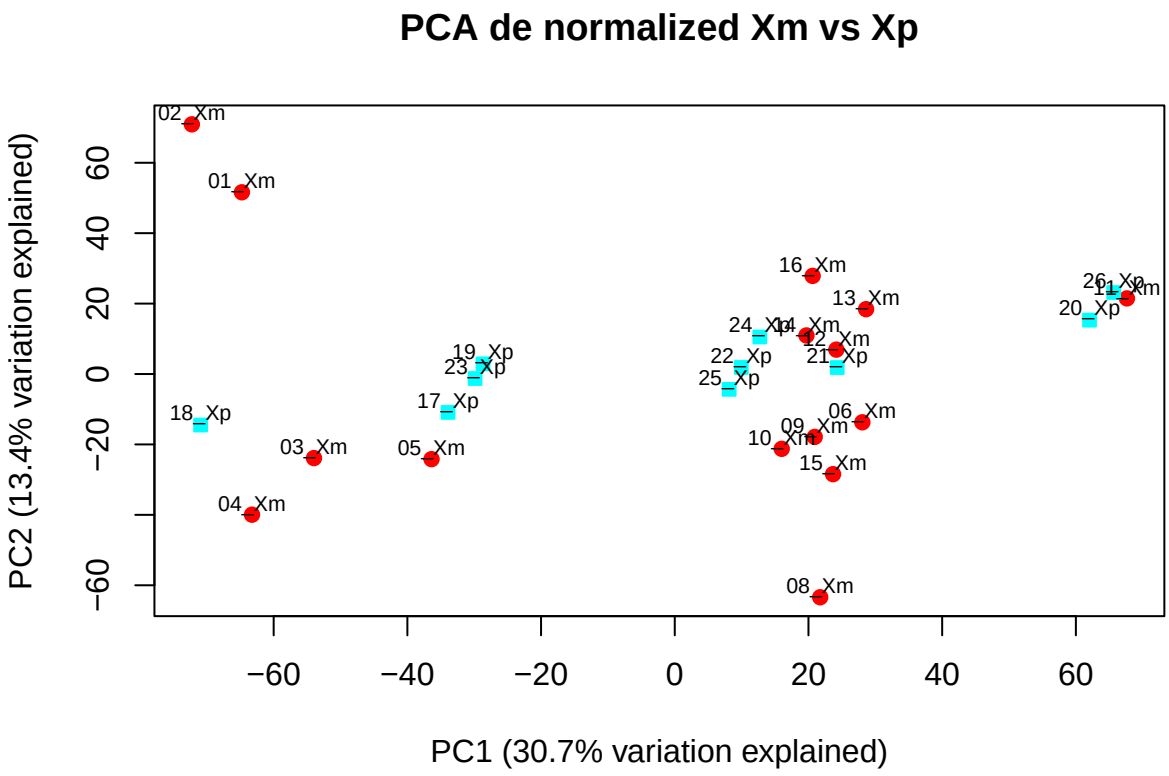
Analizando la calidad de los datos tras la normalización:



Además, se muestra el **clustering jerárquico** de las muestras normalizadas para analizar como se agrupan.



Representando el PCA de los datos normalizados:



Volviendo a realizar el informe de control de calidad para ver si ya no existe ninguna muestra con probelmas de valores atípicos:

```
arrayQualityMetrics(normData1, reporttitle = "QC_NormData1.1", force = TRUE)
```

```
annotation(normData1) <- "hgu133plus2.db"
normData_filtered1 <- nsFilter(normData1, var.func = IQR, var.cutoff = 0.75, var.filter = TRUE, require
```

Filtrado no específico

```
##
```

```
normData_filtered1
```

```
## $eset
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5206 features, 25 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
```

```
## phenoData
##   rowNames: 1 2 ... 25 (25 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: hgu133plus2.db
##
## $filter.log
## $filter.log$numDupsRemoved
## [1] 22267
##
## $filter.log$numLowVar
## [1] 15618
##
## $filter.log$numRemoved.ENTREZID
## [1] 11574
##
## $filter.log$feature.exclude
## [1] 10
```

Analizando los resultados obtenidos, se han eliminado los siguientes genes:

- 22267 valores duplicados
- 15618 genes que tienen baja variabilidad
- 11574 genes que no tienen ID Entrez

Con este filtraje no específico se ha obtenido un dataset con menor número de genes, habiendo descartado los genes que con una alta probabilidad no van a estar relacionados con el estudio.

Los genes restantes se han almacenado en la variable `FilteredEset`.

```
filtered_normData1 <- normData_filtered1$eset
filteredData1 <- exprs(filtered_normData1)
colnames(filteredData1) <- pData(normData_filtered1$eset)$ShortName
```

Análisis

La matríz de diseño El primer paso para el análisis basado en modelos lineales es crear la matriz de diseño. Básicamente es una tabla que describe la asignación de cada muestra a un grupo o condición experimental. Tiene tantas filas como muestras y tantas columnas como grupos, ya que en este caso se necesita el modelo de un factor, teniendo 2 tipos de muestras que se diferencian solo en un único factor.

```
treat1 <- pData(filtered_normData1)$Grupos

treat1 <- factor(treat1)
design1 <- model.matrix(~0 + treat1)

rownames(design1) <- sampleNames1
colnames(design1) <- levels(treat1)

design1
```

```
##       Xm Xp
## 01_Xm  1  0
## 02_Xm  1  0
## 03_Xm  1  0
## 04_Xm  1  0
## 05_Xm  1  0
## 06_Xm  1  0
## 08_Xm  1  0
## 09_Xm  1  0
## 10_Xm  1  0
## 11_Xm  1  0
## 12_Xm  1  0
## 13_Xm  1  0
## 14_Xm  1  0
## 15_Xm  1  0
## 16_Xm  1  0
## 17_Xp  0  1
## 18_Xp  0  1
## 19_Xp  0  1
## 20_Xp  0  1
## 21_Xp  0  1
## 22_Xp  0  1
## 23_Xp  0  1
## 24_Xp  0  1
## 25_Xp  0  1
## 26_Xp  0  1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$treat1
## [1] "contr.treatment"
```

La matríz de contrastes: definición de comparaciones La matriz de contrastes se utiliza para describir las comparaciones entre grupos. Consta de tantas columnas como comparaciones y tantas filas como grupos. Una comparación entre grupos se representa mediante un "1" y un "-1" en las filas de grupos a comparar.

```
cont.matrix1 <- makeContrasts(Xm.vs.Xp = Xm - Xp, levels = design1)
comparisonName1 <- "Efecto de la monosomía X heredada de la madre y el padre"
cont.matrix1
```

```
##       Contrasts
## Levels Xm.vs.Xp
##     Xm        1
##     Xp       -1
```

Estimación del modelo y selección de genes Una vez definida la matriz de diseño y los contrastes, se puede proceder a estimar el modelo, estimar los contrastes y realizar las pruebas de significación que llevarán a decidir, para cada gen y cada comparación, si pueden considerarse de expresión diferencial. Para ello, se utilizará el paquete `limma`.

Toda la información relevante para la posterior exploración de los resultados se almacena en un objeto llamado `fit.main2`.

```
fit1 <- lmFit(filteredData1, design1)
fit.main1 <- contrasts.fit(fit1, cont.matrix1)
fit.main1 <- eBayes(fit.main1)
```

El paquete `limma` implementa la función `topTable` que contiene, para un contraste dado, una lista de genes ordenados de menor a mayor p-valor que pueden considerarse de mayor a menor expresión diferencial.

Además, la instrucción `topTable2` puede aplicar un filtro automático, basado en dos criterios distintos, "log fold change (lfc)" y "p.value". En este caso, se va a emplear el valor de lfc 1.2 y el p valor de 0.05.

```
topTab1 <- topTable(fit.main1, number = nrow(fit.main1), coef = "Xm.vs.Xp", adjust = "fdr", lfc = 1.2, p

dim(topTab1)
```

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

Análisis de expresión diferencial genómica entre genotipos de Síndrome de Turner X (X = Xm y Xp) y las muestras control (XX)

En este caso, no se van a comentar todos los pasos ya que el inicio del análisis es idéntico al análisis anterior.

Creación del objeto "targets"

En la carpeta `Data` donde se encuentran los archivos .CEL, se ha creado un archivo .csv llamado "targets2". Este archivo contiene la información de las diferentes muestras del experimento para poder crear un objeto *AnnotatedDataFrame*.

fileName	grupos	ShortName	Colors
GSM1134016	XX	16_XX	blue
GSM1134017	XX	17_XX	blue
GSM1134018	XX	18_XX	blue
GSM1134019	XX	19_XX	blue
GSM1134020	XX	20_XX	blue
GSM1134021	XX	21_XX	blue
GSM1134022	XX	22_XX	blue
GSM1134023	XX	23_XX	blue
GSM1134024	XX	24_XX	blue
GSM1134025	XX	25_XX	blue
GSM1134026	X	26_X	red
GSM1134027	X	27_X	red
GSM1134028	X	28_X	red
GSM1134029	X	29_X	red
GSM1134030	X	30_X	red
GSM1134031	X	31_X	red
GSM1134032	X	32_X	red
GSM1134033	X	33_X	red
GSM1134034	X	34_X	red
GSM1134035	X	35_X	red

Figura 2: Contenido del inicio del archivo targets2.csv

Carga y lectura de los datos

```
targetsDF2 <- read.csv2(file = file.path(dataDir, "targets2.csv"), header = TRUE, sep = ";")

sampleNames2 <- as.character(targetsDF2$ShortName)
sampleColor2 <- as.character(targetsDF2$Colors)

targets2 <- AnnotatedDataFrame(targetsDF2)

targetsDF2
```

```
##          fileName Grupos ShortName Colors
## 1  GSM1134016.CEL     XX     01_XX   blue
## 2  GSM1134017.CEL     XX     02_XX   blue
## 3  GSM1134018.CEL     XX     03_XX   blue
## 4  GSM1134019.CEL     XX     04_XX   blue
## 5  GSM1134020.CEL     XX     05_XX   blue
## 6  GSM1134021.CEL     XX     06_XX   blue
## 7  GSM1134022.CEL     XX     07_XX   blue
## 8  GSM1134023.CEL     XX     08_XX   blue
## 9  GSM1134024.CEL     XX     09_XX   blue
## 10 GSM1134025.CEL     XX     10_XX   blue
## 11 GSM1134026.CEL     XO     11_XO    red
## 12 GSM1134027.CEL     XO     12_XO    red
## 13 GSM1134028.CEL     XO     13_XO    red
## 14 GSM1134029.CEL     XO     14_XO    red
## 15 GSM1134030.CEL     XO     15_XO    red
## 16 GSM1134031.CEL     XO     16_XO    red
## 17 GSM1134032.CEL     XO     17_XO    red
## 18 GSM1134033.CEL     XO     18_XO    red
## 19 GSM1134034.CEL     XO     19_XO    red
## 20 GSM1134035.CEL     XO     20_XO    red
## 21 GSM1134036.CEL     XO     21_XO    red
## 22 GSM1134037.CEL     XO     22_XO    red
## 23 GSM1134038.CEL     XO     23_XO    red
## 24 GSM1134039.CEL     XO     24_XO    red
## 25 GSM1134040.CEL     XO     25_XO    red
## 26 GSM1134041.CEL     XO     26_XO    red
## 27 GSM1134042.CEL     XO     27_XO    red
## 28 GSM1134043.CEL     XO     28_XO    red
## 29 GSM1134044.CEL     XO     29_XO    red
## 30 GSM1134045.CEL     XO     30_XO    red
## 31 GSM1134046.CEL     XO     31_XO    red
## 32 GSM1134047.CEL     XO     32_XO    red
## 33 GSM1134048.CEL     XO     33_XO    red
## 34 GSM1134049.CEL     XO     34_XO    red
## 35 GSM1134050.CEL     XO     35_XO    red
## 36 GSM1134051.CEL     XO     36_XO    red
```

A continuación, se guardan los nombres de los archivos .CEL guardados en el dataframe `targetsDF2` en un nuevo objeto llamado `CELfiles2` y se cargan los archivos del directorio que tengan el mismo nombre que los guardados en ese objeto. Estos archivos se almacenan en un objeto ExpressionFeatureSet llamado `rawData2`.

```
CELfiles2 <- targetsDF2$fileName
rawData2 <- read.celfiles(file.path(dataDir, CELfiles2), phenoData = targets2)
```

```
## Reading in : E:/TFM/Data/GSM1134016.CEL
## Reading in : E:/TFM/Data/GSM1134017.CEL
## Reading in : E:/TFM/Data/GSM1134018.CEL
## Reading in : E:/TFM/Data/GSM1134019.CEL
## Reading in : E:/TFM/Data/GSM1134020.CEL
## Reading in : E:/TFM/Data/GSM1134021.CEL
## Reading in : E:/TFM/Data/GSM1134022.CEL
## Reading in : E:/TFM/Data/GSM1134023.CEL
## Reading in : E:/TFM/Data/GSM1134024.CEL
## Reading in : E:/TFM/Data/GSM1134025.CEL
## Reading in : E:/TFM/Data/GSM1134026.CEL
## Reading in : E:/TFM/Data/GSM1134027.CEL
## Reading in : E:/TFM/Data/GSM1134028.CEL
## Reading in : E:/TFM/Data/GSM1134029.CEL
## Reading in : E:/TFM/Data/GSM1134030.CEL
## Reading in : E:/TFM/Data/GSM1134031.CEL
## Reading in : E:/TFM/Data/GSM1134032.CEL
## Reading in : E:/TFM/Data/GSM1134033.CEL
## Reading in : E:/TFM/Data/GSM1134034.CEL
## Reading in : E:/TFM/Data/GSM1134035.CEL
## Reading in : E:/TFM/Data/GSM1134036.CEL
## Reading in : E:/TFM/Data/GSM1134037.CEL
## Reading in : E:/TFM/Data/GSM1134038.CEL
## Reading in : E:/TFM/Data/GSM1134039.CEL
## Reading in : E:/TFM/Data/GSM1134040.CEL
## Reading in : E:/TFM/Data/GSM1134041.CEL
## Reading in : E:/TFM/Data/GSM1134042.CEL
## Reading in : E:/TFM/Data/GSM1134043.CEL
## Reading in : E:/TFM/Data/GSM1134044.CEL
## Reading in : E:/TFM/Data/GSM1134045.CEL
## Reading in : E:/TFM/Data/GSM1134046.CEL
## Reading in : E:/TFM/Data/GSM1134047.CEL
## Reading in : E:/TFM/Data/GSM1134048.CEL
## Reading in : E:/TFM/Data/GSM1134049.CEL
## Reading in : E:/TFM/Data/GSM1134050.CEL
## Reading in : E:/TFM/Data/GSM1134051.CEL
```

```
rawData2
```

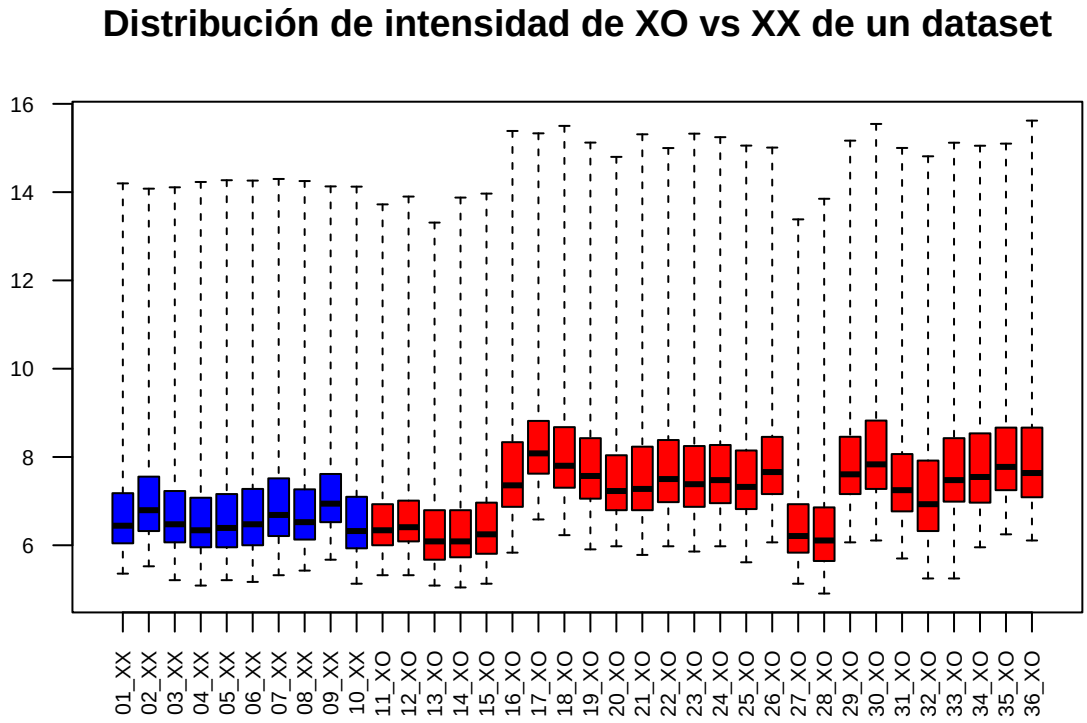
```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 1354896 features, 36 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
```

```
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

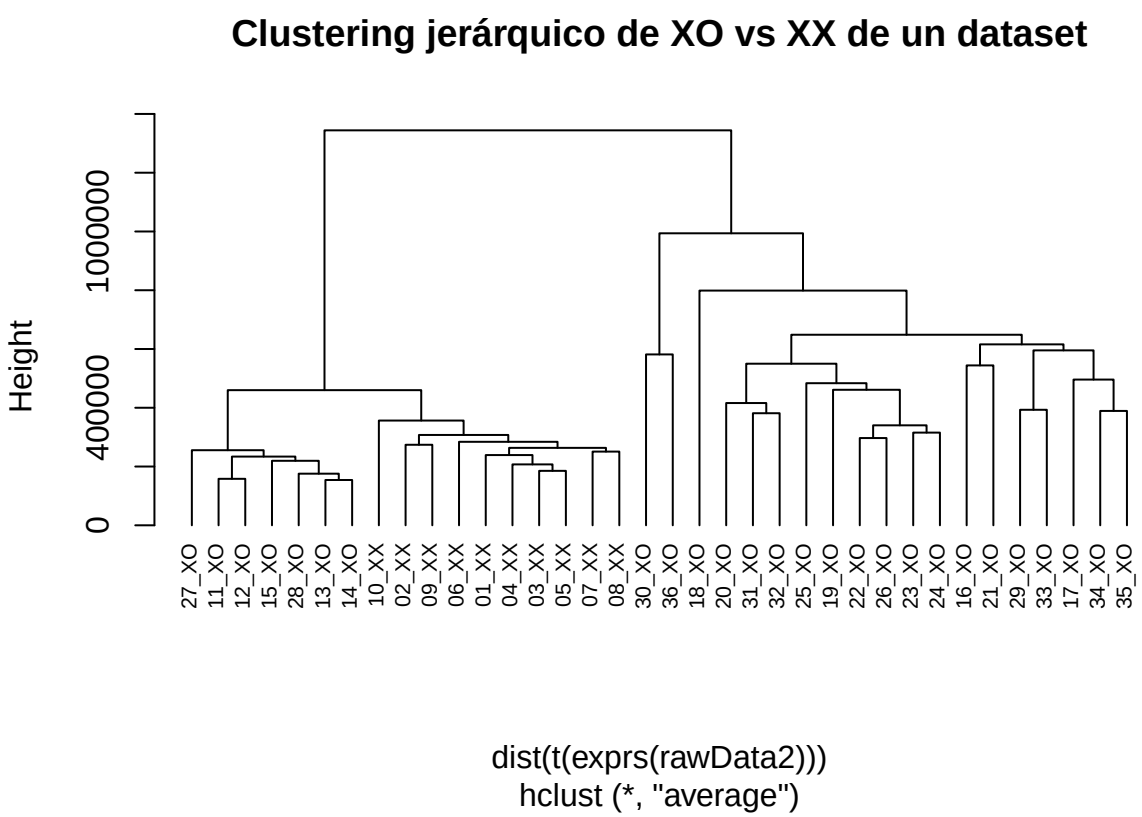
Como se puede observar, el archivo contiene 36 muestras, ya que en este caso se tienen en cuenta todas las muestras. Además, `Annotation` indica el paquete de anotaciones necesario para poder realizar este análisis, el mismo que se ha utilizado anteriormente.

Preprocesado de los datos

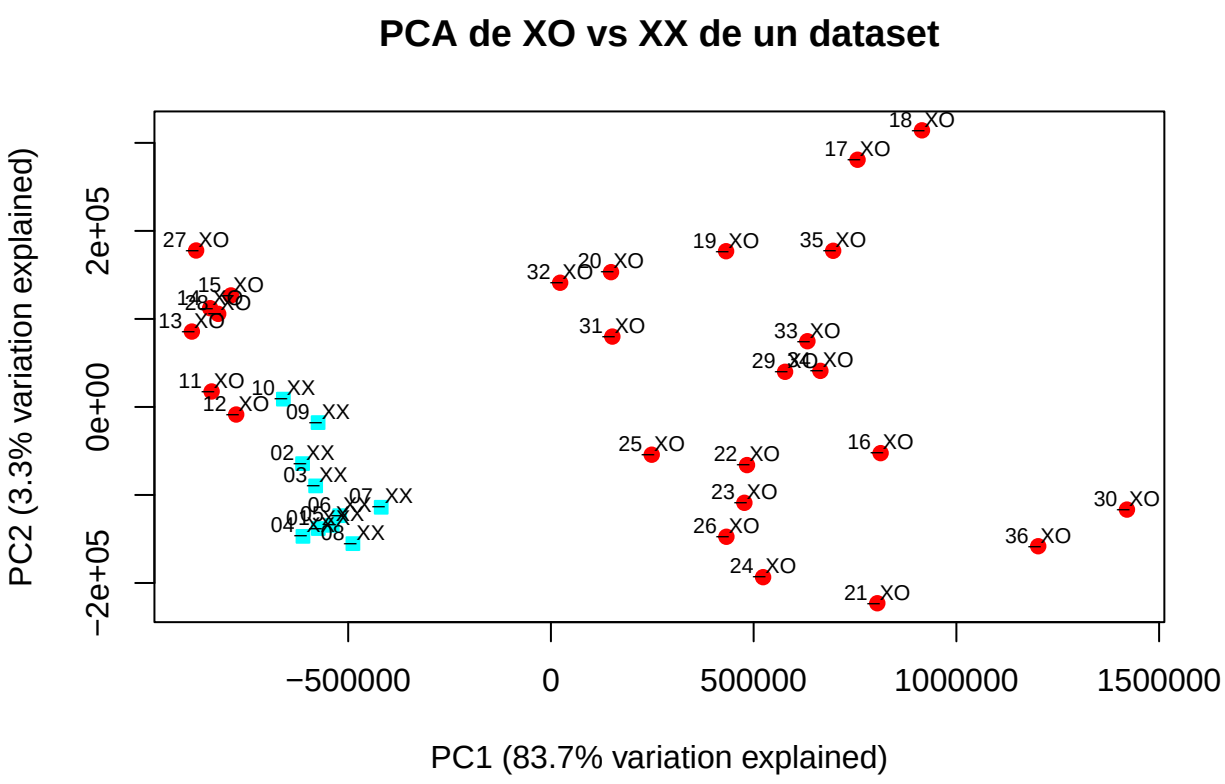
Exploración y control de calidad Mediante un **boxplot** se muestra como es la distribución de los valores.



Mediante un **clustering jerárquico** se muestra como se agrupan estas muestras.



En el **análisis de componentes principales**, se buscan las principales fuentes de variabilidad en los datos reduciendo las dimensiones.



Control de calidad con el paquete arrayQualityMetrics Con este paquete se obtienen los mismos gráficos que los anteriores pero todos a la vez en un archivo. Además, se observa si existe alguna muestra que tenga algún problema de outliers.

```
arrayQualityMetrics(rawData2, reporttitle = "QC_RawData2", force = TRUE)
```

Normalización Antes de comenzar con el análisis de expresión diferencial y vistos los problemas que muestran algunas de las muestras, es necesario hacer normalizar los arrays con el método RMA.

```
normData2 <- rma(rawData2)
```

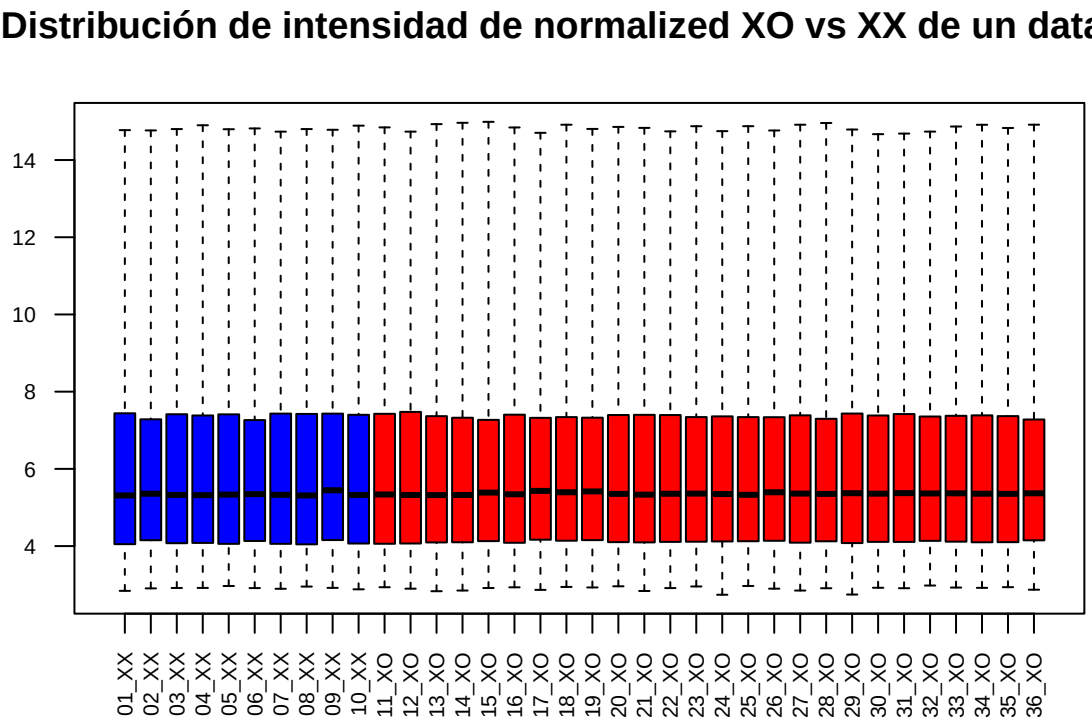
```
## Background correcting
## Normalizing
## Calculating Expression
```

```
normData2
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 36 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: exprs dates
```

```
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

Analizando la calidad de los datos tras la normalización:



Obteniendo el informe de control de calidad:

```
arrayQualityMetrics(normData2, reporttitle = "QC_NormData2", force = TRUE)
```

```
annotation(normData2) <- "hgu133plus2.db"
normData_filtered2 <- nsFilter(normData2, var.func = IQR, var.cutoff = 0.75, var.filter = TRUE, require

normData_filtered2
```

Filtrado no específico

```
## $eset
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5206 features, 36 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 36 (36 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: hgu133plus2.db
##
## $filter.log
## $filter.log$numDupsRemoved
## [1] 22267
##
## $filter.log$numLowVar
## [1] 15618
##
## $filter.log$numRemoved.ENTREZID
## [1] 11574
##
## $filter.log$feature.exclude
## [1] 10
```

Analizando los resultados obtenidos, se han eliminado los siguientes genes:

- 22267 valores duplicados
- 15618 genes que tienen baja variabilidad
- 11574 genes que no tienen ID Entrez

Con este filtraje no específico se ha obtenido un dataset con menor número de genes, habiendo descartado los genes que con una alta probabilidad no van a estar relacionados con el estudio.

Los genes restantes se han almacenado en la variable `FilteredEset`.

```
filtered_normData2 <- normData_filtered2$eset
filteredData2 <- exprs(filtered_normData2)
colnames(filteredData2) <- pData(normData_filtered2$eset)$ShortName
```

Análisis

La matríz de diseño El primer paso para el análisis basado en modelos lineales es crear la matriz de diseño. Básicamente es una tabla que describe la asignación de cada muestra a un grupo o condición experimental. Tiene tantas filas como muestras y tantas columnas como grupos, ya que en este caso se necesita el modelo de un factor, teniendo 2 tipos de muestras que se diferencian solo en un único factor.

```r
treat2 <- pData(filtered_normData2)$Grupos

treat2 <- factor(treat2)
design2 <- model.matrix(~0 + treat2)

rownames(design2) <- sampleNames2
colnames(design2) <- levels(treat2)

design2
```

```
##       XO XX
## 01_XX  0  1
## 02_XX  0  1
## 03_XX  0  1
## 04_XX  0  1
## 05_XX  0  1
## 06_XX  0  1
## 07_XX  0  1
## 08_XX  0  1
## 09_XX  0  1
## 10_XX  0  1
## 11_XO  1  0
## 12_XO  1  0
## 13_XO  1  0
## 14_XO  1  0
## 15_XO  1  0
## 16_XO  1  0
## 17_XO  1  0
## 18_XO  1  0
## 19_XO  1  0
## 20_XO  1  0
## 21_XO  1  0
## 22_XO  1  0
## 23_XO  1  0
## 24_XO  1  0
## 25_XO  1  0
## 26_XO  1  0
## 27_XO  1  0
## 28_XO  1  0
## 29_XO  1  0
## 30_XO  1  0
## 31_XO  1  0
## 32_XO  1  0
## 33_XO  1  0
## 34_XO  1  0
## 35_XO  1  0
## 36_XO  1  0
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$treat2
## [1] "contr.treatment"
```

La matríz de contrastes: definición de comparaciones La matriz de contrastes se utiliza para describir las comparaciones entre grupos. Consta de tantas columnas como comparaciones y tantas filas como grupos. Una comparación entre grupos se representa mediante un "1" y un "-1" en las filas de grupos a comparar.

```
cont.matrix2 <- makeContrasts(XO.vs.XX = XO - XX, levels = design2)
comparisonName2 <- "Efecto de la monosomía XO frente al control XX"
cont.matrix2
```

```
##       Contrasts
## Levels XO.vs.XX
##     XO        1
##     XX       -1
```

Estimación del modelo y selección de genes Una vez definida la matriz de diseño y los contrastes, se puede proceder a estimar el modelo, estimar los contrastes y realizar las pruebas de significación que llevarán a decidir, para cada gen y cada comparación, si pueden considerarse de expresión diferencial. Para ello, se utilizará el paquete `limma`.

Toda la información relevante para la posterior exploración de los resultados se almacena en un objeto llamado `fit.main2`.

```
fit2 <- lmFit(filteredData2, design2)
fit.main2 <- contrasts.fit(fit2, cont.matrix2)
fit.main2 <- eBayes(fit.main2)
```

El paquete `limma` implementa la función `topTable` que contiene, para un contraste dado, una lista de genes ordenados de menor a mayor p-valor que pueden considerarse de mayor a menor expresión diferencial.

Además, la instrucción `topTable2` puede aplicar un filtro automático, basado en dos criterios distintos, "log fold change (lfc)" y "p.value". En este caso, se va a emplear el valor de lfc 1.2 y el p valor de 0.05.

```
topTab2 <- topTable(fit.main2, number = nrow(fit.main2), coef = "XO.vs.XX", adjust = "fdr", lfc = 1.2, p

dim(topTab2)
```

```
## [1] 75  6
```

Con estos valores se obtienen 75 genes diferencialmente expresados entre los dos tipos de muestra.

Obtención de la lista de genes expresados diferencialmente Echando un vistazo a las primeras líneas del topTable, se pueden observar los genes que más cambian su expresión entre los dos grupos de muestras, ordenados por su p-valor (de menor a mayor).

```
head(topTab2)
```

```
##                 logFC  AveExpr          t      P.Value    adj.P.Val        B
## 224588_at    -8.433211 7.318187 -25.251684 1.324477e-26 6.895229e-23 47.40560
## 231592_at    -2.063240 4.606351 -14.370679 1.333327e-17 3.470649e-14 29.40494
## 203990_s_at  -1.414922 6.120874 -11.586509 1.644533e-14 2.853812e-11 22.73807
## 230532_at    -1.276882 8.079148  -9.353481 1.006671e-11 1.048145e-08 16.60723
## 76897_s_at   -2.034131 6.318440  -9.038845 2.616154e-11 2.269949e-08 15.68785
## 1558775_s_at -1.445004 6.050952  -8.429561 1.719041e-10 9.943695e-08 13.87167
```

Como se puede observar, la primera columna del `topTable2` contiene el ID del fabricante (*Affymetrix*) para cada conjunto de sondas. El siguiente paso es adivinar qué gen corresponde a cada ID de Affymetrix, utilizando para ello la anotación de genes.

Anotación de los genes Una vez que se ha obtenido la tabla con los genes diferencialmente expresados, denominada `topTab2` es útil proporcionar información adicional sobre las características que se han seleccionado. Este proceso se denomina "anotación" y lo que hace es buscar información para asociar identificadores que aparecen en la tabla superior, normalmente correspondientes a conjuntos de sondas o transcritos dependiendo del tipo de array, con nombres más familiares o intuitivos como pueden ser el símbolo del gen (`SYMBOL`) o el identificador Entrez Gene (`ENTREZID`).

```
anotaciones2 <- AnnotationDbi::select(hgu133plus2.db, keys = rownames(filteredData2), columns = c("ENTRE
head(anotaciones2)
```

```
##       PROBEID  ENTREZID   SYMBOL
## 1 216705_s_at       100      ADA
## 2 242879_x_at     10000     AKT3
## 3   222922_at     10008    KCNE3
## 4   225220_at 100093630    SNHG8
## 5 210458_s_at     10010     TANK
## 6   235798_at 100113407 TMEM170B
```

Una vez se tienen las anotaciones de todos los genes de este array, se combinaran estos nombres con los genes obtenidos en la `topTab2`, para así identificarlos con su Entrez ID y el símbolo del gen.

```
topTabAnotada2 <- topTab2 %>%
    mutate(PROBEID = rownames(topTab2)) %>%
    left_join(anotaciones2) %>%
    arrange(P.Value) %>%
    dplyr::select(7, 8, 9, 1:6)

head(topTabAnotada2)
```

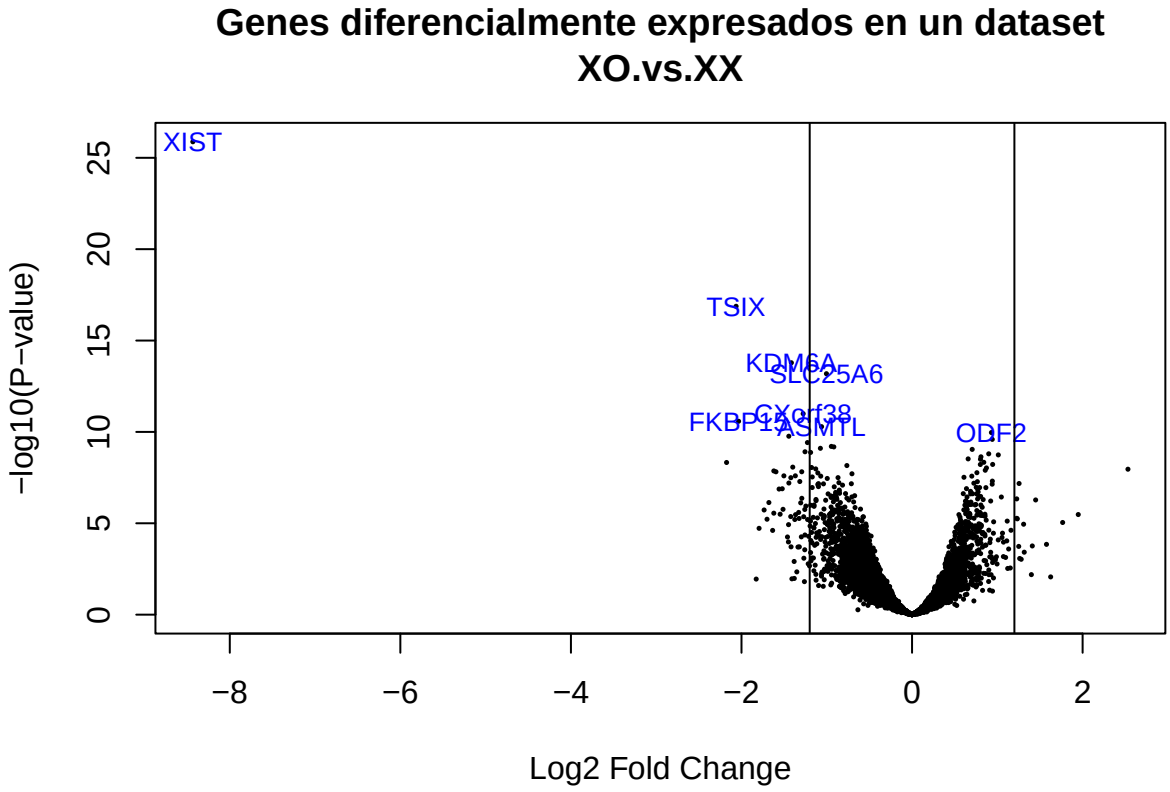
```
##        PROBEID ENTREZID  SYMBOL     logFC  AveExpr          t      P.Value
## 1    224588_at     7503    XIST -8.433211 7.318187 -25.251684 1.324477e-26
## 2    231592_at     9383    TSIX -2.063240 4.606351 -14.370679 1.333327e-17
## 3  203990_s_at     7403   KDM6A -1.414922 6.120874 -11.586509 1.644533e-14
## 4    230532_at   159013 CXorf38 -1.276882 8.079148  -9.353481 1.006671e-11
## 5   76897_s_at    23307  FKBP15 -2.034131 6.318440  -9.038845 2.616154e-11
## 6 1558775_s_at     8439   NSMAF -1.445004 6.050952  -8.429561 1.719041e-10
##      adj.P.Val        B
## 1 6.895229e-23 47.40560
## 2 3.470649e-14 29.40494
## 3 2.853812e-11 22.73807
## 4 1.048145e-08 16.60723
## 5 2.269949e-08 15.68785
## 6 9.943695e-08 13.87167
```

Se almacenan los genes epresados diferencialmente entre los dos tipos de muestras en un archivo .xlsx para, después, poder comparar los genes obtenidos con los anotados anteriormente en la búsqueda bibliográfica.

```
ruta_GSE46687 <- file.path(workingDir, "genes_diferenciales_GSE46687.xlsx")

write.xlsx(topTabAnotada2, file = ruta_GSE46687)
```

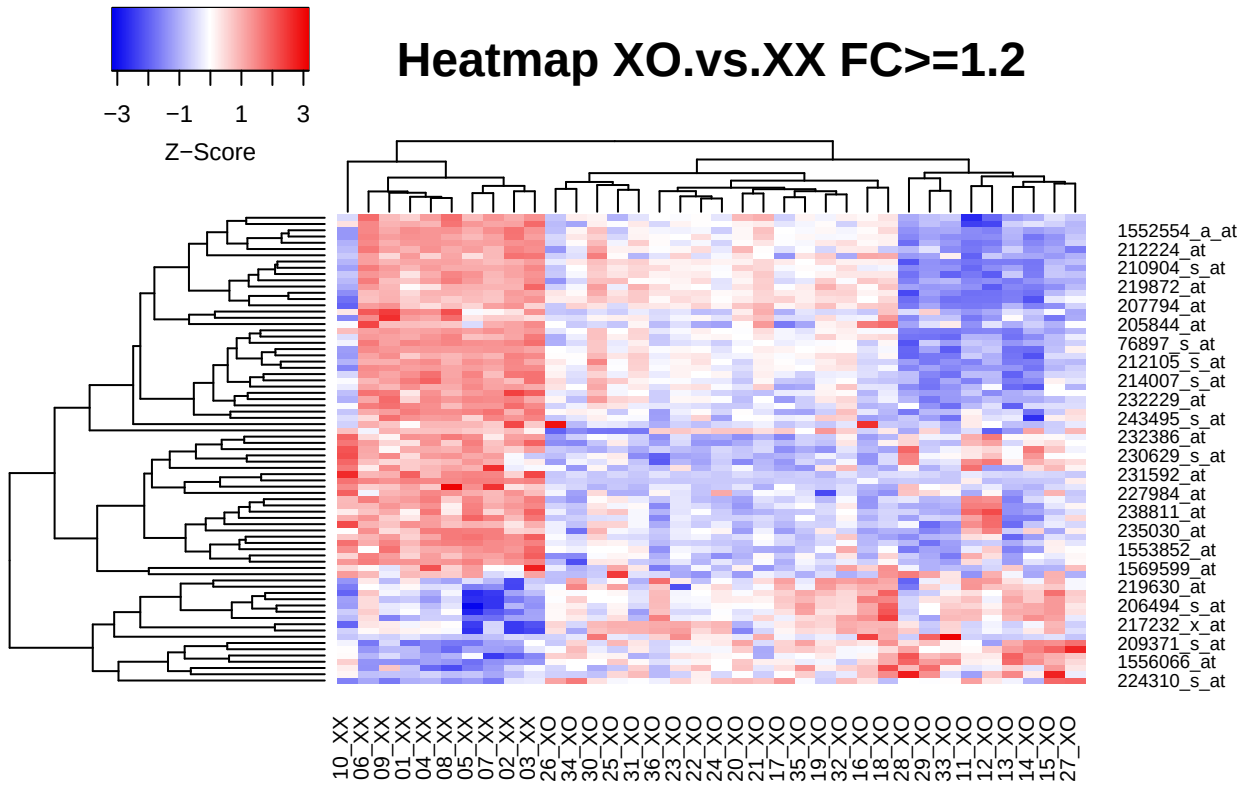
Por lo tanto, en el objeto `topTabAnotada2` se ha obtenido una lista de genes diferencialmente expresados más legible.

Visualización de los genes diferencialmente expresados Puede obtenerse una visualización de la expresión diferencial global mediante **diagramas de volcán (volcan plot)**. Estos gráficos muestran si hay muchos o pocos genes con un gran cambio de pliegue y significativamente expresados o si este número es bajo. Estos gráficos representan en el eje X los cambios de expresión en escala logarítmica (efecto biológico) y en el eje Y el logaritmo negativo del valor p (efecto estadístico).



Los genes seleccionados con expresión diferencial también pueden visualizarse mediante un **mapa de calor (heatmap)**. Estos gráficos utilizan paletas de colores para resaltar distintos valores -en este caso, expresiones significativamente diferenciales positivas (regulación al alza) o negativas (regulación a la baja).

Para hacer el mapa de calor se emplearán los genes seleccionados en los pasos anteriores.



Estudio de significación biológica Una vez obtenida una lista de genes que caracteriza la diferencia entre dos condiciones, hay que interpretarla, aunque esto requiere una buena comprensión del problema biológico subyacente.

Con este objetivo, este tipo de análisis busca establecer si, dada una lista de genes seleccionados por expresarse diferencialmente entre dos condiciones, las funciones, procesos biológicos o vías moleculares que los caracterizan aparecen en esta lista con mayor frecuencia que entre el resto de genes analizados.

Para llevarlo a cabo, se empleará el **análisis de enriquecimiento básico** y se necesitarán dos colecciones de genes:

- La lista seleccionada
- El universo de genes es decir todos los genes que se han incluído en el análisis (todos los del chip)

La mayoría de programas necesitan que los identificadores de los genes sean en formato "ENTREZ" por lo que se preparan ambas listas a la vez.

```
# Se extraen todas las sondas y se convierten a EntrezID
probesUniverse2 <- rownames(filteredData2)
entrezUniverse <- AnnotationDbi::select(hgu133plus2.db, probesUniverse2, "ENTREZID")$ENTREZID

# Se extraen las sondas de los datos seleccionados y se convierten a EntrezID
topProbes2 <- rownames(selectedData2)
entrezTop2 <- AnnotationDbi::select(hgu133plus2.db, topProbes2, "ENTREZID")$ENTREZID
```

```
# Eliminación de posibles duplicados
topGenes2 <- entrezTop2[!duplicated(entrezTop2)]
entrezUniverse <- entrezUniverse[!duplicated(entrezUniverse)]
```

Por lo tanto, se tienen 2 listas Entrez:

- **topGenes2:** con los Entrez de las sondas seleccionadas.
- **entrezUniverse:** con los Entrez de todas las sondas del array.

Se pueden utilizar muchos paquetes para realizar un análisis de enriquecimiento genético. Cada uno de ellos realiza un análisis ligeramente diferente, pero las ideas subyacentes son las mismas.

```
GOparams2 <- new("GOHyperGParams", geneIds = topGenes2, universeGeneIds = entrezUniverse,
    annotation = "hgu133plus2.db", ontology = "BP", pvalueCutoff = 0.01)

# Test de Fisher

GOhyper2 <- hyperGTest(GOparams2)

head(summary(GOhyper2), n = 10)
```

```
##        GOBPID       Pvalue OddsRatio    ExpCount Count Size
## 1  GO:0010033 3.918571e-06  3.283419 13.51483191    30  918
## 2  GO:0042221 1.136401e-05  3.003948 16.60646012    33 1128
## 3  GO:0070887 9.575911e-05  2.791772 12.60206548    26  856
## 4  GO:2001020 1.297143e-04  6.132448  1.54581411     8  105
## 5  GO:0070988 1.741286e-04 17.730159  0.29444078     4   20
## 6  GO:0030185 2.135505e-04       Inf  0.02944408     2    2
## 7  GO:0071557 2.135505e-04       Inf  0.02944408     2    2
## 8  GO:0006338 2.213438e-04  4.998851  2.11997363     9  144
## 9  GO:0006950 2.371470e-04  2.472543 19.09448473    33 1297
## 10 GO:0140719 2.405288e-04 34.984375  0.13249835     3    9
##                                          Term
## 1                  response to organic substance
## 2                           response to chemical
## 3         cellular response to chemical stimulus
## 4  regulation of response to DNA damage stimulus
## 5                                  demethylation
## 6                         nitric oxide transport
## 7                   histone H3-K27 demethylation
## 8                           chromatin remodeling
## 9                             response to stress
## 10        constitutive heterochromatin formation
```

Estas son las categorías del Gene Ontology que están más enriquecidas y el p-valor que corresponde a cada una. El valor `OddsRatio` muestra cuantas veces más abundante es ese Gene Ontology de lo que se esperaría. Por lo tanto, cuanto mayor sea este número, esta catergoría será más importante, aunque en este caso se obtienen números bastante elevados.

```r
dim(summary(GOhyper2))
```

```
## [1] 88  7
```

En total se han obtenido 88 categorías GO diferentes. Antes de finalizar el análisis, se muestran gráficamente las funciones principales de los genes diferencialmente expresados.

```r
# Filtra genes con ajuste de p-valor menor a 0.05
whichGenes2 <- topTab2["adj.P.Val"] < 0.05

# Obtiene los identificadores de fila (nombres de genes) que cumplen el criterio
selectedIDs2 <- rownames(topTab2)[whichGenes2]

# Obtén los identificadores ENTREZID correspondientes a los nombres de genes seleccionados
EntrezIDs2 <- AnnotationDbi:::select(hgu133plus2.db, selectedIDs2, c("ENTREZID"))

# Extrae la columna ENTREZID del resultado para obtener un vector de identificadores
EntrezIDs2 <- EntrezIDs2$ENTREZID

# Crea una lista con un solo elemento que contiene los identificadores ENTREZID seleccionados
listOfSelected2 <- list(EntrezIDs2)

# Obtiene el nombre de la variable (topTab2) y lo usa como nombre para la lista
topTabName2 <- deparse(substitute(topTab2))
names(listOfSelected2) <- topTabName2

# Imprime la longitud de la lista
sapply(listOfSelected2, length)
```

```
## topTab2
##      75
```

```r
# Obtiene los genes mapeados a términos GO (Gene Ontology) para Homo sapiens
mapped_genes2GO <- mappedkeys(org.Hs.egGO)

# Obtiene los genes mapeados a vías KEGG para Homo sapiens
mapped_genes2KEGG <- mappedkeys(org.Hs.egPATH)

# Combina los conjuntos de genes mapeados a términos GO y vías KEGG usando la unión
# Esto crea un conjunto único de genes que están asociados a términos GO o vías KEGG
mapped_genes <- union(mapped_genes2GO , mapped_genes2KEGG)
```

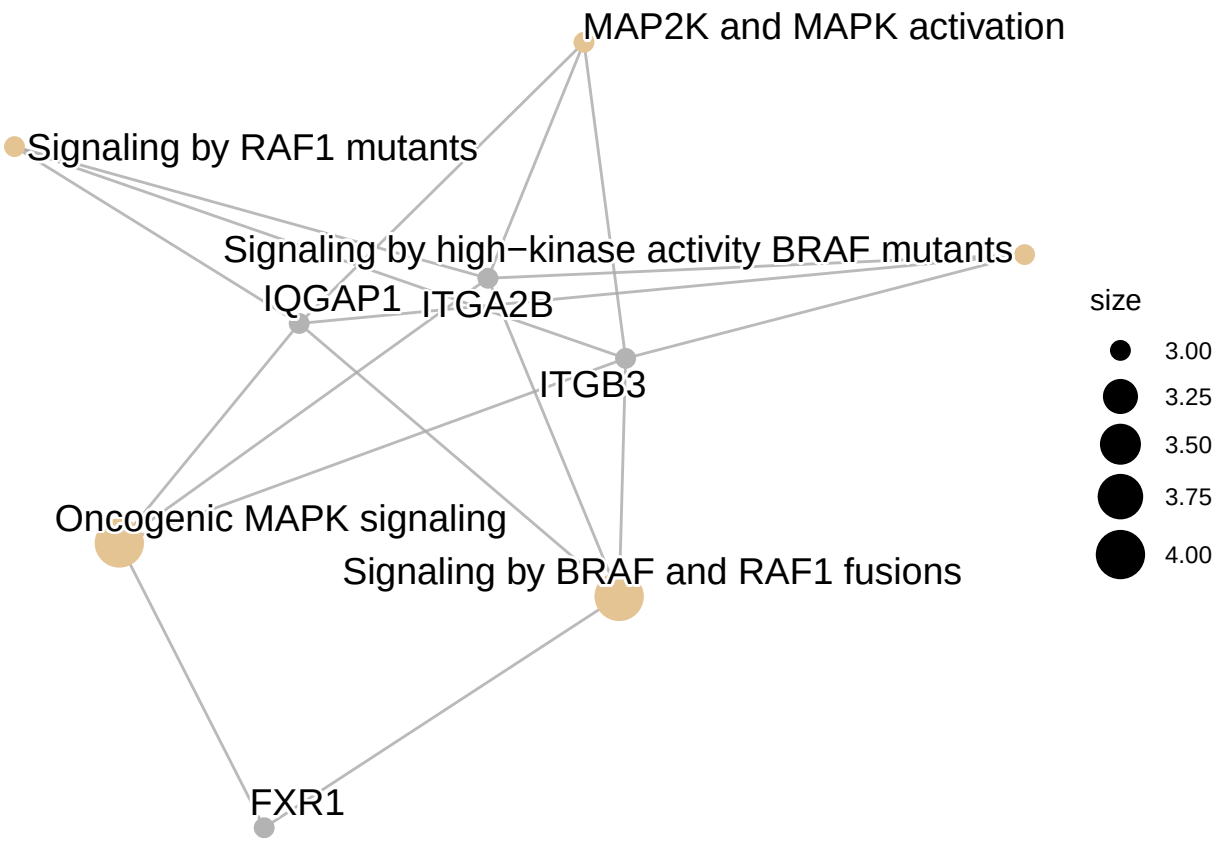
```r
# Crea una nueva lista de datos llamada listOfData2 copiando listOfSelected2
listOfData2 <- listOfSelected2

# Obtiene los nombres de las comparaciones de la nueva lista de datos
comparisonsNames2 <- names(listOfData2)

# Define el conjunto de genes universo (todos los genes asociados a términos GO o vías KEGG)
universe <- mapped_genes
```

```r
# Itera sobre cada elemento en la lista de datos
for (i in 1:length(listOfData2)){
  # Obtiene los genes para la comparación actual
  genesIn2 <- listOfData2[[i]]
  # Obtiene el nombre de la comparación actual
  comparison2 <- comparisonsNames2[i]
  # Realiza el enriquecimiento de vías para la comparación actual
  enrich.result2 <- enrichPathway(gene = genesIn2,
                                  pvalueCutoff = 0.1,
                                  readable = T,
                                  pAdjustMethod = "BH",
                                  organism = "human",
                                  universe = universe)
  # Imprime las primeras filas del resultado del enriquecimiento
  head(enrich.result2)
}
```

```r
enrichplot::cnetplot(enrich.result2, categorySize = "geneNum", schowCategory = 15, vertex.label.cex = 0
```

```
## Found more than one class "simpleUnit" in cache; using the first, from namespace 'ggbio'
```

```
## Also defined by 'hexbin'
```



Análisis diferencial genómico de dos bases de datos

En este informe se va a realizar un análisis de microarrays a partir de datos de dos estudios publicados y depositados en "*Gene Expression Omnibus*". Los estudios seleccionados son los codificados con los identificadores GEO **GSE46687** y **GSE58435** ambos incluídos en la plataforma *Affymetrix Human Genome U133 Plus 2.0 Array*.

Por un lado, el estudio **GSE46687** analiza la expresión génica diferencial en células mononucleares de sangre periférica (PBMC) de 45Xm, 45Xp y mujeres control 46XX mediante microarrays para investigar los cambios en la expresión génica de todo el genoma entre los pacientes con Síndrome de Turner con monosomía XO y las mujeres sin monosomía (sin tener en cuenta el tipo de herencia del cromosoma X).

Por otro lado, el estudio codificado como **GSE58435** tiene como objetivo comparar la expresión de ARNm en líquido amniótico en el 2º trimestre entre 5 fetos con Síndrome de Turner y 5 muestras control (Massingham, LJ *et al.*).

Selección del dataset

Se ha seleccionado dos dataset:

- **GSE46687:** trabajo de Cheng CM *et al.* llamado Gene Expression Profiling in 45X Turner Syndrome patients.
- **GSE58435:** estudio Amniotic fluid RNA gene expression profiling provides insights into the phenotype of Turner syndrome de Massingham, LJ *et al.*

Como se puede observar en las imágenes **Imagen 13** e **Imagen 14**, el primer estudio se compone de 36 muestras y el segundo de 10. Además, las anotaciones de los genes de ambos estudios se encuentran en *Affymetrix Human Genome U133 Plus 2.0 Array*.

- **XX**: 10 réplicas de muestras de control (*wild type*).
- **Xm**: 16 réplicas de muestras con Síndrome de Turner con el cromosoma X heredado de la madre
- **Xp**: 10 réplicas de muestras con Síndrome de Turner con el cromosoma X heredado del padre

Sin embargo, las muestras del estudio de Massingham, LJ *et al.* se dividen en dos grupos:

- Turner_AF: 5 réplicas de muestras con Síndrome de Turner

- Control_AF: 5 réplicas de muestras de control

Por lo tanto, las muestras de los dos estudios se agruparán en dos grupos: muestras de control y muestras de individuos con Síndrome de Turner. Además, como se ha analizado anteriormente, las muestras del estudio GSE46687 de tipo Xm y Xp se pueden agrupar todas en grupo. Teniendo todo esto en cuenta, se obtiene el siguiente esquema de muestras en total:

- **XX:** 15 réplicas de muestras de control
- **XO**: 31 réplicas de muestras con Síndrome de Turner (*wildtype*)

Se analizarán un total de 46 muestras.

	A	B	C	D
1	fileName	grupos	ShortName	Colors
2	GSM141102	X	1_X	red
3	GSM141102	X	2_X	red
4	GSM141102	X	3_X	red
5	GSM141102	X	4_X	red
6	GSM141102	X	5_X	red
7	GSM141102	XX	6_XX	blue
8	GSM141102	XX	7_XX	blue
9	GSM141102	XX	8_XX	blue
10	GSM141102	XX	9_XX	blue
11	GSM141103	XX	10_XX	blue
12	GSM113401	XX	11_XX	blue
13	GSM113401	XX	12_XX	blue
14	GSM113401	XX	13_XX	blue
15	GSM113401	XX	14_XX	blue
16	GSM113402	XX	15_XX	blue
17	GSM113402	XX	16_XX	blue
18	GSM113402	XX	17_XX	blue
19	GSM113402	XX	18_XX	blue
20	GSM113402	XX	19_XX	blue
21	GSM113402	XX	20_XX	blue
22	GSM113402	X	21_X	red
23	GSM113402	X	22_X	red
24	GSM113402	X	23_X	red
25	GSM113402	X	24_X	red

Figura 3: Contenido del inicio del archivo targets3.csv

Creación del objeto "targets"

En la carpeta `Data` donde se encuentran los archivos .CEL, se ha creado un archivo .csv llamado "targets3". Este archivo contiene la información de las diferentes muestras del experimento para poder crear un objeto *AnnotatedDataFrame*.

Carga y lectura de los datos

Para poder comenzar con el preprocesado de los datos, primero se lee el archivo `targets3.csv` y se almacenan las columnas `ShortName` y `Colors` de este archivo en dos nuevas variables para poder crear los gráficos posteriormente. Como se puede observar, el objeto `targetsDF3` contiene el dataframe que se ha creado anteriormente con la información de las muestras y algunos campos importantes para el análisis.

```
targetsDF3 <- read.csv2(file = file.path(dataDir, "targets3.csv"), header = TRUE, sep = ";")

sampleNames3 <- as.character(targetsDF3$ShortName)
sampleColor3 <- as.character(targetsDF3$Colors)

targets3 <- AnnotatedDataFrame(targetsDF3)

targetsDF3
```

```
##          fileName Grupos ShortName Colors
## 1  GSM1411021.CEL     XO      1_XO    red
## 2  GSM1411022.CEL     XO      2_XO    red
```

```
## 3  GSM1411023.CEL     XO      3_XO    red
## 4  GSM1411024.CEL     XO      4_XO    red
## 5  GSM1411025.CEL     XO      5_XO    red
## 6  GSM1411026.CEL     XX      6_XX   blue
## 7  GSM1411027.CEL     XX      7_XX   blue
## 8  GSM1411028.CEL     XX      8_XX   blue
## 9  GSM1411029.CEL     XX      9_XX   blue
## 10 GSM1411030.CEL     XX     10_XX   blue
## 11 GSM1134016.CEL     XX     11_XX   blue
## 12 GSM1134017.CEL     XX     12_XX   blue
## 13 GSM1134018.CEL     XX     13_XX   blue
## 14 GSM1134019.CEL     XX     14_XX   blue
## 15 GSM1134020.CEL     XX     15_XX   blue
## 16 GSM1134021.CEL     XX     16_XX   blue
## 17 GSM1134022.CEL     XX     17_XX   blue
## 18 GSM1134023.CEL     XX     18_XX   blue
## 19 GSM1134024.CEL     XX     19_XX   blue
## 20 GSM1134025.CEL     XX     20_XX   blue
## 21 GSM1134026.CEL     XO     21_XO    red
## 22 GSM1134027.CEL     XO     22_XO    red
## 23 GSM1134028.CEL     XO     23_XO    red
## 24 GSM1134029.CEL     XO     24_XO    red
## 25 GSM1134030.CEL     XO     25_XO    red
## 26 GSM1134031.CEL     XO     26_XO    red
## 27 GSM1134032.CEL     XO     27_XO    red
## 28 GSM1134033.CEL     XO     28_XO    red
## 29 GSM1134034.CEL     XO     29_XO    red
## 30 GSM1134035.CEL     XO     30_XO    red
## 31 GSM1134036.CEL     XO     31_XO    red
## 32 GSM1134037.CEL     XO     32_XO    red
## 33 GSM1134038.CEL     XO     33_XO    red
## 34 GSM1134039.CEL     XO     34_XO    red
## 35 GSM1134040.CEL     XO     35_XO    red
## 36 GSM1134041.CEL     XO     36_XO    red
## 37 GSM1134042.CEL     XO     37_XO    red
## 38 GSM1134043.CEL     XO     38_XO    red
## 39 GSM1134044.CEL     XO     39_XO    red
## 40 GSM1134045.CEL     XO     40_XO    red
## 41 GSM1134046.CEL     XO     41_XO    red
## 42 GSM1134047.CEL     XO     42_XO    red
## 43 GSM1134048.CEL     XO     43_XO    red
## 44 GSM1134049.CEL     XO     44_XO    red
## 45 GSM1134050.CEL     XO     45_XO    red
## 46 GSM1134051.CEL     XO     46_XO    red
```

A continuación, se guardan los nombres de los archivos .CEL guardados en el dataframe `targetsDF3` en un nuevo objeto llamado `CELfiles3` y se cargan los archivos del directorio que tengan el mismo nombre que los guardados en ese objeto. Estos archivos se almacenan en un objeto ExpressionSet llamado `rawData3`.

```
CELfiles3 <- targetsDF3$fileName
rawData3 <- read.celfiles(file.path(dataDir, CELfiles3), phenoData = targets3)
```

```
## Reading in : E:/TFM/Data/GSM1411021.CEL
```

```
## Reading in : E:/TFM/Data/GSM1411022.CEL
## Reading in : E:/TFM/Data/GSM1411023.CEL
## Reading in : E:/TFM/Data/GSM1411024.CEL
## Reading in : E:/TFM/Data/GSM1411025.CEL
## Reading in : E:/TFM/Data/GSM1411026.CEL
## Reading in : E:/TFM/Data/GSM1411027.CEL
## Reading in : E:/TFM/Data/GSM1411028.CEL
## Reading in : E:/TFM/Data/GSM1411029.CEL
## Reading in : E:/TFM/Data/GSM1411030.CEL
## Reading in : E:/TFM/Data/GSM1134016.CEL
## Reading in : E:/TFM/Data/GSM1134017.CEL
## Reading in : E:/TFM/Data/GSM1134018.CEL
## Reading in : E:/TFM/Data/GSM1134019.CEL
## Reading in : E:/TFM/Data/GSM1134020.CEL
## Reading in : E:/TFM/Data/GSM1134021.CEL
## Reading in : E:/TFM/Data/GSM1134022.CEL
## Reading in : E:/TFM/Data/GSM1134023.CEL
## Reading in : E:/TFM/Data/GSM1134024.CEL
## Reading in : E:/TFM/Data/GSM1134025.CEL
## Reading in : E:/TFM/Data/GSM1134026.CEL
## Reading in : E:/TFM/Data/GSM1134027.CEL
## Reading in : E:/TFM/Data/GSM1134028.CEL
## Reading in : E:/TFM/Data/GSM1134029.CEL
## Reading in : E:/TFM/Data/GSM1134030.CEL
## Reading in : E:/TFM/Data/GSM1134031.CEL
## Reading in : E:/TFM/Data/GSM1134032.CEL
## Reading in : E:/TFM/Data/GSM1134033.CEL
## Reading in : E:/TFM/Data/GSM1134034.CEL
## Reading in : E:/TFM/Data/GSM1134035.CEL
## Reading in : E:/TFM/Data/GSM1134036.CEL
## Reading in : E:/TFM/Data/GSM1134037.CEL
## Reading in : E:/TFM/Data/GSM1134038.CEL
## Reading in : E:/TFM/Data/GSM1134039.CEL
## Reading in : E:/TFM/Data/GSM1134040.CEL
## Reading in : E:/TFM/Data/GSM1134041.CEL
## Reading in : E:/TFM/Data/GSM1134042.CEL
## Reading in : E:/TFM/Data/GSM1134043.CEL
## Reading in : E:/TFM/Data/GSM1134044.CEL
## Reading in : E:/TFM/Data/GSM1134045.CEL
## Reading in : E:/TFM/Data/GSM1134046.CEL
## Reading in : E:/TFM/Data/GSM1134047.CEL
## Reading in : E:/TFM/Data/GSM1134048.CEL
## Reading in : E:/TFM/Data/GSM1134049.CEL
## Reading in : E:/TFM/Data/GSM1134050.CEL
## Reading in : E:/TFM/Data/GSM1134051.CEL
```

Una vez cargados todos los archivos, se analiza el objeto ExpressionSet `rawData3`, donde se incluyen las 46 muestras a analizar.

```
rawData3
```

```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 1354896 features, 46 samples
##   element names: exprs
```

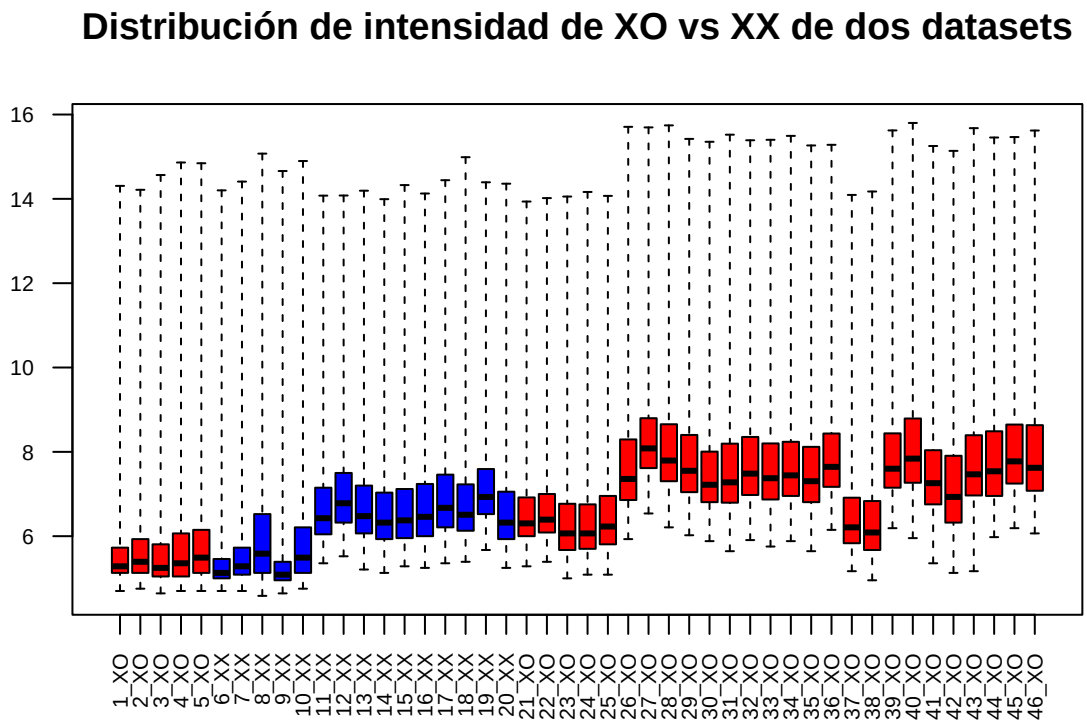
```
## protocolData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

Como se puede observar, el archivo contiene 46 muestras y una suma de 1354896 sondas en total. Además, `Annotation` indica el paquete de anotaciones necesario para poder realizar este análisis, el mismo que se ha utilizado anteriormente.

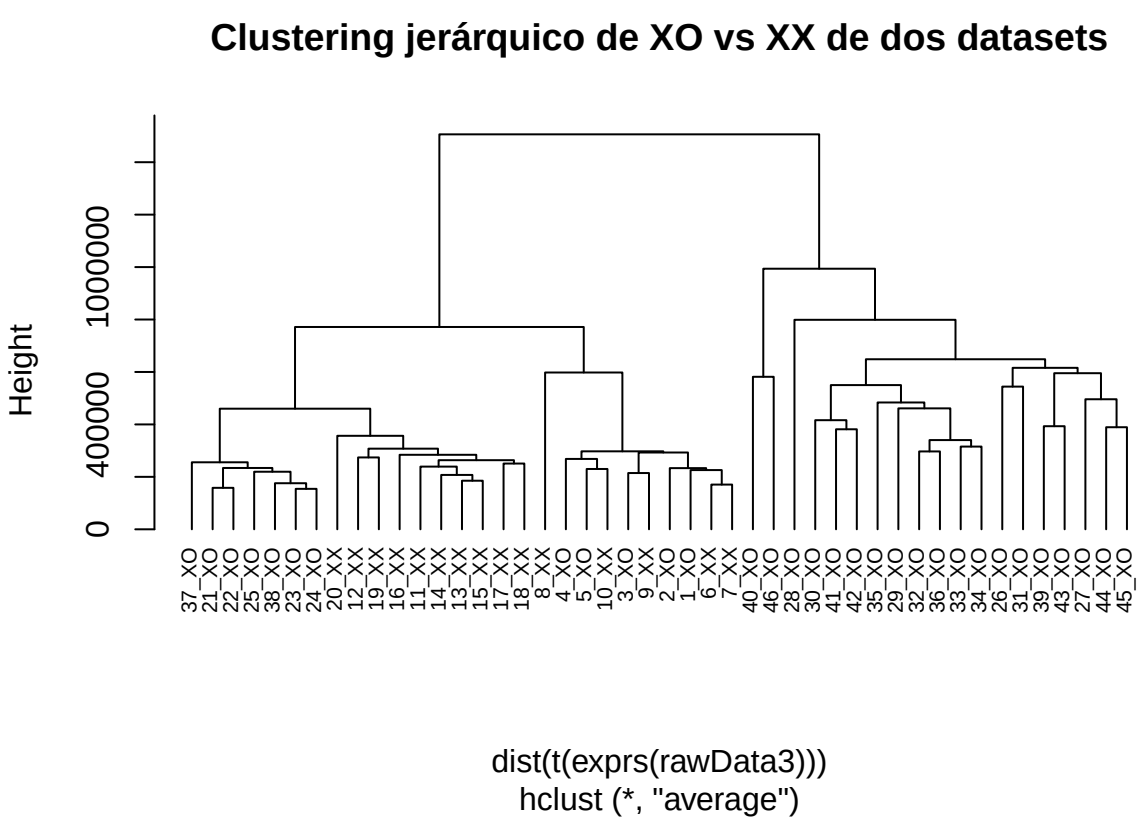
Preprocesado de los datos

Exploración y control de calidad

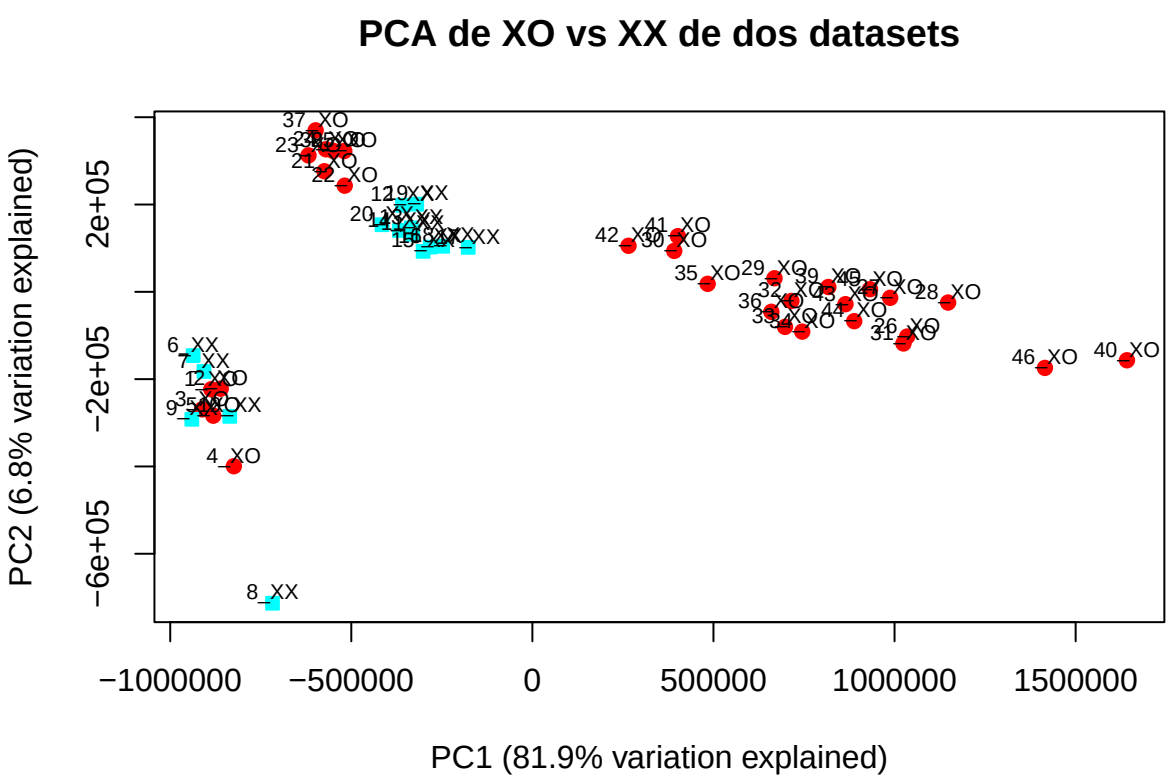
Mediante un **boxplot** se muestra como es la distribución de los valores.



Mediante un **clustering jerárquico** se muestra como se agrupan estas muestras.



En el **análisis de componentes principales**, se buscan las principales fuentes de variabilidad en los datos reduciendo las dimensiones.



Control de calidad con el paquete arrayQualityMetrics Con este paquete se obtienen los mismos gráficos que los anteriores pero todos a la vez en un archivo. Además, se observa si existe alguna muestra que tenga algún problema de outliers.

```
arrayQualityMetrics(rawData3, reporttitle = "QC_RawData3", force = TRUE)
```

Normalización

Antes de comenzar con el análisis de expresión diferencial y vistos los problemas que muestran algunas de las muestras, es necesario hacer que los arrays sean comparables entre sí y tratar de reducir, y si es posible eliminar, toda la variabilidad en las muestras que no se deba a razones biológicas. El proceso de normalización trata de asegurar que las diferencias de intensidad presentes en el array reflejen la expresión diferencial de los genes, en lugar de sesgos artificiales debidos a cuestiones técnicas. El método más utilizado para la normalización de arrays es el método RMA.

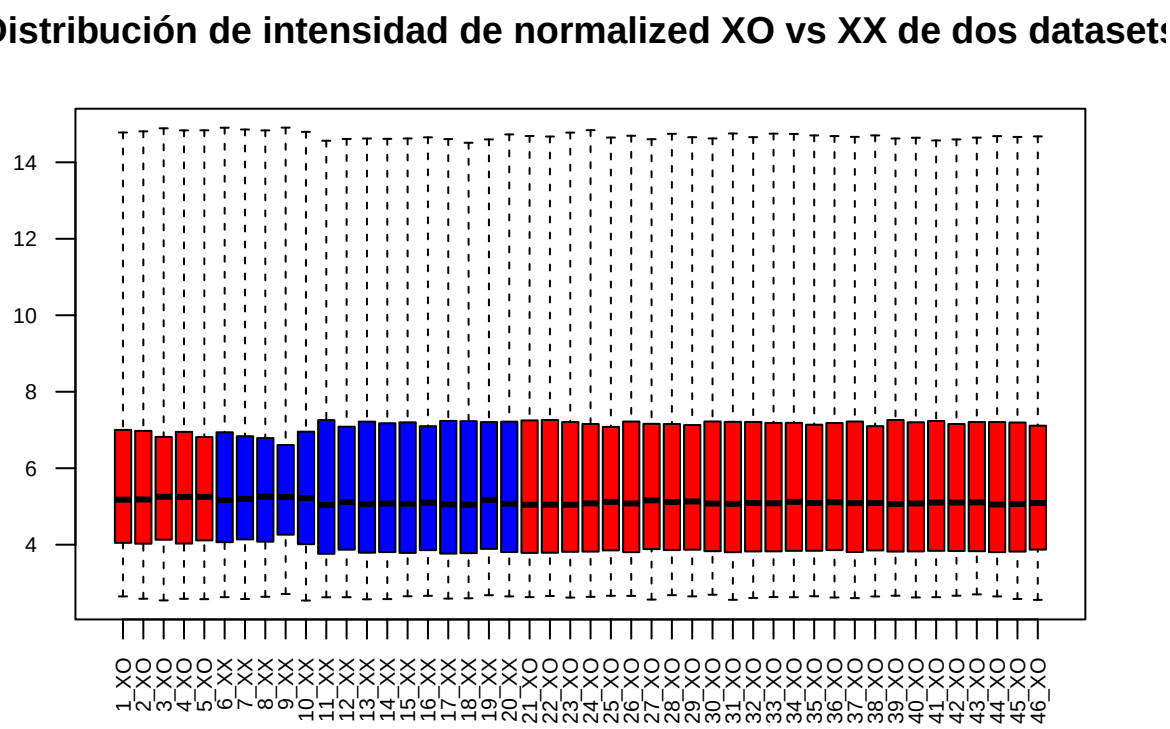
```
normData3 <- rma(rawData3)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

```
normData3
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 46 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: pd.hg.u133.plus.2
```

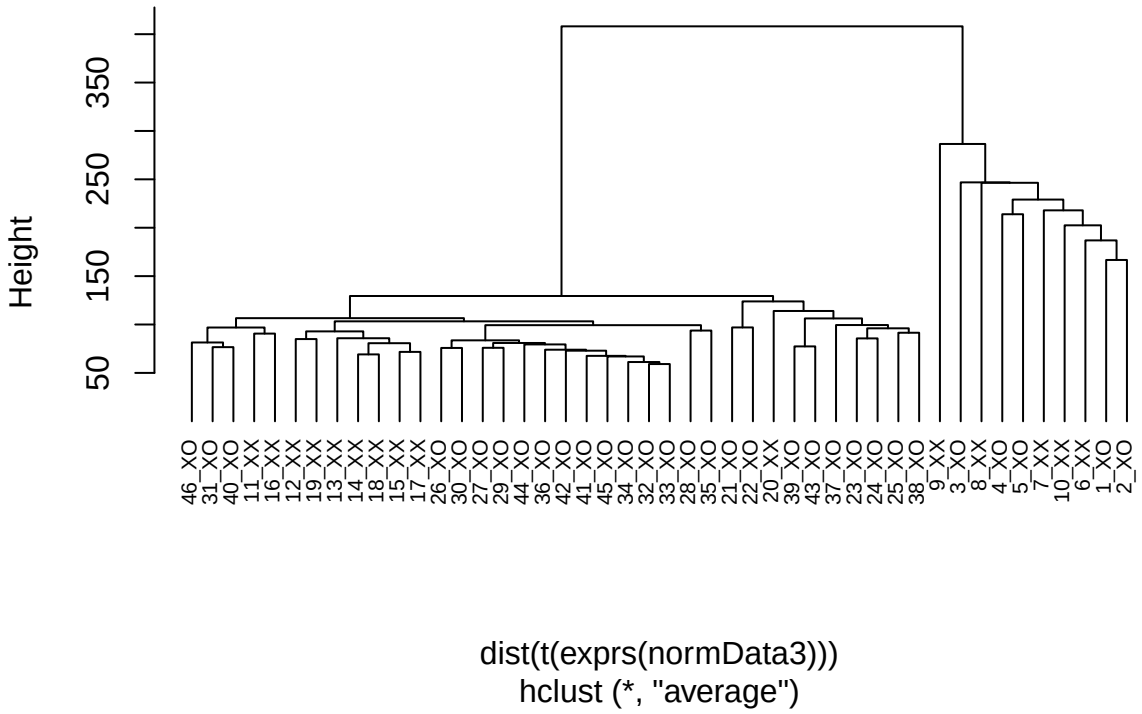
Analizando la calidad de los datos tras la normalización:

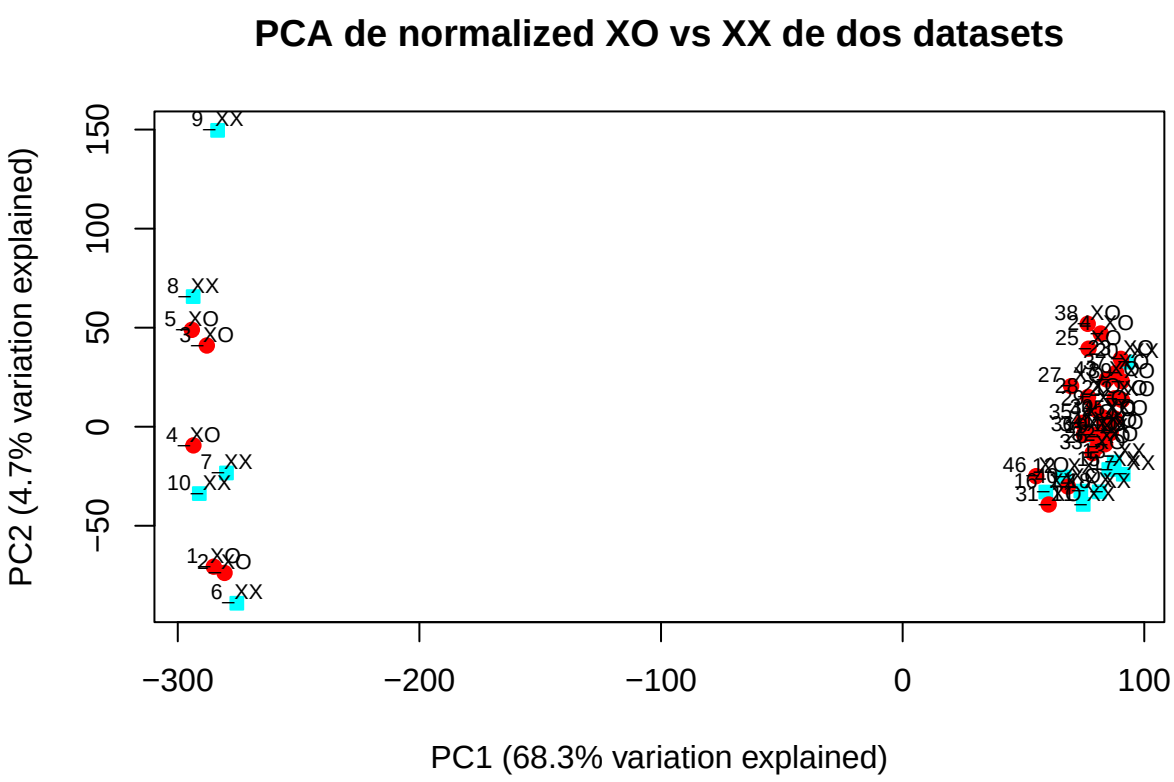


Además, se crea un nuevo archivo QC mediante el paquete `arrayQualityMetrics` para los datos normalizados.

```
arrayQualityMetrics(normData3, reporttitle = "QC_NormData3", force = TRUE)
```

Además, en el **clustering jerárquico** se observa como se agrupan las muestras tras la normalización.


Clustering jerárquico de normalized XO vs XX de dos datasets
Height
50
150
250
350
46_XO
31_XO
40_XO
11_XX
16_XX
12_XX
19_XX
13_XX
14_XX
18_XX
15_XX
17_XX
26_XO
30_XO
27_XO
29_XO
44_XO
36_XO
42_XO
41_XO
45_XO
34_XO
32_XO
33_XO
28_XO
35_XO
21_XO
22_XX
20_XO
39_XO
43_XO
37_XO
23_XO
24_XO
25_XO
38_XO
9_XX
3_XO
8_XX
4_XO
5_XO
7_XX
10_XX
6_XX
1_XO
2_XO
dist(t(exprs(normData3)))
hclust (*, "average")



Filtrado no específico

A continuación, se van a eliminar algunos genes sin basarnos en el efecto que se está estudiando, sino por alguna característica que se considere no relacionada con el tema de estudio, es decir, aquellos genes cuya variabilidad puede atribuirse a la variación aleatoria; por ejemplo, se pueden eliminar los genes que varían poco de un grupo a otro. Además, como en este caso se dispone del paquete de anotación (hgu133plus2.db), también se puede utilizar para eliminar conjuntos de sondas que no tengan asociado un identificador de gen (identificador Entrez, por ejemplo).

Empleando la función `nsFilter` del paquete `genefilter` de Bioconductor se eliminan genes basándose en un umbral de variabilidad mencionado anteriormente.

```
annotation(normData3) <- "hgu133plus2.db"
normData_filtered3 <- nsFilter(normData3, var.func = IQR, var.cutoff = 0.75, var.filter = TRUE, require

normData_filtered3
```

```
## $eset
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 5206 features, 46 samples
##   element names: exprs
## protocolData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
```

```
## phenoData
##   rowNames: 1 2 ... 46 (46 total)
##   varLabels: fileName Grupos ShortName Colors
##   varMetadata: labelDescription channel
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: hgu133plus2.db
##
## $filter.log
## $filter.log$numDupsRemoved
## [1] 22267
##
## $filter.log$numLowVar
## [1] 15618
##
## $filter.log$numRemoved.ENTREZID
## [1] 11574
##
## $filter.log$feature.exclude
## [1] 10
```

Analizando los resultados obtenidos, se han eliminado los siguientes genes:

- 22267 valores duplicados
- 15618 genes que tienen baja variabilidad
- 11574 genes que no tienen ID Entrez

Con este filtraje no específico se ha obtenido un dataset con menor número de genes, habiendo descartado los genes que con una alta probabilidad no van a estar relacionados con el estudio.

Los genes restantes se han almacenado en la variable `FilteredEset`.

```
filtered_normData3 <- normData_filtered3$eset
filteredData3 <- exprs(filtered_normData3)
colnames(filteredData3) <- pData(normData_filtered3$eset)$ShortName
```

Análisis

La selección de genes expresados de forma diferencial consiste básicamente en realizar algún tipo de prueba, normalmente por genes, para comparar la expresión génica entre grupos.

La matríz de diseño

El primer paso para el análisis basado en modelos lineales es crear la matriz de diseño. Básicamente es una tabla que describe la asignación de cada muestra a un grupo o condición experimental. Tiene tantas filas como muestras y tantas columnas como grupos, ya que en este caso se necesita el modelo de un factor, teniendo 2 tipos de muestras que se diferencian solo en un único factor. Cada fila contiene un uno en la columna del grupo al que pertenece la muestra y un cero en las demás.

```r
treat3 <- pData(filtered_normData3)$Grupos

treat3 <- factor(treat3)
design3 <- model.matrix(~0 + treat3)

rownames(design3) <- sampleNames3
colnames(design3) <- levels(treat3)

design3
```

```
##       XO XX
## 1_XO   1  0
## 2_XO   1  0
## 3_XO   1  0
## 4_XO   1  0
## 5_XO   1  0
## 6_XX   0  1
## 7_XX   0  1
## 8_XX   0  1
## 9_XX   0  1
## 10_XX  0  1
## 11_XX  0  1
## 12_XX  0  1
## 13_XX  0  1
## 14_XX  0  1
## 15_XX  0  1
## 16_XX  0  1
## 17_XX  0  1
## 18_XX  0  1
## 19_XX  0  1
## 20_XX  0  1
## 21_XO  1  0
## 22_XO  1  0
## 23_XO  1  0
## 24_XO  1  0
## 25_XO  1  0
## 26_XO  1  0
## 27_XO  1  0
## 28_XO  1  0
## 29_XO  1  0
## 30_XO  1  0
## 31_XO  1  0
## 32_XO  1  0
## 33_XO  1  0
## 34_XO  1  0
## 35_XO  1  0
## 36_XO  1  0
## 37_XO  1  0
## 38_XO  1  0
## 39_XO  1  0
## 40_XO  1  0
## 41_XO  1  0
## 42_XO  1  0
```

```
## 43_XO  1  0
## 44_XO  1  0
## 45_XO  1  0
## 46_XO  1  0
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$treat3
## [1] "contr.treatment"
```

La matríz de contrastes: definición de comparaciones

La matriz de contrastes se utiliza para describir las comparaciones entre grupos. Consta de tantas columnas como comparaciones y tantas filas como grupos. Una comparación entre grupos se representa mediante un "1" y un "-1" en las filas de grupos a comparar.

```
cont.matrix3 <- makeContrasts(XO.vs.XX = XO - XX, levels = design3)
comparisonName3 <- "Efecto de la monosomía X frente al control XX"
cont.matrix3
```

```
##       Contrasts
## Levels XO.vs.XX
##     XO        1
##     XX       -1
```

Estimación del modelo y selección de genes

Una vez definida la matriz de diseño y los contrastes, se puede proceder a estimar el modelo, estimar los contrastes y realizar las pruebas de significación que llevarán a decidir, para cada gen y cada comparación, si pueden considerarse de expresión diferencial. Para ello, se utilizará el paquete `limma`.

```
fit3 <- lmFit(filteredData3, design3)
fit.main3 <- contrasts.fit(fit3, cont.matrix3)
fit.main3 <- eBayes(fit.main3)
```

Análisis de genes diferencialmente expresados utilizando p = 0.1

El paquete `limma` implementa la función `topTable` que contiene, para un contraste dado, una lista de genes ordenados de menor a mayor p-valor que pueden considerarse de mayor a menor expresión diferencial.

```
topTab3 <- topTable(fit.main3, number = nrow(fit.main3), coef = "XO.vs.XX", adjust = "fdr", lfc = 1, p.v
dim(topTab3)
```

```
## [1] 61  6
```

Echando un vistazo a las primeras líneas del topTable, se pueden observar los genes que más cambian su expresión entre XX y XO, ordenados por su p-valor (de menor a mayor).

```
head(topTab3)
```

```
##                 logFC  AveExpr         t      P.Value    adj.P.Val         B
## 224588_at   -5.715969 6.277132 -7.657397 6.565200e-10 3.417843e-06 11.989823
## 203990_s_at -1.220397 5.756018 -5.517299 1.298507e-06 2.253343e-03  5.178456
## 235388_at   -1.383390 6.047576 -5.110740 5.330377e-06 5.549989e-03  3.902128
## 1553852_at  -1.100266 5.358177 -4.957126 9.029630e-06 7.834709e-03  3.425787
## 201058_s_at  1.764117 8.202346  4.817958 1.450281e-05 9.351609e-03  2.997719
## 202444_s_at -1.120058 5.649846 -4.625979 2.770227e-05 1.256917e-02  2.413429
```

Como se puede observar, la primera columna del `topTable3.2` contiene el ID del fabricante (*Affymetrix*) para cada conjunto de sondas. El siguiente paso es adivinar que gen corresponde a cada ID de Affymetrix, utilizando para ello la anotación de genes.

Una vez que se ha obtenido la tabla con los genes diferencialmente expresados, denominada `topTab3` es útil proporcionar información adicional sobre las características que se han seleccionado. Este proceso se denomina "anotación" y lo que hace es buscar información para asociar identificadores que aparecen en la tabla superior, normalmente correspondientes a conjuntos de sondas o transcritos dependiendo del tipo de array, con nombres más familiares o intuitivos como pueden ser el símbolo del Gen o el identificador Entrez Gene.

```
anotaciones3 <- AnnotationDbi::select(hgu133plus2.db, keys = rownames(filteredData3), columns = c("ENTRE
head(anotaciones3)
```

```
##       PROBEID ENTREZID SYMBOL
## 1   204639_at      100    ADA
## 2   222880_at    10000   AKT3
## 3   207078_at    10001   MED6
## 4 209027_s_at    10006   ABI1
## 5   227647_at    10008  KCNE3
## 6 210458_s_at    10010   TANK
```

Una vez se tienen las anotaciones de todos los genes de este array, se combinaran estos nombres con los genes obtenidos en la `topTab3`, para así identificarlos con su Entrez ID y el símbolo del gen.

```
topTabAnotada3 <- topTab3 %>%
    mutate(PROBEID = rownames(topTab3)) %>%
    left_join(anotaciones3) %>%
    arrange(P.Value) %>%
    dplyr::select(7, 8, 9, 1:6)

head(topTabAnotada3)
```

```
##       PROBEID ENTREZID SYMBOL     logFC  AveExpr         t      P.Value
## 1   224588_at     7503   XIST -5.715969 6.277132 -7.657397 6.565200e-10
## 2 203990_s_at     7403  KDM6A -1.220397 5.756018 -5.517299 1.298507e-06
## 3   235388_at    80205   CHD9 -1.383390 6.047576 -5.110740 5.330377e-06
## 4  1553852_at   157680 VPS13B -1.100266 5.358177 -4.957126 9.029630e-06
## 5 201058_s_at    10398   MYL9  1.764117 8.202346  4.817958 1.450281e-05
## 6 202444_s_at    10613 ERLIN1 -1.120058 5.649846 -4.625979 2.770227e-05
##      adj.P.Val         B
## 1 3.417843e-06 11.989823
```

```
## 2 2.253343e-03  5.178456
## 3 5.549989e-03  3.902128
## 4 7.834709e-03  3.425787
## 5 9.351609e-03  2.997719
## 6 1.256917e-02  2.413429
```

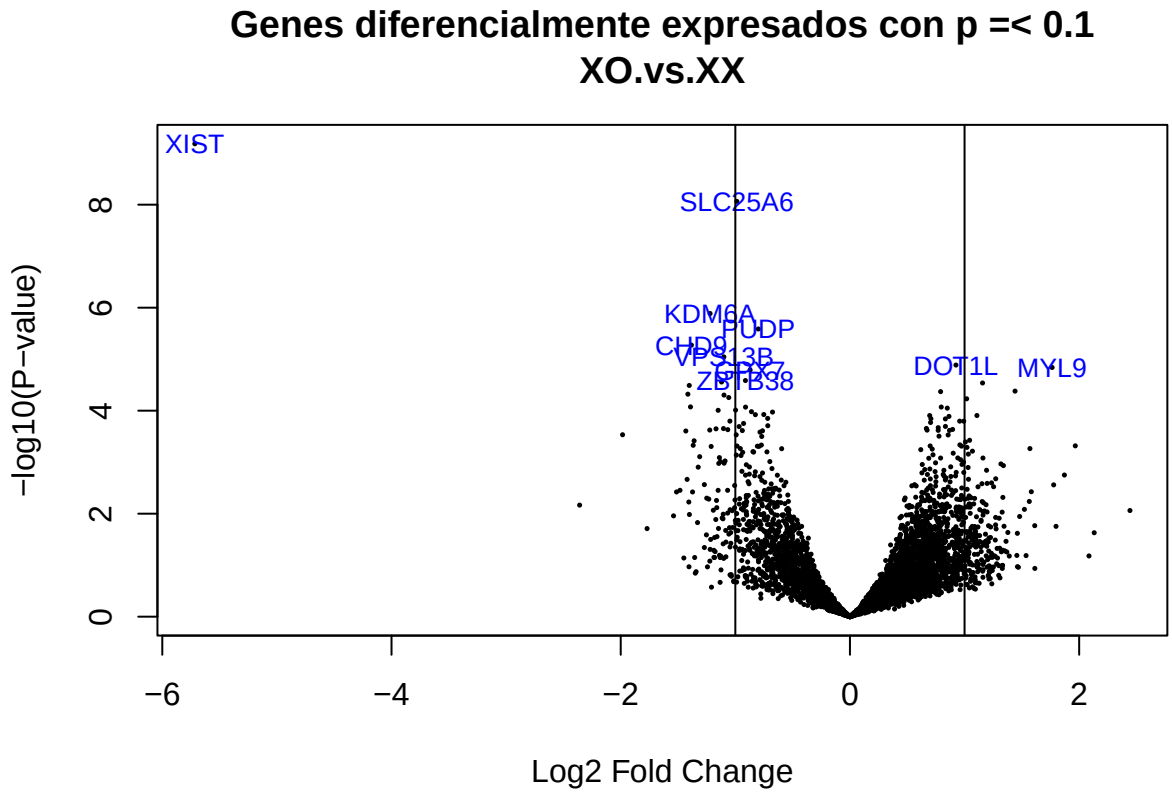
Por lo tanto, en el objeto `topTabAnotada3.` se ha obtenido una lista de genes diferencialmente expresados más legible.

Se almacenan los genes epresados diferencialmente entre los dos tipos de muestras en un archivo .xlsx para, después, poder comparar los genes obtenidos con los anotados anteriormente en la búsqueda bibliográfica.

```
ruta_GSE46687yGSE58435 <- file.path(workingDir, "genes_diferenciales_GSE46687yGSE58435.xlsx")

write.xlsx(topTabAnotada3, file = ruta_GSE46687yGSE58435)
```

Visualización de los genes diferencialmente expresados

Volcano plot Puede obtenerse una visualización de la expresión diferencial global mediante diagramas de volcán. Estos gráficos muestran si hay muchos o pocos genes con un gran cambio de pliegue y significativamente expresados o si este número es bajo. Estos gráficos representan en el eje X los cambios de expresión en escala logarítmica (efecto biológico) y en el eje Y el logaritmo negativo del valor p (efecto estadístico).

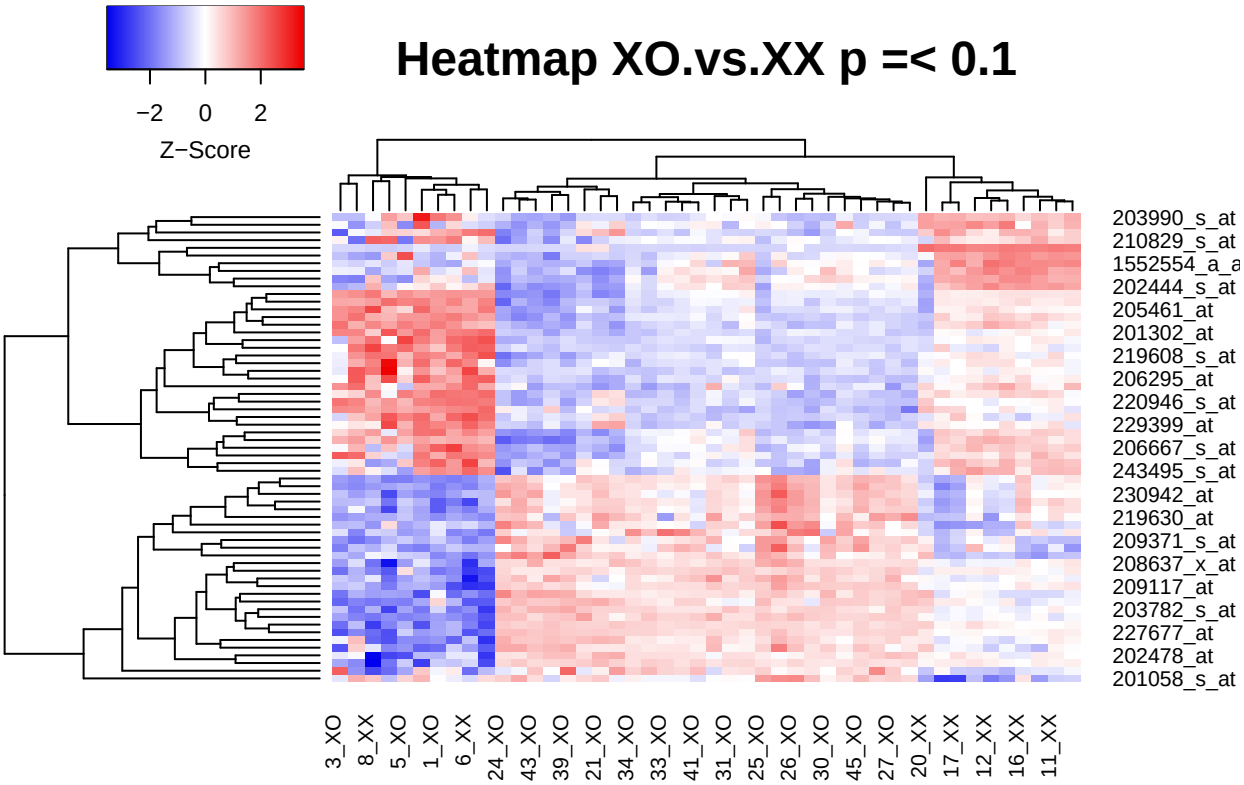


Los genes que aparecen en los primeros puestos de la `topTab3`, es decir, los que están más expresados diferencialmente, son lo que aparecen fuera de las lineas verticales (los cuales márcan el log fold = 1) y arriba.

Heatmap Los genes seleccionados con expresión diferencial pueden visualizarse mediante un mapa de calor. Estos gráficos utilizan paletas de colores para resaltar distintos valores -en este caso, expresiones significativamente diferenciales positivas (regulación al alza) o negativas (regulación a la baja).

Para hacer el mapa de calor se emplearán los genes seleccionados en los pasos anteriores.

```
selectedRows3 <- rownames(filteredData3) %in% rownames(topTab3)
selectedData3 <- filteredData3[selectedRows3, ]
coolmap(selectedData3, cluster.by = "de pattern", linkage.row = "complete", linkage.col = "complete", sl
```



Estudio de significación biológica

Una vez obtenida una lista de genes que caracteriza la diferencia entre dos condiciones, hay que interpretarla biológicamente.

Con este objetivo, este tipo de análisis busca establecer si, dada una lista de genes seleccionados por expresarse diferencialmente entre dos condiciones, las funciones, procesos biológicos o vías moleculares que los caracterizan aparecen en esta lista con mayor frecuencia que entre el resto de genes analizados.

Para llevar a cabo esto, se empleará el **análisis de enriquecimiento básico**.

```
# Se extraen las sondas de los datos seleccionados y se convierten a EntrezID
topProbes3 <- rownames(selectedData3)
entrezTop3 <- AnnotationDbi::select(hgu133plus2.db, topProbes3, "ENTREZID")$ENTREZID

#Eliminación de posibles duplicados
topGenes3 <- entrezTop3[!duplicated(entrezTop3)]
```

Por lo tanto, se tienen 2 listas Entrez:

- **topGenes:** con los Entrez de las sondas seleccionadas.
- **entrezUniverse:** con los Entrez de todas las sondas del array.

Se pueden utilizar muchos paquetes para realizar un análisis de enriquecimiento genético. Cada uno de ellos realiza un análisis ligeramente diferente, pero las ideas subyacentes son las mismas.

```
GOparams3 <- new("GOHyperGParams", geneIds = topGenes3, universeGeneIds = entrezUniverse,
    annotation = "hgu133plus2.db", ontology = "BP", pvalueCutoff = 0.05)
```

```
## Warning in makeValidParams(.Object): removing geneIds not in universeGeneIds
```

```
# Test de Fisher

GOhyper3 <- hyperGTest(GOparams3)

head(summary(GOhyper3), n = 10)
```

```
##        GOBPID       Pvalue OddsRatio    ExpCount Count Size
## 1  GO:0071557 0.0001280723       Inf  0.02285212     2    2
## 2  GO:0010628 0.0005788762  3.397181  4.74181499    13  415
## 3  GO:0036438 0.0007572179 89.940000  0.04570424     2    4
## 4  GO:0009891 0.0007921847  2.871012  7.63260822    17  668
## 5  GO:0002825 0.0008546899 19.613703  0.19424302     3   17
## 6  GO:0048332 0.0010173840 18.302041  0.20566908     3   18
## 7  GO:0010604 0.0010630467  2.510792 13.35706438    24 1169
## 8  GO:0006338 0.0011131664  4.952798  1.64535267     7  144
## 9  GO:2001020 0.0011245491  5.797101  1.19973632     6  105
## 10 GO:1903018 0.0011984364 17.154337  0.21709514     3   19
##                                                      Term
## 1                             histone H3-K27 demethylation
## 2                  positive regulation of gene expression
## 3                       maintenance of lens transparency
## 4             positive regulation of biosynthetic process
## 5           regulation of T-helper 1 type immune response
## 6                                   mesoderm morphogenesis
## 7  positive regulation of macromolecule metabolic process
## 8                                     chromatin remodeling
## 9           regulation of response to DNA damage stimulus
## 10           regulation of glycoprotein metabolic process
```

Estas son las categorías del Gene Ontology que están más enriquecidas y el p-valor que corresponde a cada una. El valor `OddsRatio` muestra cuantas veces más abundante es ese Gene Ontology de lo que se esperaría. Por lo tanto, cuanto mayor sea este número, esta catergoría será más importante. El numéro de categorías GO diferentes obtenidas es:

```
dim(summary(GOhyper3))
```

```
## [1] 406   7
```

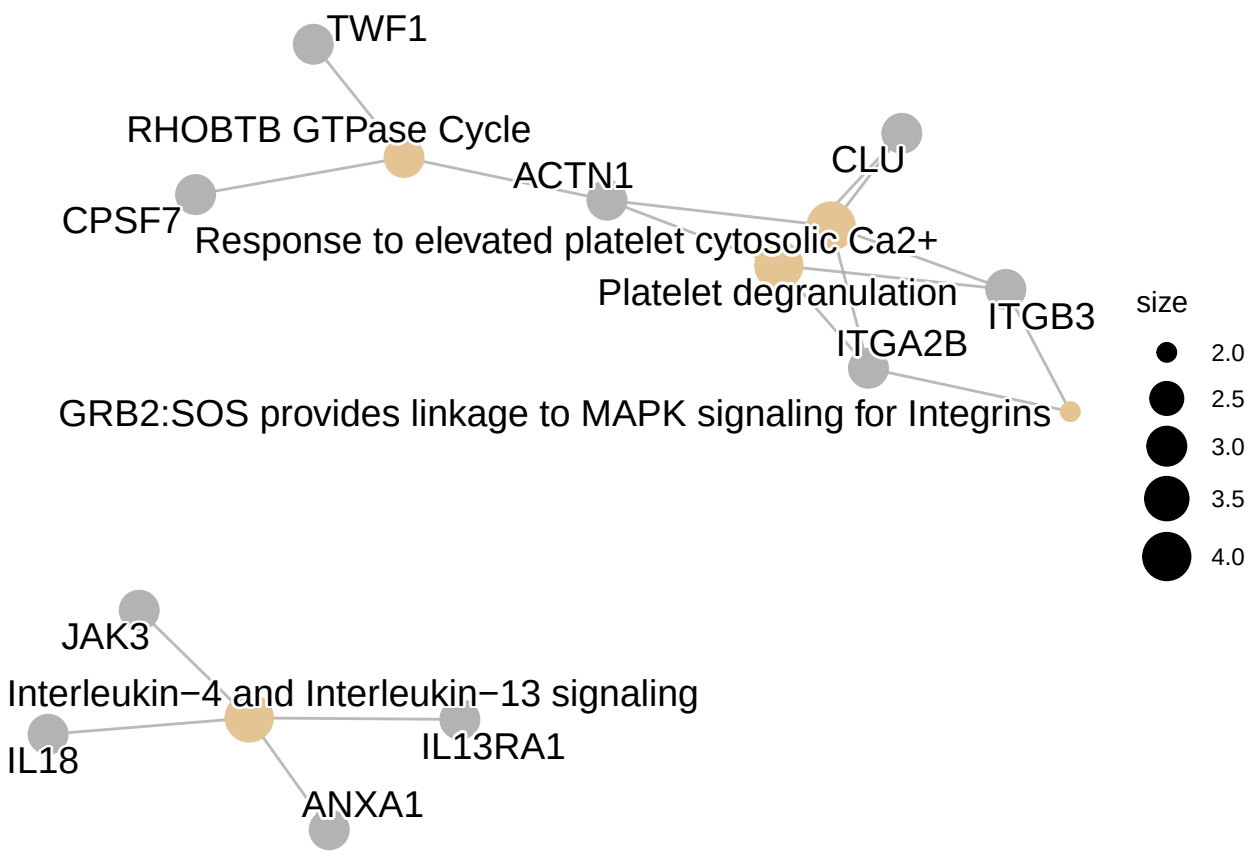
Antes de finalizar el análisis, se muestran gráficamente las funciones principales de los genes diferencialmente expresados.

```
whichGenes3 <-topTab3["adj.P.Val"]<0.15
selectedIDs3 <- rownames(topTab3)[whichGenes3]

EntrezIDs3 <- AnnotationDbi:::select(hgu133plus2.db, selectedIDs3, c("ENTREZID"))

EntrezIDs3 <- EntrezIDs3$ENTREZID

listOfSelected3 <- list(EntrezIDs3)
topTabName3 <- deparse(substitute(topTab3))
names(listOfSelected3) <- topTabName3

sapply(listOfSelected3, length)
```

```
## topTab3
##      61
```

```
listOfData3 <- listOfSelected3
comparisonsNames3 <- names(listOfData3)

for (i in 1:length(listOfData3)){
  genesIn3 <- listOfData3[[i]]
  comparison3 <- comparisonsNames3[i]
  enrich.result3 <- enrichPathway(gene = genesIn3,
                                  pvalueCutoff = 0.1,
                                  readable = T,
                                  pAdjustMethod = "BH",
                                  organism = "human",
                                  universe = universe)
  head(enrich.result3)
}
```

```
enrichplot::cnetplot(enrich.result3, categorySize = "geneNum", schowCategory = 15, vertex.label.cex = 0
```



Referencias

Zupkovitz, G., Tischler, J., Posch, M., Sadzak, I., Ramsauer, K., Egger, G., … & Seiser, C. (2006). Negative and Positive Regulation of Gene Expression by Mouse Histone Deacetylase1. *Molecular and cellular biology, 26*(21), 7913-7928. DOI: 10.1128/MCB.01220-06

Bioconductor. (2023). *Bioconductor, Open Source Software for Bioinformatics.* https://www.bioconductor.org/

Sánchez, Pla. [Alex]. (2022). *ADO-04 - Ejemplo de análisis (1) - Los datos* [recurso de aprendizaje audiovisual]. Universidad Oberta de Catalunya (UOC).

Sánchez, Pla. [Alex]. (2022). *ADO-04 - Ejemplo de análisis (2a) - Lectura y preprocesado* [recurso de aprendizaje audiovisual]. Universidad Oberta de Catalunya (UOC).

Sánchez, Pla. [Alex]. (2022). *ADO-04 - Ejemplo de análisis (2b) - Filtraje no específico* [recurso de aprendizaje audiovisual]. Universidad Oberta de Catalunya (UOC).

Sánchez, Pla. [Alex]. (2022). *ADO-04 - Ejemplo de análisis (3) - Selección de genes y análisis de enriquecimiento* [recurso de aprendizaje audiovisual]. Universidad Oberta de Catalunya (UOC).

Analisis de datos omicos - Materiales para un curso (ASPteching). (2022). *GitHub.* https://github.com/ASPteaching/Analisis_de_datos_omicos-Materiales_para_un_curso

Analisis de datos omicos - Ejemplo 1 - Microarrays (ASPteching). (2022). *GitHub.* https://github.com/ASPteaching/Analisis_de_datos_omicos-Ejemplo_1-Microarrays

Omics Data Analysis - Case Study 1 - Microarrays (ASPteching). (2022). *GitHub.* https://github.com/ASPteaching/Omics_Data_Analysis-Case_Study_1-Microarrays