Creación de una herramienta web para el análisis de datos ómicos

Isaac Morales General
Máster en Bioinformática y Bioestadística
Programación para la bioinformática

Ricardo Gonzalo Sanz
Alexandre Sánchez Pla, Antoni Pérez Navarro, Carles Ventura Royo, José Antonio Morán Moreno y María Jesús Marco Galindo
24 de mayo de 2017


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Creación de una herramienta web para el análisis de datos ómicos*
Nombre del autor:	*Isaac Morales General*
Nombre del consultor/a:	*Ricardo Gonzalo Sanz*
Nombre del PRA:	*Alexandre Sánchez Pla, Antoni Pérez Navarro, Carles Ventura Royo, José Antonio Morán Moreno y María Jesús Marco Galindo*
Fecha de entrega (mm/aaaa):	05/2017
Titulación:	*Máster de Bioinformática y Bioestadística*
Área del Trabajo Final:	*Programación para la bioinformática*
Idioma del trabajo:	*Castellano*
Palabras clave	*Análisis de microarrays, Shiny, pipeline.*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El análisis de microarrays para calcular la expresión diferencial de los genes, es una de las técnicas en las que la bioinformática desempeña un importante papel. Sin embargo, para realizar estos análisis se requiere que el usuario posea un cierto conocimiento de programación.

Este trabajo pretende crear una aplicación web que permita a usuarios que no dispongan de los conocimientos suficientes de programación, realizar el análisis de la expresión diferencial de los genes a partir de microarrays. Para ello, se desarrolla un pipeline en el lenguaje de programación R que marcará la ruta a seguir para el análisis y, posteriormente, se implementará este pipeline en una herramienta web creada mediante la herramienta Shiny.

Para comprobar la correcta funcionalidad tanto del pipeline como de la aplicación web, se descargan tres datasets que permitan realizar el análisis utilizando para ello distintos parámetros. El primer dataset necesitará de un archivo de anotación de *Mus musculus,* mientras que los otros dos datasets utilizarán de *Homo sapiens*. Además, el dataset número tres contendrá tres grupos para poder realizar comparaciones entre ellos.

Tras la implementación de la herramienta, esta muestra un correcto funcionamiento para las funciones programadas. Sin embargo, sería conveniente ampliar la herramienta otorgándole un mayor número de archivos de anotación predeterminados, distintos tipos de normalización de los datos y realizar análisis de RNA-seq y NGS.

Abstract (in English, 250 words or less):

Microarray analysis to calculate the differential expression of genes is one of the techniques in which bioinformatics plays an important role. However, these analyzes requires that user has some programming expertize.

This work aims to create a web application that allows users who do not have enough programming skills to perform the differential expression analysis of genes from microarrays. To do this, a pipeline is developed in the R programming language that will mark the path to be followed for analysis and this pipeline will be implemented in a web tool created using Shiny.

To verify the correct functionality of both the pipeline and the web application, three datasets are downloaded to allow the analysis to be performed using different parameters. The first dataset will need a *Mus musculus* annotation file, while the other two datasets will use *Homo sapiens*. In addition, dataset number three will contain three groups to make comparisons between them.

After the implementation of the tool, it shows a correct operation for the programmed functions. However, it would be advisable to extend the tool by granting a larger number of predetermined annotation files, different types of data normalization, and RNA-seq and NGS analysis.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

El procesado de grandes cantidades de datos procedentes de análisis genómicos es una de las áreas en las que la bioinformática ocupa una gran parte de su rango de estudio y ante el posible difícil acceso al desarrollo de estos análisis de los usuarios que no tengan suficientes conocimientos de programación, considero necesario el facilitar este acceso mediante el desarrollo de esta aplicación web.

Este Trabajo Final de Máster consiste en el desarrollo de un pipeline para el análisis de datos procedentes de experimentos con microarrays y su posterior implementación en una herramienta web que se desarrollará mediante el paquete “Shiny” del programa estadístico R, la cual facilitará el acceso y el análisis de estos datos a usuarios que no dispongan de los conocimientos suficientes para implantar sus propios pipelines.

1.2 Objetivos del Trabajo

Para llevar a cabo el desarrollo de este trabajo se establecen dos objetivos principales: desarrollar un pipeline para el análisis de datos de microarrays y, posteriormente, implementar ese pipeline en una aplicación web. A continuación, se describen cada uno de los objetivos:

Objetivo I. Desarrollar un pipeline para el análisis de datos de microarrays.

Para llevar a cabo este objetivo, en primer lugar, se debe conocer cada una de las etapas que constituyen el análisis de datos de microarrays, también se debe establecer qué características debe reunir el dataset objetivo del análisis y proceder a descargarlo de la base de datos pública GEO del NIH. Finalmente, se deberá desarrollar este pipeline en R y comprobar que realiza correctamente su función.

Objetivo II. Implementar el pipeline desarrollado en una aplicación web.

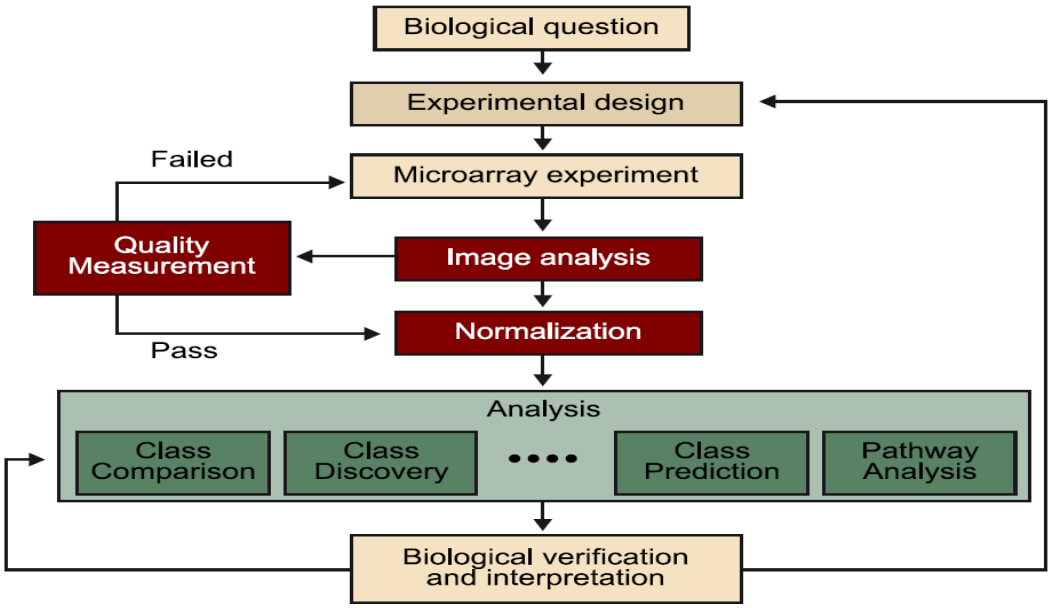
En este objetivo es importante familiarizarse con el paquete que llevará a cabo el desarrollo de la aplicación, que en este caso se trata del paquete 'Shiny' de R. A continuación, se realizará el desarrollo teórico y la programación de cada uno de los componentes que integrarán esta aplicación. Finalmente, debe llevarse a cabo un test de robustez que afirme que la aplicación cumple correctamente su función.

1.3 Enfoque y método seguido

Este TFM combina tanto la parte informática del Máster, al desarrollar mediante el programa R una aplicación web que permita analizar los datos y, posteriormente, implementarlo en una página web, como la parte más relacionada con la biología, ya que al analizar datos procedentes de análisis de microarrays se debe comprender que el fin último es dar respuesta a una pregunta biológica.

El análisis de microarrays pasa por distintas etapas, desde que el investigador se plantea la hipótesis de trabajo hasta que se obtienen los resultados finales (en nuestro caso solamente nos aplica desde que se ha realizado el experimento) (figura 1).

Figura 1. Etapas en el proceso de análisis de datos de microarrays. Análisis de datos ómicos[1].



En una primera fase se identifican aquellos puntos del proceso más relevantes y su importancia en la aplicación web futura.

1. Definir etapas del proceso de análisis (entrada de archivos, control de calidad, normalización, …)

2. Archivos necesarios de entrada al análisis

3. Etapas más costosas computacionalmente

4. ¿Qué parámetros han de ser fijos y cuáles han de ser modificables por el usuario en cada etapa del proceso?

5. ¿Qué resultados se van a generar? (archivos de texto, archivos de imágenes, …)

Posteriormente, se revisa la bibliografía para evaluar qué métodos se deben utilizar tratando de definir un pipeline lo más robusto posible.

Una vez definido teóricamente el pipeline a desarrollar, se procede a su implementación mediante el software correspondiente, el lenguaje R y el entorno de trabajo R-Studio. Las librerías y paquetes necesarios (BioConductor, limma, affy, arrayQualityMetrics, …) son seleccionados en función de lo que resulte más conveniente para el tipo de datos que queremos obtener.

Los datos a analizar se obtienen a partir de la base de datos pública GEO (Gene Expression Omnibus) del NIH (National Institutes of Health).

En una segunda etapa del TFM, el pipeline diseñado previamente se ejecuta y ajusta para que sea lo suficientemente robusto. Una vez llevado a cabo este proceso se comparan los datos obtenidos con resultados de análisis previamente realizados y/o comparándolos con los resultados que podamos obtener al realizar el mismo análisis en otras herramientas. Además, se identifican aquellas etapas en que se pueda dividir el pipeline en caso que fuese necesario (si son muy costosos computacionalmente, podría ser necesario dividir el análisis en varias aplicaciones web).

Finalmente, se implementa el pipeline desarrollado en una aplicación web utilizando el paquete “Shiny” del programa estadístico R. Este paquete permite crear aplicaciones web dinámicas sin que sea necesario ningún conocimiento de HTML ni de JavaScript ni PHP, únicamente a partir del código de R. Adicionalmente, permite a los futuros usuarios interactuar con sus datos sin que sea necesario realizar ninguna modificación en el código de la aplicación. Además, se utiliza el propio Dashboard de Shiny, lo que nos permite tener las distintas herramientas para el análisis de microarrays de forma que sean más accesibles y controlables por el usuario.

1.4 Planificación del Trabajo

Tareas

1. *Definir las etapas del proceso de análisis*. Es decir, qué tipo de archivos se deben proporcionar como entrada, qué métodos se eligen para realizar el control de calidad, normalización de los datos y análisis de la expresión diferencial de los genes.

2. *Definir los archivos necesarios de entrada para realizar el análisis*. Considerar qué características debe poseer el dataset que se utilizará para realizar el pipeline.

3. *Identificar los paquetes necesarios para realizar cada etapa del análisis*. Recabar información sobre el análisis de microarrays a través del software R. Para ello será necesario comprobar a través de bibliografía que paquetes deben utilizarse en cada etapa del análisis y, posteriormente, obtenerlos.

4. *Realizar el diseño teórico del pipeline*. Una vez se han cumplido los pasos anteriores se deben establecer uno o varios modelos de pipeline que permitan llevar a cabo el análisis.

5. *Obtener los datos que utilizaré para realizar el análisis*. A partir de las características descritas en el paso 2, buscar a través de la base de datos pública GEO del NIH, aquellos datasets que las cumplan.

6. *Testar la robustez del pipeline*. Establecer una serie de pruebas que permitan comprobar que el pipeline realiza correctamente su función.

7. *Analizar qué tipo de aplicación se va a realizar e identificar los componentes principales que integrarán el producto*. Estudiar las distintas formas que pueden resultar efectivas para el análisis de microarrays a través de una aplicación y qué tipo de análisis se quiere llevar a cabo.

8. *Proceder al diseño teórico de la aplicación, es decir, desarrollar un modelo o las especificaciones para el producto*. Establecer los detalles y características de los componentes definidos en el apartado anterior.

9. *Utilizar los modelos creados en el punto anterior para crear los componentes de la aplicación web*. Mediante R y Shiny programar cada uno de los componentes de la aplicación, intentando que resulte sencilla e intuitiva para los usuarios.

10. *Asegurarme de que los componentes individuales que integran la aplicación cumplen con los requerimientos de la especificación creada durante la etapa de diseño*. Desarrollar planes de prueba para cada componente de forma que se verifique que cada uno de ellos realiza específicamente su función.

11. *Testar la robustez de la aplicación web*. Desarrollar una batería de pruebas para analizar la ejecución global de la aplicación y comprobar que el resultado del análisis es satisfactorio, tanto por la calidad del análisis como de forma computacional en cuanto a tiempo y recursos.

Calendario

El calendario se adjunta en formato de diagrama de Gantt en el anexo I.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos a la finalización del trabajo son:

- Pipeline para el análisis de microarrays.

- Aplicación web para el análisis de datos de microarrays.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria final del TFM consta de los siguientes capítulos:

- *Expresión génica*: se realiza una introducción de las bases biológicas implicadas en la expresión de los genes.
- *Bioinformática*: se realiza una breve reseña sobre la historia de la bioinformática y sus aplicaciones actuales.
- *Análisis de microarrays*: se describe el proceso de análisis de microarrays, identificando los distintos tipos.
- *Datasets utilizados*: se presentan los datasets utilizados para comprobar la correcta ejecución de la aplicación web.
- *Desarrollo del pipeline*: se explica el procedimiento llevado a cabo para desarrollar el pipeline que servirá como base para desarrollar la aplicación web.
- *Desarrollo de la aplicación web*: se explica el procedimiento llevado a cabo para desarrollar la aplicación web.
- *Conclusiones*: se exponen las conclusiones del trabajo, se analiza el seguimiento de la planificación original y, para concluir, se plantean las posibles líneas futuras de trabajo.
- *Glosario*: definición de términos y acrónimos empleados en la memoria.
- *Bibliografía*: se ofrece información sobre la bibliografía utilizada en la elaboración del trabajo.
- *Anexos*: contiene el diagrama de Gantt para la planificación del proyecto y el código en R tanto del pipeline como de la aplicación web.

2. Expresión génica

La expresión génica es el mecanismo por el cual los organismos transforman la información contenida en los ácidos nucleicos en proteínas que se encargan de realizar funciones esenciales como las enzimas, las hormonas y receptores.

A lo largo del trabajo, aparecen un gran número de términos biológicos relacionados con la expresión génica, por lo cual es vital, entender la base biológica de este proceso.

2.1 Nucleótidos, genes y proteínas

Toda la información sobre cualquier organismo vivo conocido se encuentra codificada en complejas combinaciones de cuatro unidades estructurales conocidas como nucleótidos.

Los nucleótidos son moléculas orgánicas formadas por un azúcar de cinco carbonos (pentosa), una base nitrogenada y un grupo fosfato, unidos por un enlace covalente. Existen diversos tipos de bases nitrogenadas: Adenina (A), Guanina (G), Citosina (C), Timina (T) y Uracilo (U). Adenina y Guanina son consideradas bases púricas, mientras que Citosina, Timina y Uracilo, pirmidínicas. El grupo fosfato lleva a cabo la unión entre el carbono 5’ de un azúcar y el 3’ de otro para unir dos nucleótidos. La unión de estos nucleótidos no se realiza de forma aleatoria, sino que poseen características químicas que hacen que cada nucleótido sea atraído por otro distinto y muestre repulsión por el resto; de esta forma, se establecen las siguientes uniones: A-T o A-U (dependiendo de si se lleva a cabo en ADN o ARN, respectivamente) y C-G (véase figura 2).

El Ácido Desoxirribonucleico (ADN) y el Ácido Ribonucleico (ARN) son moléculas formadas por la unión de largas cadenas de nucleótidos. Generalmente, la molécula de ARN está formada por una cadena sencilla, mientras que la de ADN está formada por una cadena doble. La estructura de doble hélice del ADN fue descubierta en 1953 por James Watson y Francis Crick[2], los cuales también indicaron el modo el que esta molécula se

“desenrollaba” para que fuera posible proceder a su copia o lectura. Principalmente, el ADN sirve como almacén de información de los organismos, mientras que el ARN, entre otras funciones, actúa como puente para la transformación de genes en proteínas u otros productos génicos. La molécula de ADN reúne tres características que permite ser la depositaria de información genética de un organismo: la molécula de ADN contiene información basada en el orden y composición de los nucleótidos que la forman; es capaz de pasar esta información de generación en generación gracias a que cada cadena puede servir como molde para fabricar su complementaria; y es flexible, lo que permite que pueda almacenarse toda la información que requiere un ser vivo para ser como es y realizar sus funciones en un espacio tan pequeño como el interior de las células[3] (véase figura 3).

Figura 2. Unión de nucleótidos (reproducido de la página web de Affymetrix).

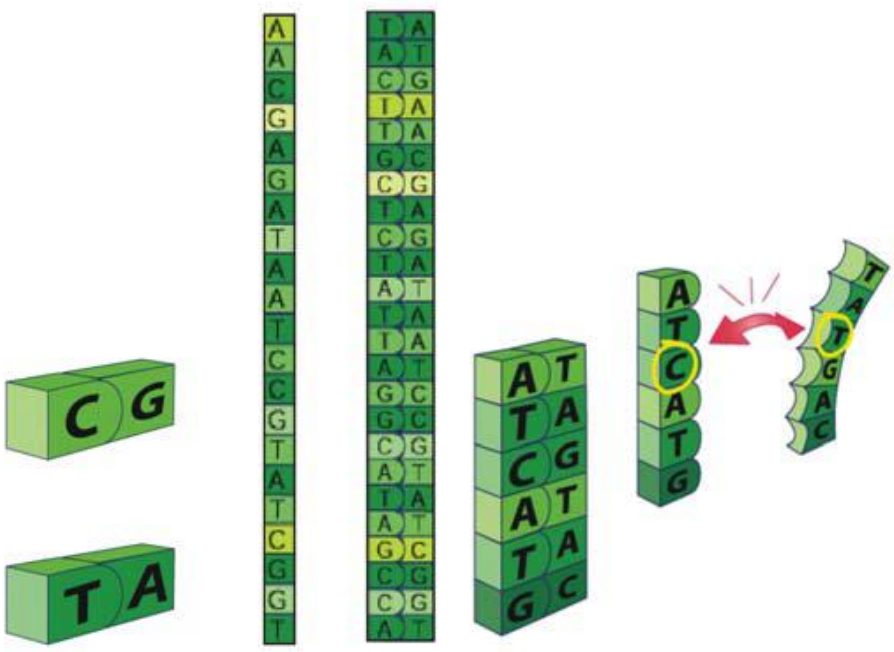
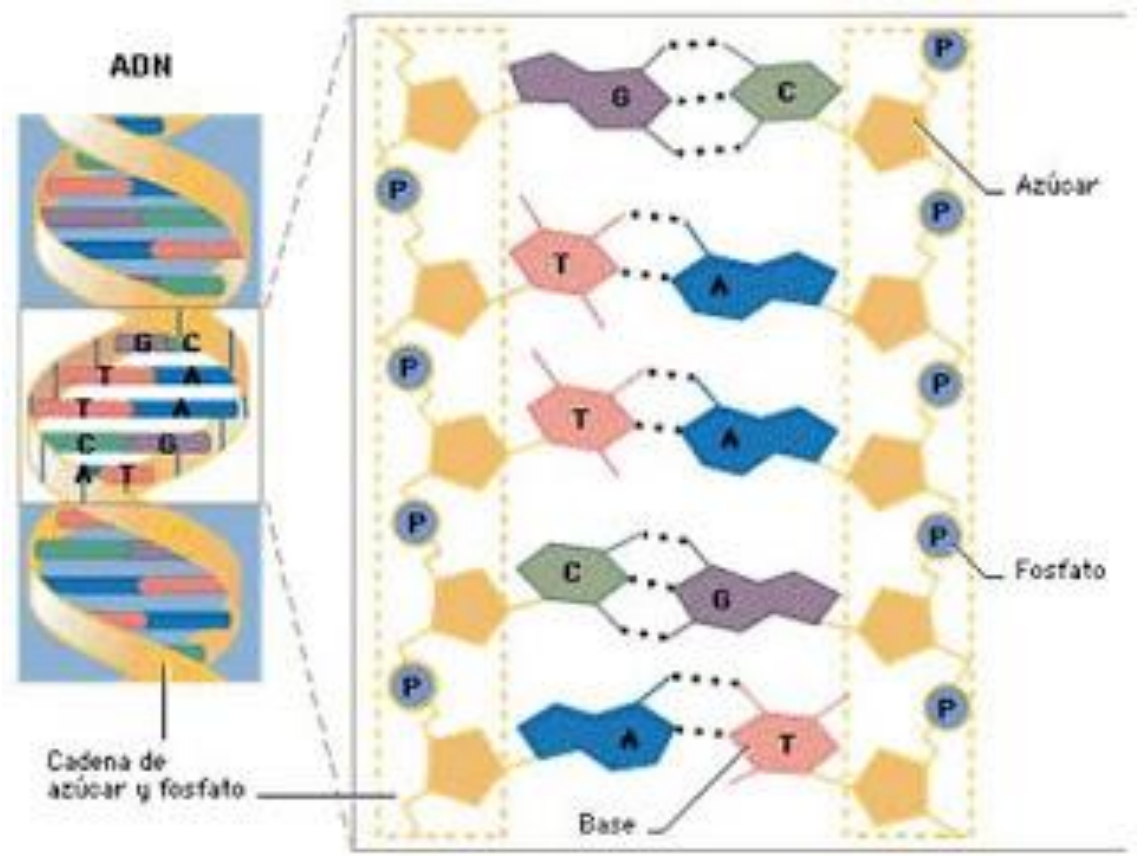
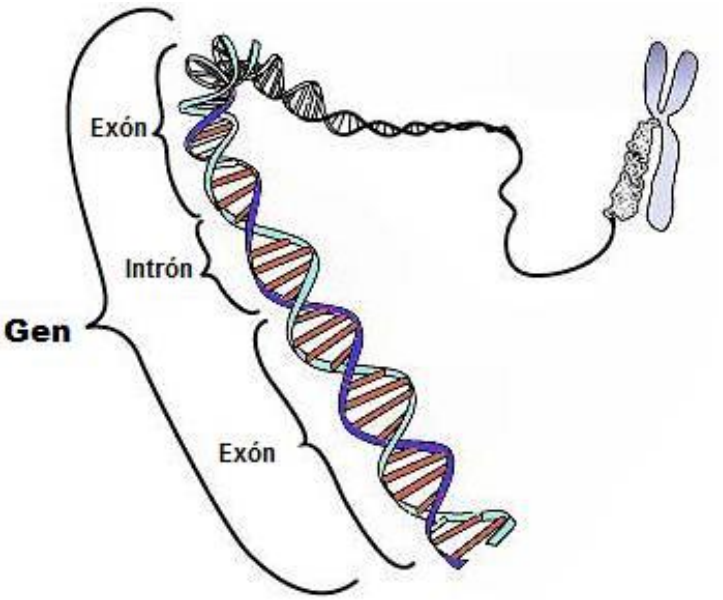


Figura 3. Estructura del ADN. (http://mtraburgos-inmunologia.blogspot.com.au/2010/11/acidos-nucleicos.html).



Un gen es una secuencia ordenada de nucleótidos que codifica un producto funcional, o ácido ribonucleico o proteínas. El gen es la unidad en la que se almacena y se transmite la información genética. Los genes ocupan posiciones específicas en los cromosomas denominadas ‘loci’. Cada una de las formas que puede tener un mismo gen se denominan ‘alelos’. El conjunto de alelos de un individuo se denomina ‘genotipo’ y el conjunto de productos que codifica ‘fenotipo’. El conjunto de genes de una especie es conocido como ‘genoma’ y dependiendo de los organismos, se encontrarán dentro de cromosomas (eucariotas) o no (procariotas). Muchos genes están constituidos por regiones codificantes (exones) dentro de las cuales se insertan regiones no codificantes (intrones) que son eliminadas durante el proceso de ‘splicing’ del ARN[4]. Esto solo ocurre en células eucariotas, ya que las células procariotas carecen de intrones (véase figura 4).

Figura 4. Esquema de un gen. (https://es.wikipedia.org/wiki/Gen).



Las proteínas son consideradas el producto final de la expresión génica. Las proteínas son moléculas formadas por cadenas lineales de aminoácidos, los cuales están unidos por enlaces conocidos como enlaces peptídicos. Las proteínas desempeñan un papel fundamental en los seres vivos; son esenciales para el crecimiento, para la síntesis y mantenimiento de diversos tejidos, para la producción de hormonas y enzimas, tienen un papel fundamental en el transporte de gases y en la transducción de señales. La cantidad de proteínas que produce un gen en una determinada unidad de tiempo varía en función de las necesidades de la célula. El nivel de expresión de los genes variará según las condiciones ambientales. Este proceso está regulado de forma precisa. Esta regulación puede ocurrir a nivel de transcripción o de traducción. El conjunto de las proteínas expresadas en una circunstancia determinada es denominado ‘proteoma’.

2.2 Transcripción, traducción y expresión

El proceso que dirige la construcción de una proteína a partir del ADN consta de dos pasos: transcripción y traducción.

La transcripción es el proceso por el cual se transmite la información del ADN al ARN. Es llevado a cabo por la ARN polimerasa que utiliza como molde una de las dos hebras del ADN (hebra codificante). La ARN polimerasa reconoce

un sitio específico de la molécula de ADN y se une, este sitio se denomina ‘promotor’ y permite que se produzca la transcripción. Genes sin promotores no pueden ser transcritos, aunque un promotor puede servir para transcribir varios genes seguidos que forman lo que se denomina un ‘operón’.

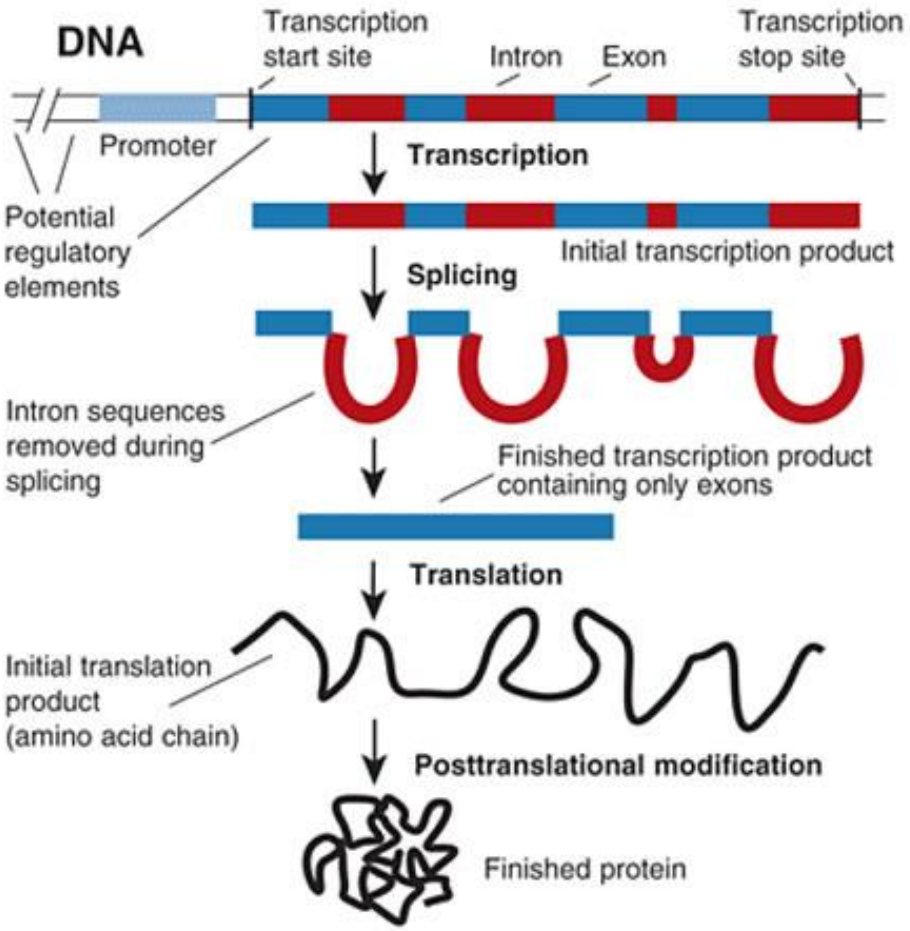
La traducción es el proceso por que la información genética que se ha transcrito a ARN va a ser procesada para formar una proteína. Este proceso tiene lugar en los ribosomas. Los ribosomas deben reconocer la secuencia por donde ha de iniciarse la traducción (codón de iniciación) y también, donde debe terminar (codón de terminación).

La transcripción, junto con la traducción constituyen el dogma central de la biología molecular[5,6] (véase figura 5).

Figura 5. Dogma central de la biología molecular.
(https://www.reeditor.com/columna/7396/4/biologia/dogma/central/la/biologia/molecular).

Sin embargo, la traducción del ARN hasta una proteína no es suficiente para que esta se expresa correctamente y desarrolle su función. Se requiere que la proteína se pliegue de forma adecuada, función de la que se encargan las chaperoninas (enzimas que ayudan al plegamiento de la proteína). Además, antes de la traducción se han tenido que producir unas modificaciones en el ARN que permitan que este sea traducido correctamente, como la eliminación de intrones o ‘splicing’, que se produce en el núcleo y deja colocados los exones para su posterior traducción (véase figura 6).

Figura 6. Proceso de formación de proteínas. (http://www.news-medical.net/life-sciences/What-is-Gene-Expression-(Spanish).aspx).



3. Bioinformática

La bioinformática es una disciplina que surge a partir del análisis computacional de secuencias de ADN y proteínas. Es un área interdisciplinar que se encarga de aplicar la tecnología computacional a la adquisición, almacenamiento, procesamiento, distribución, análisis e interpretación de información biológica. La bioinformática posee unos sólidos fundamentos de otras ciencias, como matemáticas, física, química y biología. El término bioinformática procede de Paulien Hogeweg en 1970[7], quien lo utiliza para hacer referencia al uso de procesos informáticos para estudiar sistemas biológicos.

3.1 Historia y origen

El desarrollo de la bioinformática, como no podía ser de otra forma, está muy ligado a los avances biológicos que se producen a lo largo de la historia. En 1970, Saul Needleman y Christian Wunsch desarrollan un algoritmo para el alineamiento global de dos secuencias[8], el cual se utiliza para alinear secuencias de proteínas o de ácidos nucleicos sin tener en cuenta la complejidad o longitud de las secuencias. También tiene lugar el inicio de la secuenciación del ADN y el desarrollo de software para analizar los datos obtenidos[9]. En 1982, Kurt Wüthrich publica un método para determinar la estructura terciaria de las proteínas[10], este método (Resonancia Magnética Nuclear) abre un nuevo campo en bioinformática, comienza el desarrollo de distintos métodos para predecir 'de novo' estructuras secundarias. También se produce el desarrollo de algoritmos para el alineamiento de secuencias, como el algoritmo Smith-Watherman[11], algoritmos de búsqueda en bases de datos de secuencias[12], el algoritmo FASTA que permite la comparación de secuencias[13] y la utilización de modelos ocultos de Markov para analizar patrones en las secuencias[14], lo que va a permitir localizar genes y predecir estructuras proteicas. En los años ochenta también se crean diversas bases de datos biológicas, como GenBank en 1982[15] o Swiss-Prot en 1986[16]. En los años noventa comienza la secuenciación de una gran cantidad de organismos, *Haemophilus influenzae* y *Mycoplasma genitalium* en 1995, *Saccharomyces*

cerevisae en 1996, *Escherichia coli* en 1997 y *Caenorhabditis elegans* en 1998; también se produce el desarrollo de herramientas más complejas dedicadas a la búsqueda de similitudes entre secuencias como BLAST[17] o el alineamiento múltiple de secuencias como ClustalW[18]. En los años 2000 se produce un gran avance en cuanto a la secuenciación del genoma de organismos, debido a la reducción tanto de tiempo, como de costes. Se publica el genoma de *Arabidopsis thaliana*, *Drosophila melanogaster*, *Rattus norvegicus*, *Pan paniscus*, *Felis catus* y, tal vez, el hito más importante hasta la fecha en el campo de la secuenciación y desarrollo de la bioinformática, la secuenciación del genoma humano (Proyecto Genoma Humano). Este proyecto contó con un presupuesto inical de 3 mil millones de dólares y la participación de un Consorcio Público Internacional, formado por EEUU, Reino Unido, Japón, Francia, Alemania, China y otros países; se inició en 1990 y concluyó en 2003.También se produce un incremento significativo en la cantidad de información almacenada en las bases de datos biológicas como GenBank y Swiss-Prot, además de la mejora de aplicaciones bioinformáticas para el análisis de los datos.

3.2 Objetivos y áreas de estudio

Luscombe considera tres objetivos de la bioinformática[19], el primer objetivo es organizar los datos de una manera adecuada para que los investigadores puedan acceder a ellos de forma sencilla e incluir nueva información; el segundo, consiste en el desarrollo de herramientas que ayuden en el análisis de los datos obtenidos y, finalmente, el tercero, es utilizar las herramientas para analizar los datos e interpretar los resultados de una manera biológicamente significativa.

Teniendo en cuenta los tres objetivos descritos anteriormente, se puede establecer las siguientes áreas de estudio de la bioinformática:

- Análisis de secuencias de ADN, ARN y proteínas.
- Análisis de datos de expresión génica.
- Análisis de alineamiento de secuencias.
- Predicción de estructura de proteínas.

- Análisis de la regulación génica.

Sin embargo, se debe mencionar también el apoyo que la informática ha proporcionado a biólogos evolutivos o de sistemas, permitiéndoles construir modelos computacionales a partir de los cuales poder realizar simulaciones y predecir el resultado de estos sistemas a través del tiempo.

4. Análisis de microarrays

Actualmente, la molécula de ADN es las más estudiada tanto para establecer relaciones entre las diferentes características genéticas de los seres vivos como para estudiar la predisposición a sufrir determinadas enfermedades. Esto no ha ocurrido siempre así, ya que hace unos años la molécula de ADN se consideraba demasiado compleja para su análisis. Sin embargo, en las últimas décadas se han producido diversos avances de la tecnología en biología molecular que permiten realizar análisis separando regiones del ADN, determinar la secuencia de nucleótidos o cuantificar el nivel de expresión de los genes.

4.1 Tecnología Microarray

La tecnología de microarray se ha convertido en una de las herramientas indispensables que usan los biólogos para monitorizar los niveles de expresión de los genes de un determinado organismo[20].

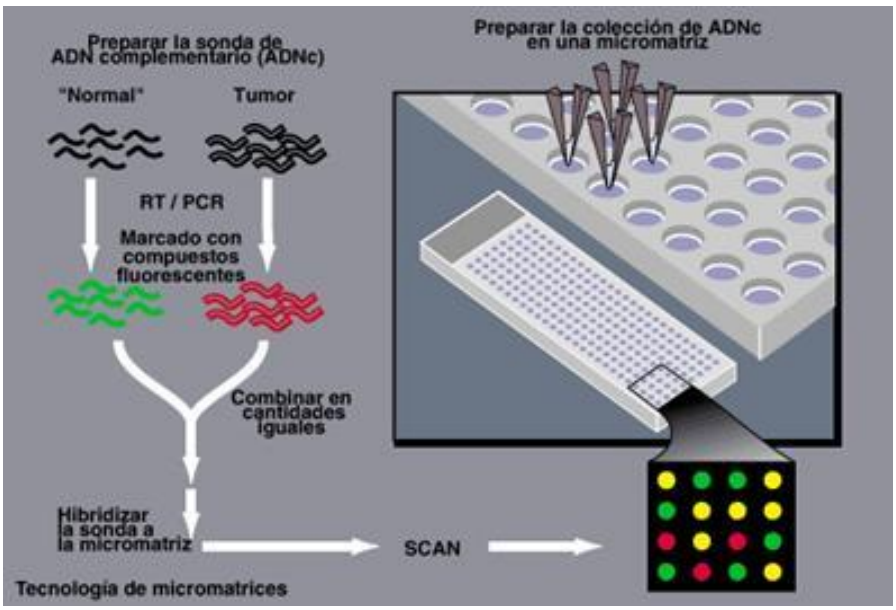
Un microarray o chip de ADN consiste en una superficie sólida donde el material genético es situado de manera que se forme una matriz de secuencias en dos dimensiones. El material genético puede consistir en secuencias cortas, llamadas oligonucleótidos, o de mayor tamaño, ADN complementario. A estos fragmentos de ADN de una sola hebra que son inmovilizados en el soporte, se les conoce como “sondas”.

Los ácidos nucleicos de las muestras a analizar se marcan por diversos métodos (enzimáticos, fluorescentes, etc.) y se incuban sobre el panel de sondas, permitiendo la hibridación (reconocimiento y unión entre moléculas complementarias) de secuencias homólogas[21] (véase figura 7).

Posteriormente a la hibridación se procede a realizar la medida de la intensidad señal de cada celda mediante el escaneo del chip. Una vez escaneado el chip se obtiene una imagen digital de las intensidades que debe ser cuantificada para poder obtener una medida de la expresión de cada gen.

A pesar de los grandes avances que ha proporcionado esta tecnología, los experimentos con microarrays no resultan infalibles y es posible que los datos obtenidos posean una gran cantidad de ruido[22,23]. Para evitarlo, se debe aplicar una serie de procesos que permita reducir el posible ruido que incorporen los datos, tales como: análisis de las imágenes, normalización de los datos, análisis estadístico de los genes diferencialmente expresados.

Figura 7. Tecnología microarray.
(http://www.quimicaviva.qb.fcen.uba.ar/contratapa/aprendiendo/capitulo15.htm).



4.2 Aplicaciones de los microarrays

El objetivo final del análisis de microarrays es comparar los niveles de expresión de los genes en diferentes condiciones. De esta forma se pueden asociar los distintos genes a determinadas funciones. Esta clase de análisis puede tener diversas aplicaciones:

- Agricultura: por ejemplo, comparando condiciones que hacen expresarse más un determinado gen cuya expresión da lugar a una característica deseada del producto.
- Farmacia: identificando genes que son regulados por un determinado fármaco.
- Diagnósticos clínicos: determinadas enfermedades pueden ser diagnosticadas mediante el análisis de la expresión génica.

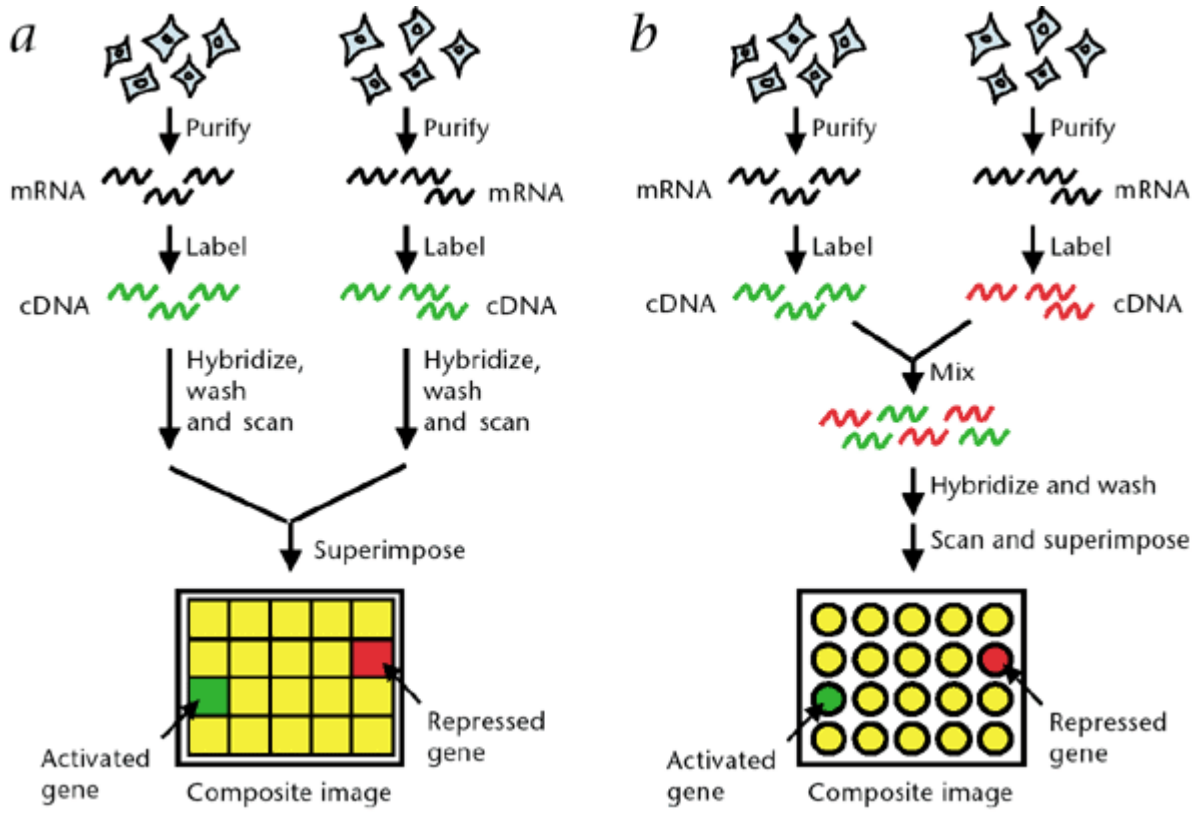
- Mapeado genético: de esta forma se puede localizar aquellas células, regiones o tejidos donde un determinado gen tiene una expresión diferencial.

4.3 Tipos de microarrays

Existen diferentes tipos de chips de ADN que se crean empleando diferentes tecnologías y se utilizarán en función de lo que se desee estudiar. Los principales tipos de microarrays son los siguientes (véase figura 8):

- Microarrays de un color o arrays de oligonucleótidos: en estos arrays solo se hibrida una muestra por lo que las muestras solo están marcadas con un marcador fluorescente. En este tipo de chips se pueden observar estimaciones del nivel de expresión, sin embargo, no pueden observarse distintas condiciones. Affymetrix© es la empresa líder en la comercialización de este tipo de microarrays.
- Microarrays de dos colores o *spotted arrays*: están basados en la hibridación competitiva de dos muestras. El ADN complementario de cada muestra se marca con un marcador fluorescente diferente (generalmente Cy3 y Cy5) y se hibridan sobre el mismo chip. A través de este proceso se puede detectar que genes se activan o reprimen en distintas condiciones.
- Chips de ADN para genotipado o *SNP arrays*: son utilizados para identificar posiciones específicas de un determinado genoma. Es decir, se pueden utilizar para identificar variaciones individuales.

Figura 8. Tipos de microarrays (a. Array de un color; b. Array de dos colores). (http://www.nature.com/nm/journal/v9/n1/fig_tab/nm0103-140_F1.html).



4.4 Proceso del análisis de datos de microarrays

Los arrays se pueden comprar prediseñados o diseñarlos con los genes y sondas que especifiquemos. Los microarrays deben ir acompañados de ficheros que indiquen la posición y naturaleza de las sondas, además de la anotación funcional de cada gen.

Para llevar a cabo un experimento con microarrays deben seguirse los siguientes pasos (véase figura 1):

1. Establecer la cuestión biológica que se quiere resolver.
    - Una vez planteada la cuestión, se debe especificar cuál es el propósito del estudio, que objetivos se persigue y que limitaciones puede presentar.

2. Realizar el diseño experimental.
    - Especificar el tipo de muestras que se van a obtener, por ejemplo, si las mezclas son individuales o si se van a realizar réplicas independientes.
    - Limitaciones en cuanto al presupuesto y el tiempo disponible. Establecer el número de arrays y de muestras que serán necesarios/posibles obtener.
3. Preparación de la muestra.
    - Extracción y aislamiento del ARN de la muestra que se quiere analizar (tejido, por ejemplo).
    - Purificación del ARN. La clave radica en evitar su degradación por la acción de ribonucleasas.
    - Comprobación de la integridad/calidad de las muestras de ARN. El ARN debe estar libre de contaminación de proteínas y ADN.
    - Una vez realizada la extracción del ARN, se lleva a cabo la amplificación del ARN mensajero.
    - Síntesis del ADN complementario mediante transcripción inversa.
    - Marcaje del ADN complementario mediante un marcador fluorescente que permita identificar posteriormente la intensidad de la expresión.
    - Hibridación en el chip de ADN.
4. Análisis y procesamiento de la imagen.
    - La cuantificación de la intensidad de la señal es un proceso muy importante ya que determina que valores se analizarán posteriormente. Este proceso se puede dividir en 3 etapas[24]: localizar los puntos a partir de los datos que proporciona el fabricante del chip, identificar qué pixeles corresponden al microarray (“foreground”) y cuales son fondo (“background”) y realizar la cuantificación, es decir, el promedio o mediana de las intensidades de los pixeles que forman el punto. También se debe cuantificar la medida del fondo, es decir, los pixeles reconocidos fuera de los puntos.

5. Control de calidad de los datos.
    - De esta forma se podrán detectar aquellos valores incorrectos y excluirlos del análisis. Estos valores incorrectos pueden originarse por accidentes en la manipulación, como rasguños o imperfecciones en la superficie del chip, o por problemas de calidad en el experimento.
    - Para realizar este control se pueden utilizar diversas técnicas, sin embargo, las más frecuentes son: realizar un histograma o boxplot, para analizar la distribución global de la fluorescencia en los distintos microarrays y dendogramas o PCA plots, para comprobar que las relaciones entre las distintas muestras se corresponden con el diseño del experimento realizado.
6. Normalización de los datos
    - El propósito de la normalización es identificar y eliminar variabilidad originada por el proceso técnico como diferente cantidad de ARN, diferentes parámetros del escáner, …
    - Existen diversas técnicas para la normalización de los datos, por ejemplo, MAS, dChip y GCRMA. Sin embargo, la técnica más usada es el RMA (Robust Multiarray Average)[25], el cual implica tres etapas:
        i. Corrección de fondo: considera que los datos de intensidades de los 'perfect match (PM)' (sondas que son diseñadas para ser complementarias de una secuencia específica) son el resultado de la suma entre la señal de fondo y la real. También asume que el nivel de fondo medio es común a cada array.
        ii. Normalización: para que se pueda comparar el valor de los distintos arrays. RMA utiliza el método de normalización por cuantiles, el cual transforma los valores de las intensidades en cada canal y en cada muestra, sin cambiarles el orden, de manera que estén distribuidos de forma idéntica. Se ordena cada uno de los microarrays en orden creciente, se toma la media o mediana por fila, se

reemplaza por la media o la mediana en todas las columnas y se vuelve al orden original.

iii. Sumarización de las diversas sondas asociadas a cada grupo de sondas para dar un único valor. RMA estima la intensidad de cada gen de forma independiente para cada sonda utilizando un método conocido como ‘median polish’.

7. Análisis de la expresión diferencial
    - Las técnicas estadísticas clásicas para comprobar si existen diferencias en la expresión de los genes no son adecuadas sin realizar algunas modificaciones[26], ya que en este caso lo que se quiere investigar es si existen diferencias en miles de variables (genes).
    - Generalmente se emplea un test estadístico clásico, como la t de Student y se calcula, para un gen, el valor observado (valor del test que compara los grupos) y se repite el mismo test un gran número de veces de forma que la asignación del grupo se cambie al azar. Al realizar este procedimiento se simula una situación en la cual no existen diferencias ya que la asignación de un grupo u otro a cada muestra es aleatoria. Por último, se calcula el valor de p, a partir del percentil que ocupe el valor observado en la distribución de los valores obtenidos por permutación. Este proceso se realiza para cada gen[27].
    - Para evitar falsos positivos se corrigen los valores de p, de esta forma, se controla el nivel de significación. Existen diversos métodos, desde el de Bonferroni, que considera significativos solo aquellos valores de p inferiores a la división entre alfa y el número de test hasta el de min-P y max-T[28].
8. Análisis de clasificación
    - Consiste en la clasificación de las muestras o de los genes según su nivel de similitud y nos permite encontrar relaciones desconocidas o verifican nuestras hipótesis. Lo podemos aplicar tanto a las muestras o a los genes.

9. Análisis funcionales
    - Consiste en comprobar si en un grupo de genes hay alguna función enriquecida. Para ello se puede utilizar la Gene Ontology (GO), la cual pretende describir de un modo consistente los genes de distintas bases de datos. GO se compone de tres ontologías dedicadas a los procesos biológicos (BP), los componentes celulares (CC) y las funciones moleculares (MF). Un gen puede tener varios términos GO asociados.

5. Datasets utilizados

Para llevar a cabo el análisis de los datos de microarrays y comprobar las distintas funcionalidades de la aplicación, se utilizan tres datasets procedentes de la base de datos pública GEO del NIH. Además, se especifica el tipo de archivo de targets que se utiliza para analizar cada uno de ellos.

El resumen de los datos de los datasets puede observarse en la tabla 1.

Tabla 1. Resumen de las variables de los datasets.

	Nº de grupos	Nº de muestras	Plataforma
Dataset 1	2	8	Affymetrix Mouse Gene 1.0 ST Array
Dataset 2	2	6	Affymetrix Human Genome U133 Plus 2.0 Array
Dataset 3	3	12	Affymetrix Human Genome U133 Plus 2.0 Array

5.1 Dataset 1

[https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE27174]

Título del estudio: Direct generation of functional dopaminergic neurons from mouse and human fibroblasts[29].

El objetivo de este estudio es analizar neuronas inducidas por dopaminérgicos a partir de fibroblastos embrionarios de ratón, los cuales han sido infectados con lentivirus que expresan factores de transcripción dopaminérgicos (Mash1 -también conocido como Ascl1-, Nurr1 -también conocido como Nr4a2- y Lmx1a). Los resultados proporcionan información sobre las bases moleculares y genéticas de la reprogramación desde fibroblastos embrionarios a neuronas. El propósito de este trabajo es comprobar si se puede mejorar el resultado clínico de la enfermedad de Parkinson, un trastorno neurológico resultante de la degeneración de neuronas dopaminérgicas mesencefálicas.

Es un dataset formado por ocho muestras, divididas en dos grupos: ‘control’ (fibroblastos no inducidos por dopaminérgicos, cuatro muestras) e ‘inducidos’

(fibroblastos inducidos por dopaminérgicos, cuatro muestras). (Véase el archivo de targets en la tabla 2)

El organismo modelo es *Mus musculus* y la plataforma utilizada es: Affymetrix Mouse Gene 1.0 ST Array.

El propósito de seleccionar este dataset es incluir un dataset con un número de muestras no muy elevado (ocho), el cual se divide en dos grupos y utiliza una plataforma basada en el genoma de ratón.

Tabla 2. Archivo targets dataset 1.

Filename	Group	Shortname	Colour
GSM671653.CEL	Induced	Induced1	Red
GSM671654.CEL	Induced	Induced2	Red
GSM671655.CEL	Induced	Induced3	Red
GSM671656.CEL	Induced	Induced4	Red
GSM671657.CEL	Not_induced	Not_induced1	Yellow
GSM671658.CEL	Not_induced	Not_induced2	Yellow
GSM671659.CEL	Not_induced	Not_induced3	Yellow
GSM671660.CEL	Not_induced	Not_induced4	Yellow

5.2 Dataset 2

[https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE62947]

Título del estudio: Identification of a Dual Inhibitor of SRPK1 and CK2 That Attenuates Pathological Angiogenesis of Macular Degeneration in Mice[30].

Este estudio se centra en el estudio de anticuerpos contra el factor de crecimiento endotelial vascular (VEGF) para impedir la angiogénesis excesiva, la cual contribuye a numerosas enfermedades como el cáncer o la retinopatía cegadora. En este estudio, se resolvió la estructura de SRPK1 unido a SRPIN340 por cristalografía de rayos X. Utilizando modelos de acoplamiento farmacóforo seguido de ensayos de quinasa in vitro, se seleccionó una biblioteca química a gran escala, y de este modo se identificó un nuevo inhibidor de SRPK1 (serina-arginina proteína quinasa 1). Este inhibidor, SRPIN803, impidió la producción de VEGF más eficazmente que SRPIN340

debido a la inhibición dual de SRPK1 y CK2 (caseína quinasa 2). En un modelo murino de degeneración macular relacionada con la edad, la administración tópica de ungüento para ojos que contenía SRPIN803 inhibió significativamente la neovascularización coroidea, lo que sugiere un potencial clínico de SRPIN803 como un ungüento tópico para la neovascularización ocular.

Este dataset está formado por seis muestras, divididas en dos grupos: 'DMSO' (el cual actúa como control en un ungüento tópico para la neovascularización ocular, tres muestras) y 'SRPIN803' (en el cual se ha añadido la molécula SRPIN803, que actúa como molécula inhibidora de SRPK1, tres muestras). (Véase el archivo de targets en la tabla 3)

El organismo modelo es *Homo sapiens* y la plataforma utilizada es: Affymetrix Human Genome U133 Plus 2.0 Array.

El propósito de seleccionar este dataset es incluir un dataset con un número de muestras no muy elevado (seis), el cual se divide en dos grupos y utiliza una plataforma basada en el genoma del ser humano.

Tabla 3. Archivo targets dataset 3.

Filename	Group	Shortname	Colour
GSM1536990_DMSO_Rep1.CEL	DMSO	1	Red
GSM1536991_DMSO_Rep2.CEL	DMSO	2	Red
GSM1536992_DMSO_Rep3.CEL	DMSO	3	Red
GSM1536993_SRPIN803_Rep1.CEL	SRPIN803	S1	Green
GSM1536994_SRPIN803_Rep2.CEL	SRPIN803	S2	Green
GSM1536995_SRPIN803_Rep3.CEL	SRPIN803	S3	Green

5.3 Dataset 3

[https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42771]

Título del estudio: Microarray gene expression profiling of kinase-dependent and kinase-independent effects of GRK2[31].

El objetivo de este estudio es comprender el papel de la inhibición de la proteína-G acoplada al receptor quinasa 2 (GRK2), ya que, a pesar de ser una

opción utilizada para el tratamiento de la insuficiencia cardíaca, se desconocen los mecanismos subyacentes a la cardioprotección. Además, se considera que esta inhibición podría estar relacionada con la proliferación de células tumorales.

Es un dataset formado por doce muestras, divididas en tres grupos de cuatro muestras cada uno: 'NOD_Scid_expanded', 'cultured' y 'Re_cult_NOD_expans'. (Véase el archivo de targets en la tabla 4)

El organismo modelo es *Homo sapiens* y la plataforma utilizada es: Affymetrix Human Genome U133 Plus 2.0 Array.

El propósito de seleccionar este dataset es incluir un dataset con un número de muestras elevado (doce), el cual se divide en tres grupos, de esta forma se podrán realizar comparaciones entre los diversos grupos dos a dos, es decir, el grupo 1 con el grupo 2, el grupo 1 con el grupo 3 y el grupo 2 con el grupo 3.

Tabla 4. Archivo targets dataset 3.

Filename	Group	Shortname	Colour
GSM1049708_Scid-GRK-1.CEL	NOD_Scid_expanded	GRK21	Red
GSM1049709_Scid-GRK-2.CEL	NOD_Scid_expanded	GRK22	Red
GSM1049710_Scid-K220R-1.CEL	NOD_Scid_expanded	GRK2-K220R1	Red
GSM1049711_Scid-K220R-2.CEL	NOD_Scid_expanded	GRK2-K220R2	Red
GSM1049712_HEK-GRK-1.CEL	cultured	GRK23	Yellow
GSM1049713_HEK-GRK-2.CEL	cultured	GRK24	Yellow
GSM1049714_HEK-K220R-1.CEL	cultured	GRK2-K220R3	Yellow
GSM1049715_HEK-K220R-2.CEL	cultured	GRK2-K220R4	Yellow
GSM1049716_Ex-Scid-GRK-1.CEL	Re_cult_NOD_expans	GRK25	Green
GSM1049717_Ex-Scid-GRK-2.CEL	Re_cult_NOD_expans	GRK26	Green
GSM1049718_Ex-Scid-K220R-1.CEL	Re_cult_NOD_expans	GRK2-K220R5	Green
GSM1049719_Ex-Scid-K220R-2.CEL	Re_cult_NOD_expans	GRK2-K220R6	Green

6. Desarrollo del pipeline

Para llevar a cabo el desarrollo de este pipeline se ha utilizado el dataset 1 y el software de libre distribución R, el cual proporciona diversos paquetes que resultan útiles para llevar a cabo de forma sencilla el análisis de datos de microarrays. Se utilizan los siguientes:

- *Oligo*: El paquete 'oligo' permite leer los datos asociados a cada fichero '.CEL', es decir, permite la lectura de la intensidad de los arrays[32].
- *Annotate*: Este paquete permite asociar cada gen symbol a la nomenclatura especificada en el chip correspondiente.
- *Ggplot2*: Utilizando este paquete podremos realizar gran parte de los gráficos en los que analizaremos y controlaremos la calidad de los datos.
- *RColorBrewer*: Este paquete permite asociar a cada muestra del experimento un color, lo cual resulta útil para comprender con mayor facilidad los gráficos.
- *Limma*: Es un paquete que usa modelos lineales para el análisis de diseños experimentales mediante expresión diferencial.

En un pipeline para el análisis de microarrays existen 3 pasos principales que se deben seguir: lectura de los datos, control de calidad y normalización y selección de genes diferencialmente expresados. Por tanto, en primer lugar, se procede al desarrollo teórico del pipeline[33,34,35]:

- *Lectura de los datos:* Este punto consiste, principalmente, en cargar los archivos '.CEL' del dataset seleccionado y el archivo de covariables (target). Para la lectura de los archivos '.CEL' se utiliza la función 'read.celfiles()' del paquete 'oligo' de R, mientras que para la lectura del target se utiliza la función 'read.csv' o 'read.table'.
- *Control de calidad y normalización:* En este apartado se va a llevar a cabo el control de calidad de los datos, es decir, comprobar si existen posibles fallos en el escaneo de la imagen, como ruido del bias, bloques dañados, etc. En cuanto a la normalización se desarrolla el método de

normalización RMA (en el apartado 2.1.1.b se explica el motivo de elegir este método), el cual se ejecuta mediante la función ‘rma()’ del paquete ‘oligo’. Para llevar a cabo el control de calidad, teniendo en cuenta que debe consumir el mínimo número de recursos posibles ya que después habrá que ejecutar este pipeline en una aplicación web, se seleccionan los siguientes gráficos: dendogramas (para comprobar que las relaciones entre las distintas muestras se corresponden con las incluidas en el target), histogramas y boxplots (para comparar las distribuciones que siguen cada uno de los chips).

- *Selección de genes diferencialmente expresados:* Para llevar a cabo este punto se deben realizar las siguientes acciones: crear la matriz de diseño mediante la función ‘model.matrix()’ del paquete ‘limma’, la cual es una matriz que contiene los grupos en que se clasifica la muestra, posteriormente se utiliza el método ‘lmFit()’ que ajusta un modelo lineal a los datos para calcular el nivel de expresión media en las muestras. Posteriormente, se crea la matriz de contrastes que básicamente sirve para indicar que grupos se quieren comparar, mediante las funciones ‘makeContrasts()’ y ‘contrasts.fit()’; como en este caso solo existen dos grupos en el dataset (inducidos, no-inducidos) solo se pueden comparar entre ellos. A continuación, se realiza el test estadístico (t-test especial para el análisis de microarrays, el cual utiliza un modelo bayesiano para compensar el bajo número de réplicas) para comprobar la expresión diferencial de los genes mediante la función ‘eBayes()’ que devuelve un data frame con los valores asignados a cada gen. Tras esto, se desarrolla un volcano plot para representar aquellos genes más diferencialmente expresados y con mayor cambio biológico. El siguiente paso es representar en una tabla aquellos genes diferencialmente expresados que superen un cierto umbral, para ello se utiliza la función ‘topTable()’, el umbral que se establece en este pipeline para este dataset es lfc = abs(3) (log fold change) y un p-valor ajustado menor de 0.001. Una vez se ha realizado este paso, se asigna a cada gen su símbolo mediante la base de datos correspondiente a la plataforma que se ha utilizado, en este caso la plataforma es ‘Affymetrix Mouse Gene 1.0 ST Array [transcript (gene) versión]’ y la base de datos asociada es

‘mogene10sttranscriptcluster’. También se hace una división para distinguir entre aquellos genes que están sobre-expresados e infra-expresados; para esto, se selecciona aquellos genes que tienen un logFC mayor que 1 como sobre-expresados e infra-expresados si el logFC es menor que -1. Finalmente, se genera un Heat Map que represente gráficamente la expresión de los genes sobre-expresados e infra-expresados en las distintas muestras.

A partir de este desarrollo teórico, se procede a implementarlo en R[36,37,38]. Sin embargo, este pipeline no es definitivo y, si bien cumple su función, deberá ser adaptado para la aplicación final desarrollada con Shiny, de manera que sea capaz de adaptarse a cualquier dataset que se introduzca. El código se encuentra en el anexo II.

7. Desarrollo de la aplicación web

El desarrollo de una aplicación web puede ser llevado a cabo a través de una gran cantidad de lenguajes de programación como podrían ser JavaScript, Python, ASP, Ruby, PHP, … Sin embargo, el lenguaje escogido para desarrollar esta aplicación es R y su paquete para el desarrollo de aplicaciones web 'Shiny', debido a que el pipeline diseñado para el análisis de datos procedentes de microarrays ha sido desarrollado en R. Además, R posee el paquete 'Bioconductor' el cual proporciona una gran cantidad de funciones para el análisis de datos biológicos. El propósito de esta aplicación es facilitar a cualquier persona que no posea un amplio conocimiento de programación el análisis de datos de microarrays, por tanto, es conveniente que la interfaz sea simple e intuitiva y el 'Dashboard' de Shiny permite cumplir este requerimiento.

7.1 R y Bioconductor

R es un lenguaje de programación dirigido, principalmente, al análisis estadístico. Sin embargo, es ampliamente usado en minería de datos, investigación biomédica y bioinformática. Se distribuye mediante licencia GNU GPL y está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. R fue desarrollado por Robert Gentleman y Ross Ihaka en el Departamento de Estadística de la Universidad de Auckland en 1993[39]. La última versión estable de R es la 3.4.0 (21/04/2017).

Bioconductor [https://www.bioconductor.org/] es un proyecto de código abierto que proporciona herramientas para el análisis y comprensión de datos genómicos[40]. Está basado en el lenguaje de programación R. El proyecto Bioconductor comenzó en 2001 y es supervisado por el Roswell Park Cancer Institute y por otros miembros procedentes de instituciones estadounidenses e internacionales. La última versión estable es la 3.5 (25/04/2017).

7.2 Shiny

Shiny es un paquete 'open source' de R, que permite desarrollar aplicaciones web usando código de R[41]. Shiny posee una gran variedad de widgets para crear de forma rápida interfaces. Sin embargo, Shiny es muy extensible y es fácil integrar contenido web utilizando HTML, CSS, JavaScript y jQuery[42,43,44].

La estructura de un programa desarrollado con Shiny es sencilla, el programa se divide en dos partes, que pueden localizarse en dos archivos distintos (server.R y ui.R) o en un mismo archivo (app.R). La parte o archivo 'ui' se centra en crear la interfaz gráfica, es decir, el resultado visual de la aplicación, mientras que el archivo o parte 'server' se centra en realizar las operaciones y procesamiento de los datos[45,46,47].

El 'dashboard' de Shiny permite desarrollar aplicaciones con una potente estética visual y mayor funcionalidad[48].

7.3 Desarrollo de la aplicación

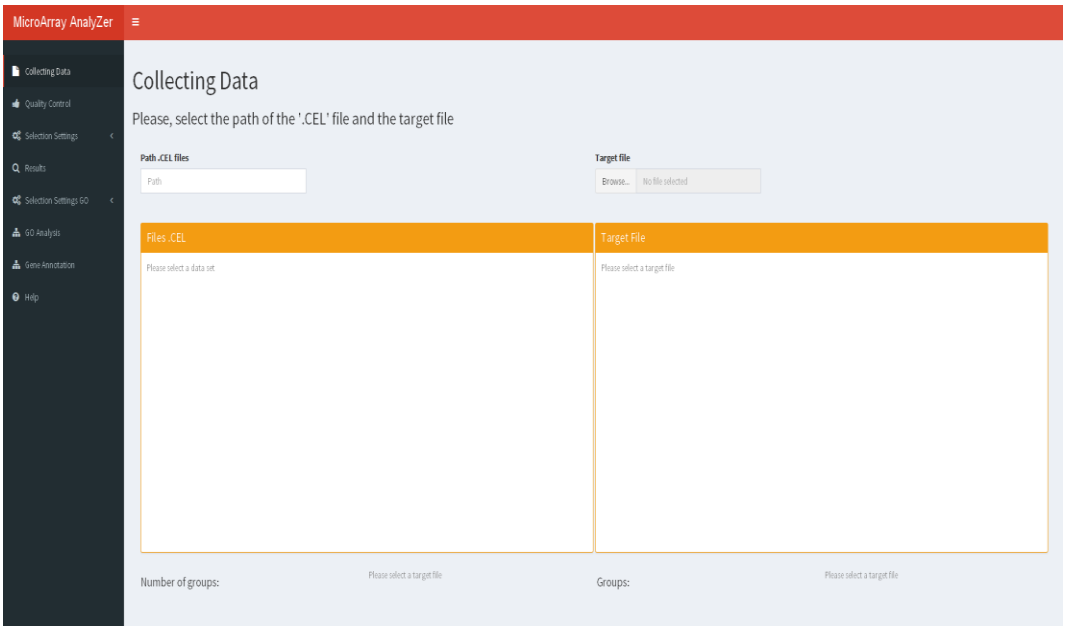
El código de la aplicación puede encontrarse en el anexo III.

La aplicación desarrollada posee los siguientes componentes:

- *Collecting data:* una sección dirigida a leer los archivos suministrados (figura 9). Existen dos procesos clave en este punto, la lectura de los archivos '.CEL', los cuales proporcionan la información del experimento recogida por el chip, y la lectura del target o archivo phenodata, el cual debe indicar la información global del experimento, es decir, los ficheros '.CEL' que lo conforman, los grupos en que se han clasificado las muestras, el nombre asignado a cada muestra y el color en el cual se representara esta muestra a la hora de realizar el control de calidad. Para la lectura de los archivos '.CEL' se ha habilitado un cuadro de texto donde se pueda introducir la ruta o directorio en el cual se encuentran los archivos '.CEL'. Una vez se ha introducido, se procede a la lectura de los archivos y, además, se mostrará por pantalla el nombre de los archivos para comprobar que no hay ningún error. Para el 'upload' del archivo phenodata se ha establecido una entrada de archivos. De este

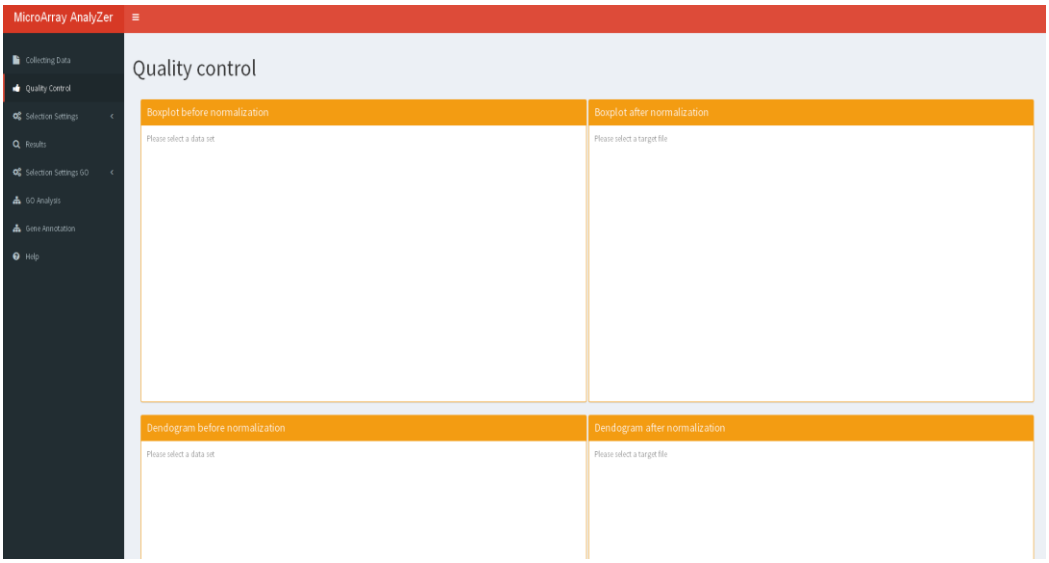
modo una vez que el archivo phenodata (el cual debe tener extensión .csv y una estructura acorde con la mostrada en la ayuda) se haya cargado, se mostrará por pantalla en formato de tabla.

Figura 9. Aspecto visual herramienta 'Collecting Data'.



- *Quality control:* un apartado que realice el control de calidad. En este punto se ha considerado el evitar tanto la redundancia de información, como podría ser el caso de mostrar histogramas junto con boxplots, como un elevado tiempo de procesamiento, por ejemplo, al cargar MA plots. Por estos factores, se ha decidido realizar el control de calidad en base a boxplots y dendogramas, antes y después de la normalización de los datos. Los boxplots darán una idea de la distribución de las intensidades, mientras que los dendogramas o clústeres jerárquicos llevan a cabo una agrupación de las muestras por grado de similaridad, lo que debería resultar en una agrupación similar a la realizada según las condiciones experimentales.

Figura 10. Aspecto visual herramienta Quality Control.



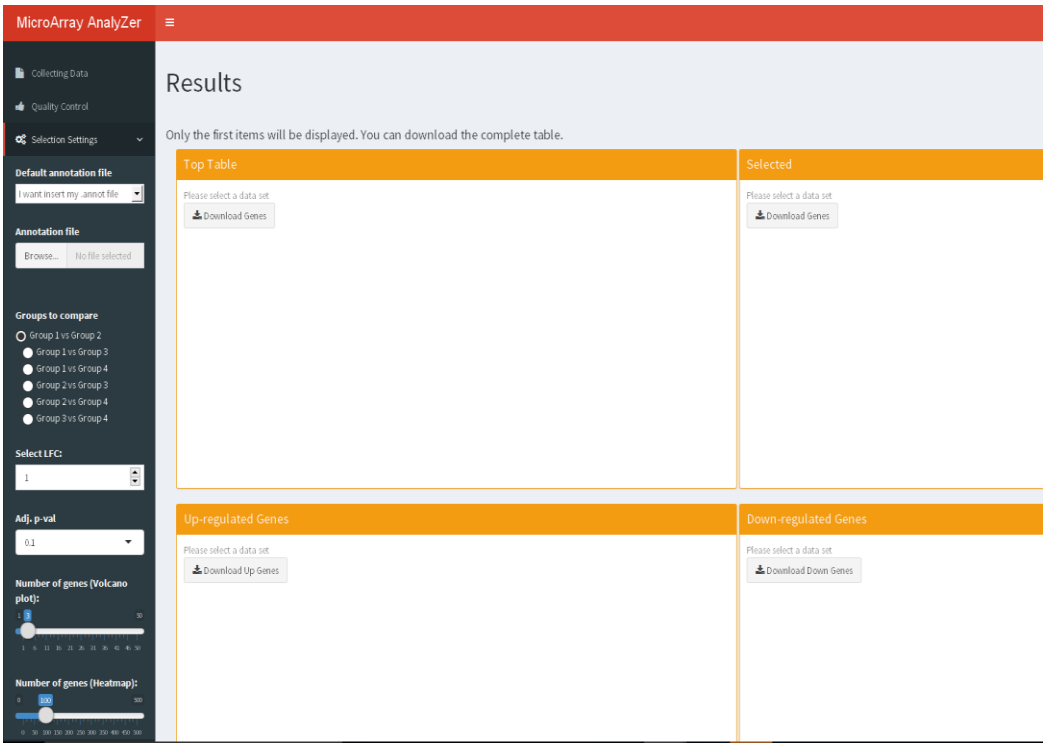
- *Selection settings:* una sección que permite ajustar los distintos parámetros que serán tenidos en cuenta a la hora de realizar la selección de genes diferencialmente expresados. Entre estos parámetros se pueden encontrar:
    - Selección del archivo de anotación. Para llevar a cabo la anotación de los genes, se han incluido dos posibilidades: cargar el archivo '.annot' que se puede descargar a partir de cualquier dataset presente en la plataforma NCBI, o cargar los ficheros de anotación predeterminados:
        - "mogene10sttranscriptcluster.db"
        - "hgu133a.db"
        - "hugene21sttranscriptcluster.db"
        - "hugene20sttranscriptcluster.db"
        - "clariomdhumantranscriptcluster.db"
        - "clariomshumanhttranscriptcluster.db"
        - "clariomshumantranscriptcluster.db"
        - "clariomsmousehttranscriptcluster.db"
        - "clariomsmousetranscriptcluster.db"
    - Elección de los grupos a comparar. Se añade la función de analizar en función de los grupos de las muestras, es decir, se puede realizar la comparación 2 a 2 de un total de hasta cuatro grupos distintos.

- Selección del Fold Change. Para ajustar el valor que se desea restringir a partir del cambio en intensidades de fluorescencia.
- Selección del p-valor ajustado. Para establecer una mayor seguridad en los resultados devueltos por el test estadístico al analizar la expresión diferencial.
- Selección del número de genes que se debe mostrar en el Volcano Plot.
- Selección del número de genes que se debe mostrar en el Heat Map.

- *Results:* una sección que mostrará por pantalla los resultados del análisis (figura 10). Se mostrarán los genes que se hayan utilizado para realizar el análisis (‘Top Table’), los genes que presentan una expresión diferencial en función de los parámetros seleccionados en el apartado anterior (‘Selected’), aquellos genes sobre-expresados (‘Up-regulated genes’) o infra-expresados (‘Down-regulated genes’), un Volcano Plot que muestre los genes en función de los cambios de expresión/p-valor y un ‘Heatmap’, en el que se visualizará las expresiones de cada gen agrupándolas en función de la muestra.

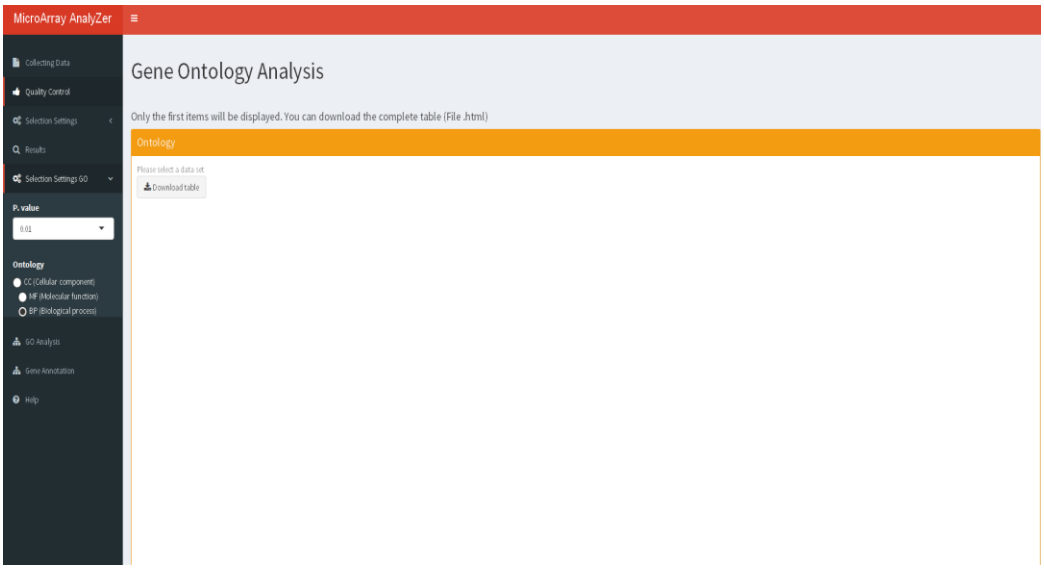
  En la aplicación solo se muestran por pantalla los 10 primeros resultados de las tablas mencionadas (‘Top Table’, ‘Selected’, ‘Up-regulated genes’ y ‘Down-regulated genes’), sin embargo, se establece la posibilidad de descargar un archivo con todos los resultados producidos.

Figura 11. Aspecto visual herramienta 'Results'.



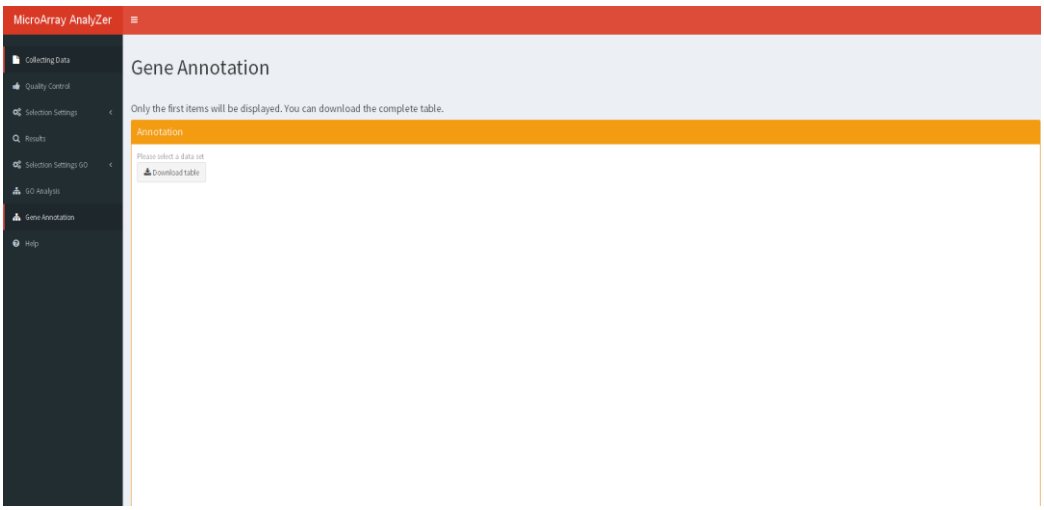
- *Selection settings GO:* una sección que permite ajustar los distintos parámetros que serán tenidos en cuenta a la hora de realizar el análisis de la ontología de los genes diferencialmente expresados. En estos parámetros se puede ajustar el p-valor que va a regir lo estricto que sea el análisis estadístico y el tipo de ontología, que puede ser:
    - CC (Cellular component)
    - MF (Molecular function)
    - BP (Biological process)
- *GO Analysis:* Esta sección permite establecer un análisis GO para estudiar a través de los genes seleccionados qué funciones están expresadas de forma diferencial con respecto al total de los genes analizados (figura 11). Para llevar a cabo este proceso, utilizo la librería 'GOstats', con la cual voy a ir comparando el nombre de los genes y asignándoles su función. En la aplicación se podrán ver los 20 primeros resultados y el resto se podrá descargar en un archivo '.html', el cual, contendrá un link por cada función que permitirá acceder a la base de datos de 'Gene Ontology' y poder recabar más información sobre ella.

Figura 12. Aspecto visual herramienta 'GO Analysis'.



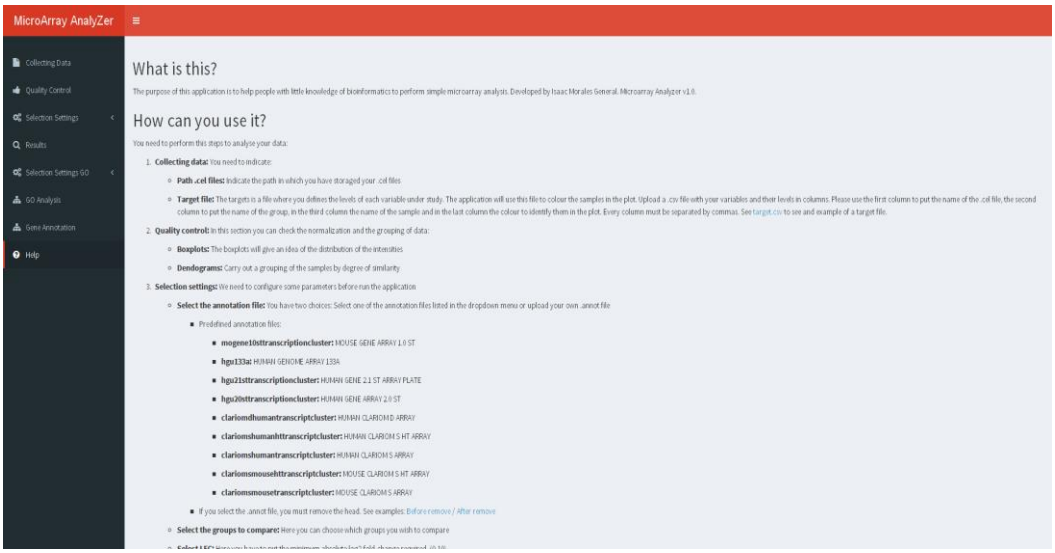
- *Gene Annotation:* En este apartado se determinan las funciones que realizan cada uno de los genes que han resultado seleccionados y se anota cada gen en diferentes bases de datos para que puedan ser fácilmente identificados (figura 12). Para llevar a cabo este proceso, utilizo la librería 'GO.db', con la cual voy a ir comparando el nombre de los genes y asignándoles su función. En la aplicación se podrán ver los 10 primeros resultados y el resto se podrá descargar en un archivo '.csv'.

Figura 13. Aspecto visual herramienta 'Gene Annotation'.



- *Help:* Se decide incluir un apartado en la aplicación que incluya una explicación sobre los procedimientos a llevar a cabo para realizar el análisis de los datos (figura 13).

Figura 14. Aspecto visual herramienta 'Help'.



7.4 Ejecución de la aplicación y batería de pruebas

A partir de los datasets seleccionados, se desarrolla la siguiente batería de pruebas que para comprobar el correcto funcionamiento de la aplicación:

- Lectura de los archivos '.CEL'
- Lectura del archivo de targets
- Visualización y comprobación de la normalización de los datos a través de los boxplots.
- Visualización y comprobación de una correcta agrupación de las muestras a través de los clústers jerárquicos.
- Comprobación de los resultados en la Top Table.
- Comprobación de los resultados en la Selected Table.
- Comprobación de los resultados en la Up-regulated Table.
- Comprobación de los resultados en la Down-regulated Table.
- Visualización del Volcano plot.
- Visualización del Heatmap.

- Comprobación de si existe una variación de los resultados al cambiar los grupos a comparar.
- Comprobación de si existe una variación de los resultados al cambiar el LFC.
- Comprobación de si existe una variación de los resultados al cambiar el valor del p-valor ajustado.
- Comprobación de si existe una variación en el Volcano Plot cambiar el número de genes que debe mostrar.
- Compruebo si existe una variación en el Heatmap al cambiar el número de genes que debe mostrar.
- Comprobación de que el cambio de anotación tenga un correcto funcionamiento.
- Comprobación de que al introducir el fichero '.annot' descargado del servidor NCBI funciones correctamente a la hora de realizar la anotación de los genes.
- Comprobación de que se realice correctamente el análisis GO.
- Comprobación de que se realice correctamente la anotación de las funciones para los genes seleccionados.
- Comprobación de que se realice correctamente la descarga de las siguientes tablas: Top Table, Selected Table, Up-regulated Table, Down-regulated Table, GO Analysis y Gene Annotation.

El resultado de la batería de pruebas para los datasets estudiados es favorable, no encontrando ningún problema en el análisis de los datos. El resumen de los resultados de la batería de pruebas puede observarse en la tabla 5. Los resultados detallados pueden observarse en el anexo IV.

Tabla 5. Resultados batería de pruebas.

PRUEBA	DATASET 1	DATASET 2	DATASET 3
Lectura de los archivos '.cel'	X	X	X
Lectura del archivo de targets	X	X	X
Visualización y comprobación de la normalización de los datos a través de boxplots	X	X	X
Visualización y comprobación de una correcta agrupación de las muestras a través de los clústers jerárquicos	X	X	X
Comprobación de los resultados en la Top Table	X	X	X
Comprobación de los resultados en la Selected Table	X	X	X
Comprobación de los resultados en la Up-regulated Table	X	X	X
Comprobación de los resultados en la Down-regulated Table	X	X	X
Visualización del Volcano Plot	X	X	X
Visualización de HeatMap	X	X	X
Compruebo si existe una variación de los resultados al cambiar los grupos a comparar	NE	NE	X
Compruebo si existe una variación de los resultados al cambiar el LFC	X	X	X
Compruebo si existe una variación de los resultados al cambiar el valor del p-valor ajustado	X	X	X
Compruebo si existe una variación en el Volcano Plot al cambiar el número de genes que debe mostrar	X	X	X
Compruebo si existe una variación en el HeatMap al cambiar el número de genes que debe mostrar	X	X	X
Compruebo que el cambio de anotación tenga un funcionamiento correcto	X	X	X
Compruebo que al introducir el fichero '.annot' descargado del servidor NCBI, funciona correctamente la anotación de los genes	X	X	X
Compruebo que se realice correctamente el análisis GO	X	X	X
Compruebo que se realice correctamente la anotación de las funciones para los genes diferencialmente expresados	X	X	X
Compruebo que se realice correctamente la descarga de las tablas: Top Table, Selected Table, Up-regulated Table, Down-regulated Table, Analysis GO, Gene Annotation.	X	X	X
Ejecución global satisfactoria	X	X	X

NE = No evaluable

8. Conclusiones

El principal interés para realizar este trabajo ha sido el poder aprender sobre el desarrollo de pipelines para el análisis de datos biológicos, en este caso de microarrays, y su posterior implementación en herramientas web que faciliten el acceso al análisis de estos datos al resto de la comunidad científica. En este trabajo, además, he aprendido como realizar correctamente el control de calidad para los datos resultantes del análisis de microarrays, las distintas formas de llevar a cabo el análisis de microarrays, es decir, los distintos tipos de normalización de los datos, qué pruebas estadísticas se pueden llevar a cabo para obtener los genes expresados diferencialmente, como realizar un análisis de la 'Gene Ontology' para comprobar aquellas funciones cuya participación es diferencial con el resto. Además, he aprendido a utilizar la herramienta 'Shiny' del software estadístico R, la cual permite crear herramientas web utilizando código de R.

Una vez completado tanto el proceso de desarrollo del pipeline para el análisis de datos procedentes de microarrays como la aplicación que permitirá llevar a cabo estos análisis de forma sencilla para aquellas personas que no tengan un amplio conocimiento de programación, se puede concluir que los objetivos propuestos en el plan de trabajo han sido alcanzados.

En cuanto a la planificación del trabajo, considero que ha sido llevada a cabo de forma adecuada, por lo que no se han excedido los plazos para cada una de las tareas propuestas. Sin embargo, la búsqueda de bibliografía ha sido constante y no se ha podido limitar a una tarea al comienzo del proyecto, ya que, a medida que se iba desarrollando tanto el pipeline como la aplicación, surgían nuevas dudas que implicaban la búsqueda de bibliografía adicional para resolverlas. En mi opinión, la parte que ha requerido un mayor esfuerzo ha sido el desarrollo del pipeline, ya que al ser un aspecto que desconocía hasta empezar este proyecto y, conjuntamente, la asignatura de análisis de datos ómicos, he requerido un mayor esfuerzo para comprender el objetivo de cada etapa del análisis.

A pesar de haber alcanzado los objetivos planteados al comienzo del proyecto, durante el desarrollo de la aplicación he considerado que hay posibilidades de ampliarla, de forma que se pueda realizar un control de calidad más exhaustivo (lo cual no se ha realizado para no aumentar el coste computacional de la aplicación), ampliar el catálogo de archivos de anotación predeterminados, ampliar el número de grupos que se pueden comparar, realizar distintos tipos de normalización de los datos (ya que se ha escogido RMA por ser el más utilizado actualmente, pero se podía haber ampliado con GCRMA, MAS, …) y, en último término, la aplicación podría ampliarse de forma que no se concentrase únicamente en el análisis de microarrays, sino llevar a cabo análisis de RNA-seq y NGS.

9. Glosario

Affymetrix: compañía estadounidense, pionera en el diseño de chips de ADN.

Bioconductor: proyecto de código abierto para el análisis de datos en genómica. Está basado en el lenguaje de programación R.

Dataset: es una colección de datos. Representada en forma de matriz, en la que cada columna representa una variable particular y cada fila un miembro del dataset.

Expresión diferencial: proceso por el cual, ante determinadas condiciones ambientales, unos genes se expresan más o menos que el resto.

Gen: unidad de información del ADN que codifica un producto funcional (ARN o proteínas).

GO: El proyecto Ontología Génica (en inglés Gene Ontology) provee un vocabulario que describe el gen y los atributos del producto génico en cualquier organismo.

Microarray: es una superficie sólida a la cual se une una colección de fragmentos de ADN.

Pipeline: conjunto de procesos que van transformando los datos, siendo la entrada de cada uno de estos procesos la salida del anterior.

R: lenguaje de programación orientado al análisis estadístico.

RMA: algoritmo utilizado para crear una matriz de expresión a partir de datos Affymetrix.

Shiny: herramienta para crear fácilmente aplicación web interactivas que permiten a los usuarios interactuar con sus datos sin necesidad de manipular el código fuente.

10. Bibliografía

[1]. M. Carme Ruíz de Villa, Alex Sánchez-Pla. Análisis de datos de microarrays (2013). Editorial UOC.

[2]. Watson, J. D., & Crick, F. H. (1974). Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. JD Watson and FHC Crick. Published in Nature, number 4356 April 25, 1953. Nature, 248(5451), 765.

[3]. Cañedo Andalia, R., & Guerrero Pupo, J. C. (2005). Nociones de bioquímica y genética útiles para los profesionales de la información del sector de la salud. ACIMED, 13(1), 1-1.

[4]. Gilbert, W. (1978). Why genes in pieces? Nature, 271(5645), 501-501.

[5]. Crick, F. H. (1958, January). On protein synthesis. In Symp Soc Exp Biol (Vol. 12, No. 138-63, p. 8)

[6]. Crick, F. (1970). Central dogma of molecular biology. Nature, 227(5258), 561-563.

[7]. Hesper, B., & Hogeweg, P. (1970). Bioinformatica: een werkconcept. Kameleon, 1(6), 28-29.

[8]. Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology, 48(3), 443-453.

[9]. Staden, R. (1977). Sequence data handling by computer. Nucleic Acids Research, 4(11), 4037-4052.

[10]. Wüthrich, K., Wider, G., Wagner, G., & Braun, W. (1982). Sequential resonance assignments as a basis for determination of spatial protein structures by high resolution proton nuclear magnetic resonance. Journal of molecular biology, 155(3), 311-319.

[11]. Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. Journal of molecular biology, 147(1), 195-197.

[12]. Wilbur, W. J., & Lipman, D. J. (1983). Rapid similarity searches of nucleic acid and protein data banks. Proceedings of the National Academy of Sciences, 80(3), 726-730.

[13]. Pearson, W. R., & Lipman, D. J. (1988). Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences, 85(8), 2444-2448.

[14]. Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. Bulletin of mathematical biology, 51(1), 79-94.

[15]. Cravedi, K. (2008). GenBank Celebrates 25 Years of Service with Two-Day Conference. Leading Scientists Will Discuss the DNA Database at April 7–8 Meeting. National Institutes of Health.

[16]. Bairoch, A. (1994). The ENZYME data bank. Nucleic acids research, 22(17), 3626-3627.

[17]. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.

[18]. Thompson, J. D., Higgins, D. G., Gibson, T.J. (1994). «CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice». Nucleic Acids Research 22 (22): 4673-80.

[19]. Luscombe, N. M., Greenbaum, D., & Gerstein, M. (2001). What is bioinformatics? An introduction and overview. Yearbook of Medical Informatics, 1(83-100), 2.

[20]. Babu, M. M. (2004). Introduction to microarray data analysis. Computational genomics: Theory and application, 17(6), 225-49.

[21]. Maestre, J. G. (2010). Análisis de datos de MicroArrays (Doctoral dissertation). [Última consulta: 14 de mayo de 2017] <https://riunet.upv.es/bitstream/handle/10251/8578/Memoria%20PFC%20Jose%20Gonzalez.pdf>.

[22]. Draghici, S., Khatri, P., Eklund, A. C., & Szallasi, Z. (2006). Reliability and reproducibility issues in DNA microarray measurements. TRENDS in Genetics, 22(2), 101-109.

[23]. Zeisel, A., Amir, A., Köstler, W. J., & Domany, E. (2010). Intensity dependent estimation of noise in microarrays improves detection of differentially expressed genes. BMC bioinformatics, 11(1), 400.

[24]. González J (2010) Análisis de datos de "microarrays". Universidad Politécnica de Valencia. Ibime. Informática Biomédica. [Última consulta: 14 de mayo de 2017] <http://tauja.ujaen.es/bitstream/10953.1/542/1/TFG_TorresGodino%2CIsabel.pdf>

[25]. Wu, Z. (2009). A review of statistical methods for preprocessing oligonucleotide microarrays. Statistical methods in medical research, 18(6), 533-541.

[26]. Tsai, C. A., Chen, Y. J., & Chen, J. J. (2003). Testing for differentially expressed genes with microarray data. Nucleic acids research, 31(9), e52-e52.

[27]. Westfal, P., & Young, L. S. (1993). Resampling-based multiple testing. [Última consulta: 14 de mayo de 2017] <http://statistics.berkeley.edu/sites/default/files/tech-reports/633.pdf>

[28]. Good P: Permutation Tests, 2nd ed. New York, Springer, 2000. [Última consulta: 14 de mayo de 2017] <http://sankhya.isical.ac.in/search/63a1/63a1_rev2.pdf>

[29]. Caiazzo, M., Dell'Anno, M. T., Dvoretskova, E., Lazarevic, D., Taverna, S., Leo, D., ... & Russo, G. (2011). Direct generation of functional

dopaminergic neurons from mouse and human fibroblasts. Nature, 476(7359), 224-227.

[30]. Morooka, S., Hoshina, M., Kii, I., Okabe, T., Kojima, H., Inoue, N., ... & Ninomiya, K. (2015). Identification of a dual inhibitor of SRPK1 and CK2 that attenuates pathological angiogenesis of macular degeneration in mice. Molecular pharmacology, 88(2), 316-325.

[31]. Fu X, Koller S, Abd Alla J, Quitterer U. Inhibition of G-protein-coupled receptor kinase 2 (GRK2) triggers the growth-promoting mitogen-activated protein kinase (MAPK) pathway. J Biol Chem 2013 Mar 15;288(11):7738-55.

[32]. Oligo User's Guide. (2015) [en línea]. Benilton S. Carvalho: BIOCONDUCTOR. [Última consulta: 20 de marzo de 2017]. <https://www.bioconductor.org/packages/release/bioc/vignettes/oligo/inst/doc/oug.pdf>

[33]. Sánchez, A., & de Villa, M. C. (2008). A tutorial review of microarray data analysis. Universitat de Barcelona. [Última consulta: 20 de marzo de 2017]. <http://www.ub.edu/stat/docencia/bioinformatica/microarrays/ADM/slides/A_Tutorial_Review_of_Microarray_data_Analysis_17-06-08.pdf>

[34]. Babu, M. M. (2004). Introduction to microarray data analysis. Computational genomics: Theory and application, 17(6), 225-49.

[35]. Santamaría Vicente, R. (2010). Visual analysis of gene expression data by means of biclustering. [Última consulta: 20 de abril de 2017]. <http://hdl.handle.net/10366/76569>

[36]. Using Bioconductor for Microarray Analysis (2016) [en línea]. BIOCONDUCTOR. [Última consulta: 20 de abril de 2017]. <https://www.bioconductor.org/help/workflows/arrays/>

[37]. Analysing microarray data in BioConductor (2011) [en línea]. THE BIOINFORMATICS KNOWLEDGEBLOG. [Última consulta: 26 de marzo de 2017]. <http://bioinformatics.knowledgeblog.org/2011/06/20/analysing-microarray-data-in-bioconductor/>

[38]. Analyze your own microarray data in R/Bioconductor (2015) [en línea]. VIB BIOINFORMATICS CORE WIKI. [Última consulta: 26 de marzo de 2017]. <http://wiki.bits.vib.be/index.php/Analyze_your_own_microarray_data_in_R/Bioconductor>.

[39]. A Brief History R: Past and Future History, Ross Ihaka, Statistics Department, The University of Auckland, Auckland, New Zealand. [Última consulta: 15 de mayo de 2017]. <https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf>

[40]. Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., ... & Hornik, K. (2004). Bioconductor: open software development for computational biology and bioinformatics. Genome biology, 5(10), R80.

[41]. Shiny by RStudio (2016) [en línea]. [Última consulta: 10 de abril de 2017] <http://shiny.rstudio-staging.com/tutorial/>

[42]. Arias, J., Giraldo, M., Montoya, V., Montoya, L., Usuga, O., & Hernández, F. (2016). Aplicación en Shiny para distribuciones muestrales [en línea]. [Última consulta: 10 de abril de 2017] <https://github.com/fhernanb/semilleroApps/tree/master/Dmuestrales>

[43]. Beeley, C. (2016). Web application development with R using Shiny. Packt Publishing Ltd.

[44]. Carmona, F. (2016). Taller 2: shiny: aplicaciones web interactivas con r. VI GENAEIO I 2015: VI Jornadas de enseñanza y aprendizaje de la estadística y la investigación operativa, 208, 21. [Última consulta: 10 de abril de 2017] <http://genaeio.seio.es/ACTAS_VI_JORNADAS_HUELVA_2015.pdf>

[45]. Mulero, J. (2016) [en línea]. Aplicaciones interactivas diseñadas con Shiny. Repositorio de la Universidad de Alicante. [Última consulta: 10 de abril de 2017] <http://hdl.handle.net/10045/54325>

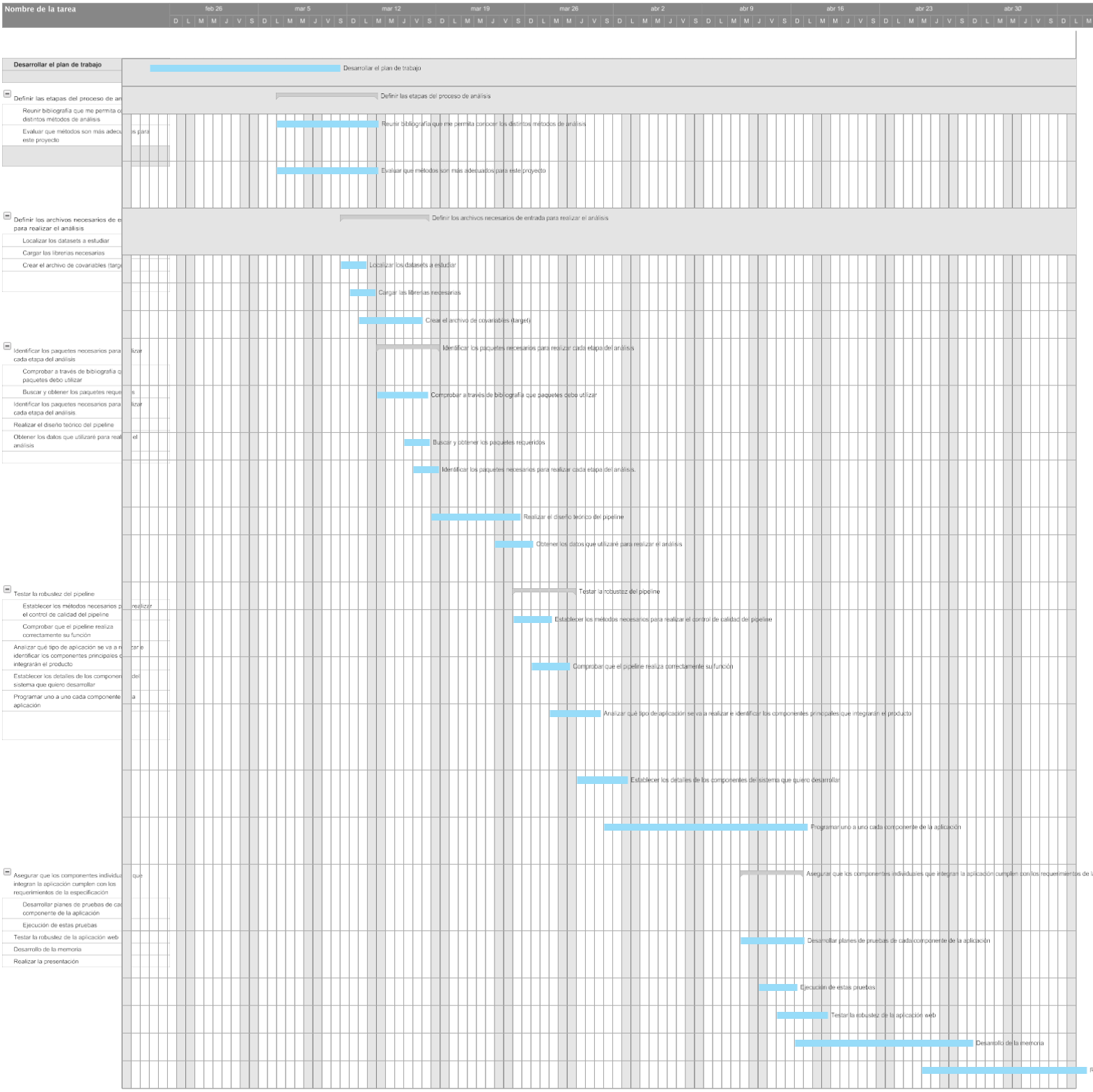
[46]. Wallinga M. (2013) [en línea]. Building an Interactive Online Fact Book with R Shiny. [Consulta: 10 de abril de 2017] <http://www.airum.org/docs/presentations/2013-wallinga-shiny.pdf>

[47]. Gómez Reverte, D. S., Molina Vila, M. D., Mulero, J., Nueda Roldán, M. J., & Pascual Romero, A. (2016). Aplicaciones diseñadas con Shiny: un recurso docente para la enseñanza de la estadística. [Última consulta: 10 de abril de 2017] <https://web.ua.es/va/ice/jornadas-redes-2016/documentos/tema-2/807250.pdf>

[48]. Winston Chang (2015) [en línea]. Dynamic dashboards with Shiny. [Última consulta: 10 de abril de 2017] <https://www.rstudio.com/resources/webinars/dynamic-dashboards-with-shiny/>

11. Anexos

Anexo I. Diagrama de Gantt



Anexo II. Código del pipeline

```
# CARGA DE PAQUETES NECESARIOS
library(oligo)
library(mogene10sttranscriptcluster.db)
library(annotate)
library(ggplot2)
library(limma)
# LECTURA DE LOS DATOS
celpath <- "E:/sync/MEGA/UOC/TFM/Dataset2/CEL"
list <- list.files(celpath,full.names=TRUE)
data <- read.celfiles(list)
phenodata <- read.csv("E:/sync/MEGA/UOC/TFM/Dataset2/target.csv", sep = ";",
header = TRUE)
# NORMALIZACIÓN
data.rma <- rma(data)
data.matrix <- exprs(data.rma)
# CONTROL DE CALIDAD
## Dendogramas
par(mfrow=c(1,2))
### Antes de normalización
eset <- exprs(data)
distance <- dist(t(eset), method = "maximum")
clusters <- hclust(distance)
plot(clusters, main="Antes de normalizar", labels = phenodata$Shortname)
### Después de normalización
eset_rma <- exprs(data.rma)
distance_rma <- dist(t(eset_rma), method = "maximum")
clusters_rma <- hclust(distance_rma)
plot(clusters_rma, main="Después de normalizar", labels =
phenodata$Shortname)
## Boxplot
par(mfrow=c(1,2))
### Antes de normalización
boxplot(data,which='pm',col=phenodata$Color,names=phenodata$Shortname,
target='core', main="Antes de normalizar")
### Después de normalización
boxplot(data.matrix,names=phenodata$Shortname,
col=phenodata$Color,main="Después de normalizar")
```

```
## Histograma
### Antes de normalización
hist(data[,1:8],lwd=2,which='pm',col=phenodata$Color,ylab='Density',xlab='Log
2 intensities', main='Histograma antes de normalizar', target='core')
legend('topright', legend=phenodata$Shortname , lty=1, col=phenodata$Color,
bty='n', cex=.75)
### Después de normalización
plotDensities(data.rma,main="Después de normalizar", legend= FALSE)
legend('topright', legend=phenodata$Shortname , lty=1, col=phenodata$Color,
bty='n', cex=.75)
# SELECCIÓN DE GENES DIFERENCIALMENTE EXPRESADOS
## Matriz de diseño
groups <- phenodata$Grupo
f <- factor(groups,levels=c("Induced","Not_induced"))
design <- model.matrix(~ 0 + f)
colnames(design) <- c("Induced","Not_induced")
data.fit <- lmFit(data.matrix,design)
## Contrastes
contrast.matrix <- makeContrasts(Induced-Not_induced,levels=design)
x <- c("Induced-Not_induced")
contrast.matrix <- makeContrasts(contrasts = x,levels=design)
data.fit.con = contrasts.fit(data.fit,contrast.matrix)
data.fit.eb = eBayes(data.fit.con)
## Volcano Plot
volcanoplot(data.fit.eb,coef=1,highlight=5, main = "Volcano Plot", names =
names(data.fit.eb$coefficients[,1]))
# Selección de genes
options(digits=2)
tab <- topTable(data.fit.eb,coef=1,number=1000, lfc=abs(3))
topgenes <- tab[tab[, "adj.P.Val"] < 0.001, ]
## Anotación de resultados
gene.symbols <- getSYMBOL(rownames(topgenes), "mogene10sttranscriptcluster")
results <- cbind(topgenes, gene.symbols)
## Up/Down regulated
topups = results[results[, "logFC"] > 1, ]
topups = na.omit(topups)
dim(topups)
head(topups)
topdowns = results[results[, "logFC"] < -1, ]
topdowns = na.omit(topdowns)
```

```
dim(topdowns)
head(topdowns)
## Heat Map
data.matrix.up <- data.matrix[(rownames(topdowns)),]
sampleNames <- vector()
featureNames <- vector()
heatlogs <- vector()
for (i in 1:8)
{
  sampleNames <- c(sampleNames,rep(phenodata$Shortname[i],dim(topdowns)[1]))
  featureNames                                                              <-
c(featureNames,rownames(data.matrix.up[1:dim(topdowns)[1],]))
  heatlogs <- c(heatlogs,data.matrix.up[1:dim(topdowns)[1],i])
}
vector = as.character(phenodata$Shortname)
for (i in 1:8)
{
  sampleNames<-replace(sampleNames,sampleNames==i,vector[i])
}
heatData                                                                    <-
data.frame(norm_logInt=heatlogs,sampleName=sampleNames,featureName=topdowns$g
ene.symbols)
dataHeat <- ggplot(heatData, aes(sampleName,featureName))
dataHeat            +            geom_tile(aes(fill=norm_logInt))            +
scale_fill_gradient(low="green", high="red")
# Gene Annotation
require(GOstats)
top <- tab
genes <- topups
entrezUniverse          =          unique(getEG(as.character(rownames(top)),
"mogene10sttranscriptcluster.db"))
geneIds<-                         unique(getEG(as.character(rownames(genes)),
"mogene10sttranscriptcluster.db"))
GOparams          =          new("GOHyperGParams",          geneIds=geneIds,
universeGeneIds=entrezUniverse,
                annotation="mogene10sttranscriptcluster.db", ontology="BP",
                pvalueCutoff=0.001, conditional=FALSE,
                testDirection="over")
KEGGparams              =              new("KEGGHyperGParams",geneIds=geneIds,
universeGeneIds=entrezUniverse,
```

```
annotation="mogene10sttranscriptcluster.db",
pvalueCutoff=0.01, testDirection="over")
GOhyper = hyperGTest(GOparams)
KEGGhyper = hyperGTest(KEGGparams)
## Se crea un informe html con los resultados
comparison = "topTab_LPS.in.AGED"
GOfilename=file.path("E:/", paste("GOResults.",comparison,".html", sep=""))
KEGGfilename=file.path("E:/",        paste("KEGGResults.",comparison,".html",
sep=""))
htmlReport(GOhyper, file = GOfilename, summary.args=list("htmlLinks"=TRUE))
htmlReport(KEGGhyper,             file             =             KEGGfilename,
summary.args=list("htmlLinks"=TRUE))
```

Anexo III. Código de la aplicación web

```
# This is a Shiny web application. You can run the application by clicking.
# This application allow you analyse microarray data.
# Author: Isaac Morales General
# Version: 1.0
# Required packages
library(shiny)
library(shinydashboard)
library(oligo)
library(annotate)
library(ggplot2)
library(limma)
library(GO.db)
# Max size Upload
options(shiny.maxRequestSize=50*1024^2)
ui <- dashboardPage(
  dashboardHeader(title = "MicroArray AnalyZer", titleWidth = 220),
  skin = "red",
  dashboardSidebar(
    width = 220,sidebarMenu(
      br(),
      menuItem("Collecting Data", tabName = "collecting", icon =
icon("file")),
      menuItem("Quality Control", tabName = "quality", icon = icon("thumbs-
up")),
      menuItem("Selection Settings", tabName = "adj", icon = icon("gears"),
               selectInput("annotation", "Default annotation file", choices =
list("I want insert my .annot file"="other",
"mogene10sttranscriptcluster"="mogene10sttranscriptcluster.db",
"hgu133a"="hgu133a.db",
"hugene21sttranscriptcluster"="hugene21sttranscriptcluster.db",
"hugene20sttranscriptcluster" = "hugene20sttranscriptcluster.db",
"clariomdhumantranscriptcluster" = "clariomdhumantranscriptcluster.db",
"clariomshumanhttranscriptcluster"="clariomshumanhttranscriptcluster.db",
"clariomshumantranscriptcluster"="clariomshumantranscriptcluster.db",
"clariomsmousehttranscriptcluster"="clariomsmousehttranscriptcluster.db",
"clariomsmousetranscriptcluster"="clariomsmousetranscriptcluster.db"),
```

```
selected = NULL, multiple = FALSE,selectize = FALSE, width = NULL, size =
NULL),
fileInput("ann", "Annotation file", multiple = FALSE),
radioButtons("groups", label="Groups to compare", inline = TRUE,
choices = list("Group 1 vs Group 2" = "1", "Group 1 vs Group 3" = "2", "Group
1 vs Group 4" = "4", "Group 2 vs Group 3" = "3", "Group 2 vs Group 4" = "5",
"Group 3 vs Group 4" = "6"), selected = 1),
numericInput("lfc", "Select LFC: ", value = 1, min = 0, max = 10, step = 1,
width = NULL),
selectInput("pval", "Adj. p-val", choices = list("0.001", "0.005",
"0.01","0.05","0.1","0.5","1"), selected = "0.1", multiple = FALSE,
selectize = TRUE, width = NULL, size = NULL),
sliderInput("volcano", "Number of genes (Volcano plot):", min = 1, max = 50,
value = 3),
sliderInput("max_genes", "Number of genes (Heatmap):", min = 0, max = 500,
value = 100) ),
menuItem("Results", tabName = "selection", icon = icon("search")),
menuItem("Selection Settings GO", tabName = "adjGO", icon = icon("gears"),
selectInput("pvalGO", "P. value", choices = list(0.001, 0.005,
0.01,0.05,0.1,0.5,1), selected = 0.01, multiple = FALSE,
selectize = TRUE, width = NULL, size = NULL),
radioButtons("ontol", label="Ontology", inline = TRUE,
choices = list("CC (Cellular component)" = "CC", "MF (Molecular function)" =
"MF", "BP (Biological process)" = "BP"), selected = "BP") ),
menuItem("GO Analysis", tabName = "ontology2", icon = icon("sitemap")),
menuItem("Gene Annotation", tabName = "ontology", icon = icon("sitemap")),
menuItem("Help", tabName = "help", icon = icon("question-circle")))),
# DASHBOARD BODY
dashboardBody(
 tabItems(
  tabItem(tabName = "collecting",
      h1("Collecting Data"),
      h3("Please, select the path of the '.CEL' file and the target file"),
      br(),
      fluidPage(splitLayout(
      textInput("CEL", "Path .CEL files", placeholder = "Path"),
      fileInput("target", "Target file", multiple = FALSE, accept =
c('text/csv','text/comma-separated-values,text/plain','.csv'), placeholder =
"No file selected"))),
      fluidPage(splitLayout(
```

```
      box(title = "Files .CEL", status = "warning", solidHeader = TRUE,
      tableOutput("celfiles"), height = 450, width = 1500),
      box(title = "Target File", status = "warning", solidHeader = TRUE,
      tableOutput("targetfile"), height = 450, width = 1500))),
      fluidPage(splitLayout(
      h4("Number of groups: "),
      textOutput("gro"),
      h4("Groups: "),
      textOutput("grou")))),
      tabItem(tabName = "quality",
      h1("Quality control"),
      br(),
      fluidPage(splitLayout(
      box(title = "Boxplot before normalization", status = "warning",
solidHeader = TRUE,
      plotOutput("box_bef"), height = 475, width = 600),
      box(title = "Boxplot after normalization", status = "warning",
solidHeader = TRUE,
       plotOutput("box_aft"), height = 475, width = 600))),
      fluidPage(splitLayout(
        box(title = "Dendogram before normalization", status = "warning",
solidHeader = TRUE,
        plotOutput("den_bef"), height = 475, width = 600),
        box(title = "Dendogram after normalization", status = "warning",
solidHeader = TRUE,
        plotOutput("den_aft"), height = 475, width = 600)))),
      tabItem(tabName = "selection",
       h1("Results"),
      br(),
      h4("Only the first items will be displayed. You can download the
complete table."),
     fluidPage(splitLayout(
    box(title = "Top Table",status = "warning", solidHeader = TRUE,
    tableOutput("selected1"), downloadButton("dwtable", "Download Genes"),
height = 450, width = 500),
    box(title = "Selected", status = "warning", solidHeader = TRUE,
    tableOutput("selected2"), downloadButton("dwtable2", "Download Genes"),
height = 450, width = 500))),
     fluidPage(splitLayout(
```

```
      box(title = "Up-regulated Genes", status = "warning", solidHeader =
TRUE,
     tableOutput("upgenes"), downloadButton("dwup", "Download Up Genes"),
height = 450, width = 500),
    box(title = "Down-regulated Genes", status = "warning", solidHeader =
TRUE,
      tableOutput("downgenes"),    downloadButton("dwdw",    "Download    Down
Genes"), height = 450, width = 500))),
     fluidPage(splitLayout(
     box(title = "Volcano Plot", status = "warning", solidHeader = TRUE,
     plotOutput("volcan"), height = 475, width = 500),
     box(title = "Heat Map", status = "warning", solidHeader = TRUE,
     plotOutput("heatm"), height = 475, width = 500))),
     textOutput("prueba")),
     tabItem(tabName = "ontology2",
    h1("Gene Ontology Analysis"),
     br(),
     h4("Only the first items will be displayed. You can download the
complete table (File .html)"),
  box(title = "Ontology", status = "warning", solidHeader = TRUE,
  tableOutput("go2") , downloadButton("dwgo2", "Download table"), height =
750, width = 1000)),
  tabItem(tabName = "ontology",
  h1("Gene Annotation"),
 br(),
  h4("Only the first items will be displayed. You can download the complete
table."),
  box(title = "Annotation", status = "warning", solidHeader = TRUE,
  tableOutput("go"), downloadButton("dwgo", "Download table"), height = 750,
width = 1000)),
 tabItem(tabName = "help",
 h2("What is this?"),
  p("The purpose of this application is to help people with little knowledge
of bioinformatics to perform simple microarray analysis. Developed by Isaac
Morales General. Microarray Analyzer v1.0."),
  h2("How can you use it?"),
 p("You need to perform this steps to analyse your data:"),
  tags$ol(
  tags$li(p(strong("Collecting data: "),"You need to indicate: "),
```

```
   tags$ul(tags$li(p(strong("Path .cel files: "), "Indicate the path in which
you have storaged your .cel files"))),
    tags$ul(tags$li(p(strong("Target file: "), "The targets is a file where
    you defines the levels of each variable under study. The application will
    use this file to colour the samples in the plot. Upload a .csv file  with
    your variables and their levels in columns. Please use the first column
    to put the name of the .cel file, the second column to put the name of
    the group, in the third column the name of the sample and in the last
    column the colour to identify them in the plot. Every column must be
    separated by commas. See"
      ,tags$a(href="target.csv", target="_blank", "target.csv"),"to see
and example of a target file."))),
      tags$li(p(strong("Quality control: "),"  In this section you can check
the normalization and the grouping of data: "),
      tags$ul(tags$li(p(strong("Boxplots:  "), "The boxplots will give an
idea of the distribution of the intensities"))),
   tags$ul(tags$li(p(strong("Dendograms:  "), "Carry out a grouping of the
samples by degree of similarity")))),
   tags$li(p(strong("Selection settings:  "), "We need to configure some
parameters before run the application"),
     tags$ul(tags$li(p(strong("Select the annotation file:"), "You have two
choices: Select one of the annotation files listed in the dropdown menu or
upload your own .annot file"))),
    tags$ul(tags$ul(tags$li(p("Predefined annotation files: "))),
    tags$ul(tags$ul(tags$li(tags$dd(p(strong("mogene10sttranscriptioncluster:
"), "MOUSE GENE ARRAY 1.0 ST"))))),
    tags$ul(tags$ul(tags$li(tags$dd(p(strong("hgu133a:   "),   "HUMAN   GENOME
ARRAY 133A"))))),
     tags$ul(tags$ul(tags$li(tags$dd(p(strong("hgu21sttranscriptioncluster:
"), "HUMAN GENE 2.1 ST ARRAY PLATE"))))),
     tags$ul(tags$ul(tags$li(tags$dd(p(strong("hgu20sttranscriptioncluster:
"), "HUMAN GENE ARRAY 2.0 ST"))))),
    tags$ul(tags$ul(tags$li(tags$dd(p(strong("clariomdhumantranscriptcluster:
"), "HUMAN CLARIOM D ARRAY"))))),
  tags$ul(tags$ul(tags$li(tags$dd(p(strong("clariomshumanhttranscriptcluster:
"), "HUMAN CLARIOM S HT ARRAY"))))),

      tags$ul(tags$ul(tags$li(tags$dd(p(strong("clariomshumantranscriptclust
      er:        "),        "HUMAN        CLARIOM        S        ARRAY"))))),
```

```
      tags$ul(tags$ul(tags$li(tags$dd(p(strong("clariomsmousehttranscriptclu
      ster: "), "MOUSE CLARIOM S HT ARRAY")))),
      tags$ul(tags$ul(tags$li(tags$dd(p(strong("clariomsmousetranscriptclust
      er: "), "MOUSE CLARIOM S ARRAY")))),
      tags$ul(tags$ul(tags$li(p("If you select the .annot file, you must
      remove the head. See examples: ",tags$a(href="annot11.jpg",
      target="_blank", "Before remove"),
      (" / ") ,tags$a(href="annot21.jpg", target="_blank", "After
      remove")))),
       tags$ul(tags$li(p(strong("Select the groups to compare:"), "Here you
can choose which groups you wish to compare"))),
       tags$ul(tags$li(p(strong("Select LFC:"), "Here you have to put the
minimum absolute log2-fold-change required. (0-10)")))),
      tags$ul(tags$li(p(strong("Select Adj. p-val:"), "Select the cutoff
value for adjusted p-values. Only genes with lower p-values are listed"))),
       tags$ul(tags$li(p(strong("Select the number of genes (Volcano
Plot):"), "Select the number of genes showed in the Volcano Plot"))),
       tags$ul(tags$li(p(strong("Select the number of genes (Heatmap):"),
"Select the number of genes showed in the Heatmap"))),
        tags$li(p(strong("Results: You can check the result of the
analysis"),": "),
       tags$ul(tags$li(p(strong("Top Table: "), "Genes used for the analysis.
Only the first items will be displayed. You can download the complete
table."))),
      tags$ul(tags$li(p(strong("Selected: "), "Genes that meet the specified
requirements. Only the first items will be displayed. You can download the
complete table."))),
      tags$ul(tags$li(p(strong("Up-regulated Genes: "), "Genes that meet the
specified requirements and are overexpressed. Only the first items will be
displayed. You can download the complete table."))),
                     tags$ul(tags$li(p(strong("Down-regulated Genes: "),
"Genes that meet the specified requirements and are under-expressed. Only the
first items will be displayed. You can download the complete table."))),
                     tags$ul(tags$li(p(strong("Volcano Plot: "), "Will show
the genes as a function of the expression changes and the p-value."))),
                     tags$ul(tags$li(p(strong("Heat Map: "), "Will display
the expressions of each gene grouping them according to the sample.")))),
                tags$li(p(strong("Selection Settings GO: "),"You need to
configure some parameter before run the GO analysis")),
```

```
                        tags$ul(tags$li(p(strong("Select the P. value:"),
"Select the cutoff value"))),
                        tags$ul(tags$li(p(strong("Select the Ontolgy:"),
"Select the Ontology which you wish analyse: "))),
                                tags$ul(tags$ul(tags$li(p("CC:      Cellular
Component")))),
                                tags$ul(tags$ul(tags$li(p("MF:     Molecular
Function")))),
                                tags$ul(tags$ul(tags$li(p("BP:    Biological
Process")))),
                tags$li(p(strong("GO Analysis: "),"Which functions are
differentially expressed with respect to the total of the analyzed genes.
** Only works with predefined annotation files **")),
                tags$li(p(strong("Gene Annotation: "),"The functions
performed by each of the genes that have been selected."))))))
# SERVER
server <- function(input, output) {
  # Read CEL files
  data <- reactive({
    validate(
      need(input$CEL != "", "Please select a data set")
    )
    list <- list.files(input$CEL,full.names=TRUE)
    read.celfiles(list)
  })
  # Read target file
  phenodata <- reactive({
    validate(
      need(input$target != "", "Please select a target file")
    )
    file1 = input$target
    data1 = read.csv(file1$datapath)
    return(data1)
  })
  # Process RMA
  data.rma <- reactive({
    d <- data()
    rma(d)
  })
  # Data Matrix
```

```
data.matrix <- reactive({
  d <- data.rma()
  exprs(d)
})
# Fit Data
data.fit <- reactive({
  f <- phenodata()
  groups <- f[,2]
  vect <- unique(groups)
  fac <- factor(groups,levels=vect)
  design <- model.matrix(~ 0 + fac)
  colnames(design) <- vect
  df <- lmFit(data.matrix(),design)
  return(df)
})
# Select groups to compare
gr_sel <- reactive({
  num <- 1
  control <- FALSE
  f <- phenodata()
  gro <- as.character(f[,2])
  vec <- unique(gro)
  cont <- length(vec)
  h <- ""
  v <- vector()
  m <- vector()
  for (i in 1:cont)
  {
    v[i] <- vec[i]
  }
  for (i in 1:cont)
  {
    for (j in cont:i)
    {
      if(v[i] != v[j])
      {
        h <- paste(v[i],v[j], sep = "-")
        m[num] <- h
        num <- num+1
      }
```

```
    }
  }
  return(m)
})
# eBayes to Fit Data
data.fit.eb <- reactive({
  f <- phenodata()
  groups <- f[,2]
  vect <- unique(groups)
  fac <- factor(groups,levels=vect)
  design <- model.matrix(~ 0 + fac)
  colnames(design) <- vect
  gru <- gr_sel()
  df <- lmFit(data.matrix(),design)
  contrast <- makeContrasts(contrasts = gru,levels=design)
  datafitcon = contrasts.fit(df,contrast)
  dfeb = eBayes(datafitcon)
  return(dfeb)
})
# Top Table
result <- reactive({
  options(digits=10)
  cf <- as.numeric(input$groups)
  d <- data()
  num <- nrow(d@featureData@data)
  topgenes <- topTable(data.fit.eb(),coef=cf, number = num)
  ID <- row.names(topgenes)
  topgenes <- cbind(topgenes, ID)
  annota <- input$annotation
  if(annota != "other") {
    library(annota, character.only = TRUE)
    contador <- nchar(annota)
    if(substr(annota, contador-2,contador) == ".db")
    {
      annota_lib <- substr(annota, 1, contador-3)
    }
    else {
      annota_lib <- annota
    }
    Gene.symbol <- getSYMBOL(rownames(topgenes), annota_lib)
```

```
    }
    else {
      annotacion <- input$ann
      data1  =  read.table(annotacion$datapath,  sep="\t",  header  =  FALSE,
fill=TRUE, stringsAsFactors = FALSE)
      data1 = data1[ ,c(1,3)]
      names(data1) <- c("ID", "Gene.symbol")
    }
    if(annota != "other") {
      results <- cbind(Gene.symbol, topgenes)
    } else {
      results <- merge(data1, topgenes, by = "ID")
    }
    results <- na.omit(results)
    row.names(results) <- results$ID
    return(results)
  })
  # Selected Genes
  selected <- reactive({
    tab <- result()
       lfc_n <- as.numeric(input$lfc) * -1
       lfc_p <- input$lfc
    max_gen <- input$max_genes
    topgen <- tab[tab[, "adj.P.Val"] < input$pval, ]
     topgen1 <- topgen[topgen[, "logFC"] < lfc_n,]
    topgen2 <- topgen[topgen[, "logFC"] > lfc_p, ]
    topgenes3 <- rbind.data.frame(topgen1,topgen2)
    topgenes3 <- topgenes3[order(topgenes3$adj.P.Val), ]
    topgen4 <- na.omit(topgenes3)
    return(topgen4)
  })
  # COLLECTING DATA
  # CEL
  output$gro <- renderText({
    f <- phenodata()
    grou <- as.character(length(levels(f[,2])))
    grou
  })

  output$grou <- renderText({
```

```
    f <- phenodata()
    gr <- as.character(levels(f[,2]))
    ob <- paste(gr, " ", sep = ";")
    ob
  })
  # SHOW CELFILES
  output$celfiles <- renderTable({
    validate(
      need(input$CEL != "", "Please select a data set")
    )
    l <- list.files(input$CEL,full.names=TRUE)
    h <- strsplit(l, "/", fixed=TRUE)
    Files.CEL <- vector()
    for(i in 1:length(h)){
      a <- length(h[[i]])
      Files.CEL <- c(Files.CEL, h[[i]][a])
    }
    as.data.frame(Files.CEL)
  })
  # PHENODATA
  output$targetfile <- renderTable({
    f <- phenodata()
    f
  })
  #BOXPLOT BEFORE
  output$box_bef <- renderPlot({
    d <- data()
    f <- phenodata()
    boxplot(d,which='pm',col=as.character(f[,4]),names=f[,3],  target='core',
main="Before", las=2)
  })
  #BOXPLOT AFTER
  output$box_aft <- renderPlot({
    f <- phenodata()
    dm <- data.matrix()
    boxplot(dm,names=f[,3], col=as.character(f[,4]),main="After", las=2)
  })
  # DENDOGRAM BEFORE
  output$den_bef <- renderPlot({
    d <- data()
```

```
    f <- phenodata()
    eset <- exprs(d)
    distance <- dist(t(eset), method = "maximum")
    clusters <- hclust(distance)
    plot(clusters, main="Before", labels = f[,3])
  })
  # DENDOGRAM AFTER
  output$den_aft <- renderPlot({
    f <- phenodata()
    dr <- data.rma()
    eset_rma <- exprs(dr)
    distance_rma <- dist(t(eset_rma), method = "maximum")
    clusters_rma <- hclust(distance_rma)
    plot(clusters_rma, main="After", labels = f[,3])
  })
  # VOLCANO PLOT
  output$volcan <- renderPlot({
    dfeb <- data.fit.eb()
    cf <- as.numeric(input$groups)
    annota <- input$annotation
    if(annota != "other") {
      a <- result()
      volcanoplot(dfeb,coef=cf,  highlight=input$volcano,  main  =  "Volcano
Plot", names = a$Gene.symbol)
   } else {
     volcanoplot(dfeb,coef=cf,  highlight=input$volcano,  main  =  "Volcano
Plot", names = names(dfeb$coefficients[,1]))
   }
    })
  # HEAT MAP
  output$heatm <- renderPlot({
    res0 <- selected()
    max_gen <- input$max_genes
    res <- res0[1:max_gen, ]
    res <- na.omit(res)
    res1 <- res[res[, "logFC"] < -1, ]
    res2 <- res[res[, "logFC"] > 1, ]
    res <- rbind.data.frame(res1,res2)
    f <- phenodata()
    dm <- data.matrix()
```

```
    dm2 <- dm[(rownames(res)),]
    sampleNames <- vector()
    featureNames <- vector()
    heatlogs <- vector()
    num_samples <- nrow(f)
    for (i in 1:num_samples)
    {
      sampleNames <- c(sampleNames,rep(f[i,3],dim(res)[1]))
      featureNames <- c(featureNames,rownames(dm2[1:dim(res)[1],]))
      heatlogs <- c(heatlogs,dm2[1:dim(res)[1],i])
    }
    vector = as.character(f[,3])
    for (i in 1:num_samples)
    {
      sampleNames<-replace(sampleNames,sampleNames==i,vector[i])
    }
    annota <- input$annotation
    if(annota != "other") {
      heatData                                                                    <-
data.frame(norm_logInt=heatlogs,sampleName=sampleNames,featureName=res$Gene.s
ymbol)
    }
    else {
      heatData                                                                    <-
data.frame(norm_logInt=heatlogs,sampleName=sampleNames,featureName=substr(res
$Gene.symbol, 1, 10))
    }
    dataHeat <- ggplot(heatData, aes(sampleName,featureName))
    dataHeat          +          geom_tile(aes(fill=norm_logInt))          +
scale_fill_gradient(low="green", high="red")
  })

  # GENE SELECTION (TopTable)
  output$selected1 <- renderTable( digits = -3,{
    res <- result()
    head(res, 10)
  })
  output$dwtable <- downloadHandler(filename = function() {
    "TableGenes.csv" },
    content = function(file) {
```

```
    write.csv(result(), file)
  })
# GENE SELECTION (SelectedGenes)
output$selected2 <- renderTable( digits = -3,{
  res2 <- selected()
  head(res2, 10)
})
output$dwtable2 <- downloadHandler(filename = function() {
  "TableGenes.csv" },
  content = function(file) {
    write.csv(selected(), file)
  })
# UP GENES SELECTION
output$upgenes <- renderTable(digits = -3,{
  res <- selected()
  topups = res[res[, "logFC"] > 1, ]
  dim(topups)
  head(topups)
})
output$dwup <- downloadHandler(filename = function() {
  "TableGenes.csv" },
  content = function(file) {
    res <- selected()
    topups = res[res[, "logFC"] > 1, ]
    write.csv(topups, file)
  })
# DOWN GENES SELECTION
output$downgenes <- renderTable(digits = -3,{
  res <- selected()
  topdown = res[res[, "logFC"] < -1, ]
  dim(topdown)
  head(topdown)
})
output$dwdw <- downloadHandler(filename = function() {
  "TableGenes.csv" },
  content = function(file) {
    res <- selected()
    topdown = res[res[, "logFC"] < -1, ]
    write.csv(topdown, file)
  })
```

```
# Gene Ontology
output$go2 <- renderTable({
  require(GOstats)
  annota <- input$annotation
  bd <- character()
  p <- as.numeric(input$pvalGO)
  if(annota != "other")
  {
    bd <- annota
  }
  top <- result()
  genes <- selected()
  entrezUniverse = unique(getEG(as.character(rownames(top)), bd))
  geneIds <- unique(getEG(as.character(rownames(genes)),bd))
  GOparams = new("GOHyperGParams",
                 geneIds=geneIds, universeGeneIds=entrezUniverse,
                 annotation=bd, ontology=input$ontol,
                 pvalueCutoff=p, conditional=FALSE,
                 testDirection="over")
  KEGGparams = new("KEGGHyperGParams",
                   geneIds=geneIds, universeGeneIds=entrezUniverse,
                   annotation=bd,
                   pvalueCutoff=p, testDirection="over")
  GOhyper = hyperGTest(GOparams)
  head(summary(GOhyper), 20)
})
output$dwgo2 <- downloadHandler(filename = function() {
  "myreport.html" },
  content = function(file) {
    require(GOstats)
    annota <- input$annotation
    bd <- character()
    p <- as.numeric(input$pvalGO)
    if(annota != "other")
    {
      bd <- annota
    }
    top <- result()
    genes <- selected()
    entrezUniverse = unique(getEG(as.character(rownames(top)), bd))
```

```
      geneIds <- unique(getEG(as.character(rownames(genes)),bd))
      GOparams = new("GOHyperGParams",
                     geneIds=geneIds, universeGeneIds=entrezUniverse,
                     annotation=bd, ontology=input$ontol,
                     pvalueCutoff=p, conditional=FALSE,
                     testDirection="over")
      KEGGparams = new("KEGGHyperGParams",
                       geneIds=geneIds, universeGeneIds=entrezUniverse,
                       annotation=bd,
                       pvalueCutoff=p, testDirection="over")
      GOhyper = hyperGTest(GOparams)
      # Creamos un informe html con los resultados
      htmlReport(GOhyper, file = file, summary.args=list("htmlLinks"=TRUE))
   })
  # Gene Annotation
  output$go <- renderTable({
    res <- selected()
    a <- as.vector(res$Gene.symbol)
    annota <- input$annotation
    if(annota != "other") {
      if(annota == "mogene10sttranscriptcluster.db")
      {
        table1     <-     select(mogene10sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "hgu133a.db")
      {
        table1 <- select(hgu133a.db, keys=a, columns=c("SYMBOL","GENENAME",
"GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "hugene21sttranscriptcluster.db")
      {
        table1     <-     select(hugene21sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "hugene20sttranscriptcluster.db")
      {
```

```
        table1     <-     select(hugene20sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "clariomdhumantranscriptcluster.db")
      {
        table1     <-     select(clariomdhumantranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "clariomshumanhttranscriptcluster.db")
      {
        table1    <-    select(clariomshumanhttranscriptcluster.db,    keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "clariomshumantranscriptcluster.db")
      {
        table1     <-     select(clariomshumantranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "clariomsmousehttranscriptcluster.db")
      {
        table1    <-    select(clariomsmousehttranscriptcluster.db,    keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      else if(annota == "clariomsmousetranscriptcluster.db")
      {
        table1     <-     select(clariomsmousetranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
      }
      b <- table1$GO
    } else {
      annotacion <- input$ann
      data1  =  read.table(annotacion$datapath,  sep="\t",  header  =  FALSE,
fill=TRUE, stringsAsFactors = FALSE)
      data1 = data1[ ,c(1,21)]
```

```
      names(data1) = c("ID", "GO")
      res <- merge(res, data1, by = "ID")
      res <- na.omit(res)
      row.names(res) <- res$ID
      a <- res$ID
      c <- res$Gene.symbol
      d <- res$GO
      d <- substr(d, start = 1, stop = 10)
      table1 <- data.frame(a,c,d)
      names(table1) <- c("ID", "Gene.symbol", "GO")
      b <- table1$GO
      b <- sapply(b, as.character)
      b <- substr(b, start = 1, stop = 10)
    }
    table2 <- select(GO.db, keys=b, columns=c("TERM", "DEFINITION"),
                     keytype="GOID")
    names(table2) <- c("GO", "TERM", "DEFINITION")
    table3 <- merge(table1,table2, by = "GO")
    table3 <- na.omit(table3)
    table3 <- unique(table3)
    head(table3, 10)
  })
  output$dwgo <- downloadHandler(filename = function() {
    "TableGenes.csv" },
    content = function(file) {
      res <- selected()
      a <- as.vector(res$Gene.symbol)
      annota <- input$annotation
      if(annota != "other") {
        if(annota == "mogene10sttranscriptcluster.db")
        {
          table1     <-     select(mogene10sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                            keytype="SYMBOL")
        }
        else if(annota == "hgu133a.db")
        {
          table1 <- select(hgu133a.db, keys=a, columns=c("SYMBOL","GENENAME",
"GO"),
                            keytype="SYMBOL")
```

```
        }
        else if(annota == "hugene21sttranscriptcluster.db")
        {
          table1     <-     select(hugene21sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "hugene20sttranscriptcluster.db")
        {
          table1     <-     select(hugene20sttranscriptcluster.db,     keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "clariomdhumantranscriptcluster.db")
        {
          table1    <-    select(clariomdhumantranscriptcluster.db,    keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "clariomshumanhttranscriptcluster.db")
        {
          table1   <-   select(clariomshumanhttranscriptcluster.db,   keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "clariomshumantranscriptcluster.db")
        {
          table1    <-    select(clariomshumantranscriptcluster.db,    keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "clariomsmousehttranscriptcluster.db")
        {
          table1   <-   select(clariomsmousehttranscriptcluster.db,   keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                           keytype="SYMBOL")
        }
        else if(annota == "clariomsmousetranscriptcluster.db")
        {
```

```
          table1    <-    select(clariomsmousetranscriptcluster.db,    keys=a,
columns=c("SYMBOL","GENENAME", "GO"),
                          keytype="SYMBOL")
        }
        b <- table1$GO
      } else {
        annotacion <- input$ann
        data1  =  read.table(annotacion$datapath,  sep="\t",  header  =  FALSE,
fill=TRUE, stringsAsFactors = FALSE)
        data1 = data1[ ,c(1,21)]
        names(data1) = c("ID", "GO")
        res <- merge(res, data1, by = "ID")
        res <- na.omit(res)
        row.names(res) <- res$ID
        a <- res$ID
        c <- res$Gene.symbol
        d <- res$GO
        d <- substr(d, start = 1, stop = 10)
        table1 <- data.frame(a,c,d)
        names(table1) <- c("ID", "Gene.symbol", "GO")
        b <- table1$GO
        b <- sapply(b, as.character)
        b <- substr(b, start = 1, stop = 10)
      }
      table2 <- select(GO.db, keys=b, columns=c("TERM", "DEFINITION"),
                       keytype="GOID")
      names(table2) <- c("GO", "TERM", "DEFINITION")
      table3 <- merge(table1,table2, by = "GO")
      table3 <- na.omit(table3)
      table3 <- unique(table3)
      write.csv(table3, file)
    })
  output$prueba <- renderText({
  })}
# Run the application
shinyApp(ui = ui, server = server)
```

Anexo IV. Resultados batería de pruebas

- Lectura de los archivos '.CEL' y del archivo de targets.

  La lectura y carga tanto de los archivos .cel como del archivo de targets es satisfactoria para todos los datasets.

Dataset 1. Collecting data.

Files .CEL

Files.CEL
GSM671653.CEL
GSM671654.CEL
GSM671655.CEL
GSM671656.CEL
GSM671657.CEL
GSM671658.CEL
GSM671659.CEL
GSM671660.CEL

Target File

FileName	Grupo	Shortname	Color
GSM671653.CEL	Induced	Induced1	Red
GSM671654.CEL	Induced	Induced2	Red
GSM671655.CEL	Induced	Induced3	Red
GSM671656.CEL	Induced	Induced4	Red
GSM671657.CEL	Not_induced	Not_induced1	Yellow
GSM671658.CEL	Not_induced	Not_induced2	Yellow
GSM671659.CEL	Not_induced	Not_induced3	Yellow
GSM671660.CEL	Not_induced	Not_induced4	Yellow

Dataset 2. Collecting data.

Files .CEL

Files.CEL
GSM1536990_DMSO_Rep1.CEL
GSM1536991_DMSO_Rep2.CEL
GSM1536992_DMSO_Rep3.CEL
GSM1536993_SRPIN803_Rep1.CEL
GSM1536994_SRPIN803_Rep2.CEL
GSM1536995_SRPIN803_Rep3.CEL

Target File

FileName	Grupo	Shortname	Color
GSM1536990_DMSO_Rep1.CEL	DMSO	1	Red
GSM1536991_DMSO_Rep2.CEL	DMSO	2	Red
GSM1536992_DMSO_Rep3.CEL	DMSO	3	Red
GSM1536993_SRPIN803_Rep1.CEL	SRPIN803	S1	Green
GSM1536994_SRPIN803_Rep2.CEL	SRPIN803	S2	Green
GSM1536995_SRPIN803_Rep3.CEL	SRPIN803	S3	Green

Dataset 3. Collecting data.

Files .CEL

Files.CEL
GSM1049708_Scid-GRK-1.CEL
GSM1049709_Scid-GRK-2.CEL
GSM1049710_Scid-K220R-1.CEL
GSM1049711_Scid-K220R-2.CEL
GSM1049712_HEK-GRK-1.CEL
GSM1049713_HEK-GRK-2.CEL
GSM1049714_HEK-K220R-1.CEL
GSM1049715_HEK-K220R-2.CEL
GSM1049716_Ex-Scid-GRK-1.CEL
GSM1049717_Ex-Scid-GRK-2.CEL
GSM1049718_Ex-Scid-K220R-1.CEL
GSM1049719_Ex-Scid-K220R-2.CEL

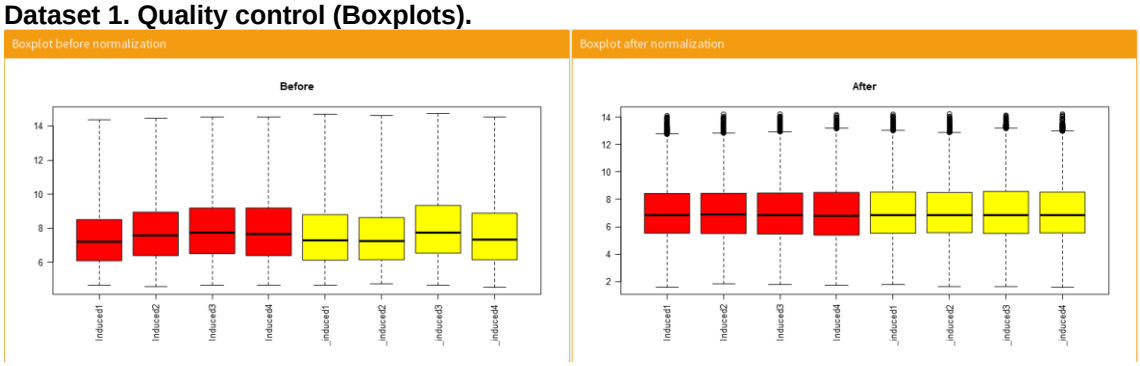
Target File

FileName	Grupo	Shortname	Color
GSM1049708_Scid-GRK-1.CEL	cultured	GRK21	Red
GSM1049709_Scid-GRK-2.CEL	cultured	GRK22	Red
GSM1049710_Scid-K220R-1.CEL	cultured	GRK2-K220R1	Red
GSM1049711_Scid-K220R-2.CEL	cultured	GRK2-K220R2	Red
GSM1049712_HEK-GRK-1.CEL.CEL	mouse_expanded	GRK23	Green
GSM1049713_HEK-GRK-2.CEL	mouse_expanded	GRK24	Green
GSM1049714_HEK-K220R-1.CEL	mouse_expanded	GRK2-K220R3	Green
GSM1049715_HEK-K220R-2.CEL	mouse_expanded	GRK2-K220R4	Green
GSM1049716_Ex-Scid-GRK-1.CEL	re_cultured	GRK25	Yellow
GSM1049717_Ex-Scid-GRK-2.CEL	re_cultured	GRK26	Yellow
GSM1049718_Ex-Scid-K220R-1.CEL	re_cultured	GRK2-K220R5	Yellow
GSM1049719_Ex-Scid-K220R-2.CEL	re_cultured	GRK2-K220R6	Yellow

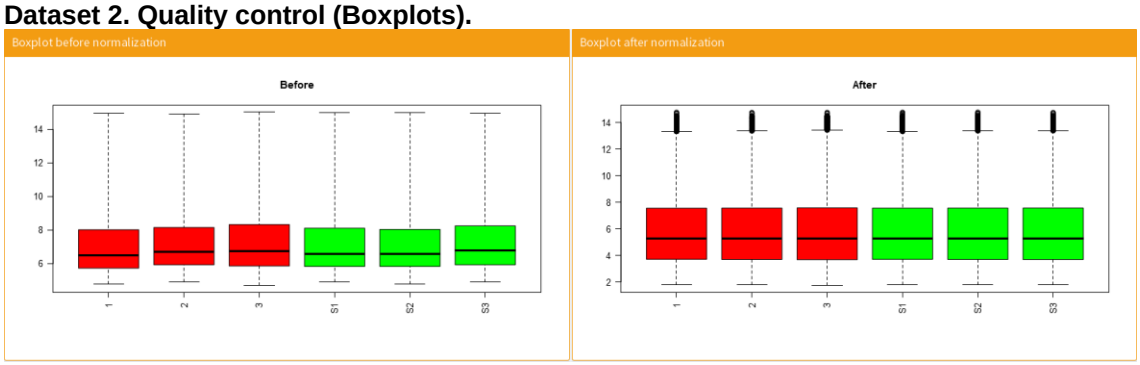
- Visualización y comprobación de la normalización de los datos a través de los boxplots.

  El control de calidad a través de los boxplots es satisfactorio para todos los datasets.

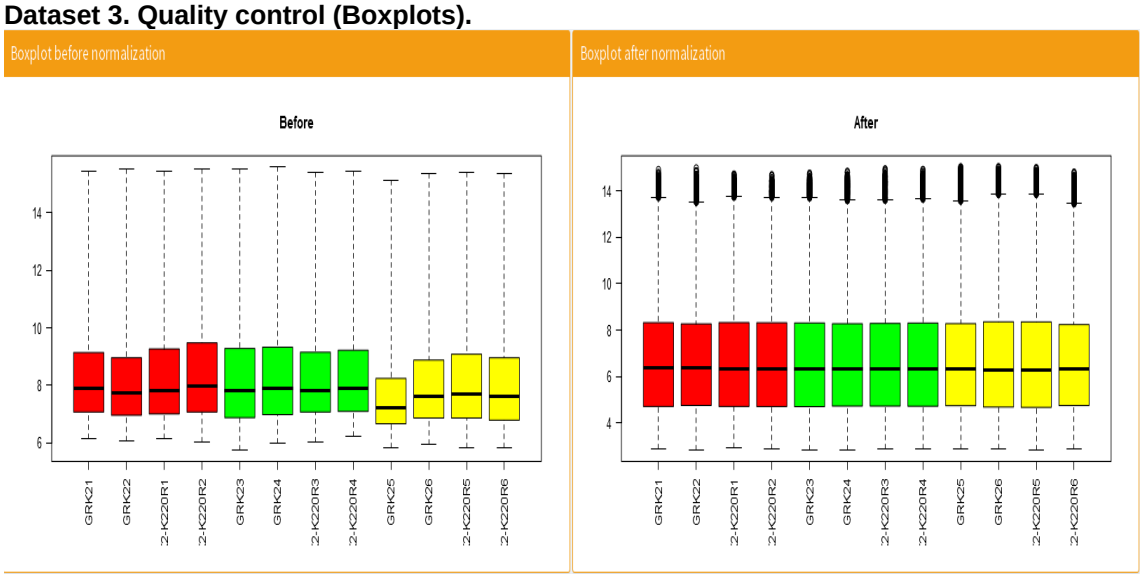
Dataset 1. Quality control (Boxplots).



Dataset 2. Quality control (Boxplots).



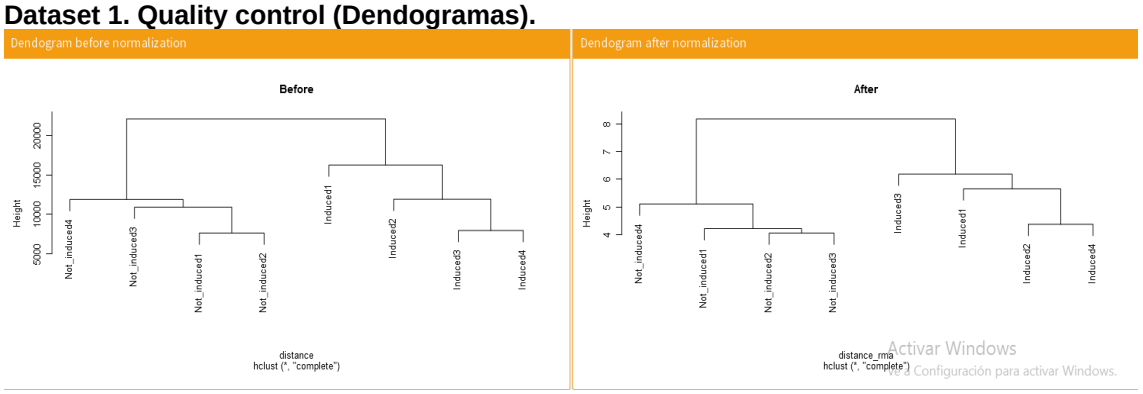
Dataset 3. Quality control (Boxplots).



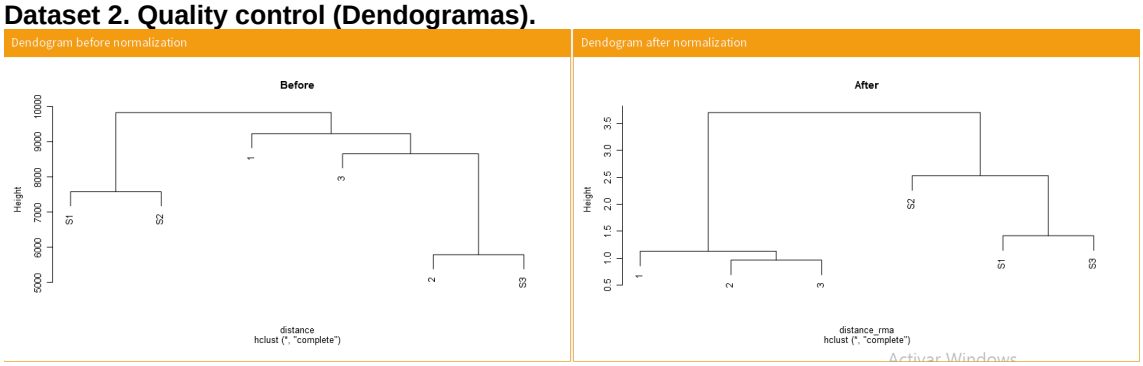
- Visualización y comprobación de una correcta agrupación de las muestras a través de los clústers jerárquicos.

  El control de calidad a través de los dendogramas es satisfactorio para todos los datasets.

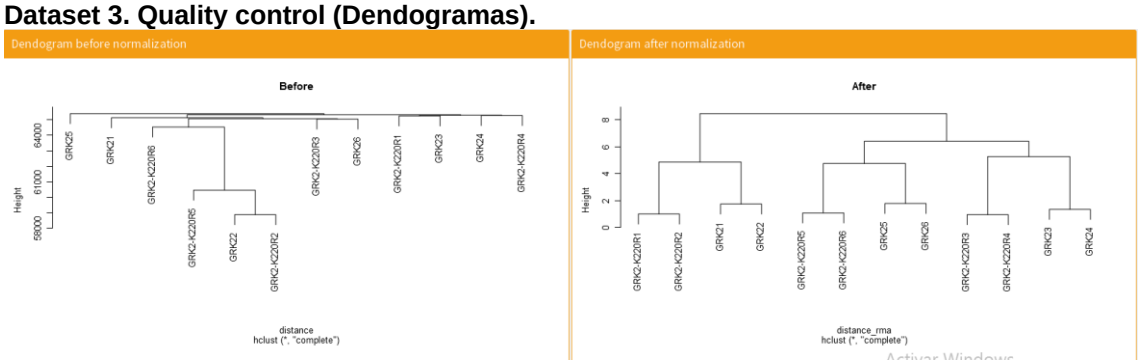
Dataset 1. Quality control (Dendogramas).



Dataset 2. Quality control (Dendogramas).



Dataset 3. Quality control (Dendogramas).



- Comprobación de los resultados en la Top Table, en la Selected Table, en la Up-regulated Table y en la Down-regulated Table.

  Los resultados de las tablas mencionadas proporcionan resultados acordes a los parámetros seleccionados en todos los datasets.

Dataset 1. Results (LFC=4, p-val=0.001).

Top Table

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
Obp2a	5.707E+00	7.364E+00	4.742E+01	1.068E-11	3.232E-07	1.619E+01	10470175
Lmx1a	5.780E+00	9.600E+00	4.456E+01	1.818E-11	3.232E-07	1.587E+01	10351443
Amph	5.654E+00	8.317E+00	4.076E+01	3.901E-11	3.468E-07	1.538E+01	10403796
Mrc1	-5.371E+00	7.748E+00	-3.893E+01	5.780E-11	3.617E-07	1.512E+01	10469358
Slc10a4	5.531E+00	8.432E+00	3.868E+01	6.104E-11	3.617E-07	1.509E+01	10522388
Fcrls	-4.973E+00	7.917E+00	-3.737E+01	8.203E-11	4.167E-07	1.487E+01	10499189
Rtn1	5.281E+00	9.127E+00	3.653E+01	9.950E-11	4.366E-07	1.473E+01	10400626
Nr4a2	4.714E+00	1.033E+01	3.608E+01	1.105E-10	4.366E-07	1.466E+01	10482772
Mapk10	5.752E+00	8.785E+00	3.510E+01	1.401E-10	4.659E-07	1.448E+01	10531869
Il1rl1	-4.246E+00	8.081E+00	-3.498E+01	1.441E-10	4.659E-07	1.446E+01	10345791

Download Genes

Selected

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
Prkcq	4.077E+00	7.410E+00	1.056E+01	3.263E-06	1.567E-04	5.014E+00	10469255
Calb1	5.802E+00	7.420E+00	1.014E+01	4.522E-06	1.926E-04	4.668E+00	10503416
Hmcn1	-4.228E+00	8.204E+00	-9.890E+00	5.520E-06	2.199E-04	4.457E+00	10358664
Pomc	4.322E+00	8.685E+00	9.741E+00	6.228E-06	2.353E-04	4.328E+00	10394240
Gpm6a	4.026E+00	7.609E+00	8.609E+00	1.646E-05	4.640E-04	3.293E+00	10571815
Ogn	-4.425E+00	9.317E+00	-7.744E+00	3.720E-05	7.984E-04	2.420E+00	10405063

Download Genes

Up-regulated Genes

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
Prkcq	4.077E+00	7.410E+00	1.056E+01	3.263E-06	1.567E-04	5.014E+00	10469255
Calb1	5.802E+00	7.420E+00	1.014E+01	4.522E-06	1.926E-04	4.668E+00	10503416
Pomc	4.322E+00	8.685E+00	9.741E+00	6.228E-06	2.353E-04	4.328E+00	10394240
Gpm6a	4.026E+00	7.609E+00	8.609E+00	1.646E-05	4.640E-04	3.293E+00	10571815

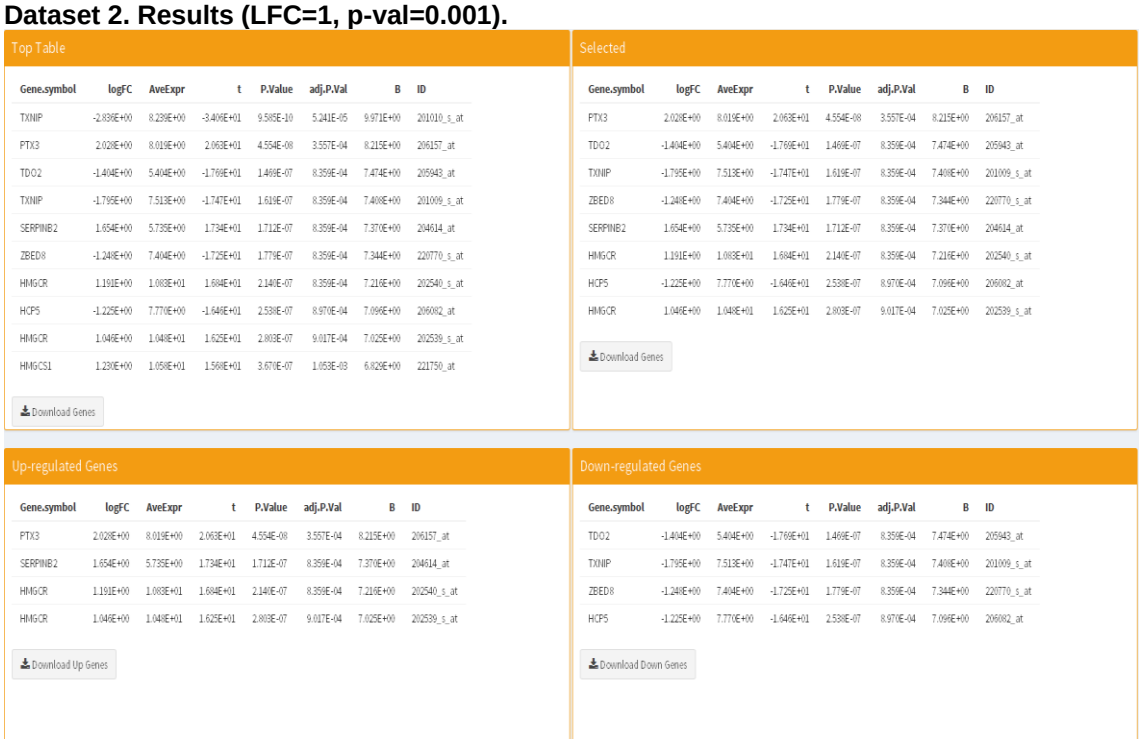
Download Up Genes

Down-regulated Genes

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
Hmcn1	-4.228E+00	8.204E+00	-9.890E+00	5.520E-06	2.199E-04	4.457E+00	10358664
Ogn	-4.425E+00	9.317E+00	-7.744E+00	3.720E-05	7.984E-04	2.420E+00	10405063

Download Down Genes

Dataset 2. Results (LFC=1, p-val=0.001).



Top Table

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
TXNIP	-2.836E+00	8.239E+00	-3.406E+01	9.585E-10	5.241E-05	9.971E+00	201010_s_at
PTX3	2.028E+00	8.019E+00	2.063E+01	4.554E-08	3.557E-04	8.215E+00	206157_at
TDO2	-1.404E+00	5.404E+00	-1.769E+01	1.469E-07	8.359E-04	7.474E+00	205943_at
TXNIP	-1.795E+00	7.513E+00	-1.747E+01	1.619E-07	8.359E-04	7.408E+00	201009_s_at
SERPINB2	1.654E+00	5.735E+00	1.734E+01	1.712E-07	8.359E-04	7.370E+00	204614_at
ZBED8	-1.248E+00	7.404E+00	-1.725E+01	1.779E-07	8.359E-04	7.344E+00	220770_s_at
HMGCR	1.191E+00	1.083E+01	1.684E+01	2.140E-07	8.359E-04	7.216E+00	202540_s_at
HCP5	-1.225E+00	7.770E+00	-1.646E+01	2.538E-07	8.970E-04	7.096E+00	206082_at
HMGCR	1.046E+00	1.048E+01	1.625E+01	2.803E-07	9.017E-04	7.025E+00	202539_s_at
HMGCS1	1.230E+00	1.058E+01	1.568E+01	3.670E-07	1.053E-03	6.829E+00	221750_at

Download Genes

Selected

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
PTX3	2.028E+00	8.019E+00	2.063E+01	4.554E-08	3.557E-04	8.215E+00	206157_at
TDO2	-1.404E+00	5.404E+00	-1.769E+01	1.469E-07	8.359E-04	7.474E+00	205943_at
TXNIP	-1.795E+00	7.513E+00	-1.747E+01	1.619E-07	8.359E-04	7.408E+00	201009_s_at
ZBED8	-1.248E+00	7.404E+00	-1.725E+01	1.779E-07	8.359E-04	7.344E+00	220770_s_at
SERPINB2	1.654E+00	5.735E+00	1.734E+01	1.712E-07	8.359E-04	7.370E+00	204614_at
HMGCR	1.191E+00	1.083E+01	1.684E+01	2.140E-07	8.359E-04	7.216E+00	202540_s_at
HCP5	-1.225E+00	7.770E+00	-1.646E+01	2.538E-07	8.970E-04	7.096E+00	206082_at
HMGCR	1.046E+00	1.048E+01	1.625E+01	2.803E-07	9.017E-04	7.025E+00	202539_s_at

Download Genes

Up-regulated Genes

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
PTX3	2.028E+00	8.019E+00	2.063E+01	4.554E-08	3.557E-04	8.215E+00	206157_at
SERPINB2	1.654E+00	5.735E+00	1.734E+01	1.712E-07	8.359E-04	7.370E+00	204614_at
HMGCR	1.191E+00	1.083E+01	1.684E+01	2.140E-07	8.359E-04	7.216E+00	202540_s_at
HMGCR	1.046E+00	1.048E+01	1.625E+01	2.803E-07	9.017E-04	7.025E+00	202539_s_at

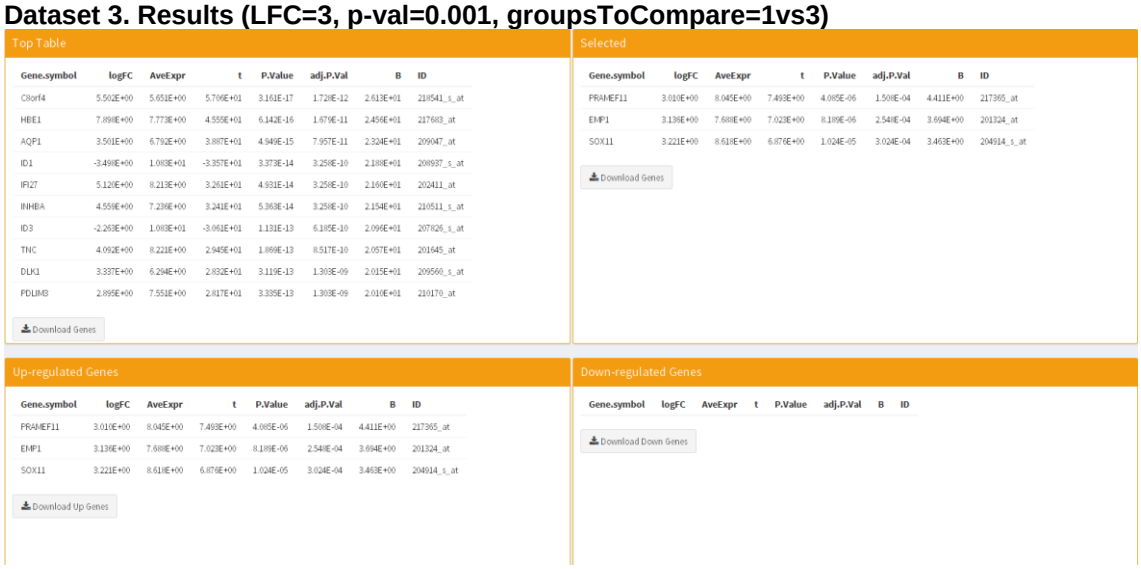
Download Up Genes

Down-regulated Genes

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
TDO2	-1.404E+00	5.404E+00	-1.769E+01	1.469E-07	8.359E-04	7.474E+00	205943_at
TXNIP	-1.795E+00	7.513E+00	-1.747E+01	1.619E-07	8.359E-04	7.408E+00	201009_s_at
ZBED8	-1.248E+00	7.404E+00	-1.725E+01	1.779E-07	8.359E-04	7.344E+00	220770_s_at
HCP5	-1.225E+00	7.770E+00	-1.646E+01	2.538E-07	8.970E-04	7.096E+00	206082_at

Download Down Genes

Dataset 3. Results (LFC=3, p-val=0.001, groupsToCompare=1vs3)



Top Table

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
C8orf4	5.502E+00	5.651E+00	5.706E+01	3.161E-17	1.728E-12	2.613E+01	218541_s_at
HBE1	7.898E+00	7.773E+00	4.555E+01	6.142E-16	1.679E-11	2.456E+01	217683_at
AQP1	3.501E+00	6.792E+00	3.887E+01	4.949E-15	7.957E-11	2.324E+01	209047_at
ID1	-3.498E+00	1.083E+01	-3.357E+01	3.373E-14	3.258E-10	2.188E+01	208937_s_at
IFI27	5.120E+00	8.213E+00	3.261E+01	4.931E-14	3.258E-10	2.160E+01	202411_at
INHBA	4.559E+00	7.236E+00	3.241E+01	5.363E-14	3.258E-10	2.154E+01	210511_s_at
ID3	-2.263E+00	1.083E+01	-3.061E+01	1.131E-13	6.185E-10	2.096E+01	207826_s_at
TNC	4.092E+00	8.221E+00	2.945E+01	1.869E-13	8.517E-10	2.057E+01	201645_at
DLK1	3.337E+00	6.294E+00	2.832E+01	3.119E-13	1.303E-09	2.015E+01	209560_s_at
PDLIM3	2.895E+00	7.551E+00	2.817E+01	3.335E-13	1.303E-09	2.010E+01	210170_at

Download Genes

Selected

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
PRAMEF11	3.010E+00	8.045E+00	7.493E+00	4.085E-06	1.508E-04	4.411E+00	217365_at
EMP1	3.136E+00	7.688E+00	7.023E+00	8.189E-06	2.548E-04	3.694E+00	201324_at
SOX11	3.221E+00	8.618E+00	6.876E+00	1.024E-05	3.024E-04	3.463E+00	204914_s_at

Download Genes

Up-regulated Genes

Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID
PRAMEF11	3.010E+00	8.045E+00	7.493E+00	4.085E-06	1.508E-04	4.411E+00	217365_at
EMP1	3.136E+00	7.688E+00	7.023E+00	8.189E-06	2.548E-04	3.694E+00	201324_at
SOX11	3.221E+00	8.618E+00	6.876E+00	1.024E-05	3.024E-04	3.463E+00	204914_s_at

Download Up Genes

Down-regulated Genes

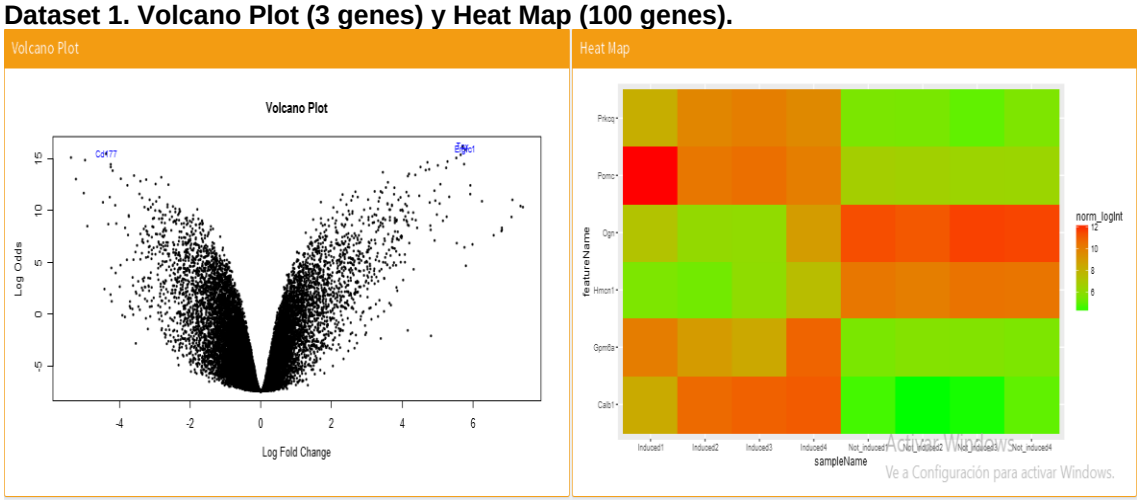
Gene.symbol	logFC	AveExpr	t	P.Value	adj.P.Val	B	ID

Download Down Genes

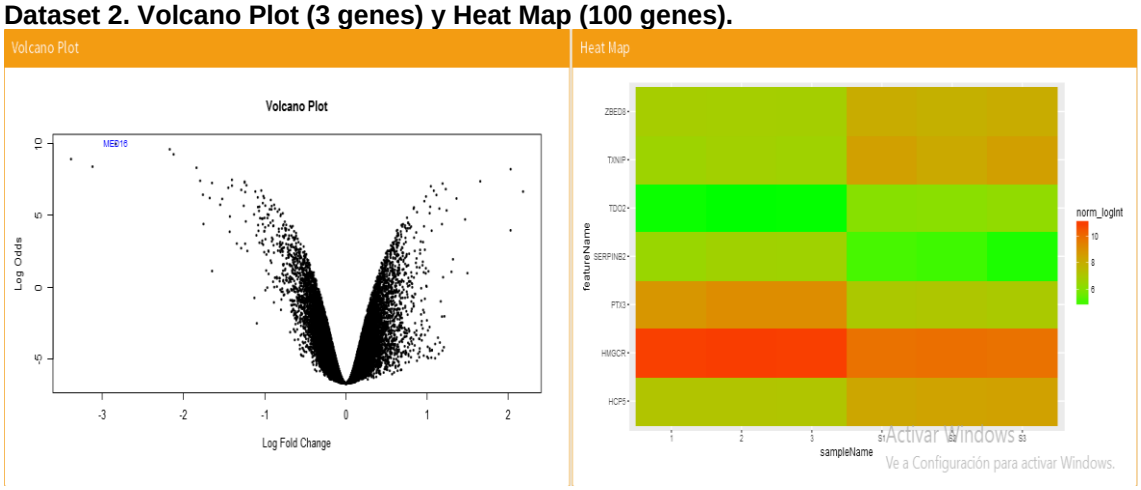
- Visualización del Volcano plot y del Heatmap.

  La visualización del Volcano plot y del Heatmap es satisfactoria para todos los datasets.

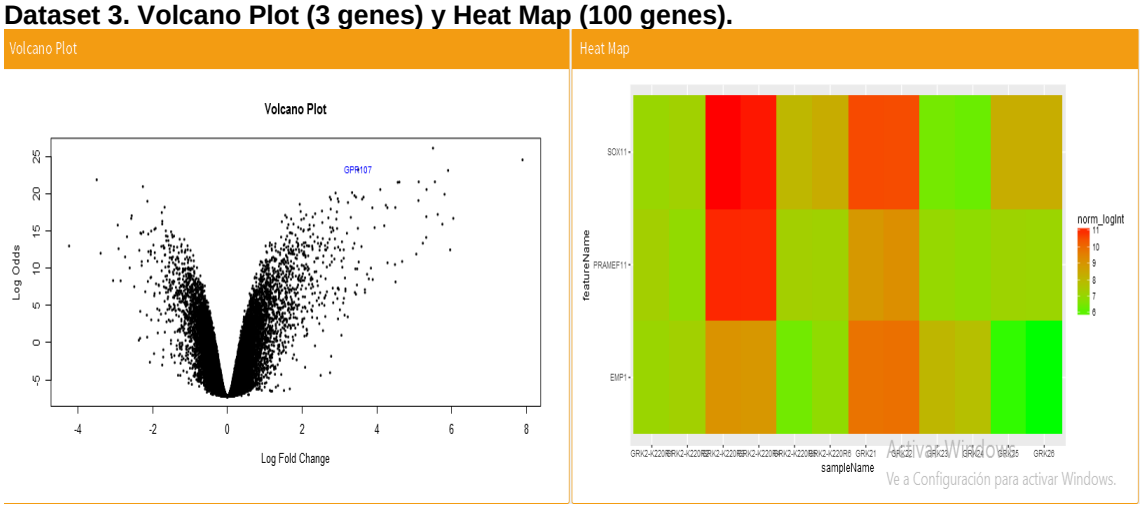
Dataset 1. Volcano Plot (3 genes) y Heat Map (100 genes).



Dataset 2. Volcano Plot (3 genes) y Heat Map (100 genes).



Dataset 3. Volcano Plot (3 genes) y Heat Map (100 genes).



- Comprobación de si existe una variación de los resultados al cambiar los grupos a comparar.

  De los tres datasets analizados, esta comprobación solo se puede realizar en el nº 3 ya que en los restantes solo hay dos grupos a

comparar y, por tanto, como cabría esperar, al intentar comparar grupos que no existe la aplicación muestra un error.

En el dataset 3 se comprueba que existe una variación en el resultado del análisis en función de qué grupos se comparen.

- Comprobación de si existe una variación de los resultados al cambiar el LFC.

  Se comprueba en los tres datasets que al provocar un cambio en el valor de LFC, se produce un cambio en los resultados del análisis. A mayor valor de LFC, la búsqueda es más restrictiva y, por tanto, se obtiene un número de genes diferencialmente expresados menor.

- Comprobación de si existe una variación de los resultados al cambiar el valor del p-valor ajustado.

  Se comprueba en los tres datasets que al provocar un cambio en el valor del p-valor ajustado, se produce un cambio en los resultados del análisis. Cuanto menor sea el p-valor, la búsqueda es más restrictiva y, por tanto, se obtiene un número de genes diferencialmente expresados menor.

- Comprobación de si existe una variación en el Volcano Plot cambiar el número de genes que debe mostrar.

  En los tres datasets al variar el número de genes que debe mostrar el Volcano Plot se produce un cambio en el gráfico mostrando el número de genes indicado.

- Compruebo si existe una variación en el Heatmap al cambiar el número de genes que debe mostrar.

  En los tres datasets al variar el número de genes que debe mostrar el Heatmap se produce un cambio en el gráfico mostrando el número de genes indicado.

- Comprobación de que el cambio de anotación tenga un correcto funcionamiento y que al introducir el fichero '.annot' descargado del servidor NCBI funciona correctamente el análisis.

  Para los datasets estudiados el cambio de anotación entre el archivo '.annot' descargado del servior NCBI y el correspondiente de entre los predeterminados tiene éxito y realiza correctamente el análisis de la expresión diferencial. Sin embargo, como es de esperar, al intentar analizar los datos con un archivo de anotación que no se corresponde con el dataset, se produce un error y la aplicación no realiza el análisis.

- Comprobación de que se realice correctamente el análisis GO.

  El resultado del análisis GO para los tres datasets es satisfactorio, siempre que se utilice como archivo de anotación alguno de los predeterminados por la aplicación, ya que para realizar este análisis la función de R requiere que se suministre un archivo '.db'.

- Comprobación de que se realice correctamente la anotación de las funciones para los genes seleccionados.

  La anotación de las funciones se realiza correctamente en los tres datasets suministrados.

- Comprobación de que se realice correctamente la descarga de las siguientes tablas: Top Table, Selected Table, Up-regulated Table, Down-regulated Table, GO Analysis y Gene Annotation.

  Se realiza correctamente la descarga de todas las tablas propuestas en los tres datasets suministrados.