Aplicación web desarrollada en Shiny para el análisis de Microarrays

Javier Bertol Chorro
Máster en Bioinformática y Bioestadística
Análisis de datos ómicos

Ricardo Gonzalo Sanz (Tutor)
Jose Antonio Morán Moreno (Profesor responsable de la asignatura)
Fecha Entrega
5 de Junio de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Descripción del trabajo*
Nombre del autor:	*Javier Bertol Chorro*
Nombre del consultor/a:	*Ricardo Gonzalo Sanz*
Nombre del PRA:	*Jose Antonio Morán Moreno*
Fecha de entrega (mm/aaaa):	06/2018
Titulación::	*Máster en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Análisis de datos ómicos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Expresión diferencial de genes, shiny, microarrays*

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

Con las ciencias ómicas en auge, los experimentos realizados por los científicos tienen una gran cantidad de datos que procesar cuya curva de aprendizaje, es de gran dificultad y requiere de conocimientos bioinformáticos.

La finalidad del TFM, es dotar a dichos científicos de una herramienta que permita realizar un análisis de DGE en microarrays de expresión clariom S (Affymetrix) en *Homo sapiens*, con unos parámetros pre-establecidos y otros controlados por el usuario. Para ello, se define una pipeline en el lenguaje de programación R, para posteriormente implementarla en una aplicación web interactiva mediante el paquete de R, Shiny.

Para comprobar la robustez y la funcionalidad de la pipeline y la herramienta desarrollada, se utilizan tres datasets para testar que el análisis funcione correctamente.

La herramienta demuestra un funcionamiento correcto para los datasets testados, así como la correcta descarga de todos los gráficos y tablas realizados.

Como líneas futuras de desarrollo, se plantea dotar de más variedad a la herramienta en cuanto a tipos de arrays y especies, así como aumentar el número de parámetros de la pipeline que pueden ser controlados por el usuario. Finalmente buscar un servidor alternativo a http://shinyapps.io para desplegar la aplicación.

Abstract (in English, 250 words or less):

With the omic sciences on the rise, the experiments carried out by scientists have a large amount of data to process. Whose learning curve is difficult and requires knowledge in bioinformatics fields The purpose of this project is to provide these scientists with a tool to perform a DGE analysis on clariom S expression microarrays (Affymetrix) in *Homo sapiens*. The application is interactive and respond to the user´s inputs. For this purpose, a pipeline of microarrays analysis is defined in the R programming language. Later the pipeline is implemented in an interactive web application using the R Shiny package. To check the robustness and functionality of the pipeline and the developed tool, three datasets were used. The tool demonstrates a correct operation with the datasets tested. All the graphs and tables reported by the application can be downloaded. As future lines of development it is proposed to provide more variety in terms of types of arrays and species to be analyzed. And provide the user with more parameters to control for a greater depth of analysis

Índice

1. Introducción 1
   1.1. Contexto y justificación del Trabajo 1
   1.2. Objetivos del Trabajo 1
   1.3. Enfoque y método seguido 2
   1.4. Planificación del Trabajo 4
   1.5. Breve sumario de productos obtenidos 5
   1.6. Breve descripción de los otros capítulos de la memoria 5
2. Estado del Arte 6
3. Tecnología Microarrays 9
   3.1 Antecedentes históricos 9
   3.2 Microarrays 9
   3.3 Tipos de Microarrays 10
   3.4 Aplicaciones de los Microarrays 11
   3.5 Análisis de datos de Microarrays 12
4. Datasets Utilizados 14
   4.1 Dataset 1 14
   4.2 Dataset 2 15
   4.3 Dataset 3 15
5. Desarrollo de la pipeline 17
   5.1 Software utilizado 17
   5.2 Pipeline 18
6. Desarrollo de la aplicación en Shiny 25
7. Resultados 31
8. Conclusiones 33
9. Glosario 35
10. Bibliografía 36
11. Anexos 40

Lista de figuras

Figura 1.Página principal de la aplicación DGER..... 2
Figura 2.Workflow estandarizado de una pipeline de análisis de Microarrays….3
Figura 3.Visualización de la aplicación desarrollada. 3
Figura 4.Chips soportados por las principales herramientas para el análisis de datos de Microarrays..... 6
Figura 5. Características de las herramientas offline 8
Figura 6. Etapas de un experimento de Microarrays..... 10
Figura 7. Tipos de arrays…. 11
Figura 8.Potenciales aplicaciones de los Microarrays..... 12
Figura 9. Workflow habitual de análisis d datos de Microarray 13
Figura 10. Consola interactiva de R-studio..... 17
Figura 11 PCA de las muestras del dataset 1 20
Figura 12.Boxplot de intensidades 21
Figura 13.Mapa de enriquecimiento usando Reactome. 24

Lista de tablas

Tabla 1. Aspecto del archivo targets correspondiente al dataset 1. 16
Tabla 2. Resultados obtenidos 31
Tabla 3. Resultados tests de cuello de botella 32

1. Introducción

1.1. Contexto y justificación del Trabajo

Con las ciencias ómicas en auge, los experimentos científicos a día de hoy recopilan una gran cantidad de datos para procesar y analizar. El software necesario para realizar dichos análisis, tiene una curva de aprendizaje de gran dificultad, ya que para llevarlos a cabo se requiere un nivel avanzado de conocimientos en materias de bioinformática[1][2].

Para resolver esta problemática, este TFM, va a desarrollar una herramienta implementando una pipeline de análisis de datos de transcriptómica (microarrays[3][4] de expresión); en una aplicación web online interactiva, con el software de programación R, mediante el uso del paquete Shiny. Facilitando con ello el análisis a todos aquellos científicos que no dispongan de los conocimientos bioinformáticos necesarios para programar e implementar sus propias pipelines.

1.2. Objetivos del Trabajo

Para desarrollar este trabajo de final de máster, se definen inicialmente dos objetivos principales bien diferenciados. A continuación se describen dichos objetivos.

Objetivo 1. Desarrollo de una pipeline para el análisis de datos de trancriptómica (microarrays de expresión).
Para desarrollar este objetivo, es necesario entender que pasos son importantes en una pipeline de análisis de microarrays; para ello hay que definir el análisis teórico de la pipeline, los archivos de entrada, los datasets que serán utilizados para probar la funcionalidad y los paquetes necesarios de R para realizar el análisis. Una vez definidos, se pasa a realizar la programación de la pipeline y probar la robustez de la misma con los datasets seleccionados.

Objetivo 2. Implementación en una aplicación web (Shiny) de la pipeline diseñada.

En primer lugar para realizar este objetivo, hay que entender el funcionamiento del paquete Shiny donde se realizará la aplicación. Posteriormente, se realizará el diseño teórico del layout de la aplicación, así como la definición de los parámetros de la pipeline que serán controlados por el usuario y la forma de presentar el informe de resultados del análisis. Finalmente, se realizará la implementación de la pipeline en Shiny, se testeará la funcionalidad de la misma con los datasets seleccionados en modo local, para después desplegar la aplicación a un servidor web (Figura1).

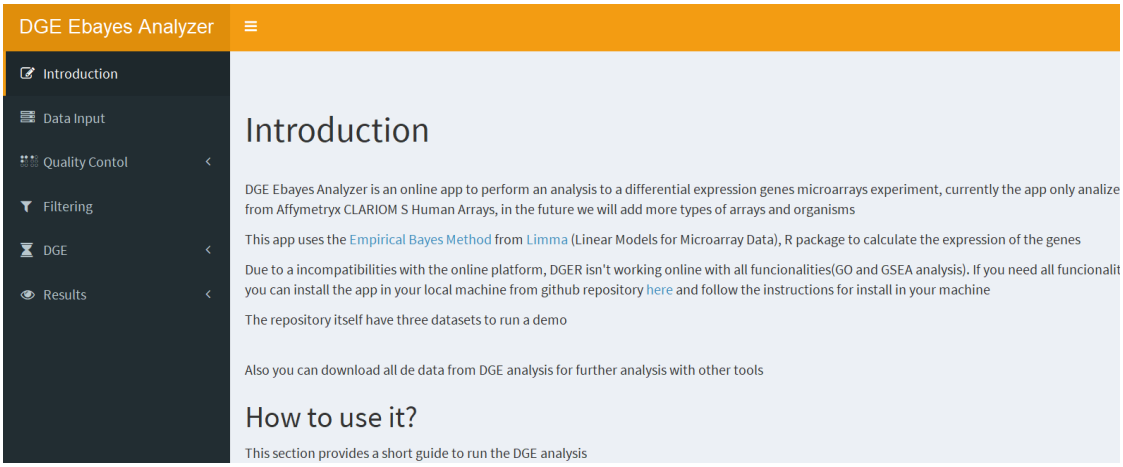


Figura 1.Página principal de la aplicación DGER.Producto final

1.3. Enfoque y método seguido

Existen multitud de aplicaciones offline y online implementando diversos workflows[5] de análisis de microarrays, todas ellas persiguiendo objetivos específicos. Una de las estrategias para desarrollar este TFM, sería ampliar cualquiera de las aplicaciones ya existentes añadiéndole nuevas funcionalidades. Finalmente, se decidió desarrollar un producto nuevo, intentado que la pipeline sea lo más estándar posible para poder realizar en ella, la mayor variedad de experimentos de microarrays diferentes.

Para desarrollar el proyecto, este de dividió en dos fases:

Fase 1 Desarrollo de la pipeline

En esta fase del proyecto se definió la pipeline (figura 2) de análisis de microarrays y se testó su robustez con los datasets seleccionados.

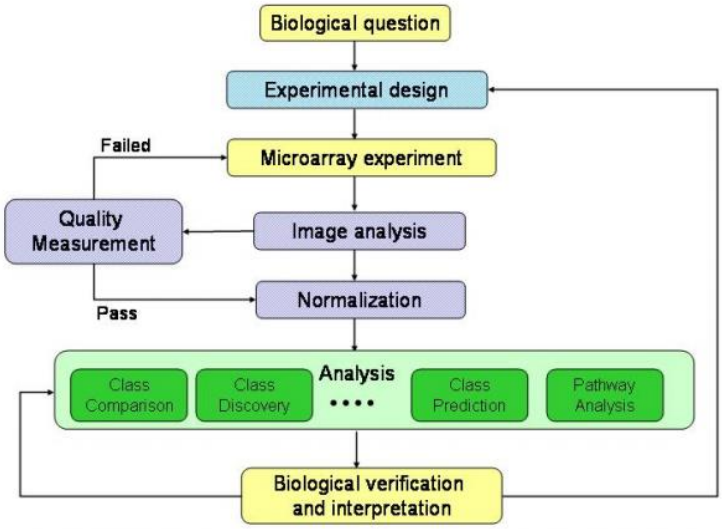


Figura 2.Workflow estandarizado de una pipeline de análisis de Microarrays[6]

Fase 2 Implementación de la pipeline a la aplicación web Shiny

Durante esta fase se implementó la pipeline diseñada en la fase 1 con el paquete de R Shiny. Para el diseño, se utiliza el framework shinydashboard (figura 3) de Shiny. Una vez testada la funcionalidad de la aplicación en el entorno local de desarrollo, se procede al despliegue de la aplicación a un servidor web. (http://www.shinyapps.io).

Figura 3.Visualización de la aplicación desarrollada.

1.4. Planificación del Trabajo

Se definieron las siguientes tareas a desarrollar en las dos fases del proyecto:

Tareas Fase 1

1. *Definir análisis teórico de la pipeline*. Se realizó una extensa revisión bibliográfica sobre análisis de Microarrays y pipelines de R para su análisis.
    - Definir etapas importantes de la pipeline
    - Definir informe de resultados del análisis
    - Archivos necesarios para utilizar la pipeline
    - Selección de los datasets para testear la aplicación
    - Definir paquetes de R necesarios para el análisis
2. *Programar la pipeline.* Se desarrolló el script en R de la pipeline basado en las características definidas anteriormente.
3. *Prueba de robustez.* Se comprobó la funcionalidad de la pipeline diseñada con los datasets seleccionados, así como posibles cuellos de botella computacionales.
4. *Diseño del Layout de la aplicación web.* Se diseñó de forma teórica la distribución que tendría la aplicación web en Shiny.

Tareas Fase 2

1. *Implementación de la pipeline a la aplicación web.* Se pasó a programar todos los componentes que serían necesarios para crear una aplicación en Shiny de la pipeline diseñada.
2. *Definir parámetros de la aplicación.* Se definieron y programaron que parámetros del análisis serian fijos y cuáles podrían ser controlados por los usuarios de la aplicación.
3. *Prueba de Robustez.* Se comprobó de forma local que la aplicación funcionase correctamente con los datasets seleccionados.
4. *Despliegue en el servidor.* Se desplegó la aplicación en el servidor gratuito que tiene R-Studio para Shiny. (https://www.shinyapps.io/)

1.5. Breve sumario de productos obtenidos

1. Script de la pipeline para el análisis de microarrays
2. Aplicación web desarrollada a partir de la pipeline
    - http://dger.ddns.net:8787/auth-sign-in
    - https://eirr.shinyapps.io/DGER/
3. Repositorio de github (https://github.com/eirr82/DGER-) con toda la información sobre la aplicación.

1.6. Breve descripción de los otros capítulos de la memoria

Los siguientes capítulos componen el cuerpo de la memoria:

- *Estado del arte.* Se hace una breve reseña sobre las diferentes herramientas que existen para analizar microarrays actualmente.
- *Tecnología Microarrays.* Capítulo donde se explica que son los microarrays de expresión y como se analizan.
- *Datasets*. Breve resumen de los datasets seleccionados para testear la aplicación.
- *Desarrollo pipeline*. Capítulo que explica el proceso del diseño de la pipeline de análisis de microarrays.
- *Desarrollo aplicación Shiny*. Se explica el proceso y los componentes que forman la aplicación web.
- *Resultados.* Breve resumen de los resultados y los productos obtenidos en el TFM.
- *Conclusiones.* Capítulo donde se exponen las conclusiones extraídas del desarrollo del TFM.
- *Glosario.* Recopilación de definiciones de términos utilizados en la memoria.
- *Bibliografía.* Capítulo que incluye todas las fuentes consultadas para la realización del TFM
- *Anexos.* Contiene documentación relativa a la aplicación (requisitos de software, resultados, código etc.)

2. Estado del Arte

Actualmente, el análisis de la expresión génica basada en la tecnología de microarrays, se ha convertido en una de las bases de la investigación biomédica[7]. Se necesita disponer de las herramientas apropiadas para manejar la cantidad y la diversidad de los datos que producen estas investigaciones. Existen multitud de herramientas gratuitas disponibles para realizar análisis en microarrays de expresión y que no requieren conocimientos en materias de programación.

La elección del tipo de herramienta de las disponibles a usar, va a venir claramente definida por el tipo de chip o microarray que hemos utilizado en nuestro experimento, ya que no todas las herramientas disponibles, tienen la capacidad de realizar análisis en todos los chips disponibles en el mercado de las plataformas comerciales (figura 4).

Tool	Affymetrix	Two-color
BASE	GeneChips, ExonChips	GenePix, Illumina
Chipster	GeneChips, ExonChips	Agilent, Illumina
EzArray	GeneChips	–
GeneX (Geoss)	GeneChips	–
MARS	GeneChips	Definition of parser for different formats possible
CARMAWeb	GeneChips	Agilent, ImaGene, BlueFuse, GenePix, QuantArray, SPOT, Arrayvision
Mayday	GeneChips	Only normalized data
TM4	Only preprocessed	GenePix

Figura 4.Chips soportados por las principales herramientas para el análisis de datos de Microarrays[5]

Podemos dividir estas herramientas en aquellas que procesan los datos crudos del experimento (intensidad de la señal) y las que procesan los datos ya normalizados y procesados. Encontramos otra división, entre aquellas que se pueden usar en modo offline (requieren instalación local) y aquellas herramientas que están alojadas en servidores webs a las que se accede de modo online y no es necesario la instalación de ningún componente adicional.

Entre las herramientas que requieren instalación destacamos (figura 5) las siguientes:

- BASE
- Chipster
- Affymetrix expression control
- EzArray
- GeneX(Geoss)
- MARS
- Mayday
- TM4
- Proyecto Bioconductor(R)

Dentro de las aplicaciones Online, destacan sin lugar a dudas aquellas basadas en R y bioconductor y las desarrolladas con Shiny

- CARMAweb (https://carmaweb.genome.tugraz.at/carma/)
- DEApp[8]
- START
- Galaxy[9] (https://usegalaxy.org/)

La elección de la herramienta apropiada para realizar el análisis de expresión, constituye actualmente un paso primordial a la hora de diseñar un experimento de microarrays.

Feature	BASE	Chipster	EzArray	Geoss	MARS/CARMAWeb	Mayday	TM4
Quality assessment							
Boxplot		√	√		√	√	
MA Plot	√	√	√		√	√	√
Nuse		√					
PLM		√	√				
Intensity distribut.		√					
Array correlation		√	√				
PCA/SVD		√				√	
Image intensity			√		√		
RNA deg.p.[a]		√	√				
Scatter plot	√	√	√	√		√	
Normalization							
RMA	√	√	√		√	√	
GCRMA		√	√		√		
PLIER	√	√					
MAS5		√	√		√		
VSN		√			√		
Lo(w)ess		√		√	√		
Print-tip lowess							
Scaling		√				√	
dChip			√				
Quantile					√		
Differential analysis							
t-test		√	√		√	√	√
SAM		√		√		√	√
ANOVA		√		√		√	√
Fold change		√	√	√		√	√
Rank produkt						√	√
Mann–Withney		√					√
Kruskal–Wallis		√					√
Wilcoxon		√	√		√		√
Multiple testing							
Benj.–Hochb. FDR[b]		√	√	√	√	√	√
Bonferroni		√	√		√	√	√
Clustering							
Hierarchichal clust.		√				√	√
k-means		√				√	√
PCA/SVD		√				√	√
SOM		√				√	√
QT		√				√	√
Visualization							
Heatmap		√				√	√
Histogram		√				√	
Volcano plot		√			√		√
Profile plot		√				√	√
Functional analysis							
KEGG		√					√
Gene Ontology		√			√		√
GSEA		√				√	√

Figura 5. Características de las herramientas offline disponibles para el análisis de Microarrays[5]

3. Tecnología Microarrays

3.1 Antecedentes históricos

La biología molecular dispone de multitud de técnicas para medir la expresión génica

- Northen blot (Niveles de RNA)
- Western blot (niveles de proteína)
- Real time qPCR (Niveles de DNA)

La molécula que mayor interés ha tenido siempre de entre todas, son los niveles de RNA transcrito, de este interés, hace unos años se acuñó el término de transcriptómica para referirse al estudio del RNA, cuyo principal objetivo es cuantificar los niveles de expresión génica bajo diferentes condiciones.

Este interés en la trancriptómica,[10] ha llevado a cambiar el paradigma de que el interés no está en lo que se puede medir, sino en la cantidad de mediciones simultáneas que se pueden realizar, lo que ha llevado al desarrollo de nuevas tecnologías y métodos para extraer y procesar la gran cantidad de datos que se generan con estas técnicas.

3.2 Microarrays

En plena era de la transcriptómica, los microarrays se erigieron como una herramienta indispensable para cuantificar los niveles de expresión génica de un determinado organismo sujeto a distintas condiciones.

Los microarrays de DNA[11] o chips, consisten en una gran cantidad de moléculas de DNA ordenadas sobre un sustrato sólido, formando de esta manera una matriz bidimensional

Cada microarray contiene multitud de spots donde están ubicados los fragmentos de DNA que actuarán como sondas (probes). Los fragmentos de DNA de las muestras (dianas) a analizar, se marcan por diversos métodos (enzimáticos, fluorescentes etc.) y se incuban sobre el panel de sondas permitiendo la hibridación entre sonda y diana de las moléculas

complementarias de las secuencias homólogas, quedando inmovilizadas sobre el soporte sólido (figura 6).

Posteriormente, se utilizan herramientas de detección(escáneres) sobre los chips hibridados , permitiendo la identificación y la cuantificación de las moléculas hibridadas para posteriormente, usar herramientas bioinformáticas para el análisis e interpretación de los resultados[12].

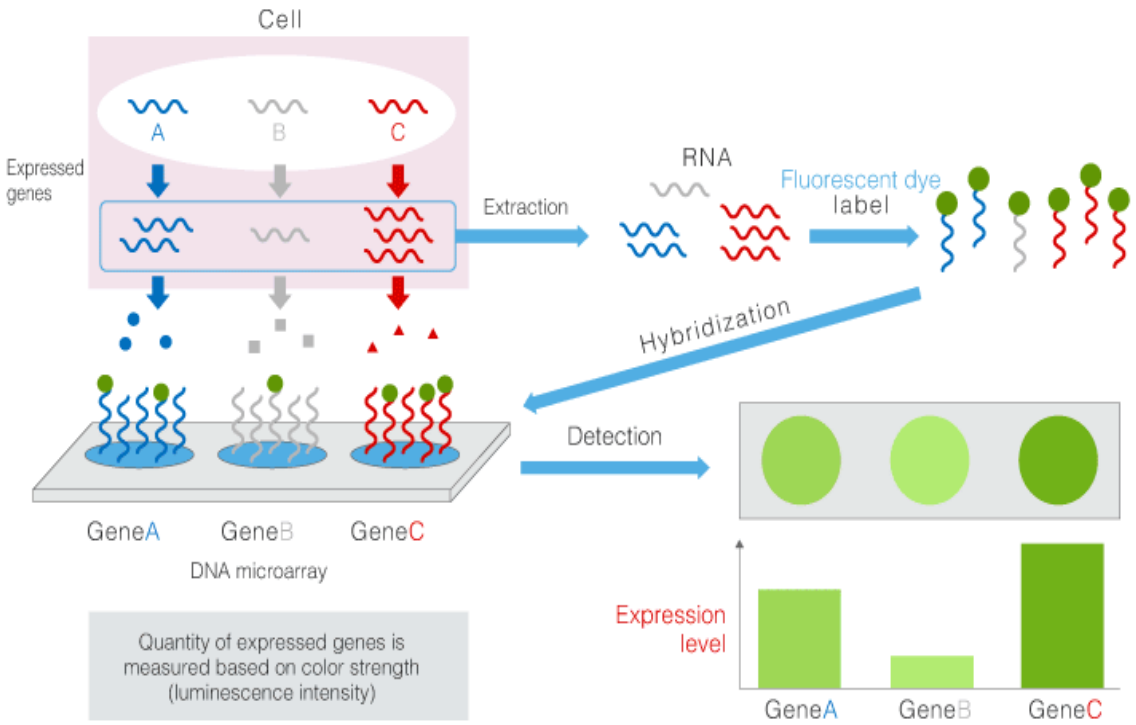


Figura 6. Etapas de un experimento de Microarrays. Desde la extracción del RNA de las muestras hasta la detección de la señal[13]

3.3 Tipos de Microarrays

Existen diferentes tipos de microarrays ya que se utilizan diferentes tecnologías para fabricarlos. Se pueden clasificar según el tipo de biomolécula utilizada, el tipo de marcaje de las dianas, la cantidad de muestras hibridadas etc.

Tipos de Microarrays más comunes[14]

- Microarrays de Tejidos
- Microarrays de Proteínas
- Microarrays de expresión(RNA)

    - Genotipado
    - SNPs
- Microarrays de DNA
    - cDNA
    - Oligonucleótidos

Una clasificación habitual de los microarrays es por el número de muestras que se hibridan simultáneamente:

- Microarrays de dos colores
- Microarrays de un color

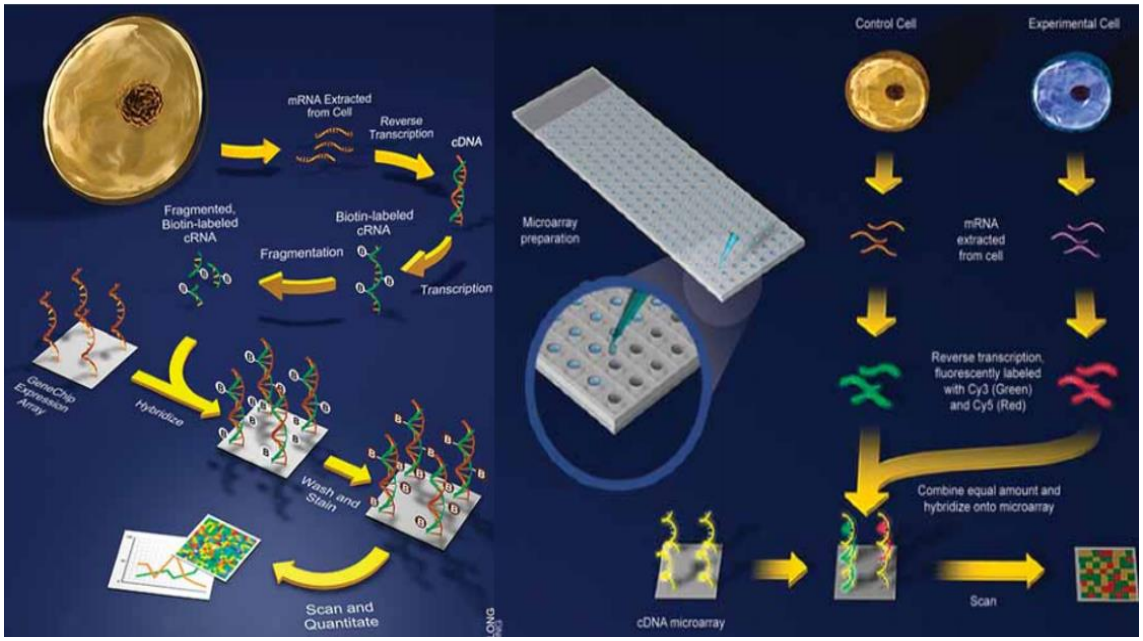


Figura 7. Arrays de oligonucleótidos(izquierda) y spotted arrays(derecha)[15]

3.4 Aplicaciones de los Microarrays

Los microarrays han supuesto una revolución en el campo de la transcriptómica y actualmente son utilizados en multitud de disciplinas(Figura 8) diferentes[16]:

- Diagnóstico Molecular
    - Cribado genético
    - Identificación de patógenos
    - Resistencia a fármacos de microorganismos infecciosos
    - Diagnóstico y pronóstico oncológico
    - Análisis de expresión génica

- Farmacogenómica
    - Identificación de genes
    - Respuesta a fármacos
    - Diagnóstico predictivo
- Desarrollo de fármacos
    - Cribado de fármacos
    - Diseño y estratificación de ensayos clínicos

	CURRENT ADOPTION BY PHARMA R&D	POTENTIAL APPLICATIONS IN BIOMARKER DISCOVERY	POTENTIAL APPLICATION IN CLINICAL SCREENING	NEED FOR HIGH DENSITY ARRAYS	OVERALL POTENTIAL IN PHARMACEUTICAL R&D	POTENTIAL FOR DIAGNOSTIC APPLICATIONS
SNP arrays	●	●	●	●	●	●
Array CGH	◣	◗	◗	◣	◣	◣
Resequencing	●	◗	◗	●	◗	◗
ChIP/Chip	○	○	○	●	◣	○
Splice variants	◣	◣	◣	●	◣	◣
MicroRNA	◣	◣	◣	◗	◗	○
RNAi arrays	◣	◣	○	◣	◣	○
Mitochondrial DNA arrays	○	○	○	○	○	○
Methylation	◗	◗	○	●	◗	○

Figura 8.Potenciales aplicaciones de los Microarrays.[17]

3.5 Análisis de datos de Microarrays

Las etapas(Figura 9) básicas que deben seguirse para llevar a cabo en un experimento de análisis[18],[19],[20] de microarrays de expresión, son las siguientes(se explicarán con más profundidad en el capítulo que hace referencia al diseño de la pipeline):

1. Identificar la cuestión biológica que se quiere resolver
2. Diseño experimental
3. Experimento con chips o microarrays
    - Extracción del material genómico que se quiere analizar(RNA)
    - Síntesis del cDNA complementario
    - Marcaje del cDNA mediante un marcador fluorescente

- Hibridación en el array de expresión

4. Análisis de datos
    - Lectura de la intensidad de la señal de los chips
    - Control de calidad de los datos
    - Normalización de los datos
    - Análisis de la expresión diferencial(DGE)
    - Análisis funcional (GO, GSEA)

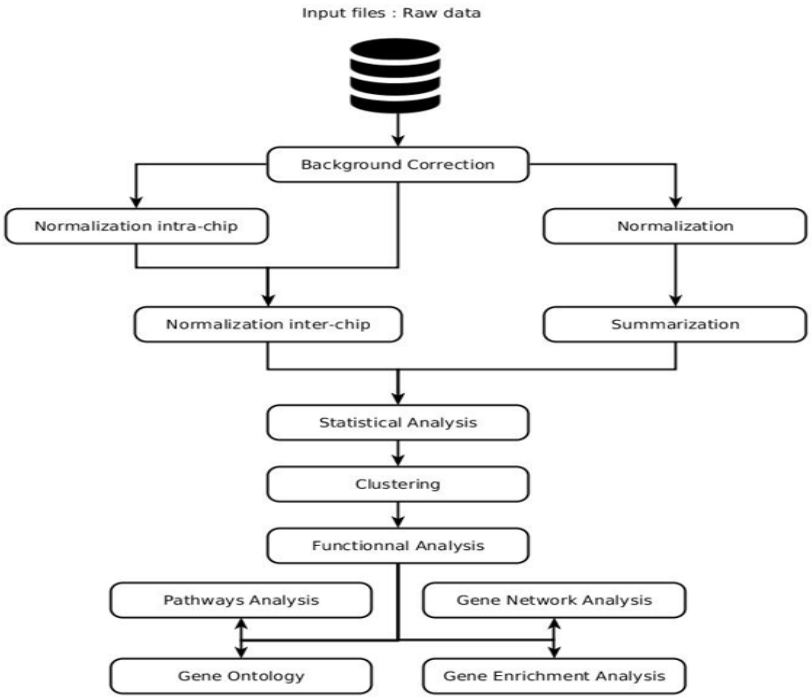


Figura 9. Workflow habitual de análisis de datos de Microarrays[9]

.

4. Datasets Utilizados

Para el desarrollo de la pipeline inicialmente se descargaron dos datasets del repositorio de datos publico Geo Expression Omnibus (https://www.ncbi.nlm.nih.gov/geo/) , debido a inconsistencias con los resultados del dataset número uno, se descargó un tercer dataset para tener una mayor fiabilidad en la robustez de la aplicación.

Todos los datasets descargados, corresponden a la misma plataforma ya que la aplicación desarrollada actualmente solo está diseñada para analizar los arrays de Affymetrix Clariom S en homo sapiens, concretamente:

GPL23159 [Clariom_S_human] Affymetrix Clariom Human array

4.1 Dataset 1

Expresion data from a human lung metastatic cell line treated with siPGC1alpha or siControl[21]

GSE99554 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE99554)

El objetivo del estudio es entender los cambios de expresión en células de cáncer metastásico pulmonar (línea celular 4175) con un knockdown del gen PGC1alpha respecto a las células que tiene este gen.

Las células se dividieron en dos grupos, un grupo fue tratado con el RNA de interferencia (siRNA) para el gen PGC-1a durante 48horas. El segundo grupo, fue tratado con un siRNA control durante 48h.

El dataset consta de un total de seis muestras divididas en dos grupos:

- Tres muestras para el grupo control(siControl)
- Tres muestras para el grupo knockdown(siPGC1alpha)

4.2 Dataset 2

Expression data from 5azaC- or Gallic acid-treated human lung cancer H1299 cell line[22]

GSE99993 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE99993)

El objetivo del estudio es comparar las diferencias de expresión génica de la línea celular H1299(cáncer pulmón), al ser tratadas con el fármaco 5azaC- o el ácido Gálico que tiene actividades anti cancerosas.

El dataset consta de un total de ocho muestras divididas de la siguiente forma:

- Dos muestras tratamiento fármaco (5azaC-)
- Dos muestras tratamiento ácido gálico (GA)
- Dos muestras control sin el fármaco (5azaC)
- Dos muestras control sin el ácido gálico (Ctrl-G)

4.3 Dataset 3

Expression data from human macrophages treated with miR-1246 mimic/inhibitor[23]

GSE107870 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE107870)

El objetivo de este experimento, fue determinar el perfil de expresión de los macrófagos M2-like cuando son tratados con un mímico de miR-1246 (promueve la tumorogénesis y la metástasis) y compararlos con un grupo control tratados con un inhibidor de miR-1246

Un total de seis muestras fueron divididas en los siguientes grupos

- Replica biológica 1
    - Una muestra con Macrófagos sin tratar (grupo control)
    - Una muestra con Macrófagos tratados con el mímico de miR-1246
    - Una muestra con Macrófagos tratados con el inhibidor de miR-1246

- Replica biológica 2

- Una muestra con Macrófagos sin tratar (grupo control)
- Una muestra con Macrófagos tratados con el mímico de miR-1246
- Una muestra con Macrófagos tratados con el inhibidor de miR-1246

Paralelamente para cada dataset, se construyó un archivo “targets” en formato .CSV (separado por ;) que proporciona la información del experimento (nombres de los archivos, grupos que lo componen etc.)

FileNames	Group	Name	PlotColor
MDAMB231_4175_sicontrol_1	control	control1	chocolate4
MDAMB231_4175_sicontrol_2	control	control2	chocolate4
MDAMB231_4175_sicontrol_3	control	control3	chocolate4
MDAMB231_4175_siPGC1alpha_1	SIPGC	SIPGC1	royalblue4
MDAMB231_4175_siPGC1alpha_2	SIPGC	SIPGC2	royalblue4
MDAMB231_4175_siPGC1alpha_3	SIPGC	SIPGC3	royalblue4

Tabla 1.Aspecto del archivo targets correspondiente al dataset 1.

5. Desarrollo de la pipeline

5.1 Software utilizado

Para desarrollar la pipeline, se ha utilizado el software de programación open source R en su versión 3.5.0 (Joy in playing), junto a la suite Bioconductor que está diseñada específicamente para el tratamiento de datos de experimentos genómicos.

The R Project for statistical computing

R es un lenguaje de programación de distribución libre para computación estadística y multitud de gráficos. Es uno de los lenguajes más utilizados en los campos de investigación biomédica y bioinformática. Actualmente cuenta con más de 2000 paquetes en su repositorio principal (CRAN). Funciona en multitud de plataformas basadas en UNIX, Windows y MacOS

R-Studio

Es un entorno de desarrollo integrado (IDE) para R. Incluye una consola, un editor de sintaxis, que admite la ejecución directa de código, así como herramientas para gestionar el espacio de trabajo (figura 10).

Bioconductor

Es una suite de herramientas para el análisis de datos genómicos[6], [24], [25]. Usa el lenguaje de programación R y es de código abierto. Actualmente cuenta con más de 1500 paquetes.

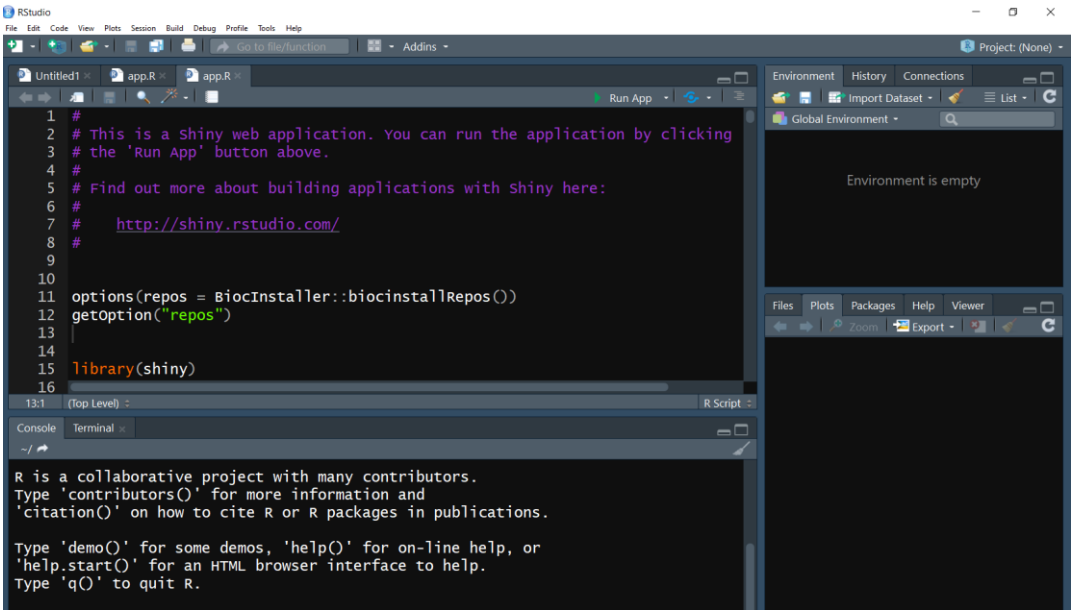


Figura 10. Consola interactiva de R-studio, entorno de trabajo donde se ha desarrollado la aplicación

Los paquetes utilizados para el desarrollo de la pipeline fueron los siguientes:

Repositorio CRAN (repositorio principal de R)

- Knitr[26]. Paquete que se utiliza para reportar el informe de resultados en formato markdown.
- DT[27]. Es una interfaz de R para la librería de JavaScript DataTables y cuya función es poder añadir filtros, paginación y otras características a las tablas.

Repositorio Bioconductor

- Oligo[28]. Herramientas para el pre procesado de arrays de oligonucleótidos y lectura de los datos crudos de los experimentos(CEL files)
- pd.clariom.s.human. Anotación de genes para los arrays de Affymetrix Clariom_S_Human
- clariomshumantranscriptcluster.db. Paquete de anotación para los arrays de Affymetrix Clariom_S_Human
- Limma[29]. Paquete para calcular la expresión diferencial de genes por el método de Ebayes
- Affycoretools. Herramientas para los arrays de Affymetrix
- Heatplus[30]. Herramienta para generar los heatmaps de la pipeline.
- topGO[31]. Herramienta para realizar el análisis GO (ontología genética)
- GofuncR[32]. Herramienta para realizar el análisis GO
- ReactomePA[33]. Herramienta para realizar análisis GSEA
- Homo.sapiens. Paquete de anotación para humanos para realizar el análisis GO.

5.2 Pipeline

La pipeline diseñada, cuenta con cinco etapas claramente diferenciadas, lectura de datos, control de calidad, normalización, selección de genes diferencialmente expresados y análisis funcional.

Lectura de Datos

Se cargan lo datos necesarios para realizar el análisis, los ficheros .CEL y el archivo targets, creado manualmente que contiene la información del experimento. Para leer los ficheros .CEL se usa la función `read.celfiles()` [28] del paquete oligo la cual crea un objeto con clase ExpressionSet. Para leer el archivo targets se utiliza la función `read.csv()`.

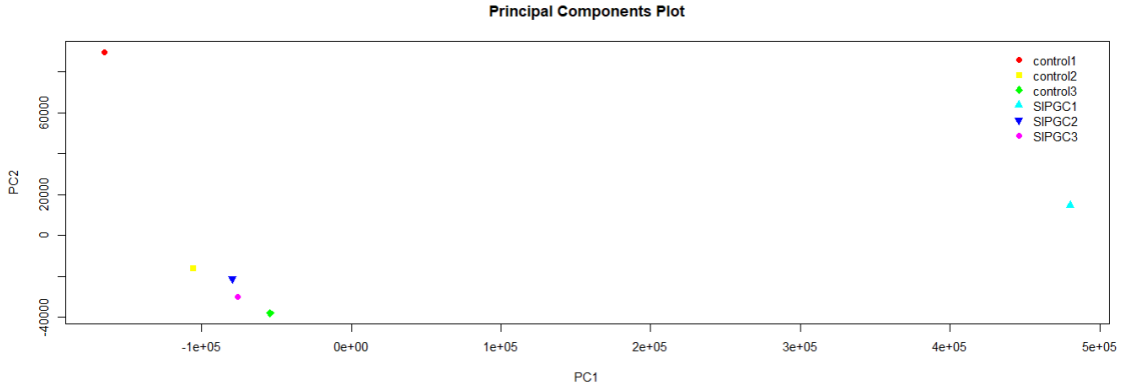
Control de Calidad

Ante de proceder al análisis de los datos, es importante realizar un control de calidad[4], [34] de los datos importados. De esta manera, podemos detectar si alguna muestra está dañada ya sea por un error en el proceso de fabricación del chip o una mala hibridación entre sonda y diana.

La pipeline realiza dos controles de calidad, unos sobre los datos crudos (raw) y otro una vez los datos han sido normalizados.

El control de calidad consta de los siguientes gráficos para realizarlo:

- Histograma. Representa la intensidad de la señal para cada muestra del experimento. Permite evaluar la calidad de las muestras.
- Boxplot. Resumen gráfico en forma de cajas de la intensidad de la señal.
- Dendograma. Representación gráfica en forma de árbol, donde se agrupan las muestras de forma jerárquica. Sirve para evaluar una correcta agrupación de las muestras o por el contrario, ver si existe algún efecto batch no deseado
- PCA. Reduce la dimensionalidad de las variables y muestra gráficamente como se agrupan las muestras del experimento (figura 11).



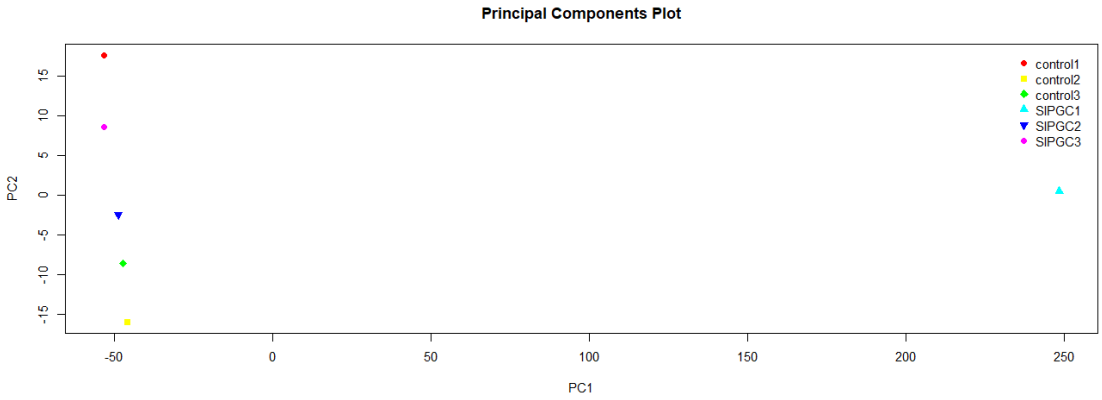


Figura 11 PCA de las muestras del dataset 1. Arriba antes de normalizar, abajo después Observamos una muestra muy diferente las demás

El control de calidad, nos sirve como primer punto de control para valorar de forma objetiva la calidad de los arrays o por el contrario ver si detectamos algún error en alguna muestra que podría llevarnos a un mal análisis de las mismas.

Normalización de las muestras

El último paso antes de empezar con la selección de genes diferencialmente expresados, es normalizar las muestras para eliminar fuentes de variación sitemáticas que no sean diferencias de expresión.(eficiencia en el marcaje del color, cantidad de RNA, efectos espaciales del chip, etc.).

Hay diversos métodos de normalización[35], [36], para la pipeline se decidió utilizar el método RMA (robust Multiarrays average) ya que actualmente es el más utilizado en este tipo de análisis. Para realizarlo se utiliza la función `rma()` del paquete oligo.

La normalización(figura 12) por RMA[37] consta de tres etapas bien diferenciadas:

- Corrección de fondo. Consiste en eliminar el ruido de fondo, es decir la hibridación no especifica entre biomoléculas que afecta a la sensibilidad y especificidad del chip.
- Normalización. Para comparar varios chips, es necesario ajustar la señal para que todas las muestras tenga la misma escala y puedan ser comparables. El método RMA normaliza por cuantiles de PM(perfect match) de las sondas

- Sumarización. Para cada gen, tenemos varias sondas que miden su intensidad.La sumarizacion corresponde al proceso de determinar la intensidad de los genes a partir de la intensidad de sus sondas. El método RMA ajusta este valor por Median Polish[36]

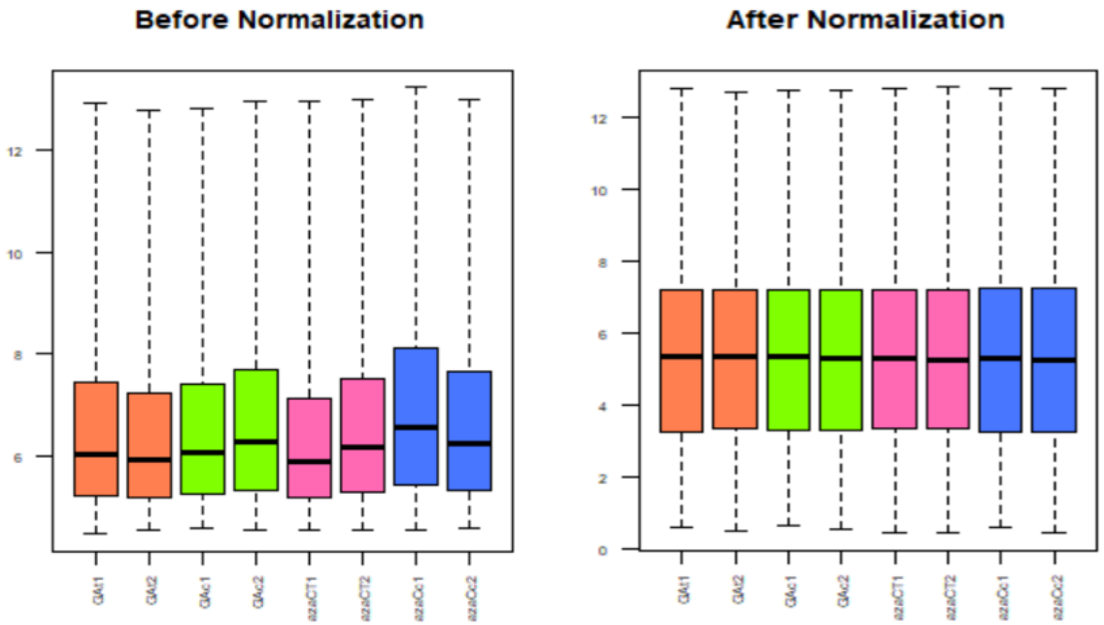


Figura 12.Boxplot de intensidades de las muestras del dataset 2. Antes de normalizar (izquierda) y después de normalizar (derecha)

Selección de genes diferencialmente expresados

Para seleccionar los genes que están diferencialmente expresados, hay diversos métodos existentes en la literatura[38]. En la pipeline diseñada se va a utilizar el paquete de Bioconductor Limma (linear models for microarrays analysis), la idea central de este tipo de análisis, es ajustar un modelo lineal para cada gen con sus valores de expresión.

El uso de los modelos lineales requiere que especifiquemos dos matrices para realizar el análisis:

- Matriz de diseño. Contiene las muestras que han sido hibridadas en los arrays. Función `model.matrix()`
- Matriz de Contrastes. Contiene la información referente a que grupos van a ser comparados en el análisis (contrastes). Función `makeContrasts()` del paquete limma.

Una vez especificadas las matrices, se realiza el test estadístico por el método ebayes[39] (empirical bayes), un test específicamente diseñado para el análisis de microarrays que utiliza un modelo bayesiano que se basa en reducir los valores de la varianza de las sondas hacia un valor común, y aumentar los grados de libertad para las varianzas individuales de cada gen.

Para ello usamos las siguientes funciones del paquete limma:

- `lmFit()` para crear el modelo lineal
- `contrasts.fit()` calcula los coeficientes del modelo lineal creado para cada set de contrastes especificados
- `eBayes()` aplica el método bayesiano al modelo linear

Una vez creado el modelo lineal, los resultados del mismo se presentan de la siguiente forma:

- Toptable. Una tabla que representa aquellos genes diferencialmente expresados, ordenados de mayor a menor. Función `topTable()` del paquete limma. La tabla se presenta con los genes anotados , creando una columna con el nombre de los genes con la función `getSYMBOL()` del paquete annotate
- Volcano Plot[40]. Gráfico que representa para cada gen, los cambios de expresión en escala logarítmica. Función `volcanoplot()` del paquete limma
- Tabla de genes seleccionados. Una tabla filtrada de la toptable a partir de unos parámetros definidos por el usuario.
- Heatmap. Grafico que representa los perfiles de expresión de los genes seleccionados. Función `regHeatmap()` del paquete Heatplus

Se han añadido los siguientes gráficos y tablas para experimentos con más de una comparación:

- Resumen decide tests[29]. Una tabla que presenta que genes presentan una expresión diferencial simultáneamente en más de una comparación. Función `decideTests()` del paquete limma.
- Diagrama de Venn. Una representación visual de la tabla generada por el decide tests. Función `vennDiagram()` del paquete limma
- Tabla indicando los genes que están sobre expresados e infra expresados en cada comparación realizada.

Análisis Funcional

Finalmente se utiliza la tabla de genes seleccionados para realizar un análisis funcional de dichos genes:

- Análisis GO. El proyecto Ontología Génica realiza un análisis de enriquecimiento sobre el set de genes proporcionado. Provee un vocabulario controlado que describe a los genes y a los atributos del producto génico de cualquier organismo. Busca los términos GO que están sobrerepresentados o infrarepresentados en el set de genes y anota la función biológica asociada. Se presentan los resultados de la siguiente forma:
    - Tabla de términos GO anotados a partir de un test hipergeométrico. Función `go_enrich()` del paquete GOfuncR
    - Gráfico con la distribución por categorías de los genes anotados. Función `plot_anno_scores()` del paquete GOfuncR
- Análisis GSEA(Reactome)[41]. Se realiza un análisis de enriquecimiento sobre el set de genes proporcionados para ver cuales están asociados con la base de datos de Reactome. Para ello se utiliza un modelo hipergeometrico. Función `enrichPathway()` del paquete reactomePA
- Los resultados se presentan de la siguiente forma:
    - Diagrama de Barras. Muestra el resultado del análisis de enriquecimiento.
    - Diagrama de puntos. Muestra el resultado del análisis de enriquecimiento.

- Mapa de enriquecimiento (figura13). Muestra el mapa de resultados del análisis de enriquecimiento
- Asociaciones génicas. Muestra el mapa de asociaciones entre genes del análisis de enriquecimiento.

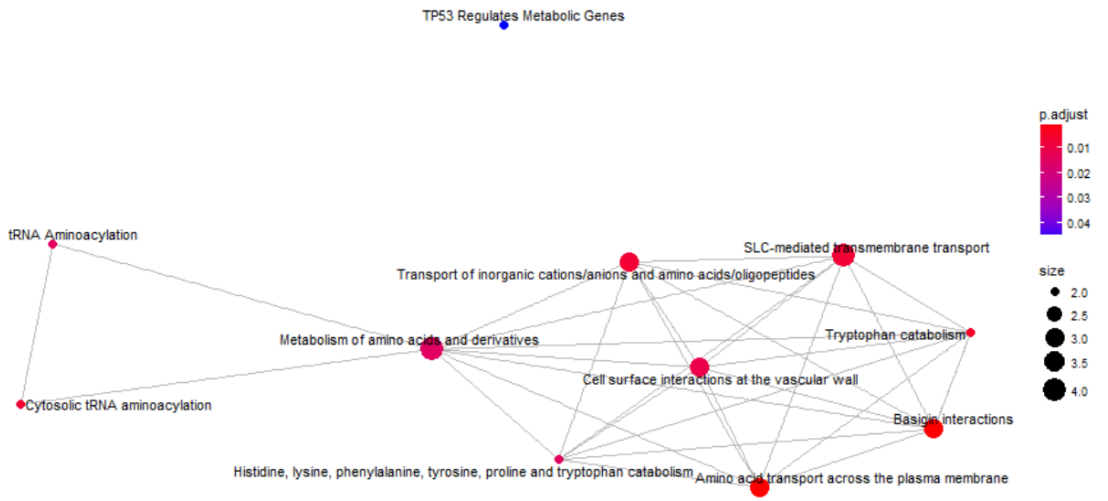


Figura 13.Mapa de enriquecimiento usando Reactome. Dataset 2

Una vez definida la pipeline, se procede a programar en R cada uno de los componentes. Una vez finalizada la programación se procede a testar la robustez de la pipeline con los datasets de prueba y a testar posibles cuellos de botella computacionales, cuyos resultados serán reportados en el capítulo específico para ellos.

6. Desarrollo de la aplicación en Shiny

Con la pipeline definida, programada y testada se procede a implementarla a una aplicación web.

Dado que la aplicación está programada en R, la herramienta utilizada para el desarrollo de la aplicación web es el paquete Shiny[42], que permite adaptar el código de R para crear aplicaciones web interactivas de forma sencilla.

Shiny implementa la programación reactiva[43], en donde los objetos (gráficos y tablas) que forman la aplicación responden a los inputs de los usuarios, dotando a estos de una gran capacidad de control a la hora de establecer los parámetros de la manera óptima para realizar el análisis de microarrays.

Las aplicaciones escritas en Shiny constan de dos partes:

- ui.R. Donde va todo el código relativo a la interfaz gráfica de la aplicación.
- server.R. Donde va todo el código relativo a las funciones necesarias para realizar el análisis.

El primer paso, fue diseñar la distribución que tendrá la pipeline en la aplicación web y la elección de Shinydashboard[44], un framework para shiny que potencia el apartado visual de la aplicación.
Posteriormente, se definieron que parámetros de la aplicación serian fijos y cuáles podrían ser controlados por el usuario.
Finalmente se procedió a implementar la pipeline diseñada, adaptando el código a las necesidades de shiny.

ShinyDashboard provee de un sistema de paneles de navegación, que hace que el uso de la aplicación por parte de los usuarios sea sencilla y muy intuitiva. La aplicación consta de los siguientes paneles de navegación[45]

- Introduction. Panel que contiene toda la información necesaria para que el usuario pueda realizar el análisis de microarrays.
- Data input. Es la sección que utilizaremos para introducir los ficheros requeridos para realizar el análisis. El panel consta de los siguientes apartados:
    - *CEL FILES*. Sección donde el usuario de la aplicación debe cargar los ficheros .CEL de las muestras a analizar
    - *TARGETS*. Sección donde el usuario debe cargar el archivo .CSV que contiene la información del experimento.
    - Input data Summary. Donde se muestran los nombres de las muestras cargadas, así como los grupos en las que están divididas.
    - *Targets Info*. Se muestra en formato tabla toda la información del archivo targets.
- Quality Control. Esta sección visualiza los resultados del control de calidad realizado a las muestras, está dividida en dos secciones:
    - Antes de normalizar. Incluye los siguientes gráficos, todos ellos descargables de forma individual en formato .PNG:
        - Histograma de intensidades
        - Boxplot de Intensidades
        - Dendograma
        - PCA
    - Después de normalizar. Incluye los siguientes gráficos todos ellos descargables de forma individual en formato .PNG:
        - Boxplot de intensidades
        - Dendograma
        - PCA

      Al final de esta sección tenemos la posibilidad de descargar los datos del experimento ya normalizados.

- DGE. Sección donde se configuran los parámetros para llevar a cabo la selección diferencial de genes, está dividido en dos subsecciones:
    - *One Group.* Sección donde se realizará la selección de grupos que formaran los contrastes del modelo líneal en los experimentos donde solo haya dos grupos a comparar. Esta sección incluye:
        - Matriz de diseño
        - Matriz de Contrastes con los grupos seleccionados
    - *Multiple Groups.* Sección para seleccionar los grupos que formaran los contrastes en aquellos experimentos con más de dos grupos a comparar, admite hasta tres comparaciones simultáneas. Esta sección incluye:
        - Matriz de diseño
        - Matriz de contrastes con los grupos seleccionados
- Results. Sección donde se presentan los resultados del análisis diferencial de expresión y se selecciona el set de genes sobre el que se realizará el análisis funcional. Está dividida en dos secciones:
    - *One Group.* Resultados para los experimentos con solo una comparación. Incluye:
        - Tabla con el resultado del DGE
        - Tabla con los genes seleccionados para su posterior análisis. Los parámetros disponibles para filtrar por parte del usuario son:
            - Adjust P Value (predefinido en 0,05)
            - Número máximo de genes a incluir (predefinido en 10)
        - Volcano Plot con el resultado del DGE. El usuario podrá controlar el número de genes a resaltar en el gráfico.(predefinido en 2)
        - HeatMap de la tabla de genes seleccionados.
    - *Multiple Groups*. Resultados para los experimentos con más de una comparación. Incluye:
        - Tabla con el resultado del DGE

- Tabla con los genes seleccionados para su posterior análisis. Los parámetros disponibles para filtrar por parte del usuario son:
    - Adjust P Value(predefinido en 0,05)
    - Número máximo de genes a incluir(predefinido en 10)
- Volcano Plot con el resultado del DGE. El usuario podrá controlar el número de genes a resaltar en el gráfico.
- HeatMap de la tabla de genes seleccionados.(predefinido en 2)
- Decide Tests. El usuario podrá controlar el P.Value.(predefinido en 0,01)
- Diagrama de Venn
- Tablas de Genes infra y sobre expresados para cada comparación.

Todos los gráficos y tablas presentados en esta sección se podrán descargar de forma individual en formato .PNG y .CSV

- GO Analysis. Sección con los resultados del análisis de Gene Ontology. El usuario debe indicar si el experimento es con dos grupos o con más de dos. Incluye:
    - Tabla con los términos GO y la función molecular anotada
    - Gráfico GO.
- GSEA. Sección donde se realiza un enriquecimiento de los genes usando la base de datos Reactome. Incluye:
    - Barplot
    - Dotplot
    - Mapa de enriquecimiento
    - Mapa de asociaciones entre genes

Antes de desplegar la aplicación a un servidor online, se testan todas las funciones programadas con los datasets de prueba y se comprueban posibles cuellos de botella computacionales. Los resultados son reportados en la sección específica para ello

Despliegue de la aplicación Online

Se decide desplegar la aplicación online en shinyapps.io, el propio servidor gratuito que ofrece el desarrollador de Shiny (Rstudio) para que la aplicación sea accesible de forma online.

Se puede acceder a la aplicación desde el siguiente enlace:

https://eirr.shinyapps.io/DGER/

El análisis GO y el análisis GSEA no se han podido implementar con éxito en el servidor gratuito de Shiny debido a los siguientes motivos:

El paquete topGO utilizado para el análisis GO inicialmente se implementó correctamente en la pipeline, pero no se pudo implementar de forma funcional en la aplicación Shiny, por lo que se buscó como alternativa el paquete GofuncR que se ha podido implantar de forma correcta y es totalmente funcional cuando se utiliza la aplicación en modo local. Cuando se realizó el despliegue al servidor de shinyapps.io, la aplicación no es capaz de realizar los análisis funcionales (GO y GSEA) debido a los requerimientos de memoria RAM necesarios(los servidores gratuitos de shinyapps.io solo tienen 1Gb de RAM).Como alternativa se dividió la aplicación en dos, una parte para el análisis diferencial de los microarrays y una segunda para realizar el análisis funcional a partir de los genes seleccionados, esta solución presenta los mismos problemas de memoria RAM y la aplicación se queda sin memoria antes de completar el análisis.

Además de los requerimientos de memoria RAM los paquetes GofuncR y reactomePA utilizan el paquete sm para realizar los gráficos, dicho paquete necesita acceder al display, algo que actualmente tampoco es compatible con el servidor Shiny de shinyapps.io. La alternativa, es utilizar paquetes diferentes para realizar el análisis funcional para líneas futuras de desarrollo de la aplicación

Para tener la aplicación web online con todas las funcionalidades se ha habilitado un servidor de RStudio provisional hasta que esté implantado definitivamente el servidor shiny

Se puede acceder al servidor habilitado desde el siguiente enlace:

http://dger.ddns.net:8787/auth-sign-in

usuario y contraseña: rstudio

Toda la información requerida para utilizar los datasets de prueba en la aplicación online se encuentra en el anexo número 3.

7. Resultados

A continuación, se reportan en forma de tabla brevemente los resultados obtenidos de la batería de pruebas realizadas, para testar el funcionamiento de la pipeline y la aplicación web. Los gráficos y tablas generados se pueden consultar de forma detallada en el anexo número 4.

	Pipeline		Aplicación Shiny Local			Aplicación Shiny Online			
Tests Realizados	Dataset 1	Dataset 2	Dataset 3	Dataset 1	Dataset 2	Dataset 3	Dataset 1	Dataset 2	Dataset 3
Carga de archivos CEL	✓	✓	✓	✓	✓	✓	✓	✓	✓
Carga de archivo Targets	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gráficos antes de normalizar	✓	✓	✓	✓	✓	✓	✓	✓	✓
Gfáficos después de normalizar	✓	✓	✓	✓	✓	✓	✓	✓	✓
Selección de contrastes	✓	✓	✓	✓	✓	✓	✓	✓	✓
Toptable	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tabla de genes seleccionados	✓	✓	✓	✓	✓	✓	✓	✓	✓
Volcano Plot	✓	✓	✓	✓	✓	✓	✓	✓	✓
Heatmap	✓	✓	✓	✓	✓	✓	✓	✓	✓
Decide Tests	*	✓	✓	*	✓	✓	*	✓	✓
Tabla de genes infra y sobre expresados	*	✓	✓	*	✓	✓	*	✓	✓
Análisis GO	✓	✓	✓	✓	✓	✓	✓	✓	✓
Anáisis GSEA	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reactividad Aplicación	✓	✓	✓	✓	✓	✓	✓	✓	✓
Descarga de gráficos y tablas	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabla 2. Resultados obtenidos de la batería de pruebas realizadas sobre la aplicación

*Pruebas no realizadas debido a que el dataset 1 solo tiene dos grupos a comparar

Test Cuello de botella computacional

Se realizaron diversos tests sobre el tiempo de computación que tardaban en realizarse algunas funciones y que hacen que haya tiempos de espera mientras se realizan los cálculos y hacen que la aplicación no vaya fluida.

Los detectados han sido los siguientes

- Carga de Datos
- Normalización
- Análisis GO
- Análisis GSEA

	Pipeline	Aplicación Shiny	
		Local	Online
Carga de Datos	1.15s	5,91s	14.64s
Normalización	9.03s	13,71s	7.21s
Análisis GO	51,25s	55,82s	35.77s
Análisis GSEA	16,48s	17.83s	13.27s

Tabla 3. Resultados en segundos de los tests de cuello de botella computacionales detectados. En rojo el que produce un tiempo de espera más elevado.

La aplicación online es más rápida a la hora de calcular las funciones necesarias para realizar el análisis debido a que cuenta con más memoria RAM, 8GB , la máquina en donde se ha desarrollado cuenta con 6GB de RAM , de ahí la diferencia en los tiempos de procesado. En cambio la aplicación online es más lenta a la hora de cargar los datos, esto es debido a que depende el ancho de banda de internet que tenga asignado el servidor en donde está alojada la aplicación.

Los resultados no han variado de forma significativa según el tamaño de los datasets de prueba. Si influye, a la hora de aumentar los tiempos de espera, si el ordenador donde se está ejecutando la aplicación tiene otros procesos abiertos consumiendo memoria.

8. Conclusiones

Durante el desarrollo de este trabajo, se ha aprendido a identificar todas las etapas necesarias para realizar un análisis de datos de microarrays y diseñar pipelines para su análisis. Se ha implementado la pipeline diseñada en una herramienta online que facilite el acceso a este tipo de análisis a científicos sin los conocimientos necesarios en bioinformática. Se ha aprendido el concepto de programación reactiva y a utilizarla en la herramienta para dotar de más control del análisis al usuario de la misma.

Se ha aprendido la importancia del entorno de trabajo (software) escogido para el desarrollo del TFM. La pipeline y la aplicación se han desarrollado con Windows como sistema operativo, y el servidor que aloja la aplicación web está basado en Ubuntu, lo que ha provocado incompatibilidades a la hora de desplegar la aplicación al servidor online.

Los objetivos propuestos en el plan de trabajo que eran diseñar una pipeline de análisis de datos e implantarla en una herramienta online se han realizado con éxito. Exceptuando la implementación del análisis funcional debido a incompatibilidades de paquetes de bioconductor y el servidor escogido para el despliegue de la misma.

La planificación ha sido la adecuada, exceptuando la parte dedicada al despliegue de la aplicación al servidor, que ha causado más dificultades de las previstas al inicio del TFM. Se han tenido que introducir cambios en los paquetes de R utilizados y eliminar funcionalidades de la aplicación para poder desplegarla de forma correcta.

Como líneas futuras de desarrollo de la aplicación , se evalúa la posibilidad de buscar un servidor alternativo para poder desplegar la aplicación con todas las funcionalidades, para ello se ha pensado en utilizar el software Docker que permite construir instancias de software , que se pueden utilizar posteriormente

en cualquier servidor con un sistema operativo que soporte la instalación de Docker.

Se ha priorizado la funcionalidad de la aplicación por encima de dotar de más control al usuario del análisis para evitar problemas a la hora de programar la reactividad de la aplicación, en futuras actualizaciones se quieren incrementar los parámetros controlados por el usuario así como el tipo de microarrays y organismos de los que realizar análisis.

9. Glosario

Affymetrix: Compañía Estadounidense, principal fabricante de Microarrays.

Bioconductor: Suite para el análisis de datos genómicos basada en R

Dataset: Colección de datos utilizados para un estudio.

DGE: Acrónimo en inglés para referirse a la expresión diferencial de genes

Ebayes: Método basado en estadística bayesiana para seleccionar genes Diferencialmente expresados

Github: Servicio de repositorios para el control de versiones de aplicaciones

GO: Acrónimo en inglés para referirse a la ontología genética.

GSEA: Acrónimo en inglés para referirse a los test de enriquecimiento de genes

Layout: Disposición o distribución de la aplicación

Microarray: Superficie sólida donde se hibridan biomolecular para su posterior análisis

PCA: Acrónimo en inglés para referirse al análisis de componentes principales

Pipeline: Etapas en las que divide un análisis de datos.

PM: Acrónimo en inglés para perfect missmatch

R: Lenguaje de programación estadística

RMA: Método utilizado para normalizar datos de Microarrays

Rstudio: Entorno de trabajo gráfico para el lenguaje R

Script: Conjunto de código informático que ejecuta una aplicación

Shiny: Paquete de R que permite desarrollar aplicaciones interactivas

ShinyDashboard: Interfaz gráfica para mejorar la visualización delas aplicaciones creadas con shiny

TFM: Acrónimo trabajo final de máster

Trancriptomica: Ciencia que estudia los transcritos de RNA

10. Bibliografía

[1] M. Kaushik and S. Mahendru, “Genomic Databases and Softwares: In Pursuit of Biological Relevance through Bioinformatics,” *Adv. Tech. Biol. Med.*, vol. 4, no. 3, 2015.

[2] A. Bayat, “Science, medicine, and the future: Bioinformatics.,” *BMJ Br. Med. J.*, vol. 324, no. 7344, pp. 1018–1022, 2002.

[3] K. Hokamp *et al.*, “ArrayPipe: A flexible processing pipeline for microarray data,” *Nucleic Acids Res.*, vol. 32, no. WEB SERVER ISS., pp. 457–459, 2004.

[4] M. C. Ruíz de Villa and A. Sánchez-Pla, “Mod 2. Análisis de Datos de Microarrays,” p. 73, 2010.

[5] A. Koschmieder, K. Zimmermann, S. Trißl, T. Stoltmann, and U. Leser, “Tools formanaging and analyzing microarray data,” *Brief. Bioinform.*, vol. 13, no. 1, pp. 46–60, 2012.

[6] S. Alex, “Ejemplo de An a ´ lisis de Datos de Microarrays con R y Bioconductor,” *Group*, pp. 1–17, 2009.

[7] A. De Stasio, M. Ertelt, W. Kemmner, U. Leser, and M. Ceccarelli, “Exploiting scientific workflows for large-scale gene expression data analysis,” *2009 24th Int. Symp. Comput. Inf. Sci.*, no. May 2014, pp. 448–453, 2009.

[8] Y. Li and J. Andrade, “DEApp: An interactive web interface for differential expression analysis of next generation sequence data,” *Source Code Biol. Med.*, vol. 12, no. 1, pp. 10–13, 2017.

[9] “Microarrays with Galaxy.” [Online]. Available: http://www.ensat.ac.ma/mobihic/microarray-galaxy.html. [Accessed: 04-Jun-2018].

[10] Z. C. Dong and Y. Chen, “Transcriptomics: Advances and approaches,” *Sci. China Life Sci.*, vol. 56, no. 10, pp. 960–967, 2013.

[11] R. Bumgarner, “DNA microarrays: Types, Applications and their future,”

Curr Protoc Mol Biol., vol. 6137, no. 206, pp. 1–17, 2013.

[12] V. Moreno, “Análisis estadístico de microarrays de ADN.”

[13] “3D-Geme.” [Online]. Available: http://www.3d-gene.com/en/about/chip/chi_003.html. [Accessed: 03-Jun-2018].

[14] M. C. Ruíz de Villa and A. Sánchez-Pla, “Mod 1. Preliminares.”

[15] C. A. Bradley Coe, “Spot your Genes, An overview of the microarray.” [Online]. Available: https://www.scq.ubc.ca/spot-your-genes-an-overview-of-the-microarray/. [Accessed: 31-May-2018].

[16] R. B. Stoughton, “Applications of Dna Microarrays in Biology,” *Annu. Rev. Biochem.*, vol. 74, no. 1, pp. 53–82, 2005.

[17] R. Fisler, “The Role of Microarray Technology,” *drug discovery world*. [Online]. Available: http://www.ddw-online.com/drug-discovery/p97050-the-role-of-microarray-technologywinter-06.html. [Accessed: 03-Jun-2018].

[18] R. Santamar, “microarray Análisis de datos de S □.”

[19] G. B. Whitworth, “An Introduction to Microarray Data Analysis and Visualization,” *Comput. Genomics Theory Appl.*, pp. 19–50, 2010.

[20] S. M. Angelsk, “Microarray pipeline The microarray pipeline,” *Transformation*, no. September, 2009.

[21] S. Andrzejewski *et al.*, “PGC-1α Promotes Breast Cancer Metastasis and Confers Bioenergetic Flexibility against Metabolic Drugs,” *Cell Metab.*, vol. 26, no. 5, p. 778–787.e5, 2017.

[22] Y.-P. Weng *et al.*, “The inhibitory activity of gallic acid against DNA methylation: Application of gallic acid on epigenetic therapy of human cancers,” *Oncotarget*, vol. 9, no. 1, pp. 361–374, 2018.

[23] T. Cooks *et al.*, “Mutant p53 cancers reprogram macrophages to tumor supporting macrophages via exosomal miR-1246,” *Nat. Commun.*, vol. 9, no. 1, 2018.

[24] B. Klaus, “An end to end workflow for differential gene expression using Affymetrix microarrays,” *F1000Research*, vol. 5, p. 1384, 2016.

[25] Bioconductor Maintainer, “Using Bioconductor for Microarray Analysis.” .

[26] Yihui, “knitr R package.” [Online]. Available: https://yihui.name/knitr/. [Accessed: 06-Jun-2018].

[27] T. Package, “Package ‘ DT ,’” 2018.

[28] B. S. Carvalho, “oligo User’s Guide,” pp. 1–51, 2015.

[29] G. K. Smyth, M. Ritchie, and N. Thorne, “Linear Models for Microarray and RNA-Seq Data User ’ s Guide,” *R*, no. September, 2015.

[30] A. Ploner, “Creating heatmaps using package Heatplus,” 2014.

[31] A. Alexa, “Gene set enrichment analysis with topGO,” 2014.

[32] S. Grote, “GOfuncR: Gene Ontology Enrichment Using FUNC.” .

[33] G. Yu and Q.-Y. He, “ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization,” *Mol. BioSyst.*, vol. 12, no. 2, pp. 477–479, 2016.

[34] T. Raman *et al.*, “Quality control in microarray assessment of gene expression in human airway epithelium,” *BMC Genomics*, vol. 10, p. 493, 2009.

[35] T. Park, S. G. Yi, S. H. Kang, S. Y. Lee, Y. S. Lee, and R. Simon, “Evaluation of normalization methods for microarray data,” *BMC Bioinformatics*, vol. 4, pp. 1–13, 2003.

[36] B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed, “A comparison of normalization methods for high density oligonucleotide array data based on variance and bias,” *Bioinformatics*, vol. 19, no. 2, pp. 185–193, 2003.

[37] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-barclay, K. J. Antonellis, and T. P. Speed, “Exploration , Normalization , and Summaries of High Density Oligonucleotide Array Probe Level Data,” no. June, pp. 249–264, 2018.

[38] C. Soneson and M. Delorenzi, “A comparison of methods for differential expression analysis of RNA-seq data,” *BMC Bioinformatics*, vol. 14, no. 1, p. 1, 2013.

[39] M. E. Ritchie *et al.*, “Limma powers differential expression analyses for RNA-sequencing and microarray studies,” *Nucleic Acids Res.*, vol. 43, no. 7, p. e47, 2015.

[40] W. Li, “Application of Volcano Plots in Analyses of mRNA Differential Expressions with Microarrays,” no. March 2011, pp. 2015–2018, 2011.

[41] R. Gentleman, “Gene Set Enrichment Analysis,” no. x, pp. 1–9, 2006.

[42] Rstudio, “Shiny Tutorial.” [Online]. Available: https://shiny.rstudio.com/tutorial/. [Accessed: 30-Apr-2018].

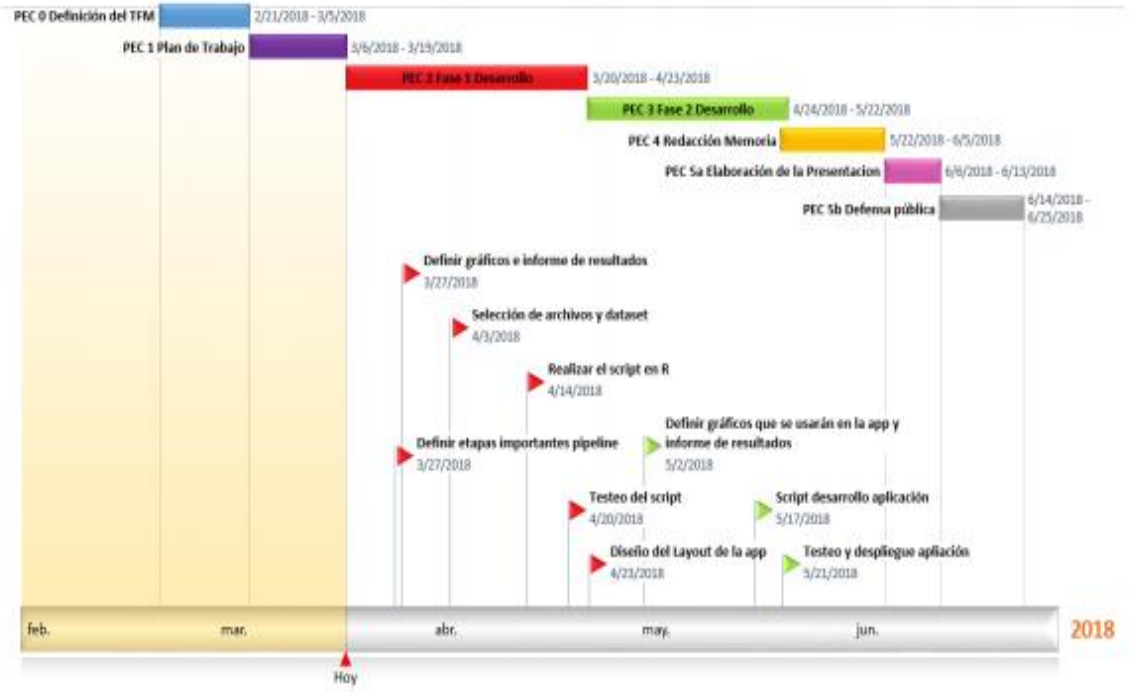
[43] Rstudio, “Reactivy - An overview.” [Online]. Available: https://shiny.rstudio.com/articles/reactivity-overview.html. [Accessed: 25-May-2018].

[44] Rstudio, “ShinyDashboard.” [Online]. Available: https://rstudio.github.io/shinydashboard/get_started.html. [Accessed: 15-May-2018].

[45] J. S. Martial Luyts, “Tutorial in R Shiny package: Developing web applications in the area of Biostatistisc & Data Science.” [Online]. Available: https://ibiostat.be/seminar/uploads/introdcution-r-shiny-package-20160330.pdf. [Accessed: 30-Apr-2018].

11. Anexos

Anexo 1 Diagrama de Gantt planificación temporal inicial



Anexo 2. Resultados

Pipeline

La pipeline diseñada obtiene los siguientes resultados satisfactorios con los tres datasets:

- Lectura de archivos CEL y archivo targets
- Control de calidad antes y después de normalizar:
    - Histograma de intensidades
    - Boxplot
    - Dendograma
    - PCA Analysis
- Selección de grupos para la matriz de contraste para los experimentos con una y más de una comparación
- Tabla con los resultados del análisis de la expresión diferencial
- Tabla de genes seleccionados con los criterios definidos por el usuario
    - Selección por Adjust.P.Value
    - Selección del máximo de genes a incluir
- Volcano plot
    - Selección de genes a mostrar
- Heatmap
- Tabla con el resumen del test de comparaciones múltiples(experimentos para más de una comparación)
    - Selección por P.Value
- Diagrama de Venn(experimentos con más de una comparación)
- Tabla con la lista de genes sobre e infra expresados(experimentos para más de una comparación)
- Análisis GO
    - Tabla con términos GO anotados
    - Resumen del análisis GO realizado
- Análisis GSEA
    - Gráfico de barras
    - Gráfico de puntos
    - Mapa de enriquecimiento
    - Mapa de asociaciones intergénicas

El código de la pipeline se puede encontrar integro en el Anexo 1

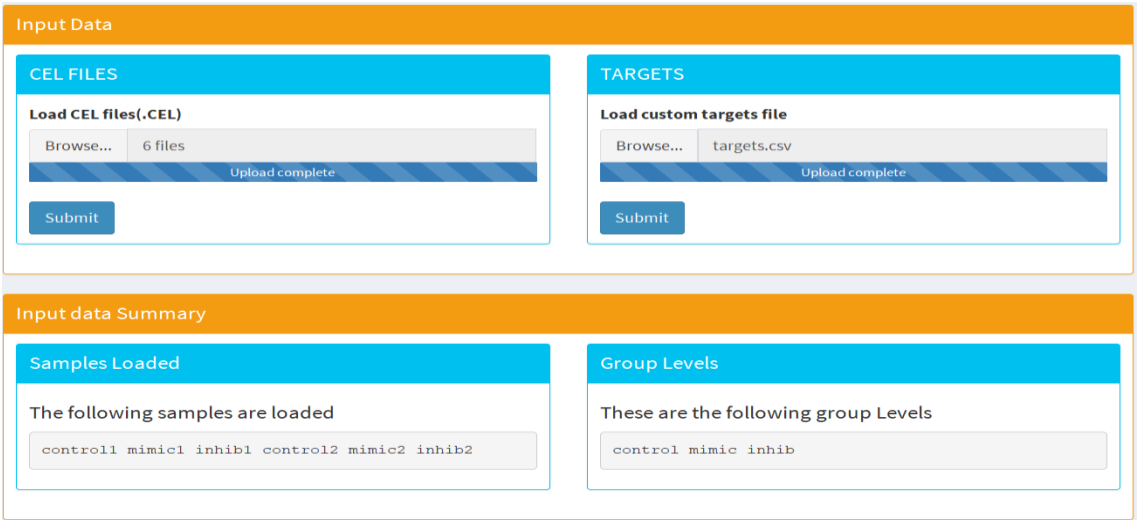
Aplicación web

Para la aplicación web, testada en local se obtienen los mismos resultados favorables que para la pipeline

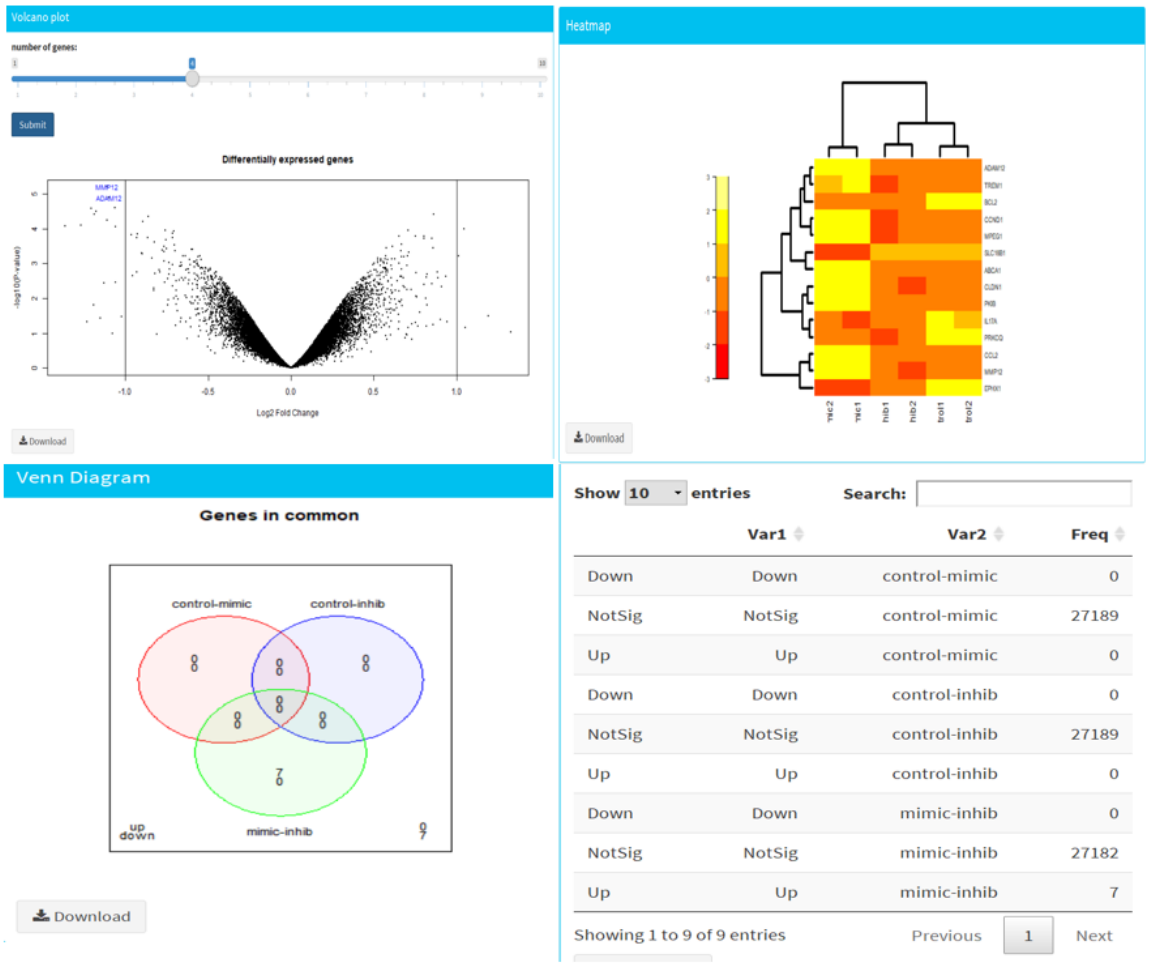
Se obtiene los siguientes resultados favorables en la aplicación web:

- Lectura de archivos CEL y archivo targets
- Control de calidad antes y después de normalizar:
    - Histograma de intensidades
    - Boxplot
    - Dendograma
    - PCA Análisis
- Selección de grupos para la matriz de contraste para los experimentos con una y más de una comparación
- Tabla con los resultados del análisis de la expresión diferencial
- Tabla de genes seleccionados con los criterios definidos por el usuario
    - Selección por Adjust.P.Value
    - Selección del máximo de genes a incluir
- Volcano plot
    - Selección de genes a mostrar
- Heatmap
- Tabla con el resumen del test de comparaciones múltiples(experimentos para más de una comparación)
    - Selección por P.Value
- Diagrama de Venn(experimentos con más de una comparación)
- Tabla con la lista de genes sobre e infra expresados(experimentos para más de una comparación)
- Análisis GO
    - Tabla con términos GO anotados
    - Resumen del análisis GO realizado
- Análisis GSEA

- Gráfico de barras
- Gráfico de puntos
- Mapa de enriquecimiento
- Mapa de asociaciones intergénicas



Pestaña de Carga de datos de la aplicación desarrollada, con el dataset 2 cargado.



Show 10 entries Search:

	Var1	Var2	Freq
Down	Down	control-mimic	0
NotSig	NotSig	control-mimic	27189
Up	Up	control-mimic	0
Down	Down	control-inhib	0
NotSig	NotSig	control-inhib	27189
Up	Up	control-inhib	0
Down	Down	mimic-inhib	0
NotSig	NotSig	mimic-inhib	27182
Up	Up	mimic-inhib	7

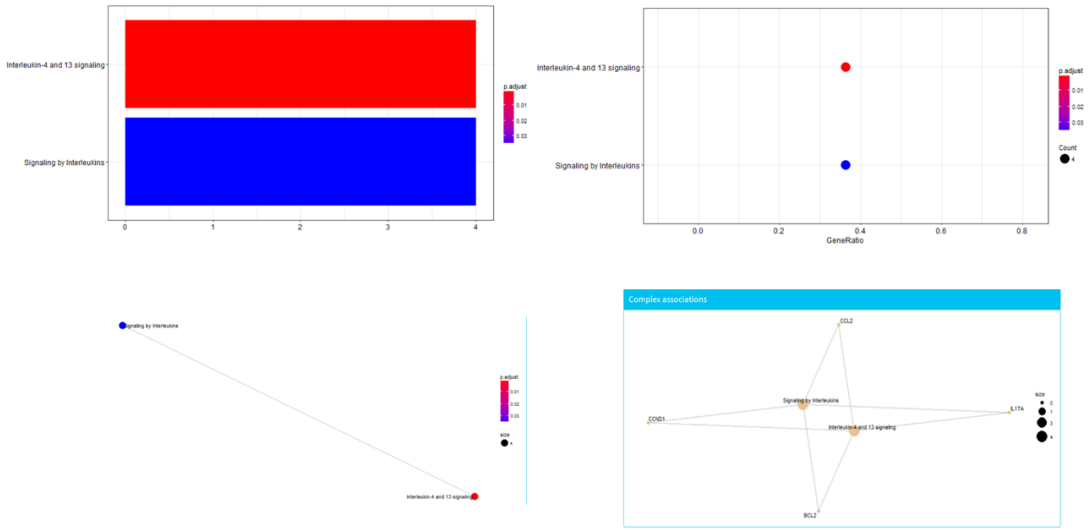
Showing 1 to 9 of 9 entries Previous 1 Next

Gráficos generados por el apartado de resultados para el dataset numero 3.

Show 10 entries Search:

	go_id	gene	go_name	root_node
1	GO:0006952	BCL2	defense response	biological_process
2	GO:0006952	CCL2	defense response	biological_process
3	GO:0006952	CLDN1	defense response	biological_process
4	GO:0006952	IL17A	defense response	biological_process
5	GO:0006952	MMP12	defense response	biological_process
6	GO:0006952	PRKCQ	defense response	biological_process
7	GO:0006952	TREM1	defense response	biological_process
8	GO:0009605	ABCA1	response to external stimulus	biological_process
9	GO:0009605	BCL2	response to external stimulus	biological_process
10	GO:0009605	CCL2	response to external stimulus	biological_process

Tabla análisis GO generada para el dataset 1



Gráficos obtenidos para el análisis GSEA del dataset 3

Anexo 3. Repositorio Github y acceso al servidor Rstudio

El código de la aplicación web se encuentra disponible para su consulta en el repositorio oficial de la aplicación en Github https://github.com/eirr82/DGER- .
En el repositorio se encuentran los siguientes archivos:

- README.md. Archivo de información que explica cómo utilizar la aplicación y que pasos a seguir si se quiere instalar de modo local.
- Intrucctions. Archivo de texto que explica como utilizar la aplicación(es el mismo texto que se utiliza en la página de ayuda de la aplicación)
- Ui.R. Código de la interfaz de la aplicación
- Server.R. Código de las funciones que generan los resultados de la aplicación.
- App.R. Archivo que reúne el código de ui.R y server.R y que puede ser ejecutado de forma local

Los datasets que se han utilizado para desarrollar la aplicación se pueden descargar desde el siguiente enlace (incluye el archivo targets):
https://drive.google.com/drive/folders/1-YoXdp7qdCX8OfT7SvpYS-CD2_7yCIcC

Para acceder al servidor online de Rstudio habilitado para ejecutar la aplicación debemos seguir los siguientes pasos

- Entrar en el siguiente enlace http://dger.ddns.net:8787/
- Loguear en el servidor usando *rstudio* como usuario y contraseña
- Apretar el botón *Run App* dentro de la consola de Rstudio

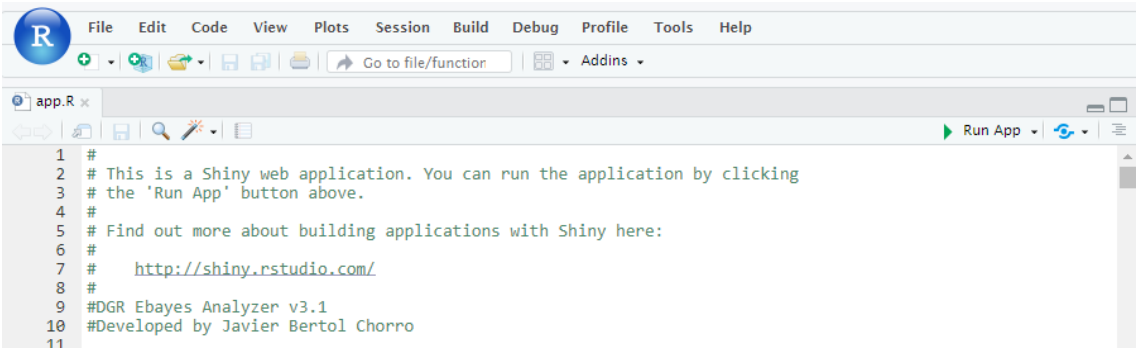


Imagen principal del servidor Rstudio habilitado para ejecutar la aplicación.

Anexo 4.Codigo Pipeline

```
##SCRIPT PIPELINE TFM##
#Cargar librerías a utilizar en el análisis
library(oligo)
library(pd.clariom.s.human)
library(limma)
library(annotate)
library(clariomshumantranscriptcluster.db)
library(topGO)
library(knitr)
library(affycoretools)
library(Heatplus)

##Carga de datos##
#carga archivo targets
targets <- read.csv("C:/Users/Cristian/Desktop/TFM/datos/set1/targets.csv", sep=";")

#carga cel files(paquete oligo)
celFiles <- list.celfiles("C:/Users/Cristian/Desktop/TFM/datos/set1/", full.names=TRUE)
rawData <- read.celfiles(celFiles)

#Creamos la información del archivo targets
info <- data.frame(grupo=c(1,1,1,2,2,2))
sampleNames <- targets$Name
color <- c("chocolate4","chocolate4","chocolate4","royalblue4","royalblue4","royalblue4")
sampleGrops <- targets$Group
sampleInfo  <- targets$FileNames

##Control de calidad##
#antes de normalizar
```

```
#histograma
hist(rawData, main="Signal distribution", col=color, lty=1:ncol(info))
legend (x="topright", legend=sampleNames , col=color, lty=1:ncol(info))

#Boxplot de las muestras
boxplot(rawData, cex.axis=0.6, col=color, las=2, names=sampleNames)
legend(x="topright", legend = sampleGrops, col=color, lty=1:ncol(info))

#Dendograma La muestras del mismo grupo deberìan agruparse juntas
clust.euclid.average <- hclust(dist(t(exprs(rawData))),method="average")
plot(clust.euclid.average,  labels=sampleNames,  main="Hierarchical  clustering
of samples",  hang=-1)

#PCA analisis
plotPCA(rawData)

##NORMALIZACION##
#Normalizar
eset_rma <- oligo::rma(rawData) #Usar rma() del paquete Oligo no del paquete
AFFY

##Control de calidad después de Normalizar##
#boxplot despues de normalizar
boxplot(eset_rma, cex.axis=0.6, col=color, las=2, names=sampleNames)
legend(x="topright", legend = sampleGrops, col=color, lty=1:ncol(info)
#dendograma
clust.euclid.average <- hclust(dist(t(exprs(eset_rma))),method="average")
plot(clust.euclid.average,  labels=sampleNames,  main="Hierarchical  clustering
of samples",  hang=-1)

#PCA Analysis
plotPCA(eset_rma)

##DGE análisis con limma##
```

```
#crear la matriz de diseño del experimento
groups <- as.character(targets$Group)
groups <- as.factor(groups)
lev<-factor(groups, levels=unique(groups))
design <- model.matrix(~0+lev)
colnames(design)<-levels(lev)
rownames(design) <-sampleInfo
kable(design)

#Crear contrastes
cont.matrix <- makeContrasts(
  controlvstratamiento = (control-tratamiento),
  levels = design
)
kable(cont.matrix)

#DGE por ebayes
fit <- lmFit(eset_rma, design)
fit.main <- contrasts.fit(fit, cont.matrix)
fit.main <- eBayes(fit.main)

#Selección de genes diferencialmente expresados
#log2fold, p.value etc(se implementará en la aplicación diretamente)

toptable <- topTable(fit.main, coef=1, n=1000)

#eliminar la columna probeid y eliminar los NA
toptable <- na.omit(toptable)
toptable$PROBEID <-NULL

#Anotación manual en caso de que la anotación via "affycoretools" cree cuello
computacional en el servidor de shiny

###Anotacion de resultados
```

```
genesymbols <- getSYMBOL(rownames(toptable1), "clariomshumantranscriptcluster.db")
results1 <- cbind(genesymbols,toptable1)
results1 <- na.omit(results1)

#show few records of a toptable
kable(toptable[1:20,])

#select top15 genes (para mostrar resultados)
#en la aplicacíon el usuario decidirá los criterios para escoger la lista de genes
selected <- toptable[1:15,]

##Volcano Plot(con solo un contraste)##
#volcano plot
volcanoplot(fit.main,coef=1,highlight=5, names=names(fit.main$coefficients[,1]),
          main=paste("Differentially expressed genes", sep="\n"))
abline(v=c(-1,1))

#Decide test y diagrama de Venn para arrays con mas de una comparacion(cuando las haya)
res<-decideTests(fit.main, method="separate", adjust.method="fdr", p.value=0.01)
sum.res.rows<-apply(abs(res),1,sum)
res.selected<-res[sum.res.rows!=0,]
print(summary(res))

#diagrama de venn
vennDiagram (res.selected[,1:3], main="Genes in common", cex=0.9, circle.col = c('red', 'blue', 'green'), include="up")

#Heatmap de la lista de genes seleccionada
probeNames<-rownames(res)
probeNames.selected<-probeNames[sum.res.rows!=0]
exprs2cluster <-exprs(eset_rma)[probeNames.selected,]
```

```
pal <- colorRampPalette(brewer.pal(11, "RdYlGn"))(100)
heatmap(exprs2cluster, col=pal)

#Heatmap para experimentos con sola un contraste
geneID = rownames(topTable(fit.main, coef=1, num=100, adjust="none",p.value=0.05))
exdat = eset_rma[geneID]
reg2 = regHeatmap(exprs(exdat), legend=2, col=heat.colors, breaks=-3:3)
plot(reg2)

##Go análisis##
##Paquete topGO##
#A partir de la lista generada por toptable con los diferencialmente expresados como selected
geneID <-  rownames(toptable)

#creamos la lista de genes interesantes que corresponde a los seleccionados de la toptable
myInterestingGenes <- rownames(selected)
geneList <- factor(as.integer(geneID %in% myInterestingGenes))
names(geneList) <- geneID

#Creamos el objeto topgo
top_GO_data <- new("topGOdata", ontology = "BP", allGenes = geneList,
                   nodeSize = 10, annot=annFUN.db, affyLib = "clariomshumantranscriptcluster.db")
#Test de Fisher
resultFisher <- runTest(top_GO_data, algorithm = "classic", statistic = "fisher")

#enrichment with Kolmogorov-Smirnov test con los modod clasico y elimin
resultKS <- runTest(top_GO_data, algorithm = "classic", statistic = "ks")
resultKS.elim <- runTest(top_GO_data, algorithm = "elim", statistic = "ks")

#resultados
```

```
allRes <- GenTable(top_GO_data, classicFisher = resultFisher,
                   classicKS = resultKS, elimKS = resultKS.elim,
                   orderBy = "elimKS", ranksOf = "classicFisher", topNodes = 10)

#mostrar resultados
kable(allRes)

#mostrar gráfico
showSigOfNodes(top_GO_data, score(resultKS.elim), firstSigNodes = 5,
useInfo = 'all')

#fin de la pipeline
```

Anexo 5. Requerimientos de Software y Hardware necesarios para instalar la aplicación en modo local

Hardware mínimo recomendado

- Intel Core i3 2.5G hz o superior
- 4GB de RAM (8GB recomendados)
- 2GB de espacio en el disco duro

Software utilizado para el desarrollo del proyecto

R version 3.5.0 (2018-04-23)

Platform: x86_64-w64-mingw32/x64 (64-bit)

Running under: Windows >= 8 x64 (build 9200)

Matrix products: default

locale:

[1] LC_COLLATE=Spanish_Spain.1252 LC_CTYPE=Spanish_Spain.1252 LC_MONETARY=Spanish_Spain.1252

[4] LC_NUMERIC=C LC_TIME=Spanish_Spain.1252

attached base packages:

[1] stats4 parallel stats graphics grDevices utils datasets methods base

other attached packages:

[1] clusterProfiler_3.9.1 ReactomePA_1.25.0

[3] affycoretools_1.53.2 Homo.sapiens_1.3.1

[5] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2 GO.db_3.6.0

[7] OrganismDbi_1.23.0 GenomicFeatures_1.33.0

[9] GenomicRanges_1.33.5 GenomeInfoDb_1.17.1

[11] shinycssloaders_0.2.0 Heatplus_2.27.0

[13] DT_0.4 knitr_1.20

[15] GOfuncR_1.1.1 vioplot_0.2

[17] sm_2.2-5.5 clariomshumantranscriptcluster.db_8.7.0

[19] org.Hs.eg.db_3.6.0 annotate_1.59.0
[21] XML_3.98-1.11 AnnotationDbi_1.43.1
[23] limma_3.37.1 pd.clariom.s.human_3.14.1
[25] DBI_1.0.0 oligo_1.45.0
[27] Biobase_2.41.0 oligoClasses_1.43.1
[29] RSQLite_2.1.1 Biostrings_2.49.0
[31] XVector_0.21.1 IRanges_2.15.13
[33] S4Vectors_0.19.7 BiocGenerics_0.27.0
[35] shinydashboard_0.7.0 shiny_1.1.0

loaded via a namespace (and not attached):

[1] R.utils_2.6.0 tidyselect_0.2.4 htmlwidgets_1.2
[4] grid_3.5.0 BiocParallel_1.15.5 munsell_0.4.3
[7] units_0.5-1 codetools_0.2-15 preprocessCore_1.43.0
[10] colorspace_1.3-2 GOSemSim_2.7.0 BiocInstaller_1.31.1
[13] Category_2.47.0 rstudioapi_0.7 DOSE_3.7.0
[16] GenomeInfoDbData_1.1.0 hwriter_1.3.2 bit64_0.9-7
[19] biovizBase_1.29.0 affxparser_1.53.0 R6_2.2.2
[22] locfit_1.5-9.1 AnnotationFilter_1.5.1 bitops_1.0-6
[25] reshape_0.8.7 fgsea_1.7.0 DelayedArray_0.7.2
[28] assertthat_0.2.0 promises_1.0.1 scales_0.5.0
[31] ggraph_1.0.1 nnet_7.3-12 enrichplot_1.1.2
[34] gtable_0.2.0 affy_1.59.0 ggbio_1.29.0
[37] ensembldb_2.5.1 rlang_0.2.1 genefilter_1.63.0
[40] splines_3.5.0 rtracklayer_1.41.2 lazyeval_0.2.1
[43] acepack_1.4.1 dichromat_2.0-0 checkmate_1.8.5
[46] yaml_2.1.19 reshape2_1.4.3 crosstalk_1.0.0
[49] backports_1.1.2 rsconnect_0.8.8 httpuv_1.4.3
[52] qvalue_2.13.0 Hmisc_4.1-1 RBGL_1.57.0
[55] tools_3.5.0 tcltk_3.5.0 ggplot2_2.2.1
[58] affyio_1.51.0 gplots_3.0.1 ff_2.2-14
[61] RColorBrewer_1.1-2 ggridges_0.5.0 Rcpp_0.12.17
[64] plyr_1.8.4 base64enc_0.1-3 progress_1.1.2
[67] zlibbioc_1.27.0 purrr_0.2.5 RCurl_1.95-4.10

[70] prettyunits_1.0.2 rpart_4.1-13 viridis_0.5.1
[73] cowplot_0.9.2 mapplots_1.5.1 ggrepel_0.8.0
[76] SummarizedExperiment_1.11.3 cluster_2.0.7-1 magrittr_1.5
[79] data.table_1.11.4 DO.db_2.9 reactome.db_1.64.0
[82] ProtGenerics_1.13.0 matrixStats_0.53.1 mime_0.5
[85] xtable_1.8-2 gcrma_2.53.0 gridExtra_2.3
[88] compiler_3.5.0 biomaRt_2.37.1 tibble_1.4.2
[91] KernSmooth_2.23-15 ReportingTools_2.21.0 R.oo_1.22.0
[94] htmltools_0.3.6 GOstats_2.47.0 later_0.7.2
[97] Formula_1.2-3 tidyr_0.8.1 udunits2_0.13
[100] geneplotter_1.59.0 tweenr_0.1.5 rappdirs_0.3.1
[103] MASS_7.3-49 Matrix_1.2-14 R.methodsS3_1.7.1
[106] gdata_2.18.0 igraph_1.2.1 bindr_0.1.1
[109] pkgconfig_2.0.1 rvcheck_0.1.0
GenomicAlignments_1.17.1
[112] foreign_0.8-70 foreach_1.4.4 AnnotationForge_1.23.0
[115] stringr_1.3.1 VariantAnnotation_1.27.1 digest_0.6.15
[118] graph_1.59.0 fastmatch_1.1-0 htmlTable_1.12
[121] edgeR_3.23.2 GSEABase_1.43.0 curl_3.2
[124] graphite_1.27.2 Rsamtools_1.33.0 gtools_3.5.0
[127] jsonlite_1.5 PFAM.db_3.6.0 bindrcpp_0.2.2
[130] viridisLite_0.3.0 BSgenome_1.49.0 pillar_1.2.3
[133] lattice_0.20-35 GGally_1.4.0 httr_1.3.1
[136] survival_2.41-3 glue_1.2.0 UpSetR_1.3.3
[139] iterators_1.0.9 bit_1.1-14 Rgraphviz_2.25.0
[142] ggforce_0.1.2 stringi_1.2.2 blob_1.1.1
[145] DESeq2_1.21.3 latticeExtra_0.6-28 caTools_1.17.1
[148] memoise_1.1.0 dplyr_0.7.5