UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Modelos estadísticos para el análisis integrativo de experimentos multi-ómicos

Máster Universitario en Ingeniería de Análisis de Datos, Mejora de Procesos y Toma de Decisiones

Autora

Blanca Tomás Riquelme

Tutora

Sonia Tarazona Campos

Cotutora externa

Ana Conesa Cegarra

DEPARTAMENTO DE ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

2018

AGRADECIMIENTOS

En primer lugar, quiero agradecer a Ana y a Sonia que me dieran la oportunidad de pertenecer a un grupo tan bueno como es el grupo de Genómica de la Expresión Génica del CIPF. Ha sido un auténtico placer. Todo lo que me llevo es bueno.

En especial quiero expresar mi agradecimiento a mi tutora de TFM. Sonia, para mí has sido mucho más que una tutora de TFM, has sido un referente. Mil gracias por enseñarme tanto profesionalmente, pero sobre todo como persona. Mis gracias nunca serán suficientes.

A mis padres, por hacer posible que consiga mis propósitos. A mi hermano y mi hermana por ser como mis segundos padres. No puedo olvidarme de mi sobrino: Jordiet das mucha guerra pero eso es lo que hace que te quiera tanto. Eres un crack. Este trabajo es mi forma de agradeceros el estar siempre conmigo. Siempre.

Y, por último, pero no menos importante, a mi compañero de viaje, Sergio. Gracias por hacerme creer que puedo, por ayudarme a levantarme cuando más lo he necesitado. Sabes lo que esto significa para mí y siempre he tenido tu apoyo y comprensión. Esto no habría sido posible sin ti. Gracias, gracias y gracias por ser y estar.

RESUMEN

Los avances recientes en las tecnologías de secuenciación nos han brindado una oportunidad sin precedentes para entender mejor el papel de la genómica, epigenética y transcriptómica en la regulación de la expresión génica. Un análisis integrativo de experimentos en los que se han medido varias de estas ómicas proporciona una descripción más clara y completa de los procesos biológicos, fenotipos o enfermedades estudiados. Sin embargo, muchas de las estrategias de integración de datos descritas en la literatura consisten en analizar por separado cada tipo de datos ómicos y comparar "integrativamente" los resultados, ya que todavía se carece de herramientas estadísticas para la integración de datos que puedan ser utilizadas por investigadores que no son expertos en estadística.

El desarrollo de una herramienta estadística para la integración de datos multi-ómicos plantea no pocos desafíos. Primero, la gran cantidad de variables ómicas (miles de genes) en comparación con los pequeños tamaños de muestra en este tipo de experimentos (por lo general, no más de decenas de muestras biológicas). En segundo lugar, el nivel inherente de ruido en las tecnologías ómicas. Y tercero, el diseño de una herramienta que sirva para una amplia gama de escenarios biológicos o diseños experimentales y ayude a interpretar los resultados y a responder a las preguntas biológicas.

En este trabajo, hemos explorado diversas estrategias estadísticas para superar estas limitaciones en el contexto de los modelos lineales generalizados de forma que se puedan obtener modelos útiles que expliquen la regulación de la expresión génica. Para facilitar la aplicación de dichas estrategias a usuarios no estadísticos, hemos generado una librería en el lenguaje estadístico R y una aplicación web mediante Shiny que se ha puesto a disposición de cualquier investigador a través del repositorio público *Bitbucket*: https://bitbucket.org/ConesaLab/more/.

Palabras clave: integración de datos multi–ómicos, regulación de la expresión génica, modelos lineales generalizados, bioinformática, paquete en R, Shiny.

RESUM

Els avanços recents en les tecnologies de seqüenciació ens han brindat una oportunitat sense precedents per a entendre millor el paper de la genòmica, epigenètica i transcriptómica en la regulació de l'expressió gènica. Una anàlisi integrativa d'experiments en què s'han mesurat diverses d'estes òmiques proporciona una descripció més clara i completa dels processos biològics, fenotips o malalties estudiats. No obstant això, moltes de les estratègies d'integració de dades descrites en la literatura consistixen a analitzar per separat cada tipus de dades òmiques i comparar "integrativament" els resultats, ja que encara no es tenen ferramentes estadístiques per a la integració de dades que puguen ser utilitzades per investigadors que no són experts en estadística.

El desenvolupament d'una ferramenta estadística per a la integració de dades multi-òmiques planteja no pocs reptes. Primer, la gran quantitat de variables òmiques (milers de gens) en comparació amb les xicotetes grandàries de mostra en aquest tipus d'experiments (generalment, no més de desenes de mostres biològiques). En segon lloc, el nivell inherent de soroll en les tecnologies òmiques. I tercer, el disseny d'una eina que servisca per a una àmplia gamma d'escenaris biològics o dissenys experimentals i ajude a interpretar els resultats i a respondre a les preguntes biològiques.

En aquest treball, hem explorat diverses estratègies estadístiques per a superar aquestes limitacions en el context dels models lineals generalitzats de manera que es puguen obtindre models útils que expliquen la regulació de l'expressió gènica. Per a facilitar l'aplicació d'aquestes estratègies a usuaris no estadístics, hem generat una llibreria en el llenguatge estadístic R i una aplicació web mitjançant Shiny que s'ha posat a la disposició de qualsevol investigador a través del repositori públic *Bitbucket*: https://bitbucket.org/ConesaLab/more/.

Paraules clau: integració de dades multi–òmiques, regulació de l'expressió gènica, models lineals generalitzats, bioinformàtica, paquet en R, Shiny.

ABSTRACT

Recent advances in sequencing technologies have given us an unprecedented opportunity to better understand the role of genomic, epigenetic and transcriptomic features in the regulation of gene expression. An integrative analysis of experiments in which several of these omics have been measured provide a more full and clear picture of the studied biological processes, phenotypes or diseases. However, many of the data integration strategies described in the literature consist on separately analysing each omic data type and "integratively" comparing the results since there is still a lack of statistical tools to perform data integration that can be used by researchers that are not expert on statistics.

Developing a statistical tool for multi-omic data integration poses no few challenges. First, the huge number of omic variables (thousands of genes) compared with the small sample sizes in this type of experiments (usually not more than tens of biological samples). Second, the inherent level of noise in the omic technologies. And third, the design of a tool that serves for a wide range of biological scenarios or experimental designs and helps to interpret the results and answer the biological questions.

In this work, we have explored various statistical strategies to overcome these limitations in the context of generalized linear models so that useful models can be obtained that explain the regulation of gene expression. To facilitate the application of these strategies to non-statistical users, we have generated a library in the R statistical language and a Shiny web application that has been made available to any researcher through the Bitbucket public repository: https://bitbucket.org/ConesaLab/more/.

Key words: multiomic data integration, gene expression regulation, generalized linear models, bioinformatics, R package, Shiny.

Lista de Figuras

Lista de Tablas

Nomenclatura

AIC	Criterio de información de Akaike
IDs	Identificadores de los genes
miRNA	microRNA
MORE	Multi-Omics REgulation
TF	Factor de transcripción
ADNc	ADN complementario
ADN	Ácido desoxirribonucleico
ARNm	ARN mensajero
ARN	Ácido ribonucleico
A	Base nitrogenada adenina
C	Base nitrogenada citosina
GLM	Modelos lineales generalizados
G	Base nitrogenada guanina
m.a.s	Muestra aleatoria simple
MLR	Modelos de regresión lineal múltiple
MOSim	Multi–Omics Simulation
RV	Razón de verosimilitud
T	Base nitrogenada timina
U	Base nitrogenada uracilo

Índice

Capítulo 1

Introducción

1.1. ¿Qué son las ómicas?

La célula es la unidad básica morfológica y funcional de vida del organismo, cuyas tres principales componentes son la membrana celular, el citoplasma y el núcleo. Este último es quien alberga el material genético: los cromosomas (que en conjunto forman el genoma) en las células eucariotas (animal). En cambio, en las células procariotas (bacterianas) el material genético está disperso en el citoplasma (capítulo 1 en [18]).

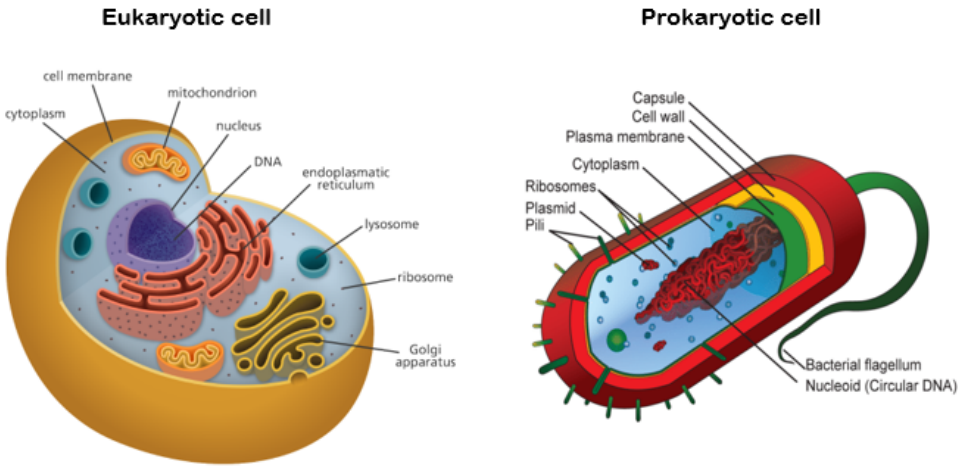

Figura 1.1: Célula procariota y eucariota.

Los cromosomas están formados por un ácido nucleico que contiene la información genética, el ADN (ácido desoxirribonucleico). El ADN posee una estructura de doble hélice, siendo cada cadena una secuencia de nucleótidos compuesta por las siguientes bases nitrogenadas: adenina (A), guanina (G), citosina (C) y timina (T), enfrentándose siempre en la doble cadena A con T y C con G, siendo las bases enfrentadas bases complementarias (véase la Figura 1.2). A lo largo de las cadenas de ADN se encuentran los genes, unidades físicas y funcionales que contienen la información sobre cómo deben funcionar las células del organismo.

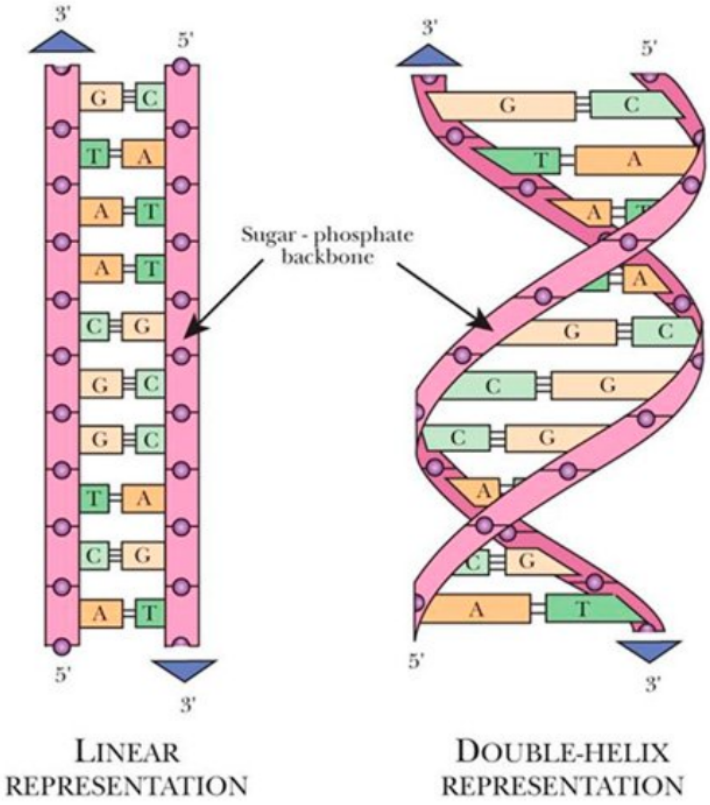

Figura 1.2: Estructura del ADN.

Sin embargo, las reacciones bioquímicas que tienen lugar en la célula (metabolismo) que permiten al organismo crecer, reproducirse, responder a cambios en diferentes entornos... son catalizadas por proteínas, llamadas enzimas. Las proteínas son grandes moléculas formadas por cadenas lineales de aminoácidos que son determinadas mayoritariamente por la genética del propio organismo. Los genes que codifican proteínas se transcriben dando lugar a los tránscritos (moléculas de ARN), que salen del núcleo y son traducidos a proteínas, que son sintetizadas por los ribosomas localizados en el citoplasma. Por otro lado, los productos intermedios y finales de las reacciones bioquímicas (metabolómicas) son los llamados metabolitos.

La necesidad de comprender el comportamiento celular a nivel molecular ha dado lugar a nuevas disciplinas de investigación conocidas como ómicas. Hasta finales del siglo pasado, se estudiaban unos pocos genes, proteínas... en un mismo experimento. La aparición de las tecnologías de alto rendimiento ha permitido estudiar en un mismo ensayo todas o una gran parte de las moléculas de interés presentes en la célula, dando lugar a los datos ómicos. Existen muchas disciplinas ómicas diferentes, algunos ejemplos son:

i) **Genómica**. Es la ciencia que estudia la composición genética de los organismos, a través del genoma (conjunto completo de todos los genes), que no debe

confundirse con la genética que estudia los genes individualmente. Tiene muchas aplicaciones, como el estudio de enfermedades genéticas o desarrollo de fármacos.

ii) **Transcriptómica**. Ver sección 1.1.1.

iii) **Proteómica**. El conjunto completo de proteínas en la célula se denomina proteoma y, consecuentemente, el estudio de la estructura y función de las proteínas se conoce como proteómica. La proteómica estudia la estructura de las proteínas, las interacciones entre estas o la identificación de aquellas proteínas cuya presencia, ausencia o alteración se relaciona con determinados estadíos fisiológicos.

iv) **Metabolómica**. Su objetivo es detectar, cuantificar y dilucidar la estructura de los metabolitos que, en conjunto, representan el metaboloma de una célula, tejido, órgano u organismo.

Estas recientes tecnologías han permitido diferentes avances en campos como la biotecnología, la ecología o medicina, ya que permiten describir los genomas de una gran variedad de organismos, estudiar los cambios en la expresión (o transcripción) de los genes por efectos del entorno celular (véase sección 1.1.1), analizar la inducción de un nuevo fármaco, etc. Por lo que cada ómica busca proporcionar información específica sobre los sistemas biológicos, los elementos que conforman la célula y la relación entre el genotipo y el fenotipo. El genotipo es la información genética de un individuo, mientras que el fenotipo son las características visibles de dicho individuo, que resultan de la interacción entre su genotipo y el medio.

Sin embargo, las disciplinas ómicas han supuesto también un gran reto, ya que han generado conjuntos de datos masivos y el científico tiene que tratar con ellos, asumiendo el desafío del manejo, procesamiento y análisis de los datos ómicos.

1.1.1. Transcriptómica

De todas las ómicas, aquella que tendrá un total protagonismo en este trabajo es la transcriptómica, por lo que en esta sección se describe en detalle.

Como se ha dicho anteriormente, las proteínas son las responsables de catalizar los procesos bioquímicos que tienen lugar en las células, por lo que el objetivo primordial es la síntesis de proteínas, es decir, el proceso mediante el cual tiene lugar la formación de las proteínas a través del código genético almacenado en el ADN. La síntesis de proteínas consta de dos procesos: la transcripción y la traducción.

La transcripción surge del hecho de que el material genético se encuentra en el núcleo y los ribosomas que son los encargados de la síntesis de proteínas se encuentran

en el citoplasma. En este primer proceso los genes son transcritos o copiados a otra molécula, el ARN mensajero (ARNm) (capítulo 8 en [31]). El ARN (ácido ribonucleico) tiene una estructura similar a la del ADN: consta de una sola cadena en lugar de la doble hélice y la base nitrogenada timina es reemplazada por el uracilo (U). En particular, el ARNm es el ácido ribonucleico que transfiere el código genético procedente del ADN del núcleo celular a un ribosoma que se encuentra en el citoplasma. Las moléculas de ARN que se transcriben a partir de los genes se llaman tránscritos, y la expresión de un gen determinado se define como la cantidad de tránscritos producidos a partir de él.

Una vez el ARNm ha sido transportado a los ribosomas, tiene lugar el proceso de traducción: dicho ARNm se traduce a una secuencia determinada de aminoácidos que constituyen una proteína. Por lo tanto, el código genético permite traducir la información escrita en el lenguaje de los ácidos nucleicos (nucleótidos) al lenguaje de las proteínas (aminoácidos), siendo válido para cualquier ser vivo.

Así pues, mientras que la información contenida en los genes es estática, la expresión génica varía con el tiempo, según las condiciones ambientales, la etapa de desarrollo del organismo, etc. Por todo esto, la transcriptómica analiza esencialmente la comparación de perfiles de expresión génica entre distintos genotipos (muestras biológicas, grupos de individuos o condiciones experimentales) para analizar y comprender los distintos procesos biológicos. Los genes que muestren un cambio en su expresión entre condiciones diferentes serán los más relevantes para caracterizar los distintos genotipos y se dirá que son genes diferencialmente expresados. La selección de los genes diferencialmente expresados se llama análisis de expresión diferencial. Estos genes serán los candidatos a ser biomarcadores o dianas terapéuticas, por ejemplo.

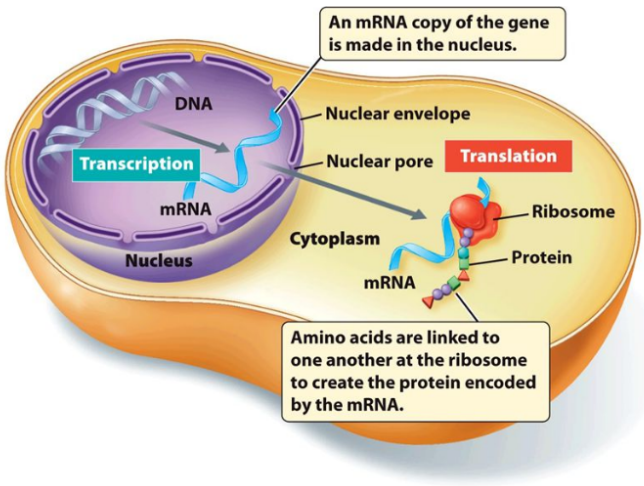

Figura 1.3: Procesos de transcripción y traducción.

1.1.2. ¿Cómo se mide la expresión génica?

La medida de la expresión génica es importante, ya que así es posible cuantificar el nivel de activación de un gen determinado en la célula. El estudio de la expresión génica permite, por ejemplo, identificar una infección vírica de la célula, si un paciente es vulnerable a padecer una enfermedad o la inducción de un factor de transcripción (proteínas reguladoras cuya función es controlar qué genes se activan o desactivan en el genoma).

Existen diversos métodos para medir el nivel de expresión génica. En particular, para medir el nivel de expresión de miles de genes (o incluso todo el genoma), es imprescindible el uso de técnicas de alto rendimiento, que incluyen la tecnología de microarrays y la secuenciación de nueva generación (*Next Generation Sequencing*, NGS [21], [37]) como RNA-seq, entendiéndose por secuenciación la determinación del orden de los nucleótidos.

i) **Microarrays**. Un microarray es un soporte sólido (portaobjetos) que contiene secuencias de ADN correspondientes a diversos genes (miles), llamadas sondas. El funcionamiento de los microarrays se basa fundamentalmente en que las secuencias complementarias de ácidos nucleicos tienden a emparejarse entre sí. Por lo tanto, las muestras de ARN se tratan para transformarlas en ADN complementario (ADNc), que es una hebra de ADN de doble cadena que constituye una secuencia totalmente complementaria del ARN que se está tratando. A continuación, estas muestras tratadas se inyectan en el microarray para que entren en contacto con las sondas y se unan para medir mediante fluorescencia el nivel de expresión. Los valores de expresión génica medidos con esta técnica son intensidades de fluorescencia y, por tanto, de naturaleza continua (Figura 1.4).

ii) **RNA-seq**. Es una forma alternativa para medir el nivel de expresión sin la necesidad de tener conocimiento previo de los genes de un organismo, como sucedía en los microarrays. Aquí, también se toma la muestra de ARN y se transforma en ADNc para secuenciarlo en plataformas de alto rendimiento como, por ejemplo, Illumina [44]. La secuenciación del ADNc da lugar a millones de lecturas, que después son asignadas a cada gen según su secuencia. Por lo tanto, los valores de expresión génica obtenidos mediante la tecnología RNA-seq son el número de lecturas asociadas a cada gen, es decir, de naturaleza discreta (Figura 1.5).

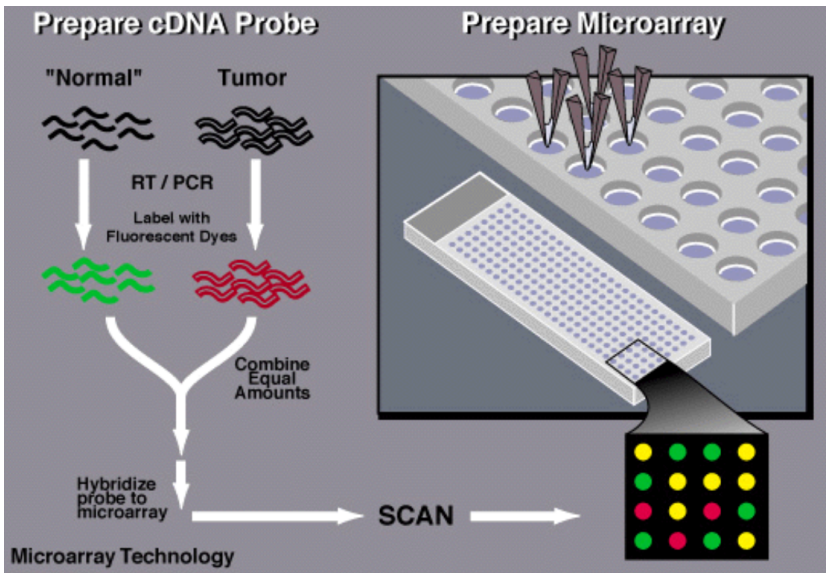

Figura 1.4: Medida de la expresión génica a través de microarrays.

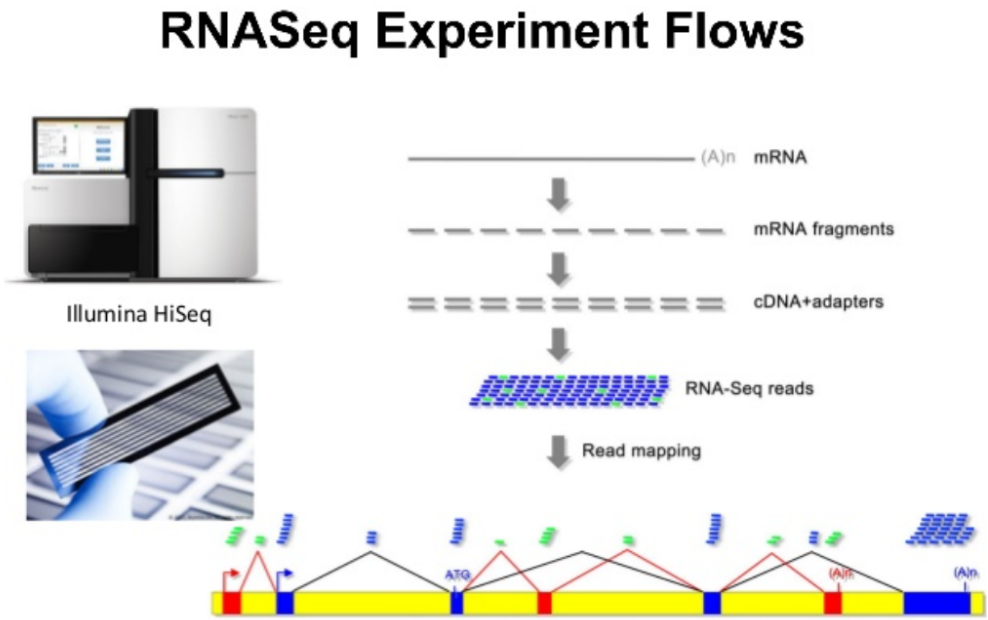

Figura 1.5: Medida de la expresión génica con RNA-seq.

1.1.3. Problemática en los datos ómicos

En el análisis estadístico de datos ómicos se presentan una serie de problemas o retos a los que el científico debe enfrentarse. Un problema común es el que se conoce como "maldición de la dimensión", que se refiere al número elevado de variables (por ejemplo, genes) que deben analizarse frente a un reducido tamaño muestral (muestras biológicas). El reducido tamaño muestral puede deberse al elevado coste de las pruebas o a la carencia de individuos que padezcan una determinada enfermedad, entre otras. Esto suele conllevar una falta de potencia estadística, que puede conducir a resultados

erróneos.

Por otro lado, la presencia de ruido a causa de la propia técnica de medición es otro problema a tener en cuenta. Las posibles consecuencias son la presencia de medidas poco fiables y que variables u observaciones no sean comparables entre sí. Estos problemas pueden solucionarse, o al menos paliarse, a través de un preprocesado de los datos que recibe el nombre de normalización.

Además, también pueden haber sesgos en las mediciones, en el sentido de que las muestras biológicas hayan sido procesadas por diferentes personas, laboratorios o en distintos momentos. Este sesgo se conoce como efecto por lotes y puede llegar a ser importante, ya que estos efectos pueden enmascarar un factor de interés o llevar a conclusiones erróneas.

Por último, como los datos pueden ser de diferente naturaleza (continua o discreta), estadísticamente tendrán que abordarse de formas diferentes, ya que tendremos distintas distribuciones de probabilidad y los modelos clásicos tendrán que ser revisados para abordar este problema. En el caso de datos discretos, la transformación de estos a continuos no es una tarea sencilla y más sabiendo que tenemos un reducido tamaño muestral. La distribución de probabilidad asociada a datos de conteos usualmente es la de Poisson. Sin embargo, esta distribución presenta el problema de no caracterizar correctamente la variabilidad biológica, ya que en una distribución de Poisson se tiene que la media y la varianza son iguales. Por lo tanto, surge la necesidad de acudir a una familia más general para modelizar la sobre-dispersión observada en los datos de conteos procedentes de técnicas de secuenciación: la Binomial Negativa, en la que la distribución de Poisson es un caso particular.

1.2. ¿Qué es la bioinformática?

En el ámbito de los datos ómicos, se pueden abordar diferentes tipos de análisis, entre otros, el estudio de la expresión génica, la determinación de las estructuras proteicas codificadas por los genes o las interacciones entre distintos tipos de moléculas. Para ello, el uso de ordenadores resulta ser indispensable para poder tratar con grandes cantidades de datos y, por lo tanto, nace el concepto de bioinformática.

La bioinformática es un campo multidisciplinar, ya que combina la biología, la informática, la matemática y la estadística, entre otras, para modelar procesos celulares a partir del análisis de datos ómicos generados mediante técnicas de alto rendimiento. Por ello, los principales objetivos de la bioinformática son:

i) Organizar los datos de forma que pueda accederse a la información existente, realizar el proceso de normalización de los datos o corregir el efecto por lotes.

ii) Desarrollar herramientas y métodos para el análisis de los datos ómicos.

iii) Aplicación de estas herramientas con el fin de analizar e interpretar los resultados de una manera significativamente biológica, en el sentido de que los resultados se correspondan con el conocimiento biológico que poseemos.

La bioinformática es un campo que está en continuo crecimiento y cada vez más son los estudios e investigaciones que se llevan a cabo dentro de esta disciplina. De hecho, entre los años 1980 y 2000 es cuando la bioinformática ha experimentado su mayor despunte [20].

Tradicionalmente, los sistemas biológicos han sido estudiados de forma individual, es decir, para cada gen, proteína..., mientras que la bioinformática va más allá: los análisis son globales (estudio de las ómicas) cuyo propósito es descubrir patrones comunes que se aplican a diversos sistemas.

1.3. Integración de datos multi-ómicos

Como hemos visto anteriormente, el análisis individual de cada ómica nos aporta información sobre aspectos diferentes del funcionamiento de un sistema biológico. Sin embargo, medir distintas ómicas en el mismo sistema nos permite obtener una visión más global de los mecanismos celulares que dan lugar a un fenotipo concreto. En el caso de la transcriptómica, en concreto, es muy interesante conocer qué elementos intervienen en la regulación de la expresión génica en un momento o condición experimental determinados. Existen distintos tipos de regulación de la expresión de un gen. Algunos ejemplos son:

- Los microRNAs (miRNAs), que son genes cortos que dan lugar a tránscritos cuya misión es degradar el ARNm y por tanto inhibir la expresión del gen. La expresión de los miRNAs se puede medir también con microarrays o protocolos de secuenciación específicos (small RNA-seq).

- Los factores de transcripción (TFs), que son proteínas que se pegan a la región promotora del gen y pueden activar o inhibir su expresión. Se puede medir la expresión de los genes que dan lugar a estas proteínas de la misma forma que se mide la expresión de cualquier otro gen, por ejemplo, mediante microarrays o RNA-seq.

- La accesibilidad de la cromatina. La cromatina es la forma en la que se dispone el ADN en el núcleo de la célula, que está totalmente compactado. Para que la maquinaria de transcripción pueda actuar en un gen determinado, la cromatina

tiene que "desplegarse", es decir, estar accesible. La accesibilidad de la cromatina se puede medir mediante técnicas de secuenciación como DNase-seq o ATAC-seq.

- La metilación del ADN, que es uno de los mecanismos epigenéticos más importantes. Es fundamental en la regulación del silenciamiento génico, ya que puede provocar alteraciones en la transcripción sin necesidad de que se produzca una alteración en la secuencia del ADN. De nuevo, la metilación se puede medir mediante técnicas de microarrays o de secuenciación masiva.

Por tanto, medir varias de estas ómicas en el mismo sistema junto con los datos de expresión génica nos ayudará a entender los procesos que hacen que los genes se activen o inhiban bajo ciertas condiciones (enfermedades, etapas de desarrollo, etc.). No obstante, para sacar partido de toda esta información multi-ómica, es clave disponer de métodos estadísticos que permitan integrar todas las ómicas en el mismo modelo y, al tiempo, den solución a los distintos problemas que, como vimos en la sección anterior, plantean este tipo de datos.

En la integración multi-ómica, como en cualquier análisis estadístico, es importante que todas las ómicas y variables adicionales que quieran añadirse al análisis, se midan en los mismos individuos (muestras biológicas, pacientes...), o al menos sean de alguna forma equiparables. Además, debe realizarse un diseño del experimento adecuado a los objetivos o cuestiones que queremos abordar, debe tenerse en cuenta el tamaño muestral para obtener una buena potencia estadística y no obtener resultados erróneos, así como el tipo de ómica, o variables adicionales a considerar en el análisis, entre otras. Una vez establecido todo lo anterior, se podrán determinar las herramientas estadísticas adecuadas para realizar la integración e interpretación de los resultados.

Encontrar métodos adecuados para el análisis estadístico de datos multi-ómicos sigue siendo un reto dentro de la bioinformática, especialmente si buscamos herramientas que estén públicamente disponibles para toda la comunidad científica, y fáciles de utilizar para investigadores que no sean expertos en estadística o lenguajes de programación. Aunque se han diseñado o adaptado diferentes metodologías estadísticas [16], que incluyen tanto métodos de aprendizaje no supervisado (análisis en componentes principales, clustering...) como supervisado (modelos de regresión lineal, regresión por mínimos cuadrados parciales, análisis discriminante...), sigue existiendo una carencia de herramientas adecuadas.

En el presente trabajo, nos centraremos en la integración de datos multi-ómicos para el estudio de la regulación de la expresión génica. Para ello, se utilizará una estrategia basada en los modelos lineales generalizados (GLM), que son una extensión de los modelos de regresión lineal múltiple que admiten más distribuciones de probabilidad

para la variable respuesta además de la distribución normal y, por tanto, proporcionan un marco más flexible para la integración de datos ómicos. En concreto, partiremos de la primera versión del método MORE (*Multi-Omics REgulation*) diseñada por el grupo de Genómica de la Expresión Génica del Centro de Investigación Príncipe Felipe, donde se ha desarrollado este trabajo. Los modelos MORE toman la expresión génica como variable respuesta y los posibles reguladores y condiciones experimentales como variables explicativas. MORE está programado en R y adaptado para ser utilizado en datos ómicos. En la nueva versión de MORE que se propone en este proyecto, no solo se amplían sus funcionalidades, como se verá más adelante, sino que se crea un paquete de R y una versión web del mismo con R Shiny para que pueda ser usado de forma fácil por la comunidad científica.

Capítulo 2

Objetivos

Los objetivos de este trabajo se fijaron a partir de una primera versión del método MORE (*Multi–Omics REgulation*) que había sido desarrollada por el grupo de Genómica de la Expresión Génica del Centro de Investigación Príncipe Felipe. En concreto, y con la idea de mejorar las funcionalidades y aplicabilidad de dicha herramienta, se establecieron los tres objetivos principales siguientes:

1. Introducir nuevas funcionalidades en el método MORE:

    - Desarrollo y corrección de funciones para estudiar ómicas reguladoras de naturaleza categórica con valores binarios (0 ó 1): filtro de baja variación de reguladores, estudio de correlación para evitar multicolinealidad y adaptación de otras funciones de MORE para que acepten y procesen correctamente este tipo de variables binarias.
    - Desarrollo de una función que extraiga una tabla resumen de los resultados de los modelos de regresión (genes, reguladores, ómicas, coeficientes de regresión...), así como la construcción de una función auxiliar que permita realizar contrastes de hipótesis sobre combinaciones lineales de los coeficientes de regresión.
    - Corrección del código de R ante los nuevos cambios planteados en MORE.

2. Desarrollo de un paquete en R: conversión de los scripts del algoritmo MORE en un paquete para que pueda ser utilizado por la comunidad científica. Esto conlleva el desarrollo y construcción del paquete, así como la redacción de la documentación de ayuda relativa a las funcionalidades principales que constituyen el paquete MORE.

3. Construcción de una aplicación web del método MORE a través de R Shiny para usuarios poco familiarizados con lenguajes de programación, es decir, evitar que

un usuario tenga que lidiar con código y manejo de funciones. Esto implica la programación en R del estilo e interfaz de usuario de la aplicación, así como la ejecución interna del propio algoritmo.

Capítulo 3

Metodología

3.1. Software R

En estadística y en particular en bioinformática, R es uno de los lenguajes de programación más usados. R es un entorno y lenguaje de programación para el cálculo, análisis de datos y gráficos. Entre otras características, R dispone de una amplia e integrada colección de herramientas para el análisis estadístico de datos, almacenamiento y manipulación efectiva de los datos, así como gráficos que incrementan potencialmente la interpretabilidad de los resultados. En [41] puede encontrarse una introducción del lenguaje R.

Existen diversos entornos de desarrollo para la programación en R, pero sin duda el más utilizado es RStudio por su potencia e interfaz intuitiva (véase Figura 3.1). RStudio es un software libre y la comunidad que aporta información y soporte es muy numerosa, lo que le hacen ser un software especialmente atractivo.

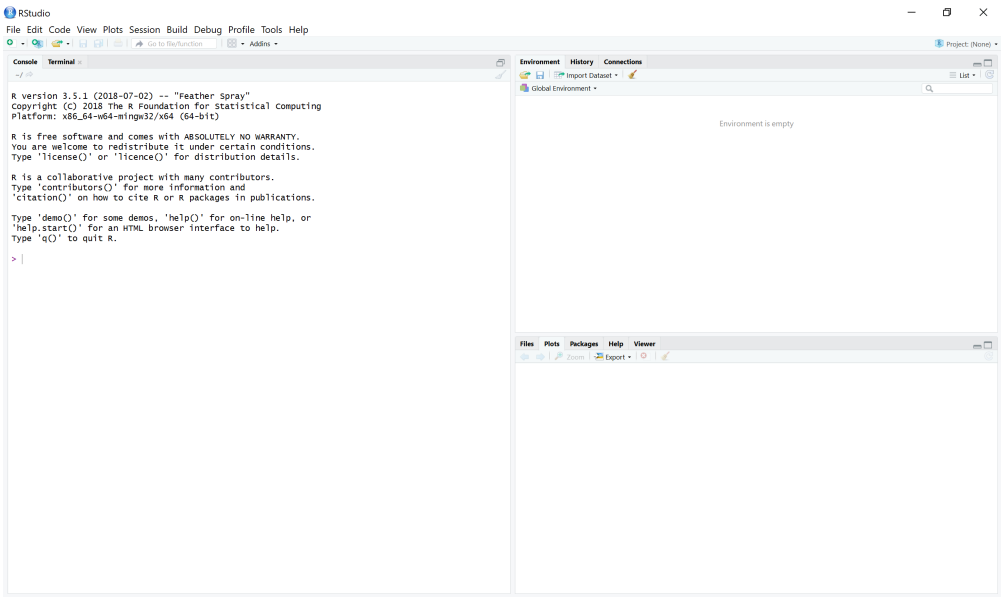

Figura 3.1: Interfaz RStudio.

Muchas de las técnicas estadísticas usadas para el análisis de datos se encuentran en el entorno base de R, mientras que otras se encuentran en las librerías (*packages*) que aporta la comunidad científica. CRAN (https://cran.r-project.org/) es el principal repositorio de librerías de R. Sin embargo, existen otros repositorios más específicos para determinadas disciplinas, como es el caso de Bioconductor (https://www.bioconductor.org/), que aglutina las librerías más utilizadas en el campo de la bioinformática. Bioconductor es un repositorio libre que proporciona a los usuarios herramientas estadísticas para el análisis e interpretación de datos ómicos [13]. Otros repositorios genéricos como *GitHub* o *Bitbucket* también suelen utilizarse para almacenar públicamente librerías de R.

3.2. Modelos Lineales Generalizados

Supongamos que se tiene un vector de observaciones, $\boldsymbol{y} \in \mathbb{R}^n$, una realización de la variable aleatoria Y. Además, sean los valores de p variables explicativas, $X_1, X_2, \ldots, X_p$, para las n observaciones consideradas. Denotando por x_{ij} el valor del individuo i-ésimo en la variable X_j, siendo $i = 1, 2, \ldots, n$ y $j = 1, 2, \ldots, p$, se tiene que los datos se pueden expresar en forma matricial, $\boldsymbol{X}_{n\times p}$.

$$\mathbf{X}_{n\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = [\mathbf{X}_1 \mid \mathbf{X}_2 \mid \ldots \mid \mathbf{X}_p] \,.$$

Supongamos que las observaciones de la variable aleatoria Y son independientes, siguen una distribución normal, son homogéneas y los valores medios de la variable respuesta dependen linealmente del valor de las variables explicativas, es decir,

- $y_i \sim N(\mu_i, \sigma^2)$, siendo $E[y_i] = \mu_i$, los valores y_i independientes y homogéneos, siendo esto último $\text{Var}[y_i] = \sigma^2$, $\forall i = 1, 2, \ldots, n$.

- La relación lineal es una especificación para la media en términos de un número de parámetros desconocidos, $\beta_1, \beta_2, \ldots, \beta_p$, que son estimados a partir de los datos disponibles a través del método de mínimos cuadrados o el método de máxima verosimilitud, que en este caso particular, son equivalentes al seguir los datos de la variable respuesta una distribución normal.

  $$E[y_i] = \mu_i = \boldsymbol{x}_i^T \boldsymbol{\beta}$$

  donde $\boldsymbol{x}_i^T$ representa la i-ésima fila de la matriz $\mathbf{X}$. Alternativamente, también

puede expresarse en forma matricial,

$$\boldsymbol{\mu} = E[Y] = \boldsymbol{X}\boldsymbol{\beta}$$

o considerando la parte especifica y una parte aleatoria que se asume que sigue una distribución normal n-variante de media $\mathbf{0}$ con matriz de varianzas–covarianzas $\sigma_\epsilon^2 \boldsymbol{I}$, siendo $\mathbf{I}$ la matriz identidad de orden n.

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Con todo esto, se tiene un modelo de regresión lineal múltiple (MLR, de su acrónimo en inglés) para el estudio de la relación entre la variable respuesta y las variables explicativas. Sin embargo, puede ocurrir que tengamos una variable respuesta de naturaleza discreta, por lo que la distribución de probabilidad ya no sería la normal, o que la relación entre la variable respuesta y las variables explicativas no sea necesariamente lineal. Por lo tanto, esto ha llevado a la revisión de los modelos clásicos para poder estudiar situaciones más generales, lo que ha dado lugar a una extensión de estos: los llamados modelos lineales generalizados (GLM, de su acrónimo en inglés).

La transición de los modelos clásicos a los modelos GLM puede definirse en dos pasos:

1. Muchas de las propiedades de la distribución normal son compartidas por una clase más amplia de distribuciones que se conocen como la familia exponencial de distribuciones.

2. A la extensión de los métodos numéricos para estimar los parámetros desconocidos, $\beta_1, \beta_2, \ldots, \beta_p$, en el sentido de que existe una función monótona diferenciable de manera que relacione μ_i y $\eta_i = \boldsymbol{x}_i^T\boldsymbol{\beta}$. A η_i se le conoce como predictor lineal y corresponde a la parte específica del modelo.

3.2.1. Función *link*

La función que relaciona el predictor lineal, η_i, y el valor esperado de y_i, μ_i, siendo $i = 1, 2, \ldots, n$, comúnmente recibe el nombre de función *link* o función enlace,

$$g(\mu_i) = \boldsymbol{x}_i^T\boldsymbol{\beta} = \sum_{j=1}^{p} x_{ij}\beta_j = \eta_i$$

donde la función $g(\cdot)$ es una función matemática monótona diferenciable. Por consiguiente, la condición que presentan los modelos GLM es que el valor esperado de cada observación puede ser expresado como una función de su predictor lineal. Esta

función es la función inversa de la función $link$, y en la literatura se le suele denominar g^{-1} [27].

$$E[y_i] = \mu_i = g^{-1}(\eta_i)$$

Los modelos lineales clásicos son un caso particular de los modelos lineales generalizados, en los que el valor esperado y el predictor lineal son iguales y, entonces, la función *link* es la identidad. Sin embargo, cuando tenemos datos relativos a conteos, la distribución conveniente es una Poisson o la distribución Binomial Negativa (donde la Poisson es un caso particular) y la identidad deja de ser la función *link* adecuada, ya que el valor esperado será positivo mientras que el predictor lineal puede tomar valores negativos. En los capítulos 2 y 3 de [23] y [10] respectivamente, pueden encontrarse las funciones *link* más utilizadas.

3.2.2. Familia exponencial de distribuciones

Sea Y una variable aleatoria cuya distribución de probabilidad depende de un único parámetro θ. Se dice que su distribución pertenece a la familia exponencial si puede escribirse como

$$f(y;\theta) = s(y)t(\theta)\mathrm{e}^{a(y)b(\theta)}$$

o, equivalentemente, puede reescribirse como

$$f(y;\theta) = \exp\{a(y)b(\theta) + c(\theta) + d(y)\}$$

siendo $s(y) = \mathrm{e}^{d(y)}$, $t(\theta) = \mathrm{e}^{c(\theta)}$ y $a(\cdot)$, $b(\cdot)$, $c(\cdot)$, $d(\cdot)$ funciones conocidas. Si $a(y) = y$, se dice que la distribución está en la forma canónica y θ es el parámetro de interés. Sin embargo, en el caso de que hubieran más parámetros además de este último θ, estos se considerarán conocidos.

Es necesario obtener expresiones para la esperanza y varianza de $a(y)$, con el objeto de determinar completamente la distribución de la variable aleatoria considerada. En el capítulo 3 de [10] se presenta el cálculo de la esperanza y varianza para una variable aleatoria continua a través de su función de densidad, considerándose de forma análoga el cálculo para variables aleatorias discretas, solo que la integral se sustituye por el sumatorio y en este caso se hablaría de función de probabilidad. Además, se debe tener en cuenta que estos resultados son aplicables a cualquier función de probabilidad (densidad) en la que el orden del sumatorio (integral) sea intercambiable con la diferenciación. Por lo que las expresiones de la esperanza y varianza son,

$$E\left[a(y)\right] = -\frac{c\prime(\theta)}{b\prime(\theta)}$$

$$Var\left[a(y)\right] = \frac{b''(\theta)c'(\theta) - c''(\theta)b'(\theta)}{b'(\theta)^3}$$

Si el lector lo desea, en [23] (capítulo 2) se presenta otra forma alternativa del cálculo de la esperanza y varianza de una variable aleatoria continua.

3.2.3. Estimación de los parámetros

En primer lugar, vamos a abordar la estimación de los parámetros en los modelos GLM. Sea $y_1, y_2, \ldots, y_n$ una muestra aleatoria simple (m.a.s) con las propiedades descritas anteriormente para los modelos GLM. La estimación de los coeficientes de regresión se hace usualmente por el método de la máxima verosimilitud, aunque también puede hacerse por mínimos cuadrados ordinarios, mediante métodos de remuestreo... El procedimiento de estimación máximo verosímil se realiza mediante los siguientes pasos (para el cálculo completo, consultar el capítulo 4 de [10]).

i) Sea la función de verosimilitud de la m.a.s,

$$\ell(\theta \mid y_i) = \prod_{i=1}^{n} f(y_i;\theta) = \exp\left\{\sum_{i=1}^{n} a(y_i)b(\theta) + c(\theta) + d(y_i)\right\}$$

ii) Por lo tanto, su función soporte es

$$L(\theta \mid y_i) = \ln \ell(\theta \mid y_i) = \sum_{i=1}^{n} a(y_i)b(\theta) + c(\theta) + d(y_i)$$

iii) Considerando un y concreto de la m.a.s se tendría que la derivada de su función soporte respecto del parámetro θ recibe el nombre de función *score*, que es una variable aleatoria cuyas expresiones para el valor esperado y varianza pueden encontrarse en [10].

$$U = \frac{\partial L(\theta \mid y)}{\partial \theta} = \frac{\partial}{\partial \theta}\left[a(y)b(\theta) + c(\theta) + d(y)\right] = a(y)b'(\theta) + c'(\theta)$$

iv) Se deriva la función soporte respecto de cada β_j con $j = 1, 2, \ldots, p$, esto es $U_j = \frac{\partial L(\theta|y_i)}{\partial \beta_j}$.

v) Por último, debe resolverse la condición de optimalidad, es decir, $U_j = 0$ para cada $j = 1, 2, \ldots, p$. Para ello, se hace uso del método numérico de Newton-Raphson (véase [49]) y de la matriz de información de Fisher, $\Im_{jk} = E[U_j U_k]$, cuya solución $\hat{\boldsymbol{\beta}}$ puede encontrarse en [10].

3.2.4. Bondad de ajuste de los modelos GLM

Antes de definir una medida de bondad de ajuste para los modelos GLM, debemos conocer qué es el método de la razón de verosimilitudes. En general, sea $\theta \in \Omega$ un parámetro p-dimensional y Ω el espacio muestral que contiene todas las hipótesis posibles. El contraste que se quiere realizar en términos generales es

$$\begin{cases} H_0 : \theta \in \Omega_0 \\ \\ H_1 : \theta \in \Omega - \Omega_0 \end{cases}$$

siendo Ω_0 un subconjunto de posibles valores del espacio total Ω.

El rechazo de la hipótesis nula dependerá de la razón de verosimilitud

$$RV = \frac{\ell(H_0)}{\ell(H_1)} \leq 1$$

que será rechazada si la razón es lo suficientemente pequeña. Además, cuando el número de observaciones consideradas es suficientemente grande, se tiene que

$$-2 \ln RV = -2 \left[\ln \ell(H_0) - \ln \ell(H_1) \right] = 2(L(H_1) - L(H_0)) \sim \chi^2_r$$

donde r es el número de restricciones impuestas en la hipótesis nula.

Ahora estamos en disposición de ver una forma de evaluar la bondad de ajuste de un modelo lineal generalizado: la comparación de nuestro modelo con un modelo con todas las variables que pueda admitir (modelo saturado o completo), considerando que el modelo saturado posee la misma distribución y función *link* del modelo de interés. Por lo tanto, el modelo saturado debe ser aquel que cumpla que $\mu_i = y_i$, con $i = 1, 2, \ldots, N$, o equivalentemente, no tenemos parte residual.

Consideremos que disponemos de un total de N observaciones de una variable aleatoria Y con distribución conocida. En general, sea m el número de parámetros máximo que pueden estimarse en el modelo completo. Sea $\boldsymbol{\beta}_{\text{máx}}$ el vector de coeficientes de regresión para el modelo saturado (completo) y $\hat{\boldsymbol{\beta}}_{\text{máx}}$ su vector de estimadores máximo verosímiles.

La función de verosimilitud para el modelo saturado evaluada en $\hat{\boldsymbol{\beta}}_{\text{máx}}$, $\ell(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y})$, será mayor que cualquier otra función de verosimilitud para estas observaciones, ya que proporciona la mejor descripción de los datos. Respecto al modelo de interés, sea $\ell(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y})$ su función máximo verosímil. Por lo tanto, la razón de verosimilitud

$$\lambda = \frac{\ell(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y})}{\ell(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y})}$$

proporciona una forma de evaluar la bondad de auste del modelo. Valores elevados indicarán que el modelo de interés no se ajusta correctamente a los datos, por lo que su

bondad de ajuste no será buena. Sin embargo, en la práctica suele usarse el logaritmo de la razón de verosimilitud

$$\ln \lambda = \ln \frac{\ell(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y})}{\ell(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y})} = \ln \ell(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y}) - \ln \ell(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y}) = L(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y}) - L(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y})$$

siendo la devianza el estadístico definido por

$$D = 2\left[L(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y}) - L(\hat{\boldsymbol{\beta}} \mid \boldsymbol{y})\right] \sim \chi^2_{m-p}$$

donde m es el número máximo de parámetros a estimar y p el número de parámetros de interés que queremos estimar. La demostración de que la devianza se distribuye asintóticamente como una chi cuadrado con $m-p$ grados de libertad puede encontrarse en el capítulo 5 de [10] y en [47] puede consultarse una prueba más rigurosa.

Por último, cabe destacar que el cociente entre la devianza del modelo restringido y la devianza del modelo completo, sería el porcentaje de variabilidad explicada por el modelo, que es el equivalente al coeficiente de determinación R^2 de los modelos de regresión lineal.

3.2.5. Contrastes de hipótesis sobre los coeficientes de regresión

Sea una variable aleatoria Y con distribución conocida e $y \in \mathbb{R}$ un valor observado o realización de esta. Vamos a considerar el caso particular en el que se tienen las variables explicativas $X_1, X_2, \ldots, X_p$ y el factor *Grupo* a dos niveles, siendo la generalización a más factores y más niveles análoga. En el contexto de este trabajo, la variable respuesta y será la expresión génica, mientras que las variables explicativas serán los potenciales elementos reguladores de dicha expresión génica, y el factor *Grupo* representará los distintos grupos experimentales que se puedan considerar, por ejemplo, pacientes control y pacientes enfermos.

Consideraremos que la relación de la variable respuesta con las variables explicativas y el factor *Grupo* viene dado por el modelo de la Ecuación 3.1, es decir, solo consideraremos interacciones dobles entre *Grupo* y el resto de variables explicativas, además de los efectos simples. Al ser la variable *Grupo* un factor con k variantes (dos en este ejemplo), tendremos que definir $k-1$ variables *dummy* (una en nuestro caso), que denotaremos como G y que representará el nivel no referencial, por ejemplo, pacientes enfermos.

$$y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \beta_{p+1} X_1 \times G + \ldots + \beta_{2p} X_p \times G + \beta_{2p+1} G \tag{3.1}$$

Es importante en este tipo de estudios saber si el regulador modula significativamente la expresión génica tanto en el grupo de pacientes control como

en el de enfermos, y si el efecto es el mismo en ambos grupos. Para responder este tipo de preguntas, necesitamos realizar contrastes de hipótesis sobre los coeficientes del regulador X_i y de su interacción con G. Los contrastes sobre los coeficientes individuales vienen dados por los propios resultados del modelo de regresión. Sin embargo, es necesario también contrastar la hipótesis nula de que la suma de ambos coeficientes sea nula. El test de hipótesis considerado será entonces el siguiente, siendo $i \neq j$ e $i, j = 1, 2, \ldots, 2p+1$,

$$\begin{cases} H_0 : \beta_i + \beta_j = 0 \\ \\ H_1 : \beta_i + \beta_j \neq 0 \end{cases}$$

La resolución del contraste pasa por utilizar el estadístico de devianza definido en la sección 3.2.4. En particular, tenemos un contraste de hipótesis con una única restricción, por lo que definiendo la devianza bajo la hipótesis nula se tiene que

$$D_0 = 2\left[L(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y}) - L(\hat{\boldsymbol{\beta}}_0 \mid \boldsymbol{y})\right] \sim \chi^2_{2p+1-1} \equiv \chi^2_{2p}$$

En cambio, bajo la hipótesis alternativa se tiene que

$$D_1 = 2\left[L(\hat{\boldsymbol{\beta}}_{\text{máx}} \mid \boldsymbol{y}) - L(\hat{\boldsymbol{\beta}}_1 \mid \boldsymbol{y})\right] \sim \chi^2_{2p+1}$$

Siendo el estadístico que se usa finalmente la diferencia de ambas devianzas,

$$\triangle D = D_0 - D_1 = 2\left[L(\hat{\boldsymbol{\beta}}_1 \mid \boldsymbol{y}) - L(\hat{\boldsymbol{\beta}}_0 \mid \boldsymbol{y})\right] \sim \chi^2_{2p+1-2p} \equiv \chi^2_1$$

donde 1 indica el número de restricciones consideradas en la hipótesis nula. La hipótesis nula será rechazada cuando el p-valor sea menor que un cierto error de primera especie dado por el usuario, que usualmente se fija en 0.05.

3.3. Selección de variables en modelos de regresión

Anteriormente se han resaltado los problemas existentes en el estudio de datos ómicos (sección 1.1.3) y, en particular, se ha enunciado la problemática del elevado número de variables en comparación al número de observaciones con el que usualmente se cuenta. Cuando utilizamos los modelos de regresión para analizar datos ómicos es común, por tanto, que el número de variables explicativas exceda el número de observaciones disponibles. Esto hace que sea necesario aplicar una estrategia de selección de variables, de forma que podamos estimar los parámetros del modelo y dispongamos de una mínima potencia estadística para hacer inferencia sobre ellos. Las técnicas de selección de variables más usadas en este contexto son los llamados métodos de regularización o *shrinkage*, como Lasso, regresión Ridge o ElasticNet, y también el método Stepwise.

A continuación, se describen brevemente los métodos de selección de variables usados en este trabajo. Para mayor detalle y profundización véase [12], [24], [28], [50] o [51]. Sea el modelo de regresión $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, asumiendo que la variable respuesta es centrada (su media es nula) y siendo $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ el vector de observaciones de una variable aleatoria Y, $\boldsymbol{X}_{n\times p}$ la matriz que contiene los valores de las p variables explicativas para cada una de las observaciones, $X_1, X_2, \ldots, X_p$, $\boldsymbol{\beta}$ el vector de coeficientes de regresión y $\boldsymbol{\epsilon}$ el vector residual.

i) Lasso.

La regresión Lasso es muy utilizada en disciplinas con grandes cantidades de datos, como es nuestro caso. Lasso no es una técnica muy robusta cuando existen regresores correlacionados, ya que elegirá uno arbitrariamente. Esto sugiere que la selección de variables mediante la técnica Lasso es adecuada cuando tenemos una pequeña cantidad de coeficientes no nulos, por lo que una pequeña parte de los predictores son los que afectan de forma sustancial a la respuesta. El estimador Lasso usa la norma ℓ_1 para encontrar la solución al problema de optimización siguiente.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|_2^2 + \lambda \| \boldsymbol{\beta} \|_1$$

siendo $\| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|_2^2 = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2$ la suma de cuadrados residual del modelo, $\| \boldsymbol{\beta} \|_1 = \sum_{j=1}^{p} \mid \beta_j \mid$ es la norma ℓ_1 que permite regular el ajuste por mínimos cuadrados y reducir coeficientes Lasso al cero absoluto y $\lambda \geq 0$ que es el parámetro de ajuste.

ii) Regresión Ridge.

La regresión Ridge es utilizada cuando tenemos una gran cantidad de predictores que poseen coeficientes no nulos y provienen de una distribución normal. Concretamente, funciona bien con predictores influyentes en la respuesta pero cuyos efectos son pequeños. Además, a diferencia de la regresión Lasso, la regresión Ridge trata bien el conjunto de variables correlacionadas haciendo que sus coeficientes tiendan a cero (no se llega a alcanzar el cero absoluto). A su vez, también se diferencia de Lasso en la norma que usa: la norma euclídea, ℓ_2.

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \|_2^2 + \lambda \| \boldsymbol{\beta} \|_2^2$$

siendo ahora $\| \boldsymbol{\beta} \|_2^2 = \sum_{j=1}^{p} \beta_j^2$.

iii) ElasticNet.

Esta técnica de selección de variables es una combinación de las dos anteriores, lo que la convierte en una técnica robusta a correlaciones extremas entre los regresores. Al ser una combinación de Lasso y Ridge, usa como penalización tanto la norma ℓ_1 como ℓ_2, siendo λ_1 y λ_2 parámetros de ajuste.

$$\hat{\boldsymbol{\beta}} = \left(1 + \frac{\lambda_2}{n}\right) \left\{ \arg\min_{\boldsymbol{\beta}} \parallel \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} \parallel_2^2 + \lambda_1 \parallel \boldsymbol{\beta} \parallel_1 + \lambda_2 \parallel \boldsymbol{\beta} \parallel_2^2 \right\}$$

iv) Método Stepwise

El método Stepwise engloba una serie de procedimientos de selección de variables explicativas, basados en la inclusión o exclusión de las mismas en el modelo de una forma secuencial. Dentro del método Stepwise se tienen,

Método Forward Se conoce como selección hacia adelante. En general, se parte del modelo básico $y = \beta_0$ y se van agregando términos de acuerdo a algún criterio, esto es, añadir al modelo la variable más significativa en cada etapa hasta una cierta regla de parada.

Método Backward Se conoce como eliminación hacia atrás. Se parte del modelo completo y en cada etapa se elimina la variable menos influyente de acuerdo a algún criterio, hasta que no proceda eliminar ningún término más.

Los criterios de eliminación (agregación) de términos al modelo completo (básico) pueden ser diferentes, desde la significación de cada coeficiente hasta criterios globales como el Criterio de Información de Akaike (AIC) o el Criterio de Información de Bayes (BIC).

3.4. Multicolinealidad

Cuando se construye un modelo de regresión, una de las hipótesis que debe cumplirse es que las variables independientes tomen valores distintos no relacionados entre sí, es decir, que no exista multicolinealidad entre los regresores. El no cumplimiento de esta hipótesis puede llevar a problemas serios:

- Para el cálculo de estimaciones mínimo cuadráticas de los coeficientes de regresión esta hipótesis es imprescindible, ya que este cálculo depende de la inversa de la matriz $\boldsymbol{X^T X}$.

- En el cálculo de las estimaciones máximo verosímiles de los coeficientes de regresión, también se tiene que es una hipótesis indispensable. En [45] se muestra que el cálculo de las estimaciones de los coeficientes de regresión a través del método de Newton–Raphson requiere la inversión de la matriz de información de Fisher (véase sección 3.2.3). Si existe multicolinealidad, hará que las soluciones del sistema $U_j = 0$, con $j = 1, 2, \ldots, p$, estén mal condicionadas (son sensibles a pequeñas variaciones de los datos del sistema) y, por lo tanto, que la matriz de información sea singular.

- Las estimaciones son muy dependientes entre sí, es decir, existe una elevada correlación entre las estimaciones de los coeficientes.

- Los estimadores de los coeficientes de regresión poseen varianzas muy elevadas.

- Pequeñas variaciones en los datos provocan variaciones sustanciales en el valor de las estimaciones de los parámetros, lo que implica la inestabilidad de los parámetros.

Por lo tanto, hay que intentar paliar el problema de la multicolinealidad en las variables explicativas consideradas. Es sabido que hay diferentes formas de estudiar la existencia de multicolinealidad en los datos, siendo el cálculo de la matriz de correlaciones de las variables explicativas, el estudio del VIF (*Variance Inflation Factor)* o el índice de condicionamiento, algunas de las más usadas.

En el método MORE se ha estudiado la correlación entre pares de reguladores (variables explicativas). Se considera que dos reguladores están "correlacionados" si su correlación supera un cierto valor límite dado por el usuario. Una vez se tienen los reguladores "correlacionados", se crea una red con todos los reguladores y se conectan con una arista solo aquellos que están "correlacionados". Se extraen de esta red sus componentes conexas y las componentes conexas con más de un regulador constituirán los grupos de reguladores correlacionados. Para evitar problemas de multicolinealidad, de cada uno de estos grupos se elige al azar un representante, que será el único que se incluirá en el modelo.

En nuestro caso, al disponer de reguladores que pueden tomar valores numéricos o categóricos (binarios), el estudio de la correlación será diferente en cada caso.

3.4.1. Medidas de correlación en variables continuas

En primer lugar, y aunque es de sobra conocida, vamos a definir la medida que ha sido utilizada en este trabajo para cuantificar la relación existente entre variables de naturaleza numérica: el coeficiente de correlación lineal de Pearson.

Sean $\mathbf{x}$ e $\mathbf{y}$ dos vectores que contienen los valores reales de dos variables aleatorias X e Y. Se define la correlación de Person, $\rho(\boldsymbol{x}, \boldsymbol{y})$, como una medida de la correlación lineal entre dos variables y se calcula mediante el cociente siguiente,

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{Cov[\boldsymbol{x}, \boldsymbol{y}]}{\sigma_{\boldsymbol{x}}\sigma_{\boldsymbol{y}}}$$

En el caso de que los vectores de observaciones estén centrados, la última expresión es equivalente a

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{E[\boldsymbol{x}\boldsymbol{y}]}{\sigma_{\boldsymbol{x}}\sigma_{\boldsymbol{y}}}$$

$Cov[\boldsymbol{x}, \boldsymbol{y}]$ es la covarianza entre $\mathbf{x}$ e $\mathbf{y}$, $E[\boldsymbol{x}\boldsymbol{y}]$ es la correlación cruzada entre los vectores de observaciones $\mathbf{x}$ e $\mathbf{y}$, $\sigma_{\boldsymbol{x}}$ y $\sigma_{\boldsymbol{y}}$ son las desviaciones típicas de los vectores $\mathbf{x}$ e $\mathbf{y}$ respectivamente. Por vectores centrados se entienden aquellos que poseen media nula, consiguiéndose mediante la diferencia entre el valor de la observación y el valor medio esperado del vector considerado.

El coeficiente de Pearson puede tomar valores reales comprendidos en el intervalo [-1,1], cuya interpretación es la siguiente.

- Si $\rho(\boldsymbol{x}, \boldsymbol{y}) = 1$ se tiene una correlación total positiva entre ambas variables, esto es que tienen una relación directamente proporcional.
- Si $\rho(\boldsymbol{x}, \boldsymbol{y}) = 0$ indicaría que no existe relación lineal entre ambas variables.
- Si $\rho(\boldsymbol{x}, \boldsymbol{y}) = -1$ obtendríamos que ambas variables poseen una correlación total negativa y, por lo tanto, tendrían una relación inversamente proporcional.

Por otro lado, cabe destacar que el coeficiente de correlación lineal de Pearson no proporciona información sobre la relación de causalidad entre las variables consideradas, en el sentido de que un coeficiente de correlación elevado (en valor absoluto) no implica la causalidad entre ellas.

3.4.2. Medidas de correlación en variables categóricas

A diferencia de las variables continuas, el estudio de la medida de relación entre variables categóricas no es una tarea tan sencilla. Los valores de una variable categórica son categorías o grupos mutuamente excluyentes, siendo el número de grupos o categorías finito. Aquí, se considera un caso particular de las variables categóricas: las variables binarias. Estas variables toman valores para dos categorías posibles (diferenciar entre dos clases, ausencia o presencia de un factor considerado, etc.), por lo que pueden codificarse como una variable de valores 0 ó 1.

Por tanto, hay que considerar nuevas herramientas para cuantificar la relación entre variables categóricas. En el presente trabajo se han considerado las siguientes medidas para el estudio de la correlación de variables binarias: los coeficientes de correlación *phi*, Cramer V, Kappa de Cohen, tetracórica e índice de Jaccard.

i) Coeficiente de correlación *phi*.

Es una medida de asociación entre dos variables categóricas. Sean dos variables binarias X e Y. El coeficiente *phi*, que denotaremos por ϕ, está estrechamente relacionado con el estadístico χ^2 en una tabla de contingencia 2×2,

$$\phi^2 = \frac{\chi^2}{n} = \frac{(n_{11}n_{22} - n_{12}n_{21})^2}{n_{1\cdot}n_{2\cdot}n_{\cdot 1}n_{\cdot 2}}$$

siendo n el número total de observaciones consideradas y n_{ij} las frecuencias absolutas observadas según la siguiente tabla de contingencia.

	Y_1	Y_2	
X_1	n_{11}	n_{12}	$n_{1\cdot}$
X_2	n_{21}	n_{22}	$n_{2\cdot}$
	$n_{\cdot 1}$	$n_{\cdot 2}$	$n_{\cdot\cdot}$

Donde X_1 y X_2 son las dos categorías posibles de la variable X; Y_1 e Y_2 son las dos categorías posibles de la variable Y; $n_{1\cdot} = n_{11} + n_{12}$; $n_{2\cdot} = n_{21} + n_{22}$; $n_{\cdot 1} = n_{11} + n_{21}$; $n_{\cdot 2} = n_{12} + n_{22}$ y $n_{\cdot\cdot} = n_{11} + n_{12} + n_{21} + n_{22}$.

El coeficiente ϕ toma valores reales en el intervalo [-1,1], donde los distintos valores tienen una interpretación similar al coeficiente de correlación de Pearson.

- Si $\phi = 1$ se tiene una correlación total positiva entre ambas variables categóricas, esto sugiere que la mayoría de valores se encuentran en la diagonal principal y se tiene una relación directamente proporcional.
- Si $\phi = 0$ indicaría que no existe relación lineal entre las dos variables y los valores de ambas se encontrarían dispersos en las distintas celdas de la tabla de contingencia.
- Si $\phi = -1$ obtendríamos que ambas variables poseen una correlación total negativa y, por lo tanto, esto supondría que la mayoría de valores

se encuentran fuera de la diagonal principal de la tabla de contingencia, existiendo una relación inversa entre ambas variables.

ii) Coeficiente de correlación Cramer V.

El coeficiente Cramer V se define como una medida de asociación entre dos variables nominales (discretas), siendo las variables categóricas codificadas como 0 ó 1 (binarias) un caso particular. Consideremos dos variables nominales, X que posee r niveles e Y con k niveles, y sea también la tabla de contingencia entre ambas.

	Y_1	Y_2	$\cdots$	Y_k	
X_1	n_{11}	n_{12}	$\cdots$	n_{1k}	$n_{1\cdot}$
X_2	n_{21}	n_{22}	$\cdots$	n_{2k}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rk}	$n_{r\cdot}$
	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot k}$	$n_{\cdot\cdot}$

siendo $n_{\cdot j} = \sum_{i=1}^{r} n_{ij}$, $n_{i\cdot} = \sum_{j=1}^{k} n_{ij}$ y $n = n_{\cdot\cdot} = \sum_{j=1}^{k} n_{\cdot j} = \sum_{i=1}^{r} n_{i\cdot}$, con $i = 1, 2, \ldots, r$ y $j = 1, 2, \ldots, k$. Se define el estadístico χ^2 como

$$\chi^2 = \sum_{i,j} \frac{\left(n_{ij} - \frac{n_{i\cdot} n_{\cdot j}}{n}\right)^2}{\frac{n_{i\cdot} n_{\cdot j}}{n}}$$

El coeficiente de correlación Cramer V que denotaremos por φ_c, se basa en el estadístico χ^2 tal y como se muestra en la siguiente expresión.

$$\varphi_c = \sqrt{\frac{\frac{\chi^2}{n}}{\text{mín}\{r-1, k-1\}}} = \sqrt{\frac{\phi^2}{\text{mín}\{r-1, k-1\}}}$$

A diferencia del coeficiente de correlación de Pearson y *phi*, este coeficiente toma valores reales comprendidos en el intervalo [0,1], siendo 1 el indicador de relación total entre las variables y 0 la inexistencia de relación entre ambas. Además, es un coeficiente simétrico, en el sentido de que no importa qué variable consideremos en la fila o columna.

El estimador Cramer V es un estimador sesgado, aún con un tamaño de muestra considerablemente grande, por lo que sobreestima la fuerza de asociación. [2] muestra que el sesgo del estadístico Cramer V es $\frac{1}{n-1}(r-1)(c-1)$, por lo que un estimador insesgado es

$$\tilde{\varphi}_c = \sqrt{\frac{\tilde{\phi}_+^2}{\text{mín}\{\tilde{r}-1, \tilde{k}-1\}}}$$

siendo $\tilde{\phi}^2 = \phi^2 - \frac{1}{n-1}(r-1)(c-1)$ la corrección del sesgo, que al no poder tomar valores negativos se tiene que $\tilde{\phi}_+^2 = \text{máx}\{0, \tilde{\phi}^2\}$, $\tilde{r} = r - \frac{(r-1)^2}{n-1}$ y $\tilde{k} = k - \frac{(k-1)^2}{n-1}$.

Intuitivamente, se tiene que en el caso de tablas de contingencia 2×2 los coeficientes de correlación de Cramer V y *phi* son equivalentes, al poseer Cramer V en su expresión el cuadrado de *phi*.

iii) Coeficiente de correlación Kappa de Cohen.

El coeficiente Kappa de Cohen es una medida del grado de concordancia entre dos variables categóricas y, en particular, de las variables binarias, o alternativamente, es la proporción de desacuerdos esperados por casualidad que no ocurren, y se denota como κ (véase [6]). Para ello se definen las siguientes proporciones:

- p_0: proporción de observaciones que concuerdan.
- p_e: proporción de observaciones que concuerdan por casualidad.

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

De acuerdo a las definiciones de p_0 y p_e, se tiene que el numerador hace referencia a la proporción de casos concordantes por no casualidad y el denominador a la proporción de casos tanto concordantes por no casualidad como discordantes. Este coeficiente toma valores reales comprendidos en el intervalo [0,1], donde el índice $\kappa = 1$ indica concordancia completa entre ambas variables, mientras que $\kappa = 0$ implicaría que no hay acuerdo entre las variables distinto al que cabría esperar por casualidad.

iv) Coeficiente de correlación tetracórica.

El coeficiente de correlación tetracórica es una medida de asociación entre dos variables aleatorias categóricas que son representadas en tablas de contingencia 2×2 [11]. Este coeficiente se obtiene a través de la correlación de dos variables latentes continuas que se distribuyen normalmente, cuya categorización da lugar a las variables categóricas originales. Por variable latente se entiende variables

que no son observadas directamente, sino que son inferidas de otras variables que sí son observadas. Además, la correlación tetracórica puede generalizarse a la correlación policórica para tratar variables que contengan más de dos categorías. El coeficiente de correlación tetracórica toma valores comprendidos en el intervalo [-1,1]. En [9] puede encontrarse el algoritmo de cálculo de este coeficiente.

v) Índice de Jaccard.

El índice de Jaccard [33] se utiliza para estudiar la similaridad de dos conjuntos de datos, comparando los elementos de cada uno de los conjuntos para determinar cuáles pertenecen a ambos conjuntos y cuáles no. El índice de Jaccard toma valores reales pertenecientes al intervalo [0,1], donde valores elevados indican elevada similaridad entre los dos conjuntos considerados. Sin embargo, no es robusto cuando el tamaño muestral es pequeño o tenemos datos faltantes, por lo que conduciría a obtener resultados erróneos. Dados dos conjuntos de datos A y B, el índice de Jaccard que denotaremos por $J(A, B)$, viene dado por

$$J(A, B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid}$$

donde $\mid \cdot \mid$ denota el cardinal de un conjunto dado, $\cap$ la intersección de conjuntos y $\cup$ la unión de conjuntos.

Al no disponer de conjuntos, debe adaptarse el uso del índice de Jaccard. Al contar con variables binarias, la idea es tomar las posiciones que ocupe, por ejemplo, el valor 1 en ambos reguladores considerados (variables explicativas). El cálculo del índice de Jaccard sería

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

siendo M_{11} el número de veces en los que ambos reguladores toman a la vez el valor 1 (misma posición), M_{01} el número de veces en los que el primer regulador toma el valor 0 y el segundo 1 y M_{10} el número de veces en los que el primer regulador toma el valor 1 y el segundo 0.

3.5. Método MORE

Para entender la utilidad del método MORE (del inglés *Multi-Omics REgulation*), recordemos que la regulación génica es el proceso mediante el cual se controla qué genes se expresan, o lo que es lo mismo, darán lugar a un producto funcional como una proteína, tal y como se vio en la sección 1. Los reguladores son elementos de la

célula que regulan el proceso de transcripción y son muy diversos: pueden ser otros genes reguladores como los microRNA (miRNA, siendo RNA el acrónimo en inglés de ARN) que son genes de secuencia más corta que no han sido traducidos a proteínas; la metilación del ADN que está relacionada con la inhibición de genes; la presencia de histonas o no en el gen (proteínas básicas que junto al ADN pueden formar la cromatina que es la sustancia que permite formar un cromosoma), entre otros muchos.

El método MORE ha sido desarrollado por el Laboratorio de Genómica de la Expresión Génica en el Centro de Investigación Príncipe Felipe con el propósito principal de facilitar la aplicación de modelos GLM a datos multi-ómicos. Concretamente, el objetivo del método MORE es modelizar la regulación de una determinada entidad biológica (gen, proteína, metabolito, etc.) bajo unas determinadas condiciones experimentales. Si, por ejemplo, se trata de entender la regulación de la expresión génica, MORE genera modelos para todos y cada uno de los genes estudiados (del orden de miles) donde la variable respuesta en cada caso es la expresión de un gen determinado y las variables explicativas serían las covariables experimentales, tales como el tiempo o tratamiento, y los reguladores potenciales de cada gen. Tras ajustar cada modelo, los reguladores con un efecto significativo sobre la expresión génica serían los candidatos a estar actuando específicamente sobre el gen en el sistema biológico que se está estudiando. MORE requiere, para crear los modelos GLM, una serie de información que se presenta a continuación.

- **Matriz de expresión génica**. Esta matriz contiene en sus filas el valor de expresión para cada gen i (variables), con $i = 1, 2, \ldots, p$, bajo una serie de condiciones experimentales (las n muestras biológicas), en columnas. Aunque en estadística las matrices de datos se suelen disponer como indica la Figura 4.2, notar que en bioinformática se representan las variables en filas y observaciones en columnas, tal y como se muestra en la Figura 3.3. Así pues, conviene tener presente que MORE generará un GLM para cada una de la filas (genes) de esta matriz.

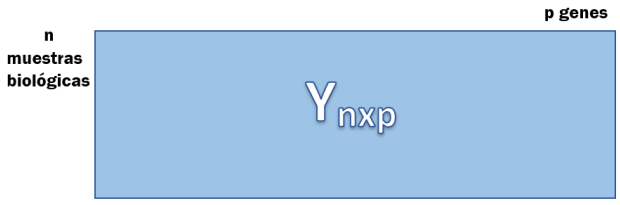

Figura 3.2: Representación usual de los datos.

- **Matriz de diseño experimental**. Tendrá tantas filas como columnas posea la matriz de expresión génica y tantas columnas como factores experimentales

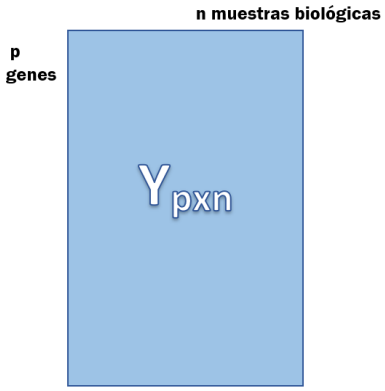

Figura 3.3: Representación de los datos en bioinformática.

consideremos (tratamiento, tipo de enfermedad...), con tal de recoger el diseño experimental considerado. Sin embargo, hay que tener en cuenta que, con tal de facilitar la interpretación de los modelos a los usuarios no estadísticos, en caso de que haya más de un factor, MORE combinará los valores de los mismos para crear un único factor.

- **Matrices de ómicas reguladoras**. Podemos tener tantas matrices como ómicas reguladoras consideremos. Cada matriz recoge los valores de los reguladores para una ómica determinada (en filas) y, de nuevo, las muestras biológicas en columnas. Los valores de los reguladores para una ómica pueden ser de tipo numérico o categórico binario (valores 0 ó 1), pero todos los reguladores de una misma ómica tendrán la misma naturaleza. La primera versión de MORE, de la que parte este trabajo, no permitía la inclusión de reguladores de tipo binario, por lo que todo lo relacionado con los reguladores de este tipo ha sido desarrollado en el marco de este TFM.

- **Matriz de asociaciones**. Tendremos tantas matrices de asociaciones como ómicas reguladoras consideradas. Cada matriz indicará las asociaciones potenciales entre genes y reguladores. Por tanto, de estas matrices podemos extraer los reguladores potenciales de un gen determinado, que conformarán junto al factor experimental, las variables explicativas que se incluirán en la ecuación inicial del modelo GLM.

Con los parámetros de entrada, MORE genera primeramente la ecuación del modelo inicial, que será diferente para cada gen, pues cada uno de ellos posee diferentes reguladores potenciales, incluyendo las correspondientes interacciones con el factor experimental, para aplicar los modelos GLM. Sin embargo, puede ocurrir que existan genes con un elevado número de reguladores potenciales en comparación al número

de muestras biológicas. Esto puede desencadenar en la imposibilidad de estimar los coeficientes de regresión y sus respectivas significaciones estadísticas, debido al reducido o inexistente número de grados de libertad. Por ello, la primera versión de MORE contaba con funcionalidades para:

i) Filtrar reguladores con datos faltantes.

ii) Filtrar reguladores (de tipo numérico) con baja variación.

iii) Filtrar reguladores con valores numéricos correlacionados, sin tener en cuenta las correlaciones negativas.

iv) Filtrar reguladores mediante los métodos de selección de variables descritos en la sección 3.3: ElasticNet, Lasso, Ridge y Stepwise.

Al estar estos filtros diseñados exclusivamente para reguladores con valores numéricos, la aplicación del método MORE estaba restringida a un determinado tipo de datos, por lo que este aspecto ha sido mejorado: se ha corregido el filtro de reguladores con baja variación, aplicándolo tanto a valores numéricos como categóricos (binarios) y se ha modificado el filtro de reguladores correlacionados, en el sentido de que ahora es posible también calcular la correlación de variables binarias y considerar correlaciones negativas. Además, inicialmente MORE centraba y escalaba (si el usuario lo deseaba) el conjunto de datos de entrada, en cambio, ahora hay que tener presente la posibilidad de contar con variables binarias, las cuales no deben centrarse ni escalarse.

Los modelos GLM finales son los obtenidos después de la aplicación de todos los filtros anteriores. Los resultados se recogen en los siguientes objetos:

◇ **ResultsPerGene**. Es una lista que contiene tantos objetos como genes tengamos y, para cada gen, se muestran los siguientes resultados:

Y Contiene los valores de expresión del gen dado (**y**), los valores ajustados por el modelo (**fitted.y**) y los valores residuales del modelo GLM (**residuals**).

X Contiene los valores de los predictores incluidos en el modelo de forma adicional.

coefficients Recoge los coeficientes estimados para los reguladores que han resultado significativos y sus significaciones (p–valores).

allRegulators Tabla resumen donde para cada gen y sus reguladores potenciales se indica la ómica a la que pertenecen estos últimos, el área, si han sido filtrados y, en ese caso, el tipo de filtro y si el regulador es significativo o no.

significantRegulators Vector que indica los reguladores significativos para el gen considerado.

◇ **GlobalSummary**. Lista que contiene información del modelo final GLM:

GoodnessOfFit Recoge p–valores, grados de libertad residuales finales, porcentaje de devianza y el criterio de información de Akaike (AIC).

ReguPerGene Información para cada ómica sobre el número de reguladores iniciales, número de reguladores incluidos en el modelo inicial y número de reguladores significativos.

GenesNOmodel Recoge aquellos genes para los que no se pudo obtener un modelo GLM final, indicando la posible razón: "*Too many missing values*", "*No predictors after EN*" y "*GLM error*", donde EN se refiere a selección de variables ElasticNet.

◇ **Arguments**. Lista con los argumentos utilizados para generar el modelo: matriz final de diseño experimental, el nivel de significación dado por el usuario, el mínimo de grados de libertad exigido por el usuario, etc.

Por último, el método MORE incluye una función adicional que permite investigar las relaciones de los genes y los reguladores de forma gráfica, haciéndolo más intuitivo y agradable. Podrá estudiarse las relaciones entre un gen y regulador dados, un gen y todos sus reguladores potenciales, un regulador y los genes a los que regula, estudiar puntos de tiempo...

Por lo tanto, las nuevas funcionalidades para la mejora del método MORE son las que se enumeran a continuación y se presentarán en el capítulo 4:

i) Corrección del centrado y escalado, esto es que ante la presencia de variables binarias, solo centre y escale las variables numéricas.

ii) Corrección de la función **LowVariatFilter**: además de filtrar reguladores (variables) numéricos por baja variabilidad, deberá filtrar también reguladores binarios.

iii) Corrección de la función **CollinearityFilter**: filtrar reguladores numéricos y binarios previa elección de un regulador representante de cada grupo correlacionado, teniendo en cuenta además el tipo de correlación del resto de reguladores del grupo con su respectivo representante.

iv) Nueva función **RegulationPerCondition**: desarrollo de una nueva función para recuperar los reguladores significativos y mostrar el efecto de cada regulador bajo cada condición experimental considerada.

v) Nueva función **BetaTest** que hará contrastes de hipótesis sobre los coeficientes de regresión con la finalidad de estudiar si el regulador modula de forma significativa la expresión génica en cada grupo. Esta función se utilizará dentro de la función definida en iv).

3.6. Datos utilizados

En esta sección se presentan los dos conjuntos de datos que se han utilizado en el desarrollo del trabajo, como ejemplo y para testar las funcionalidades desarrolladas: los datos STATegra y los datos simulados a través del paquete MOSim.

3.6.1. Datos STATegra

Los datos STATegra provienen del proyecto europeo STATegra que tiene por objetivo el desarrollo de métodos estadísticos y software para la integración de datos ómicos de secuenciación masiva (http://www.stategra.eu/).

Como parte de este proyecto, se generaron los datos STATegra [42] que son datos ómicos procedentes de un diseño temporal de un proceso de diferenciación celular en ratones, concretamente la diferenciación a células B. Las células B son un tipo de linfocito cuyo desarrollo ocurre en la médula ósea.

En particular, se estudió la línea celular B3, que está mutada de tal forma que no se produce la diferenciación espontáneamente sino que debe inducirse de algún modo. En STATegra, se indujo dicha diferenciación mediante la activación del factor de transcripción Ikaros, con lo que se establecieron dos condiciones: el grupo Control (sin diferenciación) y el grupo Ikaros, siendo en este último grupo donde ocurre la activación del factor de transcripción y, por tanto, la diferenciación.

Así pues, los datos STATegra constan de un total de 6 muestras temporales para cada condición: 0, 2, 6, 12, 18, 24 horas. Además, se replicaron 3 veces por condición y tiempo, con lo que se cuenta con un total de 36 muestras experimentales. A partir de estas muestras biológicas se obtuvieron los datos que utilizaremos en este trabajo: RNA-seq (expresión génica), miRNA-seq (expresión de miRNAs), DNase-seq (accesibilidad de la cromatina) y RRBS-seq (metilación del ADN). También utilizaremos los datos de expresión de factores de transcripción, que se extrajeron de los datos de RNA-seq. Cabe destacar que los datos habían sido pre–procesados por los

miembros del proyecto STATegra. Los datos STATegra contienen los siguientes objetos:

- ◇ **GeneExpressionDE**. Matriz de expresión génica (valores de conteo) para cada gen (5865 genes) en las 36 muestras experimentales. Notar que estos genes son una selección del total de genes de ratón realizada por el consorcio STATegra, tras filtrar aquellos genes cuya expresión no cambiaba significativamente entre condiciones ni a lo largo del tiempo.

- ◇ **data.omics**. Lista que contiene cuatro matrices de datos correspondientes a las ómicas reguladoras consideradas, donde se almacenan los datos de expresión para cada una de ellas: miRNA (106 reguladores), DNase (9846 reguladores), RRBS (metilación del ADN, 1116417 reguladores) y TF (factor de transcripción, 130 reguladores). Cabe resaltar que todos estos valores son numéricos.

- ◇ **edesign**. Matriz que contiene las covariables experimentales: tiempos y condiciones (Control o Ikaros). Notar que para este trabajo no consideraremos el tiempo. Por tanto, tendremos un único factor con dos valores: Ikaros y Control.

- ◇ **associations**. Lista que contiene cuatro matrices, proporcionando cada una de ellas las asociaciones entre reguladores y genes, que indican cuáles son los reguladores potenciales de cada gen para cada ómica reguladora.

Los datos STATegra se utilizarán en el capítulo 6 como ilustración de la aplicación del método MORE a un caso real.

3.6.2. Datos simulados

Simulamos un conjunto de datos multi–ómicos sencillo para poder incluirlo en el paquete MORE como ejemplo, y también para ilustrar el uso de la aplicación MORE con R Shiny. Para ello, utilizamos el paquete de R MOSim (https://bitbucket.org/ConesaLab/mosim).

MOSim es un paquete programado en R que ha sido desarrollado por el Laboratorio de Genómica de la Expresión Génica en el Centro de Investigación Príncipe Felipe y permite simular fácilmente expermientos multi–ómicos: datos de expresión génica, ómicas reguladoras y relaciones potenciales entre genes y reguladores. MOSim genera datos de conteos para diferentes ensayos de secuenciación con un diseño experimental flexible, incluyendo series temporales [22].

Los datos que han sido simulados mediante el paquete MOSim constan de las siguientes ómicas: RNA-seq (expresión génica), ChIP-seq (valores que indican si un TF determinado se pega o no al gen para regularlo), miRNA-seq (expresión de miRNAs) y

TF (expresión de factores de transcripción) en 8 puntos de tiempo para dos condiciones, Group1 y Group2, y para 20 genes diferencialmente expresados. Esto implica contar con un total de 16 observaciones.

Los 20 genes diferencialmente expresados se dividen en 15 que están afectados por las tres ómicas reguladoras (ChIP-seq, miRNA-seq y TF) y los 5 restantes están afectados por menos de 3 ómicas reguladoras.

Por lo tanto, los objetos que se almacenan en el fichero simulado son:

- ◇ **GeneExpressionDE**. Valores de expresión de RNA-seq (conteos) de 20 genes diferencialmente expresados para las 16 muestras experimentales.

- ◇ **data.omics**. Lista de matrices que contienen los valores de expresión para las tres ómicas reguladoras consideradas: miRNA-seq (valores de conteo, 562 reguladores), ChIP-seq (valores binarios 0 ó 1, 48 reguladores) y TF (valores de conteo, 114 reguladores).

- ◇ **associations**. Lista de matrices que contienen la información de las asociaciones entre genes y reguladores para cada ómica reguladora considerada.

- ◇ **edesign**. Matriz de diseño experimental que recoge las variables experimentales: condición (que en la matriz de diseño aparecen como Group1 y Group2 respectivamente pero a lo largo de la memoria se denotarán como A y B) y los 8 puntos de tiempo. De nuevo, los puntos de tiempo serán considerados como réplicas de cada condición, por lo que la única covariable experimental será la condición, con dos valores 0 (condición A) y 1 (condición B).

Así, este fichero de datos simulados será utilizado para ilustrar el funcionamiento del paquete MORE y la aplicación web MORE Shiny. Resaltar que la ómica reguladora ChIP-seq posee valores binarios, por lo que el lector podrá también ver las nuevas funcionalidades sobre variables de naturaleza binaria.

Capítulo 4

Nuevas funcionalidades para el método MORE

El punto de partida de este trabajo, como se ha indicado anteriormente, es una versión inicial del método MORE. Gran parte de este trabajo ha consistido en mejorar y ampliar las funcionalidades del método para hacerlo más versátil y útil. Es importante señalar en este punto que la modificación del código de R para incluir estas nuevas funcionalidades ha supuesto, no únicamente la propia programación de las nuevas funciones, sino también un profundo estudio del código inicial del que se partía para enterderlo bien antes de modificarlo, limpiarlo de funciones o argumentos obsoletos y, en ocasiones, reorganizarlo para poderlo adaptar. Además, la incorporación de nuevas funciones ha hecho necesaria en todos los casos la modificación de los objetos de salida, que resumen los resultados del algoritmo.

En las próximas secciones se describen las nuevas funcionalidades o modificaciones realizadas en MORE. Muchas de ellas han pasado por un estudio previo de los métodos estadísticos a emplear para resolver el problema con el mayor rigor estadístico posible, y los métodos utilizados han sido descritos en el capítulo anterior.

4.1. Centrado y escalado

Como se ha comentado en la sección anterior, el algoritmo MORE estaba inicialmente programado para aceptar únicamente datos de entrada de ómicas reguladoras que fueran de naturaleza numérica. Sin embargo, en algunas ómicas, se pueden generar datos de tipo categórico, con dos clases en la mayor parte de los casos. Un ejemplo son los datos de unión de factores de transcripción (TF) a genes, que se pueden cuantificar como una probabilidad o bien codificar como una variable binaria: 1 si el TF se une al gen bajo una determinada condición y 0 en caso contrario. También los datos de genómica tienen esta característica y se pueden codificar como una variable binaria: 1, si el individuo presenta una mutación en un gen determinado, y 0, si no la

presenta.

Por lo tanto, se consideró interesante que esta nueva versión del algoritmo MORE aceptara también ómicas de tipo binario y, en consecuencia, hubo que modificar algunas funciones internas que requerían de un tratamiento especial para valores binarios. Una de estas funciones a modificar fue la encargada de centrar y/o escalar los valores de las ómicas, que forma parte de la función principal **GetGLM** (véase sección 5.1.1). Este centrado o escalado se hace en MORE a petición de los usuarios (para facilitar la interpretación de los coeficientes, dar el mismo peso a todos los reguladores, etc.) o también internamente cuando se lleva a cabo la selección de variables de ElasticNet. Sin embargo, centrar o escalar reguladores binarios carece de sentido por lo que se tuvo que introducir una modificación en esta parte del código para que el algoritmo discriminara entre valores numéricos y binarios, de tal forma que centrara y/o escalara únicamente las ómicas reguladoras con valores numéricos. Para ello, se tuvo que introducir también un nuevo argumento en la función **GetGLM** de MORE (llamado `omic.type`) para que el usuario pueda indicar cuáles de las ómicas son numéricas y cuáles categóricas (binarias).

La modificación del código de R se encuentra en el Anexo B.1 y consta de un bucle que recorre el vector dado por el usuario (`omic.type`), que recoge valores 1 y/o 0, siendo 1 una ómica reguladora binaria y 0 una ómica reguladora numérica. Por lo tanto, si el valor que se recorre es 0, entonces se centrará y escalará si el usuario así lo ha indicado previamente, ya que el centrado y escalado son parámetros de entrada del método MORE (TRUE o FALSE) (véase sección 5.1.1).

4.2. Función LowVariatFilter

Esta es una función auxiliar a la que llama la función principal **GetGLM** y fue programada con el propósito de eliminar aquellos reguladores que presentaran valores con baja variabilidad, ya que aunque estos reguladores resultaran significativos en el modelo, no serían de interés en el estudio porque variarían muy poco. Al mismo tiempo, si eliminamos desde el principio estos reguladores que apenas cambian, contaremos con un número menor de predictores en el modelo y, por lo tanto, con un mayor número de grados de libertad en los residuos, con lo que aumentará la potencia estadística.

En la versión inicial del método MORE, este filtro estaba programado para reguladores con valores numéricos, pero la inclusión de reguladores binarios ha hecho que tuviera que ser revisada. El código correspondiente a esta función se puede encontrar en el Anexo B.2.

```
> LowVariatFilter(data, method, percVar, omic.type)
```

Para la ejecución de esta función se necesitan los siguientes parámetros de entrada:

- ◇ **data**. Matriz que contendrá los valores medios entre réplicas, mientras que si no tenemos réplicas, esta matriz será la matriz ómica reguladora original correspondiente.
- ◇ **percVar**. Vector que indica para cada ómica reguladora la mínima variabilidad deseable. O bien ha sido introducido por el usuario, o bien lo computa el propio algoritmo.
- ◇ **method**. Forma de filtrar reguladores numéricos.
- ◇ **omic.type**. Vector compuesto por 0 y 1, siendo 0 la indicación por parte del usuario de que la ómica correspondiente es numérica y 1 binaria. Observar que este nuevo argumento, ya descrito en el apartado anterior, fue añadido a la función **GetGLM** (véase sección 5.1.1) para permitir la inclusión de los reguladores binarios.

Por lo tanto, en este punto el algoritmo habrá calculado para cada ómica reguladora su correspondiente matriz con los valores medios de las réplicas para cada condición indicada en la matriz de diseño experimental (o la ómica reguladora original en caso de no haber réplicas). Una vez tengamos estas matrices, se tiene que discernir entre el tipo de ómica: numérica o binaria, ya que según esto, el filtro será diferente.

i) Ómica reguladora numérica.

Método sd Método aplicado en el caso en que el usuario no haya establecido un límite de mínima variabilidad (el parámetro `min.variation` de la función **GetGLM** sería NULL). Para cada una de las matrices que contienen las medias entre réplicas, se aplican los siguientes pasos:

1. Se calcula la desviación típica para cada fila de la matriz considerada, siendo las filas los reguladores potenciales.
2. Dado que el usuario no ha establecido el valor límite de mínima variabilidad, se calcula este como el 10 % de la máxima variabilidad observada, es decir, de las desviaciones típicas obtenidas en el paso 1.
3. De las desviaciones típicas calculadas en el paso 1, tomamos únicamente aquellos reguladores que presentan una desviación típica mayor que el valor calculado en el paso 2.

Método user Método aplicado cuando el usuario ha introducido el parámetro que indica el límite de mínima variabilidad deseada. Aquí, a partir de las matrices que contienen las medias entre réplicas, se hacen los siguientes pasos para cada una de ellas:

1. Se calcula para cada regulador de la matriz considerada (filas), la diferencia entre los valores máximo y mínimo.
2. Se toma el límite de mínima variabilidad deseada para la ómica correspondiente de la matriz considerada.
3. Se cogen los reguladores cuyos valores calculados en el paso 1 sean mayores que el valor de mínima variabilidad tomado en el paso 2.

ii) Ómica reguladora binaria.

En el caso de que la ómica reguladora sea binaria, hemos tenido que desarrollar una nueva estrategia para filtrar los reguladores con baja variabilidad. Como se ha especificado anteriormente, aquí también el valor de corte es introducido por el usuario, o bien, es fijado por el propio algoritmo MORE si el usuario así lo desea (indicando en valor de corte `min.variation` el valor NULL, véase sección 5.1.1). Este parámetro de mínima variabilidad ha sido tratado de forma especial en las ómicas binarias:

- Si el usuario indica `min.variation = 0`, significará que el valor de corte para todas las ómicas indistintamente es 0. En cambio, en el caso de las ómica binarias no filtrará por baja variabilidad.
- Si el usuario indica `min.variation = NULL`, el corte que computa MORE automáticamente para las ómicas binarias es 0.9.
- Si el usuario indica valores de corte para las distintas ómicas, el correspondiente a la ómica binaria se computará como el complementario de este valor (1 - valor), ya que el usuario introducirá valores pequeños.

En las ómicas reguladoras binarias, la matriz que contiene las medias entre réplicas tendrá valores determinados:

- Si no hay réplicas, los valores serán 0 ó 1, ya que se toma la ómica binaria original.
- En caso de haber réplicas, según de cuántas réplicas dispongamos, tendremos unos valores determinados u otros. Por ejemplo, considérese que hay dos réplicas.

Al ser la matriz de medias compuesta por las medias entre réplicas, se tiene que los posibles valores de las medias serán 0, $\frac{1}{2}$ ó 1 en este caso. Sin embargo, razonando del mismo modo, se tiene que con tres réplicas los posibles valores serán 0, $\frac{1}{3}$, $\frac{2}{3}$ ó 1, y así sucesivamente.

La idea principal es calcular la proporción entre el número de veces que aparece un valor determinado y el total. Este cálculo es equivalente en los dos casos descritos: tanto si hay réplicas como si no. Para ello, se implementaron los siguientes pasos:

1. Se cuenta (para cada regulador) el número de valores que tenemos (1 y 0, o bien, los determinados según el número de réplicas) y dividimos cada recuento por el número total de valores que tenemos en la matriz de medias (o número total de columnas de esta), que será el mismo valor para cada matriz de medias, pues los valores de expresión han sido tomados bajo las mismas condiciones experimentales.
2. Se toma, para cada regulador, el máximo de las proporciones calculadas en el paso anterior. Al trabajar con el máximo de las proporciones, se tiene que estas oscilan entre 0.5 y 1, por lo tanto, valores elevados indicarán que el regulador no varía suficientemente y de ahí que se haya definido de esta manera el parámetro `min.variation`.
3. Se seleccionan aquellos reguladores cuyo valor máximo de proporciones (paso 2) es mayor que el límite de mínima variabilidad (fijado por el usuario para la ómica correspondiente o computado por el algoritmo).

El resultado de la aplicación de esta función serán las matrices de medias filtradas para cada ómica reguladora, que contendrán exclusivamente aquellos reguladores que hayan pasado el filtro correspondiente. Internamente, después de esta función, el algoritmo pasará la selección de reguladores a las matrices de las ómicas reguladoras originales, de tal forma que tengamos el filtro aplicado a las matrices ómicas iniciales para proseguir con el algoritmo.

4.3. Función CollinearityFilter

Cuando se incorporan datos ómicos como variables explicativas en modelos de regresión es bastante común que aparezcan problemas de multicolinealidad, ya que

las distintas entidades biológicas (los genes, por ejemplo) suelen actuar en grupos y, por lo tanto, se puede observar en este tipo de datos que hay determinados grupos de reguladores que tienden a tener el mismo perfil a lo largo de las observaciones. En particular, dos reguladores correlacionados positivamente tendrán perfiles similares, mientras que si están correlacionados negativamente tendrán perfiles opuestos.

Por ello, es conveniente tratar este posible problema de multicolinealidad antes de crear el modelo. Con el objetivo de filtrar reguladores correlacionados y así solucionar la problemática asociada a la multicolinealidad, se desarrolló la función `CollinearityFilter()`, cuya implementación en R puede encontrarse en el Anexo B.3. Se trata de una función auxiliar utilizada por la función principal **GetGLM**.

```
> CollinearityFilter(data, reg.table, correlation = 0.8, omic.type)
```

La aplicación de `CollinearityFilter()` se hace para cada gen de forma independiente, por lo que los parámetros de entrada son:

- ◇ **data**. Matriz que contiene los reguladores que no han sido filtrados por baja variabilidad o por datos faltantes. Esta matriz la computa el propio algoritmo previamente al uso de esta función.

- ◇ **reg.table**. Tabla que recoge en columnas el gen considerado (*gene*), sus reguladores que no han sido filtrados previamente (*regulator*), la ómica a la que pertenece cada regulador (*omic*), el área (*area*) y el filtro (*filter*), siendo este último rellenado para los reguladores correlacionados.

- ◇ **correlation**. Límite indicado por el usuario a partir del que se considerará que dos reguladores están correlacionados. Dado que esta es una función auxiliar, se utiliza realmente el valor que el usuario indica en la función principal, **GetGLM** en este caso.

- ◇ **omic.type**. Vector compuesto por 0 y 1 tal y como se ha definido en las secciones anteriores.

La idea es estudiar la correlación entre reguladores para cada tipo de ómica (numérica o binaria) y, en caso de encontrar un grupo de reguladores correlacionados, escoger aleatoriamente un regulador representante, que será el regulador que entre en la ecuación inicial del modelo GLM.

Además, esta función indicará dentro de cada grupo de reguladores correlacionados, si el resto están correlacionados positivamente o negativamente con el representate escogido al azar, así como la ómica y número de grupo al que pertenecen. Por ejemplo, considérese como ómica los datos de expresión de miRNAs, y el primer grupo de reguladores correlacionados. Las etiquetas que tendrá este grupo de reguladores serán:

miRNA_mc1_R Esta etiqueta será la asignada al regulador del primer grupo de reguladores correlacionados de la ómica miRNA que ha sido escogido de forma aleatoria como representante del grupo.

miRNA_mc1_P Esta etiqueta se asignará a aquellos reguladores del primer grupo de reguladores correlacionados de la ómica miRNA que están correlacionados positivamente con el regulador representante, lo que querrá decir que tienen perfiles similares.

miRNA_mc1_N Esta etiqueta se asignará a aquellos reguladores del primer grupo de reguladores correlacionados de la ómica miRNA que están correlacionados negativamente con el regulador representante, lo que querrá decir que tienen perfiles opuestos.

Este resultado se plasmará en la tabla **reg.table** definida anteriormente, concretamente en la columna llamada *filter* que, para los reguladores correlacionados aparecerán sus respectivas etiquetas, ya que esta columna hace referencia al tipo de filtro que ha causado la eliminación del regulador correspondiente. Consecuentemente, el resultado final es una actualización del parámetro de entrada **reg.table**.

La construcción de la función `CollinearityFilter()` se basa, en primer lugar, en discernir entre las ómicas reguladoras numéricas y binarias a través del parámetro de entrada `omic.type`. Para cada ómica, realizará los siguientes pasos según sea numérica o binaria:

i) Ómicas reguladoras numéricas.

1. Al haber pasado previamente los filtros de valores faltantes y de baja variabilidad, hay que comprobar que la ómica dada posee todavía reguladores que no hayan sido eliminados. En caso de no poseer reguladores, pasará a analizar la siguiente ómica, en caso contrario seguirá con el paso 2.
2. Se calcula el coeficiente de correlación de Pearson para cada pareja de reguladores (véase sección 3.4.1). Una pareja de reguladores se considerará correlacionada si su correlación de Pearson es mayor que el valor de corte establecido como parámetro de entrada. Para tener en cuenta las correlaciones negativas, se considera el valor absoluto de las correlaciones de Pearson.
3. Una vez escogidas las parejas de reguladores correlacionados, pueden ocurrir dos cosas:

3.1. Solo se tiene una única pareja de reguladores correlacionados. Este caso es el más sencillo de los dos, ya que simplemente basta con escoger uno de los dos reguladores al azar. El regulador no seleccionado se elimina de la matriz de reguladores, ya que no podrá formar parte del modelo. En la tabla resumen final **reg.table**, concretamente en la columna *filter*, se le asignará el nombre **ómica_mc1_P** o **ómica_mc1_N** (1 al solo haber una pareja) según sea el tipo de correlación entre ambas. En cuanto al representante, en la columna *filter* aparecerá la etiqueta **ómica_mc1_R** y se añadirá una variable ficticia (regulador ficticio) cuyo nombre será la etiqueta indicada y que irá al final de la tabla, ya que el regulador escogido como representante puede ser filtrado después mediante la selección de variables: Lasso, ElasticNet... (véase sección 3.3). Esto significa que, si no añadimos esta variable ficticia, podríamos perder en el futuro la información de la correlación y elección del representante.

3.2. El segundo caso que se puede dar es que haya más de una pareja de reguladores correlacionados. Aquí, se crea un grafo con todos estos reguladores (nodos), y solo se unen por aristas aquellos pares de reguladores que estén correlacionados. A continuación, se extraen las componentes conexas de este grafo mediante la librería de R `igraph`, de tal forma que los grupos de reguladores correlacionados son las componentes conexas con más de un regulador. De cada grupo (componente conexa del grafo) se escogerá aleatoriamente un representante al azar, que recibirá la asignación: **ómica_mc1_R**, **ómica_mc2_R**... según al grupo que pertenezca. El resto de reguladores tendrán la etiqueta **ómica_mc1_P**, **ómica_mc1_N**, **ómica_mc2_P**, **ómica_mc2_N**... según el grupo y tipo de correlación con el representante. Análogamente al caso anterior, estas etiquetas se asignan en la columna *filter* de la tabla y creando las respectivas variables ficticias para evitar la pérdida de información (véase Figura 4.2).

ii) Ómicas reguladoras binarias.

En el caso de las ómicas reguladoras binarias se tiene que el procedimiento es análogo y solo varía la forma de calcular la correlación, pero la asignación de etiquetas, creación de variables ficticias... es exactamente igual.

En la sección 3.4.2 se han definido las posibles medidas de correlación para variables categóricas y ahora debemos escoger la mejor medida de estas. Para juzgar qué medida de correlación era más adecuada, se diseñaron diversos

reguladores (variables) ficticios binarios con los valores 0 ó 1 en diferentes posiciones (Figura 4.1).

```
> miPrueba
          Regulador2 Regulador3 Regulador5 Regulador6 Regulador7 Regulador9 Regulador10
Muestra1           1          1          0          1          0          1           0
Muestra2           0          1          0          1          1          0           1
Muestra3           0          1          0          0          0          0           1
Muestra4           0          1          0          1          1          0           1
Muestra5           0          0          1          1          1          0           1
Muestra6           0          0          1          1          1          0           1
Muestra7           0          0          1          1          1          0           1
Muestra8           0          0          1          1          1          0           1
Muestra9           0          0          1          1          1          0           1
Muestra10          0          0          1          1          1          0           1
Muestra11          0          0          1          1          1          0           1
Muestra12          0          0          1          1          1          0           1
Muestra13          0          0          1          1          1          0           1
Muestra14          0          0          1          1          1          0           1
```

Figura 4.1: Variables binarias ficticias para el estudio de la mejor medida de correlación. Concretamente se hicieron tablas de contingencia 2×2 con las columnas etiquetadas como Regulador 2 y Regulador 9, Regulador 9 y Regulador 10, Regulador 6 y Regulador 7.

Se calcularon y compararon las medidas de correlación descritas en la sección 3.4.2:

- **Coeficiente *Phi***. A través de la librería `psych` mediante la función `phi()`.
- **Cramer V**. A través de la librería `vcd` mediante la función `assocstats()`. Esta función también da como resultado el coeficiente *phi*. Sin embargo, esta función no presenta la corrección al sesgo de Cramer V, por lo que se optó por programar la función (véase Anexo A).
- **Kappa de Cohen**. Cargando la librería `psych` y haciendo uso de la función `cohen.kappa()`. Destacar que esta función da como resultados valores entre -1 y 1, pero en la literatura el coeficiente Kappa de Cohen se define entre 0 y 1.
- **Coeficiente de correlación tetracórica**. Mediante la librería `psych` y usando la función `tetrachoric()`.
- **Índice de Jaccard**. Se programó en R, véase Anexo A.

En el Anexo A se adjuntan las instrucciones que permitieron discernir entre las distintas medidas a partir del cálculo de tablas de contingencia 2×2. Esto dio lugar a que escogiéramos el coeficiente *phi*, siendo las razones principales que propiciaron esta elección las siguientes:

a) El coeficiente de correlación Cramer V disminuía mucho con la corrección del sesgo en casos en los que existía una evidente correlación, mientras que el coeficiente *phi* daba valores coherentes.

b) El coeficiente *phi* permitía identificar más fácilmente correlaciones negativas al tomar valores en [-1,1]. Como puede verse en el Anexo A, cuando se espera correlación exacta negativa, en CramerV se obtiene 1 y en el índice de Jaccard 0.

c) El índice de Jaccard toma valores 0.25 en casos en los que existe una evidente correlación, por lo que no se identificaría como variables correlacionadas, mientras que con la correlación tetracórica no se obtenían valores coherentes, al igual que en el coeficiente Kappa de Cohen.

Entonces, la función para reguladores binarios difiere de los reguladores numéricos únicamente en la forma de estudiar la correlación: para cada par de reguladores se crea una tabla de contingencia 2×2, que se utiliza para calcular el coeficiente *phi*, es decir, la medida de correlación entre pares de reguladores binarios. Posteriormente, todos los coeficientes *phi* entre pares de reguladores se comparan con el parámetro de correlación y aquellos que sobrepasen este límite se considerarán parejas de reguladores correlacionados.

A partir de aquí se dan los mismos casos que en el apartado de las ómicas numéricas: que solo haya una única pareja o que hayan más. En ambos casos se procede de la misma forma que la descrita en las ómicas reguladoras numéricas, obteniéndose la actualización de la tabla correspondiente a aquellos reguladores pertenecientes a ómicas binarias (Figura 4.2).

```
> res$SummaryPerGene[1:25,]
                    gene                 regulator            omic        area filter
13_5789384_5789613 "ENSMUSG00000000078" "13_5789384_5789613" "ChIP-seq"  ""   "ChIP-seq_mc1_R"
13_5803679_5803880 "ENSMUSG00000000078" "13_5803679_5803880" "ChIP-seq"  ""   "ChIP-seq_mc1_P"
13_5804696_5804875 "ENSMUSG00000000078" "13_5804696_5804875" "ChIP-seq"  ""   "ChIP-seq_mc1_P"
13_5860779_5861047 "ENSMUSG00000000078" "13_5860779_5861047" "ChIP-seq"  ""   "ChIP-seq_mc1_N"
13_5861542_5861793 "ENSMUSG00000000078" "13_5861542_5861793" "ChIP-seq"  ""   "ChIP-seq_mc1_P"
13_5926774_5926959 "ENSMUSG00000000078" "13_5926774_5926959" "ChIP-seq"  ""   "ChIP-seq_mc1_P"
mmu-miR-431-5p     "ENSMUSG00000000078" "mmu-miR-431-5p"     "miRNA-seq" ""   "Model"
mmu-miR-410-3p     "ENSMUSG00000000078" "mmu-miR-410-3p"     "miRNA-seq" ""   "Model"
mmu-miR-30a-3p     "ENSMUSG00000000078" "mmu-miR-30a-3p"     "miRNA-seq" ""   "Model"
mmu-miR-208a-5p    "ENSMUSG00000000078" "mmu-miR-208a-5p"    "miRNA-seq" ""   "Model"
mmu-miR-7081-3p    "ENSMUSG00000000078" "mmu-miR-7081-3p"    "miRNA-seq" ""   "Model"
mmu-miR-200a-3p    "ENSMUSG00000000078" "mmu-miR-200a-3p"    "miRNA-seq" ""   "Model"
mmu-miR-6935-3p    "ENSMUSG00000000078" "mmu-miR-6935-3p"    "miRNA-seq" ""   "Model"
mmu-miR-320-3p     "ENSMUSG00000000078" "mmu-miR-320-3p"     "miRNA-seq" ""   "Model"
mmu-miR-93-3p      "ENSMUSG00000000078" "mmu-miR-93-3p"      "miRNA-seq" ""   "Model"
mmu-miR-298-3p     "ENSMUSG00000000078" "mmu-miR-298-3p"     "miRNA-seq" ""   "Model"
mmu-miR-122-3p     "ENSMUSG00000000078" "mmu-miR-122-3p"     "miRNA-seq" ""   "Model"
mmu-miR-34a-3p     "ENSMUSG00000000078" "mmu-miR-34a-3p"     "miRNA-seq" ""   "Model"
mmu-miR-30e-3p     "ENSMUSG00000000078" "mmu-miR-30e-3p"     "miRNA-seq" ""   "Model"
mmu-miR-350-5p     "ENSMUSG00000000078" "mmu-miR-350-5p"     "miRNA-seq" ""   "miRNA-seq_mc1_P"
mmu-miR-744-3p     "ENSMUSG00000000078" "mmu-miR-744-3p"     "miRNA-seq" ""   "Model"
mmu-miR-188-5p     "ENSMUSG00000000078" "mmu-miR-188-5p"     "miRNA-seq" ""   "Model"
mmu-miR-5118       "ENSMUSG00000000078" "mmu-miR-5118"       "miRNA-seq" ""   "Model"
mmu-miR-294-5p     "ENSMUSG00000000078" "mmu-miR-294-5p"     "miRNA-seq" ""   "Model"
mmu-miR-466f-3p    "ENSMUSG00000000078" "mmu-miR-466f-3p"    "miRNA-seq" ""   "Model"
```

Figura 4.2: Una de las salidas de la función `CollinearityFilter()`. Aquí pueden observarse las etiquetas para los reguladores correlacionados positivamente, negativamente y el representante. Las variables ficticias aparecerán al final de tabla como se mostrará en la Figura 5.10.

4.4. Función RegulationPerCondition

La función **GetGLM** de MORE estima un modelo GLM diferente para cada uno de los genes en la matriz respuesta con sus respectivos reguladores (sección 5.1.1). Por tanto, y tal como se describe en la sección 3.5, esta función genera una cantidad ingente de información en forma de listas de R encadenadas: resultados individuales para cada gen, resumen global, argumentos utilizados...

Con el fin de facilitar al usuario el manejo de dicha información y extraer los resultados más esenciales que son las regulaciones significativas, se desarrolló la función `RegulationPerCondition()`, que es otra de las tres funciones principales del paquete MORE (véase sección 5.1). El lector puede consultar el código R de esta función en el Anexo B.4.

```
> RegulationPerCondition(getGLMoutput, betaTest = TRUE)
```

Los parámetros de entrada que necesita esta función son:

- ◇ **getGLMoutput**. Es el objeto final de la función principal `GetGLM()` que contiene todas las salidas descritas en la sección 3.5: *ResultsPerGene*, *GlobalSummary* y *Arguments*.
- ◇ **betaTest**. Es un parámetro que en caso de tomar el valor TRUE (por defecto), ejecutará internamente la función `BetaTest()` (véase la sección 4.5).

El objetivo de la función `RegulationPerCondition()` es generar una tabla compuesta por las siguientes columnas:

- ▶ **gene**. Contiene el nombre de todos los genes para los que se ha obtenido un modelo GLM final.
- ▶ **regulator**. Contiene el nombre de todos los reguladores que han resultado significativos para cada modelo GLM.
- ▶ **omic**. La ómica a la cual pertenece cada regulador de la columna anterior.
- ▶ **area**. Si en el conjunto de datos se ha especificado adicionalmente el área.
- ▶ **representative**. Columna que indica únicamente en los grupos de reguladores correlacionados el nombre original del regulador representante.
- ▶ **Group**. Habrá tantas columnas como grupos de combinaciones experimentales se obtengan de la matriz de diseño experimental y serán las que almacenen los coeficientes de regresión para cada regulador en cada covariable.

Para completar el desarrollo de esta función, se ha programado una función auxiliar llamada `GetPairs1GeneRegulator()` que generará una primera tabla con todas las columnas excepto *representative* y las columnas *Group* descritas anteriormente. El código de esta función también se adjunta en el Anexo B.4. El parámetro de entrada *gene* es un objeto que contiene los nombres de los genes que se tienen disponibles.

```
> GetPairs1GeneRegulator(gene, getGLMoutput)
```

Después de haber generado esta primera tabla con `GetPairs1GeneRegulator()`, se le añade la columna *representative* y se termina de completar según dos supuestos que determinarán el número adicional de columnas a añadir:

i) Si la matriz del diseño experimental es nula.

- Al no haber matriz de diseño experimental, se tiene que no habrá interacciones entre reguladores y condiciones experimentales, por lo que solo debe crearse una única columna que albergue los coeficientes de regresión obtenidos en los modelos GLM finales.
- Solo se indican los valores de los coeficientes de regresión en los reguladores no correlacionados y en el representante de cada grupo de reguladores correlacionados.
- La finalidad de esto último es poder asignar a los reguladores correlacionados positivamente con su correspondiente representante, el mismo coeficiente de regresión. En cambio, los correlacionados negativamente deberán tener el mismo coeficiente con signo opuesto, ya que presentan perfiles opuestos.
- En la columna *representative* se indicará, para cada regulador perteneciente a un determinado grupo de reguladores correlacionados, el nombre original del regulador que ha sido considerado al azar como representante del grupo de reguladores correlacionados.

ii) Si se tiene una matriz del diseño experimental.

Al haber matriz de diseño experimental, puede haber interacciones entre reguladores y condiciones experimentales, por lo que habrá que crear adicionalmente en la primera tabla generada, tantas columnas como condiciones experimentales tengamos. En cada una de estas columnas aparecerán los coeficientes de regresión correspondientes a la relación entre los reguladores y las condiciones experimentales, por lo que habrán sumas de coeficientes a causa de la agrupación de covariables del modelo, ya que estos indicarán el efecto del

regulador sobre la expresión génica en los distintos grupos (combinación de las condiciones experimentales).

Por ejemplo, considérese dos grupos (control y enfermos), siendo el referencial el grupo referente a los pacientes control, indicados por la variable *dummy* G, y un regulador llamado X_1. Se desea saber el efecto del regulador sobre la expresión génica (y) tanto en el grupo control como en el de pacientes enfermos. Por lo tanto, supongamos que el modelo GLM resultante es:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_1 \times G$$

Al tener dos grupos, obtendríamos dos columnas adicionales en la tabla. Para el regulador X_1 considerado, la primera se correspondería con el coeficiente β_1, ya que sería el efecto relativo al grupo referencial (control), mientras que en la segunda columna aparecería la suma de coeficientes $\beta_1 + \beta_2$ que cuantificará el efecto del regulador sobre la expresión génica en el grupo de pacientes enfermos. De aquí que surja la necesidad de aplicar contrastes de hipótesis a la suma de coeficientes de regresión.

En particular, el funcionamiento de la función `RegulationPerCondition()` a partir de aquí es:

- Añadimos a la tabla tantas columnas como condiciones experimentales se tenga.
- Para cada gen, se construyen conjuntos que contengan los reguladores que se presentan individualmente en el modelo, exclusivamente en una interacción o en ambas.
- Se evalúa para cada regulador significativo del gen correspondiente, a qué conjunto pertenece de los construidos anteriormente y, según esto, se le asigna su respectivo coeficiente que puede ser un único coeficiente o suma de coeficientes. En este punto, solo se asignan los coeficientes a los reguladores no correlacionados y el representante del grupo de reguladores correlacionados.
- Ahora, si el parámetro `betaTest` es TRUE, se aplicará la función `BetaTest()` definida en la siguiente sección, cuyo objetivo es realizar el contraste de hipótesis para la suma de coeficientes tal y como se definió en la sección 3.2.5.
- Una vez realizados los contrastes para los coeficientes que así lo necesitasen, se asignan los coeficientes correspondientes a los reguladores

correlacionados: si están correlacionados positivamente con el regulador representante, tendrán los mismos coeficientes que este, mientras que si están correlacionados negativamente serán los mismos coeficientes pero de signo opuesto.

4.5. Función BetaTest

Esta función fue construida como función auxiliar de la función definida en la sección anterior (`RegulationPerCondition()`), con la finalidad de realizar tests de hipótesis sobre sumas de coeficientes, en concreto, el contraste definido en la sección 3.2.5. El código empleado para la construcción de esta función puede consultarse en el Anexo B.5.

```
> BetaTest(coeffs, myGene, MOREresults)
```

Los parámetros de entrada de la función `BetaTest()` son:

- ◇ **coeffs**. Tabla compuesta por las columnas definidas en la sección anterior con los coeficientes de regresión asignados a los reguladores no correlacionados y al regulador representante.
- ◇ **myGene**. Gen dado para el que se aplicará el contraste de hipótesis a sus respectivos coeficientes de regresión almacenados en la tabla anterior si se corresponde.
- ◇ **MOREresults**. Es el objeto final de la función principal **GetGLM** que contiene todas las salidas descritas en la sección 3.5: *ResultsPerGene*, *GlobalSummary* y *Arguments*.

Internamente, esta función realiza los siguientes pasos:

- De la tabla contenida en *coeffs*, se toma la subtabla definida por el gen dado en *myGene*. Por lo tanto, se tendrá una tabla con las mismas columnas pero menos filas al eliminar el resto de genes.
- A partir de esta subtabla, la función pretende encontrar aquellos coeficientes que se corresponden con una suma de coeficientes. Esto es así si, y solo si, el coeficiente del grupo referencial es distinto de 0 (primera columna de las columnas *Group* adicionales), ya que si es 0 no sería necesario hacer un contraste de hipótesis adicional porque no tendríamos ya una suma de coeficientes. Por tanto, la subtabla aún se reduce más: solo se presentan aquellos reguladores cuyos coeficientes para el grupo referencial sean diferentes de 0.

- A partir de la subtabla generada en el paso anterior, comparamos cada columna con la del grupo referencial para ver si los coeficientes son distintos, ya que esto conducirá a que existe una suma de coeficientes. Ahora la subtabla vuelve a reducirse, como consecuencia de quedarnos con aquellos reguladores que presentan coeficientes distintos al referencial.

- Ahora estamos en disposición de poder aplicar el contraste. Para ello, se genera el modelo GLM y la hipótesis nula. A través de la función `linearHypothesis()` (de la librería de R `car`), se obtendrá un p–valor a partir de la distribución χ^2. Esto debe hacerse para cada coeficiente de regresión del punto anterior.

- Por último, se analiza si cada p–valor es mayor que el nivel de significación α dado por el usuario (es un parámetro de entrada de MORE, véase sección 5.1) y, en caso de ser así, se acepta la hipótesis nula de que la suma de coeficientes es nula, por lo que al correspondiente coeficiente de regresión se le asignará el valor 0. En caso contrario, se le asigna el valor de la suma de coeficientes.

Capítulo 5

Paquete MORE y MORE Shiny

En la presente sección se presentan, en primer lugar, las tres funciones principales que componen el paquete MORE, para después mostrar la construcción del paquete y su extensión a aplicaciones web mediante Shiny.

5.1. Composición del paquete

El paquete MORE se compone de tres funciones principales que son las que el usuario podrá utilizar llamándolas de la forma que se define a continuación.

5.1.1. Función GetGLM

La función **GetGLM** es la función principal del paquete, ya que ajusta un modelo lineal generalizado (GLM) para cada gen (proteína, metabolito, etc.) del objeto **GeneExpression** para determinar qué reguladores y variables experimentales tienen un efecto significativo en la variable respuesta (expresión génica, niveles de proteína, etc.).

```
GetGLM(GeneExpression,
        associations,
        data.omics,
        edesign = NULL,
        center = TRUE,
        scale = FALSE,
        Res.df = 5,
        epsilon = 0.00001,
        alfa = 0.05,
        MT.adjust = "none",
        family = negative.binomial(theta=10),
        elasticnet = 0.5,
        stepwise = "backward",
        interactions.reg = TRUE,
        min.variation = 0,
        correlation = 0.9,
        min.obs = 10,
        omic.type = 0)
```

Los argumentos de entrada de la función son los que se presentan a continuación, siendo **GeneExpression**, **associations**, **data.omics** y **edesign** las matrices de expresión génica, de asociaciones, de ómicas reguladoras y de diseño experimental descritas en las sección 3.5.

- **center**. Si es TRUE (valor por defecto), las ómicas numéricas serán centradas.
- **scale**. Si es TRUE, las ómicas numéricas serán escaladas. Su valor por defecto es FALSE.
- **Res.df**. Número de grados de libertad residuales. Por defecto, son 5. Al incrementar los grados de libertad residual, se tendrá una mayor potencia estadística y disminuirá el número de predictores significativos.
- **epsilon**. Umbral para la tolerancia de convergencia positiva en el modelo GLM. Su valor por defecto es 0.00001.
- **alfa**. Nivel de significación. Por defecto, 0.05.
- **MT.adjust**. Método de corrección de pruebas múltiples para ser utilizado dentro del procedimiento de selección de variables. Su valor por defecto es, "none" (véase las diferentes opciones en `?p.adjust` de R).
- **family**. Distribución y función *link* para ser utilizadas en el modelo (véase `?glm` en R para más información). Por defecto, `negative.binomial(theta = 10)`.
- **elasticnet**. Parámetro de ElasticNet. Por defecto, será 0.5. En cambio, puede tomar los siguientes valores:
  - ◇ **NULL**. No se hará ninguna selección de variables.
  - ◇ **Valor entre 0 y 1**. ElasticNet se aplica con estos valores que son una combinación entre la regresión Ridge y Lasso.
  - ◇ **Valor 0**. Penalización Ridge.
  - ◇ **Valor 1**. Penalización Lasso.
- **stepwise**. Selección de variables por el procedimiento Stepwise. Una de estas opciones puede ser elegida: "none", "backward" (valor por defecto), "forward", "two.ways.backward" o "two.ways.forward".
- **interactions.reg**. Si es TRUE (valor predeterminado), MORE permite las interacciones entre cada regulador y la covariable experimental.

- **min.variation**. En los reguladores con valores numéricos, indicará el cambio mínimo que un regulador debe presentar a través de las condiciones para mantenerlo en los modelos de regresión. En cambio, para los reguladores binarios, si la proporción del valor más repetido iguala o excede este valor, se considerará que el regulador tiene baja variación y se elimina de los modelos de regresión. El valor por defecto es 0.

- **correlation**. Umbral de correlación (en valor absoluto) para decidir qué reguladores están correlacionados, en cuyo caso se elige un representante del grupo de reguladores correlacionados para ingresar al modelo. Por defecto, 0.9.

- **min.obs**. Mínimo valor de observaciones que un gen debe tener para computarse el modelo GLM. Su valor por defecto es 10.

- **omic.type**. Vector compuesto por tantos elementos como número de ómicas tengamos, que indicarán si la ómica presenta valores numéricos (0, por defecto) o valores binarios (1). Cuando se da un valor único, el tipo para todas las ómicas se establece en ese valor. Por defecto, 0.

Las salidas de la función **GetGLM** son las salidas que se han descrito en la sección 3.5, donde se ha explicado la funcionalidad y objetivo del método MORE.

5.1.2. Función RegulationPerCondition

Esta función se corresponde con la descrita como nueva funcionalidad de MORE en la sección 4.4. Una vez se haya ejecutado la función **GetGLM** y guardado los resultados en un objeto, puede utilizarse la función **RegulationPerCondition** para obtener la tabla resumen que se explicó en la correspondiente sección.

5.1.3. Función plotGLM

El paquete MORE incluye la función **plotGLM** para representar gráficamente la relación entre genes y reguladores: para un par gen–regulador dado, para explorar los reguladores de un gen determinado, o para analizar qué genes están regulados por un regulador específico, entre otras.

```
plotGLM(GLMoutput,
        gene,
        regulator = NULL,
        reguValues = NULL,
        plotPerOmic = FALSE,
        gene.col = 1,
        regu.col = NULL,
        order = TRUE,
```

```
    xlab = "",
    cont.var = NULL,
    cond2plot = NULL)
```

Los parámetros de entrada de la función **plotGLM** son,

- **GLMoutput**. Objeto generado por la función **GetGLM**.
- **gene**. ID del gen que se quiere graficar.
- **regulator**. ID del regulador que se quiere graficar. Si el valor es NULL (por defecto), todos los reguladores del gen dado serán graficados.
- **reguValues**. Vector que contiene los valores de un regulador que el usuario puede proporcionar de forma opcional. Si es NULL (valor por defecto), estos valores son cogidos del objeto **GLMoutput** siempre que estén disponibles.
- **plotPerOmic**. Si es TRUE, todos los reguladores significativos del gen dado y la misma ómica se grafican en el mismo gráfico. Si es FALSE (valor predeterminado), cada regulador se grafica en gráficos separados.
- **gene.col**. Color con el que se grafica el gen, por defecto es 1 (negro).
- **regu.col**. Color con el que se grafica el regulador. Si es NULL (valor por defecto), la propia función asignará un color, que será diferente para cada ómica reguladora.
- **order**. Si es TRUE (valor por defecto), los valores en el eje X aparecerán ordenados.
- **xlab**. Etiqueta o nombre para el eje X.
- **cont.var**. Vector con una longitud igual al número de observaciones en los datos, que opcionalmente puede contener los valores de la variable numérica (por ejemplo, el tiempo) a graficar en el eje X. Por defecto, NULL.
- **cond2plot**. Vector o factor que indica el grupo experimental de cada valor para representar. Si es NULL (predeterminado), las etiquetas se toman de la matriz de diseño experimental.

A parte de los gráficos, también se da el número de reguladores gráficados para un gen dado o el número de genes graficados que son regulados por un regulador determinado.

5.2. Paquete en R

Un paquete se entiende como una extensión del sistema base R con código, datos y documentación en formato estandarizado (véase Figura 5.4). La creación y publicación de un paquete en R tiene una serie de objetivos y ventajas:

- El paquete es una herramienta para poder mantener colecciones de funciones y datos, así como la correcta documentación de estas.
- Permite la publicación de código de forma que otros usuarios pueden hacer uso de él, contribuyendo al crecimiento de R.
- La publicación de un paquete permite la realimentación de este, ya que otros usuarios tienen acceso a él y, por lo tanto, esto puede dar lugar al desarrollo y ampliación de sus funcionalidades y a la conexión con otras herramientas y proyectos.
- Los paquetes permiten una extensión fácil, transparente y multiplataforma del sistema base de R.

5.2.1. Preparación previa

En este trabajo se ha construido el paquete de R en el sistema operativo Windows. Para ello, se necesita la instalación de unas herramientas llamadas *Rtools*, que pueden ser descargadas para cada versión de R en la página web https://cran.r-project.org/bin/windows/Rtools/.

Además, en nuestro caso hace falta tener instalado una versión del procesador de textos TEX, como por ejemplo MiKTeX para desarrollar la documentación del paquete. Si se necesitase introducir código C, Fortran o C++, también se requiere la instalación de compiladores GNU [32, 5] que vienen en las herramientas *Rtools*. En cambio, en nuestro caso no ha sido necesario al no programar nada en estos lenguajes de programación.

Sin embargo, también es posible el desarrollo del paquete en otros sistemas operativos como Mac o Linux. Las herramientas que deberían instalarse en su caso pueden encontrarse en [36]. Cabe resaltar que a pesar de que el paquete se creó desde un sistema operativo concreto, el paquete puede instalarse y usarse en cualquier sistema operativo.

5.2.2. Creación del paquete

Una vez hecha la instalación de las *Rtools*, estamos en disposición de crear el paquete. Para ello, hay dos formas posibles de creación de un paquete: a través de RStudio o arrancando el comando `package.skeleton()`. Aquí se ha optado por la primera, ya que es más intuitiva y fácil. Si el lector desea saber cómo hacer uso de la segunda forma, puede consultar [19].

1. Arrancando RStudio, creamos un nuevo proyecto a través de `File > New Project...`, en el que aparecerá la ventana que se presenta en la Figura 5.1 y le pincharemos en `New Directory`.

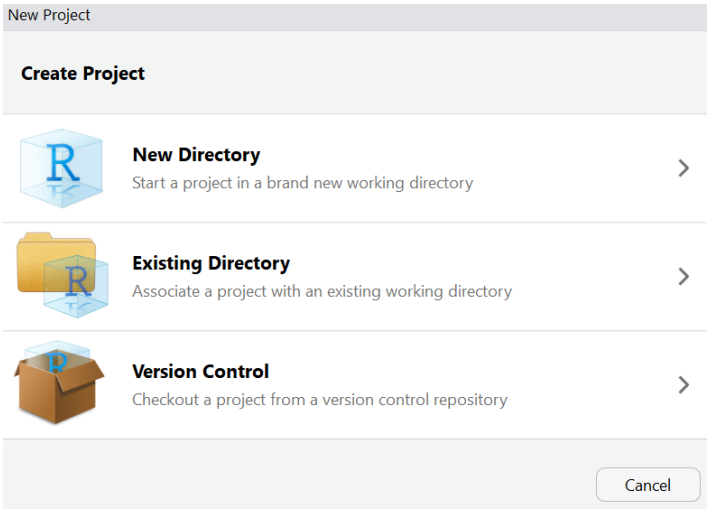

Figura 5.1: Creación de un nuevo proyecto en RStudio.

2. A continuación, se abrirá la ventana de la Figura 5.2 en la que indicaremos que el proyecto es un paquete (`R Package`).

3. El siguiente paso es darle un nombre al paquete, añadir los ficheros (scripts) que lo constituirán y el lugar de creación de este, tal y como se muestra en la Figura 5.3.

Apretando el botón `Create Project`, tendremos una carpeta llamada con el nombre asignado al paquete (MORE) en el que se encuentra el proyecto y la estructura que debe cumplir el paquete de R.

5.2.3. Estructura de un paquete en R

Dentro de la carpeta MORE, ya que así hemos llamado a nuestro proyecto, deben encontrarse las siguientes carpetas que deberán incluir ciertos ficheros.

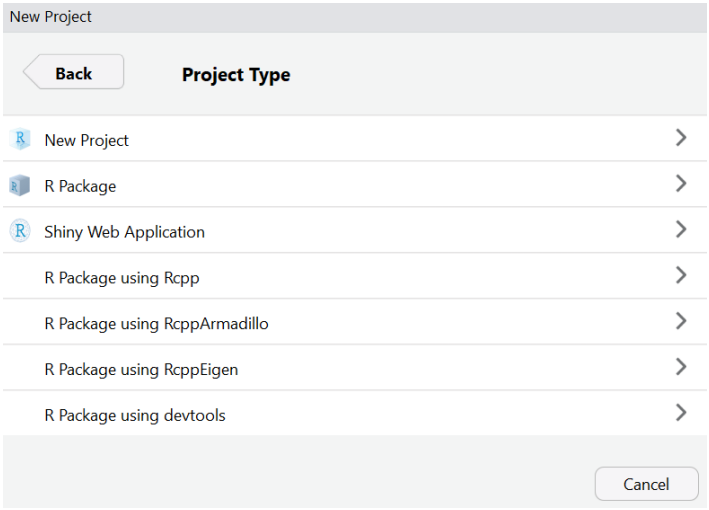

Figura 5.2: Tipo de proyecto: *R package*.

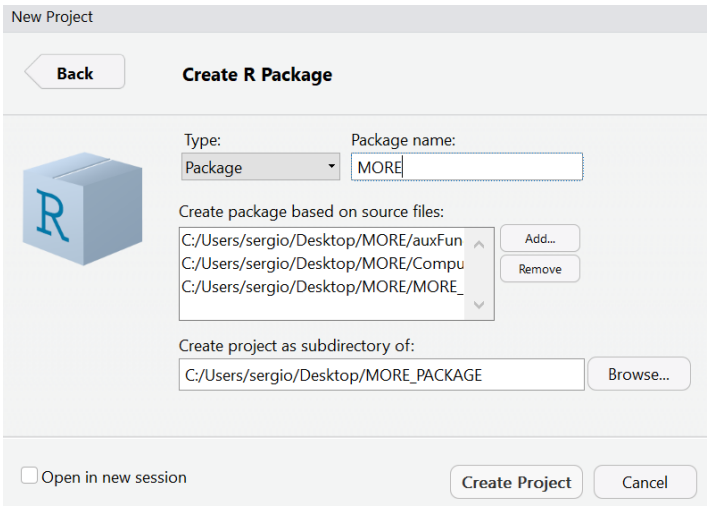

Figura 5.3: Creación del paquete: nombre, scripts y directorio donde se guardará.

◇ **data**. Es una carpeta en la que se ha incluido un fichero de datos de prueba, `TestData.RData` y se corresponden con los datos simulados descritos en la sección 3.6.2.

◇ **man**. Es una carpeta en la que debe incluirse la documentación de las funciones que el usuario usará en ficheros de extensión `.Rd`. En nuestro caso se han creado cuatro ficheros: `GetGLM.Rd`, `plotGLM.Rd`, `RegulationPerCondition.Rd` y `TestData.Rd`. En ellos se introducen una descripción, el uso de la función, los parámetros de entrada, los parámetros de salida y ejemplos de ejecución de las tres funciones principales del algoritmo MORE y del conjunto de datos de prueba (véase Figura 5.4). Esta documentación es la que aparecerá como ayuda en el cuadro *Help* de RStudio y que podrá accederse mediante la instrucción `?NombreFunción`, tal y como se muestra en la Figura 5.5.

```
\name{GetGLM}
\alias{GetGLM}
%- Also NEED an '\alias' for EACH other topic documented here.
\title{
GLM model for each gene
}
\description{
The \code{GetGLM} function adjusts a generalized linear model (GLM) for each of the genes, so that it can be analyzed which regulators
and experimental variables have a significant effect. To get to the final model, variable selection methods are applied (lasso, ridge
regression and ElasticNet), stepwise procedure, filter regulators with low variation, missing values and correlated regulators.
}
\usage{
GetGLM(GeneExpression, associations, data.omics, edesign = NULL,
center = TRUE, scale = FALSE, Res.df = 5, epsilon = 0.00001,
alfa = 0.05, MT.adjust = "none", family = negative.binomial(theta=10),
elasticnet = 0.5, stepwise = "backward", interactions.reg = TRUE,
min.variation = 0, correlation = 0.9, min.obs = 10, omic.type = 0)
}
%- maybe also 'usage' for other objects documented here.
\arguments{
  \item{GeneExpression}{Matrix or data frame containing gene expression data with genes in rows and experimental samples in columns. The
row names must be the gene IDs.}

  \item{associations}{List where each element corresponds to a different omic data type (miRNAs, transcription factors, methylation, etc
.). The names of the list will be the omics. Each element is a data frame with two columns (optionally three) describing the potential
interactions between genes and regulators for that omic. First column must contain the regulators, second the gene IDs, and an additional
column can be added to describe the type of interaction (for example, in methylation data, if a CpG site is located in the promoter
region of the gene, in the first exon, etc.).}

  \item{data.omics}{List where each element corresponds to a different omic data type (miRNAs, transcription factors, methylation, etc.).
The names of this list will be the omics and each element of the list is a  matrix or data frame with omic regulators in rows and samples
in columns.}

  \item{edesign}{Data frame or matrix describing the experimental design. Rows must be the samples, that is, the columns in the \code{Gen
eExpression}, and columns must be the experimental variables to be included in the model, such as disease, treatment, etc.}

  \item{center}{If TRUE (default), the omic data are centered.}

  \item{scale}{If TRUE, the omic data are scaled. Default value is set to FALSE.}

  \item{Res.df}{Number of degrees of freedom in the residuals. By default, 5. Increasing \code{Res.df} will increase the power of the
statistical model and decrease the number of significant predictors.}

  \item{epsilon}{A threshold for the positive convergence tolerance in the GLM model. By default, 0.00001.}

```

Figura 5.4: Estructura del fichero Rd. En particular se muestra el fichero `GetGLM.Rd`.

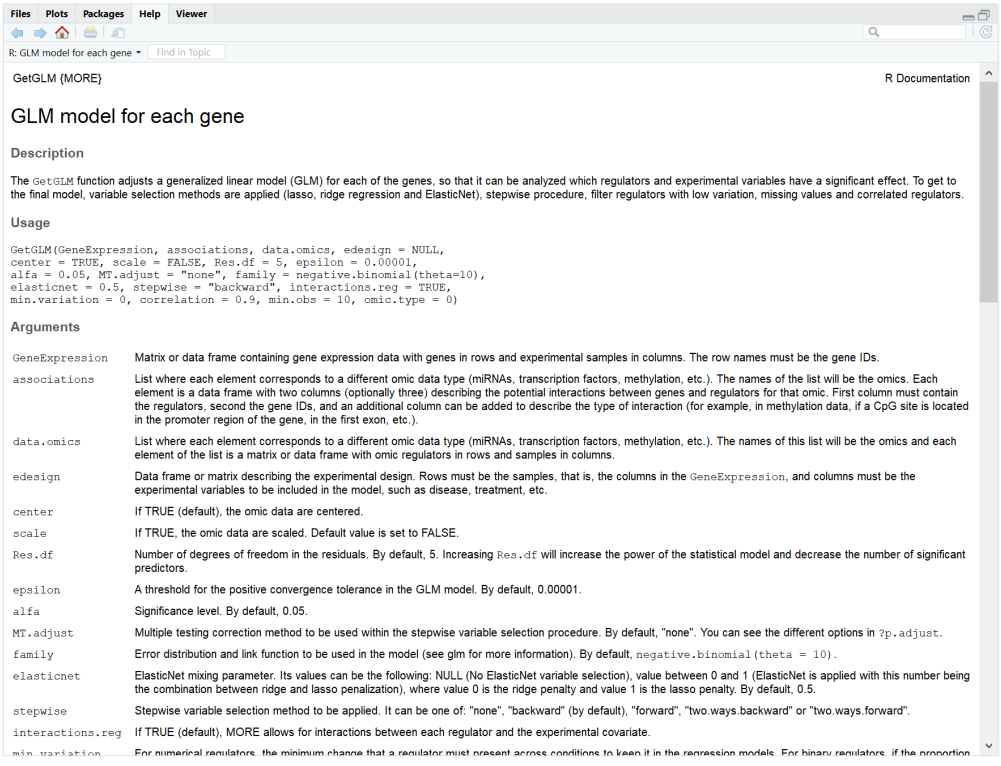

Figura 5.5: Cuadro de ayuda de la función **GetGLM** en R. Para acceder a la ayuda de esta función debe hacerse la instrucción `?GetGLM`.

- ◇ **R**. Es una carpeta donde deben estar incluidos los ficheros (scripts) que se han incluido en la ventana de la Figura 5.3 y son los que conforman la funcionalidad del paquete.

◇ **DESCRIPTION**. Es un archivo en el que se tiene una descripción general del paquete: nombre, tipo, autores, dependencia de paquetes, descripción, licencia... tal y como se muestra en la Figura 5.6.

```
Package: MORE
Type: Package
Title: Multi-Omics REgulation (MORE)
Version: 0.1.0
Author: Sonia Tarazona, Blanca Tomás Riquelme
Depends: igraph, MASS, glmnet, car, psych
Maintainer: Sonia Tarazona <starazona@cipf.es>
Description: MORE is a method which facilitate the task of applying GLMs models to multi-omic data. The goal of MORE is to
License: GPL-2
Encoding: UTF-8
LazyData: true
```

Figura 5.6: Fichero DESCRIPTION: descripción general del paquete.

◇ **NAMESPACE**. Es un archivo en el que deben incluirse la carga de las librerías de las que dependa nuestro paquete y se realiza a través del comando `Import()` (véase Figura 5.7). Con esto podemos dar más facilidades al usuario de MORE, ya que las librerías de las que depende se instalan y cargan automáticamente con la carga de nuestro paquete.

```
exportPattern("^[[:alpha:]]+")
import(igraph)
import(MASS)
import(glmnet)
import(car)
import(psych)
importFrom("grDevices", "colors")
importFrom("graphics", "abline", "arrows", "axis", "box", "lines",
           "mtext", "par", "plot", "points")
importFrom("stats", "aggregate", "anova", "as.dist", "as.formula",
           "coef", "cor", "glm", "model.matrix", "na.omit", "p.adjust",
           "residuals", "sd", "supsmu")
importFrom("utils", "combn")
```

Figura 5.7: Fichero NAMESPACE: dependencias del paquete.

◇ **MANUAL DE USUARIO**. Además, para hacer más fácil el uso del paquete MORE y el posterior uso de Shiny, se ha creado un manual de usuario en el que se explica, de forma secuencial, cómo utilizar el paquete MORE. También se detallan los parámetros de entrada y salida de las funciones del paquete, se dan ejemplos para ilustrar el uso de las funciones y se incluye una sección dedicada a la explicación del uso de la aplicación MORE Shiny que veremos en la siguiente sección. Este tipo de manuales no son obligatorios en los paquetes destinados al repositorio general CRAN, pero sí lo son en otros repositorios como Bioconductor. En nuestro caso, se decidió crear dicho manual porque facilita enormemente al usuario la utilización del paquete. El manual de usuario está disponible para cualquier usuario en https://bitbucket.org/ConesaLab/more/downloads/ (UserGuide.pdf).

5.2.4. Compilación del paquete

Una vez desarrollado todos los ficheros `Rd` descritos, los ficheros NAMESPACE y DESCRIPTION y guardado los ficheros y los datos en sus respectivas carpetas, estamos en disposición de pasar al paso final: compilar el paquete para crear un archivo y así compartirlo con el resto de usuarios de R. Este archivo es de tipo WinRAR (un archivo comprimido) de extensión `.gz` y que se construye fuera de la carpeta MORE creada en los pasos anteriores. Este proceso puede realizarse a través de la consola de R o a través de RStudio. Aquí se ha optado por hacerlo de la segunda forma, ya que es más sencillo e intuitivo. Los pasos a seguir son:

1. Checkear el paquete en busca de errores. Con el proyecto abierto, pinchamos en el botón `Check` de la Figura 5.8. En esa misma pantalla aparecerán los respectivos mensajes de error sobre los ficheros `.Rd` creados, errores en la compilación de los ejemplos, errores en el código de los scripts, posibles warnings o que todo está correctamente. En caso de errores habrá que corregirlos previamente al paso 2.

2. Cuando el checkeo del paquete sea correcto, estamos en disposición de compilar el paquete pulsando el botón `Install and Restart` de la Figura 5.8. Si hemos hecho más cambios después de pulsar este botón, podemos pinchar en el botón `Clean and Rebuild` y volverá a compilarse el paquete. En este paso se instala la librería MORE para que podamos ver cómo han quedado las ayudas, si funciona correctamente o la descripción del paquete.

3. Si todo está a nuestro gusto y no hay más cambios pasamos a generar el archivo definitivo WinRAR pulsando el botón `Build Source Package` (Figura 5.8). Automáticamente se generará, fuera de la carpeta donde se generó el proyecto, un archivo WinRAR de extensión `.gz` llamado **MORE_0.1.0.tar.gz** (nombre_versión.tar.gz), que es el archivo que se subirá al repositorio *Bitbucket* para que cualquier usuario pueda instalarse el paquete MORE.

5.2.5. Ejemplo de uso del paquete MORE

Como ilustración de la ejecución del paquete, se considerarán los datos simulados definidos en la sección 3.6.2 y los parámetros de entrada que se muestran en la Tabla 5.1. El primer paso para utilizar el paquete MORE es su instalación. El paquete MORE está disponible en https://bitbucket.org/ConesaLab/more, cuya instalación desde *Bitbucket* en R se realiza mediante las siguientes instrucciones:

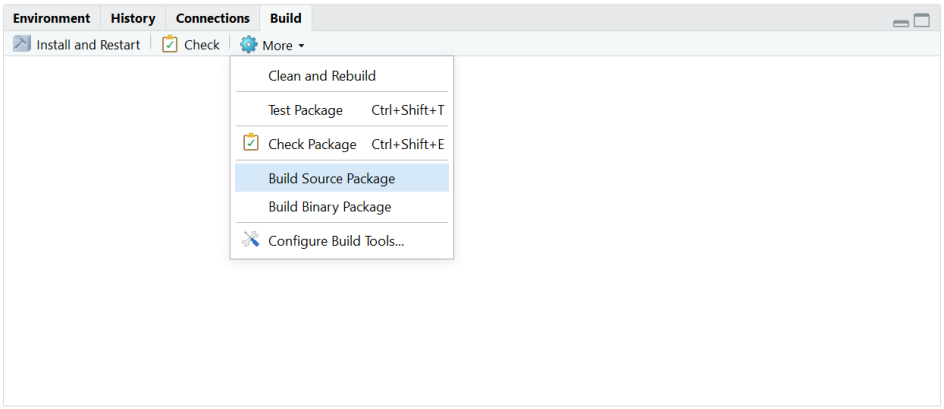

Figura 5.8: Compilación y construcción del archivo WinRAR en RStudio.

Parámetro	Valor	Parámetro	Valor
center	TRUE	elasticnet	0.3
scale	FALSE	stepwise	backward
Res.df	7	interactions.reg	TRUE
epsilon	0.00001	min.variation	NULL
alfa	0.05	correlation	0.9
MT.adjust	fdr	min.obs	10
family	negative.binomial(theta = 10)	omic.type	1,0,0

Tabla 5.1: Parámetros de entrada considerados para aplicar el método MORE a los datos simulados.

```
> install.packages("devtools")
> devtools::install_bitbucket("ConesaLab/more")
```

Ahora puede cargarse la librería MORE, los datos de prueba y arrancar el algoritmo MORE como se muestra en el Anexo C. En la Figura 5.9 se presenta cómo es la carga de la función principal `GetGLM()`, que hemos guardado en un objeto llamado `SimGLM`. Dentro de este objeto se encuentran todos los *outputs* de la función principal descritos en la sección 3.5.

Aquí se muestran, por ejemplo, para el gen ENSMUSG00000000078 los *outputs* tabla resumen **allRegulators** para los primeros 10 y últimos 15 reguladores y los coeficientes de regresión asociados a sus reguladores significativos (Figuras 5.10 y 5.11).

- En la tabla **allRegulators** puede observarse qué reguladores son significativos (columna *Sig* es 1 y columna *filter* es *Model*). En cambio, para los reguladores no significativos (columna *Sig* es 0), se presenta en la columna *filter* la razón. Por ejemplo, los seis primeros reguladores pertenecientes a la ómica ChIP–seq son un grupo de reguladores correlacionados y se ha escogido como representante el regulador 13_5861542_5861793. Este último se corresponde con el regulador ficticio ChIP–seq_mc1_R al final de la tabla. Sin embargo, no ha resultado

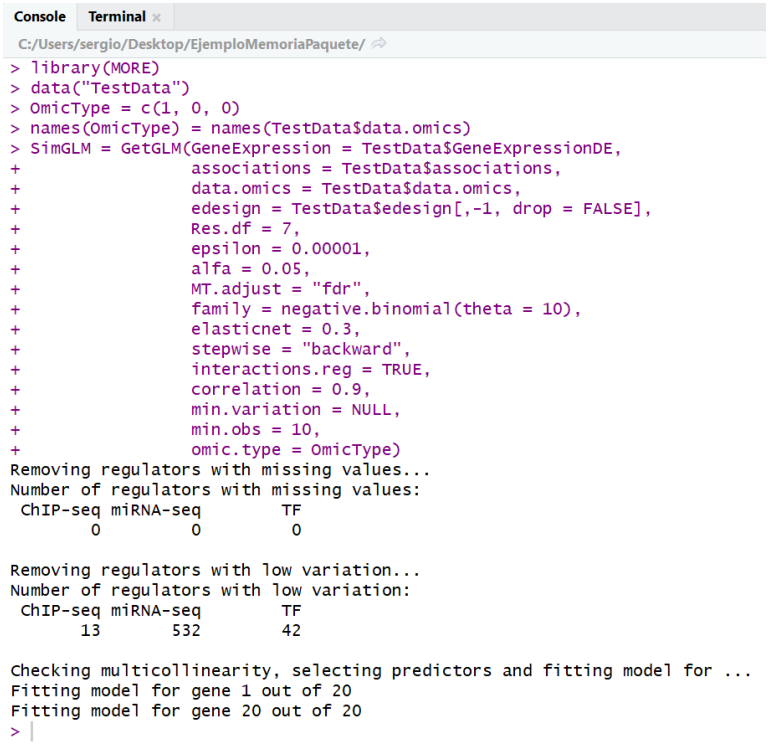

Figura 5.9: Carga del paquete, datos y función principal **GetGLM**.

significativo a causa de la selección de variables ElasticNet (columna *Sig* es 0 y columna *filter* es ElasticNet).

- En la Figura 5.11 se presentan tanto los coeficientes de regresión como el p–valor asociado a los reguladores significativos y sus interacciones para el gen ENSMUSG00000000078.

Otras salidas de interés pueden ser el porcentaje de devianza explicado por cada modelo, qué modelos son globalmente significativos o saber qué genes no han resultado tener finalmente un modelo GLM.

- En la Figura 5.12 se presenta una tabla (**GoodnessOfFit**) con los p–valores asociados a cada modelo, los grados de libertad residuales, el porcentaje de devianza explicado por cada modelo, el estadístico AIC y el número de reguladores significativos para cada gen.

- Otra salida es el número de genes que no tienen un modelo GLM y la razón. En este caso, no se ha obtenido ningún gen que no tenga modelo final. En el caso de haber, puede verse cuáles son introduciendo la instrucción en R `SimGLM$GlobalSummary$GenesNOmodel`.

- Por último, también puede ser de interés ver cuántos reguladores teníamos

```
> SimGLM$ResultsPerGene$ENSMUSG00000000078$allRegulators[1:10,]
                                gene          regulator      omic area         filter Sig
13_5789384_5789613 ENSMUSG00000000078 13_5789384_5789613  ChIP-seq      ChIP-seq_mc1_P   0
13_5803679_5803880 ENSMUSG00000000078 13_5803679_5803880  ChIP-seq      ChIP-seq_mc1_P   0
13_5804696_5804875 ENSMUSG00000000078 13_5804696_5804875  ChIP-seq      ChIP-seq_mc1_P   0
13_5860779_5861047 ENSMUSG00000000078 13_5860779_5861047  ChIP-seq      ChIP-seq_mc1_N   0
13_5861542_5861793 ENSMUSG00000000078 13_5861542_5861793  ChIP-seq      ChIP-seq_mc1_R   0
13_5926774_5926959 ENSMUSG00000000078 13_5926774_5926959  ChIP-seq      ChIP-seq_mc1_P   0
mmu-miR-431-5p     ENSMUSG00000000078     mmu-miR-431-5p miRNA-seq              Model   1
mmu-miR-18a-5p     ENSMUSG00000000078     mmu-miR-18a-5p miRNA-seq              Model   1
mmu-miR-18b-5p     ENSMUSG00000000078     mmu-miR-18b-5p miRNA-seq         ElasticNet   0
mmu-miR-25-3p      ENSMUSG00000000078      mmu-miR-25-3p miRNA-seq         ElasticNet   0
>
> SimGLM$ResultsPerGene$ENSMUSG00000000078$allRegulators[90:105,]
                              gene      regulator     omic area       filter Sig
Satb1              ENSMUSG00000000078          Satb1       TF        ElasticNet   0
Tbp                ENSMUSG00000000078            Tbp       TF          TF_mc1_N   1
Zfp628             ENSMUSG00000000078         Zfp628       TF          TF_mc1_N   1
Zfp64              ENSMUSG00000000078          Zfp64       TF        ElasticNet   0
Cux1               ENSMUSG00000000078           Cux1       TF       LowVariation   0
Gtf2b              ENSMUSG00000000078          Gtf2b       TF       LowVariation   0
Mafk               ENSMUSG00000000078           Mafk       TF       LowVariation   0
Pou6f1             ENSMUSG00000000078         Pou6f1       TF       LowVariation   0
Rara               ENSMUSG00000000078           Rara       TF       LowVariation   0
Rfx1               ENSMUSG00000000078           Rfx1       TF       LowVariation   0
Rreb1              ENSMUSG00000000078          Rreb1       TF       LowVariation   0
Rxra               ENSMUSG00000000078           Rxra       TF       LowVariation   0
Zfp692             ENSMUSG00000000078         Zfp692       TF       LowVariation   0
Zfp775             ENSMUSG00000000078         Zfp775       TF       LowVariation   0
ChIP-seq_mc1_R     ENSMUSG00000000078 ChIP-seq_mc1_R ChIP-seq         ElasticNet   0
TF_mc1_R           ENSMUSG00000000078       TF_mc1_R       TF              Model   1
```

Figura 5.10: Tabla **allRegulators** para el gen ENSMUSG00000000078. Al ser la tabla demasiado extensa, se han presentado los 10 primeros y los 15 últimos reguladores. Los dos últimos reguladores se corresponden con los reguladores ficticios descritos en la sección 4.3. Estos son los que han pasado por la selección de variables ElasticNet.

```
> SimGLM$ResultsPerGene$ENSMUSG00000000078$coefficients
                          coefficient      p-value
(Intercept)              8.768718e+00 2.043440e-23
`mmu-miR-18a-5p`         2.549854e-06 4.122175e-04
Hmga1                   -1.099662e-05 5.564550e-03
Mef2d                   -2.381339e-05 2.018984e-04
TF_mc1_R                -1.339656e-04 4.201164e-03
`Group1:mmu-miR-431-5p`  3.675612e-06 1.273422e-03
```

Figura 5.11: Coeficientes de regresión de los reguladores significativos en el modelo GLM para el gen ENSMUSG00000000078.

inicialmente, cuántos han sido incluidos en el modelo inicial y, de estos, cuántos han resultado significativos (véase Figura 5.13).

Sin embargo, para obtener una visión general de los resultados puede hacerse uso de la función **RegulationPerCondition**, desarrollada en este trabajo y explicada en el Capítulo 4.4, cuya ejecución y *ouput* se muestran en la Figura 5.14.

En la tabla se tiene para cada gen los reguladores que han resultado significativos en cada modelo GLM. En particular, la columna *representative* indica qué regulador fue escogido de forma aleatoria como representante de un grupo de reguladores correlacionados. Cuando no se proporciona información en esta columna, significa que el regulador no forma parte de ningún grupo de reguladores correlacionados. Como se indicó en la sección 4.4, los reguladores correlacionados positivamente con el representante tendrán los mismos coeficientes que este, mientras que los reguladores correlacionados negativamente con el representante tendrán los mismos coeficientes

```
> SimGLM$GlobalSummary$GoodnessOfFit
                    modelPvalue dfResiduals  Rsquared      AIC sigReg
ENSMUSG00000000078 2.416260e-24          10 0.9240024 284.9202      8
ENSMUSG00000056999 1.455313e-14          12 0.8428737 276.9555      3
ENSMUSG00000024873 6.778339e-13          12 0.8378507 207.5526      3
ENSMUSG00000015461 4.395871e-07          13 0.5570730 345.8397      2
ENSMUSG00000058135 1.862135e-24          12 0.9021445 277.6424      4
ENSMUSG00000038208 6.302497e-17          13 0.8466428 223.3464      2
ENSMUSG00000033985 4.378120e-59          12 0.9565413 224.0540      8
ENSMUSG00000048410 2.247470e-07          12 0.5502613 272.5239      2
ENSMUSG00000066036 7.941563e-06          14 0.5991302 311.2287      1
ENSMUSG00000041995 1.919478e-18          12 0.8771216 297.1618      3
ENSMUSG00000016018 2.730291e-22          12 0.8914844 327.4505      4
ENSMUSG00000012535 4.668036e-07          12 0.6552223 315.2774      5
ENSMUSG00000021583 4.894920e-11          12 0.7836433 273.8660      7
ENSMUSG00000031389 3.409565e-12          14 0.7738291 311.9902      2
ENSMUSG00000049624 4.394295e-07          14 0.6461432 215.2914      1
ENSMUSG00000026563 4.536469e-07          13 0.5908273 296.9099      3
ENSMUSG00000091297 1.339863e-05          14 0.4915643 281.3786      1
ENSMUSG00000061740 4.782011e-10          11 0.7767617 262.2062      4
ENSMUSG00000050439 2.077879e-15          13 0.8169939 104.1613      2
ENSMUSG00000036932 6.061863e-24          12 0.9036931 248.0532      4
```

Figura 5.12: Tabla **GoodnessOfFit**.

```
> SimGLM$GlobalSummary$ReguPerGene
                   ChIP-seq-Ini miRNA-seq-Ini TF-Ini ChIP-seq-Mod miRNA-seq-Mod TF-Mod ChIP-seq-Sig miRNA-seq-Sig TF-Sig
ENSMUSG00000000078            6            74     23            6            4     13            0            2      6
ENSMUSG00000056999            0            83      0            0            4      0            0            3      0
ENSMUSG00000024873            0            79      0            0            7      0            0            3      0
ENSMUSG00000015461            2            72     11            1            2      2            0            1      1
ENSMUSG00000058135            2            80     49            1            3     31            0            0      4
ENSMUSG00000038208            1            41     50            0            1      5            0            0      2
ENSMUSG00000033985            2            58     47            2            5     28            0            0      8
ENSMUSG00000048410            5            47     46            0            1      4            0            0      2
ENSMUSG00000066036            2            37     56            0            1      2            0            0      1
ENSMUSG00000041995            5            51     29            0            1      6            0            0      3
ENSMUSG00000016018            1            58     36            1            2     21            0            1      3
ENSMUSG00000012535            3            63     45            3            3     28            0            0      5
ENSMUSG00000021583            5            74     44            3            2     30            0            0      7
ENSMUSG00000031389            3            76     22            2            3     15            0            0      2
ENSMUSG00000049624            2            58     36            0            0      5            0            0      1
ENSMUSG00000026563            2            48     66            1            0     41            0            0      3
ENSMUSG00000091297            1            35      0            1            0      0            1            0      0
ENSMUSG00000061740            1            55     14            0            0      6            0            0      4
ENSMUSG00000050439            4            60      9            1            0      4            0            0      2
ENSMUSG00000036932            1            62     26            1            4     13            0            1      3
```

Figura 5.13: Número de reguladores iniciales (–Ini), en el modelo (–Mod) y significativos (–Sig) para cada uno de los 20 genes considerados en las distintas ómicas.

pero con signo opuesto.

Los coeficientes de regresión correspondientes a cada regulador para cada grupo experimental se almacenan en las últimas columnas. Concretamente, al tener nuestra matriz de diseño experimental dos condiciones (A o B), se tiene que tendremos dos columnas finales que almacenarán los coeficientes, siendo la primera de ellas (`Group0`) la correspondiente a la primera condición y la segunda (`Group1`) la correspondiente a la segunda condición.

Las conclusiones que pueden extraerse de la tabla pueden ser las siguientes:

1. Si dos grupos experimentales tienen los mismos coeficientes, significa que el regulador tiene el mismo efecto en el gen en ambos grupos. Por ejemplo, el regulador Arnt de la ómica reguladora TF posee los mismos coeficientes de regresión ($-7{,}829 \cdot 10^{-5}$) en ambas condiciones experimentales para el gen ENSMUSG00000058135, lo que indica que este regulador tiene el mismo efecto tanto en la condición A como en la condición B.

```
> misResultados = RegulationPerCondition(SimGLM)
> misResultados
                                 gene            regulator      omic area  representative     Group0     Group1
mmu-miR-18a-5p          ENSMUSG00000000078        mmu-miR-18a-5p miRNA-seq                    2.550e-06  2.550e-06
Hmga1                   ENSMUSG00000000078                Hmga1        TF                    -1.100e-05 -1.100e-05
Mef2d                   ENSMUSG00000000078                Mef2d        TF                    -2.381e-05 -2.381e-05
mmu-miR-431-5p          ENSMUSG00000000078        mmu-miR-431-5p miRNA-seq                    0.000e+00  3.676e-06
Arid5a                  ENSMUSG00000000078                Arid5a       TF              Rfxank -1.340e-04 -1.340e-04
Rfxank                  ENSMUSG00000000078                Rfxank       TF              Rfxank -1.340e-04 -1.340e-04
Tbp                     ENSMUSG00000000078                   Tbp       TF              Rfxank  1.340e-04  1.340e-04
Zfp628                  ENSMUSG00000000078                Zfp628       TF              Rfxank  1.340e-04  1.340e-04
mmu-miR-668-5p          ENSMUSG00000056999        mmu-miR-668-5p miRNA-seq                    1.453e-05  1.453e-05
mmu-miR-6931-5p         ENSMUSG00000056999       mmu-miR-6931-5p miRNA-seq                    0.000e+00 -3.536e-05
mmu-miR-543-5p          ENSMUSG00000056999        mmu-miR-543-5p miRNA-seq                    0.000e+00 -2.643e-05
mmu-miR-7648-3p         ENSMUSG00000024873       mmu-miR-7648-3p miRNA-seq                   -1.276e-05 -1.276e-05
mmu-miR-6931-5p1        ENSMUSG00000024873       mmu-miR-6931-5p miRNA-seq      mmu-miR-6931-5p -2.059e-05  0.000e+00
mmu-miR-145a-3p         ENSMUSG00000024873       mmu-miR-145a-3p miRNA-seq      mmu-miR-6931-5p -2.059e-05  0.000e+00
mmu-miR-376c-3p         ENSMUSG00000015461       mmu-miR-376c-3p miRNA-seq                    3.343e-05  3.343e-05
Arnt                    ENSMUSG00000015461                  Arnt       TF                    3.965e-04  3.965e-04
Arnt1                   ENSMUSG00000058135                  Arnt       TF                   -7.829e-05 -7.829e-05
Sp4                     ENSMUSG00000058135                   Sp4       TF                   -1.568e-04 -1.568e-04
Ep300                   ENSMUSG00000058135                 Ep300       TF              Zfp513 -3.759e-04 -3.759e-04
Zfp513                  ENSMUSG00000058135                Zfp513       TF              Zfp513 -3.759e-04 -3.759e-04
Gfi1b                   ENSMUSG00000038208                Gfi1b        TF                    3.529e-05  3.529e-05
Tcf3                    ENSMUSG00000038208                 Tcf3        TF                    0.000e+00  4.089e-05
Myb                     ENSMUSG00000033985                  Myb        TF                    1.446e-05  1.446e-05
Hbp1                    ENSMUSG00000033985                 Hbp1        TF                    0.000e+00  1.259e-04
Arid5a1                 ENSMUSG00000033985                Arid5a       TF                Nfix  3.645e-04  3.645e-04
Nfix                    ENSMUSG00000033985                 Nfix        TF                Nfix -3.645e-04 -3.645e-04
Rfxank1                 ENSMUSG00000033985                Rfxank       TF                Nfix  3.645e-04  3.645e-04
Tbp1                    ENSMUSG00000033985                   Tbp       TF                Nfix -3.645e-04 -3.645e-04
Zfp6281                 ENSMUSG00000033985                Zfp628       TF                Nfix -3.645e-04 -3.645e-04
Zfp652                  ENSMUSG00000033985                Zfp652       TF                Nfix -3.645e-04 -3.645e-04
Runx1                   ENSMUSG00000048410                 Runx1       TF                    -7.401e-04  5.345e-04
Srebf2                  ENSMUSG00000048410                Srebf2       TF                    0.000e+00 -1.374e-04
Ppard                   ENSMUSG00000066036                 Ppard       TF                    4.521e-05  4.521e-05
Hsf1                    ENSMUSG00000041995                  Hsf1       TF                   -2.065e-04 -2.065e-04
Nfe2l2                  ENSMUSG00000041995                Nfe2l2       TF                    7.861e-05  7.861e-05
Tcf31                   ENSMUSG00000041995                 Tcf3        TF                    6.240e-05  6.240e-05
mmu-miR-376c-5p         ENSMUSG00000016018       mmu-miR-376c-5p miRNA-seq                    2.413e-05  2.413e-05
Nrf1                    ENSMUSG00000016018                 Nrf1        TF                   -2.778e-04 -2.778e-04
E2f2                    ENSMUSG00000016018                 E2f2        TF              Nfe2l1  1.096e-03  1.096e-03
Nfe2l1                  ENSMUSG00000016018                Nfe2l1       TF              Nfe2l1 -1.096e-03 -1.096e-03
Hivep2                  ENSMUSG00000012535                Hivep2       TF                    3.567e-04  3.567e-04
E2f21                   ENSMUSG00000012535                 E2f2        TF              Nfe2l1  1.728e-03  1.728e-03
Hsf11                   ENSMUSG00000012535                 Hsf1        TF                Hsf1  6.247e-04  6.247e-04
Klf13                   ENSMUSG00000012535                 Klf13       TF                Hsf1  6.247e-04  6.247e-04
Nfe2l11                 ENSMUSG00000012535                Nfe2l1       TF              Nfe2l1 -1.728e-03 -1.728e-03
Nr1h2                   ENSMUSG00000021583                Nr1h2        TF                    0.000e+00 -3.310e-04
Deaf1                   ENSMUSG00000021583                Deaf1        TF               Deaf1 -7.496e-04 -7.496e-04
Myc                     ENSMUSG00000021583                  Myc        TF                Nfix  2.508e-03  2.508e-03
Nfix1                   ENSMUSG00000021583                 Nfix        TF                Nfix  2.508e-03  2.508e-03
Stat5b                  ENSMUSG00000021583                Stat5b       TF                Nfix  2.508e-03  2.508e-03
Zfp64                   ENSMUSG00000021583                 Zfp64       TF               Deaf1 -7.496e-04 -7.496e-04
Zfp787                  ENSMUSG00000021583                Zfp787       TF                Nfix -2.508e-03 -2.508e-03
Myc1                    ENSMUSG00000031389                  Myc        TF               Satb1  4.425e-05  4.425e-05
Satb1                   ENSMUSG00000031389                Satb1        TF               Satb1 -4.425e-05 -4.425e-05
Rela                    ENSMUSG00000049624                 Rela        TF                    -1.166e-04 -1.166e-04
Tcf32                   ENSMUSG00000026563                 Tcf3        TF                    -1.854e-04 -1.854e-04
Ep3001                  ENSMUSG00000026563                Ep300        TF              Zfp513  7.109e-04  7.109e-04
Zfp5131                 ENSMUSG00000026563                Zfp513       TF              Zfp513  7.109e-04  7.109e-04
4_120588893_120589079   ENSMUSG00000091297 4_120588893_120589079  ChIP-seq                    0.000e+00 -1.645e+00
Smad3                   ENSMUSG00000061740                Smad3        TF                   -1.327e-04 -1.327e-04
Sp41                    ENSMUSG00000061740                  Sp4        TF                    5.296e-04  5.296e-04
Zfp7871                 ENSMUSG00000061740                Zfp787       TF                   -1.174e-03 -1.174e-03
Tcf33                   ENSMUSG00000061740                 Tcf3        TF                    0.000e+00  1.899e-04
Tcf34                   ENSMUSG00000050439                 Tcf3        TF                    0.000e+00  1.620e-04
Zfp7872                 ENSMUSG00000050439                Zfp787       TF                    0.000e+00 -8.099e-04
mmu-miR-543-5p1         ENSMUSG00000036932        mmu-miR-543-5p miRNA-seq                    1.605e-05  1.605e-05
Mef2d1                  ENSMUSG00000036932                Mef2d        TF                    3.454e-05  3.454e-05
E2f22                   ENSMUSG00000036932                 E2f2        TF              Nfe2l1 -1.526e-04 -1.526e-04
Nfe2l12                 ENSMUSG00000036932                Nfe2l1       TF              Nfe2l1  1.526e-04  1.526e-04
```

Figura 5.14: Tabla resumen dada por la función **RegulationPerCondition**.

2. Si uno de los coeficientes es 0, significa que el regulador no tiene ningún efecto sobre el gen en esta condición experimental. Por ejemplo, para el gen ENSMUSG00000091297 se tiene que el regulador 4_120588893_120589079 de la ómica ChIP–seq tiene un efecto significativo en la condición B (`Group1`) al ser su coeficiente de regresión no nulo ($-1{,}645$).

3. Los grupos experimentales con diferentes coeficientes y distintos de cero, indican que el regulador afecta al gen en todos estos grupos experimentales, pero la magnitud del efecto no es la misma para todos estos grupos. Por ejemplo, para el gen ENSMUSG00000048410 se tiene que el regulador Runx1 de la ómica TF influye de forma diferente en ambas condiciones al tener diferentes coeficientes de regresión. Para la primera condición (`Group0`) su coeficiente de regresión es $-7{,}401 \cdot 10^{-4}$ y para la segunda condición (`Group 1`) es $5{,}345 \cdot 10^{-4}$.

Por último, puede estudiarse la relación entre genes y reguladores de forma gráfica mediante la ejecución de la función **plotGLM**. El eje X de cada uno de los gráficos se divide entre las dos condiciones y dentro de cada condición se tienen las observaciones que corresponden a diferentes puntos de tiempo en este caso. El eje Y derecho muestra los valores de expresión para el gen (dibujados en azul en la Figura 5.15), mientras que el eje Y izquierdo indica los valores para el regulador significativo (representados en verde en la Figura 5.15). A continuación se muestran los gráficos obtenidos siguiendo el orden del código en el Anexo C.

En primer lugar, se muestra el gráfico correspondiente a la relación entre un par gen–regulador (Figura 5.15), concretamente entre los valores de expresión del gen ENSMUSG00000091297 (azul) y los valores del regulador de la ómica ChIP-seq 4_120588893_120589079 (verde).

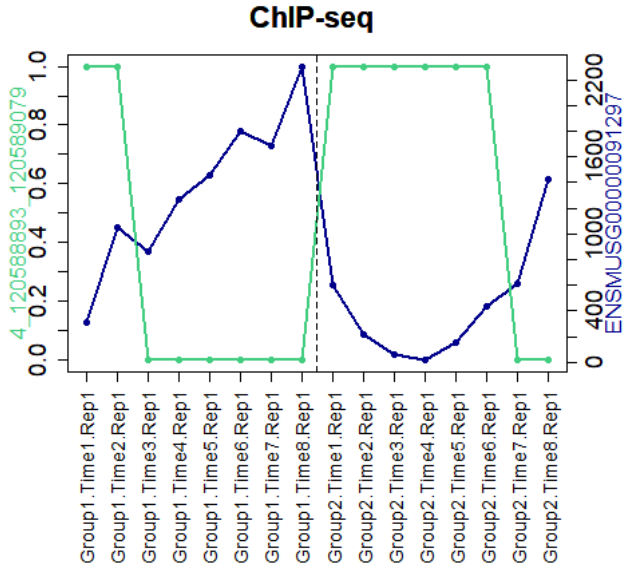

Figura 5.15: Plot del par gen–regulador: ENSMUSG00000091297 y 4_120588893_120589079. El regulador toma valores nulos en la primera condición, por lo que no tendrá efecto, mientras que en la segunda condición influye en la disminución del valor de la expresión del gen. Esto puede corroborarse a través del coeficiente de regresión ($-1{,}645$ en la segunda condición).

Otra posibilidad es estudiar un gen determinado frente a cada uno de sus reguladores significativos. Como se muestra en la Figura 5.16, se ha considerado el gen ENSMUSG00000036932 y se han obtenido cuatro gráficos, lo que indica que estos son los cuatro reguladores que regulan a este gen de manera significativa. Obsérvese que los reguladores tienen diferentes colores según la ómica a la que pertenezcan (azul TF y verde miRNA–seq), mientras que el gen se muestra en rojo.

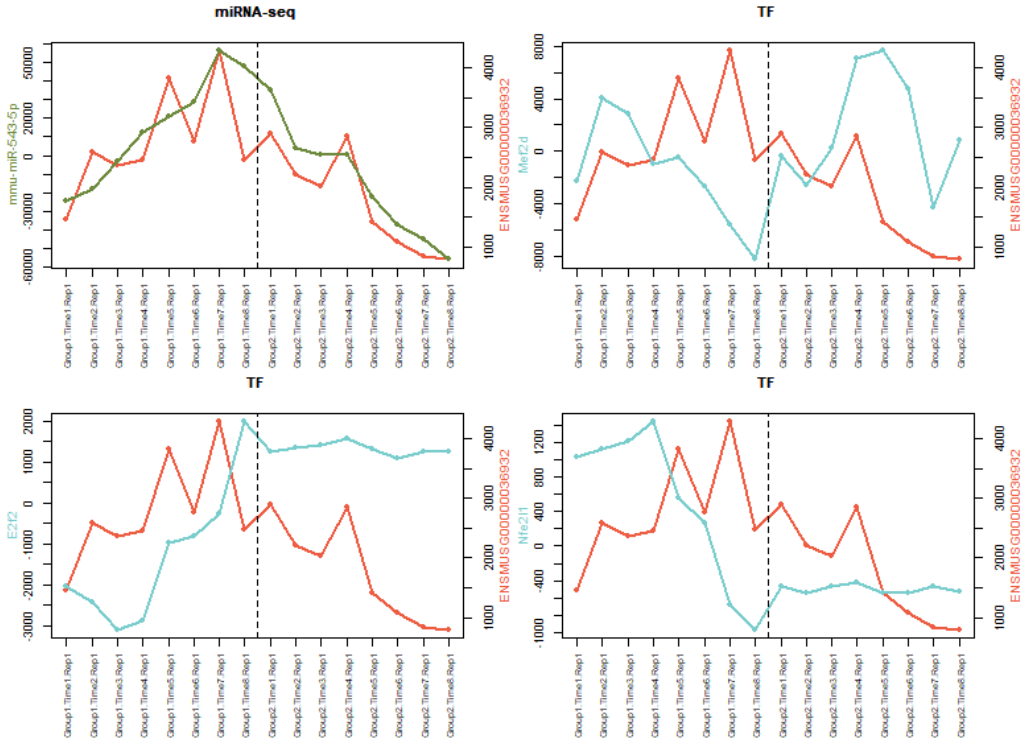

Figura 5.16: Plot del gen ENSMUSG00000036932 y todos sus reguladores significativos. Obsérvese que los reguladores pertenecientes a cada ómica se trazan en colores distintos.

También puede obtenerse de un regulador específico, tantos gráficos como a genes regule. Aquí se ha escogido el regulador Zfp513 (naranja) que regula a dos genes ENSMUSG00000058135 y ENSMUSG00000026563 (azul) y, por tanto, se obtienen dos gráficos (Figura 5.17).

Para terminar de ilustrar las funcionalidades de MORE, se muestra el gráfico para la pareja ENSMUSG00000061740 y su regulador significativo de la ómica TF, Smad3, dando los puntos de tiempo a través del parámetro de entrada **cont.var**, así como la etiqueta al eje X correspondiente para diferenciar entre ambas condiciones (Figura 5.18).

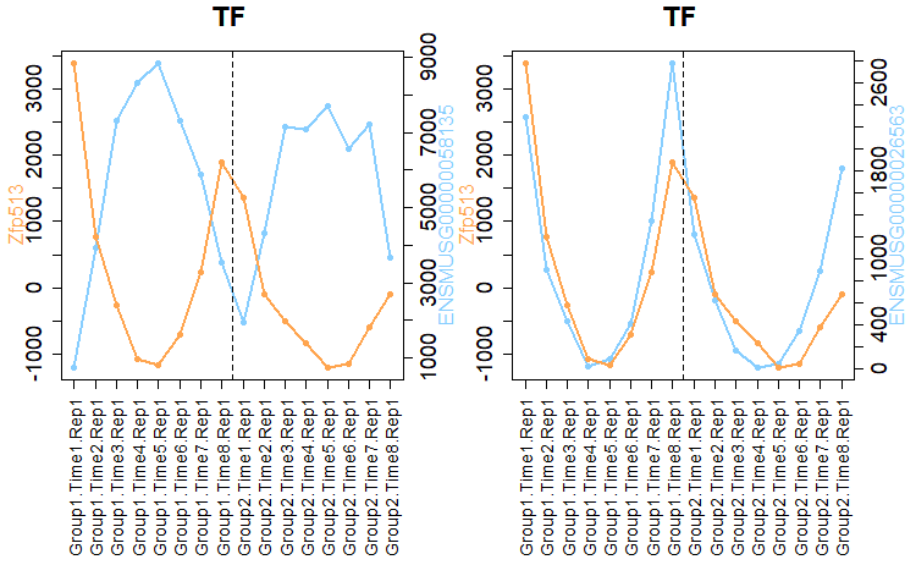

Figura 5.17: Regulador Zfp513 frente a los genes que regula (ENSMUSG00000058135 y ENSMUSG00000026563). Con el gen ENSMUSG00000058135 presenta una correlación negativa, de hecho su coeficiente de regresión es $-0{,}0003759$ en ambas condiciones, por lo que su influencia es la misma en ambas: aumentos de su valor de expresión harán que la expresión del gen disminuya y viceversa. Sin embargo, con el gen ENSMUSG00000026563 posee correlación positiva e influye de la misma manera en ambas condiciones (0,0007109 posee el mismo coeficiente de regresión en ambas), haciendo que cuando sus valores de expresión aumenten lo hagan también los valores de expresión del gen.

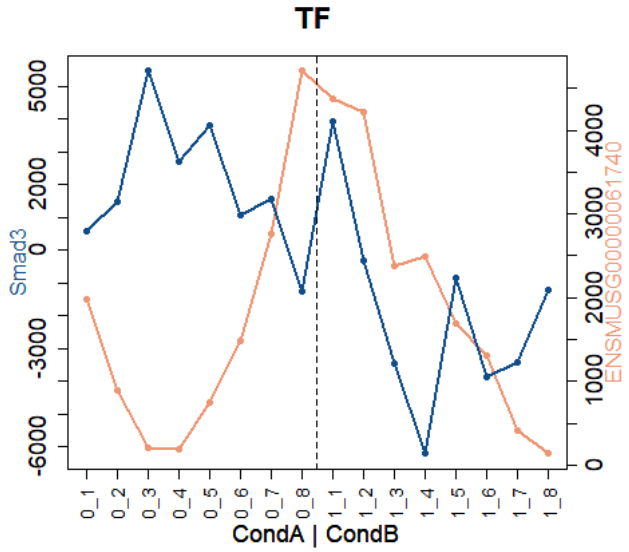

Figura 5.18: Puntos de tiempo para el gen ENSMUSG00000061740 y regulador Smad3 diferenciando entre ambas condiciones. Podría decirse que cuando el regulador toma valores de expresión más elevados, el gen toma valores de expresión más bajos, por lo que estarían correlacionados de forma negativa.

5.3. MORE en R Shiny

Una vez el paquete ha sido creado, podemos ir más allá: hay usuarios que no están familiarizados o ni siquiera conocen el lenguanje de programación R. Como el objetivo es llegar al máximo número de usuarios posible, surge la idea de construir una aplicación web de tal forma que el usuario no tenga que programar ni lidiar con el arranque de funciones y esto es posible gracias a Shiny.

5.3.1. ¿Qué es Shiny?

Shiny es una herramienta para crear fácilmente aplicaciones web interactivas para permitir a los usuarios interactuar con sus datos sin tener que manipular el código de los scripts que componen el paquete o proyecto. Shiny fue desarrollado por RStudio en 2012 y ha ido evolucionando de forma progresiva. Las principales características a destacar de Shiny son:

- La construcción de las aplicaciones mediante Shiny se hace a través del lenguaje de programación R. Sin embargo, también puede construirse mediante el uso de HTML, CSS y JavaScript [7].

- Shiny funciona en cualquier entorno de programación de R (consola de R, RStudio...) y en cualquier sistema operativo.

- El estilo de interfaz se basa, por defecto, en el *framework* Bootstrap, dándole un aspecto atractivo y amigable. Bootstrap es un conjunto de herramientas de código abierto para diseño de aplicaciones web, por lo que contiene plantillas de diseño, botones, cuadros de elección... basados en HTML y CSS, con extensión incluso a JavaScript [7] [30] .

- Las aplicaciones se desarrollan bajo la programación reactiva, esto es la incorporación de componentes activos. Por lo tanto, las aplicaciones se ejecutan a partir de las órdenes de los usuarios, convirtiéndose en aplicaciones interactivas. Por lo que un punto muy favorable es permitir a un usuario manejar los componentes de la aplicación sin la necesidad de manipular el código, con lo que podemos llegar a usuarios menos avanzados en lenguajes de programación.

- Las aplicaciones web creadas con Shiny son válidas y útiles en númerosos ámbitos: investigación, profesional o docente.

En [38] el lector puede encontrar una variedad de ejemplos de aplicaciones Shiny, así como tutoriales y ayudas para la creación de una aplicación web con Shiny.

5.3.2. Implementación de MORE en R Shiny

Para poder realizar una aplicación web con Shiny tenemos que tener instalado el paquete Shiny y el paquete de temas de Shiny. El primero es imprescindible para la creación de la aplicación, mientras que el segundo es para darle estilo a la aplicación (colores y tipo de fuente) sin necesidad de acudir a la programación CSS o HTML[1] [39]. La instalación de los paquetes puede realizarse en R mediante el comando `install.packages()`.

```
> install.packages("shiny")
> install.packages("shinythemes")
```

Una vez instalados los paquetes para Shiny, podemos crear la aplicación a través de RStudio mediante `New File > Shiny Web App...` donde nos saltará la ventana que se muestra en la Figura 5.19. En esta ventana se pide el nombre de la aplicación, el directorio donde deben crearse los archivos y la estructura de nuestra aplicación.

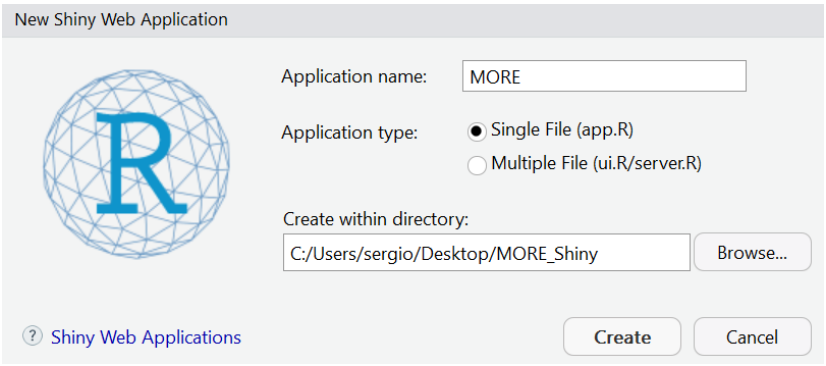

Figura 5.19: Creación de la aplicación mediante R Shiny.

5.3.3. Estructura de la aplicación Shiny

El resultado de la creación del proyecto Shiny es un directorio nuevo con el nombre que le hayamos asignado y en el que pueden aparecer uno o dos archivos, que es lo que le hemos ordenado en `Application type` en la Figura 5.19.

1. Creación de dos scripts (*Multiple File*):

   - ◇ **ui.R**: Script para desarrollar la interfaz de usuario, donde se reciben los *inputs* y se muestran los *outputs*. Aquí se define el estilo (colores, tipo de fuente, etc.) y estructura (paneles, botones, pestañas, etc.) de la aplicación web.

[1] En caso de querer introducir código CSS o HTML para la creación de estilos, el usuario debe descargarse la librería `shinyjs`.

◇ **server.R**: Script para realizar la ejecución de funciones, cálculos, análisis...

2. Otra forma alternativa es la unión de estos dos últimos scripts en uno solo (*Single File*) y que debe recibir el nombre de `app.R`. En este trabajo se ha realizado de esta manera, tal y como se muestra en la Figura 5.19. El fichero `app.R` tiene la estructura general que se presenta en la Figura 5.20.

```
library(shiny)

# UI
u<-shinyUI(TipoDePagina(

))


# SERVER
s<-shinyServer(function(input, output) {


})

shinyApp(ui = u, server = s)
```

Figura 5.20: Estructura general y básica del fichero `app.R`.

En el Anexo D puede consultarse el código completo del fichero `app.R`, donde pueden encontrarse los dos bloques definidos. Además, si quiere introducirse imágenes en la aplicación o contenidos de estilo CSS o HTML, estos deben guardarse en una carpeta adicional que debe llamarse `www`.

5.3.4. Ejecución de MORE Shiny

Una vez desarrollado el fichero `app.R` (carga de librerías y desarrollo de los bloques *ui* y *server*), estamos en disposición de arrancar la aplicación. Apretando el botón `Run App` (esquina superior derecha encuadrado rojo de la Figura 5.21), automáticamente aparecerá una ventana con la aplicación MORE, cuya interfaz de usuario se corresponde con la Figura 5.22. Una vez tenemos abierta la aplicación, el usuario debe rellenar los distintos parámetros de entrada. La aplicación MORE Shiny consta de las siguientes funcionalidades:

- **Carga de datos**. El usuario puede cargar sus datos desde su propio directorio mediante el botón `Browse...`. Los datos deben contenerse en un único fichero de extensión `.RData`. En este se almacenarán las cuatro matrices que se desean analizar (matriz de expresión génica, matriz de asociaciones, matriz de ómicas reguladoras y la matriz de diseño experimental).

- **Ejecución GetGLM y RegulationPerCondition**. Una vez el usuario rellena los *inputs* (que son los de la función **GetGLM** más el parámetro **betaTest** de la función **RegulationPerCondition**), apretando el botón `Start GLM`, la aplicación ejecuta la función **GetGLM** para generar automáticamente la tabla resumen proporcionada por la función **RegulationPerCondition**.

- **Descarga de la tabla**. La aplicación también permite al usuario la descarga de la tabla completa en un archivo CSV para su posterior manipulación. Para ello, basta apretar el botón `Download` y guardar un archivo con extensión `.csv`[2]. Además, aparecerá un botón llamado `Inputs Plot` que, clickando en dicho botón, aparecerán todos los *inputs* de la función **plotGLM**.

- **Visualización de gráficos**. Cuando se han rellenado los *inputs* de la función **plotGLM**, apretando el botón `Generate Plot` se generará debajo un gráfico que es resultado de la ejecución de la función. Cabe destacar que el usuario puede cambiar los parámetros de entrada de la función de tal manera que pueda obtener más gráficos e interactúe con sus datos (programación reactiva). Además, en el caso de que la función **plotGLM** genere más de un gráfico, si el usuario desea verlos todos, solo deberá clickar en el botón `Download` y guardar un archivo con extensión `.pdf`[3].

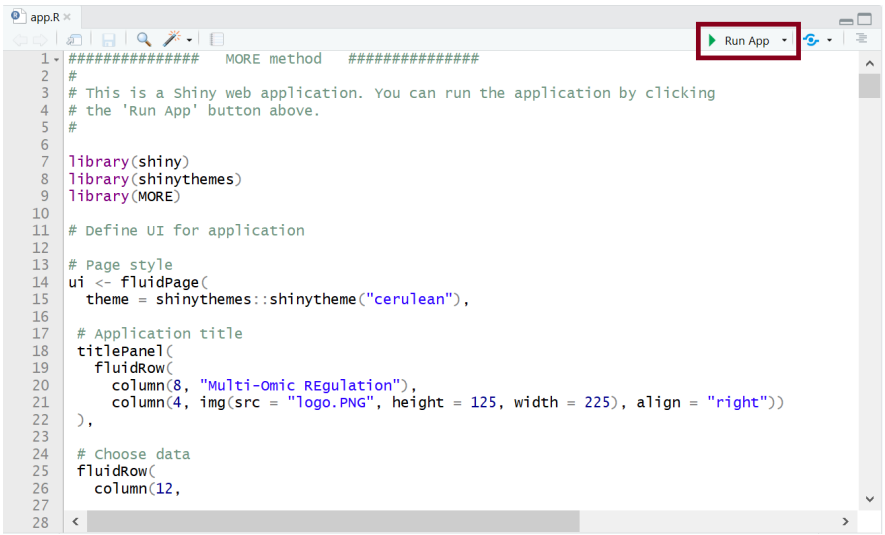

Figura 5.21: Para arrancar la aplicación hay que clickar en el botón `Run App` (encuadrado rojo).

[2]Es imprescindible que se guarde el archivo indicando la extensión: nombre.csv.
[3]Es imprescindible que se guarde el archivo indicando la extensión: nombre.pdf.

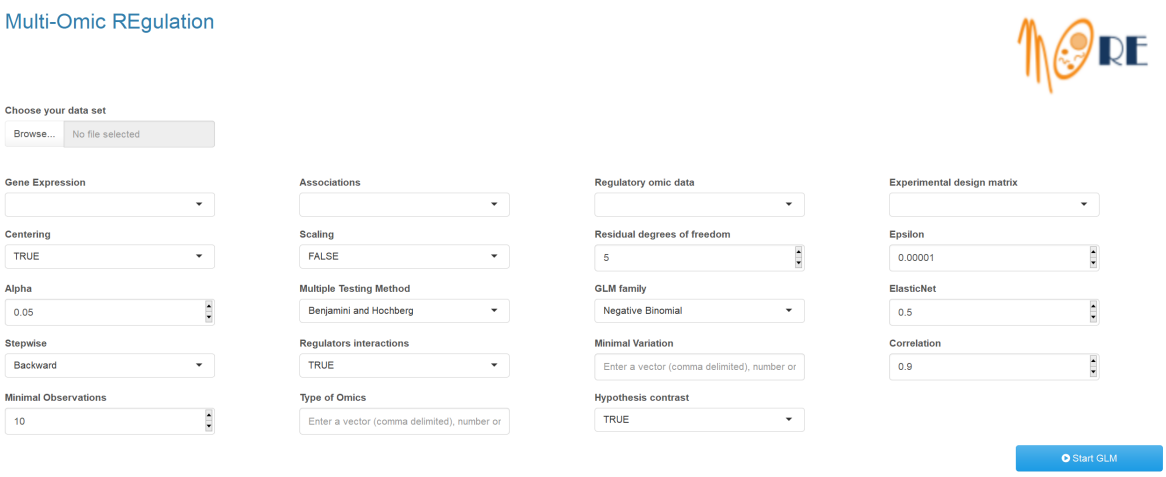

Figura 5.22: MORE Shiny.

5.3.5. Ejemplo de aplicación MORE Shiny

El fichero `app.R` está disponible en *Bitbucket* y puede ser descargado a través de https://bitbucket.org/ConesaLab/more/downloads/ (fichero MORE_Shiny.zip). Además, puede visualizarse el ejemplo mediante el siguiente vídeo que se pone a disposición de cualquier usuario https://youtu.be/SSIaeFRNsXg.

Abriendo el fichero `app.R` en RStudio y apretando el botón `Run App` aparece la ventana con la aplicación web para MORE. Primero, introducimos nuestros datos desde nuestro directorio a partir del botón `Browse...` (encuadrado azul de la Figura 5.23). MORE Shiny requiere insertar el fichero de datos (con extensión `.RData`) que contenga únicamente las matrices de datos tal y como queramos usarlas, es decir, la matriz de expresión, de asociaciones, de ómicas reguladoras y de diseño experimental, ya que desde aquí no tenemos flexibilidad para eliminar columnas o filas de *data frames* o matrices. Los datos introducidos se corresponden con los datos simulados descritos en la sección 3.6.2, solo que en la matriz de diseño experimental se han eliminado los puntos de tiempo y trabajaremos únicamente con las dos condiciones (A o B). El fichero que contiene estas cuatro matrices de datos también está disponible en la página web anterior.

Los distintos parámetros de entrada introducidos para ilustrar el funcionamiento de la aplicación MORE Shiny son los que se presentan en la Figura 5.23, donde puede observarse que ya se han cargado los datos del fichero. Cabe destacar que la mayoría de parámetros se introducen a través de un desplegable con las distintas opciones (elección de matriz de datos, elección del centrado y escalado, método Stepwise...). Los parámetros numéricos como la correlación o los grados de libertad residuales, pueden ser introducidos por teclado también. Sin embargo, los vectores deben introducirse separados únicamente por comas (por ejemplo, `1,1,0,0`). Si el

usuario desea introducir el valor NULL para algún parámetro, esto se indica dejando en blanco la correspondiente casilla.

Cuando se han introducido los datos y los valores de los parámetros de entrada en sus respectivas casillas, pinchamos en el botón `Start GLM` para arrancar la aplicación que generará automáticamente la tabla resumen que se presenta en la Figura 5.24 y que se corresponde con la salida de la función **RegulationPerCondition**. Esta tabla es fácilmente manejable en la aplicación, en el sentido de que pueden buscarse genes y/o reguladores en concreto, mostrar 10, 25, 50 o 100 entradas almacenándose el resto en páginas. Además, si el usuario quisiera manipularla puede descargarla al completo en un fichero CSV mediante el botón `Download` (encuadrado naranja en la Figura 5.24) bajo el nombre, por ejemplo, **MiTabla.csv**.

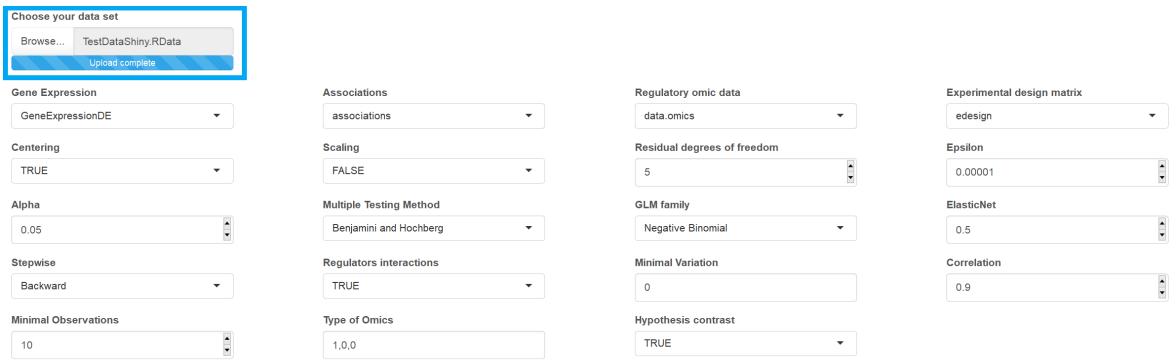

Figura 5.23: MORE Shiny: introducción de datos (encuadrado azul) y parámetros de entrada. Nótese que el vector correspondiente al parámetro `omic.type` se introduce como `1,0,0`.

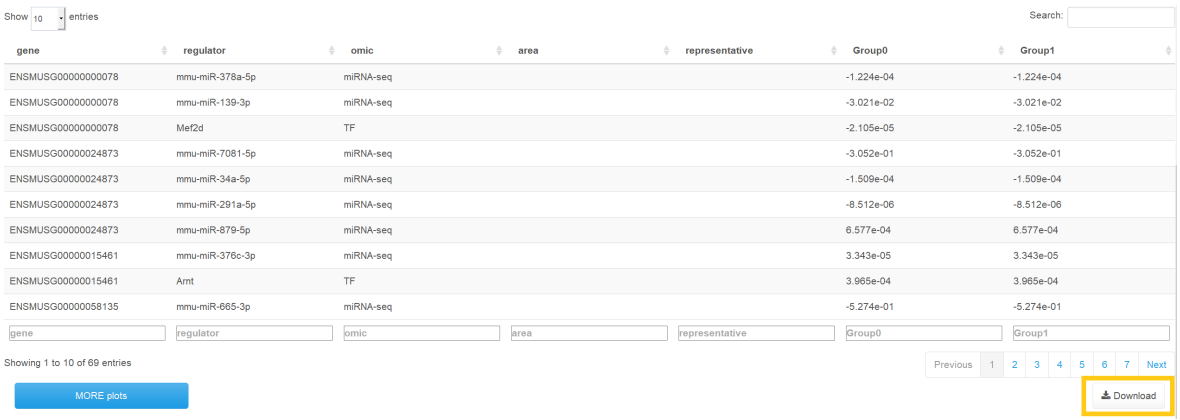

Figura 5.24: MORE Shiny: tabla resumen. Puede elegirse cuántos reguladores ver en la tabla (*Show entries*). Además, la tabla permite hacer búsquedas de genes, reguladores u ómicas que se quieran estudiar concretamente. Puede descargarse la tabla completa en formato CSV.

Ahora, si se quiere graficar genes y reguladores se tiene que pinchar en el botón `Inputs Plot` de la Figura 5.24. Automáticamente aparecerán las casillas

correspondientes a los parámetros de entrada de la función **plotGLM** (Figura 5.25). Igualmente a los parámetros de entrada anteriores, muchos de ellos pueden escogerse a través de desplegables (como los colores) y si se quiere dar el valor NULL debe dejarse la casilla en blanco (por ejemplo, *Values Regulator* será NULL al dejarlo en blanco).

Concretamente en la Figura 5.25 se han considerado el gen ENSMUSG00000000078 (naranja) y su regulador significativo Mef2d perteneciente a la ómica TF (azul). Además, se han tomado los 8 puntos de tiempo (*Values Continuous Variable*) y colocado la etiqueta *condA* | *condB* en el eje X para diferenciar entre ambas condiciones (*Title X axis*).

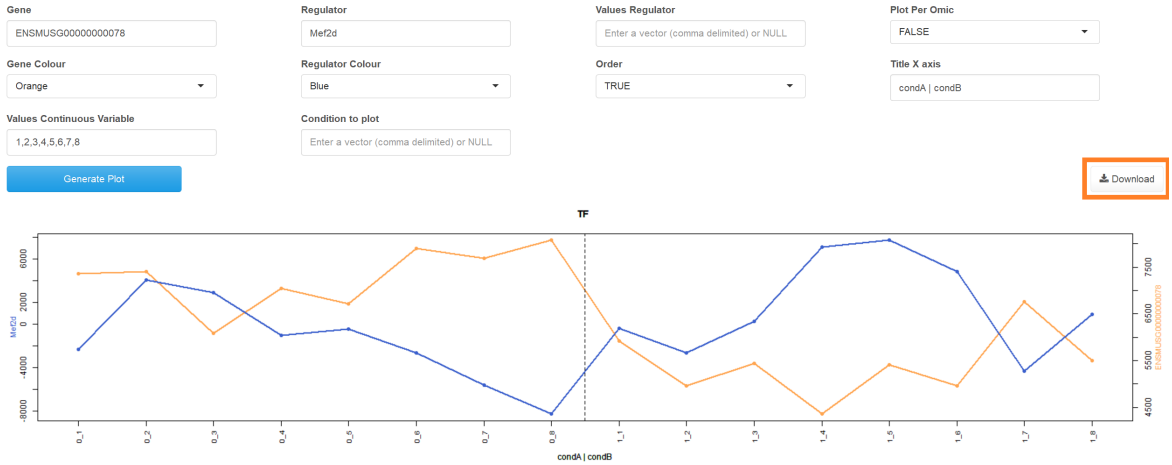

Figura 5.25: MORE Shiny: gráfico para el par gen–regulador ENSMUSG00000000078 y Mef2d. El eje Y derecho se corresponde a los valores de expresión del gen (naranja) y el eje Y izquierdo a los valores de expresión del regulador (azul). El eje X se divide entre las dos condiciones experimentales y a su vez entre los 8 puntos de tiempo. Podría decirse que poseen una correlación negativa entre ambos, ya que cuando los valores de expresión del regulador disminuyen, aumentan los del gen y viceversa.

Recordar que si quisiéramos estudiar otro gen (regulador), cambiar el color o eliminar puntos de tiempo, entre otras posibilidades, es posible y automáticamente se generará el gráfico correspondiente. Esto es gracias a la programación reactiva de Shiny, que permite cambiar los parámetros de entrada sin tener que volver a ejecutar la aplicación web desde el principio. Por otra parte, si hay más de un gráfico disponible, el usuario puede verlos todos juntos en un fichero PDF que puede guardar mediante el botón `Downoload` (encuadrado naranja Figura 5.25), por ejemplo, bajo el nombre **plots.pdf**.

Capítulo 6

Aplicación de MORE a un caso real

Como ilustración de una aplicación real del algoritmo MORE, se considerarán los datos STATegra definidos en la sección 3.6.1. Aquí solo se considerará como covariable experimental el factor condición (Ikaros y Control), descartando la variable tiempo. El resto de matrices de datos (matriz de expresión génica, matriz de ómicas reguladoras y matriz de asociaciones) serán las que se describieron anteriormente. El código con la ejecución del ejemplo STATegra puede encontrarse en el Anexo E.

Para obtener los modelos para los 5865 genes, se aplicó la función **GetGLM** sobre los datos de expresión génica centrados (pero no escalados) que no tenían valores faltantes y, dado que son datos discretos (conteos), se utilizó la distribución binomial negativa. Se exigió que los modelos tuvieran, al menos, 15 grados de libertad en los residuos, para asegurar un mínimo de potencia estadística para detectar los efectos significativos. La selección de variables se hizo mediante ElasticNet y Stepwise (opción Backward). Los valores de corte para la variación mínima de las ómicas reguladoras habían sido previamente estudiados por el grupo de Genómica de la Expresión Génica, siendo todas estas ómicas de naturaleza numérica. Así pues, los parámetros de entrada del algoritmo fueron los que se presentan en la Tabla 6.1.

Parámetro	**Valor**	**Parámetro**	**Valor**
center	TRUE	elasticnet	0.5
scale	FALSE	stepwise	backward
Res.df	15	interactions.reg	TRUE
epsilon	0.00001	min.variation	0.5,0.5,0.1,1
alfa	0.05	correlation	0.95
MT.adjust	fdr	min.obs	10
family	negative.binomial(theta = 10)	omic.type	0,0,0,0

Tabla 6.1: Parámetros de entrada para el método MORE (función **GetGLM**) aplicado a los datos reales STATegra.

Los resultados de ejecución de esta función mostraron que para 5695 de los 5865

genes iniciales (97.1 %) se obtuvo un modelo significativo (p–valor del modelo $< 0{,}05$). Para 210 de estos genes no se obtuvieron reguladores signficativos y, el resto, tienen entre 1 y 73 reguladores significativos, con una mediana de 3. Por lo tanto, contamos con un total de 5485 genes que poseen un modelo significativo y con al menos un regulador en el modelo final. Para los 5485 genes con modelo significativo, la variabilidad explicada se sitúa entre un 11.5 % y un 99.7 %, siendo la mediana 84.6 %. La Figura 6.1 muestra la relación entre el número total de reguladores obtenidos para cada uno de estos genes y su valor de devianza. Observamos que aquellos genes que presentan 2, 3, 4, 5 o más reguladores significativos poseen elevados porcentajes de devianza, aunque se observan *outliers*. Sin embargo, en aquellos genes que presentan un regulador significativo puede observarse que los porcentajes de devianza no son tan buenos, siendo esto de esperar pues solamente hay un único regulador en el modelo. Es muy probable que estos genes con un solo regulador y una devianza baja tengan otro tipo de regulación que no se ha contemplado en los experimentos ómicos realizados por el consorcio STATegra, de ahí que no consigamos explicar bien sus cambios de expresión con nuestros modelos.

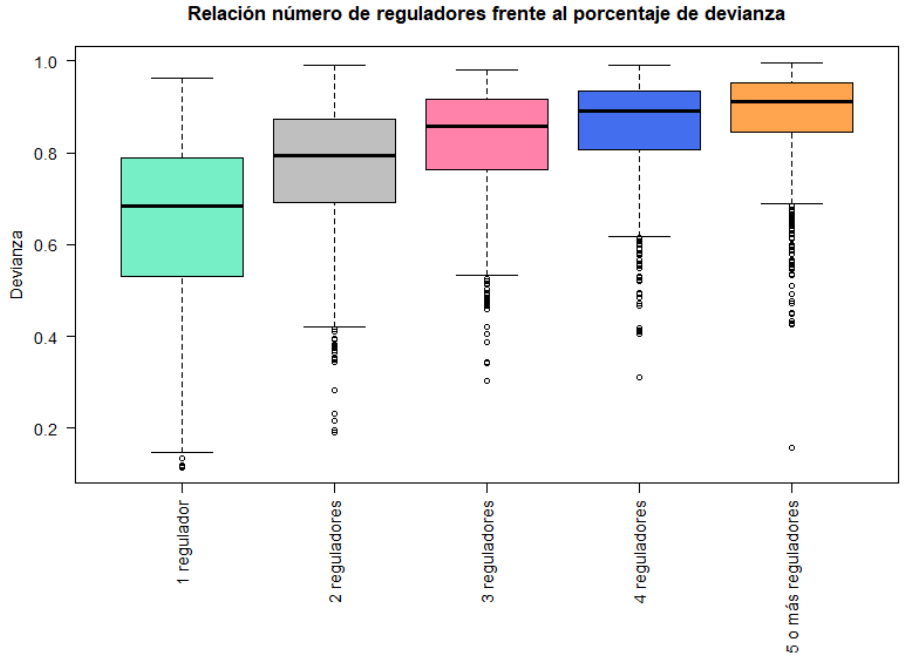

Figura 6.1: Gráfico de cajas y bigotes múltiple que representa la distribución del porcentaje de devianza explicado por el modelo según el número de reguladores significativos obtenido con MORE para los datos de STATegra.

A continuación, se aplicó la función **RegulationPerCondition** para obtener una tabla más manejable con la información de los genes y sus reguladores significativos. De esta tabla, solo se consideraron los 5485 genes indicados anteriormente.

La primera pregunta a responder, desde el punto de vista de la interpretación biológica de los resultados, es qué ómica juega un papel regulador más importante. Para ello, calculamos el porcentaje de genes de los 5865 genes estudiados inicialmente que estaban regulados por cada ómica (Figura 6.2). La barra gris nos muestra el elevado porcentaje de genes que está regulado por alguna de las ómicas estudiadas (93.5 %). La regulación por factores de transcripción (TF) es fundamental en el sistema biológico estudiado, ya que aproximadamente el 85 % de los genes están regulados por TFs. Los miRNAs le siguen en relevancia, con casi un 30 % de genes regulados. En cambio, no se observa que la accesibilidad de la cromatina (DNase-seq) o la metilación intervengan demasiado en la regulación de la expresión génica.

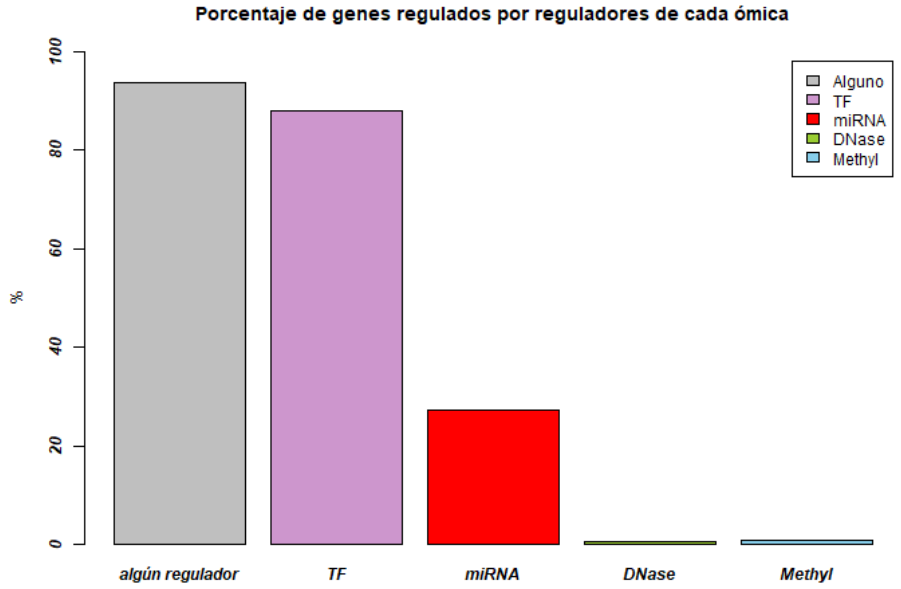

Figura 6.2: Porcentaje de genes significativamente regulados según MORE por cada ómica estudiada en los datos de STATegra.

Es interesante también tener una idea del número de reguladores de cada ómica que están regulando significativamente a cada gen. La Figura 6.3 muestra, para cada ómica, esta información. Mientras que todos los genes regulados por DNase-seq (accesibilidad de la cromatina) tienen 1 único regulador de ese tipo (región accesible a la cromatina), en metilación, cada gen puede tener hasta 80 reguladores (sitios de metilación). En miRNA-seq tenemos que más de 1000 genes solo están regulados por 1 miRNA, mientras que unos 400 se regulan por 2 miRNAS y el resto de genes por entre 3 y 9 miRNAs. La mayoría de genes están regulados por varios TFs, concretamente los números de reguladores más comunes son entre 1 y 6 TFs, siendo 2 el número de TF reguladores más común (casi 1300 genes). También se observa que unos pocos genes están regulados

por 10 ó más TFs.

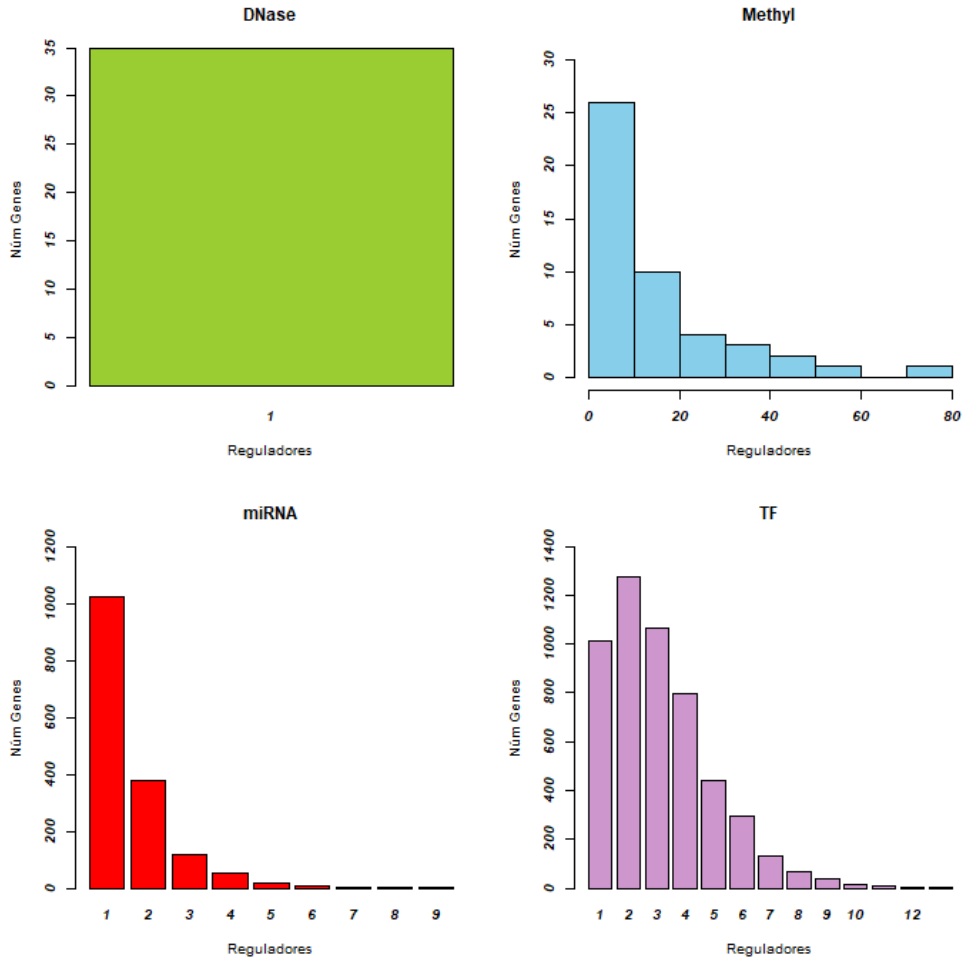

Figura 6.3: Gráfico por cada ómica reguladora que muestra cuántos genes (eje Y) tienen 1 regulador, 2 reguladores, 3 reguladores, etc. (eje X), según los resultados de MORE para los datos de STATegra.

Uno de los principales objetivos del estudio STATegra era entender el papel del factor de transcripción Ikaros en la regulación del sistema, ya que es el que desencadena la diferenciación. Por ello, no solo interesa saber cuáles eran los reguladores significativos de cada gen, sino también cuáles de ellos actúan de forma diferente en las condiciones Ikaros y Control. A partir de los resultados de la función **RegulationPerCondition**, podemos obtener esta información. De las 19356 regulaciones significativas obtenidas, buscamos aquellas con coeficientes distintos en ambas condiciones. Un 50 % de las regulaciones (9679) cumplían este requisito.

Por otra parte, existen reguladores clave en un sistema biológico, que regulan a un elevado número de genes, y se les denomina *master regulators*. En nuestro caso, los 10 reguladores más importantes por regular a un mayor número de genes se muestran en la Tabla 6.2. Como era de esperar, Ikaros (cuyo identificador es Ikzf1) resultó ser uno de

los más importantes. Myc (codificado también como ENSMUSG00000022346), el TF que regula a más genes, es regulado a su vez por Ikaros y por otro regulador interesante, Irf4, como se puede observar en la Figura 6.4. Distintas referencias bibliográficas, entre ellas [15] [25] y [43], destacan la importancia de estos factores de transcripción en la diferenciación a células B.

Regulador	Número de genes regulados	p–valor
Myc	2688	0
Ikzf1	1907	0
Irf4	1888	0
Elf1	1648	0
Sp100	1608	0
Sp4	1163	0
Mxd4	704	0
Bach2	638	3.0699e-308
Hivep2	575	5.7857e-184
Nfix	463	1.6992e-197

Tabla 6.2: Estudio STATegra, 10 reguladores más importantes según los resultados de MORE. Todos ellos son factores de transcripción (TF).

No obstante, podría suceder que los reguladores mostrados en la Tabla 6.2 no fueran tan importantes como nos han parecido a simple vista. Imaginemos que, por ejemplo, Myc estuviera caracterizado desde el principio como un regulador que puede regular potencialmente a muchos genes, mientras que el número de genes a los que potencialmente puede regular Tbx6 fuera muy reducido. Sería lógico, pues, que Myc resultara ser un regulador significativo para muchos más genes que Tbx6. El siguiente análisis se hizo para identificar aquellos reguladores que, en proporción, regulaban a más genes de forma significativa según MORE que potencial (según describe el fichero de asociaciones potenciales que se le proporciona a MORE para que genere los modelos). Para responder esta pregunta, se hizo un test de independencia para cada regulador (TF), en concreto un Test Exacto de Fisher. Este test es usado para estudiar la existencia de asociación entre dos variables cualitativas, es decir, contrastar si las proporciones de una variable son diferentes entre los valores que toma la otra variable. El Test Exacto de Fisher es usualmente aplicado en tablas de contingencia 2×2, ya que para variables cualitativas con más de dos categorías se tendría que las tablas pueden llegar a ser demasiado grandes y conducir a tiempos de ejecución muy largos. Es importante tener en cuenta que el Test Exacto de Fisher está diseñado para situaciones en las que se conocen las frecuencias marginales de filas y columnas, es decir, los totales de cada fila y columna, ya que sino el test deja de ser exacto [48]. Además, el Test Exacto de Fisher se basa en la distribución hipergeométrica, permitiendo el cálculo

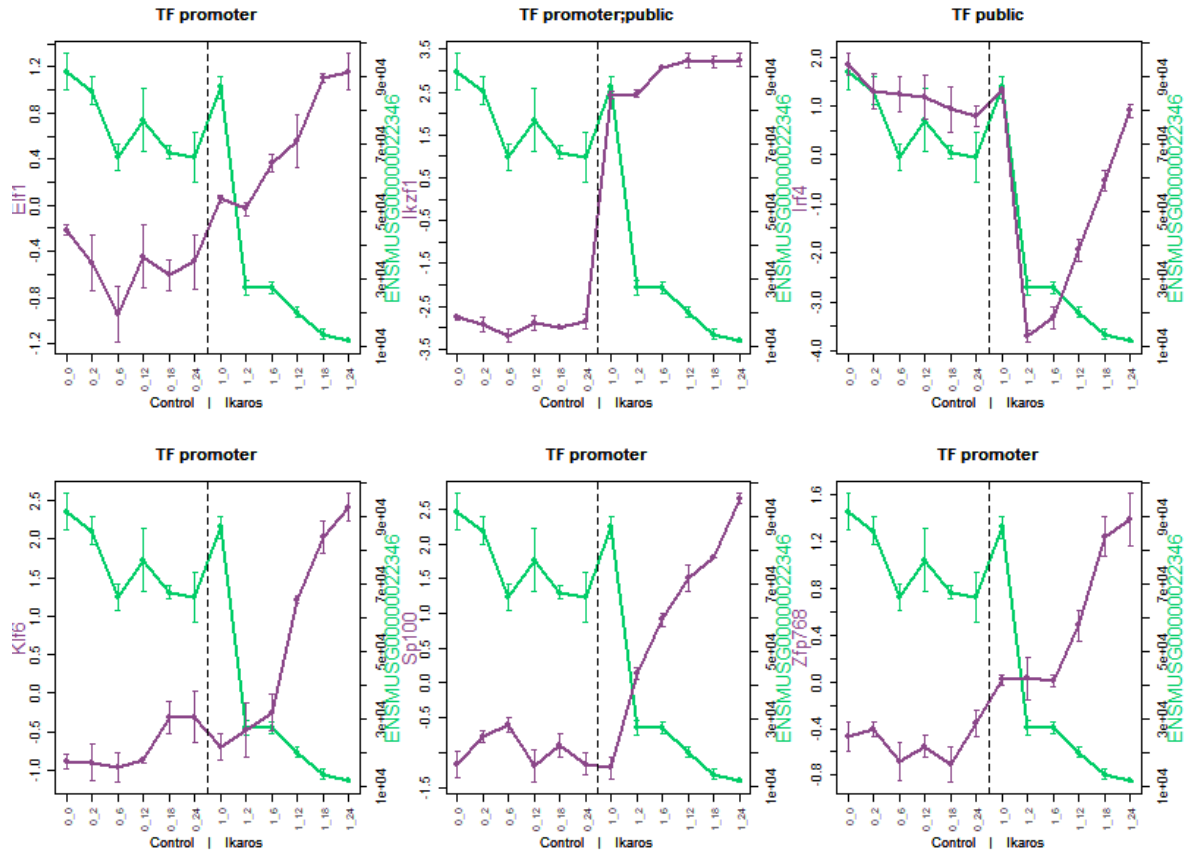

Figura 6.4: Reguladores del factor de transcripción Myc (ENSMUSG00000022346) según MORE. Cada gráfica muestra las dos condiciones experimentales en el eje X (Control e Ikaros). Dado que las observaciones se tomaron en 6 puntos de tiempo por condición, con tres réplicas cada uno, se representan en este eje los puntos de tiempo por cada condición. Las líneas representan el perfil medio del gen o regulador, tras promediar las tres réplicas de cada tiempo. El segmento en cada punto de tiempo representa la media ± error estándar. El eje izquierdo Y muestra los valores del regulador (en púrpura), mientras que el eje derecho Y muestra los valores del gen Myc (en verde).

de probabilidades exactas de obtener una determinada distribución de eventos dentro de una tabla. Bajo la hipótesis de independencia, la distribución hipergeométrica no depende de ningún parámetro desconocido y expresa la distribución de las cuatro celdas de la tabla en términos del elemento **N1S1**, ya que su valor determina los valores de las otras tres celdas dado los totales marginales (véase Tabla 6.3).

Es posible realizar el Test Exacto de Fisher en R mediante la función `fisher.test()` que se encuentra en el entorno base de R. Aquí, la hipótesis nula de independencia es equivalente a la hipótesis de que el *odd ratio* es igual a 1. Por lo tanto, para averiguar si la proporción de regulaciones significativas por parte de un regulador dado es mayor que la proporción de regulaciones de ese regulador dado en el fichero de asociaciones, se ha considerado como hipótesis alternativa que el *odd ratio* es mayor

que 1 (`alterntive = "greater"` en la función `fisher.test()`), es decir, un contraste unilateral. Por *odd ratio* se entiende el cociente de *odds*, es decir, el *odd* asociado a las regulaciones significativas en MORE por parte del regulador TF en cuestión y el *odd* correspondiente a las regulaciones potenciales (fichero de asociaciones) del regulador TF en cuestión.

Por lo tanto, se ha programado una función que, para un regulador de la ómica TF dado, construya internamente su correspondiente tabla de contingencia 2×2 a partir de los valores de las frecuencias marginales de filas y columnas de dicho regulador y del valor de la celda **N1S1** como se muestra en la Tabla 6.3. Después, aplicará el Test Exacto de Fisher a la tabla de contingencia construida teniendo en cuenta que el contraste es unilateral. La salida será el nombre del regulador y su p–valor asociado (véase Anexo E).

Ómica TF	Regulador TF	No Regulador TF	
Significativa en MORE	N1S1	N2S1 = S1 – N1S1	S1 = Número de regulaciones significativas = 16108
No Significativa en MORE	N1S2 = N1 – N1S1	N2S2 = N – N1S1 – N2S1 – N1S2	S2 = N – S1 = 158371
	N1 = Número de las N regulaciones en las que aparece el regulador	N2 = N – N1	N = Número pares de regulaciones potenciales en la ómica TF para los 5865 genes = 174479

Tabla 6.3: Tabla de contingencia 2×2. Los valores marginales se conocen a priori y en el código del Anexo E se presentan con los mismos nombres. La tabla azul es la que se pasa como argumento de entrada a la función `fisher.test()`. **N1S1** recoge el número de veces en las que el regulador en cuestión es significativo en MORE y aparece en el fichero de asociaciones. A partir de este valor se construye la tabla de contingencia 2×2 para el regulador en cuestión de la ómica TF.

Por lo tanto, para responder a la pregunta anterior se consideraron que las variables (binarias) a testar eran si la regulación gen–TF era o no significativa según MORE y si la regulación incluía o no al TF (véase la Tabla 6.3). Los p-valores de este contraste para cada TF se recogen en la tercera columna de la Tabla 6.2. Como se puede observar, todos los p-valores son significativos (incluso después de corregirlos por test múltiples mediante el método de Benjamini y Hochberg [1]). Esto resalta la relevancia de estos reguladores en el sistema biológico estudiado.

Por último, simplemente comentar que son múltiples las preguntas y análisis que se pueden hacer a partir de los resultados del MORE: análisis de enriquecimiento

funcional, generación de redes de regulación, etc. La información obtenida con MORE puede resultar muy útil al investigador para comprender mejor la regulación de la expresión génica en su sistema biológico. Aquí se han mostrado algunos ejemplos como ilustración de uso del paquete desarrollado en este trabajo, pero queda fuera del alcance del mismo realizar una interpretación biológica detallada de los resultados. También cabe recordar al lector que, en contextos biológicos como este, los análisis bioinformáticos o bioestadísticos de datos de alto rendimiento suelen servir para generar hipótesis de trabajo. Algunas de estas hipótesis habrán sido ya corroboradas por otros estudios y vienen refrendadas por la literatura pero, las novedosas, deberán ser validadadas experimentalmente en el laboratorio.

Capítulo 7

Conclusiones

La evolución de las tecnologías ómicas de alto rendimiento ha dado como resultado la aparición de nuevas fuentes de información que hacen que resulte imprescindible revisar la forma de analizar los datos ómicos o elaborar nuevas estrategias bionformáticas con tal de abordar las nuevas necesidades. En concreto, a lo largo del trabajo, se ha mostrado la problemática que envuelve la integración de datos multi–ómicos (elevado número de variables, tamaño muestral reducido...), que se ha intentado abordar de forma correcta y rigurosa mediante las diferentes herramientas y técnicas estadísticas presentadas.

El grupo de Genómica de la Expresión Génica desarrolló el algoritmo MORE, escrito en el lenguaje de programación R, con tal de estudiar la regulación de una determinada entidad biológica (por ejemplo genes o proteínas) bajo unas ciertas condiciones experimentales, aplicando para ello modelos lineales generalizados. Sin embargo, esta primera versión de MORE requería algunas mejoras con tal de cubrir mayor variedad de datos ómicos y de ofrecer más herramientas a los usuarios para interpretar los modelos obtenidos. Otra carencia de MORE es que no estaba organizado en formato de librería (o paquete) de R, ni estaba correctamente documentado, por lo que dificultaba el uso por parte de usuarios externos al grupo.

Así pues, este trabajo se ha centrado en mejorar las funcionalidades y usabilidad de MORE, como ya se estableció en los Objetivos del trabajo. En concreto, se han llevado a cabo las siguientes mejoras:

1. Se ha modificado el algoritmo para que aceptara ómicas reguladoras de tipo categórico binario, ya que en su primera versión solo permitía ómicas reguladoras numéricas.

2. Se ha facilitado una nueva función que ayuda a los usuarios a interpretar los resultados de los modelos de regresión, ya que extrae los coeficientes para cada regulador significativo en cada una de las condiciones estudiadas y realiza un

contraste de hipótesis sobre la suma de coeficientes cuando así se requiere. De esta forma, el investigador puede ver fácilmente en qué condiciones o grupos experimentales el regulador regula al gen y si la magnitud del efecto es la misma en todas estas condiciones.

3. Era objetivo primordial de este proyecto poder compartir el algoritmo con toda la comunidad científica, siendo una de las mejores vías la creación de un paquete en R que se ha puesto a disposición de los usuarios en el repositorio público *Bitbucket*.

4. Sin embargo, hay que ser conscientes de que no todos los científicos están familiarizados con el lenguaje de programación en R, por lo que también se ha presentado la programación en Shiny del algoritmo MORE para la creación de una aplicación web que resulte atractiva y facilite la interacción del usuario con sus datos.

Para completar con éxito las tareas anteriores, fue necesario, en primer lugar estudiar a fondo el código de R disponible en la primera versión de MORE, entender el funcionamiento de los modelos, las distintas estrategias de selección de variables y cómo se resumía la gran cantidad de resultados obtenidos, teniendo en cuenta que MORE genera cientos o miles de modelos de regresión en una única ejecución, dependiendo del número de entidades biológicas que se quieran estudiar. Además, hubo que limpiar el código, eliminando partes obsoletas, reorganizándolo en ocasiones para que fuera más eficiente, y corrigiendo algunos errores detectados.

Tras esta primera fase, se programaron las nuevas funcionalidades descritas en los puntos 1 y 2. Para ello, se estudiaron y valoraron distintas estrategias estadísticas para resolver los problemas planteados, como por ejemplo, la mejor medida de correlación para variables binarias o cómo resolver un contraste de hipótesis para la suma de coeficientes de regresión en modelos lineales generalizados. Implementar estas funciones implicó también la modificación de otras funciones y de los resúmenes de resultados generados por MORE, para dar cabida a estos nuevos procedimientos.

Una vez modificado todo el código base de MORE, se generó la librería de R. Como usuaria habitual de las librerías de R, resultó interesante aprender en qué consistía este proceso. La parte más laboriosa fue, realmente, documentar las funciones principales del paquete, es decir, aquellas que estarán visibles para los usuarios. En nuestro caso, habían tres funciones principales: **GetGLM**, que genera los modelos de regresión para todas y cada una de las variables ómicas estudiadas; **RegulationPerCondition**, que resume la información de las regulaciones significativas ejecutando el contraste de

hipótesis mencionado con anterioridad; y **plotGLM**, que grafica los perfiles de las entidades biológicas implicadas en la regulación (por ejemplo, un gen y uno de sus reguladores significativos). Cuando se documenta una función, se genera la información que devuelve R cuando se solicita ayuda de dicha función. Además, las funciones y su documentación se recogen (por orden alfabético) en un manual que se genera automáticamente al crear el paquete. Sin embargo, este manual no es muy práctico para aprender a utilizar un paquete, ya que no aporta ejemplos de uso ni las funciones aparecen en el orden en que deben ser usadas para llevar a cabo el análisis objeto del paquete. Por ello, se elaboró también una guía de usuario mucho más útil y completa. Se simuló un conjunto de datos multi-ómico sencillo que sirviera de ejemplo de uso, y se utilizó para ilustrar el procedimiento a seguir para aplicar MORE.

Con la idea de facilitar la aplicación de MORE a usuarios no familiarizados con el lenguaje de programación R, se decidió implementar una aplicación tipo web haciendo uso de la librería R Shiny. Esta aplicación genera una ventana en la que los usuarios pueden ejecutar las distintas funciones del paquete de forma más sencilla, simplemente eligiendo las opciones mediante desplegables, botones, cajas de diálogo, etc. El uso de esta aplicación también se describe en el manual de usuario mencionado en el párrafo anterior.

Por último, y para mostrar la utilidad y potencialidad de MORE para extraer información relevante de un gran conjunto de datos multi–ómicos, se utilizó el paquete para analizar datos experimentales (los datos STATegra). Aunque está fuera del alcance de este trabajo hacer una intepretación biológica minuciosa de los resultados obtenidos, se ofrecen algunas pinceladas para ilustrar la riqueza de información obtenida.

Como suele suceder con cualquier programa de software, esta nueva versión de MORE no será la última, ya que seguirán implementándose nuevas mejoras a nivel de funcionalidad y de aplicación web, para satisfacer las necesidades de los usuarios y mejorar las características del paquete.

Cabe subrayar que a fecha de hoy no existe una herramienta bioinformática con funcionalidades similares a MORE, por lo que es de esperar que muchos usuarios la utilicen en el futuro para analizar sus experimentos multi–ómicos y que les ayude a entender cómo se regula el funcionamiento de la célula, y por consiguiente del organismo, bajo ciertas condiciones. Este conocimiento es esencial para la búsqueda de dianas terapéuticas, de biomarcadores o para el diseño de tratamientos personalizados.

Bibliografía

[1] Benjamini, Y., and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)* (1995), 289–300.

[2] Bergsma, W. A bias-correction for cramér's v and tschuprow's t. *Journal of the Korean Statistical Society 42*, 3 (2013), 323–328.

[3] Buescher, J. M., and Driggers, E. M. Integration of omics: more than the sum of its parts. *Cancer & metabolism 4*, 1 (2016), 4.

[4] Cameron, A. C., and Windmeijer, F. A. An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics 77*, 2 (1997), 329–342.

[5] Carmona, F. Creación de paquetes de R en Windows (y Linux). Accedido a http://www.ub.edu/stat/docencia/Cursos-R/Radvanced/materials/Crear_paquetes_R.pdf, 2013.

[6] Cohen, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement 20*, 1 (1960), 37–46.

[7] Core Team. Accedido a https://getbootstrap.com/, 2018.

[8] Crick, F. Central dogma of molecular biology. *Nature 227*, 5258 (1970), 561.

[9] Divgi, D. Calculation of the tetrachoric correlation coefficient. *Psychometrika 44*, 2 (1979), 169–172.

[10] Dobson, A. J. *An introduction to generalized linear models*. Chapman and Hall/CRC, 2002.

[11] Faul, F., Erdfelder, E., Buchner, A., and Lang, A.-G. Statistical power analyses using g* power 3.1: Tests for correlation and regression analyses. *Behavior research methods 41*, 4 (2009), 1149–1160.

[12] Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software 33*, 1 (2010), 1.

[13] Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology 5*, 10 (2004), R80.

[14] Gomez-Cabrero, D., Abugessaisa, I., Maier, D., Teschendorff, A., Merkenschlager, M., Gisel, A., Ballestar, E., Bongcam-Rudloff, E., Conesa, A., and Tegnér, J. Data integration in the era of omics: current and future challenges, 2014.

[15] Heizmann, B., Kastner, P., and Chan, S. Ikaros is absolutely required for pre-b cell differentiation by attenuating il-7 signals. *Journal of Experimental Medicine 210*, 13 (2013), 2823–2832.

[16] Huang, S., Chaudhary, K., and Garmire, L. X. More is better: recent progress in multi-omics data integration methods. *Frontiers in genetics 8* (2017), 84.

[17] Huber, W., Carey, V. J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B. S., Bravo, H. C., Davis, S., Gatto, L., Girke, T., et al. Orchestrating high-throughput genomic analysis with bioconductor. *Nature methods 12*, 2 (2015), 115.

[18] Karp, G. *Cell and molecular biology: concepts and experiments*. John Wiley & Sons, 2009.

[19] Leisch, F. Creating R packages: A tutorial. (2008).

[20] Luscombe, N. M., Greenbaum, D., and Gerstein, M. What is bioinformatics? a proposed definition and overview of the field. *Methods of information in medicine 40*, 04 (2001), 346–358.

[21] Mardis, E. R. The impact of next-generation sequencing technology on genetics. *Trends in genetics 24*, 3 (2008), 133–141.

[22] Martínez-Mira, C., Conesa, A., and Tarazona, S. MOSim: Multi-Omics Simulation in R. (2018).

[23] McCullagh, P., and Nelder, J. A. *Generalized linear models*, vol. 37. CRC press, 1989.

[24] Meier, L., Van De Geer, S., and Bühlmann, P. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70*, 1 (2008), 53–71.

[25] Merkenschlager, M. Ikaros in immune receptor signaling, lymphocyte differentiation, and function. *FEBS letters 584*, 24 (2010), 4910–4914.

[26] Mulero, J. Aplicaciones interactivas diseñadas con Shiny. (2016).

[27] Nelder, J. A., and Baker, R. J. Generalized linear models. *Encyclopedia of statistical sciences 4* (2004).

[28] Ogutu, J. O., Schulz-Streeck, T., and Piepho, H.-P. Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. In *BMC proceedings* (2012), vol. 6, BioMed Central, p. S10.

[29] Palsson, B., and Zengler, K. The challenges of integrating multi-omic data sets. *Nature chemical biology 6*, 11 (2010), 787.

[30] Park, T. Accedido a https://bootswatch.com/, 2018.

[31] Pevsner, J. *Bioinformatics and functional genomics*. John Wiley & Sons, 2015.

[32] R Core Team. Writing R Extensions. Accedido a https://cran.r-project.org/doc/manuals/R-exts.html, 1999 – 2108.

[33] Real, R., and Vargas, J. M. The probabilistic basis of jaccard's index of similarity. *Systematic biology 45*, 3 (1996), 380–385.

[34] Romero Villafranca, R., and Zúnica Ramajo, L. *Métodos Estadísticos En Ingeniería*. Editorial UPV, 2005.

[35] RStudio Support. Developing Packages with RStudio. Accedido a https://support.rstudio.com/hc/en-us/articles/200486488-Developing-Packages-with-RStudio, 2017.

[36] RStudio Support. Painless package development for R. Accedido a https://www.rstudio.com/products/rpackages/devtools/, 2018.

[37] Schuster, S. C. Next-generation sequencing transforms today's biology. *Nature methods 5*, 1 (2007), 16.

[38] Shiny from RStudio. Accedido a https://shiny.rstudio.com/, 2017.

[39] Shiny Themes. Accedido a https://rstudio.github.io/shinythemes/, 2014.

[40] Tarazona Campos, S. *Statistical methods for transcriptomics: From microarrays to RNA-seq*. PhD thesis, 2015.

[41] Team, R. C. R language definition. *Vienna, Austria: R foundation for statistical computing* (2000).

[42] The STATegra Consortium. STATegra White Paper (2014). 9.

[43] Vidal, I. F., Carroll, T., Taylor, B., Terry, A., Liang, Z., Bruno, L., Dharmalingam, G., Khadayate, S., Cobb, B. S., Smale, S. T., et al. Genome-wide identification of ikaros targets elucidates its contribution to mouse b cell lineage specification and pre-b cell differentiation. *Blood* (2013), blood–2012.

[44] Wang, Z., Gerstein, M., and Snyder, M. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics 10*, 1 (2009), 57.

[45] Weissfeld, L. A., and Sereika, S. M. A multicollinearity diagnostic for generalized linear models. *Communications in Statistics-Theory and Methods 20*, 4 (1991), 1183–1198.

[46] Wickham, H. *R packages: organize, test, document and share your code*. O'Reilly Media, Inc., 2015.

[47] Wilks, S. S. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics 9*, 1 (1938), 60–62.

[48] Yates, F. Tests of significance for 2× 2 contingency tables. *Journal of the Royal Statistical Society. Series A (General)* (1984), 426–463.

[49] Ypma, T. J. Historical development of the newton–raphson method. *SIAM review 37*, 4 (1995), 531–551.

[50] Yuan, M., and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*, 1 (2006), 49–67.

[51] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*, 2 (2005), 301–320.

Anexos

Anexos A

Código de las medidas de correlación

i) Coeficiente de correlación Cramer V[1]

```
### Funcion Cramer V con la correccion de sesgo ###

CramerV = function(x){

  N = sum(x)
  chi = suppressWarnings(chisq.test(x, correct = FALSE)$statistic)
  Phi = chi/N
  R = nrow(x)
  C = ncol(x)
  Phi = max(0, Phi-((R-1)*(C-1)/(N-1)))
  CC = C-((C-1)^2/(N-1))
  RR = R-((R-1)^2/(N-1))
  CV = sqrt(Phi / min(RR-1, CC-1))
  return(CV)

}
```

ii) Índice de Jaccard

```
### Indice de Jaccard ###

jaccard = function (a,b){

        a1 = which(a == 1);
    b1 = which(b==1);
        return(length(intersect(a1,b1))/length(union(a1,b1)))

    }
```

iii) Estudio de la mejor medida de correlación

```
### Estudio de la mejor medida  de correlacion ###

prueba = matrix(c(0.92,1,1,0.83,0,1,0,0.01,
                  1,0,0.5,0.9,0.75,0,1,0.73,0,1,1,0.9,0,1,
                  0.05,0.77,0.86,0,1,0.81,0,0,0,0.2,0,1,0.99,
```

[1]Señalar que en los comentarios del código no hay acentos por problemas en la codificación.

```
                    0.88,0.87,0,1,0.9,0,1,1,0.03,0,1,0.1,0.86,
                    0.1,0,0,0.5,1,1,1,0.19,0,1,0.78,0.99,0.5,
                    0,0,0.3,1,1,1,0.62,0,1,0.84,0.8,0.2,0,0,
                    0.1,1,1,1,0.69,0,1,0.6,0.8,0.2,0,0,0.3,
                    1,1,1,0.6,0,1,0.6,0.8,0.2,0,0,0.3,1,1,1,
                    0.6,0,1,0.6,0.8,0.2,0,0,0.3,1,1,1,0.6,0,
                    1,0.6,0.8,0.2,0,0,0.3,1,1,1,0.6,0,1,0.6,
                    0.8,0.2,0,0,0.3,1,1,1,0.6,0,1,0.6,0.8,0.27,
                    0,0,0.31,1,1,1,0.06,0,1,0.46,0.48,0.2,0,0,
                    0.03,1,1,1,0.26,0,1,0.16,0.88),
                    nrow=14, byrow=T)

rownames(prueba) = c("Muestra1", "Muestra2", "Muestra3",
                     "Muestra4", "Muestra5", "Muestra6",
                     "Muestra7", "Muestra8", "Muestra9",
                     "Muestra10", "Muestra11", "Muestra12",
                     "Muestra13", "Muestra14")

colnames(prueba) = c("Regulador1", "Regulador2", "Regulador3",
                     "Regulador4", "Regulador5", "Regulador6",
                     "Regulador7", "Regulador8", "Regulador9",
                     "Regulador10", "Regulador11", "Regulador12")

miPrueba = prueba[, c(2,3,5,6,7,9,10)]

## Tablas de contingencia
tab1 = table(miPrueba[,1], miPrueba[,6]) ## espero correlacion 1
tab2 = table(miPrueba[,6], miPrueba[,7]) ## espero correlacion -1
tab3 = table(miPrueba[,4], miPrueba[,5]) ## espero una correlacion elevada

## Phi
library(psych)
phi(tab1) ## 1
phi(tab2) ## -1
phi(tab3) ## 0.68

## Correlacion tetracorica
library(psych)
tetrachoric(tab1) ## 0.89
tetrachoric(tab2) ## -0.89
tetrachoric(tab3) ## 0.8

## Coeficiente Kappa de Cohen
library(psych)
kappa1 = cohen.kappa(tab1)
kappa1$kappa ## 1

kappa2 = cohen.kappa(tab2)
kappa2$kappa ## -0.1529412

kappa3 = cohen.kappa(tab3)
kappa3$kappa ## 0.6315789

## Cramer V
CramerV(tab1) ## 1
CramerV(tab2) ## 1
CramerV(tab3) ## 0.6454972
```

```
# Indice Jaccard
jaccard(miPrueba[,1], miPrueba[,6]) ## 1
jaccard(miPrueba[,6], miPrueba[,7]) ## 0
jaccard(miPrueba[,1], miPrueba[,2]) ## 0.25. Espero un valor elevado.
jaccard(miPrueba[,2], miPrueba[,6]) ## 0.25. Espero un valor elevado.
```

Anexos B

Código de las nuevas funcionalidades

B.1. Centrado y escalado

```
### Centrado y escalado de variables continuas ###

  for (i in 1:length(omic.type)){

    if (omic.type[[i]] == 0){
      data.omics[[i]] = t(scale(t(data.omics[[i]]),
                                center = center,
                                scale = scale))
    }
  }
```

B.2. Función LowVariatFilter

```
### Funcion LowVariatFilter ###

LowVariatFilter=function(data, method, percVar, omic.type){

  SummaryRes = LV.reg = vector("list", length=length(data))
  names(SummaryRes) = names(LV.reg) = names(data)

  for (ov in names(data)) {

## Para omicas reguladoras continuas

    if (omic.type[ov] == 0) {

      if (method=="sd") {

        met=apply(data[[ov]], 1, sd, na.rm=TRUE)
        maxMet=max(met)*(percVar[ov]/100)
        myreg=met[met>maxMet]
        LV.reg[[ov]]=names(met[met<=maxMet])
        data[[ov]]=data[[ov]][names(myreg), ,drop=FALSE]

      }
```

```
      if(method=="user") {

        if (min(dim(data[[ov]])) > 0) {

          met = apply(data[[ov]],
                      1,
                      function(x) max(x, na.rm = TRUE)-min(x, na.rm = TRUE))
          maxMet = percVar[ov]
          myreg = met[met > maxMet]
          LV.reg[[ov]] = names(met[met <= maxMet])
          data[[ov]] = data[[ov]][names(myreg), , drop = FALSE]

        }
      }
    }

## Para omicas reguladoras binarias

    if (omic.type[ov] == 1) {
      if(percVar[ov] != 0){
        met = apply(data[[ov]], 1, function (x) { max(table(x)/length(x)) })
        myreg = met[met < percVar[ov]]
        LV.reg[[ov]] = names(met[met >= percVar[ov]])
        data[[ov]] = data[[ov]][names(myreg), ,drop=FALSE]
      }
    }
  }

  results = vector("list", length=2)
  results[[1]] = data
  results[[2]] = LV.reg
  names(results) = c("data", "LV.reg")

  return(results)
}
```

B.3. Función CollinearityFilter

```
### Funcion CollinearityFilter ###

CollinearityFilter = function(data,
                              reg.table,
                              correlation = 0.8,
                              omic.type) {

  row.names(reg.table) = reg.table[, "regulator"]

  for (j in 1:length(omic.type)){

    ## Omicas continuas

    if(omic.type[[j]] == 0){

      ## Reguladores con los que se cuenta
      myreg = reg.table[which(reg.table[,"omic"] ==
                              names(omic.type)[[j]]), ,drop=FALSE]
```

```
myreg = as.character(myreg[which(myreg[,"filter"] ==
                                 "Model"),"regulator"])

## Si los hay...

if (length(myreg) > 1) {

  ## Correlacion de Pearson y
  ## eleccion de los reguladores correlacionados

  mycorrelations = data.frame(t(combn(myreg,2)),
                              as.numeric(as.dist(cor(data[, myreg]))),
                              stringsAsFactors = FALSE)

  mycor = mycorrelations[abs(mycorrelations[,3]) >= correlation,]

  ## Caso 1. Solo hay una pareja: se escoge uno al azar y
  ## se asignan las etiquetas

  if (nrow(mycor) == 1) {

    correlacionados = unlist(mycor[,1:2])
    regulators = colnames(data)
    keep = sample(correlacionados, 1)
    remove = setdiff(correlacionados, keep)
    regulators = setdiff(regulators, remove)
    data = data[ ,regulators]
    index.reg = which(colnames(data) == as.character(keep))
    colnames(data)[index.reg] = paste(names(omic.type)[[j]],
                                      paste("mc", 1, sep = ""),
                                      "R",
                                      sep = "_")

    reg.table = rbind(reg.table, reg.table[keep,])
    reg.table[nrow(reg.table), "regulator"] = paste(names(omic.type)[[j]],
                                                    paste("mc", 1, sep = ""),
                                                    "R",
                                                    sep = "_")

    reg.table[keep, "filter"] = paste(names(omic.type)[[j]],
                                      paste("mc", 1, sep = ""),
                                      "R",
                                      sep = "_")

    rownames(reg.table) = reg.table[ ,"regulator"]

    if(mycor[,3] > 0){

      index = mycor[1, which(with(mycor, mycor[1 ,c(1,2)] != keep))]
      reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                         paste("mc", 1, sep = ""),
                                         "P",
                                         sep = "_")

    } else {

      index = mycor[1, which(with(mycor, mycor[1, c(1,2)] != keep))]
      reg.table[index, "filter"] = paste(names(omic.type)[[j]],
```

```
                                        paste("mc", 1, sep = ""),
                                        "N",
                                        sep = "_")
  }
}

## Caso 2. Hay mas de una pareja de reguladores

if (nrow(mycor) >= 2) {

  ## Construccion del grafo

  mygraph = graph.data.frame(mycor, directed=F)
  mycomponents = clusters(mygraph)

  ## Para cada cluster se escoge un representante
  ## y se asignan las etiquetas

  for (i in 1:mycomponents$no) {

    correlacionados = names(mycomponents$
                            membership[mycomponents$membership == i])

    regulators = colnames(data)
    keep = sample(correlacionados, 1)
    reg.remove = setdiff(correlacionados, keep)
    regulators = setdiff(regulators, reg.remove)
    data = data[ ,regulators]
    index.reg = which(colnames(data) == as.character(keep))
    colnames(data)[index.reg] = paste(names(omic.type)[[j]],
                                      paste("mc", i, sep = ""),
                                      "R",
                                      sep = "_")

    reg.table = rbind(reg.table, reg.table[keep,])
    reg.table[nrow(reg.table), "regulator"] = paste(
                                        names(omic.type)[[j]],
                                        paste("mc", i, sep = ""),
                                        "R",
                                        sep = "_")

    reg.table[keep, "filter"] = paste(names(omic.type)[[j]],
                                      paste("mc", i, sep = ""),
                                      "R",
                                      sep = "_")

    rownames(reg.table) = reg.table[ ,"regulator"]
    actual.couple = data.frame(t(combn(correlacionados,2)),
                               stringsAsFactors = FALSE)

    colnames(actual.couple) = colnames(mycorrelations[,c(1,2)])

    actual.correlation = NULL
    for(k in 1:nrow(actual.couple)){

      if (any(actual.couple[k,c(1,2)] == keep)){
        actual.correlation = rbind(actual.correlation,
                                   actual.couple[k,])
```

```
          }
        }

        actual.correlation = merge(actual.correlation[,c(1,2)],
                                   mycorrelations)

        for(k in 1:nrow(actual.correlation)){

          if(actual.correlation[k,3] > 0){
            index = as.character(
                        actual.correlation[k, which(
                                              with(actual.correlation,
                                              actual.correlation[k,c(1,2)]

            reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                               paste("mc", i, sep = ""),
                                               "P",
                                               sep = "_")

          } else {
            index = as.character(
                        actual.correlation[k, which(with(
                                              actual.correlation,
                                              actual.correlation[k,c(1,2)]

            reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                               paste("mc", i, sep = ""),
                                               "N",
                                               sep = "_")
          }
        }
      }
    }
  }

  ## Omicas binarias

} else {

  myreg = reg.table[which(
                      reg.table[,"omic"] == names(omic.type)[[j]]
                          ), ,drop=FALSE]

  myreg = as.character(
                      myreg[which(myreg[,"filter"] == "Model"),
                      "regulator"]
                      )

  ## Si hay reguladores...

  if (length(myreg) > 1){

    couple = t(combn(myreg,2))
    categorical.correlation = NULL

    ## Creacion tablas de contingencia 2x2

    for (k in 1:nrow(couple)){
```

```
  contingency.table = table(data[,couple[k,1]], data[,couple[k,2]])
  categorical.correlation = rbind(categorical.correlation,
                                  phi(contingency.table))
}

## Estudio de la correlacion

mycorrelations = data.frame(couple, categorical.correlation)
correlations = mycorrelations[abs(mycorrelations[,3]) >= 0.6,]

## Caso 1. Solo hay una pareja: se escoge un representate al
## azar y se asignan las etiquetas

if (nrow(correlations) == 1) {

  correlacionados = as.character(unlist(correlations[,1:2]))
  regulators = colnames(data)
  keep = sample(correlacionados, 1)
  remove = setdiff(correlacionados, keep)
  regulators = setdiff(regulators, remove)
  data = data[ ,regulators]
  index.reg = which(colnames(data) == as.character(keep))
  colnames(data)[index.reg] = paste(
                                    names(omic.type)[[j]],
                                    paste("mc", 1, sep = ""),
                                    "R",
                                    sep = "_")

  reg.table = rbind(reg.table, reg.table[keep,])
  reg.table[nrow(reg.table), "regulator"] = paste(
                                          names(omic.type)[[j]],
                                          paste("mc", 1, sep = ""),
                                          "R",
                                          sep = "_")

  reg.table[keep, "filter"] = paste(names(omic.type)[[j]],
                                    paste("mc", 1, sep = ""),
                                    "R",
                                    sep = "_")

  rownames(reg.table) = reg.table[ ,"regulator"]

  if(correlations[,3] > 0){

    index = as.character(correlations[1, which(
                                      with(correlations,
                                      correlations[1,c(1,2)] != keep))])

    reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                      paste("mc", 1, sep = ""),
                                      "P",
                                      sep = "_")

  } else {

    index = as.character(correlations[1, which(
                                    with(correlations,
                                    correlations[1,c(1,2)] != keep))])
```

```
    reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                       paste("mc", 1, sep = ""),
                                       "N",
                                       sep = "_")
  }
}

## Caso 2. Hay mas de una pareja de reguladores correlacionados

if (nrow(correlations) >= 2) {

  ## Construccion del grafo

  mygraph = graph.data.frame(correlations, directed=F)
  mycomponents = clusters(mygraph)

  ## Para cada cluster se escoge un representante
  ## y se asignan las etiquetas

  for (i in 1:mycomponents$no) {

    correlacionados = names(
                            mycomponents$
                            membership[mycomponents$membership == i]
                            )

    regulators = colnames(data)
    keep = sample(correlacionados, 1)
    reg.remove = setdiff(correlacionados, keep)
    regulators = setdiff(regulators, reg.remove)
    data = data[ ,regulators]
    index.reg = which(colnames(data) == as.character(keep))
    colnames(data)[index.reg] = paste(names(omic.type)[[j]],
                                      paste("mc", i, sep = ""),
                                      "R",
                                      sep = "_")

    reg.table = rbind(reg.table, reg.table[keep,])
    reg.table[nrow(reg.table),
            "regulator"] = paste(names(omic.type)[[j]],
                                 paste("mc", i, sep = ""),
                                 "R",
                                 sep = "_")

    reg.table[keep, "filter"] = paste(names(omic.type)[[j]],
                                      paste("mc", i, sep = ""),
                                      "R",
                                      sep = "_")

    rownames(reg.table) = reg.table[ ,"regulator"]
    actual.couple = as.data.frame(t(combn(correlacionados,2)))
    colnames(actual.couple) = colnames(mycorrelations[,c(1,2)])

    actual.correlation = NULL

    for(k in 1:nrow(actual.couple)){
      if (any(actual.couple[k,c(1,2)] == keep)){
```

```
                actual.correlation = rbind(actual.correlation,
                                           actual.couple[k,])
              }
            }

            actual.correlation = merge(
                                       actual.correlation[,c(1,2)],
                                       mycorrelations
                                       )

            for(k in 1:nrow(actual.correlation)){

              if(actual.correlation[k,3] > 0){

                index = as.character(
                             actual.correlation[k, which(
                                                  with(actual.correlation,
                                                  actual.correlation[k,c(1,2)]
                                                  )

                reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                                   paste("mc", i, sep = ""),
                                                   "P",
                                                   sep = "_")

              } else {

                index = as.character(
                             actual.correlation[k, which(
                                                  with(actual.correlation,
                                                  actual.correlation[k,c(1,2)]
                                                  )

                reg.table[index, "filter"] = paste(names(omic.type)[[j]],
                                                   paste("mc", i, sep = ""),
                                                   "N",
                                                   sep = "_")
              }
            }
          }
        }
      }
    }
  }

  resultado = list(RegulatorMatrix = data, SummaryPerGene = reg.table)
  rownames(resultado$SummaryPerGene) = resultado$SummaryPerGene[,"regulator"]
  return(resultado)

}
```

B.4. Función RegulationPerCondition

```
### Funcion que me ayudara a generar una primera tabla ###

GetPairs1GeneAllReg = function(gene, getGLMoutput){
```

```
  reguSignif = getGLMoutput$ResultsPerGene[[gene]]$significantRegulators

  if (is.null(reguSignif)) {

    return (NULL)

  } else {

    reguSignif = getGLMoutput$ResultsPerGene[[gene]]$allRegulators[reguSignif,]
    reguSignif = reguSignif[,c("gene", "regulator", "omic", "area", "filter")]
    return (reguSignif)

  }
}

### Funcion RegulationPerCondition ###

RegulationPerCondition = function(getGLMoutput, betaTest = TRUE){

  design = getGLMoutput$arguments$finaldesign
  Group = getGLMoutput$arguments$groups

  ## Creacion de una primera tabla y
  ## columna adicional "representative"

  genes = rownames(getGLMoutput$GlobalSummary$ReguPerGene)
  myresults = do.call("rbind", lapply(genes,
                                      GetPairs1GeneAllReg,
                                      getGLMoutput))
  colnames(myresults) = c(colnames(myresults)[1:4], "representative")
  myresults[myresults[, "representative"] == "Model", "representative"] = ""

  ## Primer caso: matriz experimental nula

  if (is.null(design)){

  ## Creo una sola columna para los coeficientes

    coeffs = matrix(1, nrow(myresults), 1)
    colnames(coeffs) = "coefficients"
    rownames(coeffs) = rownames(myresults)
    myresults = cbind(myresults, coeffs)
    myresults[grep("_N", myresults[, "representative"]), "coefficients"] = -1

  ## Para cada gen se asigna los coeficientes a
  ## sus respectivos reguladores

    for(k in unique(myresults[,"gene"])){

  ## Busqueda de representantes

      counts = grep("_R", myresults[myresults[,"gene"] == k, "representative"])
      representatives = myresults[myresults[,"gene"] == k, "regulator"][counts]
      omic.representative = myresults[myresults[,"gene"] == k,
                                      c("regulator",
                                      "representative")][counts,]

  ## Si entra en la condicion, asignara mismos coeficientes
```

```
## a los correlacionados positivamente y signo opuesto a
## los correlacionados negativamente

    if(length(representatives) != 0){

      norow.nulls = which(myresults[myresults[,"gene"] == k, "representative"]
      myresults[myresults[,"gene"] == k, "representative"][norow.nulls] =
              sub("_P",
              "",
              myresults[myresults[,"gene"] == k,
                        "representative"][norow.nulls])

      myresults[myresults[,"gene"] == k, "representative"][norow.nulls] =
                sub("_N",
                "",
                myresults[myresults[,"gene"] == k,
                          "representative"][norow.nulls])

      myresults[myresults[,"gene"] == k, "representative"][norow.nulls] =
                sub("_R",
                "",
                myresults[myresults[,"gene"] == k,
                          "representative"][norow.nulls])

      for(i in 1:length(representatives)){

        reg.rep = myresults[myresults[,"gene"] == k &
                            myresults[,"regulator"] ==
                            representatives[i],
                            "representative"]

        myresults[myresults[,"gene"] == k &
                  myresults[,"representative"] ==
                  reg.rep, "representative"] = representatives[i]
      }

      significatives = gsub("`",
                            "",
                            names(getGLMoutput$
                                  ResultsPerGene[[k]]$
                                  coefficients[2:nrow(getGLMoutput$
                                                      ResultsPerGene[[k]]$
                                                      coefficients), 1]))

      sign.glm = names(getGLMoutput$
                       ResultsPerGene[[k]]$
                       coefficients[2:nrow(getGLMoutput$
                                           ResultsPerGene[[k]]$
                                           coefficients), 1])

      for(i in 1:length(significatives)){

        if(any(significatives[i] == omic.representative[,2])){

          index.regul = rownames(omic.representative)[which(
                                                    omic.representative[,2]
                                                    == significatives[i])]
```

```
          PN = myresults[myresults[,"gene"] == k &
                         myresults[,"representative"] ==
                         index.regul, "coefficients"]

          myresults[myresults[,"gene"] == k &
                    myresults[,"representative"] == index.regul,
                    "coefficients"] = PN*getGLMoutput$
                                      ResultsPerGene[[k]]$
                                      coefficients[sign.glm[i], 1]

        } else {

          myresults[myresults[,"gene"] == k &
                    myresults[,"regulator"] == significatives[i],
                    "coefficients"] = getGLMoutput$ResultsPerGene[[k]]$
                                      coefficients[sign.glm[i], 1]

        }
      }

    } else {

    ## Asignacion sin mas de los coeficientes

      myresults[myresults[,"gene"] == k,
                "coefficients"] = getGLMoutput$
                                  ResultsPerGene[[k]]$
                                  coefficients[2:nrow(getGLMoutput$
                                                      ResultsPerGene[[k]]$
                                                      coefficients),
                                                      1]

    }
  }

} else {

## Caso en el que haya matriz experimental

## Se crean tantas columnas como condiciones experimentales

  index = unique(Group)
  names.groups = paste("Group", index, sep = "")
  conditions = matrix(0, nrow(myresults), length(names.groups))
  colnames(conditions) = names.groups
  rownames(conditions) = rownames(myresults)
  myresults = cbind(myresults, conditions)

## Para cada gen...

  for(k in unique(myresults[,"gene"])){

## Se crean tres conjuntos para evaluar a cual pertenecen los reguladores:
## con interaccion, individualmente y ambas.

    significant.regulators = getGLMoutput$
                             ResultsPerGene[[k]]$
                             significantRegulators
```

```
    model.variables = gsub("`",
                           "",
                           rownames(getGLMoutput$
                                    ResultsPerGene[[k]]$
                                    coefficients))[-1]

    interactions.model = gsub("`",
                              "",
                              rownames(getGLMoutput$
                                       ResultsPerGene[[k]]$
                                       coefficients)[grep(":",
                                                          rownames(getGLMoutput$
                                                          ResultsPerGene[[k]]$
                                                          coefficients))])

    inter.variables = unlist(strsplit(interactions.model, ":", fixed = TRUE))

    if(is.null(inter.variables)){
      inter.variables = NULL
    } else {
      inter.variables = inter.variables[seq(2,
                                            length(inter.variables),
                                            by = 2)]
    }

    variables.only = setdiff(setdiff(model.variables, interactions.model),
                             inter.variables)

    if(length(grep("Group", variables.only)) != 0){
      variables.only = variables.only[-grep("Group", variables.only)]
    }

    variables.inter.only = intersect(inter.variables, model.variables)
    variables.inter = setdiff(inter.variables, model.variables)

    for(j in 2:nrow(getGLMoutput$ResultsPerGene[[k]]$coefficients)){

      regul = unlist(strsplit(gsub("`",
                                   "",
                                   rownames(getGLMoutput$
                                            ResultsPerGene[[k]]$
                                            coefficients)[j]),
                                   ":"))
## Asignacion de coeficientes a los reguladores no correlacionados
## y representante

      if(any(regul %in% variables.only)){

        if(any(regul %in% significant.regulators)){

          myresults[myresults[,"gene"] == k &
                    myresults[,"regulator"] == regul,
                    c(names.groups)] = getGLMoutput$
                                       ResultsPerGene[[k]]$
                                       coefficients[j]

        } else {
```

```
    myresults[myresults[,"gene"] == k &
              myresults[,"representative"] == regul,
              c(names.groups)] = getGLMoutput$
                                 ResultsPerGene[[k]]$
                                 coefficients[j]

  }
}

if(any(regul %in% variables.inter)){

  if(any(regul %in% significant.regulators)){

    myresults[myresults[,"gene"] == k &
              myresults[,"regulator"] == regul[2], regul[1]] =
    myresults[myresults[,"gene"] == k &
              myresults[,"regulator"] == regul[2], regul[1]] +
              getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

  } else {

    myresults[myresults[,"gene"] == k &
              myresults[,"representative"] == regul[2], regul[1]] =
    myresults[myresults[,"gene"] == k &
              myresults[,"representative"] == regul[2], regul[1]] +
              getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

  }
}

if(any(regul %in% variables.inter.only)){

  if(any(regul %in% significant.regulators)){

    if(length(regul) == 1){

      myresults[myresults[,"gene"] == k &
                myresults[,"regulator"] == regul, c(names.groups)] =
      myresults[myresults[,"gene"] == k &
                myresults[,"regulator"] == regul, c(names.groups)] +
                getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

    } else {

      myresults[myresults[,"gene"] == k &
                myresults[,"regulator"] == regul[2], regul[1]] =
      myresults[myresults[,"gene"] == k &
                myresults[,"regulator"] == regul[2], regul[1]] +
                getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

    }

  } else {

    if(length(regul) == 1){

      myresults[myresults[,"gene"] == k &
```

```
                    myresults[,"representative"] == regul,
                                    c(names.groups)] =
            myresults[myresults[,"gene"] == k &
                    myresults[,"representative"] == regul,
                                    c(names.groups)] +
                    getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

          } else {

            myresults[myresults[,"gene"] == k &
                    myresults[,"representative"] == regul[2], regul[1]] =
            myresults[myresults[,"gene"] == k &
                    myresults[,"representative"] == regul[2], regul[1]] +
                    getGLMoutput$ResultsPerGene[[k]]$coefficients[j]

          }
        }
      }
    }

## Aplicacion del contraste

    if(betaTest == TRUE){
      myresults = BetaTest(coeffs = myresults,
                           myGene = k,
                           MOREresults = getGLMoutput)
    }

## Asignacion de coeficientes a los reguladores correlacionados segun
## el tipo de correlacion con sus respectivos representantes. Asignacion
## del nombre original del representate en la columna filter

    countsR = grep("_R",
                   myresults[myresults[,"gene"] == k,
                   "representative"])

    if(length(countsR) != 0){
      countsR = myresults[myresults[,"gene"] == k,
                          5:ncol(myresults)][countsR,]
      countsP = countsR
      countsP[,"representative"] = sub("_R", "", countsP[,"representative"])
      countsP[,"representative"] = paste(countsP[,"representative"],
                                         "_P",
                                         sep = "")

      for(l in 1:nrow(countsP)){

        myresults[myresults[,"gene"] == k &
                  myresults[,"representative"] ==
                  countsP[l,"representative"], 6:ncol(myresults)] =
      countsP[l,2:ncol(countsP)]

      }

      countsN = countsR
      countsN[,"representative"] = sub("_R", "", countsN[,"representative"])
      countsN[,"representative"] = paste(countsN[,"representative"],
                                         "_N",
```

```
                                        sep = "")

  for(l in 1:nrow(countsN)){

    myresults[myresults[,"gene"] == k &
              myresults[,"representative"] ==
              countsN[l,"representative"], 6:ncol(myresults)] =
    -countsN[l,2:ncol(countsN)]

  }

  counts = grep("_R",
                myresults[myresults[,"gene"] == k,
                "representative"])

  representatives = myresults[myresults[,"gene"] == k,
                              "regulator"][counts]

  omic.representative = myresults[myresults[,"gene"] == k,
                                  c("regulator",
                                  "representative")][counts,]

  if(length(representatives) != 0){

    norow.nulls = which(myresults[myresults[,"gene"] == k,
                        "representative"] != "")

    myresults[myresults[,"gene"] == k,
              "representative"][norow.nulls] =
    sub("_P", "", myresults[myresults[,"gene"] == k,
        "representative"][norow.nulls])

    myresults[myresults[,"gene"] == k,
              "representative"][norow.nulls] =
    sub("_N",
        "",
        myresults[myresults[,"gene"] == k, "representative"][norow.nulls])

    myresults[myresults[,"gene"] == k,
              "representative"][norow.nulls] =
    sub("_R",
        "",
        myresults[myresults[,"gene"] == k,
                          "representative"][norow.nulls])

    for(i in 1:length(representatives)){

      reg.rep = myresults[myresults[,"gene"] == k &
                          myresults[,"regulator"] == representatives[i],
                                "representative"]

      myresults[myresults[,"gene"] == k &
                myresults[,"representative"] == reg.rep,
                          "representative"] = representatives[i]

    }
  }
}
```

```
    }
  }

  myresults[, 6:ncol(myresults)] = signif(myresults[, 6:ncol(myresults)],
                                          digits = 4)

  return(myresults)

}
```

B.5. Función BetaTest

```
### Funcion BetaTest ###

BetaTest = function(coeffs, myGene, MOREresults){

  family = MOREresults$arguments$family
  alfa = MOREresults$arguments$alfa

  ## Se coge la subtabla correspondiente al gen almacenado en myGene

  mycoeffs = coeffs[coeffs[,"gene"] == myGene,]

  ## Habra suma de coeficientes si el coeficiente del grupo
  ## referencial es distinto de 0

  mytable = mycoeffs[which(mycoeffs[,6] != 0), c(1,2,5:ncol(mycoeffs))]

  ## En caso de haber coeficientes en el referencial distintos de 0...

  if(dim(mytable)[1] != 0){

  ## Comparamos entre las columas de coeficientes para
  ## quedarnos con aquellos que son distintos

    betas = NULL
    for(j in 1:nrow(mytable)){

      if(any(mytable[j,4] != mytable[j, 5:ncol(mytable)])){
        betas = rbind(betas, mytable[j,])
      }

    }

  ## Si hay suma de coeficientes, se hace el contraste

    if(!is.null(betas)){

      mySignificatives = rownames(MOREresults$
                                  ResultsPerGene[[myGene]]$
                                  coefficients)[-1]

      mySigni = gsub("`", "", mySignificatives)
      myY = MOREresults$ResultsPerGene[[myGene]]$Y[,1]
      myX = MOREresults$ResultsPerGene[[myGene]]$X
      myX = myX[, mySigni]
```

```
      for(i in 1:nrow(betas)){

        if(betas[i,"representative"] == ""){

          myRegulator = betas[i, "regulator"]

        } else {

          myRegulator = betas[i, "representative"]

        }

        aquitar = mySignificatives[grep(myRegulator, mySignificatives)]
        group = unlist(strsplit(aquitar, ":", fixed = TRUE))
        group = gsub("`",  "",group[grep("Group", group)])
        mymodel = glm(myY~., data = myX, family = family)
        myHyp = paste0(paste(aquitar, collapse = "+"), "␣=␣0")
        pvalue = try(linearHypothesis(mymodel,
                                      c(myHyp),
                                      test = "Chisq")$"Pr(>Chisq)"[2],
                     silent = TRUE)

        if(class(pvalue) == "try-error"){pvalue = NA}

  ## Vemos si el p valor es mayor que el nivel de significacion
  ## para aceptar la hipotesis nula

        if(pvalue > alfa){

          if(betas[i,"representative"] == ""){

            coeffs[coeffs[, "gene"] == myGene &
                   coeffs[, "regulator"] == myRegulator, group] = 0

          } else {

            coeffs[coeffs[, "gene"] == myGene &
                   coeffs[, "representative"] == myRegulator, group] = 0

          }
        }
      }
    }
  }

  return(coeffs)

}
```

Anexos C

Ejecución paquete MORE

```
### EJEMPLO APLICACION PAQUETE MORE ###

## Carga del paquete MORE
library(MORE)

## Carga de los datos
data("TestData")

## Ejecucion de la funcion GetGLM
OmicType = c(1, 0, 0)
names(OmicType) = names(TestData$data.omics)

SimGLM = GetGLM(GeneExpression = TestData$GeneExpressionDE,
                associations = TestData$associations,
                data.omics = TestData$data.omics,
                edesign = TestData$edesign[,-1, drop = FALSE],
                Res.df = 7,
                epsilon = 0.00001,
                alfa = 0.05,
                MT.adjust = "fdr",
                family = negative.binomial(theta = 10),
                elasticnet = 0.3,
                stepwise = "backward",
                interactions.reg = TRUE,
                correlation = 0.9,
                min.variation = NULL,
                min.obs = 10,
                omic.type = OmicType)

## Guardo los resultados
save(SimGLM, file = "SimGLM.RData")

## Outputs SimGLM
SimGLM$ResultsPerGene$ENSMUSG00000000078$coefficients
SimGLM$ResultsPerGene$ENSMUSG00000000078$allRegulators
SimGLM$GlobalSummary$GoodnessOfFit
SimGLM$GlobalSummary$ReguPerGene

## Ejecucion RegulationPerCondition
misResultados = RegulationPerCondition(SimGLM)

### PLOTS ###
```

```
## Gen y regulador
png("GenRegulador.png", width = 500, height = 450)

par(ps = 18)
par(oma = c(5,1,0,1) + 0.3)

plotGLM(GLMoutput = SimGLM,
        gene = "ENSMUSG00000091297",
        regulator = "4_120588893_120589079",
        plotPerOmic = FALSE,
        gene.col = "blue4",
        regu.col = "seagreen3")

dev.off()


## Gen
png("gen.png", width = 800, height = 600)

par(mfrow = c(2,2),
    ps = 11,
    oma = c(3,1,0,1) + 0.1)

plotGLM(GLMoutput = SimGLM,
        gene = "ENSMUSG00000036932",
        regulator = NULL,
        plotPerOmic = FALSE,
        gene.col = "tomato2")

dev.off()

## Regulador
png("regulador.png", width = 600, height = 500)

par(oma=c(5, 0, 1, 0) + 0.4)
par(ps = 18)
par(mfrow = c(1,2))

plotGLM(GLMoutput = SimGLM,
        gene = NULL,
        regulator = "Zfp513",
        plotPerOmic = FALSE,
        gene.col = "skyblue1",
        regu.col = "tan1")

dev.off()


## Valores puntos de tiempo y etiqueta eje X
png("contVarXlab.png", width = 500, height = 450)

par(ps = 18)
par(oma=c(0, 1, 0, 1) + 0.3)

plotGLM(GLMoutput = SimGLM,
        gene = "ENSMUSG00000061740",
        regulator = "Smad3",
```

```
        plotPerOmic = FALSE,
        gene.col = "lightsalmon2",
        regu.col = "dodgerblue4",
        cont.var = c(1,2,3,4,5,6,7,8),
        xlab = "CondA | CondB")

dev.off()
```

Anexos D

Código Shiny

```
###############   MORE method   ###############
#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#

library("shiny")
library("shinythemes")
library("MORE")

# Define UI for application

### Page style

ui <- fluidPage(
  theme = shinythemes::shinytheme("cerulean"),

 # Application title
 titlePanel(
   fluidRow(
     column(8, "Multi-Omic REgulation"),
     column(4, img(src = "logo.PNG",
                   height = 125,
                   width = 225),
                        align = "right"))
 ),

 # Choose data
 fluidRow(
   column(12,
          fileInput("datafile",
                    label = "Choose your data set",
                    multiple = FALSE,
                    accept = ".RData"))
 ),

 # Inputs: data sets, inputs by user...
 fluidRow(
   column(3, uiOutput("geneExpression")),
   column(3, uiOutput("associations")),
   column(3, uiOutput("dataOmics")),
   column(3, uiOutput("eDesign"))
```

```
),

fluidRow(
  column(3, selectInput("center",
                        "Centering",
                        choices = list("TRUE" = TRUE, "FALSE" = FALSE))),
  column(3, selectInput("scale",
                        "Scaling",
                        choices = list("TRUE" = TRUE, "FALSE" = FALSE),
                        selected = "FALSE")),
  column(3, numericInput("ResDF",
                         "Residual degrees of freedom",
                         value = 5,
                         min = 1)),
  column(3, numericInput("epsilon",
                         "Epsilon",
                         value = 0.00001,
                         min = 0,
                         max = 1,
                         step = 0.00001))
),

fluidRow(
  column(3, numericInput("alpha",
                         "Alpha",
                         value = 0.05,
                         min = 0,
                         max = 1,
                         step = 0.01)),
  column(3, selectInput("MTadjust",
                        "Multiple Testing Method",
                        choices = c("Benjamini and Hochberg" = "fdr",
                                    "Bonferroni" = "bonferroni",
                                    "Holm" = "holm",
                                    "Hochberg" = "hochberg",
                                    "Hommel" = "hommel",
                                    "Benjamini and Yekutieli" = "BY",
                                    "None" = "none"),
                        selected = "Benjamini and Hochberg")),
  column(3, selectInput("family",
                        "GLM family",
                        choices = c("Negative Binomial" = "negative.binomial",
                                    "Poisson" = "poisson",
                                    "Quasipoisson" = "quasipoisson",
                                    "Gaussian" = "gaussian",
                                    "Binomial" = "binomial"),
                        selected = "Negative Binomial")),
  column(3, numericInput("elasticnet",
                         "ElasticNet",
                         value = 0.5,
                         min = 0,
                         max = 1,
                         step = 0.1))
),

fluidRow(
  column(3, selectInput("stepwise",
                        "Stepwise",
```

```
                          choices = list("Backward" = "backward",
                                         "Forward" = "forward",
                                         "Two ways backward" = "two.ways.backward",
                                         "Two ways forward" = "two.ways.forward",
                                         "None" = "none"))),
    column(3, selectInput("InteractionsReg",
                          "Regulators interactions",
                          choices = list("TRUE" = TRUE, "FALSE" = FALSE))),
    column(3, textInput('vectorVariation',
                        "Minimal Variation",
                        placeholder = "Enter a vector (comma delimited),
                                       number or NULL")),
    column(3, numericInput("correlation",
                           "Correlation",
                           value = 0.9,
                           min = 0,
                           max = 1,
                           step = 0.1))
  ),

  fluidRow(
    column(3, numericInput("MinObs",
                           "Minimal Observations",
                           value = 10,
                           min = 1)),
    column(3, textInput('vectorOmicType',
                        "Type of Omics",
                        placeholder = "Enter a vector (comma delimited),
                                       number or NULL")),
    column(3, selectInput("betaTest",
                          "Hypothesis contrast",
                          choices = list("TRUE" = TRUE, "FALSE" = FALSE)))
  ),



  # Button to start the app
  fluidRow(
    column(12, actionButton("goButton",
                            label = "Start GLM",
                            class = "btn-primary",
                            width = '210px',
                            icon("play-circle")),
              align = "right")
    ),

  # Horizontal line
  tags$hr(),

  # Table from RegulationPerCondition
  fluidRow(
    dataTableOutput("tableMORE"),
    column(6, uiOutput("viewButton"), align = "left"),
    column(6, uiOutput("tableDownload"), align = "right")
  ),

  tags$hr(),
  # Plots and inputs plotGLM
```

```
fluidRow(conditionalPanel(
    condition = "input.plotConfig > 0",
    column(3, textInput("gene",
                        "Gene",
                        placeholder = "Enter a gene or NULL")),
    column(3, textInput("regulator",
                        "Regulator",
                        placeholder = "Enter a regulator or NULL")),
    column(3, textInput('reguValues',
                        "Values Regulator",
                        placeholder = "Enter a vector (comma delimited) or NULL"
    column(3, selectInput("plotPerOmic",
                          "Plot Per Omic",
                          choices = list("TRUE" = TRUE, "FALSE" = FALSE)))

    )),

fluidRow(conditionalPanel(
   condition = "input.plotConfig > 0",
   column(3, selectInput("geneCol",
                         "Gene Colour",
                         choices = list("Black" = "black",
                                        "Green" = "palegreen4",
                                        "Blue" = "royalblue3",
                                        "Red" = "tomato2",
                                        "Purple" = "darkorchid3",
                                        "Orange" = "tan1",
                                        "Yellow" = "yellow2",
                                        "Grey" = "snow3"))),
   column(3, selectInput("reguCol",
                         "Regulator Colour",
                         choices = list("Green" = "palegreen4",
                                        "Blue" = "royalblue3",
                                        "Red" = "tomato2",
                                        "Black" = "black",
                                        "Purple" = "darkorchid3",
                                        "Orange" = "tan1",
                                        "Yellow" = "yellow2",
                                        "Grey" = "snow3"))),
   column(3,selectInput("order",
                        "Order",
                        choices = list("TRUE" = TRUE, "FALSE" = FALSE))),
   column(3, textInput("xlab", "Title X axis", value = ""))

   )),

fluidRow(conditionalPanel(
   condition = "input.plotConfig > 0",
   column(3, textInput('ContVar',
                       "Values Continuous Variable",
                       placeholder = "Enter a vector (comma delimited) or NULL"))
   column(3, textInput('cond2plot',
                       "Condition to plot",
                       placeholder = "Enter a vector (comma delimited) or NULL"))

   )),
```

```
 fluidRow(conditionalPanel(
    condition = "input.plotConfig > 0",
    column(6, actionButton("generatePlot",
                           label = "Generate Plot",
                           class = "btn-primary",
                           width = '250px'),
           align = "left"),
    column(6, downloadButton('downloadPlot'), align = "right")

    )),

 fluidRow(column(12, plotOutput(outputId = "plotMORE"))),

 tags$hr()

)

# For large files
options(shiny.maxRequestSize = 30*1024*1024^2)

# Define server logic required to draw a histogram
server <- function(input, output) {

  # For choices in set data
  infile <- reactive({
    infile <- input$datafile
    if (is.null(infile)) {
      return(NULL)
    }
    objectsLoaded <- load(input$datafile$datapath, envir = .GlobalEnv)
    return(objectsLoaded)
  })

  myData <- reactive({
    df<-infile()
    if (is.null(df)) return(NULL)
    return(df)
  })

  # Select data sets
  output$geneExpression <- renderUI(
    selectInput("geneExpression", "Gene Expression", choices = myData()))

  output$associations <- renderUI(
    selectInput("associations", "Associations", choices = myData()))

  output$dataOmics <- renderUI(
    selectInput("dataOmics", "Regulatory omic data", choices = myData()))

  output$eDesign <- renderUI(
    selectInput("eDesign", "Experimental design matrix", choices = myData()))

  OutPutGLM = list()

  observeEvent(input$goButton, {

    # OmicType and min.variation names
```

```
if(nchar(input$vectorOmicType) > 1){
  omicType = as.numeric(unlist(strsplit(input$vectorOmicType, ",")))
} else if(nchar(input$vectorOmicType) == 1){
  omicType = as.numeric(input$vectorOmicType)
}
names(omicType) = names(input$dataOmics)


if(nchar(input$vectorVariation) > 1) {
  minVar = as.numeric(unlist(strsplit(input$vectorVariation, ",")))
  names(minVar) = names(input$dataOmics)

} else if(length(input$vectorVariation == 1)){
  minVar = as.numeric(input$vectorVariation)
}

MinimalVariation <- reactive({
  if (identical(input$vectorVariation, ""))
    NULL
  else
    input$vectorVariation
})

# Family GLM
myFamily <- reactive({
  if(input$family == "negative.binomial"){
     negative.binomial(theta = 10)
  } else if(input$family == "poisson"){
    poisson()
  } else if(input$family == "quasipoisson"){
    quasipoisson()
  } else if(input$family == "gaussian"){
    gaussian()
  } else if(input$family == "binomial"){
    binomial()
  }
})

# GetGLM function
OutPutGLM <<- GetGLM(GeneExpression = eval(parse(text=input$geneExpression)),
                 associations = eval(parse(text=input$associations)),
                 data.omics = eval(parse(text=input$dataOmics)),
                 edesign = eval(parse(text=input$eDesign)),
                 Res.df = input$ResDF,
                 epsilon = input$epsilon,
                 alfa = input$alpha,
                 MT.adjust = input$MTadjust,
                 family = myFamily(),
                 elasticnet = input$elasticnet,
                 stepwise = input$stepwise,
                 interactions.reg = input$InteractionsReg,
                 correlation = input$correlation,
                 min.variation = MinimalVariation(),
                 min.obs = input$MinObs,
                 omic.type = omicType)

# Table by RegulationPerCondition function
myResults = RegulationPerCondition(OutPutGLM, input$betaTest)
```

```
  output$tableMORE = renderDataTable(myResults)

  # Download table as csv file
  output$downloadTable = downloadHandler(
    filename = function(){paste("TableMORE.csv")},
    content = function(file){
      write.csv(myResults, file)
    }
  )

  output$tableDownload = renderUI({downloadButton("downloadTable",


  output$viewButton = renderUI(actionButton("plotConfig",
                                            label = "MORE plots",
                                            class = "btn-primary",
                                            width = '250px'))

})

# Plot
observeEvent(input$generatePlot, {

  # Inputs can be NULL
  regulatorValues <- reactive({
    if (identical(input$reguValues, ""))
      NULL
    else if(nchar(input$reguValues) > 1)
      ReguValues = as.numeric(unlist(strsplit(input$reguValues, ",")))
  })


  ContinuousVar <- reactive({
    if (identical(input$ContVar, ""))
      NULL
    else if(nchar(input$ContVar) > 1)
      contVar = as.numeric(unlist(strsplit(input$ContVar, ",")))
  })


  CondToPlot <- reactive({
    if (identical(input$cond2plot, ""))
      NULL
    else
      input$cond2plot
  })

  Gene <- reactive({
    if (identical(input$gene, ""))
      NULL
    else
      input$gene
  })

  Regulator <- reactive({
    if (identical(input$regulator, ""))
      NULL
    else
```

```
        input$regulator
    })

    output$plotMORE = renderPlot({
      plotGLM(OutPutGLM,
              Gene(),
              Regulator(),
              regulatorValues(),
              input$plotPerOmic,
              input$geneCol,
              input$reguCol,
              input$order,
              input$xlab,
              ContinuousVar(),
              CondToPlot())
    })

    # Save images in a pdf file
    output$downloadPlot = downloadHandler(
      filename = function(){paste0("plotsGLM.pdf")},
      content = function(file){
        pdf(file, onefile = TRUE)

        print(plotGLM(OutPutGLM,
                      Gene(),
                      Regulator(),
                      regulatorValues(),
                      input$plotPerOmic,
                      input$geneCol,
                      input$reguCol,
                      input$order,
                      input$xlab,
                      ContinuousVar(),
                      CondToPlot()))

        dev.off()
        })

})

}

# Run the application
shinyApp(ui = ui, server = server)
```

Anexos E

Ejemplo STATegra

```
###############   Ejemplo STATegra   ###############

## Carga paquete
library("MORE")

## Carga de los datos
load("DataSTATegra.RData", verbose = TRUE)

## Ejecucion de la funcion GetGLM
min.var = c(0.5, 0.5, 0.1, 1)
names(min.var) = names(data.omics)

OmicType = c(0, 0, 0, 0)
names(OmicType) = names(data.omics)

STATegraGLM = GetGLM(GeneExpression = GeneExpressionDE[1:5,],
                 associations = associations,
                 data.omics = data.omics,
                 edesign = edesign[,-1, drop = FALSE],
                 Res.df = 15,
                 epsilon = 0.00001,
                 alfa = 0.05,
                 MT.adjust = "fdr",
                 family = negative.binomial(theta = 10),
                 elasticnet = 0.5,
                 stepwise = "backward",
                 interactions.reg = TRUE,
                 correlation = 0.95,
                 min.variation = min.var,
                 min.obs = 10,
                 omic.type = OmicType)


## Guardo los resultados
save(STATegraGLM, file = "STATegraGLM.RData")

## Genes sin modelo: 13
STATegraGLM$GlobalSummary$GenesNOmodel

## 168 genes parten sin reguladores
## en el modelo (los 13 de antes entran aqui)
length(which(is.na(STATegraGLM$GlobalSummary$GoodnessOfFit[,1])))
```

```
## Filtramos estos 168 genes: 5697 genes tienen modelo
misModelos = STATegraGLM$GlobalSummary$GoodnessOfFit[
                        -which(is.na(STATegraGLM$GlobalSummary$GoodnessOfFit[,1]))

## Modelos significativos: 5695

        ## 5695, hay 2 modelos no significativos
length(which(misModelos[,1] < 0.05))
        ## Modelos significativos
misModelosSignif = misModelos[which(misModelos[,1] < 0.05),]

## Reguladores significativos

    ## 0 reguladores: 210 genes
length(which(misModelosSignif[,5] == 0))

    ## entre 1 y 73
max(misModelosSignif[,5])

    ## mediana de 3
median(misModelosSignif[,5])

## Modelos significativos con reguladores en el modelo: 5485 genes
misModelosSignif = misModelosSignif[misModelosSignif[,5] > 0, ]

## Minimo, maximo de devianza y mediana
min(misModelosSignif[,3]) ## 11.5%
max(misModelosSignif[,3]) ## 99.7%
median(misModelosSignif[,3]) ## 84.6%

## Boxplot. Figura 6.1

Regul1 = misModelosSignif[misModelosSignif[,5] == 1, 3]
Regul2 = misModelosSignif[misModelosSignif[,5] == 2, 3]
Regul3 = misModelosSignif[misModelosSignif[,5] == 3, 3]
Regul4 = misModelosSignif[misModelosSignif[,5] == 4, 3]
Regul5 = misModelosSignif[misModelosSignif[,5] >= 5, 3]

png("boxplot.png", width = 800, height = 600)
par(ps = 14)
par(oma = c(6,0,0,0) + 0.3)
boxplot(Regul1,
        Regul2,
        Regul3,
        Regul4,
        Regul5,
        las = 2,
        names = c("1 regulador",
                  "2 reguladores",
                  "3 reguladores",
                  "4 reguladores",
                  "5 o mas reguladores"),
        col = c("aquamarine2",
                "gray75",
                "palevioletred1",
                "royalblue2",
                "tan1"),
```

```
        ylab = "Devianza",
        main = "Relacion numero de reguladores frente al porcentaje de devianza")

dev.off()

## Barplot. Figura 6.2

## RegulationPerConditon
misResultados = RegulationPerCondition(STATegraGLM)
length(unique(misResultados[,"gene"])) ## 5487

## Quitamos los dos modelos no significativos
noSignificativos = misModelos[misModelos[,1] > 0.05,]
misResultados = misResultados[-c(which(misResultados[,"gene"] ==
                                 rownames(noSignificativos)[1]),
                                 which(misResultados[,"gene"] ==
                                 rownames(noSignificativos)[2])),]

        ## 5485 significativos y con reguladores
length(unique(misResultados[, "gene"]))

## Numero de genes regulados por cualquier omica

any = (5485/5865)*100
TF = (length(unique(misResultados[misResultados[,"omic"] ==
                  "TF", "gene"]))/5865)*100
miRNA = (length(unique(misResultados[misResultados[,"omic"] ==
                  "miRNA", "gene"]))/5865)*100
DNase = (length(unique(misResultados[misResultados[,"omic"] ==
                  "DNase", "gene"]))/5865)*100
Methyl = (length(unique(misResultados[misResultados[,"omic"] ==
                  "Methyl", "gene"]))/5865)*100

png("barplot.png", width = 700, height = 500)
barplot(c(any, TF, miRNA, DNase, Methyl),
        col = c("gray75",
                "plum3",
                "red1",
                "olivedrab3",
                "skyblue"),
        names.arg = c("algun regulador",
                      "TF",
                      "miRNA",
                      "DNase",
                      "Methyl"),
        ylab = "%",
        ylim = c(0, 100),
        legend.text = c("Alguno",
                        "TF",
                        "miRNA",
                        "DNase",
                        "Methyl"),
        main = "Porcentaje de genes regulados por reguladores de cada omica",
        font.axis = 4)
dev.off()

## Figura 6.3
misRegulacionesTF = misResultados[misResultados[,"omic"] == "TF", "gene"]
```

```
misRegulacionesmiRNA = misResultados[misResultados[,"omic"] == "miRNA", "gene"]
misRegulacionesDNase = misResultados[misResultados[,"omic"] == "DNase", "gene"]
misRegulacionesMethyl = misResultados[misResultados[,"omic"] == "Methyl", "gene"]

png("Figura3.png", width = 700, height = 700)
par(mfrow = c(2,2))
barplot(table(table(misRegulacionesDNase)),
        main = "DNase",
        xlab = "Reguladores",
        ylab = "Num␣Genes",
        col = "olivedrab3",
        font.axis = 4)

hist(table(misRegulacionesMethyl),
        main = "Methyl",
        xlab = "Reguladores",
        ylab = "Num␣Genes",
        col = "skyblue",
        ylim = c(0, 30),
        font.axis = 4)

barplot(table(table(misRegulacionesmiRNA)),
     main = "miRNA",
     xlab = "Reguladores",
     ylab = "Num␣Genes",
     col = "red1",
     font.axis = 4,
     ylim = c(0, 1200))

barplot(table(table(misRegulacionesTF)),
     main = "TF",
     xlab = "Reguladores",
     ylab = "Num␣Genes",
     col = "plum3",
     ylim = c(0, 1400),
     font.axis = 4)
dev.off()

## Pares gen-regulador son distintos en las condiciones
ParGenRegulador = misResultados[misResultados[,6] !=
misResultados[,7], c("gene",
                     "regulator",
                     "Group0",
                     "Group1")]

## Reguladores Top Ten
myRegulators = table(misResultados[,"regulator"])
myRegulators = myRegulators[order(myRegulators, decreasing = TRUE)][1:10]

library(xtable)
xtable(myRegulators[order(myRegulators, decreasing = TRUE)][1:10])

## Figura 6.4: plotGLM
png("plotGLM.png", width = 800, height = 600)
par(ps = 14)
par(mfrow = c(2,3))
plotGLM(STATegraGLM,
        gene = "ENSMUSG00000022346",
```

```
        plotPerOmic = FALSE,
        gene.col = "springgreen3",
        order = FALSE,
        cont.var = rep(c(0,2,6,12,18,24), each = 3),
        xlab = "Control␣␣␣|␣␣␣Ikaros")
dev.off()

## Funcion TestFisher para los reguladores de la omica TF
TestFisher = function(regulador){

  ## Pares de reguladores potenciales en TF para los 5865 genes
  misRegulaciones = associations$TF[which(associations$TF[,1] %in%
                                    rownames(GeneExpressionDE)), ]
  misAsociaciones = as.vector(apply(misRegulaciones[,1:2],
                                    1,
                                    paste,
                                    collapse = "_"))
  N = length(misAsociaciones)

  ## Regulaciones significativas
  RegulacionesSingif = misResultados[misResultados[, "omic"] == "TF", ]
  RegulacionesSingif = as.vector(apply(RegulacionesSingif[,1:2],
                                       1,
                                       paste, collapse = "_"))
  S1 = length(RegulacionesSingif)

  ## Num de las N regulaciones en las que aparece el regulador
  ReguladorSignif = as.vector(misAsociaciones[grep(regulador, misAsociaciones)])
  N1 = length(ReguladorSignif)

  ## Montamos la tabla
  N1S1 = intersect(RegulacionesSingif, ReguladorSignif)
  N1S1 = length(N1S1)

  N2S1 = S1 - N1S1
  N1S2 = N1 - N1S1
  N2S2 = N - N1S1 - N2S1 - N1S2

  ## Aplicacion del test de Fisher
  miTabla = matrix(c(N1S1, N2S1, N1S2, N2S2), byrow = TRUE, nrow = 2)
  colnames(miTabla) = c(paste("TF", regulador, sep = "_"),
                        paste("No_TF", regulador, sep = "_"))
  rownames(miTabla) = c("Significativo", "NoSignificativo")

  Fisher = fisher.test(miTabla, alternative = "greater")

  misPvalores = matrix(c(regulador,
                         signif(Fisher$p.value, 5)),
                       byrow = TRUE, nrow = 1)
  colnames(misPvalores) = c("Regulador", "Pvalor")
  misPvalores = as.data.frame(misPvalores)
}


## Aplicacion de la funcion TestFisher al top ten de reguladores

myRegulators = table(misResultados[,"regulator"])
myRegulators = myRegulators[order(myRegulators, decreasing = TRUE)][1:10]
```

```
ResultadosFisher = NULL
for(i in 1:length(names(myRegulators))){
  myResult = TestFisher(names(myRegulators)[i])
  ResultadosFisher = rbind(ResultadosFisher, myResult)
}
```