Diseño de aplicación web para el análisis de expresión de archivos en formato mirGFF3

Máster universitario en Bioinformática y bioestadística

Área 1: Análisis de secuencias de RNA regulador

Autor: Adrián Corral Bondía

Tutora: Lorena Pantano Rubiño

Carles Ventura Royo

Junio 2019

Resumen

En el presente proyecto nos hemos centrado en la problemática que envuelve el estudio de los micro-RNA (miRNA)[19] definidos como pequeñas moléculas de ARN monocatenario con una longitud entre 20 y 27 nucleótidos y la capacidad de regular la expresión de otros genes post-transcripcionalmente, así como de sus isomiR, variaciones en las secuencias de dichos miRNA.

Hoy en día el número de aplicaciones que producen archivos de análisis de estas moléculas son múltiples: seqbuster, PROST! o sRNAbench por citar algunos ejemplos. Esto da lugar a una enorme diversidad de formatos que dificultan el proceso de aunar resultados y líneas de investigación.

Para tratar de solventar este problema el proyecto mirTOP se ha centrado en el establecimiento de un formato unitario que facilite la agrupación y análisis de resultados de las diferentes herramientas. MirTOP[3] propone mirGFF3[3], derivado de GFF3 (General Feature Format) [6]y que ofrece un compromiso óptimo entre las bondades de las diferentes aplicaciones.

Con la intención de apoyar el empleo de este formato hemos desarrollado una aplicación web que aporte una interfaz visual para el análisis de expresión génica en archivos mirGFF3, así como de los metadatos asociados al mismo. Para su elaboración hemos empleado Rstudio[1] , IDE para lenguaje R, el cual gracias a su fuerte enfoque hacia el análisis estadístico, series temporales y algoritmos de clasificación constituye una base técnica ideal para el proyecto. De la misma manera la herramienta se ha beneficiado de las múltiples librerías que R posee, entre las cuales destacaremos los paquetes Bioconductor[8] y Shiny[9]. Esta última permite el diseño de aplicaciones web convirtiendo código R en HTML5/CSS/JavaScript.

El último paso del proyecto ha consistido en probar la aplicación finalizada con un conjunto de datos lo suficientemente grande como para poder ponderar tanto el rendimiento como los límites del diseño. Con esto en mente nos hemos referido el proyecto *Reproducibility of high throughput mRNA and small RNA sequencing across laboratories.* de *Peter A C 't Hoen et. al.* [18] investigación bajo el proyecto GEUVADIS (Genetic European Variation in Disease) consistente en más de 465 muestras de miRNAcon un total de 167,4 MB en información.

Palabras Clave: miRNA, Shiny, R, mirGFF3, mirTOP, GEUVADIS.

Abstract

In this project we have focused on the problems related to the study of micro-RNA (miRNA) defined as small single-stranded RNA molecules with a length between 20 and 27 nucleotides and the ability to regulate the expression of other genes post-transcriptionally, as well as their isomiR, defined as variations in the sequences of such miRNA.

Nowadays the number of applications capable of analyzing miRNA files are many: seqbuster, PROST! or sRNAbench to name a few.

This results in an enormous diversity of formats that hinder the process of combining results and investigation lines. To try to solve this problem, the mirTOP project aims to establish a unitary format that facilitates the gathering and analysis of the results from the different tools. MirTOP proposes mirGFF3, modified format derived from GFF3 (General Feature Format), which offers an optimal compromise between the benefits of the different tools.

With the intention of supporting the use of this format we have developed a web application that provides a visual interface for the analysis of gene expression in mirGFF3 files, as well as the metadata associated to it. We used Rstudio, IDE for R language, which thanks to its major focus on statistical analysis, time series and classification algorithms constitutes an ideal technical base for the project. Another of the advantages of using this platform are the multiple libraries with functionalities like Shiny and Bioconductor.

Shiny is a package for web applications design that converts the R code into HTML5/CSS/JavaScript.

The final step of the project has been testing the finished application with a large data set, big enough so we can check both performance and design limits. With this in mind we have referred to the project *Reproducibility of high throughput mRNA and small RNA sequencing across laboratories.* by Peter A C 't Hoen et. al., investigation under the GEUVADIS (Genetic European Variation in Disease) project with more than 465 miRNA samples and a total of 167.4 MB of information.

Key words:miRNA, Shiny, R, mirGFF3, mirTOP, GEUVADIS

Índice General

Índice de figuras

Índice de tablas

1

Introducción

El avance tecnológico sobretodo en el campo de la secuenciación genética ha traído un incremento exponencial en el volumen de datos de los que disponemos para investigación, esto es especialmente cierto en el campo del micro-RNA.

1.1 Micro-RNA

Los micro-RNA[20] o miRNAs son moléculas de RNA[19] transcritas a partir de genes de DNA dentro de la propia célula, pero que no son traducidas a proteínas. Se descubrieron originalmente al estudiar el desarrollo del nemátodo *Caenorhabditis elegans*. Entre sus generalidades destacamos que son monocatenarias, con un tamaño de entre 20 y 27 nucleótidos y con la capacidad de regular la expresión de otros genes post-transcripcionalmente. Esta regulación normalmente da lugar a una disminución en la expresión del gen codificado ya que se unen a regiones complementarias de mRNA[20] dando lugar a una inhibición de la traducción o un incremento en la tasa de degradación del mismo aunque en menor medida también los hay con función promotora y activadora .
En diversas investigaciones se ha observado que los miRNAs[19] participan en un enorme espectro de actividades biológicas que incluyen diferenciación, proliferación y muerte celular hasta el punto que su exceso o falta está asociado a diversos cánceres y problemas cardíacos y aunque representan solo el 2-3% del genoma en humanos pueden influir en la regulación de más del 60% de sus genes pudiendo darse que un mRNA sea regulado por varios miRNAs.
La síntesis de miRNA sigue una serie de pasos que dan lugar a diferentes estadios intermedios del mismo:

1. En el interior del núcleo celular por medio de la polimerasa II la secuencia de DNA que da origen al miRNA transcribe precursores de varios cientos de pares de bases formando una horquilla al poseer una región complementaria a la que codifica el miRNA y formando un RNA bicatenario denominado **micro-RNA primario[** el cual presenta una longitud de cientos de pares.

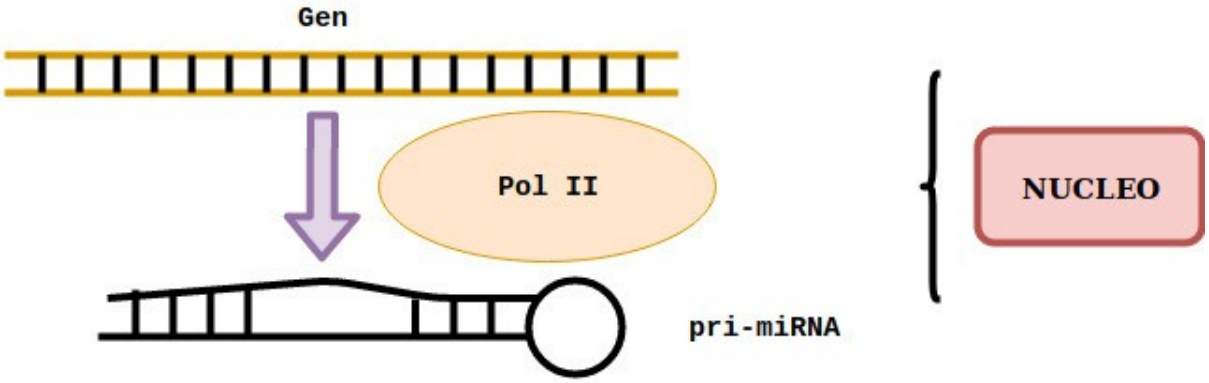


Figura 1.1 : Primer paso de la síntesis del miRNA

2. En un segundo estadío las ribonucleasas Drosha y DGCR8 cortan la base de la horquilla de los precursores en los extremos 3'y 5' lo que da lugar al **pre-micro-RNA** el cual ya acorta su longitud a unos 60-70 nucleótidos.

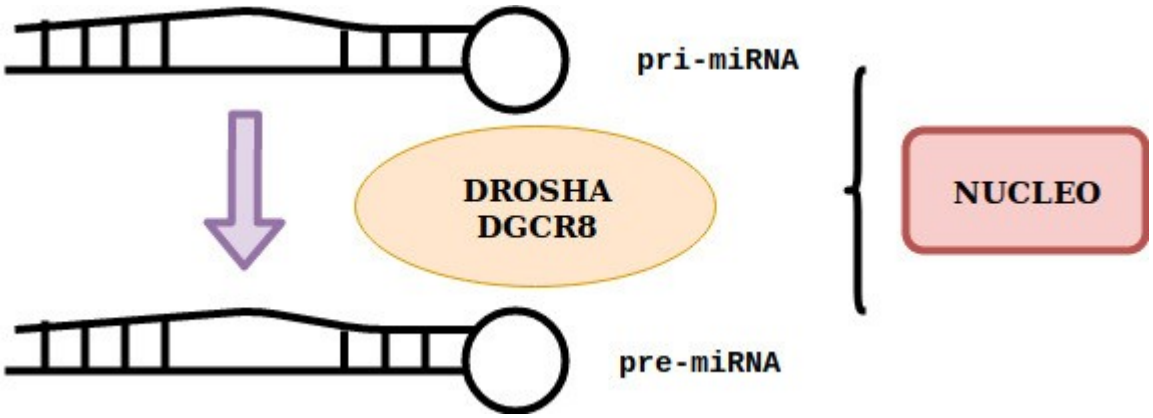


Figura 1.2 : Segundo paso de la síntesis del miRNA

3. Este **pre-micro-RNA** es transportado por medio de la exportina 5 al citoplasma.

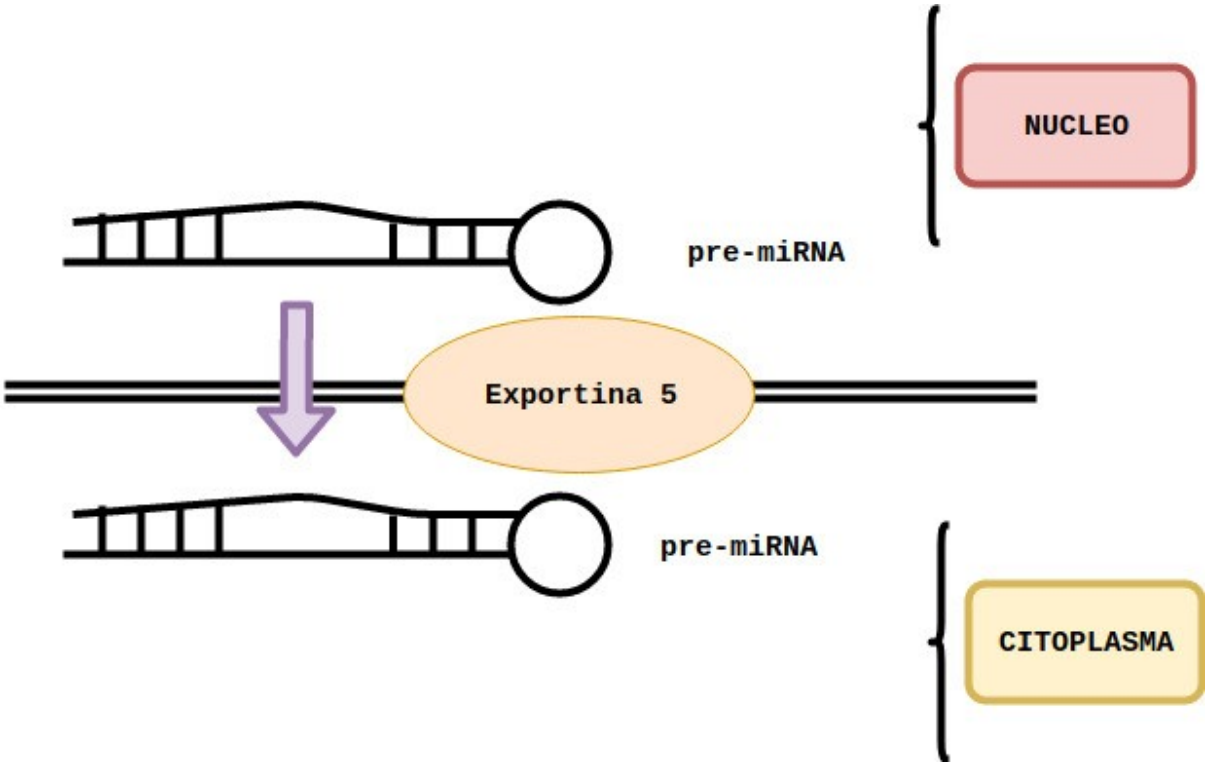


Figura 1.3 : Tercer paso de la síntesis del miRNA

4. En el citoplasma es procesado de nuevo por la ribonucleasa Dicer que separa la horquilla y libera un duplex de miRNA de un tamaño aproximado de 22 nucleótidos.

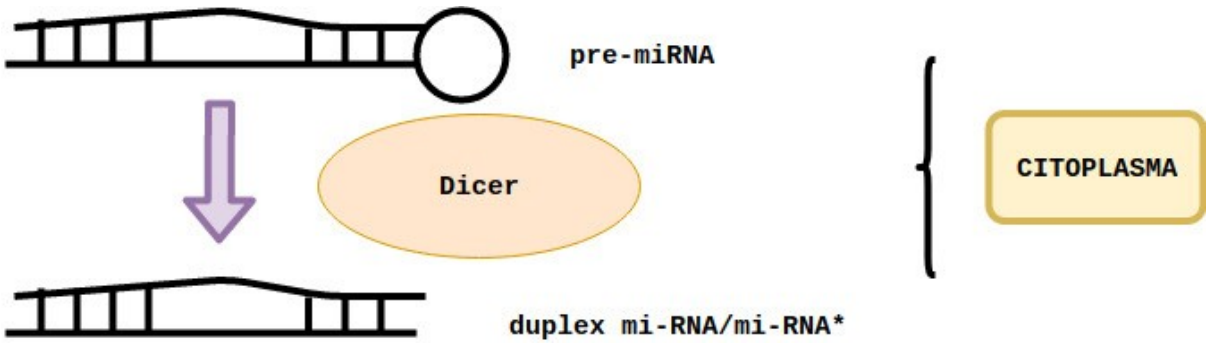


Figura 1.4 : Cuarto paso de la síntesis del miRNA

5. En este momento el **pre-micro-RNA** se incorpora en el complejo RISC (RNA Induced Silencing Complex) el cual está formado por las moléculas: RNAasa Dicer, TRPB, PRKR (proteína quinasa activadora dependiente de ARN) y Ago 2. Una vez incorporado se produce la escisión de la molécula de RNA dando lugar a una cadena de miRNA madura a la cual se priva de su complementaria.

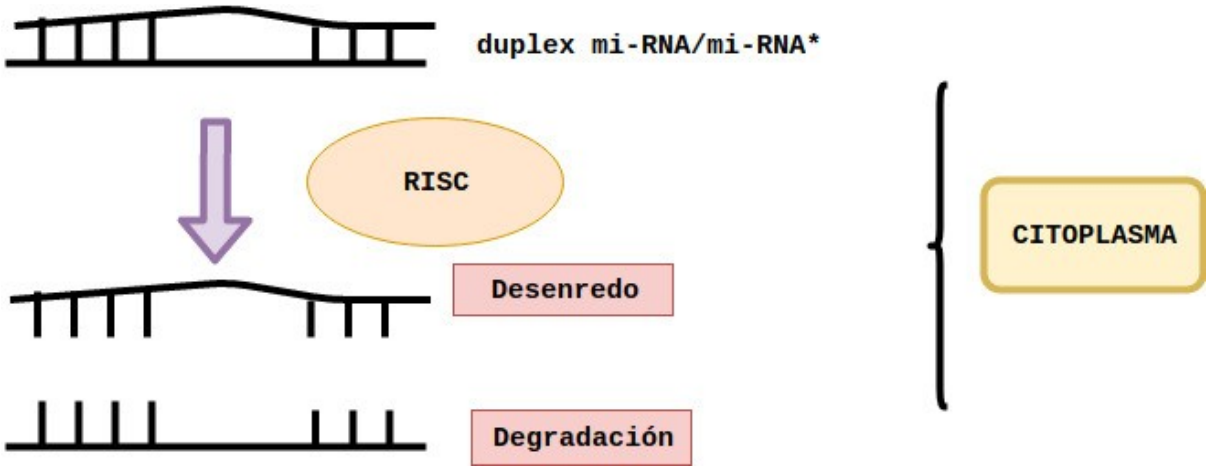


Figura 1.5 : Quinto paso de la síntesis del miRNA

1.2 mirTOP

El proyecto mirTOP[3] (miRNA Transcriptomic Open Project) se ha iniciado con el objetivo de ofrecer herramientas que permitan la unificación de los diferentes formatos de análisis con los que se trabaja en el campo de estudio de los miRNAs. Para ello y basándose en GFF3[4], formato existente de análisis de miRNA, se ha creado mirGFF3 con el que se trata de optimizar y unificar la información proveniente de las diferentes herramientas como son seqbuster, miRge2.0, isomiR-SEA, sRNAbench, PROST!, así como archivos BAM[3].
Junto a esta parte del proyecto se ha desarrollado paralelamente la herramienta del mismo nombre, **mirtop[3]**, diseñada para convertir con facilidad a mirGFF3[3] los archivos provenientes de los diferentes programas de análisis.

1.3 GFF3 y mirGFF3

El formato GFF3[4] (Generic Feature Format Version 3) consta de 9 columnas tabuladas cada una de las cuales aporta una información específica y relevante del fragmento de miRNA que describe.

Columna 1 (sequid/seqname). El identificador de la referencia empleada para establecer el sistema de coordenadas para el precursor.

Columna 2 (source). Típicamente es el nombre del software, proyecto o base de datos del que deriva el archivo.

Columna 3 (type/feature). El tipo de muestra indicada por su Secuencia Ontológica, su número de acceso, etc.

Columna 4 (start). Coordenadas que indican el inicio del precursor analizado.

Columna 5 (end). Coordenadas que indican el final del precursor analizado.

Columna 6 (score). La puntuación con coma decimal del fragmento.

Columna 7 (strand). Definida como "+"(forward) o "-" (reverse)

Columna 8 (phase/frame). Indicador de si la primera base del fragmento coincide con la primera base del codón, la segunda, etc. 0, 1, 2, etc.

Columna 9 (attributes). Una lista de elementos separados por ";" que proporcionan información adicional sobre cada fragmento de análisis.

Si comparamos el formato original GFF3[6] con el mirGFF3[3] vemos que hay modificaciones en varias de sus columnas.

Columna	GFF3	mirGFF3
sequid/seqname	V	V
source	V	V
type/feature	V	V
start	V	V
end	V	V
score	V	Opcional*
strand	V	V
phase/frame	V	X
attributes	V	Modificada**

Tabla 1.3 : Esquema comparativo del formato GFF3[6] con mirGFF3[3]

En mirGFF3[3] se conservan las columnas: sequid, source, type, start, end y strand eliminando la columna “phase”. La columna “score” pasa a ser opcional* en virtud de si la herramienta que ha producido el archivo de datos la emplea o no y se substituye la columna “atributes” por una versión modificada*. Si estudiamos en detenimiento dicha columna vemos que varios de los descriptores se han alterado desde su disposición original.

Atributo 1 (ID). Es el identificador del fragmento, debe ser único.

Atributo 2 (name). Nombre asignado al fragmento. No es necesario que sea único.

Atributo 3 (alias). Nombres alternativos del fragmento en otras bases de datos.

Atributo 4 (parent). Indica el fragmento de referencia lo que permite agrupar exones en transcritos, estos en genes, etc.

Atributo 5 (target). Indica la diana del alineamiento nucleótido-nucleótido o bién proteina-proteina.

Atributo 6 (gap). El alineamiento del fragmento al objetivo en caso de que ambos contengan huecos y sean no colineares.

Atributo 7 (derives_from). Se emplea para desambiguar la relación entre dos fragmentos cuando esta es temporal en vez de estructural.

Atributo 8 (note). Notas de texto con información adicional.

Atributo 9 (dbxref). Referencia cruzada con bases de datos externas.

Atributo 10 (ontology_term). Referencia cruzada con el sistema de clasificación de “terminos ontológicos”.

Si comparamos los atributos en el formato original GFF3[3] con la columna de atributos en

mirGFF3[6] vemos que hay múltiples modificaciones que podemos dividir en los atributos que se conservan o modifican y otros que se añaden y no están en la original.
En los atributos de mirGFF3[6] se prima la información relativa al análisis de los diferentes isomiR vinculados al fragmento de miRNA estudiado.

Atributos GFF3	Atributos mirGFF3
ID	UID
name	name
alias	variant
parent	parent
target	filter
gap	hits
derives_from	expression
note	read
bxref	cigar
ontology_term	X

Tabla 1.4 : Esquema comparativo de los atributos en GFF3[3] con respecto a mirGFF3[6]

Atributo 1 (UID). Identificador único de cada isomiR dependiente de su secuencia e independiente de su posición en el genoma.

Atributo 2 (read). Secuencia de bases nitrogenadas que constituye el fragmento.

Atributo 3 (name). Nombre de la secuencia madura.

Atributo 4 (parent). Nombre de la horquilla precursora.

Atributo 5 (variant). Describe las variaciones entre la secuencia del isomiR con el miRNA de referencia en la base de datos de elección.

Atributo 6 (cigar). Es el CIGAR de alineaciones.

Atributo 7 (hits). Indica el unmero de veces en el que la secuencia coincide con los diferentes cambios de isomiR.

Atributo 8 (expression). El numero total de lecturas de la secuencia.

Atributo 9 (filter). Indica si una variación es aceptada o no según las opciones de filtrado al adjudicarle una puntuación de confianza.

2

Metodología y plan de trabajo

2.1 Empleo de R

Para el desarrollo del proyecto hemos elegido Rstudio[1], IDE (Entorno de Desarrollo Integrado) para lenguaje R[17], el cual ofrece una base sólida para la escritura del programa al orientarse hacia el análisis estadístico y presenta multitud de paquetes y librerías que aportan las herramientas necesarias para la consecución del mismo. Estos paquetes y librerías permiten a R destacar en el abanico de funcionalidades orientadas al desarrollo de modelos lineales, algoritmos de clasificación y tests estadísticos. La gran mayoría forman parte del proyecto Bioconductor[8], conjunto de aplicaciones y herramientas para el análisis de datos genéticos.
Las librerías empleadas en el presente proyecto han sido, entre otras:

tidyverse: [21]Constituye un conjunto de paquetes orientados a la manipulación exploración y visualización de datos. La idea detrás del paquete es facilitar el establecimiento de flujos de trabajo y conectar las diferentes herramientas que dan lugar al correcto discurrir de este flujo por medio del pipe operator "%>% " el cual enfatiza la secuencia de acciones.
Dentro del núcleo que forma tidyverse [21]tenemos los siguientes paquetes agrupados en virtud a la función que desempeñan dentro del entorno.

ggplot2 (librería de visualización): Se encarga de la parte de visualización y es un sistema de creación de gráficos que permite la inclusión de gran número de variables.

tibble (librería para obtener "tidy data"): Permite una nueva reformulación del data frame dando lugar a objetos de este tipo que permiten una mayor complejidad en su estructura.

tidyr (librería para obtener "tidy data"): Permite obtener "tidy data" que se traduce

como que cada observación es una fila, cada variable una columna y cada valor una celda de la tabla de datos.

dplyr (librería de transformación): Aporta un conjunto de expresiones (*mutate( ), select( ), filter( ),* etc.) para la manipulación de datos que permite solucionar los retos más habituales a los que nos podemos llegar a enfrentar.

stringr (librería de transformación): Funciones que ayudan a trabajar fácilmente con "strings".

readr (librería de importación): Permite leer datos de archivos de tipo csv. Haciendo que el parseo de información sea más sencillo y completo.

purrrlyr (librería de programación): Añade caracerísticas de programación funcional a R sin cambiar la estructura elemental del lenguaje.

SummarizedExperiment[16]: Los objetos desarrollados por este paquete almacenan observaciones de varias muestras en matrices rectangulares junto con metadatos descriptivos adicionales. La coordinación entre los metadatos y los ensayos (assays) del objeto representa un aspecto clave en esta clase.

El objeto ***se*** es una matriz donde las filas y sus caacterísticas (rowData) representan datos de interés (genes, exones, etc.) y las columnas (colData) representan data frames con los diferentes atributos de las muestras.

Como hemos dicho cada objeto ***se*** puede tener uno o más ensayos (assays) cada uno con una estructura de matriz.

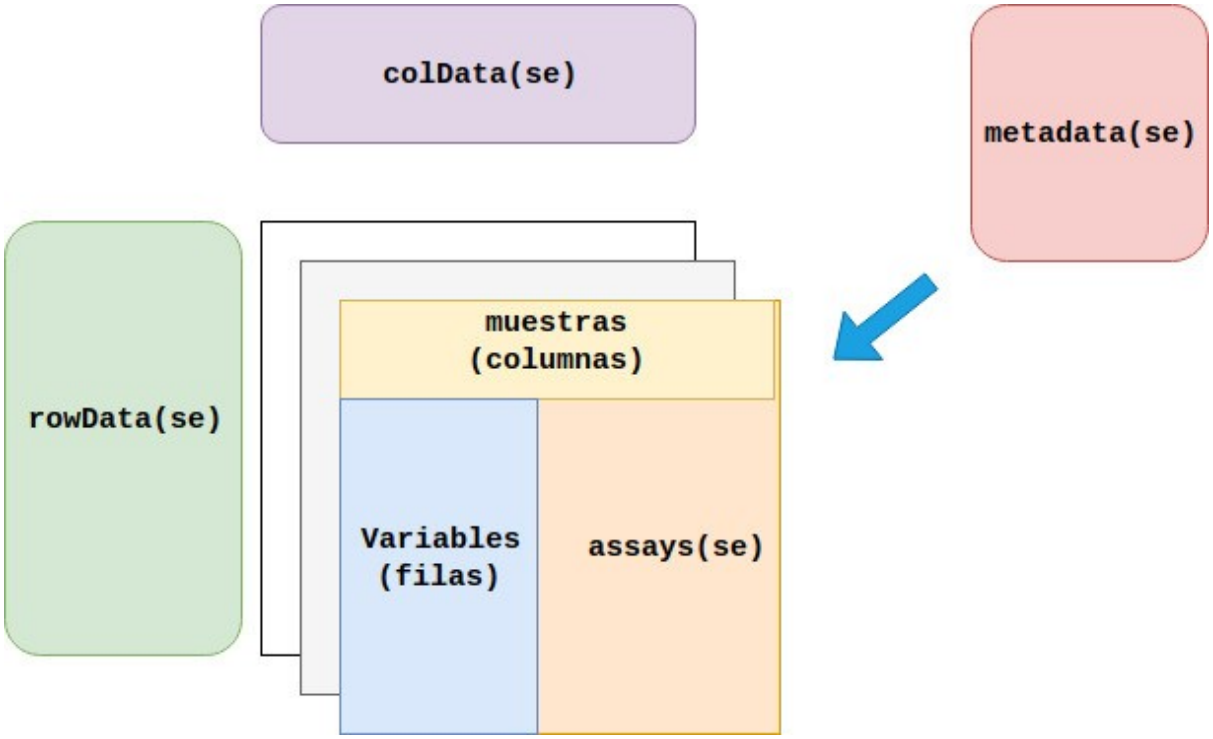


Figura 2.1: Esquema de la estructura de SummarizedExperiment

DT: Este paquete proporciona una interfaz con la librería de JavaScript "*DataTables*" de manera que objetos de R como matrices o dataframes pueden ser mostrados como tablas en páginas con formato HTML.

DESeq2[22]: Nos permite un análisis de la expresión diferencial, estimando la media de la varianza basándonos en modelos lineales generalizados de distribución binomial negativa.
El objeto creado y denominado *DESeqDataSET* es una extensión de la clase *RangedSummarizedExperiment*, muy similar a la clase *SummarizedExperiment* pero en este caso las filas de la primera representan rangos genómicos de interés y no un dataframe de atributos.
El objeto *RangedSummarizedExperiment* además debe tener una fórmula de diseño asociada que contempla las variables que serán usadas en la construcción del modelo.

DEGreport: Permite crear un informe en formato HTML de la expresión diferencial de un conjunto de datos integrando código procedente de DESeq2.
Dentro de esta hemos empleado diferentes funciones como son: *DEGpattern* la cual permite agrupar genes según perfiles de expresión.

2.2 Datos de trabajo

Los datos empleados en el ensayo final del programa corresponden con la investigación *Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories* desarrollada por Peter A C ’t Hoen, Marc R Friedländer et al. [18]Esta se basa en la secuenciación de miRNA en líneas celulares de linfoblastoide de 465 individuos en siete centros de secuenciación y con un gran número de réplicas.
Esta es una de las investigaciones bajo el proyecto GEUVADIS (Genetic European Variation in Disease) que aúna diecisiete laboratorios centrados en la secuenciación génica centrados en proyectos de gran escala como son ”los 1000 genomas” y el “Consorcio Internacional Genómico”.

El conjunto de datos final consiste en un archivo en formato mirGFF3 (geu-mirtop.gff) con un peso de **167,4 MB** conteniendo **137.844** lineas de muestras y los metadatos asociados (geu-metadata.csv) con un tamaño de **18,1 KB** y **452** lineas de muestras.

Los datos pueden descargarse en las direcciones:

https://www.dropbox.com/s/addg7wldhl1k46s/geu-mirtop.gff.gz?dl=0

https://www.dropbox.com/s/9xx5d9u1veq6oa3/geu-metadata.csv?dl=0

Por otro lado durante el desarrollo de la aplicación hemos trabajado con archivos de datos menores y más manejables que hemos empleado para testear el funcionamiento del código y a los que se pueden acceder dentro de la página de GitHub[15] del presente proyecto https://github.com/miRTop/mirgff3_shiny/tree/master/test y que consiste de un archivo principal en formato mirGFF3 (mirtop.gff) con un tamaño de **4 MB** y que contiene **17.778** lineas de muestras y el archivo de metadatos asociados en formato csv (metadata.csv) con un tamaño de **126 bytes**.

2.3 Shiny[10]

Paquete de R que emplearemos como principal herramienta para el desarrollo de nuestra

aplicación web y que permite convertir código R en HTML5/CSS/JavaScript de manera fácil y rápida . Dos características propias del paquete y que favorecen su empleo para este fin es que Shiny se basa en la ***programación reactiva[11]*** la cual actúa vinculando los valores de entrada y salida del programa de manera que los valores de salida se modifica automáticamente al modificar los valores de entrada. Esto resulta enormemente útil al diseñar una aplicación de análisis ya que nos permite jugar más con los datos seleccionando diferentes conjuntos de los mismos y actualizando automáticamente los resultados.

En segundo lugar está el empleo de **widgets**, definidos como pequeñas programas prediseñados que aportan funcionalidades de uso frecuente. De esta manera evitamos el tener que desarrollarlas de cero aportándole al paquete todo un abanico de ventajas que nos permiten añadir funcionalidad de una manera modular.

2.4. Diseño de la aplicación

En primer lugar necesitamos establecer el diseño general de la aplicación, definir los métodos de entrada y salida de datos, las variables que emplearemos y la disposición general del interfaz de usuario.
Optaremos por el formato *sidebarLayout* que se compone de una barra lateral donde colocaremos el menú de carga de archivos y la opciones de normalización, así como un botón que al pulsarlo inicie la carga de datos y el desarrollo de las variables de salida.

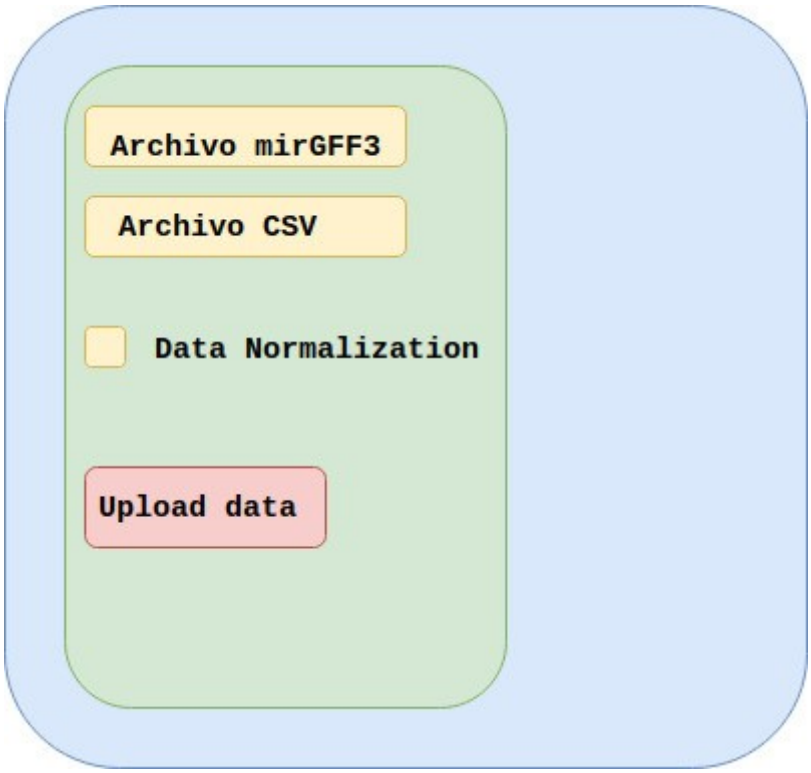


Figura 2.2: Esquema de la barra lateral en la aplicación de análisis de expresión génica en formato mirGFF3

En la sección principal estableceremos tres pestañas cada una de las cuales mostrará una información diferente relativa a los datos que queremos analizar.

Pestaña 1: Información. En esta nos limitamos a mostrar la información del assay de datos crudos del objeto *SummarizedExperiment* [16] con el que vamos a trabajar o bien la información relativa a la normalización del *DESeqDataSet [22]* después de someterlo a una transformación estabilizadora de la varianza.

Pestaña 2: PCA. El código en esta sección establece una PCA (Principal Components Analysis) [17] de los datos procesados permitiendo reducir las variables que definen el conjunto de datos a unas nuevas no correlacionadas y ordenadas por la cantidad de varianza original que representan. Estas representaciones se establecen en términos de mínimos cuadrados y codifique un sistema de colores en virtud de los metadatos. Para la selección de los mismos establecemos un menu desplegable con las diferentes columnas de metadatos y un botón de inicio que indica al programa la necesidad de crear la gráfica actualizada en virtud de la selección en el menú de metadatos.

Pestaña 3: Selector. Creamos una tabla de filas seleccionables con los diferentes isomiR y creamos una gráfica individual para cada una de las selecciones.

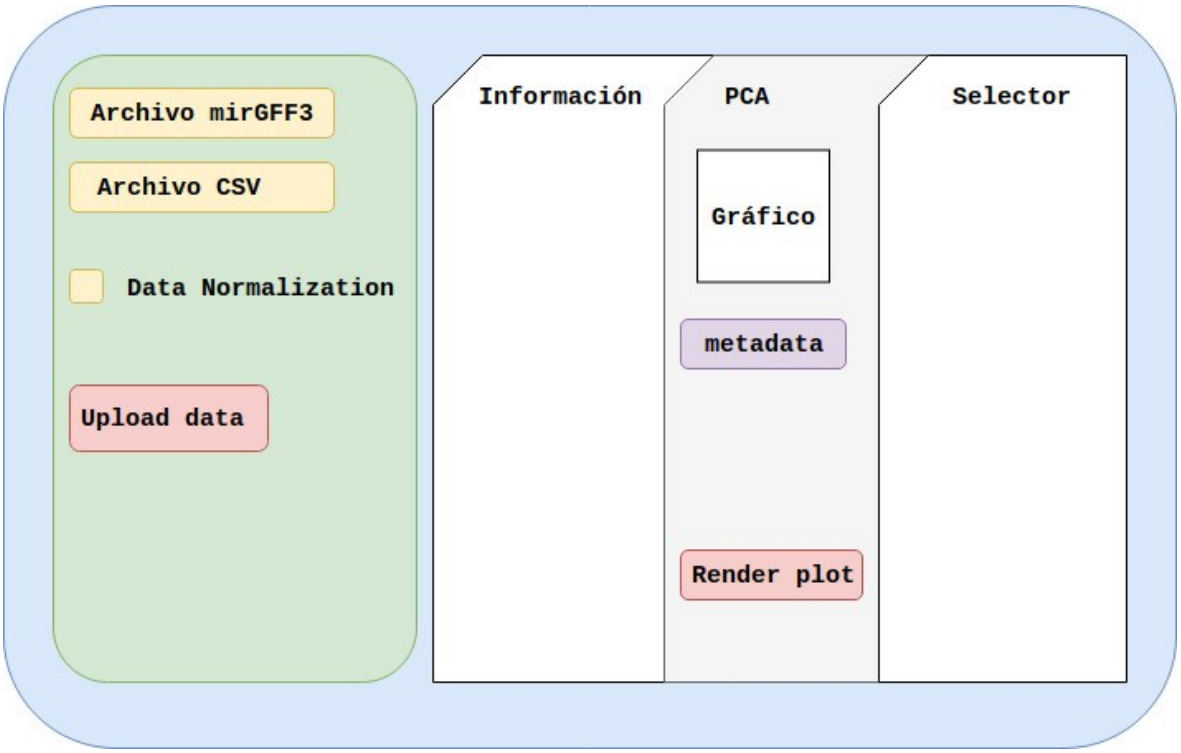


Figura 2.3: Esquema de la plicación de análisis de expresión génica en formato mirGFF3

2.5 Github[15]

Durante el desarrollo del proyecto hemos empleado la plataforma Github para compartir el código y discutir las modificaciones del mismo: https://github.com/miRTop/mirgff3_shiny
Se trata de un servicio web para el desarrollo colaborativo de software que permite llevar un registro y control de las diferentes versiones del código así como almacenar y distribuir el mismo.
El funcionamiento se basa en los repositorios Git, los cuales establecen una copia local del código para así poder llevar un seguimiento de todos los cambios que se establecen y recuperar cualquier versión del producto. Además permite clonar el repositorio donde está el código, modificarlo y enviar tu versión al administrador el cual evalúa los cambios y añadirlo o no al código maestro.
En definitiva, una herramienta enormemente útil y casi indispensable en cualquier proyecto de desarrollo informático.

3

Resultados

3.1 Código

En esta sección vamos a analizar el código empleado en la aplicación. Al tratarse de una aplicación en Shiny, este se divide en dos partes (**server.R** y **ui.R**) que en este caso desarrollaremos en sendos archivos interconectados.

Contiene la lógica interna del programa.

Definida como “interfaz de usuario”, representa el aspecto de la aplicación y las opciones de interacción de la misma (*input* de datos y *output* de informes y gráficos).

Las diferentes funciones definidas dentro de ***server.R*** se organizan en varios bloques según la función que representan y estos se interconectan por medio de un sistema de entrada y salida de variables con el código en ***ui.R*** el cual presenta la información de dichas variables al usuario así como los resultados al ejecutar las diferentes funciones de procesamiento.

Lectura de archivos mirGFF3 y csv

El primer bloque de código se centra en la entrada de datos.

Creamos el objeto **shinyServer** [11] que contendrá la información relativa a las funcionalidades de la aplicación y establecemos inicialmente un límite mayor al tamaño de datos de entrada en 200 MB, esto es fundamental si deseamos probar el programa acabado con el conjunto de datos final. A continuación definimos las variables con los datos de entrada que apuntarán a los archivos *mirGFF3* y *csv* como información inicial y almacenará

los mismos en sendas variables.

Esta función la establecemos del tipo “reactive” de manera que cada vez que modifiquemos alguno de los parámetros dentro de la misma esto se traducirá en las variables de salida.

```
shinyServer(function(input, output, session) {
    options(shiny.maxRequestSize=200*1024^2)
    dataInput <-reactive({
        inFile1 <-input$file1
        inFile2 <-input$file2
```

Figura 3.1: Código de server.R para introducir los datos de la aplicación.

En la siguiente sección definimos las dos funciones de lectura. La primera para el formato *mirGFF3* y la segunda para el archivo *csv*. Estas funciones se limitan a permitir la lectura de datos sin ningún procesamiento con la excepción de renombrar las columnas del archivo *mirGFF3*.

```
lecturaGFF<-function(gff){resultado<- as_tibble(read_tsv(gff, col_names=
c("seqname","source","feature","start","end","score","strand","frame","attribute"),
comment="#"))
return (resultado)
                        }
```

Figura 3.2: Código de server.R para leer archivos mirGFF3.

```
coldata_extract_csv <- function(csv){md_raw <- read.csv(csv, row.names = 1)
return(md_raw)
```

Figura 3.3: Código de server.R para leer archivos csv.

ui.R En esta sección indicamos el tema gráfico de la aplicación dentro de las opciones que aporta *Shinythemes[11]* y el título de la aplicación.

```
shinyUI(fluidPage(
    #Tema de diseño
    theme = shinytheme("united"),
    #Titulo de la aplicación
    titlePanel("mirGFF3 Reader"),
    #Disposición general del interface de la aplicación.
```

Figura 3.4: Código de ui.R para establecer el tema y título del programa.

A continuación desarrollamos la configuración de la barra lateral que permitirá la entrada de datos únicamente en los formatos *csv* y *mirGFF3* respectivamente, indicando además donde encontrar los archivos de prueba en la página de gitHub donde se encuentra el proyecto.

```
 sidebarLayout(
      #Disposición del panel lateral.
      sidebarPanel(
          #Menú para cargar el arhivo GFF y CSV.
          fileInput("file1", "Choose CSV File",
                    multiple = FALSE,placeholder = "test/metadata.csv",
                    accept = c("text/csv",
                               "text/comma-separated-values,text/plain",
                               ".csv")),
          fileInput("file2", "Choose GFF File",
                    multiple = FALSE,placeholder = "test/mirtop.gff",
                    accept = c(".gff")),
          #Casilla activada por defecto que selecciona la normalización
o no de los datos
          checkboxInput(inputId = "normalize",
                        label = strong("Data normalization"),
                        value = TRUE),
          #Botón de acción para lanzar la aplicación.
          actionButton("upload", "Upload Data")
      ),
```

Figura 3.5: Código de ui.R para establecer el interfaz de entrada de datos y normalización de los mismos.

Seguidamente establecemos una casilla que derivará al empleo o no del conjunto de datos normalizado y un botón que iniciará el proceso de cálculo y análisis una vez los datos se han cargado correctamente.

Funciones para obtener *colData, rowData, counts* y *metadata*

server.R En esta sección definimos las funciones necesarias para obtener los parámetros que constituirán el ensayo de datos sin procesar de este objeto.

El primer fragmento de código crea la variable ***colData*** [16] que contiene un *data frame* con columnas descriptivas de información relativa a las filas de muestras. Esta viene indicada en las primeras líneas de comentarios del archivo mirGFF3 a continuación del epígrafe "COLDATA".
El código se encarga de leer las 3 primeras líneas de texto de la variable donde hemos almacenado este archivo y extrae la información señalada.

```
coldata_extract <- function(gff_coldata) {
primeras_lineas = readLines(gff_coldata, n = 3)
coldata <- grep("COLDATA",primeras_lineas, value = TRUE)
            coldata2<-coldata %>% str_replace("## COLDATA: ","")
            coldata3<-unlist(str_split(coldata2, ","))
            return(coldata3)
                                        }
```

Figura 3.6: Código de server.R para crear la variable colData.

A continuación creamos la variable **attributes (rowdata)** [16] que contiene parámetros de interés de las diferentes muestras. La función se encarga de editar esta información y almacenarla en esta variable, para ello lee el archivo mirGFF3, lo convierte en un objeto de tipo *tibble* y lo procesa devolviendo una tabla con las filas.

```
atributes_extract<-function(gff){
            datos1<-lecturaGFF(gff)
            as_tibble(expresion <-datos1 %>%
                          select(attribute))
            gff_table<-datos1 %>%
                mutate(uid = str_extract(attribute,"iso-[0-9]+-\\w+;")) %>%
                separate_rows(attribute, sep=";") %>%
                mutate(attribute = gsub("=", " ", trimws(attribute))) %>%
                separate(attribute, sep = " ", into = c("att", "value")) %>%
                spread(att, value) %>%
                select(-uid) # remove temporal ID
            return (gff_table)
                                }
```

Figura 3.7: Código de server.R para crear la variable rowData/attribute.

La siguiente define la variable counts que almacena la puntuación de los diferentes miRNAs y sus isomiR.

En esta sección empleamos la función que obtiene los *rowData/attribute* y transformamos en un tibble la información de las columnas de atributos UID y Expression. Editamos levemente la información y la devolvemos en un formato de matriz numérica.

```
   counts_extract<-function(gff, colnames){
           datos1<-atributes_extract(gff)
           as_tibble(uid_exp <-datos1 %>%
                        select(c(UID,Expression))) %>%
               select(c(UID, Expression)) %>%
               separate(Expression, sep=",", into = colnames, convert =
TRUE) %>%
               distinct()  %>%
               as.data.frame() %>%
               column_to_rownames("UID") %>%
               as.matrix()
       }
```

Figura 3.8: Código de server.R para crear la variable counts.

Barra de progresos y constitución del objeto *SummarizedExperiment*

En esta sección establecemos una barra de progreso que nos permitirá ver si la aplicación sigue trabajando a pesar de que no muestre los resultados de inmediato. Esto puede ser redundante en el caso de conjuntos de datos pequeños ya que el tiempo de espera es corto pero con datasets de mayor tamaño como el de los "*datos finales*" de este proyecto es necesario un indicador. Esto lo logramos con la función *withProgress* y añadiendo puntos de control por medio de *incProgress* para cada una de las partes que constituye el procesamiento de los elementos que forman el ensayo de *SummarizedExperiment*.

Las líneas siguientes se centran en llamar a las funciones definidas anteriormente para obtener las variables con la información de *colData*, *rowData*, *counts* y *metadata* y aplicarlas a los archivos de entrada del programa definiendo las variables dando lugar al *assay* de *SummarizedExperiment.*

```
withProgress(message = 'Calculating',
                     detail = 'Hold your horses...', value = 0, {
                     n<-7

                     incProgress(1/n, detail = paste("Reading coldata"))

                     colnames<- coldata_extract(inFile2$datapath)
                     incProgress(1/n, detail = paste("Reading coldata"))
                     attributes<-atributes_extract(inFile2$datapath)
                     incProgress(1/n, detail = paste("Reading counts"))
                     counts<-counts_extract(inFile2$datapath, colnames)
                     incProgress(1/n, detail = paste("Reading metadata"))
                     metadata<-coldata_extract_csv(inFile1$datapath)
                     updateSelectInput(session, "datadrop", choices =
colnames(metadata))
                     incProgress(1/n, detail = paste("Creating object"))
                     keep <- rowSums(counts>0) > (ncol(counts) * 0.2)
                     attributes <- attributes[keep,]
                     counts <- counts[keep,]
                     se<-SummarizedExperiment(assays = list(raw =
counts[,rownames(metadata)]),colData = metadata, rowData = attributes)
                     good_samples <- colSums(assays(se)[[1]]>0) > 50
                     se <- se[,good_samples]
                     dds<-DESeqDataSetFromMatrix(assays(se)[[1]],
colData(se),design = ~1)
                     incProgress(1/n, detail = paste("Normalizing"))
                     vst <- varianceStabilizingTransformation(dds)
                     assays(se)[["vst"]] <- assay(vst)
                     if(input$normalize) {
                         assays(se)[["use"]] <- assays(se)[["vst"]]
                     }else{
                         assays(se)[["use"]] <- assays(se)[["raw"]]
                     }
                     incProgress(1/n, detail = paste("Done"))
                     })
        se
```

Figura 3.8: Código de server.R para crear las variables que formarán SummarizedExperiment a partir de los archivos de entrada y establecer el medidor de progreso.

Otras líneas que vale la pena destacar en detalle son:

```
                     if(input$normalize) {
                         assays(se)[["use"]] <- assays(se)[["vst"]]
                     }else{
                         assays(se)[["use"]] <- assays(se)[["raw"]]
                     }
                     incProgress(1/n, detail = paste("Done"))
                     })
          se
```

Figura 3.9: Código de server.R para crear el assay vst dentro de SummarizedExperiment con los datos normalizados.

Este fragmento modifica el *assay* que se va a devolver para su análisis en función de si la casilla de "*normalización de datos*" está activada o no siendo los datos crudos de *SummarizedExperiment* si no lo está y el ensayo "*vst*" de datos normalizados si lo está.

```
                    dds<-DESeqDataSetFromMatrix(assays(se)[[1]],
colData(se),design = ~1)
                    incProgress(1/n, detail = paste("Normalizing"))
                    vst <- varianceStabilizingTransformation(dds)
                    assays(se)[["vst"]] <- assay(vst)
```

Figura 3.10: Código de server.R para crear el assay vst con los datos normalizados.

En esta sección desarrollamos el objeto *DESeqDataSet* a partir del *SummarizedExperiment* de datos crudos y a continuación aplicamos una normalización de los resultados por medio de *varianceStabilizingTransformation* creando otro assay con estos datos modificados que denominaremos "*vst*".

Información sobre el objeto de estudio

server.R En estas líneas indicamos como condición que se presione el botón dentro de ui.R de carga de datos, en caso de que esto suceda se crea la variable se con los datos procedentes de dataInput() y se muestra el contenido de esta variable en modo texto.

```
observeEvent(input$upload, {
       #Definimos la variable "se" (summarizedExperiment) en esta sección
para que puedan utilizarla el resto de funciones.
       se<-dataInput()
       #Mostramos el contenido del objeto "contenido" que es un objeto
SummarizedExperiment.
       output$contenido<- renderPrint({
          se
       })
```

Figura 3.11: Código de server.R para mostrar la información básica de SummarizedExperiment.

ui.R Definimos el panel central de la aplicación y establecemos un formato de pestañas siendo la primera una que muestra información general sobre el objeto de estudio.

```
mainPanel(
           tabsetPanel(type = "tabs",
                       #Pestaña con información sobre el
SummarizedExperiment
                       tabPanel("Info", verbatimTextOutput("contenido")),
```

Figura 3.12: Código de ui.R para mostrar la información básica de SummarizedExperiment.

Análisis de componentes principales (PCA)

server.R El código en esta sección establece una PCA[17] de los datos procesados permitiendo reducir las variables que definen el conjunto de datos a unas nuevas no correlacionadas y ordenadas por la cantidad de varianza original que representan. Estas representaciones se establecen en términos de mínimos cuadrados y que codifique un sistema de colores en virtud de los metadatos.

En el código definimos la variable de salida (*pca*) del tipo gráfica y le indicamos que la realice por medio de la función *degPCA* empleando la variable definida por *dataInput()* y que devuelve un *assay* de *SummarizedExperiment* con los datos normalizados o no según la selección inicial en el menú lateral de la aplicación al cargar los datos iniciales.

```
output$pca<- renderPlot({
            degPCA(assays(dataInput())[["use"]], metadata = colData(se),
data = FALSE)
        })
```

Figura 3.13: Código de server.R para mostrar la gráfica de PCA.

ui.R Desarrollamos otra pestaña dentro del panel central para mostrar la gráfica con los componentes principales. Adjudicamos un identificador (“PCA”) y un output de salida de tipo gráfica (*pca*). A continuación establecemos un menú selector que cargue las opciones del archivo de *metadatos* en un “datadrop” comunicando con la línea de código explicada en la sección *server.R* de esta función.

Por último establecemos un botón de inicio que señala cuando el program debe leer la selección del metadato y mostrar la gráfica correspondiente, esta vez diferenciando los puntos de la misma con un código de colores.

```
tabPanel("PCA", plotOutput("pca"),
                                selectInput("datadrop","metadata", choices = ""),
                                actionButton("upload2", "Render plot")
                    ),
```

Figura 3.14: Código de ui.R para mostrar la gráfica de PCA además de mostrar la selección de metadata y establecer un botón de inicio.

Selector de isomiRs

server.R En esta sección desarrollamos una tabla interactiva que permite seleccionar sus filas y crea una gráfica de las mismas.

En el código inicialmente convertimos el contenido que va a ir en la tabla (conjunto de isomiRs) en un data frame y seguidamente definimos la misma y enviamos el resultado a la variable de salida que comunicará con el *ui.R*.

En las siguientes líneas definimos las gráficos de salida y el selector de las filas (*fila5*).

Por último podemos ver un condicional el cual define que si al menos se selecciona una fila

se desarrolla la gráfica de la misma (*degPlot*) y el parámetro genes dentro de la función de la gráfica utilizará la fila dentro de la variable *rownames* del *SummarizedExperiment[16]* que hayamos seleccionado en la tabla.

```
rowdataDF<-as.data.frame(rowData(se))
        #Creamos el output que es una tabla a partir de rowData con lineas
seleccionables.
        output$tabla4<-DT::renderDataTable(rowdataDF, server = TRUE,filter
= 'top')
        #Creamos un output que son gráficos de los isomeros seleccionados.
        output$graph <- renderPlot({
            #Creamos la variable que almacenará las filas seleccionadas.
            filas5 <-input$tabla4_rows_selected
            #Creamos metadata a partir de la variable se
            #Desarrollamos los gráficos de las filas seleccionadas.
            if (!is.null(filas5)){
                degPlot(se, genes = rownames(se)[filas5], slot = "use",
                        xs = input$datadrop, log2 = FALSE)
            }
        })
```

Figura 3.15: Código de server.R para crear una tabla interactiva que permita seleccionar isomiRs y muestre gráficos de los mismos en virtud a los metadatos.

ui.R

Desarrollamos una última pestaña en el panel central que consta de dos secciones. En la superior establecemos una tabla con los diferentes isomiR de manera que se puedan seleccionar las filas de interés de la misma. En la inferior mostramos gráficas con las selecciones que hemos establecido en el panel superior.

En el código definimos el tercer panel con cuatro secciones, el texto que encontraremos en el encabezado del panel, la tabla reactiva, la información de salida señalando las filas seleccionadas y el gráfico de cada selección.

```
  tabPanel("Selector", DT::dataTableOutput("tabla4"),
                                verbatimTextOutput("info"),plotOutput("graph")),
```

Figura 3.16: Código de ui.R para crear una tabla interactiva que permita seleccionar isomiRs y muestre gráficos de las filas seleccionadas.

3.2. Resultados finales de la aplicación

Al lanzar el programa obtenemos las siguientes opciones en el interfaz gráfico.

Figura 3.2: Menú inicial de la aplicación “mirGff3 Reader”.

Una vez cargamos los archivos de datos y activamos “Upload data” podemos observar una barra de progresos que nos indica el ritmo de procesamiento para el conjunto de datos presente.

Otra de las opciones en el menú lateral es la normalización de datos que está activada por defecto ya que suele ser recomendable en este tipo de análisis aunque tenemos la opción de des-seleccionarla en caso de que fuera necesario en nuestra investigación.

3.2.1 Datos de prueba normalizados

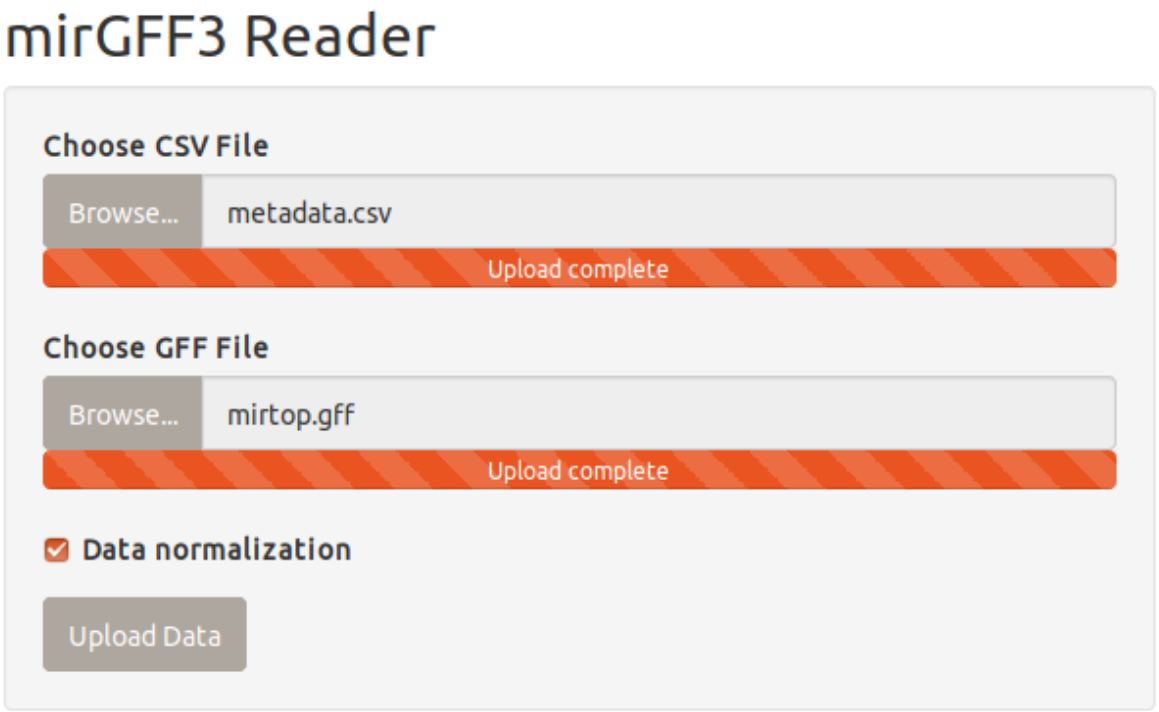


Figura 3.3: Detalle de la carga de datos y las pestaña de normalización en la aplicación "mirGff3 Reader".

El primer resultado que observamos en las pestañas centrales es la información general del objeto *SummarizedExperiment* con el que trabajamos. En este caso empleando el conjunto de datos de prueba (https://github.com/miRTop/mirgff3_shiny/tree/master/test) con la pestaña de normalización seleccionada.
Podemos observar que el número de *assays* del objeto es 3; los datos en crudo (*raw*) sin normalizar, los datos normalizados (*vst*) y el tercer *assay* que representa el que está en uso en ese momento determinado, *vst* si seleccionamos la normalización o *raw* si no.

Figura 3.4: Pestaña principal de información del objeto SummarizedExperiment en la aplicación “mirGff3 Reader” con el conjunto de datos de testeo normalizado.

La siguiente pestaña es la destinada para el análisis de componentes principales (PCA). Inicialmente se muestra la gráfica con los resultados y debajo tenemos la opción de seleccionar uno de los diferentes metadatos provenientes del archivo csv cargado al inicio. En este caso el metadato pre-seleccionado es el primero del datadrop.

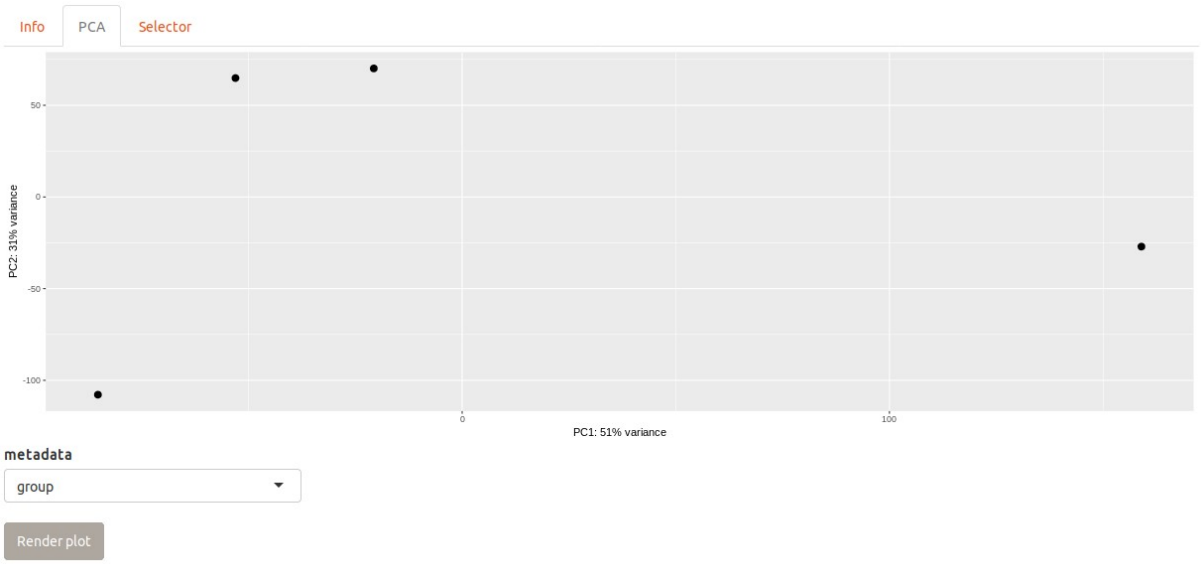


Figura 3.5: Pestaña de PCA en la aplicación “mirGff3 Reader” con el conjunto de datos de testeo normalizado.

Si seleccionamos uno de los metadatos y pulsamos el botón “Render plot” se muestra la PCA relativa al metadato seleccionado codificada por colores. En este caso el archivo de metadatos

sólo cuenta con una columna y dos variables dentro de la misma.

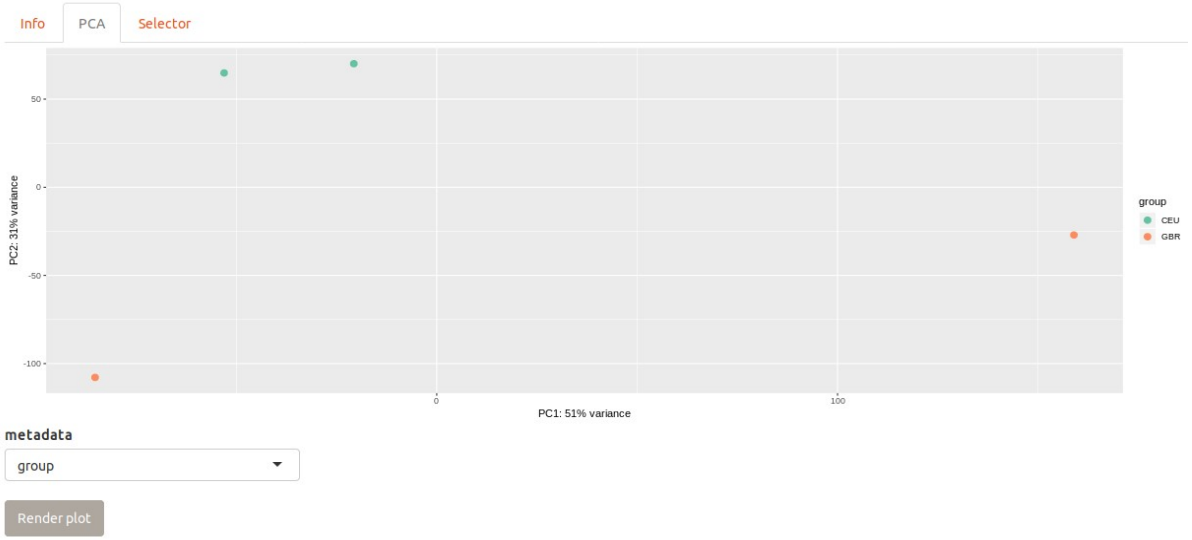
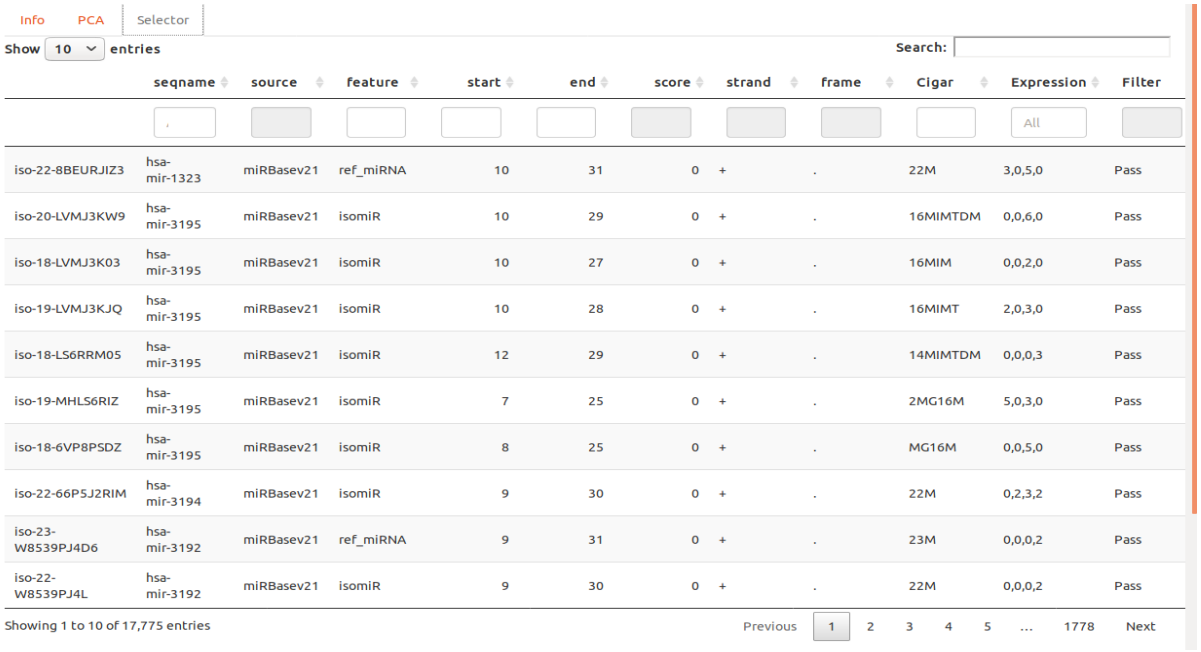


Figura 3.6: Pestaña de PCA en colores dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo normalizado.

La última pestaña consta de una tabla interactiva que nos muestra los atributos de los diferentes isomiR como pueden ser la puntuación del mismo, el origen de la muestra etc. todos ellos referidos anteriormente al explicar el formato mirGFF3.



Info PCA Selector

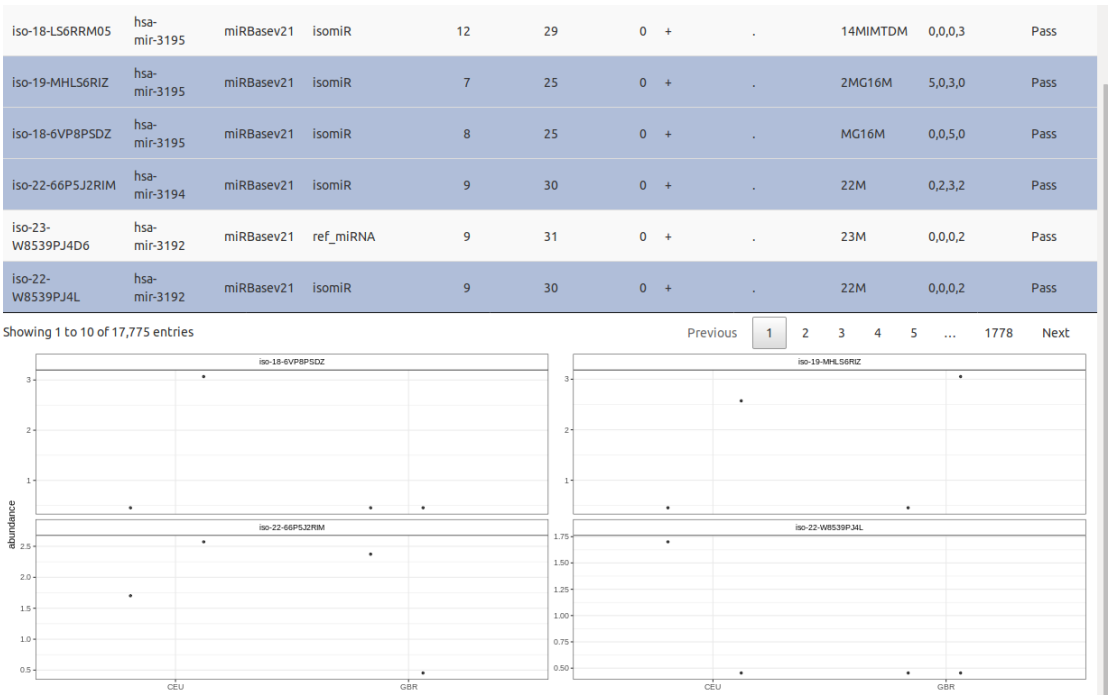
Show 10 entries Search:

	seqname	source	feature	start	end	score	strand	frame	Cigar	Expression	Filter
										All	
iso-22-8BEURJIZ3	hsa-mir-1323	miRBasev21	ref_miRNA	10	31	0	+	.	22M	3,0,5,0	Pass
iso-20-LVMJ3KW9	hsa-mir-3195	miRBasev21	isomiR	10	29	0	+	.	16MIMTDM	0,0,6,0	Pass
iso-18-LVMJ3K03	hsa-mir-3195	miRBasev21	isomiR	10	27	0	+	.	16MIM	0,0,2,0	Pass
iso-19-LVMJ3KJQ	hsa-mir-3195	miRBasev21	isomiR	10	28	0	+	.	16MIMT	2,0,3,0	Pass
iso-18-LS6RRM05	hsa-mir-3195	miRBasev21	isomiR	12	29	0	+	.	14MIMTDM	0,0,0,3	Pass
iso-19-MHLS6RIZ	hsa-mir-3195	miRBasev21	isomiR	7	25	0	+	.	2MG16M	5,0,3,0	Pass
iso-18-6VP8PSDZ	hsa-mir-3195	miRBasev21	isomiR	8	25	0	+	.	MG16M	0,0,5,0	Pass
iso-22-66P5J2RIM	hsa-mir-3194	miRBasev21	isomiR	9	30	0	+	.	22M	0,2,3,2	Pass
iso-23-W8539PJ4D6	hsa-mir-3192	miRBasev21	ref_miRNA	9	31	0	+	.	23M	0,0,0,2	Pass
iso-22-W8539PJ4L	hsa-mir-3192	miRBasev21	isomiR	9	30	0	+	.	22M	0,0,0,2	Pass

Showing 1 to 10 of 17,775 entries Previous 1 2 3 4 5 ... 1778 Next

Figura 3.7: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo normalizado y seleccionando.

Esta nos permite seleccionar los isomiR que deseemos investigar para observar una gráfica de los mismos.



iso-18-LS6RRM05	hsa-mir-3195	miRBasev21	isomiR	12	29	0	+	.	14MIMTDM	0,0,0,3	Pass
iso-19-MHLS6RIZ	hsa-mir-3195	miRBasev21	isomiR	7	25	0	+	.	2MG16M	5,0,3,0	Pass
iso-18-6VP8PSDZ	hsa-mir-3195	miRBasev21	isomiR	8	25	0	+	.	MG16M	0,0,5,0	Pass
iso-22-66P5J2RIM	hsa-mir-3194	miRBasev21	isomiR	9	30	0	+	.	22M	0,2,3,2	Pass
iso-23-W8539PJ4D6	hsa-mir-3192	miRBasev21	ref_miRNA	9	31	0	+	.	23M	0,0,0,2	Pass
iso-22-W8539PJ4L	hsa-mir-3192	miRBasev21	isomiR	9	30	0	+	.	22M	0,0,0,2	Pass

Showing 1 to 10 of 17,775 entries Previous 1 2 3 4 5 ... 1778 Next

Figura 3.8: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo normalizado mostrando los gráficos de las filas seleccionadas.

3.2.2 Datos de prueba no-normalizados

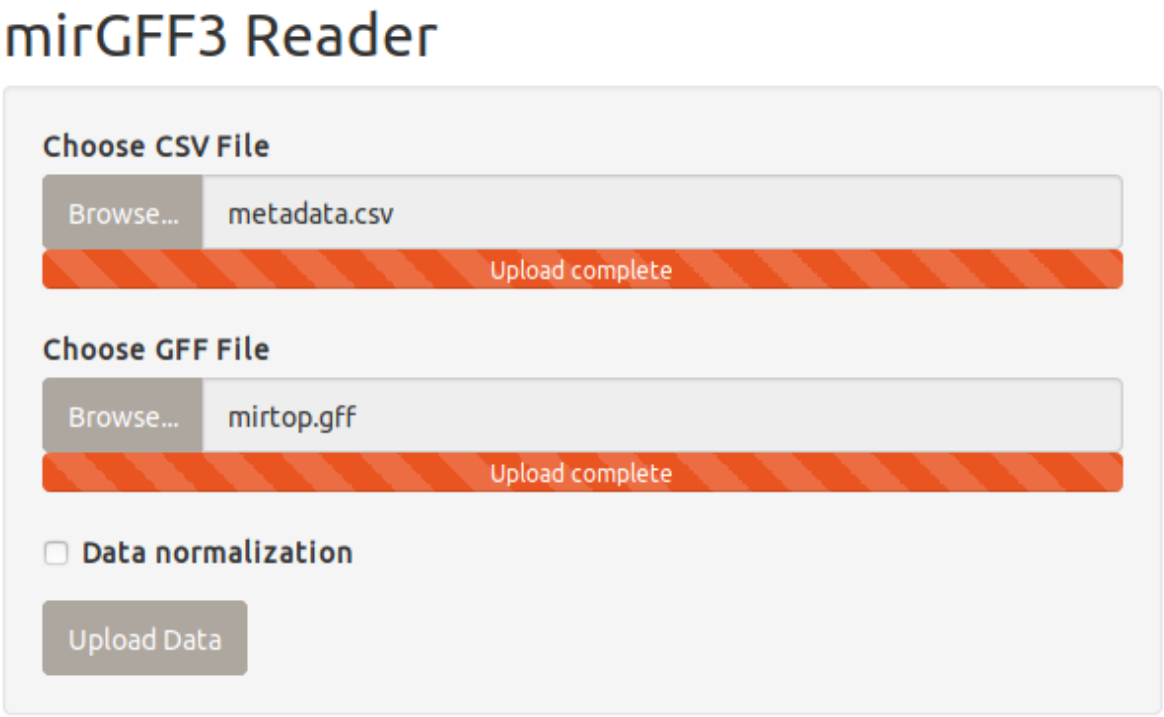


Figura 3.9: Detalle de la carga de datos con la pestaña de normalización desactivada en la aplicación “mirGff3 Reader”.

En este caso des-seleccionamos la pestaña de normalización de los datos y cargamos de la misma manera el archivo *mirGFF3* y *csv* de testeo.
Podemos ver que la información del objeto *SummarizedExperiment* sigue igual que en el caso de que seleccionemos la normalización de datos. Esto es debido a que al querer diseñar un código reactivo y que podamos modificar mientras lo analizamos se procesan los diferentes assays de *SummarizedExperiment* automáticamente al cargar los datos estos incluyen como explicamos anteriormente: los datos sin normalizar (assay(raw)), los normalizados (assay(*vst)*) y el tercer *assay* que representa el que está en uso.

Figura 3.10: Pestaña principal de información del objeto SummarizedExperiment en la aplicación “mirGff3 Reader” con el conjunto de datos de testeo no-normalizados.

La PCA de datos sin normalizar difiere de la anterior al no controlar los datos que están más alejados de la media o con varianzas de gran tamaño.

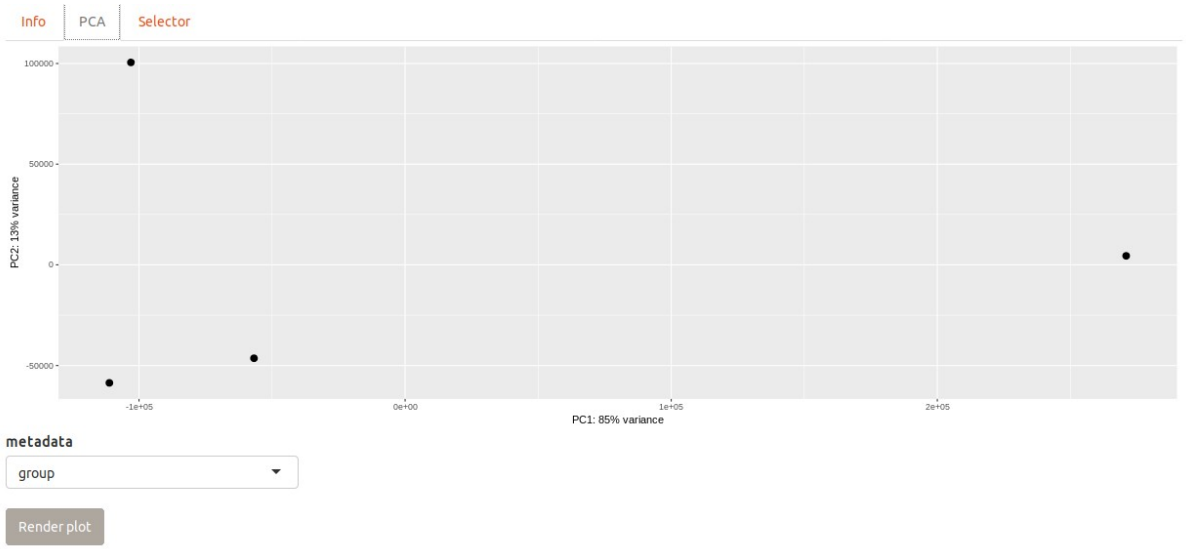


Figura 3.11: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo no -normalizados.

Lo mismo sucede al seleccionar la columna de metadatos y establecer la PCA en virtud de las variables dentro de la misma.

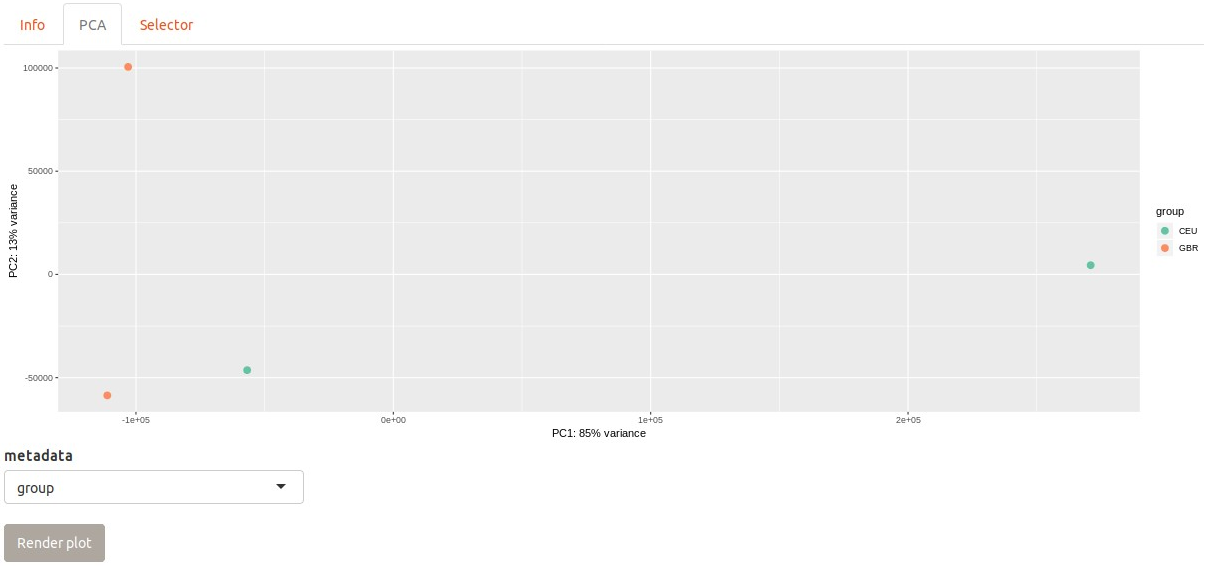


Figura 3.12: Pestaña de PCA en colores dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo no-normalizados.

En el caso del selector de isomiR también hay variación debido a los outliers que puedan haber en el conjunto de datos. Esto puede no verse a simple vista al observar los valores de los diferentes atributos pero sí se aprecia al realizar las gráficas.

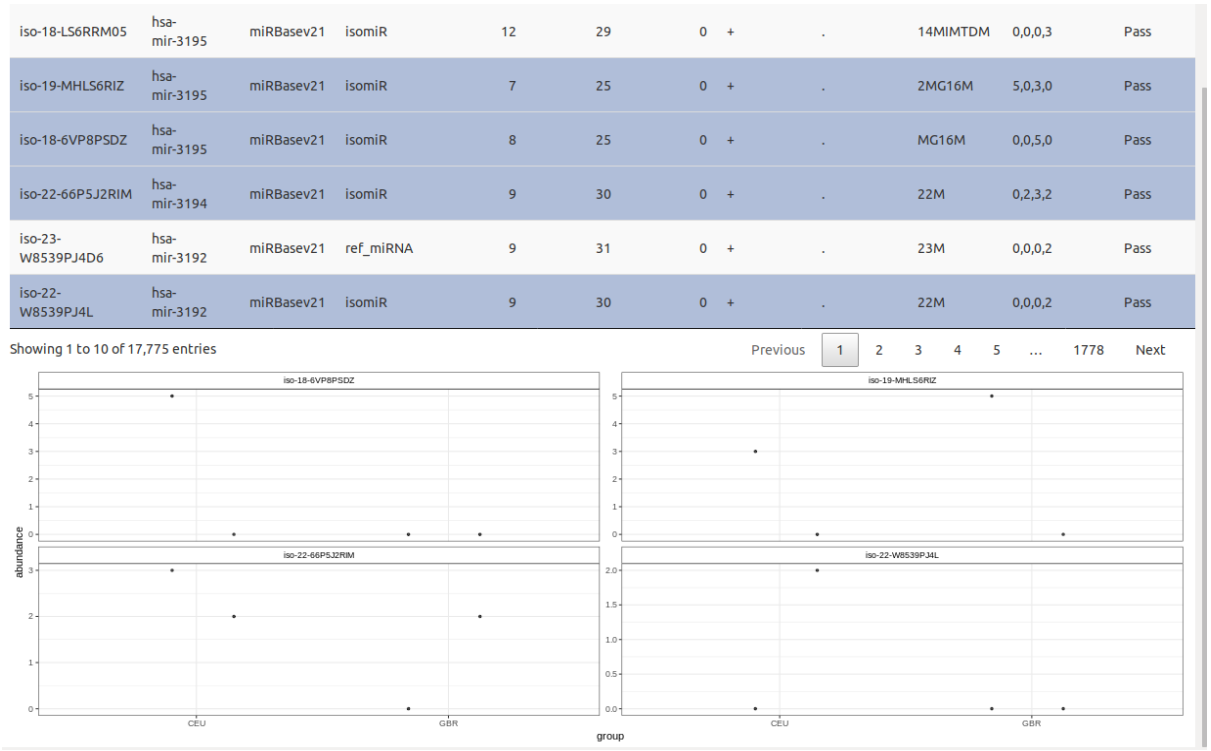


Figura 3.13: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de testeo no-normalizados.

3.2.3 Datos finales normalizados

El conjunto de datos final es ostensiblemente más grande que el de prueba y consta de un mayor número de muestras así como de metadatos.
En este caso al cargar los datos es necesario esperar mucho más para que los archivos se carguen así como para que se procesen.
Por lo demás la aplicación responde virtualmente de la misma manera y sin dar ningún error.

En la primera pestaña de información vemos que también se han procesado los tres assays para el objeto *SummarizedExperiment* así como las variables necesarias para el mismo.

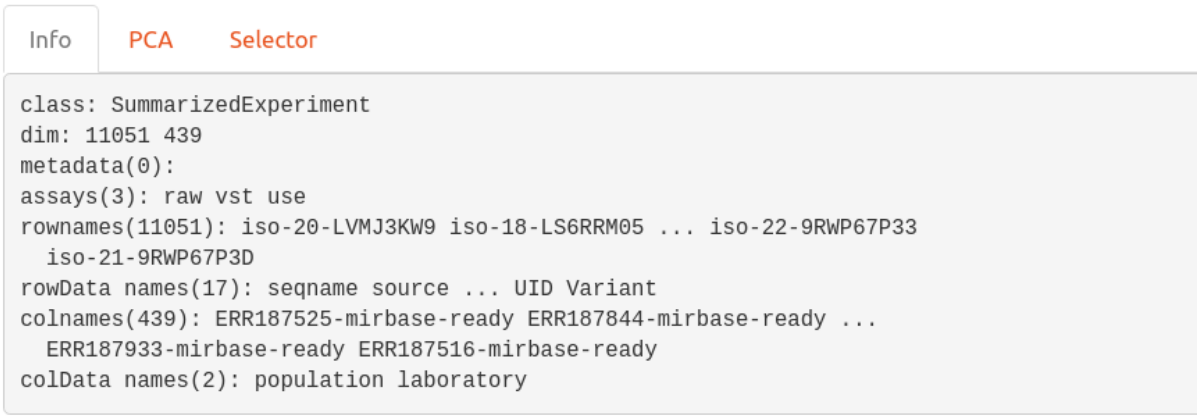


Figura 3.14: Pestaña principal de información del objeto SummarizedExperiment en la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizados.

Al desplegar la pestaña de la PCA vemos que el número de puntos es mucho mayor aunque al no tener seleccionadas ninguna de la variables para la línea de metadata desconocemos si se agrupan en función de este parámetro.

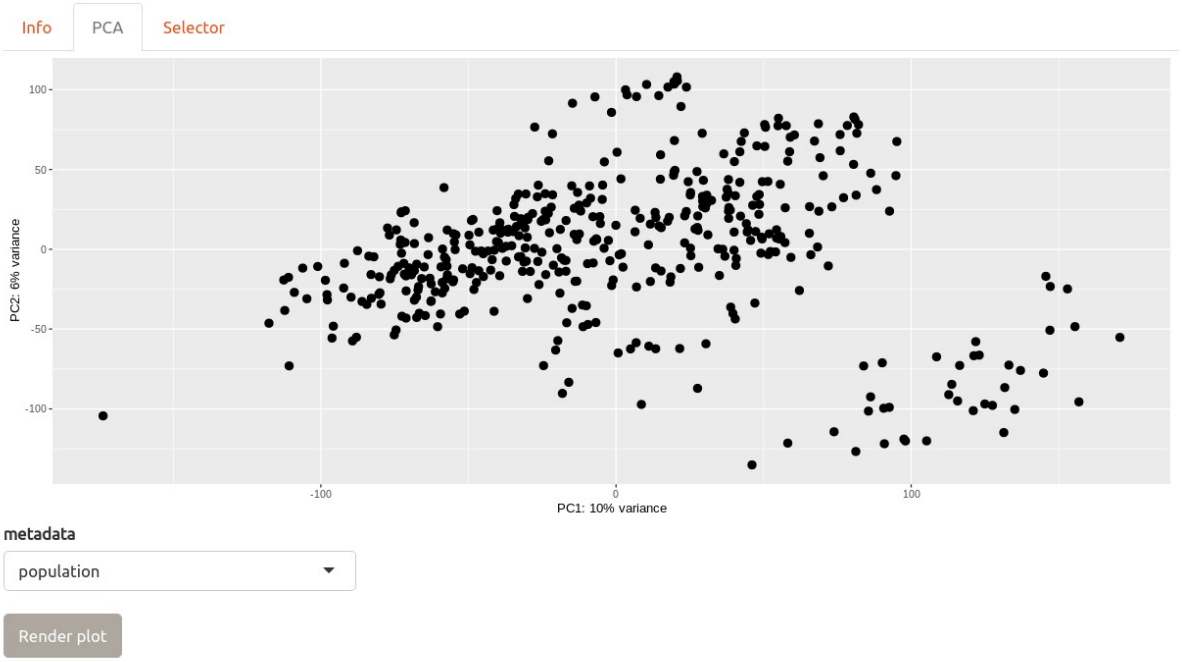


Figura 3.15: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizados.

Al seleccionar el metadato “*population*” el código de colores se encuentra mezclado sin aparecer una agrupación de los componentes principales a simple vista aunque sería necesario un análisis más concienzudo.

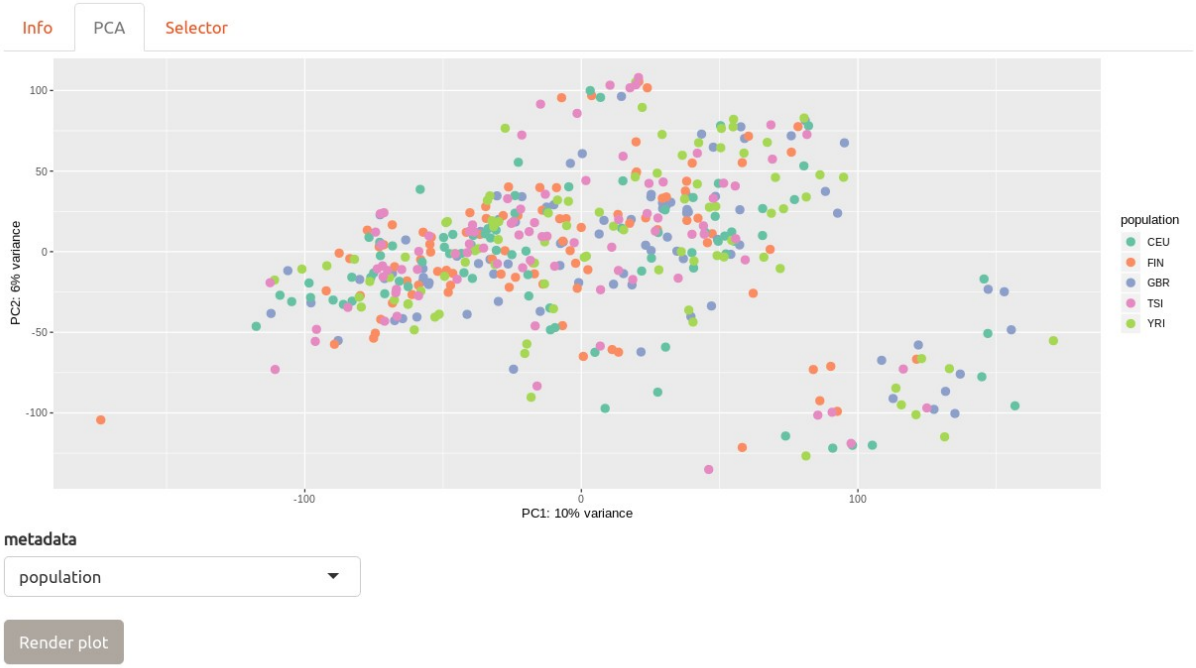


Figura 3.16: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizados y la metadata “population” seleccionada.

Sin embargo con el metadato “*laboratory*” podemos ver que si hay una agrupación de colores por lo que es buen indicativo de variable clasificatoria del conjunto de datos.

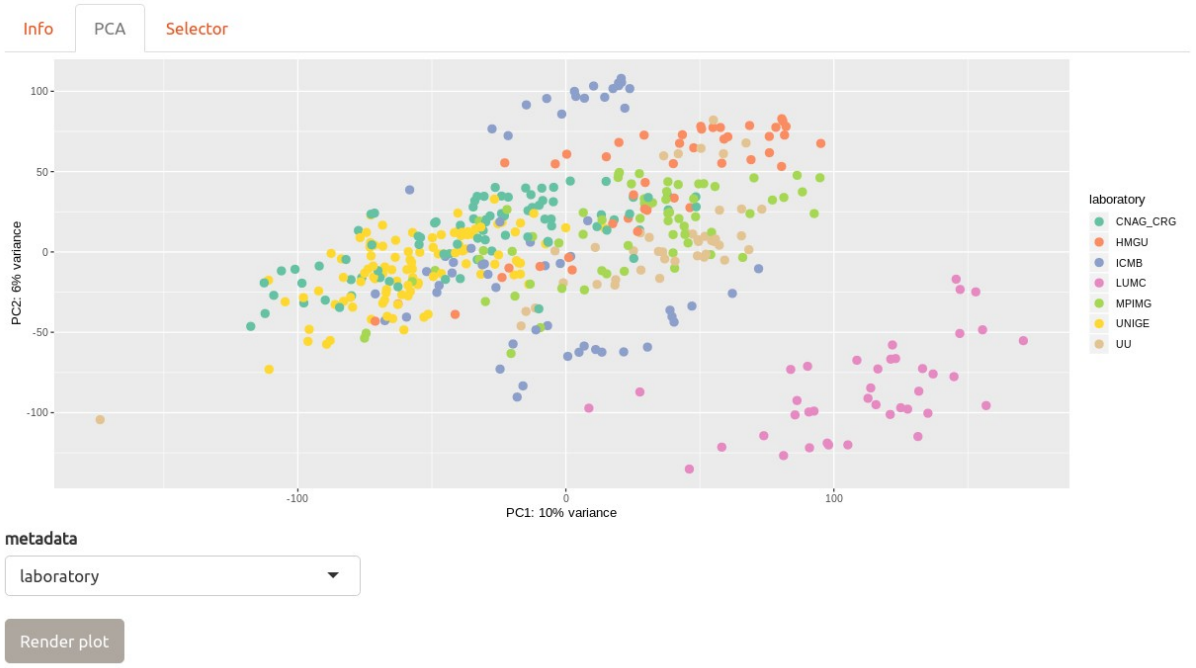
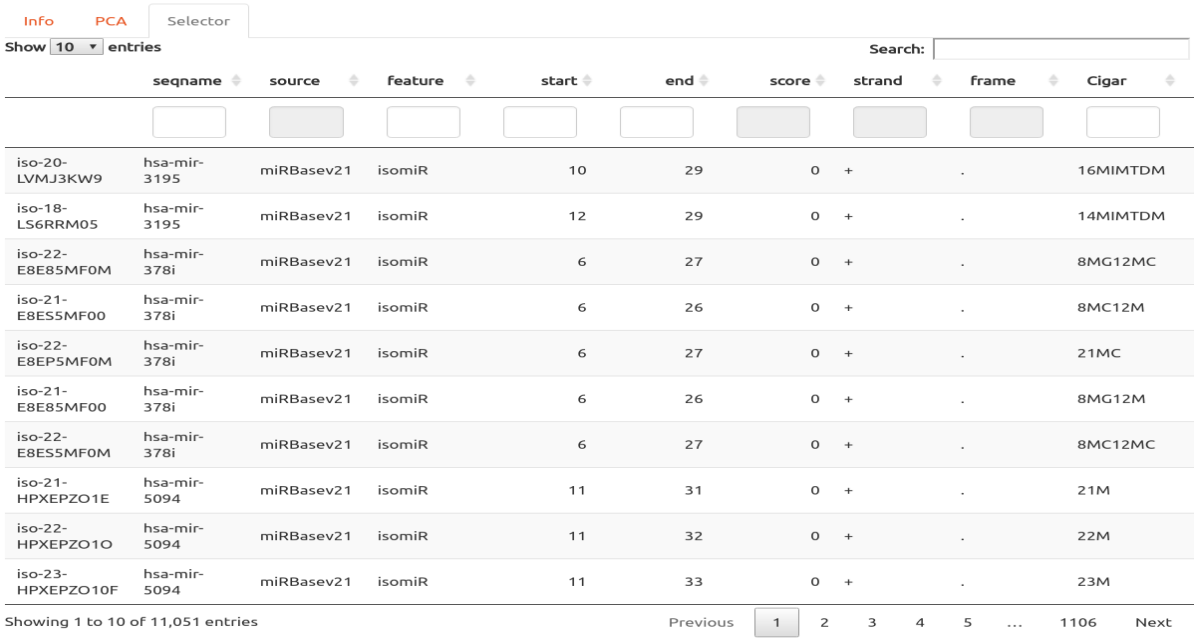


Figura 3.17: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizados y la metadata “laboratory” seleccionada.

En la última pestaña “Selector” la presentación de los isomiR es igual a los del conjunto de testeo ya que en esencia el formato de archivo es el mismo.



Info PCA Selector

Show 10 entries Search:

	seqname	source	feature	start	end	score	strand	frame	Cigar
iso-20-LVMJ3KW9	hsa-mir-3195	miRBasev21	isomiR	10	29	0	+	.	16MIMTDM
iso-18-LS6RRM05	hsa-mir-3195	miRBasev21	isomiR	12	29	0	+	.	14MIMTDM
iso-22-E8E85MF0M	hsa-mir-378i	miRBasev21	isomiR	6	27	0	+	.	8MG12MC
iso-21-E8ES5MF00	hsa-mir-378i	miRBasev21	isomiR	6	26	0	+	.	8MC12M
iso-22-E8EP5MF0M	hsa-mir-378i	miRBasev21	isomiR	6	27	0	+	.	21MC
iso-21-E8E85MF00	hsa-mir-378i	miRBasev21	isomiR	6	26	0	+	.	8MG12M
iso-22-E8ES5MF0M	hsa-mir-378i	miRBasev21	isomiR	6	27	0	+	.	8MC12MC
iso-21-HPXEPZO1E	hsa-mir-5094	miRBasev21	isomiR	11	31	0	+	.	21M
iso-22-HPXEPZO1O	hsa-mir-5094	miRBasev21	isomiR	11	32	0	+	.	22M
iso-23-HPXEPZO10F	hsa-mir-5094	miRBasev21	isomiR	11	33	0	+	.	23M

Showing 1 to 10 of 11,051 entries Previous 1 2 3 4 5 … 1106 Next

Figura 3.18: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizados.

Los gráficos sin embargo corroboran el gráfico de PCA anterior al mostrar las diferentes muestras de cada isomiR agrupadas en función de la variable “*laboratory*”.

iso-21-E8ES5MF00	hsa-mir-378i	miRBasev21	isomiR	6	26	0	+	.	8MC12M	0,
iso-22-E8EP5MF0M	hsa-mir-378i	miRBasev21	isomiR	6	27	0	+	.	21MC	0,
iso-21-E8E85MF00	hsa-mir-378i	miRBasev21	isomiR	6	26	0	+	.	8MG12M	0,
iso-22-E8ES5MF0M	hsa-mir-378i	miRBasev21	isomiR	6	27	0	+	.	8MC12MC	0,
iso-21-HPXEPZO1E	hsa-mir-5094	miRBasev21	isomiR	11	31	0	+	.	21M	0,
iso-22-HPXEPZO1O	hsa-mir-5094	miRBasev21	isomiR	11	32	0	+	.	22M	2,
iso-23-HPXEPZO10F	hsa-mir-5094	miRBasev21	isomiR	11	33	0	+	.	23M	0,

Showing 1 to 10 of 11,051 entries Previous 1 2 3 4 5 … 1106 Next

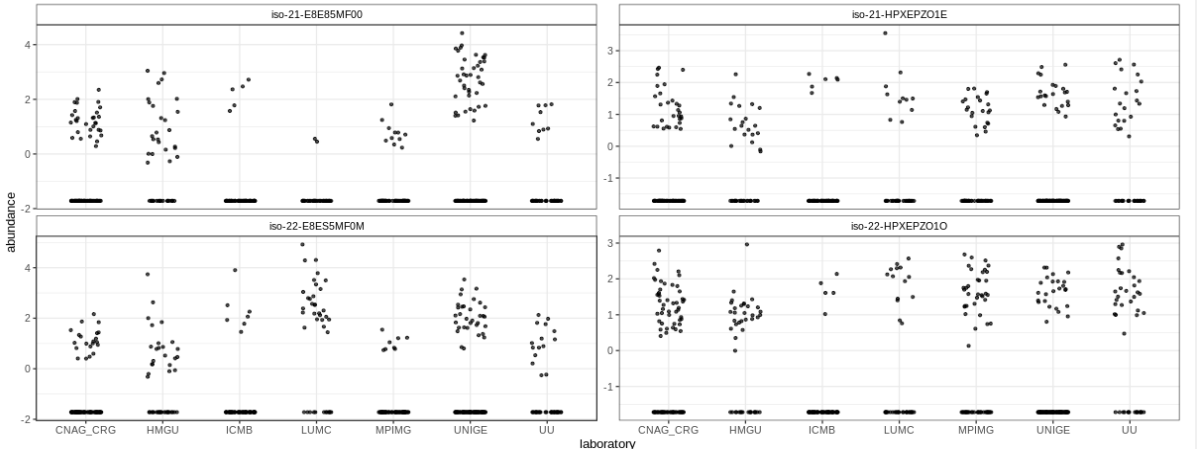


Figura 3.19: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales normalizado mostrando los gráficos de las filas seleccionadas.

3.2.4 Datos finales no-normalizados

Como hemos explicado anteriormente tanto la constitución de los diferentes assays de SummarizedExperiment como el resumen mostrado en la pestaña de información no varía con respecto al objeto cuando empleamos los datos normalizados.

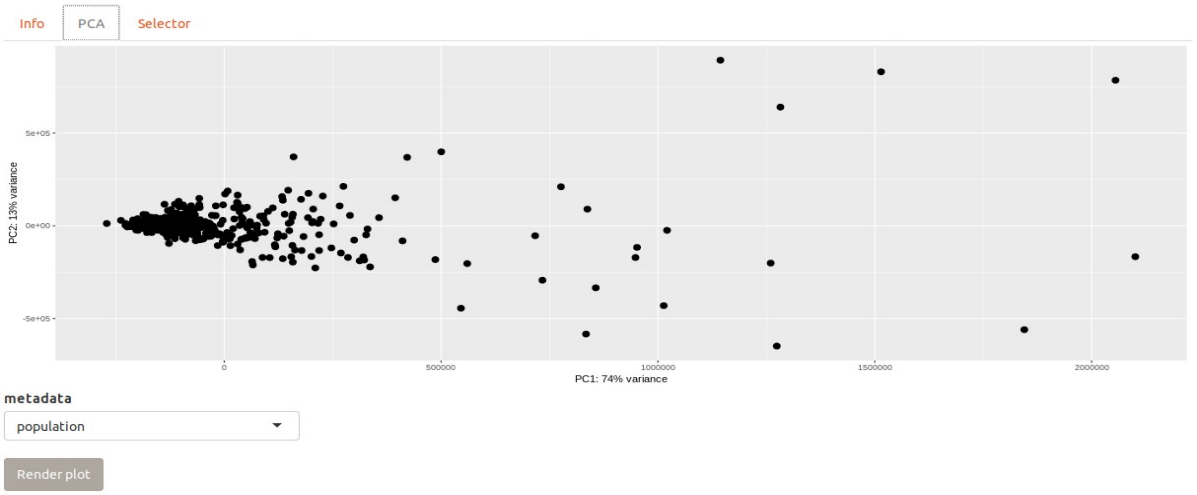


Figura 3.20: Pestaña de PCA en colores dentro de la aplicación “mirGff3 Reader” con el conjunto de datos de finales no- normalizado.

Al observar la pestaña de PCA vemos que difiere con respecto al conjunto de datos normalizado y se agrupan mucho más.

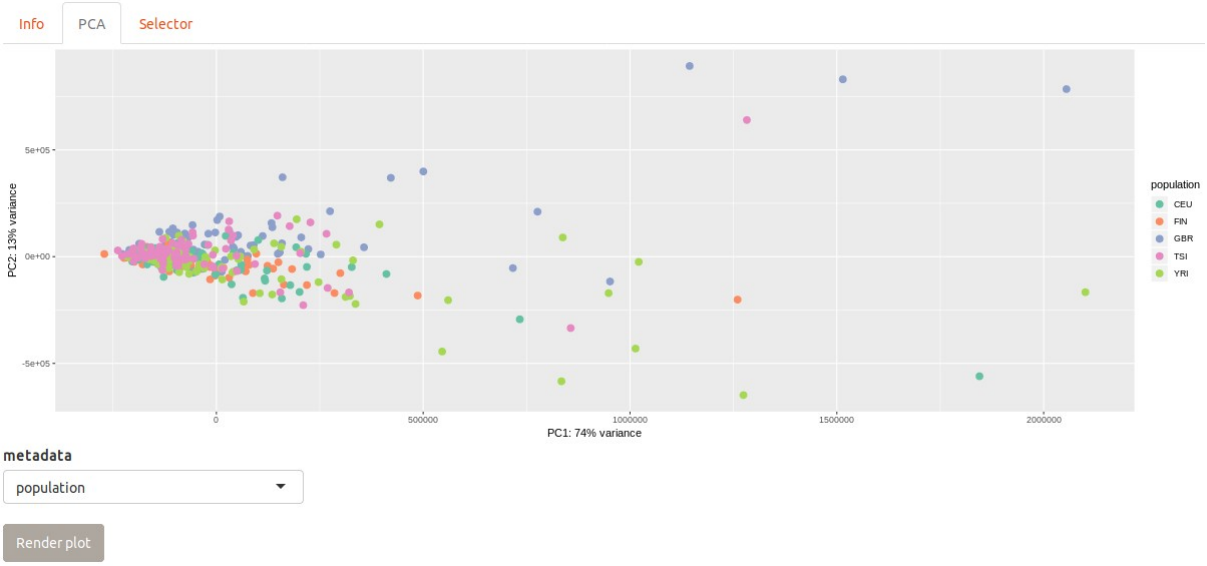


Figura 3.21: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales no- normalizados y la metadata “population” seleccionada.

En la PCA diferenciada por población con respecto a la por laboratorios se aprecia que la

distribución de puntos no varía y podría haber una correspondencia entre alguno de los laboratorios y los conjuntos de población aunque esto requeriría pruebas posteriores para que se comprobara.

Figura 3.22: Pestaña de PCA dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales no- normalizados y la metadata “laboratory” seleccionada.

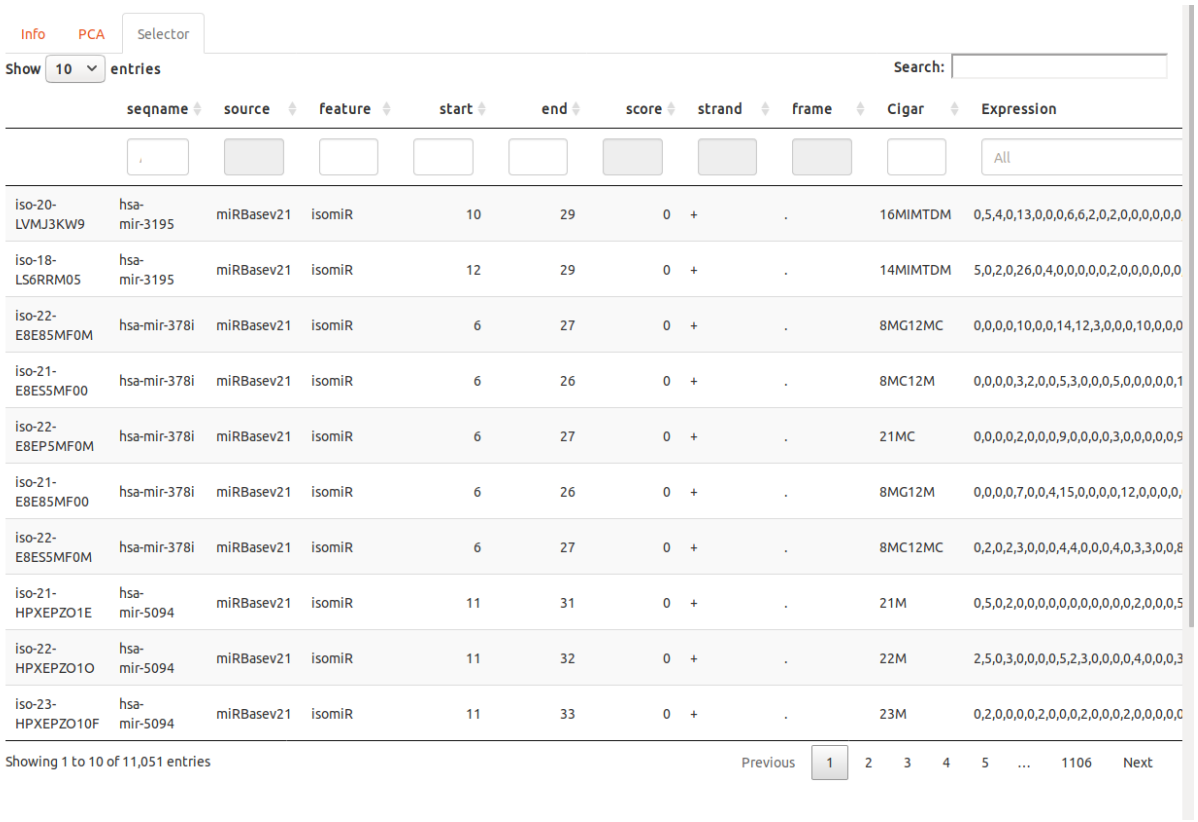


Figura 3.23: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales no-normalizados.

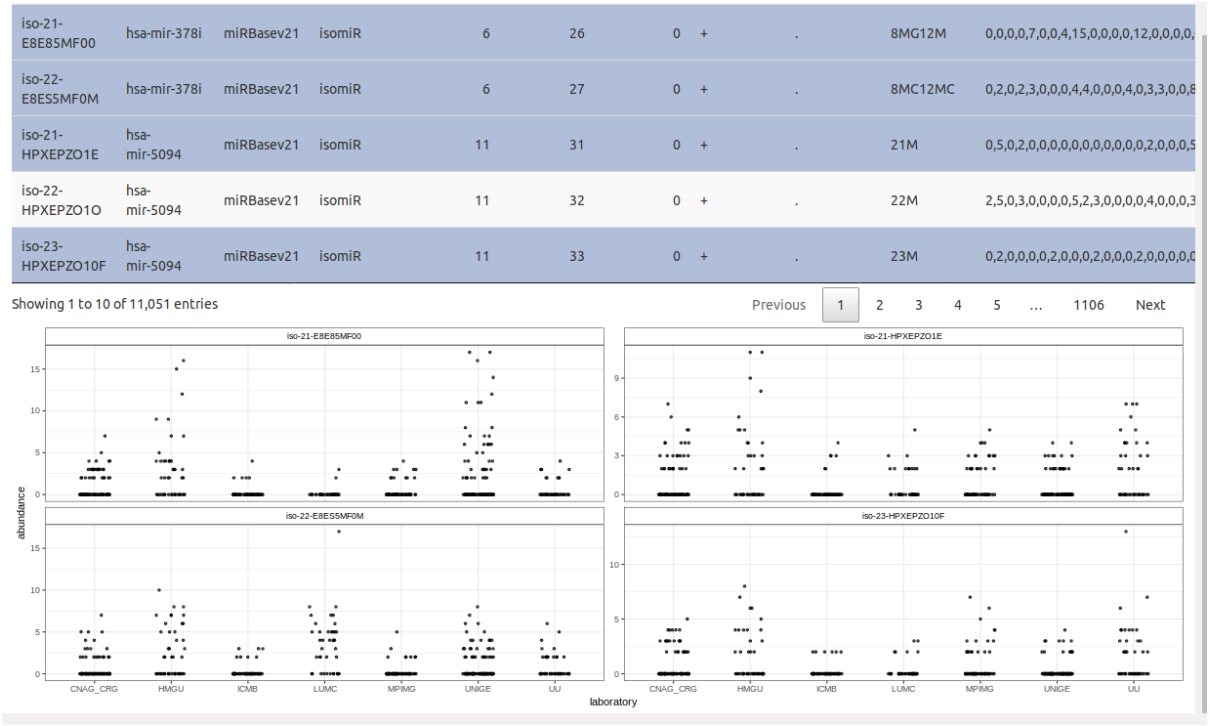


Figura 3.24: Pestaña del selector de isomiRs dentro de la aplicación “mirGff3 Reader” con el conjunto de datos finales no-normalizados mostrando los gráficos de las filas seleccionadas.

En la pestaña del selector podemos ver como las gráficas de los diferentes isomiR difieren de cuando los datos están normalizados al igual que sucedía en el conjunto de datos de testeo.

3.3 Rendimiento

A la hora de media la carga de procesamiento de los datos es necesario indicar las especificaciones de ordenador en el que se han llevado a cabo los cálculo. En este caso se trata de un:

```
Intel i7-7700HQ 4 núcleos y 8 hilos

RAM DDR4 2400 en 8GB
```

A continuación hemos cronometrado el tiempo de carga de los archivos de datos (principalmente el mirGFF3) y el tiempo de procesamiento posterior.

En el caso de los archivos de **datos de testeo** el tiempo de carga de los archivos ha sido inapreciable y el tiempo de procesamiento ha sido de:

Si tenemos en cuenta el peso del archivo de 4MB

En el caso de los de los archivos de **datos finales** la carga de los mismos dentro de la aplicación ha tardado un total de:

y el procesamiento de los mismos ha tardado:

Si comparamos el peso de los datos de testeo de 4MB y 9 segundos de tiempo con el de los datos finales de 167,4MB y 438 segundos vemos que la relación no es exactamente lineal entre peso del archivo y tiempo de procesamiento entre estos dos conjuntos aunque si próxima.

Datos de testeo: **0,44**

Datos finales: **0,38**

Por lo que sería recomendable tener en cuenta más factores como el número de variables en los metadatos, estos últimos y el número de líneas en los archivos.

4

Conclusiones y trabajo futuro

Una vez finalizado este proyecto podemos afirmar que el desarrollo de esta aplicación colaborará con la consolidación de mirGFF3 como formato unitario para el análisis de miRNA y apoyará el desarrollo de herramientas de código abierto que permitan unificar líneas de estudio así como facilitar la eliminación de barreras de tipo técnico en este ámbito de investigación.

Otro aspecto del programa es el diseño en diferentes pestañas de información y la estructura casi modular que permite Shiny y tidyR hacen que modificar el código para añadir implementaciones sea más sencillo que con otros lenguajes de programación o entornos.

Por otro lado la prueba final con los datos del proyecto de investigación *Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories* desarrollada por Peter A C ’t Hoen, Marc R Friedländer et al. dentro del proyecto GEUVADIS y el enorme volumen de datos que representa sobretodo en comparación con los datasets que se suelen analizar en un día a día normal enfatiza la versatilidad y capacidad de la aplicación, dicho esto, e independientemente de la potencia de cálculo a la que optemos podemos afirmar que si se dispone de una máquina relativamente actual no debería haber ningún impedimento dentro del diseño del programa para trabajar con casi cualquier conjunto de datos, más aún con los de menor peso que se suelen estilar.

Dentro de los planes de futuro está el ampliar la capacidad del programa para introducir el análisis de expresión diferencial así como incrementar las opciones de las funcionalidades existentes.

Agradecimientos

A mi tutora Lorena por arriesgarse conmigo otro año, por su paciencia y por lo mucho que he aprendido de ella durante el transcurso de este master.
A mi madre y a Eva, por aguantarme los malos humores y las miradas perdidas mientras pensaba cómo hacer funcionar un pedazo de código rebelde.

Glosario de términos

BAM: Formato binario para almacenar datos de secuenciación.

Bioconductor: Conjunto de librerias y paquetes para el lenguaje de programación R que se centra en el análisis de datos genómicos.

CSS: Acrónimo para “Cascading Style Sheets”, es un lenguaje que permite diseñar y modificar la presentación visual de un documento web.

Dicer: *Ribonucleasa presente en el citoplasma que se encarga de cortar el pre-microRNA durante su procesamiento.*

DGCR8: *Proteína requerida para el procesamiento de miRNAs en el nucleo celular y que se une a Drosha para cortar el pri-miRNA .*

Drosha: *Enzima ribonucleasa que se une a DGCR8 en el nucelo celular formando un complejo encargado de la maduración del miRNA.*

GFF3: Acronimo para «General Feature Format versión 3». Es un formato de archivo para miRNAs que consiste en una linea por elemento cada una con 9 columnas que contemplan parámetros como el origen de la muestra, la hebra de la que proviene, el identificador de la misma, etc.

HTML5: Acrónimo para “HyperText Markup Language, versión 5”. Lenguaje básico de la World Wide Web.

IDE:Acrónimo para “Integrated Development Environment”, aplicación que ofrece un entorno para el desarrollo de programas.

isomiR: Secuencias de miRNA con variaciones respecto a las secuencias de referencia debido a adiciones, deleciones o substituciones de bases nitrogenadas con respecto a la secuencia inicial.

isomir-SEA: Herramienta capaz de extraer niveles de expresión de miRNAs a partir de una alta tasa de transferencia efectiva.

JavaScript: Lenguaje de programación orientado a objetos y que forma parte de un navegador web permitiendo de esta manera la interacción con páginas dinámicas.

Linfoblastoide: Células con modificaciones morfológicas precursoras de los Linfocitos.

miRge2: Herramienta para el análisis secuencial de miRNA.

mirGFF3: Formato para el análisis de miRNA derivado del formato GFF3.

miRNA: También denominados microRNA. Se trata de pequeñas moléculas de ARN monocatenario con una longitud de entre 20 y 27 nucleótidos y con la capacidad de regular la expresión de otros genes.

mirTOP: Acrónimo para “miRNA Transcriptomic Open Project”,

Número de acceso: En el campo de la bioinformática se trata de un identicador único que se otorga a una secuencia de proteinas o de DNA y que permite el seguimiento de las diferentes versiones de dicha secuencia.

PROST!: Acrónimo para «Processing Of Small Transcripts», es una aplicación basada en python que cuantifica y anota el grado de expresión de miRNA.

R: Lenguaje de programación orientado a objetos desarrollado por R. Gentleman y R Ihaka, y diseñado para el análisis estadístico.

reactividad: Propiedad en la que al cambiar la entrada o input del programa se cambia el valor de salida o output en virtud de esta modificación.

Rstudio: Entorno de desarrollo integrado (IDE) para lenguaje R.

seqbbuster: Pipeline para el análisis de miRNA.

sRNAbench: Aplicación para el procesamiento de miRNA obtenidos de plataformas como illumina y SOILiD

tidy data: Definimos así a conjuntos de datos donde cada observación es una fila, cada variable una columna y cada valor una celda de la tabla.

Widget: Aplicación de pequeño tamaño que permiten el empleo de funciones de uso frencuente.

Bibliografía

[1] Rstudio Homepage, https://www.rstudio.com/

[2] Kevin-Rue-Albretch, Federico Marini, Charlotte Soneson, Aaron T.L. Lunes (2018): *SEE: Interactive SummarizedExperiment Explorer*, Publicación online. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6013759/

[3] Thomas Desvignes, Phillipe Loher, Karen Eilbeck, Jeffery Ma, Gianvito Urgese, Bastian Fromm, Jason Sydes, Ernesto Aparicio-Puerta, Victor Barrera, Roderic Espín, Eric Londin, Aristeidis G. Telonis, Elisa Ficarra, Marc R. Friedländer, John H. Postlethwait, Isidore Rigoutsos, Michael Hackenberg, Ioannis S. Vlachos, Marc K. Halushka, Lorena Pantano (2018),*Unification of miRNA and isomiR research: the mirGFF3 format and the mirtop API*, Unification Biorxiv.

[4]Lincoln Stein. *Generic Feature Format Version 3 (GFF3)*,(26 Febrero 2013): https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md

[5]Gereric Model Organism Database (GMOD): http://gmod.org/wiki/GFF3

[6]*GFF3 File Format - Definition and supported options:* https://www.ensembl.org/info/website/upload/gff3.html#fields

[7]Manuel Oviedo de la Fuente , *Shiny: crear una aplicación web interactiva desde R*,(24 noviembre 2015): https://rpubs.com/moviedo/shiny

[8]Bioconductor home page: https://www.bioconductor.org

[9]Shiny home page: https://shiny.rstudio.com/

[10]OxShex: Shiny: https://shiny.oxshef.io/tutorials_shiny101.html

[11]Dean Attali's R blog: *Building Shiny apps – an interactive tutorial:* https://www.r-

bloggers.com/building-shiny-apps-an-interactive-tutorial/

[12]Weicheng Zhu, *Shiny Tutorial:* https://bookdown.org/weicheng/shinyTutorial/

[13]Michael Love (2019), *RNA-seq workflow: gene-level exploratory analysis and differential*
expression. Publicación online:
https://bioconductor.org/packages/release/workflows/html/rnaseqGene.html

[14]Yu G, Wang L, Han Y, He Q (2012). "*clusterProfiler: an R package for comparing biological themes among gene clusters.*" OMICS: A Journal of Integrative Biology

[15] GitHub Guides: https://guides.github.com/

[16] Martin Morgan, Valerie Obenchain, Jim Hester, Hervé Pagès *Package "SummarizedExperiment"* (2019)

[17] Paul Teetor, R Cookbook (2011) O'Reilly Media

[18] Peter A C 't Hoen , Marc R Friedländer , Jonas Almlöf , et.al. *Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories*. (2013) Nature biotechnology.

[19] Que són los miRNA? http://www.feedbackciencia.com/que-son-los-micrornas/

[20] Fazli Wahid, Adeeb Shehzad, et. al. MicroRNAs: Synthesis, mechanism, function, and recent clinical trials, Biochimica et Biophysica Acta (2010)

[21] Tidyverse https://www.tidyverse.org/

[22] DESeqDataSet
https://www.rdocumentation.org/packages/DESeq2/versions/1.12.3/topics/DESeqDataSet-class