Universidad de Granada

Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores

Biología Computacional con Big Data-Omics e Ingeniería Biomédica

---

Guión II: Análisis de Expresión Diferencial empleando KnowSeq mediante Docker

---

Profesores:
Daniel Castillo Secilla
Luis Javier Herrera Maldonado
Ignacio Rojas Ruiz

6 de febrero de 2020

Índice

1. Objetivos

Durante el desarrollo de este guión se pretenden alcanzar los siguientes objetivos principales:

- Familiarizarse con el análisis de expresión diferencial con RNA-seq.
- Conocer y utilizar el entorno R, así como KnowSeq, un paquete R desarrollado para llevar a cabo análisis de expresión diferencial y de aprendizaje automático para datos de RNA-Seq.
- Familiarizarse con el uso de Docker, una herramienta cada día más usada para encapsular softwares y servicios, evitando problemas de dependencias y versiones.
- Llevar a cabo un análisis para el conjunto de datos de Lung Cancer disponible en SWAD.

2. Requisitos

Es necesario disponer de los siguientes pre-requisitos:

- Instalación de Docker en el host.
- Obtención de la imagen de KnowSeq disponible para Docker.
- Descarga del conjunto de datos disponible en SWAD.

3. Introducción

El principal objetivo del presente guión es la realización de un análisis completo de datos de RNA-seq, y más concretamente de expresión diferencial. RNA-seq está resultando de gran utilidad en esta última década para el conocimiento de la funcionalidad de sistemas biológicos completos a nivel genómico y molecular. Antes de la inclusión en el mercado de esta tecnología, estos estudios se realizaban mediante el uso de microarray, una tecnología muy confiable y de la que han salido un gran número de estudios, pero menos potente y precisa que RNA-seq [1].

Dentro del desarrollo de las tecnologías para el estudio de las diferentes ómicas, el proyecto Bioconductor ha supuesto un gran incremento en la utilización de estas en el estudio de dichas ómicas, principalmente por su apuesta por facilitar el análisis de datos de tanto de microarray como de datos provenientes de NGS (RNA-seq, DNA-seq, CHiP-seq, etc. . .) combinando además el conocimiento de diferentes áreas como la estadística, la informática o la biología molecular [2].

Una de las mayores utilidades de RNA-seq son los análisis de expresión diferencial. El objetivo de estos análisis es la identificación de conjuntos de genes que se expresan a diferentes niveles bajo diferentes condiciones [3].

Este tipo de estudios permiten detectar aquellos genes que pueden estar asociados a diferentes fenotipos, por ejemplo, células tumorales frente a células sanas. Sin embargo, incluso para los análisis más simples, es necesario tener especial cuidado en las estrategias elegidas para realizar una selección adecuada de los genes involucrados en la expresión diferencial.

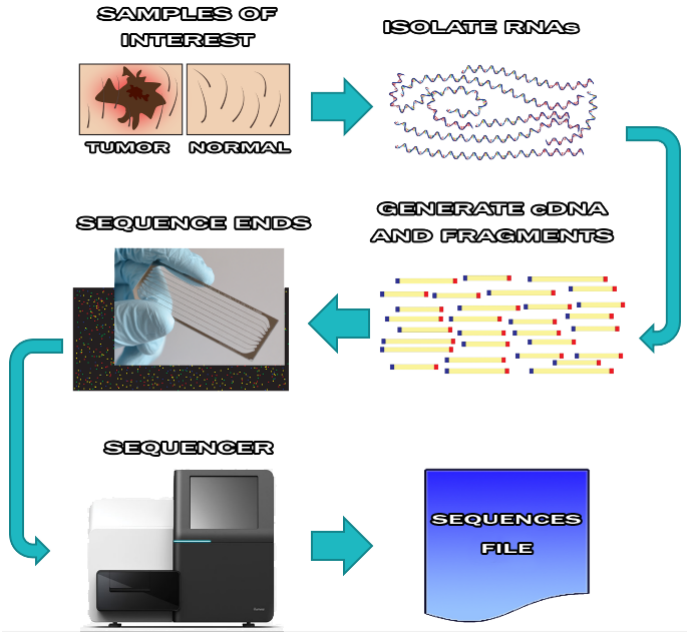

Figura 1: **Proceso de secuenciación de muestras de NGS. A partir de muestras de RNA procedentes del tejido deseado, se genera los fragmentos de ADN complementario a secuenciar. Dichos fragmentos son introducidos en un secuenciador que, finalmente crea un fichero (fastq) con los fragmentos de ADN secuenciados, dicho fichero está en "crudo" y necesita un preprocesamiento hasta llegar a los ficheros count, que finalmente son los que se usan para el estudio transcriptómico y de expresión de gen.**

4. KnowSeq. Software automatizado para análisis de RNA-Seq

Para llevar a cabo dichos análisis, se ha desarrollado una herramienta actualmente pública en el repositorio más importante en Bioinformática (Bioconductor). Dicha herramienta es conocida como KnowSeq y se diseñó con el propósito de dar a los expertos un paquete software que encapsule todos los pasos necesarios para llevar a cabo análisis de biomarcardores y su posterior evaluación mediante Machine Learning.

Dicha herramienta es totalmente modular y permite modificar el pipeline propuesto dependiendo del uso y del estudio que se quiera abordar. Para más información acerca de KnowSeq. se puede visitar la página oficial de KnowSeq en Bioconductor (https://bioconductor.org/packages/release/bioc/html/KnowSeq.html), así como su manual de usuario, que desgrana paso a paso cada uno de los procesos incluidos en ella (https://bioconductor.org/packages/release/bioc/vignettes/KnowSeq/inst/doc/KnowSeq.pdf).

Para tener una idea acerca de los procesos y sub-procesos implementados en la herramienta, la Figura 2 muestra gráficamente todo el pipeline y posibilidades que ofrece KnowSeq.

Para instalar localmente KnowSeq, se requiere la instalación de las herramientas de Bioconductor que permiten gestionar la instalación de los paquetes software que dicho repositorio contiene. Además, KnowSeq está disponible para R 3.6 y superiores, lo que implica que en una versión anterior no se podrá instalar. Para llevar a cabo la instalación tanto de Bioconductor como de KnowSeq es necesario introducir en R el siguiente código:

```
# Instalamos BiocManager de Bioconductor
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

# Instalamos KnowSeq
BiocManager::install("KnowSeq")
```

No obstante, se ha creado una versión encapsulada de la herramienta mediante Docker, sacando provecho de este método de virtualización software para evitar la instalación de dependencias necesarias. Para poder llevar a cabo la instalación de Docker en Windows es necesario contar con Windows 10 Pro o enterprise debido a que la tecnología Hyper-V es requerida para poder ejecutar correctamente Docker. Por ello, si no se tiene una de dichas versiones de Windows, se recomienda instalar KnowSeq localmente o crear una Máquina Virtual para ello.

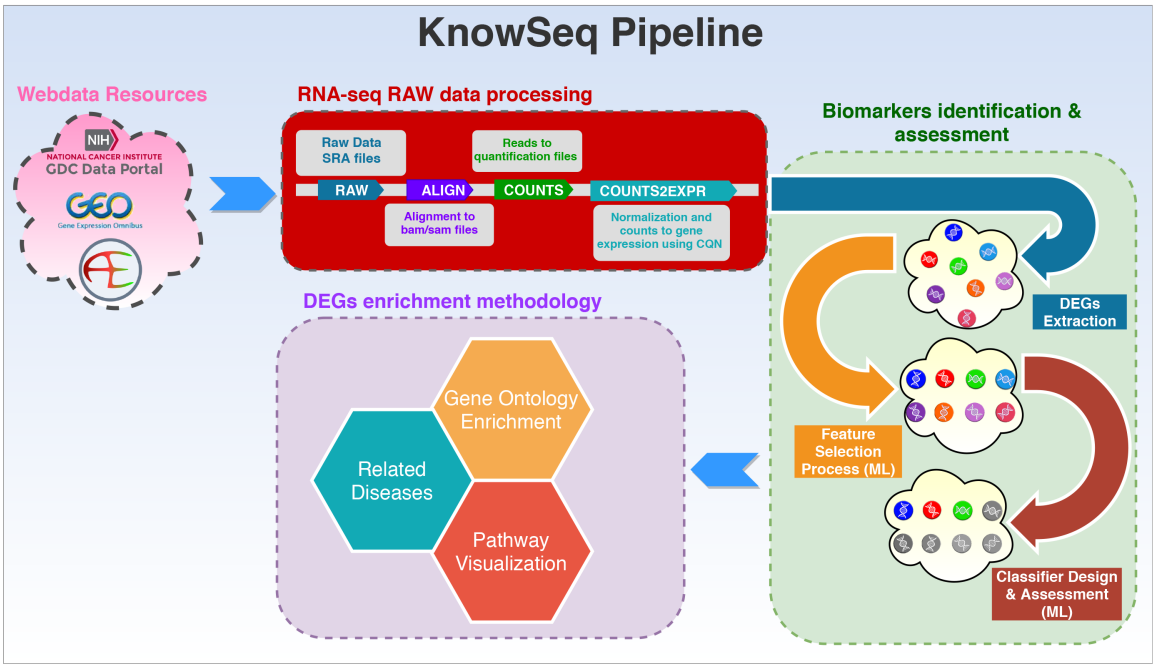

Figura 2: **Pipeline implementado en la herramienta KnowSeq.**

5. Preparando Docker

La idea detrás de Docker es crear contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues.

La principal diferencia con una Máquina Virtual es que los recursos del host son compartidos con el contenedor, evitando así el consumo de recursos que genera una Máquina Virtual al virtualizar el Hardware necesario.

Aunque Docker se pensó para contener pequeños microservicios (Pequeños servicios como BBDD, servidores web, o procesos para controlar tareas concretas), puede usarse y se usa para contener también despliegues de aplicaciones mayores e incluso Sistemas Operativos.

5.1. Instalando Docker

Paso 1: Instalando docker

Instalar Docker descargándolo mediante su sitio web oficial o, si se usa un entorno GNU/Linux mediante los siguientes comandos:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg —
sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"

sudo apt get update

apt search docker-ce; sudo apt install docker-ce

sudo docker run hello-world

Para instalar Docker es necesario seguir las instrucciones del cuadro indicativo del Paso 1. Una vez instalado, se probará el hello world de docker para asegurar que efectivamente esta todo correctamente instalado.

Una vez comprobada la instalación, el contenedor de KnowSeq puede ser descargado para trabajar con la herramienta. A modo de curiosidad, esta imagen contiene un Ubuntu encapsulado con la versión de R y de Bioconductor necesarias para instalar KnowSeq, además de todas las dependencias y librerías requeridas. Siguiendo el comando del cuadro indicativo del Paso 2, hará falta indicarle a Docker la imagen de KnowSeq, así como el parámetro -it para entrar al modo interactivo. Si el parámetro -it no se indica, el contenedor se desplegará y automáticamente se cerrara ya que no hay ninguna tarea indicada para que realice.

Paso 2: Descargando contenedor de KnowSeq

Para descargar el contenedor de KnowSeq, hará falta introducir el siguiente comando:

sudo docker run -it casedugr/knowseq:latest

Si ya se dispone de una imagen en .tar, solo habrá que cargarla mediante el siguiente comando:

sudo docker load -i knowSeqDockerImage.tar

Es necesario saber como comunicar al Host (Nuestro PC) con el contenedor debido a que para cargar los datos en el contenedor, estos han de ser enviados desde el Host. Para ello Docker provee de la función cp, la cual actúa de forma prácticamente idéntica al cp de entornos UNIX. Hay que tener en cuenta que por tema de permisos es recomendable crear una carpeta para recibir archivos desde el contenedor tales como gráficas, matrices de datos, etc... Para todo ello, los cuadros indicativos de los Pasos 3 y 4 muestran los comandos necesarios para todo ello.

Paso 3: Enviando los datos del Host al Contenedor

Para enviar los datos en SWAD al contenedor desde el Host es necesario los siguientes comandos:

sudo docker ps (Para consultar el Container ID)
sudo docker cp knowSeqData.zip
ContainerID:/~/workDir/knowSeqData.zip

Paso 3: Enviando ficheros del contenedor al Host

Primero es recomendable crear una carpeta específica para ello y dotarla de todos los permisos:

mkdir dockerFiles
sudo chmod 777 dockerFiles

Finalmente, se realiza la copia desde el contenedor al Host:

sudo docker cp ContainerID:/ /workDir/knowSeqData/grafica.png dockerFiles/grafica.png

6. Análisis de muestras de Lung Cancer con KnowSeq

Una vez KnowSeq esté disponible ya sea en local o mediante el despliegue del contenedor Docker, se va a realizar el análisis de expresión diferencial de las muestras de Lung cancer propuestas y subidas a SWAD. El objetivo principal de este análisis es encontrar un conjunto robusto de biomarcadores (Genes en este caso) con el potencial a la capacidad de discernir muestras de los grupos abordados en el análisis (Primary Tumor y Solid Tissue Normal). Este tipo de muestras son muy comunes cuando se realizan análisis de expresión de gen. Primary Tumor es tejido tumoral extraído directamente del tumor a estudiar. Por otro lado, Solid Tissue Normal es tejido normal adyacente al tumor del cual se ha extraído la muestra de Primary Tumor.

6.1. Preparando los datos

Lo primero que se ha de hacer una vez se ha descargado el dataset del SWAD es necesario descomprimirlo para extraer todos los ficheros count y el csv con la información de las muestras. Esto se puede hacer con una descompresión desde el propio R aprovechando justo después para indicar que el directorio de trabajo va a ser la carpeta descomprimida. Para ello, se ejecuta al siguiente código:

```
# Descomprimimos los datos
unzip("knowSeqData.zip")

# Listamos los archivos y comprobamos
list.files()

```

```
# Establecemos la carpeta descomprimida como directorio de trabajo
setwd("knowSeqData")

# Comprobamos que se ha establecido correctamente
list.files()
```

Una vez se tienen los datos correctamente extraídos y el directorio de trabajo fijado, se ha de leer el fichero *samplesSheetKnowseq.csv*, el cual contiene toda la información referente a los ficheros count y los labels de cada muestra. En dicho fichero, la columna File.ID representa el nombre de la carpeta de la muestra mientras que la columna File.Name representa el nombre del fichero count dentro de cada carpeta. Finalmente la columna Sample.Type nos indica el label de cada muestra (Primary Tumor o Solid Tissue Normal). El resto de columnas son meramente informativas y no son necesarias para llevar a cabo el análisis. Para más comodidad se crearan variables concretas para cada una de las tres columnas necesarias.

Posteriormente se creará también un DataFrame a partir de las variables que representan dicha información necesaria. Finalmente, se exportará el DataFrame creado a un CSV para tener esa información guardada y lista para los pasos posteriores. Todo ello se puede realizar ejecutando el siguiente código:

```
# Leemos el CSV
samplesInfo <- read.csv("samplesSheetKnowseq.csv", sep = ",")

# Creamos las variables necesarias
Run <- samplesInfo$File.Name
Path <- samplesInfo$File.ID
Class <- samplesInfo$Sample.Type

# Unificamos las muestras tumorales
Class <- gsub("Recurrent Tumor", "Primary Tumor", Class)

# Imprimimos numero de muestras por clase
table(Class)

# Exportamos DataFrame a CSV
write.csv(SamplesDataFrame, file = "SamplesDataFrame.csv")
```

6.2. Extrayendo los valores de expresión de gen

Una vez se tienen los datos cargados y el CSV creado, es necesario unir todos los count en una misma matriz para poder así trabajar con ella. Para ello, KnowSeq cuenta con la función *countsToMatrix*, la cual hace un merge de todos los ficheros aunándolos en una misma variable. Además, esta función también nos devolverá los labels de las muestras. Para ello se debe ejecutar el siguiente código:

```
# Cargamos y aunamos los ficheros count
countsInformation <- countsToMatrix("SamplesDataFrame.csv")

# Exportamos tanto la matriz de datos como los labels a nuevas
    variables
countsMatrix <- countsInformation$countsMatrix
labels <- countsInformation$labels
```

Cuando la matriz de count esta preparada, se puede empezar a trabajar con los datos de los genes propiamente dichos. Cada count está formado por dos columnas, la primera de ellas indica el identificador de cada gen según la notación Ensembl Id. La segunda columna indica el número de reads de cada gen o las veces que se lee ese gen en el genoma. Para el cálculo de los valores de expresión de gen hay que normalizar esos reads de cada gen en función del contenido de Guanina-Citosina (GC Content) de cada gen. Además hay que cambiar los Ensembl ID de los genes por los Gene Symbols, o el nombre propiamente de cada gen. Por ello, es necesario consultar para cada gen su gene Symbol y su GC Content. Este proceso puede hacerse mediante la función de KnowSeq *getAnnotationFromEnsembl*. A dicha función se le pasa los Ensembl IDs y el genoma de referencia con el que fueron alineadas las muestras (38 para este estudio). Una vez se tiene eso, mediante la función *calculateGeneExpressionValues* se extraen los valores de expresión de gen. Para llevar a cabo todo ello se debe ejecutar el siguiente código:

```
# Consultamos los Gene Symbols y el GC content de cada gen
myAnnotation <- getAnnotationFromEnsembl(rownames(countsMatrix),
    referenceGenome=38)

# Calculamos los valores de expresion usando la matriz de count y la
    anotacion previamente adquirida
expressionMatrix <- calculateGeneExpressionValues(countsMatrix,
    myAnnotation, genesNames = TRUE)
```

Lo ideal una vez se consigue la matriz de expresión es realizar un análisis de calidad en busca de posibles outliers. Para ello la función *RNAseqQA* incorpora diferentes métodos de búsqueda de los mismos. Esta función crea un directorio nuevo que contiene imágenes e información acerca de los diferentes test de calidad y los outliers considerados por cada test. No obstante, al contar con un número muy elevado de genes y muestras, este proceso se demorará mucho, por lo que es opcional en el desarrollo de este guión. Para hacer el análisis de calidad se ha de ejecutar el siguiente código:

```
# Realizamos el analisis de calidad
RNAseqQA(expressionMatrix)
```

Aparte de eliminar los posibles outliers presentes en las muestras, es necesario tratar

el efecto batch, una desviación intrínseca a los datos debido a diversos factores en el proceso de adquisición de las muestras. Si se conocen los batches o grupos de batches existentes en los datos, se aplica un algoritmos basado en el teorema de Bayes llamado ComBat. En este caso, como no se conoce si existe siquiera efecto batch entre las muestras, se debe de aplicar otro tipo de algoritmos que buscan posibles variables afectadas por estas desviaciones. Para ello existen dos grandes algoritmos, Surrogate Variable Analysis (SVA) y RUV (Remove Unwanted Variation). KnowSeq implementa ComBat y SVA como métodos de tratamiento de efecto batch mediante la función *batchEffectRemoval*. Para aplicar SVA en este caso, primero hay que crear un modelo que tendrá en cuenta que variables están afectadas para que posteriormente, la función que extrae los genes diferencialmente destacados tenga en cuenta dichas variables surrogadas.

Una vez se ha creado el modelo de SVA, es necesario imponer ciertas restricciones estadísticas para así filtrar los genes que no albergan diferencia alguna entre muestras sanas y tumorales. Para ello, se suele utilizar el Log-Fold Change (LFC) que mide la magnitud de la diferencia de expresión de cada gen en la muestra tumoral con respecto a la sana. Cuanto mayor sea dicha magnitud, más diferencia a nivel de expresión existe. Además si esa magnitud es negativa significa que el gen esta inhibido en la muestra tumoral con respecto a la normal. Por otro lado, si esa magnitud es positiva significa que el gen esta sobre-expresado en la muestra tumoral con respecto a la normal. Para llevar a cabo tanto el modelo del algoritmo SVA como la extracción de genes destacados mediante la función *limmaDEGsExtraction*, hay que ejecutar el siguiente código:

```
# Creamos el modelo SVA de variables surrogadas para tratar el efecto batch
svaMod <- batchEffectRemoval(expressionMatrix, as.factor(labels), method = "sva")

# Extraemos los genes diferencialmente destacados teniendo en cuenta las correccion mediante SVA
DEGsInformation <- limmaDEGsExtraction(expressionMatrix, as.factor(labels), lfc = 3.0, pvalue = 0.01, number = Inf, svaCorrection = TRUE, svaMod = svaMod)

%# Extraemos la tabla de estadisticas de los genes diferencialmente expresados, asi como la matriz ya filtrada con dichos genes.
topTable <- DEGsInformation$Table
DEGsMatrix <- DEGsInformation$DEGsMatrix
```

Una vez se tienen los genes diferencialmente expresados, es recomendable realizar una serie de comprobaciones gráficas para comprobar si realmente se observan cambios significativos entre ambos grupos (Tumor y Normal) mediante la funcion *dataPlot*

con distintos modos.

```
# Plotting the expression of the first 12 DEGs separatelly for all the samples
dataPlot(DEGsMatrix[1:12,], labels, mode = "genesBoxplot", toPNG = TRUE, toPDF = TRUE)

# Plotting the heatmap of the first 12 DEGs separatelly for all the samples
dataPlot(DEGsMatrix[1:12,], labels, mode = "heatmap", toPNG = TRUE, toPDF = TRUE)
```

7. Autoría

Para la reproducción del pipeline utilizado en este trabajo, es necesario la cita de este documento o de los artículos de investigación en los que se ha aplicado previamente:

- Castillo, D., Gálvez, J. M., Herrera, L. J., San Román, B., Rojas, F., & Rojas, I. (2017). Integration of RNA-Seq data with heterogeneous microarray data for breast cancer profiling. BMC bioinformatics, 18(1), 506: https://rdcu.be/bjTwh
- Gálvez, J. M., Castillo, D., Herrera, L. J., San Román, B., Valenzuela, O., Ortuño, F. M., & Rojas, I. (2018). Multiclass classification for skin cancer profiling based on the integration of heterogeneous gene expression series. PloS one, 13(5), e0196836.: https://doi.org/10.1371/journal.pone.0196836
- Castillo, D., Galvez, J. M., Herrera, L. J., Rojas, F., Valenzuela, O., Caba, O., ... & Rojas, I. (2019). Leukemia multiclass assessment and classification from Microarray and RNA-seq technologies integration at gene expression level. PloS one, 14(2). https://doi.org/10.1371/journal.pone.0212127
- Castillo-Secilla, D., Galvez, J.M, Ortuno, F.M., Herrera, L.J. & Rojas, I. KnowSeq R/bioc package: Beyond the traditional RNA-seq pipeline. A breast cancer case study., 08 November 2019, PREPRINT (Version 1) available at Research Square https://doi.org/10.21203/rs.2.16962/v1
- Gálvez, J. M., Castillo, D., Herrera, L. J., Valenzuela, O., Caba, O., Prados, J. C., ... & Rojas, I. (2019). Towards Improving Skin Cancer Diagnosis by Integrating Microarray and RNA-seq Datasets. IEEE Journal of Biomedical and Health Informatics. https://doi.org/10.1109/JBHI.2019.2953978

Referencias

[1] F. Hahne, W. Huber, R. Gentleman and S. Falcon Bioconductor Case Studies, 1st Edition Springer Publishing Company, Incorporated, 2008.

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

[3] J. W. Davis, Bioinformatics and computational biology solutions using r and bioconductor, Journal of the American Statistical Association 102 (477).