Pipeline RNASeq con *c2eYs for H. Sapiens*

*control de **c**alidad, tabla de **c**ontajes, **e**xpresión diferencial **Y s**ignificación biológica*

Manuel Ramón Gutiérrez Covarrubias
Máster en BioInformática y Bioestadística
Desarrollo de software para el análisis de datos ómicos

Consultor: **Ricardo Gonzalo Sanz**
Profesores responsables: **Alexandre Sánchez Pla, Antoni Pérez Navarro, Carlos Ventura Royo, José Antonio Morán Moreno y María Jesús Marco Galindo**

24 de mayo de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Pipeline RNASeq con c2eYs for H. Sapiens*
Nombre del autor:	*Manuel Ramón Gutiérrez Covarrubias*
Nombre del consultor/a:	*Ricardo Gonzalo Sanz*
Nombre del PRA:	*Alexandre Sánchez Pla, Antoni Pérez Navarro, Carlos Ventura Royo, José Antonio Morán Moreno y María Jesús Marco Galindo*
Fecha de entrega (mm/aaaa):	05/2017
Titulación::	*Máster Bioinformática y Bioestadística*
Área del Trabajo Final:	*Desarrollo de software para el análisis de datos ómicos*
Idioma del trabajo:	*Castellano*
Palabras clave	*Pipeline RNASeq Web*

Resumen del Trabajo

La finalidad de este trabajo fin de máster es el desarrollo de una herramienta software para el análisis de datos ómicos.

Esta propuesta se materializa en la creación de una interfaz accesible e intuitiva sobre un entorno web, que permite ejecutar un pipeline de análisis RNASeq, que pueda ser utilizado por un usuario sin tener grandes conocimientos de bioestadística y bioinformática.

Se ha partido de un pequeño catálogo previo de métodos susceptibles de implementación y uso, para cada una de las fases del pipeline (adquisición de datos, control de calidad, creación/selección de tabla de contajes, análisis de expresión diferencial y análisis de significación biológica). La metodología empleada ha consistido en analizar los métodos indicados en este catálogo, estudiándolos por separado y comparándolos mediante análisis empíricos (consumos de recursos hardware y tiempo, básicamente) y literatura científica.

La implementación final se ha desarrollado en base a los métodos seleccionados previamente, obteniéndose un producto funcional que permite la ejecución del análisis RNASeq.

Abstract

The aim in this master's degree job is the development of software tool for the analysis of omic data.

This proposal is materialized in the creation of an accessible and intuitive interface on a web environment, that allows to execute a pipeline of analysis RNASeq, and can be used by a user without great knowledge of biostatistics and bioinformatics.

First, a small preliminary catalogue of methods that can be implemented and used for each of the pipeline phases (data acquisition, quality control, creation / selection of counting tables, differential expression analysis and analysis of biological significance) has been the starting point. The methodology used has consisted of analyzing the methods of that catalogue, studying them separately and comparing them through empirical analysis (consumption of hardware and time resources, basically) and scientific literature.

The final implementation has been developed based on previously selected methods, obtaining a functional product that allows the execution of RNASeq analysis.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

Hoy en día, la utilización de las nuevas tecnologías en la investigación científica no es tema de debate, se asume que las *TIC* aportan un valor con gran relevancia, cualitativa y cuantitativa, desde la obtención de los datos en crudo mediante interfaces especializados (medidores de aire, por ejemplo), hasta la generación de decisiones (*machine learning*, por ejemplo) pasando por diferentes procesamientos de dichos datos.

En el ámbito de la Bioinformática y Bioestadística, al igual que en otras disciplinas científicas, se hace necesaria la automatización de procesos a través de herramientas especificas que permitan obtener resultados óptimos, permitiendo ahorrar recursos.

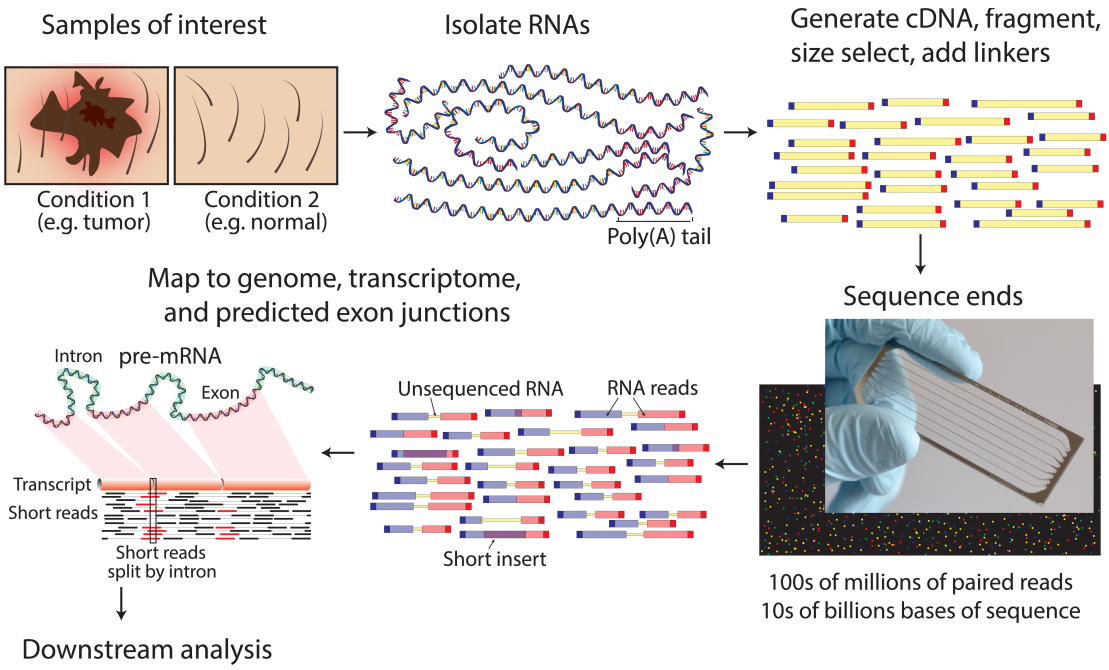


Fig.1.1. Análisis RNASeq (https://en.wikipedia.org/wiki/RNA-Seq)

RNASeq es una tecnología *NGS* (*Next Generation Sequencing*) que realiza una secuenciación directa de *ARN*[1]. Actualmente, los análisis *RNASeq*, si bien se realizan mediante sistemas informáticos potentes, las herramientas existentes son un compendio de procesos, a veces en entornos integrados y otras veces como una sucesión de procesos desarrollados en diferentes lenguajes de programación y/o plataformas; no existe una gran disponibilidad de herramientas que sirvan de guía a un usuario investigador.

La finalidad de este *TFM* (Trabajo Fin de Máster) es la creación una herramienta web que ayude al investigador en la realización de un análisis *RNASeq*, para diferentes muestras de origen ómico, siendo guiado por un pipeline en el que cada proceso está especializado en una tarea concreta utilizando unos métodos seleccionados en base a criterios computacionales, empíricos y/o basados en el estudio de diferentes publicaciones científicas.

Tanto mi perfil académico como el profesional están encuadrados entre el mundo de las *TIC* y el ámbito de la salud. La creación de herramientas para el análisis de datos ómicos está directamente relacionado con mis intereses, el tándem *TIC-salud*.

1.2. Objetivos del Trabajo

1. Objetivos generales.

    1.1. Crear el pipeline para análisis de datos *RNASeq*.

    1.2. Implementar pipeline en plataforma web mediante *Shiny*.

2. Objetivos específicos.

    2.1. Crear pipeline para el análisis de datos teniendo en cuenta el coste computacional de los métodos empleados.

        2.1.1. Evaluar el mejor método, entre *FastQC* y *NGS_QC Toolkit*, para realizar el control de calidad.

        2.1.2. Evaluar el mejor método para creación de tabla de contajes. Los métodos a evaluar son *Tophat*, *Cufflinks* y *HTSeq*.

        2.1.3. Evaluar el mejor método, entre *Cuffdiff2* y *DESeq*, para analizar la expresión diferencial.

        2.1.4. Evaluar el mejor método para el análisis de significación biológica.

        2.1.5. Evaluar el mejor método para el análisis de significación biológica.

    2.2. Implementar plataforma web mediante *Shiny*.

        2.2.1. Crear procesos de servidor que realicen las siguientes tareas:

            2.2.1.1. Control de calidad basado en el método seleccionado en el punto 2.1.1.

2.2.1.2. Creación de la tabla de contajes según el método elegido en el punto 2.1.2.

2.2.1.3 Análisis de la expresión diferencial a partir del mejor método del punto 2.1.3.

2.2.1.4. Análisis de significación biológica mediante el método obtenido en el punto 2.1.4.

2.2.2. Crear interfaz de usuario que sirva de guía del pipeline, incluyendo definición de formatos y estructuras de ficheros de entrada y salida, e implementación de ayudas que permitan orientar al usuario en el uso de la herramienta.

1.3. Enfoque y método seguido

Con el fin de obtener el software objetivo de este *TFM* con las máximas garantías posibles, intentando evitar deficiencias en el propio desarrollo que afecta al producto final, se establece como estrategia básica la realización de una evaluación previa (obtención de conocimiento) antes de comenzar la implementación del pipeline. Tal y como marcan los objetivos definidos, el desarrollo de este *TFM* se lleva a cabo en 2 fases: una de evaluación de métodos a utilizar, que permite seleccionar los óptimos entre los evaluados, y otra de implementación de dichos métodos, en un plataforma web sobre *Shiny*.

La evaluación (Fase 1) de los métodos sigue el siguiente guión.

- De los métodos a comparar, intentar localizar recursos en fuentes fidedignas, además de documentación comparativa en Internet, que ayude en la selección del método a implementar en la correspondiente tarea del pipeline.

- Analizar los algoritmos empleados en los métodos comparados. Intentar valorar la complejidad computacional de cada uno de los métodos, haciendo hincapié en el consumo de recursos y dificultad de implementación. Este proceso arroja la significancia suficiente para realizar la selección del método apropiado.

La implementación / desarrollo (Fase 2) sigue el siguiente guión.

- Ajuste de métodos. Detectar funciones y módulos necesarios para una correcta implementación del algoritmo del método seleccionado. Esta detección consiste en la enumeración de los subprocesos ya identificados en el análisis de los métodos, indicando cuáles se implementarán en funciones y cuáles en módulos, con su justificación.

- Análisis del pipeline. Con el fin de detectar y corregir posibles fallos iniciales, se revisa el pipeline completo y se prepara la implementación.

- Desarrollar, en lenguaje *R*, cada una de las funciones y módulos detectados. Se revisa el código para evitar problemas de recursividad, entre otros.

- Documentar cada una de las funciones y módulos implementados así como las decisiones tomadas.

- Realizar pruebas de ejecución con datos de ejemplo, sobre cada una de las funciones y módulos implementados, para detección de errores.

- Corregir errores detectados. Seguidamente a la corrección, se vuelven a realizar las pruebas hasta conseguir un código sin errores.

El interfaz de usuario se desarrolla siguiente criterios de usabilidad y accesibilidad, que permiten una cómoda experiencia para el usuario. Todas las opciones se identifican correctamente evitando dudas en el uso y construyendo un entorno intuitivo.

Todas las implementaciones se realizan en la Fase 2, exceptuando aquellas que se hayan de realizar en la Fase 1 por necesitarse en los procesos de evaluación de los métodos. Se adjunta justificación de estas decisiones. Para la aplicación web se utiliza *Dashboard* de *Shiny*.

1.4. Planificación del Trabajo

Los siguiente puntos se corresponden con la planificación de hitos y temporización, incluyendo la elaboración del propio Plan de Trabajo y creación de la memoria y presentación del *TFM*.

1.4.1. Tareas

Se identifican con letras y se subdividen numéricamente.

a. Plan de Trabajo (15 días).

a.1. Análisis de requerimientos (5 días). Se realiza una evaluación de las necesidades documentales y software, para llevar a cabo el *TFM*.

a.2. Gestión del tiempo (2 días). Se desglosan las tareas y se las asigna una temporización.

a.3. Borrador del Plan de Trabajo (7 días).

a.4. Entrega Plan de Trabajo (1 día).

b. Preparación - Fase 1 (21 días).

b.1. Control de calidad (5 días). Estudio y selección de métodos.

b.1.1. *FastQC* (2 días).

b.1.2. *NGS_QC Toolkit* (3 días).

b.2. Crear tabla de contajes (4 días). Estudio y selección de métodos.

b.2.1. *Tophat* (2 días).

b.2.2. *Cufflinks* (1 día).

b.2.3. *HTSeq* (1 día).

b.3. Análisis de expresión diferencial (5 días). Estudio y selección de métodos.

b.3.1. *Cuffdiff2* (2 días).

b.3.2. *DESeq* (3 días).

b.4. Análisis de significación biológica (6 días). Estudio y selección de metodología a implementar.

b.5. Entrega Fase 1 (1 día).

c. Desarrollo - Fase 2 (35 días).

c.1. Ajuste de métodos (10 días).

c.2. Análisis del pipeline (5 días).

c.3. Implementación en *Shiny* (19 días).

c.3.1. Control de calidad (5 días).

c.3.2. Tabla de contajes (4 días).

c.3.3. Análisis de expresión diferencial (4 días).

c.3.4. Análisis de significación biológica (3 días).

c.3.5. Interfaz de usuario (3 días).

c.4. Revisión continua del desarrollo (19 días). Tarea auditora / correctiva sobre el desarrollo que optimiza procesos y mitiga riesgos. Se realiza durante todo el desarrollo y su tiempo no es acumulable al resto de las tareas.

c.5. Entrega Fase 2 (1 día).

d. Memoria y presentación *TFM* (+ 10 días).

d.1. Memoria. Se trata de una tarea realizada en el tiempo, desde la primera tarea del TFM del Plan de Trabajo hasta la entrega de la Fase de Desarrollo (2).

d.2. Presentación (9 días).

c.3. Entrega TFM (1 día).

1.4.2. Hitos

1.4.2.1. Entrega del Plan de Trabajo.

1.4.2.2. Entrega de la Fase 1, consistente en la parte de la memoria realizada hasta el momento, que incluye únicamente la obtención de conocimiento mediante la evaluación de los distintos métodos que comprenden cada una de las tareas de dicha fase.

1.4.2.3. Entrega de la Fase 2, compuesta por la aplicación desarrollada en plataforma web sobre *Shiny* (*Shiny Dashboard*).

1.4.2.4. Entrega de memoria y presentación del *TFM*.

1.4.3. Análisis de riesgos

1.4.5.1. Temporización. Una mala temporización puede impedir el alcance de los objetivos marcados. Si se puede, cuando se invierta más tiempo del planificado en un tarea, se aprovecha el tiempo sobrante de otras tareas. Se pretende aprovechar al máximo el tiempo disponible, utilizando el sobrante como factor correctivo.

1.4.5.2. Evaluación errónea de métodos. Puede conllevar la no comprensión correcta de los mismos con el consiguiente problema en su implementación, pudiendo requerir de la utilización de más tiempo del definido, retrasando otras tareas.

1.4.5.3. Problemas computacionales. En el momento de la implementación, y tras un análisis incompleto o una selección de *datasets* muy grandes, pueden aparecen problemas de computación que ralentizan la ejecución de los distintos procesos e incluso generan timeouts que imposibilitan su ejecución.

1.4.5.4. Errores en la implementación. Pueden bloquear la consecución parcial o total del *TFM*. Es necesario una comprensión previa así como experiencia en el uso de las herramientas de desarrollo a emplear, antes de comenzar la implementación.

1.4.5.5. Estrategia errónea de implementación. Una distribución errónea de procesos para la resolución de cada una de las tareas, puede resultar en un código que no funcione.

1.4.5.6. Falta de actualización periódica de la Memoria. Este documento principal debe estar al día, aprovechando el calendario establecido, sin utilizar tiempo asignado a otras tareas, permitiendo realizar la entrega en plazo.

1.4.5.7. Retrasos imprevistos. Un imprevisto puede generar retrasos y cuellos de botella. Se realiza un ajuste continuado a la temporización planificada, intentando realizar las tareas en el menor tiempo posible, ahorrando tiempo para poder asignarlo a la resolución de actividades imprevistas.

1.4.4. Calendario - Diagrama *Gantt*

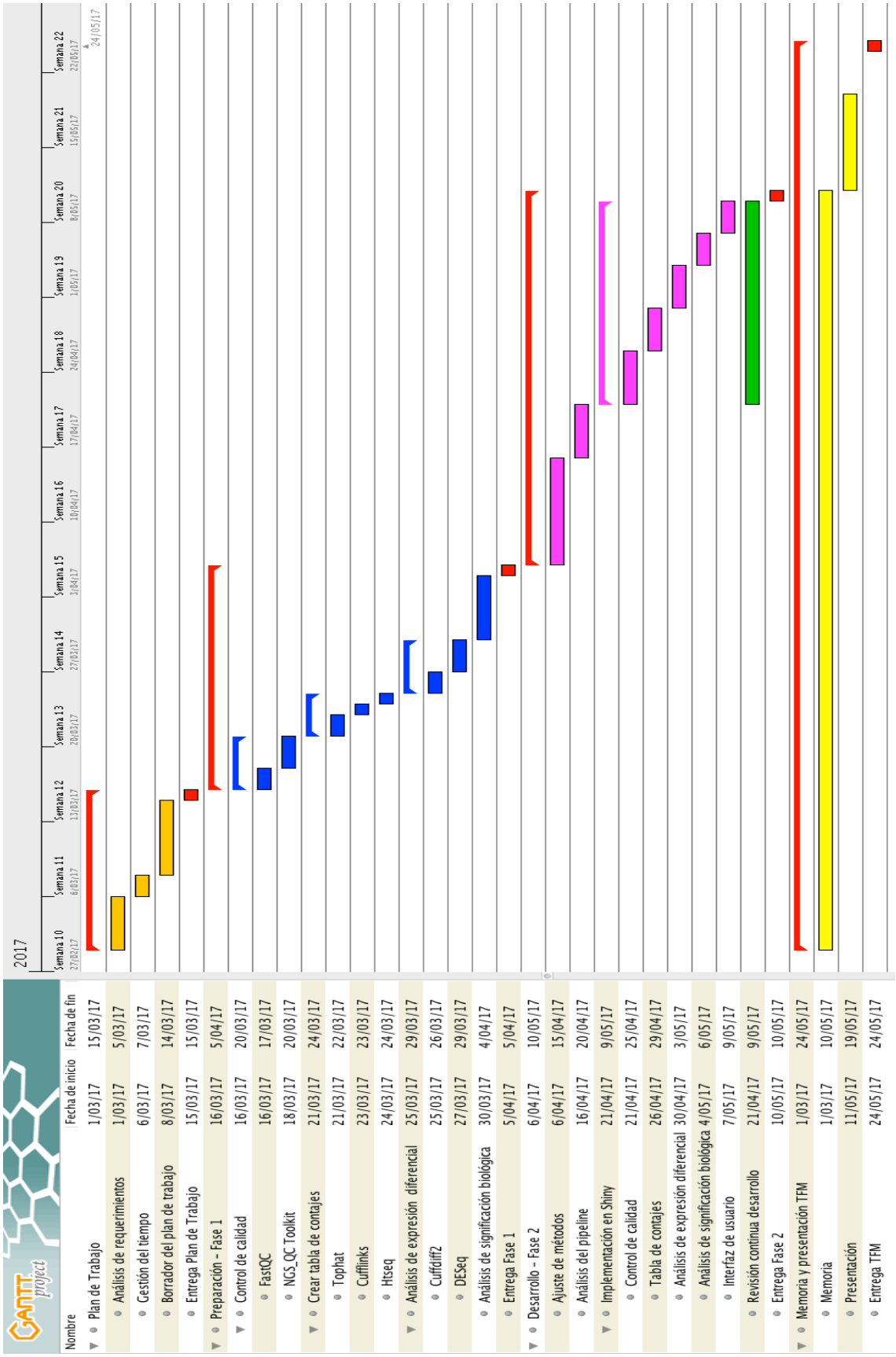


Fig.1.2. Plan de trabajo

1.5. Breve sumario de productos obtenidos

1. Plan de Trabajo del *TFM*.

2. Memoria. Documento principal generado, actualizado y corregido durante todas las etapas del *TFM*.

3. Producto. Software en plataforma web *Shiny*, que permite guiar a un usuario, normalmente investigador, en el análisis de datos de *RNASeq*. Se incluyen URL para su ejecución así como los ficheros fuente de este software.

4. Presentación virtual. Ficheros en formato *PDF* y *HTML* (visualizable con cualquier navegador estándar) de la presentación de todas las fases del *TFM* del que es base el Plan de Trabajo. Se adjunta un video con voz en off de la presentación del *TFM*.

5. Autoevaluación del proyecto. Este documento mostrará una crítica del *TFM* una vez se ha concluido todas las tareas. Se hará una revisión del mismo indicando las posibles mejoras (*TO-DO*), así como aquellos "defectos". Esta documentación aparece incluida en el propio TFM en el capítulo 4, de Conclusiones.

1.6. Otros capítulos de la memoria

1.6.1. Fase 1. Preparación

Fase de evaluación teórica de métodos empleados en cada uno de los pasos del pipeline a implementar: control de calidad, creación de tablas de contajes, análisis de expresión diferencial y de significación biológica. Se analizan los métodos indicados en el Plan de Trabajo, realizando tests de rendimiento, comprobado recursos utilizados, tamaños de muestras, tiempos, …, que permitan seleccionar los mejores métodos para su posterior implementación.

1.6.2. Fase 2. Desarrollo

Diseño e implementación de todos los métodos seleccionados en la fase de preparación, para poder ser utilizados en una plataforma web a partir de *Dashboard* de *Shiny*. Se intenta optimizar y minimizar el código utilizado, siendo éste ampliamente documentado.

2. Fase 1 - Preparación

La evaluación de los métodos a emplear en las diferentes fases de este *TFM* se ha realizado siguiendo el siguiente guión:

- De los métodos a comparar, se han intentado localizar recursos desde fuentes fidedignas. Además, se ha buscado documentación comparativa en Internet que la ayudado en la selección de los métodos a implementar.

- Se analizan los algoritmos empleados en los métodos comparados. Se valoran los requerimientos computacionales de cada uno de los métodos además de su complejidad a la hora de realizar su implementación. En base a esos criterios, se obtiene la significancia suficiente para realizar la selección de los métodos, dependiendo de la tarea.

2.1. Conjunto de datos

Como conjunto de datos, se ha seleccionado el titulado *E-MTAB-1147 - Transcription profiling by high throughput sequencing of HNRNPC knockdown and control Hella cells*, accesible desde la URL:

`http://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-1147/`

La motivación de la selección de este conjunto de datos ha venido dada por las referencias realizadas durante el Máster. Se trata de un conjunto de datos con pocas muestras, pero con gran cantidad de información.

La preparación de este conjunto de datos se realizó de acuerdo a la *Guía de Preparación de Muestras de Secuenciación de ARNm (Illumina, Parte # 1004898 REV.D)* que consta de muestras *knockdown* y *control*, secuenciadas juntas en una *flowcell* en una y dos vías, respectivamente. Se utilizan los datos disponibles en la última actualización, de fecha 22 de agosto de 2016.

Muestra	Réplica	Ficheros	Genotipo	Variable
Control	1	4	control	control
Control	2	4	control	control
knockdown 1	1	2	HNRNPC knockdown with siRNA1	KD1
knockdown 1	2	2	HNRNPC knockdown with siRNA1	KD1
knockdown 2	1	2	HNRNPC knockdown with siRNA2	KD2
knockdown 2	2	2	HNRNPC knockdown with siRNA2	KD2

Tab.2.1. Descripción conjunto de datos (Elaboración propia)

Este conjunto de datos consta de 6 muestras, *control* y *knockdown*, correspondientes al organismo *Homo Sapiens* para individuos de sexo femenino.

Las muestras de *control* son 2, divididas en 8 ficheros, mientras que las muestras de *knockdown*, se agrupan en 4 réplicas, 2 por tipo de *knockdown* (KD1 y KD2).

2.2. Control de calidad. *FastQC vs NGS QC Toolkit*

2.2.1. *FastQC*

Permite realizar controles de calidad sobre datos de secuencias sin procesar, mediante un conjunto modular de análisis que alerta de problemas en dichos datos, a tener en cuenta para posteriores análisis de los mismos.

Desde la web de *Babraham BioInformatics*, se puede obtener una aplicación en lenguaje Java, de código abierto, que implementa el método *FastQC***[2]** y que tiene las siguientes características destacables:

- Importación de datos de archivos en formatos *BAM***[3]**, *SAM***[4]** ó *FastQ***[5]**.
- Generación de resúmenes rápidos para indicar áreas problemáticas.
- Gráficos resumen y tablas para la evaluación ágil de datos.
- Exportación de resultados a informe estático en formato *HTML***[6]**.
- Automatización de generación de informes sin uso interactivo.

Un ejemplo de uso de esta aplicación Java se puede encontrar en *QuadCRS***[7]**, herramienta de código abierto, desarrollada en lenguaje *Python*, que ayuda al control de calidad de datos *RNASeq*, e implementa el uso de tres métodos: *FastQC*, *RNA-SeQC***[8][9]** y funciones de *RSeQC***[10][11]**. La ejecución de *FastQC* es dependiente de software de *Babraham BioInformatics*.

FastQC lee los ficheros de secuencia, incluidos en la línea de comandos, y emite un informe de calidad de cada uno de ellos. Si no se introducen nombres de ficheros de secuencia, se abre una aplicación gráfica.

Para realizar el análisis de la herramienta se ha utilizado la versión 0.11.5 de fecha 8 de marzo de 2016, obtenida desde la siguiente *URL*:

```
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
fastqc_v0.11.5.dmg
```

Un pseudocódigo básico de esta aplicación podría ser el siguiente. Entre paréntesis se indica la clase, en lenguaje Java, que realiza la operación.

```
programa FastQC;
seleccionar los ficheros en formato .fastq (FastQCApplication);
para cada fichero de secuencias (SequenceFile)
  para cada secuencia del fichero de secuencias
    ejecutar análisis (AnalysisRunner).
      hacer estadísticas básicas (BasicStats)
        mostrar
          nombre de fichero,
          tipo de fichero,
          codificación,
          total de secuencias,
          secuencias marcadas de calidad pobre,
          longitud de secuencias,
          % GC;
```

```
        fin mostrar;
      fin hacer estadísticas básicas;
      hacer puntuaciones por base (PerBaseQualityScores)
        mostrar
          media,
          mediana,
          cuartil más bajo,
          cuartil más alto,
          Percentil 10%,
          Percentil 90%
        fin mostrar;
      fin hacer puntuaciones por base;
      hacer puntuaciones por posición (PerTileQualityScores)
        mostrar
          Media por base;
        fin mostrar;
      fin hacer puntuaciones por posición;
      hacer distribución de la media (PerSequenceQualityScores);
      hacer porcentaje de base por secuencia (PerBaseSequenceContent);
      hacer distribución de GC (PerSequenceGCContent);
      hacer análisis de "Adapters" (AdapterContent);
      hacer análisis Kmer (KmerContent);
    fin ejecutar análisis;
  fin para cada secuencia;
fin para cada fichero;
fin programa;
```

2.2.2. *NGS_QC Toolkit*

Es un conjunto de herramientas**[12]**, desarrollado en lenguaje *Perl* por el *National Institute Of Plant Genome Research*. Una descripción de esta herramienta se puede encontrar en la publicación científica titulada *NGS QC Toolkit: A toolkit for Quality Control of Next Generation Sequencing Data de Ravi K. Patel y Mukesh Jain***[13]**.

`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3270013/`

El análisis de esta herramienta se ha realizado con la versión 2.3.3, de fecha 3 de febrero de 2014, descargada desde la siguiente *URL*:

```
http://59.163.192.90:8080/ngsqctoolkit/cgi-bin/download.pl?
toolkit=NGSQCToolkit_v2.3.3.zip
```

Esta aplicación puede realizar el control de calidad con dos herramientas básicas especializadas en aprovechar los ordenadores con varias *CPUs*. Realiza el tratamiento de ficheros en formatos *FastQ* ó *FASTA*[**14**].

En este caso, el análisis se ha realizado con las herramientas que utilizan el formato *FastQ*: `IlluQC.pl` e `IlluQC_PRLL.pl`, siendo esta última la adaptada para el aprovechamiento de varias CPUs.

Un pseudocódigo de esta aplicación podría ser:

```
programa IlluQC;
  para cada fichero de secuencias;
    analizar estructura del fichero completo (checkFastQFormat);
    leer fichero (processSingleEndFiles);
    crear gráficos (drawGCDist y drawBaseComp);
    crear estadísticas (printStat, printStatTab);
    grabar salida HTML (htmlPrint)
  fin para cada fichero;
fin program;
```

Para mejorar el rendimiento del análisis, se ha evitado el uso de ficheros comprimidos, realizando una descompresión previa a la utilización de los métodos, es decir, los ficheros originales estaban comprimidos con *Gzip***[15]** (.fastq.gz) y al descomprimirse han quedado en formato *FastQ* (.fastq).

La siguiente tabla muestra una comparativa entre los métodos anteriores, para el análisis de 6 GB y 29.741.852 secuencias cada uno.

2 Ficheros de 6 GB	FastQC	NGS QC Toolkit (1 CPU)	NGS QC Toolkit (2 CPUs)
Tiempo de CPU	15:05 minutos	> 15 minutos *	> 15 minutos *
Memoria - real - virtual	 161.6 MB 3.90 GB	 > 2.3 MB 2.34 GB	 > 135 MB > 2.52 GB
Estrategia por fichero	No se realiza chequeo completo del fichero. Por cada secuencia leída, se realizan los cálculos estadísticos	Se chequea por completo el fichero a comparar y se vuelve a leer secuencia por secuencia para realizar los cálculos estadísticos	
Visualización / Gráficos	Alertas con iconos y facilidad de comparativa entre muestras utilizando el sistema de pestañas	Comparativo en el mismo documento resumen HTML	

**Proceso cancelado con 1 fichero, al 5.37% del proceso (1.600.000 secuencias)*
(Fuente: estadísticas del sistema operativo)

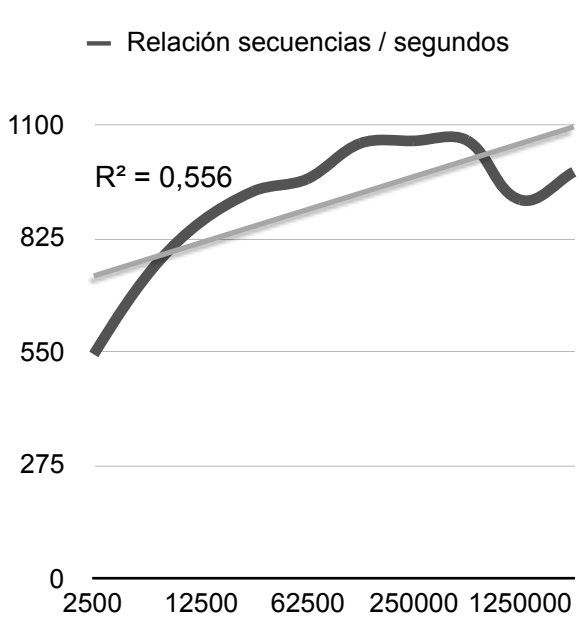
Tab.2.2. Pruebas de rendimiento - FastQC (Elaboración propia)

La elección del formato *FastQ* ha estado motivada por la estandarización del mismo en los secuenciadores. Este será el formato utilizado como ficheros de input en el pipeline a desarrollar.

Se observa que *FastQC*, cuando se utiliza a través del interfaz de usuario, consume menos tiempo de proceso (7:45 minutos) aunque mayor espacio en memoria (25.6 MB + 5.9 GB).

La tabla y gráfica de la siguiente página, muestran el consumo de recursos computacionales mediante *NGS QC Toolkit*, medido con diferentes tamaños de muestras. También se indica el estadístico R^2 y una línea de tendencia.

Nº Secuencias Fichero	Tiempo CPU
2500	4s 600ms
6250	8s 300ms
12500	14s 600ms
25000	26s 600ms
62500	1min 4s 500ms
125000	1min 58s 400ms
250000	3min 55s 500ms
625000	9min 48s 200ms
1250000	22min 41s
2500000	42min 14s 500s



Tab.2.2.3. Rendimiento Bowtie (Elaboración propia)

2.2.3. Decisión

Con ficheros de secuencias de menores dimensiones, los tiempos son menores en ambos métodos, pero a medida que el tamaño es mayor, el método *NGS QC Toolkit* se vuelve lento aunque el consumo de memoria es menor que *FastQC*.

Con respecto a las características propias de los resultados obtenidos con ambas aplicaciones, se concluye que ambas disponen de diferentes elementos que permiten realizar correctamente el control de calidad.

La posibilidad de invocar ambas aplicaciones desde lenguaje *R* es viable, pero las estadísticas de rendimiento computacional indican que la eficiencia de *FastQC* es mayor que la de *NGS QC Toolkit*. Este es el motivo principal por el que se ha seleccionado *FastQC* como herramienta para realizar el control de calidad en el pipeline.

2.3. Tabla de contajes. *Tophat* vs *Cufflinks* vs *HTSeq*

Se ha realizado un análisis individual de cada una de las tres herramientas y se ha comprobado que la creación de la tabla de contajes**[16][17][18]** se puede realizar de manera individual o con la unión de algunas de ellas**[19]**.

2.3.1. *Tophat - A splices read mapper for RNA-seq*

Se constata que esta herramienta es necesaria para poder crear la tabla de contajes, tanto si se utiliza *Cufflinks* o *HTSeq*.

En este caso, se ha elegido el formato *FASTA* para utilizar con *Tophat***[20]** (versión 2.1.1). La conversión desde el fichero en formato *FastQ* se ha conseguido con la herramienta *Seqtk***[21]** (versión 1.2-r101-dirty). La conversión entre formatos se ha realizado mediante una orden similar a la siguiente:

```
seqtk seq -a muestra.fastq > muestra.fasta
```

Se debe indicar que el tiempo empleado en la conversión de un fichero de unos 500 MB, ha sido de 28 segundos.

A continuación se necesitan preparar los datos, generando un fichero de índices que puede ser creado con la herramienta *bowtie-build* (versión 0.12.9) y un comando similar al siguiente:

```
bowtie-build -f muestra.fasta muestra.bowtie
```

Este proceso, para un fichero de unos 300 MB, en un ordenador de 4 *CPUs*[1], se extendió durante 5 minutos y 35 segundos. El resultado ha sido una serie de ficheros con extensión .ebwt; los más interesantes son los que tienen extensión `.rev.1.ebwt` y `.rev.2.ebwt`.

En este punto, ya se disponen de todos los datos necesarios para ejecutar *Tophat*, con una orden similar a la siguiente:

```
tophat -o muestra —no-novel-juncs muestra.bowtie muestra.fastq
mv muestra/accepted_hists.bam muestra.bam
```

La ejecución de esta orden, en un ordenados con 4 *CPUs*, duró más de 19 minutos, creándose un directorio llamado muestra, con diferentes ficheros, siendo el más importante el que está en formato *BAM*, llamado `accepted_hists.bam`, y que será utilizado tanto por Cufflinks como por *HTSeq*.

[1] Intel QuadcoreCore i5-6500 3.20 Ghz, 8 GB RAM y 200 GB SATA

En caso de necesitar ficheros en formato *SAM*, se puede realizar la conversión desde el formato *BAM*, con el conjunto de herramientas *SAMtools***[22]**, ejecutando una orden similar a:

```
samtools view -h -o muestra.sam muestra.bam
```

Para obtener los ficheros en formatos *GFF/GTF***[23]**, con las anotaciones de los genes, para el organismo Homo sapiens, versión *GRCh38*, se ha accedido a la siguiente *URL*, en *Ensembl*:

```
http://www.ensembl.org/info/data/ftp/index.html
```

2.3.2. *Cufflinks - Transcriptome assembly and differential expression analysis for RNA-Seq*

A continuación se ha ejecutado la herramienta *Cufflinks***[24]** (versión 2.2.1) mediante la siguiente orden:

```
cufflinks –library-type fr-secondstrand \
      -o cufflinks.log \
      -G Homo_sapiens.GRCh38.88.gff3 \
      muestra.bam
```

En la salida estándar se obtiene algo similar a:

```
Loading reference annotation.
Inspecting reads and determining fragment length distribution.
> Processed 34391 loci.
[*************************] 100%
> Map Properties:
>    Normalized Map Mass: 2207817.00
>    Raw Map Mass: 2207817.00
>    Fragment Length Distribution: Truncated Gaussian (default)
>                  Default Mean: 200
>              Default Std Dev: 80
Estimating transcript abundances.
> Processed 34391 loci.
```

Como resultados, se obtienen 4 ficheros en el directorio *cufflinks.log*. Su tiempo de ejecución sobre un fichero de 260 MB, en un ordenador de 4 *CPUs*, ha sido de 8 minutos, utilizando un máximo de 1.6 GB de RAM. Los 4 ficheros obtenidos son:

```
-rw-rw-r-- 1 user user   6046246 abr  2 20:59 genes.fpkm_tracking
-rw-rw-r-- 1 user user  27167507 abr  2 20:59 isoforms.fpkm_tracking
-rw-rw-r-- 1 user user         0 abr  2 20:52 skipped.gtf
-rw-rw-r-- 1 user user 337000093 abr  2 20:59 transcripts.gtf
```

2.3.3. *HTSeq - Analysing high-throughput sequencing data with Python*[26]

Se ha utilizado la versión 0.6.1 de *HTSeq*[25] y el contaje se ha realizado con el comando:

```
htseq-count -q --mode=union --stranded=no --type=exon \
         -f bam -o counts.sam \
         muestra_2.bam \
         Homo_sapiens.GRCh38.88.gtf > counts.log
```

La ejecución del comando se ha realizado en una máquina con 4 *CPUs*, durante prácticamente 3 minutos, consumiendo 360 MB de memoria *RAM*, durante ese tiempo. La longitud del fichero *BAM* utilizada es de 260 MB.

Como resultado se obtiene una salida en formato *SAM* y un log del proceso. En ese log, las últimas líneas muestras el resumen, por ejemplo:

```
__no_feature  1571913
__ambiguous   0
__too_low_aQual    0
__not_aligned 0
__alignment_not_unique  3108913
```

2.3.4. Decisión

Los resultados obtenidos mediante *Cufflinks*, se han conseguido consumiendo más memoria *RAM* y tiempo que con *HTSeq*, pero *Cufflinks* devuelve información más completa (genes, isoformas, obviados y tránscritos); *HTSeq* sólo devuelve contajes totales mientras que *Cufflinks* devuelve *RPKM*[27].

En un principio, se decide implementar *Tophat* y *Cufflinks* en el pipeline por la intención de utilizar una librería R llamada *cummeRbund* en el desarrollo final. La inclusión de esta librería en el desarrollo ha generado graves problemas de compatibilidad entre ella y las otras librerías y no se ha conseguido implementarla. Dado que la única necesidad era para la creación de la tabla de contajes, se buscan otras vías y se localiza la librería *RSubread* que incluye una función llamada featureCounts que permite la creación de la tabla de contajes desde los ficheros accepted_hits.bam generados por *Tophat*. Por todo ello, finalmente se decide implementar únicamente *Tophat*.

2.4. Expresión diferencial. *Cuffdiff2* vs *DESeq*

2.4.1. *Cuffdiff2*

Se ha evaluado la versión 2.2.1. Esta herramienta**[23]** forma parte del paquete Cufflinks. Una sentencia de ejecución de ejemplo, podría ser:

```
cuffdiff -p 2 -o cuffdiff.log \
         Homo_sapiens.GRCh38.88.gff3 \
         muestra1.bam,muestra2.bam
```

Con la anterior orden, se realiza el análisis para las muestras 1 y 2, y la salida se envía al directorio cuffdiff.log, donde se crean varios ficheros de resultados (*FPKM tracking, count tracking, rad group tracking, differential expression tests, differential splicing tests, differential coding output, differential promoter use* y *read group info*). El proceso, en un ordenador de 4 *CPUs*, invierte 13 minutos de tiempo de *CPU* y utiliza hasta cerca de 1.9 GB de memoria *RAM*. Las muestras tienen un tamaño en disco de alrededor de 260 MB.

2.4.2. *DESeq*

Se ha localizado una librería, en lenguaje *R*, que permite realizar análisis de expresión diferencial con *DESeq***[28]**. La librería se llama igual que el método, *DESeq*, y pertenece al conjunto de paquetes *BioConductor***[29]**. Este puede ser instalado, en R, con las siguientes instrucciones:

```
source("http://bioconductor.org/biocLite.R")
biocLite("DESeq")
```

No se ha realizado test alguno con este método debido a la complejidad inicial en la preparación de datos. Como apoyo a la decisión se ha tenido en cuenta el artículo científico de *Cole Trapnell et al*, titulado *Differential analysis of gene regulation at transcript resolution with RNA-seq***[30]**, donde se indica que *Cuffdiff* obtiene muy buenos resultados, pero es *DESeq* (también *edgeR*) el método que más genes diferencialmente expresados localiza, pero en algunos casos (p.e. transcritos y genes) muestra mayor número de falsos positivos que *Cuffdiff*.

2.4.3. Decisión

Cuffdiff tiene el gran inconveniente de la utilización de muchos recursos computacionales, pero tiene buenos resultados y muestra mucha información, aunque no es recomendable para análisis de expresión diferencial de genes, si la profundidad de secuenciación es baja**[31]**. Por otro lado, *DESeq* obtiene un número alto de falsos positivos cuando el número de réplicas es baja aunque para un número alto de réplicas podría considerarse la mejor**[32]**.

A partir de esta información y de la incluida en las publicaciones analizadas, se decide realizar la implementación del método *edgeR***[33][34]** (*Empirical Analysis of Digital Gene Expression Data in R*), que no era uno de los métodos previstos para implementar en el pipeline, pero que ha resultado ser el más tolerante a pesar de la inclusión de algunos falsos positivos (menos que *DESeq*), para realizar análisis de expresiones diferenciales de genes.

2.5. Significación biológica[35]. *goana* vs *topGO*

2.5.1. *goana*

Chequea la sobre-representación**[36]** de términos *GO* (*Gene Ontology*) ó vías (pathways) *KEGG* de los genes expresados diferencialmente de un ajuste de modelo lineal.

Realiza análisis de enriquecimiento, en el que se debe suministrar un *Entrez Gene Id***[37]** por cada gen. Los cálculos se pueden realizar con una prueba similar a un test exacto de *Fisher* (prueba hipergeométrica unilateral), o a partir del ajuste del método con un test ajustado que utiliza una distribución hipergeométrica no central de *Wallenius***[38][39]**.

El resultado es una tabla con una fila por cada término *GO* y las siguientes columnas: término *GO*, ontología (*BP*, *CC* y *MF*), número de genes en el término GO, número de genes diferencialmente expresados por arriba, número de genes diferencialmente expresados por abajo, *p-value* de la sobre-representación por arriba, *p-value* de la sobre-representación por abajo. La anotación utilizada es la de *NCBI RefSeq*.

2.5.2. *topGO*

Paquete de *Bioconductor* que está diseñado para realizar análisis semiautomáticos de enriquecimiento**[40]** para términos de *GO*. Instalable con:

```
source("http://bioconductor.org/biocLite.R")
biocLite("topGO")
```

Partiendo de datos con medidas normalizadas de expresión génica, correlaciones genéticas o análisis de expresión diferencial, interpreta y visualiza los resultados.

Una ventaja de este método, aunque no utilizable en este *TFM*, es la posibilidad de implementación, por parte del usuario, de nuevas pruebas estadísticas o nuevos algoritmos.

También implementa la posibilidad de visualización de gráficos con los resultados, creando un gráfico de estructura.

2.5.3. Decisión

Tomando como base la relación directa entre el método *edgeR* y *goana*, aconsejada por *Bioconductor*, se decide implementar la significación biológica con el método *goana* debido a su sencillez, velocidad y eficacia en cuanto a los resultados esperados en esta fase.

Finalmente, dada la cantidad de información a descargar, se decide realizar la implementación únicamente para el organismo *Homo Sapiens*. Este cambio se refleja en el nombre de la aplicación que pasa a llamarse *c2eYs for H. Sapiens*.

2.6. Métodos de análisis del coste computacional

Se realiza un análisis empírico de los métodos, sobre una misma máquina los mismos métodos, mediante comparación de sus costes espacial[2] y temporal[3]; se considera un método mejor aquel que obtiene menores costes.

Para comprobar el comportamiento de cada método, por cada uno de ellos, se realizan tres análisis con muestras pequeñas (<12.500 secuencias), medianas (entre 12.500 y 125.000 secuencias) y grandes (> 125.000 secuencias).

[2] cantidad de memoria necesaria para ejecutar el algoritmo

[3] tiempo total en la ejecución del algoritmo

2.7. Riesgos aparecidos

La complejidad en el estudio de los métodos *FastQC* y *NGS_QC Toolkit*, los tests con *Tophat* y *Cufflinks*, imprevistos externos de carácter personal y problemas en descargas de ficheros de gran volumen, han sido eventualidades que han requerido una mayor inversión de tiempo que el previsto, generando retrasos tanto en los análisis iniciados como en las sucesivas tareas.

La implementación del proyecto, en algunos momentos, se ha complicado por el alcance previsto del mismo, dado que se pretenden realizar todas las fases de un pipeline RNASeq desde la propia aplicación, teniendo en cuenta que los procesos no son inmediatos y que algunos de ellos pueden invertir horas en su finalización.

En una fase inicial, se instalaron *RStudio Server***[41]** y *Shiny Server***[42]** en un servidor *Linux* (*Ubuntu* 17.04, 64 bits**[43]**) con 4 CPUs y 8 GB de RAM. El requerimiento de instalar el software para la realización del Control de Calidad y la generación de la Tabla de Contajes, era una limitación para utilizar *Github***[44]** y *shinyapps.io***[45]**, webs especializadas en la ayuda del desarrollo del software.

Seguidamente, se desarrolló el código necesario para realizar la carga masiva de datos a analizar, mediante la propia aplicación, en el servidor Linux. Inicialmente, se configuró *Shiny* para permitir el envío de ficheros de hasta 7 GB; luego se desactivo. Esta carga generó consumos altos en los recursos del ordenador, sobre todo a nivel de entrada / salida, es decir, de disco duro.

A continuación se descargó, instaló y configuró el software necesario para implementar el código que realiza el Control de Calidad. En este punto, la dificultad encontrada fue cómo ejecutar un programa *Java*, *FastQC*, desde *Shiny* para luego captar los resultados generados. La solución se tardó en optimizar, generando otra vez más retrasos en el desarrollo de este TFM.

Los problemas serios aparecieron cuando se comenzó a trabajar con la siguiente fase, la creación de la Tabla de Contajes, con el software *TopHat* y *Cufflinks*, básicamente; su uso no es trivial. Los ficheros que se generaron y los tiempos y recursos utilizados en su generación, penalizaron el avance del proyecto. Se había decidido utilizar *Cufflinks* para obtener, mediante la librería *cummeRbund*, la tabla de contajes, pero es incompatible con el resto de librerías utilizadas en la aplicación y después de dar muchas vueltas se ha decidido descartar la implementación de *cummeRbund* y de *Cufflinks*. En su lugar, a partir de la salida de *Tophat* se obtiene la tabla de contajes con la librería *Rsubread* y su función featureCounts apoyándose en otras librerías de anotaciones compatibles.

La dedicación a solventar los problemas derivados de esas situaciones han sido un quebradero de cabeza, y como consecuencia, se han debido realizar múltiples modificaciones en el software. Una de las modificaciones realizadas ha sido la ejecución en *background* de aquellos procesos más costosos en

tiempo y/o recursos, configurando dichos procesos en modo secuencial o paralelo, con utilización de una o más CPUs, en función de si tiene un consumo menor o mayor de recursos, sobre todo de entrada / salida. El estado de los procesos más importantes se puede monitorizar desde el *Dashboard* de la aplicación *c2eYs*.

El resto de los métodos para los análisis de expresión diferencial y significación biológica, no han necesitado una inversión extra en el tiempo.

2.8. Software base en el desarrollo

2.8.1. Lenguaje *R*

El lenguaje R**[46]** es un proyecto *GNU*[4], similar al lenguaje S de los Laboratorios Bell, que está compuesto de un lenguaje propio y un entorno para la informática estadística, con capacidad de generación de una gran variedad de gráficos, con las siguientes características principales:

- gran potencia en la utilización de operaciones de cálculo, especialmente con matrices,
- facilidad de captura y manipulación de datos,
- amplia variedad de modelos estadísticos y técnicas gráficas,
- salida de resultados, datos y gráficos, a impresora, fichero o pantalla,
- lenguaje de programación modular que incluye condicionales, bloques de control, recursividad y gestión de entrada-salida,
- gran repositorio de librerías de código,
- multiplataforma

2.8.1. *RStudio*

Es un entorno**[47]** integrado (IDE) para trabajar con el sistema operativo R. Incluye un editor capaz de resaltar la sintaxis y se pueden realizar ejecuciones directas de código. Existen versiones para la muchos sistemas operativos.

Para este TFM, además de utilizarse de aplicación clásica *standalone*, con instalación local, se ha decidido trabajar con el software *RStudio Server* que tiene las mismas capacidades que la versión standalone de la aplicación, pero funciona en plataforma web y es instalada en un servidor; este tipo de instalación tiene mayores ventajas respecto a una instalación local.

2.8.1.1. *Shiny*

Es un potente paquete**[48]** desarrollado para lenguaje *R* (*framework*[5]) , incluido en *RStudio*, que ayuda a convertir los desarrollos en lenguaje *R* en una aplicación web interactiva sin que el usuario necesite tener conocimientos de *HTML*, *CSS* ó *Javascript*.

Al igual que con *RStudio*, para realizar este *TFM*, dado que requiere una carga de datos a un servidor y una instalación de aplicaciones no *R* (*FastQC*, *Tophat*,

---

[4] Proyecto colaborativo de software libre.

[5] Plataforma para desarrollar aplicaciones de software.

Cufflinks, *HTseq*, …) para su validación y desarrollo, se ha decidido instalar la versión servidor de *Shiny*, es decir, una instancia de *ShinyServer*. Este servidor es el que da servicio a la aplicación *c2eYs for H.Sapiens*.

2.8.1.2. *Shiny Dashboard*

Es un paquete R, que crea cuadros de mando**[49]** con Shiny, creando una atractiva y sencilla capa de visualización.

2.8.2. Librerías utilizadas *R*

La aplicación de los métodos seleccionados en la fase de preparación requiere el uso de software externo, como *FastQC ó Tophat*, y la utilización de librerías de los repositorios de *R*.

A continuación se listan las librerías que formarán parte del desarrollo final, sin incluir las básicas del sistema *R* ni de *Shiny*:

- *DT***[50]**. Provee al interfaz de usuario de una implementación de *DataTables* muy útil a la hora de presentar data frames en la aplicación *Shiny*.

- *RColorBrewer***[51]**. Permite utilizar una paleta de colores diseñada por Cynthia Brewer.

- *gageData***[52]**. Paquete de *Bioconductor* que ofrece datasets demo de varios organismos e incluso de datos cartográficos. La utilización de este paquete viene dada por la necesidad de acceso a una tabla de contajes *hnrnp.cnts* (*RNA-seq dataset on HNRNPC knockdown and control HeLa cells*) que está directamente relacionada con el conjunto de datos utilizado para realizar este TFM.

- *edgeR***[34]**. Paquete de *Bioconductor*, basado en distribuciones binomiales negativas, que permite realizar el análisis de expresión diferencial mediante implementación de métodos estadísticos exactos (clásicos) para varios grupos, así como otros métodos basados en modelo líneales generalizados (glms).

- org.Hs.eg.db**[53]**. Paquete de *Bioconductor* que da acceso a una amplia anotación del genoma humano basado en cartografía con identificadores de *Entrez genes*. Necesario para realizar el análisis de significancia biológica con *goana*.

- *Rsubread***[54]**. Paquete de *Bioconductor* utilizado para la creación de la tabla de contajes a partir de los ficheros bam generados por *Tophat*.

3. Fase 2. Desarrollo

3.1. Análisis y diseño de la aplicación *c2eYs*

Este software se ha diseñado para ser utilizado en un servidor *Shiny*. La implementación se ha realizado principalmente en lenguaje *R*, con el interfaz *Shiny Dashboard*, utilizando scripts en *Bash***[55]** para *Linux*, en las llamadas a herramientas externas al entorno *R*.

Esta aplicación cubre las siguientes características generales:

- Interfaz web basado en *Shiny Dashboard*. La aplicación se utiliza desde un navegador web. El interfaz lo genera el servidor a partir de código en lenguaje *R.*

- Capacidad del servidor para invocar programas externos. El servidor ejecuta una aplicación *Java* (*FastQC*) y otros programas binarios desde *Bash scripts*. El lenguaje *R* se utiliza para realizar todas las llamadas al sistema y generar las diferentes pantallas y resultados que son publicados en el interfaz de usuario a partir de la interacción del propio usuario.

- Monousuario. Dada las limitaciones de recursos en los equipos donde se han realizado los desarrollos, se prepara una configuración monousuario que no tiene en cuenta las sesiones.

- Toda la información mostrada en la aplicación aparecerá en idioma inglés.

El siguiente diagrama muestra un escenario posible de uso de la aplicación.

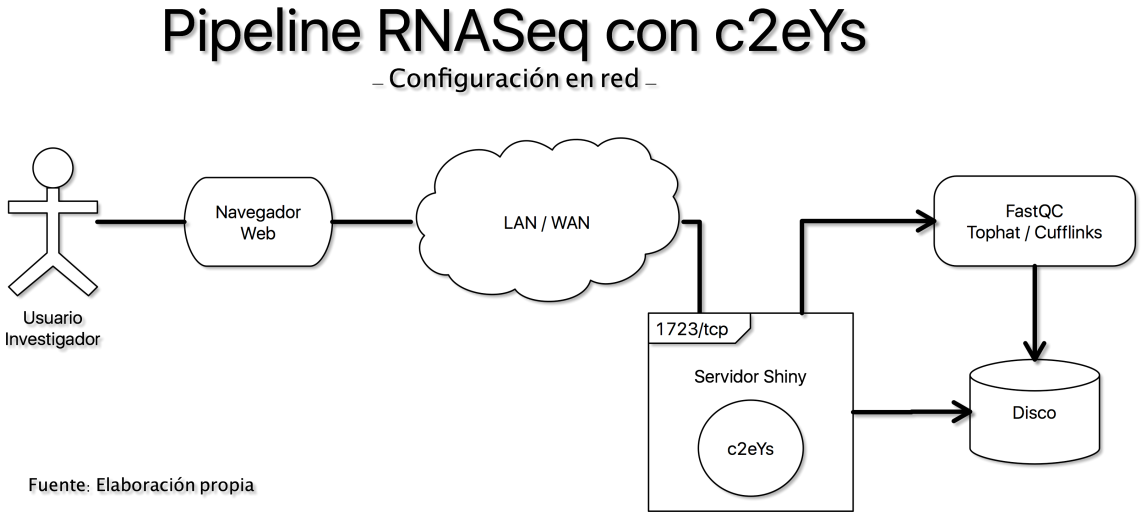


Fig.3.1.1. Configuración en red

Sus elementos son:

- Usuario Investigador. Usuario que utiliza la aplicación. Para ello se conecta a una navegador web e introduce la URL que da acceso a dicha aplicación. EN nuestro caso: `http://193.146.74.35:1723/c2eYs`

- Navegador web**[56]**. Es un programa que sirve para utilizar aplicaciones diseñadas para Internet que normalmente se conectan a servidores web mediante el protocolo de comunicaciones *HTTP***[57]**. El navegador se conecta a través de la *LAN / WAN***[58]** y utiliza el interfaz de usuario de la aplicación Shiny.

- *LAN / WAN.* Medio de conexión entre cliente (navegador w*eb*) y el servidor.

- Servidor *Shiny*. Software de servidor que gestiona las peticiones de la aplicación *c2eYs*, tanto la entrada como la salida. El puerto definido de escucha del servicio es el 1723/tcp.

- *c2eYs*. Aplicación desarrollada en lenguaje *R* que crea un pipeline para un análisis *RNASeq*, instalada sobre el servidor *Shiny* (*Shiny Dashboard*).

- *FastQC* y *Tophat*. Programas a ser invocados desde la aplicación *c2eYs*, a través de Shiny.

- Disco. Espacio físico donde se almacenan los resultados de las aplicaciones y programas anteriores, así como ficheros temporales. También es el lugar donde se graban los ficheros que contienen las muestras utilizadas para realizar los análisis. La aplicación *Shiny* lee de aquí los resultados generados, para después enviarlos al navegador web del usuario.

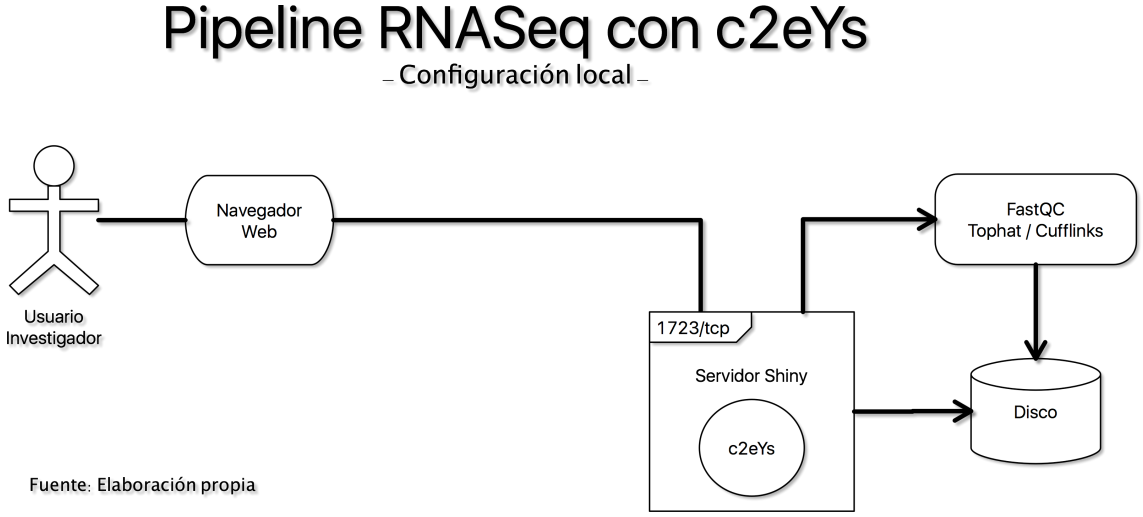


Fig.3.1.2. Configuración local

Dada la limitación en el diseño monousuario de la aplicación (concurrencia máxima de 1 usuario), una configuración local puede ser otro escenario posible e interesante. El diagrama anterior muestra esta posibilidad.

Se debe tener en cuenta que, si bien una instalación local normalmente se realiza en un ordenador personal, con configuración más limitada que la que pueda tener un servidor dedicado, pueden aparecer diferentes eventualidades debido al consumo excesivo de los diferentes recursos hardware (*CPUs*, memoria y espacio en disco), penalizando el rendimiento en la ejecución de la aplicación.

3.2. Casos de uso

En esta aplicación no existe mecanismo de autenticación por lo que el acceso a la misma no necesita de la introducción de unas credenciales iniciales. Es por ello, que una vez se ha accedido a la URL de la aplicación, se muestran todas las opciones principales. En la zona izquierda, aparecen dichas opciones. La zona de la derecha es el área principal de trabajo donde se seleccionan, introducen y muestran los resultados, es decir, la zona de interacción principal del usuario.

El siguiente diagrama muestras los casos de uso generales:

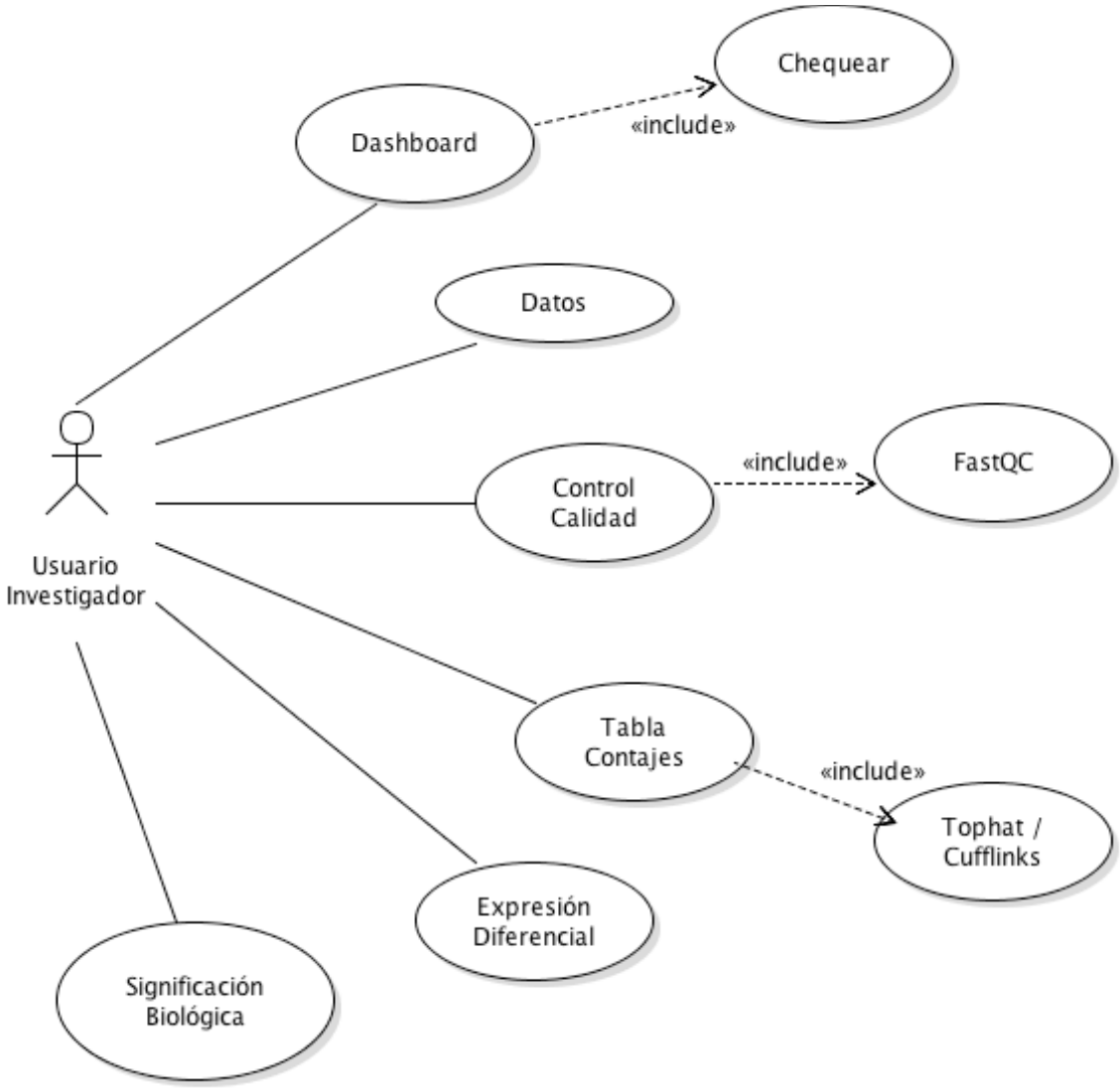


Fig.3.2. Diagrama de casos de uso genéricos (Elaboración propia)

A continuación se realiza un breve análisis de estos casos de uso. En todos ellos el único actor definido es el usuario investigador.

3.2.1. Dashboard

Permite visualizar el estado de las tareas de cada una de las fases del pipeline; se muestra un resumen en base a cajas, iconos y colores. No existen precondiciones ya que la información mostrada es la disponible y no hay restricción alguna para mostrarla.

3.2.1.1. Chequear

Realiza un chequeo de todos los paneles de datos a mostrar en el dashboard. Este caso queda integrado en la lógica del Dashboard.

3.2.3. Datos

Desde esta opción se decide de cuantas muestras va a constar el pipeline y cuáles son los ficheros que tendrán sus datos. Los datos pueden estar ya precargados de otro momento o modificarse según criterio.

3.2.4. Control de Calidad

Muestra una pantalla de gestión que permite generar o cargar un análisis de control de calidad ya generado. Se visualizan estados de los test del análisis con tablas de datos y gráficos disponibles generados por FastQC. La única precondición para que este caso se puede realizar es la realización del caso de uso Datos.

3.2.4.1. *FastQC*

Se encarga de ejecutar el control de calidad mediante la aplicación FastQC. Devuelve el estado de cada uno de los pasos tomados. La llamada a este caso queda integrada en el caso de uso Control de Calidad.

3.2.5. Tabla de Contajes

Desde esta opción, se puede generar una nueva tabla de contajes o utilizar alguna creada previamente. Esa tabla puede visualizarse y descargarse.

3.2.5.1. *Tophat*

Es el ocupado de realizar todas las operaciones necesarias para obtener la tabla de contajes. Devuelve el estado de cada una de las tareas realizadas. El caso de uso Tabla de Contajes es el responsable de ejecutar este caso.

3.2.6. Expresión Diferencial

Una vez se ha seleccionado una tabla de contajes, se puede utilizar esta opción. El usuario introduce agrupaciones para cada muestra y puede realizar diferentes tests con ellos, mostrando tablas de datos y gráficos con los resultados obtenidos.

La precondición necesaria para este caso de uso es la realización del caso de uso Tabla de Contajes, en especial, tener seleccionada una tabla de contajes.

3.2.7. Significación Biológica

La utilización de esta pantalla es dependiente de la ejecución del Test de Decisión, realizado en la opción Expresión Diferencial. Si se ha ejecutado dicho test, se realizará el análisis de significación biológica, obteniendo los resultados en una tabla en pantalla.

La precondición de este caso de uso es la realización del caso de uso Expresión Diferencial.

3.3. Configuración para desarrollo y pruebas

A fin de valorar la implementación y realizar las pruebas en distintos entornos, se han utilizado 3 ordenadores distintos con características hardware diferentes.

Servidor	Procesadores / Ghz	Memoria (GB) / Mhz	Disco / Tipo	S.O.
Core2Duo i3	2 / 2.8 Ghz	6 / 800 DDR2	512 / SSD	OSX 10.11.6 64 bits
Quadcore i5	4 / 3.2 Ghz	8 / 3200 DDR4	450 / SATA	Ubuntu 17.04 64 bits
Atom n450	2 / 1.66 Ghz	1 / 666 DDR2	160 / SATA	Ubuntu 17.04 64 bits

Tab.3.3. Configuración para desarrollo y pruebas (Elaboración propia)

En ellos, se ha instalado y configurado el siguiente software:

- Lenguaje *R* versión 3.3.3, con una serie de paquetes del repositorio principal de *CRAN* y del proyecto *BioConductor*.
- *RStudio Server* 1.0.136. *IDE*[6] para implementación del código.
- *Shiny Server* 1.5.1.834. Servidor de aplicación.
- *Java SE Runtime Environment* 1.8.0_121. Necesario para la ejecución de *FastQC*.

Como era de esperar, el mejor rendimiento se obtuvo con el ordenador dotado con *Quadcore*, mayor número de procesadores y tamaño de memoria, ambos con velocidades mayores que el resto.

Las tareas que permiten seleccionar el uso de procesadores se han modificado para utilizar únicamente uno y permitir la utilización del ordenador para otras tareas al margen de esta aplicación.

Se ha podido constatar que las versiones de *Shiny Server* y *RStudio Server*, en su versión libre, no realizan aprovechamiento alguno de recursos, pudiendo llegar a ralentizar totalmente el ordenador por falta de ellos.

[6] Entorno integrado de desarrollo

3.4. Interfaz de usuario

La finalidad de esta aplicación es el desarrollo de un pipeline en un entorno monousuario por lo que, en un principio, se considera innecesaria la creación de una pantalla de acceso donde el usuario introduce sus credenciales. Esta tarea se deja como posible *TO-DO*.

El usuario, arrancará un navegador web, donde introducirá la URL de acceso a la aplicación y aparecerá la pantalla principal. En esta pantalla principal se muestra el nombre de la aplicación en la zona superior, además de un botón de mostrar / ocultar el menú principal.

El resto de la pantalla queda repartida en dos partes: izquierda con menú de opciones principales y derecha con la gestión de la opción de menú seleccionada.

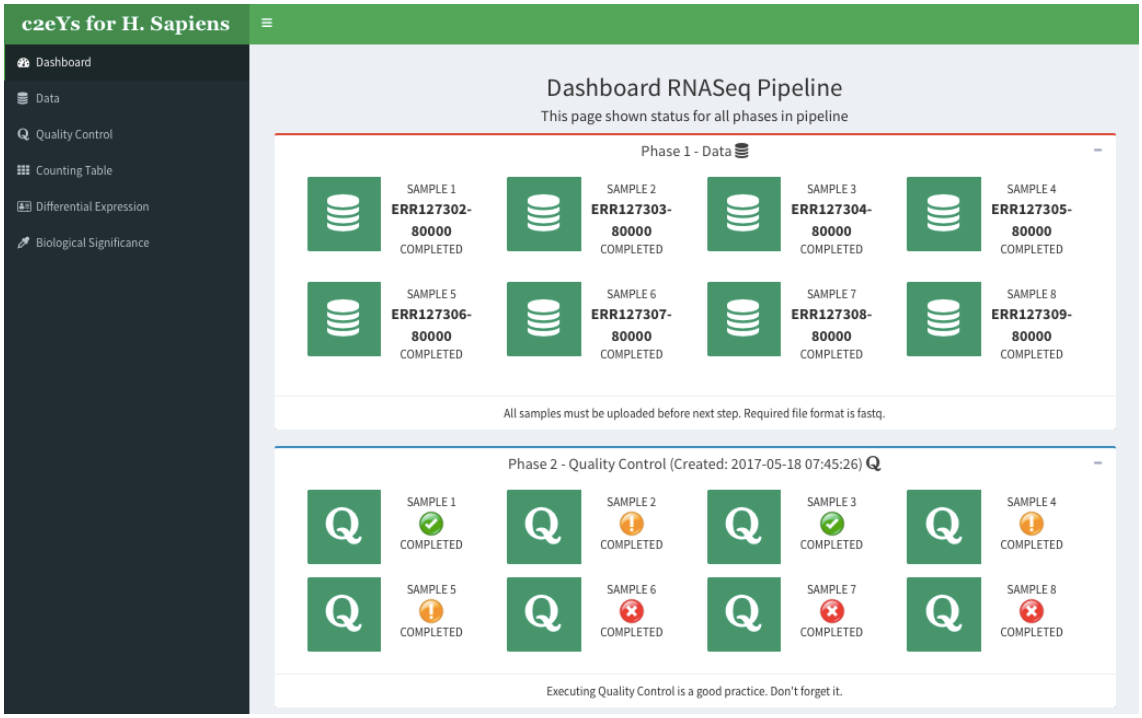
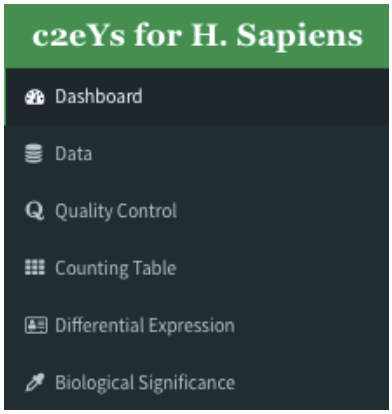


Fig.3.4. Interfaz de usuario

En todas las pantallas existen textos con indicaciones y ayudas, para saber qué es lo que hacer en todo momento. Hay referencias a pasos previos que hay que realizar antes de abordar los siguiente. La navegación por la aplicación se realiza de manera cómoda e intuitiva.

3.4.1. Menú de opciones



Muestra la lista de opciones disponibles: Dashboard, Control de Calidad, Tabla de Contajes, Expresión Diferencia y Significación Biológica.

Por defecto, aparece seleccionada la primera de las opciones, que se corresponde con el Dashboard.

Todas las opciones, aunque con sus restricciones de uso, son accesibles en cualquier momento.

Fig.3.4.1. Menú de opciones

3.4.2. Área de trabajo

Se corresponde con la mayor parte de la página y es la zona donde el usuario interacciona con la aplicación y se muestran los resultado al detalle.

3.4.2.1. Dashboard

Esta opción es la que aparece activada en el momento de acceder a la aplicación. Muestra un resumen del estado del análisis que se está realizando. La única interacción prevista con el usuario es la capacidad de los paneles de esconder o mostrar sus contenidos; por defecto, son mostrados. Existen tareas que son más costosas en tiempo que otras, algunas de ellas se ejecutan en segundo plano, y en esta página también se muestra su estado.

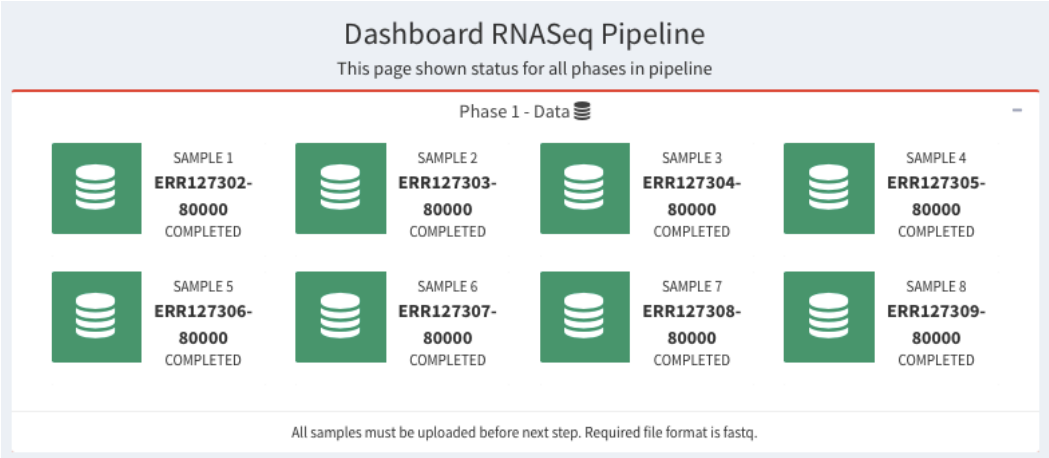


Fig.3.4.2.1.1. Dashboard Datos

La página se subdivide en 5 paneles que se corresponden con las 5 fases del análisis: Carga de Datos, Control de Calidad, Tabla de Contajes, Expresión Diferencial y Significancia Biológica.

- Fase 1 - Datos. Muestra tantas cajas como el número de ficheros seleccionados para cargar. En cada caja aparece una icono con un fondo, verde ó rojo, en función de si se ha realizado la carga o no, un titulo (SAMPLE) que indica el número de muestra cargada y que es la referencia en el resto de opciones de la aplicación, el nombre del fichero que contiene la muestra y el estado en el que se encuentra: COMPLETED (cargado correctamente) ó PENDING (pendiente de cargar). La figura 3.4.2.1.1 muestra un ejemplo de esta fase.

- Fase 2 - Control de Calidad. La información que aparece en este panel, proviene de los resultados generados por la aplicación FastQC. En la propia cabecera, se indica cuando se inició el último control de calidad. Esta información se actualiza automáticamente cada 5 segundos. Al igual que la fase anterior, se muestran tantas cajas como muestras definidas en el análisis. Cada una de ellas muestra un icono identificativo con un color de fondo, verde, naranja ó rojo, dependiendo de si el estado del análisis de control de calidad se ha completado, está en proceso o no se ha iniciado, respectivamente. En la parte derecha-inferior de cada caja, se muestra una leyenda indicando dicho estado. Encima de dicha leyenda se muestra un icono representativo del estado final e individual de cómo ha concluido el análisis de la muestra en cuestión. Una ejemplo de este panel puede visualizarse en el figura 3.4.2.1.2. El icono de cada test puede ser uno de los siguientes:

  todos los tests han resultado correctos,

  en algún test existen avisos,

  algún test ha fallado.

Fig.3.4.2.1.2. Dashboard Control de Calidad

- Fase 3 - Tabla de contajes. Este panel muestra información dependiente de la configuración existente en la opción de Tabla de Contajes. Así tenemos las siguiente posibilidades de selección:

    - Generar Tabla de Contajes, este panel muestra 2 cajas que se corresponden con las 2 tareas a realizar en la generación de la tabla de contajes: Análisis Individual y Proceso de Creación.

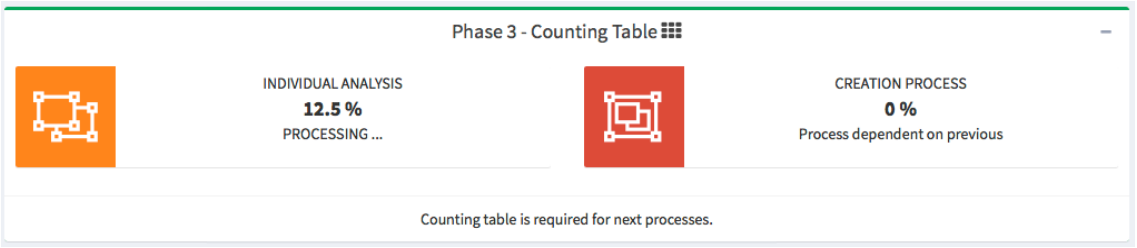


Fig.3.4.2.1.3. Dashboard Tabla de Contajes - Generada

Cada una de esas cajas tiene un icono identificativo con un color en base al estado de la tarea en sí, siendo posibles 4 colores: verde que indica que todo ha ido bien, naranja cuando se está realizando la ejecución, violeta si la ejecución de dicha tarea ha terminado con error y rojo si la tarea no se ha iniciado. Además se incluye un número que indica el porcentaje de ejecución de la tarea. Hay que tener en cuenta que este indicador no es proporcional al tiempo si no al número de subtareas en cada una de las tareas.

- Cargar una Tabla, si se carga el fichero se muestra el nombre del mismo; en caso contrario, se indica que el fichero está pendiente de cargar.

- Usar Tabla Generada, se indica el nombre y fecha de la tabla que se va a utilizar. Esta tabla es el resultado de la opción Generar Tabla de Contajes una vez han concluido correctamente todas las tareas.

- Usar Tabla *Demo*. Se carga una tabla por defecto, para realizar pruebas en la aplicación. La tabla cargada se obtiene del paquete R *gageData*.

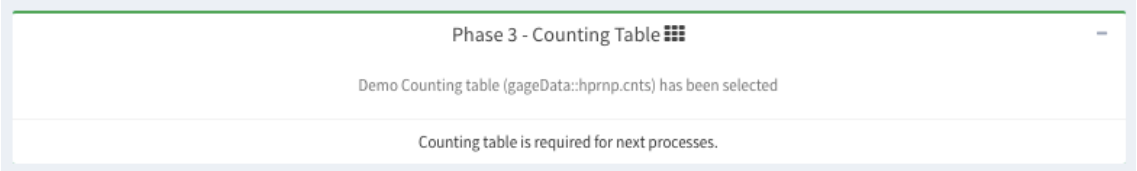


Fig.3.4.2.1.4. Dashboard Tabla de Contajes - Demo

- Fase 4 - Expresión diferencial. La utilización de esta opción pasa por tener una tabla de contajes disponible, ya sea generada, seleccionada o cargada. Si no existe, aparece un mensaje informándolo. En este panel, además del mensaje de ausencia de tabla de contajes, pueden aparecer mensajes de la actividad llevada a cabo en la opción de Expresión Diferencial del menú principal, como por ejemplo “Mostrando Filtrado y Normalización”. Si el usuario accede a la opción Expresión diferencial, puede observar los resultados de dicho análisis.

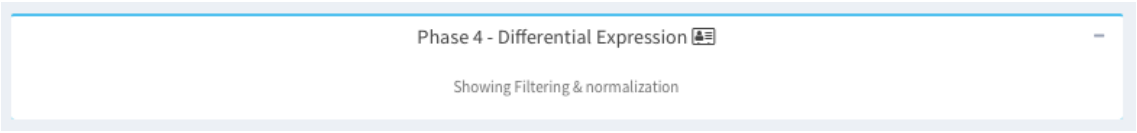


Fig.3.4.2.1.5. Dashboard Expresión Diferencial

- Fase 5 - Significación Biológica. Este panel muestra información sobre los datos que se están mostrando en la opción específica de Significación Biológica. Si no se ha realizado el último test del análisis de expresión diferencial, llamado Test de Decisión, se muestra un mensaje indicando que está pendiente de ejecución; en caso contrario, informa que el resultado se está mostrando.

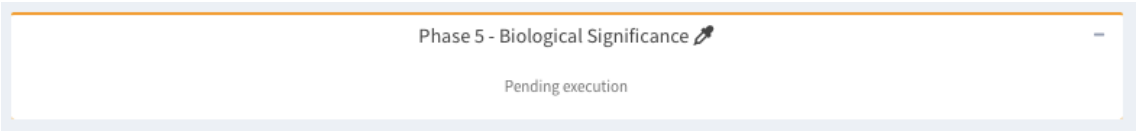
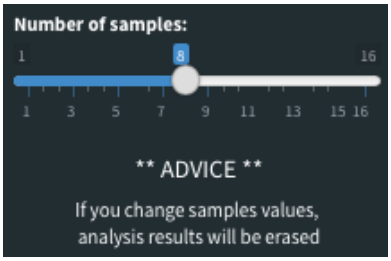


Fig.3.4.2.1.6. Dashboard Significación Biológica

3.4.2.2. Datos

No se puede realizar el análisis sin datos por lo que esta opción es una de las más importantes por ser la que nutre de información al posterior análisis.



Cuando se selecciona la opción, en la parte inferior del menú (zona izquierda de la pantalla) aparece un control (*slider*) que permite al usuario, seleccionar el número de muestras con las que se va a trabajar.

Fig.3.4.2.2.1. Slider Muestras

Se ha establecido un máximo de 16 muestras, que es parametrizable desde el código fuente de la aplicación[7]. Se debe prestar atención a la advertencia que se muestra sobre el cambio de las muestras ya que, una vez se han cambiado las muestras, el análisis ya no se corresponde con ellas.

[7] Fichero genericos.R

El “área de trabajo” para esta opción, permite seleccionar y cargar los ficheros de las muestras. Según se van cargando, el panel de Dashboard relacionado con esta fase, se va actualizando.

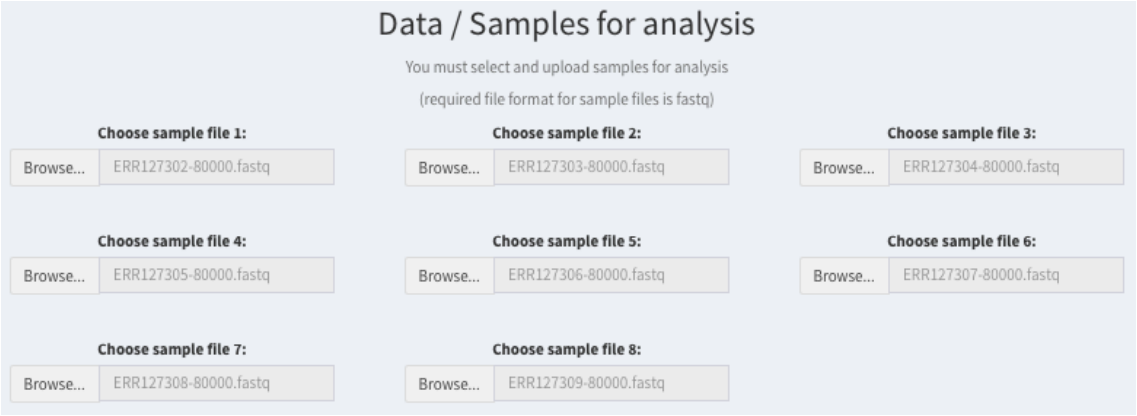


Fig.3.4.2.2.2. Carga de Datos

El formato requerido de los ficheros a cargar en el servidor es el FastQ. Si no se introduce un fichero de este tipo, se indica que el fichero tiene un formato incorrecto.

3.4.2.3. Control de Calidad (*FastQC*)

Esta opción permite realizar un control de calidad mediante la aplicación FastQC ó leer los resultados ya obtenidos y que están almacenados en el servidor. El botón de leer los resultados previos, únicamente aparecerá si la aplicación lo detecta. Dado que una vez se empiece a utilizar la aplicación, siempre habrá unos datos de control de calidad previo, el botón de lectura de resultados aparecerá siempre a no ser que se borren dichos datos, en el servidor.

La ejecución de un control de calidad, borra todos los resultados previamente calculados, si no se cargan todos los ficheros de las muestras definidas. Mientras se está ejecutando el análisis, se puede ver su evolución desde el respectivo panel en el Dashboard.

Una vez completado el análisis de todas las muestras propuestas, se cargan automáticamente los resultados de la misma forma que si se hace clic en el botón Leer Resultados. En ambos casos, aparecen dos nuevos botones de selección: uno para seleccionar si se quieren visualizar datos o gráficos y el otro para mostrar el resultado del test.

Después de hacer la selección, aparece un panel con pestañas que se corresponden con los resultados generados para cada una de las muestras en el test seleccionado. El nombre de la pestaña se forma con el número de muestra y la fecha y hora de obtención de los resultados del test seleccionado.

También se incluye un icono con el resultado global del mismo. Este icono es el mismo al mostrado en el Dashboard.

A continuación se muestra un ejemplo de salida gráfica, donde está seleccionada la muestra 1 generada en fecha 18 de mayo de 2017 a las 07:45:34 horas con un *warning* en el test *Kmer*.

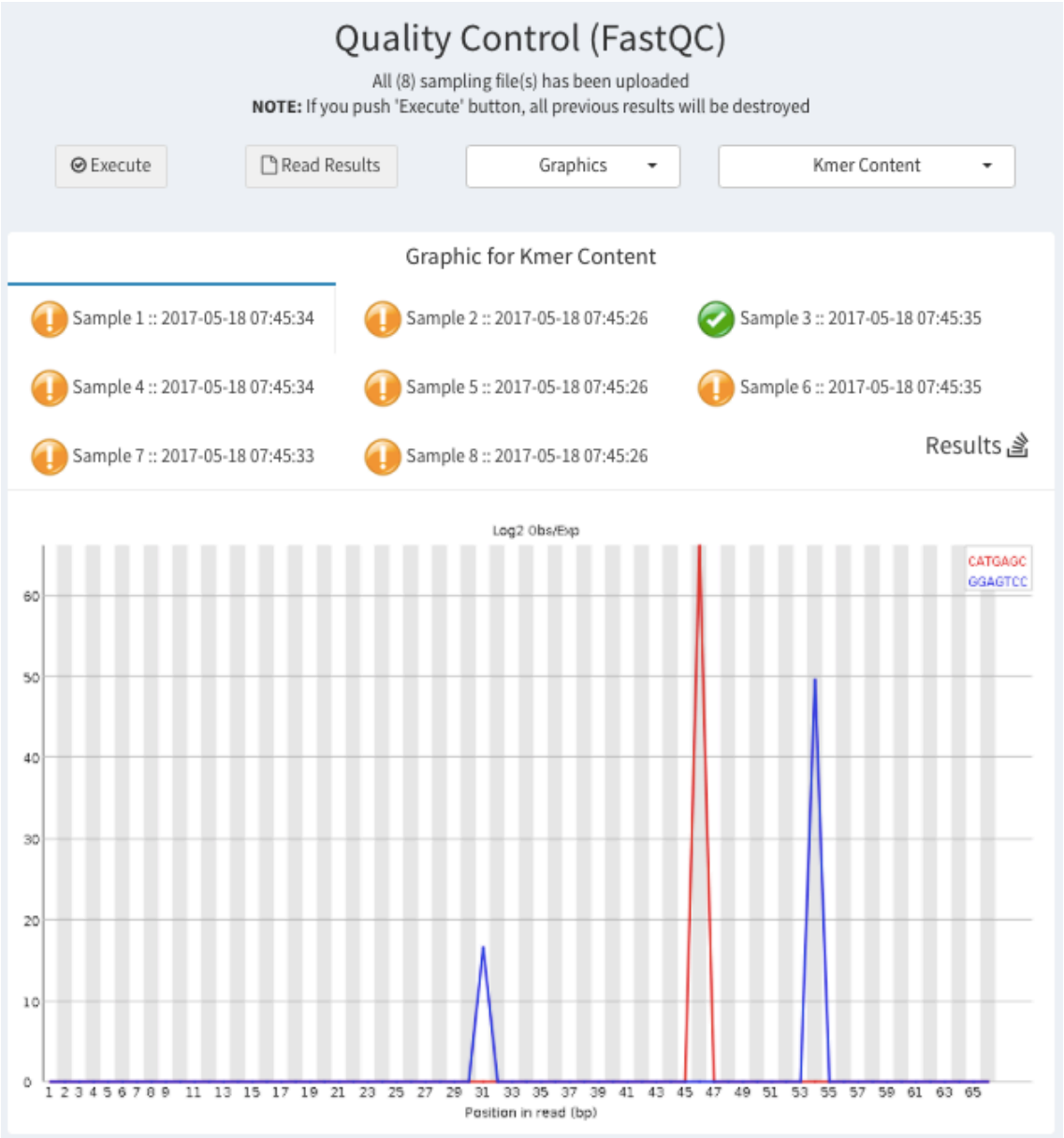


Fig.3.4.2.3.1. Control de Calidad - Gráfica

En la figura 3.4.2.3.2, se muestra el siguiente ejemplo con la tabla de datos del mismo test, para la misma muestra que en el gráfico anterior.

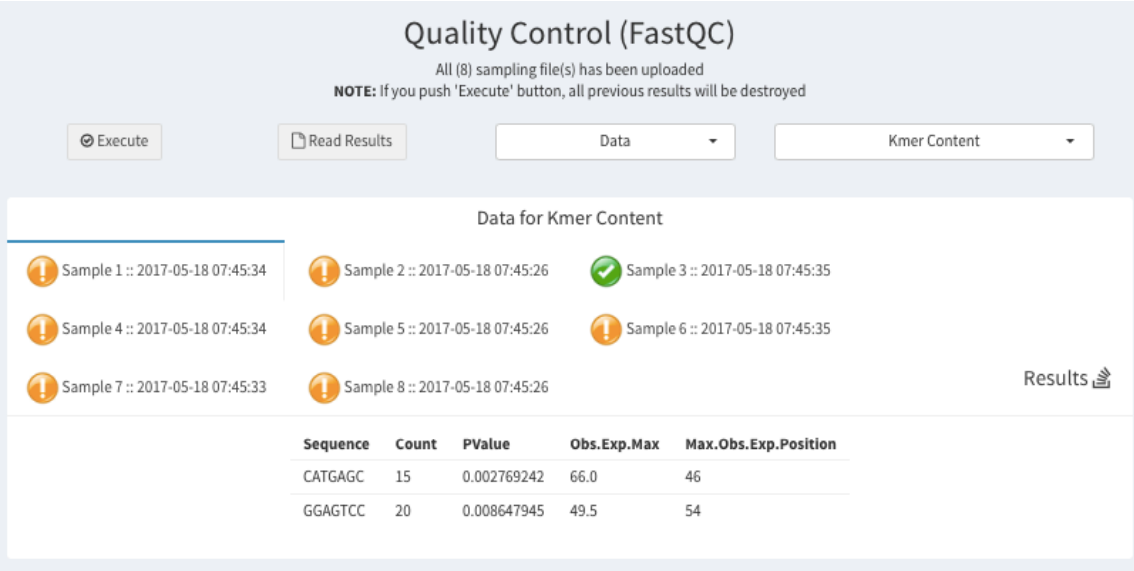


Fig.3.4.2.3.2. Control de Calidad - Datos

La ejecución de la aplicación *Java FastQC*, conlleva diversas comprobaciones sobre los datos en bruto para asegurarse que no hay problemas o sesgos que puedan afectar al análisis posterior. Una vez ejecutado el control de calidad, *FastQC* asigna un resultado a cada evaluación por muestra y módulo, pudiendo ser correcto, con avisos o con errores.

La única premisa para la ejecución de este control de calidad es tener cargados los ficheros y que estos tengan el formato .fastq.

La salida generada por *FastQC* consiste en una serie de ficheros entre los que se encuentran `fastqc_data.txt` y `summary.txt`. Este último contiene un resumen del estado final de cada uno de los test realizados durante el control de calidad. Por su parte, `fastqc_data.txt`, contiene una detallado informe sobre esos mismos tests.

Los tests disponibles en *c2eYs* son los mismos que los generados por la aplicación *FastQC***[2]**:

- Basic Statistics. Contiene los datos básicos estadísticos con:
    - el nombre del archivo que se analizó,
    - el tipo de archivo, bases o espacio de colores,
    - la codificación ASCII,
    - el recuento del número total de secuencias procesadas (reales o estimadas),
    - las secuencias marcadas como de “calidad pobre”,
    - la longitud de la secuencia más corta y más larga (si es la misma sólo se muestra una),
    - y el % GC de toda las bases en todas las secuencias.

Este test no devuelve fallo ni aviso en caso alguno.

- Per base sequence quality. Muestra una visión general del rango de valores de calidad a través de todas las bases en cada una de las posiciones del archivo *FastQ*. Por cada una de las posiciones se dibuja un gráfico tipo *Boxplot* donde:

    - la línea roja es el valor de la mediana,
    - el cuadro amarillo representa el intervalo entre cuartiles (25 a 75%),
    - las líneas horizontales superiores e inferiores (bigotes) representan los puntos del 10% y 90%,
    - la línea azul representa la calidad media,
    - el eje *Y* muestra la calidad. A mayor valor, mejor calidad (verde - buena, naranja - razonable, roja - mala).

    El test muestra advertencias si el cuartil inferior para cualquier base es menor de 10 o si la mediana para cualquier base es menor a 25. Por otro lado, fallará si el cuartil inferior es menor de 5 para cualquier base o si la mediana para cualquier base es menor de 20.

- Per sequence quality scores. Con este test se puede comprobar si una proporción significativa de secuencias tiene una calidad global baja; esto podría indicar algún problema sistemático. Este test genera una advertencia si la calidad media más frecuentemente observada es inferior a 27, lo que equivale a una tasa de error de 0.2%. Fallará si la calidad media frecuentemente observada es inferior a 20, lo que equivale a una tasa de error del 1%.

- Per base sequence content. Muestra la proporción de cada posición de cada una de las cuatro bases, en un archivo. En condiciones normales, la cantidad relativa de cada base, refleja la cantidad total de esa base en el genoma. Se muestra una advertencia si la diferencia entre A y T ó G y C, es mayor que 10% en alguna posición. Este test mostrará fallo si la diferencia entre A y T ó G y C es mayor que el 20 % en alguna posición.

- Per base GC content. Se representa el contenido de GC en cada posición del archivo. Se muestra una advertencia si el contenido de GC de alguna base se desvía más del 5% del contenido medio de GC. Cuando el contenido de alguna base se desvía más de un 10% del contenido medio de GC, se mostrará un fallo.

- Per sequence GC content. Mide el contenido de GC a lo largo de la longitud de cada secuencia del archivo y la compara con una distribución normal modelada del contenido de GC. La advertencia aparece si la suma de las desviaciones de la distribución normal representa más del 15% de las lecturas. Por el contrario, se mostrará una fallo si la suma de las desviaciones de la distribución normal representa más del 30% de las lecturas.

- Per base N content. El módulo representa el porcentaje de bases por cada posición que fue sustituida por N en lugar de por una base convencional. No es raro ver una proporción muy baja de Ns especialmente al final de la secuencia. Si esta proporción crece por encima de un pequeño porcentaje sugiere que el análisis ha podido ser incorrecto. Este test muestra una advertencia si alguna posición muestra una contenido de N mayor al 5%. Si alguna posición muestra un contenido superior a 20%, se devolverá un fallo.

- Sequence Length Distribution. La mayor parte de los secuenciadores de alto rendimiento generan fragmentos de secuencia de longitud uniforme, pero a veces pueden aparecer lecturas de longitudes con una gran variabilidad. Este test crea un gráfico con las distribución de los tamaños de los fragmentos existentes en el archivo. Se muestra una advertencia si todas las secuencias no tienen la misma longitud. Por el contrario, aparecerá un fallo si alguna de las secuencias tiene longitud 0.

- Sequence Duplication Levels. Se muestra el grado de duplicación de cada secuencia en el conjunto y se crea un gráfico que muestra el número relativo de secuencias con diferentes grados de duplicación. Sólo analiza las secuencias que se producen en las primeras 200.000 secuencias en cada archivo. Se muestra una advertencia si las secuencias duplicadas superan el 20% del total. Si superan el 50% se mostrará un fallo.

- Overrepresented sequences. Una secuencia que está muy sobre-representada significa que tiene un alto valor de significación biológica. Sólo se rastrean en todo el archivo, las secuencias que aparecen en las 200.000 secuencias primeras. Por cada secuencia sobre-representada se buscan coincidencias en una base de datos de contaminantes. Los hits deben ser de al menos 20 pb de longitud y un fallo como máximo. Este test devuelve una advertencia si se encuentra alguna secuencia representada más del 0.1% del total; si se representa más del 1%, se muestra un error.

- Overrepresented Kmers. Se cuenta el enriquecimiento de cada 5-mer dentro de la librería de secuencias. Calcula un nivel esperado en el que el k-mer[8] debiera aparecer y luego usa el recuento real para calcular una relación entre observada / esperada para ese k-mer, muestra una lista de hits y yb gráfico para los 6 primeros hits con un patrón de enriquecimiento de ese k-mer en toda la longitud de las secuencias. Este test se limita al análisis del 20% de la librería de secuencias y lo extrapola al resto. Mostrará una advertencia si algún k-mer se enriquece más de 3 veces en total o más de 5 veces en alguna posición individual. Se devolverá un fallo si algún k-mer se enriquece más de 10 veces en alguna posición.

La opción de control de calidad de c2eYs se muestra en la figura 3.4.2.3.3. Después de que el usuario haga clic en el botón de ejecutar, se ejecuta un script en Bash que a su vez ejecuta, en paralelo y en segundo plano, el control de calidad con FastQC, para cada una de los ficheros FastQ previamente cargados. Los resultados se almacenan en un directorio distinto por fichero.

[8] Referido a todas las subcadenas posibles de longitud k contenidas en una cadena

baja el subdirectorio `www/results/qc`. Dichos directorios se enumeran de 1 a n, en función del número secuencia de muestra asignado en la carga del mismo. Además se crean un fichero <número>.err y <número>.log. El primero de ellos recoge los errores durante la ejecución, pero este dato no es leído desde la aplicación. *c2eYs* consulta periódicamente el segundo fichero (log) buscando la cadena OK que marca el fin del control de calidad. En este punto no se tienen en cuenta los errores posibles; si el proceso falla, nos se mostrarán datos en la aplicación.

Los datos generados son leídos mediante el botón Leer Resultados. En ese momento, c2eYs accede a los ficheros generados por FastQC en los diferentes directorios.

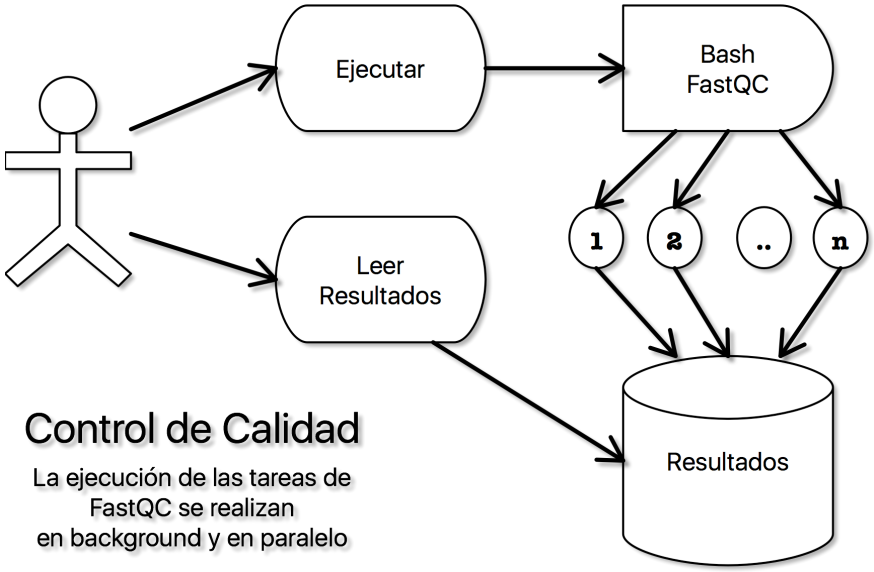


Fig.3.4.2.3.3. Flujo Control de Calidad (Elaboración propia)

3.4.2.4. Tabla de Contajes (*Tophat*)

Mediante esta opción es posible la creación de una tabla de contajes en función de las muestras cargadas, la selección de una tabla de contajes ya generada o de una tabla que se cargue en el servidor, o de una tabla de demo preestablecida, para los posteriores análisis con los datos ya cargados.

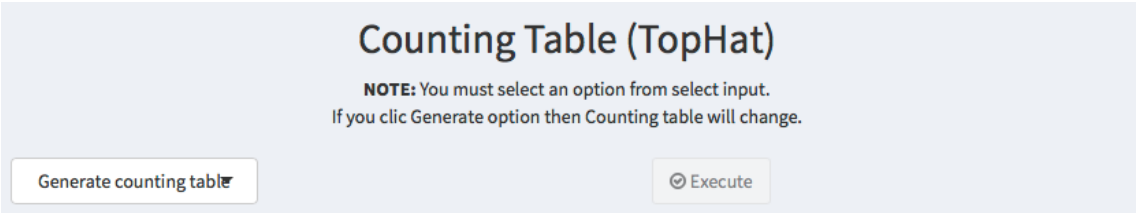


Fig.3.4.2.4.1. Generación de Tabla de Contajes (Generate)

En la figura anterior, se muestra cómo se ha configurado la pantalla después de seleccionar que se va a generar una nueva tabla de contajes. En este caso, la aplicación muestra un botón Ejecutar que permite iniciar el proceso.

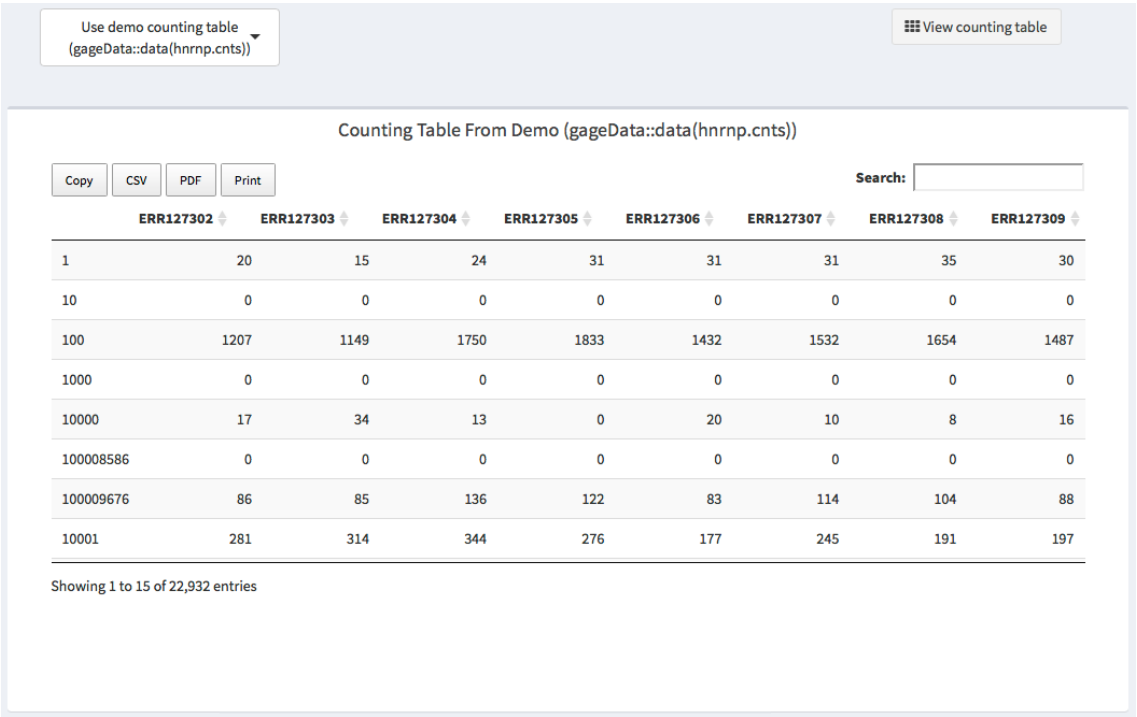
Una vez iniciado el proceso, se debe esperar a que termine. Para posteriores versiones se puede implementar la opción de cancelar el proceso mediante otro botón (*TO-DO*).

Otra opción seleccionable consiste en la carga de una tabla de contajes personalizada, en el servidor. Se realiza como en la siguiente figura.

Fig.3.4.2.4.2. Generación de Tabla de Contajes (Upload)

Donde aparece un botón de búsqueda y selección del fichero que contiene la tabla de contajes a cargar. El formato de dicha tabla debe ser delimitado por tabuladores.

La otra posibilidad es utilizar una tabla demo. Esta tabla se obtiene del paquete *gageData* y está relacionado directamente con el conjunto de datos seleccionado en este TFM.



	ERR127302	ERR127303	ERR127304	ERR127305	ERR127306	ERR127307	ERR127308	ERR127309
1	20	15	24	31	31	31	35	30
10	0	0	0	0	0	0	0	0
100	1207	1149	1750	1833	1432	1532	1654	1487
1000	0	0	0	0	0	0	0	0
10000	17	34	13	0	20	10	8	16
100008586	0	0	0	0	0	0	0	0
100009676	86	85	136	122	83	114	104	88
10001	281	314	344	276	177	245	191	197

Fig.3.4.2.4.3. Visualización de Tabla de Contajes

Una vez se ha seleccionado una tabla de contajes, aparece un nuevo botón, que permite visualizar la tabla de contajes seleccionada. Este hecho queda representado en la figura anterior Fig.3.4.2.4.3 donde, se incluyen controles para poder realizar búsquedas, ordenaciones varias y exportaciones de los datos en distintos formatos.

Se debe tener en cuenta que no es posible realizar los análisis de Expresión Diferencial y Significación Biológica sin haber seleccionado una tabla de contajes, ya sea generada o cargada.

El proceso de generación de la tabla de contajes se ha dividido en 2 partes:

- Ejecución de *Tophat* sobre cada una de las muestras,

- Creación de tabla de contajes, una vez se han obtenido los ficheros .bam.

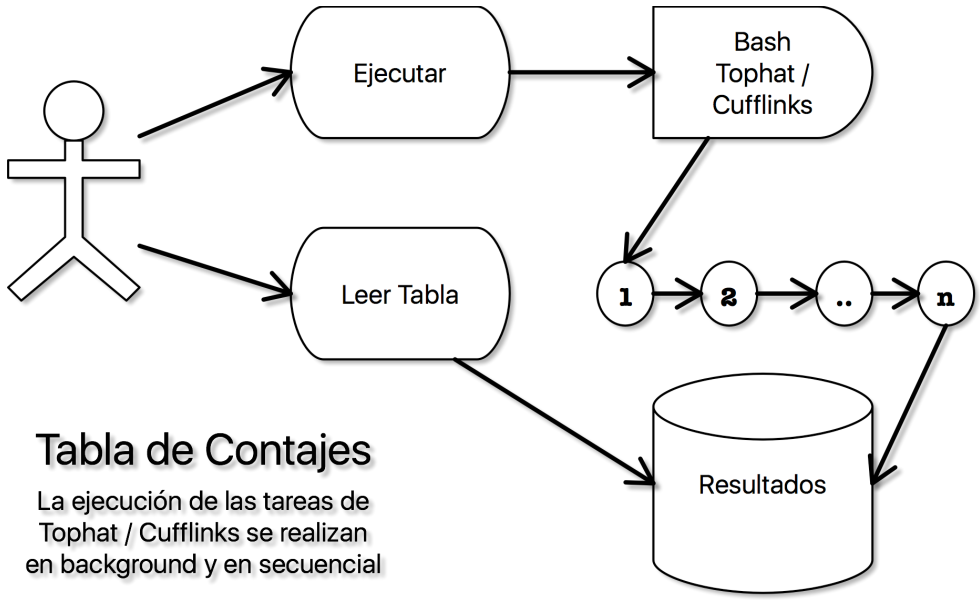


Fig.3.4.2.4.4. Flujo Tabla de Contajes (Elaboración propia)

Antes de ejecutar la primera fase, se realiza una preparación para crear un *script Bash* que será el ejecutado en segundo plano (background). Este script contiene tantas llamadas al *script* `ct.0.bash` como número de ficheros de muestras se hayan definido, para posteriormente, en un directorio llamado `bam`, agrupar enlaces a los diferentes ficheros .bam de cada muestra para crear las tablas de contajes a partir del *script* `crearCT.R`

La figura 3.4.2.4.4. muestra cómo se gestiona la opción de Tabla de Contajes. por un lado la ejecución se realiza en secuencia, es decir, cuando un proceso termina se lanza el siguiente, como una cadena, hasta que terminan todos. Por otro lado, la opción de Leer Tabla permite la captura de la tabla ya generada.

El script `ct.0.bash` realiza varios pasos para cada fichero:

- chequeo de parámetros de llamada,
- chequeo de espacio libre en disco,
- ejecución de Tophat.

El script `crearCT.R` crea la tabla de contajes a partir la librería *Rsubread* y la función *featureCounts*.

Estos procesos consumen muchos recursos, sobre todo memoria y *CPU*. Para realizar la aplicación se han ejecutado para utilizar una única *CPU*, ya que el servidor donde se instala realiza otras operaciones y se quieren evitar problemas de rendimiento en otras aplicaciones de ese mismo servidor. Esta decisión también se podría aprovechar si se realizasen distintas instalaciones de *c2eYs* en el mismo servidor, pero en distintos directorios, en un máquina con varias CPUs, pudiéndose utilizar una *CPU* por instalación.

Al final del proceso, se crea un fichero llamado `tabla.info` que advierte a la aplicación que ya se ha creado la tabla de contajes. Es entonces, cuando el proceso se da por finalizado.

3.4.2.5. Expresión Diferencial (edgeR)

Al acceder a esta opción, si ya se ha seleccionado la tabla de contajes, se muestra una pantalla similar a la expuesta en la siguiente figura.

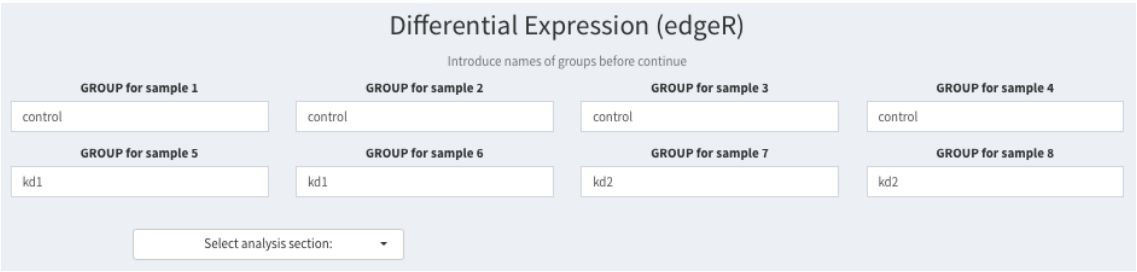


Fig.3.4.2.5.1. Asignación grupos Expresión Diferencial

En un principio, no aparece el botón de selección de acción inferior. En su lugar aparece un mensaje indicando al usuario que primero debe asignar grupos a cada una de las muestras a analizar; para poder avanzar, se deben asignar identificadores de grupo a cada una de ellas.

En las acciones que requieren tiempo de espera, se muestra una imagen como la de la figura siguiente.

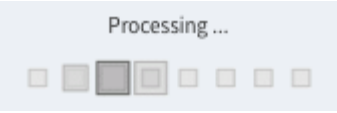


Fig.3.4.2.5.2. Espera proceso

Las secciones (tests) posibles del análisis son:

- Filtrado y normalización. En este caso, no se realiza una normalización estadística, realizando una transformación a una distribución normal. Se intenta mitigar los errores introducidos por los secuenciadores. Esta operación se realiza mediante la función calcNormFactors que calcula factores de normalización para alinear columnas de una tabla de contajes.

  La siguiente figura muestra la apariencia de pantalla cuando se selecciona esta opción. Consta de 2 pestañas: Data y Plot. En la primera se muestran los datos en forma de tabla, en la segunda se da la posibilidad de generar, al vuelo, varios tipos de gráficos: MDS-BCV, MDS-logFC, Dendograma y Heatmap, mostrados en las figuras inferiores. El control de muestras únicamente es válido para los gráficos MDS.

Fig.3.4.2.5.3. Filtrado y Normalización

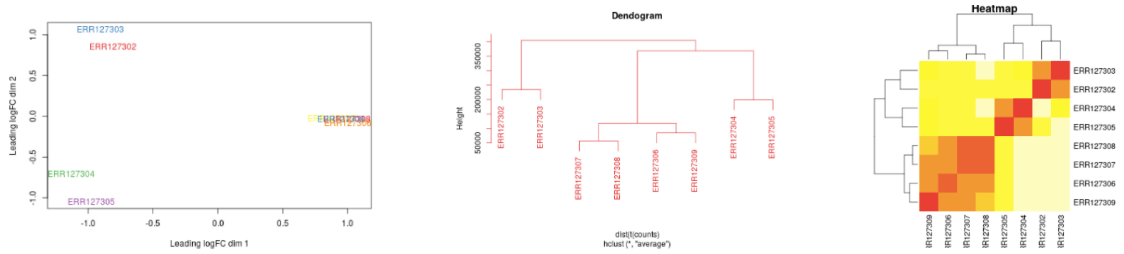


Fig.3.4.2.5.4. Gráficos Filtrado y Normalización

- Estimación de la dispersión GLM (Common, Trended y Tagwise). Los modelos lineales generalizados ó GLMs, son una extensión de los modelos lineales clásicos a datos distribuidos de forma no uniforme. Especifican distribuciones de probabilidad de acuerdo con su relación de media-varianza.

  Para experimentos generales con múltiples factores, edgeR utiliza el método de probabilidad ajustada (CR) para estimar las dispersiones. Esta opción realiza las 3 estimaciones indicadas: común, tendencia y tagwise, en cascada.

  La figura siguiente presenta cómo se muestra la salida de la estimación final, en formato de tabla de datos. Al igual que en la sección (test) anterior, también existen dos pestañas: datos y gráficos. En este caso, el gráfico es único y muestra el coeficiente biológico de variación, sin posibilidad de manipulación.

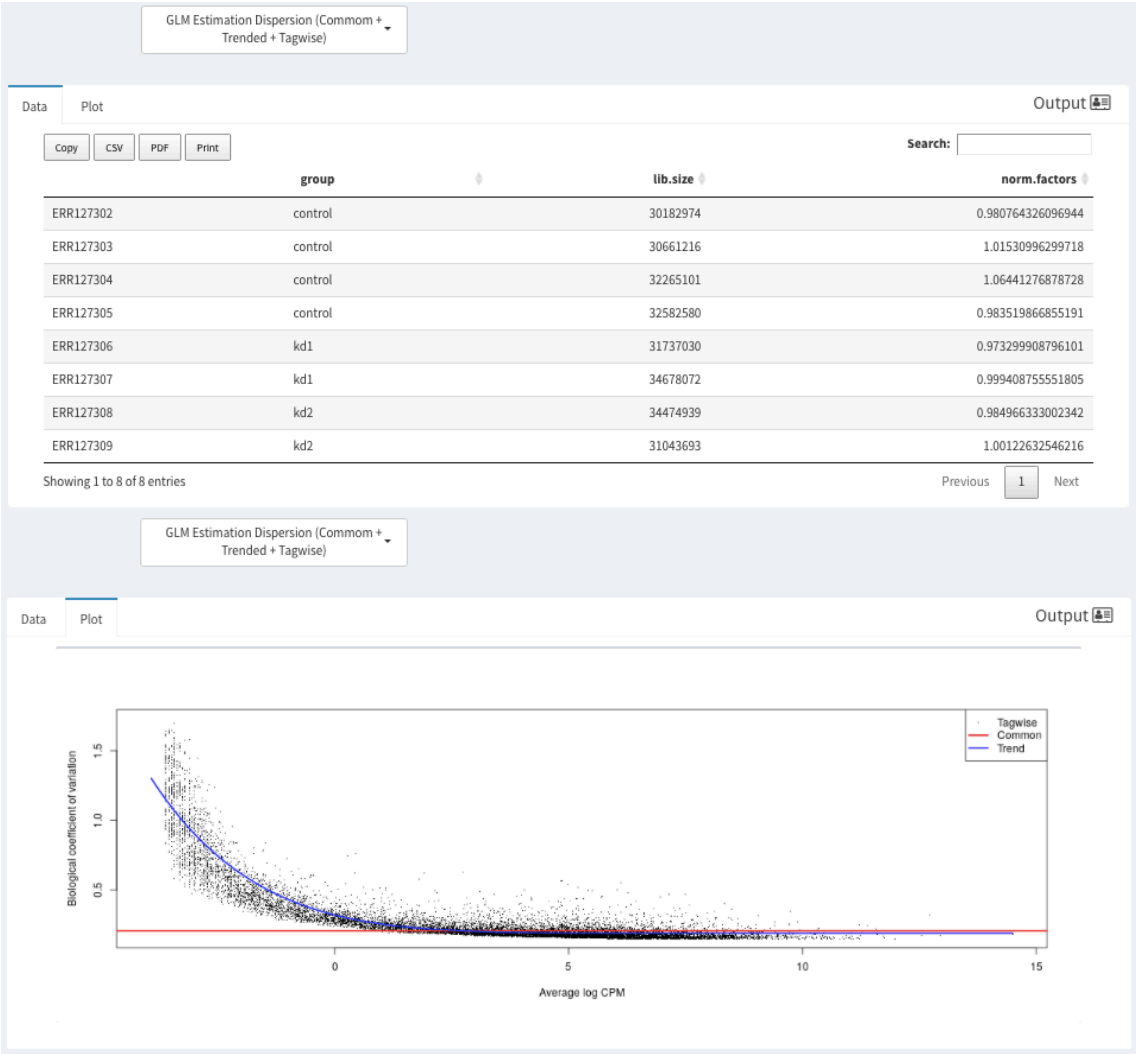


Fig.3.4.2.5.5. Estimación GLM - Datos y Gráfico

- Creación de matriz de diseño. Este proceso únicamente muestra una tabla con los datos de la matriz de diseño generada a partir de la tabla de contajes. La apariencia es la de la siguiente figura.

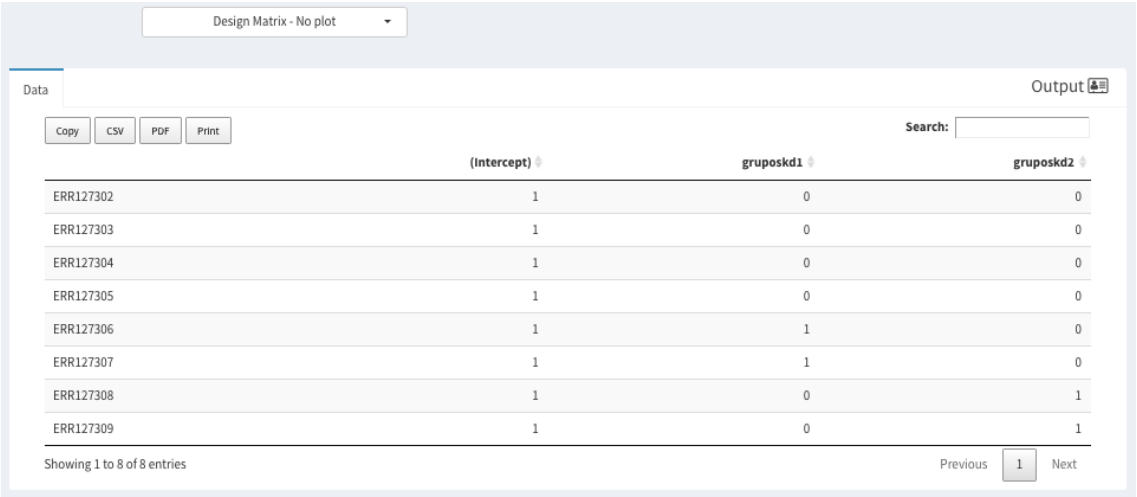


Fig.3.4.2.5.6. Matriz de Diseño

- Comparación de grupos GLM. Mediante este test se realizan comparaciones entre los distintos grupos definidos. Para que este test se pueda realizar, es necesario calcular la estimación de la dispersión GLM. Una vez se hace clic, se ejecutan tanto la normalización como la estimación de la dispersión, con independencia de si se ha ejecutado o no anteriormente, y un ajuste *glmFit* seguido de un *glmLRT*. Esto podría ser mejorado en una revisión futura (*TO-DO*).

  La pantalla, después de un momento de proceso, aparece con una configuración similar a la siguiente figura.

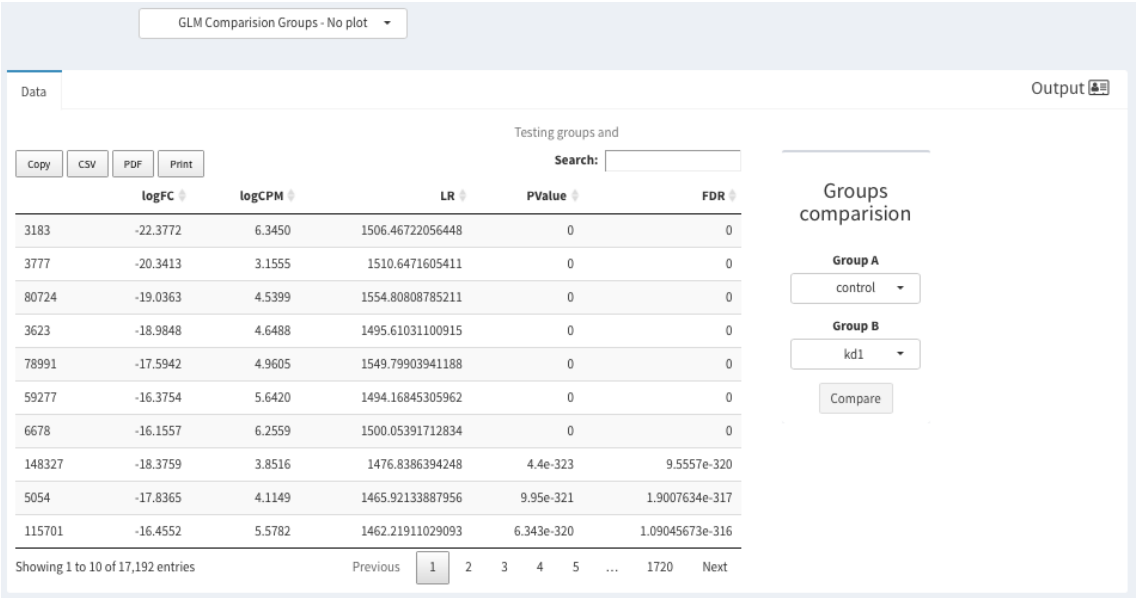


Fig.3.4.2.5.7. Comparación GLM de grupos

- Test de decisión. Finalmente, el test de decisión se genera a partir de todos los datos previos de la comparación de grupos, es decir, si se modifican los grupos, el cálculo se verá afectado.

  Para realizar los cálculos de las comparaciones entre grupos se empleó el método BH. Para calcular este valor también se usa el método BH (menos costoso computacionalmente) y un p-value 0.05, es decir, se localizan aquellos genes diferencialmente expresados al 5% FDR.

  La salida de este test muestra dos pestañas con tabla de datos y gráfico, como las que aparecen en las siguiente figuras.

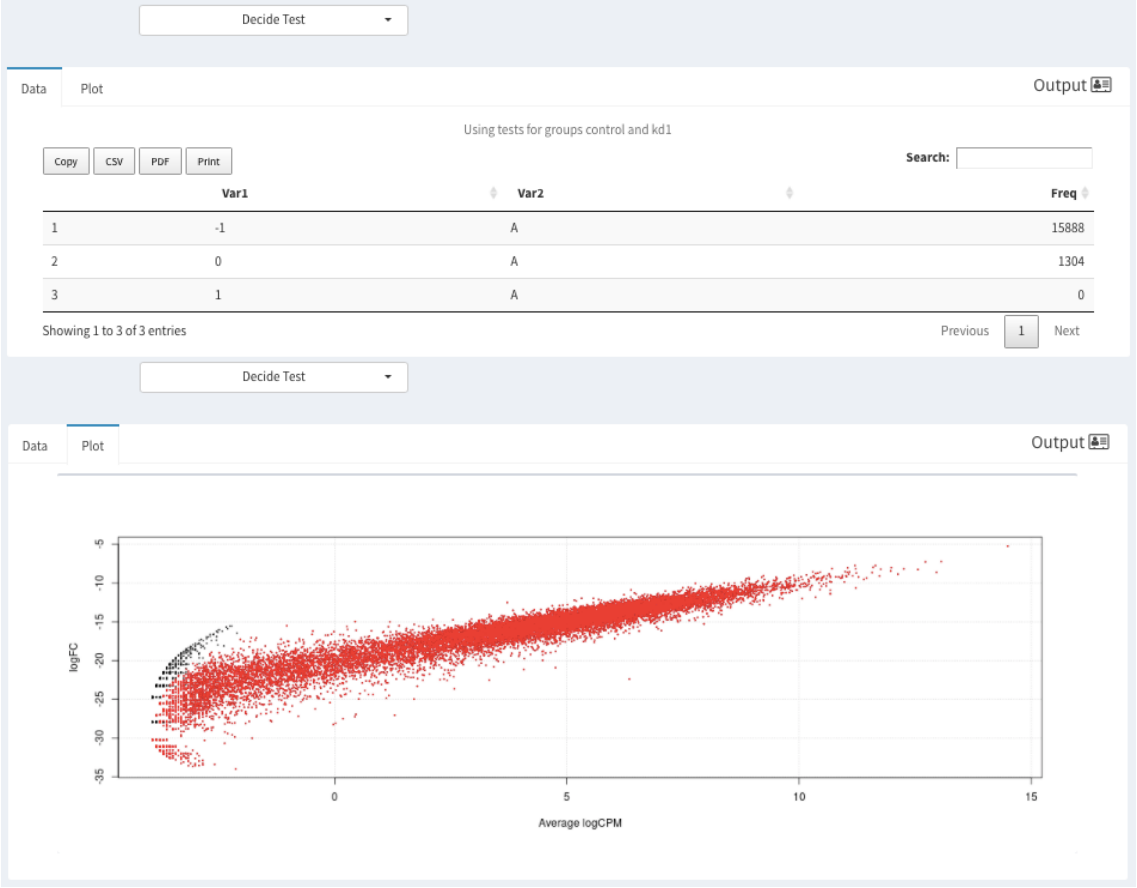


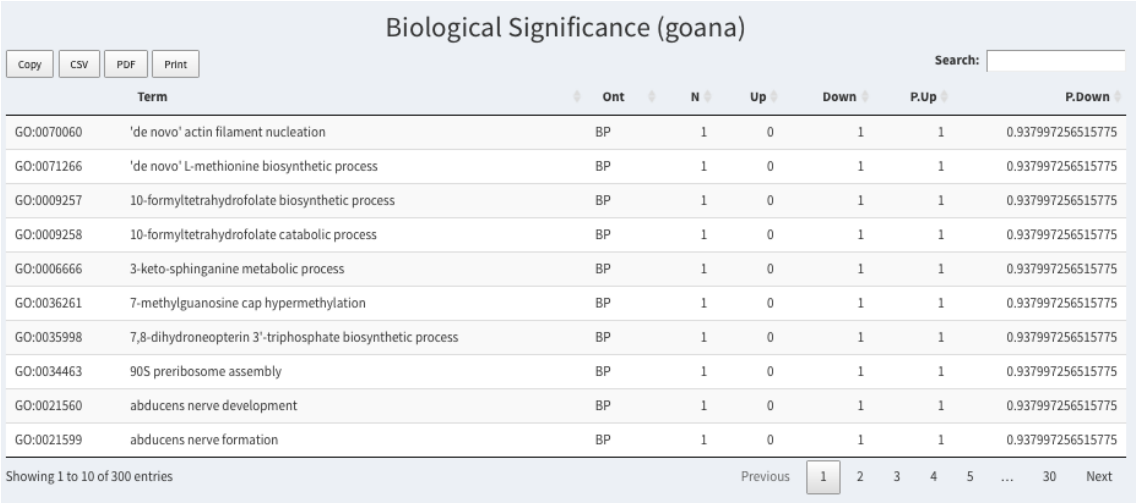
Fig.3.4.2.5.8. Test de decisión - Datos y Gráfico

3.4.2.6. Significación Biológica (goana).

Este análisis únicamente se ejecutará si se ha realizado el Test de Decisión en el análisis de Expresión Diferencial.

Este análisis toma los datos de entrada desde el Test de Decisión calculado en el análisis de Expresión Diferencial y ejecuta *goana* con el coeficiente *B.PregVsLac* para el organismo Homo Sapiens.

La salida se muestra en una tabla similar a la siguiente figura.


Biological Significance (goana)

Copy CSV PDF Print Search:

	Term	Ont	N	Up	Down	P.Up	P.Down
GO:0070060	'de novo' actin filament nucleation	BP	1	0	1	1	0.937997256515775
GO:0071266	'de novo' L-methionine biosynthetic process	BP	1	0	1	1	0.937997256515775
GO:0009257	10-formyltetrahydrofolate biosynthetic process	BP	1	0	1	1	0.937997256515775
GO:0009258	10-formyltetrahydrofolate catabolic process	BP	1	0	1	1	0.937997256515775
GO:0006666	3-keto-sphinganine metabolic process	BP	1	0	1	1	0.937997256515775
GO:0036261	7-methylguanosine cap hypermethylation	BP	1	0	1	1	0.937997256515775
GO:0035998	7,8-dihydroneopterin 3'-triphosphate biosynthetic process	BP	1	0	1	1	0.937997256515775
GO:0034463	90S preribosome assembly	BP	1	0	1	1	0.937997256515775
GO:0021560	abducens nerve development	BP	1	0	1	1	0.937997256515775
GO:0021599	abducens nerve formation	BP	1	0	1	1	0.937997256515775

Showing 1 to 10 of 300 entries Previous 1 2 3 4 5 … 30 Next

Fig.3.4.2.6. Significación Biológica

3.5. Resumen de la implementación realizada

Todo el código se implementa en el mismo servidor. *Shiny* se encarga de crear y ofrecer el interfaz de usuario, y de ejecutar directamente todas las peticiones de tareas o delegarlas en aplicaciones externas de las que más tarde lee los resultados.

A continuación se muestra un diagrama de relaciones entre los diferentes módulos desarrollados.

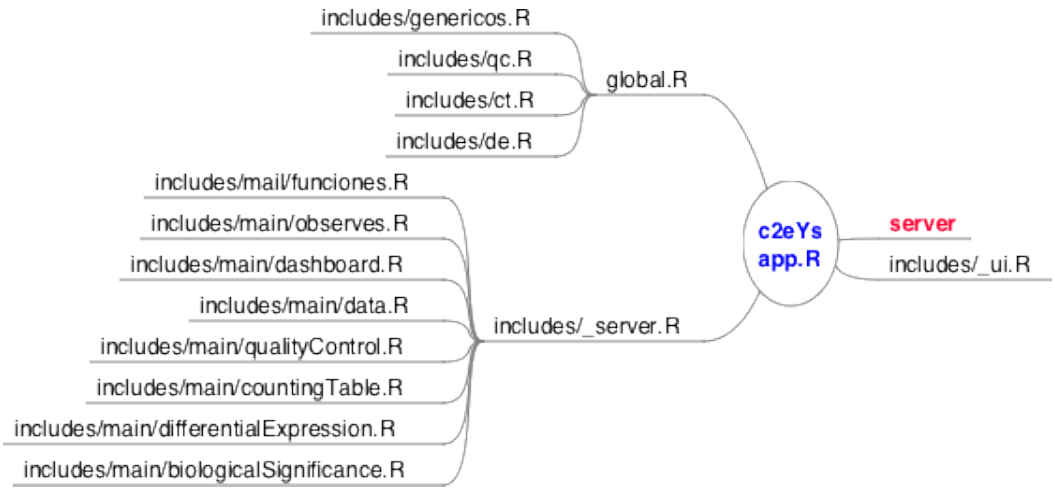


Fig.3.5.1. Implementación realizada

La aplicación se basa en el módulo principal `app.R`, que es desde donde se arranca el servidor (server) y se crea el interfaz de usuario, `_ui.R` que es publicado en el navegador web.

El fichero `global.R` incluye ficheros de código fuente que no utiliza objetos Shiny ni ShinyDashboard. El fichero `genericos.R` contiene los valores iniciales y carga de librerías requeridas por la aplicación. Por su parte los ficheros `qc.R`, `ct.R` y `de.R` contienen el código especializado para el control de calidad, tabla de contajes y expresión diferencial, respectivamente; todos ellos contienen variables y funciones que son invocados por funciones propias del servidor, `_server.R`.

El fichero `_server.R` es el dedicado a gestionar todas las tareas de la aplicación así como actualizar el interfaz de usuario. Contiene ficheros de funciones, `funciones.R`, observeEvents, `observes.R`, y gestores de cada una de las opciones del menú principal de la aplicación.

3.6. Test de Robustez del pipeline

Los tests se han realizado tanto en la capa de cliente como en la de servidor.

Cliente / Interfaz de usuario. Se detallan las pruebas realizadas para cada una de las opciones y controles en la capa cliente.

- El Dashboard, en función de la configuración definida, muestra correctamente la información según se va actualizando.

- Datos. Si se cambia la configuración de número de ficheros, aumentando o decreciendo, el comportamiento queda actualizado correctamente sin mostrar mensaje de error alguno.

- Control de calidad. Se han hecho pruebas para ver el comportamiento de este panel en función de los resultados FastQC. Se ha probado a cancelar los procesos en el servidor, mostrándose correctamente los estados en el Dashboard. Se utilizan ficheros con y sin el formato adecuado. Con el formato adecuado el interfaz no muestra errores; con otro formato, no se muestra el icono del resultado del análisis, pero no rompe el programa ni muestra errores en pantalla. En el caso de no tener datos, aparece el icono de FastQC.

- Tabla de contajes. Cada interacción con esta fase, queda reflejada en el Dashboard. Se comprueba que si se envía un fichero con el formato inadecuado, el flujo de la aplicación no se rompe.

- Expresión diferencial. Comprobado que muestra correctamente el mensaje. Se analizan y prueban diversas situaciones en cuanto al cambio de la tabla de contajes. Se corrige un problema en la detección de genes por no realizar una búsqueda correcta por la clave adecuada. Se decide que independientemente de la tabla de contajes utilizada, la aplicación intente detectar la clave de búsqueda; se comprueba que tiene un funcionamiento correcto. Se comprueba que no se puede acceder a realizar una análisis de este tipo si antes no se ha seleccionado una tabla de contajes.

- Significación biológica. Comprobado que muestra correctamente el mensaje. Después de muchas pruebas, se mejora la información que aparece, cuando y cómo. Se detecta un error para la ejecución de la significación biológica, cuando se utiliza la tabla de contajes creada desde la aplicación. La búsqueda con *goana* no devuelve datos. Se corrige el error en la aplicación y se muestra un mensaje “controlado” con el mensaje de *goana*. Se comprueba que el acceso a este análisis se realiza únicamente si se ha hecho el análisis de expresión diferencial (al menos, la comparación de grupos GLM).

- Las pruebas de carga de ficheros en el servidor también pueden afectar al cliente. En este caso, sin tener en cuenta el tiempo de proceso invertido en cada una de las cargas de los ficheros, el interfaz cliente se ha comportado de manera estable.

- Servidor. Se detallan las operaciones realizadas en la capa de servidor.

Se han hecho pruebas de carga con ficheros de hasta 7 GB, ejecutándose para una sesión de *Tophat*, limitada al uso de 1 y 2 procesadores; el sistema se ha comportado bastante bien aunque ha tenido unos picos bastante altos de entrada / salida. No se trata de un servidor dedicado para estas tareas por lo que no se activan todos los procesadores. La ejecución de *FastQC* sobre estos ficheros tampoco ha generado problemas de rendimiento en el servidor aunque tanto este como *Tophat* han tenido un tiempo de proceso bastante alto.

Al final de las pruebas, para evitar posibles problemas de rendimiento, el tamaño máximo de carga de fichero se ha dejado en el valor por defecto de Shiny Server, es decir 5MB.

Durante el tiempo de desarrollo y pruebas se ha hecho un seguimiento de los ficheros de logs generados por el servidor Shiny, buscando algún error; cuando se han localizado, se han corregido.

3.7. Instalación de la aplicación.

El software base mínimo que debe instalarse para poder ejecutar la aplicación con garantías, es el siguiente:

- *RStudio Server* 1.0.136
- *Shiny Server* 1.5.1.834
- *Java SE Runtime Environment* 1.8.0_121. Necesario para la ejecución de *FastQC*.

La aplicación ha sido validada en un servidor con *Linux Ubuntu 17.04* por lo que se considera que es el sistema operativo para ejecutar la aplicación. Se ejecutan órdenes del sistema (por ejemplo, `ps -ef)` que no son compatibles con otros sistemas operativos.

Además, para poder ejecutar la generación de la Tabla de Contajes, es necesario tener instalada la aplicaciones Tophat. Desde Ubuntu, se puede instalar con la siguiente orden:

```
apt-get install tophat
```

Una vez configurados los programas anteriores, incluyendo las librerías R especificadas en el punto 2.8 de esta memoria, desde el directorio `/srv/shiny-server` (o el que se haya configurado para servir las aplicaciones), se debe desempaquetar el software de c2eYs. En un sistema Linux / Unix, sería:

```
cd /srv/shiny-server
tar xvfz c2eYs.tgz
```

Esto creará la lista de ficheros incluidos en el anexo, así como una serie de directorios.

Para concluir la instalación, en el directorio `/srv/shiny-server`, se debe incluir otros dos directorios:

`Bowtie2Index` —> el directorio de índices para bowtie2, que debe estar compuesto por el contenido del directorio `/Homo_sapiens/NCBI/GRCh38/Sequence/Bowtie2Index` que es parte del Genoma Humano.

`gtf` —> el directorio donde están las anotaciones de genes, que debe estar compuesto por el contenido del directorio `/Homo_sapiens/NCBI/GRCh38/Annotation/Genes` que también es parte del Genoma Humano.

La descarga del genoma se puede realizar con la siguiente orden:

```
wget ftp://igenome:G3nom3s4u@ussd-ftp.illumina.com/Homo_sapiens/
NCBI/GRCh38/Homo_sapiens_NCBI_GRCh38.tar.gz
```

4. Conclusiones

¿ Qué se ha aprendido del trabajo ?

En el ámbito de la Bioinformática, existe una gran cantidad de software especializado de aplicación, con una alta complejidad en su utilización que va en sintonía con la dificultad de resolución de los problemas biológicos objetivo de estas aplicaciones informáticas. El uso de esas herramientas no queda bien delimitado y muchas de ellas son como “navajas suizas”. Eso genera un gran inconveniente a la hora de afrontar un problema, ya que hay utilidades software que “bajo el papel” pretenden hacer lo mismo, aunque con grandes matices ya que se utilizan métodos distintos para supuestamente realizar análisis similares. Por otra parte, la interpretación de los resultados visuales tiene un peso aún grande.

¿ Se han alcanzado los objetivos ?

Si como alcanzar los objetivos se entiende cubrir aquellos propuestos por el Plan de Trabajo, la respuesta es sí, aunque con una aclaración. Si bien se ha realizado el trabajo propuesto, el desarrollo final creo que se podría haber implementado de otra manera. No cabe duda que R junto a *RStudio* forman un tándem realmente potente, pero desde mi punto de vista, creo que *Shiny* y *Shiny Dashboard* han complicado este desarrollo; cuando tenía una idea, debía desecharla porque el *framework* no lo permitía, eso sí, después de realizar una alta inversión en tiempo para validarlo. Entiendo que para un usuario novel, trabajar con un framework es una avance, pero a un programador le limita la libertad de programación.

Seguimiento de la planificación y metodología

A lo largo del desarrollo de este TFM me he intentado ceñir a la temporización y metodología marcada, pero de manera clara únicamente lo he conseguido con la metodología. Haciendo computo de recursos invertidos en el desarrollo de este TFM, a pesar de haber tenido que recuperar retrasos generados por elementos externos, he excedido con creces los tiempos marcados. Me hubiese gustado tener más tiempo o menor carga de trabajo, pero quizás el planteamiento debió ser menos exigente. Respecto a los cambios introducidos, sí que ha habido y han ido desde cambios entre ordenadores (con más recursos y delegando procesos) para evitar retrasos, a cambios en los métodos seleccionados en primera instancia por no poderlos implementar debido a incompatibilidades. Al mismo tiempo que realizaba el trabajo, recibía un aprendizaje sobre la problemática del procesamiento de grandes volúmenes de datos en ordenadores con pocos recursos.

Líneas de trabajo futuro. TO-DO.

El primer replanteamiento, ya avanzado, sería cambiar de entorno de programación a un lenguaje mas flexible aunque sí en entorno web, lenguaje *PHP* con un servidor *NGinx*, por ejemplo. He hecho pruebas de rendimiento con *Rscript* y tanto los gráficos como las salidas de datos tienen unos tiempos muy buenos.

Como mejoras tempranas a realizar:

- En el análisis de expresión diferencial, en la versión actual, los tests se recalculan. Una mejora consistiría en guardar esos cálculos para posteriores necesidades y evitar consumo innecesario de *CPU* y memoria.

- En el análisis de expresión diferencial, en la comparativa de grupos, mejorar la aparición de los controles de selección de grupos y *DT* ya que a veces aparecen intercalados.

- Incorporación de botón de cancelar proceso, para los procesos que tardan en demasía. Implementable en el *Dashboard*, por ejemplo.

- Realizar avisos por mail cuando una tarea “pesada” haya concluido.

- Mejoras en la aplicación de la reactividad *Shiny*, entre las distintas opciones.

- Optimización del análisis de significación biológica.

5. Glosario

Accesibilidad. Cualidad de poder acceder o facilidad de hacerlo.
Actor. Rol de un usuario o sistema cuando interacciona con otro/s.
Ajuste. Ponderación, redefinición de variables y transformación de escalas.
Algoritmo. Conjunto ordenado de operaciones sistemáticas.
Alinear. Representar y comparar dos o más secuencias para resaltar zonas de similitud.
Anotación. Descripción de la ubicación de elementos con una función potencial.
ARNm. ARN con la función de mensaje de información genética.
ASCII. Sistema de codificación de caracteres.
Autenticación. acceso mediante credenciales a un servicio.
Background. Aplicado a la ejecución de un proceso en segundo plano.
Binomial. Aplicado a una distribución de Bernoulli.
Bit. Unidad mínima de información 1 ó 0.
Boxplot. Diagrama de caja.
Colaborativo. Aplicado a procesos que se realizan en un grupo.
Credenciales. Autorización normalmente en base a un par login-password..
CSS. Hojas de estilos aplicable en páginas HTML.
Dataset. Conjunto de datos.
Dispersión. Grado de distanciamiento de una valores respecto a su valor medio.
Distribución. Distribución de probabilidad.
Enriquecimiento. Sobrerrepresentación de anotaciones.
Estadístico. Medida cuantitativa derivada de un conjuntos de datos de una muestra.
Estandarización. Proceso de estandarizar, a un valor se le resta su media aritmética y el resultado se divide por su desviación estándar.
Estimación. Búsqueda de un valor aproximado de una parámetro de una población a partir de parámetros de una muestra.
FDR. Acrónimo de False Discovery Rate.
Heatmap. Mapa de calor de representación de datos, más intensidad mayor representación.
Interfaz. Área de comunicación entre sistemas, normalmente con el usuario.
Isoforma. Una de las distintas formas de una proteína.
Learning (machine). Aprendizaje automático.
MB. Megabyte, 1024 Kilobyes.
MDS. Acrónimo de MultiDimensional Scaling.
Media. Valor característico de un conjunto de datos.
Mediana. Valor de la variable de la posición central de un conjunto de datos ordenados.
NCBI. Acrónimo de National Center for Biotechnology Information.
Nginx. Servidor web ligero de alto rendimiento.
NGS. Acrónimo de Next Generation Sequencing.
Ómica. Neologismo utilizado en biología molecular para referirse a los genes, organismos, proteínas, ...

Ontología. Estudio de las propiedades de un ser.
Percentil. Valor del elemento que divide una serie de datos en cien grupos de igual valor o intervalo.
PHP. Lenguaje de programación de código abierto ampliamente utilizado en aplicaciones web.
Pipeline. Conjunto de técnicas ejecutadas en secuencia.
Pseudocódigo. Descripción de alto nivel del funcionamiento de un algoritmo.
P-value. Probabilidad de cometer una error de tipo I (falso positivo) en un análisis estadístico.
Recursividad. Capacidad de un algortimo de llamarse a sí mismo.
Repositorio. Contenedor.
Rscript. Programa que ejecuta scripts R.
SATA. Acrónimo se Serial Advanced Technology Attachment.
Secuenciación. Respecto al ADN, conjunto de métodos y técnicas bioquímicas para determinar el orden de los nucleótidos (ACGT).
Secuenciador. Equipo que secuencia.
Secuencia. Respecto a genética, sucesión de letras ACGT en una banda de ADN.
Sentencia. Respecto a la informática, elemento básico en que se divide el código de programación.
Sesgo. Influencia directa o indirecta, en los resultados de un proceso estadístico.
Sesión. Entorno de conexión de una usuario en un servidor, con sus variables y valores.
SO. Acrónimo de Sistema Operativo.
SSD. Acrónimo de Solid State Drive, aplicado a dispositivos de almacenamiento.
Standalone. De un proceso cuando se puede ejecutar de manera independiente.
Timeouts. Espera máxima a la aparición de un evento.
TO-DO. Del inglés, "para hacer".
Toolkit. Conjunto de herramientas.
Usabilidad. Cualidad de un programa informática para ser sencillo e intuitivo de usar.

6. Bibliografía

[1] *Johana Carolina Soto Sedano y Camilo Ernesto López Carrascal*
RNA-seq: herramienta transcriptómica útil para el estudio de interacciones planta-patógeno
Fitosanidad, 101-113, 16 / 2, 2012
http://www.redalyc.org/pdf/2091/209126216009.pdf

[2] Babraham Bioinformatics - 22 mayo 2017
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

[3] Illumina - TruSight Tumor - 22 mayo 2017
https://support.illumina.com/help/BS_App_TruSightTumor15_OLH_1000000001133/Content/Source/Informatics/BAM-Format.htm

[4] Sequence Alignment/MAP Format Specification - 22 mayo 2017
https://samtools.github.io/hts-specs/SAMv1.pdf

[5] Illumina Support CASAVA - 22 mayo 2017
https://support.illumina.com/help/SequencingAnalysisWorkflow/Content/Vault/Informatics/Sequencing_Analysis/CASAVA/swSEQ_mCA_FASTQFiles.htm

[6] W3C - 22 mayo 2017
https://www.w3.org/html/

[7] The Ohio University Comprehensive Cancer Center - 22 mayo 2017
http://bioserv.mps.ohio-state.edu/QuaCRS/

[8] CGA - 22 mayo 2017
http://archive.broadinstitute.org/cancer/cga/rna-seqc

[9] RNA-SeQC: RNA-seq metrics for quality control and process optimization - 22 mayo 2017
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3356847/

[10] RSeQC - 22 mayo 2017
http://rseqc.sourceforge.net

[11] RSeQC: quality control of RNA-seq experiments - 22 mayo 2017
https://www.ncbi.nlm.nih.gov/pubmed/22743226

[12] NGC QC Toolkit - National Institue of Plant Genome Research - 22 mayo 2017
http://www.nipgr.res.in/ngsqctoolkit.html

[13] *Ravi K. Patel and Mukesh Jain*
NGS QC Toolkit: A Toolkit for Quality Control of Next Generation Sequencing Data
PLoS One, e30619, 7(2), 2012
https://dx.doi.org/10.1371%2Fjournal.pone.0030619

[14] Zhang Lab - What is FASTA - 22 mayo 2017
http://zhanglab.ccmb.med.umich.edu/FASTA/

[15] GNU Gzip - 22 mayo 2017
https://www.gnu.org/software/gzip/

[16] RNA seq data analysis - Tophat, HTseq and DESeq2 analysis - 22 mayo 2017
https://digibio.blogspot.com.es/2015/08/rna-seq-data-analysis-tophat-htseq-and.html

[17] Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks de Cole Trapnell et al, publicado el 1 de marzo de 2012 - 22 mayo de 2017
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3334321/

[18] RNA-seq: mapping to a reference genome with tophat and counting with HT-seq - 22 mayo 2017
http://angus.readthedocs.io/en/latest/drosophila_rnaseq1.html

[19] *Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Ring & Liar Pachter*
Differential gene and transcript expression analysis of RNA-Seq experiments with Tophat and Cufflinks
Nature Protocols, 562-578, 7, 2012
http://cole-trapnell-lab.github.io/pdfs/papers/trapnell-tuxedo-protocol.pdf

[20] Tophat - 22 mayo 2017
https://ccb.jhu.edu/software/tophat/index.shtml

[21] Seqtk - 22 mayo 2017
https://github.com/lh3/seqtk

[22] SAMtools - 22 mayo 2017
http://samtools.sourceforge.net

[23] Ensembl - GFF/GTF File Format - 22 mayo 2017
http://www.ensembl.org/info/website/upload/gff.html

[24] Cufflinks - 22 mayo 2017
http://cole-trapnell-lab.github.io/cufflinks/

[25] HTSeq - 22 mayo 2017
http://www-huber.embl.de/HTSeq/doc/overview.html

[26] Python - 22 mayo 2017
https://www.python.org

[27] RPKM, FPKM and TPM, clearly explained - 22 mayo 2017
http://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/

[28] DESeq - R Documentation - 22 mayo 2017
https://www.rdocumentation.org/packages/DESeq/versions/1.24.0/topics/DESeq

[29] Bioconductor - DESeq - 22 mayo 2017
http://bioconductor.org/packages/release/bioc/html/DESeq.html

[30] *Cole Trapnell, David G Hendrickson, Martin Sauvageau, Loyal Goff, John L Rinn y Lior Pachter*
Differential analysis of gene regulation at transcript resolution with RNA-seq
Nature Biotechnology, 46-53, 31, 2013
http://cole-trapnell-lab.github.io/papers/trapnell-cuffdiff2/

[31] *Zong Hong Zhang, Dhanisha J. Jhaveri, Vikki M. Marshall, Denis C. Bauer, Janette Edson, Ramesh K. Nrayanan, Gregory J. Robinson, Andreas E. Lundberg, Perry F. Barlett, Naomi R. Wray, Qionj-Yi Zhao*
A Comparative Study of Techniques for Differential Expression Analysis on RNA-Seq Data
PLoS One, e103207, 9(8), 2014
http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0103207

[32] *Schurch NJ, Schofield P, Glerlinski M, Cole C, Sherstnev A, Singh V, Wrobel N, Hgarbi K, Simpson GG, Owen-Hughes T, Blaxter M, Barton GJ*
How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use?
RNA, 839-51, 22(6), 2016
https://www.ncbi.nlm.nih.gov/pubmed/27022035

[33] edgeR: a Bioconductor package for differential expression analysis of digital gene expression data - 22 mayo 2017
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/

[34] edgeR: differential expression analysis of digital gene expression data - 22 mayo 2017
https://www.bioconductor.org/packages/devel/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf

[35] GOC - 22 mayo 2017
http://www.geneontology.org

[36] goana - 22 mayo 2017
https://www.rdocumentation.org/packages/edgeR/versions/3.14.0/topics/goana.DGELRT

[37] Entrez Gene: gene-centered information at NCBI - 22 mayo 2017
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3013746/

[38] Calculation Methods for Wallenius' Noncentral - 22 mayo 2017
http://www.agner.org/random/nchyp1.pdf

[39] Gene Ontology analysis for RNA-seq: accounting for selection bias - 22 mayo 2017
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2872874/

[40] Gene set enrichment analysis with topGO - 22 mayo 2017
https://www.bioconductor.org/packages/devel/bioc/vignettes/topGO/inst/doc/topGO.pdf

[41] RStudio Server - 22 mayo 2017
https://www.rstudio.com/products/rstudio/download-server/

[42] Shiny Server - 22 mayo 2017
https://www.rstudio.com/products/shiny/download-server/

[43] Ubuntu 17.04 64 bits - 22 mayo 2017
http://releases.ubuntu.com/17.04/

[44] Github - 22 mayo 2017
https://github.com

[45] shinyapps.io - 22 mayo 2017
https://www.shinyapps.io

[46] R - 22 mayo 2017
https://www.r-project.org/

[47] RStudio - 22 mayo 2017
https://www.rstudio.com/

[48] Shiny - 22 mayo 2017
https://shiny.rstudio.com/

[49] Shiny Dashboard - 22 mayo 2017
https://rstudio.github.io/shinydashboard/

[50] DT: An R interface to the DataTables library - 22 mayo 2017
http://rstudio.github.io/DT/

[51] RColorBrewer - 22 mayo 2017
https://cran.r-project.org/web/packages/RColorBrewer/index.html

[52] gageData - 22 mayo 2017
https://bioconductor.org/packages/release/data/experiment/html/gageData.html

[53] org.Hs.eg.db - 22 mayo 2017
https://bioconductor.org/packages/release/data/annotation/html/org.Hs.eg.db.html

[54] Rsubread - 22 mayo 2017
http://bioconductor.org/packages/release/bioc/vignettes/Rsubread/inst/doc/Rsubread.pdf

[55] Bash - 22 mayo 2017
https://www.gnu.org/software/bash/

[56] Navegador web - 22 mayo 2017
https://es.wikipedia.org/wiki/Navegador_web

[57] Protocolo HTTP - 22 mayo 2017
https://www.w3.org/Protocols/rfc2616/rfc2616.html

[58] Redes LAN / WAN / MAN y otras redes - 22 mayo 2017
http://www.uv.es/uvweb/master-ingenieria-telecomunicacion/es/blog/lan-wan-man-otras-redes-1285954593702/GasetaRecerca.html?id=1285959494096

7. Anexos

7.1. Scripts R.

app.R

```
# Suprimir warnings
#options(warn = -1)
options(warn = 0)

# Servidor monousuario

# Global
source("global.R", local = TRUE)

# UI
source("includes/_ui.R", local = TRUE)

# Server
source("includes/_server.R", local = TRUE)

# Main
shinyApp(ui, server)
```

global.R

```
# Elementos estaticos
source("includes/genericos.R", local = TRUE)

# Control de calidad
source("includes/qc.R", local = TRUE)

# Tabla de contajes
source("includes/ct.R", local = TRUE)

# Expresion diferencial
source("includes/de.R", local = TRUE)
```

crearCT.R

```
# Crear la tabla de contajes a partir de la informacion de
Tophat / Cufflinks

library(Rsubread)
setwd("/srv/shiny-server/c2eYs.new/www/results/ct")
ficheros <- list.files(pattern="accepted_hits.bam$",
                       recursive = TRUE)

fc <- featureCounts(files =
list.files(pattern="accepted_hits.bam$",
                                       recursive = TRUE),
                    annot.inbuilt="hg38")

# Renombar columnas
colnames(fc$counts) <- c(paste0("S", seq(1, ncol(fc$counts))))

# Anotaciones
sample.table <- fc$counts
gtf.file <- file.path("/srv/shiny-server/c2eYs.new/gtf",
"genes.gtf")

library(Rsamtools)
bam.list <- BamFileList(ficheros)

library(GenomicFeatures)
txdb <- makeTxDbFromGFF(gtf.file, format="gtf")

exons.by.gene <- exonsBy(txdb, by="gene")
library(GenomicAlignments)
se <- summarizeOverlaps(exons.by.gene, bam.list,
                        mode="Union",
                        singleEnd=FALSE,
                        ignore.strand=TRUE,
                        fragments=TRUE)
fc <- featureCounts(bam.files, annot.ext=gtf.file,
                    isGTFAnnotationFile=TRUE,
                    isPaired=TRUE)
write.table(fc$counts, "tabla_contajes.txt")
write("OK", "tabla.info")
```

includes/_ui.R

```
# UI
ui <- dashboardPage(
  skin = "green",
  dashboardHeader(title = "c2eYs for H. Sapiens",
                  titleWidth = "300px"),
  dashboardSidebar(
    width="300px",
    sidebarMenu(
      id = "sidebarmenu",
      menuItem("Dashboard",
               tabName = "db",
               icon = icon("dashboard")),
      menuItem("Data",
               tabName = "dt",
               icon = icon("database")),
      menuItem("Quality Control",
               tabName = "qc",
               icon = icon("quora")),
      menuItem("Counting Table",
               tabName = "ct",
               icon = icon("th")),
      menuItem("Differential Expression",
               tabName = "de",
               icon = icon("vcard-o")),
      menuItem("Biological Significance",
               tabName = "bs",
               icon = icon("eyedropper")),
      conditionalPanel("input.sidebarmenu === 'dt'",
                       sliderInput("nf", "Number of samples:",
1, tf, nf),
                       column(12,
                              align="center",
                              h4("** ADVICE **"),
                              "If you change samples values,",
                              br(),
                              "analysis results will be erased"
                       ))
    )
  ),
  dashboardBody(
    useShinyjs(),
    extendShinyjs(text = jsCode),
    tags$head(
      tags$link(rel = "stylesheet",
                type = "text/css",
                href = "c2eYs.css"),
      tags$script(type="text/javascript",
                  src = "c2eYs.js")),
    tabItems(
      tabItem(tabName = "db",
              fluidRow(
                column(12,
                       align = "center",
                       h2("Dashboard RNASeq Pipeline"),
                   h4("This page shown status for all phases in
pipeline"),
```

```
                uiOutput("ui_db")))),

tabItem(tabName = "qc",
        fluidRow(
          column(12,
                 align = "center",
                 h2("Quality Control (FastQC)")),
          column(12,
                 align = "center",
                 uiOutput("ui_qc")),
          column(12,
                 align = "center",
                 uiOutput("bt_qc")),
          column(12,
                 align = "center",
                 uiOutput("ms_qc")),
          column(12,
                 align = "center",
                 uiOutput("ui_results_qc")))),

tabItem(tabName = "ct",
        fluidRow(
          column(12,
                 align = "center",
                 h2("Counting Table (TopHat)")),
          column(12,
                 align = "center",
                 uiOutput("ui_ct")),
          column(12,
                 align = "center",
                 uiOutput("ms_ct")),
          column(12,
                 align = "center",
                 hidden(
                   div(id = "wt_ct",
                       p("Processing ..."),
                       img(src="cargando.gif")))),
          column(12,
                 align = "center",
                 uiOutput("ui_results_ct")))),

tabItem(tabName = "de",
        fluidRow(
          column(12,
                 align = "center",
                 h2("Differential Expression (edgeR)")),
          column(12,
                 align = "center",
                 uiOutput("ui_de")),
          column(6,
                 align = "center",
                 uiOutput("bt_de")),
          column(6,
                 align = "center",
                 hidden(
                   div(id = "wt_de",
                       p("Processing ..."),
                       img(src="cargando.gif")))),
```

```
                column(12,
                       align = "center",
                       uiOutput("sb_de")),
                column(12,
                       align = "center",
                       uiOutput("ui_results_de")))),

      tabItem(tabName = "bs",
              fluidRow(
                column(12,
                       align="center",
                       h2("Biological Significance (goana)")),
                column(12,
                       align = "center",
                       hidden(
                         div(id = "wt_bs",
                             p("Processing ..."),
                             img(src="cargando.gif")))),
                column(12,
                       align="center",
                       uiOutput("ui_bs")))),

      tabItem(tabName = "dt",
              fluidRow(
                column(12,
                       align="center",
                       h2("Data / Samples for analysis")),
                column(12,
                       align="center",
                            helpText("You must select and upload
samples for analysis"),
                             helpText("(required file format for
sample files is fastq)"),
                       uiOutput("ui_dt"))))
    )
  )
)
```

includes/_server.R

```
# Servidor
server <- function(input, output, session) {

  # Funciones
  source("includes/main/funciones.R", local = TRUE)

  # ObserveEvents
  source("includes/main/observes.R", local = TRUE)

  # Dashboard
  source("includes/main/dashboard.R", local = TRUE)

  # Quality Control
  source("includes/main/qualityControl.R", local = TRUE)

  # Counting Table
  source("includes/main/countingTable.R", local = TRUE)

  # Differential Expression
  source("includes/main/differentialExpression.R", local = TRUE)

  # Biological Significance
  source("includes/main/biologicalSignificance.R", local = TRUE)

  # Data
  source("includes/main/data.R", local = TRUE)

  # Reactivar cada 5 segundos
  autoInvalidate <- reactiveTimer(5000)
  autoInvalidate1 <- reactiveTimer(1000)

  # Reconexion en sesion
  session$allowReconnect(TRUE)

}
```

includes/genericos.R

```
# Elementos genericos a c2eYs

# Eliminar todos los objetos de memoria
rm(list = ls())

# Paquetes Bioconductor

#if (!(require("edgeR", character.only = T))) {
#  source("http://bioconductor.org/biocLite.R")
#  biocLite("edgeR")
#}

#if (!(require("gageData", character.only = T))) {
#  source("http://bioconductor.org/biocLite.R")
#  biocLite("gageData")
#}

#if (!(require("gage", character.only = T))) {
#  source("http://bioconductor.org/biocLite.R")
#  biocLite("gage")
#}

#if (!(require("org.Hs.eg.db", character.only = T))) {
#  source("http://bioconductor.org/biocLite.R")
#  biocLite("org.Hs.eg.db")
#}

# Otros paquetes

#if (!(require("png", character.only = T)))
install.packages("png")
#if (!(require("DT", character.only = T)))
install.packages("DT")
#if (!(require("RColorBrewer", character.only = T)))
install.packages("RColorBrewer")
#if (!(require("shiny", character.only = T)))
install.packages("shiny")
#if (!(require("shinyjs", character.only = T)))
install.packages("shinyjs")
#if (!(require("shinyBS", character.only = T)))
install.packages("shinyBS")
#if (!(require("shinydashboard", character.only = T)))
install.packages("shinydashboard")

# Libraries
library(edgeR)
library(gageData)
#library(gage)
library(org.Hs.eg.db)
#library(png)
library(DT)
library(RColorBrewer)

#library(shinyBS)
library(shinyjs)
library(shinydashboard)
```

```
library(shiny)

library(Rsubread)
library(Rsamtools)
library(GenomicFeatures)
library(GenomicAlignments)

# Options Shiny (hasta 7 GB)
options(shiny.maxRequestSize = 7*1024^3)
options(shiny.reactlog = FALSE)
#options(shiny.port = 3333)
options(shiny.trace = FALSE)
options(digits = 3)
#options(error = browser())

# Codigo Javascript
jsCode <- "
shinyjs.tabqc = function(p) {

  var b = {
    n: 0,
    i: 0
  };

  q = shinyjs.getParams(p, b);
  var d = document.getElementById('qctab');
  if (d) {
    var a = d.getElementsByTagName('a');
      var c = '<div> <img src=\"icons/' + b.i + '.png\"/> ' +
a[b.n - 1].text.trim() + ' </div>';
    a[b.n - 1].setAttribute('data-value', c);
    a[b.n - 1].innerHTML =  c;
  }
}
"

# Set de colores a utilizar
brewer.cols <- brewer.pal(6, "Set1")

# Carga inicial de la tabla de contajes demo
# para evitar el tiempo en la seleccion
data("hnrnp.cnts")

# Directorio de uploads de ficheros
uploads <- "./www/uploads/"

# Maximo numero de samples a importar
tf <- 16

# Leer numero de samples seleccionados,
# si lo hubiera
nf <- 2
if (file.exists(paste0(uploads, "n.info"))) {
  nf <- readLines(paste0(uploads, "n.info"))
} else write(nf, paste0(uploads, "n.info"))

# Leer nombres de ficheros ya cargados
uf <- matrix(cbind(""), # Nombre de los ficheros cargados
```

```
              nrow = 1,
              ncol = tf,
              byrow = TRUE)
for (i in 1:nf) {
  if (file.exists(paste0(uploads, i, ".info")))
    uf[i] <- readLines(paste0(uploads, i, ".info"))
}

# Especie
especie <- "Homo_sapiens.GRCh38.88.gff3"
```

includes/qc.R

```
# Funciones genericas para Control de calidad

# Modulos y graficos estadisticos
qcModulos <- c("Basic Statistics",
               "Per base sequence quality",
               "Per tile sequence quality",
               "Per sequence quality scores",
               "Per base sequence content",
               "Per sequence GC content",
               "Per base N content",
               "Sequence Length Distribution",
               "Sequence Duplication Levels",
               "Overrepresented sequences",
               "Adapter Content",
               "Kmer Content")

qcGraficas <- c(NA,
                "per_base_quality.png",
                "per_tile_quality.png",
                "per_sequence_quality.png",
                "per_base_sequence_content.png",
                "per_sequence_gc_content.png",
                "per_base_n_content.png",
                "sequence_length_distribution.png",
                "duplication_levels.png",
                NA,
                "adapter_content.png",
                "kmer_profiles.png")

# Variables globales
qcProgram <- "/srv/shiny-server/c2eYs/FastQC/fastqc"
qcResultados <- "./www/results/qc/"
qcImagenes <- "results/qc/"
qcExe <- "./qc.bash"

# Otras variables
cc.programa <- NA
cc.version <- NA

# Procesar modulo hasta encontrar >>END_MODULE
# @f: handler del fichero ya abierto
# @m: numero de modulo a procesar
# @e: estado del modulo (pass, fail o warn)
# @u: muestra
# @r: resumen
qcDataSample <- function(f, m, e, u, r) {

  # Filas y columnas
  fil <- c()
  col <- c()

  while (TRUE) {
    l <- readLines(f, n = 1)
    if (length(l) == 0) break
```

```
    # Fin de módulo
    if (l == ">>END_MODULE") break
    else {

      # Cabecera y datos
      if (startsWith(l, "#"))
        col <- strsplit(gsub("#", "", l), "\t")[[1]]
      else {
        v <- strsplit(l, "\t")[[1]]
        if (v[1] != "Filename") # No interesa
          for (x in v) fil[length(fil) + 1] <- x
      }
    }
  }

  # Devolver estructura
  d <- NA
  if (length(fil) > 0) {
    t <- matrix(data = fil, ncol = length(col), byrow = TRUE)
    colnames(t) <- col
    d <- as.data.frame(t)
  }

  return(d)
}

# Mostrar grafica si existe
# @u: muestra a obtener la grafica
# @m: modulo del que mostrar la grafica
# @sesion: datos de la sesion
qcGraphSample <- function(u, m, sesion) {

  r <- ""
  if (!is.na(qcGraficas[m])) {
    file <- paste0(qcImagenes,
                   gsub("\\.", "_", u),
                   "/", u, "_fastqc/Images/", qcGraficas[m])

    # Si la grafica no existe, mostrar imagen aviso
    if (!file.exists(paste0("www/",file)))
      #t <-  paste0("tags$img(src='noimage.png')")
      t <- helpText("There's no graphic for this option")
        else  t  <-    paste0("tags$img(src='",  file,  "',
width='100%')")
    r <- eval(parse(text = t))
  } else r <- helpText("There's no graphic for this option")

  return(list(r, NULL))
}


# Procesar muestra
# @modulo: modulo a leer
# @muestra: muestra a leer
# @sesion: datos de la sesion
qcData <- function(muestra, modulo, sesion) {

  # Comprobar que el fichero sumario exista
```

```
  sumario <- paste0(qcResultados,
                    gsub("\\.", "_", muestra),
                    "/", muestra, "_fastqc/summary.txt")

  # Terminar si no existen datos
  if (!file.exists(sumario))
    return(list(NULL, NULL))

  # Leer fichero resumen QC
  resumen <- read.delim(sumario,
                        header = FALSE)
  colnames(resumen) <- c("estado", "modulo", "fichero")

  file <- paste0(qcResultados,
                 gsub("\\.", "_", muestra),
                 "/", muestra, "_fastqc/fastqc_data.txt")

  retorno <- NA
  estado <- "" # status de cada dato / grafico

  nl <- 0
  m <- 0
  f <- file(file, "r")
  while (TRUE) {

    l <- readLines(f, n = 1)
    if (length(l) == 0) break

    # Contar líneas
    nl <- nl + 1

    # La primera fila es el programa y la versión
    if (nl == 1) {
      if (startsWith(l, "##")) {
        v <- strsplit(gsub("##", "", l), "\t")[[1]]
        if (length(v) == 2) {
          cc.programa <<- v[1]
          cc.version <<- v[2]
        }
        next
      }
    }

    # Inicio de módulo
    if (startsWith(l, ">>")) {
      ti <- strsplit(gsub(">>", "", l), "\t")[[1]]
      m <- match(ti[1], qcModulos)
      if (!is.na(m)) {
        estado <- ti[2]
        if (modulo == 0) # Todos
            retorno <- paste0(retorno, qcDataSample(f, m, ti[2],
muestra, resumen))
         else if (modulo == m) { # Procesar el modulo indicado y
terminar
          retorno <- qcDataSample(f, m, ti[2], muestra, resumen)
          break
        }
      }
```

```
    }
  }
  close(f)

  # Si no hay datos, mostrar mensaje
  if (is.na(retorno)) {
    if (modulo == 10)
                retorno  <-  as.data.frame(c("No  overrepresented
sequences"))
    else if (modulo == 12)
      retorno <- as.data.frame(c("No overrepresented Kmers"))
    else retorno <- as.data.frame(c("No data for this option"))
    colnames(retorno) <- c("Result")
  }

  # Devolver lista con data.frame y vector de status
  return(list(retorno, estado))
}

# Ejecutar control de calidad para los ficheros de secuencias
# @muestras: lista de muestras a analizar
exeQC <- function(muestras) {

  # Ejecutar
  for (i in 1:length(muestras)) {
    dir_r <- paste0(qcResultados, i)

    # Si existe, se borra
    if (dir.exists(dir_r)) {
      unlink(dir_r, recursive = TRUE)
    }
    dir.create(dir_r)

    comando <- paste(qcExe,
                     qcProgram,
                     dir_r,
                     paste0("./www/uploads/", i, ".fastq"),
                      paste0(">", dir_r, ".log 2>", dir_r, ".err
&"))
    #cat(comando, "\n")
    # La ejecución se realiza en background (&) para
    # asegurar que el proceso se puede completar
    # incluso si se realiza una desconexion de la
    # aplicacion
    # TO-DO: incluir aviso por email para cuando
    # finalice el proceso
    system(comando)
  }
}

# Devuelve la estructura UI para acceder a cada uno de los
# resultados existentes en los diferentes directorios.
# Ademas comprueba el estado del estadistico
oldQC <- function(n, o) {

  if (is.null(o))
    return("helpText('There is no result')")
```

```
  #cat("Pasa\n")

  # Datos por muestra
  ui <- ""
  resultados <- dir(qcResultados)
  for (i in 1:n) {
    time <- file.info(paste0(qcResultados, resultados[i]))$mtime
    q <- qcData(i, as.numeric(o), session)[2]
    if (is.null(q)) q <- "warn"
    if (is.null(q[[1]])) q <- "fastqc"
    #show(q)
     ui <- paste0(ui, ", tabPanel(title=div(img(src='icons/", q,
".png'), 'Sample " , i, " :: ", time, "'),",
                  "uiOutput('ui_qc", i, "'))")
  }

  # Devolver mensaje si no hay resultados
  if (ui == "")
    return("helpText('There is no result')")

  # Completar
  ti <- "Results"
     ui  <-  paste0("div(id='dqctab',  br(),  column(12,
align='center',",
                "uiOutput('cb_qc'),",
                   "fluidRow(tabBox(title = tagList('", ti, " ',
shiny::icon('stack-overflow')),",
                "id='qctab', width='100%' ",
                ui, "))))")
  return(ui)
}

# Opciones QC
# Submenu de QC en vector
optionsQC <- function() {

  r <- ""
  for (i in 1:length(qcModulos)) {
    if (r != "") r <- paste0(r, ",")
    r <- paste0(r, "'", qcModulos[i], "' = '", i, "'")
  }
  r <- paste0("c(", r, ")")
  return(eval(parse(text = r)))
}
```

includes/ct.R

```
# Funciones genericas para Tabla de contajes

# Variables globales
ctExe0 <- "./ct.0.bash"
#ctExe1 <- "./ct.1.bash"
#ctExe2 <- "./ct.2.bash"
ctResultados <- "./www/results/ct/"

# Ejecutar la generación tabla de contajes de secuencias
# No se ejecutan en background porque el consumo de recursos
aumenta significativamente
# @muestras: lista de muestras a analizar
# TO-DO: incluir aviso por email para cuando finalice el proceso
exeCT <- function(muestras) {

    # Borrar todos los subdirectorios de ./results/ antes de
continuar
  unlink(ctResultados, recursive = TRUE)
  dir.create(ctResultados)
  dir.create(paste0(ctResultados, "bam")) # Links de bam

  # Links
  links <- ""

  # Primera fase (0)
  comandos <- ""
  for (i in 1:length(muestras)) {
    dir_r <- paste0(ctResultados, i)
    if (!dir.exists(dir_r)) dir.create(dir_r)
    comando <- paste(ctExe0,
                     i,
                     paste0("./www/uploads/", i, ".fastq"),
                     dir_r,
                     especie,
                       ">", paste0(ctResultados, i, ".log"), "2>
", paste0(ctResultados, i, ".err\n"))
    comandos <- paste0(comandos, comando, "\n")
          links <- paste0(links, "ln -s ../", i, "/tophat/
accepted_hits.bam ", i, ".bam\n")
    #cat(comando, "\n")
  }

  # Fase final - Crear directorios para links de bam de Tophat
  comandos <- paste0(comandos, "cd ", ctResultados, "bam\n")
  comandos <- paste0(comandos, links)
  comandos <- paste0(comandos, "Rscript /srv/shiny-server/c2eYs/
crearCT.R\n")

  # Creacion de shell bash para ejecucion en secuencia
  f <- file("ct.bash")
  writeLines(comandos, f)
  close(f)
```

```
  # Ejecucion en background
  system("bash ./ct.bash > ct.log 2> ct.log &")
}

# Ejecutar procesos comunes a todos los resultados de exeCT0
# Segunda fase
#exeCT1 <- function() {
#  comando <- paste0(ctExe1, " ", ctResultados, " > ct.1.log 2>
ct.1.err")
#  system(comando)
#}

# Devuelve la estructura UI para acceder a la tabla de contajes
# @n: origen de la tabla de contajes
oldCT <- function(n) {

  # Datos para la tabla de contajes
  ti <- "Counting Table"
    if (n == "D") ti <- paste0(ti, " From Demo
(gageData::data(hnrnp.cnts))")
  else if (n == "C") ti <- paste0(ti, " Generated")
  else if (n == "U") ti <- paste0(ti, " Uploaded")

  # Ejecutando
  if (n == "X")
    return(div(class = "busy",
               p("Processing ..."),
               img(src="cargando.gif")
    ))

    ui <- paste0("div(id='dcttab', br(), column(12,
align='center',",
               "uiOutput('cb_ct'),",
               "fluidRow(box(title = '", ti, "', ",
               "id='cttab', width='100%',",
               "datatable(values$tabla, rownames = TRUE,",
               "extensions = c('Buttons', 'Scroller'),",
                 "options = list(dom = 'Bfrtip',",
                                 "deferRender = TRUE, scrollY =
300, scroller = TRUE,",
                                    "buttons = c('copy', 'csv',
'excel', 'pdf', 'print')))))))")

  return(ui)
}

# Devuelve la estructura UI para acceder a la tabla de contajes
generada
oldCTGen <- function() {

  # Datos para la tabla de contajes
  ti <- "Counting Table Generated"

    ui <- paste0("div(id='dcttab', br(), column(12,
align='center',",
               "uiOutput('cb_ct'),",
               "fluidRow(box(title = '", ti, "', ",
               "id='cttab', width='100%',",
```

```
                "datatable(values$tabla, rownames = TRUE,",
                "extensions = c('Buttons', 'Scroller'),",
                "options = list(dom = 'Bfrtip',",
                   "deferRender = TRUE, scrollY = 300, scroller =
TRUE,",
                      "buttons = c('copy', 'csv', 'excel', 'pdf',
'print')))))))")

  ui <- NULL
  return(ui)
}

# Comprueba si existe una tabla de contajes
#  Se  busca  el  fichero  cuffData.info  que  indica  que  se  ha
terminado la ejecucić
existeCT <- function() {

  r <- FALSE
  #cat("Buscando ", paste0(ctResultados, "tabla.info\n"))
   if (file.exists(paste0(ctResultados, "bam/tabla.info"))) r <-
TRUE
  return(r)
}

# Crear tabla de contajes a partir de los resultados de tophat
```

includes/de.R

```
# Basado en:
# RNA Sequence Analysis in R: edgeR
#  https://web.stanford.edu/class/bios221/labs/rnaseq/
lab_4_rnaseq.html

# Datos expresion diferencial
data_de_y <- NULL
data_de_t <- NULL
data_de_d <- NULL
data_de_g <- NULL
data_de_l <- NULL

# Campos por fila
ncampos_de <- 4

# Normalizar
# @y: datos
normalizar <- function(y) {
  y <- calcNormFactors(y)
  return(y)
}

# Estimar dispersiones - complejo - GLM method
# @y: datos
# @m: matriz de diseño
# @e: metodo Trended (p.e. power)
estimarDispersionesGLM <- function(y, m, e) {
  # Comun
  y <- estimateGLMCommonDisp(y, m)
  # Tendencia
  y <- estimateGLMTrendedDisp(y, m, method = e)
  # Tagwise
  y <- estimateGLMTagwiseDisp(y, m)
  return(y)
}

# Realizar comparacion con matriz de diseño m
# para los grupos g, entre los elementos a y b
# a debe ser menor que b
# @y: datos
# @m: matriz de diseño
# @a: elemento menor a comparar
# @b: elemento mayor a comparar
# @n: numero de filas a mostrar (p.e. 10)
compararGrupos <- function(y, m, g, a, b, n) {
  fit <- glmFit(y, m)

  # Crear contraste
  contraste = c()
  for (i in 1:length(g)) {
    if (i == a) c[i] = 1
    else if (i == b) c[i] = -1
    else c[i] = 0
  }

  lrt <- glmLRT(fit,
```

```
                contrast = contraste)
  return(topTags(lrt, n = n))
}

# Hacer un test de decision
# @y: datos
# @m: metodo (p.e. BH)
# @p: pvalues (p.e. 0.05)
testDecision <- function(y, m, p) {
  return(decideTestsDGE(y, adjust.method = m, p.value = p))
}

# Prepara pantalla para procesar
de_off <- function(n) {
  shinyjs::hide(selector = "#detab li a[data-value=deplot]")
  #shinyjs::disable("typde")
  shinyjs::disable("detab")
  shinyjs::show("wt_de")

  for (i in 1:n)
    eval(parse(text = paste0("shinyjs::disable('nsg", i, "')")))
}

# Dejar pantalla como estaba antes de procesar
de_on <- function(n, t) {
   if (t != 3 && t != 4) shinyjs::show(selector = "#detab li
a[data-value=deplot]")
  shinyjs::hide("wt_de")
  shinyjs::enable("detab")
  shinyjs::enable("typde")

  for (i in 1:n)
    eval(parse(text = paste0("shinyjs::enable('nsg", i, "')")))

}
```

includes/main/countingTable.R

```
# Configuracion para Counting Table
output$ui_ct <- renderUI({

  # Ejecutar, leer, seleccionar tipo, seleccionar analiis
  btnExe <- NULL

  # Numero de ficheros cargados
  n <- countUploaded()

  # Ejecutar si ya existen datos
  # La ejecución se realiza sobre los datos cargados
  if (n >= values$nf)
    btnExe <- actionButton(inputId = "exect",
                           label = "Execute",
                           icon = icon("check-circle-o"))

  #isolate({
   #avisoTabla <- HTML("Counting table is missing. Please, do
Generate or Select")
  #  if (!is.null(values$tabla))
  avisoTabla <- HTML(paste0("<strong>NOTE:</strong> ",
                                "You must select an option from
select input.<br/>",
                              "If you clic Generate option then
Counting table will change."))
  #})
  #avisoTabla <- ""

  if (existeCT())
    opciones <- c("Select counting table:" = "-",
                  "Use generated counting table" = "C",
                  "Generate counting table" = "X",
                  "Upload counting table (matrix) ..." = "U",
                                    "Use demo counting table
(gageData::data(hnrnp.cnts))" = "D")
  else opciones <- c("Select counting table:" = "-",
                     "Generate counting table" = "X",
                     "Upload counting table (matrix) ..." = "U",
                                     "Use demo counting table
(gageData::data(hnrnp.cnts))" = "D")

  # Panel de botones
  fluidPage(
    column(12,
           align = "center",
           avisoTabla),
    column(3,
           selectInput(inputId = "typct",
                       label = "",
                       choices = opciones)),
    column(3,
           hidden(
             fileInput("ficct",
```

```
                        "",
                        multiple = FALSE,
                               placeholder="Select counting table
file"))),
    column(3, br(), hidden(btnExe)),
    column(3, br(),
           hidden(
             actionButton(inputId = "viect",
                          label = "View counting table",
                          icon = icon("th"))))
  )
})

# Mensaje de estado de resultados CT
output$ms_ct <- renderUI({

  # Dependencias
  input$exect
  values$ctProcesando
  values$ctProcesado

  # Comprobar el estado de Execute
  #ejecutandoCT()

  r <- NULL
   #cat("Procesando:", ifelse(values$ctProcesando, "SI", "NO"),
"\n")
    #cat("Procesado:", ifelse(values$ctProcesado, "SI", "NO"),
"\n")
    if (values$qcProcesado) r <- HTML("<p style='color:
darkgreen'><strong>CT completed. Tabs have been reloaded with
new values.</strong></p>")
  else if (values$ctProcesando && !values$ctProcesado)
    r <- HTML("<p style='color: red'><strong>Executing CT ... </
strong>You can see progress status in dashboard.</p>")
  return(r)
})

# Configuracion para resultados Counting table
output$ui_results_ct <- renderUI({

  # Dependencias
  #values$tabla
  input$viect
  values$ctProcesando
  values$ctProcesado

  # Estado ejecucion CT
  ejecutandoCT()

  # Mostrar mensaje de ocupado
  div(class = "busy",
      p("Processing ..."),
      img(src="cargando.gif")
  )
})
```

includes/main/observes.R

```
# Cuando cambia el numero de ficheros propuestos
# para cargar, actualizar values$nf
observeEvent(input$nf, {
  if (input$nf != values$nf) values$nf <- input$nf
})

# Cuando se pulsa el boton de ejecucion del QC (exeqc)
observeEvent(input$exeqc, {

  # Marcar como procesando
  values$qcProcesando <- TRUE
  values$qcProcesado <- FALSE
  values$qcLeido <- FALSE

  # Deshabilitar botones mientras se ejecuta
  deshabilitarQC()

  # Crear vector de muestras
  muestras <- c()
  isolate({
    for (i in 1:values$nf) {
      muestras[i] <- values$uf[i]
    }
  })

  # Ejecutar QC sobre el directorio qcResultados
  # para las muestras del vector
  exeQC(muestras)

  # Volver a habilitar los botones
  habilitarQC()

})

# Cuando se cambie el tipo de qc,
# reiniciar funcion de configuracion
# de tabs en QC
observeEvent(input$typqc, {

  # Constantes
  isolate({
    t <- input$typqc
  })
  r <- ifelse(t == "D", "renderTable", "renderUI")
  s <- ifelse(t == "D", "qcData", "qcGraphSample")

  # Para todos los resultados
  for (i in 1:length(dir(qcResultados))) {

    # Creacion de tab, (reescribir)
    h <- paste0("output$ui_qc", i, " <- ", r, "({",
                "o <- as.numeric(input$optqc)\n",
```

```
                "t <- input$typqc\n",
                "s <- ", s, "(", i, ", o, session)\n",
                "return(s[1])",
                "})")
    eval(parse(text = h))

    # Cabecera tab
    #eval(parse(text = paste0("js$tabqc(", i, ", qcData(", i, ",
as.numeric(input$optqc), session)[2])")))
  }
})

# Cuando se pulsa el boton Read
# se ejecuta leen todos los resultados
# almacenados en el directorio qcResultados
observeEvent(input$oldqc, {
  leerQC()
})

# Seleccionar tipo de tabla de contajes
observeEvent(input$typct, {

  isolate({ t <- input$typct })

  #cat("typct cambiado a", t, "\n")

  # Reiniciar DE
  data_de_l <<- NULL
  values$deEjecutado <- FALSE

  # No seleccionar tabla
  if (t == "-") {
    shinyjs::hide("ficct")
    shinyjs::hide("viect")
    shinyjs::hide("exect")
    values$tabla <- NULL
  }

  # Se pueden realizar los calculos de nuevo
  if (t == "X") {
    #cat("Pasa a X\n")
    ejecutandoCT()
    shinyjs::hide("ficct")
    shinyjs::hide("viect")
    shinyjs::show("exect")
    values$tabla <- NULL
  }

  # La tabla es la obtenida en los calculos anteriores
  # del paso Counting table
  if (t == "C") {
    shinyjs::hide("ficct")
         values$tabla  <-  read.table(paste0(ctResultados,  "bam/
tabla_contajes.txt"), header = TRUE)
    shinyjs::show("viect")
    shinyjs::hide("exect")
    ejecutandoCT()
  }
```

```
  # La tabla sera cargada a partir de un fichero
  # Debe ser de tipo matrix
  if (t == "U") {
    shinyjs::show("ficct")
    if (is.null(values$tabla)) shinyjs::hide("viect")
    else shinyjs:show("viect")
    shinyjs::hide("exect")
  }

  # Se utilizara una tabla demo del paquete gageData
  if (t == "D") {
    shinyjs::hide("ficct")
    values$tabla <- hnrnp.cnts # Precargada
    shinyjs::show("viect")
    shinyjs::hide("exect")
  }

})

# Cuando se carga un fichero
observeEvent(input$ficct, {
   cat("Cargando tabla de contajes desde ", input$ficct$name,
"\n")

  js_string <- ''
  result = tryCatch({
    t <- read.table(input$ficct$datapath, header = TRUE)
    values$tabla <- t
  }, warning = function(w) {
    cat("Warning CT\n")
     js$alert("File format warning. Please choose one valid file
format (Tab-separated)")
  }, error = function(e) {
    cat("Error CT\n")
     js$alert("File format error. Please choose one valid file
format (Tab-separated)")
  }, finally = {
    cat("Finally CT\n")
  })

  cat("Tabla de contajes cargada\n")
  shinyjs::show("viect")
  shinyjs::hide("exect")
})

# Cuando se haga clic en View Counting table
observeEvent(input$viect, {
  #shinyjs::show("ui_results_ct")
  leerCT()
})

# Cuando se hace clic en Execute Counting Table
observeEvent(input$exect, {

  # Marcar como procesando
  values$ctProcesando <- TRUE
  values$ctProcesado <- FALSE
```

```
  values$ctGenerando <- TRUE
  values$ctGenerada <- FALSE
  values$ctLeido <- FALSE

  # Crear vector de muestras
  muestras <- c()
  isolate({
    for (i in 1:values$nf) {
      muestras[i] <- values$uf[i]
    }
  })

  # Ejecutar QC sobre el directorio qcResultados
  # para las muestras del vector
  exeCT(muestras)
})

# Cuando se selecciona una seccion Differential Expression
observeEvent(input$typde, {

  # Ocultar / Mostrar UI de resultados
   if (is.null(values$tabla) || input$typde == "" || input$typde
== "0")
    shinyjs::hide("ddetab")
  else {
    # Seleccionar tab 1
    updateTabsetPanel(session, "detab", selected="dedata")
    shinyjs::show("ddetab")
    # Ocultar tab Plot
    if (input$typde == "3" || input$typde == "4")
      shinyjs::hide(selector = "#detab li a[data-value=deplot]")
  }

})

# Seleccion de Plot MDS - BCV
observeEvent(input$t1a, {
  # Dependencias
  values$btt1 <- "a"
})

# Seleccion de Plot MDS - logFC
observeEvent(input$t1b, {
  # Dependencias
  values$btt1 <- "b"
})

# Seleccion de Dendograma
observeEvent(input$t1c, {
  # Dependencias
  values$btt1 <- "c"
})

# Seleccion de Heatmap
observeEvent(input$t1d, {
  # Dependencias
  values$btt1 <- "d"
})
```

```
# Compare pulsado
observeEvent(input$t4, {
  # Dependencias
  data_de_l <<- NULL
  values$deEjecutado <- FALSE
  values$exede <- rnorm(1)
})

# Cambio grupo A
observeEvent(input$grade, {
  isolate({
    values$grade <- input$grade
  })
})

# Cambio grupo B
observeEvent(input$grbde, {
  isolate({
    values$grbde <- input$grbde
  })
})
```

includes/main/funciones.R

```
# Valores reactivos
values <- reactiveValues(tf = tf, # Maximo numero de ficheros
                         nf = nf, # Numero de ficheros cargados
(por defecto, 2)
                        uf = uf, # Nombre de los ficheros
                             qcProcesando = FALSE, # No se esta
procesando una QC
                                qcProcesado = FALSE, # No se ha
completado el QC
                        qcLeido = FALSE, # No se ha leido el QC
                             ctProcesando = FALSE, # No se esta
procesando una CT
                                ctProcesado = FALSE, # No se ha
completado el CT
                         ctGenerada = FALSE, # No se ha generado
la CT
                              ctGenerando = FALSE, # No se esta
generando la CT
                         ctLeido = FALSE, # No se ha leido la CT
generada
                            ob = rep(FALSE, tf), # observeEvent
para cada fichero
                        tabla = NULL, # Tabla de contajes
                         deDataGenerada = "0", # Tipo de últimos
datos generados DE
                               dePlotGenerado = NULL,  # Tipo de
último plot generado en DE
                          bsEjecutando = FALSE, # No se muestra
goana
                           bsEjecutado = FALSE, # No se muestra
goana
                        btt1 = "a", # Boton pulsado en T1
                             exede = 0, # Control para boton de
comparacion
                        grade = NULL, # Grupo A en DE
                        grbde = NULL # Grupo B en DE
)

# Deshabilitar botones QC
deshabilitarQC <- function() {
  shinyjs::disable("exeqc")
  shinyjs::disable("oldqc")
  shinyjs::disable("typqc")
  shinyjs::disable("optqc")
}

# Habilitar botones QC
habilitarQC <- function() {
  shinyjs::enable("exeqc")
  shinyjs::enable("oldqc")
  shinyjs::enable("typqc")
  shinyjs::enable("optqc")
}
```

```
# Reconstruir cabeceras de tabs en QC
rebuildTabsQC <- function() {
  # Para todos los resultados
  for (i in 1:length(dir(qcResultados))) {
    # Cabecera tab
     eval(parse(text = paste0("js$tabqc(", i, ", qcData(", i, ",
as.numeric(input$optqc), session)[2])")))
  }
}

# Leer estatus fichero QC
# @n: numero de muestra
leerStatusQC <- function(n) {

  #options(show.error.messages = FALSE)

  f <- paste0(qcResultados, n, "/", n, "_fastqc/summary.txt")
  if (file.exists(f)) {

    # Buscar 1 FAIL
      s <- try(system(paste0("grep '^FAIL' ", f, " | wc -l "),
intern = TRUE))
    if (s[1] == 1) return("fail")

    # Buscar 1 WARN
      s <- try(system(paste0("grep '^WARN' ", f, " | wc -l "),
intern = TRUE))
    if (s[1] == 1) return("warn")

  } else return("fail")

  #options(show.error.messages = TRUE)
  return("pass")
}

# Lectura de los datos generados en QC
leerQC <- function() {

  # Deshabilitar botones mientras se ejecute
  deshabilitarQC()

  # Cuantos ficheros cargados
  n <- countUploaded()

  # Construir UI para los resultados
  removeUI(selector = "#dqctab")
  insertUI(selector = "#ui_results_qc",
           ui = eval(parse(text = oldQC(n, input$optqc))))

  # Tipo de salida definida actualmente
  t <- input$typqc

  # Comprobar la seleccion de typqc
  r <- ifelse(t == "D", "renderTable", "renderUI")
  s <- ifelse(t == "D", "qcData", "qcGraphSample")

  # Crear funciones para la seleccion de cada muestra
  # y tipologia de los datos a mostrar
```

```
  for (i in 1:length(dir(qcResultados))) {

    # Creacion de tab,
    # por defecto se muestran datos de Basic Statistics
    h <- paste0("output$ui_qc", i, " <- ", r , "({",
                "o <- as.numeric(input$optqc)\n",
                "t <- input$typqc\n",
                "s <- ", s, "(", i, ", o, session)\n",
                "return(s[1])",
                "})")
    eval(parse(text = h))
  }

  # Mostrar opciones
  shinyjs::show("typqc")
  shinyjs::show("optqc")

  # Volver a habilitar botones
  habilitarQC()
}

# Ocultar boton Execute de CT si hay algun proceso ct arrancado
ejecutandoCT <- function() {
  #autoInvalidate1()
  #cat("Comprobado ejecucion CT\n")
   s <- system("ps -ef | grep ct.bash |grep -v grep", intern =
TRUE)
  #cat("L:", length(s), "\n", s, "\n")
  if (length(s) > 0) shinyjs::disable("exect")
  else shinyjs::enable("exect")
}

# Leer tabla de contajes
leerCT <- function() {

  # UI
  shinyjs::show("wt_ct")
  uict <- eval(parse(text = oldCT(input$typct)))

  # Construir UI para los resultados
  removeUI(selector = "#dcttab")
  insertUI(selector = "#ui_results_ct", ui = uict)
  shinyjs::hide("wt_ct")
}

# Leer tabla de contajes generada
#leerCTGen <- function() {

#  # Leer datos
#  cuff <- readCufflinks("cuffdiff")
#  values$tabla <- countMatrix(genes(cuff))

#  # UI
#  shinyjs::show("wt_ct")
#  uict <- eval(parse(text = oldCTGen()))

#  # Construir UI para los resultados
#  removeUI(selector = "#dcttab")
```

```
#  insertUI(selector = "#ui_results_ct", ui = uict)
#  shinyjs::hide("wt_ct")
#}

# Creacion dinamica de los botones de carga de ficheros
createUploadButtons <- function(n) {

  t <- ""
  for (i in 1:n) {
    if (t != "") t <- paste(t, ", ")
    isolate({
          nu <- ifelse(values$uf[i] == "", "no file selected",
values$uf[i])
    })
    t <- paste0(t,
                "column(4, align='center', fileInput('f", i,
                "', 'Choose sample file ", i,
                    ":', multiple = FALSE, accept = c('.fastq'),
width = '90%'",
                ", placeholder=\"", nu , "\"))")

    # Añadir CSS para atributo ajuste del UI
    #t <- paste0(t,
    #            ", tags$head(tags$style(type='text/css', '#f",
    #            i, " {margin-left: 10px}')))")

    # Creacion dinamica de "observe events"
    # para recordar los ficheros cargados
    # Cuando se carga un fichero, este es movido a un directorio
       # estatico para posterior uso. Si caducase la sesion o
hubiera
     # algun problema en el servidor, podria perderse el fichero
cargado.
    # Además se actualiza el nombre de la muestra en el fichero
    # correspondiente <nmuestra>.info para futuros accesos

    # Si existe no volver a crear
    # Se crean dinamicamente la primera vez que se necesitan
    if (!values$ob[i]) {
      o <- paste0("observeEvent(input$f", i, ", {",
                  "nombre <- input$f", i, "$name\n",
                  "origen <- input$f", i, "$datapath\n",
                         "write(nombre, paste0(uploads, '", i,
".info'))\n",
                  "values$uf[", i, "] <- nombre\n",
                   "destino <- paste0(uploads, '", i, ".fastq')
\n",

                        # Si el destino ya existe, borrar sus
resultados QC

                  "if (file.exists(destino)) {\n",
                    "dir_r <- paste0(qcResultados,", i, ")\n",
                    "unlink(paste0(dir_r, '.err'))\n",
                    "unlink(paste0(dir_r, '.log'))\n",
                    "if (dir.exists(dir_r)) {\n",
                      "unlink(dir_r, recursive = TRUE)\n",
                    "}\n",
```

```
                    "}\n",
                    "if (file.exists(origen)) {\n",
                      "file.rename(origen, destino)\n",
                    "}\n",
                  "})")
      e <- eval(parse(text = o))

      # Marcar como hecho
      values$ob[i] <- TRUE
    }
  }

  # Actualizar el numero de muestras
  write(n, paste0(uploads, "n.info"))

  return(t)
}

# Contar los ficheros cargados en la sesion
countUploaded <- function() {
  n <- 0 # Numero de ficheros cargados
  for (i in 1:values$nf) {
    t <- paste0("!is.null(values$uf[", i, "]) && values$uf[", i,
"] != ''")
    if (eval(parse(text = t))) n <- n + 1
  }
  return(n)
}
```

includes/main/dashboard.R

```
# Configuracion Dashboard
output$ui_db <- renderUI({

  infodir <- ""
  if (dir.exists(qcResultados)) {
      time <- file.info(qcResultados)$mtime
      infodir <- paste0(" (Created: ", time, ")")
  }

  # Crear UI
  fluidRow(
    column(12,
           align="center",
           box(title = tagList("Phase 1 - Data",
                               shiny::icon("database")),
                   footer = "All samples must be uploaded before
next step. Required file format is fastq.",
               width = 12,
               status = "danger",
               collapsible = TRUE,
               uiOutput("ui_db_dt"))),
    column(12,
           align="center",
                  box(title = tagList(paste0("Phase 2 - Quality
Control", infodir),
                               shiny::icon("quora")),
                   footer = "Executing Quality Control is a good
practice. Don't forget it.",
               width = 12,
               status = "primary",
               collapsible = TRUE,
               uiOutput("ui_db_qc"))),
    column(12,
           align="center",
           box(title = tagList("Phase 3 - Counting Table",
                               shiny::icon("th")),
                   footer = "Counting table is required for next
processes.",
               width = 12,
               status = "success",
               collapsible = TRUE,
               uiOutput("ui_db_ct"))),
    column(12,
           align="center",
                   box(title = tagList("Phase 4 - Differential
Expression",
                               shiny::icon("vcard-o")),
               width = 12,
               status = "info",
               collapsible = TRUE,
               uiOutput("ui_db_de"))),
    column(12,
           align="center",
```

```
                    box(title = tagList("Phase 5 - Biological
Significance",
                                        shiny::icon("eyedropper")),
                width = 12,
                status = "warning",
                collapsible = TRUE,
                uiOutput("ui_db_bs")))
  )
})

# Configuracion para panel DB Data
output$ui_db_dt <- renderUI({

  # Comprobar que se han cargado todos los ficheros propuestos
  t <- ""
  for (i in 1:values$nf) {
    if (t != "") t <- paste0(t, ",")
    info <- paste0(uploads, i, ".info")

    # Comprobar si existe informacion del fichero a cargar
    if (file.exists(info)) {
      nombre <- readLines(info)

      # Separar nombre y extension
      dnombre <- strsplit(basename(nombre), "\\.")[[1]]
      fnombre <- paste0(uploads, i, ".fastq")

      # Comprobar si existe el fichero
      if (file.exists(paste0(uploads, i, ".fastq")))
        t <- paste0(t, "infoBox(title = 'Sample ", i, "',",
                    "subtitle = 'COMPLETED',",
                    "value = '",  dnombre[1], "',",
                     "icon=icon('database'), color='olive', width
= 3)")
      else
        t <- paste0(t, "infoBox(title = 'Sample ", i, "', ",
                    "subtitle = 'PENDING',",
                    "value = '",  dnombre[1], "',",
                     "icon=icon('database'), color = 'red', width
= 3)")
    } else
      t <- paste0(t, "infoBox(title = 'Sample ", i, "',",
                  "subtitle = 'PENDING',",
                  "value = '",  dnombre[1], "',",
                   "icon=icon('database'), color = 'red', width =
3)")
  }
  t <- paste0("fluidPage(", t, ")")
  eval(parse(text = t))
})

# Configuracion para panel DB Quality Control
output$ui_db_qc <- renderUI({

  # Cada segundo comprobar estado de la tarea
  autoInvalidate()

  # Contar los ficheros ya cargados
```

```
  n <- countUploaded()

  # Por defecto, no se ha procesado
  if (values$qcProcesando)
    values$qcProcesado <- FALSE

  t <- ""
  r <- 0
  for (i in 1:n) {
    if (t != "") t <- paste0(t, ",")
    f <- paste0(qcResultados, i, ".log")

    # Separar nombre y extension
    dnombre <- strsplit(basename(values$uf[i]), "\\.")[[1]]
    fnombre <- paste0(uploads, i, ".fastq")

    st <- paste0(leerStatusQC(i), ".png")
    cat("Muestra ", i, " = ", st, "\n")

    if (file.exists(f)) {
      if (file.size(f) == 0) {
        t <- paste0(t, "infoBox(title = 'Sample ", i, "', ",
                    "value = 'PROCESSING ...',",
                    "subtitle = '',",
                     "icon=icon('quora'), color = 'orange', width
= 3)")
      } else {
        o <- readLines(f)
        if (o == "OK") {
          t <- paste0(t, "infoBox(title = 'Sample ", i, "', ",
                       "subtitle = 'COMPLETED',",
                             "value = div(style='top-margin: 0px;
bottom-margin: 0px', img(src='icons/", st, "')),",
                            "icon=icon('quora'), color = 'olive',
width = 3)")
          r <- r + 1
        } else {
          t <- paste0(t, "infoBox(title = 'Sample ", i, "', ",
                       "value = 'PROCESSING ...',",
                       "subtitle = '',",
                           "icon=icon('quora'), color = 'orange',
width = 3)")
        }
      }
    } else {
      t <- paste0(t, "infoBox(title = 'Sample ", i, "', ",
                  "value = 'PENDING',",
                  "subtitle = '',",
                     "icon=icon('quora'), color = 'red', width =
3)")
    }
  }

    # Marcar como terminado si todas las muestras han sido
procesadas
  if (r == n) {
    if (values$qcProcesando) {
      values$qcProcesado <- TRUE
```

```
      values$qcProcesando <- FALSE
    }

    # Leer los datos generados
    if (!values$qcLeido) {
      #leerQC()
      values$qcLeido <- TRUE
    }
  }

  t <- paste0("fluidPage(", t, ")")
  eval(parse(text = t))
})

# Configuracion para panel DB Counting Table
output$ui_db_ct <- renderUI({

  # Refrescar panel
  autoInvalidate()

  # Tipo de tabla
  t <- input$typct

  # Test pendiente
  if (is.null(t) || t == "-") return(helpText("Pending Select or
Generate"))
  if (t == "U")
    if (is.null(input$ficct))
      return(helpText("Pending select counting table file"))
    else if (input$ficct$name == "") return(helpText("Pending
select counting table file"))

   else return(helpText(paste0("File ", input$ficct$name, " has
beed uploaded")))
    if (t == "D") return(helpText("Demo Counting table
(gageData::hprnp.cnts) has been selected"))
    if (t == "C") return(helpText("Execution completed and
counting table available"))

   # Los procesos generados se diferencian en las siguientes
fases:
  # - Análisis de muestras indviduales
  # - Creacion de tabla

  # Generacion de una tabla nueva
  # Contar los ficheros ya cargados
  n <- countUploaded()

  # Por defecto, no se ha procesado
  if (values$ctProcesando)
    values$ctProcesado <- FALSE

  # Continuar
  t <- ""
  r <- 0     # OK
  e <- FALSE # Error
  m <- 0     # Sample
  s <- ""    # Estado
```

```
  for (i in 1:n) {
    if (t != "") t <- paste0(t, ",")
    f <- paste0(ctResultados, i, ".log")

    #cat("Leyendo ", f, "\n")
    # Recordar muestra
    m <- i

    # Separar nombre y extension
    dnombre <- strsplit(basename(values$uf[i]), "\\.")[[1]]
    fnombre <- paste0(uploads, i, ".fastq")

    if (file.exists(f)) {
      if (file.size(f) == 0) s <- "PROCESSING ..."
      else {
        o <- readLines(f)
        oo <- strsplit(o[length(o)], ":")[[1]]
        if (oo[1] == ">OK")
          if (oo[2] == "Fin") {
            s <- "COMPLETED"
            r <- r + 1
          } else s <- "PROCESSING ..."
        else { # Error
          s <- paste0("ERROR: ", oo[2])
          e <- TRUE
          values$ctProcesando = FALSE
          values$ctProcesado = FALSE
          break # Se termina con error
        }
      }
    } else if (s == "") s <- "PENDING"

    #cat("Estado ", s, "-", r, "\n")
  }

  # Info para analisis individual, fase 1
  v <- ""
  p <- "0 %"
  c <- ""

  # Si hay error
  if (e) c <- "violet"
  else {
    if (n != 0 && r != 0) p <- paste0( r * 100 / n, " %")
    if (r == 0) # No ha empezado
      c <- "red"
    else if (r == n) # Se todo ha ido OK
      c <- "olive"
    else c <- "orange"
  }

  t <- paste0(t, "infoBox(title = 'Individual Analysis',",
              "value = '", p, "',",
              "subtitle = '", s, "',",
               "icon = icon('object-ungroup'), color = '", c, "',
width = 6)")

  # Si todas las muestras no han sido procesadas OK
```

```
  if (r != n) {
    t <- paste0(t, ",infoBox(title = 'Creation Process',",
                "value = '0 %',",
                "subtitle = 'Process dependent on previous',",
                    "icon = icon('object-group'), color = 'red',
width = 6)")
    t <- paste0("fluidRow(", t, ")")
    return(eval(parse(text = t)))
  }

  if (file.exists(paste0(ctResultados, "/bam/tabla.info"))) {
    ok <- readLines(paste0(ctResultados, "/bam/tabla.info"))
    if (ok == "OK") {
      values$ctGenerada <- TRUE
      values$ctGenerando <- FALSE
      values$ctProcesando = FALSE
      values$ctProcesado = TRUE
         values$tabla <- read.table(paste0(ctResultados, "/bam/
tabla_contajes.txt"), header = TRUE)
    }
  }

  # Continuar con la segunda fase
  s <- "PENDING"
  c <- "red"
  if (values$ctProcesando) {
    s <- "PROCESSING ..."
    c <- "orange"
  }

  # Comprobar segunda fase
  if (values$ctGenerada)
    t <- paste0(t, ",infoBox(title = 'Creation Results',",
                "value = '100%',",
                "subtitle = 'COMPLETED',",
                  "icon = icon('object-group'), color = 'olive',
width = 6)")
  else
    t <- paste0(t, ",infoBox(title = 'Creation Results',",
                "value = '0%',",
                "subtitle = '", s, "',",
                 "icon = icon('object-group'), color = '", c, "',
width = 6)")

  t <- paste0("fluidRow(", t, ")")
  eval(parse(text = t))

})

# Configuracion para panel DB Differential Expression
output$ui_db_de <- renderUI({

  # Dependencias
  t <- input$typde

   if (is.null(t) || t == "0" || t == "") return(helpText("No
previous test generated"))
```

```
    if (t == "1") return(helpText("Showing Filtering &
normalization"))
    if (t == "2") return(helpText("Showing GLM Estimation
Dispersion (Common + Trended + Tagwise)"))
  if (t == "3") return(helpText("Showing Design Matrix"))
    if (t == "4") return(helpText("Showing GLM Comparision
Groups"))
  if (t == "5") return(helpText("Showing Decide Test"))
})

# Configuracion para panel DB Biological Significance
output$ui_db_bs <- renderUI({
  if (values$bsEjecutando) return(helpText("Executing Biological
Significance ..."))
    else if (values$bsEjecutado) return(helpText("Showing last
analysis results"))
    else return(helpText("Pending execution"))
})
```

includes/main/biologicalSignificance.R

```
# Configuracion para Biological Significance
output$ui_bs <- renderUI({

  return(
    fluidPage(
      column(12, uiOutput("ui_bs_al")),
      column(12, dataTableOutput("ui_bs_dt"))
    )
  )
})

# Mensaje de aviso si es necesario
output$ui_bs_al <- renderUI({

  # Dependencias
  values$tabla

     # Chequear si se tiene la informacion necesaria para
ejecutarse
  if (is.null(data_de_l) || !values$deEjecutado) {
    shinyjs::hide("wt_bs")
    shinyjs::hide("bs_dt")
    return(fluidPage(
      column(12,
             align = "center",
             HTML(paste0("<strong>NOTE:</strong> ",
                         "Do execute Differential Expression
before")))))
  }
})

# Datos del analisis
output$ui_bs_dt <- DT::renderDataTable({


  # Dependencias
  if (!values$deEjecutado) return (NULL)

  shinyjs::show("wt_bs")

  # No mostrar datos si no se puede
  if (is.null(data_de_l)) return(NULL)

  # Analisis de Significacion Biologica (goana)
  values$bsEjecutado = FALSE
  values$bsEjecutando = TRUE

  go <- NULL
  try(go <- goana(data_de_l, species = "Hs"), silent = TRUE)
   if (is.null(go)) js$alert("No genes found in universe. Please
verify counting table.")

  #go <- goana(data_de_l, coef="B.PregVsLac", species = "Hs")
```

```
  if (!is.null(go))
     d <- datatable(as.data.frame(topGO(go, on = "BP", "up", n =
300)),
                    rownames = TRUE,
                    extensions = 'Buttons', options = list(
                      dom = 'Bfrtip',
                       buttons = c('copy', 'csv', 'excel', 'pdf',
'print')
                       ))
  else {
      s = c("No genes found in universe. Please verify counting
table.")
    d = data.frame(s)
    colnames(d) <- c("")
    rownames(d) <- c("")
  }
  values$bsEjecutado = TRUE
  values$bsEjecutando = FALSE
  shinyjs::hide("wt_bs")
  shinyjs::show("bs_dt")

  return(d)
})
```

includes/qualityControl.R

```
# Configuracion para QC
output$ui_qc <- renderUI({

  # Contar ficheros cargados
  n <- countUploaded()

  # Mostar avisos
  t <- ""

  # Comprobar si se han cargado todos los ficheros
  if (n < values$nf)
     t <- paste0("Before analysis can be execute, you may upload
all sample data files (",
                 n, " of ", values$nf, " file(s) uploaded yet)")
   else t <- paste0("All (", n, ") sampling file(s) has been
uploaded",
                    "<br/>",
                    "<strong>NOTE:</strong> ",
                      "If you push 'Execute' button, all previous
results will be destroyed")

  # Crear UI
  fluidPage(
    column(12,
           align="center",
           HTML(t))
  )
})

# Botones para QC
output$bt_qc <- renderUI({

  # Ejecutar, leer, seleccionar tipo, seleccionar analisis
  btnExe <- NULL
  btnRead <- NULL
  btnType <- NULL
  btnOptions <- NULL

  # Contar ficheros cargados
  n <- countUploaded()

  # Ejecutar si se han cargado todos los ficheros
  if (n >= values$nf)
    btnExe <- actionButton(inputId = "exeqc",
                           label = "Execute",
                           icon = icon("check-circle-o"))

  # Si hay resultados
     if (dir.exists(paste0(qcResultados, "1")) || values
$qcProcesado) {

    # Lectura de resultados almacenados en último analisis
    btnRead <- actionButton(inputId = "oldqc",
                            label = "Read Results",
```

```
                            icon = icon("file-o"))

    # Tipo de resultado
    btnType <- hidden(
      selectInput(inputId = "typqc",
                  label = "",
                  choices = c("Data" = "D",
                              "Graphics" = "G")))

    # Seleccion de analisis
    btnOptions <- hidden(
      selectInput(inputId = "optqc",
                  label = "",
                  choices = optionsQC()))
  }

  # Panel de botones
  fluidPage(
    column(2, br(), btnExe),
    column(3, br(), btnRead),
    column(3, btnType),
    column(4, btnOptions)
  )
})

# Mensaje de estado de resultados QC
output$ms_qc <- renderUI({

  # Dependencias
  input$exeqc
  values$qcProcesando
  values$qcProcesado

  r <- NULL
     if (values$qcProcesado) r <- HTML("<p style='color:
darkgreen'><strong>QC completed. Tabs have been reloaded with
new values.</strong></p>")
  else if (values$qcProcesando && !values$qcProcesado)
    r <- HTML("<p style='color: red'><strong>Executing QC ... </
strong>You can see progress status in dashboard.</p>")
  return(r)
})

# Configuracion para resultados QC
output$ui_results_qc <- renderUI({

  # Dependencias
  input$exeqc
  input$oldqc

  # Mostrar mensaje de ocupado
  div(class = "busy",
      p("Processing ..."),
      img(src="cargando.gif")
  )
})

# Configuración titulo de los datos mostrados
```

```
output$cb_qc <- renderUI({

  t <- ifelse(input$typqc == "D", "Data", "Graphic")
  d <- paste0(t, " for ", qcModulos[as.numeric(input$optqc)])
  rebuildTabsQC()

  column(12,
         align = "center",
         h4(d))
})
```

includes/differentialExpression.R

```
# Configuracion para Differential Expression
output$ui_de <- renderUI({

  # Muestras en tabla
  n <- 0
  if (!is.null(values$tabla)) {
    n <- ncol(values$tabla)
    if (n < 1)
      return(fluidPage(
        column(12,
               align = "center",
               HTML(paste0("<strong>NOTE:</strong> ",
                                    "Counting table has wrong
format")))))
  }

  # Crear dinamicamente los campos para los grupos
  # Crear 4 campos por fila
  t <- ""
  f <- 1
  c <- 0
  for (i in 1:n) {
    if (t != "") t <- paste0(t, ",")
          t  <-  paste0(t,  "column(",  (12  /  ncampos_de),  ",
align='center',",
                "textInput('nsg", i, "', 'GROUP for sample ", i,
                  "', value='', placeholder='Group for sample ",
i, "'))")
  }

  # Terminar cadena
  if (t != "") t <- paste0("fluidRow(", t, ")")

  opciones <- NULL
  if (t != "") opciones <- eval(parse(text = t))

  if (is.null(values$tabla))
    fluidPage(
      column(12,
             align = "center",
             HTML(paste0("<strong>NOTE:</strong> ",
                                  "Counting table must be provided
before"))))
  else fluidPage(
    column(12,
                     helpText("Introduce  names  of  groups  before
continue")),
    fluidPage(opciones)
  )

})

# Botones de accion para Differential Expression
```

```
output$bt_de <- renderUI({

  retorno <- NULL

  # Si no hay tabla de contaje, no mostrar datos
  if (is.null(values$tabla))
    return(NULL)

  # Chequear si se ha introducido nombre para todos
  # los samples de la tabla de contajes
  n <- ncol(values$tabla)
  if (n < 1)
    return(fluidPage(
      column(12,
             align = "center",
             HTML(paste0("<strong>NOTE:</strong> ",
                         "Counting table has wrong format")))))

  #isolate({
  nt <- 0
  gn <- c()
  for (i in 1:n) {
    t <- eval(parse(text = paste0("input$nsg", i)))
    if (!is.null(t))
      if (trimws(t, "both") != "") {
        nt <- nt + 1
        gn[length(gn) + 1] = t
      }
  }
  #})

  if (nt != n) # No estan todos los datos introducidos
    return(fluidPage(
      column(12,
             align = "center",
             HTML(paste0("<strong>NOTE:</strong> ",
                         "You must fill all grupos before")))))

  # Comprobar que hay más de un grupo
  if (length(unique(gn)) == 1)
    return(column(12,
                  align = "center",
                  HTML(paste0("<strong>NOTE:</strong> ",
                              "Contrasts can be applied only to
factors with 2 or more levels"))))

  # Si existe la tabla de contaje
  retorno <- fluidPage(
    column(12,
           align = "center",
           selectInput(inputId = "typde",
                       label = "",
                       selected = "0",
                        choices = c("Select analysis section:" =
"0",
                                    "Filtering & Normalization" =
"1",
```

```
                                        "GLM Estimation Dispersion
(Commom + Trended + Tagwise)" = "2",
                                      "Design Matrix - No plot" =
"3",
                                     "GLM Comparision Groups - No
plot" = "4",
                                    "Decide Test" = "5"))))

  return(retorno)

})

# Resultados de Differential Expression
output$ui_results_de <- renderUI({

  if (is.null(values$tabla))
    return(NULL)

  #isolate({
  n <- ncol(values$tabla)
  nt <- 0
  for (i in 1:n) {
    t <- eval(parse(text = paste0("input$nsg", i)))
    if (!is.null(t))
      if (trimws(t, "both") != "") nt <- nt + 1
  }
  #})

  if (nt != n) # No estan todos los datos introducidos
    return(NULL)

  # Se crea oculto y se activa cuando haya seleccion
  retorno <- hidden(
    div(id = "ddetab",
        br(),
        column(12,
               align = "center",
               fluidRow(tabBox(title = tagList("Output",

shiny::icon("vcard-o")),
                               id="detab",
                               width = 12,
                               tabPanel(title = "Data",
                                        value = "dedata",

#dataTableOutput("ui_ded")),
                                        uiOutput("ui_ded")),
                               tabPanel(title = "Plot",
                                        value = "deplot",
                                        uiOutput("ui_deg")))))))

  return(retorno)
})

# Salida para los datos Data
output$ui_ded <- renderUI({

  isolate({
```

```
    grade <- values$grade
    grbde <- values$grbde
  })

  if (input$typde == "4")
    return(
      fluidRow(
        #column(12,
         #       helpText(paste0("Testing groups ", grade, " and
", grbde))),
        column(8,
               dataTableOutput("data_de_out")),
        column(4,
               align = "center",
               uiOutput("data_de_ctl"))))
  else if (input$typde == "5")
    return(
      fluidPage(
        column(12,
               helpText(paste0("Using tests for groups ", grade,
" and ", grbde))),
        column(12,
               dataTableOutput("data_de_out"))
      )
    )
  else return(
    fluidPage(
      column(12,
             dataTableOutput("data_de_out"))
    )
  )
})

# Salida para datos de resultados
output$data_de_out <- DT::renderDataTable({

  # Dependencias
  values$exede

  # Si no hay seleccion, no hacer nada
  if (input$typde == "" ||
      input$typde == "0") return(NULL)

  n <- ncol(values$tabla)

  # Leer los valores de los grupos en un vector
  # para crear los grupos
  isolate({
    grupos <- c()
    for (i in 1:ncol(values$tabla)) {
      t <- eval(parse(text = paste0("input$nsg", i)))
      if (is.null(t)) {
        de_on(n, input$typde)
        return(NULL) # Teminar si se detecta error
      } else grupos[i] <- t
    }
  })
```

```
  # Deshabilitar la seleccion mientras
  # se realizan los calculos
  de_off(n)

  # Matriz de diseño (necesaria)
  matriz.diseno <- model.matrix(~grupos)

  # Si es la matriz de diseño, no hay que hacer calculos
  if (input$typde == "3") {

    de_on(n, input$typde)
    nm.matriz.diseno <- matriz.diseno
    rownames(nm.matriz.diseno) <- colnames(values$tabla)
    return(datatable(nm.matriz.diseno,
                     rownames = TRUE,
                     extensions = 'Buttons', options = list(
                       dom = 'Bfrtip',
                             buttons = c('copy', 'csv', 'excel',
'pdf', 'print')
                     )))

  } else {

     # TO-DO. Ejecutar solamente si no se dispone de los datos a
este nivel de proceso

    cat("Procesando 1\n")

    # Empezar
    y <- NULL

    # Filtrar y Normalizar
    y <- DGEList(counts = values$tabla,
                 group = factor(grupos),
                 remove.zeros = TRUE)

     # Prescindir de los genes con una frecuencia inferior a 100
cuentas por millon (CPM)
      # Se mantienen aquellos genes con 100 o mas para al menos
dos muestras
       apply(y$counts, 2, sum) # Total de cuentas de gen por
muestras
    k <- rowSums(cpm(y) > 100) >= 2
    k <- y[k,]

    # Restablecer los tamaños de las librerias
    # Han sido borrados varios
    y$samples$lib.size <- colSums(y$counts)

    # Normalizar
    y <- normalizar(y)
    data_de_y <<- y

    cat("Fin proceso 1\n")

    # Si se ha elegido Normalizar, terminar con la lista
    isolate({
      if (input$typde == "1") {
```

```
        values$deDataGenerada = "1"
        de_on(n, input$typde)
        return(datatable(data_de_y$samples,
                         rownames = TRUE,
                         extensions = 'Buttons', options = list(
                           dom = 'Bfrtip',
                             buttons = c('copy', 'csv', 'excel',
'pdf', 'print')
                         )))
      }
    })

    cat("Procesando 2\n")

    # Estimacion de la dispersion
         y  <-  estimarDispersionesGLM(data_de_y,  matriz.diseno,
"power")
    data_de_y <<- y

    cat("Fin proceso 2\n")

    # Si se ha elegido Estimacion de la dispersion por GLM
    isolate({
      if (input$typde == "2") {
        values$deDataGenerada = "2"
        de_on(n, input$typde)
        return(datatable(data_de_y$samples,
                         rownames = TRUE,
                         extensions = 'Buttons', options = list(
                           dom = 'Bfrtip',
                             buttons = c('copy', 'csv', 'excel',
'pdf', 'print')
                         )))
      }
    })

    cat("Procesando 4\n")

    # Por defecto, generar para el grupo 1 y 2 con ajuste BH
    # Ajuste
    y <- glmFit(data_de_y, matriz.diseno)

    # Rellenar e vector para el contraste dinamico
    # segun seleccion de grupos
    ug <- unique(grupos)
    contraste <- rep(0, length(ug))
    isolate({
      gra <- values$grade
      grb <- values$grbde
    })
    if (is.null(gra)) gra <- ug[1]
    if (is.null(grb)) grb <- ug[2]
    for (i in 1:length(ug)) {
      if (gra == ug[i]) contraste[i] <- 1
      if (grb == ug[i]) contraste[i] <- -1
    }

    isolate({
```

```
      values$grade <- gra
      values$grbde <- grb
    })

    g <- glmLRT(y, contrast = contraste)
    t <- topTags(g, n = nrow(g), sort.by = "PValue")
    data_de_l <<- g

    g$table$fdr <- p.adjust(g$table$PValue, method = "BH")
    o <- order(g$table$PValue)
    oo <- cpm(data_de_y)[o,]
    #oo <- cpm(dd1)[o[1:10],]

    # Añadir FDR
    #met <- input$metde
    #if (is.null(met)) met <- "BH"
    #t$table$fdr <- p.adjust(g$table$PValue, method = met)

    updateTabsetPanel(session, "detab", selected = "dedat")

    data_de_t <<- t
     values$deEjecutado <- TRUE   # Tenemos data_de_l y se puede
ejecutar el BS

    cat("Fin proceso 4\n")

    # Comparar grupos -> delegado a otra funcion
    isolate({
      if (input$typde == "4") {
        values$deDataGenerada = "4"
        de_on(n, input$typde)
        return(datatable(as.data.frame(data_de_t),
                         rownames = TRUE,
                         extensions = 'Buttons', options = list(
                           dom = 'Bfrtip',
                             buttons = c('copy', 'csv', 'excel',
'pdf', 'print')
                         )) %>%
                  formatRound("logFC", digits = 4) %>%
                  formatRound("logCPM", digits = 4)# %>%
                #formatRound("LR", digits = 3) #%>%
                #formatRound("PValue", digits = 3) %>%
                #formatRound("FDR", digits = 3)
        )
      }
    })

    cat("Procesando 5\n")

    # Comprobar si se ha calculado el paso anterior
    if (is.null(data_de_l)) {
      de_on(n, input$typde)
      return(NULL)
    }

      # Anotaciones (data_de_l contiene los datos de la ultima
comparacion seleccionada)
```

```
    # Si la tabla ha sido generada por la aplicacion, keytype =
SYMBOL
    # Si la tabla es la DEMO, keytype = ENTREZID
    # Eso es lo normal, en cualquier caso, por si cambiara la db
de la aplicacion
    # siempre se intentara adivinar la clave de busqueda
    # Intentar descubrir la clave de busqueda entre las opciones

    claves <- keytypes(org.Hs.eg.db)
    ann <- NULL
    for (i in 1:length(claves)) {
      try(ann <- select(org.Hs.eg.db,
                        keys = rownames(data_de_l),
                        keytype = claves[i],
                                                    columns =
c("ENTREZID","SYMBOL","GENENAME")),
          silent = TRUE)
      if (!is.null(ann)) break
    }

    if (is.null(ann)) {
              js$alert("Impossible locate key for querying at
org.Hs.eg.db")
      de_on(n, input$typde)
      return()
    }

    table(ann$ENTREZID==rownames(data_de_l))
    data_de_l$genes <- ann

    # Total de genes diferencialmente expresados al 5% FDR
    y <- decideTestsDGE(data_de_l, adjust.method = "BH", p.value
= 0.05)
    data_de_d <<- rownames(data_de_y)[as.logical(y)]
    data_de_y <<- y
    #table(l1_12$table$fdr < 0.05)
    #table(l1_12$table$fdr < 0.05) / nrow(l1_12$table)

    cat("Fin proceso 5\n")

    # Test de decicion
    isolate({
      if (input$typde == "5") {
        values$deDataGenerada = "5"
        de_on(n, input$typde)
        #cat("CLASE1:", class(summary(data_de_y)), "\n")
        sm <- as.data.frame(summary(data_de_y))
        return(datatable(sm,
                         rownames = TRUE,
                         extensions = 'Buttons', options = list(
                           dom = 'Bfrtip',
                             buttons = c('copy', 'csv', 'excel',
'pdf', 'print')
                         ))) # %>%
        #formatRound("logFC", digits = 3))
      }
    })
  }
```

```
  # Habilitar control
  de_on(n, input$typde)

})

# Salida para graficos de resultados
output$ui_deg <- renderUI({ #renderPlot({

  # Dependencias
  input$typde

  # No mostrar Plot para estos tipos
  if (input$typde == "3" || input$typde == "4")
    return(NULL)

  n <- ncol(values$tabla)

  # Deshabilitar la seleccion mientras
  # se realizan los calculos
  #de_off(n)
  shinyjs::show(selector = "#detab li a[data-value=deplot]")

  if (input$typde == "1") {
    return(
      fluidPage(
        uiOutput("plot_de_out"),
        uiOutput("plot_de_ctl")
      )
    )
  }

  if (input$typde == "2") {
    return(
      fluidPage(
        column(12,
               align = "center",
               uiOutput("plot_de_out",
                        width = "100%"))
      )
    )
  }

  if (input$typde == "5") {
    return(
      fluidPage(
        column(12,
               align = "center",
               uiOutput("plot_de_out", width = "100%"))
      )
    )
  }

  # Habilitar control
  #de_on(n, input$typde)

})
```

```
# Plot
output$plot_de_out <- renderUI({

  # Dependencias
  values$deDataGenerada

  box(width = ifelse(input$typde == "1", 6, 12),
      plotOutput("plot_de_plot"))
})

# Plot
output$plot_de_plot <- renderPlot({

  # Dependencias
  #input$typde
  values$deDataGenerada
  values$btt1
  input$t2

  isolate({ n <- ncol(values$tabla) })

  isolate({

    # PlotMDS en Normalization
    if (input$typde == "1") {
      de_off(n)

      if (values$btt1 == "a")
        plotMDS(data_de_y,
                top = ifelse(is.null(input$t2), 500, input$t2),
                method = "BCV",
                     col = brewer.cols) #   as.numeric(data_de_y
$samples$group))
      else if (values$btt1 == "b")
        plotMDS(data_de_y,
                top = ifelse(is.null(input$t2), 500, input$t2),
                method = "logFC",
                     col = brewer.cols) #   as.numeric(data_de_y
$samples$group))
      else if(values$btt1 == "c") {
        x <- dist(t(data_de_y$counts))
        y <- hclust(x, "average")
        plot(y,
             main = "Dendogram",
             col = brewer.cols,
             xlab="dist(t(counts)")
      } else if(values$btt1 == "d") {
        x <- dist(t(data_de_y$counts))
        y <- hclust(x, "average")
        heatmap(as.matrix(x),
                col = heat.colors(8),
                main = "Heatmap")
      }

      de_on(n, input$typde)
      values$dePlotGenerado = "1"
    }
```

```
    # Plot BCV
    if (input$typde == "2") {
      plotBCV(data_de_y)

      values$dePlotGenerado = "2"
      de_on(n, input$typde)
    }

    # Plotsmear
    if (input$typde == "5") {
      plotSmear(data_de_l, de.tags = data_de_d)
      #abline(h = c(-2, 2), col = "blue")
      # Los genes el rojo son los obtenidos por el Test
      #plotMD(data_de_l, status = data_de_y)
      values$dePlotGenerado = "5"
      de_on(n, input$typde)
    }

  })
})

# Control del Plot
output$plot_de_ctl <- renderUI({

  # Que plot ha sido generado
  #input$typde
  values$deDataGenerada
  values$dePlotGenerado

  isolate({
    # PlotMDS en Normalization
    if (input$typde == "1") {
      return(box(
        width = 6,
        h3("Plots"), br(),
        HTML("<strong>Select graph type:</strong>"), br(), br(),
        actionButton(inputId = "t1a", label = "MDS - BCV"),
        actionButton(inputId = "t1b", label = "MDS - logFC"),
        actionButton(inputId = "t1c", label = "Dendogram"),
        actionButton(inputId = "t1d", label = "Heatmap"),
        br(), br(),
        sliderInput("t2", "Number of genes:", 1, 10000, 500),
        helpText("Note: slider only for Plot MDS")
      ))
    }
  })
})

# Box en data comparacion
output$data_de_ctl <- renderUI({

  # Que plot ha sido generado
  #input$typde
  values$deDataGenerada
  values$dePlotGenerado

  isolate({
    # Comparaciones por pares de grupos
```

```
    if (input$typde == "4") {

      # Contar grupos unicos
      gn <- c()
      n <- ncol(values$tabla)
      for (i in 1:n) {
        t <- eval(parse(text = paste0("input$nsg", i)))
        if (!is.null(t))
          if (trimws(t, "both") != "") gn[length(gn) + 1] = t
      }
      gr <- c()
      gu <- unique(gn)
      for (i in 1:length(gu)) {
          gr[i] = eval(parse(text = paste0("c('", gu[i], "'='",
gu[i], "')")))
      }
      de_on(n, input$typde)

      isolate({
        values$grade <- gr[1]
        values$grbde <- gr[2]
      })

      # Ocultar pestaña
      shinyjs::hide(selector = "#detab li a[data-value=deplot]")

      return(box(
        width = 6,
        h3("Groups comparision"),
        br(),

        # Group 1
        selectInput(inputId = "grade",
                    label = "Group A",
                    selected = gr[1],
                    choices = gr),

        # Grupo 2
        selectInput(inputId = "grbde",
                    label = "Group B",
                    selected = gr[2],
                    choices = gr),

        #selectInput(inputId = "metde",
        #            label = "Adjust method for FDR:",
        #            selected = "BH",
          #              choices = c("BH", "holm", "hochberg",
"hommel", "bonferroni", "BY", "fdr", "none")),

        actionButton(inputId = "t4", label = "Compare")

      ))
    }
  })
})
```

includes/main/data.R

```
# Configuracion para Data
output$ui_dt <- renderUI({

  # Botones a mostrar
  n <- ifelse(values$nf == 0, 2, values$nf)

  # Mostrarlos
  t <- paste0("fluidRow(", createUploadButtons(n), ")")
  eval(parse(text = t))

})
```

7.2. Scripts Bash.

qc.bash

```
#!/bin/bash
export JAVA_HOME=/opt/jdk1.8.0_121
export PATH=$JAVA_HOME/bin:$PATH
export RESULTS=./www/results/qc
FASTQC=$1
if [ ! -e $FASTQC ]; then
 echo "FastQC no existe"
 exit
fi
shift
RESULTS=$1
export RESULTS
shift
#echo "$FASTQC -o $RESULTS --extract -f fastq $* > /dev/null
2> /dev/null"
$FASTQC -o $RESULTS --extract -f fastq $* > /dev/null 2> /dev/
null

# Marcar fin
echo "OK"
```

ct.0.bash

```
#!/bin/bash

# CPUs a usar cuando se puedan ejecutar los comandos por threads
CPUS=1

# Chequear parametros
if [ "$#" -ne 4 ]; then
  echo ">ER:Incorrect number of parameters"
  exit
fi

# Obtener valores de los parametros
NUMERO=$1
MUESTRA=$2
RESULTADOS=$3
ESPECIE=$4

echo ">OK:1:Creating environment ..."

# Cambiar al directorio de resultados
if [ ! -d "$RESULTADOS" ]; then
  echo ">ER:Results folder does not exist"
  exit
fi
cd $RESULTADOS

echo ">OK:2:Testing free space ..."

# Comprobar si hay espacio en disco para
# realizar la copia de la muestra al directorio
LIBRE=`echo $(($(stat -f --format="%a*%S" .)))`
TAMANO=`wc -c < ../../../../$MUESTRA`
if [ "$TAMANO" -gt "$LIBRE" ]; then
  echo ">ER:There is not enough free space"
  exit
fi

echo ">OK:3:Copying sample file ..."

# Si hay espacio en disco, copiar fichero a directorio de
trabajo
cp ../../../../$MUESTRA muestra.fastq
if [ "$?" -ne "0" ]; then
  echo ">ER:Error while copying sample file"
  exit
fi

echo ">OK:4:Executing Tophat ..."

# Ejecuta Tophat
tophat -p $CPUS -G ../../../../../gtf/genes.gtf --library-
type=fr-firststrand -o tophat ../../../../../Bowtie2Index/genome
muestra.fastq 2> tophat.err > tophat.log
OK=`tail -1 tophat.err | grep -c "Run complete"`
if [ "$OK" -ne "1" ]; then
  echo ">ER:Error executing Tophat"
```

```
  exit
fi


# Terminar
echo ">OK:Fin"
```