Creación de un pipeline para el análisis de datos procedentes de PCR cuantitativa

Elena García Bravo
Máster Universitario de Bioinformática y Bioestadística
Trabajo Fin de Máster

Nombre del Consultor: Ricardo Gonzalo Sanz

Fecha de Entrega: 24 de Mayo de 2017

Universitat Oberta de Catalunya

Ficha del trabajo final.

Título del trabajo	Creación de un pipeline para el análisis de datos procedentes de PCR cuantitativa
Nombre del autor	Elena García Bravo
Nombre del consultor	Ricardo Gonzalo Sanz
Profesores responsables de la asignatura	Alexandre Sánchez Pla, Antoni Pérez Navarro, Carles Ventura Royo, Jose Antonio Morán Moreno y María Jesús Marco Galindo
Fecha de entrega	24 de Mayo de 2017
Titulación	Máster Universitario de Bioinformática y Bioestadística
Área del Trabajo Final	Trabajo Fin de Máster
Idioma del trabajo	Castellano
Palabras clave	qPCR, pipeline, R

Resumen

En este proyecto fin de Máster se ha diseñado y desarrollado un *pipeline* para el análisis estadístico de datos procedentes de PCR cuantitativa (qPCR) obtenidos en un aparato ABI7000 y medidos en placas prediseñadas de qPCR, miRCURY LNA Universal RT microRNA PCR assays (Exiqon), para miRNAs encontrados típicamente en exosomas. El pipeline está desarrollado en el lenguaje de programación `R`, haciendo uso de librerías desarrolladas especialmente para procesar datos biológicos, entre ellas el proyecto Bioconductor y el paquete `HTqPCR`. Como resultado del proyecto se ha obtenido un producto final que ha sido ejecutado sobre unos datos de muestra y se han generado unos resultados que nos han permitido obtener conclusiones y responder a las preguntas inicialmente planteadas. Dichos resultados se presentan en un documento que integra código y comentarios aclaratorios de texto dinámico en un archivo Rmd compilable en formato html y pdf. Asimismo, se ha comparado el funcionamiento del pipeline realizado con otras aplicaciones de análisis de qPCR ya existentes, de cara a implementar posibles mejoras futuras. La realización de este proyecto ha contribuido satisfactoriamente a consolidar los conocimientos de bioestadística y bioinformática adquiridos durante el máster y complementar mi formación como bióloga.

Abstract

This Master Thesis project consists in the elaboration of a pipeline for the statistical analysis of quantitative PCR data (qPCR) obtained in an ABI7000 thermocycler and measured in pre-designed plates of qPCR, miRCURY LNA Universal RT microRNA PCR assays (Exiqon), for quantifying miRNAs typically found in exosomes. The pipeline was developed with R programming language, making use of specific libraries to process biological data. Those libraries include the Bioconductor project and the package HTqPCR. As a result, I have obtained a final pipeline that has been executed on some sample data and generated results that allowed us to draw conclusions about the data and answer the biological questions initially raised. These results are presented in a document that integrates code and explanatory comments of dynamic text in an Rmd file, compilable as an html as well as a pdf file. Likewise, the performance of the pipeline has been compared with other existing qPCR analysis applications, in order to implement possible future improvements. The realization of this project has contributed to consolidate my knowledge of biostatistics and bioinformatics acquired during this master's degree and to complement my training as a biologist.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto y justificación del trabajo

El objetivo global de este proyecto ha sido el desarrollo de un flujo de trabajo (en adelante *pipeline*) para el análisis bioestadístico de datos procedentes de PCR cuantitativa (qPCR) basado en el lenguaje de programación R. El pipeline ha sido desarrollado en base a unos datos de qPCR previamente obtenidos por un grupo de investigación, y se evaluará su funcionalidad comparándolo con otras herramientas de análisis de qPCR ya existentes.

La PCR cuantitativa es hoy en día una técnica rutinaria en la mayoría de laboratorios de biología molecular. Es utilizada para numerosas aplicaciones, tanto en ciencia básica como aplicada, y se ha convertido en un método estándar para la cuantificación de la expresión génica.

Tras llevar a cabo el diseño experimental y la manipulación en laboratorio para llevar a cabo la reacción, los datos generados por la qPCR han de ser correctamente procesados y analizados para extraer información biológica relevante que pueda responder a las preguntas planteadas por los investigadores. Este proceso de tratamiento de los datos sigue un flujo de trabajo o pipeline [1].

El fin de este trabajo de fin de máster (TFM) es desarrollar un pipeline para análisis de datos procedentes de qPCR, que se desarrollará utilizando el lenguaje de programación R. La principal razón de esta elección es que es uno de los principales lenguajes utilizados en bioinformática y bioestadística hoy en día, y existen numerosas librerías desarrolladas para procesar datos biológicos. Además, se trata de un lenguaje de código abierto y multiplataforma, lo que ha contribuido a su popularidad.

La elección de esta temática como proyecto de fin de máster me permitirá aprender a manejar datos biológicos de forma personalizada utilizando un lenguaje de programación, habilidad muy solicitada actualmente debido a gran cantidad de datos que están siendo generados en el campo de la biología, en áreas como la genómica, transcriptómica y resto de ómicas. Este proyecto me ayudará también a consolidar conocimientos de bioestadística y bioinformática adquiridos durante el máster y complementar satisfactoriamente mi formación como bióloga.

1.2. Objetivos del Trabajo

1.3. Objetivos generales

Objetivo 1: Diseño y creación de un pipeline para analizar datos de qPCR.

Objetivo 2: Comparar el resultado con otras herramientas ya existentes para el análisis de datos de qPCR.

1.4. Objetivos específicos

Objetivo 1

- Diseño del pipeline, especificando y justificando cada uno de los pasos de los que constará.
- Implementación de cada uno de los pasos del pipeline con las librerías de R.
- Generar un informe automático para visualizar los resultados de una forma más intuitiva.

Objetivo 2

- Ejecución del pipeline utilizando datos de qPCR públicos de una base de datos.
- Evaluación del pipeline, comparando los resultados con otro software para qPCR ya existente, e implementar posibles mejoras.

1.5. Enfoque y método seguido

Se comenzó el proyecto realizando una revisión bibliográfica para conocer las metodologías utilizadas en qPCR, especialmente en el análisis de los datos.

Para implementar el pipeline se utilizó R como lenguaje de prgramación y R-Studio como entorno de trabajo, un entorno ampliamente utilizado en análisis de datos biológicos y de qPCR en concreto ([2]).

La revisión bibliográfica también incluyó la lectura de manuales de los paquetes y librerías de Bioconductor más utilizados en análisis de qPCR: ReadqPCR, EasyqqpcR, HtqPCR, etc.

Durante el desarrollo del código se fueron guardando versiones de éste, por si en algún momento el código fallase, poder rescatar el trabajo en un punto en el que funcionaba.

Una de los subobjetivos propuestos ha sido generar un informe dinámico con los resultados. Para ello he utilizado R-Markdown junto con el paquete knitr, que permite integrar código R con los lenguajes Markdown y Latex.

Inicialmente busqué en bases de datos biológicas (NCBI, EBI, GEO) un set de datos de qPCR para analizar. Finalmente, los datos escogidos fueron un set de datos de expresión de miRNA medida con qPCR, cedidos por el profesor supervisor.

Para evaluar y comparar los resultados finales del pipeline se buscaron otras herramientas de análisis de datos de qPCR ya creadas, con el fin de utilizarlas con el mismo set de datos. La herramienta escogida fue pyQPCR [3], [4].

Durante el semestre he trabajado bajo supervisión y y estableciendo comunicación periódica con el director del trabajo a través del foro y del correo electrónico.

1.6. Planificación del Trabajo

En este apartado se enumeran las etapas de planificación que se expusieron en el plan de trabajo, y que han servido de guía temporal durante el avance del proyecto.

PEC1

- Redacción del plan de trabajo.

PEC2

- Búsqueda bibliográfica sobre análisis de qPCR, artículos, documentación, pipelines documentados y publicados.
- Diseñar un pipeline teniendo en cuenta los siguientes pasos:
    1. Determinar el formato y la lectura de los datos de entrada, entre ellos, el formato estándar RDML.
    2. Fase de control de calidad de los datos crudos.
    3. Elección de un gen normalizador y normalización de los datos.
    4. Fase de control de calidad de los datos normalizados.
    5. Cálculo de la expresión diferencial.
    6. Obtención de los genes con mayor expresión diferencial.

7. Generación de mapas de expresión diferencial (heatmap).
8. Obtención de genes comunes a todos los grupos experimentales.
9. Generación de un informe con los resultados.

- Búsqueda del set de datos con el que se trabajaría durante la elaboración del pipeline.
- Investigación de los paquetes y librerías de R que nos ayuden a realizar cada uno de los pasos.
- Instalación de dichos paquetes e implementación dentro de nuestro pipeline.
- Desarrollar el pipeline en el contexto de un informe que se genere de forma automática con Markdown.

PEC3

- Análisis del set de datos con el pipeline desarrollado, comprobando su correcto funcionamiento.
- Búsqueda de otras herramientas de análisis de qPCR con las que comparar mis resultados (Pabinger, 2014).
- Evaluación del resultado de mi pipeline con los resultados ofrecidos por estas otras herramientas.
- Implementar las posibles mejoras en mi pipeline y cerrar el proyecto.

Memoria final

- Redacción de la memoria final.
- Preparación de la presentación oral del trabajo.
- Exposición oral del trabajo.

De cara a la entrega de esta memoria, la planificación del trabajo ha sido modificada varias veces. Uno de los principales motivos de modificación se ha debido a la inversión de una gran parte del tiempo en la búsqueda bibliográfica y familiarización con los paquetes de Bioconductor. También tomó más tiempo del esperado la búsqueda de unos datos de qPCR adecuados para incorporarlos al *pipeline*. En consecuencia, la planificación resultante se presenta en la figura 1.

1.7. Breve sumario de productos obtenidos

Finalizado el TFM, se han generado los siguientes documentos entregables:

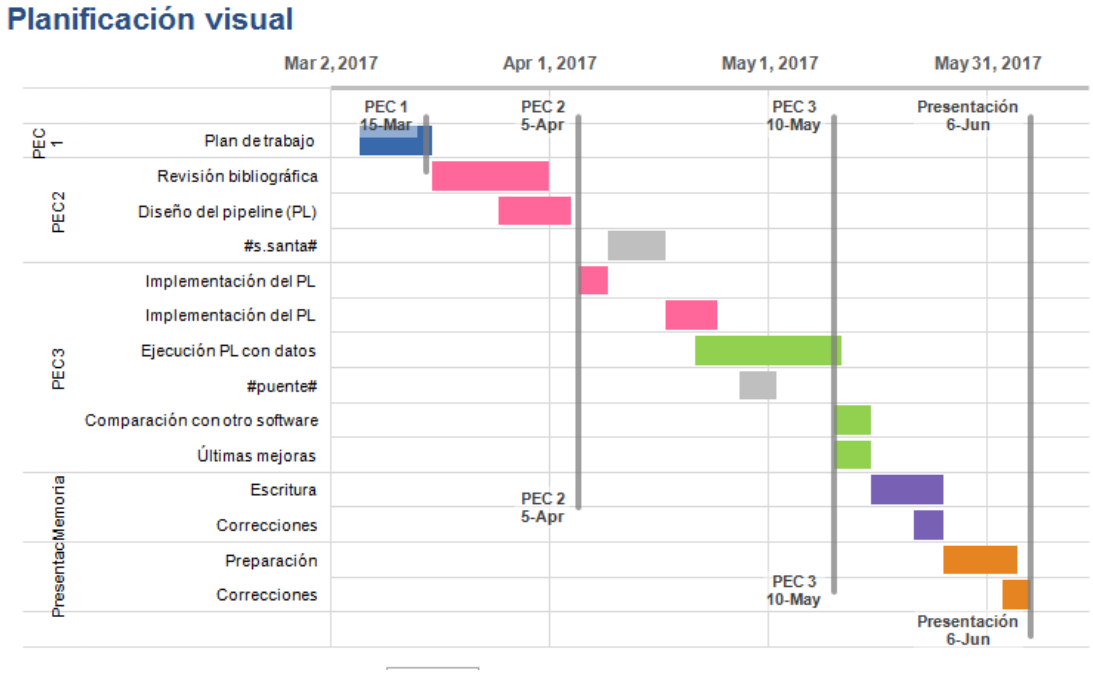

Figura 1: Diagrama de Gannt con la planificación del trabajo.

1. **Plan de trabajo** entregado como PEC1.
2. **Avances del proyecto** entregado como PEC2 y PEC3.
3. **Memoria** escrita que describe el contexto del trabajo, y el pipeline diseñado, y los resultados arrojados.
4. **Pipeline** desarrollado como un documento compilable en forma de informe.
5. **Informe de resultados** generado tras la lectura de los datos de muestra y tras compilar el pipeline. Existen dos versiones de éste: html y pdf.
6. **Presentación virtual** presentación oral del proyecto en formato vídeo.

1.8. Breve descripción de los otros capítulos de la memoria

- En el **capítulo 2: Fundamentos biológicos**, se realizará una introducción a los conceptos biológicos necesarios para entender el objetivo del pipeline. También se hará una descripción del origen biológico de los datos con los que se ha trabajado durante la elaboración del pipeline.
- En el **capítulo 3: Pipeline para análisis de los datos de PCR cuan-**

titativa, se desarrolla el trabajo porpuesto en el objetivo 1 del presente proyecto, y parte del objetivo 2: utilizar el pipeline para analizar unos datos de qPCR. Se explicará el proceso de construcción del pipeline, exponiendo los motivos de cada uno de los pasos implementandos y mostrando los resultados arrojados al aplicarlo al set de datos de prueba.

- En el **capítulo 4: Comparar el resultado con otras herramientas de análisis de qPCR**, se desarrolla el trabajo realizado relacioado con el resto del objetivo 2. Se exploró el software pyQPCR para ver su fucionamiento y para aplicarlo al set de datos de prueba. Sin embargo, no se ha podido generar resultados para compararlos con los obtenidos en el pipeline. Se expondrán las conclusiones obtenidas.

2. Fundamentos biológicos

2.1. ¿Qué es una PCR cuantitativa?

La PCR cuantitativa es hoy es una herramienta básica en la mayoría de laboratorios de biología molecular. La finalidad de esta técnica es cuantificar la concentración de ADN en una muestra, y se ha convertido en un método estándar para medir expresión génica.

Una reacción de **PCR** (Reacción en Cadena de la Polimerasa) consiste en generar más copias de una molécula de ADN específica (amplificación) para su posterior detección/cuantificación. En una reacción de PCR convencional, la cuantificación se realiza al final de la reacción, cuando todo el ADN posible ha sido sintetizado y la reacción llega a su fase de *plateau* tras 40 ciclos aproximadamente (Ver figura 2).

Por contra, la **PCR cuantitativa** o a tiempo real, en adelante **qPCR**, permite medir la concentración o cantidad de ADN de la muestra durante todo el tiempo en el que transcurre la reacción. El ADN, conforme se va sintetizando, emite fluorescencia gracias a un fluoróforo que se adiciona en la mezcla de reacción. La cantidad de florescencia es detectada por el termociclador en cada ciclo de amplificación (ver figura 2, [5]), generándose una curva que indica la cantidad de ADN acumulado en cada ciclo.

En cada ciclo de reacción, el número de moléculas de ADN se multiplica por dos. La qPCR permite contabilizar el ciclo en el que se empieza a detectar ADN en la muestra, denominado **ciclo de cuantificación o Cq** (otra forma muy extendida de denominar a este valor es Ct, pero no está recogida en las directrices MIQE). El valor de Ct es inversamente proporcional a la concentración de cDNA inicial: cuantos más ciclos de amplificación se requieran para la detección de la fluorescencia, menos cantidad de ADN inicial habrá en la muestra. De esta manera se puede determinar con gran precisión la cantidad de ADN inicial de una muestra, comparando la curva generada con la curva generada por una muestra estándar cuya concentración es conocida. Es por este motivo que es una de las herramientas estándar para medir expresión génica.

2.2. Medir la expresión génica

En una qPCR, cuantificar **expresión génica** es sinónimo de determinar el número de copias de ARN de un gen de interés presente en una muestra.

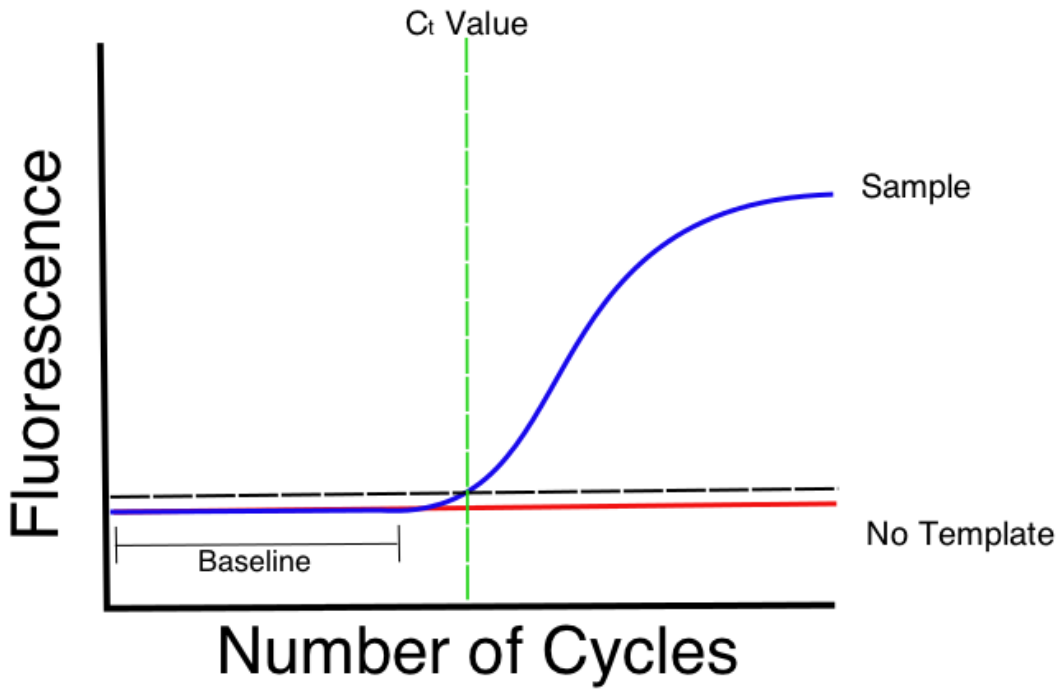

Figura 2: Gráfico de una reacción de qPCR.

En todos los organismos, los genes se encuentran codificados en el ADN, y ejercen su función transcribiendo esta información que albergan a una **molécula de ARN**. El resultado de este proceso es el que se cuantifica cuando se mide expresión génica mediante qPCR.

Así pues, la molécula inicial que se quiere cuantificar no tiene por qué ser ADN, sino que puede ser ARN procedente de la expresión de un gen, que se aísla y se transforma en **ADN complementario (cDNA)** mediante una reacción de **retrotranscripción**. Después, se llevará a cabo la reacción de qPCR normalmente para cuantificar la cantidad inicial de ARN.

A partir de una qPCR se puede obtener una **cuantificación relativa** de la expresión génica, comparando el gen de interés con un **gen de referencia**. De esta manera se puede conocer si el gen o genes de interés se sobreexpresan o inhiben con respecto al gen de referencia, en diferentes condiciones experimentales.

Uno de los pasos más importante a la hora de realizar un experimento con qPCR es la elección de uno o varios genes de referencia que sirva como base para concluir si los genes de interés se sobreexpresan o inhiben. Son buenos candidatos los genes que presenten expresión estable en todas las muestras, y un nivel de expresión

correlacionado con la cantidad total de ARN de la muestra, y que no varíen su expresión entre muestras experimentales. Siempre que sea posible, se escogerán genes que ya hayan sido validados en condiciones similares a las del experimento que vayamos a realizar, y publicados en la bibliografía [6].

2.3. Cuantifiación de miRNAs

La técnica de qPCR es ampliamente utilizada para medir expresión génica a partir del ARN transcrito por el gen. Entre los ARNs cuya expresión se puede cuantificar están no sólo los que se traducen a proteínas o ARNs codificantes, sino también otros ARNs cuya función no es ser traducidos, como los microRNAs [7].

Los **microRNAs o miRNAs** son moléculas de ARN no codificante que actúan, a su vez, como reguladores de la expresión génica de otros genes, regulando multitud de procesos biológicos. Se expresan en una amplia variedad de organismos, entre ellos los humanos. Los miRNA podrían representar como mínimo el 3 % de todos los genes humanos [8]. Funcionan regulando la expresión de otros genes mediante **ribointerferencia**, es decir, uniéndose a otros ARNs, los ARNs mensajeros, y evitando que se traduzcan a proteínas. Ver figura 3, [9].

El interés biomédico de estas moléculas radica en su uso potencial como **biomarcadores**. Se ha comprobado que la expresión génica de los miRNAs se ve alterada alterada en muchos tipos de cánceres humanos y otras enfermedades, ya que funcionan como moléculas supresoras de tumores y otros tipos de regulación [10]. Por ello en la actualidad, el análisis de los perfiles de expresión de miRNAs en humanos tiene aplicaciones potenciales como una nueva técnica no invasiva para diagnóstico y pronóstico de enfermedades como cáncer de vejiga, diabetes, asma ([11], [12], [13]).

En este trabajo, los datos de qPCR con los que se ha trabajado forman parte de un análisis de la expresión de genes de microRNAs de exosomas de orina en humanos.

2.4. El conjunto de datos

Tras llevar a cabo el diseño experimental y la manipulación en laboratorio para llevar a cabo la reacción, los **datos** generados por la qPCR han de ser correctamente procesados y analizados para extraer información biológica relevante que pueda responder a las preguntas planteadas por los investigadores. Este proceso de tratamiento de los datos sigue un flujo de trabajo o pipeline [1] debe llevarse a cabo por un técnico que tenga conocimientos tanto del fundamento biológico del

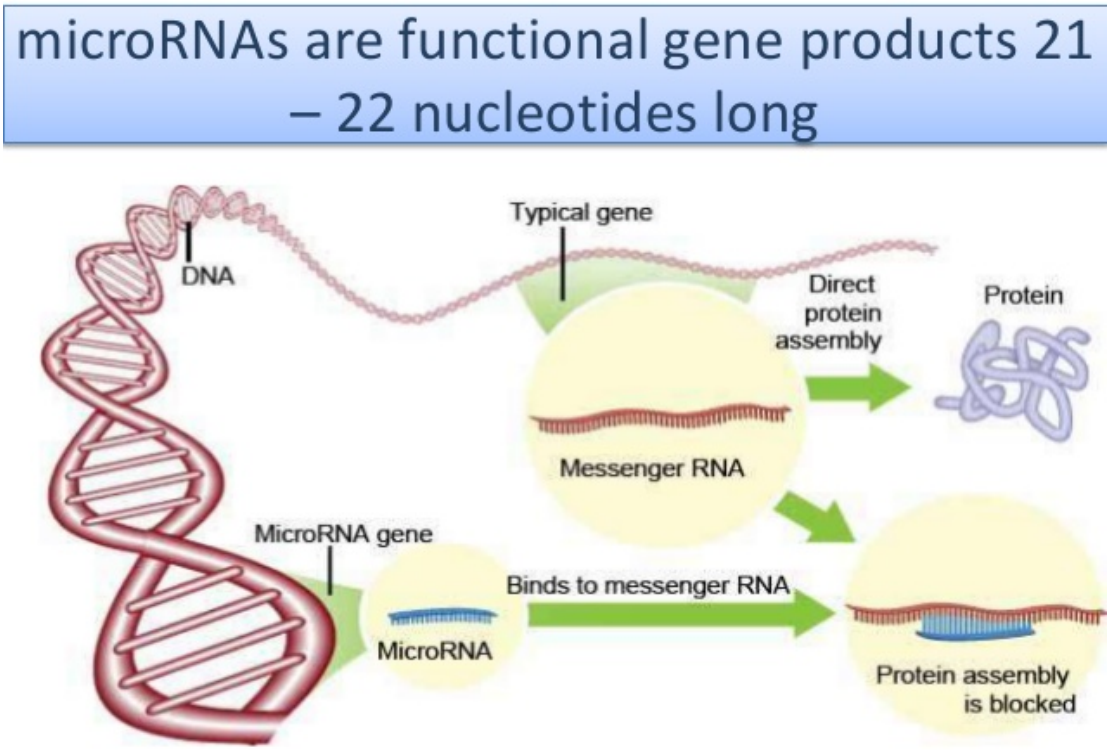

Figura 3: La función reguladora de los miRNAs consiste en la inhibición de la traducción de los ARN mensajeros a proteínas.

experimento como de las herramientas analíticas necesarias para sacar conclusiones de dichos datos.

Para implementar el pipeline era necesario contar con unos datos sobre los que trabajar. Tras investigar varios datos de qPCR públicos, incluyendo los publicados por [14], decidí decantarme por un conjunto de datos cedidos por el profesor supervisor, y que aún no han sido publicados.

Como ya se ha mencionado previamente, los datos de qPCR con los que he trabajado forman parte de un análisis de expresión génica de **microRNA aislados de exosomas de orina**.

Los **exosomas** son vesículas membranosas liberadas por muchos tipos de células al medio extracelular. Actúan como comunicadores intercelulares y regulan procesos biológicos mediante el transporte de biomoléculas funcionales. Pueden aislarse fácilmente a partir de fluidos biológicos como la saliva, la leche materna o la orina, y además contienen en su interior proteínas, lípidos, ARNs mensajeros y miRNAs, por lo que son excelentes candidatos para aislar estos biomarcadores de forma no invasiva [15].

Los datos de muestra utilizados para el presente proyecto forman parte de un análisis de la expresión de genes de microRNA, purificados de exosomas aislados de muestras de orina. El experimento consta de 28 muestras, cada muestra perteneciente a un paciente. Las 28 muestras se dividieron en dos grupos/tratamientos de pacientes (grupo 1 y grupo 2), donde se observaron dos tipos de respuesta (A y B).

Las reacciones de qPCR se llevaron a cabo en placas prediseñadas de qPCR, miRCURY LNA Universal RT microRNA PCR assays (Exiqon), para miRNAs encontrados típicamente en exosomas. Existen modelos de 96 y 384, en este caso nosotros disponemos datos de placas de 96. Estas placas utilizan SYBR Green I como marcador de producto de PCR. La reacción se llevó a cabo en un aparato ABI7000.

Estas placas prediseñadas contienen los cebadores o primers necesarios para amplificar, en cada pocillo de la placa, un tipo de miRNA o un ARN/ADN control. Para llevar a cabo la qPCR sólo es necesario añadir el cDNA de muestra y los componentes de la mezcla de reacción. Los controles presentes en la placa consisten en ARN o ADN sintético adicionado a la muestra durante los procesos de aislamiento del ARN, reversotranscripción a cDNA, y amplificación mediante qPCR, y servirán para monitorizar dichos procesos.

2.5. Preguntas biológicas a responder

El objetivo principal que se busca con el análisis de estos datos, y en función del cual se ha diseñado el pipeline, es el de comparar los perfiles de expresión de miRNAs entre grupos experimentales con el fin de encontrar diferencias en los patrones de expresión.

Los perfiles de expresión de cada uno de los grupos, dos a dos, se compararán mediante un contraste de hipótesis con un test-t. Como resultado, de cada comparación se generará una lista de los **genes más diferencialmente expresados** y su nivel de significación para cada una de las comparaciones.

En nuestros datos de muestra desconocemos qué diferencias experimentales hay entre los grupos 1 y 2, y tampoco sabemos en qué consisten cada una de las respuestas A y B. Al margen de esta información, las comparaciones a realizar por el pipeline serán:

- Respuesta A del grupo 1 vs Respuesta B del grupo 1
- Respuesta A del grupo 2 vs respuesta B del grupo 2
- Respuesta A del grupo 1 vs Respuesta A del grupo 2
- Respuesta B del grupo 1 vs Respuesta B del grupo 2

3. Pipeline para análisis de datos de PCR cuantitativa

3.1. Investigación bibliográfica y paquetes necesarios

Para diseñar el flujo de trabajo, así como para conocer qué paquetes del proyecto bioconductor [16], se recurrió a una consulta bibliográfica en la que las publicaciones más destacadas fueron [4] y [2]. En los pasos en los que han sido aplicables se han seguido las directrices MiQE [6], una serie de guías para estandarizar la publicación de datos de qPCR.

3.1.1. Librerías y preparativos necesarios

Para desarrollar el pipeline instalé el proyecto Bioconductor con el comando:

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

Adicionalmente, también se recurrió a otros paquetes externos a Bioconductor durante el periodo de construcción del pipeline, como por ejemplo el paquete `pander` para crear tablas a partir de `datasets`. Con la función `install.package()` se instalaron conforme se fueron necesitando, y con la función `library()` se llaman al principio del script.

3.1.2. Instalación oculta de paquetes

En el documento Rmd he implementado una parte del código que instalará el/los paquetes de R y bioconductor necesarios para ejecutar el pipeline, en el caso de que el usuario no los tuviese instalados. El código utilizado para ello se encuentra en el Anexo y en el archivo Rmd entregado.

3.1.3. Diseño del pipeline

Con ayuda de los pasos indicados por el profesor supervisor y leyendo bibliografía específica [2,4,17] diseñé un diagrama de flujo del *pipeline*. Este esquema me ayudó a seguir una dirección de trabajo y a visualizar qué apartados y subapartados deberían mostrarse en el informe final. Finalmente, debido al set de datos usado como muestra, el resultado ha sido un pipeline específico para leer datos de Cq de placas prediseñadas de qPCR miRCURY LNAT (Exiqon), medidas en un

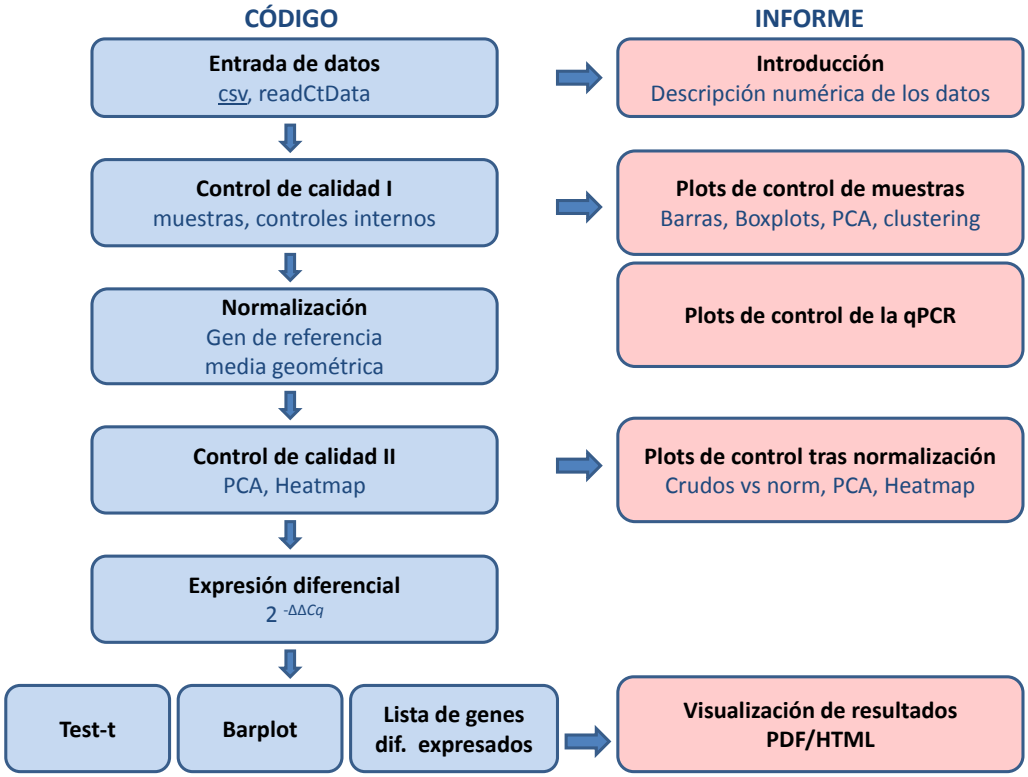

Figura 4: Diagrama de flujo del pipeline. Azul: código; Rosa: resultados.

termociclador ABI PRISM 7000 Sequence Detection System. Sin embargo, con ligeras modificaciones puede adaptarse a nuevos datos con el mismo formato (otros modelos de placas de miRNA de Exiqon).

Dicho pipeline ha sufrido diversas modificaciones hasta llegar a la versión final, mostrada en la figura 4. En el diagrama he indicado cada parte de un color en función de si se trata de código a desarrollar (azul) o si se trata de parte de los resultados (rosa).

3.2. Lectura de los datos

Los datos de muestra utilizados para el presente proyecto forman parte de un análisis de la expresión de genes de microRNA, purificados de exosomas aislados de muestras de orina. El experimento consta de 28 muestras. Cada muestra pertenece a un paciente y se ha medido en una placa de 96 pocillos. Tras la reacción de qPCR cada muestra generó un archivo de 96 medidas de valores Cq. Como ejemplo de

cómo debe ser el formato de lectura, las primeras lineas de uno de estos archivos (`sample01.csv`) se muestran en la tabla 1.

Tabla 1: Formato del archivo de texto que debe generar cada placa.

Detector	Reporter	Start	End	Threshold
1		25.7754	NA	NA
2		26.1430	NA	NA
3		32.7190	NA	NA
4		28.1366	NA	NA
5		17.7910	NA	NA
6		27.1528	NA	NA

Para la lectura de los datos usamos el paquete `HTqPCR` ([17]). Este paquete nos permite leer multitud de formatos generados por equipos de distintas casas comerciales, así como formatos de texto plano. El código de lectura de los datos se adjunta en el Anexo.

Para poder leer nuestros datos con la función `readCtData` hubo que crear de forma manual un fichero adicional llamado `Genesinfo.txt` con la información sobre el nombre del gen o el control que se evalúa en cada pocillo. Para más información sobre el formato de este archivo ver la tabla 5.

Las 28 muestras se dividen en dos grupos/tratamientos de pacientes (grupo 1 y grupo 2), donde se observaron dos tipos de respuesta (A y B). Con el fin de introducir estos datos durante la lectura de los valores Cq, hubo que crear manualmente otro fichero adicional al que se denominó `samplesinfo.txt`. Este fichero relaciona el nombre del fichero de cada muestra con las condiciones experimentales correspondientes. Para más detalles sobre el formato, ver la tabla 4.

Durante la lectura de los datos, puede suceder que falten datos de Cq para algunas medidas. Cuando se da el caso de que faltan algunos valores, para poder continuar con el análisis se recomienda sustituir estos valores ausentes por un valor numérico elevado, como por ejemplo 40 [17]. Si el umbral de detección está en el ciclo número 40 se puede considerar que no hay amplificación de ARN que detectar, y la expresión será por tanto cercana a 0. En el pipeline desarrollado los valores

NA se sustituyen por valores de 40 de forma automática, aunque este valor es modificable por el usuario.

3.3. Fase de control de calidad de los datos crudos

Una vez tenemos nuestro dataset leído de forma adecuada, hemos de hacer una exploración de los valores de Cq. Para ello se pedirá de forma inicial un resumen estadístico de los valores Cq de cada muestra. En la tabla 2 aparece una muestra del resumen generado. Esta tabla será uno de los primeros resultados generados por el pipeline, y se guardará automáticamente en un archivo de texto denominado `Results_raw_summary.csv`, en la misma carpeta donde el usuario se encuentre ejecutando el pipeline.

Tabla 2: Resumen de los datos crudos de expresión de cada muestra.

X	Min....	X1st.Qu.	Median.	Mean...	X3rd.Qu.	Max....
sample01	15.7	22.7	27.2	26.7	30.2	41.6
sample02	16.1	25.4	28.7	29	32.1	41.6
sample03	16.3	24.8	27.3	27.2	29.2	40
sample05	15.6	24.8	26.9	26.8	28.6	41.3
sample06	15.8	20.4	24.4	24.2	27.2	40

Tras el resumen numérico se decidió incluir gráficos preliminares de los valores de expresión Cq en crudo de cada uno de los genes/controles medidos, así como de los valores y la distribución de expresión de cada muestra. Estos gráficos se generaron con las funciones `plotCtOverview` y `plotCtBoxes` implementadas en el paquete `HTqPCR`. Además, para evaluar si existe agrupación las muestras por grupos experimentales se han generado los gráficos de jerarquía y de análisis de componentes principales (PCA) con las funciones `clusterCt()` y `plotCtPCA` del mismo paquete. Estos gráficos se guardarán automáticamente en la carpeta de ejecución del usuario.

A continuación se muestran los resultados preliminares obtenidos al ejecutar el control de calidad del pipeline sobre los datos de muestra (figuras 5, 6 y 7).

En esta etapa de exploración y de control de calidad de los datos, estos gráficos

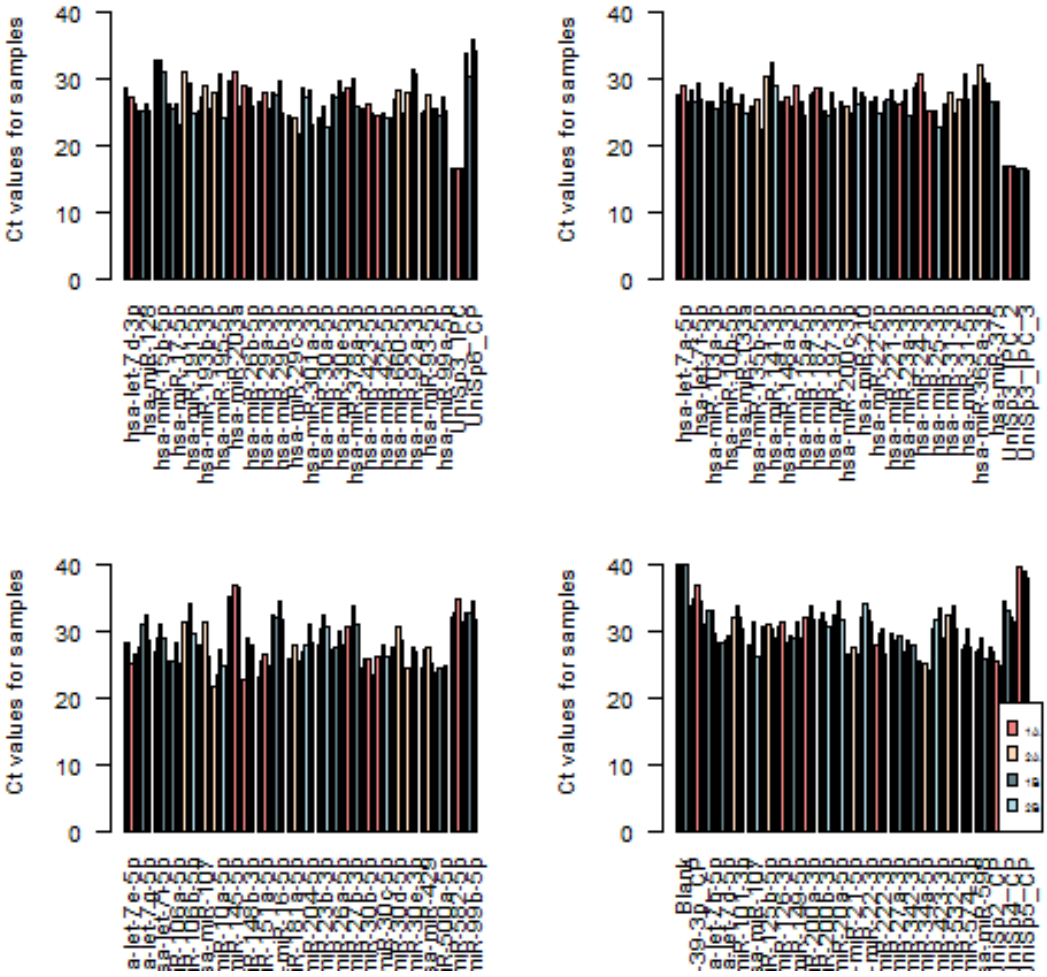

Figura 5: Valores Cq en crudo de cada uno de los genes/controles.

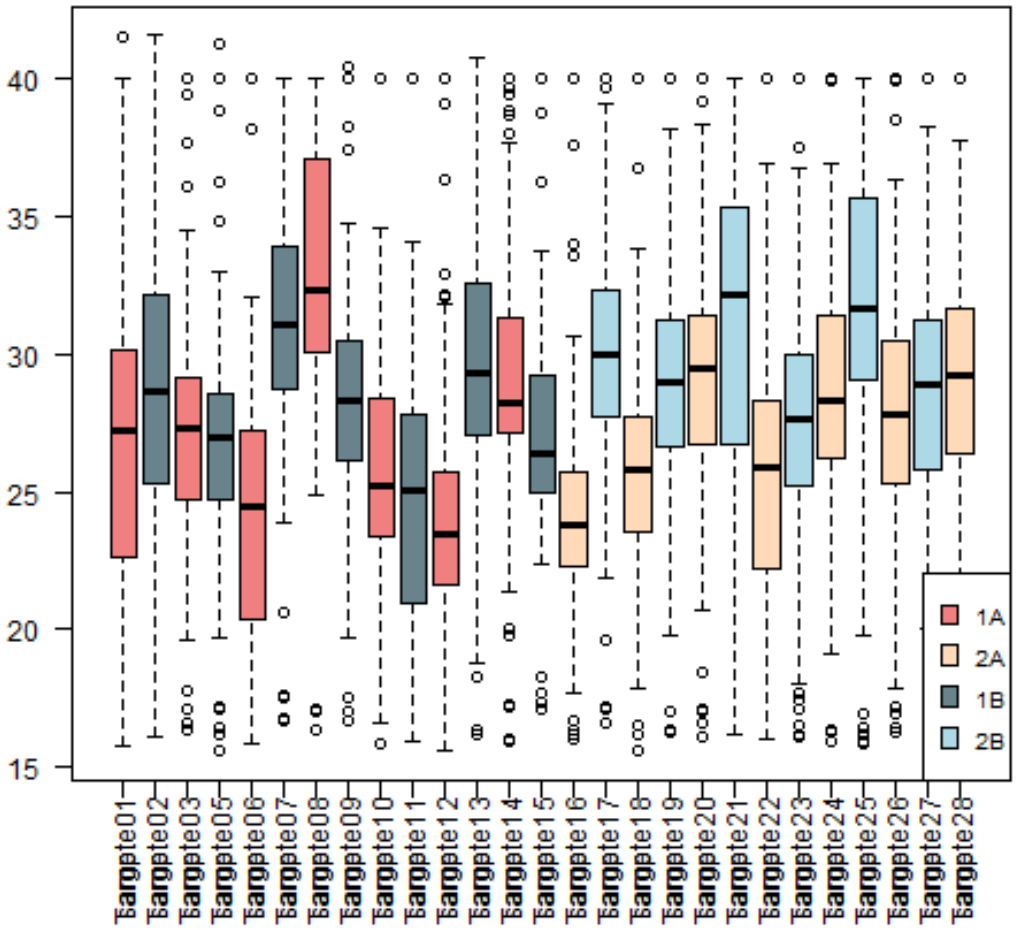

Figura 6: Distribución de valores Cq en crudo de cada una de las muestras.

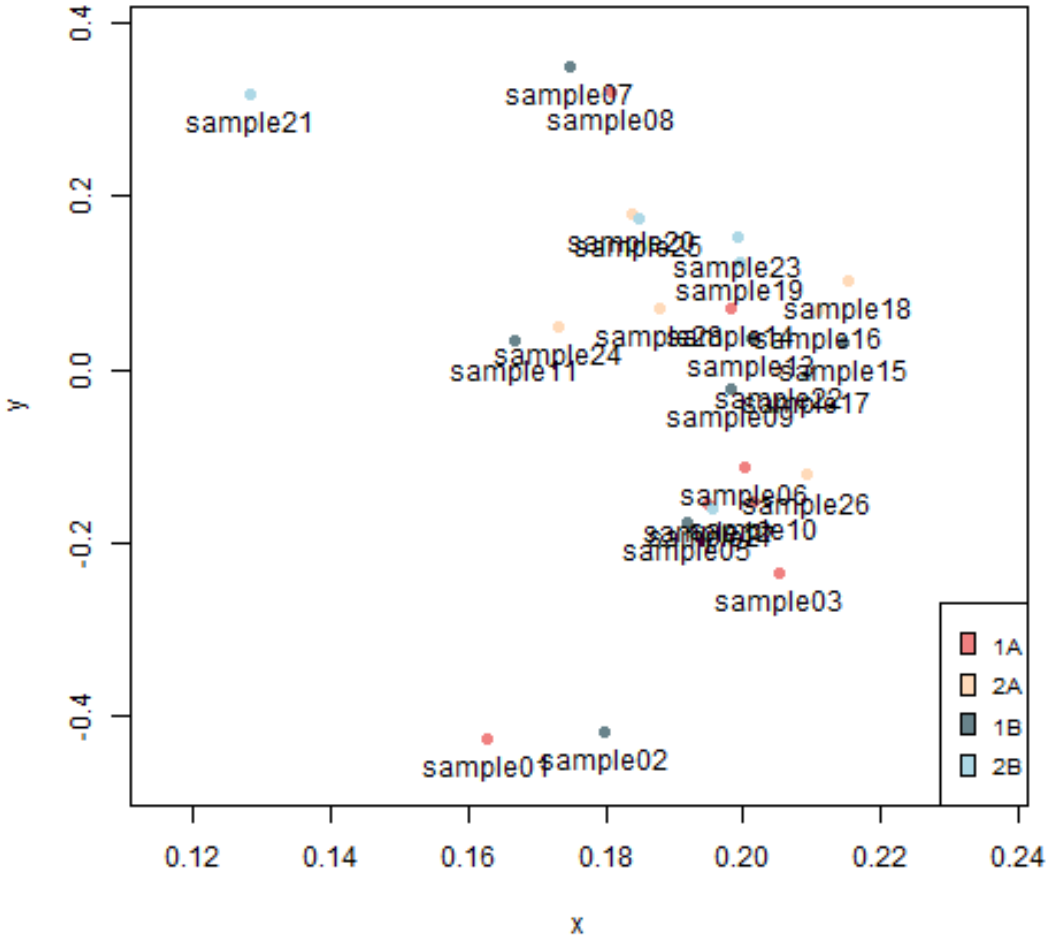

Figura 7: Principales fuentes de variación (PCA) en las muestras.

ayudan a observar si existen muestras o genes con comportamiento anómalo, de una forma rápida. Por ejemplo, en nuestros datos de muestra, deberíamos decidir si descartar la muestra 4, que posee unos valores de Cq muy diferentes al resto y que no se agrupa con el resto de su mismo grupo experimental en el gráfico PCA.

Adicionalmente, en esta etapa del pipeline se podría haber incluido, de forma complementaria, un análisis de las curvas de amplificación y de disociación (raw y derivada) a partir de los datos crudos de fluorescencia. Con esos datos se podrían delimitar el nivel de ruido de fondo (baseline) y el umbral de detección. Se hará el cálculo de Cq (ciclo umbral) y la eficiencia de la reacción de qPCR. Sin embargo, con el formato de entrada de nuestros datos no estaban disponibles estos datos crudos, con lo cual este paso se descartó.

3.3.1. Control de calidad de la reacción de qPCR

El protocolo para qPCR miRCURY LNAT (Exiqon) incluye una serie de ARNs y ADNs control que se adicionan a la muestra en diferentes etapas de la preparación de la qPCR para monitorizar los procesos de aislamiento del ARN, reversotranscripción a cDNA, y rendimiento de la reacción de amplificación.

Estos controles, además, pueden servir para identificar y excluir muestras atípicas o 'outliers'. En esta parte del pipeline se ha implementado código que muestra los Cq medidos a partir de estos controles.

Las placas prediseñadas qPCR miRCURY LNAT (Exiqon) contienen ya los cebadores para amplificar estos controles. En este caso el código está preparado para medir placas miRCURY LNA Universal RT microRNA PCR assays (Exiqon), pero la posición de estos controles en la placa varía según el modelo. Por eso, según el modelo de placa utilizado, en esta etapa el usuario habrá de introducir qué pocillos contienen cada uno de estos controles, siguiendo las plantillas de la página de Exiqon..

A continuación se muestra el código utilizado para señalar la posición de los controles de la placa, y que puede ser modificable por el usuario antes de compilar el archivo Rmd.

```
# MODIFICABLE POR EL USUARIO SI ES NECESARIO
# Asignar posiciones de los controles en la placa:
UniSp3 <- c(12, 36, 48)
UniSp6 <- 24
CelmiR39 <- 92
UniSp2 <- 93
UniSp4 <- 94
```

```
UniSp5 <- 95
Blank <- 96
```

3.3.1.1. Control del proceso de extracción de ARN

Los controles UniSp2, UniSp4 y UniSp5 aparecen en los pocillos de 93 al 95. Es ARN que añade a la muestra previamente a la extracción de ARN total, y sirven para monitorizar dicho proceso. Cada uno de los controles tiene una concentración final 100 veces menor que el anterior, siendo el más concentrado UniSp2 y UniSp5 el más diluido. Tras la reacción de PCR se espera que las diferencias en Cq entre dichos controles sea de 5-7 Cq.

En el pipeline se ha implementado una parte de código que muestra el resultado de amplificación de estos controles, en forma de gráfica,que se guardará como `Results_plotCt_unisp245.png`. En la figura 8 se muestra el gráfico generado por nuestros datos de muestra.

3.3.1.2. Control de síntesis de cDNA

Un segundo control consiste en la adición de los ARNs UniSp6 y cel-39-3p tras el aislamiento de ARN y previamente a la síntesis de cDNA. Es una manera de diagnosticar problemas en la reacción de retrotranscripción. El pipeline también incluye una parte de código que genera un gráfico para visualizar estos controles, y será almacenado como `Results_plotCt_unisp6CelmiR39.png`. En la figura 9 se muestra dicho gráfico.

3.3.1.3. Control de eficiencia de la qPCR

Este control consiste en la adición de ADN junto al cDNA muestral, en un paso previo a la reacción de qPCR. Permite distinguir resultados negativos verdaderos de falsos resultados negativos causados por un mal funcionamiento de la reacción de PCR. Esos controles permiten también comparar entre placas, ya que la cantidad de ADN amplificado debe ser el mismo en todas las muestras. Si todo ha ido bien, los valores Cq deberían ser muy similares dentro de la misma placa y especialmente entre placas.

En esta parte del pipeline se ha implementado también la generación de un gráfico con los valores Cq de estos controles, marcados con el nombre UniSp3. A continuación (fig. 10)se muestran los niveles de amplificación de estos controles en nuestros datos de muestra.este gráfico generado automáticamente se guardará con el nombre `Results_plotCt_Uni3.png`.

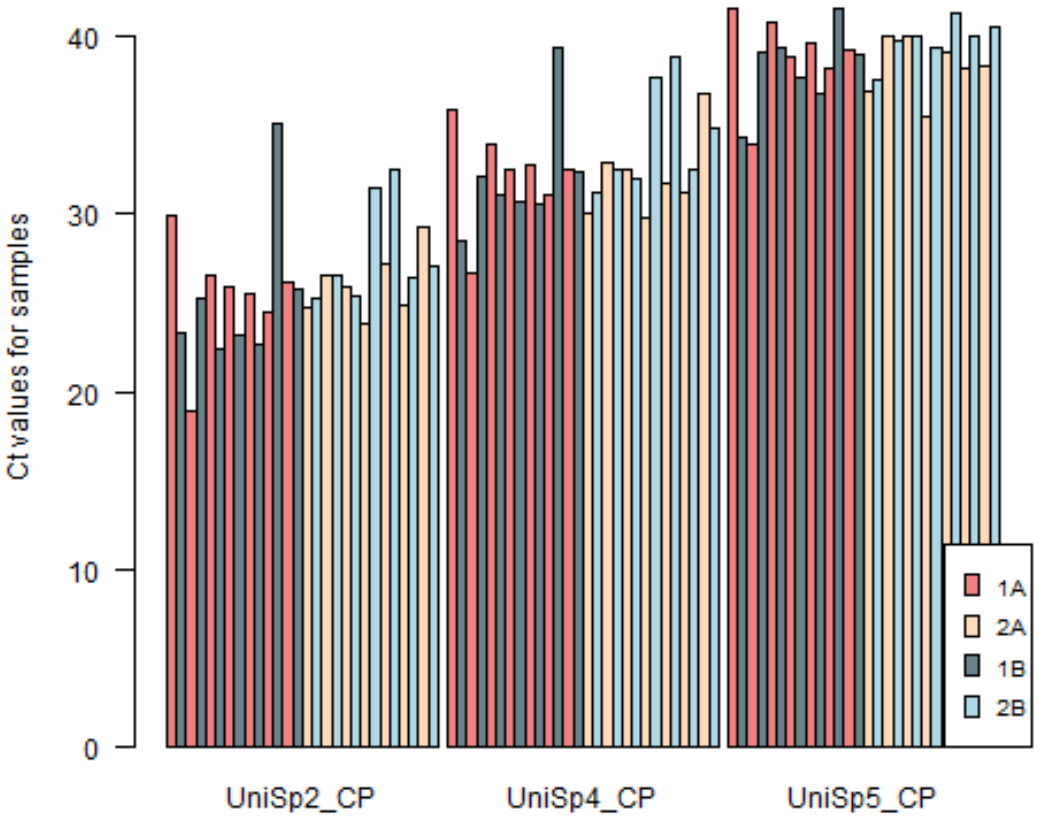

Figura 8: Controles UniSp2, UniSp4 and UniSp5 en los datos de muestra. Observamos que las diferencias en Cq entre dichos controles es alrededor de 5 Cq, como es de esperar.

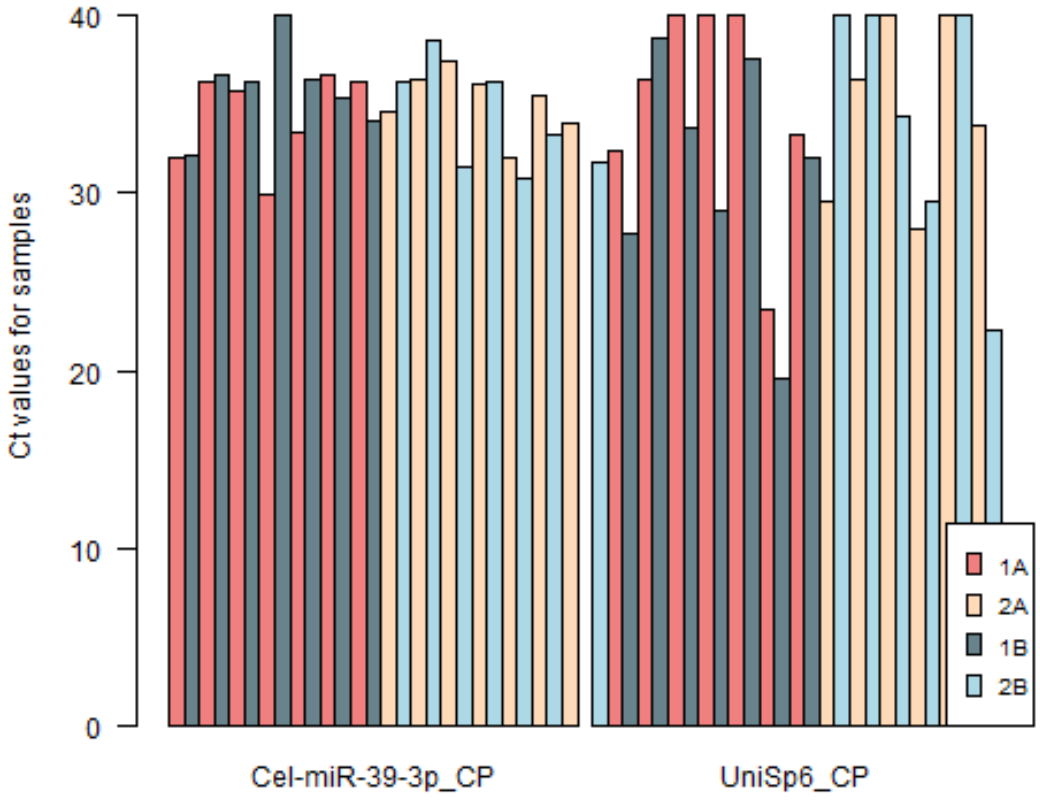

Figura 9: Controles UniSp6 y cel-39-3p, que demuestran que la reacción de retro-transcripción ha sido exitosa.

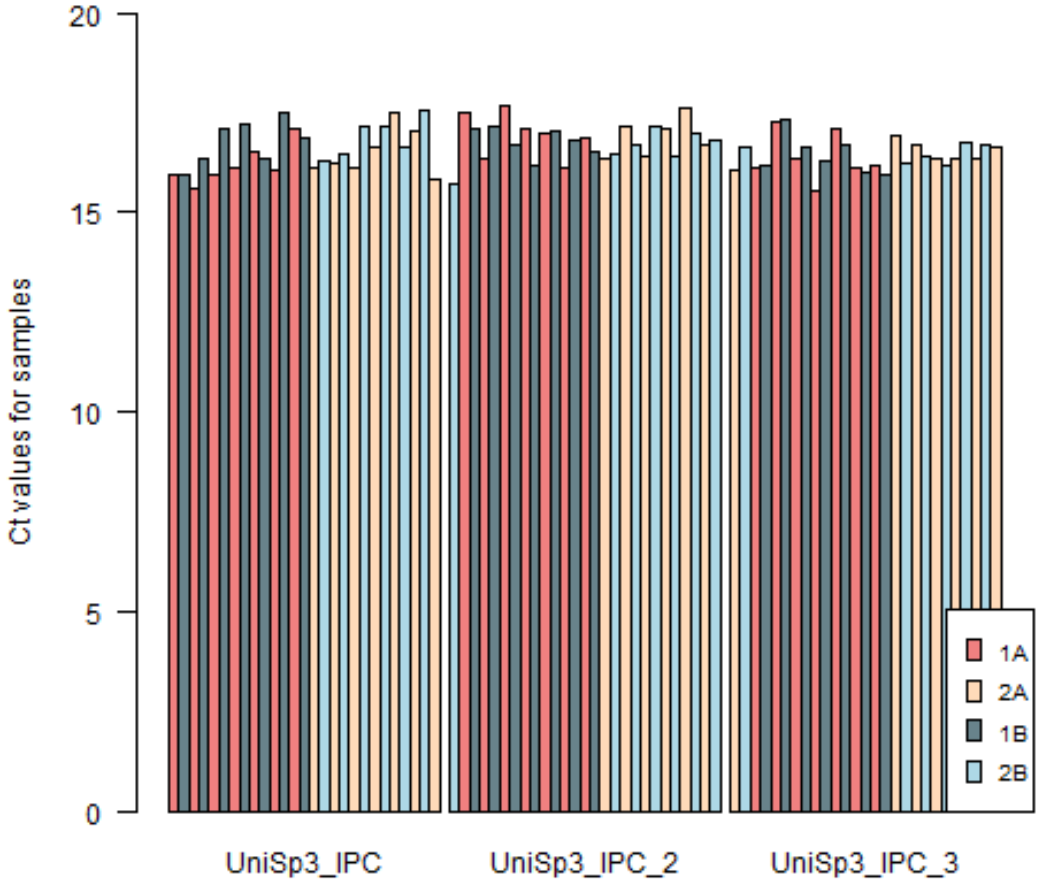

Figura 10: Controles UniSp3 de los datos de muestra, que deben encontrarse todos alrededor del mismo Cq.

3.4. Fase de Filtrado y Normalización

3.4.1. Pretratamiento y filtrado

En esta etapa del pipeline se incluye un paso de filtrado para asignar categorías a las medidas Cq en función de su calidad o fiabilidad. Las tres categorías posibles son 'Ok', 'Unreliable' o 'Undetermined', y son asignadas con la función `setCategory` del paquete `HTqPCR`.

En el pipeline, el límite máximo está establecido en 35 por defecto. Las medidas que superen el número máximo de ciclos serán catalogadas como 'Undetermined', y serán aquellas para las que no se ha detectado expresión. En esta categoría se incluirán los valores NA, que previamente el usuario habrá podido establecer en un valor elevado como 40 Cq.

El límite mínimo está establecido en 10 por defecto. Las medidas que queden por debajo del número máximo de ciclos serán catalogadas como 'Unreliable', y serán aquellas cuyo umbral de detección Cq esté por encima de un umbral de corte que puede ser especificado por el usuario.

En este pipeline el usuario puede establecer ambos límites en el trozo de código previsto para ello, que se reproduce a continuación:

```
# MODIFICABLE POR EL USUARIO
Cqmax <- 35
Cqmin <- 10
```

Tras la clasificación en categorías, se genera un gráfico en el que se podrá ver la proporción de muestras de cada una de las categorías. Dicho gráfico también se puede usar de guía para descartar una muestra poco fiable. Por ejemplo, en nuestro set de datos de muestra, parecería buena idea descartar la muestra 4 como habíamos señalado anteriormente (ver figura 11). Este gráfico se guardará automáticamente en `Results_categories.png`.

3.4.2. Elección de un gen normalizador

Uno de los pasos más importante a la hora de procesar los resultados de una qPCR es la elección de **uno o varios genes de referencia** respecto a los que normalizar los datos. Los datos de qPCR se pueden **cuantificar de forma absoluta** para averiguar la cantidad de ARN inicial en la muestra. Requiere que la eficiencia de amplificación sea igual para todas las muestras. En nuestro caso nos interesa

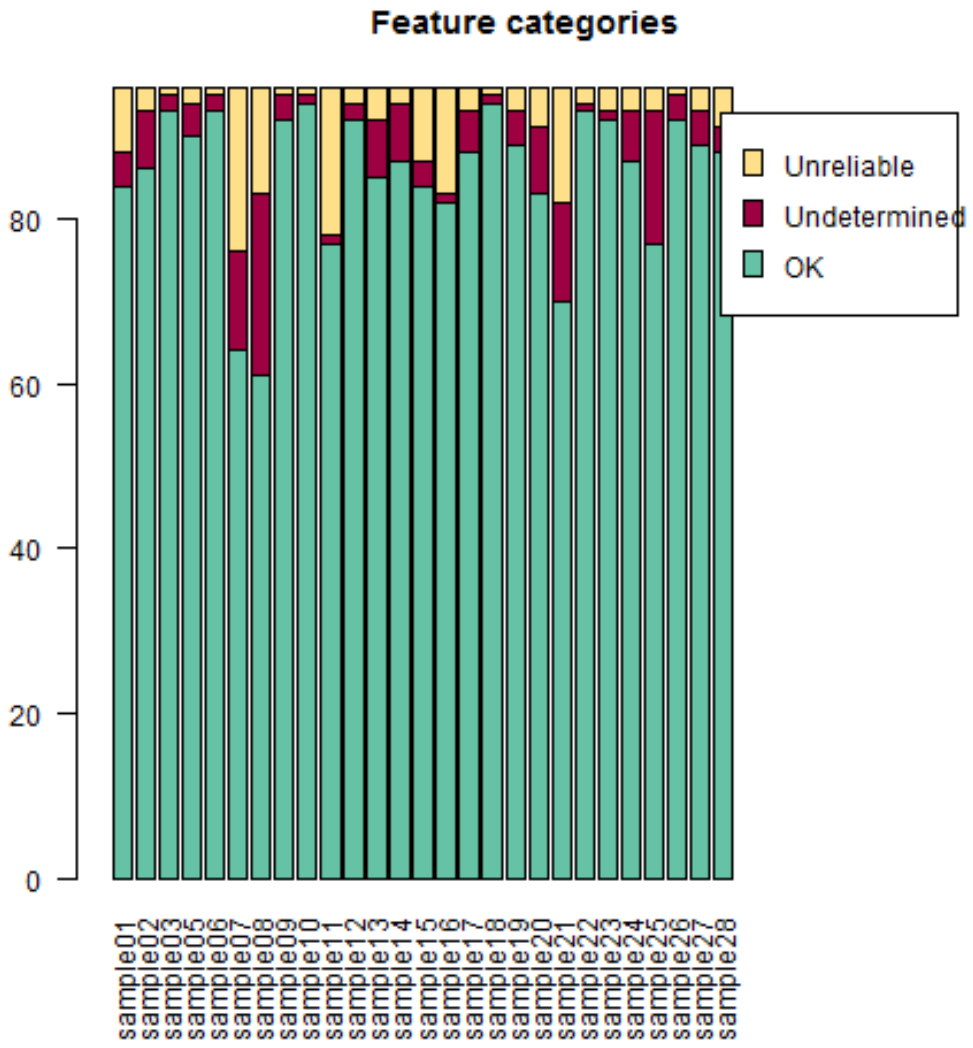

Figura 11: Proporción de las categorías 'Ok', 'Unreliable' o 'Undetermined' en cada una de las muestras. EN este caso la muestra 4 contiene tantas medidas 'Unreliable' que podríamos descartarla.

realizar una **cuantificación relativa** para analizar expresión diferencial. En este caso, es necesario hacer una comparación de las medidas Cq con respecto a otras.

El **gen normalizador** es un gen cuya expresión Cq sirve de base para calcular la expresión relativa del resto de genes a analizar. Normalmente, habrá sido escogido por el usuario durante el diseño del experimento. En tal caso, la elección debe estar debidamente justificada y dicho gen deberá estar validado experimentalmente como un buen gen de referencia [6].

En el caso de nuestros datos de muestra no se disponía de un gen de referencia establecido. En los casos en los que no se dispone de un gen validado experimentalmente, son buenos candidatos a escoger los genes que presenten expresión estable en todas las muestras y condiciones experimentales y un nivel de expresión correlacionado con la cantidad total de ARN de la muestra. La normalización respecto a **un único gen de referencia** es el método más común, pero no es el más recomendado por la normativa MiQE, a menos que ese gen esté completamente validado. Hay métodos que incluyen varios genes de referencia y otros que no incluyen ninguno ([18]).

En este pipeline se ha implementado de forma manual un código que calcula de manera automática los genes que menos variación presentan entre muestras, es decir, cuya **desviación estándar relativa es menor**. El usuario deberá escoger cuántos genes normalizadores quiere encontrar. Por defecto, en este pipeline se han escogido los 3 con menor variación, aunque el usuario es libre de modificar este número. El código que contiene los cálculos y sirve para escoger estos genes se muestra en el Anexo.

Finalmente, el pipeline genera una lista donde se muestran los 10 primeros genes que menos varían en su expresión entre diferentes muestras o placas. Lo normal es que aparezcan de los controles que se utilizan para evaluar la calidad de la qPCR. No es adecuado utilizarlos como genes como genes normalizadores, pues provienen de ARN que no formaba parte de la muestra original, y por tanto deben descartarse. Este descarte también se ha implementado de forma automática en el código.

3.4.3. Normalización

Normalizando los niveles de fluorescencia con respecto a una muestra eliminaremos la variabilidad introducida durante los procesos de extracción del ARN, transcripción y eficiencia de amplificación, que nos enmascararían las diferencias experimentales reales. Con los datos normalizados ya podremos hacer comparaciones entre muestras.

Ya hemos mencionado previamente que la normalización respecto a un gen de referencia es el método más común, pero existen otros métodos que no incluyen genes de referencia [18], [17]. En el siguiente paso del análisis el usuario debe escoger un método de normalización para hacer comparables los valores de expresión entre genes y muestras. La normalización se llevará a cabo con la función `normalizeCtData` del paquete `HTqPCR`.

3.4.3.1. Método 1: Genes normalizadores

Si el usuario escoge este método de normalización, el pipeline calculará la diferencia relativa en el ciclo de cuantificación, ΔCq, según describieron Livak y Schmittgen [19]. El argumento `deltaCt` del paquete `HTqPCR` se utilizará para calcular el nivel de expresión promedio (Cq) de los genes de referencia escogidos en el apartado anterior, y calcula la diferencia de este promedio con respecto al resto de valores de expresión génica.

3.4.3.2. Método 2: Media geométrica

Otro de los métodos disponibles en el pipeline implementado es la normalización por media geométrica, disponible en el paquete `HTqPCR`. El argumento `geometric.mean` de la función normalizadora calcula el valor Cq promedio para cada muestra, y reescala todos los valores Cq de las muestras respecto al promedio de una de ellas. Por defecto será respecto a la muestra número 1. Algunos estudios han demostrado que este método es el idóneo para normalizar expresión de miRNA [20].

A continuación, aparecerá un fragmento de código en el que el usuario deberá especificar qué método de normalización prefiere. Por defecto se aplicará el método ΔCq. El código de normalización se muestra en el Anexo.

```
# MODIFICABLE POR EL USUARIO
tipo_de_normalizacion <- "deltaCt"
# opciones: "geometric.mean", "deltaCt"
```

Tras la etapa de normalización se han implementado la generación de gráficos para comparar los valores Cq antes y después de normalizar. En el siguiente gráfico (figura 12) se muestran los valores Cq crudos (x) y normalizados (y) de nuestros datos de muestra, utilizando el método del Δ Cq respecto a 3 genes normalizadores. Se guardarán automáticamente en el archivo `Results_norm_vs_raw.png`.

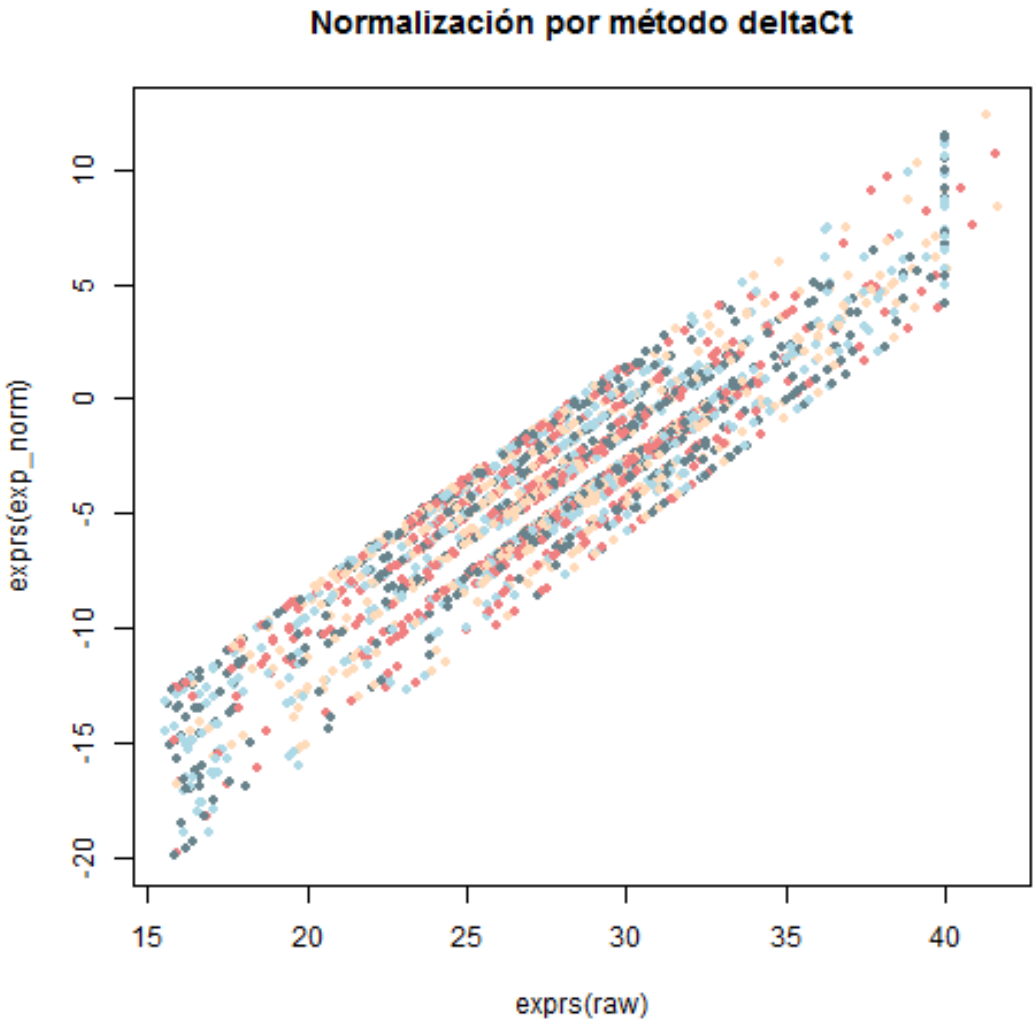

Figura 12: Valores Cq crudos (x) y normalizados (y) de los datos de muestra, utilizando el método del Δ Cq respecto a 3 genes normalizadores.

3.4.4. Filtrado de datos no experimentales

Como última última fase de la etapa de normalización y filtrado, se eliminarán los datos marcados como 'Unreliable' y/o 'Undetermined'. También, de forma automática, se eliminan las medidas de miRNA con más de 3 medidas marcadas como "Unreliable". El usuario puede cambiar este límite en el siguiente código.

```
#MODIFICABLE POR EL USUARIO
excluir <- 3
```

Los controles que se adicionan a la muestra y que permiten monitorizar los procesos de extracción y amplificación, una vez se ha comprobado que dichos procesos han ido bien, no aportan ninguna información sobre la expresión génica de las muestras, y por tanto también se habrán de excluir. La función utilizada para filtrar es `filterCtData`, y el código de filtrado se muestra en el Anexo.

3.5. Fase de control de calidad de los datos normalizados

En esta fase del pipeline se generarán de forma automática un par de gráficos, *heatmap* y PCA, para comprobar de nuevo la agrupación de las muestras una vez normalizados los valores Cq. De nuevo, esta es una fase donde podremos decidir si descartamos alguna muestra que se comporte de forma anómala o distinta a las demás.

En el *heatmap* generado para nuestros datos de muestra (figura 13) tenemos un resumen visual de cómo se agrupan las muestras experimentales (horizontal), así como de los valores de expresión relativa de cada una de los genes (vertical) en cada una de las muestras. Este gráfico se genera con la función `plotCtHeatmap` y se guardará como `Results_heatmap.png`.

Asimismo, también se generará un gráfico PCA con los datos normalizados, mostrando las dos principales fuentes de variación entre muestras (x e y). Si los puntos o muestras aparecen formando agrupaciones es indicativo de que existirán diferencias experimentales entre dichas muestras. El gráfico almacenará automáticamente como `Results_norm_PCA.png`. En la figura 14 se muestra un ejemplo generado con los datos de muestra.

3.6. Fase final: Resultados obtenidos

En este último apartado se calculará la expresión génica relativa entre los grupos experimentales presentes en el set de datos. Este es el apartado menos dinámico

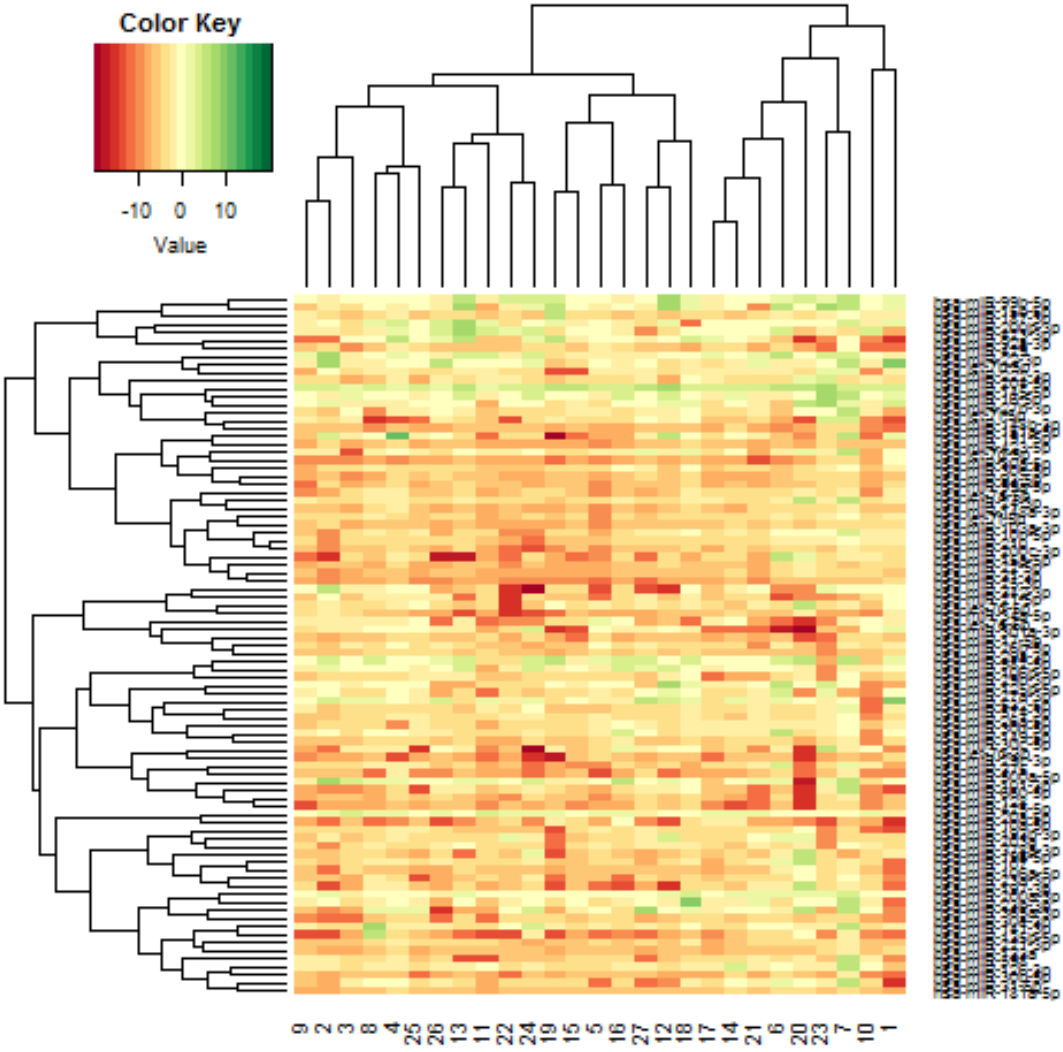

Figura 13: Heatmap de los datos normalizados y filtrados

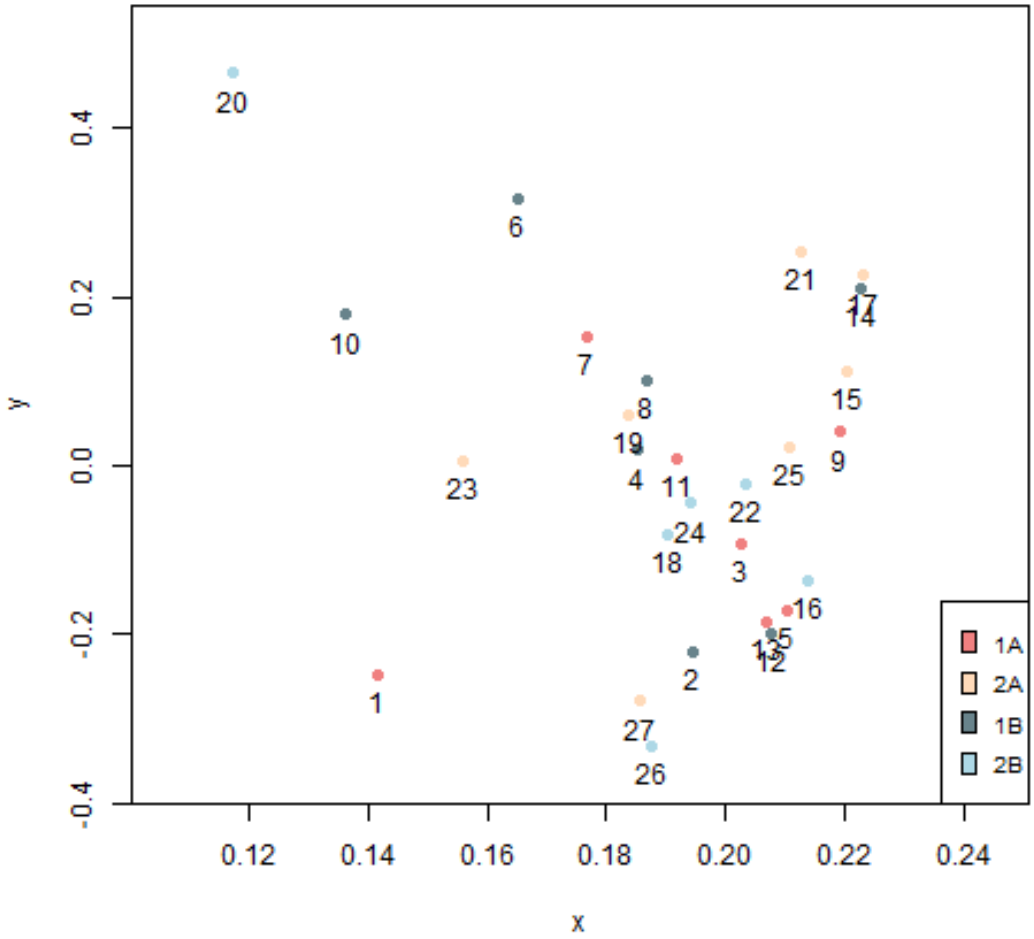

Figura 14: PCA de los datos de muestra normalizados y filtrados, donde no se observa agrupación de las muestras por categorías experimentales.

del informe, pues se ha adaptado a los 4 grupos experimentales que existían en nuestros datos de muestra.

En este apartado se han realizado contrastes 2 a 2 de cada uno de los grupos experimentales mediante un test t. El usuario podrá escoger el nivel de significación que deben alcanzar los genes diferencialmente expresados, que por defecto está establecido en α=0,05, en el siguiente código:

```
# MODIFICABLE POR EL USUARIO
sig <- 0.05
```

El test-t determina si dos poblaciones varían en su promedio. El p-valor obtenido es un indicador de si debemos rechazar o no la hipótesis nula (las poblaciones son iguales). El p-valor de corte se calculará ajustado por el método de Benjamini y Hochberg ([21]), debido a las comparaciones múltiples que se realizarán. Para realizar el test-t, para cada una de las comparaciones se utilizó la función `ttestCtData`. Una vez realizado el test, la tabla de resultados obtenida se guardará en un fichero de texto csv. A continuación se muestra un ejemplo del código usado. El código para cada test se muestra en el Anexo.

```
DE.1A.1B.ttest <- ttestCtData(exp_Filt[,1:14], groups = targets$Group[1:14])
write.csv(DE.1A.1B.ttest, file = "Results_DE_1Avs1B.csv")
```

La tabla de resultados generada por este código presenta el formato siguiente (ver tabla 3).

Tabla 3: Ejemplo de tabla de resultados del test-t entre dos grupos experimentales.

X	genes	p.value	adj.p.value	ddCt	categoryTarget
27	hsa-miR-15a-5p	0.01085	0.8388	-8.151	OK
24	hsa-miR-148b-3p	0.02941	0.8388	6.314	OK
26	hsa-miR-151a-5p	0.03425	0.8388	2.59	OK
35	hsa-miR-195-5p	0.03857	0.8388	-2.973	OK
46	hsa-miR-22-5p	0.08613	0.9404	-1.34	OK
52	hsa-miR-25-3p	0.09158	0.9404	-1.701	OK

X	genes	p.value	adj.p.value	ddCt	categoryTarget
45	hsa-miR-22-3p	0.1167	0.9404	2.498	OK
1	hsa-let-7a-5p	0.12	0.9404	-2.798	OK
77	hsa-miR-429	0.129	0.9404	2.929	OK
7	hsa-let-7g-5p	0.1355	0.9404	3.736	OK
20	hsa-miR-135b-5p	0.1413	0.9404	-4.544	OK
40	hsa-miR-203a	0.1577	0.9404	-2.174	OK
44	hsa-miR-210	0.1616	0.9404	-2.147	OK
3	hsa-let-7d-3p	0.176	0.9404	-3.139	OK
67	hsa-miR-31-3p	0.2189	0.9404	-1.288	OK

Finalmente, como último resultado del pipeline, para cada gen se mostrarán los 15 genes más diferencialmente expresados en un gráfico, que se guardará automáticamente. Los gráficos generados para el set de datos de muestra se presentan en las figuras 15-18. Un ejemplo del código para generar cada gráfico aparece a continuación:

```
plotCtRQ(DE.1A.1B.ttest, genes = 1:15)
```

Finalmente, en el pipeline se redactan los resultados de test en un texto dinámico, en el que se expone el número de genes con expresión diferencial significativa encontrados. En nuestro set de datos, ninguna de las comparaciones entre grupos ha resultado ser significativa. Este resultado ya se podía predecir viendo que en el gráfico heatmap (fig. 13) las muestras no presentaban patrones muy diferentes de expresión, y tampoco se agrupaban por categorías experimentales en el gráfico del PCA (fig. 14).

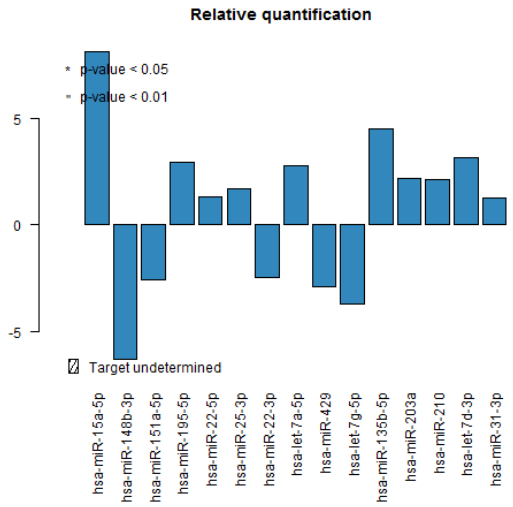

Figura 15: Expresión diferencial 1A vs 1B.

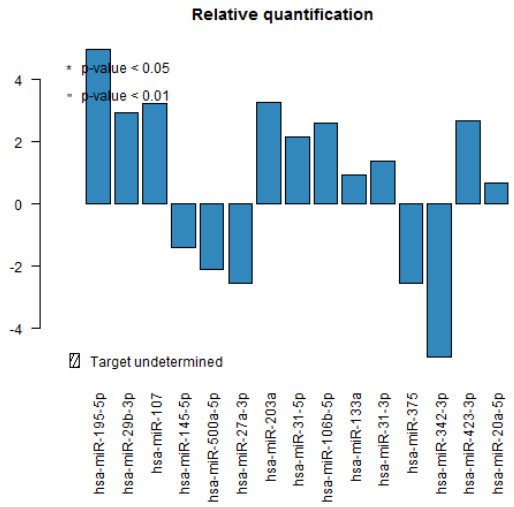

Figura 16: Expresión diferencial 2A vs 2B.

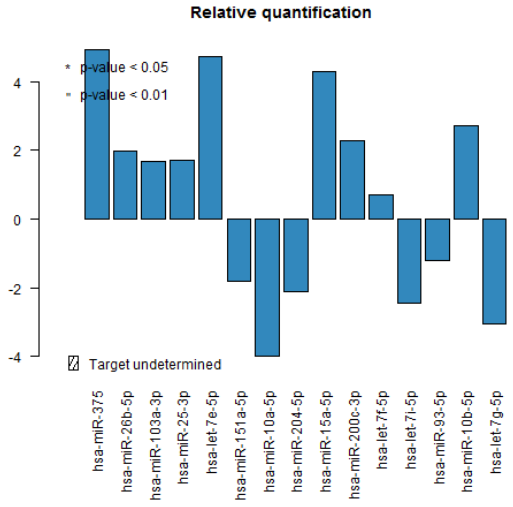

Figura 17: Expresión diferencial 1A vs 2A.

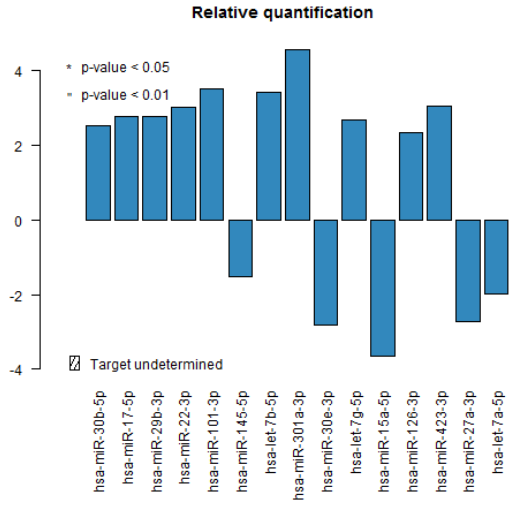

Figura 18: Expresión diferencial 1B vs 2B.

3.7. Información de la sesión de R

Como última parte del pipeline, se ha implementado una línea de código que generará información conteniendo la versión de R con la que se ha ejecutado el pipeline, así como los paquetes usados. Esta información se guardará también automáticamente el archivo `Results_sessionInfo.txt`, y puede ser de utilidad a la hora de diagnosticar problemas de ejecución.

Ejemplo de una sesión tras ejecutar el pipeline:

```r
readLines("Results_sessionInfo.txt")
```

```
##  [1] "R version 3.3.3 (2017-03-06)"
##  [2] "Platform: x86_64-w64-mingw32/x64 (64-bit)"
##  [3] "Running under: Windows 7 x64 (build 7601) Service Pack 1"
##  [4] ""
##  [5] "locale:"
##  [6] "[1] LC_COLLATE=Spanish_Spain.1252  LC_CTYPE=Spanish_Spain.1252   "
##  [7] "[3] LC_MONETARY=Spanish_Spain.1252 LC_NUMERIC=C                  "
##  [8] "[5] LC_TIME=Spanish_Spain.1252    "
##  [9] ""
## [10] "attached base packages:"
## [11] "[1] parallel  stats     graphics  grDevices utils     datasets  methods   b
## [12] ""
## [13] "other attached packages:"
## [14] "[1] HTqPCR_1.28.0       limma_3.30.13       RColorBrewer_1.1-2  Biobase_2.3
## [15] "[5] BiocGenerics_0.20.0 pander_0.6.0        knitr_1.15.1       "
## [16] ""
## [17] "loaded via a namespace (and not attached):"
## [18] " [1] Rcpp_0.12.10          magrittr_1.5          zlibbioc_1.20.0      "
## [19] " [4] stringr_1.2.0         caTools_1.17.1        tools_3.3.3          "
## [20] " [7] KernSmooth_2.23-15    affy_1.52.0           htmltools_0.3.5      "
## [21] "[10] gtools_3.5.0          yaml_2.1.14           rprojroot_1.2        "
## [22] "[13] digest_0.6.12         preprocessCore_1.36.0 affyio_1.44.0        "
## [23] "[16] bitops_1.0-6          codetools_0.2-15      evaluate_0.10        "
## [24] "[19] rmarkdown_1.5         gdata_2.17.0          stringi_1.1.5        "
## [25] "[22] BiocInstaller_1.24.0  gplots_3.0.1          backports_1.0.5      "
## [26] "[25] stats4_3.3.3         "
```

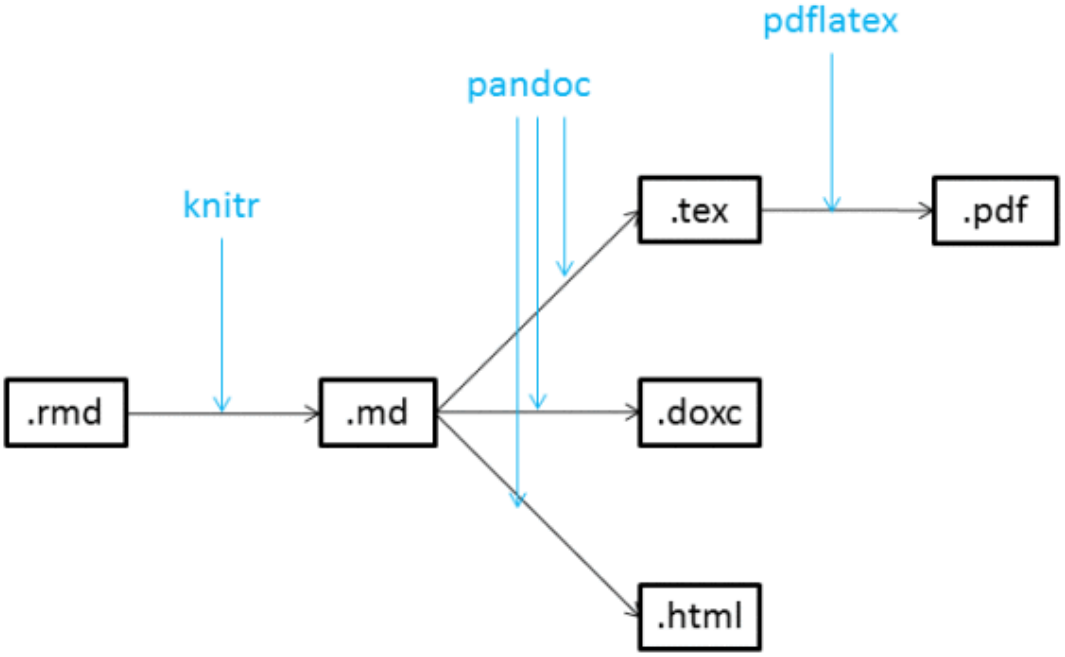

Figura 19: Papel de los paquetes knitr, pandoc y pdflatex al compilar un archivo Rmd.

3.8. Generación del informe dinámico

El documento que contiene el código necesario para el análisis conjuntamente con explicaciones de cada etapa está en formato Rmd. Este tipo de archivos son editables en R studio y se pueden compilar para ejecutar el código contenido y generar un informe con los resultados.

En la cabecera del archivo Rmd, delimitada por líneas `---`, se han indicado las instrucciones necesarias para generar bien un informe en PDF, o bien un informe en formato HTML. El formato de salida es customizable por el usuario, y se explica cómo hacerlo en el apartado siguiente, *Requisitos para ejecutar el pipeline*.

En la figura 19 se muestra un esquema de la conversión del archivo Rmd hasta pdflatex o html, mediante el paquete `knitr`, y los programas `pandoc` y `pdflatex` [22].

Antes de compilar, el usuario también podrá realizar cambios en los siguientes apartados de la cabecera:

- El **título del informe**, proponiendo uno nuevo siempre que vaya entre comillas simples o dobles.
- El **autor**, que aparecerá donde se indica "Nombre y Apellidos".
- La **fecha** del día de compilación, que se genera automáticamente (toma la fecha del equipo). El usuario podrá cambiar esta parte y escoger la fecha deseada, cambiando '`format(Sys.time(), "%d de%B de%Y")`' por 'Día, Mes, Año' o cualquier otro formato, siempre entre comillas.

```
title: "Informe de Resultados de qPCR"
author: "Nombre y Apellidos"
date: '`r format(Sys.time(), "%d de %B de %Y")`'
```

El código que el usuario no necesita saber o modificar ha sido marcado como echo=FALSE, include=FALSE y no aparece en el informe final.

Tras la compilación del documento, se generará un informe con el análisis y los resultados de los datos de qPCR introducidos por el usuario. Como se ha ido indicando a lo largo de este capítulo, el pipeline ha sido diseñado para guardar los resultados obtenidos de forma automática en forma de gráficos y tablas, por si el usuario quiere utilizarlos para incluirlos en otro documento o trabajar con las tablas en una hoja de cálculo.

3.9. Requisitos para ejecutar el pipeline

Para que el usuario pueda ejecutar el pipeline deberá:

1. Tener instalados R, R-studio y PDFLaTeX.
2. Tener instalados en R los paquetes pander y knitr para compilar el informe. Estos paquetes se instalarán introduciendo en la consola la instrucción:

```r
install.packages("knitr", "pander")
```

3. El usuario deberá establecer como directorio de trabajo el directorio donde se encuentran sus datos. Será necesario modificar la variable `ruta <- "./dades"` y cambiarlo por la ruta del usuario actual, en el documento Rmd. Esto debe hacerse antes de la compilación.
4. Opcionalmente, algunas partes de código están preparadas para que puedan ser modificadas por el usuario final en el fichero Rmd. Estos códigos estarán marcados como: `# MODIFICABLE POR EL USUARIO`.

5. Una vez modificadas todos las partes de código necesarias, el usuario debe compilar el informe para obtener los resultados. En la cabecera del archivo Rmd, delimitada por líneas `---`, se podrá comentar y por tanto 'ocultar' (escribiendo `#` al inicio de la línea) parte de ella para generar, bien un informe en PDF para imprimir, o bien un informe html en el que resulta más fácil navegar por él. A continuación se muestra la cabecera que genera el informe pdf:

```
---
title: "Informe de Resultados de qPCR"
author: "Nombre y Apellidos"
date: '`r format(Sys.time(), "%d de %B de %Y")`'

#output:
#  html_document:
#     toc: true
#     toc_float: true
#     number_sections: yes
#     highlight: default

output:
 pdf_document:
   toc: true
   number_sections: yes

header-includes:
- \usepackage[utf8]{inputenc}
- \usepackage[spanish, es-tabla]{babel}
- \usepackage{amsmath,amssymb}

bibliography: bibliografia.bib
---
```

A continuación, se muestra la cabecera que debe generar el informe html:

```
---
title: "Informe de Resultados de qPCR"
author: "Nombre y Apellidos"
date: '`r format(Sys.time(), "%d de %B de %Y")`'

output:
  html_document:
     toc: true
```

```
     toc_float: true
     number_sections: yes
     highlight: default

#output:
# pdf_document:
#   toc: true
#   number_sections: yes

header-includes:
- \usepackage[utf8]{inputenc}
- \usepackage[spanish, es-tabla]{babel}
- \usepackage{amsmath,amssymb}

bibliography: bibliografia.bib
---
```

3.9.1. Formato de los datos de entrada

Este pipeline ha sido diseñado para leer valores de Cq de placas prediseñadas miRCURY LNAT de la casa Exiqon, medidas en un termociclador ABI PRISM 7000 Sequence Detection System. La etapa de entrada de datos está preparada para leer automáticamente datos cuyo formato será el siguiente:

- Cada placa de qPCR tendrá la misma distribución: cada gen medido se encuentra en el mismo pocillo en todas las placas.
- A cada placa le corresponde un único fichero de texto con 1 columna y tantas filas como pocillos, con los valores Cq.
- Los ficheros han de ser de texto plano (csv, txt).
- Se necesita otro fichero adicional `samplesinfo.txt` que relaciona el nombre del fichero con las condiciones experimentales correspondientes. Deberá aparecer como mínimo los nombres de los archivos que contienen los datos en la primera columna. De forma adicional se pueden añadir columnas con la información relativa a las condiciones experimentales de cada muestra. Se muestra un ejemplo del contenido de este fichero en la tabla 4.

Tabla 4: Ejemplo del formato que debe tener el fichero samplesinfo.txt.

Sample	Treatment	Response
sample01.csv	1	A
sample02.csv	1	B
sample03.csv	1	A
sample04.csv	1	B
sample05.csv	1	A
sample06.csv	1	B

- Se necesita un fichero adicional `Genesinfo.txt` con información sobre el nombre del gen que se evalúa en cada pocillo. La distribución de columnas debe ser la que aparece en la tabla 5, y como mínimo deberá aparecer el nombre y la posición (pocillo) de cada gen o control medido en la placa. Se podrán añadir columnas opcionales con la secuencia amplificada y otras informaciones.

Tabla 5: Resumen de los datos crudos de expresión de cada muestra.

microRNA_Name	Posicion_Placa	Target_sequence
hsa-miR-17-5p	1	CAAAGUGCUUACAGUGCAGGUAG
hsa-miR-29c-3p	2	UAGCACCAUUUGAAAUCGGUUA
hsa-miR-191-5p	3	CAACGGAAUCCCAAAAGCAGCUG
hsa-miR-26b-5p	4	UUCAAGUAAUUCAGGAUAGGU
hsa-miR-193b-3p	5	AACUGGCCCUCAAAGUCCCGCU
hsa-miR-203a	6	GUGAAAUGUUUAGGACCACUAG

El documento dinámico viene acompañado de un set de datos de muestra que cumple las condiciones de formato.

4. Comparación con otras herramientas de análisis de qPCR

4.1. Funcionamiento de pyqPCR

La aplicación pyqPCR [3], desarrollada en lenguaje de programación Python, es una herramienta con interfaz gráfica de usuario que permite hacer cálculos básicos a partir de datos Cq de experimentos de qPCR. La aplicación está disponible de forma completamente gratuita en la página oficial de los creadores, en el siguiente enlace: http://pyqPCR.sourceforge.net/.

A partir de los valores Cq, la aplicación puede calcular:

- calcula la curva estándar y la eficiencia de amplificación.
- Calcula cuantificación absoluta de ADN, a partir de un estándar de concentraciones.
- Calcula expresión génica relativa mediante el método $\Delta\Delta$Cq desarrollado por Livak y Schmittgen [19]. Incluye un método de normalización en el que se requieren genes normalizadores.

La aplicación admite archivos en formato crudo csv, txt, directamente importados del aparato de PCR, y también lee el formato XML, basándose en las propuestas MiQE [6]. A la hora de cargar los archivos, debes escoger el tipo de cuantificación que quieres calcular (absoluta o relativa), y escoger el aparato de qPCR en el que se han generado los datos, para que la aplicación pueda leer adecuadamente el formato. En la figura 4.1 se muestra la pantalla de carga.

Una vez cargados los datos de muestra de nuestro experimento (28 archivos), es necesario indicar, de forma manual, la disposición de cada tipo de muestra: control, gen de referencia, etc. en la placa (fig. 4.1).

Llegados a este punto, el marcaje de cada uno de los pocillos en *target*, *sample* y *control* no fue nada intuitiva y la nomenclatura era confusa. La documentación del programa es algo pobre en este paso y no me permitió continuar de forma intuitiva, generando muchos errores e imposibilitando la generación de un gráfico de expresión relativa con mis datos. Algunos de estos intentos se recogen en las figuras 22-25.

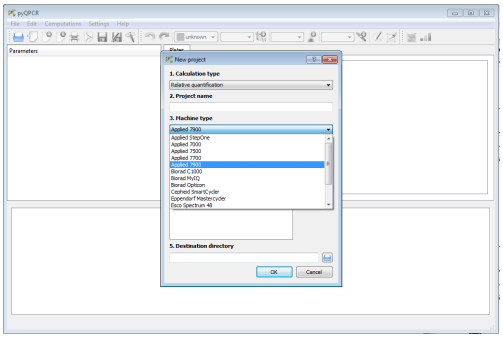

Figura 20: Errores con los valores ausentes.

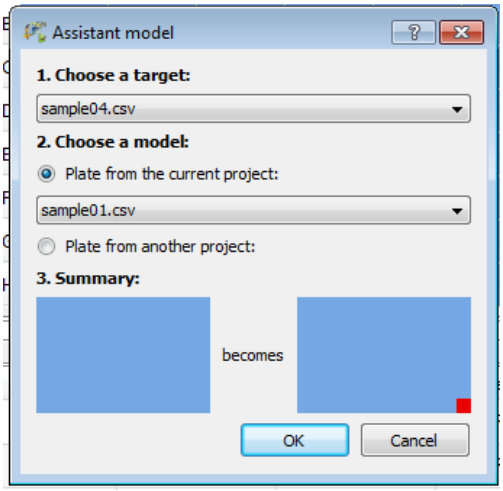

Figura 21: Error en la adjudicación de un gen de referencia.

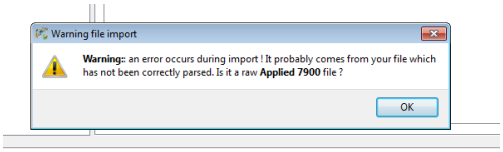

Figura 22: Un error durante la carga de los archivos, que fue solventado modificando los ficheros csv.

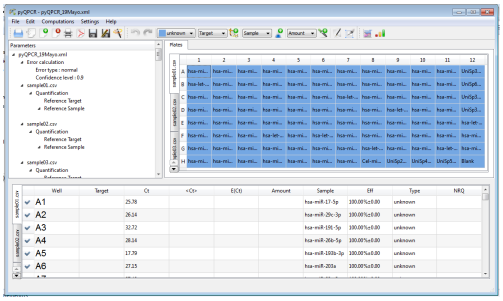

Figura 23: Placa cargada en el programa.

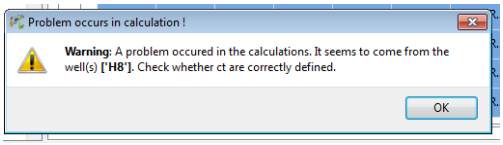

Figura 24: Errores con los valores ausentes.

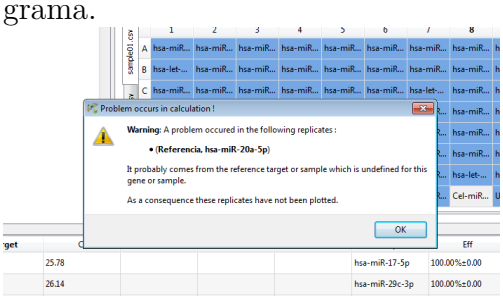

Figura 25: Error en la adjudicación de un gen de referencia.

No obstante, existe un vídeo tutorial donde se muestra el funcionamiento de esta aplicación, aunque sin explicaciones ni comentarios adicionales aclarando cada paso: https://vimeo.com/12695306.

En este caso no se llegó a completar el análisis de nuestros datos. No obstante, la aplicación permite generar un informe en PDF con los resultados (tablas, curvas estándar y gráficos de cuantificación), así como guardarlos en formato XML.

4.2. Ventajas y desventajas

En este apartado se resumen las ventajas y desventajas encontradas en cada uno de los programas de análisis de qPCR utilizados: pyqPCR (tabla 6) y el pipeline desarrollado en este trabajo (tabla 7).

Tabla 6: Fortalezas y debilidades del programa gratuito pyqPCR, desarrollado en lenguaje Python.

Fortalezas	Debilidades
La interfaz gráfica permite su utilización a usuarios sin nociones de programación	La aplicación es muy poco intuitiva a la hora de designar las muestras problema, los genes de referencia, los genes normalizadores, etc.
Permite calcular fácilmente la eficiencia de la PCR a partir de una serie de diluciones estándar	Cuando se tienen que analizar numerosas muestras/placas de qPCR, preparar las placas para su correcta lectura y obtener los gráficos se convierte en una tarea ardua.
Cumple las directrices MiQE, generando los resultados en un archivo XML	El método de normalización y cálculo de ΔCq requieren la elección de genes normalizadores
Permite generar un pdf con los resultados	Actualizado por última vez en 2012
Es gratuito y está basado en un lenguaje de código abierto (Python)	No admite placas de 384 pocillos

Tabla 7: Fortalezas y debilidades del pipeline desarrollado en lenguaje R.

Fortalezas	Debilidades
Está preparado para generar un informe dinámico, con los resultados redactados de forma automática	No existe un apartado para calcular la eficiencia de la PCR a partir de un estándar
Permite analizar un gran número de muestras sin implicar más tiempo de manipulación para el usuario	No genera los resultados en formato XML tal y como indica MiQE
El informe se puede compilar tanto en formato html como pdf	No posee interfaz gráfica, se requieren unas mínimas nociones de programación en R por parte del usuario
Está basado en un lenguaje de código abierto (R) y es modificable y mejorable por otros usuarios	Hay secciones en las que es necesario mejorar el código para que sea dinámico y aplicable a otros datos
Permite utilizar dos métodos de normalización: respecto a un gen de referencia y un método que no requiere de genes normalizadores, específico para miRNA	
Permite flexibilidad en el manejo de errores si el usuario conoce el entorno R y los datos que maneja	

5. Conclusiones

5.1. Conclusiones generales

A continuación se describe las conclusiones obtenidas tras la elaboración del presente trabajo:

1. El lenguaje de programación R es versátil y está diseñado especialmente para facilitar la manipulación de datos. Además, se trata de un lenguaje de código abierto y multiplataforma, lo que lo hace idóneo para su uso en investigación.

2. Existen numerosas librerías desarrolladas para procesar datos biológicos en R, entre ellas el proyecto bioconductor y el paquete `HTqPCR`, utilizado para implementar el pipeline desarrollado en el presente trabajo. Es por eso que es uno de los principales lenguajes utilizados en bioinformática y bioestadística hoy en día.

3. R-markdown es una herramienta sencilla, cómoda y potente que permite generar informes dinámicos en pdf y html, imprescindibles hoy en día para asegurar una investigación reproducible.

4. El pipeline desarrollado me ha servido tanto para aprender a manejar datos biológicos de forma personalizada utilizando un lenguaje de programación, como para obtener un producto final que, con algunas mejoras y modificaciones, me servirá de base para automatizar otros análisis similares que tenga que elaborar en un futuro.

5. El aprendizaje de un lenguaje de programación (como R o Python) es básico para poder manejar de forma automática y personalizada grades cantidades de muestras, ahorrándonos tiempo de manipulación y tareas repetitivas que habría que hacer en otros programas de interfaz gráfica como pyQPCR o una hoja excel.

5.2. Logro de los objetivos

- **Objetivo 1**
- Diseño del pipeline, especificando y justificando cada uno de los pasos de los que constará.

Este objetivo llevó más tiempo de lo esperado debido al tiempo dedicado a la revisión bibliográfica y a la obtención de los datos. Completarlo correctamente fue

un logro y un paso crítico para después poder implementar todos los pasos en R.

- Implementación de cada uno de los pasos del pipeline con las librerías de R.

Este objetivo es el principal y el que da título al Trabajo Fin de Máster. Ha sido cumplido en su totalidad, obteniendo como resultado un pipeline ejecutable.

- Generar un informe automático para visualizar los resultados de una forma más intuitiva.

Este objetivo también se ha cumplido satisfactoriamente, pues al final del proyecto se ha obtenido un documento que integra código y comentarios aclaratorios de texto dinámico en un archivo Rmd compilable en formato html y pdf.

- **Objetivo 2**
- Ejecución del pipeline utilizando datos de qPCR públicos de una base de datos.

Este objetivo ha sido también correctamente cumplimentado, puesto que el pipeline se elaboró en base a los datos aportados por el profesor supervisor. El pipeline ha sido ejecutado con los datos de muestra y se han generado unos resultados que nos han permitido obtener conclusiones y responder a las preguntas inicialmente planteadas.

- Evaluación del pipeline, comparando los resultados con otro software para qPCR ya existente, e implementar posibles mejoras.

Este objetivo ha sido parcialmente cumplimentado. Se escogieron al inicio varios programas de análisis de qPCR (Thermo Fisher Cloud, pyPCR), todo ellos propuestos en las PEC2. Finalmente se escogió pyQPCR por su sencilla interfaz e instalación. Pese a todo, ha habido problemas a la hora de utilizarlo y generar unos resultados.

5.3. Análisis de la planificación y dificultades observadas

La planificación propuesta en el diagrama de Gantt no se ha cumplido en su totalidad, sobre todo en las etapas finales del proyecto. Sin embargo, los hitos importantes (PEC1, PEC2 y PEC3) sí han sido respetados.

Asimismo, subestimé el tiempo calculado para la revisión bibliográfica y análisis de paquetes de R hasta encontrar el adecuado y aprender a manejarlo. Se probaron también numerosos paquetes, como EximiR, ReadqPCR, NormqPCR, hasta escoger el definitivo: HTqPCR.

Tampoco tuve en cuenta la dificultad de encontrar una plataforma con datos de qPCR disponibles, y la dificultad de trabajar con unos datos heredados, de los cuales no se dispone de mucha información.

También he encontrado dificultad en realizar un trabajo TFM supervisado a distancia, en el cual se requiere gran capacidad de organización, motivación y trabajo autónomo por parte del alumno. Sin embargo, la comunicación periódica con mi profesor supervisor ha sido clave para poder superar esta dificultad.

Finalmente, algo que ha beneficiado y compensado el tiempo de más invertido ha sido la utilización de herramientas como Markdown y Latex para la redacción de esta memoria, puesto que me permitió centrarme en la redacción y generar automáticamente elementos como el índice, las numeraciones de tablas y figuras, el formato final, etc.

5.4. Mejoras futuras

Conforme mis conocimentos de R sean mayores, me será posible seguir mejorando el informe dinámico para adaptarlo de forma completamente automática a nuevos datos, siempre y cuando se respete el formato descrito.

Una posible mejora sería la integración del pipeline en una interfaz gráfica, que no requiera de conocimientos de R por parte del usuario. Una manera sencilla proporcionada por R es Shiny, de manera que el usuario pueda subir online datos a la aplicación y escoger los parámetros, las muestras a descartar, etc, para finalmente generar el informe en PDF.

6. Glosario

ARNm	Acido ribonucléico mensajero. Biomolécula que transporta la información genética desde el ADN hasta la traducción a proteína.
cDNA	ADN complementario. ADN obtenido a partir de una molécula de ARN mediante una reacción de reversotranscripción.
Cq	Ciclo de cuantificación. En una qPCR, ciclo de amplificación en el que se comienza a detectar el producto amplificado, mediante fluorescencia.
Ct	Sinónimo de Cq.
ΔCq	Diferencia entre el ciclo de cuantificación Cq de una muestra con respecto al Cq de un gen normalizador.
$\Delta\Delta$Cq	Diferencia el ciclo de cuantificación Cq de dos muestras normalizadas respecto a un gen normalizador.
Exosoma	Vesículas membranosas liberadas por muchas células al medio intracelular y presentes en fluidos como sangre, orina y leche. Contienen numerosas moléculas que sirven como biomarcadores, entre ellas, miRNAs.
Gen normalizador	Sinónimo de gen de referencia.
Gen de referencia	Gen respecto al cual se calcula la expresión relativa de otros genes. Normalmente es un gen que presenta una expresión génica constante en todas las condiciones experimentales del experimento.
miRNA	micro ARN. ARN no codificante que inhibe la expresión génica mediante unión al ARNm.
PCR	Reacción en cadena de la polimerasa. Técnica utilizada para amplificar una secuencia de ADN específica para su posterior detección, cuantificación o utilización.
Pipeline	En lenguaje de programación, flujo de trabajo que comprende una serie de pasos, en los que entran unos datos como input y se obtienen dichos datos procesados como salida.

Pocillo(s)	Cada uno de las cavidades de una placa de qPCR donde se llevará a cabo la reacción. Estas placas pueden contener 96 ó 384 de estas cavidades.
qPCR	Modalidad de PCR utilizada para amplificar y simultáneamente cuantificar el producto de amplificación de ADN.
Retrotranscripción	Reacción llevada a cabo por una enzima tipo ADN-polimerasa denominada retrotranscriptasa, que consiste en copiar una molécula de ARN a ADN.
Termociclador	Equipo de laboratorio en el que se lleva a cabo la reacción de PCR o qPCR.

Bibliografía

1. Conesa A, Madrigal P, Tarazona S *et al.* A survey of best practices for rna-seq data analysis. *Genome biology* 2016;**17**:13.

2. Rodiger S, Burdukiewicz M, Blagodatskikh K *et al.* R as an environment for the reproducible analysis of dna amplification experiments. *The R Journal* 2015;**7**:127–50.

3. PyQPCR, a free software for qPCR. *http://pyqpcrsourceforgenet/?static5/about* Accessed: 2017-05-1919AD.

4. Pabinger S, Rodiger S, Kriegner A *et al.* A survey of tools for the analysis of quantitative pcr (qPCR) data. *Biomolecular Detection and Quantification* 2014;**1**:23–33.

5. Porterfield A. BitesizeBio. *http://bitesizebiocom/24581/what-is-a-ct-value/* Accessed: 2017-05-1111AD.

6. Bustin SA, Benes V, Garson JA *et al.* The miqe guidelines: Minimum information for publication of quantitative real-time pcr experiments. *Clinical chemistry* 2009;**55**:611–22.

7. Benes V, Castoldi M. Expression profiling of microRNA using real-time quantitative pcr, how to use it and what is available. *Methods* 2010;**50**:244–9.

8. Bentwich I, Avniel A, Karov Y *et al.* Identification of hundreds of conserved and nonconserved human microRNAs. *Nature genetics* 2005;**37**:766–70.

9. Rao S. Role of transcriptomics in gene expression studies. *https://wwwslidesharenet/sarlayadav71/ role-of-transcriptomics-in-gene-expression-studies-and* Accessed: 2017-05-1717AD.

10. Frontela Noda M. MicroRNAs en el cáncer: De la investigación a la práctica clínica. *Revista Cubana de Medicina* 2012;**51**:325–35.

11. Homami A, Ghazi F. MicroRNAs as biomarkers associated with bladder cancer. *Medical Journal of The Islamic Republic of Iran (MJIRI)* 2016;**30**:1282–9.

12. Heffler E, Allegra A, Pioggia G *et al.* MicroRnas profiling in asthma: Potential biomarkers and therapeutic targets. *American Journal of Respiratory Cell and Molecular Biology* 2017.

13. Banerjee J, Nema V, Dhas Y *et al.* Role of micrornas in type 2 diabetes and associated vascular complications. *Biochimie* 2017.

14. Vermeulen J, De Preter K, Naranjo A *et al.* Predicting outcomes for chil-

dren with neuroblastoma using a multigene-expression signature: A retrospective siopen/cog/gpoh study. *The lancet oncology* 2009;**10**:663–71.

15. Zhang J, Li S, Li L *et al.* Exosome and exosomal microRNA: Trafficking, sorting, and function. *Genomics, proteomics & bioinformatics* 2015;**13**:17–24.

16. Huber, W., Carey *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods* 2015;**12**:115–21.

17. Dvinge H, Bertone P. HTqPCR: High-throughput analysis and visualization of quantitative real-time pcr data in r. *Bioinformatics* 2009;**25**:3325–6.

18. Matz MV, Wright RM, Scott JG. No control genes required: Bayesian analysis of qRT-pcr data. 2013.

19. Livak KJ, Schmittgen TD. Analysis of relative gene expression data using real-time quantitative pcr and the 2- $\Delta\Delta$ct method. *methods* 2001;**25**:402–8.

20. Mestdagh P, Van Vlierberghe P, De Weer A *et al.* A novel and universal method for microRNA rt-qPCR data normalization. *Genome biology* 2009;**10**:R64.

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

22. Hespen RV. Writing your thesis with r markdown. *https:// rosannavanhespen-research wordpress com/ 2016/03/30/ writing-your-thesis-with-r-markdown-5-the-thesis-layout/* Accessed: 2017-05-2121AD.

Anexo

En este anexo se adjunta todo el código R contenido en el pipeline.

1. Cabecera del archivo Rmd

```
---
output:
  pdf_document:
    number_sections: yes

header-includes:
- \usepackage[utf8]{inputenc}
- \usepackage[spanish, es-tabla]{babel}
- \usepackage{amsmath,amssymb}
- \usepackage{placeins}
- \usepackage{tabularx}

- \renewcommand{\thesubsection}{\thesection.\Roman{subsection}}
- \renewcommand{\arraystretch}{1.5}

---
```

2. Opciones de los fragmentos de código

```
## ----setup, include=FALSE-----------------------------------------------
knitr::opts_chunk$set(echo = TRUE, comment = NULL, error=FALSE, message = FALSE, warning = FALSE)
options(width = 90, digits = 4)
library(knitr)
```

3. Instalación automática de paquetes

```
## ----instalar_paquetes, echo=FALSE, include=FALSE------------------------
# listas de paquetes
paquetes_bioc <- c("HTqPCR")
paquetes_nuevos <- c("pander")


# cada paquete ha de descargarse de un repositorio
# respositorio bioconductor
nuevos_bioc <- paquetes_bioc[!(paquetes_bioc %in% installed.packages()[,"Package"])]
if(length(nuevos_bioc)){
  source("https://bioconductor.org/biocLite.R")
  biocLite(nuevos_bioc)
}
```

```
# respositorio CRAN
nuevos <- paquetes_nuevos[!(paquetes_nuevos %in% installed.packages()[,"Package"])]
if(length(nuevos)){
  install.packages(nuevos)
}

## ----cargar_paquetes, echo=FALSE, results='hide'-------------------------
# Una vez instalados los paquetes necesarios, vamos a cargarlos
library(pander)
library(HTqPCR)
```

4. Lectura de los datos

```
## ----directorio----------------------------------------------------------
# MODIFICABLE POR EL USUARIO
# ruta de carpetas donde se encuentran los datos a analizar:
ruta <- "./dades"

## ----pocillos------------------------------------------------------------
# MODIFICABLE POR EL USUARIO
# Escoge el número de pocillos de la placa (96/384):
pocillos <- 96

## ----setwd, echo=FALSE, include=FALSE------------------------------------
# Para que R trabaje en esta carpeta:
setwd(ruta)

## ----leerCq, echo=FALSE, include=FALSE-----------------------------------
# leemos el fichero con la informacion de las muestras:
targets <- read.table(file.path(ruta, "samplesinfo.txt"),sep = ",", header=TRUE)

# leemos los ficheros con los valores de Cq:
raw <- readCtData(files = targets$Sample, path = ruta,
                  format = "plain", n.features = pocillos,
                  column.info = list(position=1, Ct=3),
                  n.data = 1,
                  sep = ",", header = TRUE, skip = 2)



## ------------------------------------------------------------------------
# Para que no aparezca el aviso de los valores NA (ausentes),
# se suele sustituir por un valor numérico elevado como 40
# MODIFICABLE POR EL USUARIO
limitcq <- 40
exprs(raw)[is.na(exprs(raw))] <- limitcq

## ----genes, echo=FALSE, include=FALSE------------------------------------
# leemos y almacenamos los nombres de los genes:
genes <- file.path(ruta, "Genesinfo.csv")
genesinfo <- read.csv(genes, header = TRUE)
```

```
genelist <- genesinfo[,1]

# asignamos los nombres de los genes a nuestros datos de qPCR:
featureNames(raw) <- as.vector(genelist)

## ----genelist, echo=FALSE, include=FALSE---------------------------------
# Creamos una tabla con la lista de genes:
genes <- paste(c(1:length(genelist)), genelist)
pander(t(matrix(genes, nrow = 8, ncol = 12)), justify="left")
```

5. Resumen numérico de los datos

```
## ----resumen, echo=FALSE-------------------------------------------------
# Creamos una tabla-resumen:
pander(t(as.matrix(summary(raw))))

## ----resumencsv, echo=FALSE, include=FALSE-------------------------------
# salvamos el resumen en un archivo de texto:
write.csv(t(as.matrix(summary(raw))), file="Results_raw_summary.csv")
```

6. Clasificación de los datos por grupos experimentales

```
## ----grupos, echo=FALSE, include=FALSE-----------------------------------
# ESTE TROZO DE CÓDIGO ES ESPECÍFICO DE LOS DATOS DEL PROYECTO
# En el dataset 'targets', añadimos una nueva variable 'Group'
# que indique grupo y respuesta:
targets$Group <- NA

# recodificamos la variable Group:
targets$Group[targets$Treatment == 1 & targets$Response == "A"] <- "1A"
targets$Group[targets$Treatment == 1 & targets$Response == "B"] <- "1B"
targets$Group[targets$Treatment == 2 & targets$Response == "A"] <- "2A"
targets$Group[targets$Treatment == 2 & targets$Response == "B"] <- "2B"
```

7. Colores y leyenda

```
## ----colores, echo=FALSE, include=FALSE----------------------------------
# preparación de elementos para plots de visualizacion preliminar

# paletas de color
colores2 <- c('lightcoral', 'lightblue4')
colores4 <- c(
  rep(c('lightcoral', 'lightblue4'), 7),
  rep(c('peachpuff', 'lightblue'), 7)
```

```
  )

# leyenda
leyenda_grupos <- c("bottomright", cex = 0.8,
                    legend = c("1A", "2A", "1B", "2B"),
                    fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))
```

8. Gráficos de Cq de cada gen en crudo

```
## ----plotCt_grupos, echo=FALSE, message=FALSE, results='hide'------------

### PLOT EXPRESION Cq RAW

# muestras agrupadas por grupo y respuesta

par(mfrow=c(2,2))


plotCtOverview(raw[1:24],
               genes = featureNames(raw), groups = (targets$Group),
               ylim = c(0, 45),
               col=colores4,
               legend = FALSE
)

plotCtOverview(raw[25:48],
               genes = featureNames(raw), groups = (targets$Group),
               ylim = c(0, 45),
               col=colores4,
               legend = FALSE
)

plotCtOverview(raw[49:72],
               genes = featureNames(raw), groups = (targets$Group),
               ylim = c(0, 45),
               col=colores4,
               legend = FALSE
)

plotCtOverview(raw[72:96],
               genes = featureNames(raw), groups = (targets$Group),
               ylim = c(0, 45),
               col=colores4,
               legend = FALSE
)

legend("bottomright", cex = 0.5,
                    legend = c("1A", "2A", "1B", "2B"),
                    fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))
```

```
dev.copy(png,'Results_raw_plotCtOverview.png')
dev.off()
```

9. Gráficos de Cq de cada muestra en crudo

```
## ----raw_boxplot, echo=FALSE, message=FALSE, results='hide'--------------
### BOXPLOT de cada paciente
plotCtBoxes(raw, col = colores4, las=2, names = paste("muestra", c(1:28)))

legend("bottomright", cex = 0.8,
        legend = c("1A", "2A", "1B", "2B"),
        fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue')
)

dev.copy(png,'Results_raw_plotCtBoxes.png')
dev.off()
```

10. Gráficos de agrupación

```
## ---- echo=FALSE, results='hide'----------------------------------------
clusterCt(raw, type = "samples")

## ----PCA_raw, echo=FALSE, results='hide'---------------------------------
plotCtPCA(raw, features=FALSE, col = colores4)
legend("bottomright", cex = 0.8,
       legend = c("1A", "2A", "1B", "2B"),
       fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))
dev.copy(png,'Results_raw_PCA.png')
dev.off()
```

11. Controles internos

```
## ----controles Uni-------------------------------------------------------
# MODIFICABLE POR EL USUARIO SI ES NECESARIO
# Asignar posiciones de los controles en la placa:
UniSp3 <- c(12, 36, 48)
UniSp6 <- 24
CelmiR39 <- 92
UniSp2 <- 93
```

```
UniSp4 <- 94
UniSp5 <- 95
Blank <- 96

## ---- echo=FALSE, include=FALSE-----------------------------------------
# creamos listas con las posiciones y los nombres de los controles
controles_pocillos <- c(UniSp3, UniSp6, CelmiR39, UniSp2, UniSp4, UniSp5, Blank)
controles_nombre <- rep("Control", length(controles_pocillos))

# asignamos los nombres de los genes a nuestros datos de qPCR:
featureType(raw)[controles_pocillos] <- controles_nombre

## ----extraccion ARN, echo=FALSE, results='hide'--------------------------

# CONTROL DEL PROCESO DE EXTRACCIÓN de ARN
# UniSp2, UniSp4 and UniSp5
# stepwise diferences btwn thm is expected to be 5-7 Cq

plotCtOverview(raw[c(UniSp2, UniSp4, UniSp5)],
               genes = featureNames(raw),
               ylim = c(0, 45),
               las=1,
               col=colores4,
               legend = FALSE
)

legend("bottomright", cex = 0.8,
       legend = c("1A", "2A", "1B", "2B"),
       fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))

dev.copy(png,'Results_plotCt_unisp245.png')
dev.off()

## ----cDNA, echo=FALSE, results='hide'------------------------------------

plotCtOverview(raw[c(UniSp6, CelmiR39)],
               genes = featureNames(raw),
               ylim = c(0, 45),
               las=1,
               col=colores4,
               legend = FALSE
)

legend("bottomright", cex = 0.8,
       legend = c("1A", "2A", "1B", "2B"),
       fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))

dev.copy(png,'Results_plotCt_unisp6CelmiR39.png')
dev.off()

## ----interplate, echo=FALSE, results='hide'------------------------------
# UniSp3 IPC
plotCtOverview(raw[UniSp3],
               genes = featureNames(raw),
```

```
               ylim = c(0, 20),
               las=1,
               legend = FALSE,
               col = colores4
)

legend("bottomright", cex = 0.8,
       legend = c("1A", "2A", "1B", "2B"),
       fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))

dev.copy(png,'Results_plotCt_Uni3.png')
dev.off()
```

12. Filtrado por categorías

```
## ------------------------------------------------------------------------
# MODIFICABLE POR EL USUARIO
Cqmax <- 35
Cqmin <- 10

## ---- echo=FALSE, results='hide'-----------------------------------------
raw.cat <- setCategory(raw, groups = targets$Group, Ct.max = 38, Ct.min = 1)
plotCtCategory(raw.cat)
dev.copy(png,'Results_categories.png')
dev.off()
```

13. Elección de gen normalizador

```
## ------------------------------------------------------------------------
# MODIFICABLE POR EL USUARIO
numero_genes_ref <- 3

## ---- echo=FALSE---------------------------------------------------------
# cálculo de expresión media
exp_desvest <- apply(X = exprs(raw), FUN = sd, MARGIN = 1)
exp_mean <- apply(X = exprs(raw), FUN = mean, MARGIN = 1)

# cálculo de la desviación estándar relativa a la expresión media
rel_desvest <- sort(exp_desvest/exp_mean)

candidatos <- setdiff(names(rel_desvest), as.character(genelist)[controles_pocillos])

genes_ref <- candidatos[1:numero_genes_ref]
```

```
## ---- echo=FALSE-----------------------------------------------------------
pander(as.matrix(rel_desvest[1:10]), justify="left")
```

14. Normalización

```
## ---------------------------------------------------------------------------
# MODIFICABLE POR EL USUARIO
tipo_de_normalizacion <- "deltaCt" # opciones: "geometric.mean", "deltaCt"


## ----normalizacion, echo=FALSE, include=FALSE----------------------------

# normalización
exp_norm <- normalizeCtData(raw, norm=tipo_de_normalizacion, deltaCt.genes = c(genes_ref))

## ----pre_post_norm, echo=FALSE, results='hide'---------------------------
plot(exprs(raw), exprs(exp_norm), pch = 20, main = paste("Normalización por método",
tipo_de_normalizacion), col = rep(colores4))
dev.copy(png,'Results_norm_vs_raw.png')
dev.off()

## ----bx_raw, echo=FALSE, results='hide'----------------------------------
### BOXPLOT de cada paciente
plotCtBoxes(raw, col = colores4,
            las = 2,
            names = paste("muestra", c(1:28)),
            mar = c(5,5,1,1),
            main = "Muestras sin normalizar",
            )
legend("topright", cex = 0.4,
        legend = c("1A", "2A", "1B", "2B"),
        fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue')
)

dev.off()

## ----bx_norm, echo=FALSE, results='hide'----------------------------------
plotCtBoxes(exp_norm, col = colores4,
            las = 2,
            names = paste("muestra", c(1:28)),
            mar = c(5,5,1,1),
            main = "Muestras normalizadas",
            )
legend("topright", cex = 0.4,
        legend = c("1A", "2A", "1B", "2B"),
        fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue')
```

```
)

dev.off()
```

15. Filtrado tras normalización

```
## ----------------------------------------------------------------------
#MODIFICABLE POR EL USUARIO
excluir <- 3

## ---- echo=FALSE, results='asis'----------------------------------------
# filtro de controles,valores y 'Unreliable' y/o 'Undetermined'
exp_Filt <- filterCtData(exp_norm, remove.type = "Control",
remove.category = "Unreliable", n.category = excluir)
```

16. Heatmap y PCA tras normalización

```
## ---- echo=FALSE, results='hide'----------------------------------------
plotCtHeatmap(exp_Filt, dist = "pearson")
dev.copy(png,'Results_heatmap.png')
dev.off()

## ----PCA_exp_Filt, echo=FALSE, results='hide'----------------------------
plotCtPCA(exp_Filt, features=FALSE, col = colores4)
legend("bottomright", cex = 0.8,
       legend = c("1A", "2A", "1B", "2B"),
       fill = c('lightcoral', 'peachpuff', 'lightblue4', 'lightblue'))
dev.copy(png,'Results_norm_PCA.png')
dev.off()
```

17. Expresión diferencial y tests estadísticos

```
## ----------------------------------------------------------------------
# MODIFICABLE POR EL USUARIO
sig <- 0.05

## ---- echo=FALSE, include=FALSE----------------------------------------
```

```
DE.1A.1B.ttest <- ttestCtData(exp_Filt[,1:14], groups = targets$Group[1:14])

## ---- echo=FALSE-----------------------------------------------------------
# creamos nueva variable "REGULATED":
DE.1A.1B.ttest$Regulated <- NA

# recodificamos la variable:
DE.1A.1B.ttest$Regulated[DE.1A.1B.ttest$ddCt>0] <- "up"
DE.1A.1B.ttest$Regulated[DE.1A.1B.ttest$ddCt<0] <- "down"

pander(DE.1A.1B.ttest[1:15,c(1, 5, 6, 7, 12)])
write.csv(DE.1A.1B.ttest, file = "Results_DE_1Avs1B.csv")

## ---- echo=FALSE, results='hide'-------------------------------------------
plotCtRQ(DE.1A.1B.ttest, genes = 1:15, cex.lab = 0.1)
dev.copy(png,'Results_1A1B.png')
dev.off()

## ----include=FALSE---------------------------------------------------------
DE.2A.2B.ttest <- ttestCtData(exp_Filt[,15:28], groups = targets$Group[15:28])

## ----echo=FALSE------------------------------------------------------------
# creamos nueva variable "REGULATED":
DE.2A.2B.ttest$Regulated <- NA

# recodificamos la variable:
DE.2A.2B.ttest$Regulated[DE.2A.2B.ttest$ddCt>0] <- "up"
DE.2A.2B.ttest$Regulated[DE.2A.2B.ttest$ddCt<0] <- "down"

pander(DE.2A.2B.ttest[1:15,c(1, 5, 6, 7, 12)])
write.csv(DE.2A.2B.ttest, file = "Results_DE_2Avs2B.csv")

## ---- echo=FALSE, results='hide'-------------------------------------------
plotCtRQ(DE.2A.2B.ttest, genes = 1:15, cex.lab = 0.1)
dev.copy(png,'Results_2A2B.png')
dev.off()

## ----include=FALSE---------------------------------------------------------
DE.1A.2A.ttest <- ttestCtData(exp_Filt[,seq(1, 27, 2)], groups = targets$Group[seq(1, 27, 2)])

## ----echo=FALSE------------------------------------------------------------
# creamos nueva variable "REGULATED":
DE.1A.2A.ttest$Regulated <- NA

# recodificamos la variable:
DE.1A.2A.ttest$Regulated[DE.1A.2A.ttest$ddCt>0] <- "up"
DE.1A.2A.ttest$Regulated[DE.1A.2A.ttest$ddCt<0] <- "down"

pander(DE.1A.2A.ttest[1:15,c(1, 5, 6, 7, 12)])
write.csv(DE.1A.2A.ttest, file = "Results_DE_1Avs2A.csv")

## ---- echo=FALSE, results='hide'-------------------------------------------
plotCtRQ(DE.1A.2A.ttest, genes = 1:15, cex.lab = 0.1)
dev.copy(png,'Results_1A2A.png')
```

```
dev.off()

## ----include=FALSE-------------------------------------------------------
DE.1B.2B.ttest <- ttestCtData(exp_Filt[,seq(2, 28, 2)], groups = targets$Group[seq(2, 28, 2)])

## ----echo=FALSE----------------------------------------------------------
# creamos nueva variable "REGULATED":
DE.1B.2B.ttest$Regulated <- NA

# recodificamos la variable:
DE.1B.2B.ttest$Regulated[DE.1B.2B.ttest$ddCt>0] <- "up"
DE.1B.2B.ttest$Regulated[DE.1B.2B.ttest$ddCt<0] <- "down"

pander(DE.1B.2B.ttest[1:15,c(1, 5, 6, 7, 12)])
write.csv(DE.1B.2B.ttest, file = "Results_DE_1Bvs2B.csv")

## ---- echo=FALSE, results='hide'-----------------------------------------
plotCtRQ(DE.1B.2B.ttest, genes = 1:15, cex.lab = 0.1)
dev.copy(png,'Results_1B2B.png')
dev.off()
```

18. Información de la sesión R

```
## ----sesioninfo, echo=FALSE----------------------------------------------
sessionInfo()

writeLines(capture.output(sessionInfo()), "Results_sessionInfo.txt")
```