Desarrollo de una aplicación web para obtener información sobre datos de genomas y transcriptomas

Odei Barreñada Taleb
Máster en Bioinformática y Bioestadística
Análisis de datos ómicos

Guillem Ylla Bou
Carles Ventura Royo y **David Merino Arranz**

2 de enero de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Desarrollo de una aplicación web para obtener información sobre datos de genomas y transcriptomas
Nombre del autor:	Odei Barreñada Taleb
Nombre del consultor/a:	Guillem Ylla Bou
Nombre del PRA:	Carles Ventura Royo y David Merino Arranz
Fecha de entrega:	02/2018
Titulación:	Master Bioinformática y bioestadística
Área del Trabajo Final:	Análisis de datos ómicos
Idioma del trabajo:	Español
Palabras clave	RNA-seq, Shiny, *Blattella germanica*

Resumen del Trabajo:

Este proyecto surge de la necesidad de facilitar el acceso y visualización de datos transcriptomicos del laboratorio de Xavier Bellés en el instituto de biología evolutiva (IBE). Estos datos son, mayoritariamente, resultados de análisis de expresión génica mediante RNA-seq del insecto *Blattela germanica* obtenidos en diferentes momentos del desarrollo de la metamorfosis.

El resultado de este proyecto es la creación de una aplicación web que permite la búsqueda, navegación y visualización de datos transcriptomicos albergados por este laboratorio. La aplicación se ha creado mediante el uso de R y su paquete Shiny. Esta combinación de herramientas permite la creación de interfaces interactivas con todo el potencial del lenguaje de R.

La aplicación está diseñada de manera que cada pestaña permite la realización de una función diferente. Hay pestañas para la consulta de valores de expresión, para crear gráficos de barras, buscar genes según su secuencia mediante el software BLAST, analizar los datos de co-expresión entre genes o consultar el diccionario de genes ortólogos con *Drosophila melanogaster*.

Abstract:

This project arises from the need to allow an easy the access and visualization of transcriptomic data of the laboratory of Xavier Bellés from the institute of evolutionary biology (IBE). These data are, mainly, results of analysis of gene expression by RNA-seq of the insect *Blattela germanica* obtained at different stages of the development of metamorphosis.

The final product of this project is the creation of a web application that allows to search, navigate and visualize transcriptomic data hosted by this laboratory. The application was created using R and the Shiny package. This combination of tools allows the creation of interactive interfaces with all the potential of R.

The application is designed in different tabs that each can perform a different function. There are tabs to query expression values, to create bar graphs, search for genes according to their sequence using BLAST software, analyze co-expression data between genes or consult the orthologous gene dictionary with Drosophila melanogaster.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación

Era de las técnicas *ómicas*

No faltan evidencias de que actualmente nos encontramos en la era de los datos ómicos (genómica, transcriptomica, epigenómica, etc.), el aumento de las distintas técnicas para de secuenciación masiva y la reducción de su coste es una de los principales motivos. Esta era se ha caracterizado con este nombre debido al interés de conocer el contenido de los genomas, el código completo que caracteriza cada genoma. Desde la secuenciación completa del primer organismo vivo ,la bacteria *Haemophilus influenzae* [1], se fue aumentando la capacidad y reduciendo el coste a pasos agigantados. El primer borrador del genoma humano se terminó de secuenciar en 2001 [2] y hasta día de hoy se han ido aumentando el número de genomas secuenciados hasta un total de 33.000 especies diferentes totalmente secuenciadas y disponibles para uso público en el NBCI. Desde la primera técnica de secuenciación, propuesta por F. Sanger en 1975 [3] el panorama ha cambiado enormemente. Si el método de Sanger requería de mucho material, tiempo y dinero para llevarse a cabo, el estado del arte en este campo utiliza técnicas de ultra secuenciación (o secuenciación de alto rendimiento) capaces de

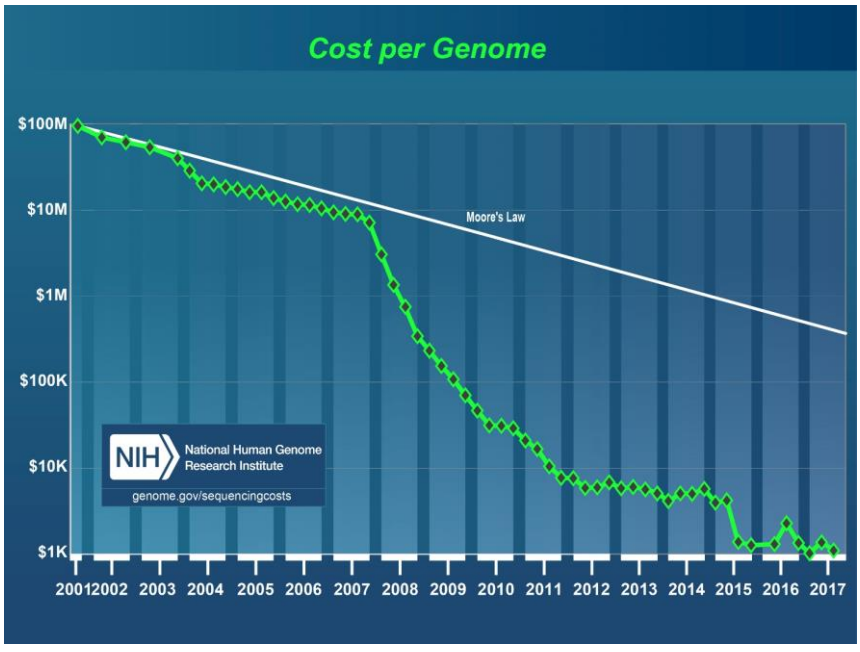


Ilustración 1: Evolución del precio por secuenciación de genoma.

secuenciar genomas enteros en un tiempo récord y a precios muchísimo más asequibles, incluso por debajo de los 1000 $/genoma.

Esta disminución del precio ha provocado el uso exponencial de la secuenciación en todos los ámbitos científicos generando un crecimiento igual de los datos genómicos generados. Concretamente en el repositorio público de secuencias de nucleótidos Sequence Read Archive (SRA) están albergados actualmente 3.608.535 secuencias diferentes (de las cuales 1.152.109 son genomas completos) pero este no es ni mucho menos la única base de datos de secuencias genómicas. Cabe destacar que estos son únicamente los datos de secuenciación brutos, directamente obtenidos de la secuenciación, el procesamiento de estos datos crea datos derivados que no siempre son los mismos debido a las configuraciones de las funciones de ensamblado de secuencias, lo que requiere de que se guarden ambos datos para poder corregir los cálculos si fuera necesario.

Una vez alcanzado un hito de este calibre, la secuenciación de los genomas, la comunidad científica se ha volcado en el siguiente objetivo, la comprensión del genoma y su efecto en el fenotipo. Y para este paso ha sido crucial el desarrollo de las técnicas que analizan no solo en DNA sino el intermediario entre este y el fenotipo resultante, el RNA. De hecho, este campo ha cobrado tanta importancia que ya es reconcomido como un campo independiente denominado transcriptomica.

La transcriptomica estudia todas las variantes del RNA (RNAr, RNAt, RNAm, RNAi, miRNA) aunque debido a su relevancia se centra principalmente en el RNA mensajero y en el microRNA. Su estudio se realiza mediante microarrays (técnica ya practicante obsoleta) y sobre todo, mediante la tecnología emergente de secuenciadores de alto rendimiento, denominada RNA-seq [5]. La tecnología RNA-seq se basa en la secuenciación del DNA complementario de un RNA, donde a posteriori se realiza una cuenta del número de veces que se secuencia cada transcrito para hacer una estimación de la cantidad de RNA presente en la muestro lo que se puede traducir en la expresión total de ese transcrito o gen.

Secuenciación de RNA.

La tecnología de RNA-seq cubre todo el proceso a partir de una muestra de RNA hasta la obtención del transcriptoma completo de la muestra. En siguiente imagen (Ilustración 2) se puede observar un protocolo genérico de RNA-seq cosiste en 5 pasos principales: 1) preparación de la muestra de RNA, 2) construcción de la biblioteca de DNA complementario, 3) secuenciar los fragmentos de DNA, 4) Alineamiento de los fragmentos, 5) Obtención de los niveles de expresión de cada gen [6]. Cada uno de los pasos genera un nuevo objeto o dato nuevo derivado del análisis del anterior, lo que implica por un lado la creación de más datos y la generación de datos nuevos que inevitablemente supone una pequeña alteración de los datos reales. Contando con un ligero margen de error en cada análisis, al final de todo el proceso se puede acabar con datos muy alejados de la realidad (aunque se intente evitar esta situación a toda costa). Al final de todo el resultado típico es una matriz de expresión donde se indica en cada fila los genes analizados y en las columnas el nivel de expresión de cada gen en cada muestra.

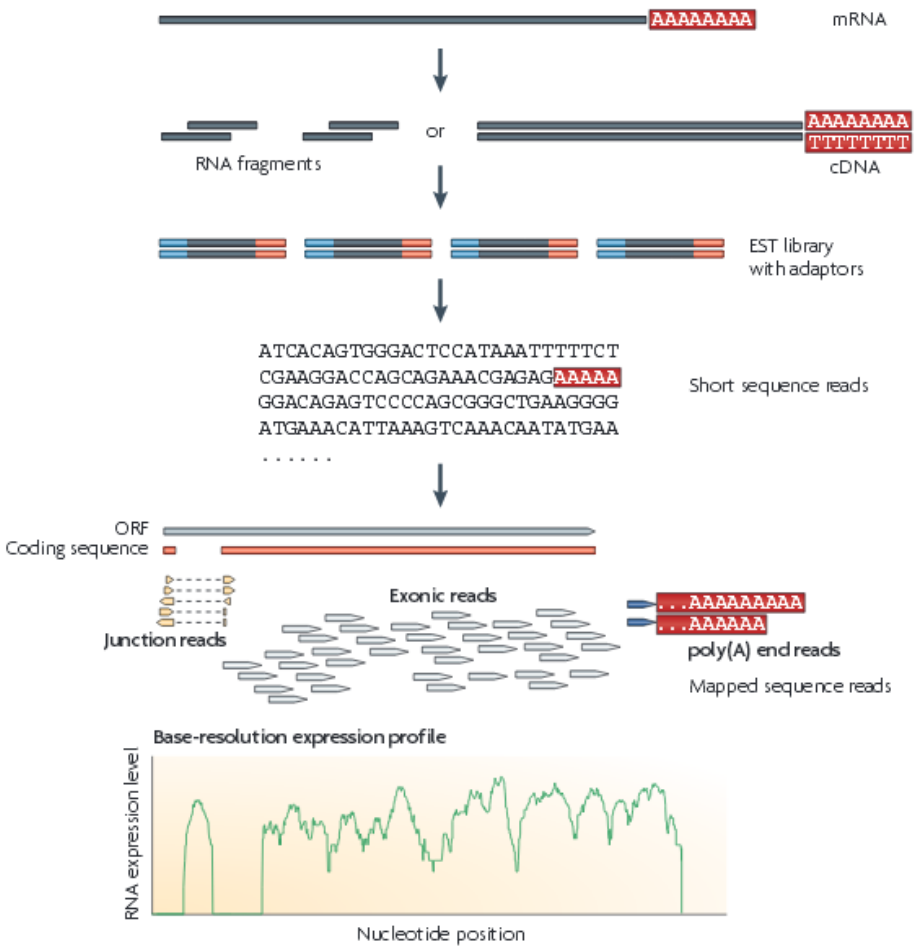


Ilustración 2: Proceso esquemático estándar de la metodología de RNA-seq.

Instituto de biología evolutiva

El laboratorio de la evolución de la metamorfosis en insectos, liderado por Xavier Bellés forma parte del instituto de biología evolutiva. Son grupo dedicado al estudio de la evolución de los insectos, y para esta finalidad entre otras metodologías utilizan las técnicas de RNA-seq para la obtención de datos transcriptomicos. En este laboratorio trabajan con diferentes modelos de insecto con el fin de entender el proceso evolutivo de la metamorfosis. Entre los animales más estudiados están *Drosophila melanogaster*, *Blattella germanica*, *Tribolium castaneum*.

Uno de los más estudiados en este laboratorio es la cucaracha común o *B. germanica*, debido a las características de su metamorfosis. Se trata de un insecto con una metamorfosis hemimetabola que se considera un ancestro de la ampliamente estudiada metamorfosis común o holometábola.

A lo largo de los últimos años, este laboratorio a utilizado técnicas de secuenciado de nueva generación para el análisis de *B. germanica* bajo diferentes condiciones. En total se tienen un total de 55 sets de datos de genes diferentes.

Uno de los sets de datos más extensos son los datos de RNA mensajero obtenidos en diferentes estadios de desarrollo de la cucaracha. Se obtuvieron muestras de *B. germanica* en 11 etapas diferentes del desarrollo y se secuenciaron los mRNA por duplicado. El resultado son un total 22 sets de mRNA en 11 estados de desarrollo diferentes. Estas etapas han sido ampliamente estudiadas por este laboratorio, cada una de ellas representa un momento especial del desarrollo y pose un contexto hormonal característico para la metamorfosis de *B. germanica* [7].

Usando el mismo esquema anterior también se han analizado la expresión de microRNA (o miRNA son moléculas de RNA pequeñas, de una longitud de entre 20 y 25 pares de bases), a lo largo de estas 11 etapas del desarrollo de *B. germanica*. En este caso el resultado son otras 22 bibliotecas (o sets de datos) de hasta 167 miRNA diferentes [7].

Adicionalmente, en el laboratorio se han obtenido otros sets de datos RNA-seq de *B. germanica* en distintos tejidos. En este caso no han sido los datos específicos respecto a las etapas del desarrollo, sino más bien sets relativos a la expresión en distintos tejidos. En total son otros 11 sets, sin replicados, de hasta 4 tejidos diferentes (Glandula targal, Ovario, Ala y cuerpo graso) y obtenidas en etapas evolutivas especificas (N6, N5) o bajo condiciones concretas (grupo control o tratado con: hormona del crecimiento, ecdisoma, o mediante estrés hídrico) [8].

Todos los datos previamente descritos están publicados en la base de datos Gene Expression Omnibus [9] con distintos números asociados detallados en los artículos [7], [10].

Aprovechando los datos descritos se han generado ficheros con información necesarios para la completa compresión de la especie modelo *B. germanica*, Entre estos ficheros generados están: una matriz con coeficientes de co-expresión entre genes de *B. germanica* al largo del desarrollo, un diccionario de genes ortólogos entre esta especie y *D. melanogaster*, y el proteoma conocido de la especia estudiada.

El primero se trata, como indica su nombre, una matriz en la que se comparan por parejas todos los genes anotados de *B. germanica*, y se puntúa según su coeficiente de correlación de Pearson. Este coeficiente valora la similitud del perfil general de ascenso/descenso de niveles de expresión entre diferentes fases del desarrollo. El resultado es la valoración de co-expresión entre cada pareja genes.

La tabla de genes ortólogos se ha obtenido mediante el protocolo de “reciprocal best blast hit (RBBH)” en la cual se realizan dos ciclos de blast. Primero uno de cada gen de la cucaracha contra la base de datos de *Drosophila* y luego otra tirada de cada gen de la mosca contra la base de datos de *B. germanica*, de esta manera los genes que coincidan con mejor puntuación en ambos casos serán considerados como ortólogos.

También se ha utilizado el proteoma completo conocido de *Blatella germanica* obtenido a partir del recientemente secuenciado genoma gracias a la colaboración de diferentes laboratorio al proyecto i5k [11].

Visualización de los datos

Uno de los mayores retos de la actualidad científica es sin duda el análisis e interpretación de los datos generados. Teniendo en cuanta la cantidad enorme de posibilidades y variantes presenten en proceso del análisis de datos es lógico que no existe un claro consenso en las herramientas y parámetros a utilizar [12]. Lo que está claro es que las herramientas capaces de visualizar el resultado de unos datos tan complejo son una excelente ayuda para los investigadores. Por

ejemplo, sin los navegadores genómicos la exploración y observación de datos genómicos sería mucho más complicado y, por ende, quedaría relegado a un grupo más pequeños de personas lo que repercutiría negativamente a la difusión de los mismos. Las herramientas de visualización de datos genómicos son un puente entre los datos generados o raw data y su interpretación biológica.

1.2 Objetivos del Trabajo

El objetivo último de este proyecto es facilitar el acceso de datos genómicos a personal sin conocimiento de bioinformatico y de programación. Para ello se ha planificado la creación de una herramienta web, una aplicación, donde albergar los datos generados, navegar por ellos y poder visualizarlo, todo esto de una manera clara y sencilla. Los tres pilares fundamentales para esta aplicación son: que la herramienta esté basada en la web, para que cualquiera pueda acceder desde cualquier plataforma; que sea interactiva, evitando así la introducción de datos mediante líneas de comando complejas; y que sea *user-friendly* (fácil de usar), sencillo y con la visualización de datos clara y bonita.

Las distintas funcionalidades están separadas en diferentes pestañas para saber en todo momento los datos que se usan y el significado de los datos de salida. La herramienta se caracteriza por ofrecer la visualización de los datos albergados y para ello se han programado distintas funcionalidades entre las que elegir. Las principales capacidades de la herramienta son: la búsqueda de genes según su ID, su secuencia (tanto nucleótididica o de aminoácidos) o sus ortólogos en otras especies como *D. melanogaster,* la visualización de datos de expresión de los genes y la comparación de los datos de expresión entre distintos genes, más adelante se presentarán con detalle todas estas funcionalidades y sus peculiaridades.

1.3 Enfoque y métodos

Pilares fundamentales de la app

Para alcanzar el objetivo planteado, se necesita crear un producto que cumpla las características deseadas. Un producto accesible desde cualquier plataforma y lugar, una interfaz interactiva que facilite la interacción con el usuario y un aspecto sencillo e intuitivo.

Accesibilidad

En un principio esta aplicación está construida para un uso a nivel interno, dentro del propio instituto ya que está diseñada para cubrir las necesidades de estos investigadores. Pero como puede ser una herramienta útil se construye pensando en la posibilidad de que sea abierta a toda la comunidad científica, pero especialmente al personal interesado en los datos que permite visualizar. Estos investigadores serán aquellos que trabajen con *B. germanica* y más concretamente que estudien el desarrollo de la metamorfosis en este organismo.

User-friendly

Otra de las bases de la aplicación es que tiene que poder utilizarse con facilidad por cualquier persona. Por esta razón se ha fijado como objetivo crear la aplicación con una interfaz lo más sencilla posible manteniendo las máximas funcionalidades. La idea es evitar que personal sin avanzados conocimientos en bioinformática tener que lidiar con complicados programas sin interfaz gráfica que funcionan mediante líneas de comando.
User-friendly es un término anglosajón ampliamente utilizado en programación que hace referencia a sistemas construidos de pensando en su facilidad de uso y su buen diseño.

Interactiva

En la misma línea, y estrechamente relacionado con la experiencia del usuario final, se ha tomado la decisión de crear una herramienta que se manipule de manera interactiva. Con esta decisión se facilita la interacción entre el usuario y la aplicación.

Herramientas utilizadas

Se han utilizado principalmente tres herramientas; R como base de toda la aplicación, Rstudio como entorno de desarrollo (IDE) de R y, por último, el paquete de R Shiny capaz de crear aplicaciones web interactivas a partir de código de R.

De manera secundaria se han utilizado otros lenguajes para este proyecto como algunas líneas en HTML (lenguaje dedicada a la creación de páginas web) y CSS (lengua de diseño gráfico). U otros paquetes de R como son DT (visualización de tablas de datos) o igraph (generación de gráficos de redes) entre otros.

R (lenguaje de programación)

Es el principal lenguaje de programación en el ámbito de la bioinformática y el análisis de datos. Es un lenguaje de código abierto y colaborativo donde cada usuario puede aportar su código a la comunidad. Uno de sus mayores fortalezas es la posibilidad de usar código de aportado por otros programadores. Esto se realiza mediante “paquetes”, pequeñas aportaciones de código fácilmente instalables que aportan a cualquier usuario el trabajo de otra gente. El lenguaje de R está especializado en el manejo de datos, realización de cálculos estadísticos y visualización de datos.

Para este proyecto será utilizado como la base de la aplicación. La aplicación completa se construirá con código de R, además de otras funcionalidades implementadas en forma de paquetes de este programa.

Rstudio

Rstudio es la interfaz de trabajo (Entorno de desarrollo integrado, IDE) más popular de R. Permite que el trabajo de R sea en una interfaz visualmente más atractiva con muchísimas facilidades a la hora de escribir código.

Shiny

Shiny es un paquete de R con el cual se pueden crear aplicaciones interactivas. Este paquete convierte funciones escritas en R en código de HTML (propio de páginas web) y de JavaScript (utilizado ampliamente en conjunto con HTML para

crear páginas web dinámicas). Una vez instalado este paquete ya se pueden acceder a toda una serie de funciones nuevas que permiten la creación de la aplicación.

Esquema

La Ilustración 3 resumen esquemáticamente el funcionamiento de las herramientas utilizadas. De manera muy resumida se utilizan estos elementos (R y Shiny) para programar una aplicación, en forma de página web, basada en HTML, JavaScript y CSS.

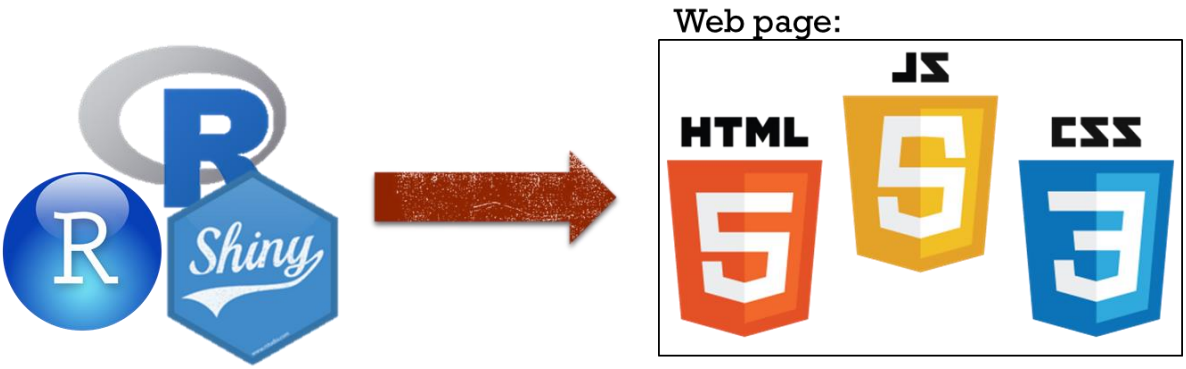


Ilustración 3: Esquema de metodología utilizada

Metodología

La manera de estructurar de la aplicación será modular. Debido a que se quieren integrar muchas funcionalidades de diferente índole en una misma aplicación, se ha optado por separar cada función de la aplicación y presentarla en un espacio diferente. Para ello la mejor manera es la creación de un menú de pestañas en la parte superior de la pantalla donde se pueda navegar entre las pestañas. Dentro de cada pestaña se creará la interfaz que sirva para la realización de cada una de las funciones que se requieran.

1.4 Planificación del Trabajo

Recursos

Las necesidades mínimas de este proyecto son durante la primera fase un ordenador en el cual poder instalar el software adecuado para poder trabajar. En la segunda fase la aplicación, se necesita un servidor donde albergar la aplicación. En este caso, el ordenador utilizado ha sido mi propio ordenador portátil, el cual ha estado a la altura para este proyecto. Respecto al servidor, se ha montado uno para la ocasión ejecutando Ubuntu 16.04 donde se ha instalada las herramientas necesarias.

Planificación y calendario

Dado la estructura modular de la aplicación, se ha planificado según cada una de sus unidades principales, las distintas funcionalidades. Se han decidido una serie de tareas a cumplir y una temporización para cada uno de ellas. El calendario se ha organizado con tiempos de ejecución de 1 semana, aunque se ha adaptado este tiempo según la dificultad estimada de cada tarea.
Las semanas con menor carga de trabajo se aprovechaba el tiempo para redactar las entregas parciales y/o la memoria final.

Tabla 1: Calendario de planificación de tiempo

Semanas / Metas	S1 (16 Oct)	S2 (23 Oct)	S3 (30 Oct)	S4 (6 Nov)	S5 (13 Nov)	S6 (20 Nov)	S7 (27 Nov)	S8 (4 Dic	S9 (11 Dic)	S10 18 Dic)	S11 (25 Dic)	S12 (1 Ene)
Creación de la base de la aplicación												
Añadir buscador de genes por ID												
Añadir visualización de datos por diagrama de barras												
Introducir datos de secuencias												
Añadir buscador por secuencias (BLAST)												
Permitir búsquedas por genes ortólogos												
Demostración en el laboratorio												
Añadir un visualizador genómico												
Redacción de la memoria												
Añadir datos de co-expresión												

1.5 Resumen de productos obtenidos

El producto final, tal y como se había planeado es una aplicación web que permite la visualización de datos transcriptomicos obtenidos por el laboratorio de biología evolutiva. El servidor se ejecuta mediante dos archivos con código de R y una carpeta con los datos a utilizar. Tal y como indica los manuales de Shiny la aplicación se ejecuta por gracias a dos archivos con código de R. Por un lado *ui.R* (con el código para crear la interfaz gráfica con la que el usuario interactúa) y por otro *server.R* (con el código de R capaz de preparar los datos adecuados que el usuario demanda). En total más de 500 líneas de código nuevo que permiten la ejecución de la aplicación creada.

La aplicación está organizada de manera que tenga tantas pestañas como herramientas diferentes tenga. Actualmente se ha terminado de trabajar en el proyecto con un total de 7 pestañas. Los títulos de cada pestaña informan de su función, y son (traducidos del inglés): Tabla de expresión, Graficas de expresión, multi-lineal, Blast, Co-expresion, Ortologos Dmel, Sobre la web. En los próximos capítulos se detalla la función de cada una de estas herramientas.
El servidor va a ser alojado en un servidor propio del IBE, pero provisionalmente se ha montado un servidor propio al cual se puede acceder mediante su IP: 130.206.125.92:3838/obt/. Cuando sea publicado en el IBE existirá un enlace directamente desde la página web del IBE.

1.6 Introducción de siguientes capítulos

Una vez detallado el marco actual, en los siguientes capítulos se profundiza sobre el producto creado. Empezando por la descripción detallada del producto, continuar enmarcando las limitaciones detectadas durante su creación para terminar comentando las posibilidades de mejorar y ampliar la aplicación.

Descripción del producto creado

A lo largo de este capítulo se va a describir detalladamente toda la aplicación. Primero detallando la funcionalidad de cada una de las pestañas incluidas para posteriormente, y ayudado de ejemplos, simular el uso que se le daría a la aplicación

Limitaciones

Otro capítulo esta memoria será el comentario sobre las limitaciones actuales de la aplicación. Ya que son aspectos a tener en cuenta si posteriormente se quieren ampliar la aplicación.

Perspectivas de futuro

Por último, se intentará plantear las posibles vías de crecimiento de este proyecto en caso de poder ampliar el tiempo de dedicación.

2. Descripción del producto creado

Como se ha detallado anteriormente, el fruto de este proyecto es la creación de una aplicación mediante la cual facilitar la visualización de datos transcriptomicos. En este capítulo se detallará la aplicación construida. Además, para facilitar la comprensión de la utilidad del mismo se utilizarán algunos ejemplos de su uso más básico.

2.1 Funcionamiento de la aplicación

La app se ha estructurado en 7 pestañas diferentes, cada una de ellas con una funcionalidad concreta. Los títulos para cada una de las pestañas (traducidos del inglés son): Tabla de expresión, Graficas de expresión, multi-lineal, Blast, Co-expresion, Ortologos Dmel. A continuación, se detalla su funcionamiento uno a uno.

Antes de comenzar, comentar la barra de selección de pestañas (Ilustración 4). Está ubicada en la parte superior de la pantalla y tiene un título y 7 pestañas con sus respectivos rótulos. Está presente en todas las pestañas para permitir una navegación fluida a través de todas las herramientas de la aplicación. El título es el nombre de la aplicación, que de manera provisional se la ha denominado “ShinyAPP TFM”. Sucesivo al mismo se declara el número de versión de la misma. Las siguientes pestañas son los títulos junto a un icono representativo. La segunda pestaña “graficas de expresión” es la única que posee un pequeño meno desplegable para acceder a tres sub-pestañas diferentes, pero las tres con la misma función, la realización de gráficos de barras.

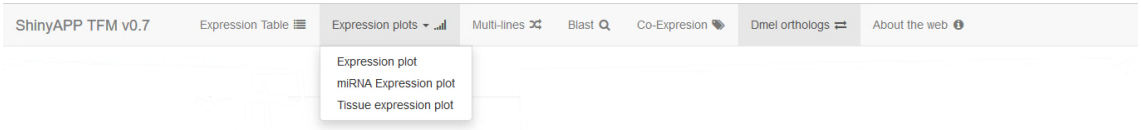


Ilustración 4: Aspecto del menú de navegación de la aplicación.

Tabla de expresión – Expression table

Esta pestaña está dedicada a la consulta de los datos necesarios en formato tabla. En la zona superior aparece durante unos segundos una notificación con las primeras indicaciones, aunque no son necesarias ya que la interfaz es sencilla y comprensible. La mayor parte de la pantalla la ocupa una tabla con 13 columnas (2); la primera, la ID de los genes de *B. germanica* ordenados alfabéticamente, la segunda el nombre del gen (únicamente cuando si es conocido) y las 11 columnas restantes son los datos de expresión en FPKM de cada uno de los estadios de desarrollo estudiados. En el extremo superior izquierdo se ha colocado un controlador deslizante (1) con el cual personalizar el número de decimales a representar en la tabla. Por encima de la tabla en la esquina superior derecha se ubica un cuadro de búsqueda (3) en el cual se puede escribir el ID o nombre de los genes para acceder directamente a la información deseada.

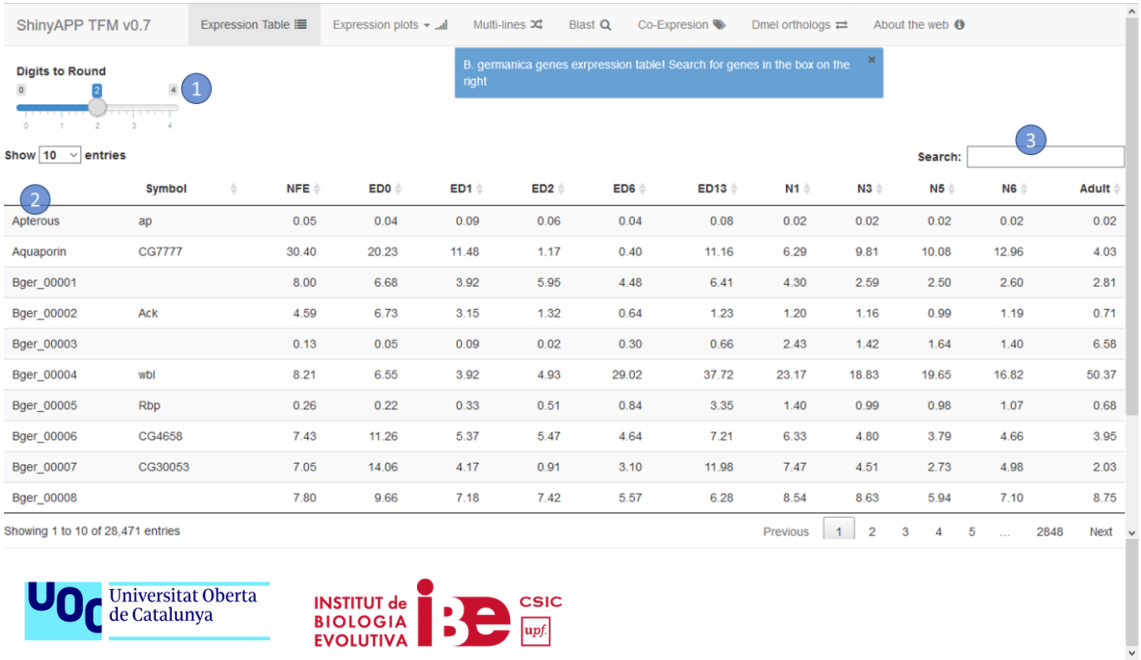


Ilustración 5: Tabla de expresión de genes de B. germanica

Graficas de expresión – Expression plots

Esta pestaña permite la visualización de los datos de expresión del gen de interés en las 11 etapas de desarrollo estudiadas. En la parte superior hay, por un lado sección para buscar o seleccionar el gen deseado y por otro un enlace con la base de datos apollo [13]. El enlace envía al navegador genómico apollo, directamente a la posición del gen consultado. Este navegador permite la visualización de la estructura del gen (exones, CDS, zonas UTR…), al mismo tiempo permite el acceso a la secuencia completa o a las secuencias de cada una de las regiones. Debajo de esto, se crea el grafico de barras que permite la visualización de los datos de expresión por cada una de las etapas del desarrollo. El gen se puede cambiar a en cualquier momento solamente clicando en el meno desplegable y eligiendo (o escribiendo) el nuevo gen de interés. Inmediatamente se actualizará el enlace y la gráfica a la nueva selección.

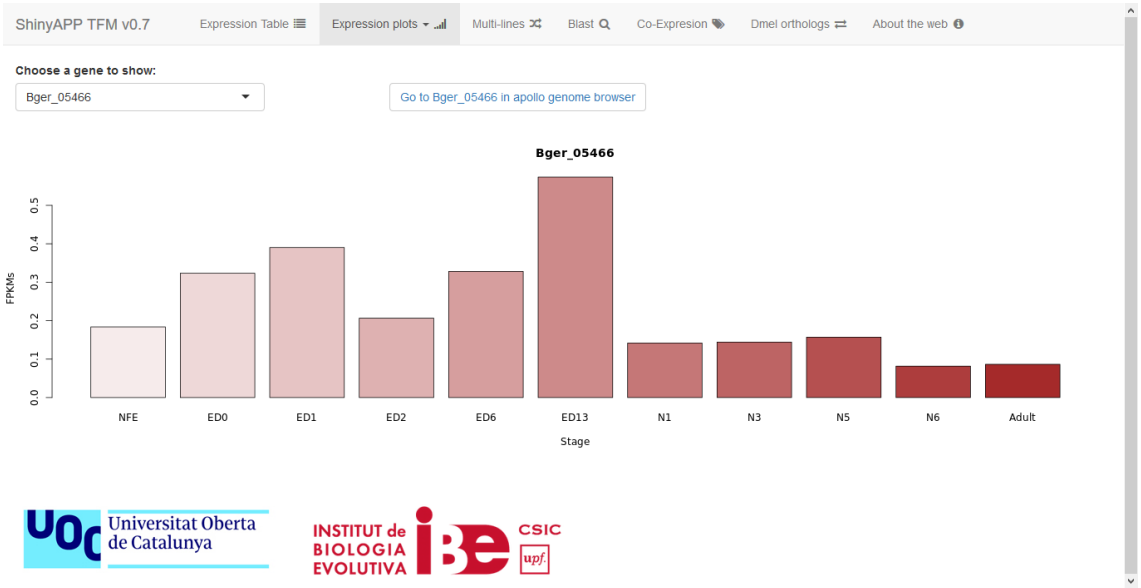


Ilustración 6: Pestaña "Expression plot"

Las restantes subpestañas (*miRna expresión plot* y *Tissue expresión plot*) únicamente generan la gráfica de expresión de la misma manera explicada anteriormente. La única diferencia entre ambos es que aceden a bases de datos diferentes, la primera representa los datos de expresión de 167 miRNA conocidos de *B. germanica* en los mismos estadios de desarrollo, la segunda

representa visualmente la expresión génica de 11 situaciones concretas de 4 tejidos diferentes: Glándula targal (4), Ovario (4), Alas (2) y cuerpos grasos (1).

Ilustración 7: Aspecto de las pestañas "miRNA expresion plot" (izq.) y "Tissue expresion plot" (dcha.)

Multi-lineal

Esta pestaña se creó bajo el interés de poder comparar varios perfiles de expresión simultáneamente. En la parte superior, como en los otros casos, se encuentra una caja donde introducir (o seleccionar de la lista desplegable) los genes que quieres visualizar en la gráfica. En este caso para facilitar la visualización de múltiples datos de expresión, se trata de un gráfico delineas, donde cada línea representa los datos de expresión del gen en cada uno de los 11 estadios del desarrollo. Con cada gen elegido se genera la línea con los datos correspondiente y se añade el gen a la leyenda de la gráfica.

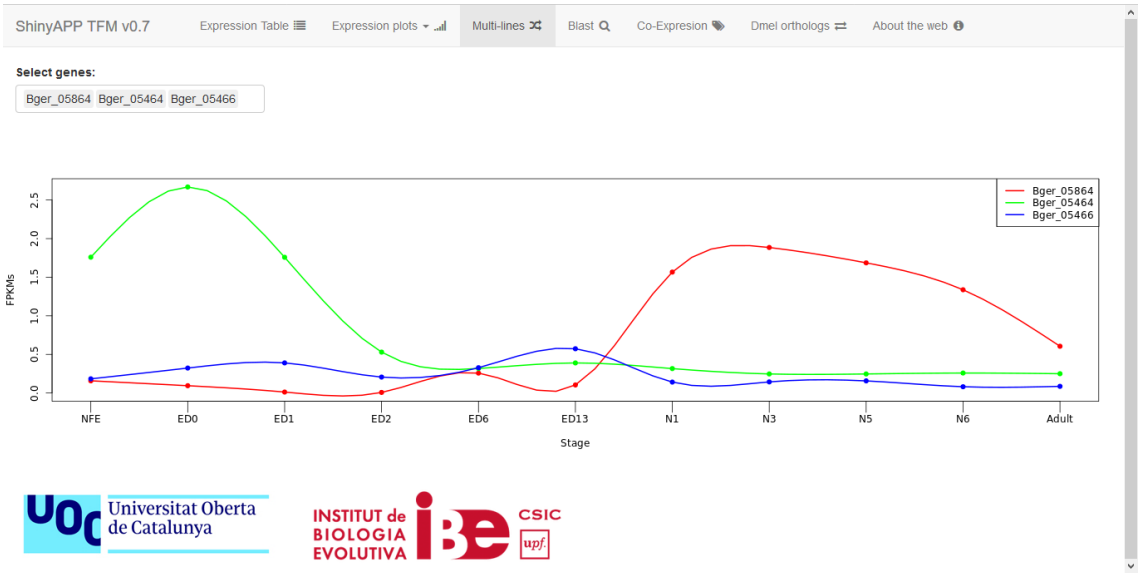


Ilustración 8: Demostración de la funcionalidad para comparar genes mediante gráficos de múltiples líneas con los primeros tres genes de la base de datos.

Blast

BLAST son las siglas para Basic Local Alignment Search Tool (traducido al castellano, herramienta básica de búsqueda de alineación local). Se trata de una herramienta informática para el alineamiento de secuencias implementada en el NBCI. La aplicación creada utiliza este mismo software, instalado en el servidor para su uso local.

Esta pestaña (Ilustración 9) tiene dos maneras de introducir la secuencia a consultar, en primer lugar, una caja de texto donde pegar la secuencia de interés (1), en segundo lugar, se facilita una opción para subir el archivo con la secuencia deseada (2). El formato ideal es el fasta (una primera línea opcional precedida por el signo *mayor que* “>” indicando el nombre de la secuencia, y a continuación, a partir de la segunda línea la secuencia). Funciona tanto con secuencias nuecleotidicas como para secuencias peptídicas.

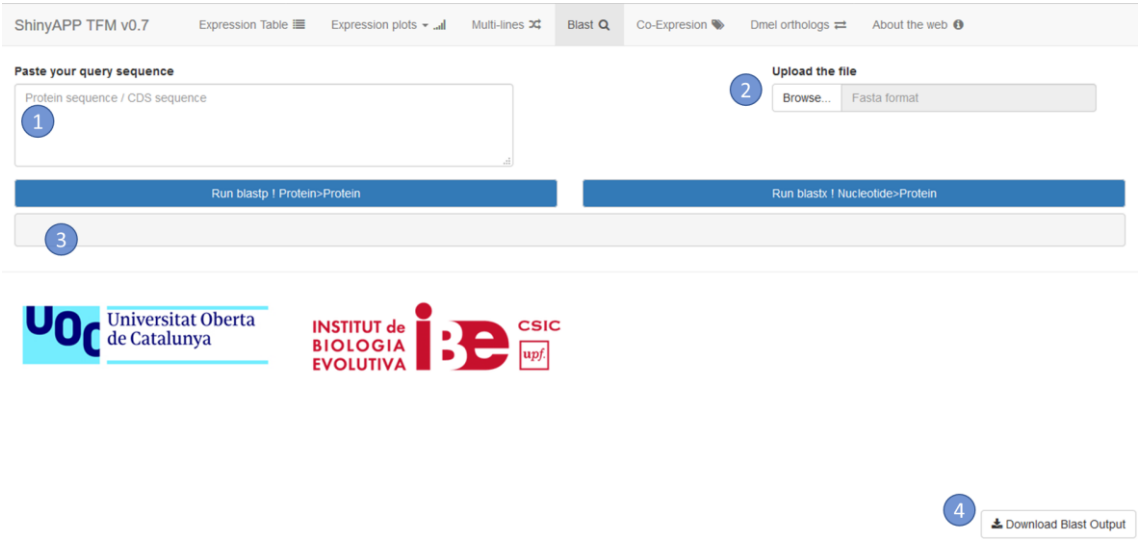


Ilustración 9: Pestaña de BLAST con inputs en blanco.

Debajo de las opciones de introducción de la secuencia se ubican dos botones: uno para realizar un *blastp* (una variante del alineamiento BLAST) optimizado para la alineación de una secuencia de aminoácidos contra una base de datos de secuencias peptídicas. Y el otro botón para realizar un *blastx*, otra variante, que antes del BLAST realiza una conversión de la secuencia de nucleótidos a aminoácidos para alinear esta secuencia peptídica traducida contra un base de datos de proteínas. Cada botón realiza una acción diferente pero el resultado es similar, por debajo de estos se genera el resultado del BLAST seleccionado dentro del recuadro remarcado (3).

Por último, fijado en la parte inferior de la pantalla se encuentra un botón etiquetado como “Download Blast Output” (4) que permite la descarga en formato de texto del resultado del BLAST directamente al ordenador.

En la siguiente imagen (Ilustración 10) se puede observar cómo tras la introducción de una secuencia en la caja izquierda (1) y pulsar el botón correspondiente se genera el resultado por debajo de los botones (3).

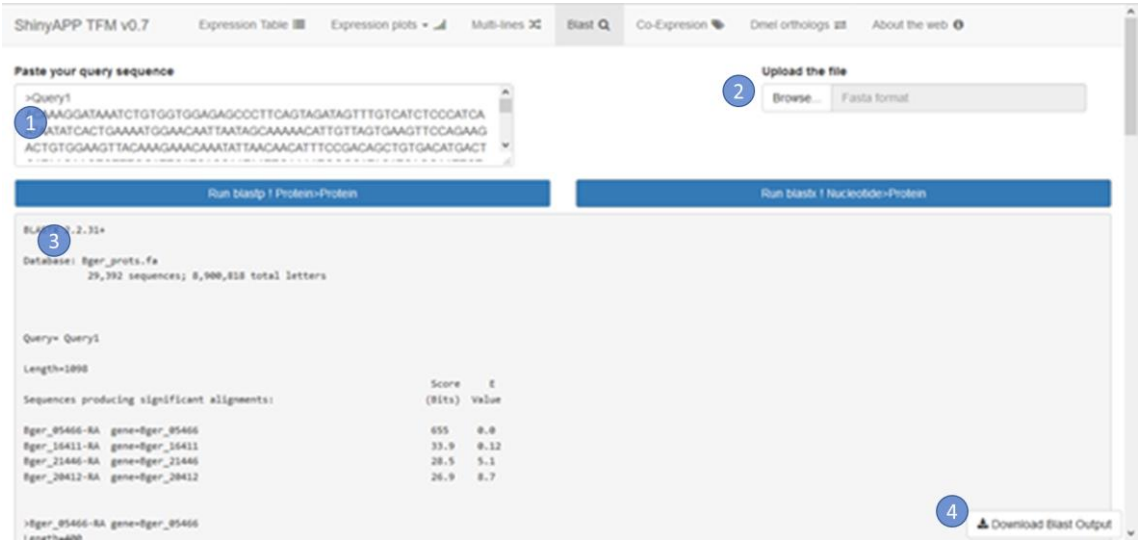


Ilustración 10: Ejemplo de introducción de secuencia y de resultado

Co-expresion

Esta pestaña permite la exploración por los datos de co-expresion obtenidos mediante el análisis de expresión entre todas las parejas de genes estudiados. Esta pestaña esta divida en dos partes, en el lado izquierdo hay un panel con las opciones de configuración (1) y a la derecha una zona donde se pueden visualizar los datos que posee dos pestañas diferentes (2), aunque a primera vista este totalmente en blanco (Ilustración 11).

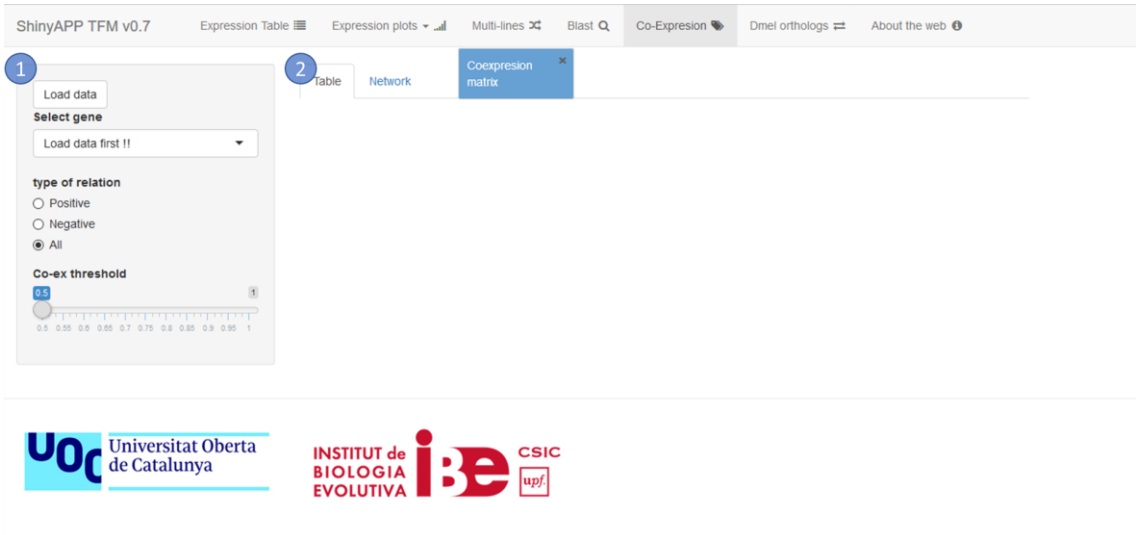


Ilustración 11: Pestaña "Co-expresión"

Esta pestaña es dinámica, cambia la interfaz cuando se realizan algunas acciones. Antes de nada y como sugiere el primer botón del panel de configuración “Load data”, se ha de clicar en el mismo para cargar los datos de expresión. Este botón permite visualizar los datos de co-expresión en el panel derecho dentro de la pestaña “table” (2) (Ilustración 12).

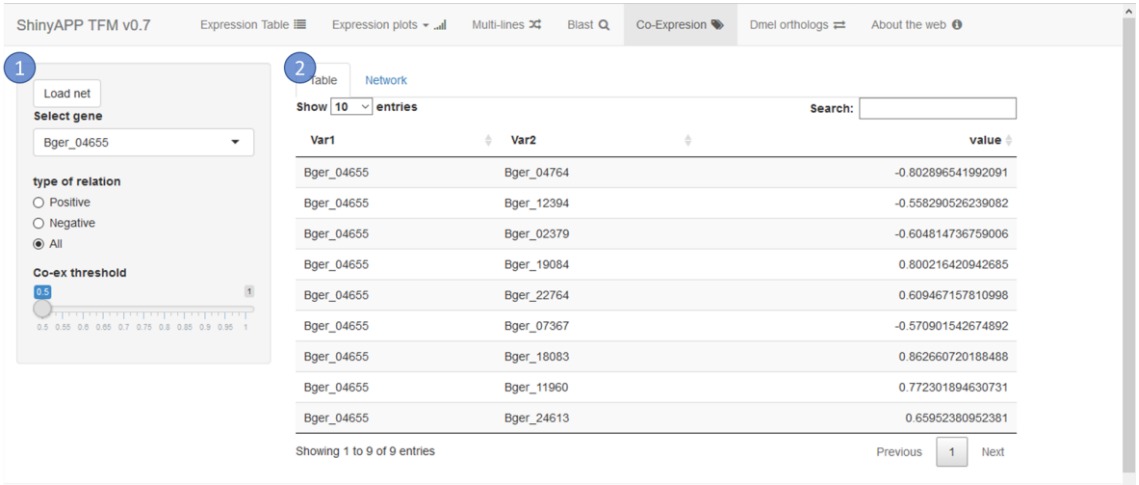


Ilustración 12: Pestaña "Co-expresión" tras cargar los datos.

Tras esta acción, además de la visualización de la tabla cambia el botón y es sustituido por un botón nievo etiquetado como “load net” al mismo tiempo que permite utilizar el menú desplegable para seleccionar cualquier gen del genoma de *B. germanica* sobre el cual mostrar los datos de expresión. Mostrar la tabla entera seria ineficaz a la vez que un derroche de recursos del servidor debido a la magnitud de la matriz de datos. Por debajo del menú desplegable hay dos configuraciones más para filtrar el resultado de los genes visibles en la tabla. Por un lado, un filtro para visualizar correlaciones positivas (que mantiene el mismo perfil de expresión), negativas (con perfiles de expresión opuesto) o todas (mostrar todos los genes correlacionados). Por otro lado, se puede filtrar las correlaciones hasta cierto umbral mediante un controlador deslizante. El controlador tiene posiciones desde 0.5 hasta 1, comienza en 0.5 y cuando más alto se coloque más restrictivo será el filtro.

Cada vez que se clique en el botón de “Load net” se genera una gráfica tipo red con el gen seleccionado en el centro y los genes con correlación (y que superen los filtros) enlazados al primero dentro del panel derecho en la pestaña “network” tal cual se muestra en la siguiente imagen (Ilustración 13).

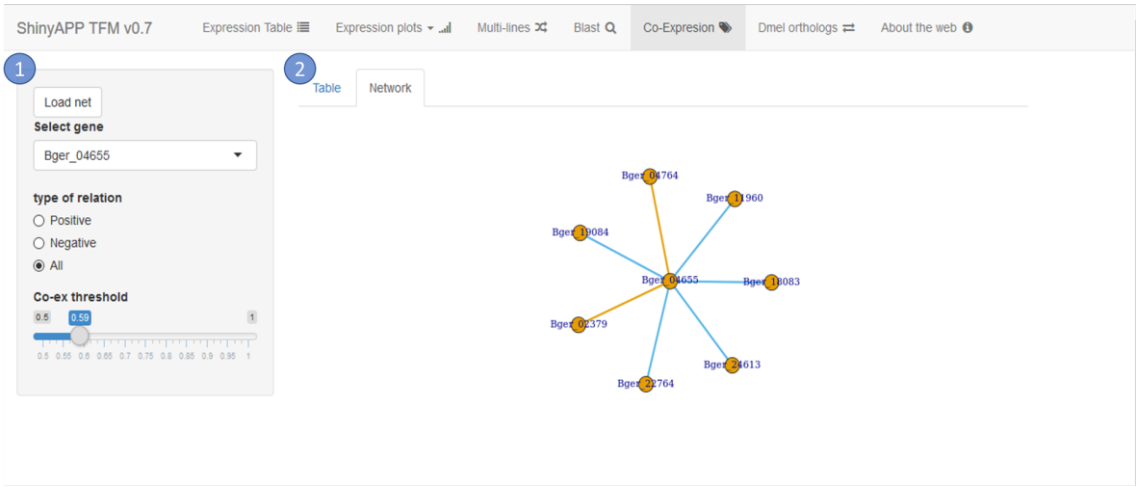
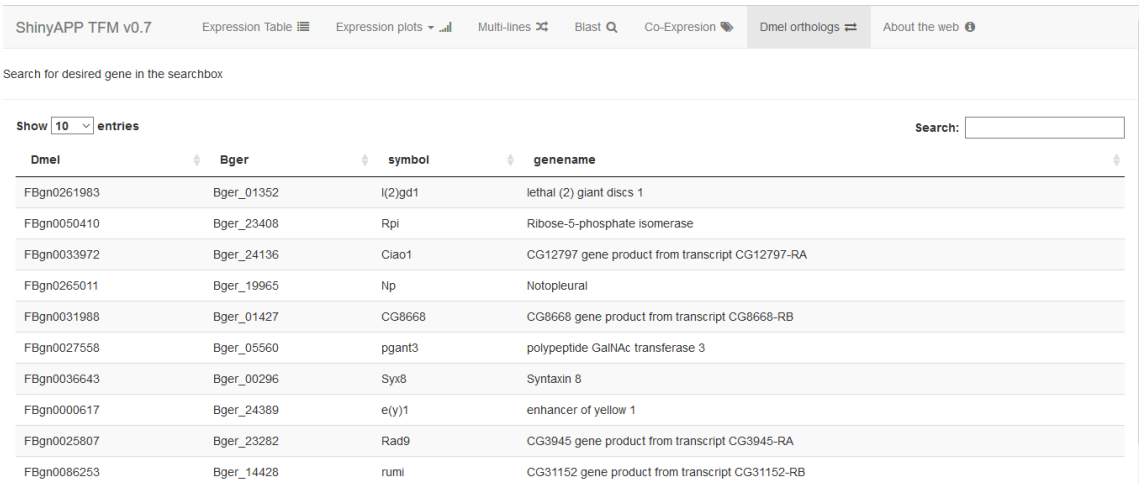


Ilustración 13: Ejemplo de grafica de red generada por la pestaña "Co-expresión".

Ortologos Dmel – Dmel Orthologs

En esta pestaña se puede visualizar la tabla de genes ortólogos entre *B. germanica* y *D. melanogaster*. La tabla de genes ocupa toda la pantalla y está compuesta por 4 columnas (ver Ilustración 14). La primera tiene las ID de los genes de melanogaster obtenida de Flybase [14], la segunda columna los genes ortólogos detectados de *B. germanica*, la tercera las abreviaturas de los nombres de los genes y la última columna el nombre completo de los genes. De manera



Dmel	Bger	symbol	genename
FBgn0261983	Bger_01352	l(2)gd1	lethal (2) giant discs 1
FBgn0050410	Bger_23408	Rpi	Ribose-5-phosphate isomerase
FBgn0033972	Bger_24136	Ciao1	CG12797 gene product from transcript CG12797-RA
FBgn0265011	Bger_19965	Np	Notopleural
FBgn0031988	Bger_01427	CG8668	CG8668 gene product from transcript CG8668-RB
FBgn0027558	Bger_05560	pgant3	polypeptide GalNAc transferase 3
FBgn0036643	Bger_00296	Syx8	Syntaxin 8
FBgn0000617	Bger_24389	e(y)1	enhancer of yellow 1
FBgn0025807	Bger_23282	Rad9	CG3945 gene product from transcript CG3945-RA
FBgn0086253	Bger_14428	rumi	CG31152 gene product from transcript CG31152-RB

Ilustración 14: Pestaña de genes ortologos entre Drosophila y Blattella

similar a otras tablas de la aplicación, posee un cuadro de búsqueda con el cual escribir el gen deseado (tanto por nombre, abreviación o ID) y acceder directamente al mismo.

Sobre la web - About the web

Esta es la última pestaña de la aplicación, pero es a la que se accede primero por defecto. Posee tres secciones diferenciadas; primero una información introductoria sobre el proyecto (1), segundo, algunos detalles respecto al origine de los datos (2) y tercero, el articulo al cual acudir para acceder a más información o citar la herramienta si te ha sido de utilidad (3).

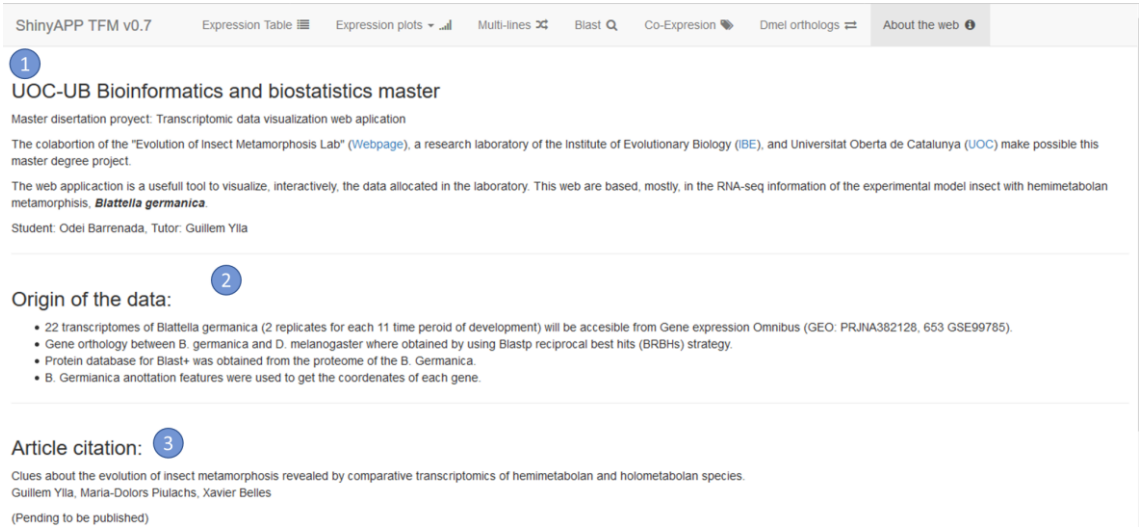


Ilustración 15: Muestra de la pestaña "About the web" de la aplicación.

2.2 Ejemplo de uso

Dependiendo la información de partida y el objetivo la aplicación se utilizará de manera distinta. La aplicación está diseñada para que sea muy intuitiva y se pueda utilizar sin ningún tipo de conocimiento previo. En caso de tener problemas y resolver dudas, en este capítulo vamos a repasar las situaciones más habituales a las que esta aplicación nos podrá ser de gran utilidad y la manera de actuar para obtener el resultado deseado. Para ello vamos a simular hasta 6 situaciones habituales:

Datos de partida	Objetivo
ID *B. germanica*	Datos de expresión
Nombre del gen *B. germanica*	Gráfico de expresión
ID del gen *B. germanica*	Secuencia / estructura genómica
ID Drosophila	Gráfico de expresión
2 genes de *B. germanica*	Comparativa de niéveles de expresión
Fragmento de un gen	Gráfico de expresión

Empezando por el caso más sencillo en el que conocemos el ID del gen en *B. germanica* y queremos conocer sus datos de expresión. La primera pestaña, Expresion table, permite la búsqueda de genes de *B. germanica* mediante su ID (formato Bger_XXXXX) o su nombre en caso de conocerse. Si el gen de interés está presente en la tabla de datos de expresión el programa devolverá una tabla con únicamente una fila con los datos de expresión del gen. Adicionalmente, se podrá manipular el número de dígitos a mostrar en la tabla.

Si partimos del nombre del gen de *B. germanica* (pero no la ID) y queremos obtener el grafico de barras del perfil de expresión. Necesitamos primero obtener la ID del gen de *Blattella* (a partir de la tabla de la pestaña Expression table) para posteriormente introducir (o seleccionar) el ID en la pestaña Expresion plot. De esta manera se genera automáticamente el grafico de barras del gen deseado.

En el caso de que queramos conocer la estructura genómica de un gen cuyo ID conocemos. Será suficiente acceder a la pestaña Expresion plot y una vez seleccionado el gen deseado se habilitará un botón con el que acceder al navegador apollo directamente a la posición de este gen donde se mostrará la estructura génica y en caso de requerir más detalles incluso se puede acceder a la secuencia completa del gen o a la de cada una de sus características.

Otra posible situación, sería el caso en el que conocemos un gen importante para el desarrollo de *D. melanogaster* y queremos comprobar si en *B. germanica* pudiera expresarse y tener un efecto similar. El primer paso serio conocer el

ortólogo del gen en *B. germanica,* para lo que utilizaremos el diccionario de genes ortolgoos de la pestaña Dmel orthologs. Una vez encontrado el ID de *B. germanica* procederemos utilizando la herramienta de la pestaña Expresion plot para obtener la gráfica correspondiente.

También puede pasarnos que queramos comparar los niveles de expresión de dos genes. Actualmente la herramienta de generación de gráficos de barras no permite la visualización de datos de expresión de dos genes al mismo tiempo por lo que habrá que utilizar la herramienta multi-lines que permite la selección (y/o introducción) de dos genes y la aplicación genera un alinea con los datos de expresión de ambos genes para que sean fácilmente comparable. Adicionalmente si queremos un enfoque más numérico, podemos acceder a la pestaña de Co-expression y una vez introducido uno de los dos genes de interés devuelve el coeficiente de correlación de Pearson (en caso de que sea superior a 0.5).

Como último ejemplo, podríamos encontrarnos en la situación de no conocer ni el ID ni el nombre de gen, únicamente conoceríamos un fragmento de la secuencia que lo compone. Para estas situaciones se ideo la herramienta de BLAST, mediante la introducción de la secuencia conocido se puede saber los genes más similares a este fragmento. En caso de que exista, tras lanzar la consulta a la aplicación, esta devolverá el listado con los genes más similares y en primera posición el gen buscado.

3. Limitaciones

Durante la creación de la aplicación se han encontrado algunos puntos que pueden limitar las capacidades de la aplicación. En este capítulo se comentan las principales limitaciones encontradas.

3.1 Análisis de datos

La aplicación funciona únicamente como herramienta de visualización de datos. Utiliza los datos almacenados en el servidor para generar las gráficas correspondientes según las peticiones del usuario. La aplicación no realiza ningún tipo de análisis de datos, los datos hay que dárselos previamente procesados y listos para mostrar. Esto se convierte en problema a la hora de cargar nuevos datos, requiere hacer un procesamiento manual de los datos cada vez.

3.2 Potencial de procesamiento

Para la visualización de los datos y la generación de gráficas, las capacidades de procesamiento del servidor son sencillas, ya que estas acciones no requieren grandes cálculos. No obstante, esta aplicación posee algunas matrices de datos de grandes dimensiones, las cuales necesitan cargar estos datos directamente en la memoria del servidor. Por lo general no trabajamos con matrices demasiado grandes (entre los 1 y los 10 Mb de tamaño), lo que permite gestionar tablas de hasta 60.000 filas con una decena de columnas. Pero hay una excepción, la matriz de co-expresión. Esta tabla se genera calculando el coeficiente de relación de Pearson para cada pareja de genes, si se conocen 28.471 genes de *B. germanica*, existen 28.471*28.471 = 810.597.841 parejas de genes. Una tabla de esas dimensiones es difícilmente manipulable por lo que se ha optado por el filtrado de todas las parejas con coeficientes bajos (concretamente por debajo de 0.5). El resultado es una matriz de 108.327.530 filas todavía demasiado grande para el servidor provisional. Por lo que hasta que se obtenga acceso un servidor de mayor potencia se utiliza una muestra parcial de esta matriz.

3.3 Tamaño de la aplicación

La construcción modular por pestañas permite flexibilidad para añadir más funcionalidades conforme se vayan creando y necesitando. Pero este proceso no puede utilizarse infinitamente, ya que un número demasiado elevado de pestañas podría saturar el espacio y confundir al usuario. En resumen, a pesar de que las pestañas es una buena idea para la construcción de la aplicación mediante funciona lides no es un sistema que se deba mantener en caso de un gran crecimiento en el número de funcionalidades de la aplicación.

4. Perspectivas de futuro

La duración de una asignatura del master es un tiempo bastante escueto como para crear un producto totalmente terminado. Se tuvo en cuenta en la planificación del proyecto y se nota en el resultado final. Para un resultado óptimo se requiere de una dedicación exclusiva a este proyecto y un periodo de tiempo más largo. En este capítulo se detalla lo que se podría hacer en caso de seguir con este proyecto durante más tiempo.

4.1 Nuevas funcionalidades

La aplicación nace de la necesidad de visualizar los datos transcriptomicos. Pero puede servir para un objetivo mucho más general, como es el facilitar el acceso a estos datos a cualquier persona que le interese y que la falta de conocimientos de informática le complica esta tarea. El primer punto que se puede mejorar de la aplicación es la adición de nuevas funcionalidades.

4.2 Abierto a toda comunidad científica

De momento está en un servidor provisional con acceso único mediante enlace. Ideado para cumplir las necesidades del laboratorio de Xavier Bellés y su equipo. No obstante, una vez la aplicación cumpla unos mínimos y pueda ser útil en un ámbito más general se publicará de manera global para el beneficio de toda a la comunidad científica y en especial a los más interesados que son los investigadores que trabajan con *B. germanica* estudiando la metamorfosis.

4.3 Ampliación a nuevas fuentes de datos

El público interesado en *B. germanica* y la metamorfosis es realmente pequeño. Actualmente la aplicación se limita a la visualización de los datos de este insecto, y únicamente los albergados por el laboratorio de la evolución de la metamorfosis de insectos. En un futuro se podría ampliar los datos albergados por el servidor para ofrecer un abanico de posibilidades mucho mayor.

5. Conclusiones

Ha sido el primer proyecto de este estilo que he realizado y la verdad es que me ha hecho crecer enormemente tanto a nivel de estudiante tanto a nivel de investigador.

El master de la UOC se imparte totalmente a distancia lo que exige al estudiante un grado de autonomía y gestión del tiempo de manera totalmente distinta a la realizada durante todo el resto de la etapa educativa. Y aunque lo distinto no siempre tiene que ser mejor, en este caso ha sido una experiencia muy positiva. El formato de trabajo, basado en pequeñas entregas que miden el progreso, me ha ayudado a familiarizarme con el sistema de planificación de trabajos y metas temporales de corto plazo. A pesar de que el tutor de la carrera ha estado siempre dispuesto a prestar apoyo, la distancia ha hecho que me enfrente al proyecto con un grado de autonomía que me ha brindado una oportunidad excelente para autoevaluar mis capacidades tanto de aprendizaje como de desarrollo de proyecto.

Ya tenía algo de experiencia con el manejo de datos con R por lo que la mayor carga de trabajo ha sido el diseño web y la creación del servidor web. Es muy diferente escribir el código para realizar un *pipeline* concreto que manipule unos datos, a programar un servidor web que tiene que actuar según ciertas entradas de datos introducidos por los usuarios, actuar en consecuencia para posteriormente enviar otros datos a la pantalla del usuario.

El trabajo de creación de la interfaz web ha sido muy costoso, a pesar de que conocía bien el lenguaje de programación R, nunca lo había utilizado con esta finalidad. Al tratarse de una página web, el código de R se tenía que complementar con otros lenguajes para sacarle el máximo provecho a la aplicación. El código escrito combina el lenguaje de R con trazas de HTML y CSS, dos lenguajes que apenas conocía y que para según que características he tenido que aprender a utilizar.

En un inicio, el objetivo a conseguir era ligeramente ambiguo y con multitud de cambios. Algunas funcionalidades se han modificado, y otras nuevas se han añadido conforme avanzada el proyecto. Pero el resultado conseguido es una

aplicación totalmente funcional que cumple con todos los criterios inicialmente establecidos.

La planificación al igual que el proyecto ha tenido algunas alteraciones en consecuencia de los cambios. Pero esto no ha impedido la correcta ejecución del calendario de actividades propuesto, con los cambios efectuados.

6. Glosario

RNA-seq: Es un método de secuenciación que utiliza técnicas de que permite la secuenciación del todo el conjunto de RNA de la muestra.

Servidor: Un ordenador en la nube capaz de ejecutar código.

Aplicación: Una serie de software informático diseñado específicamente para una función. En nuestro caso será la ejecución de la herramienta de visualización de datos transcriptomicos.

Set de datos: Un conjunto de datos agrupados.

Transcriptoma: Conjunto de todas las moléculas RNA de una célula.

7. Bibliografía

[1] R. D. Fleischmann *et al.*, “Whole-genome random sequencing and assembly of Haemophilus influenzae Rd.,” *Science*, vol. 269, no. 5223, pp. 496–512, Jul. 1995.

[2] J. C. Venter *et al.*, “The Sequence of the Human Genome,” *Science (80-.).*, vol. 291, no. 5507, pp. 1304–1351, Feb. 2001.

[3] F. Sanger and A. R. Coulson, “A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase,” *J. Mol. Biol.*, vol. 94, no. 3, pp. 441–448, May 1975.

[4] Y. Kodama, M. Shumway, and R. Leinonen, “The sequence read archive: explosive growth of sequencing data,” *Nucleic Acids Res.*, vol. 40, no. D1, pp. D54–D56, Jan. 2012.

[5] R. Lowe, N. Shirley, M. Bleackley, S. Dolan, and T. Shafee, “Transcriptomics technologies,” *PLOS Comput. Biol.*, vol. 13, no. 5, p. e1005457, May 2017.

[6] Z. Wang, M. Gerstein, and M. Snyder, “RNA-Seq: a revolutionary tool for transcriptomics.,” *Nat. Rev. Genet.*, vol. 10, no. 1, pp. 57–63, Jan. 2009.

[7] G. Ylla, M.-D. Piulachs, and X. Belles, “Comparative analysis of miRNA expression during the development of insects of different metamorphosis modes and germ-band types,” *BMC Genomics*, vol. 18, no. 1, p. 774, 2017.

[8] X. Belles and G. Ylla, “Towards understanding the molecular basis of cockroach tergal gland morphogenesis. A transcriptomic approach,” *Insect Biochem. Mol. Biol.*, vol. 63, pp. 104–112, 2015.

[9] R. Edgar, M. Domrachev, and A. E. Lash, “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.,” *Nucleic Acids Res.*, vol. 30, no. 1, pp. 207–10, Jan. 2002.

[10] G. Ylla, B. Fromm, M.-D. Piulachs, and X. Belles, “The microRNA toolkit of insects,” *Sci. Rep.*, vol. 6, no. 1, p. 37736, Dec. 2016.

[11] M. C. Harrison *et al.*, “Hemimetabolous genomes reveal molecular basis of termite eusociality,” *bioRxiv*, 2017.

[12] A. Conesa *et al.*, “A survey of best practices for RNA-seq data analysis,”

Genome Biol., vol. 17, no. 1, p. 13, 2016.

[13] E. Lee *et al.*, “Web Apollo: a web-based genomic annotation editing platform,” *Genome Biol.*, vol. 14, no. 8, p. R93, Aug. 2013.

[14] L. S. Gramates *et al.*, “FlyBase at 25: looking to the future,” *Nucleic Acids Res.*, vol. 45, no. D1, pp. D663–D671, Jan. 2017.

8. Anexos

4.4 Scripts completos para el funcionamiento del servidor:

ui.R

Script que crea la interfaz del usuario y envía las consultas al servidor.

```
V = "v0.7" # version de la app

##############  navbarPage("ShinyAPP TFM")  ##############

navbarPage(paste("ShinyAPP TFM",V), selected = "About the web",id =
"tabs",

          footer = HTML('<hr/><img
src="http://www.uoc.edu/portal/_resources/common/imatges/marca_UOC/UO
C_Masterbrand.jpg"/>
                        <img
src="http://professorsiciencia.cat/sites/default/files/styles/llistat
s-480/public/logo_ibe_sencer_color.png"
style="width:346px;height:104px" />'),

          tags$head( #Personaliza la ubicacion y forma de la
notificacion
            tags$style(
              HTML(".shiny-notification {
             position:fixed;
             top: 50px;
             left: 40%;
             color: #fff; background-color: #4c91cd
             }"
              )
            )
          ),

################# Table mRNA expression  ########################

          tabPanel("Expression Table", icon = icon("list"),value =
"EXT",

                   fluidRow(
                     column(2,
                            sliderInput("Round","Digits to
Round",0,4,2))
                   ),

                   fluidRow(
                     DT::dataTableOutput("table2")
                   )
           ),

#################### Plot mRNA expression  ####################

        navbarMenu("Expression plots",icon = icon("signal"),
```

```
            tabPanel("Expression plot",

                     fluidRow(
                       column(4,
                              selectizeInput(
                                inputId = 'searchID', label = 'Choose
a gene to show: ',
                                choices = NULL,
                                selected = 1)
                       ),
                       column(8,
                              htmlOutput("LINK",
                                         class="btn btn-default
action-button shiny-bound-input"
                                         ,
                                         style="margin-top: 25px")
                             )
                     ),

                     fluidRow(
                       plotOutput("plot0")
                     ),

                     fluidRow( # no funcional por el momento
                       column(12, align="center",
                              tableOutput("table3")
                       )
                     )
            ),

#################### Plot miRNA expression  ######################

            tabPanel("miRNA Expression plot",
                     fluidRow(
                              selectizeInput(
                                inputId = 'miRNA', label = 'Choose a
gene to show: ',
                                choices = NULL,
                                selected = 1)
                       ),
                     fluidRow(
                       plotOutput("plotmiRNA")
                     )
            ),

######################### Plot tissue expression
#########################
            tabPanel("Tissue expression plot",
                     fluidRow(
                       selectizeInput(
                         inputId = 'tissue', label = 'Choose a gene
to show: ',
                         choices = NULL,
                         selected = 1)
                     ),

                     fluidRow(
                       plotOutput("plottissue")
```

```
                    )
            )),


######################## Multiple  plot
########################################
            tabPanel("Multi-lines", icon = icon("random"),
                     fluidRow(
                       column(4,
                              selectizeInput(
                                inputId = 'genenumber2', label =
'Select genes:',
                                choices = NULL,multiple=TRUE,
                                selected = 1)
                              )
                     ),
                     fluidRow(
                       plotOutput("Multiplot")
                     )
            ),

################## Local Blast ######################
            tabPanel("Blast",icon = icon("search"),
                     fluidRow(
                       column(8,
                              textAreaInput("QuerySeq", "Paste your
query sequence",
                                            width = 600,height = 100,
                                            placeholder = "Protein
sequence / CDS sequence",
                                            resize = "vertical"
                                            )
                       ),
                       column(4,
                              fileInput("UploadSeq", "Upload the
file",
                                            width = 390,accept =
"text",placeholder = "Fasta format")
                              )
                       ),
                     fluidRow(
                       column(6,
                              actionButton("RunBlastp", "Run blastp !
Protein>Protein", width="100%",
                              style="color: #fff; background-color:
#337ab7")
                              ),
                       column(6,
                              actionButton("RunBlastx", "Run blastx !
Nucleotide>Protein", width="100%",
                              style="color: #fff; background-color:
#337ab7")
                       ),
                       tags$hr()
                     ),

                     fluidRow(
                       column(12,
```

```
verbatimTextOutput("Blast_out",placeholder = TRUE)
                          )
                        ),

                        fixedPanel(downloadButton(outputId='BotonFijo',
label= "Download Blast Output"
                                                  ,icon =
icon("download")
                                                 ),bottom = 2,right = 2
                                   )
                        ),

###################### Co-expression ############################
             tabPanel("Co-Expresion", icon = icon("tags"),

                        sidebarLayout(
                          sidebarPanel(
                            conditionalPanel(condition = 'input.mylist
== "Load data first !!"',

actionButton("loadCoE","Load data")
                            ),
                            conditionalPanel(condition = 'input.mylist
!= "Load data first !!"',

actionButton("loadNet","Load net")
                            ),

                            selectizeInput(
                              inputId = 'mylist', label = 'Select gene',
                              choices = "Load data first !!",
                              selected = 1),

                            radioButtons("PN", "type of relation",
                                         choices = c(Positive =
"Positive",
                                                     Negative =
"Negative",
                                                     All = "All"
                                                     ),
                                         selected = "All"),

                            sliderInput("Umbral","Co-ex threshold",
                                        min= 0.5, max = 1,step = 0.01,
                                        value = 0.5)
                           ,width = 3
                          ),

                          mainPanel(
                            tabsetPanel(type = "tabs",
                                        tabPanel("Table",
DT::dataTableOutput("CoEtable")),
                                        tabPanel("Network",
plotOutput("Net"))
                            )
                          )
                        )
             ),
```

```
#################### Orthologos ##############################

            tabPanel("Dmel orthologs", icon = icon("exchange"),

                     fluidRow(
                       HTML("Search for desired gene in the
searchbox"),
                       tags$hr()

                     ),

                     fluidRow(
                       column(12,DT::dataTableOutput("Ortho_Out"))
                     )
            ),

############################## About us
#####################################

            tabPanel("About the web", icon = icon("info-circle"),

                     fluidRow(
                       HTML('<div class= "row" style="margin-
left:10px;margin-right:10px">
                             <h3>UOC-UB Bioinformatics and
biostatistics master</h3>
                            <P></P> Master disertation proyect:
Transcriptomic data visualization web aplication
                            <P></P> The colabortion of the "Evolution
of Insect Metamorphosis Lab" (<a
href="https:http://www.biologiaevolutiva.org/xbelles/index.html">Webp
age</a>),
                             a research laboratory of the Institute of
Evolutionary Biology (<a href="https://www.ibe.upf-csic.es/
">IBE</a>), and Universitat Oberta
                             de Catalunya (<a
href="http://www.uoc.edu/portal/en/index.html">UOC</a>) make possible
this master degree project.
                            <P></P> The web applicaction is a usefull
tool to visualize, interactively, the data allocated in the
laboratory. This web are based, mostly, in the RNA-seq information of
the experimental model insect with hemimetabolan metamorphisis,<b><i>
Blattella germanica</b></i>.
                            <P></P> Student: Odei Barrenada, Tutor:
Guillem Ylla <hr> </div>')
                     ),

                     fluidRow(
                       HTML('<div class= "row" style="margin-
left:10px;margin-right:10px">

                        <h3>Origin of the data:</h3>

                          <ul>
                              <li>22 transcriptomes of Blattella
germanica (2 replicates for each 11 time peroid of development) will
be accesible from Gene expression Omnibus (GEO: PRJNA382128, 653
GSE99785).</li>
```

```
                            <li>Gene orthology between B. germanica
and D. melanogaster where obtained by using Blastp reciprocal best
hits (BRBHs) strategy.</li>
                            <li>Protein database for Blast+ was
obtained from the proteome of the B. Germanica.</li>
                            <li> B. Germianica anottation features
were used to get the coordenates of each gene.</li>
                          </ul>
                        <hr> </div>'
                            )
                      ),

                      fluidRow(
                        HTML('<div class= "row" style="margin-
left:10px;margin-right:10px">
                            <h3>Article citation:</h3>
                            <P></P>Clues about the evolution of
insect metamorphosis revealed by comparative transcriptomics of
hemimetabolan and holometabolan species.
                            <P>Guillem Ylla, Maria-Dolors Piulachs,
Xavier Belles </P>
                            <P>(Pending to be published)</P>

                            </div>')
                      )
            )

################## Cierre ######################

) #cierre de navbarPage
```

server.R

Script recibe las indicaciones del usuario y genera los gráficos y tablas para visualizarse en la pantalla.

```
#DATA LOAD

#setwd("../../srv/shiny-server/obt/") # directorio de la app en el
sevidor

load("data/normalized_RPKM_sum_S_v2")
load("data/normalized_RPKM_sum_v2")
load("data/RPKMs_manual_models_v2")
load("data/orthologs_Names")
gffS <- read.table("data/gffsimplified.txt",stringsAsFactors = F)
load("data/miRNA_table_genome_precs_sum")
load(file = "data/RPKMs_manual_models_v3_Tissues")

normalized_RPKM_sum <- normalized_RPKM_sum/2  # convertir en FPKM

# PACKAGE LOAD
require(igraph)
require(DT)
```

```
require(splines)

# SERVER.R

function(input, output,session){

######### exp table ################

  output$table2 <- DT::renderDataTable(
    datatable(normalized_RPKM_sum_S) %>%
      formatRound(~ NFE + ED0 + ED1 + ED2 + ED6 + ED13 + N1 + N3 + N5
+ N6 + Adult,digits = input$Round))

######### exp plot ################

  #la funcion updateSelectizeInput() permite cargar los menus
desplegables en del lado del servidor
  updateSelectizeInput(session = session, inputId = 'searchID',
choices = rownames(normalized_RPKM_sum), server = TRUE, selected =
"Bger_05466")

  #crea la paleta de colores que se utilizará
  colPal <- colorRampPalette(c("White", "brown"))

  output$plot0 <- renderPlot({
    if (input$searchID == ""){  #grafico en "en blanco"
      barplot(normalized_RPKM_sum["Bger_19809",]
              ,xlab = "Stage"
              ,ylab = "FPKMs",ylim=c(0,1)
              ,col = colPal(12)[-1])
      }else{  #creacion del grafico de barras con el gen indicado
      barplot(normalized_RPKM_sum[input$searchID,]
              ,main = input$searchID
              ,xlab = "Stage"
              ,ylab = "FPKMs"
              ,col = colPal(12)[-1])
      }
  })


######### miRNA exp plot ################

  updateSelectizeInput(session = session, inputId = 'miRNA', choices
= rownames(CPM_miRNA_sum), server = TRUE,selected = "Bger_05466")

  colPal <- colorRampPalette(c("White", "brown"))
  output$plotmiRNA <- renderPlot({
    if (input$miRNA == ""){
      barplot(normalized_RPKM_sum["Bger_19809",]
              ,xlab = "Stage"
              ,ylab = "CPMs",ylim=c(0,1)
              ,col = colPal(12)[-1])
    }else{
      barplot(as.numeric(CPM_miRNA_sum[input$miRNA,])
              ,main = input$miRNA
              ,names.arg = colnames(CPM_miRNA_sum)
              ,ylab = "CPMs"
              ,xlab = "Stage"
              ,col = colPal(12)[-1])
```

```
    }
  })

######### tissue exp plot ################

  updateSelectizeInput(session = session, inputId = 'tissue', choices
= rownames(normalized_RPKM_tissue), server = TRUE,selected =
"Bger_05466")

  colPal <- colorRampPalette(c("White", "brown"))

  output$plottissue <- renderPlot({
    if (input$tissue == ""){
      barplot(normalized_RPKM_sum["Bger_19809",]
              ,xlab = "Condition"
              ,names.arg = colnames(normalized_RPKM_tissue)
              ,ylab = "RPKMs",ylim=c(0,1)
              ,col = colPal(12)[-1])
    }else{
      barplot(as.numeric(normalized_RPKM_tissue[input$tissue,])
              ,main = input$tissue
              ,xlab = "Condition"
              ,names.arg = colnames(normalized_RPKM_tissue)
              ,ylab = "RPKMs"
              ,col = colPal(12)[-1])
    }
  })

################## link button exp-plot ######################

  output$LINK <- renderUI({
    if (input$searchID == ""){
      return("Go to Nowhere")
    }else{
      #generacion dinamica del link al navegador apollo (utiliza los
datos de gffS)
    A <- input$searchID
    tagList("", a(paste0("Go to ",A," in apollo genome browser"),

href=paste0("https://apollo.nal.usda.gov/blager/jbrowse/?loc="

,gffS$genes.seqname[which(rownames(gffS)==A)],"%3A"

,gffS$genes.start[which(rownames(gffS)==A)],".."

,gffS$genes.end[which(rownames(gffS)==A)],

"&tracks=DNA%2CAnnotations%2C1.%20BGER_v0.6.2-Models&highlight="),
                  #&tracks=DNA%2CAnnotations%2C1.%20BGER_v0.6.2-
Models&highlight=
                  target="_blank"))
    }
  })

########### multplines ##########

  selectList2 <- rownames(normalized_RPKM_sum)
```

```
  updateSelectizeInput(session = session, inputId = 'genenumber2',
choices = selectList2, server = TRUE,selected =
c("Bger_05864","Bger_05464"))

  output$Multiplot <- renderPlot({#primero genera el grafico en
blanco
    plot(0,col="white",ylab = "FPKMs",xlab = "Stage"
         ,ylim = if(length(input$genenumber2)==0){c(0,1)} else
c(0,max(normalized_RPKM_sum[input$genenumber2,]))
         ,xlim = c(1,11),xaxt='n')
    axis(side = 1, at = 1:11, labels = colnames(normalized_RPKM_sum),
tck=-.05)

    for (i in input$genenumber2){ #para cada ID selecionada crea los
puntos, las lineas y la leyenda
      points(normalized_RPKM_sum[i,], pch = 19,
             col = if (length(input$genenumber2) <=1){"red"} else
{palette(rainbow(length(input$genenumber2)))[which(input$genenumber2=
=i)]})

      smoth <-
predict(interpSpline(c(1:length(normalized_RPKM_sum[i,])),
normalized_RPKM_sum[i,]))
      lines(smoth,lwd=2,
            col=if (length(input$genenumber2) <=1){"red"} else
{palette(rainbow(length(input$genenumber2)))[which(input$genenumber2=
=i)]})

      legend("topright",legend=input$genenumber2
           ,col = if (length(input$genenumber2) <=1){"red"} else
{palette(rainbow(length(input$genenumber2)))}
           ,bg="white",lwd=2)
    }
  })

  ################## blastp ######################
    # runS <- "cmd.exe"  # Para las pruebas en un ordenador Windows
    runS <- "/bin/sh"

    observeEvent(input$RunBlastp, {

      textQ <- input$UploadSeq$datapath  #guarda la ubicacion del
archivo

      if (is.null(textQ)){
        # cuando le das al boton sin subir archivo:
        QSequence <- input$QuerySeq
        writeChar(QSequence,"results/TMP_queryseq.txt",eos = "\n")
        system2(paste(runS), input = paste0("blastp -query
","results/TMP_queryseq.txt -db data/db/Bger_prots.fa -out
results/Out2.txt"))
      }
      else {
        # pulsando el boton con archivo subido
        system(paste(runS), input = paste0("blastp -query ",textQ," -
db data/db/Bger_prots.fa -out results/Out2.txt"))
      }

      output$Blast_out <- renderText({
```

```
        BlastOUTFILE <- "results/Out2.txt"
        paste(  # selecina que partes del resultado mostrar y cuales
omitir
          readChar(BlastOUTFILE,15),"\n",
          substr(readChar(BlastOUTFILE,
file.info(BlastOUTFILE)$size),grepRaw("Database",readChar(BlastOUTFIL
E, file.info(BlastOUTFILE)$size)),100000)
          ,sep = ""
        )
        })

     })

################## blastx ######################

    observeEvent(input$RunBlastx, {

      textQ <- input$UploadSeq$datapath  #guarda la ubicacion del
archivo

      if (is.null(textQ)){
        # cuando le das al boton sin subir archivo:
        QSequence <- input$QuerySeq
        writeChar(QSequence,"results/TMP_queryseq.txt",eos = "\n")
        system2(paste(runS), input = paste0("blastx -query
","results/TMP_queryseq.txt -db data/db/Bger_prots.fa -out
results/Out2.txt"))
      } else {
        # pulsando el boton con archivo subido
        system(paste(runS), input = paste0("blastx -query ",textQ," -
db data/db/Bger_prots.fa -out results/Out2.txt"))
      }

      output$Blast_out <- renderText({
        BlastOUTFILE <- "results/Out2.txt"
        paste(   # selecina que partes del resultado mostrar y cuales
omitir
          readChar(BlastOUTFILE,15),"\n",
                   substr(readChar(BlastOUTFILE,
file.info(BlastOUTFILE)$size),grepRaw("Database",readChar(BlastOUTFIL
E, file.info(BlastOUTFILE)$size)),100000)
          ,sep = ""
        )
      })

    })

#### co-expr ###############################################

    MiniPairwiseC <- data.frame(var1=NA,var2=NA,value=NA)

    observeEvent(input$loadCoE, {

      load("data/MiniPairwiseC")

      updateSelectizeInput(session = session, inputId = 'CoEgenlist',
choices = MiniPairwiseC$Var1,selected ="Bger_04655" , server = TRUE)

      output$CoEtable <- DT::renderDataTable(
```

```
        switch (input$PN,
          Positive =
MiniPairwiseC[MiniPairwiseC$Var1==input$CoEgenlist &
MiniPairwiseC$value >= input$Umbral,],
          Negative =
MiniPairwiseC[MiniPairwiseC$Var1==input$CoEgenlist &
MiniPairwiseC$value <= -input$Umbral,],
          All = MiniPairwiseC[MiniPairwiseC$Var1==input$CoEgenlist &
abs(MiniPairwiseC$value) >= input$Umbral,]
        ),rownames=FALSE)

    })

     observeEvent(input$loadNet, {

       library(igraph)
       load("data/MiniPairwiseC")

       #preparacion del dataframe
       links <-
MiniPairwiseC[MiniPairwiseC$Var1==input$CoEgenlist,];links$relation
<- NULL
       links$Var2 <- as.character(links$Var2)
       for (i in 1:nrow(links)){
         links$relation[i] <- if (links$value[i]>0) {"pos"}else
{"neg"}
         links$value[i] <- abs(links$value[i])
       }
       #filtra segun umbral
       links <- links[links$value>= input$Umbral,]

       # define los colores de los links
       if(length(unique(links$relation)) == 1){
         links$relation <-factor(links$relation, labels = c("blue"))
       }else{
         links$relation <-factor(links$relation, labels = c("orange",
"blue"))
       }

       #creacion del objeto graficable
       nodes1 <- c(links$Var2,input$CoEgenlist)
       netCoE <- graph_from_data_frame(d=links,vertices = nodes1,
directed=F)

       output$Net <- renderPlot(

plot(netCoE,vertex.label.cex=0.9,edge.color=links$relation,edge.width
=3)
       )
    })


################## ortologos ######################

    output$Ortho_Out <- DT::renderDataTable({
      Orth[,-c(1)]
    },rownames= FALSE)
```

```
#################### boton fijo  ##########

    output$BotonFijo <- downloadHandler(
      filename = function() {
        paste0("BlastOutput_", Sys.Date(), ".txt")
      },
      content <- function(file) {
        file.copy("results/Out2.txt", file)

      },
      contentType = "text"
    )

########### notifications ########

     dur <- 12
     id <- "000"

     observe(
       switch(input$tabs, # realiza una notificaicon segun la pestaña
a la que se accede
              'About the web'={removeNotification(id)},
              EXT={removeNotification(id);id <<- showNotification("B.
germanica genes exrpression table! Search for genes in the box on the
right",duration = dur)},
              Blast={removeNotification(id);id <<-
showNotification("Local Blast against Blattella's proteome",duration
= dur)},
              'Multi-lines'={removeNotification(id);id <<-
showNotification("Comparison between gene expression patterns, select
genes on the left box",duration =dur)},
              'Co-Expresion'={removeNotification(id);id <<-
showNotification("Coexpresion matrix ",duration =dur)},
              'Multi-lines'={removeNotification(id);id <<-
showNotification("Coexpresion matrix ",duration =dur)},
              #'Dmel orthologs'={removeNotification(id);id <<-
showNotification("Orthologs ",duration =dur)},
              'Expression plots'={removeNotification(id);id <<-
showNotification("Beautifull expression barplots ",duration =dur)}
       )
     )

}  # cierre funcion server
```