MEMORIA

TFM: Plataforma web para la predicción de *targets* miRNA

Fernando Varela Martínez

11 de mayo de 2017

Plataforma web de predicción de targets miRNA

Fernando Baltasar Varela Martínez
Máster universitario en bioinformática y bioestadística UOC-UB

Responsable: Alexandre Sánchez Pla
Tutor: Albert Pla

MEMORIA

TFM: Plataforma web para la predicción de *targets* miRNA

Título del trabajo:	Plataforma web para la predicción de targets miRNA
Nombre del autor:	Fernando Baltasar Varela Martínez
Nombre del consultor/a:	Albert Pla
Nombre del PRA:	Alexandre Sánchez Pla
Fecha de entrega (mm/aaaa):	24/05/2017
Titulación:	Máster en bioinformática y bioestadística
Área del Trabajo Final:	Programación para la bioinformática
Idioma del trabajo	Castellano
Palabras clave	microRNA, predicción de *targets*, algoritmo

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones

Los microRNA (miRNA) son secuencias de 19 a 25 nucleótidos de RNA no codificante de animales y plantas que regulan la expresión génica postranscripcional, generalmente con efectos de inhibición al unirse a un RNA mensajero (mRNA) e impidiendo su traducción en los ribosomas. La predicción de targets para estos miRNA es muy importante por su relación con enfermedades graves como el cáncer y otras patologías además de regular procesos celulares relacionados con el sistema inmunitario o el estrés en plantas producido por condiciones extremas de falta de agua o altas temperaturas.
Con este proyecto se intenta dar a los investigadores una aplicación web para consultar predicciones de targets miRNA y a los desarrolladores de software bioinformático una interfaz de integración para la implementación de sus propios algoritmos a través de una interfaz de integración.
Además se mostrará información sobre miRNA y mRNA recogida mediante técnicas de *text mining*.
Se ha utilizado un proceso de desarrollo de software que permite su crear aplicaciones robustas, de fácil mantenimiento y preparadas para futuros desarrollos ya que dada la complejidad de los procesos implicados sería necesario continuar con el desarrollo en una segunda fase.

Abstract (in English, 250 words or less):

MicroRNA (miRNA) are 19-25 nucleotides non-coding RNA sequence in animals and plants that are key regulators of post-transcriptional genomic expression interacting with messenger RNA (mRNA). This interaction is very important to understand certain types of pathologies like cancer and other pathologies.
This project creates a web integrated platform that allows users making predictions about a miRNA and 3'UTR sequences and look up former results and TargetScan release info and also gives to bioinformatic programmers an R language interface to

code its own algorithms.
Also it shows information about miRNA and mRNA retrieved by text mining process.
The interaction between miRNA and mRNA is hardly understood and modeling that complexity requires a lot of effort from researchers and software developers.
A software development process has been used to create a robust, easy to maintain and prepared for future developments computer application.
The source code is available to anyone in GitHub.

Índice

1. Introducción

2.1. Contexto y justificación

En esta sección se explican las razones que justifican este proyecto, como se resuelve actualmente y su importancia dentro del mundo de la investigación. También se pone en antecedentes sobre el concepto fundamental sobre el que gira todo el proyecto: los microRNA (miRNA) y su interacción con los RNA mensajeros (mRNA).

Existen muchas herramientas en la web para la predicción de *targets* miRNA como *PicTar* [1] *TargetScan* [2] [3] *mirTar* [4] o *RNA22* [5]. Todas ellas implementan un determinado algoritmo (propio o de terceros a través de un API ("Application Programming Interface") pero no permite a los usuarios utilizar sus propios algoritmos. Además no siempre proporcionan el código fuente por lo que no se conoce exactamente qué procedimientos utilizan para hacer las predicciones.

La creación de una aplicación web de predicción de targets mirna para ejecutar algoritmos propios supone mucho trabajo para un investigador de forma que este proyecto intenta facilitar este proceso proporcionándole una plataforma web que le permita hacer ambas cosas. De esta forma se evita a los investigadores el tener que consultar varias herramientas de predicción distintas y les ahorra el tener que diseñar ellos mismos una aplicación web, lo que lleva tiempo y esfuerzo.

Contexto [6] [7] [8]

Los microRNA (miRNA) tienen una secuencia de 19 a 25 nucleótidos de RNA no codificante que regula la expresión génica postranscripcional, generalmente con efectos de inhibición, actuando en alguna de las fases de la traducción a proteínas al interaccionar con un RNA mensajero o *target*. Fueron descritos por primera vez en 1993 por *Lee* y colaboradores pero no se acuñó el término microRNA hasta el año 2001 en el que se publicaron tres artículos en la revista *Science* (26 de octubre de 2001) [9]

La figura 1 [10] representa el ciclo de formación de un miRNA y su interacción con un mRNA en animales y plantas (página siguiente):

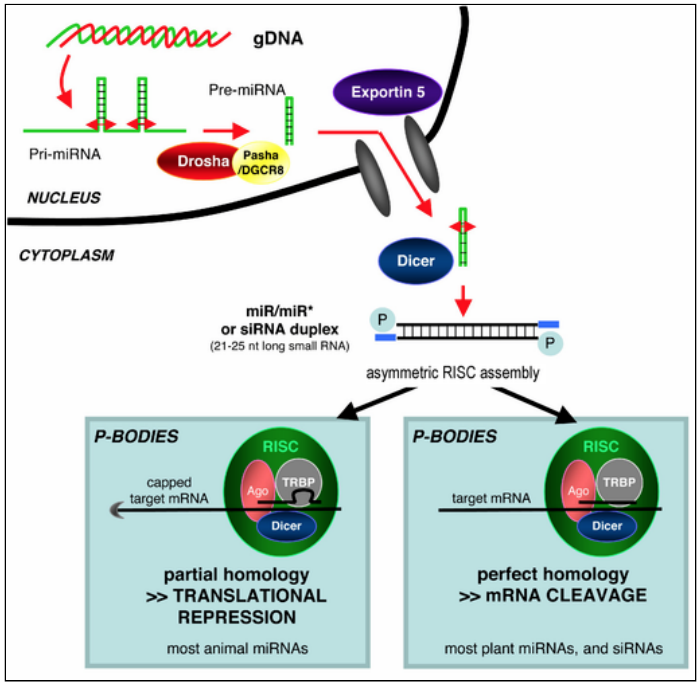


Fig. 1: formación de los miRNA

Un target de miRNA es un RNA mensajero, que es un tipo de RNA que contiene la secuencia que dará lugar a las proteínas en el proceso de traducción, que se inhibe cuando un miRNA se une a él.

Un miRNA puede tener varios *targets* (hasta cientos) y un *target* puede serlo de más de un miRNA.

La figura 2 [8] representa la interacción entre un miRNA y un mRNA:

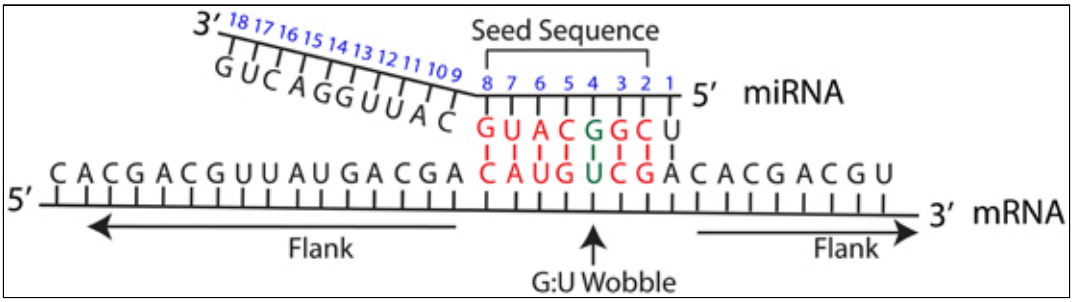


Fig. 2: interacción entre miRNA y mRNA

En ella se observa la unión de ambos RNA por una parte de la secuencia de un miRNA maduro, llamada semilla, produciéndose apareamiento de bases. Las secuencia de nucleótidos alrededor de la semilla se llama flanco.

Un apareamiento de bases de tipo *Watson-Crick* es el formado por las uniones G—C (guanina - citosina), A—T (adenina - timina, en DNA) o A—U (adenina - uracilo, en RNA) y son los más estables desde el punto de vista termodinámico. Se observa como en la semilla se pueden producir apareamientos menos

estables termodinámicamente, como el de la posición 4 del miRN de tipo G—U (guanina - uracilo).
También se ve cómo la interacción se produce por el extremo 3' UTR (menos frecuente por el extremo 5' UTR o CDS) del mRNA y el extremo 5' del miRNA (menos frecuente por el extremo 3')[11] **[11]** en lo que se conoce como *binding-site*. Los alineamientos y distintos tipos de apareamiento en el *binding-site* hace que cada interacción tenga unas características determinadas o *features* como se ve en la figura 3. **[8]**

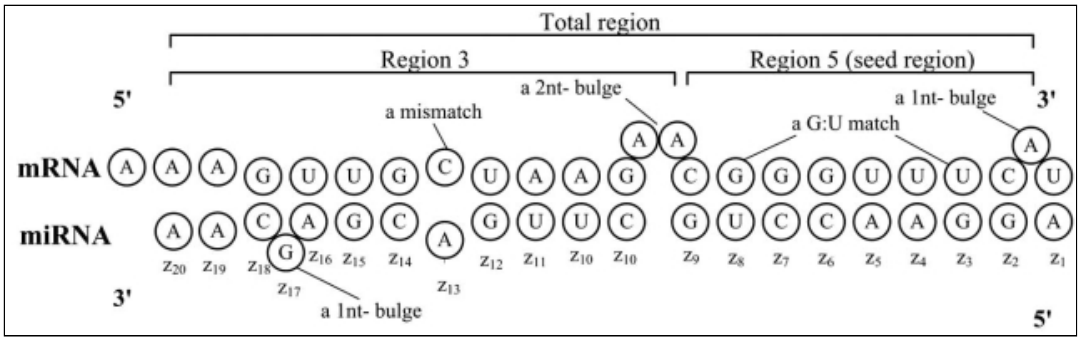


Fig. 3: *binding-site* y *features*

Los algoritmos de predicción utilizan los *features* como un elemento fundamental para sus cálculos ya que su extracción incide en la sensibilidad y la especificidad de la predicción.
Si se tiene en cuenta que $s = \{A, T, C, G\}$ donde s son los distintos nucleótidos y $z = \{z_1,...,z_K\}$ donde z es la composición de nucleótidos de la secuencia de miRNA desde el extremo 5' siendo el subíndice su posición dentro de la cadena, algunos de los *features* más importantes y que se han tenido en cuenta en este trabajo son[6]:

1. *Seed region match:* la semilla es una región de la secuencia de miRNA, también llamada región 5, que comprende desde la posición 1 a la posición 8 en el extremo 5'. El alineamiento de esta semilla con el mRNA es uno de los *features* más importantes. Bajo esta perspectiva se distinguen 8 tipos de secuencias semilla:
    1. Tipo 1: $x \in \{0,1\}$ and $x = 1$ si hay un apareamiento de bases perfecto en z_2 - z_8 de tipo *Watson-Crick*.
    2. Tipo 2: $x \in \{0,1\}$ and $x = 1$ si hay un apareamiento de bases perfecto en z_2 - z_8 de tipo *Watson-Crick* con una A (adenina) en el mRNA unida a la posición z_1.
    3. Type 3: $x \in \{0,1\}$ and $x = 1$ si hay un alineamiento en z_1 - z_7 de tipo *Watson-Crick*.
    4. Type 4: $x \in \{0,1\}$ and x = 1 si hay un apareamiento de bases perfecto en z_2 - z_8 de tipo *Watson-Crick* o apareamientos G-U y como mucho un alineamiento G-U.

5. Type 5: $x \in \{0,1\}$ and $x = 1$ si hay un apareamiento de bases perfecto en z_2 - z_7 de tipo *Watson-Crick* o apareamiento G-U (guanina-uracilo) y como mucho uno de ellos.
6. Type 6: $x \in \{0,1\}$ and $x = 1$ si el número de apareamientos perfectos en z_1 - z_8 de tipo *Watson-Crick* es mayor que un determinado valor.
7. Type 7: $x \in \{0,1\}$ and $x = 1$ si el número de apareamientos perfectos consecutivos en z_1 - z_8 de tipo *Watson-Crick* es mayor que un determinado valor.

2. *Conservation:* tanto los *targets* como los miRNA tienen secuencias muy conservadas entre especies. Cuando el mismo alineamiento de una semilla se produce en el extremo 3' UTR de una especie y además en un extremo ortólogo de otra especie, se considera que el alineamiento se caracteriza como "conservado".
3. *Free energy*: hace referencia a la cantidad mínima de energía libre y en general indica la estabilidad de un sistema biológico. En este caso indica la fuerza de la interacción entre el miRNA y su *target*. Normalmente es un valor real negativo y se mide en kcal/mol.
4. *In-site feature*: la semilla no es la única parte de la secuencia del extremo 3' UTR de la que se pueden extraer *features*. La zona de unión con un miRNA se puede dividir en la región 5 o semilla, región 3 (z_9 - z_{20}) y la unión de ambas (z_1 . z_{20}). En cada una de estas tres zonas se pueden extraer *features* como por ejemplo *free energy*, número de coincidencias en el alineamiento, número de no coincidencias, lo mismo para los apareamientos G-C, A-U, G-U, etc.
5. *Accessibility energy*: energía necesaria para hacer accesible el sitio de unión entre el miRNA y el mRNA. Cuanto menor sea, más accesible. Se mide en kcal/mol.

Los algoritmos de predicción de *targets* se pueden dividir en dos categorías principales:

1. **Basados en reglas:** utilizan un conjunto de reglas que tienen que cumplirse en un determinado orden, analizando la estructura del extremo 3' UTR.
   Ejemplo: *TargetScan*.
2. **Basados en datos**: toman decisiones en función de datos recopilados que se utilizan para la toma de decisiones.
   Ejemplos: *PicTar* y *mirTarget.*

En el proyecto se podrán implementar de los dos tipos de algoritmos.

La investigaciones sobre miRNA han experimentado un gran auge en los últimos años al revelar diversos estudios **[12]** su importancia en el desarrollo de algunos tipos de cáncer, como el colorrectal **[13]** y de pulmón **[14]** y otras

enfermedades graves [15] o procesos fisiológicos como la respuesta al estrés [16] o el desarrollo celular. [17]

En concreto, la predicción de *targets* es muy importante para conocer la regulación génica postranscripcional que se produce al unirse un miRNA con un mRNA produciendo generalmente una inhibición de la expresión aunque en algunos casos también puede aumentarla.

Por esta razón la existencia de herramientas de predicción de *targets* es muy importante para el avance de estas investigaciones.

La predicción de *targets* se hace actualmente a través de herramientas de terceros (generalmente a través de interfaces web) y mostrando los resultados obtenidos a partir de los datos introducidos por el usuario mediante de un formulario. Dependiendo de la aplicación hay distintas opciones, siendo imprescindible al menos un ID de miRNA o un ID de gen o extremo 3' UTR (ej: *mirDB*) [18] [19] [20]. En algunos casos se pueden introducir secuencias completas (ej: *PITA*) [21], alguna característica como p-valores o puntos de corte para algún cálculo estadístico (ej: *miRGate*) [22] o alguna características del alineamiento (ej: *miRTar*). En algunos casos las predicciones sólo se hacen para una especie animal en particular: *Homo sapiens*, *Zebrafish* (*Danio rerio*) o grupos filogenéticos como los vertebrados o los mamíferos.

También existen miRNA en plantas que están relacionados con procesos de crecimiento, desarrollo y responsables de regular el estrés (por calor, sequía, etc).

Con este trabajo se intenta obtener una plataforma web que se pueda instalar de forma sencilla, proporcionando el software necesario (incluido el código fuente) que permita tanto la predicción de targets como la implementación de algoritmos.

También se quiere recuperar, junto con los resultados del algoritmo, información sobre miRNA y genes en la literatura científica sin tener que recurrir a distintas herramientas de terceros por separado (ej: *PubMed* [nbci] y *miRBase* [23], etc) mediante conexiones API y de forma transparente para los usuarios de la aplicación. Esta recuperación se haría mediante técnicas de *text mining* o minería de datos.

2.2. Objetivos

En esta sección se plantean los objetivos del proyecto establecidos en su fase preliminar.

El objetivo principal es el diseño de una plataforma web que permita integrar herramientas de predicción de *targets* miRNA para facilitar a los investigadores hacer predicciones a partir de datos introducidos en la aplicación y a los desarrolladores de software el uso y la publicación de sus propios algoritmos.

Los objetivos específicos planteados son:

1. Identificar cuáles son los elementos necesarios para permitir a los investigadores/usuarios la selección de la información útil para hacer sus consultas de predicción de *targets*.
   Tiene como resultado la interfaz web de la aplicación.
2. Minimizar el número de acciones/pasos a ejecutar durante la predicción de nuevos *targets* miRNA.
   Tiene como resultado la conexión mediante API con herramientas de predicción online de terceros.
3. Reducir los tiempos de respuesta de las consultas almacenadas con respecto a las mismas consultas hechas por primera vez.
   Tiene como resultado la creación de una base de datos.
4. Identificar los métodos de la interfaz para integrar herramientas de otras desarrolladores.
   Tiene como resultado crear una interfaz de integración para desarrolladores de software.
5. Identificar la información relevante mediante procesos de *text mining*.
   Tiene como resultado recabar información sobre miRNA y *targets* de la literatura científica.
6. Minimizar el número de conexiones a terceras aplicaciones en cada consulta.
   Tiene como resultado la revisión y refactorización del código.

2.3. Enfoque y método seguido

En esta sección se discuten las posibles estrategias para lograr la consecución del proyecto.

La estrategia seguida ha sido el desarrollo de un producto nuevo en forma de plataforma web al no existir ninguna herramienta que permita tanto la predicción de *targets* de miRNA como el uso y publicación de cualquier algoritmo propios creados por los desarrolladores.

En caso de tener acceso al código fuente de alguna herramienta web, se podría haber utilizado y modificado para crear algunas partes de la plataforma, especialmente los formularios de introducción de datos.

También se podría haber reutilizado algún esquema de base de datos haciendo los cambios pertinentes pero tampoco se ha encontrado ninguno disponible.

Se ha escogido un modelo de desarrollo de software que permite un desarrollo rápido de la aplicación, corrigiendo errores antes de pasar a una nueva fase y creando un código simple y sencillo de mantener que facilite las modificaciones que se puedan hacer en fases posteriores de desarrollo.

En general, el enfoque utilizado se ha basado en dos factores fundamentales: la existencia de un solo desarrollador y el poco tiempo disponible para completarlo.

2.4. Planificación

En esta sección se describe la planificación inicial del proyecto y su grado de cumplimiento, tanto de los objetivos como de las tareas e hitos.

En la figura 4 se muestra el cronograma en el que ha basado la planificación:

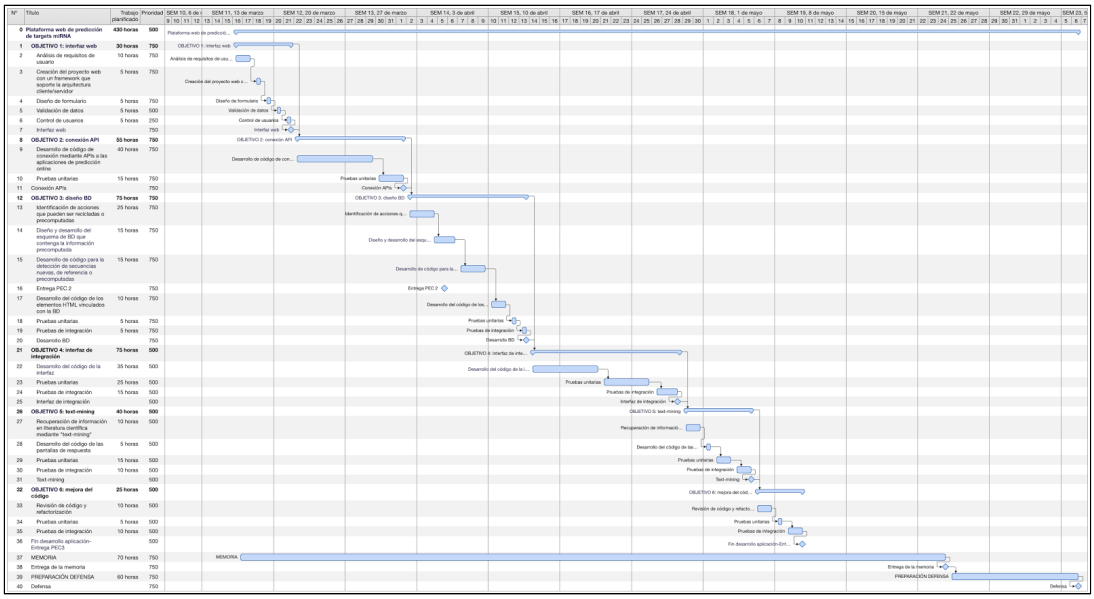


Fig. 4: cronograma completo del proyecto

A continuación se detallan las tareas realizadas agrupadas por objetivo:

- **Objetivo 1: Interfaz web**

  Identificar cuáles son los elementos necesarios para permitir a los investigadores/desarrolladores la selección de los datos necesaria para hacer las consultas (tabla 1)

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 1	Interfaz web	16/03/17	21/03/17	Alta	6	71%
Tarea 1	Análisis de requisitos de usuario / Herramientas de predicción online	16/03/17	17/03/17	Alta	2	100%
Tarea 2	Creación del proyecto	18/03/17	18/03/17	Alta	1	100%
Tarea 3	Diseño del formulario	19/03/17	19/03/17	Alta	1	100%
Tarea 4	Validación de	20/03/17	20/03/17	Alta	1	100%

	datos					
Tarea 5	Control de usuarios	21/03/17	21/03/17	Baja	1	0%
HITO 1	**Interfaz web**	**22/03/17**	**22/03/17**	**Alta**		

Tabla 1: planificación de objetivo interfaz web

Tareas

1. **Análisis de requisitos de usuario y estudio de la representación de resultados en las herramientas de predicción online**
   Se han evaluado las necesidades de los usuarios (ver sección 2.1: "Análisis de requisitos de usuario").
   Las herramientas de predicción online estudiadas han sido *Pictar* y *TargetScan* (ver sección 3.1: "Interfaz gráfica de usuario").
2. **Creación del proyecto web con un *framework* que soporte la arquitectura cliente/servidor**
   Se ha creado un proyecto web (ver sección 5.1: "Estructura de archivos").
3. **Diseño y desarrollo de la interfaz gráfica**
   Se ha creado una interfaz adaptada a las necesidades de los usuarios(ver sección 3.1: "Interfaz gráfica de usuario").
4. **Validación de datos**
   Se han codificado validaciones para los ID e miRNA y gen. También se valida que el formato de los archivos fasta sea correcto (ver sección 2.1: "Análisis de requisitos de usuario").
5. **Control de usuarios**
   Se consigue que cada usuario solo pueda ver la información determinada por su nivel de acceso.

- **Objetivo 2: Conexión API**

  Minimizar el número de acciones/pasos a ejecutar durante la predicción de nuevos *targets* miRNA (tabla 2)

  Terminología

  - **Pruebas unitarias:** sirven para comprobar el funcionamiento correcto de una unidad de código determinada.

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 2	Conexión API	22/03/17	01/04/17	Alta	11	86%
Tarea 1	Desarrollo de código de conexión mediante APIs	22/03/17	29/03/17	Alta	8	100%
Tarea 2	Pruebas unitarias	30/03/17	01/04/17	Alta	3	50%

HITO 2	**Conexión API**	**01/04/17**	**01/04/17**	**Alta**		

Tabla 2: planificación de objetivo conexión API

Tareas

1. **Desarrollo de código de conexión mediante APIs a las aplicaciones web**
   Se conecta con dos aplicaciones web, *mirBase* y *Ensembl* **[24]** a través de sus API correspondientes.
2. **Pruebas unitarias**
   Se hacen pruebas del código relacionado con las conexiones y se elabora un documento de pruebas con su resultado.

- **Objetivo 3: Base de datos**
  Reducir los tiempos de respuesta de las consultas almacenadas con respecto a las mismas consultas hechas por primera vez (tabla 3)

Terminología

- **Pruebas de integración:** sirven para comprobar el funcionamiento correcto de un conjunto de unidades de código relacionadas entre sí.

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 3	Diseño de BBDD	02/04/17	14/04/17	Alta	13	78%
Tarea 1	Esquema de BBDD	02/04/17	04/04/17	Baja	3	100%
Tarea 2	Desarrollo del código de los elementos HTML vinculados con la BBDD	05/04/17	07/04/17	Alta	3	100%
Tarea 3	Desarrollo de código para la detección de secuencias nuevas, de referencia o precomputadas	08/04/17	10/04/17	Alta	3	100%
HITO 3	**Entrega PEC1**	**14/04/17**	**14/04/17**	**Alta**		
Tarea 4	Desarrollo del código de los elementos HTML	11/04/17	12/04/17	Alta	2	100%
Tarea 5	Pruebas unitarias	13/04/17	13/04/17	Alta	1	50%

Tarea 6	Pruebas de integración	14/04/17	14/04/17	Alta	1	50%

Tabla 3: planificación de objetivo base de datos

Tareas

1. **Identificación de acciones que pueden ser recicladas o precomputadas**
   Se guarda información en la fase de identificación de *binding-sites* potenciales, obtención de *features* y evaluación de *binding-sites* potenciales (ver sección 4.3: Interfaz de integración).
   Diseño y desarrollo del esquema de BBDD que contendrá la información del punto anterior
   Se crea un esquema de base datos con tablas, secuencias, índices y tipos de dato compuestos
2. **Desarrollo de código para la detección de secuencias nuevas, de referencia o precomputadas**
   Se crea código para detectar si una secuencia es nueva o de referencia y para saber si han sido precomputados.
3. **Desarrollo del código de los elementos HTML vinculados con la BBDD**
   Se crea una lista desplegable con datasets almacenados en la base de datos.
4. **Pruebas unitarias** (tarea no completada)
   Se hacen pruebas del código relacionado con la base de datos y se elabora un documento de pruebas con su resultado.
5. **Pruebas de integración** (tarea no completada)
   Se hacen pruebas del código de los puntos anteriores y se elabora un documento de pruebas con su resultado.

- **Objetivo 4: Interfaz de integración para desarrolladores**
  Identificar los métodos de la interfaz para integrar herramientas de otras desarrolladores (tabla 4)

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 4	Interfaz de integración	15/04/17	29/04/17	Alta	13	78%
Tarea 1	Código	15/04/17	21/04/17	Baja	3	100%
Tarea 2	Pruebas unitarias	22/04/17	26/04/17	Alta	1	50%
Tarea 3	Pruebas de integración	27/04/17	29/04/17	Alta	1	50%

HITO 4	**Interfaz**	**29/04/17**	**29/04/17**	**Alta**		

Tabla 4: planificación de objetivo interfaz de integración

Tareas

1. **Desarrollo del código de la interfaz**
   La interfaz de integración que contendrá el algoritmo se ha dividido en cuatro partes. En cada una de ellas se comprueba si los parámetros de entrada que han sido calculados anteriormente para no tener que hacerlo de nuevo y reducir el tiempo de proceso (ver sección 4.3: Interfaz de integración).
2. **Pruebas unitarias** (tarea no completada)
   Se hacen pruebas del código relacionado con la interfaz mediante un ejemplo y se elabora un documento de pruebas con su resultado.
3. **Pruebas de integración** (tarea no completada)
   Se hacen pruebas del código de los puntos anteriores y se elabora un documento de pruebas con su resultado.

- **Objetivo 5: *Text mining***
  Identificar la información relevante sobre miRNA y *targets* en la literatura científica mediante procesos de *text mining* (Tabla 5)

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 5	*Text-mining*	30/04/17	06/05/17	Baja	7	2%
Tarea 1	Recuperación de información	30/04/17	01/05/17	Baja	2	0%
Tarea 2	Desarrollo del código de las pantallas de respuesta	02/05/17	02/05/17	Baja	1	20%
Tarea 3	Pruebas unitarias	03/05/17	04/05/17	Baja	2	0%
Tarea 4	Pruebas de integración	05/05/17	06/05/17	Baja	2	0%
HITO 5	***Text-mining***	**06/05/17**	**06/05/17**	**Baja**		

Tabla 5: planificación de objetivo *text mining*

Tareas

1. **Recuperación de información en literatura científica mediante *text mining*** (tarea no completada)
   Se desarrolla código para seleccionar la información y mostrarla al usuario. Se crean las tablas y campos necesarios en la base de datos.

2. **Desarrollo del código de las pantallas de respuesta** (tarea no completada)
   Se desarrolla código para mostrar la información al usuario.
3. **Pruebas unitarias** (tarea no completada)
   Se hacen pruebas del código relacionado con la interfaz mediante un ejemplo de prueba y se elabora un documento de pruebas con su resultado.
4. **Pruebas de integración** (tarea no completada)
   Se hacen pruebas del código de los puntos anteriores y se elabora un documento de pruebas con su resultado.

- **Objetivo 6: Revisión y refactorización de código**
  Minimizar el número de conexiones a terceras aplicaciones en cada consulta (Tabla 6)

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Objetivo 6	Revisión y refactorización de código	07/05/17	11/05/17	Alta	5	50%
Tarea 1	Revisión y refactorización de código	07/04/17	08/05/17	Alta	2	50%
Tarea 2	Pruebas unitarias	09/05/17	09/05/17	Alta	1	50%
Tarea 3	Pruebas de integración	10/05/17	11/05/17	Alta	2	50%
HITO 6	**Fin de desarrollo**	**11/05/17**	**11/05/17**	**Alta**		

Tabla 6: planificación de objetivo revisión de código

Tareas

1. **Revisión y refactorización de código**
   Se modifican algunas partes del código para que sea más legible, reutilizable y eficiente.
2. **Pruebas unitarias**
   Se hacen pruebas del código relacionado con la revisión de código y se elabora un documento de pruebas con su resultado.
3. **Pruebas de integración**
   Se hacen pruebas del código de los puntos anteriores y se elabora un documento de pruebas con su resultado.

- **Memoria**

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Tarea 1	Escritura	11/05/17	24/05/17	Alta	70	100%
HITO 7	**Memoria**	**24/05/17**	**24/05/17**	**Alta**		

Tabla 6: planificación de la memoria

- **Objetivo 8: Preparación de la defensa**

Descripción	Nombre	Inicio	Fin	Prioridad	Duración (jornadas)	Progreso
Tarea	Preparación	11/05/17	24/05/17	Alta	70	100%
HITO 8	**Presentación**	**25/05/17**	**06/06/17***	**Alta**	**60**	**100%**

Tabla 7: planificación de la defensa

*Ampliado al 06/06/17

2.5. Breve sumario de productos obtenidos

El producto obtenido es una plataforma web para hacer consultas de predicción de *targets* miRNA en la que también se pueden integrar algoritmos de predicción propios. La plataforma está formada por un conjunto de archivos estructurados en forma de aplicación web más dos archivos para la creación e inserción de datos en un gestor de base de datos de tipo relacional.

- **Archivos y directorios de aplicación**
  - Directorio mirnaPlatformPredict
    - ui.R
    - server.R
    - mirnaPlatformPredict.Rproj
    - api_script.R
    - bd_script.R
    - busquedatargets_script.R
    - interfaz_script.R
    - librerias_script.R
    - prediccion targets_script.R
    - validaciones_script.R
    - global.R
    - Directorio www
      - result_utr_API.fasta
      - Predicted_Targets_Info.default_predictions.txt.zip
      - ejemplo[...].fasta

- Directorio log
  - mirnaplatform.log

●**Archivos y directorios de base de datos**
- sql_create_db.sql
- sql_insert_aux_tables.sql

Para una descripción más detallada ver sección ver sección 5.1: "Estructura de archivos".

El proyecto está acompañado de una memoria y de una presentación en vídeo sobre la base de diapositivas.

2.6. Breve descripción de los otros capítulos de la memoria

- **Análisis**
  - **Análisis de requisitos de usuario**
    Describe forma de usar la aplicación en base a unas necesidades previas de los usuarios.
- **Diseño**
  - **Interfaz de usuario**
    Descripción de los elementos de cada pantalla de la aplicación.
  - **Base de datos**
    Descripción de las tablas, campos, secuencias, índices, tipos compuestos y demás elementos que forman el esquema.
- **Implementación**
  - **Tecnologías empleadas**
    Descripción del software utilizado en el proyecto.
  - **Conexiones API**
    Descripción de las conexiones a aplicaciones de terceros a través de sus API.
  - **Interfaz de integración**
    Descripción de la interfaz integrada para que los usuarios desarrolladores implementen sus propios algoritmos de predicción.
- **Resultado**
  - **Estructura de archivos**
    Conjunto de archivos y directorios que componen el proyecto, memoria y de una presentación en vídeo.

2. Análisis

2.1. Análisis de requisitos de usuario

Es un documento en el que se recogen las necesidades de los usuarios de una aplicación, proporcionadas por ellos mismos o por el propio desarrollador en base a su experiencia y conocimiento. En este caso en el que no ha habido usuarios a los que hacer entrevistas, encuestas, etc han sido de gran ayuda los comentarios del tutor del proyecto. Están limitadas por el tipo de software y el hardware que se va a utilizar, el coste de implementarlas y el tiempo disponible.

En general se espera que la plataforma que proporcione flexibilidad a la hora de hacer consultas de predicciones a partir de distintos identificadores y secuencias de miRNA, genes y extremo 3' UTR. También que muestre mensajes en pantalla cuando se produzca alguna incidencia o error.

En el caso de los usuarios desarrolladores, se espera que el código esté bien documentado y sea fácil la implementación de sus algoritmos y hacer modificaciones en caso de desarrollos futuros.

Terminología

-API: permite la comunicación entre dos aplicaciones distintas de forma sencilla para el programador.

-**R**: lenguaje de programación multiplataforma muy utilizado en el ámbito de la bioinformática. **[25]**

-Paquetes de R: conjunto de funciones, datos y código R que se almacenan en una carpeta conforme a una estructura definida llamada librería y de acceso fácil para R.

-Ensembl: es un navegador genómico de vertebrados que permite la consulta de genes, anotaciones, alineamientos, etc. Dispone de varias herramientas para los investigadores: BLAST/BLAT **[26]** (alineamientos locales de secuencias de nucleótidos y proteínas), VEP **[27]** (efecto de SNPs, inserciones, deleciones y otras variantes del genoma en secuencias de genes, transcriptos y proteínas) y *BioMart*.

-BioMart: herramienta de Ensembl que permite la extracción de información genómica de forma sencilla, vía web o a través de otras aplicaciones mediante un API y es la que se utiliza en esta plataforma.

Las consultas se hacen a uno o varios datasets que agrupan la información por especie.

Se ha utilizado esta herramienta y no otra al ser el único que tiene APIs compatibles con R, aparte de ser una referencia en el sector

***-biomaRt*:** paquete de R que provee un API para conectar a *BioMart* desde otra aplicación diferente.

-ID: identificador una secuencia. Pueden tener distinta nomenclatura dependiendo de la base de datos online que se consulte.

-Secuencia de referencia: es aquella que ya ha sido catalogada e incorporada a alguna base de datos reconocida dentro del mundo de la investigación.

-Release: conjunto de datos genómicos e ima base de datos científica relacionados y actualizados a medida que se avanza en su conocimiento.

-Formato fasta: utilizado en bioinformática para representar secuencias de nucleótidos y proteínas. Admite identificadores y comentarios, que se sitúan antes de la secuencia. En un archivo fasta (basado en texto) se puede incluir una o varias secuencias.

1. Tipos de usuario

La plataforma está pensada para dos tipos de usuario: por un lado, investigadores que quieren consultar predicciones de *targets* miRNA a partir de secuencias de nucleótidos de miRNA y del extremo 3' UTR y por otro, desarrolladores de software que quieran integrar sus propios algoritmos de predicción.

1.1. Usuario investigador

A través de una aplicación web el usuario introducirá información sobre miRNA y extremo 3'UTR para determinar si alguno de éstos últimos es target de alguno de los primeros

1. Búsqueda de predicciones existentes en la BBDD

El usuario podrá ver datos descargados de herramientas de predicción online (*release*) y de predicciones hechas anteriormente desde la plataforma.

Datos de entrada

Datos de salida

- Información sobre *targets* de miRNA

Se mostrará al usuario un listado con información de *targets*. Se podrá buscar, filtrar y ordenar por cualquiera de los campos.

2. Nuevas predicciones

Se podrán hacer nuevas predicciones utilizando un algoritmo creado por un usuario desarrollador.

Datos de entrada

- Identificador del algoritmo

Selección del algoritmo de predicción de *targets* que se va a ejecutar.

Valor por defecto: texto "Algoritmo de usuario 1".

Validación: no.

- Identificador de dataset de *Ensembl*

Selección del dataset de *Ensembl* utilizado como fuente de datos de extremo 3' UTR. Se podrá seleccionar de uno a todos.

Valor por defecto: "hsapiens_gene_ensembl".

Validación: se debe seleccionar al menos uno.

- Identificador de secuencias

Entrada de ID de miRNA y extremo 3' UTR en forma de cadena de caracteres alfanuméricos.

No se validará los ID de miRNA en la aplicación de forma que se aceptarán ID de secuencias maduras e inmaduras (si el ID es incorrecto las herramientas online no devolverán resultados).

Se admitirán ID de genes de bases de datos *Ensembl*, *NCBI gene* y *UniGene*.

- *Ensembl ID*: se comprueba que las primeras 3 letras sean "ENSG" y que la longitud sea de 15 caracteres (ej: *ENSG00000121410 - alpha-1-B glycoprotein*).
- *NCBI gene ID* **[28]** ("Entrez gene"): se comprueba que sea un número entero (ej: *27 - ABL proto-oncogene 2, non-receptor tyrosine kinase*).
- *UniGene ID* **[29]:** se comprueba que comience por 2 letras y un punto "." y a continuación números enteros (ej: *Hs.100057 - Serine/threonine kinase 35*).

En caso de haber más de un identificador estarán separados por comas.

Para que sea fácil modificar el tipo de ID permitidos, estas opciones están guardadas en la variable "genes_id_aceptados" en "global.R". Los cambios en esta variable se verán reflejados automáticamente en el campo de opciones "Base de datos referencia" de la pantalla de nuevas predicciones.

Las validaciones para ID de gene están codificadas en la función:

Archivo: `validaciones_script.R`

Función:
```
ValidarTextGenID(gen_id, tipo_ref=
    "ensembl_gene_id")
```

Los ID consultados tienen que ser todos de la misma base de datos en cada ejecución.

Se podrá consultar además por la secuencia del miRNA o la del extremo 3' UTR. No se ha implementado al considerar que si el usuario quiere hacer consultas por secuencia tiene la opción de utilizar un archivo *fasta*.

Valor por defecto: vacío.

Validación: No más de 500 ID (límite de consultas con *BioMart*).

En el caso de admitir secuencias se validará según su número (una o varias) y su tipo (RNA o DNA). Ver punto anterior.

En la tabla 7 se muestran los tipos y procedencias de las secuencias:

Entrada	Tipo	Procedencia
miRNA	Referencia	BBDD de nucleótidos online
	Nueva	Usuario
Extremo 3' UTR	Referencia	BBDD de nucleótidos online
	Nueva	Usuario

Tabla 7: tipos y procedencia de secuencias

3. Identificador de base de datos de referencia

Selección de base de datos de secuencias de referencia. Hay tres opciones: "Ensembl ID", "NCBI gene ID" y "UniGene ID:"

Valor por defecto: opción "Ensembl ID"
Validación: no.

4. Archivo fasta de secuencias

El usuario podrá seleccionar un archivo de texto (extensión ".txt", ".fa" y ".fasta") con formato *fasta* con una o varias secuencia de miRNA y un archivo con una o varias secuencias de extremo 3' UTR que se utilizarán para consultar las predicciones de *targets*.

Valor por defecto: ningún archivo seleccionado.
Validación: se comprobará su formato antes de ser enviados. Dependiendo del tipo de secuencia (RNA o DNA) se hará distintos tipos de validaciones.

En la tabla 8 se hace una relación de las validaciones:

	Entrada del usuario	Validación
1	miRNA ID	
	Gen ID	Validar
2	miRNA ID	
	Extremo 3' UTR fasta	Validar formato y tipo de secuencia (DNA)
3	miRNA fasta	Validar formato y tipo de secuencia (RNA)
	Gen ID	Validar nomenclatura
4	miRNA fasta	Validar formato y tipo de secuencia (RNA)
	Extremo 3' UTR fasta	Validar tipo de secuencia (DNA)

Tabla 8: validaciones

Datos de salida

Los datos de respuesta del algoritmo se mostrarán en un visor en la misma ventana.

1.2. Usuario desarrollador

Integración de algoritmos en la plataforma

Datos de entrada

- **Algoritmos**

El usuario podrá ejecutar su propio algoritmo de predicción de *targets* miRNA. Para ello se le proporcionará un conjunto archivos estructurados en forma de proyecto web y dentro de ellos una interfaz de integración para implementarlos.

Una vez instalada correctamente, el usuario podrá ejecutar sus algoritmos de predicción a través de una aplicación web como se indica en el punto 1.1.

Valor por defecto: no.

Validación: para implementar el algoritmo deberá cumplir los requisitos impuestos por la interfaz de integración suministrada.

2. Diagrama de flujo de datos de la aplicación

La figura 5 representa el flujo de datos desde que el usuario los introduce a través de un formulario hasta la muestra de los resultados del algoritmo de predicción.

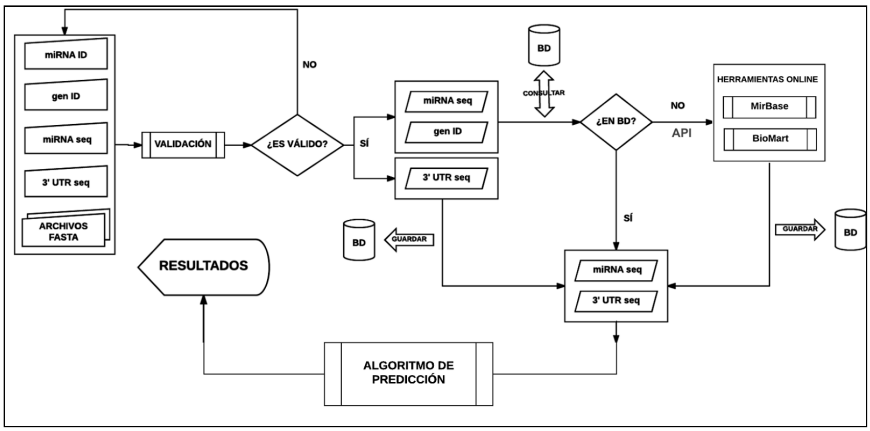


Fig. 5: diagrama de flujo de datos de Nuevas predicciones

Pasos:

1. El usuario introduce datos sobre miRNA y extremo 3' UTR.
2. Se validan los datos.
3. Se comprueba si están precomputados.
4. Si no lo están, se recuperan a través de API .
5. Se guardan los datos.
6. Se ejecuta el algoritmo.
7. Se muestran los datos al usuario.

3. Diseño

En este capítulo se hace una descripción de la fase diseño de la interfaz de usuario y la base de datos. En el caso de la interfaz se comentan todas las pantallas y las posibilidades de navegación del usuario mientras que en el caso de la base de datos se describe cómo se ha modelado el problema e incluye el diagrama de entidad relación y el esquema.

3.1. Interfaz gráfica de usuario

Se ha creado una interfaz gráfica sencilla, fácil de usar e interpretar y con coherencia en la disposición de los elementos y los colores en las distintas pantallas.

Tiene una barra de navegación en la parte superior siempre visible, sin pestañas ni listas desplegables para seleccionar las opciones. Esta sencillez aumenta la accesibilidad de la aplicación al no haber animaciones ni otro tipo de efectos.

Está compuesta por tres pantallas más una barra de navegación en la parte superior de todas ellas. Los mensajes aparecen en color rojo para resaltar y los resultados se muestran debajo de los formularios en forma de tabla de datos.

Todos los campos tienen una leyenda asociada que indica su función.

Otra opción habría sido diseñar una interfaz de línea comandos que aumentaría la rapidez de uso para usuarios avanzados pero que sería muy poco atractiva y difícil de manejar para usuarios básicos. Además su programación sería más complicada.

1. Pantallas

1.1. Barra de navegación

Contiene un menú que permite navegar por las distintas opciones de la aplicación (fig. 6). Aparece en la parte superior de todas las pantallas.

Fig. 6: Barra de navegación

1.2. Pantalla de Inicio

Pantalla informativa con un mensaje describiendo las funciones de la plataforma y otros detalles cuando se accede a la aplicación (fig. 7).

Acceso: barra de navegación -> Inicio

Fig. 7: Pantalla de inicio

Se han añadido enlaces a información sobre *targets* de miRNA y al código fuente al repositorio de código *GitHub* desde donde se podrá descargar, utilizar y modificar.

1.3. Pantalla de Búsqueda

Pantalla para consultar información sobre resultados de predicciones almacenadas en la base de datos (fig. 8).

Esta pantalla no se planificó en un principio pero después se entendió que podía ser de utilidad para los usuarios investigadores tener la posibilidad de consultar información de bases de datos online de miRNA y la relación de resultados computados por la aplicación.

Acceso: barra de navegación -> Búsqueda

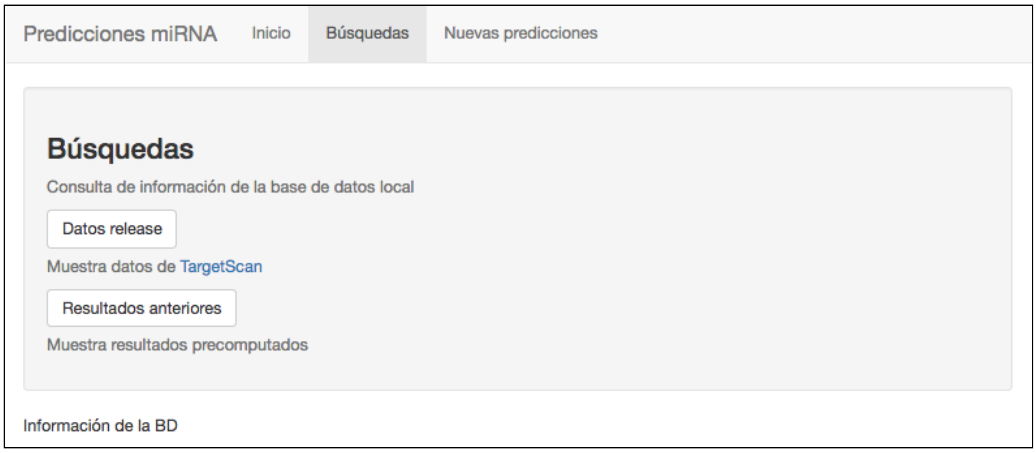


Fig. 8: Pantalla de búsqueda

- **Campos**

1. **Consulta de la release de *TargetScan*:** botón para consultar datos obtenidos de herramientas online. Se ha añadido un enlace a *TargetScan.*

   En la figura 9 se muestra la tabla de datos de la release.

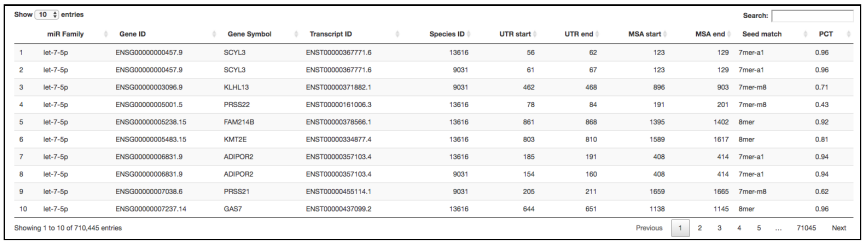


Fig. 9: Datos de release de TargetScan

- "miR Family": familia miRNA.
- "Gene ID": Identificador de gen.
- "Gene Symbol": símbolo del gen.
- "Transcript ID": Identificador del extremo 3' UTR.
- "Species ID": Identificador de especie.

- "UTR start": posición de comienzo de UTR sin contar los huecos en el alineamiento.
- "UTR end": posición de final de UTR sin contar los huecos en el alineamiento.
- "MSA start": posición de comienzo de UTR contando los huecos en el alineamiento múltiple.
- "MSA end": posición de final de UTR contando los huecos en el alineamiento múltiple.
- "Seed match": tipo de apareamiento en la semilla.
- "PCT": "Probability Conserved Targeting". Es la conversión del ratio señal/fondo (S/B) a una probabilidad de target conservado.

2. **Consulta resultados previos:** botón para consultar los resultados de la ejecución almacenados en la base de datos (figura 10).

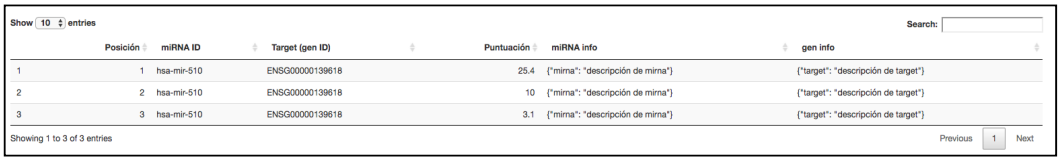


Fig. 10: Resultados de algoritmo de predicción

- Los datos mostrados son todos los resultados almacenados de la ejecución del algoritmo de predicción del usuario (ver punto 4: información de salida).

1.4. Pantalla de Predicciones

Pantalla desde la que se hacen las predicciones y se muestran los resultados (figura 11).

Acceso: barra de navegación -> Nuevas predicciones

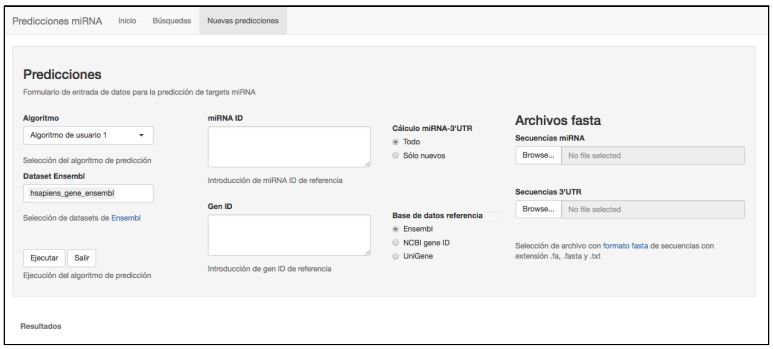


Fig. 11: Pantalla de nuevas predicciones

- **Campos**

1. **Algoritmo:** lista desplegable para seleccionar el algoritmo de predicción a ejecutar. El valor por defecto es el texto "Algoritmo de usuario 1".

2. **Dataset *Ensembl*:** lista desplegable para seleccionar los datasets de *Ensembl* con información sobre genes por especie. El valor por defecto es "hsapiens_gene_ensembl". Se pueden seleccionar de uno a todos.
3. **miRNA ID:** caja de texto para introducir los ID de miRNA de referencia.
4. **Cálculo miRNA-3'UTR:** "radio button" para elegir si se hace la predicción de todos los ID introducidos ("Todo") o sólo de los no computados previamente ("Sólo nuevos").
5. **Gen ID:** caja de texto para introducir los ID de genes de referencia.
6. **Base de datos referencia:** "radio button" para elegir la base de datos de referencia. El valor por defecto es el texto "Ensembl".
7. **Archivos fasta miRNA:** "file upload" para seleccionar archivos *fasta* de secuencias de miRNA.
8. **Archivos fasta 3' UTR:** "file upload" para seleccionar archivos *fasta* de secuencias de extremo 3' UTR.
9. **Ejecutar algoritmo:** botón para lanzar la ejecución del algoritmo seleccionado.
10. **Salir:** botón para parar la aplicación y cerrar la ventana del navegador.
11. **Actualizar fasta:** botón para eliminar los archivos fasta de los controles (se incluye por la existencia de un *bug* de *Shiny* aunque no siempre es efectivo).

    NOTA: no se ha podido encontrar una solución satisfactoria así que se ha eliminado.

    Se ha añadido un enlace a información sobre el formato de archivo *fasta*.

2. Mensajes

Los mensajes se muestran en todas las pantallas debajo de los formularios en color rojo (fig. 12).

- **Errores**

  Cuando se produzca algún error (ej: de programación o de conexión a aplicaciones de terceros o de la base de datos) se mostrará un mensaje al usuario.

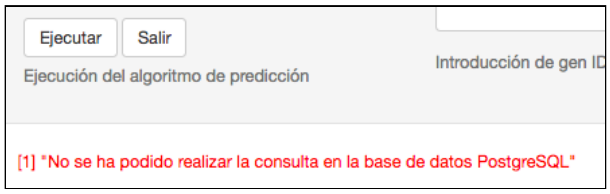


Fig. 12: detalle de mensaje de la aplicación

- **Validaciones**

  Estos mensajes aparecen cuando no se cumplen las reglas de validación, por ejemplo, al seleccionar campos del formulario o al comprobar que los

archivos fasta utilizados como entrada de datos no tienen el formato correcto (figura 13).

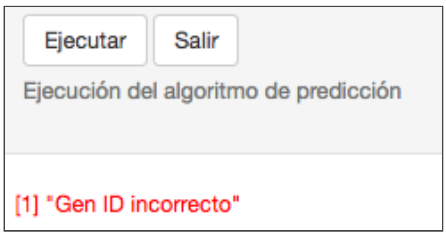


Fig. 13: detalle de validación gen ID

3. Información de salida

Los datos de salida se muestran en forma de tabla de resultados (figura 16) con funciones de búsqueda, ordenación y paginado (ver punto ·Comparación entre aplicaciones").

4.Muestra de resultados en otras aplicaciones

Se ha estudiado la representación de datos de dos herramientas online de predicción de *targets* de miRNA: *PicTar* y *TargetScan* para tener una idea más clara de cómo debe ser el diseño de la aplicación.

En ambas se puede consultar por un miRNA ID, un gen ID y especie.

- **Comparación entre aplicaciones**

  En la figuras 14 y 15 se muestra una captura de pantalla consultando por el miRNA de ejemplo**:** "hsa-let-7d-5p"**.** [30]

  - ***PicTar***

    Centrado sobre todo en mostrar datos sobre miRNA y targets junto con sus anotaciones. Uno de los datos mostrados es "Rank" que indica la posición del target ordenado de más probable a menos probable (de ser Target) en base a la puntuación otorgada por el algoritmo.

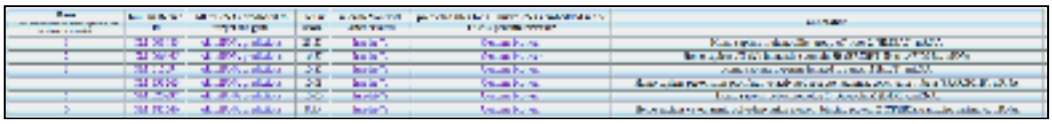


Fig. 14: captura de pantalla de la interfaz web de *PicTar* [31]

Los datos mostrados por esta herramienta son:

- "Rank": posición del *target* según su puntuación (*PicTar score*).
- "Human Refseq Id": ID de *RefSeq* [32] del *target*. Enlace a la base de datos de *RefSeq*.
- "All miRNAs predicted to target the gene": enlace al detalle de las predicciones.
- "PicTar score": puntuación del algoritmo al *target*.
- "microRNAs with Anchor sites"*:* miRNA ID seleccionado. Enlace con mirBase.

- "Predicted sites for all microRNAs embedded in the UCSC genome browser": enlace al navegador genómico *UCSC* [33] con el detalle de los sitios de unión miRNA-mRNA.
- "Annotation": detalles sobre anotación del *target*.

● ***TargetScan***

Centrado en mostrar datos sobre *features*, en especial los relacionados con el tipo de alineamiento de la semilla y la conservación de secuencias (desde el punto de vista filogenético).

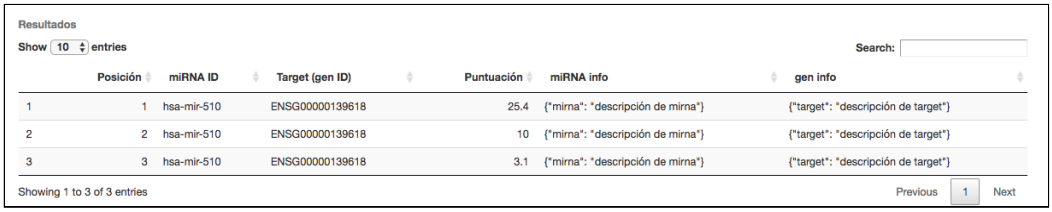
Target gene	Representative transcript	Gene name	Number of 3P-seq tags supporting UTR + 5	Link to sites in UTRs	Conserved sites				Poorly conserved sites				6mer sites	Representative miRNA	Cumulative weighted context++ score	Total context++ score	Aggregate PCT	Previous TargetScan publication(s)
					total	8mer	7mer-m8	7mer-A1	total	8mer	7mer-m8	7mer-A1						
HMGA2	ENST00000403681.2	high mobility group AT-hook 2	8021	Sites in UTR	7	1	3	3	0	0	0	0	1	hsa-let-7d-5p	-2.67	-2.69	> 0.99	2005, 2007, 2009, 2011
ARID3B	ENST00000346246.5	AT rich interactive domain 3B (BRIGHT-like)	11	Sites in UTR	5	0	1	4	0	0	0	0	0	hsa-let-7d-5p	-1.68	-1.68	> 0.99	2005, 2007, 2009, 2011
LIN28B	ENST00000345080.4	lin-28 homolog B (C. elegans)	118	Sites in UTR	4	2	2	0	1	0	0	1	0	hsa-let-7d-5p	-1.58	-1.58	> 0.99	2007, 2009, 2011
FIGN	ENST00000333129.3	fidgetin	10	Sites in UTR	6	3	0	3	0	0	0	0	0	hsa-let-7d-5p	-1.46	-2.11	> 0.99	2007, 2009, 2011

Fig. 15: captura de pantalla de la interfaz web TargetScan [34]

Los datos mostrados por esta herramienta son:

- "Target gene": símbolo del gen.
- "Representative transcript": transcripto del gen con el perfil UTR más prevalente.
- "Gene name": nombre del gen.
- "Number of 3P-seq tags supporting UTR + 5": número de etiquetas *3p-seq* que indican la localización del punto de unión miRNA-UTR.
- "Link to sites in UTRs": enlace a sitios de unión en UTR.
- "Conserved sites": número de coincidencias conservadas en el alineamiento de los tipos: *8mer*, *7mer-m8* y *7mer-1A* . [35]
- "Poorly conserved sites": número de coincidencias menos conservadas en el alineamiento de los tipos: *8mer*, *7mer-m8* y *7mer-1A*.
- "6mer sites": número de coincidencias en el alineamiento del tipo *6mer*.
- "Representative miRNA": miRNA de la familia con el menor puntuación del campo *Cumulative weighted context++ score.*
- "Cumulative weighted context++ score": puntuación acumulada ponderada de *context++* calculada en base a la contribución de 14 *features*. [36]
- "Total context++ score": puntuación total de "context++" .
- "Aggregate PCT": probabilidad de ser un *target* conservado.
- "Previous TargetScan publication(s)": año de publicación previa en *TargetScan*.

●***Aplicación***


Resultados

Show 10 entries Search:

	Posición	miRNA ID	Target (gen ID)	Puntuación	miRNA info	gen info
1	1	hsa-mir-510	ENSG00000139618	25.4	{"mirna": "descripción de mirna"}	{"target": "descripción de target"}
2	2	hsa-mir-510	ENSG00000139618	10	{"mirna": "descripción de mirna"}	{"target": "descripción de target"}
3	3	hsa-mir-510	ENSG00000139618	3.1	{"mirna": "descripción de mirna"}	{"target": "descripción de target"}

Showing 1 to 3 of 3 entries Previous 1 Next

Fig. 16: captura de pantalla de la aplicación (datos ficticios)

Los datos mostrados por esta herramienta son:

- *"*Posición*":* posición del *target* según la puntuación calculada por el algoritmo (campo "Puntuación").
- "miRNA ID"*:* ID de *miRNA*
- "Target (gen ID)"*:* ID del gen identificado como *target*.
- *"*Puntuación"*:* calculada por el algoritmo de predicción.
- *"*miRNA info": información en la literatura científica del miRNA.
- "miRNA info"*:* información en la literatura científica de gen *target*.

Estos dos últimos campos se han incluido en la base de datos y en la visualización pero no contienen datos reales recuperados mediante *text mining*. A completar en una segunda fase de desarrollo.

También tiene un campo con de ranking ("posición") y de puntuación del algoritmo similares a *PicTar*. En la aplicación se muestra menos información que en las herramientas de terceros online debido sobre todo a la falta de tiempo para desarrollar la visualización de los resultados. Sin embargo la base de datos sí está preparada para hacer cálculos sobre *binding-sites* y *features* utilizados por los algoritmos de predicción de manera que en una segunda fase de desarrollo sí se podrían mostrar datos equivalentes (ver sección 3.2: "Base de datos").

3.2. Base de datos

En esta sección se describe el gestor de base de datos elegido y como se ha implementado su uso en la aplicación a través de paquetes de R.

Terminología

- **Gestor de base de datos:** es un conjunto de software que permite almacenar, modificar y consultar información de una base de datos.
- **Modelo relacional:** los datos son guardados como relaciones entre conjuntos de datos (tuplas o filas de las tablas). Utiliza SQL [37] como lenguaje de consulta.
- **SQL:** lenguaje para manipular un gestor de bases de datos relacionales.
- **Esquema:** define los objetos de la base de datos (tablas, campos, índices, etc) y las relaciones entre ellos.
- **Clave primaria**: uno o más campos de una tabla que identifica cada fila como única.

- Clave ajena: uno o más campos de una tabla que hacen referencia a uno o más campos que son clave primaria en otra.

-PostgreSQL: gestor de bases de datos relacional de código abierto. Es la que se utiliza en el proyecto.

-Administrador de base de datos: software que proporciona una interfaz gráfica para administrar una base de datos. En este proyecto se ha usado *PgAdmin*.

-Driver de base de datos: software que permite la comunicación entre un programa y una base de datos. Son específicos para cada BBDD y lenguaje de programación**.** En la aplicación se ha usado el implementado por el paquete DBI de R.

***-RpostgreSQL*:** paquete de R para gestionar base de datos *PostgreSQL*. [38]

***-noSQL*:** tipo de gestor de bases de datos que no tiene un esquema fijo como los modelos relacionales. La base de datos se organiza en colecciones y documentos, estos últimos almacenados en forma de clave/valor como json. No usan SQL como lenguaje de acceso a datos.

- *MongoDB*: implementación de un gestor de bases de datos de tipo noSQL.

- *mongolite, rmongodb y RMongo*: paquetes de R para gestionar bases de datos *MongoDB*. [39] [40]

1. Introducción

Se ha utilizado el gestor de base de datos relacional de código abierto *PostgreSQL* y una interfaz gráfica de administración.

- **Ventajas**

-SQL como lenguaje de operaciones, lo que facilita mucho las consultas de datos.

- Fiabilidad: ya que lleva muchos años en funcionando.

- Facilidad de encontrar software relacionado con ellos como herramientas de administración gráfica, librerías, etc.

- **Desventajas**

El diseño del esquema puede resultar complejo.

Se han creado dos bases de datos en el mismo servidor: una para hacer pruebas ("mirnaplatform_bd_test″) y otra para la utilizar en entornos de producción ("mirnaplatform_bd″). Ambas tienen las mismas tablas y un esquema con el mismo nombre ("mirnaplatform″).

En un primer momento del desarrollo se pensó en utilizar un gestor de base de datos de tipo *noSQL* , en concreto la implementación de *MongoDB*. Este tipo de base de datos se consideró apropiada porque guarda la información con formato *json*, que es un formato muy utilizado para la comunicación entre aplicaciones online y además porque tiene una estructura lógica similar al tipo de dato lista de R, lo que simplifica su manipulación. Sin embargo se desechó

su uso debido a que los paquetes de R no tienen la suficiente potencia para cumplir con los objetivos del proyecto. En concreto, no devuelven los datos correctamente cuando se hacen consultas sobre subdocumentos de una colección lo que dificulta enormemente la programación.

2. Usuarios

No se ha creado ningún rol ni usuario específico. Se utiliza el usuario por defecto "postgres". La seguridad es la implementada por defecto.

3. Implementación del código relacionado con la base de datos

Los parámetros de conexión a la base de datos, los nombres de la tablas y consultas SQL se guardan en variables dentro del archivo "global.R". No es necesario guardar las secuencias ya que su valor se incrementa en una unidad enviando como valor del campo correspondiente la cadena "DEFAULT" en las consultas SQL, de forma que no es necesario guardar su referencia en variables.

- **Gestión de conexiones**

  La gestión de las conexiones a la base de datos se hace a través de funciones, lo que facilita el mantenimiento del código.

  El cierre de la conexión se realiza tanto si las operaciones tienen éxito como si no. De esta forma se evita la saturación del *driver PostgreSQL*. La cancelación de procesos de conexiones abiertas se puede hacer en el administrador de la base de datos por medio de una consulta *SQL.*

- **Paquetes de R relacionadas con la base de datos**
  - ***RpostgreSQL***: permite la conexión con una gestor de base de datos *PostgreSQL*. Gestiona las conexiones y permite ejecutar consultas SQL de tipo *DDL (**D**ata **D**efinition **L**anguage*) que crea las estructuras de la base de base de datos y *DML* (***D**ata **M**anipulation **L**anguage*) que permite la consulta, inserción y modificación de tablas.
    - **Inserciones:** se utilizan las funciones "RPostgreSQL::dbWriteTable" para insertar datos en una tabla a partir de una variable *data frame de* R y la función "RPostgreSQL::dbSendQuery", que inserta datos a partir de una cadena con una sentencia *SQL* de tipo "INSERT INTO" o "DROP".
    - **Selección, actualización y borrado:** mediante las función "RPostgreSQL::dbGetQuery", pasando como parámetro una sentencia *SQL* de tipo "SELECT", "UPDATE" o "DELETE".
  - ***DBI:*** se encarga de la gestión del *driver* de conexión a base de datos. Tiene que cargarse antes de abrir la conexión a base de datos.

- **Nuevos *Features***

  Para que el usuario desarrollador pueda crear *features* nuevos que no se contemplen en el esquema como campos específicos de la tabla "features" se crea el tipo compuesto "*new_feature*".

  Su campo "properties" es de tipo *jsonb* (json binario) y permite la introducción de datos de la forma clave / valor.La razón de utilizar json binario y no *json* es que es más eficiente al hacer consultas sobre él y permite indexación.

- **Datos de la Release de *TargetScan***

  La tabla “mirnatargets_release_targetscan” contiene datos descargados de *TargetScan* para mostrar en la pantalla de búsquedas (ver sección 3.1: "Interfaz gráfica de usuario").

  Para ello se descarga de la web un archivo comprimido llamado "Predicted_Targets_Info.default_predictions.txt.zip” **[41].** Este archivo se descomprime y se guarda en un *data frame*. Mediante la función "dbSendQuery" se borra la tabla, si existe, y la función "dbWriteTable" crea la tabla e inserta los datos del *data frame*.

  Archivo: **`busquedatargets.R`**
  Función: **`InsertarReleaseBD("archivo_release.zip", "archivo_release.txt")`**

  Esta función no está asociada a ningún evento, hay que ejecutarla directamente desde la consola del entorno integrado de desarrollo de R.

- **Clasificación de miRNA y extremo 3' UTR en precomputados o no**

  La clasificación se hace sobre miRNA ID y *transcript* ID del extremo 3’ UTR. Se crean dos funciones para hacer la clasificación que devuelven un vector con 0 (nueva o no precomputada) / 1 (referencia o precomputada) como elementos y el ID como nombre del elemento:

  Archivo: **`prediccion targets_script.R`**
  Función: **`ClasificarIDReferencia (id, data_sel, tipo, updateProgress = NULL)`**

  Archivo: **`prediccion targets_script.R`**
  Función: **`ClasificarIDPrecomputado(df_id, tipo)`**

  Como el usuario tiene que introducir siempre IDs de miRNA y extremo 3' UTR, es más útil clasificar estos pares y no por separado mediante la siguiente función:

  Archivo: **`prediccion targets_script.R`**

  Función: **`ClasificarMirnaUtrPrecomputado(mirna_id, utr_id, updateProgress = NULL)`**

Esta función recibe como parámetro miRNA IDs y gen IDs y hace una consulta de todos los pares posibles clasificándolos en "0": no precomputados o "1": precomputados.

- **Elementos del formulario conectados a la base de datos**
  Pantalla: Nuevas predicciones
  Campo: Desplegable “Dataset Ensembl”

  En el caso de que fallara la conexión con *Biomart Ensembl*, los datasets se obtendrían de la tabla “datasets_ensembl” (campo “ds_name”). El proceso de inserción se tiene que hacer manualmente. Los datasets se obtienen con la sentencia "listDatasets(dat_ensembl)" que devuelve un objeto de R con el nombre, descripción y versión de los datasets.

4. Esquema

Se crea el esquema "mirnaplatform" formado por tablas, campos, índices, tipos compuestos, secuencias, claves primarias, ajenas y valores por defecto.

- **Diagrama entidad-relación**

El diagrama entidad-relación muestra cómo se relacionan las tablas de la base datos entre ellas, como se muestra en la figura 18.

Tipos de relaciones:

-1 a 1 **:** un registro de una tabla se corresponde con un registro de otra.

-1 a muchos **:** un registro de una tabla se corresponde con más de un registro de la otra.

-Muchos a muchos **:** más de un registro de una tabla se corresponde con más de un registro de la otra.

Si aparece un círculo en la relación es que puede no haber ningún registro relacionado.

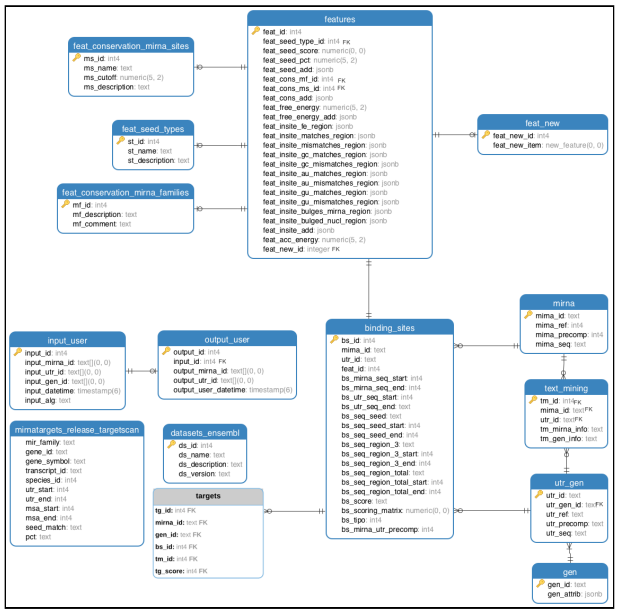


Fig. 18: diagrama entidad-relación

- **Tablas (**para ver el listado completo de campos ver Anexo I: "Base de datos").

1. ***mirna***
   Datos sobre secuencias de miRNA. Incluye el ID, secuencia, si es de referencia y si ha sido precomputado.
2. ***utr_gen***
   Datos sobre secuencias extremo 3' UTR. Incluye el ID, gen ID del que procede, si es de referencia y si ha sido precomputado.
3. ***gen***
   Datos sobre secuencias de genes. Incluye el ID y un campo con sus atributos.
4. ***binding_sites***
   Datos sobre *binding-site* de pares miRNA - extremo 3' UTR. Representa un *binding-site* con campos para guardar características de las distintas regiones de la secuencia o si ha sido precomputado.
5. ***features***
   Datos *sobre features* de pares miRNA - extremo 3' UTR. Incluye un campo para añadir *features* nuevos que no estén incluidos.
6. ***feature_new***
   Nuevos *features* creados por el usuario desarrollador.
7. ***text-mining***
   Datos sobre información de la literatura científica obtenida mediante procesos de *text mining* tanto del el miRNA *como* del el extremo 3' UTR.
8. ***input_user***
   Datos de entrada que el usuario introduce a través de la aplicación como la fecha y hora, miRNAs, extremo 3' UTR, genes consultados y el nombre del algoritmo utilizado.
9. ***output_user***
   Información de salida del algoritmo como la fecha y hora y miRNA y targets encontrados. Está relacionado con el input a través del campo "input_id".
10. ***targets***
    Datos de targets resultado de la ejecución del algoritmo de predicción. Se utiliza para mostrar los resultados al usuario.

- **Tablas auxiliares**
  Son tablas con datos que no suelen modificarse y que contienen información fija con datos como el nombre y la descripción de alguna característica como por ejemplo los distintos *features*. En caso de tener que hacer modificaciones en alguno de sus campos, sólo se modificaría esta tabla y no en todas en las que apareciera alguno de ellos. Además hace más sencillas las consultas entre varias tablas al relacionar por un ID y no por una cadena de texto.

Generalmente los campos son un identificador, un nombre y una descripción. Su clave primaria aparece en otras tablas como clave ajena.

1. ***datasets_ensembl***
   Datos de los datasets de *Ensembl*. Incluye un identificador, nombre, descripción y versión.
2. ***feat_conservation_mirna_family***
   Datos sobre el atributo "miRNA family" del *feature* "conservación de secuencias".
3. ***feat_conservation_mirna_sites***
   Datos sobre el atributo "sites miRNA" del *feature* "conservación de secuencias".
4. ***feat_seed_regions***
   Datos sobre la posición de la secuencia semilla.
5. ***feat_seed_types***
   Datos sobre el tipo de alineamiento de la secuencia semilla.

- **Otras tablas**
  1. ***mirnatargets_release_targetscan***
     El contenido de esta tabla se corresponde con el archivo con extensión *csv* descargado de *TargetScan* en el apartado "*Default predictions (conserved sites of conserved miRNA families) File: Predicted Targets (default predictions) - (8.22 MB)*".

- **Tipos**

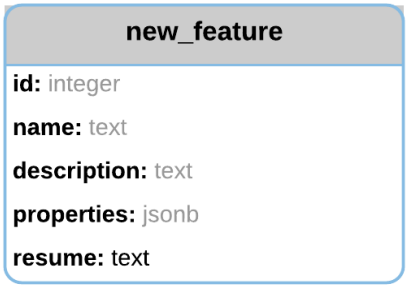


Fig. 19: tipo *new_feature*

Es un tipo compuesto que se utiliza para añadir nuevos *features* no incluidos en la tabla "binding-sites". Para ellos se utiliza el campo "properties" de tipo *json*: clave/valor.

- **Secuencias**
  Permiten generar automáticamente secuencias de números enteros correlativos. Se utilizan para los identificadores (ID) de las tablas y cuando aparecen se marcan como clave primaria y única.
- **Índices**
  Es una estructura de datos de una tabla que mejora el rendimiento de las operaciones sobre ella, asignando un identificador único a cada fila.

- **Scripts**

1. **Creación de la base de datos**
   El script de creación de la base de datos junto con el esquema y todos sus componentes está en el archivo "sql_create_db.sql". Es el mismo para la base de datos del entorno de producción o real y la del entorno de prueba, cambiando solo su nombre en script (ver anexo V: Ejemplos de código)
2. **Inserción de datos en las tablas auxiliares**
   Las sentencias de inserción de datos se guardan en el archivo "sql_insert_aux_tables.sql"

4. Implementación

En esta capítulo se describe cómo se han materializado las ideas planteadas en los capítulos anteriores en forma de código y las herramientas utilizadas para ello.

4.1. Tecnologías empleadas

Terminología

- **IDE (Integrated Development Environment)**: software que proporciona una interfaz gráfica para el desarrollo de software. Entre otras funcionalidades puede tener un editor de texto, un depurador de código, intérprete, compilador, árbol de directorios, etc. Facilita enormemente la tarea de organizar y codificar.
  En el proyecto proyecto se ha utilizado R-*RStudio.* [42]
- ***Framework*:** software que configura un esquema para el desarrollo de una aplicación. Formado por librerías con funciones que facilitan, entre otras cosas, la implementación del acceso a bases de datos y servidor web, la estructura de archivos y otras funcionalidades que simplifican el desarrollo.
  En el proyecto se ha utilizado *Shiny RStudio.* [43]
- **Servidor de aplicaciones web:** software que recibe y envía datos a un cliente (en este caso un navegador web) después de procesarlos.
  En el proyecto se ha utilizado el servidor integrado en *Shiny RStudio*.
- **Front-end:** en aplicaciones web, es la parte de la aplicación que se ejecuta en el cliente (navegador web). Es lo que ve el usuario y se basa en los lenguajes HTML, JavaScript y CSS.
- **Back-end**: en aplicaciones web, es la parte de la aplicación que se ejecuta en el servidor y se encarga del acceso a datos y lógica del programa. Interpreta lenguajes como R, *Java* o *Python*.
- **Repositorio** de código: software que hace el control de versiones de una aplicación pudiendo restaurar estados anteriores en caso de necesidad. Puede crearse en local o en la web.
- **Código abierto:** software con una licencia de uso y distribución centrada en el acceso libre al código fuente

En esta aplicación se utiliza ***GitHub*** [44] como repositorio tanto en local (integrado en *RStudio* y la aplicación *GitHub desktop*) como repositorio en la web, donde permite compartir el código fuente con otros desarrolladores

Se ha procurado utilizar software gratuito y de código abierto al no disponer de ningún presupuesto. En el caso de utilizar software de pago se ha usado versiones de prueba o sin poder acceder a toda su funcionalidad.

A continuación se hace un sumario de las tecnologías empleadas

1. **Equipo**: hace referencia a la máquina utilizada para construir el proyecto.
   - Hardware
     - Nombre del modelo: *iMac*

- Nombre del procesador: *Intel Core i7*
- Velocidad del procesador: 3,1 GHz
- Cantidad de procesadores: 1
- Cantidad total de núcleos: 4
- Caché de nivel 2 (por núcleo): 256 KB
- Caché de nivel 3: 8 MB
- Memoria: 16 GB tipo DDR3 1600 MHz
- Almacenamiento:
  - Tipo de soporte: *SSD*
  - Tamaño*:* 120,99 GB
  - Protocolo: *PCI*
  - Tipo de mapa de particiones: *GPT* (Tabla de particiones *GUID*)
  - Tamaño: 999,35 GB
  - Tipo de soporte: Rotatorio
  - Protocolo: *SATA*
  - Tipo de mapa de particiones: *GPT* (Tabla de particiones *GUID*)
- Software
  - Versión del sistema: *macOS 10.12.4*
  - Versión del kernel: *Darwin 16.5.0*

2. **Software principal**: utilizado para la creación del proyecto web.

2.1. Lenguaje de programación R v.3.3.3

Lenguaje principal con licencia *GNU General Public License (GPL) version 2* **[45]** en el que se ha codificado la aplicación. Es una implementación del lenguaje *S* con orientación estadística muy utilizado por la comunidad científica bioinformática. La mayor ventaja de R respecto de otros lenguajes como *Java* o *Python* es la gran variedad de paquetes disponibles, destacando dentro del campo de la bioinformática las que están disponibles dentro del proyecto *Bioconductor* **[46]** que proporciona herramientas para el análisis genómico y que en su versión 3.5 está formado por 1383 paquetes. De *Bioconductor* se ha utilizado *biomaRt* (proporciona una interfaz de conexión con las bases de datos de *Biomart*), *Biostrings* (permite la manipulación de secuencias de nucleótidos) **[47]** y es utilizado en el proyecto para la validación de archivos fasta y *mirbase.db* (datos sobre miRNA de bases de datos de *miRBase)*.

El entorno gráfico de desarrollo integrado (IDE) utilizado ha sido *RStudio v.1.0.136* para escritorio, también de código abierto (licencia *AGPL v3*) **[48]**, que incluye una consola, un editor de código, ayuda y documentación integradas y herramientas para la depuración y la gestión del espacio de trabajo. En esta versión está incluida la posibilidad gestionar la creación de un repositorio local de *GitHub*.

- **Ventajas**
  - Muy utilizado en el mundo de la bioinformática
  - Gran cantidad de paquetes para mejorar su funcionalidad
- **Desventajas**
  - Implementación del control de errores mejorable

2.2. *Shiny v1.0.3*

Es un framework de aplicaciones web para R. Incluye un servidor web de prueba para desplegar aplicaciones de forma transparente al desarrollador. Existe una aplicación de servidor específica para *Shiny* pero no existe todavía una versión disponible para *macOS*.

Aunque R es el lenguaje principal también se utilizan algunas sentencias código *HTML* y *JavaScript* para la interfaz de usuario.

Para crear una aplicación con *Shiny* es necesario instalar y cargar el paquete *shiny* **[49].**

Se utilizan dos paquetes para aumentar las funciones de *Shiny*: el paquete *shinyjs* **[50],** que permite hacer llamadas a funciones de *JavaScript* propias y se utiliza para mostrar mensajes al usuario, y el paquete *DT* **[51]** que permite utilizar rejillas de datos *JavaScript* de forma sencilla y que se utilizan para mostrar los resultados del algoritmo. Entre otras características, este paquete crea automáticamente un campo de búsqueda por cualquier columna, aplica un orden por columna al pinchar en cualquiera de sus encabezados y también crea la paginación si es necesaria.

- **Ventajas**
  - Ofrece un entorno web para aplicaciones codificadas en R con una curva de aprendizaje corta sin perder ninguna de las funcionalidades del lenguaje.
  - Servidor web integrado en el *framework*.
- **Desventajas**
  - Menos potente que otros frameworks de *Java* o *Python*

Se valoró la posibilidad de utilizar otros lenguajes de programación y *frameworks*:

- *Meteor* **[52]:** basado en *JavaScript* , es muy apropiado para construir la capa de presentación de la aplicación dada su sencillez. También se consideró porque implementa *MongoDB* como base de datos. Sin embargo es un lenguaje poco potente para implementar algoritmos complejos por lo que se desechó al no encontrar la forma de poder ejecutar código escrito en otros lenguajes más potente como *Java* o *Python* para desarrollar la interfaz de integración.
  Frameworks similares a *Meteor* como *ANGULARJS* **[53]** o *React* **[54]** presentan las mismas ventajas y desventajas.

- *Java*: es un lenguaje muy potente y una buena opción para hacer el proyecto dada la gran cantidad de *frameworks* disponibles como *Play* [55], *Spark* [56], *Spring* [57], *Struts* [58] y otros pero se determinó que su curva de aprendizaje era demasiado larga (especialmente *Java* 8). Además no tiene tantos paquetes en el campo de la bioinformática como R.

2.3. *PostgreSQL v9.6*

Gestor de bases de datos relacional de código abierto.

Para tareas de administración se ha utilizado *pgAdmin4* [59] y para diseñar el modelo entidad-relación se ha usado la aplicación *Navicat for PostgreSQL* (versión de prueba).

Se ha elegido *PostgreSQL* por las siguientes razones:

- Software de código abierto.
- Gran variedad de tipos de datos (*json*, fechas tipos compuestos, etc)-
- Fiable: 15 años de desarrollo.

Opciones similares a *PostgreSQL* como *MySQL* [60], *MariaDB* [61] o *SQLite* [62] se tuvieron en cuenta pero no tienen ninguna característica que destaque sobre la opción elegida.

3. **Otro software:** utilizado para otras tareas no directamente relacionadas con la creación de la aplicación.

3.1. *GitHub desktop v222 (Deer types)*: permite la gestión de repositorios en *Github*. Se utiliza esta aplicación por ser más completa que la que viene integrada en el *IDE RStudio* permitiendo no sólo gestionar varios repositorios locales sino publicar el proyecto en la web si se tiene una cuenta de usuario

3.2. *Merlin project v.4* [63]: para la creación del cronograma (versión de prueba).

3.3. *Lucidchart* [64]: para los diagramas de flujo (versión de prueba). Esta aplicación web se ha utilizado para el diagrama general de funcionamiento de la aplicación y para el diagrama de ejecución de la interfaz.

3.4. *Google drive* [65]: software ofimático online para crear toda la documentación del proyecto.

3.5. *Open refine* [66]: software con licencia de código abierto para convertir archivos grandes con formato *csv* a formato *json*.

4.2. Conexiones API

Terminología

- API: ver "Terminología" en la sección 2.1: interfaz de usuario.

- *Ensembl*: ver "Terminología" en la sección 2.1: interfaz de usuario.

- *Biomart*: ver "Terminología" en la sección 2.1: interfaz de usuario.

- *biomaRt*: ver "Terminología" en la sección 2.1: interfaz de usuario.

- ID: ver punto "Terminología" en la sección 2.1: interfaz de usuario.

- *Release*: ver punto "Terminología" en la sección 2.1: interfaz de usuario.

- Formato *fasta*: ver "Terminología" en la sección 2.1: interfaz de usuario.

- Formato *json* (JavaScript **O**bject **N**otation): formato de texto para datos estructurados en forma de clave/valor. Muy utilizado para la comunicación entre aplicaciones web. Muchos gestores de base de datos lo incorporan como tipo de dato.

- *mirBase*: base de datos de miRNA publicados datos de secuencia y anotación.

- *mirbase.db*: paquete de R que permite la conexión mediante API a la base de datos de mirBase. Se ha utilizado esta base de datos y no otra al ser el único que tiene APIs compatibles con R, aparte de ser una de las más completas

- Llamada HTTP: protocolo que entre otras funciones permite enviar datos como parámetros a un servidor web a través de una URL. Estos datos tienen que tener un formato reconocible por el servidor (ej: XML, *json*, HTML, etc).

1.Introducción

La plataforma se conecta a aplicaciones online de terceros mediante servicios de conexión API para acceder a sus bases de datos genómicas, en concreto a *Ensembl* (*release* "ensembl gene 88") y a miRBase (*release* 21).

La herramienta *BioMart* de *Ensembl* pone a disposición de los usuarios una aplicación web de consulta genómica y una gran variedad de recursos de programación, entre ellos un API de conexión a dicha web.

En este proyecto se ha optado por el paquete de R *biomaRt v2.30.0*, que proporciona una interfaz para acceder a las bases de datos de *BioMart y el* paquete *mirbase.db* para acceder a *miRBase*.

Estos paquetes están incluidos dentro del proyecto de software para bioinformáticos *Bioconductor.*

2.Implementación en el código

Las funciones donde se ha implementado la funcionalidad relacionada con las conexiones API son las siguientes:

- **Recuperación de secuencias de extremo 3' UTR a partir de un gen ID**

  Archivo: `api_script.R`

  Función: `RecuperarSecuenciasExtremo3utr(gen_id, data_sel, gen_id_ref, objeto_fasta = NULL, updateProgress = NULL)`

  En esta función se realizan los siguientes pasos:

1. Se comprueba en la base de datos si los gen ID aportados por el usuario existen en la base de datos (computados anteriormente) para reducir el tiempo de espera.
2. Si no es así, se conecta con Biomart a través de un API implementado por el paquete *biomaRt* para consultar los gen ID a través de una llamada HTTP pasando un código XML con el dataset(s) seleccionado, filtros (gen ID) y atributos (datos de salida: base de datos, extremos 3'UTR ID y su secuencia). Si existen se descarga en local un archivo fasta con los resultados (ver punto: **Acceso a la lista de datasets de *BioMart*).**

   Es necesario ejecutar el comando "wget" del sistema operativo pasándolo como parámetro en la función "system" **[67]** del paquete base, que se utiliza para ejecutar comandos del sistema operativo a través de R
3. Se valida el archivo anterior para eliminar las secuencias no válidas.

   Con este código se descarga un archivo de texto con formato fasta que contiene las secuencias del extremo 3' UTR en la carpeta local indicada como parámetro. Algunas secuencias no están disponibles y se recuperan con el siguiente formato (ejemplo):

   ```
   >ENSG00000139620|ENST00000550870
    Sequence unavailable
   ```

   Por esta razón se crea otro archivo de secuencias válidas al que le añaden sólo las secuencias correctas, que son las que se utilizarán para hacer las predicciones.

   El nombre y la ruta de estos archivos junto con los se guarda en el archivo "global.R" (ver sección 5.1: "Estructura de archivos") para facilitar modificaciones posteriores del código.

- **Acceso a la lista de datasets de *BioMart***

  - Archivo: **`conexionapi_script.R`**
  - Función: **`buscar_datasets`**

  En esta función se realizan los siguientes pasos:

  1. Se conecta con *BioMart* a través del API proporcionado por el paquete biomaRt para recuperar la relación de nombres de los dataset.
  2. Si falla la conexión carga los dataset guardados en la base de datos ver sección 3.2: "Base de datos").
  3. Los nombres de los datasets devueltos por la función se cargan posteriormente en la lista desplegable correspondiente (ver sección 3.1: "Interfaz gráfica de usuario").

  Para acceder a las bases de datos genómicas de *BioMart* es necesario indicar qué dataset se quiere utilizar. Cada uno de ellos pertenece a una

especie y para consultar más de uno es necesario hacer llamadas repetidas dentro de un bucle.

Dentro de la aplicación se pueden seleccionar en el campo del formulario "Datasets Ensembl" de la pantalla "Nuevas predicciones" (ver sección 3.1: "Interfaz gráfica de usuario").

- **Recuperación de secuencias de miRNA a partir de un miRNA ID**

  El paquete de R *mirbase.db p*roporciona una conexión API con las bases de datos de *miRBase*. Los datos de este paquete se corresponden con la *release* 21. Una vez instalada y cargada se puede hacer uso de sus funciones para recuperar las secuencias de miRNA a partir de un ID.

  En la tabla 9 se muestran datos de los cinco primeros registros recuperados mediante *mirbase.db*:

miRNA	Secuencia
cel-let-7	UACACUGUGGAUCCGGUGAGGUAGUAGGUUGUAUAGUUUGGAAUAUUACCACCG GUGAACUAUGCAAUUUUCUACCUUACCGGAGACAGAACUCUUCGA
cel-lin-4	AUGCUUCCGGCCUGUUCCCUGAGACCUCAAGUGUGAGUGUACUAUUGAUGCUUC ACACCUGGGCUCUCCGGGUACCAGGACGGUUUGAGCAGAU
cel-mir-1	AAAGUGACCGUACCGAGCUGCAUACUUCCUUACAUGCCCAUACUAUAUCAUAAA UGGAUAUGGAAUGUAAAGAAGUAUGUAGAACGGGGUGGUAGU
cel-mir-2	UAAACAGUAUACAGAAAGCCAUCAAAGCGGUGGUUGAUGUGUUGCAAAUUAUGA CUUUCAUAUCACAGCCAGCUUUGAUGUGCUGCCUGUUGCACUGU
cel-mir-34	CGGACAAUGCUCGAGAGGCAGUGUGGUUAGCUGGUUGCAUAUUUCCUUGACAAC GGCUACCUUCACUGCCACCCCGAACAUGUCGUCCAUCUUUGAA

Tabla 9: ejemplo de datos recuperados con mirbase.db

Como en el caso de los extremo 3' UTR se comprueba primero si el miRNA está en la base de datos local para reducir el tiempo de espera.

Para hacer la aplicación más robusta y tolerante a fallos, se guarda en variables del archivo "global.R" servidores alternativos ("mirror") de conexión a *BioMart*.

De este modo, en caso de fallo del servidor principal, se puede modificar el código fácilmente para que se conecte a uno de los alternativos.

Estos servidores se pueden pasar como parámetro en la conexión a Ensembl mediante el parámetro "mirror" y son: "uswest", "useast" y "asia".

4.3. Interfaz de integración

Proporciona a los desarrolladores de software las herramientas para implementar sus propios algoritmos en la herramienta. Para ello se han creado cuatro funciones sin código que se corresponden con los cuatro procesos de los que constan la mayoría de los algoritmos de predicción. El

desarrollador completará las funciones siguiendo las indicaciones y comentarios puestos a su disposición que describen los parámetros de entrada, datos de salida y ejemplos prácticos. Una vez codificado el algoritmo podrá ejecutarlo a través de la aplicación (ver código anexo III: comentarios de funciones de la interfaz de integración).
Entre cada proceso se consultará la base de datos y se harán las operaciones necesarias de lectura, inserción y actualización.
Al no estar implementado ningún algoritmo se ha creado un ejemplo para probar el funcionamiento de la aplicación (ver anexo IV: Ejemplo de prueba para el algoritmo).
Los 4 procesos/funciones son:

1. Identificar *binding-sites* potenciales

- Datos de entrada:
  miRNA y extremo 3' UTR aportados por el usuario.
  Pares miRNA - extremo 3' UTR (todos los guardados por el usuario o los que no hayan sido computados previamente).
- Datos de salida:
  Binding-sites potenciales.

Archivo: **`interfaz_script.R`**

Función:
```
IdentificarBindingSitesPotenciales(mirna_info,
utr_info, mirna_utr_no_precomp = NULL,
updateProgress = NULL)
```

En esta función se identifican y caracterizan los *binding-sites* de los miRNA y extremo 3'UTR introducidos por el usuario. Los datos de salida serán los *binding-sites* identificados como potenciales.

2. Obtener *features*

- Datos de entrada:
  Binding-features calculados en el paso anterior.
- Datos de salida:
  Features calculados por el algoritmo.

Archivo: **`interfaz_script.R`**

Función:
```
ObtenerFeatures(binding_sites_potenciales,
updateProgress = NULL)
```

En esta función se calculan los *features* correspondientes a los *binding-sites* identificados en el paso anterior. El desarrollador tiene la libertad de crear *features* nuevos que se guardan en la base de datos mediante el tipo compuesto "new_feature".

3. Evaluar *binding-sites* potenciales

- Datos de entrada:
  Binding-sites potenciales calculados en el primer paso.

Features calculados en el paso anterior.

- Datos de salida:
  Binding-sites definitivos.

Archivo: `interfaz_script.R`

Función: `EvaluarBindingSitesPotenciales(bs_potenciales, features)`

En esta función se evalúan los *binding-sites* potenciales para clasificarlos o no como definitivos para hacer la predicción final

4. Predicción final

- Datos de entrada:
  Binding-sites definitivos calculados en el paso anterior más los precalculados.
- Datos de salida:
  Targets miRNA

Archivo: `interfaz_script.R`

Función `PrediccionFinalTargets(binding_sites_definitivos, updateProgress)`

En está función se calcula la predicción final de *targets.*

- **Ejemplo**
  Al no existir ningún algoritmo implementado no se puede ejecutar la aplicación para mostrar el funcionamiento de interfaz de integración. Para simularlo, se ha creado un ejemplo con datos ficticios (ver ANEXO III: Ejemplo de prueba para el algoritmo).

- **Muestra de los resultados del algoritmo al usuario**
  Una vez obtenido el resultado del algoritmo se guardan los datos en la tabla "targets" (tabla 10), cuyo contenido se mostrará al usuario (basada en la interfaz de *PicTar*):

targets	
Campo	**Descripción**
Posición	Orden por puntuación agrupando por miRNA-target
miRNA ID	Identificador de miRNA
Target (gen ID)	Identificador de gen
Puntuación	Puntuación del binding-site
miRNA info	Información obtenida con *text mining* sobre miRNAs
gen info	Información obtenida con *text mining* sobre genes

Tabla 10: tabla *targets*

4.4. Revisión y refactorización de código

En esta sección se describen cambios en el código para detectar errores, mejorar su eficiencia, usabilidad y legibilidad teniendo en cuenta la posibilidad de desarrollos futuros.

- **Terminología**

  -Refactorización: técnica de ingeniería de software que altera la estructura del código para para mejorar su comprensión. Se considera una de las tareas a realizar en el mantenimiento de una aplicación.

  -Transacción: agrupa operaciones sobre una base de datos de forma que el resultado final es la ejecución de todas o de ninguna.

  ***-Rollback*:** acción sobre la base de datos que anula las operación incluidas en una transacción. En el proyecto se hace por medio del paquete *RPostgresSQL*.

  -***Commit*:** acción sobre una base de datos que hace efectivas de forma definitiva las operaciones sobre una base de datos.

- **Revisión de código**
  - En cada fase del proyecto se ha revisado y probado el código para detectar errores.
  - Se ha creado un control de errores para que el usuario no pierda el control sobre la aplicación en caso de error.
  - Como ejemplo están las modificaciones que se han hecho en las conexiones a la base de datos: en un principio se abría y cerraba la conexión a base de datos en cada consulta. Para hacer el código más eficiente se han creado transacciones en aquellos procesos en los que se insertan o actualizan varias tablas de forma consecutiva logrando un doble objetivo: reducir el número de conexiones y anular todas las operaciones en caso de cualquiera de ellas falle mediante un *rollback* para no producir inconsistencias de información en la base de datos (ver anexo 5: ejemplos de código).

- **Refactorización**
  - Se ha modificado el nombre de las funciones para adaptarlas a la nomenclatura consistente en todo el código **[68]**.
  - Se ha agrupado el código en archivos y funciones según su función para facilitar las tareas de mantenimiento.

5. Resultado

En este capítulo se describen los resultados obtenidos en el proyecto,

Se ha creado una herramienta formada por un conjunto de archivos y carpetas estructurados que constituyen una aplicación web. Además se han creado dos archivos para crear la base de datos (incluyendo su esquema) y para insertar datos en las tablas auxiliares; por último se aporta una memoria y presentación en vídeo basada en diapositivas.

Excepto el control de usuarios y la la recuperación de información mediante *text mining,* la aplicación es funcional. Permite la búsqueda de predicciones de *targets* y en está preparada para la implementación de algoritmos. Sin embargo,. Los datos que se visualizan al ejecutarla se corresponden a un caso de ejemplo necesario para simular el funcionamiento de la aplicación, al no haber ningún algoritmo implementado.

La base de datos es 100% funcional una vez que se ejecuten los archivos de script correspondientes.

Está programada para que su mantenimiento y modificaciones sea sencillo para los desarrolladores que además podrán descargar el código fuente del repositorio de proyectos Github.

Su punto fuerte respecto a otras aplicaciones de predicción de targets miRNA es la posibilidad implementar algoritmos de predicción propios y en lo que respecta a la interfaz de usuario es más completa e intuitiva que las herramientas de terceros estudiadas.

También se ha creado una memoria técnica y una presentación en vídeo basada en diapositivas.

5.1. Estructura de archivos

Está formada por un directorio que contiene los archivos necesarios para la ejecución de la aplicación, además de otros directorios auxiliares que no son imprescindibles pero permiten que la estructura sea más ordenada y fácil de mantener.

- **Archivos y directorios de la aplicación**
  - Directorio mirnaPlatformPredict
    - ui.R
    - server.R: contiene el código con la capa de acceso de datos (lógica y conexión con la base de datos).
    - mirnaPlatformPredict.Rproj
    - api_script.R
    - bd_script.R
    - busquedatargets_script.R
    - interfaz_script.R
    - librerias_script.R

- predicciontargets_script.R
- validaciones_script.R
- global.R
- Directorio www
  - result_utr_API.fasta
  - Predicted_Targets_Info.default_predictions.txt.zip
  - ejemplo[...].fasta
- Directorio log
  - mirnaplatform.log

● **Descripción**

- **Directorio "mirnaPlatformPredict"**: es el directorio principal o raíz de la aplicación que contiene todos sus archivos y directorios. Cualquier archivo necesario para la ejecución o funcionamiento de la aplicación debe estar dentro de este directorio.
- **Archivo "ui.R"**: contiene el código de la capa de presentación. Junto con "server.R″ es uno de los archivos principales de la aplicación ya que crea los formularios con todos sus campos y la barra de navegación con el menú. Contiene código *HTML* y *JavaScript* embebido en código específico de *Shiny.*

  En él se utilizan los siguientes paquetes:
  - ***shiny***: proporciona toda la funcionalidad de una aplicación *Shiny*.
  - ***shinyjs****:* permite hacer llamadas a funciones escritas en *JavaScript* propias. En esta aplicación se utiliza para mostrar mensajes al usuario por ejemplo en caso de que se produzca un error.
  - ***DT*****:** que permite utilizar rejillas de datos *JavaScript* de forma sencilla y que se utilizan para mostrar los resultados del algoritmo. Entre otras características, este paquete crea automáticamente un campo de búsqueda por cualquier columna, aplica un orden por columna al pinchar en cualquiera de sus encabezados y también crea la paginación si es necesaria.
- **Archivo "server.R″:** contiene el código que se ejecuta cada vez que se arranca la aplicación y en él se hacen las llamadas a las funciones que contienen la lógica de la pantalla de búsqueda de resultados y la de nuevas predicciones pasando como parámetros las variables "input", "output" y "session" que permiten el acceso a los campos de la interfaz (cajas de texto, botones, listas, etc). Esta lógica está separada en distintos archivos para que el código en su conjunto tenga más claridad al quede separado por por pantallas y funciones.

  En este archivo de servidor se importan todos los archivos con código del proyecto para que sus variables sean visibles en todos ellos.

Por último también contiene la creación de la variable "logger", necesaria para que funcione el *log* de la aplicación.

- **Archivo "busquedatargets_script.R"**: contiene el código relacionado con la pantalla de búsqueda de *targets*. Incluye la funcionalidad relacionada con los dos botones de la pantalla para consultar resultados anteriores y para consultar los datos de la *release* de *TargetScan*.
- **Archivo "predicciontargets_script.R"**: contiene el código relacionado con la pantalla de predicción de *targets*: validación de datos introducidos por el usuario, carga de los datasets de *Ensembl* en la lista de selección, inserción de datos de usuario en la base de datos y llamada para la ejecución del algoritmo.
- **Archivo "global.R"**: muy importante para el mantenimiento de la aplicación. Todos sus elementos son de acceso global en toda la aplicación por lo que cualquier modificaci en ellos se hace en este archivo (un solo sitio) y no en todo el código. Esto disminuye drásticamente la posibilidad de cometer errores.
  - Variables y constantes.
  - Consultas SQL, nombre de tablas, rutas de archivos, etc.
  - Parámetros de conexión a la base de datos.
    - Usuario
    - Contraseña
    - Host
    - Puerto
    - URL
  - Parámetros de conexión a herramientas de terceros.
    - URL y parámetros
    - Filtros
    - Atributos
- **Directorio *log*:** contiene los archivos de *log*.
- **Archivo "mirnaplatform.log"**: se guardan los mensajes que lanza la aplicación con el formato:

  [yyyy-mm-dd hh:MM:ss] ERROR [mensaje]

  Ejemplo: [2017-05-04 17:20:42] ERROR Error in (function (classes, fdef, mtable) : unable to find an inherited method for function 'dbGetQuery' for signature '"function", "character"'

  Para usar el *log* se utiliza el paquete *log4r* **[69]** que implementa un *log* basado en *log4j* **[70]** para el lenguaje Java.

- **Instrucciones para ejecutar la aplicación**

  Para poner en marcha la aplicación es necesario instalar al menos el software R, un gestor de base de datos *PostgreSQL v9.6*.

No es necesario tener instalado el *IDE RStudio* porque se puede lanzar la aplicación a través de la terminal de comandos mediante el comando "R" [71] [72].

Para esta aplicación el comando completo sería:

```
R -e "shiny::runApp('~/mirnaPlatformPredict')"
```

En las aplicaciones programadas con *Shiny* es necesario que exista en el directorio principal un archivo con el nombre "server.R" y otro con el nombre "ui.R" para que la aplicación se pueda ejecutar.

- **Archivos de la base de datos**
  - sql_create_db.sql: contiene el *script* de creación de la base de datos completa.
  - sql_insert_aux_tables.sql: contiene el script para insertar datos en las tablas auxiliares.

- **Instrucciones para crear la base de datos**

  Se abre una terminal de comandos de la base de datos y se ejecutan las instrucciones del archivo "sql_create_db.sql" que creará la base de datos junto con su esquema.

  En la misma terminal o mediante una aplicación de administración de base de datos se ejecutan las instrucciones del archivo "sql_insert_aux_tables.sql" que insertará datos en las tablas auxiliares.

4. Conclusiones

El desarrollo de esta plataforma me ha puesto de manifiesto que el funcionamiento de los miRNA y su interacción con los extremo 3' UTR es complejo y todavía en gran parte desconocido debido la gran cantidad de posibles interacciones existentes y la dificultad para crear un modelo adecuado para su comprensión, especialmente en lo referente a los *binding-sites* y *features*.

Para conseguir buenos resultados es fundamental una correcta interpretación a priori del objeto de estudio para poder plantear unos objetivos coherentes y a partir de ahí hacer una planificación apropiada.

En general se ha cumplido con los objetivos planteados, excepto el número 5: recuperación de datos en la literatura científica mediante técnicas de *text mining*. También se han terminado la mayoría de las tareas con prioridad alta, quedando pendiente alguna tarea de baja prioridad. La única tarea de alta prioridad que no se han podido desarrollar al completo tal y como estaba recogido en el plan de trabajo inicial ha sido las pruebas unitarias y de integración.

El hecho de poner el código fuente, tanto de la herramienta en sí, como de los algoritmos implementados a disposición de la comunidad científica a través de un repositorio de código online, puede derivar en un desarrollo futuro de la plataforma, añadiendo funciones nuevas y en definitiva, permitiendo a la comunidad científica de este campo disponer de una herramienta útil que contribuya al avance en su investigación.

- **Razones para no haber completado a no haber cumplido todos los objetivos el proyecto**
  - Interpretación errónea del propósito del proyecto: la interpretación inicial que hice no fue correcta del todo, especialmente en lo que respecta a la parte de interfaz de integración. Esto hizo que buena parte del trabajo hecho hasta entonces (básicamente de tipo conceptual) tuviera que ser desechado.
  - Elección incorrecta del software para crear la aplicación web y sobre todo, el cambio del modelo de base de datos de uno de tipo *noSQL* a otro relacional.
  - Falta de tiempo para el desarrollo de la aplicación, agravado por los dos puntos anteriores.
- **Trabajos futuros**
  - Probar la aplicación con un algoritmo real para saber si la interfaz se adapta a las necesidades de los desarrolladores.
  - Ampliar la información sobre resultados que se muestran al usuario estudiando las posibilidades de agrupación y visualización de datos sobre *features* especialmente.
  - Recuperar la información recogida en la literatura científica sobre miRNA y targets mediantes técnicas de *text mining* ya que este objetivo no se ha cumplido.

- permitir la consulta de microRNAs y genes o extremo 3'UTR solo, para no calcular solamente las combinaciones entre ambos.
- Implementar seguridad tanto en la aplicación (gestión de usuarios) como en la base de datos.
- Hacer pruebas de carga con varios usuarios solicitando predicciones al mismo para saber si el software utilizado es el adecuado y conocer con más exactitud las necesidades de hardware.

ANEXO I: ESQUEMA DE BASE DE DATOS PostGreSQL

Términos de la cabecera de las tablas

- **Nombre:** identificador del campo.
- **Descripción:** descripción del campo.
- **Tipo:** tipo del campo. [72]
- **Clave:** "primaria" si clave de la tabla y "ajena" si es clave primaria de otra tabla.
- **Único:** si puede repetirse**.**
- **Nulo:** si puede tener valores nulos.
- **Valor:** si tiene un valor fijo.

Tabla mirna						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
mirna_id	**ID de la secuencia miRNA**	**text**	**Primaria**	**SÍ**	**NO**	
mirna_ref	**miRNA de referencia**	**integer**		**NO**	**NO**	**0 (no) | 1 (sí)**
mirna_precomp	**Precomputado**	**integer**		**NO**	**NO**	**0 (no) | 1 (sí)**
mirna_seq	**Secuencia de nucleótidos**	**text**		**SÍ**	**NO**	

Tabla 11: tabla mirna

Tabla utr_gen						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
utr_id	**ID de la secuencia 3' UTR**	**text**	**Primaria**	**SÍ**	**NO**	
utr_gen_id	**ID del gen asociado**	**text**	**Ajena**	**NO**	**NO**	
utr_ref	**3' UTR de referencia**	**integer**		**NO**	**NO**	**0 (no) | 1 (sí)**
utr_precomp	**3' UTR precomputado**	**integer**		**NO**	**NO**	**0 (no) | 1 (sí)**
utr_seq	**Secuencia de nucleótidos de 3' UTR**	**text**		**SÍ**	**NO**	

Tabla 12: tabla utr_gen

Tabla gen						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
gen_id	**ID de la secuencia del gen**	**text**	**Primaria**	**SÍ**	**NO**	
gen_attrib	**Atributos del gen**	**jsonb**		**NO**	**SÍ**	**clave/valor**

Tabla 13: tabla gen

Tabla binding_sites						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
bs_id	**ID de fila**	**integer**	**Primaria**	**SÍ**	**NO**	**secuencial correlativo**
mirna_id	**ID de la secuencia miRNA**	**text**	**Ajena**	**NO**	**NO**	
utr_id	**ID de la secuencia 3' UTR**	**text**	**Ajena**	**NO**	**NO**	
feat_id	**ID de tabla features**	**integer**	**Ajena**	**NO**	**NO**	
bs_mirna_seq_start	**Comienzo de binding-site en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_mirna_seq_end	**Fin de binding-site en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_utr_seq_start	**Comienzo de binding-site en la secuencia de 3' UTR**	**integer**		**NO**	**NO**	
bs_utr_seq_end	**Comienzo de binding-site en la secuencia de 3' UTR**	**integer**		**NO**	**NO**	
bs_seq_seed	**Secuencia de la región 5 o semilla de binding-site**	**text**		**NO**	**NO**	
bs_seq_seed_start	**Comienzo de semilla en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_seq_seed_end	**Final de semilla en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_seq_region_3	**Secuencia de la región 3 de binding-site**	**text**		**NO**	**NO**	
bs_seq_region_3_start	**Posición de inicio de semilla en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_seq_region_3_end	**Posición de fin de región 3 en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_seq_region_total	**región 5 (semilla) + región 3**	**text**		**NO**	**NO**	
bs_seq_region_ total_start	**Posición de inicio de región total en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_seq_region_ total_end	**Posición de fin de región total en la secuencia de miRNA**	**integer**		**NO**	**NO**	
bs_score	**Puntuación**	**numeric(6,2)**		**NO**	**NO**	
bs_scoring_matrix	**Matriz de puntuación**	**numeric(2)[]**		**NO**	**NO**	
bs_other	**Otros binding-sites**	**jsonb**		**NO**	**NO**	**clave/valor**

Tabla 14: tabla binding_sites

Tabla features						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
feat_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
feat_seed_type_id	ID tipo de semilla	integer	Ajena	NO	SÍ	
feat_seed_score	Puntuación	numeric(6,2)		NO	SÍ	
feat_seed_pct	P_{CT} agregado	numeric(6,2)		NO	SÍ	
feat_seed_add	Otros atributos feat seed	jsonb		NO	SÍ	clave/valor
feat_cons_mf_id	ID mirRNA family	integer	Ajena	NO	SÍ	
feat_cons_ms_id	ID mirRNA sites	integer	Ajena	NO	SÍ	
feat_cons_add	Otros atributos feat conservation	jsonb		NO	SÍ	clave/valor
feat_free_energy	Energía libre	numeric(6,2)		NO	SÍ	
feat_free_energy_add	Otros atributos feat free energy	jsonb		NO	SÍ	clave/valor
feat_insite_fe_region	Energia libre por región	jsonb		NO	SÍ	clave/valor
feat_insite_match_region	Matches por región	jsonb		NO	SÍ	clave/valor
feat_insite_ mismatch_region	Mismatches por región	jsonb		NO	SÍ	clave/valor
feat_insite_gc_ match_region	GC matches por región	jsonb		NO	SÍ	clave/valor
feat_insite_gc_ mismatch_region	GC mismatches por región	jsonb		NO	SÍ	clave/valor
feat_insite_au_ match_region	AU matches por región	jsonb		NO	SÍ	clave/valor
feat_insite_au_ mismatch_region	AU mismatches por región	jsonb		NO	SÍ	clave/valor
feat_insite_gu_ match_region	GU matches por región	jsonb		NO	SÍ	clave/valor
feat_insite_gu_ mismatch_region	GU mismatches por región	jsonb		NO	SÍ	clave/valor
feat_insite_bulge_ mirna_region	Bulges miRNA	jsonb		NO	SÍ	clave/valor
feat_insite_bulged_ nucl_region	Bulged nucleotides	jsonb		NO	SÍ	clave/valor
feat_insite_add	Otros atributos feat free energy	jsonb		NO	SÍ	clave/valor
feat_acc_energy	Energía de acceso	numeric(6,2)		NO	SÍ	
feat_ae_add	Otros atributos de energía de acceso	jsonb		NO	SÍ	clave/valor

feat_new_id	**Nuevo feature**	**integer**		**NO**	**SÍ**	

Tabla 15: tabla *features*

Tabla feature_new						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
feat_new_id	**ID de fila**	**integer**	**Primaria**	**SÍ**	**NO**	**secuencial correlativo**
feat_new_item	**Nuevo feature**	**new_feature**		**NO**	**NO**	

Tabla 16: tabla features_new

Tabla text_mining						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
tm_id	**ID de fila**	**integer**	**Primaria**	**SÍ**	**NO**	**secuencial correlativo**
mirna_id	**ID de la secuencia miRNA**	**text**	**Ajena**	**NO**	**NO**	
utr_id	**ID de la secuencia 3' UTR**	**text**	**Ajena**	**NO**	**NO**	
tm_mirna_info	**Información sobre miRNA**	**jsonb**		**NO**	**SÍ**	**clave/valor**
tm_gen_info	**información sobre targets**	**jsonb**		**NO**	**SÍ**	**clave/valor**

Tabla 17: tabla text_mining

Tabla input_id						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
input_id	**ID de fila**	**integer**	**Primaria**	**SÍ**	**NO**	**secuencial correlativo**
input_datetime	**Fecha y hora del input**	**Text**		**SÍ**	**NO**	
input_mirna_id	**miRNA ID**	**text array**		**NO**	**NO**	
input_utr_id	**3' UTR ID**	**text array**		**NO**	**NO**	
input_gen_id	**gen ID**	**text**		**NO**	**NO**	
input_alg	**Nombre del algoritmo utilizado**	**text**		**NO**	**NO**	

Tabla 18: tabla input_id

Tabla output_id						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
output_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
output_datetime	Fecha y hora del input	text		SÍ	NO	
input_id	ID del input	integer	Ajena	NO	NO	
output_target	pares miRNA-targets	jsonb		NO	NO	

Tabla 19: tabla output_id

Tabla targets						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
tg_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
mirna_id	miRNA ID	Text	Ajena	NO	NO	
utr_id	3' UTR ID	Text	Ajena	NO	NO	
bs_id	*binding_sites* ID	integer	Ajena	NO	NO	
tx_id	*text_mining* ID	Text	Ajena	NO	NO	
tg_score	Puntuación	Numeric(6,2)		NO	NO	

Tabla 20: tabla targets

Tabla datasets_ensembl						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
ds_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
ds_name	Nombre	text		SÍ	NO	
ds_description	Descripción	text		NO	SÎ	
ds_version	Versión	text		NO	SÎ	

Tabla 21: tabla datasets_ensembl

Tabla feat_conservation_mirna_family						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
mf_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo

mf_name	Nombre	text		SÍ	NO	
mf_description	Descripción	text		NO	SÍ	

Tabla 19: tabla feat_conservation_mirna_family

Tabla feat_conservation_mirna_sites						
NOMBRE	DESCRIPCIÓN	TIPO	CLAVE	ÚNICO	NULO	VALOR
ms_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
ms_name	Nombre	text		SÍ	NO	
ms_threshold	Threshold	numeric(5,2)		NO	SÍ	
ms_description	Descripción	text		NO	SÍ	

Tabla 20: tabla feat_conservation_mirna_sites

Tabla feat_seed_regions						
NOMBRE	DESCRIPCIÓN	TIPO	CLAVE	ÚNICO	NULO	VALOR
sr_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
sr_name	Nombre	text		NO	NO	
sr_start	Posición de Inicio de secuencia	integer		NO	NO	
sr_end	Posición de final de secuencia	integer		NO	NO	
sr_description	Descripción	text		NO	SÍ	

Tabla 21: tabla feat_seed_regions

Tabla feat_seed_types						
NOMBRE	DESCRIPCIÓN	TIPO	CLAVE	ÚNICO	NULO	VALOR
st_id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
st_name	Nombre	text		SÍ	NO	
st_description	Descripción	text		NO	SÍ	

Tabla 22: tabla feat_seed_types

Tabla mirnatargets_release_targetscan						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
mir_family	Familia miRNA	integer	Primaria	SÍ	NO	secuencial correlativo
gene_id	gen ID	new_feature		NO	SÍ	
gene_symbol	Símbolo de gen	text		NO	SÍ	
transcript_id	ID de transcripto	text		NO	SÍ	
species_id	ID de especie	text		NO	SÍ	
utr_start	Posición de inicio de secuencia 3’ UTR	integer		NO	SÍ	
utr_end	Posición de fin de secuencia 3’ UTR	integer		NO	SÍ	
msa_start	Posición de inicio de secuencia UTR (contando gaps)	integer		NO	SÍ	
msa_end	Posición de fin de secuencia UTR (contando gaps)	integer		NO	SÍ	
seed_match	seed match feature	text		NO	SÍ	
pct	P_{CT} agregado	text		NO	SÍ	

Tabla 23: tabla mirnatargets_release_targetscan

Tipo new_feature						
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**CLAVE**	**ÚNICO**	**NULO**	**VALOR**
id	ID de fila	integer	Primaria	SÍ	NO	secuencial correlativo
name	Nombre	text		NO	NO	
description	Descripción	text		NO	NO	
properties	Atributos	jsonb		NO	NO	clave/valor
resume	Resumen del feature	text		NO	SÍ	

Tabla 24: tabla mirnatargets_release_targetscan

- **Secuencias**

NOMBRE	TABLA	CAMPO
bs_id_seq	binding_sites	bs_id
feat_id_seq	features	feat_id
tm_id_seq	text_mining	tm_id
input_id_seq	input_id	input_id
output_id_seq	output_id	output_id
feat_new_id_seq	feat_new	feat_new_id

tg_id_seq	targets	tg_id
datasets_ensembl_id_seq	datasets_ensembl	ds_id
new_feature_id_seq	new_feature	id
feat_cons_mirna_family_id_seq	feat_conservation_mirna_family	mf_id
feat_cons_mirna_sites_id_seq	feat_conservation_mirna_sites	ms_id
feat_seed_regions_id_seq	feat_seed_regions	sr_id

Tabla 25: tabla mirnatargets_release_targetscan

ANEXO II: ESQUEMA DE BASE DE DATOS *MongoDB*

Esquema de *MongoDB*

El esquema de base de datos creado para el gestor *MongoDB* y posteriormente desechado es:

1. Descripción

Base de datos: *MongoDB* v3.4.2 (gestor de base de datos de tipo *noSQL*).

Las bases de datos se crean con una estructura formada por “colecciones” (equivalente a “tablas” en el modelo relacional) y dentro de ellas uno o varios “documentos” (equivalente a “registros”).

2. Usuarios

Se utilizan los paquetes de R: *mongolite* , *RMongo* y *rmongodb*

Se ha creado el usuario administrador "fvarela" sin asignarlo a ningún rol específico.

3. Esquema

MongoDb no tiene un esquema como los habituales en una base de datos relacional y no exige una estructura rígida de tablas, campos y registros. A pesar de ello se puede hacer un modelo que represente cómo se almacenan y relacionan los datos. No es necesario que un documento contenga todos los campos del modelo para insertar datos en ellos.

La base de datos “mirnaplatformpredict” está formada por cinco colecciones: la colección “input”, que contiene la información de entrada del usuario; “output”, que representa la información de salida con los resultados del algoritmo; “bindingsites”, que representa la información referente a los sitios de unión miRNA-mRNA; “features”, que representa la información sobre los atributos utilizados por el algoritmo para predecir *targets* miRNA. Por último, la colección "textmining" representa la información asociada a miRNA y sus *target* recogida de forma automática de la literatura científica (*miRBase* y *PubMed*) mediante procesos de *text mining*.

La figura 20 muestra la relación entre las colecciones.

- **Relaciones**

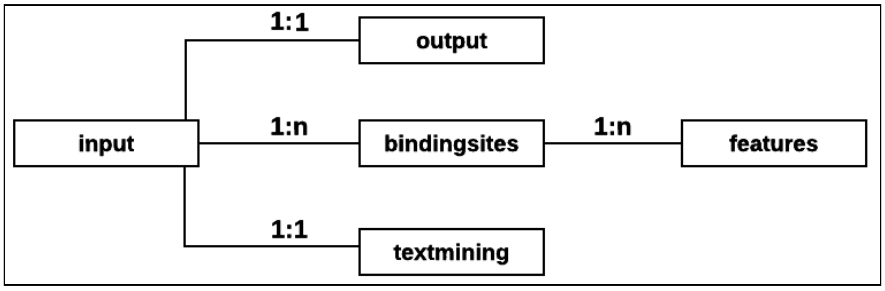


Fig.20: relaciones entre las colecciones

- **Colecciones**
  - ***input***

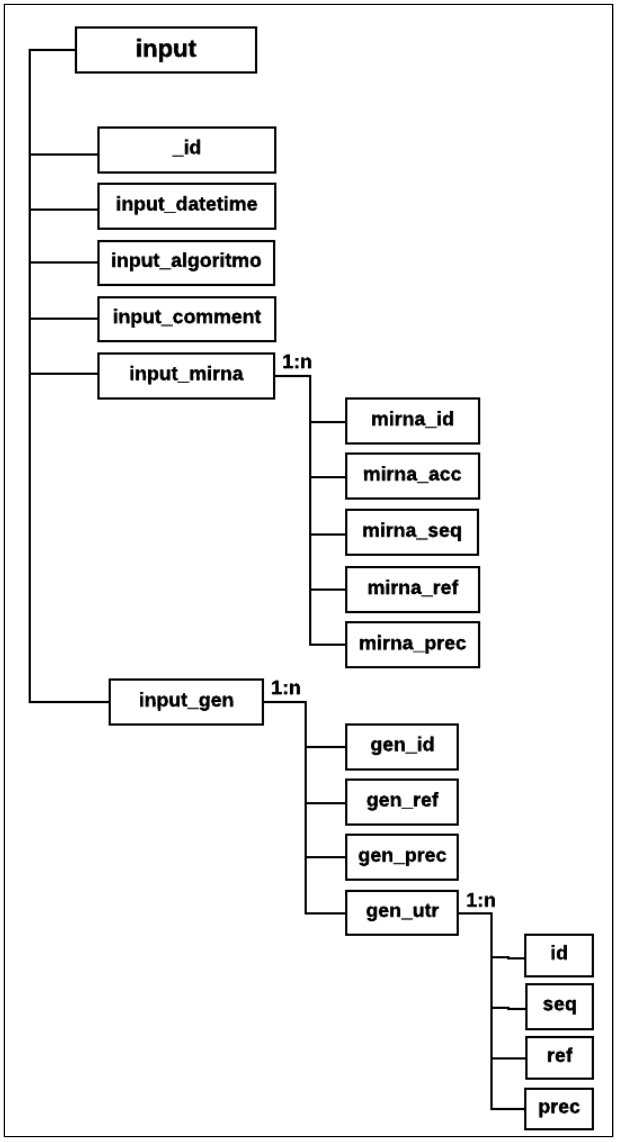


Fig. 21: validaciones

- ***output***

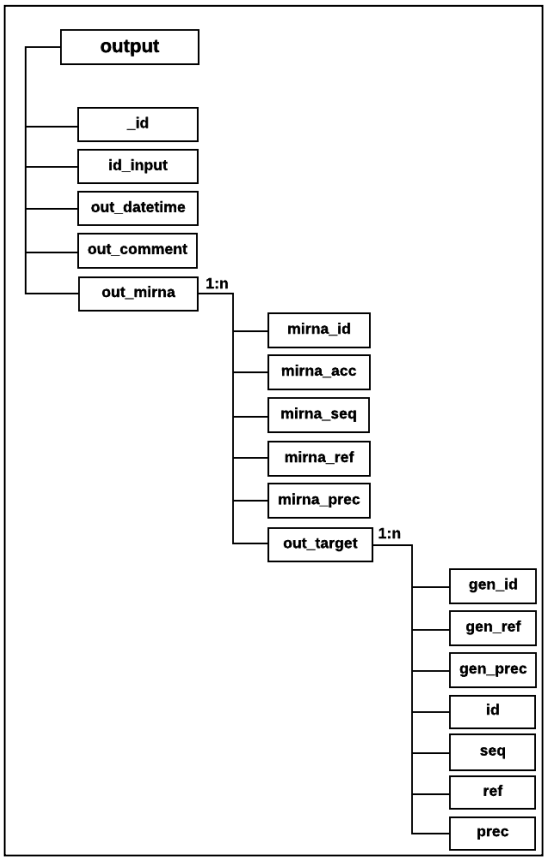


Fig. 22: colección output

- ***bindingsites***

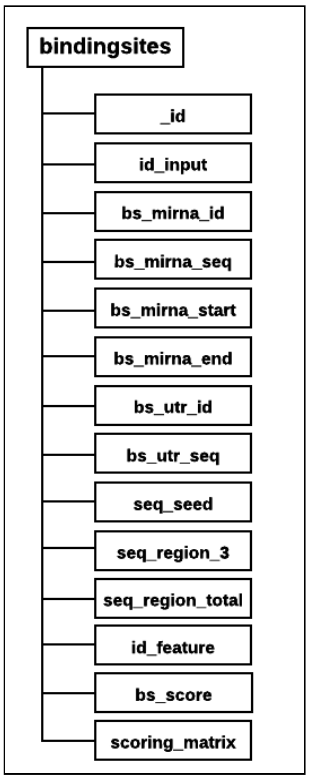


Fig. 23: colección *bindingsites*

- ***textmining***

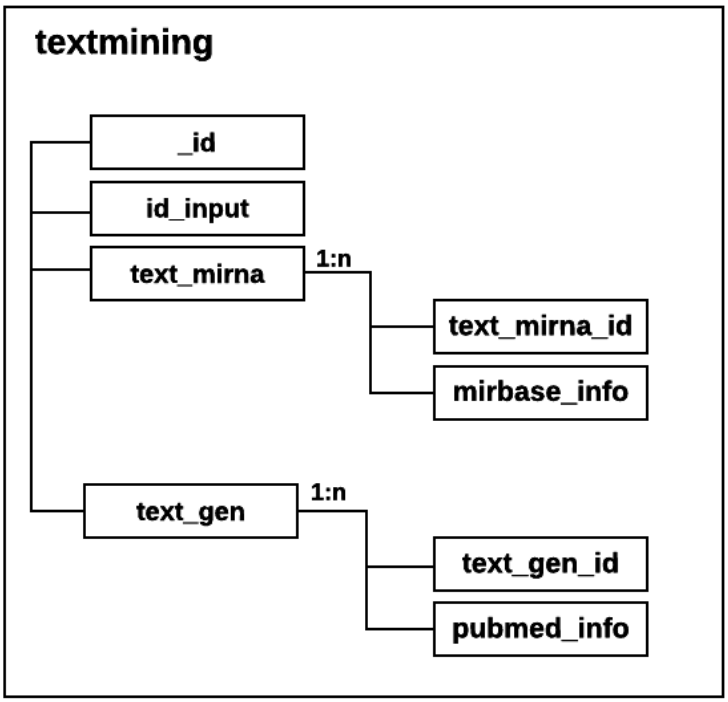


Fig 25: colección *textmining*

- ***features***

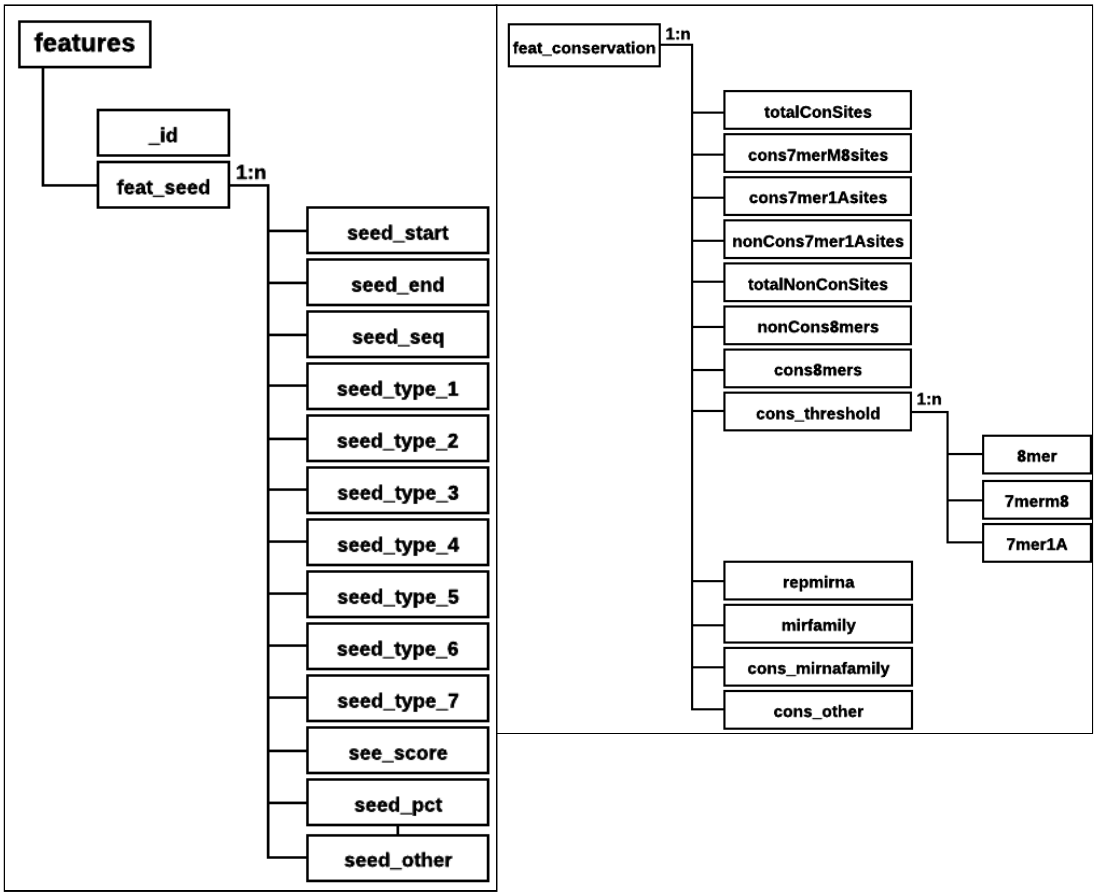


Fig. 26: colección *features*

Colección input				
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**NULO**	**VALOR**
_id	**ID documento**	**Object ID**	**NO**	**Único**
input_datetime	**Fecha de creación del documento**	**Timestamp**	**NO**	
input_algoritmo	**Nombre del algoritmo**	**String**	**NO**	
input_comment	**Notas adicionales**	**String**	**SÍ**	
input_mirna	**Atributos del mirna**	**Array**	**NO**	
mirna_id	**ID miRNA**	**String**	**NO**	
mirna_acc	**Accession number miRNA**	**String**	**SÍ**	
mirna_ref	**Secuencia de referencia**	**Integer**	**NO**	**1: Referencia | 0: Nueva**
mirna_prec	**Secuencia precomputada**	**Integer**	**NO**	**1: Precomputada | 0: no precomputada**
mirna_seq	**Secuencia de nucleótidos de miRNA**	**String**	**NO**	
input_gen	**Atributos del gen**	**Array**	**NO**	
gen_id	**ID gen**	**String**	**NO**	
gen_utr	**Atributos extremo 3'UTR**	**Array**	**SI**	
id	**ID extremo 3'UTR**	**String**	**SI**	
seq	**Secuencia de nucleótidos extremo 3'UTR**	**String**	**SI**	
ref	**Secuencia de referencia**	**Integer**	**SI**	**1: Referencia | 0: Nueva**
prec	**Secuencia precomputada**	**Integer**	**SI**	**1: Precomputada | 0: no precomputada**

Tabla 26: colección input

Colección output				
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**NULO**	**VALOR**
_id	**ID documento**	**Object ID**	**NO**	**Único**
id_input	**ID**	**String**	**NO**	**Asociado al ID del input**
out_datetime	**Fecha de creación del documento**	**Timestamp**	**NO**	
out_comment	**Notas adicionales**	**String**	**SÍ**	

out_mirna	**Atributos sobre los miRNA**	**Array**	**SÍ**	
mirna_id	**ID miRNA**	**String**	**SÍ**	
mirna_acc	**Accession number**	**String**	**SÍ**	
mirna_seq	**Secuencia de nucleótidos de miRNA**	**String**	**SÍ**	
mirna_ref	**Secuencia de referencia**	**Integer**	**SÍ**	**1: Referencia | 0: Nueva**
mirna_prec	**Secuencia precomputada**	**Integer**	**SÍ**	**1: Precomputada | 0: no precomputada**
out_target	**Atributos sobre los target miRNA**	**Array**	**SÍ**	
gen_id	**ID gen**	**String**	**SÍ**	
gen_ref	**Secuencia de referencia**	**Integer**	**SÍ**	**1: Referencia | 0: Nueva**
gen_prec	**Secuencia precomputada**	**String**	**SÍ**	
id	**ID extremo 3'UTR**	**String**	**SÍ**	
seq	**Secuencia de nucleótidos extremo 3'UTR**	**String**	**SÍ**	
ref	**Secuencia de referencia**	**Integer**	**SÍ**	**1: Referencia | 0: Nueva**
prec	**Secuencia precomputada**	**Integer**	**SÍ**	**1: Precomputada | 0: no precomputada**

Tabla 27: colección output

Colección bindingsites				
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**NULO**	**VALOR**
_id	**ID documento**	**Object ID**	**NO**	**Único**
id_input	**ID**	**String**	**NO**	**Asociado al ID del input**
bs_mirna_id	**ID miRNA**	**String**	**NO**	
bs_mirna_seq	**Secuencia de nucleótidos miRNA**	**String**	**NO**	
bs_utr_id	**ID extremo 3'UTR**	**String**	**NO**	
bs_utr_seq	**Secuencia de nucleótidos extremo 3'UTR**	**String**	**SÍ**	
bs_mirna_start	**Posición de inicio en la secuencia de miRNA**	**Integer**	**SÍ**	
bs_mirna_end	**Posición de fin la secuencia de miRNA**	**Integer**	**SÍ**	
bs_utr_start	**Posición de inicio en la secuencia de extremo**	**Integer**	**SÍ**	

	3'UTR			
bs_utr_end	**Posición de fin en la secuencia de extremo 3'UTR**	**Integer**	**SÍ**	
seq_seed	**Región 5 (semilla)**	**String**	**SÍ**	
seq_region_3	**Región 3**	**String**	**SÍ**	
seq_region_total	**Región completa**	**String**	**SÍ**	
id_feature	**ID**	**String**	**SÍ**	**Asociado al _id de la colección features**
bs_score	**Puntuación**	**Double**	**SÍ**	
scoring_matrix	**Matriz de alineamiento**	**Array**	**SÍ**	

Tabla 28: colección bindingsites

Colección textmining				
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**NULO**	**VALOR**
_id	**ID documento**	**Object ID**	**NO**	**Único**
id_input	**ID**	**String**	**NO**	**Asociado al ID del input**
text_mirna	**Atributos sobre los miRNA**	**Array**	**NO**	
text_mirna_id	**ID miRNA**	**String**	**SÍ**	
mirbase_info	**Información relacionada de MirBase**	**Array**	**SÍ**	
text_gen	**Atributos sobre los genes**	**Array**	**SÍ**	
gen_id	**ID extremo 3'UTR**	**String**	**SÍ**	
pubmed_info	**Información relacionada de Pubmed**	**Array**	**SÍ**	

Tabla 29: colección textmining

Colección features				
NOMBRE	**DESCRIPCIÓN**	**TIPO**	**NULO**	**VALOR**
_id	**ID documento**	**Object ID**	**NO**	**Único**
feat_seed	**Atributos del feature semilla**	**Array**	**SÍ**	**No definidos totalmente**
seed_start	**Comienzo de la semilla**	**Integer**	**SÍ**	

seed_end	Final de la semilla	Integer	SÍ	
seed_seq	Secuencia de la semilla	String	SÍ	
seed_type_1	Feature sí o no	Integer	SÍ	0 | 1
seed_type_2	Feature sí o no	Integer	SÍ	0 | 1
seed_type_3	Feature sí o no	Integer	SÍ	0 | 1
seed_type_4	Feature sí o no	Integer	SÍ	0 | 1
seed_type_5	Feature sí o no	Integer	SÍ	0 | 1
seed_type_6	Feature sí o no	Integer	SÍ	0 | 1
seed_type_7	Feature sí o no	Integer	SÍ	0 | 1
seed_score	Puntuación	Double	SÍ	
seed_pct	PCT agregado	Double	SÍ	
seed_other	Otros atributos	Array	SÍ	
feat_conservation	Feature conservation	Array	SÍ	true/false
totalConSites	Sites conservados	Integer	SÍ	0 | 1
cons7merM8sites	Feature sí o no	Integer	SÍ	0 | 1
cons7mer1Asites	Feature sí o no	Integer	SÍ	0 | 1
nonCons7mer1Asites	Feature sí o no	Integer	SÍ	0 | 1
totalNonConSites	Total sitios no conservados	Integer	SÍ	0 | 1
nonCons8mers	Feature sí o no	Integer	SÍ	0 | 1
cons8mers	Feature sí o no	Integer	SÍ	0 | 1
cons_threshold	Atributos de cons_threshold	Array	SÍ	
8mer	8mer	Double	SÍ	Por defecto: 0.8
7merm8	7mer-m8	Double	SÍ	Por defecto: 1.3
7mer1A	7mer-1A	Double	SÍ	Por defecto: 1.6
repmirna	miRNA representativo	String	SÍ	
mirfamily	Familia miR	String	SÍ	
cons_mirnafamily	Conservación de la familia miRNA	Integer	SÍ	1: “broadly conserved” 2: “conserved” 3: “poorly conserved”
cons_others	Otros atributos	Array	SÍ	
feat_free_energy	Feature Mínima energía libre	Array	SÍ	
free_energy	Energía libre	Double	SÍ	

feat_insite	Feature in-site	Array	SÍ	
insite_free_energy	Atributos de in-site free energy	Array	SÍ	Para la región 3, 5 y completa
insite_match	Atributos de in-site match/mismatch	Array	SÍ	Para la región 3, 5 y completa
gc_match	Atributos de in-site GC match/mismatch	Array	SÍ	
au_match	Atributos de in-site AU match/mismatch	Array	SÍ	
gu_match	Atributos de in-site AU match/mismatch	Array	SÍ	
bulge_mirna	Atributos de in-site bulge en miRNA	Array	SÍ	
bulged_nucleotides	Atributos de in-site bulged nucleótidos en miRNA	Array	SÍ	
insite_other	Otros atributos feature in-site	Array	SÍ	
feat_acc_energy	Feature energía de accesibilidad	Double	SÍ	
feat_otros	Otros features definidos por el usuario	Array	SÍ	

Tabla 30: colección features

- **Otras colecciones**

1. **Colección *mirnatargets***

   Se corresponde con la release de *TargetScan*. El archivo descargado se pasa a formato *json*. Después se importa a *MongoDB* desde la consola del sistema operativo mediante el comando "mongoimport" con el servidor de base de datos *MongoDB* arrancado:

```
> mongod
> mongoimport -h localhost --db mirna --collection mirnatargets
--type json --file ~/downloads/targeScanPreds16092016_total.json
--jsonArray;
```

```
2017-04-15T20:49:14.700+0200 connected to: localhost
2017-04-15T20:49:17.692+0200 [####....................] mirna.mirnatargets 51.1MB/247MB
(20.7%)
2017-04-15T20:49:20.695+0200 [#########...............] mirna.mirnatargets 102MB/247MB
(41.1%)
2017-04-15T20:49:23.696+0200 [###############.........] mirna.mirnatargets 152MB/247MB
(61.6%)
2017-04-15T20:49:26.695+0200 [####################.....] mirna.mirnatargets 204MB/247MB
(82.4%)
2017-04-15T20:49:29.285+0200 [########################] mirna.mirnatargets 247MB/247MB
(100.0%)
2017-04-15T20:49:29.285+0200 imported 710445 documents
```

ANEXO III: COMENTARIOS DE CÓDIGO DE LA INTERFAZ DE INTEGRACIÓN

1. Identificar *binding-sites* potenciales

- Datos de entrada:

  miRNA y extremo 3' UTR aportados por el usuario.

  Pares miRNA - extremo 3' UTR (todos los guardados por el usuario o los que no hayan sido computados previamente).

- Datos de salida:

  Binding-sites potenciales.

Archivo: **`interfaz_script.R`**

Función: **`IdentificarBindingSitesPotenciales(mirna_info, utr_info, mirna_utr_no_precomp = NULL, updateProgress = NULL)`**

Comentarios para el usuario desarrollador:

```
#' Función IdentificarBindingSitesPotenciales(mirna_info, utr_info,
mirna_utr_no_precomp = NULL)
#' Ejecuta el primer paso del algoritmo de predicción para identificar
los binding-sites potenciales
#' Se pueden calcular todos los pares miRNA y 3'UTR o sólo aquellos no
precomputados#'
#' @param mirna_info data.frame - Datos de ID y secuencia de miRNA
#' mirna_id: character - ID de miRNA
#' mirna_seq: character - secuencia de nucleótidos de miRNA
#'
#' @param utr_info data.frame - Datos de ID y secuencia de extremo
3'UTR
#' utr_id: character - ID de 3'UTR
#' utr_gen: character - ID de gen
#' utr_seq: character - secuencia de nucleótidos de extremos 3'UTR
#'
#' @param mirna_utr_no_precomp data.frame - ID de pares miRNA y 3'UTR
no precomputados
#' bs_id: integer
#' mirna_id: character
#' utr_id: character
#' precomp: integer
#'
#' @param updateProgress función - barra de espera (valor por defecto:
NULL)
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' @return df_binding_sites_potenciales: data.frame - info de
binding-sites potenciales
#'
#' Estructura de df_binding_sites_potenciales
#' bs_id integer - PK NOT NULL (valor: 0)
```

```
#' mirna_id - character
#' utr_id - character
#' feat_id - integer (valor: 0)
#' bs_mirna_seq_start - integer
#' bs_mirna_seq_end - integer
#' bs_utr_seq_start - integer
#' bs_utr_seq_end - integer
#' bs_seq_seed - character
#' bs_seq_seed_start - integer
#' bs_seq_seed_end - integer
#' bs_seq_region_3 - character
#' bs_seq_region_3_start - integer
#' bs_seq_region_3_end - integer
#' bs_seq_region_total - character
#' bs_seq_region_total_start - integer
#' bs_seq_region_total_end - integer
#' bs_score - character
#' bs_scoring_matrix - array
#' bs_type - integer NOT NULL valor 0: potencial
#' bs_other - json (clave: valor)
#'
#' @examples IdentificarBindingSitesPotenciales(c("hsa-mir-510", 1, 1,
"GUGGUGUCCUACUCAGGAGAGUGGCAAUCACAUGUAAUUAGGUGUGAUUGAAACCUCUAAGAGUGGAGUA
ACAC"),
#' c("ENST00000367029", 1, 1,
"GAACTGTGGGAGACCAGCGGAGTGGGAGGGAGACGCAGTAGACAGAGACAGACCGAGAGAGGAATGGAGA
GACAGAGGGGGCGCGCGCACAGGAGCCTGACTCCGCTGGGAGAGTGCAG
#'
GAGCACGTGCTGTTTTTTATTTGGACTTAACTTCAGAGAAACCGCTGACATCTAGAACTGACCTACCACAA
GCATCCACCAAAGGAGTTTGGGATTGAGTTTTGCTGCTGTGCAGCACTGCATTGTCATGACATTTCCAACA
CTGTGTGA
#'
ATTATCTAAATGCGTCTACCATTTTGCACTAGGGAGGAAGGATAAATGCTTTTTATGTTATTATTATTAAT
TATTACAATGACCACCATTTTGCATTTTGAAATAAAAAAACTTTTTATACCA"), updateProgress)
```

2. Obtener *features*

- Datos de entrada:

  Binding-features calculados en el paso anterior.

- Datos de salida:

  Features calculados.

Archivo: **`interfaz_script.R`**
Función: **`ObtenerFeatures(binding_sites_potenciales, updateProgress = NULL)`**
Comentarios para el usuario desarrollador:

```
#' Función ObtenerFeatures(binding_sites_potenciales, updateProgress =
NULL)
#' Ejecuta el segundo paso del algoritmo de predicción para identificar
los features
#'
```

```
#' @param binding_sites_potenciales data.frame - info sobre
binding-sites
#'
#' Estructura de binding_sites_potenciales
#' bs_id integer - PK NOT NULL
#' mirna_id - character
#' utr_id - character
#' feat_id - integer,
#' bs_mirna_seq_start - integer
#' bs_mirna_seq_end - integer
#' bs_utr_seq_start - integer
#' bs_utr_seq_end - integer
#' bs_seq_seed - character
#' bs_seq_seed_start - integer
#' bs_seq_seed_end - integer
#' bs_seq_region_3 - character
#' bs_seq_region_3_start - integer
#' bs_seq_region_3_end - integer
#' bs_seq_region_total - character
#' bs_seq_region_total_start - integer
#' bs_seq_region_total_end - integer
#' bs_score - character
#' bs_scoring_matrix - array
#' bs_type - integer NOT NULL
#' bs_other - json (clave: valor)
#'
#' @param updateProgress función - barra de espera (valor por defecto:
NULL)
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' @return df_features: data.frame - Datos de los features.
#' Importante: hay que incluir el ID de binding_sites (campo bs_id)
pasado en el parámetro binding_sites_potenciales
#'
#' Estructura de df_features
#' bs_id integer (binding_sites ID)
#' feat_seed_type_id - integer
#' feat_seed_score- double(6, 2)
#' feat_seed_pct - double(6, 2)
#' feat_seed_add - json (clave: valor)
#' feat_cons_mf_id - integer
#' feat_cons_ms_id - integer
#' feat_cons_add - json (clave: valor)
#' feat_free_energy - double(6, 2)
#' feat_free_energy_add - json (clave: valor)
#' feat_insite_fe_region - json (clave: valor)
#' feat_insite_match_region - json (clave: valor)
#' feat_insite_mismatch_region - json (clave: valor)
#' feat_insite_gc_match_region - json (clave: valor)
#' feat_insite_gc_mismatch_region - json (clave: valor)
```

```
#' feat_insite_au_match_region - json (clave: valor)
#' feat_insite_au_mismatch_region - json (clave: valor)
#' feat_insite_gu_match_region - json (clave: valor)
#' feat_insite_gu_mismatch_region - json (clave: valor)
#' feat_insite_bulges_mirna_region - json (clave: valor)
#' feat_insite_bulged_nucl_region - json (clave: valor)
#' feat_insite_add - json (clave: valor)
#' feat_acc_energy - double(5, 2)
#' feat_ae_add - json (clave: valor)
#' feat_new -  string ("(name: string, description: string, attributes:
jsonb resume: text)") Si no, cadena vacía ("")
#'
#' @examples ObtenerFeatures(df_binding_sites, updateProgress)
#' df_binding_sites <- data.frame(26, 'hsa-mir-510', 'ENST00000367029',
NULL, 9, 4, 54, 65, 'AGUAACAC', 2, 54,
#' 'CUAAGAGUGG', 3, 32, 'CUAAGAGUGGAGUAACAC', 5, 23, 152.2, '{{1, 2,
3},{2, 3, 4}}', 0, '{}');)
#'
#' names(df_binding_sites) <- c("bs_id", "mirna_id", "utr_id",
"bs_mirna_seq_start", "bs_mirna_seq_end",
#' "bs_utr_seq_start", "bs_utr_seq_end", "bs_seq_seed",
"bs_seq_seed_start", "bs_seq_seed_end", "bs_seq_region_3",
#' "bs_seq_region_3_start", "bs_seq_region_3_end",
"bs_seq_region_total", "bs_seq_region_total_start",
#' "bs_seq_region_total_end", "bs_score", "bs_scoring_matrix",
"bs_type", "bs_other")
```

3. Evaluar *binding-sites* potenciales

- Datos de entrada:

  Binding-sites potenciales calculados en el primer paso.

  Features calculados en el paso anterior.

- Datos de salida:

  Binding-sites definitivos.

Archivo: **`interfaz_script.R`**

Función: **`EvaluarBindingSitesPotenciales(bs_potenciales, features)`**

Comentarios para el usuario desarrollador:

```
#' Función EvaluarBindingSitesPotenciales(bs_potenciales, features)
#' Ejecuta el tercer paso del algoritmo de predicción: evaluación de
los binding-sites potenciales
#'
#' @param bs_potenciales data.frame - info de binding-sites potenciales
#'
#' Estructura de bs_potenciales
#' bs_id integer - PK NOT NULL
#' mirna_id - character
#' utr_id - character
#' feat_id - integer,
#' bs_mirna_seq_start - integer
#' bs_mirna_seq_end - integer
```

```
#' bs_utr_seq_start - integer
#' bs_utr_seq_end - integer
#' bs_seq_seed - character
#' bs_seq_seed_start - integer
#' bs_seq_seed_end - integer
#' bs_seq_region_3 - character
#' bs_seq_region_3_start - integer
#' bs_seq_region_3_end - integer
#' bs_seq_region_total - character
#' bs_seq_region_total_start - integer
#' bs_seq_region_total_end - integer
#' bs_score - character
#' bs_scoring_matrix - array
#' bs_type - integer NOT NULL
#' bs_other - json (clave: valor)
#'
#' @param features data.frame - info de features asociados a los
binding-sites
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' Estructura de features
#' bs_id - integer (binding_sites ID)
#' feat_seed_type_id - integer
#' feat_seed_score - double(6, 2)
#' feat_seed_pct - double(6, 2)
#' feat_seed_add - json (clave: valor)
#' feat_cons_mf_id - integer
#' feat_cons_ms_id - integer
#' feat_cons_add - json (clave: valor)
#' feat_free_energy - double(6, 2)
#' feat_free_energy_add - json (clave: valor)
#' feat_insite_fe_region - json (clave: valor)
#' feat_insite_match_region - json (clave: valor)
#' feat_insite_mismatch_region - json (clave: valor)
#' feat_insite_gc_match_region - json (clave: valor)
#' feat_insite_gc_mismatch_region - json (clave: valor)
#' feat_insite_au_match_region - json (clave: valor)
#' feat_insite_au_mismatch_region - json (clave: valor)
#' feat_insite_gu_match_region - json (clave: valor)
#' feat_insite_gu_mismatch_region - json (clave: valor)
#' feat_insite_bulges_mirna_region - json (clave: valor)
#' feat_insite_bulged_nucl_region - json (clave: valor)
#' feat_insite_add - json (clave: valor)
#' feat_acc_energy - double(5, 2)
#' feat_ae_add - json (clave: valor)
#' feat_new_id - integer
#'
#' @param updateProgress función - barra de espera (valor por defecto:
NULL)
```

```
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' @return df_binding_sites_evaluados - integer vector - ID de
binding-sites definitivos
#' bs_id integer - ID de binding_site
#'
#' @examples EvaluarBindingSitesPotenciales (df_binding_sites,
df_features, updateProgress)
#' df_binding_sites <- data.frame(2, 13, "hsa-mir-181a-2",
"ENST00000354071", "", 25, 4, 54, 65, "AAACACAACAAAACCAT", 2,
#' 54, "AAACACA", 3, 32, "AAACACAACAAAAAACACAACAAA", 5, 23, 152.2,"{1,
2, 3},{2, 3, 4}", 1, "{}", stringsAsFactors = FALSE)
#'
#' names(df_binding_sites) <- c("bs_id", "mirna_id", "utr_id",
"feat_id", "bs_mirna_seq_start", "bs_mirna_seq_end",
#' "bs_utr_seq_start", "bs_utr_seq_end", "bs_seq_seed",
"bs_seq_seed_start", "bs_seq_seed_end", "bs_seq_region_3",
#' "bs_seq_region_3_start", "bs_seq_region_3_end",
"bs_seq_region_total", "bs_seq_region_total_start",
#'
"bs_seq_region_total_end","bs_score","bs_scoring_matrix","bs_type","bs_
other")
#'
#' df_features <- data.frame(8, 15, 128.30, 54.20, "{'seed_k': 23}", 4,
1, "{'cons_x': 'conservationx'}", 123.30,
#' "{free_energy_x': 332.3}", "{rt': 21.3, r2': 34, r3':
332.3}","{'rt': 5.3, 'r2': 43, 'r3': 23.3}",
#' "{'rt': 4.3, 'r2': 34, 'r3': 43.3}", "{'rt': 2.3, 'r2': 1, 'r3':
65.3}", "{'rt': 0.3, 'r2': 32, 'r3': 3.3}",
#' "{'rt': 7.3, 'r2': 6, 'r3': 67.3}", "{'rt': 5.3, 'r2': 5, 'r3':
4.3}", "{'rt': 21.3, 'r2': 34, 'r3': 5.3}",
#' "{'rt': 2.3, 'r2': 34, 'r3': 78.3}", "{'rt': 21.3, 'r2': 7.8, 'r3':
98.3}", "{'rt': 6.3, 'r2': 34.8, 'r3': 332.3}",
#' "{}", 232.00, "", 1, stringsAsFactors = FALSE)
#'
#' names(df_features) <- c("feat_id", "feat_seed_type_id",
"feat_seed_score", "feat_seed_pct", "feat_seed_add", "feat_cons_mf_id",
#' "feat_cons_ms_id", "feat_cons_add", "feat_free_energy",
"feat_free_energy_add", "feat_insite_fe_region",
"feat_insite_match_region",
#' "feat_insite_mismatch_region", "feat_insite_gc_match_region",
"feat_insite_gc_mismatch_region", "feat_insite_au_match_region",
#' "feat_insite_au_mismatch_region", "feat_insite_gu_match_region",
"feat_insite_gu_mismatch_region", "feat_insite_bulges_mirna_region",
#' "feat_insite_bulged_nucl_region", "feat_insite_add",
"feat_acc_energy", "feat_ae_add", "feat_new_id")
```

4. Predicción final

- Datos de entrada:

  Binding-sites definitivos calculados en el paso anterior más los precalculados.

- Datos de salida:

  Targets miRNA

Archivo: **`interfaz_script.R`**

Función: **`PrediccionFinalTargets(binding_sites_definitivos, updateProgress)`**

Comentarios para el usuario desarrollador:

```
#' Función PrediccionFinalTargets(binding_sites_definitivos,
updateProgress):
#' Ejecuta el cuarto paso del algoritmo de predicción para hacer la
predicción final
#'
#' @param binding_sites_definitivos data.frame - info de binding-sites
definitivos evaluados en el paso anterior
#'
#' Estructura de binding_sites_evaluados
#' bs_id integer - PK NOT NULL
#' mirna_id - character
#' utr_id - character
#' feat_id - integer,
#' bs_mirna_seq_start - integer
#' bs_mirna_seq_end - integer
#' bs_utr_seq_start - integer
#' bs_utr_seq_end - integer
#' bs_seq_seed - character
#' bs_seq_seed_start - integer
#' bs_seq_seed_end - integer
#' bs_seq_region_3 - character
#' bs_seq_region_3_start - integer
#' bs_seq_region_3_end - integer
#' bs_seq_region_total - character
#' bs_seq_region_total_start - integer
#' bs_seq_region_total_end - integer
#' bs_score - character
#' bs_scoring_matrix - array
#' bs_type - integer NOT NULL
#' bs_other - json - clave - valor
#'
#' @param updateProgress función - barra de espera (valor por defecto -
NULL)
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' @return df_target - dataframe - Info de las predicciones
#' Estructura de df_target
#' mirna_id - character - miRNA ID
#' utr_id - character - 3'UTR ID
#' bs_id - integer- ID de binding-site definitivo
#' score - numeric (6,2) - Puntuación para el binding site
#'
```

```
#' @examples PrediccionFinalTargets(binding_sites_definitivos,
updateProgress)

#' Función PrediccionFinalTargets(binding_sites_definitivos,
updateProgress):
#' Ejecuta el cuarto paso del algoritmo de predicción para hacer la
predicción final
#'
#' @param binding_sites_definitivos data.frame - info de binding-sites
definitivos evaluados en el paso anterior
#'
#' Estructura de binding_sites_evaluados
#' bs_id integer - PK NOT NULL
#' mirna_id - character
#' utr_id - character
#' feat_id - integer,
#' bs_mirna_seq_start - integer
#' bs_mirna_seq_end - integer
#' bs_utr_seq_start - integer
#' bs_utr_seq_end - integer
#' bs_seq_seed - character
#' bs_seq_seed_start - integer
#' bs_seq_seed_end - integer
#' bs_seq_region_3 - character
#' bs_seq_region_3_start - integer
#' bs_seq_region_3_end - integer
#' bs_seq_region_total - character
#' bs_seq_region_total_start - integer
#' bs_seq_region_total_end - integer
#' bs_score - character
#' bs_scoring_matrix - array
#' bs_type - integer NOT NULL
#' bs_other - json - clave - valor
#'
#' @param updateProgress función - barra de espera (valor por defecto -
NULL)
#' Código dentro de la función: if (is.function(updateProgress)) { text
<- "miRNA ID -> seq miRNA"; updateProgress(detail = text) }
#'
#' @return df_target - dataframe - Info de las predicciones
#' Estructura de df_target
#' mirna_id - character - miRNA ID
#' utr_id - character - 3'UTR ID
#' bs_id - integer- ID de binding-site definitivo
#' score - numeric (6,2) - Puntuación para el binding site
#'
#' @examples PrediccionFinalTargets(binding_sites_definitivos,
updateProgress)
```

ANEXO IV: CASO DE EJEMPLO PARA EL ALGORITMO

mirna_info			
mirna_id	**mirna_seq**	**mirna_ref**	**mirna_precomp**
hsa-mir-510	GUGGUGUCCUACUCAGGAGAGUGGCAAUCACAUGUAAUUAGGUGUGAUUG AAACCUCUAAGAGUGGAGUAACAC	1	0

Tabla 31:ejemplo miRNA algoritmo

utr_info				
gen_id	**utr_id**	**mirna_seq**	**mirna_ ref**	**mirna_ precomp**
ENSG00000139618	ENST00000380152	GTATTGGTATACTTTTGCTTCAGTTGCATATCTTAAA ACTAAATGTAATTTATTAACTAATCAAGAAAAACATC TTTGGCTGAGCTCGGTGGCTCATGCCTGTAATCCCAA CACTTTGAGAAGCTGAGGTGGGAGGAGTGCTTGAGGC CAGGAGTTCAAGACCAGCCTGGGCAACATAGGGAGAC CCCCATCTTTACAAAGAAAAAAAAAAGGGGAAAAGAA AATCTTTTAAATCTTTGGATTTGATCACTACAAGTAT TATTTTACAAGTGAAATAAACATACCATTTTCTTTTA GATTGTGTCATTAAATGGAATGAGGTCTCTTAGTACA GTTATTTTGATGCAGATAATTCCTTTTAGTTTAGCTA CTATTTTAGGGGATTTTTTTTAGAGGTAACTCACTAT GAAATAGTTCTCCTTAATGCAAATATGTTGGTTCTGC TATAGTTCCATCCTGTTCAAAAGTCAGGATGAATATG AAGAGTGGTGTTTCCTTTTGAGCAATTCTTCATCCTT AAGTCAGCATGATTATAAGAAAAATAGAACCCTCAGT GTAACTCTAATTCCTTTTTACTATTCCAGTGTGATCT CTGAAATTAAATTACTTCAACTAAAAATTCAAATACT TTAAATCAGAAGATTTCATAGTTAATTTATTTTTTTT TTCAACAAAATGGTCATCCAAACTCAAACTTGAGAAA ATATCTTGCTTTCAAATTGGCACTGATTCTGCCTGCT TTATTTTTAGCGCTATCACAGGACCCAGAGCCTATGC CCTTTTAAACTTACCACAAAAGCAGAAGATTAATTCA ATTTAAGATGATA...	1	0
ENSG00000139618	ENST00000380152	CCTCCCAAGTAGCTGGGACTACAGAATATTGACATAC TTTGCAATGAAGCAGAAAACAAGCTTATGCATATACT GCATGCAAATGATCCCAAGTGGTCCACCCCAACTAAA GACTGTACTTCAGGGCCGTACACTGCTCAAATCATTC CTGGTACAGGAAACAAGCTTCTGATGTCTTCTCCTAA TTGTGAGATATATTATCAAAGTCCTTTATCACTTTGT ATGGCCAAAAGGAAGTCTGTTTCCACACCTGTCTCAG CCCAGATGACTTCAAAGTCTTGT	1	0
ENSG00000139618	ENST00000528762	AAACACAACAAAACCATATTTACCATCACGTGCACTA ACAAGACAGCAAGTTCGTGCTTTGCAAGATGGTGCAG AGCTTTATGAAGCAGTGAAGAATGCAGCAGACCCAGC TTACCTTGAGGGTTATTTCAGTGAAGAGCAGTTAAGA GCCTTGAATAATCACAGGCAAATGTTGAATGATAAGA AACAAGCTCAGATCCAGTTGGAAATTAGGAAGGCCAT GGAATCTGCTGAACAAAAGGAACAAGGTTTATCAAGG GATGTCACAACCGTGTGGAAGTTGCGTATTGTAAGCT ATTC	1	0

ENSG00000139618	ENST00000544455	GCATTTGCAAAGGCGACAATAAATTATTGACGCTTAA CCTTTCCAGTTTATAAGACTGGAATATAATTTCAAAC CACACATTAGTACTTATGTTGCACAATGAGAAAAGAA ATTAGTTTCAAATTTACCTCAGCGTTTGTGTATCGGG CAAAAATCGTTTTGCCCGATTCCGTATTGGTATACTT TTGCTTCAGTTGCATATCTTAAAACTAAATGTAATTT ATTAACTAATCAAGAAAAACATCTTTGGCTGAGCTCG GTGGCTCATGCCTGTAATCCCAACACTTTGAGAAGCT GAGGTGGGAGGAGTGCTTGAGGCCAGGAGTTCAAGAC CAGCCTGGGCAACATAGGGAGACCCCCATCTTTACAA AGAAAAAAAAAAAGGGGAAAAGAAAATCTTTTAAATCT TTGGATTTGATCACTACAAGTATTATTTTACAATCAA CAAAATGGTCATCCAAACTCAAACTTGAGAAAATATC TTGCTTTCAAATTGGCACT	1	0

Tabla 32: ejemplo extremo 3' UTR algoritmo

- Fase 1: Identificación de *binding-sites* potenciales

Datos de entrada:

Variables *"mirna_info"* y "*utr_info"* del paso anterior.

Datos de salida:

Binding-sites potenciales.

***Binding-sites* potenciales**									
mirna_id	**utr_id**	**bs_mirna_seq_start**	**bs_mirna_seq_end**	**bs_utr_seq_start**	**bs_utr_seq_end**	**bs_seq_seed**	**bs_seq_seed_start**	**bs_seq_seed_end**	**bs_seq_region_3**
hsa-mir-510	ENST00000470094	25	34	54	65	AGTCTTGT	1	8	GATGACTTCA
hsa-mir-510	ENST00000380152	21	29	30	41	AAAAAATT	2	9	TAAATCTTTT
hsa-mir-510	ENST00000528762	15	24	10	21	AGCTATTC	3	10	GCGTATTGTA
hsa-mir-510	ENST00000544455	1	9	11	20	TTGGCACT	4	11	TTGCTTTCAA

Tabla 33: ejemplo Binding-sites potenciales 1

bs_seq_region_3_start	**bs_seq_region_3_end**	**bs_seq_region_total**	**bs_seq_region_total_start**	**bs_seq_region_total_end**	**bs_score**	**bs_scoring_matrix**	**bs_other**	**bs_type**

9	19	AGTCTTGTGATGACTTCA	1	19	60	{{1, 2, 3},{2, 3, 4}}	{}	0
10	20	AAAAAATTTAAATCTTTT	2	20	41.1	{{5, 6, 7},{8, 9, 10}}	{}	0
3	13	AGCTATTCGCGTATTGTA	1	21	54	{{3, 4, 5},{6, 7, 8}}	{}	0
5	15	TTGGCACTTTGCTTTCAA	1	22	152.2	{{1, 3, 5},{7, 9, 11}}	{}	0

Tabla 34: ejemplo Binding-sites potenciales 2

- Fase 2: Obtener *features*

Datos de entrada:

Binding-sites potenciales del paso anterior.

Datos de salida:

Features.

Features							
bs_id	**feat_seed_type_id**	**feat_seed_score**	**feat_seed_pct**	**feat_seed_add**	**feat_cons_mf_id**	**feat_cons_ms_id**	**feat_cons_add**
284	15	23.2	3.3	{"seed": {"7mer-m8": 12}}	4	3	{"conservation": "conserved"}
280	15	43	1.5	{"seed": {"7mer-m8": 23}}	4	3	{"conservation": "conserved"}
285	15	34.1	53.9	{"seed": {"7mer-m8": 87}}	4	3	{"conservation": "conserved"}
281	15	352.1	0.3	{"seed": {"7mer-m8": 32}}	4	3	{"conservation": "poorly conserved"}

Tabla 35: ejemplo features 1

feat_free_ energy	feat_free_ energy_add	feat_insite_ fe_region	feat_insite_ match_region	feat_insite_ mismatch_region
43.2	{"free_energy1" : 23.2}	{"r3": 34.3, "r2": 2, "rt": 3}	{"r3": 332.3, "r2": 34, "rt": 21}	{"r3": 23.3, "r2": 43, "rt": 5}
98.2	{"free_energy1" : 45.2}	{"r3": 54.8, "r2": 21, "rt": 2}	{"r3": 18.3, "r2": 34, "rt": 11}	{"r3": 11.9, "r2": 333, "rt": 123}
1.2	{"free_energy1" : 89.1}	{"r3": 40.2, "r2": 19, "rt": 3}	{"r3": 52.2, "r2": 58, "rt": 21.2}	{"r3": 11, "r2": 23.2, "rt": 5}
66.4	{"free_energy": 1.2}	{"r3": 2, "r2": 27, "rt": 14}	{"r3": 332.3, "r2": 29, "rt": 44}	{"r3": 3.5, "r2": 3.3, "rt": 5.2}

Tabla 36: ejemplo *features* 2

eat_insite_ gc_match_ region	feat_insite_ gc_mismatch_ region	feat_insite_ au_match_ region	feat_insite_ au_mismatch_ region	feat_insite_ gu_match_ region
{"r3": 43.3, "r2": 34, "rt": 4}	{"r3": 65.3, "r2": 1, "rt": 2.3}	{"r3": 3.3, "r2": 32, "rt": 0}	{"r3": 67.3, "r2": 6, "rt": 7.3}	{"r3": 4.3, "r2": 5, "rt": 5.3}
{"r3": 12.3, "r2": 31, "rt": 34}	{"r3": 35.2, "r2": 1, "rt": 1.2}	{"r3": 8.8, "r2": 21, "rt": 23.3}	{"r3": 31.3, "r2": 98, "rt": 8.1}	{"r3": 41.8, "r2": 5, "rt": 3.9}
{"r3": 62, "r2": 66, "rt": 4}	{"r3": 20, "r2": 97, "rt": 2.8}	{"r3": 79, "r2": 80.8, "rt": 0}	{"r3": 59.2, "r2": 64, "rt": 7.3}	{"r3": 60.8, "r2": 8, "rt": 5.1}
{"r3": 34, "r2": 41, "rt": 374}	{"r3": 65.3, "r2": 45, "rt": 15.3}	{"r3": 1.2, "r2": 32.1, "rt": 22}	{"r3": 50, "r2": 43, "rt": 49}	{"r3": 4.3, "r2": 5, "rt": 18.2}

Tabla 37: ejemplo features 3

feat_ insite_gu_ mismatch_ region	feat_ insite_ bulges_ mirna_ region	feat_ insite_ bulged_ nucl_ region	feat_ insite_ add	feat_ acc_ energy	feat_ ae_ add
{"r3": 5.3, "r2": 34, "rt": 21.3}	{"r3": 78.3, "r2": 34, "rt": 2.3}	{"r3": 98.3, "r2": 7.8, "rt": 21.3}	{"r3": 332.3, "r2": 34.8, "rt": 6.3}	34.3	{"access_energy": 432.1}
{"r3": 6.3, "r2": 23.1, "rt": 34.2}	{"r3": 61.3, "r2": 1, "rt": 3.3}	{"r3": 8.5, "r2": 7.8, "rt": 76.2}	{"r3": 213.1, "r2": 38, "rt": 16.2}	232.34	{"access_energy": 150.8}
{"r3": 100, "r2": 5, "rt": 21.3}	{"r3": 16, "r2": 12, "rt": 2.3}	{"r3": 71, "r2": 44, "rt": 21.3}	{"r3": 5.2, "r2": 41.3, "rt": 6.3}	23.2	{"access_energy": 332.3}

{"r3": 25.9, "r2": 34, "rt": 2.6}	{"r3": 7, "r2": 17, "rt": 13}	{"r3": 98.3, "r2": 4, "rt": 21.8}	{"r3": 22.4, "r2": 3.7, "rt": 2.3}	145.2	{"access_energy": 12.7}

Tabla 38: ejemplo *features* 3

- Fase 3: Evaluación de *binding-sites* potenciales

Datos de entrada:

Binding-sites potenciales obtenidos en el paso 1 y *features* obtenidos en el paso anterior.

Datos de salida:

ID de los *binding-sites* evaluados como definitivos.

ID de binding-sites definitivos
bs_id
1
2
3

Tabla 39: ejemplo ID binding-sites definitivos

- Fase 4: Predicción final

Datos de entrada:

Binding-sites definitivos cuyos ID se han obtenido en el paso anterior.

Datos de salida:

Targets de cada miRNA.

df_prediccion_final			
mirna_id	**utr_id**	**bs_id**	**score**
hsa-mir-510	ENST00000470094	1	10.0
hsa-mir-510	ENST00000380152	2	25.4
hsa-mir-510	ENST00000528762	3	3.1

Tabla 40: ejemplo prediccion_final

***bs_id*:** campo ID correspondiente de la tabla "binding_sites"
***score*:** puntuación asignada por el algoritmo a un *binding-site*.

ANEXO V: EJEMPLOS DE CÓDIGO

- Scripts de creación de la base de datos
  Real:
  ```
  CREATE DATABASE mirnaplatform_bd WITH TEMPLATE = template0 OWNER = postgres;
  ```
  Prueba:
  ```
  CREATE DATABASE mirnaplatform_bd_test WITH TEMPLATE = template0 OWNER = postgres;
  ```

- **Liberación de conexiones de base datos**
  ```
  SELECT pg_terminate_backend(pg_stat_activity.pid)
  FROM pg_stat_activity
  WHERE datname = current_database()
  AND pid <> pg_backend_pid();
  ```

- **Muestra al usuario de resultados del algoritmo**
  Sentencia *SQL*:
  ```
  SELECT RANK() OVER (PARTITION BY  tg.mirna_id, tg.gen_id ORDER BY
  tg_score DESC) as "Posición", tg.mirna_id as "miRNA ID", tg.gen_id as
  "gen ID", tg_score as "Puntuación", tm_mirna_info::text as "miRNA info",
  tm_gen_info::text as "gen info"
  FROM mirnaplatform.targets AS tg, mirnaplatform.text_mining AS tm WHERE
  tg.tm_id = tm.tm_id ORDER BY tg.mirna_id, gen_id
  ```
  Se hace un *join* entre la tabla "targets" y "text_mining" por el campo común "tm_id". Para calcular el ranking (campo "posición") se calcula con los comandos *SQL* "RANK() OVER", se agrupa por "mirna_id" y "gen_id" con "PARTITION BY" y se ordena de mayor a menor con "ORDER BY tg_score".

- **Ejemplo de revisión y refactorización de código: gestión de transacciones**
  Con este código se pretende actualizar (comando SQL: "UPDATE") una tabla varias veces en un bucle. Si se produjera un error en alguna consulta se quiere que ninguna actualización realizada antes del error tenga efecto. Para ello se utiliza una solo objeto de conexión a la base de datos que se abre al inicio y se cierra al final del código, y se crea una transacción que agrupará a todas las consultas. Si se produce un error, se ejecutaría la instrucción "dbRollback(conn = con)" que anularía todos los cambios hechos en la base de datos en la transacción. Si todo es correcto se ejecuta "dbCommit(conn = con)" que haría permanentes los cambios.
  Antes de utilizar transacciones abría y cerraba una conexión con cada actualización de forma que si alguna fallaba se podían producir inconsistencias en los datos.
  ```
  con < ConectarBD()
   if (is.null(mensaje_error) == FALSE) { return(FALSE) }
        consulta_sql < "BEGIN TRANSACTION"
        EjecutarConsultaBD(con_db = con, consulta_sql = consulta_sql)
        if (is.null(mensaje_error) == FALSE) {
  ```

```
 dbRollback(conn = con);
 DesconectarBD(con_db = con); return(FALSE) }
 for(i in 1 : length(v_binding_sites_definitivos)){
   if (is.function(updateProgress)) {
       text < "Evaluando bindingsites";
       updateProgress(detail = text) }
       # Se actualizan los bindingsites a definitivos (valor: 1)
               consulta_sql < sprintf(SQL_UPDATE_BINDING_SITES_ID,
         1, v_binding_sites_definitivos[i])
   EjecutarConsultaBD(con_db=con, consulta_sql=consulta_sql)
    if (is.null(mensaje_error) == FALSE) {
         dbRollback(conn = con);
         DesconectarBD(con_db = con);
         return(FALSE) }
   }
    dbCommit(conn = con)
DesconectarBD(con_db = con) # Se cierra la conexión
```

5.Referencias

1 Krek, A., Grün, D., Poy, M. N., Wolf, R., Rosenberg, L., Epstein, E. J., ... & Rajewsky, N. (2005). Combinatorial microRNA target predictions. Nature genetics, 37(5), 495-500.

2 Lewis, B. P., Shih, I. H., Jones-Rhoades, M. W., Bartel, D. P., & Burge, C. B. (2003). Prediction of mammalian microRNA targets. *Cell*, *115*(7), 787-798.

3 Friedman, R. C., Farh, K. K. H., Burge, C. B., & Bartel, D. P. (2009). Most mammalian mRNAs are conserved targets of microRNAs. Genome research, 19(1), 92-105.

4 Hsu, J. B. K., Chiu, C. M., Hsu, S. D., Huang, W. Y., Chien, C. H., Lee, T. Y., & Huang, H. D. (2011). miRTar: an integrated system for identifying miRNA-target interactions in human. *BMC bioinformatics*, *12*(1), 300.

5 Miranda, K. C., Huynh, T., Tay, Y., Ang, Y. S., Tam, W. L., Thomson, A. M., ... & Rigoutsos, I. (2006). A pattern-based method for the identification of MicroRNA binding sites and their corresponding heteroduplexes. *Cell*, *126*(6), 1203-1217.

6 Yue, D., Liu, H., & Huang, Y. (2009). Survey of computational algorithms for microRNA target prediction. *Current genomics*, *10*(7), 478-492.

7 Zhang, B., Pan, X., Cobb, G. P., & Anderson, T. A. (2006). Plant microRNA: a small regulatory molecule with big impact. *Developmental biology*, *289*(1), 3-16.

8 Peterson, S. M., Thompson, J. A., Ufkin, M. L., Sathyanarayana, P., Liaw, L., & Congdon, C. B. (2014). Common features of microRNA target prediction tools. *Frontiers in genetics*, *5*, 23.

9 Ruvkun, G. (2001). Glimpses of a tiny RNA world. *Science*, *294*(5543), 797-799.

10 Saumet, A., & Lecellier, C. H. (2006). Anti-viral RNA silencing: do we look like plants?. *Retrovirology*, *3*(1), 3.

11 Lee, I., Ajay, S. S., Yook, J. I., Kim, H. S., Hong, S. H., Kim, N. H., ... & Athey, B. D. (2009). New class of microRNA targets containing simultaneous 5′-UTR and 3′-UTR interaction sites. *Genome research*, *19*(7), 1175-1183.

12 Kluiver, J., Poppema, S., de Jong, D., Blokzijl, T., Harms, G., Jacobs, S., ... & van den Berg, A. (2005). BIC and miR−155 are highly expressed in Hodgkin, primary mediastinal and diffuse large B cell lymphomas. *The Journal of pathology*, *207*(2), 243-249.

13 Lanza, G., Ferracin, M., Gafà, R., Veronese, A., Spizzo, R., Pichiorri, F., ... & Negrini, M. (2007). mRNA/microRNA gene expression profile in microsatellite unstable colorectal cancer. *Molecular cancer*, *6*(1), 54.

14 Matsubara, H., Takeuchi, T., Nishikawa, E., Yanagisawa, K., Hayashita, Y., Ebi, H., ... & Osada, H. (2007). Apoptosis induction by antisense oligonucleotides against miR-17-5p and miR-20a in lung cancers overexpressing miR-17-92. *Oncogene*, *26*(41), 6099-6105.

15 Rodriguez, A., Vigorito, E., Clare, S., Warren, M. V., Couttet, P., Soond, D. R., ... & Vetrie, D. (2007). Requirement of bic/microRNA-155 for normal immune function. *Science*, *316*(5824), 608-611.

16 Lu, X. Y., & Huang, X. L. (2008). Plant miRNAs and abiotic stress responses. *Biochemical and biophysical research* communications, *368*(3), 458-462.

17 Baltimore D, Boldin MP, O'Connell RM, Rao DS, Taganov KD. MicroRNAs: new regulators of immune cell development and function. Nat. Immunol. 2008;9(8):839–845.

18 Wang, X. (2008). miRDB: a microRNA target prediction and functional annotation database with a wiki interface. Rna, 14(6), 1012-1017.

19 Wang, X. (2016). Improving microRNA target prediction by modeling with unambiguously identified microRNA-target pairs from CLIP-ligation studies. Bioinformatics, 32(9), 1316-1322.

20 Wong, N., & Wang, X. (2015). miRDB: an online resource for microRNA target prediction and functional annotations. Nucleic acids research, 43(D1), D146-D152.

21 Kertesz, M., Iovino, N., Unnerstall, U., Gaul, U., & Segal, E. (2007). The role of site accessibility in microRNA target recognition. *Nature genetics*, *39*(10), 1278-1284.

22 Andrés-León, E., González Peña, D., Gómez-López, G., & Pisano, D. G. (2015). miRGate: a curated database of human, mouse and rat miRNA–mRNA targets. *Database*, *2015*.

23 Kozomara, A., & Griffiths-Jones, S. (2014). miRBase: annotating high confidence microRNAs using deep sequencing data. *Nucleic acids research*, *42*(D1), D68-D73.

24 Aken, B. L., Ayling, S., Barrell, D., Clarke, L., Curwen, V., Fairley, S., ... & Howe, K. (2016). The Ensembl gene annotation system. *Database*, *2016*.

25 R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

26 BLAST/BLAT search (Mar 2017). Retrieved from http://www.ensembl.org/Multi/Tools/Blast

27 Variant Effect Predictor (Mar 2017).Retrieved from http://www.ensembl.org/info/docs/tools/vep/index.html

28 Gene. Retrieved from https://www.ncbi.nlm.nih.gov/gene

29 Unigene. Retrieved from https://www.ncbi.nlm.nih.gov/unigene

30 Stem-loop sequence hsa-let-7d. Retrieved May 10, 2017, from http://mirbase.org/cgi-bin/mirna_entry.pl?acc=hsa-let-7d-5p

31 Con permiso del dr. Nikolaus Rajewsky

32 O'Leary, N. A., Wright, M. W., Brister, J. R., Ciufo, S., Haddad, D., McVeigh, R., ... & Astashyn, A. (2016). Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic acids research*, *44*(D1), D733-D745.

33 Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, D. (2002). The human genome browser at UCSC. *Genome research*, *12*(6), 996-1006.

34 Human | let-7-5p/98-5p. Retrieved May 10, 2017, from http://www.targetscan.org/cgi-bin/targetscan/vert_71/targetscan.cgi?species=Human&gid=&mir_sc=&mir_c=&mir_nc=&mir_vnc=&mirg=hsa-let-7d-5p

35 TargetScan Frequently Asked Questions (FAQs) Data download Punto 3. Retrieved from http://www.targetscan.org/faqs.Release_7.html

36 Agarwal, V., Bell, G. W., Nam, J. W., & Bartel, D. P. (2015). Predicting effective microRNA target sites in mammalian mRNAs. Elife, 4, e05005.

37 Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377-387.

38 Joe Conway, Dirk Eddelbuettel, Tomoaki Nishiyama, Sameer Kumar Prayaga and Neil Tiffin (2016). RPostgreSQL: R interface to the PostgreSQL database system. R package version 0.4-1. https://CRAN.R-project.org/package=RPostgreSQL

39 Chheng, T. (2011). RMongo: MongoDB client for R.

40 Lindsly, G. (2012). rmongodb: R-MongoDB driver.

41 Data Download Default predictions (conserved sites of conserved miRNA families) Predicted Targets (default predictions) - (8.22 MB)
Retrieved from http://www.targetscan.org/cgi-bin/targetscan/data_download.vert71.cgi

42 Team, R. (2015). RStudio: integrated development for R. *RStudio, Inc., Boston, MA URL* http*://www. rstudio. com*.

43 RStudio, S. a web application framework for r (2015). URL*:* http://shiny.rstudio.com

44 The new native. Retrieved from https://desktop.github.com

45 License, G. G. P. (1991). Version 2, June 1991. Free Software Foundation.

46 Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., ... & Hornik, K. (2004). Bioconductor: open software development for computational biology and bioinformatics. Genome biology, 5(10), R80.

47 H. Pagès, P. Aboyoun, R. Gentleman and S. DebRoy (2016). Biostrings: String objects representing biological sequences, and matching algorithms. R package version 2.42.1.

48 License, G. G. P. (2007). Version 3, June 29, 2007. Free Software Foundation.

49 Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2015). Shiny: web application framework for R. *R package version 1.0.3*.

50 Attali, D. shinyjs: Perform Common JavaScript Operations in Shiny Apps using Plain R Code, 2016. *R package version 0.9*

51 Yihui Xie (2016). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.2. https://CRAN.R-project.org/package=DT

52 Patil, A. A., Oundhakar, S. A., Sheth, A. P., & Verma, K. (2004, May). Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web* (pp. 553-562). ACM.

53 ANGULARJS [Computer software]. Retrieved from https://angularjs.org

54 React A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES [Computer software]. Retrieved from https://facebook.github.io/react

55 The High Velocity Web Framework For Java and Scala. [Computer software] Retrieved from https://www.playframework.com

56 Spark - A micro framework for creating web applications in Kotlin and Java 8 with minimal effort. [Computer software]. Retrieved from http://sparkjava.com

57 Spring. Let's build a better Enterprise [Computer software]. Retrieved from https://spring.io

58 Apache Struts [Computer software]. Retrieved from https://struts.apache.org

59 pgAdmin4 [Computer software]. Retrieved from https://www.pgadmin.org

60 MySQL [Computer software]. Retrieved from https://www.mysql.com

61 MariaDB [Computer software]. Retrieved from https://mariadb.org

62 Miranskyy, A. V., Al-zanbouri, Z., Godwin, D., & Bener, A. B. (2017). Database Engines: Evolution of Greenness. *arXiv preprint arXiv:1701.02344*.

63 Merlin project [Computer software]. Retrieved from https://projectwizards.net/en/products/merlin-project/what-is

64 Lucidchart [Computer software]. Retrieved from https://www.lucidchart.com

65 Google docs [Computer software]. Retrieved from https://www.google.com/intl/es_ALL/drive

66 Open refine [Computer software]. Retrieved from http://openrefine.org

67 Invoke a System Command [R package]. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/system.html

68 Bååth, R. (2012). The state of naming conventions in R. *The R Journal*, *4*(2), 74-75.

69 John Myles White (2014). log4r: A simple logging system for R, based on log4j. R package version 0.2. https://CRAN.R-project.org/package=log4r

70 Apache Log4j 2 [Java package]. Retrieved from https://logging.apache.org/log4j

71 How to launch a Shiny app. (2014, June 06). Retrieved June 02, 2017, from https://shiny.rstudio.com/articles/running.html

72 Run Shiny Application. (n.d.). Retrieved June 02, 2017, from https://shiny.rstudio.com/reference/shiny/latest/runApp.html