Métodos computacionales para la caracterización de dianas de microRNA en datos CLIP-seq

José Antonio Piqueras Mellado

Máster universitario en Bioinformática y Bioestadística

Aplicación de técnicas de inteligencia artificial al estudio de microARNs

Consultor: Albert Pla Planas
Profesor: Jose Antonio Morán Moreno

Junio de 2018

Copyright © 2018 JOSÉ A. PIQUERAS.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Métodos computacionales para la caracterización de dianas de microRNA en datos CLIP-seq*
Nombre del autor:	*José Antonio Piqueras Mellado*
Nombre del consultor/a:	*Albert Pla Planas*
Nombre del PRA:	*Jose Antonio Morán Moreno*
Fecha de entrega (mm/aaaa):	06/2018
Titulación:	*Máster universitario en Bioinformática y Bioestadística*
Área del Trabajo Final:	*Aplicación de técnicas de inteligencia artificial al estudio de microARNs*
Idioma del trabajo:	*Castellano*
Palabras clave:	*MicroRNA, CLIP-seq, Minería de datos*

Resumen del Trabajo:

Los microARNs (miRNAs) son pequeños ARNs que intervienen en la regulación de la expresión génica inhibiendo el proceso de traducción de las dianas o ARNs mensajeros (mRNAs) con los que interaccionan. Se sabe que estas moléculas juegan un rol importante en el control de ciertos procesos celulares y que están evolutivamente conservadas. El objetivo principal de este TFM consiste en caracterizar las interacciones que se dan entre los microARNs y sus dianas utilizando datos CLIP-seq obtenidos en ensayos en humano (*Homo sapiens*) y ratón (*Mus musculus*). Primeramente se extraen algunas variables descriptivas relacionadas con las secuencias nucleotídicas y con la estructura secundaria de los dímeros resultantes de la interacción miRNA-diana. Más tarde se emplean algunos algoritmos de *clustering* para estudiar la relación entre los atributos extraídos. Aunque los resultados obtenidos no pueden considerarse concluyentes es posible apreciar patrones consistentes en las dos especies sometidas a estudio. No obstante, todavía quedan interrogantes sin despejar y se requieren esfuerzos adicionales para solventar algunas limitaciones metodológicas.

Abstract:

MicroRNAs (miRNAs) are small RNA molecules involved in the regulation of gene expression. They act by inhibiting the translation process of the targets or messenger RNAs (mRNAs) which they interact with. It is known that these molecules play an important role in the control of certain cellular processes and they are also evolutionarily conserved. The main goal of this Master's Thesis is to characterize the interactions that occur between microRNAs and their targets by using CLIP-seq data obtained in human (*Homo sapiens*) and mouse (*Mus musculus*) assays. First, some descriptive variables related to the nucleotide sequences and to the secondary structure of the dimer resulting from the miRNA-target interaction are extracted. Later, several clustering algorithms are used to study the relationship between those attributes. Although the results obtained can not be considered as conclusive, it is possible to appreciate consistent patterns in the two species under study. However, there are still unsolved questions and additional efforts are required to deal with some methodological limitations.

Índice

1. Introducción **1**
- 1.1. Contexto y justificación del trabajo . . . 1
- 1.2. Objetivos . . . 1
- 1.3. Enfoque y método a seguir . . . 2
- 1.4. Planificación con hitos y temporización . . . 3
- 1.5. Resultados esperados . . . 6

2. Marco teórico **8**
- 2.1. Biología de los miRNAs . . . 8
- 2.2. Tecnología CLIP-seq . . . 10
- 2.3. Análisis de *clustering* . . . 11

3. Material y métodos **13**
- 3.1. Recopilación y procesamiento de los datos . . . 13
- 3.2. Caracterización de quimeras miRNA-mRNA . . . 15
- 3.3. Análisis de los datos . . . 17

4. Resultados **20**

5. Discusión **23**

6. Conclusiones **26**

7. Glosario **28**

8. Bibliografía **29**

9. Anexo I: Gráficos y tablas de resultados **32**

10. Anexo II: Procesamiento de los datos **43**
- 10.1. Paquetes . . . 43
- 10.2. Funciones . . . 44
- 10.3. Datos CLIP-seq . . . 50
- 10.4. Datos DIANA-TarBase . . . 57
- 10.5. Cruce de bases de datos . . . 57

11. Anexo III: Caracterización de las interacciones miRNA-diana **60**
- 11.1. Paquetes . . . 60
- 11.2. ViennaRNA . . . 60
- 11.3. Extracción de atributos . . . 62
- 11.4. Lista de atributos extraídos . . . 65

12. Anexo IV: Análisis de los datos **67**
- 12.1. Paquetes . . . 67

12.2. Funciones 68
12.3. Preparación de los datos 73
12.4. Reducción dimensional de los datos 74
12.5. CLARA 76
12.6. Expectation–Maximization (EM) 79
12.7. OPTICS 81
12.8. Selección de atributos 83

Índice de figuras

1. Diagrama de Gantt con la organización temporal del proyecto. 7
2. Principales tipos de interacción miRNA-diana descritos en la bibliografía. Algunos autores distinguen las dianas 6mer del resto de dianas canónicas. Imagen tomada de la publicación de Bartel [13]. 9
3. Diagrama de flujo con las principales fases de un análisis de análisis de *clustering*. Imagen tomada de la publicación de Xu y Wunsch [24]. 11
4. Grupos de distinta forma, tamaño y densidad identificados con OPTICS. Imagen tomada de la publicación de Ankerst *et al.* [29]. 12
5. Diagrama de flujo con los elementos que conforman las distintas fases del proyecto. 13
6. (A) Representación de una interacción miRNA-diana en formato dot-bracket y con una notación modificada. Las barras verticales '|' representan pares de bases según el modelo de interacción Watson y Crick, los signos de exclamación '!' señalan los pares de bases G-U y los corchetes '[ ]' permiten identificar pares de bases intramoleculares formando horquillas. (B) Representación gráfica del dímero miRNA-diana obtenida con el paquete ViennaRNA. 16
7. Ejemplos de motivos estructurales. (A) *Three-way junction*. (B) *Multi-way junction*. (C) No se evalúan las horquillas fuera de la zona de interacción miRNA-mRNA. 18
8. Proporción de la varianza explicada en función del número de componentes principales escogido. (A) Variables normalizadas y (B) variables estandarizadas. 19
9. Anchura media de la silueta de los agrupamientos en función de la transformación de los datos y del número de *clusters*. 21
10. *Reachability distance* con OPTICS. Ambos gráficos corresponden a un mismo conjunto de datos. Dependiendo del punto de corte escogido, el número y tamaño de los *clusters* varía. (A) Varios grupos de tamaño medio y muchos de pequeño tamaño. (B) Un grupo de gran tamaño y unos pocos de tamaño reducido. Las barras en negro se corresponden con puntos considerados como ruido. 22
11. Resultados obtenidos con CLARA en humano. (A-B) Diagramas de barras representando las proporciones de dianas funcionales y de los distintos tipos de interacción en los *clusters*. (C) Boxplot para ilustrar la distribución del MFE predicho para las interacciones de los grupos. (D) Boxplot para mostrar la distribución de la energía libre predicha cuando se utilizan secuencias extendidas de las dianas. (E-F) Gráficos de radar que condensan la información anterior. 24

12. Resultados obtenidos con CLARA en ratón. (A-B) Diagramas de barras representando las proporciones de dianas funcionales y de los distintos tipos de interacción en los *clusters*. (C) Boxplot para ilustrar la distribución del MFE predicho para las interacciones de los grupos. (D) Boxplot para mostrar la distribución de la energía libre predicha cuando se utilizan secuencias extendidas de las dianas. (E-F) Gráficos de radar que condensan la información anterior. 25
13. Detalle de la silueta de los tres mejores agrupamientos para cada especie. Cuando los puntos registran un coeficiente de la silueta negativo se debe a que probablemente estén mal clasificados. 32
14. Gráficos de puntos para visualizar la distribución de los datos en humano. (A) Funcionalidad de las interacciones. (B) Tipos de interacción. (C) MFE de los dímeros. (D) Agrupamiento obtenido con el algoritmo CLARA. 33
15. Gráficos de puntos para visualizar la distribución de los datos en ratón. (A) Funcionalidad de las interacciones. (B) Tipos de interacción. (C) MFE de los dímeros. (D) Agrupamiento obtenido con el algoritmo CLARA. 34
16. Criterio de información bayesiano (BIC) según modelo y número de *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA. . . . 35
17. Resultados obtenidos con el algoritmo EM en humano. Diagramas de barras representando las proporciones de interacciones funcionales y de los distintos tipos de dianas en los *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA. 36
18. Resultados obtenidos con el algoritmo EM en ratón. Diagramas de barras representando las proporciones de interacciones funcionales y de los distintos tipos de dianas en los *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA. 37
19. Gráfico de distancia a los k vecinos más cercanos para identificar el punto de corte del parámetro épsilon; hay que buscar el punto de inflexión de la curva. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA. . 38
20. *Reachability distance* con OPTICS. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA. 39

Índice de cuadros

1. Calendario de tareas. 6
2. Conjuntos de datos CLIP-seq recabados. 14
3. Test de la t para identificar diferencias en funcionalidad de las dianas según la variable en cuestión y medida de la magnitud del efecto con la d de Cohen (continúa en el cuadro 4). 40
4. Test de la t para identificar diferencias en funcionalidad de las dianas según la variable en cuestión y medida de la magnitud del efecto con la d de Cohen. Ver Lista de atributos extraídos, en Anexo III, para comprender la nomenclatura de las variables. 41
5. Conjunto de variables resultante tras la selección de atributos relevantes. Ver Lista de atributos extraídos, en Anexo III, para comprender la nomenclatura de las variables. 42

1. Introducción

1.1. Contexto y justificación del trabajo

Se conoce que los microRNAs (miRNAs) están conservados evolutivamente y que participan en la regulación de muchos procesos celulares, algunos de gran importancia. Las anomalías relacionadas con el funcionamiento de estas moléculas se asocian a patologías de diversa índole, como el cáncer [1]. Semejante escenario ha propiciado que el estudio de los miRNAs se haya convertido en un asunto de relevancia. No obstante, la experimentación biológica supone un elevado coste de tiempo y recursos, por lo que ha sido necesario desarrollar nuevos métodos computacionales que faciliten la labor de los investigadores. Por otra parte, las interacciones mejor descritas en la bibliografía son aquellas en las que existe una complementariedad perfecta entre los primeros nucleótidos en el extremo 5' del miRNA, región conocida como semilla, y algún punto del mRNA diana. Cuando se dan este tipo de uniones se dice que las dianas son canónicas, aunque no se trata de las únicas interacciones posibles. El empleo de la tecnología CLIP-seq ha permitido reconocer y secuenciar una cantidad considerable de quimeras miRNA-mRNA que corroboran la existencia de un número considerable de dianas no canónicas. Sin embargo, los estudios sobre la funcionalidad de estas dianas arrojan resultados dispares [2], por lo que el rol y la importancia de las mismas todavía sigue planteando interrogantes.

Este TFM aborda la problemática descrita desde un punto de vista computacional. En términos generales, la idea consiste en explotar la información relacionada con las quimeras miRNA-mRNA para caracterizar funcional, molecular y estructuralmente las interacciones entre los miRNAs y sus dianas. Para cumplir este cometido, se desarrollan una serie de procedimientos orientados a la extracción de atributos a partir de las quimeras disponibles y posteriormente se recurre a un conjunto de métodos de *machine learning*, como son los algoritmos de *clustering* y las técnicas de selección de atributos relevantes.

Más allá de la repercusión biológica de los miRNAs y de la necesidad de solventar las limitaciones metodológicas que plantea la investigación experimental, este proyecto responde a un marco de competencias asociado al Máster de Bioinformática y Bioestadística de la UOC. Cobran especial significación aquellas competencias específicas relacionadas con la comprensión de las bases de la biología molecular, la gestión de grandes volúmenes de datos, el manejo de *software* estadístico, la aplicación de métodos de aprendizaje automático y la interpretación biológica de los resultados. Aunque no de forma exclusiva, estas capacidades se relacionan fundamentalmente con las siguientes asignaturas del plan docente: Biología molecular, Herramientas informáticas para la bioinformática, Software para el análisis de datos y Machine learning.

1.2. Objetivos

Los objetivos generales del proyecto son los siguientes:

1. Construir un conjunto de datos y herramientas adecuadas a la caracterización quimeras miRNA-mRNA.
2. Identificar características frecuentes o distintivas de las dianas miRNA no canónicas.
3. Interpretar las posibles implicaciones biológicas de los resultados obtenidos.

Establecidos los objetivos de carácter general, se detallan a continuación los objetivos específicos que derivan de los anteriores:

1. Recabar y explotar datos generados a partir de distintos métodos de validación experimental que integren generación de quimeras miRNA-mRNA y secuenciación de alto rendimiento.
2. Elaborar un paquete de R orientado a la extracción de atributos a partir de quimeras miRNA-mRNA.
3. Especificar una relación de atributos que describa cada interacción miRNA-mRNA considerando aspectos moleculares, estructurales y funcionales.
4. Analizar la relación existente entre los diferentes atributos o variables mediante algoritmos de *clustering*.
5. Identificar aquellos atributos de mayor relevancia.
6. Señalar las estructuras moleculares predominantes en los distintos tipos de quimeras identificados.
7. Cuantificar la accesibilidad y estabilidad estructural de los diferentes tipos de quimeras reconocidos.

1.3. Enfoque y método a seguir

El elevado coste de la experimentación biológica y la disponibilidad de un ingente volumen de datos justifican la trascendencia de los procedimientos computacionales en el marco de la investigación científica. Atendiendo a dicho contexto, este TFM aborda la caracterización de dianas miRNA bajo un esquema de trabajo propio de un proyecto bioinformático.

En un primer paso, se consultan diversas fuentes bibliográficas relacionadas con la biología de los miRNAs y los atributos o características que normalmente se emplean para describir las interacciones miRAN-mRNA. Posteriormente se reúne una colección de datos que se agrupan según la especie y se procesa para caracterizar las quimeras anotadas. En etapas subsiguientes, tiene lugar el diseño e implementación de un protocolo para el análisis bioinformático propiamente dicho. En respuesta a los objetivos del proyecto, la naturaleza de este análisis tiene un marcado carácter exploratorio o descriptivo. Se pretende estudiar la relación entre los atributos extraídos a partir de las quimeras miRNA-mRNA e identificar aquellos de mayor importancia. Finalmente, se hace una lectura de los resultados reseñando y comparando aquellos aspectos de mayor interés en el ámbito biológico.

En favor de la transparencia y reproducibilidad de todo el proceso, la recopilación de datos queda restringida a repositorios de acceso abierto. Las quimeras sometidas a estudio fueron publicadas por [3], [4] y [5] y se encuentran en el material suplementario de sus artículos. La única excepción a esta norma la constituye *DIANA-TarBase v7.0* [6]; los datos se encuentran bajo licencia y no pueden ser compartidos con terceros aunque cualquiera puede solicitar acceso. También se prioriza el uso de *software* libre o, en su defecto, gratuito. La terminal de UNIX resulta útil en momentos puntuales del proyecto, permitiendo el manejo del paquete ViennaRNA [7] y facilitando el procesamiento de los datos generados a través del mismo paquete. Por otra parte, los métodos de *machine learning* se implementan utilizando distintos paquetes del lenguaje de programación estadística R [8], muy popular entre la comunidad científica.

1.4. Planificación con hitos y temporización

1.4.1. Tareas

Del conjunto de objetivos propuestos se desglosa la siguiente lista de tareas y subtareas asociadas:

1. Investigación bibliográfica:
    a. Adquisición de conocimientos relacionados con la estructura y función de los miRNAs.
    b. Revisión de conceptos y técnicas empleadas en *machine learning*.
2. Recopilación de datos:
    a. Búsqueda y selección de datos CLIP-seq previamente publicados.
    b. Recuperación de las secuencias nucleotídicas referentes a las quimeras miRNA-mRNA.
    c. Validación funcional de las interacciones miRNA-gen a partir de datos experimentales.
3. Caracterización de quimeras miRNA-diana:
    a. Desarrollo de un conjunto de funciones al uso que se incluyen en un paquete de R.
    b. Caracterización molecular y estructural de los dímeros miRNA-mRNA.
4. Análisis bioinformático:
    a. Diseño de un protocolo para el análisis de los datos.
    b. Implementación de algoritmos de *clustering*.
    c. Implementación de algoritmos para la selección de atributos.

5. Discusión de los resultados:
    a. Comparación de estructuras moleculares relevantes de las distintas quimeras.
    b. Comparación de la estabilidad química de las diferentes interacciones miRNA-mRNA.
6. Confección de documentos:
    a. Documentación del paquete en R.
    b. Redacción y revisión de la memoria del proyecto.
    c. Elaboración de la presentación del proyecto.

1.4.2. Calendario

En el cuadro 1 se muestra la fecha de inicio y de fin para cada tarea programada, incluyendo también los hitos marcados en el proyecto. En la figura 1 se puede consultar el diagrama de Gantt asociado a dicho calendario.

1.4.3. Hitos

Hitos marcados por el plan docente del Trabajo de Fin de Máster:

- PEC1: Se hace entrega de un plan de trabajo donde se especifica el enfoque del TFM, los objetivos y tareas necesarias para el cumplimiento de los mismos, y la organización temporal de todo el proyecto.
- PEC2: Se presenta un informe evaluando el desarrollo del proyecto y justificando los cambios organizativos que se pudieran producir. Las entregas parciales realizadas antes de la fecha de entrega servirán para verificar la exactitud del informe de evaluación.
- PEC3: Se aplica el mismo procedimiento y criterio que en la PEC2.
- PEC4: Se deposita la memoria del TFM; un documento de carácter finalista apto para ser evaluado por un tribunal. La memoria del proyecto debe atestiguar la capacidad del alumno para cumplir con las competencias del Máster.
- PEC5a: Se entrega la presentación del TFM; un documento de síntesis de carácter audiovisual. El alumno cuenta con 20 minutos de exposición para resumir su proyecto ante un tribunal evaluador.

Hitos relacionados con el cumplimiento de las tareas:

- Datos procesados: El conjunto de datos que se presenta ya incorpora información funcional miRNA-gen y las secuencias correspondientes a cada quimera miRNA-mRNA. Este hito está marcado por el cruce de bases de datos.

- Relación de atributos extraídos: Tras desarrollar el *software* necesario, cada quimera miRNA-mRNA ha sido caracterizada molecular y estructuralmente. El conjunto de datos a entregar reúne todos los elementos necesarios para ser sometido a análisis.
- Protocolo para el análisis de los datos: El documento que se deposita describe esquemáticamente cada paso del análisis, considerando los datos de entrada y de salida de cada algoritmo hasta llegar a los resultados esperados. El protocolo también incluye una relación de posibles ventajas y desventajas relacionadas con el *software* y métodos empleados.
- Resultados: Se entrega un sumario con los resultados obtenidos a partir del análisis bioinformático. En este punto también se hace entrega del paquete de funciones en R debidamente documentado de acuerdo con la política de la red de servidores CRAN (*Comprehensive R Archive Network*).

1.4.4. Análisis de riesgos

Los factores de riesgo a considerar son:

1. Disponibilidad de tiempo: De acuerdo con el plan docente de la asignatura, la elaboración del TFM debe comprender unas 375 horas.
2. Alcance del proyecto: Los objetivos y tareas que se propongan han de ser realistas, respaldando la especificidad y concreción del proyecto.
3. Organización del proyecto: Una mala gestión de los ficheros que conforman el proyecto puede conllevar errores graves y pérdida de información.
4. Incidencias no predecibles: Tanto problemas técnicos como asuntos personales.

A fin de evitar o paliar las contingencias que puedan afectar al desarrollo del proyecto, se adoptan las siguientes estrategias:

1. Mayor carga de trabajo al inicio de cada tarea: Se pretende minimizar la posibilidad de que surjan problemas de última hora que pongan en riesgo el cumplimiento del calendario estipulado.
2. Anticipación: Siempre que no comprometa la tarea actual, resulta ventajoso listar y resolver dudas relacionadas con actividades subsiguientes.
3. Días intensivos: Recurso indeseable al que recurrir en caso de que se produzcan retrasos graves, sobre todo cuando las causas sean ajenas al proyecto.
4. Empleo un sistema de control de versiones: Es importante contar con un registro histórico de los cambios que se van produciendo y poder recuperar versiones anteriores en caso necesario.

Tarea	Fecha de inicio	Fecha de fin
•PEC1 - Plan de trabajo	06-03-2018	18-03-2018
•Entrega PEC1	19-03-2018	19-03-2018
•PEC2 - Desarrollo del trabajo - Fase 1	20-03-2018	22-04-2018
•Investigación bibliográfica - miRNAs	12-03-2018	26-03-2018
•Recopilación y selección de datos CLIP-seq	19-03-2018	21-03-2018
•Recuperación de secuencias nucleotídicas	22-03-2018	26-03-2018
•Validación funcional con datos experimentales	27-03-2018	01-04-2018
•Entrega - Datos procesados	02-04-2018	02-04-2018
•Paquete de R - Funciones para caracterizar quimeras	03-04-2018	18-04-2018
•Caracterización molecular y estructural de las quimeras	19-04-2018	22-04-2018
•Entrega - Relación de atributos extraídos	23-04-2018	23-04-2018
•Entrega PEC2	23-04-2018	23-04-2018
•PEC3 - Desarrollo del trabajo - Fase 2	24-04-2018	20-05-2018
•Investigación bibliográfica - *machine learning*	09-04-2018	23-04-2018
•Diseño de un protocolo para el análisis de datos	24-04-2018	29-04-2018
•Entrega - Protocolo de análisis	30-04-2018	30-04-2018
•Implementación de algoritmos de *clustering*	01-05-2018	10-05-2018
•Implementación de algoritmos para selección de atributos	11-05-2018	15-05-2018
•Paquete R - Documentación del paquete	12-04-2018	15-05-2018
•Entrega - Resultados	16-05-2018	16-05-2018
•Comparación de estructuras moleculares relevantes	17-05-2018	20-05-2018
•Comparación de la estabilidad química de las quimeras	17-05-2018	20-05-2018
•Entrega PEC3	21-05-2018	21-05-2018
•PEC4 - Redacción de la memoria	06-03-2018	04-06-2018
•Entrega PEC4	05-06-2018	05-06-2018
•PEC5a - Elaboración de la presentación	06-06-2018	12-06-2018
•Entrega PEC5a	13-06-2018	13-06-2018
•PEC5b - Defensa pública	14-06-2018	24-06-2018
•Entrega PEC5b	25-06-2018	25-06-2018

Cuadro 1: Calendario de tareas.

1.5. Resultados esperados

Los elementos esperados al final de proyecto son:

- Plan de trabajo
- Memoria
- Sumario de resultados
- Código fuente
- Paquete de R
- Presentación virtual
- Autoevaluación del proyecto

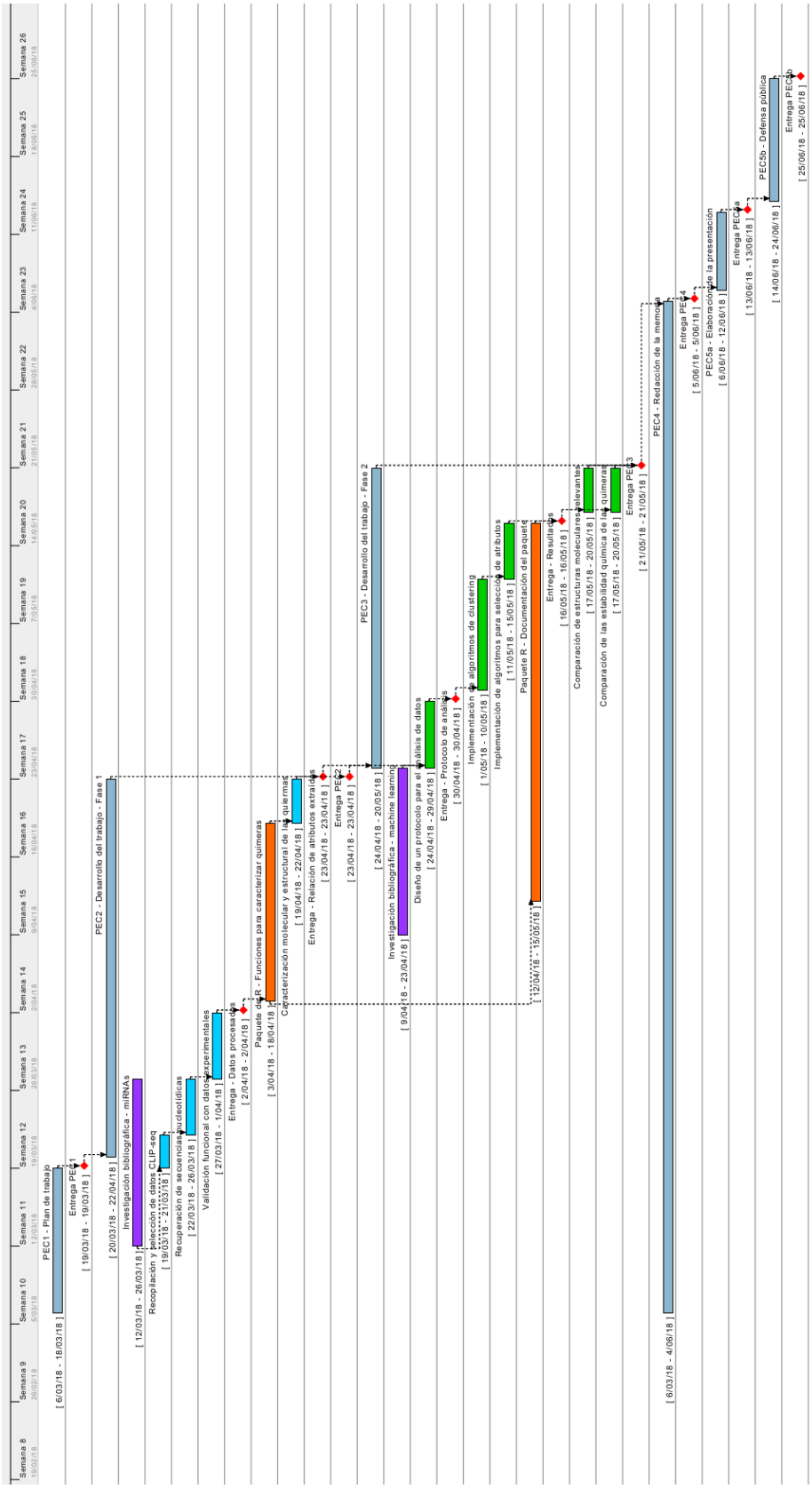

Figura 1: Diagrama de Gantt con la organización temporal del proyecto.

2. Marco teórico

2.1. Biología de los miRNAs

Los microRNAs (miRNAs) son pequeños ARNs no codificantes que intervienen en la regulación de la expresión génica inhibiendo la traducción de los mRNAs con los que interaccionan. El primer paso en la ruta de biosíntesis de estas moléculas tiene lugar en el núcleo de la célula. Los genes que codifican miRNAs son transcritos por la enzima ARN polimerasa II, generándose un miRNA primario (pri-miRNA) que consta de una caperuza y una cola de poli-A. Más tarde estas estructuras son escindidas por la acción de un complejo proteico formado por la nucleasa Drosha y la proteína de unión a RNA de doble hélice DGCR8 [9]. La escisión de la caperuza y la cola de poli-A genera un precursor de miRNA (pre-miRNA) que conserva la horquilla o *stem-loop* del pri-miRNA y que es trasportado al citoplasma tras ser reconocido por la proteína Exportina 5 [10]. Una vez en el citoplasma, el pre-miRNA se somete a la acción de otro complejo proteico, en el que interviene la ribonucleasa Dicer, dando origen a un dímero miRNA-miRNA*. Este dímero cuenta con una hebra que dará lugar al miRNA maduro y otra hebra que se conoce como hebra pasajera, normalmente designada con un asterisco [11]. El miRNA maduro interacciona con una proteína de la familia Argonauta pata terminar formando un el complejo RISC (*RNA-induced silencing complex*), responsable de guiar al miRNA hasta el mRNA diana [1]. Finalmente el mRNA es silenciado mediante la inhibición del proceso de traducción y/o la desestabilización y degradación del propio mRNA [12].

Se conoce que un mismo miRNA puede regular la expresión de distintas dianas y que a su vez un único mRNA puede estar regulado por diferentes miRNAs [11]. No obstante, todos los miRNAs presentan en su extremo 5' una región semilla (nucleótidos 2-7 o 2-8 según la publicación) responsable del reconocimiento específico las dianas; de modo que si existe una complementariedad perfecta entre las bases de dicha región y algún punto del mRNA, comúnmente en la región 3'-UTR, se dice que la diana es canónica [13]. Los casos de complementariedad total entre miRNA y mRNA resultan poco frecuentes, pero cuando se trata de una diana canónica, un simple apareamiento parcial se considera suficiente para que se produzca algún tipo de regulación [14]. Por otra parte, también puede darse el caso de que la interacción no solamente quede circunscrita al extremo 5' del miRNA. De hecho, algunos autores han encontrado evidencias de que una complementariedad adicional en la región 3' del miRNA puede intervenir mejorando el reconocimiento de las dianas o incluso compensar un emparejamiento no canónico, es decir, cuando existe algún desajuste en la región semilla [2]. En relación a las dianas no canónicas, es preciso remarcar que se ha informado de la existencia de un número importante estas interacciones y que algunas de ellas se han demostrado funcionales, si bien los mecanismos de regulación de estas dianas siguen en proceso de investigación [15].

A continuación se detallan los principales tipos de interacción miRNA-mRNA descritos en la bibliografía [13,16,17] (ver figura 2):

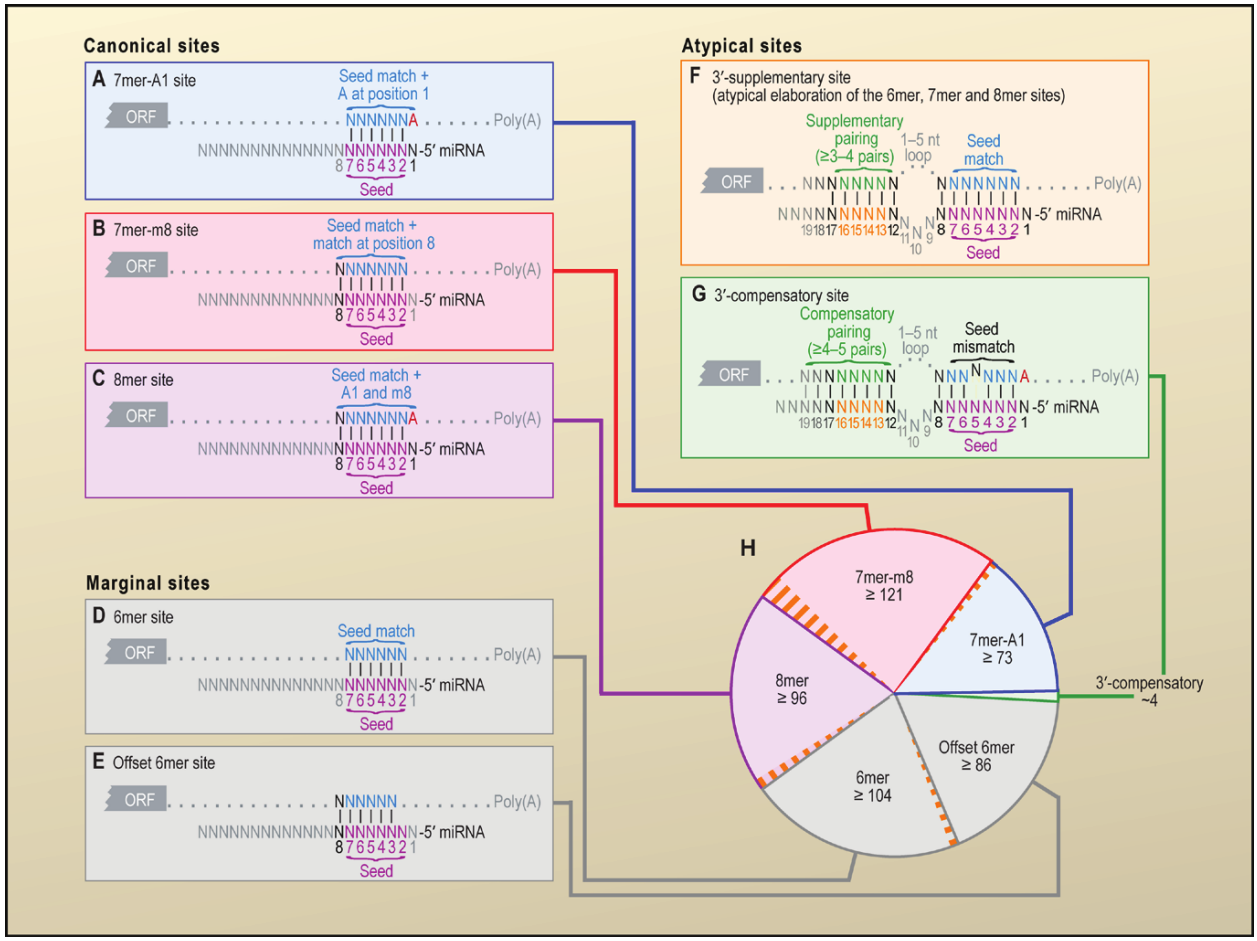

Figura 2: Principales tipos de interacción miRNA-diana descritos en la bibliografía. Algunos autores distinguen las dianas 6mer del resto de dianas canónicas. Imagen tomada de la publicación de Bartel [13].

- offset 6mer: Complementariedad Watson y Crick entre el mRNA diana y los nucleótidos 3-8 del miRNA.
- 6mer: Complementariedad Watson y Crick entre el mRNA diana y los nucleótidos 2-7 del miRNA.
- 7mer-m8: Complementariedad Watson y Crick entre el mRNA diana y los nucleótidos 2-8 del miRNA.
- 7mer-A1: Complementariedad Watson y Crick entre el mRNA diana y los nucleótidos 2-7 del miRNA, además de una adenina (A) en posición 1.
- 8mer: Complementariedad Watson y Crick entre el mRNA diana y los nucleótidos 2-8 del miRNA, además de una adenina (A) en posición 1.
- Unión suplementaria en 3': Interacciones 6mer, 7mer u 8mer seguidas de 3 o 4 pares de bases consecutivos en la región 3' del miRNA.
- Unión compensatoria en 3': Interacción no canónica, alguna base de la región semilla no está emparejada según el modelo de Watson y Crick, seguida de 4 o 5 pares de bases consecutivos en la región 3' del miRNA.
- Unión centrada: Complementariedad Watson y Crick entre el mRNA diana y las nucleótidos 4-15 o 5-16 del miRNA.

2.2. Tecnología CLIP-seq

El reducido tamaño de los miRNAs y su capacidad para interaccionar de forma parcial aunque efectiva con distintos mRNAs, son dos factores que dificultan la identificación de nuevas dianas [18]. A fin de solventar este problema, se han ido desarrollando diversos métodos computacionales dirigidos a la predicción de dianas de miRNA; *TargetScan* [19] y *miRanda* [20] son algunos de los más conocidos. Con todo, estos procedimientos no quedan exentos de inconvenientes tales como la inconsistencia de resultados al comparar distintos algoritmos y una tendencia, más o menos acusada, a generar falsos positivos [21]. El empleo de la tecnología CLIP-seq (*cross-linking immunoprecipitation with sequencing*) se plantea como una alternativa a las técnicas mencionadas anteriormente, pues permite reconocer y secuenciar los puntos de interacción entre la proteína Argonauta y los mRNAs o dianas. El principal problema asociado a este tipo de tecnología es que no permite identificar fehacientemente el miRNA asociado a la proteína Argonauta y éste ha de ser inferido por vías computacionales [15]. Sin embargo, algunos grupos de investigadores han utilizado los métodos CLIP-seq como base para desarrollar una serie de técnicas que se sirven de la generación de quimeras o híbridos miRNA-mRNA para identificar interacciones *in vivo* de forma fidedigna [3–5].

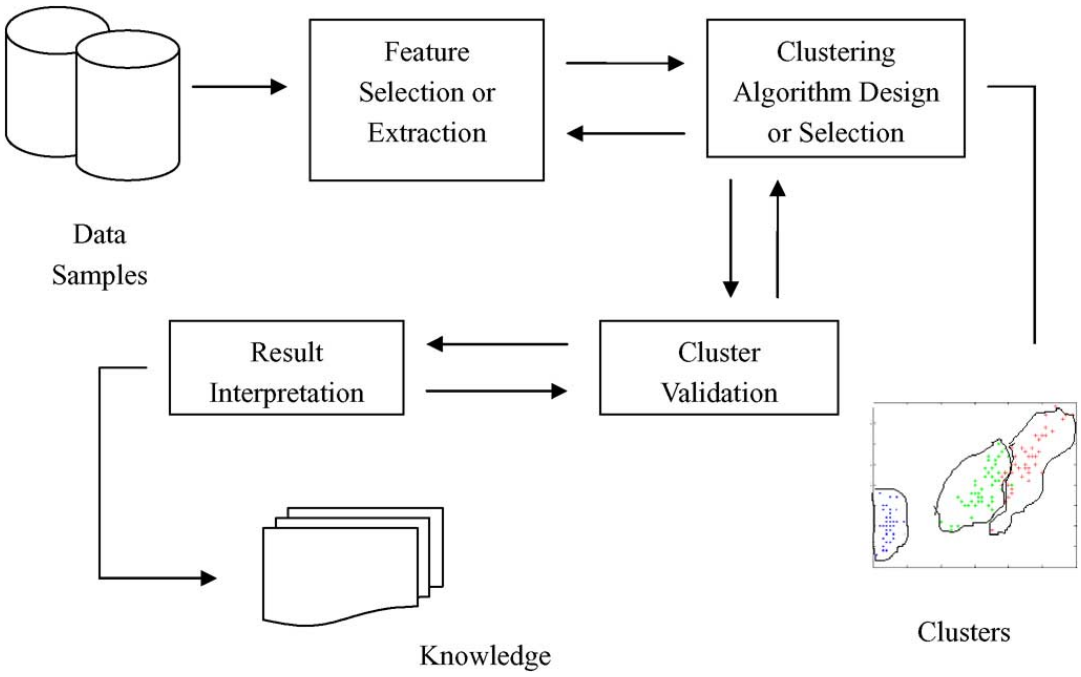

Figura 3: Diagrama de flujo con las principales fases de un análisis de análisis de *clustering*. Imagen tomada de la publicación de Xu y Wunsch [24].

2.3. Análisis de *clustering*

Los algoritmos de agrupamiento o *clustering* son una serie de técnicas enmarcadas en lo que se conoce como aprendizaje no supervisado y cuya finalidad es la de dividir un conjunto de datos sin etiquetar en una serie de grupos basándose en la similitud de las observaciones [22,23]. Hay que destacar que no existe una única definición de *cluster* y las implicaciones aparejadas a este término pueden variar en función del contexto y la naturaleza de los datos que se manejen [24]. Con todo, detrás estas definiciones siempre se repite la idea fundamental de que las observaciones de un mismo grupo deben ser tan parecidas entre sí como sea posible, mientras que las observaciones de grupos distintos deben diferir tanto como se pueda [24,25]. Del mismo modo, la similitud de los datos puede evaluarse mediante distintas métricas y funciones, como la distancia euclídea o el índice de Jaccard entre otros [25]. En cuanto a las etapas propias de un análisis de *clustering*, se pueden distinguir cuatro fases bien definidas [23] (ver figura 3): (i) extracción de atributos, (ii) implementación del algoritmo de agrupamiento, (iii) validación de los resultados e (iv) interpretación de los resultados.

La variedad de algoritmos de agrupamiento es extensa y no es la intención del proyecto abordarlos todos. Por lo tanto, solamente se detallan ciertos aspectos relacionados con algunas técnicas de *clustering* que sirvieron de base para el desarrollo de los algoritmos que finalmente se emplean en este TFM:

Algoritmos de agrupamiento por particiones: Se trata de aquellos procedimientos que se limitan a dividir los datos en varios grupos sin generar ningún tipo de estructura

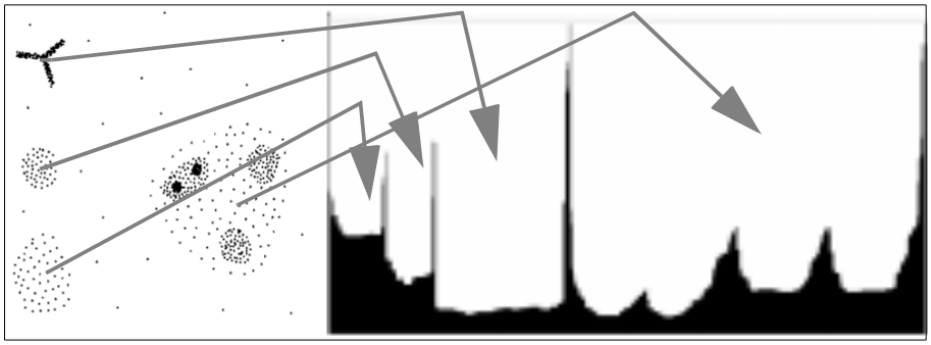

Figura 4: Grupos de distinta forma, tamaño y densidad identificados con OPTICS. Imagen tomada de la publicación de Ankerst *et al.* [29].

jerárquica [22]. Dentro de esta clase de técnicas se encuentra el algoritmo *K-means*; muy popular por a su sencillez y eficiencia para abordar grandes conjuntos de datos. Sin embargo, no todo son ventajas. Además de resultar sensible a los *outliers* o valores atípicos, los resultados obtenidos dependen de la ordenación de los datos y la selección azarosa de las particiones iniciales, pudiendo darse casos de convergencia en mínimos locales [22,26]. En respuesta a estas limitaciones surgió el algoritmo *K-medoids* o PAM (*Partitioning Around Medoids*), que permite lidiar con datos de tipo discreto y reduce la sensibilidad de los resultados frete a los *outliers* [25,26]. Así mismo, también se han desarrollado algunas técnicas encaminadas a mejorar el rendimiento cuando se analizan datos a gran escala [23]. Éste es el caso de CLARA (*Clustering Large Applications*).

Algoritmos basados en modelos: Este tipo de técnicas se basan en modelos probabilísticos que asumen que los datos observados son producto de una mezcla de grupos o componentes y que cada componente tiene su propia distribución de probabilidad [27]. El algoritmo EM (*Expectation-Maximization*) es uno de los procedimientos más empleados para estimar los parámetros de este tipo de modelos gracias a su simpleza, versatilidad y fácil implementación [28].

Algoritmos basados en densidad: Esta clase de procedimientos evalúa la densidad de los datos dentro del espacio multidimensional que los contiene, de tal forma que los puntos u observaciones presentes en una región de alta densidad forman un mismo *cluster* [25]. En oposición a los algoritmos de *clustering* por particiones, la gran ventaja de esta clase de técnicas reside en su capacidad para reconocer grupos de cualquier forma geométrica, al mismo tiempo que permiten la identificación de *outliers* o puntos considerados como ruido [24,26]. DBSCAN (*Density-based spatial clustering of applications with noise*) es uno de los algoritmos más conocidos dentro de este grupo, aunque tiene los inconvenientes de ser muy sensible a la elección de los parámetros y de obtener pobres resultados cuando la densidad de los datos es heterogénea [25]. Por su parte, el algoritmo de *clustering* OPTICS (*Ordering points to identify the clustering structure*) se plantea como una solución a estos problemas. Esta técnica de agrupamiento produce una ordenación de los datos que permite identificar la estructura de grupos con diferente forma, tamaño y densidad [29] (ver figura 4).

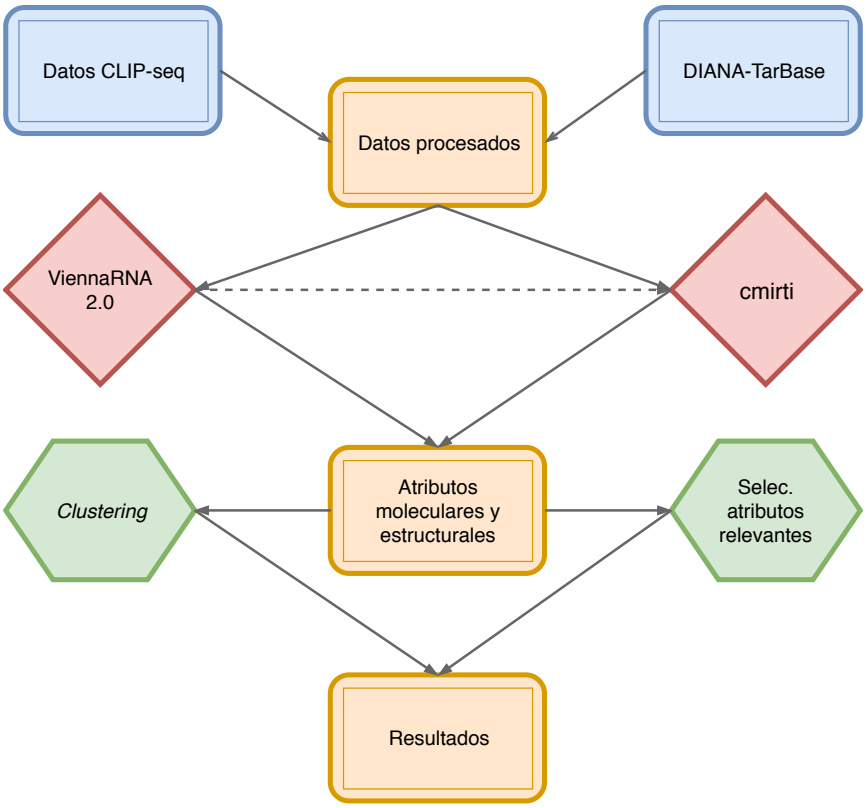

Figura 5: Diagrama de flujo con los elementos que conforman las distintas fases del proyecto.

3. Material y métodos

El esquema de trabajo del proyecto consta de tres grandes fases (ver figura 5). La etapa inicial del proyecto está dedicada a la recopilación y procesamiento de datos CLIP-seq y a la validación funcional de las interacciones miRNA-mRNA con la información contenida en el repositorio de *DIANA-TarBase v7.0*. Una vez se dispone de los datos procesados, el siguiente paso del plan de trabajo se centra en la caracterización de las interacciones miRNA-diana mediante la extracción de atributos de tipo molecular y estructural. A tal efecto, se utiliza el paquete *software* `ViennaRNA 2.0` y se desarrolla otro paquete de funciones para R, de nombre `cmirti`. Finalmente se procede a analizar la relación entre las distintas variables descriptivas que se han obtenido en la fase anterior mediante algoritmos de *clustering*. De forma secundaria también se lleva a cabo una selección de atributos relevantes mediante métodos de filtrado.

3.1. Recopilación y procesamiento de los datos

La primera fase del proyecto se centró en la búsqueda y selección de datos CLIP-seq obtenidos a partir de ensayos en humano (*Homo sapiens*) [3–5] y ratón (*Mus musculus*) [4,5] (ver cuadro 2). Dependiendo de la publicación, los conjuntos de datos recabados

Especie	Método	Genoma	Publicación
Homo sapiens	CLEAR-CLIP	NCBI36/hg18	Moore *et al.* [5]
Homo sapiens	iPAR-CLIP	GRCh37/hg19	Grosswendt *et al.* [4]
Homo sapiens	CLASH	GRCh37/hg19	Helwak *et al.* [3]
Mus musculus	CLEAR-CLIP	NCBI37/mm9	Moore *et al.* [5]
Mus musculus	iPAR-CLIP	NCBI37/mm9	Grosswendt *et al.* [4]

Cuadro 2: Conjuntos de datos CLIP-seq recabados.

incorporan distintos detalles relacionados con las quimeras miRNA-mRNA, aunque todos reúnen una serie de especificaciones comunes. Estas especificaciones incluyen la localización cromosómica donde se produce cada interacción (cromosoma, posición de inicio, posición de fin y hebra) y el identificador miRBase del miRNA involucrado en dicha interacción. Partiendo de esa información se pueden recuperar las secuencias nucleotídicas de cada quimera y los identificadores Entrez (*Entrez Gene ID*) y Ensembl (*Ensembl Gene ID*) del gen o genes asociados a una determinada región cromosómica. A modo de excepción, en el conjunto de datos publicado por Helwak *et al.* [3] no se detalla la localización cromosómica de las interacciones sino que cada quimera queda definida por un miRNA y un transcrito debidamente identificado. A fin de validar funcionalmente las quimeras disponibles, durante esta fase inicial también se procedió a descargar la base de datos *DIANA-TarBase v7.0*, pues registra varios cientos de miles de interacciones miRNA-gen validadas experimentalmente. Llegado este punto ya se contaba con todos los elementos necesarios para iniciar el procesamiento de los datos (ver Anexo II).

Con el fin de recuperar las secuencias nucleotídicas de los mRNAs se emplearon distintos paquetes en función de la especie y versión del genoma en cuestión; todos ellos dependientes del paquete `BSgenome` [30]. En aquellos casos en los que no se contaba con ningún identificador para los genes, se utilizaron los paquetes `GenomicFeatures`, `GenomicRanges` [31] y varias librerías `TxDb`, como `TxDb.Hsapiens.UCSC.hg19.knownGene`, para obtener dichos identificadores; normalmente identificadores Entrez. Cuando ya se contaba con algún tipo de identificador, el paquete `biomaRt` [32] sirvió para realizar conversiones entre identificadores de distinta clase. En lo referente a los miRNAs, cada conjunto de datos contenía un identificador miRBase para cada registro o quimera. Sin embargo, puede darse el caso de que los identificadores para un mismo miRNA varíen según la versión de miRBase empleada. El paquete `miRNAmeConverter` [33] permitió determinar a qué versión pertenecía cada conjunto de identificadores según el caso. Durante esta etapa del proyecto surgieron algunas contradicciones entre la versión señalada por los autores de los datos y la versión indicada por el *software* mencionado. Fue necesario entonces realizar algunas comprobaciones manuales que evidenciaron la corrección de las propuestas emitidas por el programa. Para cada miRNA se extrajeron las secuencias nucleotídicas, siempre que no estuvieran presentes en los datos de partida; el código MIMAT y el identificador miRBase correspondiente a la versión 21. En marzo de 2018 se lanzó la versión 22.

Antes de cruzar los datos CLIP-seq con la base de datos DIANA-TarBase, fue necesario aplicar una serie de procedimientos sobre ésta última. Primeramente se dividió el conjunto de datos en función de la especie y se seleccionaron los campos o variables a extraer

(funcionalidad, tipo de regulación, gen y miRNA). Los miRNAs anotados en DIANA-TarBase ya contaban con un identificador miRBase de acuerdo a la versión 21. Luego se trató cada subconjunto de datos con el objetivo de descartar todos aquellos registros que ofrecieran resultados contradictorios en cuanto a la funcionalidad y tipo de regulación de las interacciones miRNA-gen. Finalmente se cruzaron las bases de datos empleando el paquete `dplyr` [34].

3.2. Caracterización de quimeras miRNA-mRNA

Tras la recopilación y el procesamiento de los datos CLIP-seq, la siguiente etapa del proyecto se dedicó a la caracterización de las quimeras mediante la extracción de atributos moleculares y estructurales (ver Anexo III). Si bien era posible calcular de forma directa algunos atributos relacionados con las secuencias nucleotídicas de las quimeras, resultó imprescindible hacer uso de esas mismas secuencias para inferir la estructura secundaria resultante de la interacción entre un miRNA dado y su diana. Determinadas características de tipo estructural solamente podían extraerse a partir de dicha estructura. El programa `RNAcofold`, integrado en el paquete `ViennaRNA 2.0`, se empleó con el fin de predecir la estructura secundaria de cada quimera, representada en formato *dot-bracket*, y la estabilidad de la unión en términos de energía libre ΔG y MFE (*Minimum free energy*). Utilizando el programa `RNAup`, incluido en el mismo paquete de *software*, se estudió la accesibilidad estructural a la diana, descrita por los puntos de inicio y fin de la interacción y la energía libre total de la unión. Es preciso señalar que para el análisis de la accesibilidad estructural se manejaron secuencias extendidas de las dianas (100 nucleótidos aguas arriba y 100 nucleótidos aguas abajo) y que la energía libre de la unión se calculó considerando una penalización por la apertura del sitio de la interacción en el mRNA.

Si bien el paquete `ViennaRNA 2.0` permitió la predicción de la estructuras secundarias de los dímeros miRNA-mRNA y su caracterización desde el punto de vista de la accesibilidad estructural y estabilidad de las interacciones, la mayor parte de los atributos se calcularon utilizando un conjunto de funciones propias. Estas funciones se incluyen en un paquete de R desarrollado durante el proyecto y bautizado como `cmirti` (*Characterization of MiRNA-Target Interactions*). Se prevé iniciar el proceso de envío a los repositorios CRAN al término del TFM y tras revisar exhaustivamente que se cumplen todas las condiciones necesarias para tal fin. En cualquier caso, a la memoria del proyecto se adjunta una carpeta de archivos que recoge el código, tal cual se usó en el TFM, y la documentación del paquete.

El primer cometido llevado a cabo con el paquete `cmirti` consistió en transformar las notaciones *dot-bracket* en otro sistema de símbolos que permitiera reconocer con mayor facilidad algunos elementos tales como interacciones intramoleculares o pares de bases G-U (ver figura 6). Posteriormente se realizó una clasificación de las interacciones miRNA-diana considerando un total de 18 tipos de uniones que pueden agruparse de la siguiente manera: canónicas (offset-6mer, 6mer, 7mer-A1, 7mer-m8 y 8mer), canónicas con unión suplementaria en la región 3' (offset-6mer-supp, . . . , 8mer-supp), no canónicas centradas

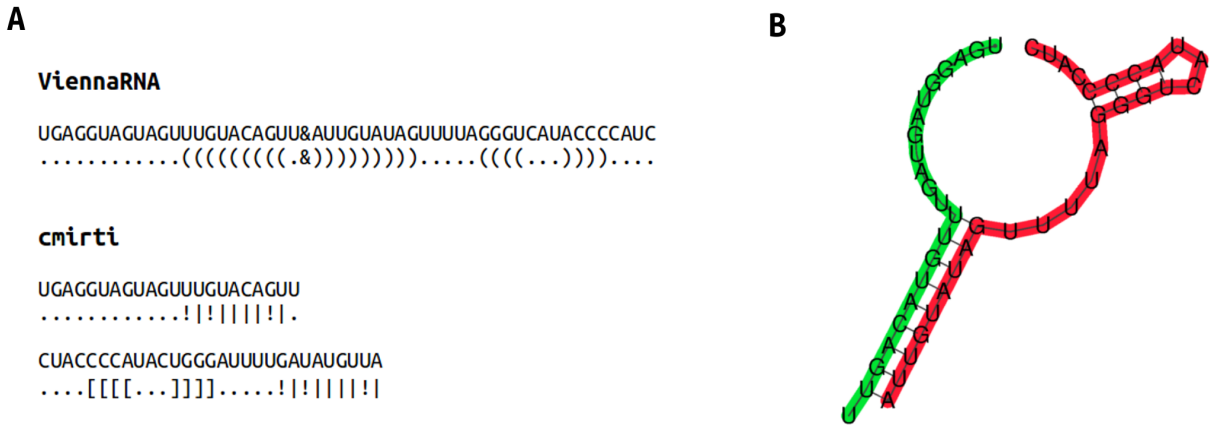

Figura 6: (A) Representación de una interacción miRNA-diana en formato dot-bracket y con una notación modificada. Las barras verticales '|' representan pares de bases según el modelo de interacción Watson y Crick, los signos de exclamación '!' señalan los pares de bases G-U y los corchetes '[ ]' permiten identificar pares de bases intramoleculares formando horquillas. (B) Representación gráfica del dímero miRNA-diana obtenida con el paquete ViennaRNA.

(NonCa-centred), no canónicas con unión compensatoria en la región 3' (NonCa-comp), y no canónicas; en este último caso se diferencian varias clases dependiendo del número de bases no apareadas o *mismatches* en la región semilla (NonCa-1mism, . . . , NonCa-6mism). Durante esta fase de clasificación se descubrieron algunos casos en los que no había interacción entre los RNAs de las quimeras y otros en los que la interacción consistía en un único par de bases con apareamiento Watson y Crick. En ambas ocasiones se optó por eliminar dichos registros con vistas a los análisis ulteriores, pues determinados atributos numéricos no se pueden calcular de forma correcta cuando se dan estas situaciones. Finalmente, se procedió a la extracción de un conjunto de atributos que fundamentalmente se relacionan con tres aspectos diferentes: (i) la cantidad, tipo y disposición de los pares de bases entre miRNA y diana, (ii) la cantidad y tamaño de distintos motivos en la estructura secundaria de los dímeros y (iii) la composición nucleotídica de los miRNAs.

En relación a los pares de bases que intervienen en la formación de los dímeros, se emplearon los miRNAs de las quimeras con la finalidad de realizar las siguientes mediciones:

- Número de pares de bases, número específico de pares de bases Watson y Crick y número de pares G-U o *wobbles*; tanto en conjunto como dentro y fuera de la región semilla.
- Número de pares de bases que interaccionan intramolecularmente formando horquillas.
- Posiciones del primer y último par de bases.
- Distancia mínima, media y máxima entre pares de bases correlativos.
- Número de regiones de pares de bases consecutivos; cada región está separada

por una o varias bases no apareadas.

- Longitud mínima, media y máxima de las regiones de pares de bases consecutivos. La longitud máxima de las regiones también se calculó de forma específica dentro y fuera de la región semilla.

Las mediciones de los cuatro último puntos se realizaron por duplicado y se registraron en distintas variables en función de si se consideraba que los pares G-U participan en la interacción o si, por el contrario, solamente se evaluaban los pares de bases de tipo Watson y Crick. Parte de estos cálculos se efectuaron además con las dianas o mRNAs de las interacciones, aunque en esta ocasión no se hizo distinción alguna entre pares G-U y bases apareadas según el modelo de Watson y Crick.

Entre los motivos estructurales que el paquete `cmirti` es capaz de reconocer se encuentran los siguientes elementos: (i) protuberancia o *bulge*, cuando en una de las cadenas existe una base desapareada flanqueada por dos bases que interaccionan con la otra cadena; (ii) bucle o *loop* simétrico, cuando entre ambas cadenas se enfrentan el mismo número de bases desapareadas consecutivas estando flanqueadas por bases que interaccionan con la otra cadena; y (iii) bucle o *loop* asimétrico, igual que el bucle simétrico pero con la particularidad de que el número de bases desapareadas es diferente en cada cadena. Para cada una de estos elementos se evaluaron los mismos atributos: número de motivos en el dímero, longitud o tamaño del motivo más pequeño, longitud o tamaño del motivo más grande, longitud media de los motivos y longitud acumulada o número total de nucleótidos que participan en motivos de una misma clase. Cuando se trataba de protuberancias y bucles asimétricos, las mediciones se realizaron en el miRNA y en la diana por separado. Otras configuraciones evaluadas son las uniones de tres o más brazos, *three-way junctions* o *multi-way junctions* respectivamente. La primera estructura consiste en una horquilla que puede estar o no incluida en un bucle pero que siempre se encuentra flanqueada por pares de bases que participan en la interacción miRNA-mRNA, mientras que la segunda estructura se identifica con un bucle que contiene al menos dos horquillas (ver figura 7). En estos casos más complejos solamente se registró el número de veces que aparecía cada motivo en la estructura secundaria del dímero.

En cuanto a los atributos relacionados con las secuencias nucleotídicas de los miRNA, se calculó el porcentaje de nucleótidos y dinucleótidos presentes en las cadenas, así como el contenido GC, contenido AU y contenido de bases púricas y pirimidínicas. Al final del proceso de extracción de atributos se contaba con un total de 96 variables de tipo numérico (ver Lista de atributos extraídos en Anexo III).

3.3. Análisis de los datos

Dado el elevado número de mediciones o variables numéricas anotadas durante la fase de caracterización de las quimeras, fue necesario implementar una serie de procedimientos de reducción dimensional de forma previa a los análisis. Concretamente se emplearon dos técnicas diferentes: análisis de componentes principales o PCA y análisis de componentes

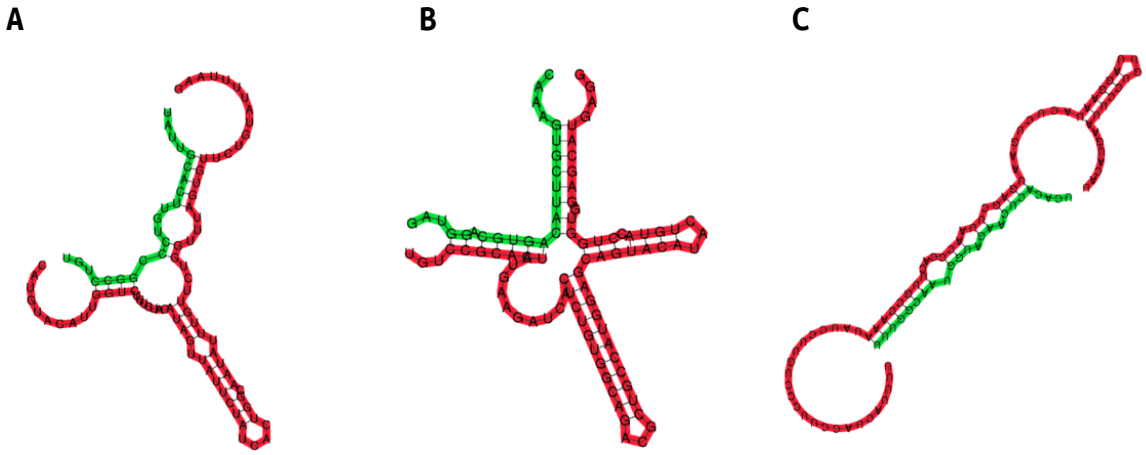

Figura 7: Ejemplos de motivos estructurales. (A) *Three-way junction*. (B) *Multi-way junction*. (C) No se evalúan las horquillas fuera de la zona de interacción miRNA-mRNA.

independientes o ICA. A su vez, cada técnica se aplicó sobre dos conjuntos de datos obtenidos a partir de distintas transformaciones de las variables numéricas originales. En una ocasión las variables fueron normalizadas o reescaladas en el rango $[0, 1]$, mientras que en la otra se llevó a cabo una estandarización (*z-score*) de los atributos. Primeramente se realizó un PCA con cada conjunto de datos y en cada caso se tomaron los primeros componentes principales que explicaban el $95\,\%$ de la varianza total (ver figura 8). Más tarde, se aplicó un ICA a los mismos conjuntos de datos, considerando en cada ocasión el mismo número de componentes escogidos tras realizar el PCA.

A fin de analizar la relación entre las distintas variables o atributos de las interacciones miRNA-diana se utilizaron tres algoritmos de *clustering*: CLARA, EM y OPTICS. Otros algoritmos más clásicos, como las técnicas de *clustering* jerárquico, no pudieron utilizarse debido a restricciones computacionales. Para llevar a cabo la implementación el algoritmo CLARA se hizo uso del paquete de R `cluster` [35] y se evaluaron diferentes valores del parámetro k, que determina el número de grupos o *clusters*. La selección del mejor agrupamiento se realizó en base al coeficiente de la silueta, un índice de validación interna. Por otro lado, el paquete `mclust` [36] permitió emplear el algoritmo EM con el propósito de ajustar varios modelos de mixturas gaussianas. En esta ocasión, el propio programa se encargó de determinar el número de componentes y modelo óptimo utilizando el criterio de información bayesiano o BIC. Finalmente, se manejó el paquete `dbscan` [37] para poner en funcionamiento el algoritmo OPTICS. Aunque no resultaba estrictamente necesario, en cada caso se estimó la distancia media entre cada punto y sus 50 vecinos más cercanos y posteriormente se representó en orden ascendente. Este tipo de gráficos permite identificar una región en la que se encuentra el valor óptimo del parámetro ϵ o épsilon, que describe el radio de la vecindad de los puntos (ver figura 19 en Anexo I).

La última fase de los análisis se dedicó a la selección de atributos relevantes para la predicción de la funcionalidad de las interacciones miRNA-mRNA. Primeramente se

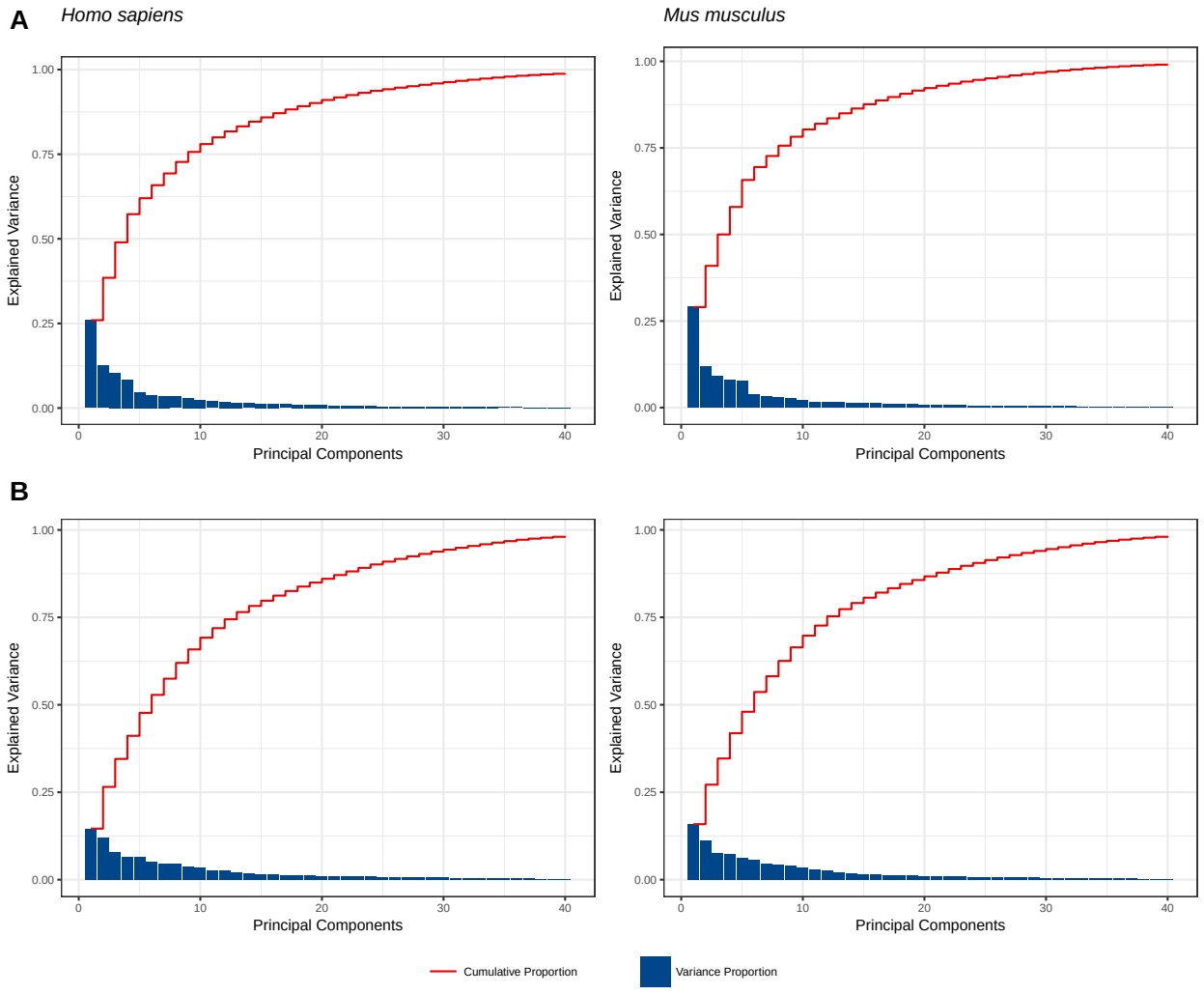

Figura 8: Proporción de la varianza explicada en función del número de componentes principales escogido. (A) Variables normalizadas y (B) variables estandarizadas.

realizó una prueba *t* de Student con cada variable para determinar si existían diferencias significativas entre los registros que se sabían funcionales y aquellos cuya funcionalidad era desconocida. Sin embargo, cuando las muestras son muy grandes es frecuente encontrar diferencias estadísticamente significativas incluso cuando la magnitud del efecto resulta despreciable [38]. Por este motivo, de forma paralela a la prueba *t* de Student, se midió el tamaño del efecto con la *d* de Cohen. De forma posterior, se llevó a cabo la selección de atributos propiamente dicha con el paquete `caret` [39]. Como consecuencia del elevado número de variables, se seleccionó un método de filtrado univariante con modelos aditivos generalizados o GAM seguido del ajuste de un modelo *random forest*. Uno de los problemas asociados a los métodos de filtrado radica en que no se evalúa la posible relación entre las distintas variables, por lo que pueden seleccionarse variables redundantes [40]. A fin de paliar este inconveniente, se descartaron algunas variables altamente correlacionadas (coeficiente de correlación mayor a $0,9$) antes de implementar la selección de atributos por filtrado.

4. Resultados

En relación al algoritmo de *clustering* CLARA, se comprobó que tanto en humano como en ratón, los mejores agrupamientos correspondían a los datos obtenidos como producto de la normalización y posterior reducción dimensional de las variables utilizando un análisis de componentes principales (ver figura 9). También se pudo advertir que en ambas especies la mayor anchura media de la silueta se registró al adoptar cinco grupos o *clusters* ($k = 5$), habiendo descartado la opción de k igual a dos. Sin embargo, dada la complejidad de los datos, se optó por escoger valores de k más altos ($k = 8$ en humano y $k = 6$ en ratón), puesto que el valor medio de la silueta de estos agrupamientos no era mucho menor que el máximo registrado en cada especie (ver figura 13 en Anexo I). Así pues, se fijó en ocho y en seis el número de grupos seleccionados en humano y ratón respectivamente. Resulta conveniente aclarar que en ningún caso se anotó un coeficiente de la silueta realmente alto; el motivo seguramente tenga que ver con la disposición aglomerada de los puntos en el espacio multidimensional de las variables (ver figuras 14 y 15 en Anexo I).

De los ocho grupos establecidos en humano, en cuatro de ellos la proporción de dianas que se saben funcionales ronda el $25\,\%$, mientras que en los cuatro restantes este porcentaje se ve reducido aproximadamente a la mitad. En ratón, donde no se observa una polarización tan clara, las proporciones oscilan entre el $12\,\%$ y el $19\,\%$ según el grupo (ver figuras 11-A y 12-A). Respecto a los tipos de interacción, es importante resaltar el hecho de que en ambas especies puede identificarse un patrón de distribución similar (ver figuras 11-[B, F] y 12-[B, F]). Existen dos grupos en humano (grupos 1 y 2) y uno en ratón (grupo 2) que destacan por una alta proporción de dianas de tipo canónico con unión suplementaria en la región 3' del miRNA. Al mismo tiempo, estos grupos son los que acumulan el porcentaje más pequeño de interacciones no canónicas y en humano son aquellos que exhiben la mayor proporción de dianas funcionales. Además, las dianas no funcionales que se encuentran en estos *clusters* solamente presentan una o dos bases

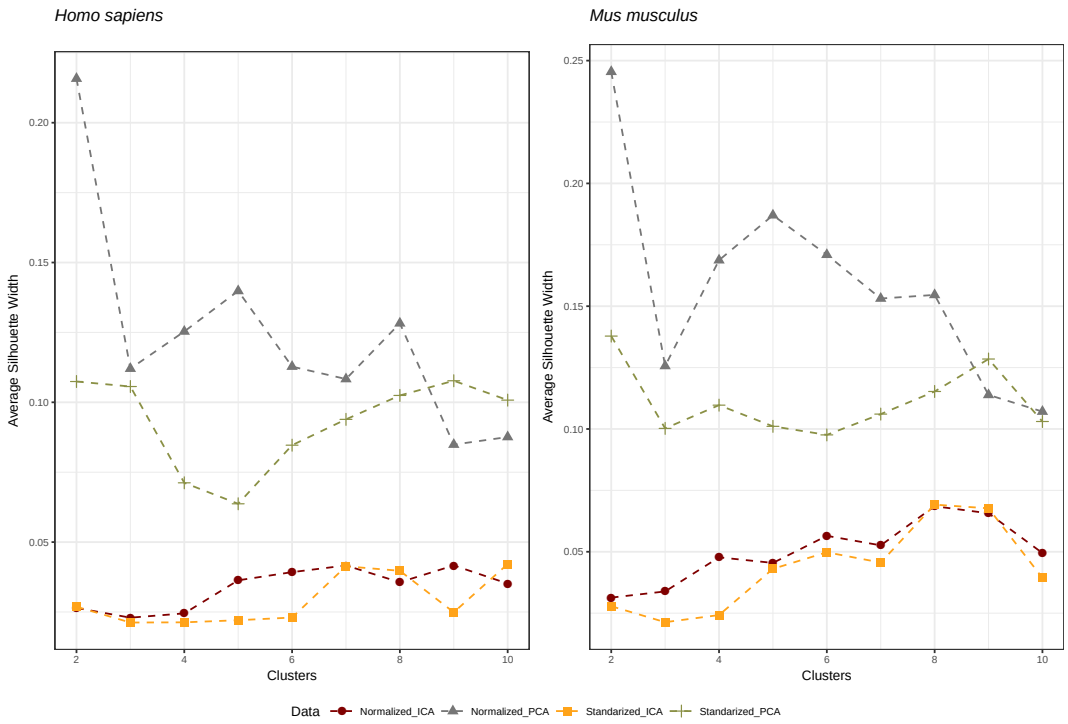

Figura 9: Anchura media de la silueta de los agrupamientos en función de la transformación de los datos y del número de *clusters*.

desapareadas en la región semilla. Por otra parte, en cada especie se reconoce un grupo (grupo 3 en humano y grupo 1 en ratón) caracterizado por el predominio, en relación a los demás grupos, de las dianas de tipo canónico sin unión suplementaria en la región 3' del miRNA; aunque paradójicamente el mismo *cluster* esta formado mayoritariamente por una amalgama de interacciones no canónicas. También resulta sencillo advertir que en ambas especies se halla un grupo (grupo 6 en humano y grupo 5 en ratón) que aglutina casi todos los casos en los que la región semilla del miRNA no interviene en la interacción miRNA-diana, es decir, cuando la diana es no canónica con seis bases desapareadas en la región semilla. El resto se trata de grupos mestizos, pues incluyen todo tipo de interacciones, en los que predominan las dianas no canónicas. Por otro lado, las dianas no canónicas con unión compensatoria en la región 3' del miRNA se distribuyen de forma más o menos homogénea en todos los *clusters*, a excepción de los grupos 3 y 6 en humano y 1 y 5 en ratón; mientras que las dianas no canónicas con unión centrada resultan escasas y se concentran en los grupos 1 y 8 en el caso de humano y en el grupo 2 en el caso de ratón. En lo referente a la estabilidad química de las interacciones, no se aprecia una gran diferencia entre grupos (ver figuras 11-[C, D] y 12-[C, D]). Sin embargo, llama la atención que en las dos especies el grupo más desfavorecido desde un punto de vista termodinámico sea aquel que acumula una mayor proporción de dianas canónicas (grupo 3 en humano y grupo 1 en ratón). Asimismo, el segundo grupo con peor estabilidad de las interacciones es aquel que reúne las dianas no canónicas sin interacción en la región semilla (grupo 6 en humano y grupo 5 en ratón).

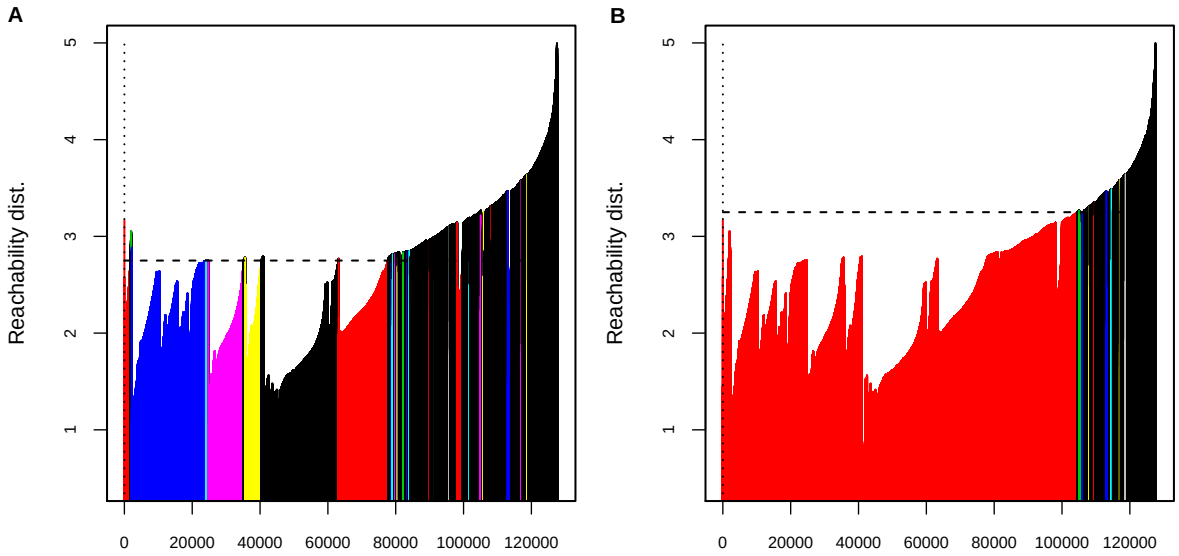

Figura 10: *Reachability distance* con OPTICS. Ambos gráficos corresponden a un mismo conjunto de datos. Dependiendo del punto de corte escogido, el número y tamaño de los *clusters* varía. (A) Varios grupos de tamaño medio y muchos de pequeño tamaño. (B) Un grupo de gran tamaño y unos pocos de tamaño reducido. Las barras en negro se corresponden con puntos considerados como ruido.

En cuanto al algoritmo EM, en casi todos los casos, con independencia del organismo y transformación de las variables, el número óptimo de grupos según el BIC fue igual al máximo evaluado, 10 *clusters* (ver figura 16 en Anexo I). Del mismo modo, los resultados obtenidos mediante la implementación de este algoritmo fueron siempre muy parecidos. Se trata de grupos bastante homogéneos desde el punto de vista de la funcionalidad y del tipo de las interacciones, lo que dificulta gravemente la extracción de conocimiento a partir de los datos (ver figuras 17 y 18 en Anexo I). Tampoco resultaron satisfactorios los resultados alcanzados tras la aplicación del algoritmo OPTICS. En esta ocasión, no fue posible encontrar un punto de corte para el parámetro épsilon que permitiera obtener un agrupamiento razonable dada la naturaleza de los datos y del problema a tratar (ver figura 20 en Anexo I). Aunque en el mejor de los casos se podían diferenciar algunos grupos de tamaño aceptable, la suma de puntos considerados como ruido y la cantidad de pequeños *clusters* seguía siendo demasiado alta (ver figura 10).

Respecto a la distribución de los atributos de las quimeras según la funcionalidad de las interacciones, se encontraron diferencias significativas en la mayor parte de las variables. No obstante, la magnitud del efecto resultó casi siempre despreciable; solamente en humano se anotaron algunas variables en las que el tamaño del efecto podía clasificarse como pequeño (ver cuadros 3 y 4 en Anexo I). Por otra parte, la selección de atributos relevantes mediante métodos de filtrado se saldó con un total de 55 variables escogidas en humano y otras 56 variables en ratón (ver cuadro 5 en Anexo I). Sin embargo, la escasa capacidad predictiva de los modelos *random forest*, ajustados con los atributos

seleccionados, pone en tela de juicio la relevancia de estas variables para la funcionalidad de las interacciones miRNA-diana. La alta precisión de los modelos (humano: $0,81$, ratón: $0,84$) solamente se explica porque una de las clases es mucho más numerosa que la otra. Concretamente, el número de interacciones de funcionalidad desconocida es mayor que el número de dianas funcionales. Por el contrario, el índice kappa, que evalúa la precisión del modelo considerando la posibilidad de acierto por azar [41], mostró valores muy bajos (humano: $0,18$, ratón: $0,05$), especialmente en ratón. Es necesario matizar que la creación de un modelo de predicción queda fuera del ámbito del proyecto y que el ajuste de un clasificador con *random forest* solamente forma parte del método de selección de atributos empleado.

5. Discusión

Si bien los resultados obtenidos no pueden considerarse concluyentes, es posible apreciar algunos patrones bastante consistentes en las dos especies sometidas a estudio. Por ejemplo, la distribución del tipo de las interacciones sigue una pauta robusta en ambos organismos y la estabilidad química de las dianas también guarda similitudes entre humano y ratón. Sin embargo, y a pesar de las coincidencias, existen inconvenientes notables. Uno de los principales tiene que ver con que los patrones observados no destacan por arrojar nueva luz sobre las interacciones no canónicas, pues la mayor parte de las clases definidas se reparte de forma parecida a través de los diferentes *clusters*. Estos patrones sirven más bien para reafirmar el conocimiento establecido: la mayor parte de las dianas funcionales son canónicas y la complementariedad de bases en la región 3' del miRNA mejora la estabilidad y la efectividad de las interacciones miRNA-mRNA. Otra de las limitaciones que han marcado el proyecto, quizá sea la que más impedimentos genere a la hora de extraer conclusiones a partir de los datos, se corresponde con la práctica ausencia de dianas cuya funcionalidad se haya probado negativa. Aunque se recabó un número considerable de interacciones miRNA-gen funcionales, se desconoce el rol biológico de la mayor parte de las interacciones miRNA-diana que fueron sometidas a análisis. Asimismo, como consecuencia de las restricciones temporales del TFM, durante la fase de caracterización de las quimeras tampoco se pudo evaluar el grado de conservación evolutiva de los miRNAs involucrados en las mismas. Este último aspecto es importante ya que se trata de una cualidad no necesaria pero común para considerar un miRNA como tal [42], además de que el grado de conservación de la región semilla se considera como uno de los mejores indicadores de la funcionalidad de los miRNAs [15].

Al mismo tiempo, es necesario tener en cuenta la eventualidad de los resultados como fruto de la elección de un determinado *software*. Aunque cada quimera reconoce inequívocamente a los RNAs involucrados en la interacción, se precisa el uso de un programa para predecir la estructura del dímero, de forma que la posterior extracción de atributos queda enteramente condicionada por dicha predicción. Si en lugar de utilizar el paquete ViennaRNA se manejara un programa diferente, como RNAhybrid, los resultados podrían presentar ciertas diferencias. Esto supone una doble limitación metodológica a la hora

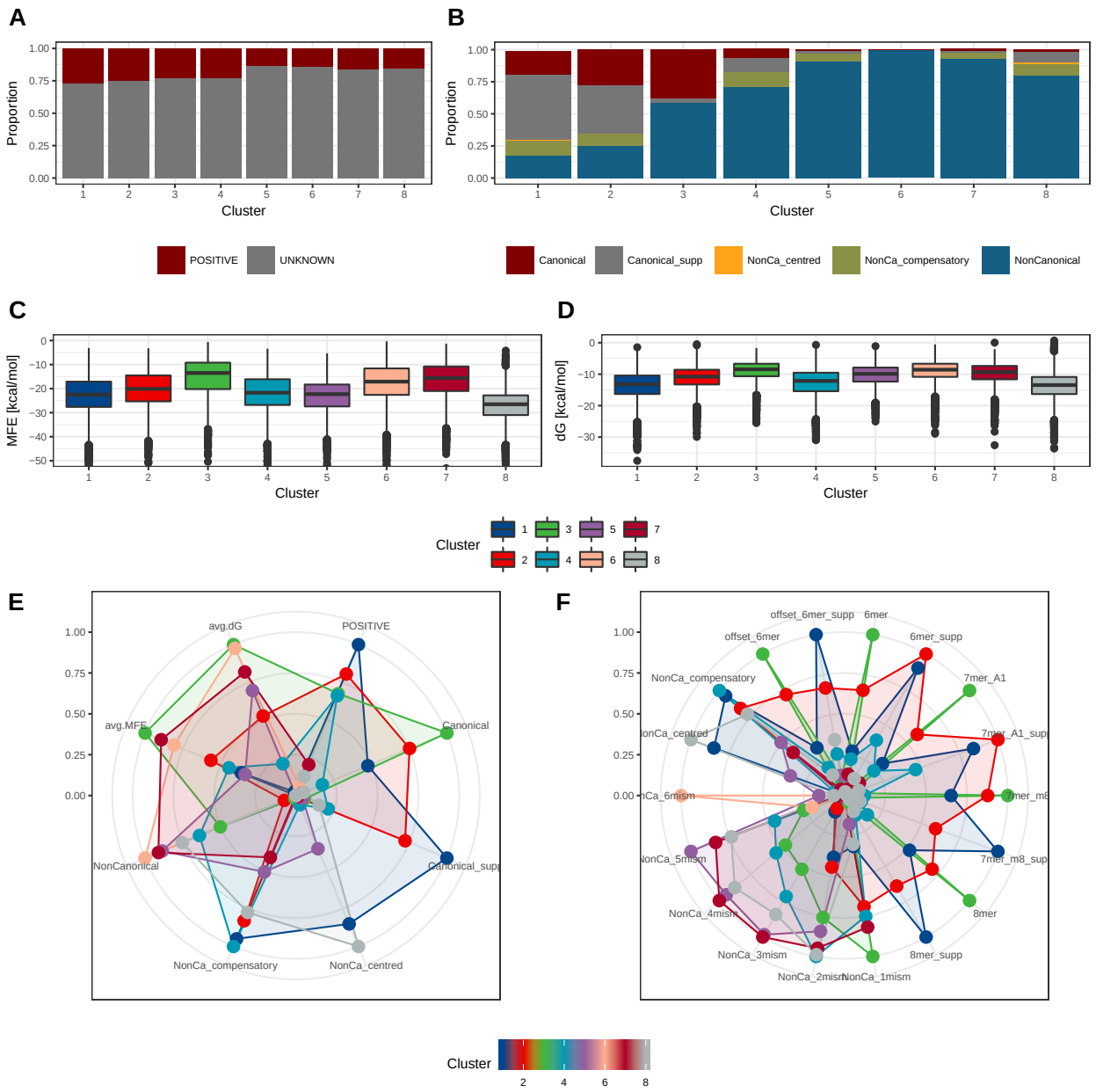

Figura 11: Resultados obtenidos con CLARA en humano. (A-B) Diagramas de barras representando las proporciones de dianas funcionales y de los distintos tipos de interacción en los *clusters*. (C) Boxplot para ilustrar la distribución del MFE predicho para las interacciones de los grupos. (D) Boxplot para mostrar la distribución de la energía libre predicha cuando se utilizan secuencias extendidas de las dianas. (E-F) Gráficos de radar que condensan la información anterior.

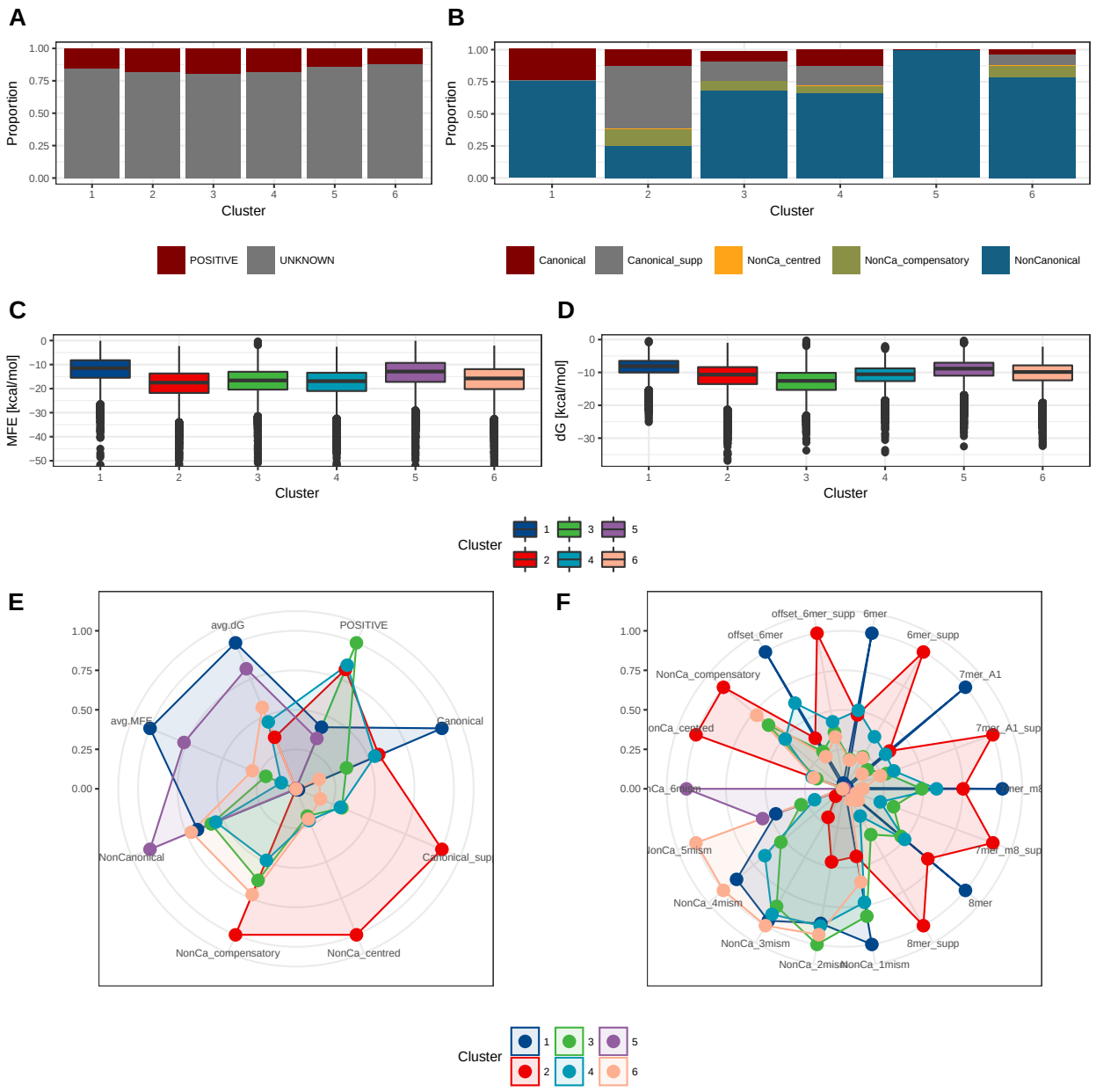

Figura 12: Resultados obtenidos con CLARA en ratón. (A-B) Diagramas de barras representando las proporciones de dianas funcionales y de los distintos tipos de interacción en los *clusters*. (C) Boxplot para ilustrar la distribución del MFE predicho para las interacciones de los grupos. (D) Boxplot para mostrar la distribución de la energía libre predicha cuando se utilizan secuencias extendidas de las dianas. (E-F) Gráficos de radar que condensan la información anterior.

de inferir nuevas propiedades de las dianas de microRNA a partir de datos CLIP-seq, tanto por la posible inconsistencia de los resultados como por el desconocimiento de la estructura *in vivo* de las interacciones miRNA-mRNA. Finalmente, a lo anterior también hay que añadir que el volumen y el carácter multidimensional de los datos, incluso después de usar técnicas como PCA o ICA, supuso un factor limitante por varios motivos. Por un lado, la implementación de algunos algoritmos de *clustering* y el cálculo de muchos índices de validación interna requieren la generación de matrices de disimilitud, que en este caso resultaban demasiado grandes y causaban insalvables problemas de memoria con los recursos de los que se disponía. Este escollo, como trabajo futuro, se podría salvar mediante el uso de superordenadores o de *clusters*. Por otro lado, se sabe que ciertos algoritmos de agrupamiento, entre los que se encuentra CLARA, reducen su rendimiento cuando la dimensionalidad de los datos es alta [24].

6. Conclusiones

El grado de cumplimiento de los objetivos propuestos al inicio del proyecto varía en función de las distintas fases o bloques que componen este TFM. Entre las metas cumplidas de acuerdo a lo planeado, se encuentran todas aquellas relacionadas con el procesamiento de los datos y la extracción de atributos a partir de las quimeras miRNA-diana. La existencia de proyectos de *software* libre de gran envergadura, tales como Bioconductor, facilita enormemente el tratamiento de una amplia variedad de datos genómicos y permite obtener una gran cantidad de información de forma rápida y sencilla. No ocurre lo mismo con los programas dirigidos a la caracterización de interacciones miRNA-mRNA. El repertorio de posibles atributos a extraer es enorme y dependiendo del enfoque de la investigación es posible hacer énfasis en variables de diferente naturaleza. Además, siempre puede surgir la necesidad de calcular y poner a prueba nuevos descriptores. Quizá, la escasez de este tipo de *software* esté motivada por la necesidad de los investigadores de calcular sus propias variables descriptivas. Sin embargo, ésta no siempre resulta una tarea sencilla y puede convertirse en un proceso tedioso. Por esta razón, uno de los propósitos del proyecto ha sido la creación de un paquete de R que aborde, en la medida de lo posible, este tipo de cometidos. En otro orden de ideas, aquellos objetivos que no se han conseguido de forma totalmente satisfactoria, son los relacionados con la caracterización y la identificación de atributos relevantes en las dianas no canónicas. Las limitaciones relativas a este asunto ya se han discutido anteriormente.

En todo momento se ha respetado la planificación del proyecto. No se han producido cambios en el orden o la naturaleza de las tareas propuestas y tampoco ha habido retrasos en el cumplimiento de los hitos marcados. Pese a que los resultados obtenidos siguen sin despejar algunos interrogantes, la metodología definida en la fases iniciales del TFM es adecuada a los objetivos planteados. En caso de disponer de más tiempo, la estrategia a seguir hubiera pasado por calcular más atributos y por emplear otros programas para predecir la estructura secundaria de las interacciones; si bien, en términos generales, el esquema de trabajo seguiría siendo el mismo o muy parecido. En cuanto a

las líneas de trabajo futuro, resulta evidente la necesidad de desarrollar nuevos métodos de *clustering* que se adapten al volumen y complejidad de los datos generados en el marco de la investigación biomédica [24]. Por otra parte, aunque en referencia al volumen creciente de datos, podría resultar interesante llevar a cabo la implementación de algunos procedimientos, como la extracción de atributos, utilizando lenguajes de reciente desarrollo tales como Julia [43], un lenguaje de alto nivel y de alto rendimiento orientado a la computación científica y numérica.

7. Glosario

- Diana: ARN mensajero (mRNA) con el que interacciona un microARN (miRNA).
- Semilla: Región formada por los primeros nucleótidos del extremo 5' de los miRNAs que juega un rol importante en el reconocimiento específico de las dianas.
- UTR: Del inglés *untranslated region*. Se trata de regiones no traducidas que se encuentran a ambos lados de un mRNA.
- Quimera: Molécula de ARN monocatenario que se forma al ligar las cadenas nucleotídicas de un miRNA y el mRNA diana con el que interacciona. La formación de estos híbridos permite reconocer interacciones *in vivo*.
- Dímero: En este contexto hace referencia a un miRNA unido a un mRNA a través de la interacción entre pares de bases.
- MFE: Del inglés *minimum free energy*. Mide la estabilidad termodinámica de una estructura química.
- *Clustering*: Conjunto de técnicas enmarcadas dentro del aprendizaje no supervisado que se emplean para estudiar la relación entre un conjunto de variables.

8. Bibliografía

1. Miyoshi K, Miyoshi T, Siomi H. 2010 Many ways to generate microRNA-like small RNAs: Non-canonical pathways for microRNA production. *Molecular Genetics and Genomics* **284**, 95–103. (doi:10.1007/s00438-010-0556-1)

2. Agarwal V, Bell GW, Nam J-W, Bartel DP. 2015 Predicting effective microRNA target sites in mammalian mRNAs. *eLife* **4**, e05005. (doi:10.7554/eLife.05005)

3. Helwak A, Kudla G, Dudnakova T, Tollervey D. 2013 Mapping the Human miRNA Interactome by CLASH Reveals Frequent Noncanonical Binding. *Cell* **153**, 654–665. (doi:10.1016/j.cell.2013.03.043)

4. Grosswendt S, Filipchyk A, Manzano M, Klironomos F, Schilling M, Herzog M, Gottwein E, Rajewsky N. 2014 Unambiguous Identification of miRNA:target site Interactions by Different Types of Ligation Reactions. *Molecular cell* **54**, 1042–1054. (doi:10.1016/j.molcel.2014.03.049)

5. Moore MJ, Scheel TKH, Luna JM, Park CY, Fak JJ, Nishiuchi E, Rice CM, Darnell RB. 2015 miRNA–target chimeras reveal miRNA 3'-end pairing as a major determinant of Argonaute target specificity. *Nature Communications* **6**. (doi:10.1038/ncomms9864)

6. Vlachos IS *et al.* 2015 DIANA-TarBase v7.0: Indexing more than half a million experimentally supported miRNA:mRNA interactions. *Nucleic Acids Research* **43**, D153–D159. (doi:10.1093/nar/gku1215)

7. Lorenz R, Bernhart SH, Höner zu Siederdissen C, Tafer H, Flamm C, Stadler PF, Hofacker IL. 2011 ViennaRNA Package 2.0. *Algorithms for Molecular Biology* **6**, 26. (doi:10.1186/1748-7188-6-26)

8. R Core Team. 2017 *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. See https://www.R-project.org/.

9. Ha M, Kim VN. 2014 Regulation of microRNA biogenesis. *Nature Reviews. Molecular Cell Biology* **15**, 509–524. (doi:10.1038/nrm3838)

10. Lin S, Gregory RI. 2015 MicroRNA biogenesis pathways in cancer. *Nature Reviews. Cancer* **15**, 321–333. (doi:10.1038/nrc3932)

11. MacFarlane L-A, Murphy PR. 2010 MicroRNA: Biogenesis, Function and Role in Cancer. *Current Genomics* **11**, 537–561. (doi:10.2174/138920210793175895)

12. Fabian MR, Sonenberg N. 2012 The mechanics of miRNA-mediated gene silencing: A look under the hood of miRISC. *Nature Structural & Molecular Biology* **19**, 586–593. (doi:10.1038/nsmb.2296)

13. Bartel DP. 2009 MicroRNAs: Target Recognition and Regulatory Functions. *Cell* **136**, 215–233. (doi:10.1016/j.cell.2009.01.002)

14. Pasquinelli AE. 2012 MicroRNAs and their targets: Recognition, regulation and an emer-

ging reciprocal relationship. *Nature Reviews. Genetics* **13**, 271–282. (doi:10.1038/nrg3162)

15. Hausser J, Zavolan M. 2014 Identification and consequences of miRNA-target interactions–beyond repression of gene expression. *Nature Reviews. Genetics* **15**, 599–612. (doi:10.1038/nrg3765)

16. Friedman RC, Farh KK-H, Burge CB, Bartel DP. 2009 Most mammalian mRNAs are conserved targets of microRNAs. *Genome Research* **19**, 92–105. (doi:10.1101/gr.082701.108)

17. Moszyńska A, Gebert M, Collawn JF, Bartoszewski R. 2017 SNPs in microRNA target sites and their potential role in human disease. *Open Biology* **7**, 170019. (doi:10.1098/rsob.170019)

18. Broughton JP, Pasquinelli AE. 2016 A tale of two sequences: microRNA-target chimeric reads. *Genetics, Selection, Evolution : GSE* **48**. (doi:10.1186/s12711-016-0209-x)

19. Lewis BP, Shih I-h, Jones-Rhoades MW, Bartel DP, Burge CB. 2003 Prediction of Mammalian MicroRNA Targets. *Cell* **115**, 787–798. (doi:10.1016/S0092-8674(03)01018-3)

20. Enright AJ, John B, Gaul U, Tuschl T, Sander C, Marks DS. 2003 MicroRNA targets in Drosophila. *Genome Biology* **5**, R1. (doi:10.1186/gb-2003-5-1-r1)

21. Min H, Yoon S. 2010 Got target?: Computational methods for microRNA target prediction and their extension. *Experimental & Molecular Medicine* **42**, 233–244. (doi:10.3858/emm.2010.42.4.032)

22. Jain AK, Murty MN, Flynn PJ. 1999 Data Clustering: A Review. *ACM Comput. Surv.* **31**, 264–323. (doi:10.1145/331499.331504)

23. Xu R, Wunsch D. 2005 Survey of clustering algorithms. *IEEE Transactions on Neural Networks* **16**, 645–678. (doi:10.1109/TNN.2005.845141)

24. Xu R, Wunsch DC. 2010 Clustering Algorithms in Biomedical Research: A Review. *IEEE Reviews in Biomedical Engineering* **3**, 120–154. (doi:10.1109/RBME.2010.2083647)

25. Xu D, Tian Y. 2015 A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science* **2**, 165–193. (doi:10.1007/s40745-015-0040-1)

26. Kassambara A. 2017 *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning*. STHDA.

27. Fraley C, Raftery AE. 2002 Model-Based Clustering, Discriminant Analysis, and Density Estimation. *Journal of the American Statistical Association* **97**, 611–631. (doi:10.1198/016214502760047131)

28. Couvreur C. 1997 The EM Algorithm: A Guided Tour. In *Computer Intensive Methods in Control and Signal Processing*, pp. 209–222. Birkhäuser, Boston, MA. (doi:10.1007/978-1-4612-1996-5_12)

29. Ankerst M, Breunig MM, Kriegel H-P, Sander J. 1999 OPTICS: Ordering Points to Identify the Clustering Structure. In *Proceedings of the 1999 ACM SIGMOD Inter-*

national Conference on Management of Data, pp. 49–60. New York, NY, USA: ACM. (doi:10.1145/304182.304187)

30. Pagès H. 2017 *BSgenome: Software infrastructure for efficient representation of full genomes and their snps*.

31. Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, Morgan M, Carey V. 2013 Software for computing and annotating genomic ranges. *PLoS Computational Biology* **9**. (doi:10.1371/journal.pcbi.1003118)

32. Durinck S, Spellman PT, Birney E, Huber W. 2009 Mapping identifiers for the integration of genomic datasets with the r/bioconductor package biomaRt. *Nature Protocols* **4**, 1184–1191.

33. Haunsberger SJ, Connolly NMC, Prehn JHM. 2016 miRNAmeConverter: An r/bioconductor package for translating mature miRNA names to different miRBase versions. *Bioinformatics* (doi:10.1093/bioinformatics/btw660)

34. Wickham H, Francois R, Henry L, Müller K. 2017 *Dplyr: A grammar of data manipulation*. See https://CRAN.R-project.org/package=dplyr.

35. Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K. 2018 *Cluster: Cluster analysis basics and extensions*.

36. Scrucca L, Fop M, Murphy TB, Raftery AE. 2016 mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal* **8**, 205–233.

37. Hahsler M, Piekenbrock M. 2017 *Dbscan: Density based clustering of applications with noise (dbscan) and related algorithms*. See https://CRAN.R-project.org/package=dbscan.

38. Sullivan GM, Feinn R. 2012 Using Effect Size—or Why the P Value Is Not Enough. *Journal of Graduate Medical Education* **4**, 279–282. (doi:10.4300/JGME-D-12-00156.1)

39. Jed Wing MKC from *et al.* 2017 *Caret: Classification and regression training*. See https://CRAN.R-project.org/package=caret.

40. Latha K. 2017 *Experiment and Evaluation in Information Retrieval Models*. CRC Press.

41. Lantz B. 2015 *Machine Learning with R: Discover how to Build Machine Learning Algorithms, Prepare Data, and Dig Deep Into Data Prediction Techniques with R*. Packt Publishing.

42. Berezikov E, Cuppen E, Plasterk RHA. 2006 Approaches to microRNA discovery. *Nature Genetics* **38 Suppl**, S2–7. (doi:10.1038/ng1794)

43. Bezanson J, Edelman A, Karpinski S, Shah V. 2017 Julia: A Fresh Approach to Numerical Computing. *SIAM Review* **59**, 65–98. (doi:10.1137/141000671)

9. Anexo I: Gráficos y tablas de resultados

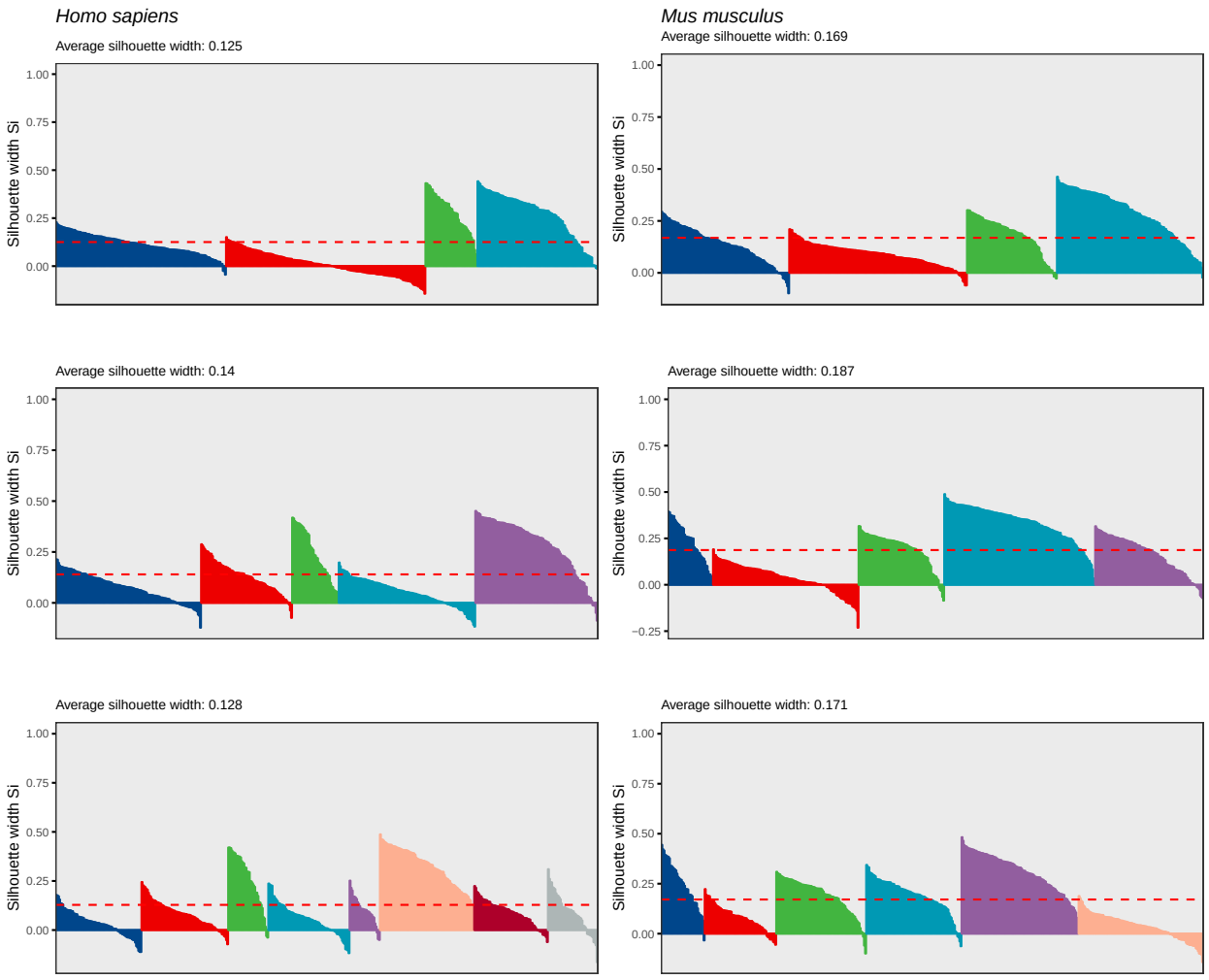

Figura 13: Detalle de la silueta de los tres mejores agrupamientos para cada especie. Cuando los puntos registran un coeficiente de la silueta negativo se debe a que probablemente estén mal clasificados.

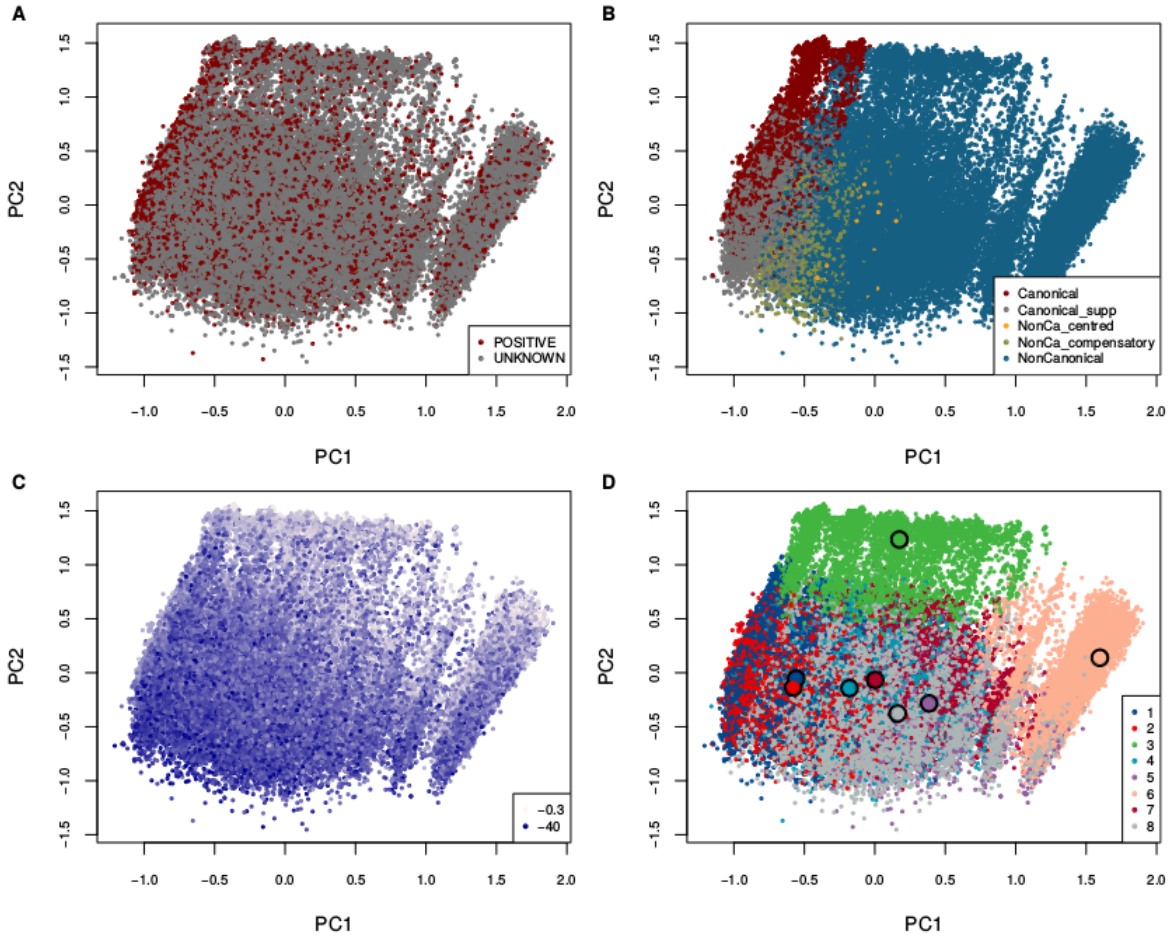

Figura 14: Gráficos de puntos para visualizar la distribución de los datos en humano. (A) Funcionalidad de las interacciones. (B) Tipos de interacción. (C) MFE de los dímeros. (D) Agrupamiento obtenido con el algoritmo CLARA.

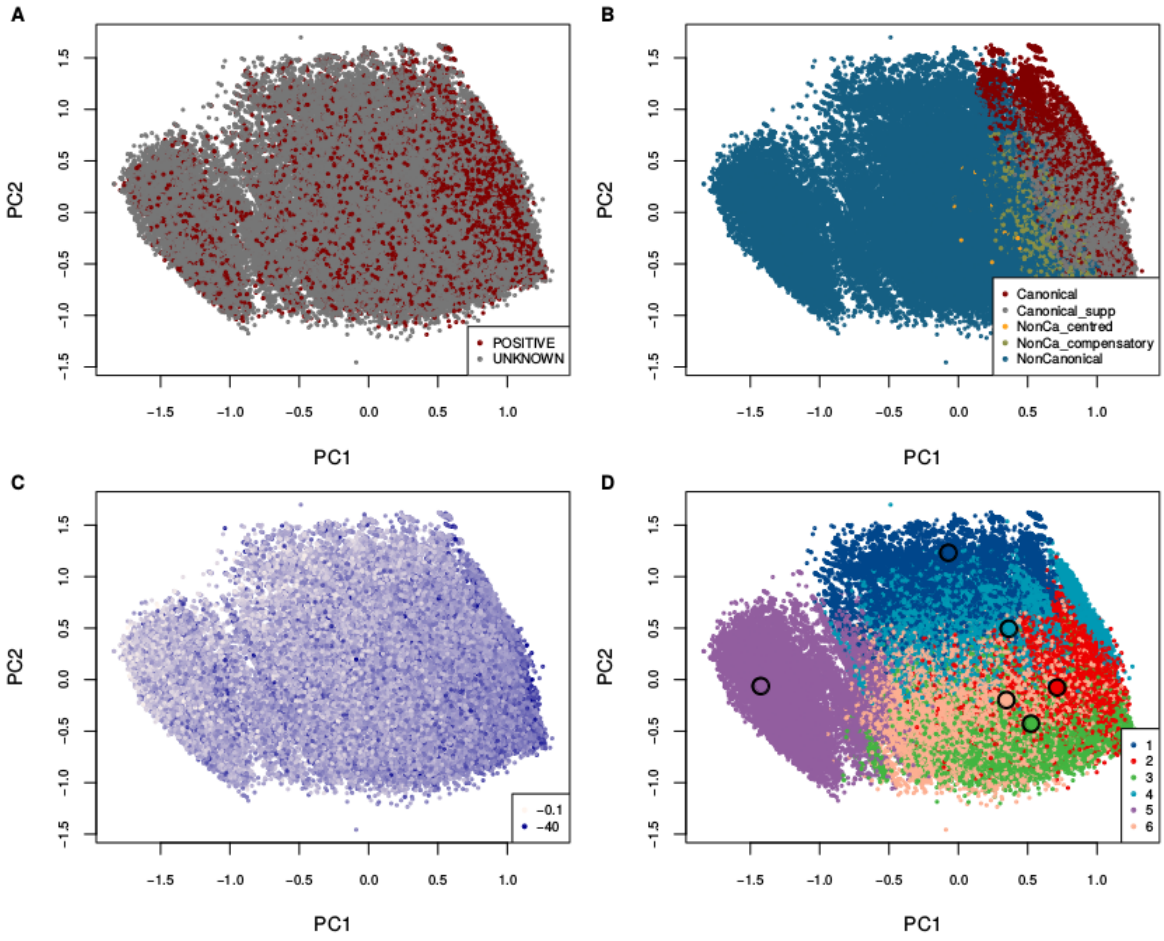

Figura 15: Gráficos de puntos para visualizar la distribución de los datos en ratón. (A) Funcionalidad de las interacciones. (B) Tipos de interacción. (C) MFE de los dímeros. (D) Agrupamiento obtenido con el algoritmo CLARA.

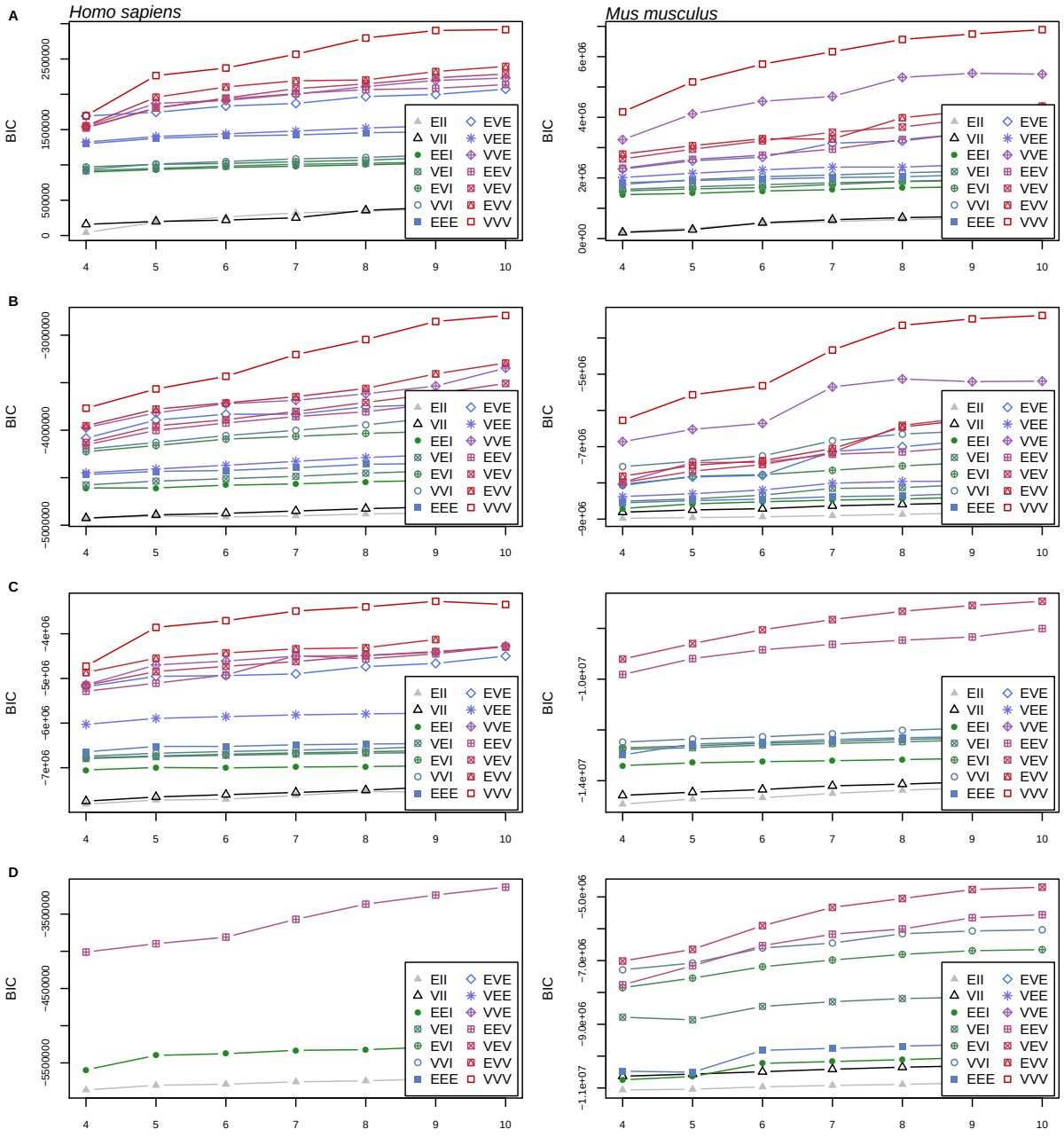

Figura 16: Criterio de información bayesiano (BIC) según modelo y número de *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA.

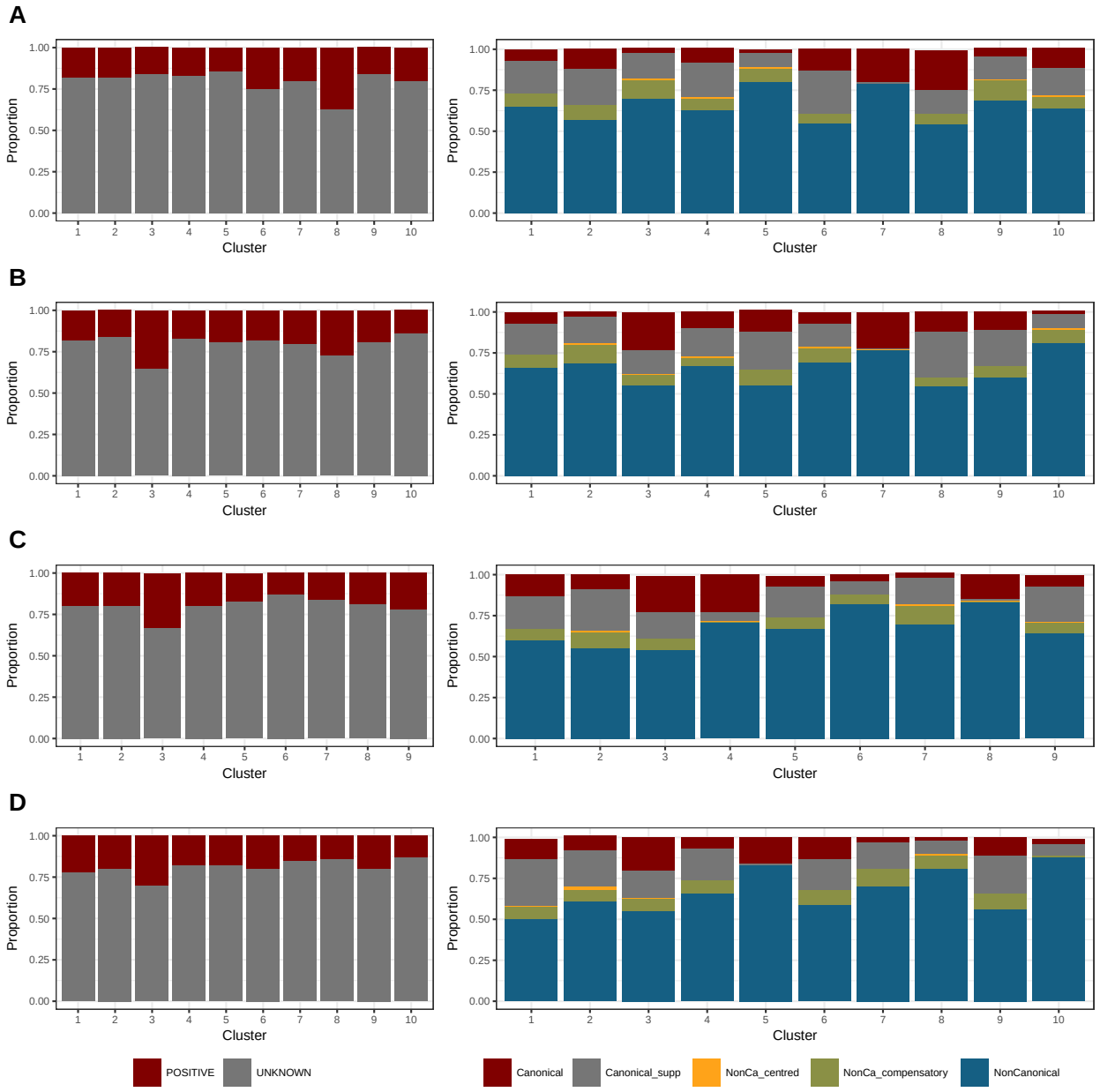

Figura 17: Resultados obtenidos con el algoritmo EM en humano. Diagramas de barras representando las proporciones de interacciones funcionales y de los distintos tipos de dianas en los *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA.

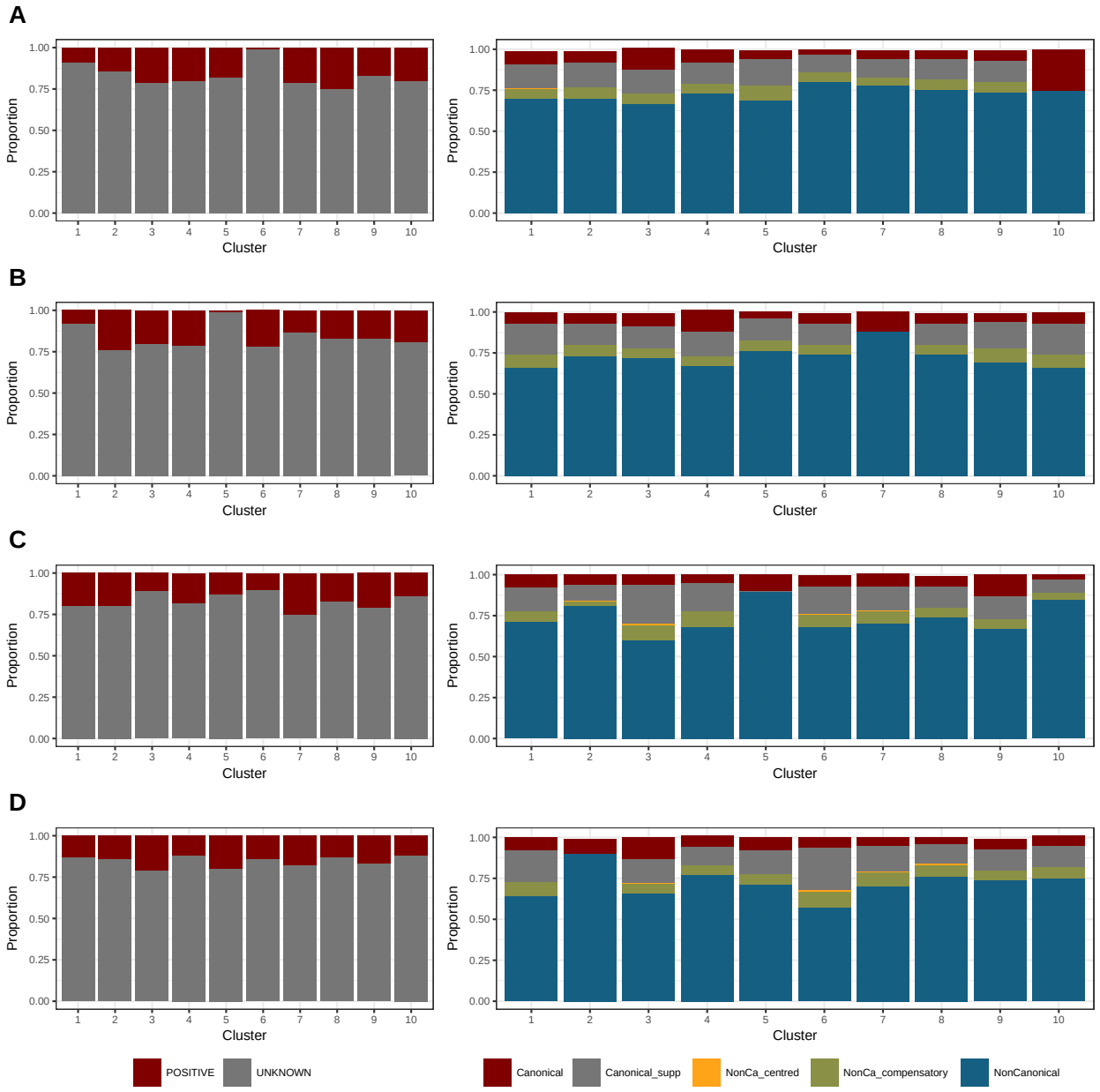

Figura 18: Resultados obtenidos con el algoritmo EM en ratón. Diagramas de barras representando las proporciones de interacciones funcionales y de los distintos tipos de dianas en los *clusters*. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA.

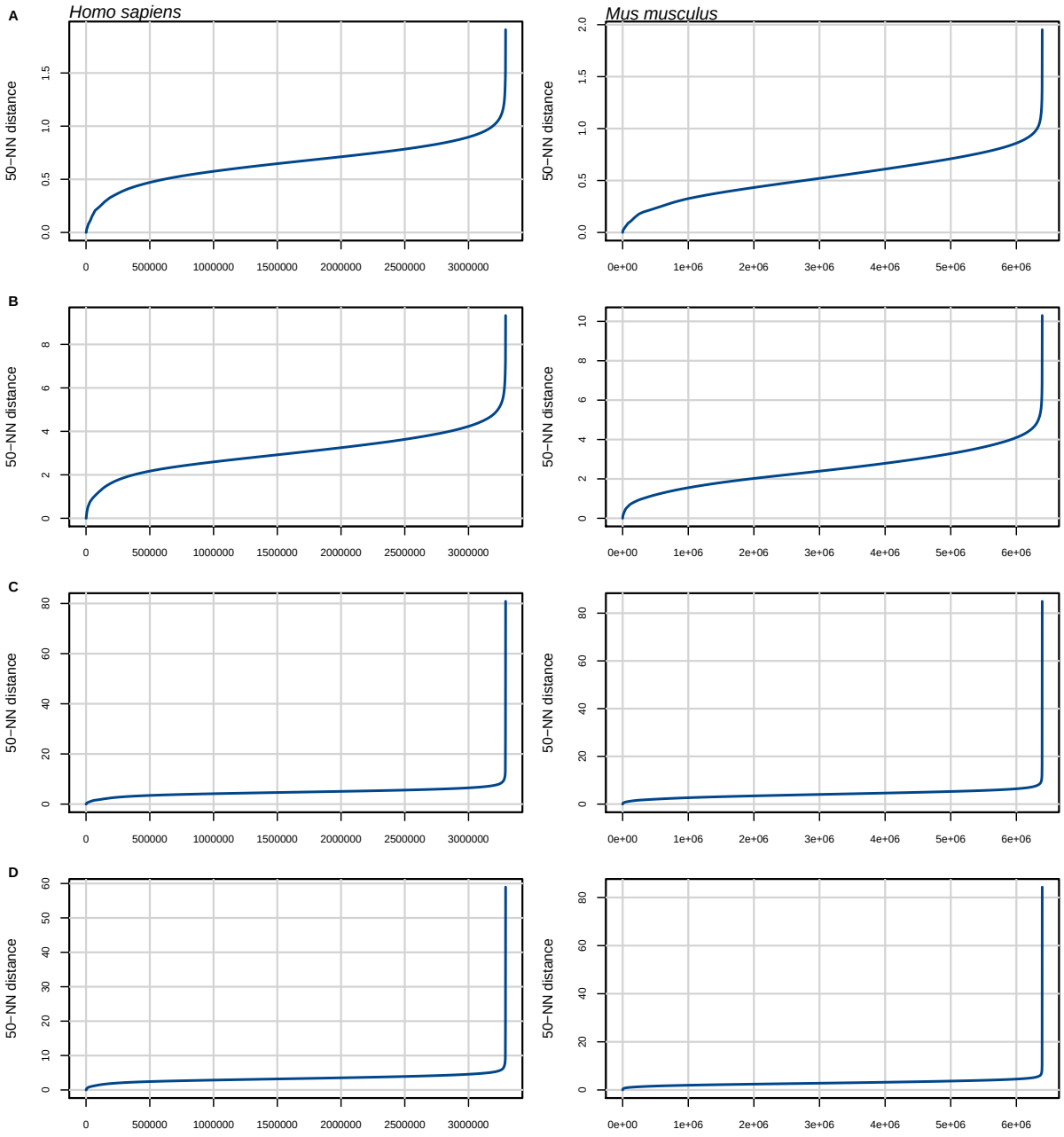

Figura 19: Gráfico de distancia a los k vecinos más cercanos para identificar el punto de corte del parámetro épsilon; hay que buscar el punto de inflexión de la curva. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA.

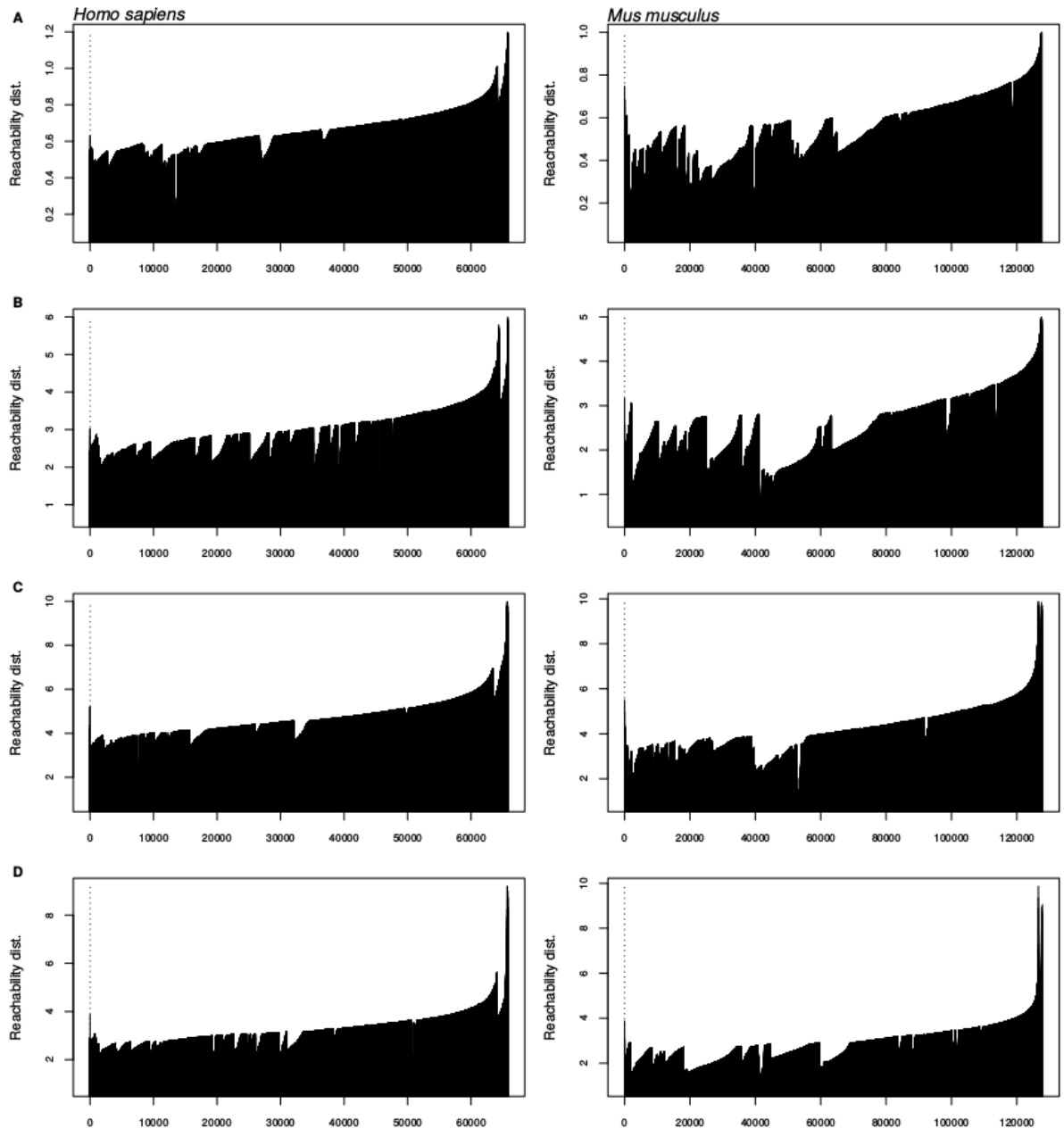

Figura 20: *Reachability distance* con OPTICS. (A) Variables normalizadas y PCA, (B) variables normalizadas e ICA, (C) variables estandarizadas y PCA y (D) variables estandarizadas e ICA.

	Homo sapiens				*Mus musculus*			
	variable	p-valor	d	efecto	variable	p-valor	d	efecto
1	mir.CWC.seed.maxlen	0	0,373	pequeño	mir.CT	0	0,196	despreciable
2	mir.WC.seed	0	0,340	pequeño	mir.GT	0	0,179	despreciable
3	mir.CBP.seed.maxlen	0	0,338	pequeño	mir.CC	0	0,152	despreciable
4	mir.BP.seed	0	0,308	pequeño	mir.CWC.seed.maxlen	0	0,144	despreciable
5	mir.GT	0	0,249	pequeño	mir.GA	0	0,141	despreciable
6	mir.TA	0	0,242	pequeño	mir.C	0	0,139	despreciable
7	mir.C	0	0,231	pequeño	mir.TA	0	0,137	despreciable
8	mir.CT	0	0,231	pequeño	mir.WC.seed	0	0,125	despreciable
9	mir.AG	0	0,228	pequeño	mir.AA	0	0,124	despreciable
10	mir.CWC.maxlen	0	0,224	pequeño	mir.CBP.seed.maxlen	0	0,119	despreciable
11	mir.WC.1stpos	0	0,219	pequeño	mir.TC	0	0,115	despreciable
12	mir.BP.1stpos	0	0,216	pequeño	mir.purine	0	0,107	despreciable
13	mir.weak	0	0,197	despreciable	mir.pyrimidine	0	0,107	despreciable
14	mir.strong	0	0,197	despreciable	mir.BP.seed	0	0,104	despreciable
15	tar.CBP.maxlen	0	0,193	despreciable	mir.G	0	0,087	despreciable
16	dG.cofold	0	0,191	despreciable	tar.MRE.lenght	0	0,082	despreciable
17	mir.CC	0	0,186	despreciable	duplex.BP	0	0,079	despreciable
18	tar.CBP.avglen	0	0,157	despreciable	mir.CG	0	0,076	despreciable
19	mir.TC	0	0,155	despreciable	duplex.WC	0	0,075	despreciable
20	duplex.WC	0	0,152	despreciable	mir.WC.1stpos	0	0,070	despreciable
21	mir.T	0	0,151	despreciable	mir.AT	0	0,070	despreciable
22	mir.AC	0	0,139	despreciable	mir.GU.outseed	0	0,069	despreciable
23	duplex.BP	0	0,137	despreciable	mir.BP.1stpos	0	0,069	despreciable
24	MFE	0	0,135	despreciable	mir.GU.seed	0	0,064	despreciable
25	mir.CWC.avglen	0	0,133	despreciable	mir.weak	0	0,064	despreciable
26	mir.CBP.maxlen	0	0,133	despreciable	mir.strong	0	0,064	despreciable
27	mir.purine	0	0,121	despreciable	mir.CWC.maxlen	0	0,061	despreciable
28	mir.pyrimidine	0	0,121	despreciable	mir.CWC.outseed.maxlen	0	0,060	despreciable
29	dG.rnaup	0	0,115	despreciable	tar.BP.maxdist	0	0,058	despreciable
30	mir.CWC.outseed.maxlen	0	0,110	despreciable	mir.AG	0	0,058	despreciable
31	tar.BP.avgdist	0	0,109	despreciable	mir.TG	0	0,055	despreciable
32	tar.BP.maxdist	0	0,101	despreciable	tar.AsymLoop.sumlen	0	0,051	despreciable
33	tar.CBP.minlen	0	0,100	despreciable	mir.CA	0	0,050	despreciable
34	tar.MRE.IntramolBP	0	0,100	despreciable	mir.WC.maxdist	0	0,049	despreciable
35	mir.G	0	0,100	despreciable	mir.BP.maxdist	0	0,049	despreciable
36	mir.CBP.avglen	0	0,096	despreciable	tar.AsymLoop.avglen	0	0,049	despreciable
37	mir.WC.outseed	0	0,095	despreciable	tar.AsymLoop.maxlen	0	0,049	despreciable
38	mir.TT	0	0,095	despreciable	mir.CBP.maxlen	0	0,047	despreciable
39	mir.CBP.outseed.maxlen	0	0,093	despreciable	tar.CBP.maxlen	0	0,047	despreciable
40	mir.GA	0	0,090	despreciable	dGu.tar	0	0,046	despreciable
41	mir.GU.seed	0	0,089	despreciable	mir.CBP.numreg	0	0,045	despreciable
42	dG.int	0	0,083	despreciable	tar.AsymLoop.minlen	0	0,044	despreciable
43	tar.3WJ.num	0	0,079	despreciable	mir.CWC.numreg	0	0,043	despreciable
44	mir.BP.outseed	0	0,074	despreciable	tar.BP.avgdist	0	0,043	despreciable
45	mir.CG	0	0,068	despreciable	mir.AC	0	0,043	despreciable
46	mir.WC.lastpos	0	0,064	despreciable	tar.CBP.numreg	0	0,041	despreciable
47	mir.BP.lastpos	0	0,062	despreciable	mir.AsymLoop.sumlen	0	0,039	despreciable
48	duplex.multiWJ.num	0	0,060	despreciable	mir.AsymLoop.maxlen	0	0,038	despreciable

Cuadro 3: Test de la t para identificar diferencias en funcionalidad de las dianas según la variable en cuestión y medida de la magnitud del efecto con la d de Cohen (continúa en el cuadro 4).

	Homo sapiens				*Mus musculus*			
	variable	p-valor	d	efecto	variable	p-valor	d	efecto
48	duplex.multiWJ.num	0,000	0,060	despreciable	mir.AsymLoop.maxlen	0,000	0,038	despreciable
49	mir.A	0,000	0,060	despreciable	mir.IntramolBP	0,000	0,037	despreciable
50	mir.AsymLoop.maxlen	0,000	0,056	despreciable	mir.AsymLoop.avglen	0,000	0,037	despreciable
51	mir.AsymLoop.avglen	0,000	0,055	despreciable	tar.3WJ.num	0,000	0,037	despreciable
52	mir.AsymLoop.sumlen	0,000	0,052	despreciable	mir.A	0,000	0,037	despreciable
53	mir.CBP.minlen	0,000	0,051	despreciable	tar.Bulge.maxlen	0,000	0,036	despreciable
54	mir.CWC.minlen	0,000	0,050	despreciable	tar.MRE.IntramolBP	0,000	0,035	despreciable
55	mir.AsymLoop.minlen	0,000	0,050	despreciable	tar.Bulge.avglen	0,000	0,035	despreciable
56	mir.Bulge.sumlen	0,000	0,049	despreciable	tar.Bulge.sumlen	0,000	0,034	despreciable
57	mir.Bulge.minlen	0,000	0,048	despreciable	mir.AsymLoop.minlen	0,000	0,034	despreciable
58	mir.Bulge.avglen	0,000	0,048	despreciable	mir.TT	0,000	0,034	despreciable
59	mir.Bulge.maxlen	0,000	0,048	despreciable	dG.int	0,000	0,033	despreciable
60	tar.MRE.lenght	0,000	0,046	despreciable	duplex.AsymLoop.num	0,000	0,033	despreciable
61	mir.WC.avgdist	0,000	0,045	despreciable	duplex.GU	0,000	0,032	despreciable
62	tar.AsymLoop.minlen	0,000	0,043	despreciable	tar.Bulge.minlen	0,000	0,032	despreciable
63	tar.AsymLoop.avglen	0,000	0,042	despreciable	mir.WC.lastpos	0,000	0,031	despreciable
64	mir.WC.maxdist	0,000	0,040	despreciable	mir.BP.lastpos	0,000	0,030	despreciable
65	mir.BP.maxdist	0,000	0,040	despreciable	mir.T	0,000	0,030	despreciable
66	mir.Bulge.num	0,000	0,037	despreciable	mir.GC	0,000	0,030	despreciable
67	tar.AsymLoop.maxlen	0,000	0,037	despreciable	mir.CBP.avglen	0,000	0,029	despreciable
68	mir.GU.outseed	0,000	0,036	despreciable	tar.CBP.avglen	0,001	0,027	despreciable
69	tar.AsymLoop.sumlen	0,000	0,035	despreciable	mir.CWC.avglen	0,001	0,026	despreciable
70	tar.Bulge.num	0,001	0,033	despreciable	mir.WC.avgdist	0,003	0,023	despreciable
71	tar.CBP.numreg	0,001	0,032	despreciable	mir.BP.avgdist	0,003	0,023	despreciable
72	mir.GG	0,001	0,030	despreciable	mir.CBP.outseed.maxlen	0,003	0,023	despreciable
73	mir.GC	0,010	0,025	despreciable	mir.GG	0,003	0,019	despreciable
74	mir.TG	0,007	0,024	despreciable	mir.WC.outseed	0,019	0,018	despreciable
75	mir.IntramolBP	0,022	0,022	despreciable	mir.Bulge.num	0,114	0,012	despreciable
76	duplex.AsymLoop.num	0,026	0,022	despreciable	dG.rnaup	0,172	0,010	despreciable
77	tar.Bulge.sumlen	0,023	0,021	despreciable	mir.WC.mindist	0,187	0,009	despreciable
78	mir.CBP.numreg	0,042	0,020	despreciable	mir.BP.mindist	0,187	0,009	despreciable
79	mir.AA	0,037	0,020	despreciable	mir.CBP.minlen	0,258	0,009	despreciable
80	mir.WC.mindist	0,024	0,019	despreciable	mir.Bulge.sumlen	0,242	0,009	despreciable
81	mir.BP.mindist	0,024	0,019	despreciable	mir.Bulge.maxlen	0,330	0,008	despreciable
82	tar.Bulge.maxlen	0,049	0,018	despreciable	dG.cofold	0,319	0,007	despreciable
83	tar.Bulge.avglen	0,110	0,015	despreciable	mir.BP.outseed	0,347	0,007	despreciable
84	duplex.SymLoop.num	0,114	0,015	despreciable	MFE	0,389	0,006	despreciable
85	tar.Bulge.minlen	0,271	0,010	despreciable	mir.Bulge.avglen	0,425	0,006	despreciable
86	duplex.GU	0,334	0,009	despreciable	tar.Bulge.num	0,418	0,006	despreciable
87	mir.CWC.numreg	0,462	0,007	despreciable	mir.Bulge.minlen	0,538	0,005	despreciable
88	duplex.SymLoop.maxlen	0,501	0,007	despreciable	duplex.SymLoop.minlen	0,523	0,005	despreciable
89	dGu.tar	0,517	0,006	despreciable	duplex.SymLoop.avglen	0,564	0,004	despreciable
90	mir.3WJ.num	0,549	0,006	despreciable	duplex.SymLoop.maxlen	0,626	0,004	despreciable
91	mir.BP.avgdist	0,588	0,005	despreciable	mir.3WJ.num	0,554	0,004	despreciable
92	mir.AT	0,629	0,005	despreciable	duplex.multiWJ.num	0,652	0,004	despreciable
93	duplex.SymLoop.avglen	0,711	0,004	despreciable	tar.CBP.minlen	0,690	0,003	despreciable
94	duplex.SymLoop.sumlen	0,713	0,004	despreciable	duplex.SymLoop.sumlen	0,712	0,003	despreciable
95	mir.CA	0,674	0,004	despreciable	duplex.SymLoop.num	0,870	0,001	despreciable
96	duplex.SymLoop.minlen	0,994	0,000	despreciable	mir.CWC.minlen	0,952	0,000	despreciable

Cuadro 4: Test de la t para identificar diferencias en funcionalidad de las dianas según la variable en cuestión y medida de la magnitud del efecto con la d de Cohen. Ver Lista de atributos extraídos, en Anexo III, para comprender la nomenclatura de las variables.

	Homo sapiens	Mus musculus
1	MFE	dG.int
2	dG.cofold	dGu.tar
3	dG.rnaup	duplex.WC
4	dG.int	duplex.GU
5	duplex.WC	mir.GU.seed
6	mir.GU.seed	mir.WC.outseed
7	mir.WC.outseed	mir.GU.outseed
8	mir.GU.outseed	mir.IntramolBP
9	mir.IntramolBP	mir.WC.1stpos
10	mir.WC.1stpos	mir.WC.lastpos
11	mir.WC.lastpos	mir.WC.avgdist
12	mir.WC.mindist	mir.BP.avgdist
13	mir.WC.avgdist	mir.BP.maxdist
14	mir.BP.maxdist	mir.CWC.numreg
15	mir.CWC.minlen	mir.CWC.avglen
16	mir.CWC.avglen	mir.CWC.maxlen
17	mir.CWC.maxlen	mir.CWC.seed.maxlen
18	mir.CWC.seed.maxlen	mir.CWC.outseed.maxlen
19	mir.CWC.outseed.maxlen	mir.CBP.numreg
20	mir.CBP.numreg	mir.CBP.avglen
21	mir.CBP.minlen	mir.CBP.maxlen
22	mir.CBP.maxlen	mir.CBP.outseed.maxlen
23	mir.CBP.outseed.maxlen	tar.MRE.lenght
24	tar.BP.maxdist	tar.BP.maxdist
25	tar.CBP.numreg	tar.CBP.numreg
26	tar.CBP.minlen	tar.CBP.avglen
27	tar.CBP.maxlen	tar.CBP.maxlen
28	mir.Bulge.num	tar.MRE.IntramolBP
29	mir.Bulge.sumlen	tar.Bulge.minlen
30	tar.Bulge.num	tar.Bulge.sumlen
31	duplex.AsymLoop.num	duplex.AsymLoop.num
32	mir.AsymLoop.minlen	mir.AsymLoop.minlen
33	tar.AsymLoop.minlen	tar.AsymLoop.avglen
34	tar.3WJ.num	tar.3WJ.num
35	duplex.multiWJ.num	mir.A
36	mir.A	mir.C
37	mir.C	mir.G
38	mir.G	mir.T
39	mir.T	mir.weak
40	mir.weak	mir.pyrimidine
41	mir.pyrimidine	mir.AA
42	mir.AA	mir.AC
43	mir.AC	mir.AG
44	mir.AG	mir.AT
45	mir.CC	mir.CA
46	mir.CG	mir.CC
47	mir.CT	mir.CG
48	mir.GA	mir.CT
49	mir.GC	mir.GA
50	mir.GG	mir.GC
51	mir.GT	mir.GG
52	mir.TA	mir.GT
53	mir.TC	mir.TA
54	mir.TG	mir.TC
55	mir.TT	mir.TG
56		mir.TT

Cuadro 5: Conjunto de variables resultante tras la selección de atributos relevantes. Ver Lista de atributos extraídos, en Anexo III, para comprender la nomenclatura de las variables.

10. Anexo II: Procesamiento de los datos

10.1. Paquetes

Paquetes requeridos durante el procesamiento de los datos:

```
if (!require(biomaRt)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("biomaRt")
}
if (!require(GenomicFeatures)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("GenomicFeatures")
}
if (!require(GenomicRanges)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("GenomicRanges")
}
if (!require(miRNAmeConverter)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("miRNAmeConverter")
}
if (!require(BSgenome.Hsapiens.UCSC.hg18)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Hsapiens.UCSC.hg18")
}
if (!require(BSgenome.Hsapiens.UCSC.hg19)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Hsapiens.UCSC.hg19")
}
if (!require(BSgenome.Mmusculus.UCSC.mm9)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("BSgenome.Mmusculus.UCSC.mm9")
}
if (!require(TxDb.Hsapiens.UCSC.hg19.knownGene)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("TxDb.Hsapiens.UCSC.hg19.knownGene")
}
if (!require(TxDb.Mmusculus.UCSC.mm9.knownGene)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("TxDb.Mmusculus.UCSC.mm9.knownGene")
}
if (!require(dplyr)) {
  install.packages("dplyr")
}
if (!require(plyr)) {
  install.packages("plyr")}
```

10.2. Funciones

10.2.1. Obtención de información

Función `coordToGeneID`: Identifica los genes que solapan con una localización cromosómica y muestra los identificadores de dichos genes.

```
coordToGeneID <- function(txdb, chr, start, end, strand, type = "entrez") {
  # Args:
  #   txdb              Objeto TxDb.
  #   chr               Vector de tipo carácter para indicar el cromosoma;
  #                     ejemplo: "chr1".
  #   start, end        Vectores de números enteros para indicar posición
  #                     de inicio y posición de fin en el cromosoma.
  #   strand            Vector de tipo carácter para indicar hebra de ADN.
  #                     Solamente admite "+" y "-".
  #   type              Nombre para el vector de identificadores génicos.
  #
  # Return:
  #   Devuelve un data.frame en el que cada localización cromosómica queda
  #   representada por un número de consulta o query. Cada query se asocia
  #   con uno o varios identificadores génicos; NA si no existe solapamiento.

  require(GenomicRanges)
  # Localizaciones cromosómicas a consultar y genes anotados.
  query <- GRanges(chr, IRanges(start, end), strand)
  subject <- genes(txdb)
  # Identificación de localizaciones que solapan con algún gen.
  overlap <- findOverlaps(query, subject)
  overlapDF <- data.frame(overlap@from, subject$gene_id[overlap@to])
  names(overlapDF) <- c("query", type)
  # Identificación de localizaciones que no solapan con ningún gen.
  n <- seq(along = chr)
  notOverlapDF <- data.frame(n[!(n %in% overlap@from)], NA)
  names(notOverlapDF) <- c("query", type)
  # Tabla de resultados.
  queryToGeneID <- rbind(overlapDF, notOverlapDF)
  queryToGeneID <- queryToGeneID[order(queryToGeneID$query), ]

  return(queryToGeneID)
}
```

Función `mirnaInfo`: Facilita el código MIMAT y el identificador de cada miRNA en la versión 21.0 de miRBase. También permite recuperar la secuencia de cada miRNA en la

versión que se indique.

```
mirnaInfo <- function(mirna, verSeq = 21, addSeq = TRUE) {
  # Args:
  #   mrina    Vector de tipo carácter con el nombre de los miRNAs.
  #   verSeq   Número entero indicando la versión de miRBase para
  #            recuperar las secuencias nucleotídicas.
  #   addSeq   Valor lógico: TRUE para incorporar las secuencias de
  #            nucleótidos en la salida de la función.
  # Return:
  #   Devuelve un data.frame en el cada miRNA va a asociado a un código
  #   MIMAT y el nombre del propio miRNA en la versión 21.0 de miRBase.
  #   También puede incluir la secuencia de nucleótidos del miRNA en la
  #   versión que se indique.

  require(MiRNANameConverter())
  # Obtención de información.
  nc <- MiRNANameConverter()
  info <- translateMiRNAName(nc, mirna, versions = c(verSeq, 21))
  # Selección de información.
  mirna <- info$input
  mimat <- info$mimat
  mirna.v21 <- info$v21.0
  selInfo <- data.frame(mirna, mimat, mirna.v21, stringsAsFactors = F)
  # Incorporación de las secuencias nucleotídicas si procede.
  if (addSeq) {
    selInfo$mirna.seq <- attr(info, "sequence")[, 3]
  }

  return(selInfo)
}
```

Función `extendCoord`: Proporciona coordenadas extendidas evitando el problema de que las coordenadas de inicio puedan resultar negativas o que las coordenadas de fin excedan la longitud del cromosoma.

```
extendCoord <- function(bsg, chr, start, end, strand = NA, up = 0, down = 0) {
  # Args:
  #   bsg              Objeto BSgenome.
  #   chr              Vector de tipo carácter para indicar el cromosoma;
  #                    ejemplo: "chr1".
  #   start, end       Vectores de números enteros para indicar posición
  #                    de inicio y posición de fin en el cromosoma.
  #   strand           Vector de tipo carácter para indicar hebra de ADN.
  #                    Solamente admite "+" y "-".
  #   up, down         Número de nucleótidos a incluir aguas arriba y
  #                    aguas abajo en las coordenadas facilitadas.
  # Return:
  #   Devuelve un data.frame de coordenadas extendidas. Si las coordenadas
```

```
  #   de inicio (start - up) resultan en un número negativo, se adopta el
  #   1 como posición de partida. Si las coordenadas de fin (end + down)
  #   exceden la longitud del cromosoma, se toma la longitud máxima de
  #   dicho cromosoma como posición de fin.

  require(BSgenome)
  # Errores.
  if (!all(chr %in% seqnames(bsg))) {
    stop("Some chromosomes are not listed in the BSgenome object")
  }
  if (any(c(start, end, up, end) < 0)) {
    stop("Parameters star, end, up and down must be positive numbers")
  }
  # Información sobre la longitud de los cromosomas.
  info <- as.data.frame(bsg@seqinfo)
  info$chr <- rownames(info)
  info <- left_join(data.frame(chr, stringsAsFactors = F), info, by = "chr")
  # Coordenadas de inicio extendidas.
  extStart <- start - up
  extStart[extStart <= 0] <- 1
  # Coordenadas de fin extendidas.
  extEnd <- end + down
  cond <- extEnd > info$seqlengths
  extEnd[cond] <- info$seqlengt[cond]
  # Avisos.
  if (any((start - up) <= 0)) {
    warning("Some extended start coordinates lead to a negative value")
  }
  if (any((end + down) > info$seqlengths)) {
    warning("Some extended end coordinates exceed the chromosome lenght")
  }
  # Conjunto de datos de salida.
  extdCoord <- data.frame(chr = chr, start = extStart,
                          end = extEnd, strand = strand)

  return(extdCoord)
}
```

Función `getCDNA`: Permite obtener las secuencias nucleotídicas de los cDNAs asociados a un grupo de transcritos.

```
getCDNA <- function(trc, start, end, mart, up = 0, down = 0) {
  # Args:
  #   trc              Vector de tipo carácter con los identificares
  #                    ensembl de los transcritos.
  #   start, end       Vectores de números enteros para indicar posición
  #                    de inicio y posición de fin. No son coordenadas
  #                    cromosómicas, indican posición dentro del cDNA.
```

```
#   mart              Objeto Mart asociado al genoma de interés.
#   up, down          Número de nucleótidos a incluir aguas arriba y
#                     aguas abajo en las coordenadas facilitadas.
# Return:
#   Devuelve un data.frame con las secuencias nucleotídicas de los cDNAs.
#   Si las coordenadas de inicio (start - up) resultan en un número
#   negativo, se adopta el 1 como posición de partida. Si las coordenadas
#   de fin (end + down) exceden la longitud del cDNA, se toma la longitud
#   máxima de dicho cDNA como posición de fin.

require(biomaRt)
# Errores.
if (any(c(start, end, up, end) < 0)) {
  stop("Parameters star, end, up and down must be positive numbers")
}
# Recuperación de las secuencias completas de los cDNAs.
seq <- getSequence(id = unique(trc), type = "ensembl_transcript_id",
                   seqType = "cdna", mart = mart)
# Asociación de cada transcrito con su secuencia cDNA.
info <- left_join(data.frame(trc, stringsAsFactors = F),
                  data.frame(seq),
                  by = c("trc" = "ensembl_transcript_id"))
# Longitud máxima de cada cDNA.
info$maxlen <- unlist(lapply(info$cdna, nchar))
# Coordenadas de inicio extendidas.
extStart <- start - up
extStart[extStart <= 0] <- 1
# Coordenadas de fin extendidas.
extEnd <- end + down
cond <- (extEnd > info$maxlen) & !is.na(info$maxlen)
extEnd[cond] <- info$maxlen[cond]
# Avisos.
if (any((start - up) <= 0)) {
  warning("Some extended start coordinates lead to a negative value")
}
if (any((end + down) > info$maxlen, nr.rm = TRUE)) {
  warning("Some extended start coordinates exceed the cDNA length")
}
# Acotación de las secuencias completas según coordenadas.
for (i in seq(along = info$cdna)) {
  if (is.na(info$cdna[i])) {
    next
  } else {
    bases <- strsplit(info$cdna[i], split = "")[[1]]
    info$cdna[i] <- paste(bases[extStart[i]:extEnd[i]], collapse = "")
  }
}
```

```
  info$start <- extStart
  info$end <- extEnd

  return(info)
}
```

10.2.2. Conversión de identificadores

Función `geneIDconverter`: Clausura para generar distintas funciones orientadas a la conversión de identificadores génicos.

```
geneIDconverter <- function(from, to) {
  # Args:
  #   from  Atributo biomaRt de partida.
  #   to    Atributo biomaRt de retorno.
  #
  # Return:
  #   Devuelve una función con los siguientes atributos:
  #      val   Valores del atributo biomaRt de partida.
  #      mart  Objeto Mart asociado al genoma de interés.

  require(biomaRt)

    function(val, mart) {
    attab <- getBM(attributes = c(from, to), filters = from,
                   values = val, mart = mart)
    toID <- attab[, to]
    names(toID) <- attab[, from]
    toID <- toID[as.character(val)]
    return(toID)
  }
}
```

Algunos conversores generados con `geneIDconverter`:

```
ensemblToEntrez <- geneIDconverter(from = "ensembl_gene_id", to = "entrezgene")
entrezToEnsembl <- geneIDconverter(from = "entrezgene", to = "ensembl_gene_id")
entrezToHgncSymbol <- geneIDconverter(from = "entrezgene", to = "hgnc_symbol")
entrezToMgiSymbol <- geneIDconverter(from = "entrezgene", to = "mgi_symbol")
```

10.2.3. Modificación o eliminación de registros

Función `inVersion21`: Indica si un miRNA está incluido en la versión 21.0 de miRBase.

```
inVersion21 <- function(mirna) {
  # Args:
  #   mirna  Vector de tipo carácter con el nombre de los miRNAs.
  #
  # Return:
  #   Devuelve un vector lógico que señala aquellos miRNAs incluidos
  #   en la versión 21.0 de miRBase.

  require(miRNAmeConverter)
  nc <- MiRNANameConverter()
  info <- translateMiRNAName(nc, mirna)
  included <- mirna %in% info$input

  return(included)
}
```

Función `oneMiRNAbyRow`: Genera un `data.frame` en el que cada fila solamente registra un miRNA.

```
oneMiRNAbyRow <- function(dataset, mirnaCol = "mirna", sep = ",") {
  # Args:
  #   dataset   Conjunto de datos en formato data.frame.
  #   mirnaCol  Nombre de la columna del dataset que contiene grupos de
  #             miRNAs formando una única cadena de texto.
  #   sep       Carácter que separa los miRNAs en el dataset de partida.
  #
  # Return:
  #   Devuelve un data.frame con la misma información que el dataset de
  #   partida, pero cada fila solamente registra un miRNA.

  # Conteo de miRNAs agrupados por fila.
  grouped <- dataset[, mirnaCol]
  rowCounter <- lapply(grouped, function(x) length(strsplit(x, sep)[[1]]))
  rowIndex <- rep(seq(along = grouped), rowCounter)
  # Separación de los miRNAs agrupados.
  isolated <- lapply(grouped, function(x) unlist(strsplit(x, sep)))
  # Nuevo dataset.
  dataset <- dataset[rowIndex, ]
  dataset[, mirnaCol] <- unlist(isolated)

  return(dataset)
}
```

Función `rmDissenting`: Elimina aquellos registros de la base de datos DIANA-TarBase en los que exista contrariedad respecto a la funcionalidad y al tipo de regulación de una pareja miRNA-gen.

```
rmDissenting <- function(dataset, geneId = "geneId", mirna = "mirna") {
  # Args:
  #   dataset   Conjunto de datos generado a partir de la base de datos
  #             DIANA-TarBase. Debe tratarse de una versión reducida con
  #             cuatro columnas: gen, miRNA, funcionalidad y regulación.
  #   geneId    Nombre de la columna para identificar los genes.
  #   mirna     Nombre de la columna para identificar los miRNAs.
  #
  # Return:
  #   Devuelve el mismo conjunto de datos una vez eliminados los registros
  #   en los que se encuentre información contradictoria.

  # Identificación parejas miRNA-gen con registros contradictorios.
  dataset <- dataset[!duplicated(dataset), ]
  pairs <- dataset[c(geneId, mirna)]
  dissentingPairs <- pairs[duplicated(pairs), ]
  # Eliminación de las parejas miRNA-gen identificadas.
  if (nrow(dissentingPairs) == 0) {
    return(dataset)
  } else {
    for (i in 1:nrow(dissentingPairs)) {
      remove <- (dataset[geneId] == dissentingPairs[i, geneId] &
                 dataset[mirna] == dissentingPairs[i, mirna])
      dataset <- dataset[!remove, ]
    }
    return(dataset)
  }
}
```

10.3. Datos CLIP-seq

10.3.1. CLEAR-CLIP

Carga de los datos:

```
humanCC <- read.delim("Data/CLEAR_CLIP_human.csv", stringsAsFactors = F)
mouseCC <- read.delim("Data/CLEAR_CLIP_mouse.csv", stringsAsFactors = F)
```

Selección de los campos o columnas de interés:

```
select <- c("chr", "start", "end", "strand", "gene.id", "gene.symbol", "miRNA")
humanCC <- subset(humanCC, select = select)
mouseCC <- subset(mouseCC, select = select)
```

Modificación del nombre de algunas columnas:

```
names(humanCC)[5:7] <- c("entrez", "symbol", "mirna")
names(mouseCC)[5:7] <- c("entrez", "symbol", "mirna")
```

Recuperación de las secuencias nucleotídicas de los *targets* a partir de las coordenadas genéticas presentes en el conjunto de datos original:

```
# Homo sapiens
hg18 <- BSgenome.Hsapiens.UCSC.hg18::Hsapiens
humanCC$target.seq <- getSeq(hg18, humanCC$chr, humanCC$start, humanCC$end,
                             strand = humanCC$strand, as.character = T)
# Mus musculus
mm9 <- BSgenome.Mmusculus.UCSC.mm9::Mmusculus
mouseCC$target.seq <- getSeq(mm9, mouseCC$chr, mouseCC$start, mouseCC$end,
                             strand = mouseCC$strand, as.character = T)
```

Recuperación de las secuencias nucleotídicas de los *targets* incorporando 100 nucleótidos aguas arriba y 100 aguas abajo siempre que sea posible:

```
# Homo sapiens
extdCoord <- extendCoord(
  hg18, humanCC$chr, humanCC$start, humanCC$end, humanCC$strand,
  up = 100, down = 100
)
humanCC$target.extdseq <- getSeq(
  hg18, extdCoord$chr, extdCoord$start, extdCoord$end,
  strand = extdCoord$strand, as.character = TRUE
)
humanCC$start.extdseq <- extdCoord$start
humanCC$end.extdseq <- extdCoord$end
# Mus musculus
extdCoord <- extendCoord(
  mm9, mouseCC$chr, mouseCC$start, mouseCC$end, mouseCC$strand,
  up = 100, down = 100
)
mouseCC$target.extdseq <- getSeq(
  mm9, extdCoord$chr, extdCoord$start, extdCoord$end,
  strand = extdCoord$strand, as.character = TRUE)
mouseCC$start.extdseq <- extdCoord$start
mouseCC$end.extdseq <- extdCoord$end
```

Selección del conjunto de datos BioMart según la versión del genoma citada en la publicación de los datos:

```
# Versión correspondiente a NCBI36/hg18
humanEnsembl54 <- useMart(biomart = 'ENSEMBL_MART_ENSEMBL',
                          dataset = 'hsapiens_gene_ensembl',
                          host = 'may2009.archive.ensembl.org')
# Versión correspondiente a NCBI37/mm9
mouseEnsembl67 <- useMart(biomart = "ENSEMBL_MART_ENSEMBL",
```

```
                              dataset = "mmusculus_gene_ensembl",
                              host = "may2012.archive.ensembl.org")
```

Recuperación de los identificadores Ensembl a partir de los identificadores Entrez proporcionados en los datos de partida:

```
humanCC$ensembl <- entrezToEnsembl(humanCC$entrez, humanEnsembl54)
mouseCC$ensembl <- entrezToEnsembl(mouseCC$entrez, mouseEnsembl67)
```

Identificación de la versión de miRBase a la que pertenecen los miRNAs de los datos:

```
nc <- MiRNANameConverter()
humanCC$mirbase <- assessVersion(nc, humanCC$mirna)$version[1]
mouseCC$mirbase <- assessVersion(nc, mouseCC$mirna)$version[1]
```

Eliminación de aquellos registros en los que el miRNA de la quimera haya sido eliminado de la versión 21 de miRBase:

```
humanCC <- humanCC[inVersion21(humanCC$mirna), ]
mouseCC <- mouseCC[inVersion21(mouseCC$mirna), ]
```

Recuperación del código de indentificación MIMAT, el nombre en la versión 21.0 de miRBase y la secuencia nucleotídica en la versión indicada:

```
humanCC <- left_join(humanCC, mirnaInfo(humanCC$mirna, humanCC$mirbase))
mouseCC <- left_join(mouseCC, mirnaInfo(mouseCC$mirna, mouseCC$mirbase))
```

Información o detalles adicionales:

```
# Homo sapiens
humanCC$publi <- "Moore et al. 2015"
humanCC$genome <- "NCBI36/hg18"
humanCC$method <- "CLEAR-CLIP"
# Mus musculus
mouseCC$publi <- "Moore et al. 2015"
mouseCC$genome <- "NCBI37/mm9"
mouseCC$method <- "CLEAR-CLIP"
```

10.3.2. iPAR-CLIP

Carga de los datos:

```
humanPC <- read.delim("Data/iPAR_CLIP_human.csv", stringsAsFactors = F)
mousePC <- read.delim("Data/iPAR_CLIP_mouse.csv", stringsAsFactors = F)
```

Selección de los campos o columnas de interés:

```
select <- c("chromosome", "start", "end", "strand", "miRNA.ID", "target.sequence")
humanPC <- subset(humanPC, select = select)
mousePC <- subset(mousePC, select = select)
```

Modificación del nombre de algunas columnas:

```
names(humanPC)[c(1, 5, 6)] <- c("chr", "mirna", "target.seq")
names(mousePC)[c(1, 5, 6)] <- c("chr", "mirna", "target.seq")
```

Modificación de las coordenadas de inicio de cada quimera; la secuencia nucleotídica de los *targets* comienza una base por delante de la posición indicada:

```
humanPC$start <- humanPC$start + 1
```

Recuperación de las secuencias nucleotídicas de los *targets* incorporando 100 nucleótidos aguas arriba y 100 aguas abajo siempre que sea posible:

```
# BSgenome
hg19 <- BSgenome.Hsapiens.UCSC.hg19::Hsapiens
mm9 <- BSgenome.Mmusculus.UCSC.mm9::Mmusculus
# Filtrado por cromosomas
humanPC <- humanPC[humanPC$chr %in% seqnames(hg19), ]
mousePC <- mousePC[mousePC$chr %in% seqnames(mm9), ]
# Homos sampiens
extdCoord <- extendCoord(
  hg19, humanPC$chr, humanPC$start, humanPC$end, humanPC$strand,
  up = 100, down = 100
)
humanPC$target.extdseq <- getSeq(
  hg19, extdCoord$chr, extdCoord$start, extdCoord$end,
  strand =  extdCoord$strand, as.character = TRUE
)
humanPC$star.extdseq <- extdCoord$start
humanPC$end.extdseq <- extdCoord$end
# Mus musculus
extdCoord <- extendCoord(
  mm9, mousePC$chr, mousePC$start, mousePC$end, mousePC$strand,
  up = 100, down = 100
)
mousePC$target.extdseq <- getSeq(
  mm9, extdCoord$chr, extdCoord$start, extdCoord$end,
  strand = extdCoord$strand, as.character = TRUE
)
mousePC$star.extdseq <- extdCoord$start
mousePC$end.extdseq <- extdCoord$end
```

Obtención de los identificadores Entrez asociados a los genes que solapan con las coordenadas genéticas registradas para cada interacción o quimera. Durante el proceso se emplean las anotaciones génicas generadas por UCSC en formato `TxDb`:

```
# Homo sapiens
humanCoordToGene <- coordToGeneID(TxDb.Hsapiens.UCSC.hg19.knownGene, humanPC$chr,
                                  humanPC$start, humanPC$end, humanPC$strand)
humanPC <- humanPC[humanCoordToGene$query, ]
```

```
humanPC$entrez <- humanCoordToGene$entrez
# Mus musculus
mouseCoordToGene <- coordToGeneID(TxDb.Mmusculus.UCSC.mm9.knownGene, mousePC$chr,
                                  mousePC$start, mousePC$end, mousePC$strand)
mousePC <- mousePC[mouseCoordToGene$query, ]
mousePC$entrez <- mouseCoordToGene$entrez
```

Selección del conjunto de datos de *BioMart* según la versión del genoma citada en la publicación de los datos:

```
# Versión correspondiente a GRCh37/hg19
humanEnsembl75 <- useMart(biomart = "ENSEMBL_MART_ENSEMBL",
                          dataset = "hsapiens_gene_ensembl",
                          host = "feb2014.archive.ensembl.org")
# Versión correspondiente a NCBI37/mm9
mouseEnsembl67 <- useMart(biomart = "ENSEMBL_MART_ENSEMBL",
                          dataset = "mmusculus_gene_ensembl",
                          host = "may2012.archive.ensembl.org")
```

Recuperación de los identificares Ensembl y el símbolo oficial de cada gen a partir de los identificares Entrez previamente recabados:

```
# Homo sapiens
humanPC$ensembl <- entrezToEnsembl(humanPC$entrez, humanEnsembl75)
humanPC$symbol <- entrezToHgncSymbol(humanPC$entrez, humanEnsembl75)
# Mus musculus
mousePC$ensembl <- entrezToEnsembl(mousePC$entrez, mouseEnsembl67)
mousePC$symbol <- entrezToMgiSymbol(mousePC$entrez, mouseEnsembl67)
```

Modificación de la estructura del conjunto de datos de forma que cada fila o registro solamente contenga un miRNA:

```
humanPC <- oneMiRNAbyRow(humanPC)
mousePC <- oneMiRNAbyRow(mousePC)
```

Identificación de la versión de miRBase a la que pertenecen los miRNAs de los datos:

```
nc <- MiRNANameConverter()
humanPC$mirbase <- assessVersion(nc, humanPC$mirna)$version[1]
mousePC$mirbase <- assessVersion(nc, mousePC$mirna)$version[1]
```

Eliminación de aquellos registros en los que el miRNA de la quimera haya sido eliminado de la versión 21 de miRBase:

```
humanPC <- humanPC[inVersion21(humanPC$mirna), ]
mousePC <- mousePC[inVersion21(mousePC$mirna), ]
```

Recuperación del código de indentificación MIMAT, el nombre en la versión 21.0 de miRBase y la secuencia nucleotídica en la versión indicada:

```
humanPC <- left_join(humanPC, mirnaInfo(humanPC$mirna, humanPC$mirbase))
mousePC <- left_join(mousePC, mirnaInfo(mousePC$mirna, mousePC$mirbase))
```

Información o detalles adicionales:

```
# Homo sapiens
humanPC$publi <- "Grosswendt et al. 2014"
humanPC$genome <- "GRCh37/hg19"
humanPC$method <- "iPAR-CLIP"
# Mus musculus
mousePC$publi <- "Grosswendt et al. 2014"
mousePC$genome <- "NCBI37/mm9"
mousePC$method <- "iPAR-CLIP"
```

10.3.3. CLASH

Carga de los datos:

```
humanCL <- read.delim("Data/CLASH_human.csv", stringsAsFactors = F)
```

Selección de los campos o columnas de interés:

```
select <- c("microRNA_name", "miRNA_seq", "mRNA_name", "mRNA_start",
            "mRNA_end_extended", "mRNA_seq_extended")
humanCL <- subset(humanCL, select = select)
```

Modificación del nombre de las columnas:

```
colNames <- c("mirna", "mirna.seq", "symbol", "cdna.start", "cdna.end",
              "target.seq")
colnames(humanCL) <- colNames
```

Manipulación de cadenas de texto para extraer información presente en los datos de partida:

```
# miRNAs
humanCL$mirna <- gsub(".*Base_|_micro.*", "", humanCL$mirna)
humanCL$mirna <- paste("hsa-", humanCL$mirna, sep = "")
# Identificadores Ensembl
humanCL$ensembl <- gsub("_.*", "", humanCL$symbol)
humanCL$transcript <- substring(humanCL$symbol, 17, 31)
# Símbolo asociado a cada gen
humanCL$symbol <- gsub("_mRNA", "", humanCL$symbol)
humanCL$symbol <- gsub(".*_", "", humanCL$symbol)
```

Selección del conjunto de datos de *BioMart*; la versión Ensembl 60 no está disponible por lo que se emplea la versión Ensembl 75, ambas corresponden a la misma versión del genoma:

```
# Versión correspondiente a GRCh37/hg19
humanEnsembl75 <- useMart(biomart = "ENSEMBL_MART_ENSEMBL",
                          dataset = "hsapiens_gene_ensembl",
                          host = 'feb2014.archive.ensembl.org')
```

Obtención de los identificares Entrez a partir de los identificares Ensembl disponibles:

```
humanCL$entrez <- ensemblToEntrez(humanCL$ensembl, humanEnsembl75)
```

Recuperación de las secuencias nucleotídicas de los *targets* incorporando 100 nucleótidos aguas arriba y 100 aguas abajo siempre que sea posible:

```
extdCDNA <- getCDNA(humanCL$transcript, humanCL$cdna.start,
                    humanCL$cdna.end, mart = humanEnsembl75,
                    up = 100, down = 100)
humanCL$target.extdseq <- extdCDNA$cdna
humanCL$cdna.extdstart <- extdCDNA$start
humanCL$cdna.extdend <- extdCDNA$end
```

Debido a que se emplea una versión del genoma que no es exactamente igual a la versión Ensembl 60, algunas secuencias extendidas no se pueden recuperar:

```
id <- is.na(humanCL$target.extdseq) | grepl("NANANA", humanCL$target.extdseq)
humanCL$target.extdseq[id] <- humanCL$target.seq
humanCL$cdna.extdstart[id] <- humanCL$cdna.start
humanCL$cdna.extdend[id] <- humanCL$cdna.end
```

Identificación de la versión de miRBase a la que pertenecen los miRNAs de los datos:

```
nc <- MiRNANameConverter()
humanCL$mirbase <- assessVersion(nc, humanCL$mirna)$version[1]
```

Eliminación de aquellos registros en los que el miRNA de la quimera haya sido eliminado de la versión 21 de miRBase:

```
humanCL <- humanCL[inVersion21(humanCL$mirna), ]
```

Recuperación del código de indentificación MIMAT, el nombre en la versión 21.0 de miRBase y la secuencia nucleotídica en la versión indicada:

```
humanCL <- left_join(humanCL, mirnaInfo(humanCL$mirna, humanCL$mirbase, F))
```

Información o detalles adicionales:

```
humanCL$publi <- "Helwak et al. 2014"
humanCL$genome <- "GRCh37/hg19"
humanCL$method <- "CLASH"
```

10.4. Datos DIANA-TarBase

Carga de los datos:

```
tarbase <- read.delim("Data/tarbase_data.csv", stringsAsFactors = F)
```

Eliminación de las interacciones miRNA-gen cuya regulación sea desconocida:

```
tarbase <- tarbase[!(is.na(tarbase$up_down) | tarbase$up_down == "UNKNOWN"), ]
```

División de la base de datos según especie:

```
tarbaseHuman <- tarbase[tarbase$species == "Homo sapiens", ]
tarbaseMouse <- tarbase[tarbase$species == "Mus musculus", ]
```

Selección de los campos o columnas de interés:

```
select <- c("geneId", "mirna", "positive_negative", "up_down")
tarbaseHuman <- subset(tarbaseHuman, select = select)
tarbaseMouse <- subset(tarbaseMouse, select = select)
```

Eliminación de aquellas interacciones miRNA-gen que registren discrepancias respecto a su funcionalidad o tipo de regulación:

```
tarbaseHuman <- rmDissenting(tarbaseHuman)
tarbaseMouse <- rmDissenting(tarbaseMouse)
```

Identificación de la versión de miRBase a la que pertenecen los miRNAs de los datos:

```
nc <- MiRNANameConverter()
head(assessVersion(nc, tarbase$mirna))
```

Modificación del nombre de las columnas:

```
colNames <- c("ensembl", "mirna.v21", "functionality", "regulation")
colnames(tarbaseHuman) <- colNames
colnames(tarbaseMouse) <- colNames
```

10.5. Cruce de bases de datos

Combinación de datos por especie:

```
humanData <- rbind.fill(humanCC, humanPC, humanCL)
mouseData <- rbind.fill(mouseCC, mousePC)
```

Identificación de cada registro en el nuevo conjunto de datos mediante un código:

```
humanData$record.id <- paste0("human", sprintf("%0.5d", 1:nrow(humanData)))
mouseData$record.id <- paste0("mouse", sprintf("%0.6d", 1:nrow(mouseData)))
```

Establecimiento de un criterio único para las secuencias nucleotídicas de los miRNAs; todas ellas se presentan en formato DNA, con “T” en lugar de “U”:

```
humanData$mirna.seq <- gsub("U", "T", humanData$mirna.seq)
mouseData$mirna.seq <- gsub("U", "T", mouseData$mirna.seq)
```

Cruce de datos con DIANA-TarBase por especie según el identificador Ensembl de cada gen y el nombre del miRNA con el que interacciona (v21.0 miRBase):

```
humanData <- left_join(humanData, tarbaseHuman)
mouseData <- left_join(mouseData, tarbaseMouse)
```

Reordenamiento de las columnas:

```
colNames <- c("record.id", "chr", "start", "end", "strand",
              "start.extdseq", "end.extdseq", "transcript",
              "cdna.start", "cdna.end", "cdna.extdstart", "cdna.extdend",
              "entrez", "ensembl", "symbol", "mimat", "mirna", "mirna.v21",
              "target.seq", "target.extdseq", "mirna.seq", "functionality",
              "regulation", "publi", "genome", "mirbase", "method")

humanData <- humanData[, colNames]
mouseData <- mouseData[, colNames[- c(8:12)]]
```

Almacenamiento de los datos generados:

```
write.table(humanData, "ProcessedData/human_data.txt", sep = "\t",
            row.names = FALSE)
write.table(mouseData, "ProcessedData/mouse_data.txt", sep = "\t",
            row.names = FALSE)
```

Cada conjunto de datos registra las siguientes columnas:

- record: Identificador de cada registro; propio de este proyecto.
- chr: Cromosoma.
- start: Posición de inicio en el cromosoma.
- end: Posición de final en el cromosoma.
- strand: Hebra del cromosoma.
- start.extdseq: Posición de inicio en el cromosoma de target.extdseq.
- end.extdseq: Posición de final en el cromosoma de target.extdseq.
- transcript*: Transcrito del mRNA de la quimera.
- cdna.start*: Posición de inicio en el cDNA del transcrito.
- cdna.end*: Posición de fin en el cDNA del transcrito.
- cdna.extdstart*: Posición de inicio en el cDNA de target.extdseq.
- cdna.extdend*: Posición de fin en el cDNA de target.extdseq.
- entrez: Identificador Entrez del gen que codifica el mRNA de la quimera.
- ensembl: Identificador Ensembl del gen que codifica el mRNA de la quimera.
- symbol: Símbolo oficial asociado al gen que codifica el mRNA de la quimera.
- mimat: Código MIMAT del miRNA de la quimera.

- mirna: Identificador miRBase del miRNA presente en los datos de partida.
- mirna.v21: Identificador miRBase del miRNA en la versión 21.
- target.seq: Secuencia nucleotídica del mRNA de la quimera.
- target.extdseq: Secuencia nucleotídica extendida (100nt + start... end + 100nt).
- mirna.seq: Secuencia nucleotídica del miRNA de la quiemra.
- functionality: Funcionalidad de la interacción miRNA-mRNA.
- regulation: Tipo de regulación de la interacción miRNA-mRNA.
- publi: Publicación que registra la quimera.
- genome: Versión del genoma empleada para el mapeo de la quimera.
- method: Método empleado para la obtención de la quimera.
- mirbase: Versión miRBase empleada en los datos de partida.

* Solamente en el conjunto de datos para *Homo sapiens*.

11. Anexo III: Caracterización de las interacciones miRNA-diana

11.1. Paquetes

```
if (!require(dplyr)) {
  install.packages("dplyr")
}
if (!require(seqinr)) {
  install.packages("seqinr")
}
if (!require(stringr)){
  install.packages("stringr")
}
if(!require(parallel)){
  install.packages("parallel")
}
```

Paquete desarrollado durante el proyecto:

```
require(cmirti)
```

11.2. ViennaRNA

Carga de los datos procesados:

```
humanData <- read.delim("ProcessedData/human_data.txt", stringsAsFactors = F)
mouseData <- read.delim("ProcessedData/mouse_data.txt", stringsAsFactors = F)
```

Adecuación de las secuencias nucleotídicas de las quimeras para que puedan ser procesadas por el paquete ViennaRNA:

```
# Secuencias
humanSeq <- paste(humanData$mirna.seq, humanData$target.seq, sep = "&")
humanSeq <- as.list(humanSeq)
mouseSeq <- paste(mouseData$mirna.seq, mouseData$target.seq, sep = "&")
mouseSeq <- as.list(mouseSeq)
# Secuencias extendidas
humanExtdSeq <- paste(humanData$mirna.seq, humanData$target.extdseq, sep = "&")
humanExtdSeq <- as.list(humanExtdSeq)
mouseExtdSeq <- paste(mouseData$mirna.seq, mouseData$target.extdseq, sep = "&")
mouseExtdSeq <- as.list(mouseExtdSeq)
```

Almacenamiento de las secuencias en formato FASTA:

```
# Secuencias
write.fasta(sequences = humanSeq, names = humanData$record.id,
            file.out = "ProcessedData/human_sequences.fa")
write.fasta(sequences = mouseSeq, names = mouseData$record.id,
            file.out = "ProcessedData/mouse_sequences.fa")
# Secuencias extendidas
write.fasta(sequences = humanExtdSeq, names = humanData$record.id,
            file.out = "ProcessedData/human_extd_sequences.fa")
write.fasta(sequences = mouseExtdSeq, names = mouseData$record.id,
            file.out = "ProcessedData/mouse_extd_sequences.fa")
```

Ejecución del programa `RNAcofold` en la terminal de Unix:

```
cd ProcessedData
# Homo sapiens
in=human_sequences.fa; out=human_cofold.txt
RNAcofold -p -d2 --noLP --noPS --output-format 'D' --csv-delim $'\t' < $in > $out
for file in *dp.ps; do rm $file; done
# Mus musculus
in=mouse_sequences.fa; out=mouse_cofold.txt
RNAcofold -p -d2 --noLP --noPS --output-format 'D' --csv-delim $'\t' < $in > $out
for file in *dp.ps; do rm $file; done
```

Carga de los datos generados con el programa `RNAcofold`:

```
humanCofold <- read.delim("ProcessedData/human_cofold.txt", header = F, skip = 1,
                          stringsAsFactors = F)
mouseCofold <- read.delim("ProcessedData/mouse_cofold.txt", header = F, skip = 1,
                          stringsAsFactors = F)
```

Asignación del nombre de las columnas de los objetos `humanCofold` y `mouseCofold`:

```
colNames <- c("row", "record.id", "seq", "dotbracket", "MFE",
              "ens.dotbracket", "ens.FE", "ens.MFE.freq", "dG")
colnames(humanCofold) <- colNames
colnames(mouseCofold) <- colNames
```

Ejecución del programa `RNAup` en la terminal de Unix:

```
cd ProcessedData
# Homo sapiens
in=human_extd_sequences.fa; out=human_rnaup.txt
RNAup -b -d2 --noLP -o -c 'S' --interaction_pairwise < $in > $out
# Mus musculues
in=mouse_extd_sequences.fa; out=mouse_rnaup.txt
RNAup -b -d2 --noLP -o -c 'S' --interaction_pairwise < $in > $out
```

Carga de los datos generados con el programa `RNAup`:

```
humanRNAup <- readLines("ProcessedData/human_rnaup.txt")
mouseRNAup <- readLines("ProcessedData/mouse_rnaup.txt")
```

Selección de atributos numéricos mediante la manipulación de las cadenas de texto:

```
linePattern <- "[\\)|\\.|\\(]&"
featPattern <- "\\d+|[\\(\\-\\ ]\\d+\\.\\d+"
# Homo sapiens
humanRNAup <- humanRNAup[grepl(linePattern, humanRNAup)]
humanRNAup <- str_extract_all(humanRNAup, featPattern, simplify = T)
humanRNAup <- gsub(pattern = "\\(", replacement = "", humanRNAup)
humanRNAup <- apply(humanRNAup, MARGIN = 1, FUN = as.numeric)
humanRNAup <- data.frame(t(humanRNAup))
# Mus muculus
mouseRNAup <- mouseRNAup[grepl(linePattern, mouseRNAup)]
mouseRNAup <- str_extract_all(mouseRNAup, featPattern, simplify = T)
mouseRNAup <- gsub(pattern = "\\(", replacement = "", mouseRNAup)
mouseRNAup <- apply(mouseRNAup, MARGIN = 1, FUN = as.numeric)
mouseRNAup <- data.frame(t(mouseRNAup))
```

Asignación del nombre de las columnas de los objetos `humanRNAup` y `mouseRNAup`:

```
colNames <- c("i", "j", "k", "l", "dG", "dGint", "dGu_l")
colnames(humanRNAup) <- colNames
colnames(mouseRNAup) <- colNames
```

11.3. Extracción de atributos

Detección del número de núcleos del procesador para la paralelización de los cómputos:

```
cores <- detectCores()
```

Generación de una esquema o cartografía para cada quimera miRNA-diana. A diferencia de la notación dot-bracket, los esquemas generados permiten distinguir los distintos elementos de la interacción con mayor facilidad (| = Watson-Crick, **!** = G-U *wobble*, **[** = *hairpin*, **.** = *mismatch*):

```
# Homo sapiens
humanDuplex <- mcmapply(getDuplexMap, humanCofold$seq, humanCofold$dotbracket,
                        USE.NAMES = FALSE, mc.cores = cores)
humanDuplex <- data.frame(t(humanDuplex), stringsAsFactors = FALSE)
# Mus musculus
mouseDuplex<- mcmapply(getDuplexMap, mouseCofold$seq, mouseCofold$dotbracket,
                           USE.NAMES = FALSE, mc.cores = cores)
mouseDuplex <- data.frame(t(mouseDuplex), stringsAsFactors = FALSE)
```

Modificaciones del nombre de las columnas de los objetos `humanDuplex` y `mouseDuplex`:

```
colNames <- c("mir.seq", "mir.map", "tar.seq", "tar.map")
colnames(humanDuplex) <- colNames
colnames(mouseDuplex) <- colNames
```

Adición de algunos atributos calculados con el paquete `ViennaRNA`:

```
# Homo sapiens
humanDuplex$record.id <- humanCofold$record
humanDuplex$MFE       <- humanCofold$MFE
humanDuplex$dG.cofold <- humanCofold$dG
humanDuplex$dG.rnaup  <- humanRNAup$dG
humanDuplex$dG.int    <- humanRNAup$dGint
humanDuplex$dGu.tar   <- humanRNAup$dGu_l
# Mus musculus
mouseDuplex$record.id <- mouseCofold$record
mouseDuplex$MFE       <- mouseCofold$MFE
mouseDuplex$dG.cofold <- mouseCofold$dG
mouseDuplex$dG.rnaup  <- mouseRNAup$dG
mouseDuplex$dG.int    <- mouseRNAup$dGint
mouseDuplex$dGu.tar   <- mouseRNAup$dGu_l
```

Identificación del tipo de interacción miRNA-mRNA de cada quimera:

```
humanSite <- mcmapply(getInteracType, humanDuplex$mir.map, humanDuplex$tar.map,
                      humanDuplex$tar.seq, USE.NAMES = FALSE, mc.cores = cores)
mouseSite <- mcmapply(getInteracType, mouseDuplex$mir.map, mouseDuplex$tar.map,
                      mouseDuplex$tar.seq, USE.NAMES = FALSE, mc.cores = cores)
```

Eliminación de aquellos registros en los que no exista interacción miRNA-mRNA o no esté definida; se requieren al menos dos pares de bases apareadas según el modelo de Watson y Crick:

```
humanSiteCond <- humanSite != "no_binding" & humanSite != "not_defined"
humanDuplex <- humanDuplex[humanSiteCond, ]
mouseSiteCond <- mouseSite != "no_binding" & mouseSite != "not_defined"
mouseDuplex <- mouseDuplex[mouseSiteCond, ]
```

Cálculo de atributos relacionados con el número y la distancia entre pares de bases:

```
# Homo sapiens
humanBPfeat <- mcmapply(getDuplexBPfeat, humanDuplex$mir.map, humanDuplex$tar.map,
                        USE.NAMES = FALSE, mc.cores = cores)
humanBPfeat <- data.frame(t(humanBPfeat))
# Mus musculus
mouseBPfeat <- mcmapply(getDuplexBPfeat, mouseDuplex$mir.map, mouseDuplex$tar.map,
                        USE.NAMES = FALSE, mc.cores = cores)
mouseBPfeat <- data.frame(t(mouseBPfeat))
```

Cálculo de atributos relacionados con la estructura secundaria del acoplamiento miRNA-diana:

```
# Homo sapiens
humanSSfeat <- mcmapply(getDuplexSSfeat, humanDuplex$mir.map, humanDuplex$tar.map,
                        USE.NAMES = FALSE, mc.cores = cores)
humanSSfeat <- data.frame(t(humanSSfeat))
# Mus musculus
mouseSSfeat <- mcmapply(getDuplexSSfeat, mouseDuplex$mir.map, mouseDuplex$tar.map,
                        USE.NAMES = FALSE, mc.cores = cores)
mouseSSfeat <- data.frame(t(mouseSSfeat))
```

Cálculo de atributos relacionados con la composición nucleotídica del miRNA de cada quimera:

```
# Homo sapiens
humanMirNtComp <- mcmapply(getSeqBaseComp, humanDuplex$mir.seq,
                           USE.NAMES = FALSE, mc.cores = cores)
humanMirNtComp <- data.frame(t(humanMirNtComp))
# Mus musculus
mouseMirNtComp <- mcmapply(getSeqBaseComp, mouseDuplex$mir.seq,
                           USE.NAMES = FALSE, mc.cores = cores)
mouseMirNtComp <- data.frame(t(mouseMirNtComp))
```

Modificación del nombre de las columnas de los objetos `humanMirNtComp` y `mouseMirNtComp`:

```
colNames <- paste("mir", colnames(humanMirNtComp), sep = ".")
colnames(humanMirNtComp) <- colNames
colnames(mouseMirNtComp) <- colNames
```

Combinación de los atributos:

```
# Homo sapiens
humanNumfeat <- cbind(humanDuplex, humanBPfeat, humanSSfeat, humanMirNtComp)
humanCatfeat <- cbind(humanData[, c("record.id", "functionality", "regulation")],
                      interact.type = humanSite)
humanFeatures <- left_join(humanNumfeat, humanCatfeat)
# Mus musculus
mouseNumfeat <- cbind(mouseDuplex, mouseBPfeat, mouseSSfeat, mouseMirNtComp)
mouseCatfeat <- cbind(mouseData[, c("record.id", "functionality", "regulation")],
                      interact.type = mouseSite)
mouseFeatures <- left_join(mouseNumfeat, mouseCatfeat)
```

Reordenamiento de las columnas de los objetos `humanFeatures` y `mouseFeatrues`:

```
humanFeatures <- humanFeatures[, c(5, 102, 103, 1:4, 104, 6:101)]
mouseFeatures <- mouseFeatures[, c(5, 102, 103, 1:4, 104, 6:101)]
```

Almacenamiento de los datos generados:

```
write.table(humanFeatures, "ProcessedData/human_features.txt", sep = "\t",
            row.names = FALSE)
```

```
write.table(mouseFeatures, "ProcessedData/mouse_features.txt", sep = "\t",
            row.names = FALSE)
```

11.4. Lista de atributos extraídos

Atributos extraídos con el paquete `ViennRNA`:

- Mínima energía libre calculado con `RNAcofold` (`MFE`).
- ΔG de la unión del heterodímero calculada con `RNAcofold` (`dG.cofold`).
- ΔG de la unión calculada con `RNAup` (`dG.rnaup` = `dG.int` + `dGu.tar`).
- Energía de la interacción (`dG.int`).
- Energía de apertura de la diana (`dGu.tar`).

Atributos relacionados con la cantidad y disposición de los pares de bases que forman la interacción miRNA-mRNA:

- Número total de pares de bases Watson-Crick (`duplex.WC`).
- Número total de pares de bases GU o *wobbles* (`duplex.GU`).
- Número total de pares de bases, BP = WC + GU (`duplex.BP`).
- Número de pb WC dentro y fuera de la región semilla (`mir.WC.seed` y `mir.WC.outseed`).
- Número de pb GU dentro y fuera de la región semilla (`mir.GU.seed` y `mir.GU.outseed`).
- Número de pb dentro y fuera de la región semilla (`mir.BP.seed` y `mir.BP.outseed`).
- Numero total de pares de bases que interaccionan intramolecularmente (`mir.IntramolBP`).

En relación a pares de bases en general (BP) y pares de bases Watson-Crick (WC):

- Posición del primer par de bases (ej: `mir.WC.1stpos`).
- Posición del último par de bases (ej: `mir.WC.1stpos`).
- Distancia mínima registrada entre pares de bases correlativos (ej: `mir.WC.mindist`).
- Distancia media entre pares de bases correlativas (ej: `mir.WC.avgdist`).
- Distancia máxima resitrada entre pares de bases correlativos (ej: `mir.WC.maxdist`).

En relación a regiones de pares de bases consecutivos (CBP) y pares de bases Watson-Crick consecutivos (CWC):

- Número de regiones de pares de bases consecutivos (ej: `mir.CBP.numreg`).
- Longitud de la región de menor tamaño (ej: `mir.CBP.minlen`).
- Longitud media de las regiones (ej: `mir.CBP.avglen`).
- Longitud de la región de mayor tamaño (ej: `mir.CBP.maxlen`).
- Longitud de la región de mayor tamaño dentro de la región semilla (ej: `mir.CBP.seed.maxlen`).
- Longitud de la región de mayor tamaño fuera de la región semilla (ej: `mir.CBP.outseed.maxlen`).

En relación al mRNA:

- Distancia entre el primer y último par de bases (`tar.MRE.lenght`).
- Distancia media entre pares de bases correlativos (`tar.BP.avgdist`).
- Distancia máxima entre pares de bases correlativos (`tar.BP.maxdist`).
- Número de regiones de pares de bases consecutivos (ej: `tar.CBP.numreg`).
- Longitud de la región de menor tamaño (ej: `tar.CBP.minlen`).
- Longitud media de las regiones (ej: `tar.CBP.avglen`).
- Longitud de la región de mayor tamaño (ej: `tar.CBP.maxlen`).
- Numero total de pares de bases que interaccionan intramolecularmente, solamente se consideran los pares comprendidos entre el primer y último par de bases con interacción intermolecular (`tar.IntramolBP`).

Atributos relacionados con la estructura secundaria de cada dímero miRNA-mRNA:

- Número total de protuberancias en el miRNA (`mir.Bulge.num`).
- Número total de protuberancias en el mRNA (`tar.Bulge.num`).
- Número total de bucles simétricos en el dímero (`duplex.SymLoop.num`).
- Número total de bucles asimétricos en el dímero (`duplex.AsymLoop.num`).
- Número total de 3WJ (*three way junction*) con horquilla en el miRNA (`mir.3WJ.num`).
- Número total de 3WJ (*three way junction*) con horquilla en el mRNA (`tar.3WJ.num`).
- Número total de multiWJ o bucle con al menos dos horquillas (`duplex.multiWJ.num`).

Para los elementos estructurales protuberancia, bucle simétrico y bucle asimétrico:

- Longitud de la formación más pequeña para cada elemento estructural (ej: `mir.Bulge.minlen`).
- Longitud o tamaño medio de cada elemento estructural (ej: `mir.Bulge.avglen`).
- Longitud de la formación más grande para cada elemento estructural (ej: `mir.Bulge.maxlen`).
- Longitud acumulada de todas las formaciones de un mismo elemento estructural (ej: `mir.Bulge.sumlen`).
- Para los bucles asimétricos los atributos se calculan tanto en el miRNA como en el mRNA (ej: `mir.AsymLoop.minlen` y `tar.AsymLoop.minlen`).

Atributos relacionados con la composición nucleotídica de cada miRNA:

- Proporción de nucleótidos dentro de la secuencia (`mir.X`; `X` $\in A, T, C, G$).
- Contenido AT y GC (`mir.weak` y `mir.strong`).
- Proporción de bases púricas y pirimidínicas (`mir.purine` y `mir.pyrimidine`).
- Proporción de dinucleótidos dentro de la secuencia (`mir.XY`; `X,Y` $\in A, T, C, G$).

Atributos de tipo categórico:

- Funcionalidad de la interacción miRNA-diana (`functionality`).
- Regulación de la interacción miRNA-diana (`regulation`).
- Tipo de interacción miRNA-diana (`interact.type`).

12. Anexo IV: Análisis de los datos

12.1. Paquetes

```
if (!require(fastICA)) {
  install.packages("fastICA")
}
if (!require(cluster)) {
  install.packages("cluster")
}
if (!require(mclust)) {
  install.packages("mclust")
}
if (!require(dbscan)) {
  install.packages("dbscan")
}
if (!require(caret)) {
  install.packages("caret")
}
if (!require(effsize)) {
  install.packages("effsize")
}
if (!require(factoextra)) {
  install.packages("factoextra")
}
if (!require(reshape2)) {
  install.packages("reshape2")
}
if (!require(ggplot2)) {
  install.packages("ggplot2")
}
if (!require(ggsci)) {
  install.packages("ggsci")
}
if (!require(ggpubr)) {
  install.packages("ggpubr")
}
if (!require(ggiraphExtra)) {
  install.packages("ggiraphExtra")
}
if (!require(plotrix)) {
  install.packages("plotrix")
}
```

12.2. Funciones

12.2.1. Tratamiento y análisis de datos

Función `normalize`: Escala los valores de una variable en el rango $[0, 1]$.

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

Función `numOfPC`: Permite escoger el número de componentes principales según el porcentaje de la varianza total explicada.

```
numOfPC <- function(pca, cut) {
  pcaSummary <- summary(pca)
  importance <- t(pcaSummary$importance)
  return(which(importance[, 3] >= cut)[1])
}
```

Función `claraMultiK`: Sobre un mismo conjunto de datos, permite aplicar el algoritmo de clustering CLARA evaluando distintos valores de `k`.

```
claraMultiK <- function(x, k, ...) {
  claraList <- vector(mode = "list", length = length(k))
  for (i in seq(along = claraList)) {
    claraList[[i]] <- clara(x, k[i], ...)
  }
  names(claraList) <- paste("k", k, sep = "")
  return(claraList)
}
```

Función `applytTest`: Aplica la prueba t de Student a todas las variables numéricas de un data.frame en función de una variable categórica de dos clases.

```
applytTest <- function(x, y) {
  pVal <- rep(NA, ncol(x))
  for (i in 1:ncol(x)) {
    test <- t.test(x[y, i], x[!y, i])
    pVal[i] <- test$p.value
  }
  names(pVal) <- colnames(x)
  return(pVal)
}
```

Función `applyCohenD`: Calcula el estadístico d de Cohen sobre todas las variables numéricas de un data.frame en función de una variable categórica de dos clases.

```
applyCohenD <- function(x, y) {
  d <- rep(NA, ncol(x))
  for (i in 1:ncol(x)) {
```

```
    test <- cohen.d(x[y, i], x[!y, i])
    d[i] <- test$estimate
  }
  names(d) <- colnames(x)
  return(abs(d))
}
```

12.2.2. Generación de gráficos

Función `plotPCAsummary`: Genera un gráfico que permite visualizar el porcentaje de varianza total explicada en función del número de componentes principales.

```
plotPCAsummary <- function(pca, numPC = NULL, title = "", subtitle = "",
                           fill = "#00468BFF", col = "#ED0000FF", ...) {

  # Porcentaje de varianza explicada por número de componentes
  pcaSummary <- summary(pca)
  if (is.null(numPC)) {
    pcaSummary <- t(pcaSummary$importance)
  } else {
    pcaSummary <- t(pcaSummary$importance)[1:numPC, ]
  }
  pcaSummary <- as.data.frame(pcaSummary)
  pcaSummary$PC <- 1:nrow(pcaSummary)
  colnames(pcaSummary) <- c("sdev", "propvar", "cumprop", "PC")
  # Escala de colores
  scalecolor <- c("Variance Proportion" = fill, "Cumulative Proportion" = col)
  # Generación del gráfico con ggplot2
  g <- ggplot(pcaSummary)
  g <- g + geom_bar(
    aes(x = PC, y = propvar, fill = "Variance Proportion"), stat = "identity"
  )
  g <- g + geom_step(aes(x = PC, y = cumprop, colour = "Cumulative Proportion"))
  g <- g + scale_fill_manual(name = "", values = fill)
  g <- g + scale_color_manual(name = "", values = scalecolor)
  g <- g + labs(y = "Explained Variance", x = "Principal Components")
  g <- g + ggtitle(title, subtitle = subtitle)
  g <- g + theme_bw(base_size = 8)
  g <- g + theme(...)
  g
}
```

Función `plotAvgSilWidth`: Elabora un gráfico donde se representa el valor de la anchura media de la silueta en función del número de *clusters*.

```
plotAvgSilWidth <- function(claraList, title = "", subtitle = "", ...) {
```

```
  # Anchura media de la silueta
  avgSil <- lapply(
    claraList, function(data) lapply(data, function(k) k$silinfo$avg.width)
  )
  avgSil <- melt(avgSil)
  colnames(avgSil) <- c("value", "k", "Data")
  avgSil$k <- as.numeric(gsub("k", "", avgSil$k))
  # Generación del gráfico con ggplot2
  g <- ggplot(avgSil, aes(x = k, y = value, shape = Data, color = Data))
  g <- g + geom_line(linetype = "dashed", size = .5)
  g <- g + geom_point(size = 2)
  g <- g + labs(y = "Average Silhouette Width", x = "Clusters")
  g <- g + ggtitle(title, subtitle = subtitle)
  g <- g + scale_color_uchicago()
  g <- g + theme_bw(base_size = 8)
  g <- g + theme(...)
  g
}
```

Función `plotSilhouette`: Genera un gráfico con la silueta de un determinado agrupamiento.

```
plotSilhouette <- function(x, title = "", ...) {

  # Silueta del agrupamiento
  avgSil <- round(x$silinfo$avg.width, 3)
  # Subtítulo del gráfico
  subtitle <- paste("Average silhouette width:", avgSil, sep = " ")
  # Generación del gráfico con ggplot2
  g <- fviz_silhouette(x, ggtheme = theme_bw(base_size = 8), print.summary = F)
  g <- g + ggtitle(title, subtitle = subtitle)
  g <- g + scale_color_lancet()
  g <- g + scale_fill_lancet()
  g <- g + theme(...)
  g
}
```

Función `tableToBarplot`: Representa mediante un gráfico de barras la información contenida en una tabla de frecuencias.

```
tableToBarPlot <- function(table, xlab = "", ylab = "", title = "",
                           subtitle = "", ...) {

  # Tabla de frecuencias
  table <- as.data.frame(table)
  table$Freq <- round(table$Freq, 2)
  # Nombre de las columnas de la tabla
  colnames(table)[1] <- "Var2"
```

```
  colnames(table)[2] <- "Var1"
  # Generación del gráfico con ggplot2
  g <- ggplot(table, aes(x = Var1, y = Freq, fill = Var2))
  g <- g + geom_bar(stat = "identity")
  g <- g + labs(y = ylab, x = xlab, fill = "")
  g <- g + ggtitle(title, subtitle = subtitle)
  g <- g + scale_fill_uchicago()
  g <- g + theme_bw(base_size = 8)
  g <- g + theme(...)
  g
}
```

Función `boxplotByCluster`: Permite elaborar varios diagramas de cajas y bigotes de una misma variable en función de los distintos *clusters*.

```
boxploByCluster <- function(x, clus, ylab = "", title = "", subtitle = "", ...) {

  # Preparación del data.frame con los datos
  data <- data.frame(x, as.factor(clus))
  names(data) <- c("Var1", "Cluster")
  # Generación del gráfico con ggplot2
  g <- ggplot(data, aes(x = Cluster, y = Var1, fill = Cluster))
  g <- g + geom_boxplot()
  g <- g + labs(y = ylab)
  g <- g + ggtitle(title, subtitle = subtitle)
  g <- g + scale_fill_lancet()
  g <- g + theme_bw(base_size = 8)
  g <- g + theme(...)
  g
}
```

Función `plotResults`: Elabora un conjunto de gráficos (diagramas de barras, boxplots y gráficos de radar) para la interpretación de los agrupamientos.

```
plotResults <- function(functionality, site, site2, MFE, dG, clustering) {

  # Tablas para gráficos de barras
  functTab <- prop.table(table(functionality, clustering), 2)
  site2Tab <- prop.table(table(site2, clustering), 2)
  # Gráficos de barras
  g1 <- tableToBarPlot(functTab, xlab = "Cluster", ylab = "Proportion",
                       legend.position = "bottom")
  g2 <- tableToBarPlot(site2Tab, xlab = "Cluster", ylab = "Proportion",
                       legend.position = "bottom")
  # Boxplots
  g3 <- boxploByCluster(MFE, clustering, ylab = "MFE [kcal/mol]")
  g3 <- g3 + coord_cartesian(ylim = c(0, -50))
  g4 <- boxploByCluster(dG, clustering, ylab = "dG [kcal/mol]")
```

```
  # Tabla para gráfico tipo radar 1
  avgMFE <- aggregate(MFE, by = list(clustering), mean)
  avgdG <- aggregate(dG, by = list(clustering), mean)
  radarTab1 <- cbind(t(functTab), t(site2Tab), avgMFE$x, avgdG$x)
  colnames(radarTab1)[8:9] <- c("avg.MFE", "avg.dG")
  radarTab1 <- as.data.frame(radarTab1)
  radarTab1 <- radarTab1[-2]
  radarTab1$Cluster <- 1:max(clustering)
  # Gráfico tipo radar 1
  rg1 <- ggRadar(radarTab1, aes(group = Cluster), alpha = .1)
  rg1 <- rg1 + theme_bw(base_size = 8)
  # Tabla para gráfico tipo radar 2
  siteTab <- prop.table(table(clustering, site), 1)
  radarTab2 <- cbind(siteTab)
  radarTab2 <- as.data.frame(radarTab2)
  radarTab2$Cluster <- 1:max(clustering)
  # Gráfico tipo radar 2
  rg2 <- ggRadar(radarTab2, aes(group = Cluster), alpha = .1)
  rg2 <- rg2 + theme_bw(base_size = 8)
  # Colores para gráficos tipo radar
  if (max(clustering) <= 6) {
    rg1 <- rg1 + scale_color_lancet() + scale_fill_lancet()
    rg2 <- rg2 + scale_color_lancet() + scale_fill_lancet()
  } else {
    mypal <- colorRampPalette(get_palette("lancet", 8))
    sc <- scale_color_gradientn(colors = mypal(8), limits = c(1, 8))
    scf <- scale_fill_gradientn(colors = mypal(8), limits = c(1, 8))
    rg1 <- rg1 + sc + scf
    rg2 <- rg2 + sc + scf
  }
  # Disposición de los gráficos
  gg1 <- ggarrange(
    g1, g2, nrow = 1, ncol = 2, widths = c(1, 1.5), labels = LETTERS[1:2]
  )
  gg2 <- ggarrange(
    g3, g4, nrow = 1, ncol = 2, legend = "bottom",
    common.legend = TRUE, labels = LETTERS[3:4]
  )
  gg3 <- ggarrange(
    rg1, rg2, nrow = 1, ncol = 2, legend = "bottom", labels = LETTERS[5:6],
    common.legend = TRUE
  )
  ggarng <- ggarrange(gg1, gg2, gg3, nrow = 3, ncol = 1, heights = c(1, 1, 1.75))
  # Mostrar gráficos
  return(ggarng)
}
```

Función `plotKNNdist`: Genera un gráfico que permite determinar el valor óptimo del parámetro épsilon cuando se va a usar DBSCAN u OPTICS:

```
plotKNNdist <- function(data, k) {
  dist <- sort(kNNdist(data, k))
  xaxis <- 1:(nrow(data) * k)
  if (length(xaxis) > 3e3) {
    id <- seq(1, max(xaxis), len = 3e3)
    dist <- dist[sort(id)]
    xaxis <- xaxis[sort(id)]
  }
  plot(x = xaxis, y = dist, type = "n",
       ylab = paste(k, "-NN distance", sep = ""),
       xlab = "Points (sample) sorted by distance")
  grid(lty = 1)
  lines(x = xaxis, y = dist, lwd = 2, col = "#00468BFF")
}
```

12.3. Preparación de los datos

Carga de los datos o atributos extraídos para cada interacción miRNA-diana:

```
humanFeatures <- read.delim("ProcessedData/human_features.txt",
                            stringsAsFactors = F)
mouseFeatures <- read.delim("ProcessedData/mouse_features.txt",
                            stringsAsFactors = F)
```

División del conjunto de datos en función de la naturaleza de los atributos; de tipo categórico o de tipo numérico:

```
# Homo sapiens
humanCatFeat <- humanFeatures[1:8]
humanNumFeat <- humanFeatures[9:ncol(humanFeatures)]
# Mus musculus
mouseCatFeat <- mouseFeatures[1:8]
mouseNumFeat <- mouseFeatures[9:ncol(mouseFeatures)]
```

Los registros `NEGATIVE` en la variable `functionality` resultan anecdóticos (uno o dos casos por especie); la gran mayoría son `POSITIVE` o `NA`. Por este motivo se establecen únicamente las clases `POSITIVE` y `UNKNOWN`:

```
# Homo sapiens
x <- humanCatFeat$functionality
x[x != "POSITIVE" | is.na(x)] <- "UNKNOWN"
humanCatFeat$functionality <- x
# Mus musculus
x <- mouseCatFeat$functionality
x[x != "POSITIVE" | is.na(x)] <- "UNKNOWN"
```

```
mouseCatFeat$functionality <- x
```

Agrupación de los tipos de interacción registrados en cinco clases según se trate de uniones canónicas (`Canonical`), canónicas con unión suplementaria en 3' (`Canonical_supp`), no canónicas (`NonCanonical`), no canónicas centradas (`NonCa_centred`) y no canónicas con unión compensatoria en 3' (`NonCa_comp`):

```
# Homo sapiens
x <- humanCatFeat$interact.type
x[grepl("mism", x)] <- "NonCanonical"
x[grepl("supp", x)] <- "Canonical_supp"
x[grepl("mer", x)] <- "Canonical"
humanCatFeat$interact.type2 <- x
# Mus musculus
x <- mouseCatFeat$interact.type
x[grepl("mism", x)] <- "NonCanonical"
x[grepl("supp", x)] <- "Canonical_supp"
x[grepl("mer", x)] <- "Canonical"
mouseCatFeat$interact.type2 <- x
```

12.4. Reducción dimensional de los datos

Se emplean dos técnicas diferentes: PCA (*Principal component analysis*) e ICA (*Idependent component analysis*). Cada técnica se aplica sobre dos conjuntos de datos, uno con las variables numéricas normalizadas y otro con la mismas variables estandarizadas (*Z-score*).

Variables numéricas normalizadas y estandarizadas:

```
# Homo sapiens
# Normalización de los datos
humanNormFeat <- as.data.frame(lapply(humanNumFeat, normalize))
# Estandarización de los datos
humanStdFeat <- scale(humanNumFeat)

# Mus musculus
# Normalización de los datos
mouseNormFeat <- as.data.frame(lapply(mouseNumFeat, normalize))
# Estandarización de los datos
mouseStdFeat <- scale(mouseNumFeat)
```

Reducción dimensional mediante PCA:

```
# Homo sapiens
# PCA sobre datos normalizados
humanNormPCA <- prcomp(humanNormFeat)
humanNormPC <- data.frame(humanNormPCA$x[, 1:numOfPC(humanNormPCA, .95)])
```

```
# PCA sobre datos estandarizados
humanStdPCA <- prcomp(humanStdFeat)
humanStdPC <- data.frame(humanStdPCA$x[, 1:numOfPC(humanStdPCA, .95)])

# Mus musculus
# PCA sobre datos normalizados
mouseNormPCA <- prcomp(mouseNormFeat)
mouseNormPC <- data.frame(mouseNormPCA$x[, 1:numOfPC(mouseNormPCA, .95)])
# PCA sobre datos estandarizados
mouseStdPCA <- prcomp(mouseStdFeat)
mouseStdPC <- data.frame(mouseStdPCA$x[, 1:numOfPC(mouseStdPCA, .95)])
```

Reducción dimensional mediante ICA; el número de componentes utilizado depende del número de componentes seleccionados en el caso del PCA:

```
# Homo sapiens
# ICA sobre datos normalizados
humanNormICA <- fastICA(humanNormFeat, n.comp = ncol(humanNormPC))
humanNormIC <- data.frame(humanNormICA$S)
# ICA sobre datos estandarizados
humanStdICA <- fastICA(humanStdFeat, n.comp = ncol(humanStdPC))
humanStdIC <- data.frame(humanStdICA$S)

# Mus musculus
# ICA sobre datos normalizados
mouseNormICA <- fastICA(mouseNormFeat, n.comp = ncol(mouseNormPC))
mouseNormIC <- data.frame(mouseNormICA$S)
# ICA sobre datos estandarizados
mouseStdICA <- fastICA(mouseStdFeat, n.comp = ncol(mouseStdPC))
mouseStdIC <- data.frame(mouseStdICA$S)
```

Elaboración de una serie de gráficos para visualizar el porcentaje o proporción de la varianza total explicada según el número de componentes principales:

```
# Generación de los gráficos
g1 <- plotPCAsummary(humanNormPCA, 40, title = expression(italic("Homo sapiens")))
g2 <- plotPCAsummary(humanStdPCA, 40)
g3 <- plotPCAsummary(mouseNormPCA, 40, title = expression(italic("Mus musculus")))
g4 <- plotPCAsummary(mouseStdPCA, 40)
ggarng <- ggarrange(
  g1, g3, g2, g4, nrow = 2, ncol = 2, legend = "bottom", common.legend = TRUE,
  labels = c("A", "", "B", "")
)
# Guardado de los gráficos
pdf("Results/PCA_Explained_Variance.pdf", width = 8.75, height = 7.25)
ggarng
dev.off()
```

12.5. CLARA

Aplicación del algoritmo de *clustering* CLARA (k = 2, . . . , 10) a cada conjunto de datos generado:

```
# Homo sapiens
humanClaraList <- list(
  claraMultiK(humanNormPC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(humanNormIC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(humanStdPC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(humanStdIC, 2:10, samples = 100, sampsize = 500, medoids.x = F)
)
names(humanClaraList) <- c("Normalized_PCA", "Normalized_ICA",
                           "Standarized_PCA", "Standarized_ICA")
# Mus musculus
mouseClaraList <- list(
  claraMultiK(mouseNormPC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(mouseNormIC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(mouseStdPC, 2:10, samples = 100, sampsize = 500, medoids.x = F),
  claraMultiK(mouseStdIC, 2:10, samples = 100, sampsize = 500, medoids.x = F)
)
names(mouseClaraList) <- c("Normalized_PCA", "Normalized_ICA",
                           "Standarized_PCA", "Standarized_ICA")
```

Generación de gráficos para la visualización de la anchura media de la silueta según el número de grupos:

```
# Generación de los gráficos
g1 <- plotAvgSilWidth(humanClaraList, expression(italic("Homo sapiens")))
g2 <- plotAvgSilWidth(mouseClaraList, expression(italic("Mus musculus")))
ggarng <- ggarrange(
  g1, g2, nrow = 1, ncol = 2, legend = "bottom", common.legend = TRUE
)
# Guardado de los gráficos
pdf("Results/CLARA_AvgSil.pdf", width = 8.75, height = 6)
ggarng
dev.off()
```

Generación de gráficos para la visualización de la silueta en aquellos casos que registren la mayor anchura media de la silueta:

```
# Homo sapiens
g1 <- plotSilhouette(humanClaraList$Normalized_PCA$k4,
                     expression(italic("Homo sapiens")))
g2 <- plotSilhouette(humanClaraList$Normalized_PCA$k5)
g3 <- plotSilhouette(humanClaraList$Normalized_PCA$k8)
# Mus musculus
g4 <- plotSilhouette(mouseClaraList$Normalized_PCA$k4,
                     expression(italic("Mus musculus")))
```

```
g5 <- plotSilhouette(mouseClaraList$Normalized_PCA$k5)
g6 <- plotSilhouette(mouseClaraList$Normalized_PCA$k6)
# Disposición de los gráficos
gg1 <- ggarrange(g1, g2, g3, nrow = 3, ncol = 1, legend = "none")
gg2 <- ggarrange(g4, g5, g6, nrow = 3, ncol = 1, legend = "none")
ggarng <- ggarrange(gg1, gg2, nrow = 1, ncol = 2)
# Guardado de los gráficos
pdf("Results/CLARA_Silhouette.pdf", width = 8.75, height = 7.25)
ggarng
dev.off()
```

Elaboración de un arreglo de gráficos (diagramas de barras, boxplots y gráficos de radar) para la visualización de los resultados tras seleccionar el mejor agrupamiento para cada especie:

```
# Homo sapiens
# Generación de los gráficos
clustering <- humanClaraList$Normalized_PCA$k8$clustering
ggarng <- plotResults(
  functionality = humanCatFeat$functionality,
  site  = humanCatFeat$interact.type,
  site2 = humanCatFeat$interact.type2,
  MFE   = humanNumFeat$MFE,
  dG    = humanNumFeat$dG.rnaup,
  clustering
)
# Guardado de los gráficos
pdf("Results/CLARA_Human_Results.pdf", width = 8.75, height = 8.75)
ggarng
dev.off()

# Mus musculus
# Generación de los gráficos
clustering <- mouseClaraList$Normalized_PCA$k6$clustering
ggarng <- plotResults(
  functionality = mouseCatFeat$functionality,
  site  = mouseCatFeat$interact.type,
  site2 = mouseCatFeat$interact.type2,
  MFE   = mouseNumFeat$MFE,
  dG    = mouseNumFeat$dG.rnaup,
  clustering
)
# Guardado de los gráficos
pdf("Results/CLARA_Mouse_Results.pdf", width = 8.75, height = 8.75)
ggarng
dev.off()
```

Elaboración de un conjunto de gráficos de puntos para visualizar los datos en humano:

```
clarak8 <- clara(humanNormPC, k = 8, samples = 100, sampsize = 500)
pdf("Results/CLARA_Human_Data_Vis.pdf", width = 8.75, height = 7.25, bg = "white")
# Disposición de lo gráficos
par(mfrow = c(2, 2), mar = c(4, 4, 1, 1),
    oma = c(1, 1, 2.5, 1), cex.axis = .7)
# Funcionalidad
colID <- factor(humanCatFeat$functionality)
colpal <- get_palette("uchicago", 2)
plot(humanNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("A")), figcorner = TRUE)
legend("bottomright", legend = c("POSITIVE", "UNKNOWN"),
       pch = 20, cex = .7, col = colpal)
# Tipo de interacción
colID <- factor(humanCatFeat$interact.type2)
colpal <- get_palette("uchicago", 5)
plot(humanNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("B")), figcorner = TRUE)
lgd <- c("Canonical", "Canonical_supp", "NonCa_centred",
         "NonCa_compensatory", "NonCanonical")
legend("bottomright", legend = lgd, pch = 20, cex = .7, col = colpal)
# MFE
MFE <- humanNumFeat$MFE
MFE[MFE < -40] <- -40
cutMFE <- cut(MFE, breaks = 50)
colorScale <- colorRampPalette(c("darkblue", "seashell"))
plot(humanNormPC[1:2], pch = 20, cex = .5, col = colorScale(50)[cutMFE])
corner.label(bquote(bold("C")), figcorner = TRUE)
legend("bottomright", legend = c(max(MFE), min(MFE)), pch = 20, cex = .7,
       col = c("seashell", "darkblue"))
# Clustering
colID <- clarak8$clustering
colpal <- get_palette("lancet", 8)
plot(humanNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("D")), figcorner = TRUE)
points(clarak8$medoids, pch = 20, cex = 3, col = "black")
points(clarak8$medoids, pch = 20, cex = 2, col = colpal)
legend("bottomright", legend = 1:8, pch = 20, cex = .7, col = colpal)
dev.off()
```

Elaboración de un conjunto de gráficos de puntos para visualizar los datos en ratón:

```
clarak6 <- clara(mouseNormPC, k = 6, samples = 100, sampsize = 500)
pdf("Results/CLARA_Mouse_Data_Vis.pdf", width = 8.75, height = 7.25, bg = "white")
# Disposición de lo gráficos
par(mfrow = c(2, 2), mar = c(4, 4, 1, 1),
    oma = c(1, 1, 2.5, 1), cex.axis = .7)
# Funcionalidad
colID <- factor(mouseCatFeat$functionality)
```

```
colpal <- get_palette("uchicago", 2)
plot(mouseNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("A")), figcorner = TRUE)
legend("bottomright", legend = c("POSITIVE", "UNKNOWN"),
       pch = 20, cex = .7, col = colpal)
# Tipo de interacción
colID <- factor(mouseCatFeat$interact.type2)
colpal <- get_palette("uchicago", 5)
plot(mouseNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("B")), figcorner = TRUE)
lgd <- c("Canonical", "Canonical_supp", "NonCa_centred",
         "NonCa_compensatory", "NonCanonical")
legend("bottomright", legend = lgd, pch = 20, cex = .7, col = colpal)
# MFE
MFE <- mouseNumFeat$MFE
MFE[MFE < -40] <- -40
cutMFE <- cut(MFE, breaks = 50)
colorScale <- colorRampPalette(c("darkblue", "seashell"))
plot(mouseNormPC[1:2], pch = 20, cex = .5, col = colorScale(50)[cutMFE])
corner.label(bquote(bold("C")), figcorner = TRUE)
legend("bottomright", legend = c(max(MFE), min(MFE)), pch = 20, cex = .7,
       col = c("seashell", "darkblue"))
# Clustering
colID <- clarak6$clustering
colpal <- get_palette("lancet", 6)
plot(mouseNormPC[1:2], pch = 20, cex = .5, col = colpal[colID])
corner.label(bquote(bold("D")), figcorner = TRUE)
points(clarak6$medoids, pch = 20, cex = 3, col = "black")
points(clarak6$medoids, pch = 20, cex = 2, col = colpal)
legend("bottomright", legend = 1:6, pch = 20, cex = .7, col = colpal)
dev.off()
```

12.6. Expectation–Maximization (EM)

Aplicación del algoritmo de *clustering* EM (G = 4, . . . , 10) a cada conjunto de datos generado:

```
# Homo sapiens
humanEMlist <- list(
  "Normalized_PCA" = Mclust(humanNormPC, 4:10),
  "Normalized_ICA" = Mclust(humanNormIC, 4:10),
  "Standarized_PCA" = Mclust(humanStdPC, 4:10),
  "Standarized_ICA" = Mclust(humanStdIC, 4:10)
)
# Mus musculus
mouseEMlist <- list(
```

```
  "Normalized_PCA" = Mclust(mouseNormPC, 4:10),
  "Normalized_ICA" = Mclust(mouseNormIC, 4:10),
  "Standarized_PCA" = Mclust(mouseStdPC, 4:10),
  "Standarized_ICA" = Mclust(mouseStdIC, 4:10)
)
```

Generación de un arreglo de gráficos para la visualización del criterio de información bayesiano (BIC) en función del conjunto de datos y número de grupos:

```
pdf("Results/EM_BIC.pdf", width = 8.75, height = 9.5, bg = "white")
par(mfrow = c(4, 2), mar = c(3, 4, 1, 1),
    oma = c(1, 1, 2.5, 1), cex.axis = .75)
plot(humanEMlist[[1]], what = "BIC")
title(expression(italic("Homo sapiens")), adj = 0, line =  0.5)
corner.label(bquote(bold("A")), figcorner = TRUE)
plot(mouseEMlist[[1]], what = "BIC")
title(expression(italic("Mus musculus")), adj = 0, line =  0.5)
for (i in 2:4) {
  plot(humanEMlist[[i]], what = "BIC")
  label <- bquote(bold(.(LETTERS[i])))
  corner.label(label, figcorner = T)
  plot(mouseEMlist[[i]], what = "BIC")
}
dev.off()
```

Confección de una serie de diagramas de barras para la visualización de los resultados del algoritmo EM en humano:

```
# Variables
functionality <- humanCatFeat$functionality
site2 <- humanCatFeat$interact.type2
clustering <- lapply(humanEMlist, function(x) x$classification)
# Tablas de proporciones
functTab <- lapply(clustering, function(x) prop.table(table(functionality, x), 2))
site2Tab <- lapply(clustering, function(x) prop.table(table(site2, x), 2))
# Gráficos de barras
g1 <- tableToBarPlot(functTab[[1]], xlab = "Cluster", ylab = "Proportion")
g2 <- tableToBarPlot(site2Tab[[1]], xlab = "Cluster", ylab = "Proportion")
g3 <- tableToBarPlot(functTab[[2]], xlab = "Cluster", ylab = "Proportion")
g4 <- tableToBarPlot(site2Tab[[2]], xlab = "Cluster", ylab = "Proportion")
g5 <- tableToBarPlot(functTab[[3]], xlab = "Cluster", ylab = "Proportion")
g6 <- tableToBarPlot(site2Tab[[3]], xlab = "Cluster", ylab = "Proportion")
g7 <- tableToBarPlot(functTab[[4]], xlab = "Cluster", ylab = "Proportion")
g8 <- tableToBarPlot(site2Tab[[4]], xlab = "Cluster", ylab = "Proportion")
# Disposición de los gráficos
gg1 <- ggarrange(g1, g3, g5, g7, nrow = 4, ncol = 1, legend = "bottom",
                 common.legend = TRUE, labels = LETTERS[1:4])
gg2 <- ggarrange(g2, g4, g6, g8, nrow = 4, ncol = 1, legend = "bottom",
```

```
                  common.legend = TRUE)
ggarng <- ggarrange(gg1, gg2, nrow = 1, ncol = 2, widths = c(1, 1.5))
# Guardado de los gráficos
pdf("Results/EM_Human_Barplot.pdf", width = 8.75, height = 8.75)
ggarng
dev.off()
```

Confección de una serie de diagramas de barras para la visualización de los resultados del algoritmo EM en ratón:

```
# Variables
functionality <- mouseCatFeat$functionality
site2 <- mouseCatFeat$interact.type2
clustering <- lapply(mouseEMlist, function(x) x$classification)
# Tablas de proporciones
functTab <- lapply(clustering, function(x) prop.table(table(functionality, x), 2))
site2Tab <- lapply(clustering, function(x) prop.table(table(site2, x), 2))
# Gráficos de barras
g1 <- tableToBarPlot(functTab[[1]], xlab = "Cluster", ylab = "Proportion")
g2 <- tableToBarPlot(site2Tab[[1]], xlab = "Cluster", ylab = "Proportion")
g3 <- tableToBarPlot(functTab[[2]], xlab = "Cluster", ylab = "Proportion")
g4 <- tableToBarPlot(site2Tab[[2]], xlab = "Cluster", ylab = "Proportion")
g5 <- tableToBarPlot(functTab[[3]], xlab = "Cluster", ylab = "Proportion")
g6 <- tableToBarPlot(site2Tab[[3]], xlab = "Cluster", ylab = "Proportion")
g7 <- tableToBarPlot(functTab[[4]], xlab = "Cluster", ylab = "Proportion")
g8 <- tableToBarPlot(site2Tab[[4]], xlab = "Cluster", ylab = "Proportion")
# Disposición de los gráficos
gg1 <- ggarrange(g1, g3, g5, g7, nrow = 4, ncol = 1, legend = "bottom",
                 common.legend = TRUE, labels = LETTERS[1:4])
gg2 <- ggarrange(g2, g4, g6, g8, nrow = 4, ncol = 1, legend = "bottom",
                 common.legend = TRUE)
ggarng <- ggarrange(gg1, gg2, nrow = 1, ncol = 2, widths = c(1, 1.5))
# Guardado de los gráficos
pdf("Results/EM_Mouse_Barplot.pdf", width = 8.75, height = 8.75)
ggarng
dev.off()
```

12.7. OPTICS

Elaboración de un conjunto de gráficos para visualizar la distancia entre cada punto y sus k vecinos más cercanos; esta información resulta útil a la hora de escoger el máximo valor del parámetro `epsilon` cuando se aplica el algoritmo de clustering OPTICS:

```
humanDataList <- list(humanNormPC, humanNormIC, humanStdPC, humanStdIC)
mouseDataList <- list(mouseNormPC, mouseNormIC, mouseStdPC, mouseStdIC)
pdf("Results/OPTICS_kNNdistplot.pdf", width = 8.75, height = 9.5)
```

```
par(mfrow = c(4, 2), mar = c(3, 4, 1, 1), oma = c(1, 1, 2.5, 1),
    cex.axis = .75, lwd = 1.5)
plotKNNdist(humanDataList[[1]], k = 50)
title(expression(italic("Homo sapiens")), adj = 0, line =  0.5)
corner.label(bquote(bold("A")), figcorner = TRUE)
plotKNNdist(mouseDataList[[1]], k = 50)
title(expression(italic("Mus musculus")), adj = 0, line =  0.5)
for (i in 2:4) {
  plotKNNdist(humanDataList[[i]], k = 50)
  label <- bquote(bold(.(LETTERS[i])))
  corner.label(label, figcorner = T)
  plotKNNdist(mouseDataList[[i]], k = 50)
}
dev.off()
```

Aplicación del algoritmo de *clustering* OPTICS (`minPts` = 50) a cada conjunto de datos generado:

```
# Homo sapiens
humanOPTICSlist <- list(
  "Normalized_PCA" = optics(humanNormPC, eps = 1.2, minPts = 50),
  "Normalized_ICA" = optics(humanNormIC, eps = 6, minPts = 50),
  "Standarized_PCA" = optics(humanStdPC, eps = 10, minPts = 50),
  "Standarized_ICA" = optics(humanStdIC, eps = 10, minPts = 50)
)
# Mus musculus
mouseOPTICSlist <- list(
  "Normalized_PCA" = optics(mouseNormPC, eps = 1, minPts = 50),
  "Normalized_ICA" = optics(mouseNormIC, eps = 5, minPts = 50),
  "Standarized_PCA" = optics(mouseStdPC, eps = 10, minPts = 50),
  "Standarized_ICA" = optics(mouseStdIC, eps = 10, minPts = 50)
)
```

Generación de los gráficos de distancia de alcance o *reachability distance* para cada especie y conjunto de datos:

```
pdf("Results/OPTICS_ReachDist.pdf", width = 8.75, height = 9.5)
par(mfrow = c(4, 2), mar = c(3, 4, 1, 1),
    oma = c(1, 1, 2.5, 1), cex.axis = .75)
plot(humanOPTICSlist$Normalized_PCA, main = "")
title(expression(italic("Homo sapiens")), adj = 0, line =  0.5)
corner.label(bquote(bold("A")), figcorner = TRUE)
plot(mouseOPTICSlist$Normalized_PCA, main = "")
title(expression(italic("Mus musculus")), adj = 0, line =  0.5)
for (i in 2:4) {
  plot(humanOPTICSlist[[i]], main = "")
  label <- bquote(bold(.(LETTERS[i])))
  corner.label(label, figcorner = T)
```

```
  plot(mouseOPTICSlist[[i]], main = "")
}
dev.off()
```

12.8. Selección de atributos

Previo a la selección de atributos, para cada variable numérica de `humanNumFeat` y `mouseNumFeat`, se comprueba si existen diferencias significativas en la distribución de cada variable numérica atendiendo a la funcionalidad de las interacciones:

```
# Homo sapiens
# Atributos o variables numéricas
features <- colnames(humanNumFeat)
# Prueba t de Student para cada variable
p_val <- applytTest(humanNumFeat, humanCatFeat$functionality == "POSITIVE")
p_val <- round(p_val, 3)
# Pureba d de Cohen para cada variable
cohen_d <- applyCohenD(humanNumFeat, humanCatFeat$functionality == "POSITIVE")
cohen_d <- round(cohen_d, 3)
# Guardado de los resultados
results <- data.frame(features, p_val, cohen_d, row.names = NULL,
                      stringsAsFactors = F)
write.table(results, "Results/Human_tTest_Cohend.txt", sep = "\t")

# Mus musculus
# Atributos o variables numéricas
features <- colnames(mouseNumFeat)
# Prueba t de Student para cada variable
p_val <- applytTest(mouseNumFeat, mouseCatFeat$functionality == "POSITIVE")
p_val <- round(p_val, 3)
# Pureba d de Cohen para cada variable
cohen_d <- applyCohenD(mouseNumFeat, mouseCatFeat$functionality == "POSITIVE")
cohen_d <- round(cohen_d, 3)
# Guardado de los resultados
results <- data.frame(features, p_val, cohen_d, row.names = NULL,
                      stringsAsFactors = F)
write.table(results, "Results/Mouse_tTest_Cohend.txt", sep = "\t")
```

Selección de atributos de cara a la predicción de la funcionalidad de las interacciones mediante un método de filtrado con *Random forest*; previamente se descartan algunas variables altamente correlacionadas:

```
set.seed(122333)
# Homo sapiens
# Variables altamente correlacionadas
highCor <- findCorrelation(cor(humanNumFeat))
```

```
# Selección de atributos
filterCtrl <- sbfControl(functions = rfSBF, method = "cv", number = 5)
humanFeatSel <- sbf(humanNumFeat[, -highCor], factor(humanCatFeat$functionality),
                    sbfControl = filterCtrl)
# Guardado de los resultados
sink("Results/Human_Feature_Selection.txt")
humanFeatSel$results
writeLines("============================================")
humanFeatSel$optVariables
sink()


# Mus musculus
# Variables altamente correlacionadas
highCor <- findCorrelation(cor(mouseNumFeat))
# Selección de atributos
filterCtrl <- sbfControl(functions = rfSBF, method = "cv", number = 5)
mouseFeatSel <- sbf(mouseNumFeat[, -highCor], factor(mouseCatFeat$functionality),
                    sbfControl = filterCtrl)
# Guardado de los resultados
sink("Results/Mouse_Feature_Selection.txt")
mouseFeatSel$results
writeLines("============================================")
mouseFeatSel$optVariables
sink()
```