Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Doctorado en Ciencias en Ciencias de la Computación

Métodos para la predicción de genes de ARNnc

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Doctor en Ciencias

Presenta:

Hugo Armando Guillén Ramírez

Ensenada, Baja California, México

2018

Tesis defendida por

Hugo Armando Guillén Ramírez

y aprobada por el siguiente Comité

Dr. Israel Marck Martínez Pérez
Director de tesis

Dr. Carlos Alberto Brizuela Rodríguez

Dr. Hugo Homero Hidalgo Silva

Dra. Kristina Marie Herbert

Dr. Everardo Gutiérrez López

Dr. Jesús Favela Vara
Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta Hugo Armando Guillén Ramírez como requisito parcial para la obtención del grado de Doctor en Ciencias en Ciencias de la Computación.

Métodos para la predicción de genes de ARNnc

Resumen aprobado por:

Dr. Israel Marck Martínez Pérez
Director de tesis

Durante la década pasada, la investigación del ARN no codificante (ncRNA por sus siglas en inglés) ha crecido dramáticamente debido a su intervención en procesos celulares y aplicaciones terapéuticas. Estas moléculas provienen en su mayoría de regiones del genoma que se creían “basura”, las cuales ahora son denominadas genes de ARN. Comparando con los métodos computacionales para detectar y predecir genes codificantes de proteínas, todavía existe una gran brecha en los niveles de precisión alcanzados. Sin embargo, muchas de las limitantes provienen de las características inherentes del ncRNA tales como la existencia de conservación en el nivel estructural secundario pero no así en el primario de las secuencias. El proceso de anotación, es decir, identificar la posición en el genoma o la función de estas moléculas, es un proceso complejo que se puede dividir en varias etapas. Una de ellas consiste en la discriminación inter-clase de secuencias putativas. En este trabajo se aborda la solución de esta tarea mediante metodologías basadas en aprendizaje máquina. Primero, a nivel de aplicación, se discriminó entre secuencias de riboswitches. Los riboswitches son ncRNAs que regulan la expresión génica alterando la conformación estructural de transcritos de ARN mensajero. No obstante la importancia de la estructura en la función de estas moléculas, en este trabajo se muestra que es posible discriminar de manera precisa entre 16 familias de riboswitches utilizando aprendizaje supervisado y un conjunto de características propuesto derivado de la secuencia, logrando superar resultados del estado del arte. Posteriormente, aplicando características basadas en secuencia, se discriminó entre ARN de interferencia capaz de producir un efecto inmunomodulador deseado o no. Como resultado, se mejoró la eficiencia en términos de precisión, especificidad y coeficiente de correlación de Matthews respecto a lo reportado por métodos del estado del arte. Más adelante, se estudian las propiedades de ciertas características basadas en secuencia. Se muestra que estas características son redundantes, por lo que se propone un esquema de selección basado en grafos que se apoya en dos funciones: la frecuencia *k*-mérica intra-clase y una medida de calidad inter-clase. Finalmente, se estudian y desarrollan abstracciones de la estructura secundaria del ARN, con las cuales fue posible calcular la redundancia intra-clase y la similitud inter-clase entre distintas familias de ARN sin realizar alineamientos múltiples. Este trabajo puede servir de base para el desarrollo de enfoques más integrales capaces de la identificación *de novo* de genes de ARN en genomas completos.

Palabras clave: ARN no codificante, aprendizaje máquina, riboswitch, estructura secundaria del ARN

Abstract of the thesis presented by Hugo Armando Guillén Ramírez as a partial requirement to obtain the Doctor of Science degree in Computer Science.

Methods for non-coding RNA genes prediction

Abstract approved by:

Dr. Israel Marck Martínez Pérez
Thesis Director

Interest in non-coding RNAs (ncRNAs) over the last decade has grown due to their discovered roles in many essential cellular processes and potential therapeutic applications. These molecules, which are now called RNA genes are located in regions of the genome formerly denoted as “junk DNA”. While interest in these ncRNAs is great, the accuracy and speed of the computational methods for identifying, detecting, or predicting RNA genes are still far behind their protein-coding gene counterparts. Some of the challenges reside in intrinsic characteristics of ncRNAs, e.g., conservation of the secondary structure, but not the primary sequence. The annotation process attempts to identify the location or function of the genes in a given genome. This endeavor as a whole is complex, but it could be divided hierarchically into substages like the inter-discrimination of putative sequences. This thesis approaches such a classification task with supervised learning and feature engineering. First, we show a newly developed methodology for classifying riboswitch sequences. Riboswitches are ncRNA segments of messenger RNAs that regulate gene expression by adopting different structural conformations. Even though the structure is an essential feature of this kind of molecule, we outperformed the state-of-the-art classification performance of 16 riboswitch families using a set of sequence-based features. Next, we approach the problem of determining whether an interfering RNA is capable of generating a controlled immunomodulatory response or not. This work shows improvement in the classification performance over the latest methodology used to predict immune response induced by interfering RNAs in terms of precision, specificity, and Mathews correlation coefficient. Later, we study the engineering of features based on primary-sequence structure, specifically, k-mer frequencies. We find that such features are inherently redundant, so we propose a graph-based selection strategy built over two functions: intra-class k-mer coverage and an inter-class quality value. Additionally, we develop alternative representations for the secondary structure, and as a proof of concept, we show an alignment-free method that estimates intra-class redundancy and inter-class similarity. This thesis could serve as a foundation for developing comprehensive approaches capable of identifying RNA genes de novo in whole genomes.

Keywords: ncRNA, machine learning, riboswitch, RNA secondary structure

Dedicatoria

A Elena

Agradecimientos

A mi esposa Gabriela, por estar conmigo en este viaje en el que decidimos embarcarnos juntos, gracias por tu amor y apoyo. Elena, eres mi motivación más grande. Te agradezco por mostrarme el infinito amor que puede llegar a sentirse. También agradezco a mis padres y hermanos que siempre han sido un pilar para mi todo este tiempo.

Agradezco sinceramente al Dr. Israel Martínez por su guía constante durante estos años tanto en lo académico como en lo personal.

A los miembros de mi comité de tesis, el Dr. Carlos Brizuela, el Dr. Hugo Hidalgo, la Dra. Kristina Herbert y el Dr. Everardo Gutiérrez. Muchas gracias por su apoyo y guía durante el transcurso de este proyecto.

Al Centro de Investigación Científica y de Educación Superior de Ensenada, especialmente a los miembros del Departamento de Ciencias de la Computación. Gracias a los profesores por siempre tener las puertas abiertas, así como al personal técnico y administrativo que siempre tiene la mejor disposición para hacer más llevadera la estancia, especialmente a Angie y Karina.

Gracias a todos los compañeros del posgrado, especialmente a Armando, Franceli, Julio, el Dr. Colbes, Julia, Nelson, Loge, Luis, Roilhi y Neftali.

A mis amigos de Tepic, Voz Amares y muchas más personas que me han dado su apoyo y afecto.

Finalmente, agradezco al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de doctorado. No. de becario: 261188.

Tabla de contenido

Página

Tabla de contenido (continuación)

Tabla de contenido (continuación)

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Figura Página

Lista de tablas

Tabla Página

Lista de tablas (continuación)

Capítulo 1. Introducción

Anteriormente se creía que la mayor parte del ADN era “basura” debido a que sólo el 1.2 % del genoma humano codifica para proteínas, sin embargo, cerca del 90 % se transcribe (Brosius y Gould, 1992), lo que despertó la hipótesis de que existen muchas secuencias de ARN codificadas en el ADN cuya función no es traducirse en proteínas, sino algo más. Ésta hipótesis se ha confirmado en la última década con el descubrimiento de las funciones catalíticas, regulatorias y enzimáticas del ARN, por lo que se acuñó el término de gen de ARN para describir una porción de ADN que codifica no para proteínas, sino para una cadena de ARN. Estas cadenas de ARN son denominadas ARN no codificante (ncRNA por sus siglas en inglés) (Eddy, 2001).

Por medio de métodos como RNA-seq es posible obtener una instantánea del ARN contenido en una célula en un determinado momento. La pregunta que muchos investigadores se plantean es si todas las secuencias de ARN obtenidas por este procedimiento realmente tienen una función, o sólo son subproductos del mecanismo de transcripción. Para categorizar estas secuencias, se realiza la anotación (Yandell y Ence, 2012). La anotación se divide en dos áreas: la anotación *estructural* y *funcional*. La anotación estructural es la identificación de genes y sus estructuras intrón-exón, mientras que la funcional consiste en añadir metadatos a las relaciones estructurales. Dentro de la anotación se encuentra la predicción de genes, es decir, la identificación de los sitios del genoma que guardan alguna función.

Los métodos de predicción de genes pueden combinar evidencia biológica previa y características extraídas de manera computacional para identificar genes en secuencias de ADN. Para los genes codificantes de proteínas se han desarrollado muchos métodos y heurísticas que realizan el proceso rápidamente y de manera precisa, sin embargo este tipo de predicciones para genes de ARN está en su infancia, por lo que todavía es muy costoso computacionalmente. A la fecha, no hay un genoma totalmente anotado de genes de ARN, por lo que es un área de oportunidad en este campo de rápido crecimiento.

1.1. Retos en la predicción de genes ncRNA

- El ARN no codificante largo (mayor a 200 nucleótidos, lncRNA por sus siglas en inglés) en general no está bien anotado, debido a que faltan las posiciones de inicio y término. La similitud a nivel primario que presentan no es suficiente para seleccionar un patrón de búsqueda de una familia en específico debido a la falta de conservación de los genes ncRNA, por lo que se requiere de un experto que los organice e integre manualmente (Gorodkin *et al.*, 2014).
- No existe un genoma completamente anotado en genes de ncRNA, por lo que no es posible estimar del todo la precisión y sensibilidad del predictor; muchos ncRNA son de longitud pequeña (< 100 nt (nucleótidos)), sin embargo por lo general son de longitud variable; y a la hora de entrenar un modelo de reconocimiento probabilístico, si se excluye una secuencia cuya mutación sea realmente una característica de la familia, ésta no será considerada en la búsqueda (Menzel *et al.*, 2009).
- Es difícil encontrar homólogos fuera del rango filogenético de ejemplos conocidos (Freyhult *et al.*, 2007).
- Actualmente, la búsqueda y la predicción de estructuras tienen un alto costo computacional, además, la mayoría de los métodos no consideran pseudonudos a pesar de que existen y tienen funciones dentro de las familias de ncRNA; y el método estándar para comparar predicciones con algún *estándar de oro* es problemático debido a que de los verdaderos positivos disponibles, esto es, ncRNA conocidos, a pesar de que son bien recuperados por los métodos, pueden no ser representativos de ncRNA sin descubrir (Gorodkin *et al.*, 2010).

1.2. Planteamiento del problema

El problema de predicción de genes es un problema desafiante y que todavía necesita mejoras, especialmente al analizar genomas grandes (Maji y Garg, 2013). A pesar de que el estado del arte ha mejorado el desempeño de los algoritmos continuamente durante las últimas dos décadas, todavía no existe una manera de generar automáti-

camente modelos génicos de alta calidad para un genoma entero, incluso en uno tan estudiado como el humano (Alioto, 2012).

Los métodos de predicción de ncRNA todavía están en su infancia comparados con los predictores de genes codificanes de proteínas, sin embargo, es un área en rápido crecimiento (Yandell y Ence, 2012). A diferencia de los genes codificantes de proteínas, los genes de ncRNA usualmente no están bien conservados en un nivel primario; incluso cuando lo están, las homologías a nivel de nucleótidos no son tan fáciles de detectar como las homologías a nivel de proteínas, lo cual limita el poder de los enfoques basados en evidencia y hace necesario el uso de modelos de clasificación (Yuan y Sun, 2013). Es por esto que para la predicción de ncRNA, los biólogos utilizan diferentes herramientas bioinformáticas con distintos métodos, para después hacer un análisis combinado de todas las evidencias recabadas y decidir si la secuencia es un ncRNA potencial (Arruda *et al.*, 2013).

Existen ciertas pistas biológicas que pueden ayudar a los predictores, como el uso de codones, sustituciones sinónimas y no sinónimas, y la energía de mínimo plegamiento (McCaskill, 1990b). En el contexto experimental, la resecuenciación de ARN es una adición a la identificación de ncRNA. Por ejemplo, se pueden identificar miARNs directamente utilizando métodos de extracción de ARN específicos y protocolos de secuenciación (Chen *et al.*, 2005; QIAGEN, 2011). Incluso con estas herramientas sofisticadas, distinguir entre los genes ncRNA reales, falsos y genes codificadores de proteínas pobremente conservados que producen pequeños péptidos continua siendo una tarea difícil, especialmente en los casos de regiones intergénicas largas de ncRNA (lincRNA) (Mercer *et al.*, 2009; van Leeuwen y Mikkers, 2010; Hung y Chang, 2010) y pseudogenes expresados (Tam *et al.*, 2008; Zhang *et al.*, 2004).

El problema consiste en desarrollar un clasificador de genes ncRNA que integre las diversas evidencias tanto biológicas como computacionales que sea comparable ya sea en precisión o tiempo de ejecución a los enfoques actuales.

1.3. Justificación

Los predictores de genes ncRNA representan una necesidad real en los campos de la genómica, medicina y áreas afines, debido a la relevancia biológica de estas moléculas y la cantidad inmensa de información generada por las técnicas de secuenciación de nueva generación (NGS por sus siglas en inglés) (Yandell y Ence, 2012). Cada vez se descubren más propiedades relevantes en los ncRNA, desde su relación con las formas alternativas de *splicing* en la traducción, hasta como indicadores para el diagnóstico de enfermedades (Mercer *et al.*, 2009; Hung y Chang, 2010; van Leeuwen y Mikkers, 2010), por lo que se espera que esta investigación ayude a la identificación de nuevos genes ncRNA como parte de un esquema de trabajo mayor.

La gran mayoría de los predictores de genes se centran en la identificación de genes codificantes. Gran parte de ellos trabajan con organismos bastante estudiados y tratan de adaptar los modelos génicos a otros organismos que pudieran no guardar una relación estrecha, lo cual no tiene un soporte biológico fuerte.

Los predictores de genes existentes basados en heurísticas han adquirido un nivel de precisión significante en esta tarea, sin embargo, aún no superan a los otros métodos propuestos. Por otro lado, además de ser un enfoque poco abordado en la literatura, la mayoría de estos predictores son desarrollados y entrenados para genomas específicos, por lo que las técnicas se aplican de una manera muy limitada (Goel et al., 2013). Además, los predictores ncRNA existentes tienen una eficiencia bastante pobre tanto en sensibilidad como en la utilización de recursos computacionales, por lo que se limitan al análisis de secuencias cortas (Arruda *et al.*, 2013). Por lo tanto, desde el punto de vista metodológico, el problema representa una oportunidad para la exploración de nuevas técnicas computacionales que intenten modelar el problema de una manera diferente y mejoren la eficiencia de los enfoques actuales.

1.4. Trabajo previo

A continuación se describen algunas de las herramientas computacionales para la predicción de ncRNA.

Infernal (Eddy y Durbin, 1994; Nawrocki *et al.*, 2009) es un predictor que identifica ncRNA mediante la búsqueda de las estructuras secundarias de ARN. Construye perfiles de consensos entre las estructuras espaciales de ARN, conocidos como modelos de covarianza, con el fin de encontrar similitudes entre la secuencia investigada y la estructura de consenso secundaria de cada una de las familias de ARN almacenadas en la base de datos Rfam (Griffiths-Jones *et al.*, 2003).

La herramienta tRNAscan (Lowe y Eddy, 1997) se considera uno de los predictores de ARN de transferencia (ARNt) más preciso. Combina tres programas: dos predictores de ARNt y el modelo de covarianza de Eddy y Durbin (1994) entrenado previamente con secuencias de ARNt. La ejecución de los tres programas resulta en un identificador de ARNt con una alta sensibilidad (99 % - 100 %) y alta especificidad al presentar una tasa de falsos positivos menor a 0.00007 por Mb, en una velocidad razonable. Portrait (Arrial *et al.*, 2009) identifica ncRNA mediante una máquina de soporte de vectores en transcriptomas incompletos o de especies no totalmente caraterizadas. El resultado de este programa es la probabilidad de que un transcrito sea o no un codificador de proteínas.

Vienna RNA (Hofacker *et al.*, 1994) es un conjunto de paquetes usados para generar o comparar estructuras secundarias de ADN. El plegado en esta herramienta usa algoritmos de predicción basados en la energía libre del ARN y la probabilidad del pareado de bases (McCaskill, 1990b). De manera particular, el paquete RNAfold (Hofacker *et al.*, 1994, 2002) se basa en la hipótesis de que la molécula de ARN se pliega en la estructura termodinámica más estable, es decir, la que tiene mínima energía libre. RNAmmer (Lagesen *et al.*, 2007) predice ARN ribosomal (ARNr) utilizando la unidad ribosomal 5S y la base de datos de ARNr Europea para generar alineamientos estructurales, los cuales se utilizan para la construcción de librerías de cadenas de Markov.

trCYK (Kolbe y Eddy, 2009) y RNA-CODE (Yuan y Sun, 2013) son propuestas recientes que se enfocan en predecir ncRNA utilizando como entrada las lecturas generadas por los NGS. Finalmente, ncRNA-Agents (Arruda *et al.*, 2013) es un predictor que utiliza predictores existentes mediante un enfoque de multiagentes.

Las bases de datos de ncRNA son una parte fundamental de la predicción. Estas bases de datos son creadas a partir de datos experimentales y computacionales. Entre las más conocidas se encuentran NONCODE (Liu *et al.*, 2005), RNAdb (Pang *et al.*, 2007), miRBase (Griffiths-Jones *et al.*, 2006), snoRNA (Lestrade y Weber, 2006), fRNAdb (Mituyama *et al.*, 2009) y RFAM (Griffiths-Jones *et al.*, 2003).

1.5. Objetivo general

Desarrollar un predictor de genes ncRNA basado en clasificación, apoyado por la evidencia biológica del estado del arte y las características intrínsecas de las secuencias de ADN.

1.6. Objetivos específicos

- Recopilar evidencia biológica de la literatura para la identificación de ncRNA.
- Proponer nuevos conjuntos de características para el problema de clasificación de ncRNA basados en diversos niveles de organización estructural.
- Determinar qué subconjunto de características se adapta mejor a un conjunto de familias de ncRNA en particular.
- Construir clasificadores para diversas familias de ncRNA y comparar con los existentes en la literatura.

1.7. Contribuciones

Las contribuciones principales de este trabajo se listan a continuación:

- La extracción de un conjunto de características basado en la secuencia primaria del ARN para la clasificación de 16 familias de *riboswitches* y un desempeño de clasificación mayor al reportado en el estado del arte, además de la propuesta de una visualización de las frecuencias *k*-méricas en un conjunto de secuencias multiclase y su agrupamiento para permitir una discriminación visual. Este trabajo

fue publicado en una revista internacional arbitrada (Guillén-Ramírez y Martínez-Pérez, 2018).

- La extracción de un conjunto de características basado en la secuencia primaria del ARN para la clasificación de ARN inmunomodulador y un desempeño de clasificación mayor al reportado en el estado del arte. Este trabajo resultó en una colaboración de integrantes del Laboratorio de Biocomputación y fue publicado en un congreso internacional (Guillen-Ramirez *et al.*, 2017).
- La descripción de los retos que presenta el área de investigación de predicción de genes de ncRNA. Este trabajo fue presentado en una conferencia nacional (Guillen-Ramirez y Martinez-Perez, 2014).
- El desarrollo de dos funciones de puntaje para frecuencias *k*-méricas y tres métodos potenciales de selección de *k*-meros para clasificación de secuencias.
- La propuesta de un *framework* que permite generar representaciones basadas en grafos de la estructura secundaria del ARN.

1.8. Organización de la tesis

La organización de esta tesis es la siguiente:

En el Capítulo 2 se presentan conceptos básicos generales relacionados a computación, biología molecular y bioinformática del ARN. Cada capítulo subsecuente es autocontenido respecto a los conceptos específicos que llegase a abordar y al trabajo relacionado al mismo.

En el Capítulo 3 se presenta la clasificación de secuencias de riboswitches putativos utilizando un conjunto extendido de características a nivel de secuencia, del cuál se seleccionó un subconjunto que mejora la discriminación. Con la intención de hacer un análisis más profundo, se propone una representación visual de las características presentadas llamada covMap. Finalmente, se comparan los resultados de clasificación con el enfoque previo (Singh y Singh, 2017) y una búsqueda con BLAST (Camacho *et al.*, 2009).

En el Capítulo 4 se ataca el problema de la clasificación de ARN inmunomodulador (IMORN). Primero, se propone un conjunto de características basadas en secuencia, así como la utilización de un algoritmo de selección de características y distintos métodos de clasificación para mejorar los resultados previos en la discriminación de IMORNs (Chaudhary *et al.*, 2016). En el capítulo se demuestra que el uso de características a nivel de secuencia permite obtener resultados similares en sensibilidad, mientras que se mejoran las métricas de precisión, especificidad y el coeficiente de correlación de Matthews (MCC).

En el Capítulo 5 se profundiza en la noción de que los k-meros encontrados en los capítulos 3 y 4 pudieran considerarse motivos discriminantes multiclase. En primer lugar se introduce la función de evaluación q para evaluar de manera rápida la unicidad de un k-mero, así como el conjunto de éstos con la función de cobertura cov. En conjunto, ambas funciones conforman el framework QCOV. Finalmente, se muestran tres estrategias para la selección de k-meros basadas en este framework y se comparan con los seleccionados en el Capítulo 3.

El Capítulo 6 introduce las representaciones alternativas de la estructura secundaria. En primer lugar se presenta un panorama general de las tareas del procesamiento de secuencias de ncRNA que involucran representaciones alternativas de la estructura secundaria. Segundo, se presenta el framework `NR-PR`, el cuál permite generar otras representaciones basadas en grafos. Tercero, se abordan las posibles aplicaciones de una de las representaciones propuestas sobre un conjunto de secuencias de riboswitches, en particular, la búsqueda de motivos estructurales.

Finalmente, el Capítulo 7 presenta un sumario de los puntos más relevantes de este trabajo, así como las conclusiones finales y algunas ideas de trabajo a futuro.

Capítulo 2. Marco Teórico

2.1. Fundamentos de computación

Los conceptos que se abordan en esta sección fueron recuperados de los libros de Hopcroft y Ullman (1979) y Cormen *et al.* (2009).

2.1.1. Teoría de conjuntos

2.1.1.1. Conjuntos

Un conjunto es una colección de objetos no repetidos llamados elementos o miembros del conjunto. Si un objeto x es un elemento del conjunto A, entonces se dice que $x \in A$, y en caso contrario, $x \notin A$. Se puede describir un conjunto listando explícitamente sus miembros en una lista dentro de llaves, o mediante una forma compacta que exprese las características de los miembros. En un conjunto no importa el orden de los elementos. El número de elementos en un conjunto A se denomina la cardinalidad de A y se escribe $|A|$. Al conjunto sin elementos se le denomina conjunto vacío y se denota como $\emptyset$.

Si todos los elementos de un conjunto A están contenidos dentro de un conjunto B, esto es, si $x \in A$ implica que $x \in B$, entonces A es un subconjunto de B y se denota como $A \subseteq B$. Para cualquier conjunto A, $\emptyset \subseteq A$ y $A \subseteq A$. Para tres conjuntos cualesquiera A, B, y C, si $A \subseteq B$ y $B \subseteq C$, entonces $A \subseteq C$. Se dice que si $A \subseteq B$ y $B \subseteq A$ entonces $A = B$. Si $A \subseteq B$ pero $A \neq B$ entonces decimos que A es un subconjunto propio de B y se denota como $A \subset B$.

Siendo A y B conjuntos, definimos las siguientes operaciones:

- *Intersección.*

$$A \cap B = \{x \mid x \in A \ \wedge \ x \in B\}. \tag{1}$$

Se dice que dos conjuntos A y B son disjuntos si no tienen elementos en común, esto es, si $A \cap B = \emptyset$.

- *Unión.*

$$A \cup B = \{x \mid x \in A \ \vee \ x \in B\}. \tag{2}$$

- *Diferencia o resta.*

$$A \backslash B = \{x \mid x \in A \ \wedge \ x \notin B\}. \tag{3}$$

2.1.1.2. Particiones

Una colección $P = \{S_{i \in I}\}$ de conjuntos no vacíos indizados por I forma una partición de un conjunto $\mathcal{S}$ si 1) los conjuntos son disjuntos entre sí, esto es, para todo $S_i, S_j \in P, i \neq j$, implica que $S_i \cap S_j = \emptyset$; y 2) su unión es $\mathcal{S}$, esto es:

$$\mathcal{S} = \bigcup_{i \in I} S_i. \tag{4}$$

En otras palabras, P forma una partición de $\mathcal{S}$ si cada elemento de $\mathcal{S}$ aparece en exactamente un $S_i \in P$.

2.1.2. Lenguajes formales

Un alfabeto es un conjunto finito no vacío cuyos elementos reciben el nombre de símbolos. Cualquier secuencia de símbolos w de un alfabeto Σ se llama una cadena sobre Σ. La longitud de w es la cantidad de símbolos que contiene y se denota como $|w|$. La cantidad de apariciones de un símbolo $a \in \Sigma$ en la palabra w se denota como $|w|_a$. Dadas dos palabras w_1 y w_2 sobre un alfabeto Σ, la operación de concatenación en w_1 y w_2 se denota como $w_1 w_2$ o $w_1 \cdot w_2$ y produce una palabra que consiste en

los símbolos de w_1 en el mismo orden, seguidos por los símbolos de w_2 en el mismo orden.

Para un alfabeto Σ se denota por Σ^* la cerradura de Kleene, la cual es el conjunto (infinito) de todas las cadenas sobre Σ. La cadena vacía se denota por λ, y el conjunto de cadenas no vacías sobre Σ, $\Sigma^+ = \Sigma^* \backslash \{\lambda\}$ se llama la cerradura positiva. Cada subconjunto de Σ^* se llama un lenguaje sobre Σ.

Para cada palabra $w \in \Sigma^*$ se define la operación potencia como:

(i) $w^0 = \lambda$, y

(ii) $w^{n+1} = ww^n$ para $n \in \mathbb{N}^+$.

Si $w = w_1 w_2 w_3$ para alguna $w_1, w_2, w_3 \in \Sigma^*$, entonces w_2 es una subcadena de w; si $w_1 \neq \lambda$ entonces w_1 es un prefijo de w, y si $w_3 \neq \lambda$, entonces w_3 es un sufijo de w. Los conjuntos de todos los prefijos, sufijos y subcadenas de una cadena w se denotan por *Pref(w), Suf(w), Sub(w)*, respectivamente.

Dado que los lenguajes son conjuntos, se pueden realizar las mismas operaciones de unión, intersección, resta y complemento. La concatenación de dos lenguajes L_1, L_2 es una operación que produce el lenguaje $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Mediante esta operación se define la exponenciación de un lenguaje L:

$$L^0 = \{\lambda\}, \tag{5}$$

$$L^{i+1} = LL^i, i \geq 0. \tag{6}$$

De la misma manera se puede definir las cerraduras de un lenguaje L:

$$L^* = \bigcup_{i=0}^{\infty} L^i, \tag{7}$$

$$L^+ = \bigcup_{i=1}^{\infty} L^i. \tag{8}$$

2.1.3. Teoría de grafos

Un grafo no dirigido es un par $G = (V, E)$, donde V es un conjunto finito llamado el conjunto de vértices, y $E \subseteq V \times V$ es un conjunto de pares no ordenados de vértices (v_i, v_j) denominados aristas, tal que $v_i, v_j \in V$ e $i \neq j$. El orden de un grafo G es el número de vértices y está dado por $|V|$. Una forma de representar un grafo G de orden n es mediante una matriz $n \times n$ booleana simétrica $A = (a_{ij})$ conocida como la matriz de adyacencias de G, donde $a_{ij} = 1$ si y sólo si $i \neq j$ y $(v_i, v_j) \in E$. Otra representación de grafos es mediante una lista de adyacencias, la cual consiste en un arreglo Adj de n listas, una para cada vértice en V. Para cada $v_i \in V$, la lista de adyacencias $Adj[v_i]$ contiene todos los vértices v_j tales que exista una arista $(v_i, v_j) \in E$.

Cualquier arista (v_i, v_j) de un grafo $G = (V, E)$ es incidente a los vértices v_i y v_j. Dos vértices u y v son adyacentes si la arista $(u, v) \in E$. El grado de un vértice v de G, $deg_G(v)$, es el número de aristas incidentes a v. El grado de un grafo $G = (V, E)$ es

$$deg(G) = max\{deg_G(v) \mid v \in V\}. \tag{9}$$

El vecindario $N(v_i)$ de vértices adyacentes a v_i se define como:

$$N(v_i) = \{v_j \mid (v_i, v_j) \in E\}. \tag{10}$$

Un grafo es dirigido cuando $(v_i, v_j) \neq (v_j, v_i)$ si $i \neq j$, en otras palabras, las aristas tienen un sentido. La definición de grado de un vértice se sustituye por las funciones

grado de entrada (*indegree*) y grado de salida (*outdegree*), donde:

$$indegree(v_i) = |\{v_j \mid (v_j, v_i) \in E\}| \text{ y} \tag{11}$$

$$outdegree(v_i) = |\{v_j \mid (v_i, v_j) \in E\}|. \tag{12}$$

Una trayectoria es una secuencia de vértices tal que cada par consecutivo de vértices está conectado por una arista. Un ciclo es un conjunto de vértices tal que existe una trayectoria donde un vértice puede aparecer dos veces. Una trayectoria cerrada es aquella que inicia y termina en el mismo vértice. Un ciclo simple es una trayectoria cerrada donde no se repiten vértices ni aristas. Un árbol es un grafo dirigido sin ciclos. Los vértices del árbol que tienen grado de salida igual a cero son llamados hojas.

2.2. Biología molecular

Los conceptos utilizados en esta sección están condensados del libro de Alberts *et al.* (2014).

2.2.1. ADN y ARN

El ADN, ácido desoxirribonucleico, es una molécula que codifica, regula y transmite las características de un ser vivo. El ARN, ácido ribonucleico, es un ácido nucléico con diversas funciones dentro de la célula. El ADN y ARN se forman de subunidades llamadas nucleótidos, las cuales se componen de tres partes: un grupo fosfato, un azúcar desoxirribosa (ribosa en el caso del ARN) y una de cuatro bases nitrogenadas, las cuales se dividen en purinas (adenina (`A`) y guanina (`G`)) y pirimidinas (timina (`T`) y citosina (`C`)). En el ARN, la timina es sustituida por el uracilo (`U`).

La desoxirribosa se compone de un anillo de cinco carbonos ($1', \ldots, 5'$). El grupo fosfato (P) une el carbono 5′ de la desoxirribosa de un nucleótido con un grupo hidroxilo en el carbono 3′ del siguiente. Una cadena de ADN tendrá un grupo fosfato libre en un extremo y un grupo hidroxilo (OH) en el otro. Por convención, el sentido de la cadena es en esta dirección y se denota como 5′ a 3′.

Las bases nitrogenadas forman puentes de hidrógeno entre sí, por lo que dos cadenas sencillas de ADN pueden enlazarse entre sí para formar una cadena doble de ADN en forma de hélice. La complementariedad se da entre los pares de bases A–T (A–U en el ARN) y G–C, lo cual se denomina como la complementariedad de Watson-Crick. Sin embargo, también hay un tipo de enlaces que no siguen esta regla (G–U) y son llamados enlaces *wobble*.

Para que se pueda formar la hebra doble, es necesario que las cadenas sencillas estén en sentidos opuestos y que las bases en las posiciones correspondientes del alineamiento sean complementarias. Los enlaces que unen una cadena doble pueden ser rotos mediante calor. Este fenómeno es llamado desnaturalización. La cadena desnaturalizada puede volver a unirse mediante enfriamiento lento, lo cual es llamado hibridación o alineamiento.

2.2.2. Genes codificantes de proteínas

Las células de todos los organismos vivos se dividen en dos grandes grupos: las células procariotas, que tienen su material genético flotando sin una membrana que lo delimite, y las eucariotas, que tienen un núcleo bien definido que contiene dicho material. Las células procariotas abarcan un gran número de seres unicelulares como los ciliados y bacterias. Por otro lado, las eucariotas forman parte de todos los seres multicelulares, además de seres unicelulares complejos como la levadura.

Un gen es un segmento de ADN que codifica para un producto funcional determinado (por ejemplo una proteína o una molécula de ARN no codificante (ncRNA)). En los organismos procariotas los genes son segmentos codificantes largos conocidos como marcos de lectura abiertos (ORF por sus siglas en inglés), mientras que en los eucariotas son regiones compuestas por subregiones codificantes (exones) y subregiones no codificantes (intrones). La región que divide un exón de un intrón es llamado sitio de *splicing*. El límite exón-intrón es llamado *sitio donador*, mientras que el límite intrón-exón es el *sitio aceptador* (Figura 1). Los sitios donadores y de aceptación canónicos son GT y AG respectivamente, sin embargo, se ha encontrado evidencia biológica de que existen otros sitios. La parte del ADN que no contiene ningún gen es conocida

como región intergénica.

Los exones se clasifican en iniciales (se extienden desde el codón de inicio hasta el primer donador), internos (ubicados entre un sitio de aceptación y el siguiente sitio donador) y los finales (se extienden desde el último sitio de aceptación hasta el codón de terminación). Los exones sencillos ocurren en los genes sin intrones, y se extienden desde el codón de inicio hasta el de terminación (Figura 2).

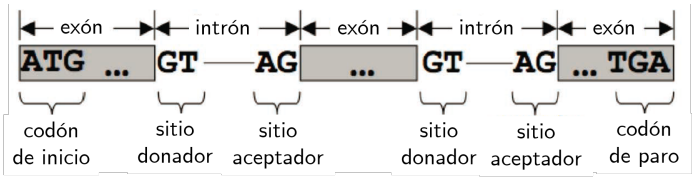

Figura 1: Estructura de un gen eucariota.

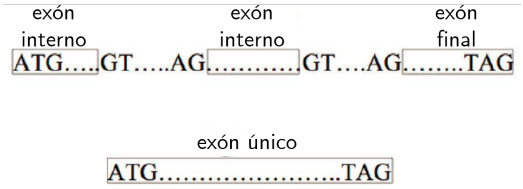

Figura 2: Tipos de exones.

De manera general, la expresión genética es el proceso donde la información codificada por el ADN se convierte en un producto génico.

2.2.3. Genes de ARN

Antes se creía que los productos génicos sólo eran proteínas, sin embargo también abarcan distintos tipos de ARN funcionales conocidos como ARN no codificantes (ncRNA), entre los que se encuentran los microARN, ARN de transferencia, ARN ribosomal y los riboswitches (Clancy, 2008).

2.2.3.1. Evolución y conservación de los genes de ARN

Una secuencia de ARN conservada es aquella que es altamente similar entre varias especies. Un ejemplo de ARN altamente conservado son los micro-ARN y el ARN nucleolar pequeño (miRNA y snoRNA por sus siglas en inglés, respectivamente) (Pang

et al., 2006). Sin embargo, la gran mayoría de los genes de ARN no son conservados a nivel primario, sino a nivel secundario y terciario (Menzel *et al.*, 2009). Por ejemplo, un par G–C en un organismo pudo haber evolucionado en un par A–U en otro. Esto se llama un cambio de base compensatorio.

2.3. Bioinformática del ARN

2.3.1. Estructura secundaria y terciaria del ARN

Es posible que una hebra sencilla de ADN y ARN se alinee (pliegue) consigo misma formando una estructura en el espacio tridimensional. La vista de la estructura en 2D es llamada estructura secundaria, y terciaria para 3D. La predicción de estas estructuras es un área de investigación activa en bioinformática del ARN (Gorodkin y Hofacker, 2011a).

En la estructura secundaria es posible distinguir pares de bases unidos, los cuales pueden ser canónicos (A–U y C–G) o *wobble* (G–U). Existen varias representaciones de la estructura secundaria del ARN, algunas de las cuales se muestran en la Figura 3. Otras representaciones son en árbol, conjunto de pares de bases y arcos anotados.

Es posible identificar subestructuras comunes según los patrones de bases pareadas y no pareadas (Figura 4). Los *stems* (también llamados hélices dobles, segmento de doble cadena o *duplex*) son regiones pareadas completamente con su cadena complementaria. Los *bulges* son porciones consecutivas no pareadas de uno o más nucleótidos de longitud en una parte de una hélice. Los *internal loops* se presentan cuando existen *bulges* en una región no pareada en ambos extremos de la hélice. Son llamados simétricos cuando ambos segmentos son de la misma longitud y asimétricos en caso contrario. Finalmente, los *hairpins* o *stem-loops* son hélices con vértices no pareados en el extremo (Tian *et al.*, 2004).

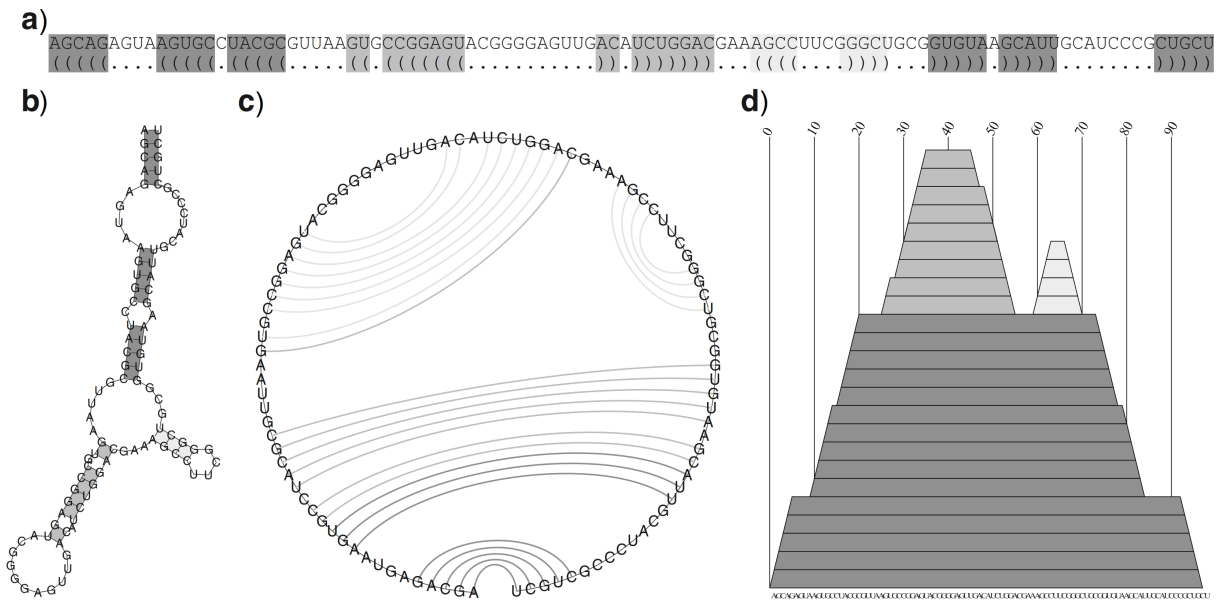

Figura 3: Representaciones de la estructura secundaria del riboswitch tetrahidrofolato. a) Representación punto-paréntesis. b) Diagrama de estructura secundaria (*squiggle plot*). c) Gráfico circular (outerplanar). D) Gráfico de montaña. Tomado de Gorodkin *et al.* (2014).

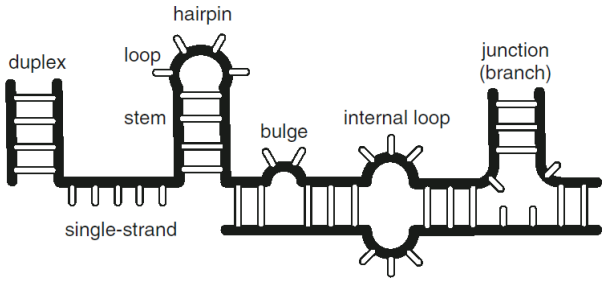

Figura 4: Convenciones para las subestructuras de ARN. Tomado de Zwieb (2014).

En la figura se puede apreciar la representación de grafo outerplanar de la Figura 3(c) que las líneas que representan los enlaces entre pares de bases no se cruzan. Sin embargo, los cruces se dan en la naturaleza, y son llamados pseudonudos (Figura 5).

2.3.2. Alineamientos

El alineamiento de secuencias es un proceso que compara dos o más secuencias biológicas (ADN, ARN o proteínas) con el fin de inferir si son homólogas, es decir, si comparten una historia evolutiva común (Rosenberg, 2009). El resultado de un alinea-

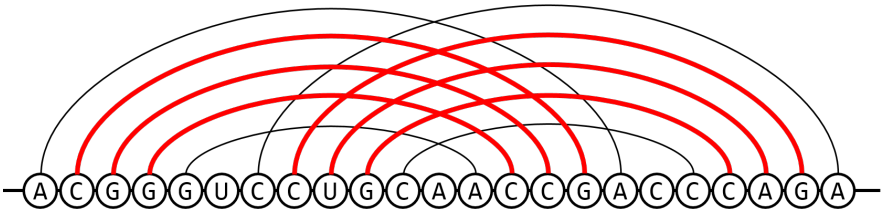

Figura 5: Pseudonudo canónico en la estructura secundaria del ARN. Adaptado de Washietl *et al.* (2012).

miento es una medida de similitud entre las dos secuencias, y puede ser a cualquiera de los tres niveles de estructura. Sin embargo, una alta similitud en la estructura secundaria de dos cadenas de ARN no implica que tengan una alta similitud en su secuencia de bases (Figura 6).

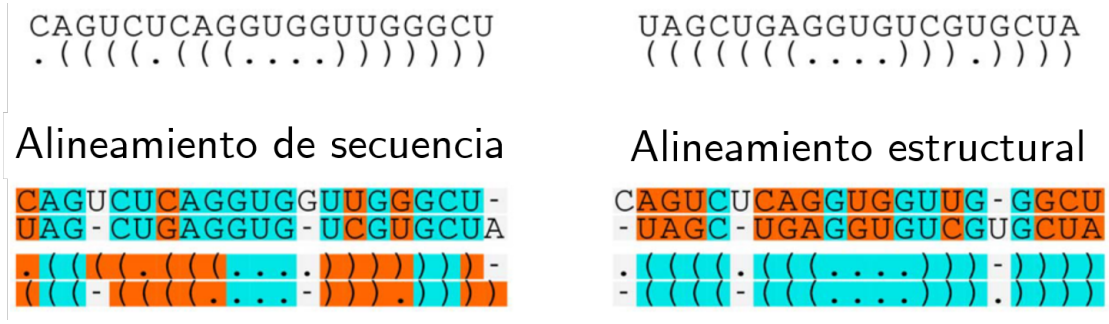

Figura 6: Ejemplo de alineamientos en dos secuencias. Realizar el alineamiento a nivel de secuencia (parte izquierda) no implica un alineamiento correcto a nivel de estructura (parte derecha). Adaptado de Gorodkin y Hofacker (2011a).

2.3.3. El proceso de anotación genómica

La anotación es el proceso de añadir información a las secuencias genómicas, por ejemplo su procedencia y función, entre otras. La anotación se realiza de manera experimental, sin embargo, realizar la anotación de todos los genomas es inviable por cuestiones de tiempo y esfuerzo, por lo que existen herramientas computacionales de soporte. Una parte importante del proceso de anotación de un nuevo genoma es la identificación de los genes, tanto en ubicación como en función. Para la localización, se utilizan algoritmos llamados predictores de genes.

2.3.3.1. Predicción de genes codificantes

El problema de la predicción de genes consiste en identificar las porciones de ADN que son biológicamente funcionales. Tradicionalmente, las regiones de interés son aquellas que codifican para proteínas, sin embargo, también se consideran elementos como el ncRNA. En esta sección se trata únicamente el problema de la predicción de genes codificantes de proteínas en organismos eucariotas.

El problema puede enunciarse como uno de clasificación, donde el objetivo es etiquetar correctamente cada elemento de la secuencia de ADN como perteneciente a un exón, o una región no codificante (un intrón, región intergénica o región sin traducir). A continuación se presenta el enunciado formal del problema propuesto por Bandyopadhyay *et al.* (2008):

- *Entrada:* una secuencia de ADN

$$X = (x_1, \ldots, x_n) \in \Sigma^*, \text{ donde } \Sigma = \{\texttt{A}, \texttt{T}, \texttt{C}, \texttt{G}\}. \tag{13}$$

- *Salida:* el etiquetado correcto de cada elemento en X perteneciendo a un exón, o a una región no codificante.

Por otro lado, el problema puede interpretarse como la extracción de exones putativos en una secuencia de ADN (Majoros, 2007):

- *Entrada:* una secuencia de ADN

$$X = (x_1, \ldots, x_n) \in \Sigma^*, \text{ donde } \Sigma = \{\texttt{A}, \texttt{T}, \texttt{C}, \texttt{G}\} \tag{14}$$

  y una función de puntuación de exones $f(x)$.

- *Salida:* el conjunto de pares $(x_i, x_j), 1 \leq i < j \leq n$, tales que la subregión $X[i:j]$ representa un exón putativo y $f(X[i:j])$ es máximo.

Los métodos de predicción pueden descomponerse en dos partes: la recopilación de evidencia (detección de señales y sensado de contenido) y la integración de la evidencia mediante un modelo de estructura de genes (Mathé *et al.*, 2002).

2.3.3.2. Sensores de señales

Los sensores de señales identifican la presencia de motivos en una secuencia que son reconocidos por la maquinaria celular. En general, las señales se relacionan con los procesos de traducción, *splicing* y transcripción. Dentro de las señales de traducción y *splicing* suelen considerarse los sitios donadores y aceptadores canónicos (`GT-AG`). El conjunto mínimo de señales que describe la estructura de una secuencia codificante incluye los codones de inicio (`ATG`) y terminación (`TGA`, `TAG` o `TAA`), y si es una secuencia multiexónica, los sitios donadores y aceptadores para cada intrón presente. Adicionalmente, se consideran los puntos de quiebre en las secuencias multiexónicas (`CU[A/G]A[C/U]`) ubicados de 20 a 50 bases río arriba del aceptador `AG`.

Otras señales utilizadas son los potenciadores y silenciadores. Los sensores transcripcionales incluyen la caja `TATA` y la señal de poliadenilación (el hexámero consenso `AATAAA`). Respecto a los sensores de traducción, se incluye la secuencia consenso de Kozak (`[A/G]XXAUGG`).

Algunas señales pueden ser modeladas como matrices de peso posicional, las cuales intentan capturar la variabilidad intrínseca de los patrones. Las matrices se obtienen a partir de secuencias alineadas con funcionalidad relacionada. Las matrices de pesos fallan en capturar las dependencias entre diferentes posiciones, por lo que se utilizan las matrices de arreglos de pesos. Las probabilidades en la matriz se calculan como probabilidades condicionales de que una determinada secuencia sea una instancia de un motivo en particular. Debido a que el estado en una posición está condicionado sólo por el estado que lo precede inmediatamente, se cumple la suposición de Markov (por lo que son cadenas de orden cero y primer orden). Otros enfoques para modelar estas relaciones son la descomposición de dependencia maximal (para posiciones no adyacentes), redes bayesianas (para la predicción de sitios donadores) y máquinas de vectores de soporte (para la identificación de los sitios de *splicing*).

2.3.3.3. Sensores de contenido

En teoría, una vez identificadas todas las señales de una secuencia, la estructura exón-intrón debería estar completamente especificada. Sin embargo, la predicción del inicio y fin de la transcripción, así como la clasificación todos los codones potenciales de inicio y los sitios de *splicing* es todavía un reto. Para esto surgen los sensores de contenido, los cuales son medidas que intentan discriminar una región de ADN en codificante o no codificante.

- Extrínsecos (homologías). Los sensores de contenido extrínsecos toman una secuencia de ADN genómica y calculan su similitud con alguna evidencia existente en las bases de datos. Es posible utilizar tres tipos diferentes de secuencias para encontrar la información sobre las regiones codificantes del genoma: secuencias de proteínas, ADN codificante (cDNA por sus siglas en inglés) o marcadores de secuencias expresadas (EST por sus siglas en inglés) y secuencias de ADN.

  El estándar en la anotación de la estructura exónica es el alineamiento de secuencias completas de cDNA derivadas de la misma especie (idealmente del mismo individuo) a la secuencia genómica de interés. Cuando la secuencia homóloga procede de la misma especie y región de ADN que la secuencia de interés, entonces basta con resolver el problema del *splicing alignment* (Gelfand *et al.*, 1996). En el caso que se tengan todos los sitios *splicing* identificados, el problema se reduce al de los genes procariotas, es decir, encontrar el ORF más grande, discriminando entre los codones de inicio por el más probable. Un método de discriminación es la secuencia consenso de Kozak (`[A/G]XXAUGG`). La limitación de los sensores de contenido extrínsecos es la baja calidad de las bases de datos, la eficacia insuficiente y la falta de sensibilidad ante exones pequeños.

- Intrínsecos. Los sensores de contenido intrínsecos se basan en el hecho de que tres bases sucesivas en el marco correcto definen un codón que se traduce en un aminoácido específico en la proteína. Debido a que en las secuencias eucariotas la región traducida es muy pequeña, se utilizan otras métricas para categorizar la secuencia, por ejemplo, el uso de codones, el contenido `G+C`, composición de nucleótidos, frecuencia de hexámeros y periodicidad de ocurrencia de bases. Ac-

tualmente los programas utilizan dos tipos de sensores de contenido: uno para las secuencias codificantes y otro para las no codificantes (intrones, regiones terminales sin traducir (UTRs) y regiones intergénicas).

2.3.3.4. Integración de la evidencia

Según la forma en que integra la evidencia adquirida, un predictor se puede clasificar en empírico, *ab initio*, o de enfoque combinado (Figura 7).

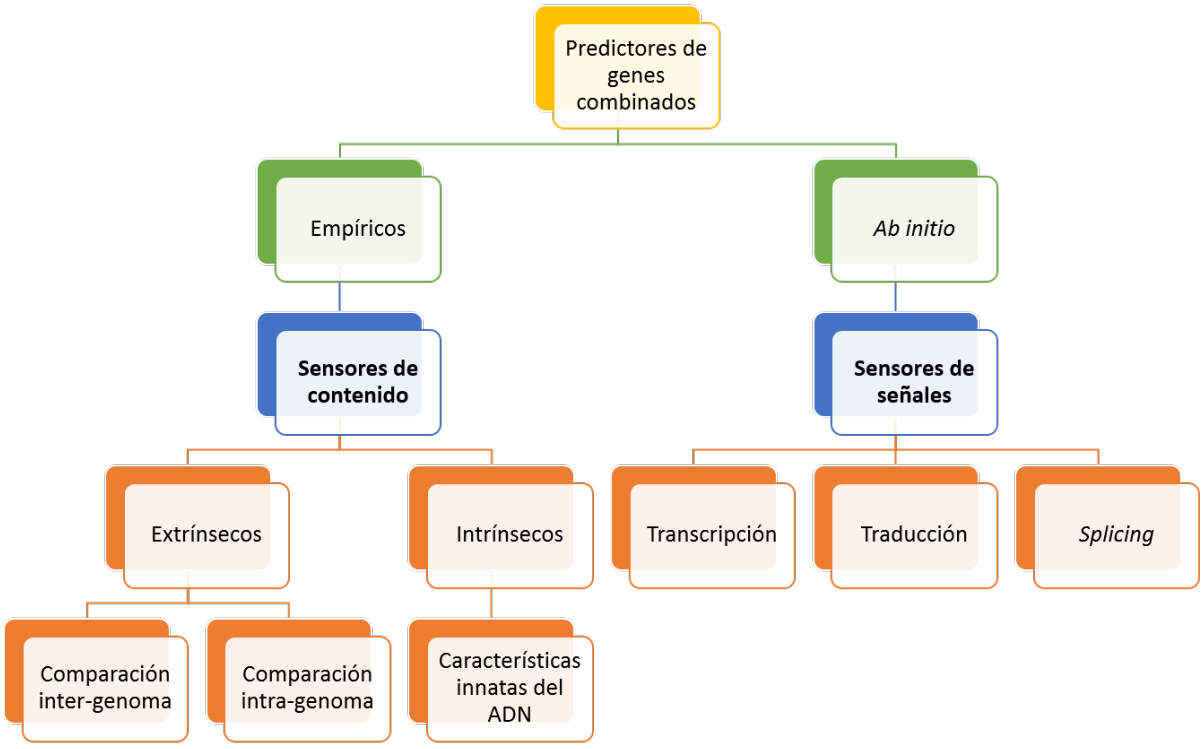

Figura 7: Clasificación de predictores en base a la integración de la evidencia. Adaptado de Sleator (2010)

Los predictores que no utilizan homologías para resolver el problema son llamados *ab initio*. Las técnicas utilizadas en la mayoría de estos predictores son la programación dinámica y los modelos ocultos de Markov (HMM por sus siglas en inglés) (Alioto, 2012).

2.3.4. Predicción de ncRNA

La ubicación del problema de predicción de genes dentro del contexto de la bioinformática del ARN se muestra en la Figura 8.

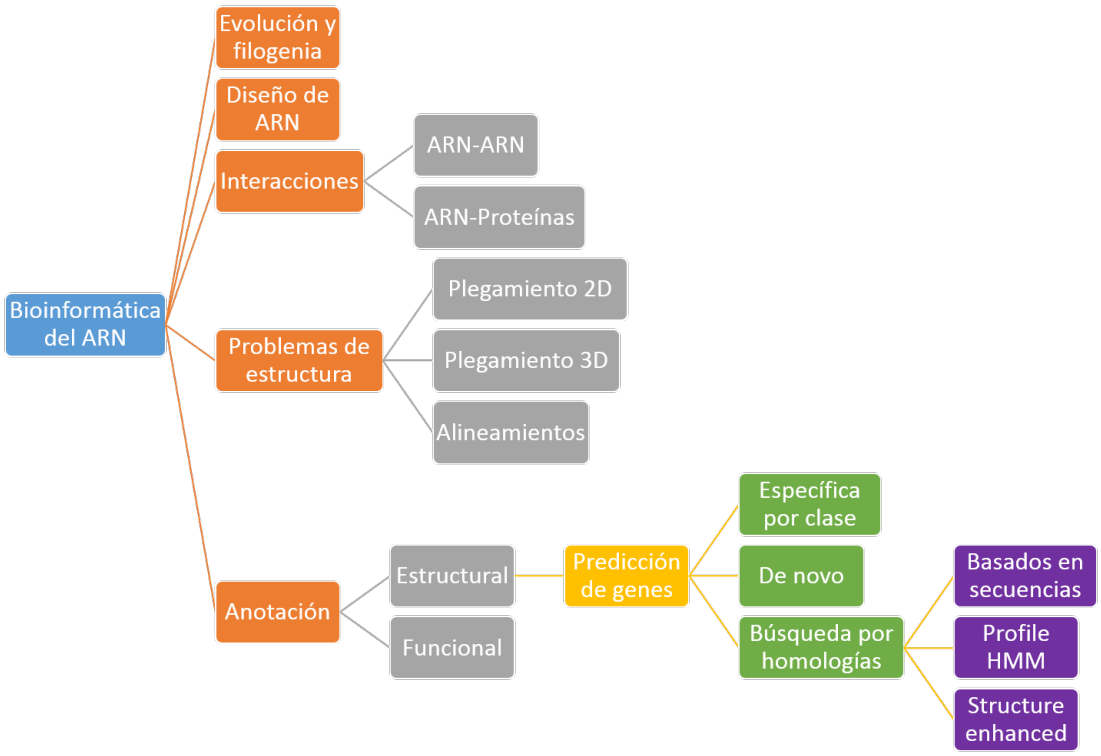

Figura 8: Algunos problemas relacionados a la bioinformática del ARN.

Por lo general, en los predictores de genes codificantes no se toman en consideración los intrones ni las regiones intergénicas, sin embargo, estas regiones posiblemente contengan alguna función reguladora. Los métodos para identificar y clasificar el ncRNA en estas regiones utilizan diferentes estrategias basadas en las características soportadas por evidencia experimental e *in silico*.

2.3.4.1. *De novo*

Debido a que los predictores *de novo* se basan solamente en las características del genoma, la estrategia general que utilizan consiste en utilizar ventanas y predecir el plegamiento del ARN en la ventana utilizando un modelo de energía libre. Sin embargo, se ha demostrado que las estructuras predichas de los genes de ARN no

son significativamente más estables que secuencias aleatorias generadas a partir del trasfondo genómico, por lo que no puede usarse ese criterio para distinguirlos (Rivas y Eddy, 2000). Como resultado, los esfuerzos se han enfocado en integrar al modelo información de distintas fuentes que puedan generar una estructura secundaria consenso.

2.3.4.2. Búsqueda por homologías y búsqueda específica de familias

La búsqueda por homologías consiste en utilizar clases de ARN conocidas (en este contexto también denominadas familias) en una porción de algún genoma. En el caso de la búsqueda específica de familias, los métodos se basan en entrenar modelos de aprendizaje máquina dedicados a ese tipo de ARN.

La búsqueda por homologías (al igual que la predicción *de novo*) está íntimamente relacionada con los problemas estructurales en el ARN: predicción del plegamiento y alineamientos estructurales. De forma bastante genérica, un predictor podría seleccionar una ventana de n-nucleótidos, tratar de predecir la estructura secundaria de esa secuencia de tamaño n, y por medio de alineamiento estructural evaluar la similitud con las familias que contenga la base de datos. Sin embargo, la sola estructura secundaria no es un hecho determinante en la similitud, dado que como se menciona en la Sección 2.3.4.1, si la estructura se predice con un modelo de energía libre, no es tan relevante como para distinguirla de una secuencia aleatoria. Por ello, los métodos de búsqueda por homologías utilizan diferentes evidencias para generar las estructuras secundarias ya sea de las familias, o de la consulta. Entre estos métodos se encuentra la unión de la información de alineamientos a nivel primario y la generalización de los modelos ocultos de Markov llamada gramáticas estocásticas libres de contexto (SCFG, por sus siglas en inglés).

Debido a las particularidades de los genes ncRNA, los criterios comúnmente considerados al intentar detectarlos son los siguientes:

- Un gran número de clases conocidas de ncRNA no presentan ORFs largos.

- Sus secuencias tienen codones de terminación inesperados.
- Las moléculas de ARN usualmente presentan conservación en su estructura secundaria y raramente en su estructura primaria.
- Algunos ncRNA conocidos tienen estructuras tridimensionales complejas, así como funciones catalíticas o estructurales.

Estas características impiden el uso de las homologías desarrolladas para la clasificación de genes codificantes de proteínas (Pang *et al.*, 2006), por lo que un enfoque común en los predictores es identificar los genes de ncRNA utilizando estructuras secundarias y motivos. Ejemplos de estas herramientas incluyen tRNAscan-SE (Lowe y Eddy, 1997) y Snoscan (Schattner *et al.*, 2005), y la propuesta por Lewis *et al.* (2003) para el caso de microARN (miRNA).

Un enfoque más general para resolver el problema consiste en primero alinear las secuencias de nucleótidos (genómicas, RNA-seq y ESTs) de organismos relacionados cercanamente al genoma objetivo y entonces buscar por señales de estructuras secundarias conservadas. No obstante, este es un proceso complejo que todavía arroja una alta tasa de falsos positivos y requiere recursos computacionales sustanciales (Yandell y Ence, 2012). Ejemplos de estas herramientas son qRNA (Eddy, 2002), StemLoc (Holmes y Rubin, 2002), pmcomp y pmmulti (Hofacker *et al.*, 2004) e Infernal (Nawrocki *et al.*, 2009) el cual utiliza la base de datos Rfam (Griffiths-Jones *et al.*, 2003).

En la práctica, es posible unir y complementar los enfoques *de novo*, específicos y por homologías para desarrollar un flujo de trabajo (*pipeline*) para la predicción (Figura 9).

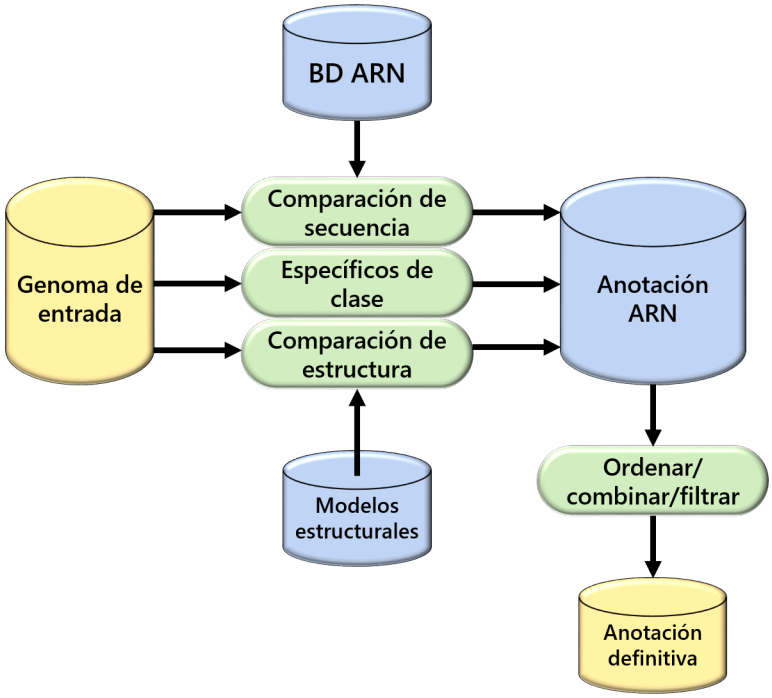

Figura 9: *Pipeline* para la predicción de genes ncRNA. Adaptado de Gorodkin y Hofacker (2011a).

Capítulo 3. Clasificación de riboswitches utilizando frecuencias *k*-méricas

3.1. Introducción

Los riboswitches (también llamados riboconmutadores) son moléculas de ARN no codificante que regulan la expresión génica a nivel de ARN mensajero cuando se unen a ligandos pequeños sin la necesidad de moléculas intermediarias (Mandal *et al.*, 2003). Un riboswitch se compone de dos regiones funcionales: el *aptámero* que detecta un ligando específico y la *plataforma de expresión*, la cual cambia su conformación cuando el ligando se encuentra unido. La plataforma de expresión comúnmente se encuentra junto al aptámero (río abajo) y se unen por una secuencia en común que puede alinearse con cualquiera de los dos dominios. Ambos dominios, aptámero y plataforma de expresión, difieren entre sí en términos de conservación de secuencia y estructura. Mientras que las plataformas de expresión se encuentran pobremente preservadas, las regiones de aptámero tienen a estar muy conservadas incluso en distancias evolutivas grandes (Breaker, 2012). Es mediante la estructura tridimensional del aptámero que un riboswitch detecta y se une a moléculas objetivo (Aghdam *et al.*, 2016) tales como purinas y sus derivados, coenzimas y compuestos relacionados, aminoácidos, azúcares fosforilados y metales (Serganov y Nudler, 2013).

Sin embargo, el interés en riboswitches va más allá de la regulación génica. Su rol como objetivo de fármacos para el desarrollo de nuevos tipos de antibióticos aparenta ser prometedor porque 1) la unión al ligando es altamente selectiva y específica, 2) la mayoría de los riboswitches se encuentran solo en bacterias y 3) los riboswitches usualmente se asocian con transcritos de ARNm que codifican proteínas de supervivencia y virulencia (Serganov y Nudler, 2013; Aghdam *et al.*, 2016; Blount y Breaker, 2006; Howe *et al.*, 2015). Otra área de investigación relacionada es el diseño de riboswitches que actúen como biosensores en aplicaciones que requieran el ligado específico a ciertas moléculas (Aghdam *et al.*, 2016). No obstante, la eficiencia de estos sensores artificiales todavía no es comparable a la de sus contrapartes naturales y la mejora en el diseño es una tarea difícil que demanda el apoyo de enfoques compu-

tacionales (Berens y Suess, 2015).

Los enfoques *in silico* son capaces de encontrar clases conocidas de riboswitches en un amplio espectro de organismos buscando en genomas completos (Lesnik *et al.*, 2005; Nawrocki y Eddy, 2013; Bengert y Dandekar, 2004; Abreu-Goodger y Merino, 2005; Chang *et al.*, 2009; Singh *et al.*, 2009; Havill *et al.*, 2014; Zhang *et al.*, 2006; Huang *et al.*, 2006; Senter *et al.*, 2012; Chang *et al.*, 2013; Retwitzer *et al.*, 2015; Manzourolajdad y Arnold, 2015; Mukherjee y Sengupta, 2015). Una subtarea del problema de la predicción de riboswitches es la clasificación en familias conocidas de secuencias putativas. Típicamente, la clasificación de ARN no codificante requiere de la extracción de características basadas en la estructura secundaria para compensar la falta de conservación de las secuencias a nivel primario (Karklin *et al.*, 2005; Childs *et al.*, 2009; Gudyś *et al.*, 2013; Panwar *et al.*, 2014). Tal estrategia es necesaria en el descubrimiento de riboswitches cuando la secuencia del aptámero es desconocida (métodos *de novo*). También es de ayuda considerar el modelo biofísico más simple para la operación de riboswitches (llamado el modelo de dos estados), en el cual el ARN mensajero adopta dos conformaciones mutuamente excluyentes conocidas como libre de ligando (*ligand-free*) y unido al ligando (*ligand-bound*) (Fürtig *et al.*, 2015). Los pseudonudos juegan un papel importante en la conformación libre de ligando al permitir la formación de los huecos o bolsillos donde ocurre la unión así como contactos terciarios de larga distancia (Serganov y Nudler, 2013). Por consiguiente, las interacciones a nivel terciario son una fuente potencial de información para modelos de clasificación, aunque esto conlleva un alto costo computacional (Manzourolajdad y Arnold, 2015). Por ejemplo, los métodos exactos para el cálculo de estructuras secundarias que contengan una limitada cantidad de clases de pseudonudos tienen una complejidad en tiempo de al menos $O(n^4)$ (Sato *et al.*, 2011). Además, investigación reciente muestra evidencia de múltiples conformaciones en los estados libre y unido al ligando, lo que incrementa el espacio de búsqueda (Fürtig *et al.*, 2015).

No obstante, cuando la secuencia del aptámero es conocida, su conservación puede explotarse para calcular características que no sean tan demandantes computacionalmente, evitando las derivadas de algoritmos de plegado (2D y 3D) y alineamientos múltiples. Las características basadas en secuencia han sido usadas exitosamente pa-

ra inducir clasificadores capaces de discriminar si un transcrito es codificante o no (Liu *et al.*, 2006; Kong *et al.*, 2007; Panwar *et al.*, 2014; Tripathi *et al.*, 2016). Recientemente, Singh y Singh (2017) aplicaron esta idea a riboswitches utilizando frecuencias a nivel nucleótido y dinucleótido como vectores de características para generar clasificadores. Si bien el cálculo de estos vectores fue realizado en tiempo lineal, el desempeño en la clasificación resulta bajo para las 16 familias de riboswitches evaluadas.

En este capítulo se presenta la clasificación de secuencias de riboswitches putativos utilizando un conjunto extendido de características a nivel de secuencia, del cuál se seleccionó un subconjunto que mejora la discriminación. Con la intención de hacer un análisis más a fondo, se propone una representación visual de las características presentadas llamada covMap. Finalmente, se comparan los resultados obtenidos con el enfoque previo (Singh y Singh, 2017) y una búsqueda con BLAST (Camacho *et al.*, 2009)[1].

3.2. Metodología

3.2.1. Conjunto de datos

Con el objetivo de realizar una comparación justa, el conjunto de datos fue construido según la metodología propuesta por Singh y Singh (2017). Se extrajeron las secuencias de 16 clases de riboswitches de RFAM 11.0 (Tabla 1). RFAM agrupa las secuencias de ARN no codificante en familias, las cuales tienen un identificador con prefijo RF-. De aquí en adelante nos referiremos a una familia en particular usando su identificador de RFAM (columna RFAM_ACC en la Tabla 1) o mediante su descripción.

Respecto al preprocesamiento de los datos, en la versión utilizada de RFAM las secuencias ya se encuentran filtradas a $< 90\,\%$ de identidad. Los conjuntos de entrenamiento y prueba son particiones del conjunto completo y se generaron utilizando muestreo aleatorio estratificado (60 % entrenamiento, 40 % prueba).

[1] Este capítulo es una versión extendida del artículo publicado en la revista Biosystems (Guillén-Ramírez y Martínez-Pérez, 2018).

Tabla 1: Número total de secuencias por familia. **Completo:** todas las secuencias de riboswitches recuperadas de RFAM 11.0; **Entrenamiento:** secuencias de entrenamiento; **Prueba:** secuencias de prueba.

RFAM_ACC	Descripción	Completo	Entrenamiento	Prueba
RF00050	FMN riboswitch (RFN element)	1211	726	485
RF00059	TPP riboswitch (THI element)	3616	2169	1447
RF00162	S-box leader (SAM-I)	1241	744	497
RF00167	Purine riboswitch	592	355	237
RF00168	Lysine riboswitch	562	337	225
RF00174	Cobalamin riboswitch	4286	2571	1715
RF00234	Glucosamine-6-phosphate activated ribozyme (glmS)	275	165	110
RF00380	ykoK leader	217	130	87
RF00504	Glycine riboswitch	2125	1275	850
RF00521	SAM riboswitch (alpha-proteobacteria) (SAM-II)	167	100	67
RF00522	PreQ1 riboswitch	116	69	47
RF00634	S-adenosyl methionine riboswitch (SAM-IV)	216	129	87
RF01051	Cyclic di-GMP-I riboswitch (c-di-GMP-I)	1020	612	408
RF01054	PreQ1-II (pre queuosine) riboswitch	39	23	16
RF01055	Molybdenum cofactor riboswitch (Moco RNA motif)	303	181	122
RF01057	S-adenosyl-L-homocysteine riboswitch (SAH)	198	118	80
TOTAL		16184	9704	6480

3.2.2. Ingeniería de características

Las características generadas se derivan de la estructura primaria de las secuencias recuperadas. A continuación se describe su obtención.

3.2.2.1. Composición K-mérica

Un k-mero se define como una cadena de longitud k. Sean p un k-mero y s una cadena, ambos sobre el alfabeto $\Sigma_{RNA} = \{\texttt{A},\texttt{U},\texttt{C},\texttt{G}\}$. La frecuencia de p en s se define como la razón del número de veces que p aparece como una subcadena de s y el número de k-meros en s, esto es,

$$freq(p, s) = \frac{count(p, s)}{|s| - k + 1}. \tag{15}$$

Ahora sea $K \subseteq \mathbb{Z}^+$. Se define la *composición K-mérica* de una secuencia s como las frecuencias de todos los posibles k-meros para $k \in K$.

Por ejemplo, si $K = \{1, 2\}$, los k-meros posibles serán todas las cadenas de longitud

1 y 2 sobre el alfabeto Σ_{RNA}, es decir, A, U, C, G, AA, AU, AC, AG, UA, UU, UC, UG, CA, CU, CC, CG, GA, GU, GC y GG, por lo que la composición K-mérica de la secuencia $s = accg$ está dada por {A=$\frac{1}{4}$, U=0, C=$\frac{1}{2}$, G=$\frac{1}{4}$, AA=0, AU=0, AC=$\frac{1}{3}$, AG=0, UA=0, UU=0, UC=0, UG=0, CA=0, CU=0, CC$\frac{1}{3}$, CG=$\frac{1}{3}$, GA=0, GU=0, GC=0, GG=0}. De manera general, para un alfabeto de cuatro letras y un conjunto $K = \{k \mid 1 \leq k \leq n\}$, la cardinalidad de la composición K-mérica está dada por

$$\sum_{k=1}^{n} 4^k = \frac{4}{3}\left(4^n - 1\right). \tag{16}$$

3.2.2.2. Extracción de características

En este estudio, consideramos la composición $\{k \mid 1 \leq k \leq n\}$-mérica como un conjunto de características para la clasificación. El conjunto de características usado por Singh y Singh (2017) corresponde a la composición K-mérica con $K = \{1, 2\}$. De aquí en adelante nos referiremos a este conjunto como DNC.

Intuitivamente, al incrementar el valor de n pudieran aumentarse el número de características discriminantes. Sin embargo, esto también hace que la cardinalidad de la composición K-mérica crezca exponencialmente (Ecuación 16). Esto resulta problemático debido a que tener demasiadas características pudiera inadvertidamente inducir modelos de clasificación sobreajustados (Domingos, 2012). Algunos autores sugieren que el número óptimo de características sea menor que el número de instancias de entrenamiento (Hua *et al.*, 2004) y, en nuestro caso, $n = 6$ es el mayor valor que satisface esta condición. Por lo tanto, en este trabajo proponemos el conjunto de características HEX, formado mediante la composición K-mérica con $K = \{1, 2, 3, 4, 5, 6\}$. Las 5460 características resultantes del conjunto HEX fueron calculadas mediante un programa escrito en Python 3.

3.2.2.3. Selección de características

Se redujo la dimensionalidad del conjunto HEX en las secuencias de entrenamiento por medio del algoritmo de selección de subconjuntos de características basado

en correlación (CfsSubsetEval, Correlation-based característica Subset Selection algorithm (Hall, 1999)) implementado en Weka 3.8 (Hall *et al.*, 2009b). A partir de ahora el conjunto resultante de características será referido como HEXCFS.

3.2.3. Configuración experimental

3.2.3.1. Clasificación

Se entrenaron seis clasificadores de cuatro tipos distintos usando HEXCFS: redes neuronales artificiales (Multilayer perceptron (MLP)), bayesianos (Naive Bayes (NB) (John y Langley, 1995)), árboles de decisión (Random Forest (RF) (Breiman, 2001) y J48 (Quinlan, 1993)), máquinas de soporte vectorial (Sequential Minimal Optimization algorithm for training a Support Vector Machine (SMO) (Platt, 1999)) e HyperPipes (HP) (Kalapanidas *et al.*, 2003b). Existen trabajos que han usado máquinas de soporte vectorial para la clasificación de moléculas de ARN no codificante (Panwar *et al.*, 2014; Liu *et al.*, 2006; Kong *et al.*, 2007). Los árboles de decisión, en particular RF, han probado ser aptos para una amplia gama de problemas (Fernández-Delgado *et al.*, 2014). NB y HP son algoritmos sencillos usados en problemas de clasificación de textos (Danger *et al.*, 2010).

En primer lugar, se realizó la sintonización de parámetros de los algoritmos. HP no requiere parámetros de entrada y NB se usó con los parámetros por omisión de Weka. Respecto a MLP, se utilizó la configuración reportada por Singh y Singh (2017). La sintonización de SMO, RF y J48 se realizó al evaluar la métrica F1 en validación cruzada de 5-pliegues con las siguientes configuraciones:

- SMO: búsqueda en rejilla sobre el parámetro de complejidad $C = [0.1, 1.0, 10.0]$ y el valor del exponente del kernel polinomial $E = [1, 2, 3]$.
- RF: número de árboles $I = [50, 100, 200]$.
- J48: valor de confianza $C = [0.05, 0.25, 0.75]$.

La lista completa de configuraciones de Weka resultantes se encuentra en la Tabla

2.

Tabla 2: Lista de configuraciones de Weka para generar los clasificadores usados y realizar la selección de características. N/A significa que el algoritmo no recibe parámetros de entrada.

Esquema	Opciones
NaiveBayes	N/A
SMO	-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007calibrator "weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
HyperPipes	N/A
J48	-C 0.25 -M 2
RandomForest	-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
MultilayerPerceptron	-L 0.3 -M 0.2 -N 500 -V 40 -S 0 -E 20 -H a
weka.attributeSelection.CfsSubsetEval -P 1 -E 1	weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1 -num-slots 1

El desempeño de la clasificación fue estimado usando validación cruzada estratificada de 10 pliegues. Posteriormente, se crearon los modelos de clasificación con todos los datos de entrenamiento y se probaron con los datos de prueba utilizando Weka Knowledgeflow. Los resultados obtenidos fueron comparados con los presentados por Singh y Singh (2017).

Puesto que el dominio del aptámero es conservado, un enfoque directo para resolver el problema de determinar a cuál familia pertenece un riboswitch putativo consiste en realizar una búsqueda con BLAST. Con el fin de comparar los resultados obtenidos por los clasificadores con las de este tipo de búsqueda, se utilizó BLAST+ versión 2 (Camacho *et al.*, 2009) para construir una base de datos con las secuencias de entrenamiento y después consultar las secuencias de prueba mediante la aplicación blastn con un umbral de *E*-value alto (*E*-value $\leq$ 1E+06). La familia a la que pertenece el mejor resultado (hit) se consideró como la predicción de BLAST.

Adicionalmente, se obtuvo el valor del mínimo *E*-value entre los hits incorrectos, y se recuperaron todas las secuencias del conjunto de prueba con un *E*-value mayor o igual a dicho valor. Las predicciones de los clasificadores HEXCFS en este subconjunto de las secuencias de prueba fueron comparadas con las predicciones de BLAST.

3.2.3.2. Métricas de evaluación

Para evaluar el desempeño de los clasificadores propuestos, se calcularon los promedios ponderados de la sensibilidad (*Sen*), especificidad (*Spe*), precisión (*ACC*) y el valor-F (*F*1). Sean *TP*, *FN*, *TN* y *FP* los números de verdaderos positivos, falsos negativos, verdaderos negativos y falsos positivos, respectivamente. Entonces,

$$Sen = \frac{TP}{TP+FN}, \tag{17}$$

$$Spe = \frac{TN}{TN+FP}, \tag{18}$$

$$ACC = \frac{TP+TN}{TP+FN+TN+FP}, \tag{19}$$

$$F1 = \frac{2TP}{2TP+FP+FN}. \tag{20}$$

Dado que este problema es de clasificación multiclase, se obtuvieron los valores TP_i, FN_i, TN_i y FP_i para cada clase C_i mediante la matriz de confusión (Sokolova y Lapalme, 2009). Sea l el número total de clases, $|C_i|$ la cardinalidad de la clase C_i (también conocido como el soporte de C_i) y $n = \sum_{i=1}^{l} |C_i|$ el número total de instancias. El promedio ponderado wM de la métrica M se define como

$$wM = \sum_{i=1}^{l} w_i M_i, \tag{21}$$

donde M_i es el valor de la métrica para la clase C_i y $w_i = \frac{|C_i|}{n}$.

3.2.4. Visualizando los *k*-meros seleccionados

En esta sección se introduce y propone el mapa de cobertura *k*-mérica (*k*-mer covMap), una herramienta de visualización para analizar la frecuencia en que aparecen ciertos motivos *k*-méricos en un conjunto de secuencias. En este caso en particular, la intención es analizar las características seleccionadas presentes en el conjunto HEXCFS.

Sean s y p cadenas y sea I una función tal que:

$$I(p,s) = \begin{cases} 1 & \text{si } p \text{ es una subcadena de } s, \\ 0 & \text{otro caso.} \end{cases} \tag{22}$$

Es posible obtener una formulación alternativa de $I(p,s)$ utilizando la frecuencia de p en s (Ec. 15):

$$I(p,s) = \begin{cases} 1 & \text{si } freq(p,s) > 0, \\ 0 & freq(p,s) = 0. \end{cases} \tag{23}$$

Sea S un conjunto de secuencias sobre el alfabeto Σ_{RNA} y C un conjunto de etiquetas (categorías). Cada secuencia $s \in S$ está asociada solo con una $c \in C$, por lo que denotamos S_c como la partición de secuencias de S etiquetadas con la categoría c.

Entonces, la cobertura del k-mero p en la categoría c se define como:

$$cov(p,c) = \frac{\sum_{s \in S_c} I(p,s)}{|S_c|}. \tag{24}$$

Los valores de $cov(p,c)$ están en el intervalo de 0 a 1, donde $cov(p,c) = 0$ indica que ninguna $s \in S_c$ contiene al k-mero p como subcadena, mientras que $cov(p,c) = 1$ significa que p está presente en todas las $s \in S_c$ (independientemente de cuántas veces esté en cada cadena y la posición que tenga). Ahora, sea P un conjunto de k-meros. El k-mer covMap es el mapa de calor que muestra los valores $cov(p,c)$ para todas las $p \in P$ y $c \in C$.

Adicionalmente, se presenta una versión agrupada del mapa mediante la organización de los elementos de P en un conjunto parcialmente ordenado P'. Sea H_i el grupo (clúster) de secuencias asociado a la clase C_i. Para cada $p \in P$, p se asigna al clúster H_f tal que

$$f = \text{mín}\left(\underset{j \in [1,|C|]}{\text{argmáx}} \{cov(p, C_j)\} \right). \tag{25}$$

Suponga que $p_i \in C_q$ y $p_j \in C_r$. Entonces, el orden en P' está dado por

$$p_i < p_j \text{ si } \begin{cases} q < r & \text{si } q \neq r, \\ cov(p_i, C_q) \leq cov(p_j, C_q) & \text{otro caso.} \end{cases} \tag{26}$$

En otras palabras, este algoritmo asigna los k-meros a la familia donde presenten más cobertura. Después, se ordenan según el orden que tengan las familias (en este trabajo es lexicográfico pero puede ser un orden arbitrario), y en cada familia se ordenan por cobertura de manera descendente. La ventaja de esta visualización es que permite ver qué k-meros son más representativos para una familia que para las demás por simple inspección. En caso de no distinguirse los grupos claramente, uno pudiera tomar la decisión de generar más características o utilizar otra categoría que permita una mejor discriminación.

3.2.4.1. Resaltado de estructuras consenso

Con el objetivo de integrar la información de los valores *cov* y los k-meros seleccionados, se recuperaron las secuencias y estructuras consenso de las familias consideradas en este trabajo localizadas en la base de datos RFAM. Estos consensos fueron generados utilizando alineamientos múltiples y se encuentran en formato Stockholm, por lo que fue necesario desarrollar un *parser* para interpretarlos. Se identificaron las posiciones del consenso estructural identificados como *gaps* y se eliminaron de ambos consensos. Después, utilizando VARNA (Darty *et al.*, 2009), se dibujaron de manera conjunta la secuencia y estructura consenso para cada familia considerada y, en cada una de ellas, se resaltaron las regiones correspondientes a los k-meros que pertenecen al clúster asociado a dicha familia (Figura 10).

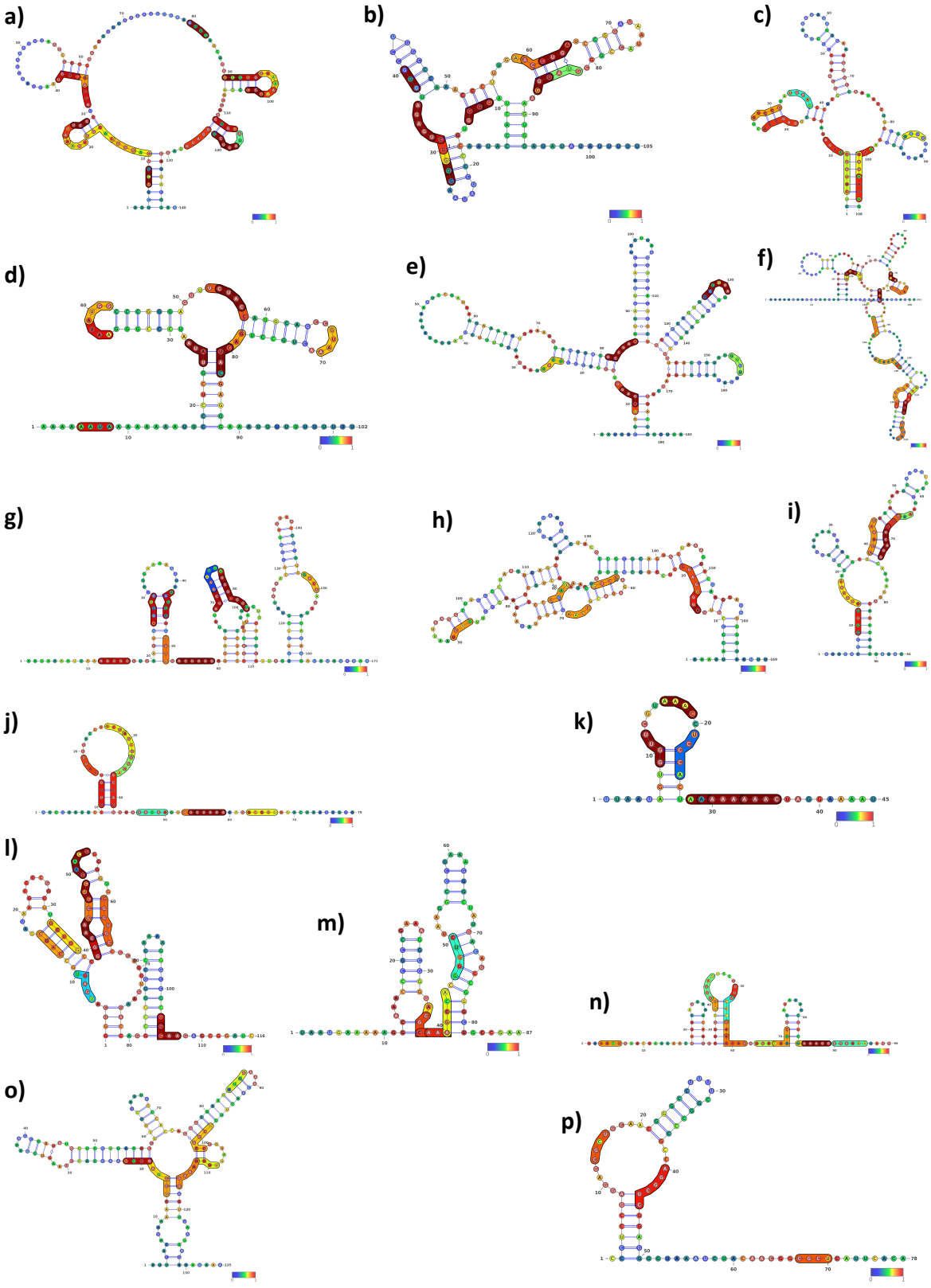

Figura 10: Estructuras consenso resaltadas. a) RF00050; b) RF00059; c) RF00162; d) RF00167; e) RF00168; f) RF00174; g) RF00234; h) RF00380; i) RF00504; j) RF00521; k) RF00522; l) RF00634; m) RF01051; n) RF01054; o) RF01055; p) RF01057.

3.3. Resultados y discusión

3.3.1. Datos desbalanceados

Las 16 familias de riboswitches consideradas en este estudio y el número de secuencias que las conforman se presentan en la Tabla 1. Como resultado de la división 60/40, los conjuntos de entrenamiento y prueba consistieron de 9704 y 6480 secuencias, respectivamente. Al observar el número de secuencias en cada familia podemos afirmar que el conjunto de datos es uno desbalanceado. En el conjunto completo de secuencias, por ejemplo, hay cuatro familias con menos de 200 secuencias (preQ1-II: 39, preQ1: 116, SAM-II: 167 y SAH: 198) y tres familias con más de 2000 secuencias (Glycine: 2125, TPP: 3616 y Cobalamin: 4268), por lo que las proporciones entre el número de secuencias por clase llegan a ser hasta de 109 a 1 (Cobalamin vs. preQ1-II).

El aprendizaje de datos desbalanceados se vuelve un problema cuando las clases que son minoría contienen tan pocas instancias que sus características esenciales pudieran resultar no representadas (Japkowicz y Stephen, 2002). En la práctica, esta situación se pudiera aminorar al rebalancear los datos de entrenamiento (He y Garcia, 2009). Sin embargo, puesto que se busca la comparación utilizando el mismo número de secuencias que el enfoque previo, este trabajo se centra en evaluar el desempeño de los clasificadores con el conjunto de secuencias original (desbalanceado).

3.3.2. Ingeniería de características

En este estudio se proponen dos conjuntos de características llamados HEX y HEXCFS. El primero consiste en la composición K-mérica con $K = \{1, 2, 3, 4, 5, 6\}$ y el segundo es el resultado de aplicar el algoritmo CfsSubsetEval al conjunto HEX. La Tabla 3 resume el número total de características por conjunto y por longitud de k-mero. Puede observarse que el número de características en el conjunto HEXCFS (156) representa tan solo el 2.85 % de las presentes en el conjunto HEX (5460). Tal reducción pudiera explicarse mediante la redundancia inherente en las características de la composición K-mérica. Puesto que un k-mero p dado tiene $\sum_{i=1}^{k}(k-i+1) = k(k+1)/2$ subcadenas, esto significa que potencialmente $O(n^2)$ características pudieran estar altamente co-

rrelacionadas con p para una clase dada. En vista de que el algoritmo CfsSubsetEval prefiere características que están altamente correlacionadas con la etiqueta pero que al mismo tiempo tengan baja intercorrelación entre ellas (Hall, 1999), se puede hacer la conjetura de que las características redundantes no se van a seleccionar.

Tabla 3: Contenido de los conjuntos de características.

Características	DNC	HEX	HEXCFS
1-meros	4	4	0
2-meros	16	16	10
3-meros	0	64	5
4-meros	0	256	46
5-meros	0	1024	56
6-meros	0	4096	39
TOTAL	20	5460	156

3.3.2.1. Motivos discriminantes

Como muestra la Tabla 3, si bien 90 % de las características HEXCFS son k-meros de longitudes 4, 5 y 6, su conteo en estas categorías no es proporcional a 4^k — el número esperado de características en cada categoría del conjunto HEX.

Aunque monómeros, dímeros y trímeros se utilizan comúnmente para clasificar secuencias de nucleótidos (Singh y Singh, 2017; Tripathi *et al.*, 2016; Panwar *et al.*, 2014), en este estudio se obtuvieron k-meros más largos para la clasificación. Para profundizar en esto, suponga que s sea una secuencia, p un k-mero y r una subcadena de p. De acuerdo con la definición de frecuencia dada en la Ecuación 15, intuitivamente, se cumple que $freq(p, s) \leq freq(r, s)$. Por lo tanto, mientras la longitud de los k-meros se incrementa, se cumple que $cov(p, c_i) \leq cov(r, c_i)$ para una determinada familia c_i. Por otro lado, si $cov(p, c_i)$ fuera mayor que cualquier otra $cov(p, c_j)$, entonces puede afirmarse que p es más específico para la familia c_i que para cualquier otra c_j, i.e., p es un motivo discriminante para c_i (Redhead y Bailey, 2007).

3.3.2.2. *k*-mer covMap de las características HEXCFS

Para explorar los *k*-meros seleccionados por el algoritmo CfsSubsetEval, se generó el *k*-mer covMap (Sección 3.2.4) con los siguientes parámetros de entrada:

- *S*: las secuencias de entrenamiento,
- *C*: las etiquetas de las 16 familias de riboswitches usadas en el estudio y
- *P*: los *k*-meros seleccionados por CfsSubsetEval (las características HEXCFS).

La Fig. 11 muestra el covMap resultante. Las filas están etiquetadas con los *k*-meros en orden alfabético y las columnas con las familias. Puede observarse que las primeras 10 filas corresponden a los valores *cov* para los dímeros seleccionados. Dada su baja especificidad, el promedio de valores *cov* en las celdas correspondientes a estas filas es de 0.99, y un total de 98.75 % de estas celdas tienen valores mayores a $cov = 0.8$. Los dímeros, trímeros e incluso tetrámeros son frecuentes en todas las familias. Sin embargo, se puede observar que mientras *k* se incrementa, el número de celdas con un valor alto de *cov* decrece. Para apoyar esta observación, se calcularon los pares de *cov* promedio y el porcentaje de celdas arriba del umbral de 0.8 para $3 \leq k \leq 6$, lo cuales resultaron ser 0.75 y 55 %, 0.34 y 5.3 %, 0.14 y 2.8 %, y 0.08 y 3.2 %, respectivamente.

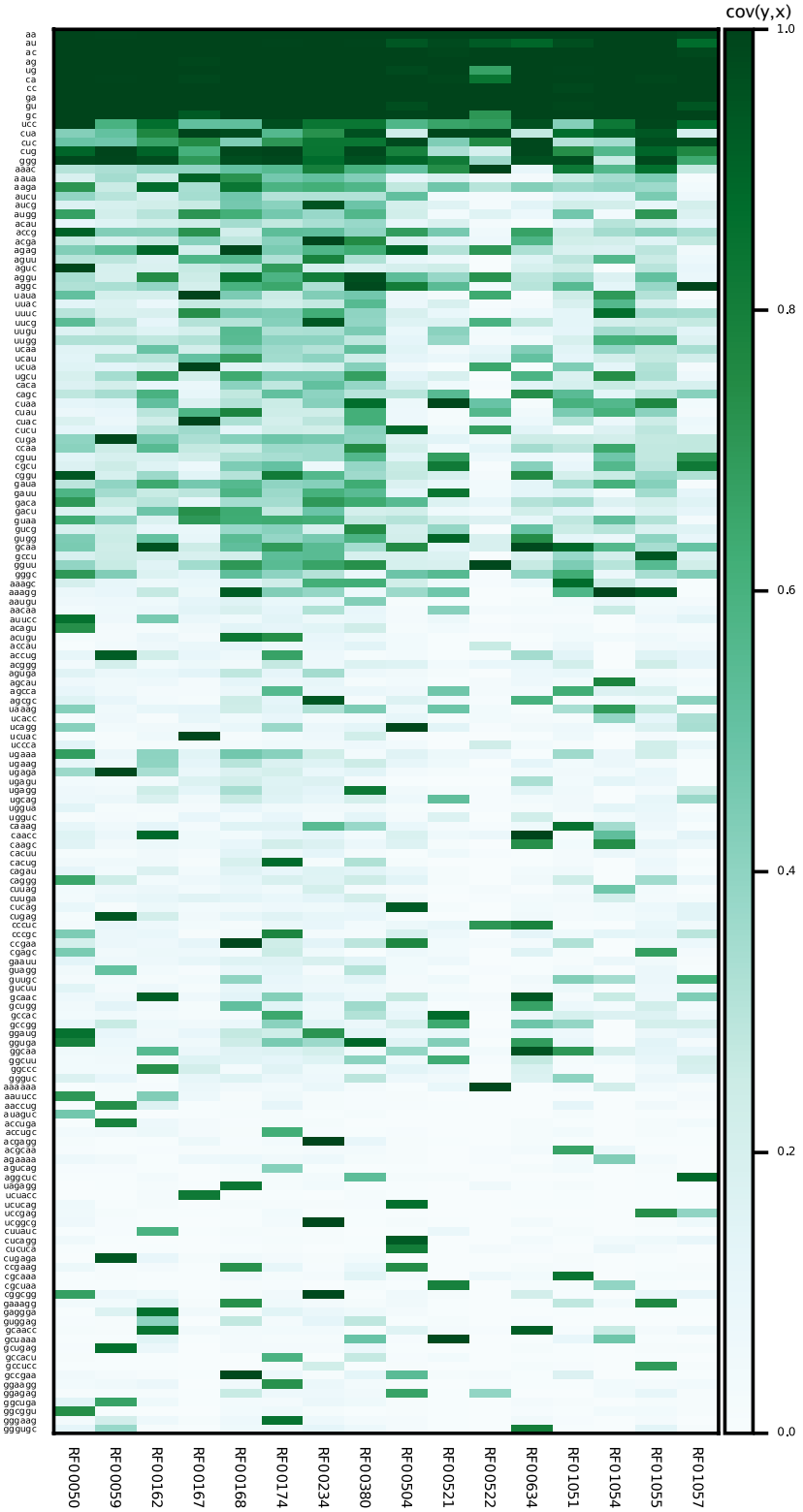

Figura 11: *k*-mer covMap generado utilizando como parámetros de entrada las 16 familias de riboswitches consideradas en el estudio, los *k*-meros del conjunto HEXCFS y las secuencias de entrenamiento. Las filas (y) corresponden a los *k*-meros seleccionados, las columnas (x) a las familias de riboswitches, y los valores representados en la celdas a $cov(y, x)$.

Adicionalmente, se puede notar que hay pentámeros y hexámeros que son frecuentes simultáneamente en varios grupos o en varias familias. Por ejemplo, el pentámero `AAAGG` aparece en las familias RF00168 (0.920), RF01051 (0.577), RF01054 (1.000) y RF01055 (0.939); `CGGCGG` en RF00050 (0.672) y RF00234 (0.982); `GCCGAA` en RF00168 (0.982) y RF00504 (0.543); y `GCAACC` en RF00162 (SAM-I, 0.844) y RF00634 (SAM-IV, 0.922). Además, la diferencia entre valores *cov* dentro de una misma fila se vuelve más identificable, lo que significa que un *k*-mero puede estar mucho más presente en las secuencias de una familia que en las otras familias. Por ejemplo, el hexámero `GGCGGU` en RF00050 tiene una cobertura de 0.740, mientras que la máxima cobertura alcanzada en las otras familias por este mismo *k*-mero es de 0.061 (RF00174). Esta observación confirma que, independientemente del valor de la cobertura, algunos *k*-meros seleccionados tienden a ocurrir más en secuencias de una familia en particular, por lo que podrían ser considerados como motivos discriminantes multiclase (Redhead y Bailey, 2007). La versión agrupada del covMap se generó para identificar más fácilmente estos *k*-meros (Fig. 12).

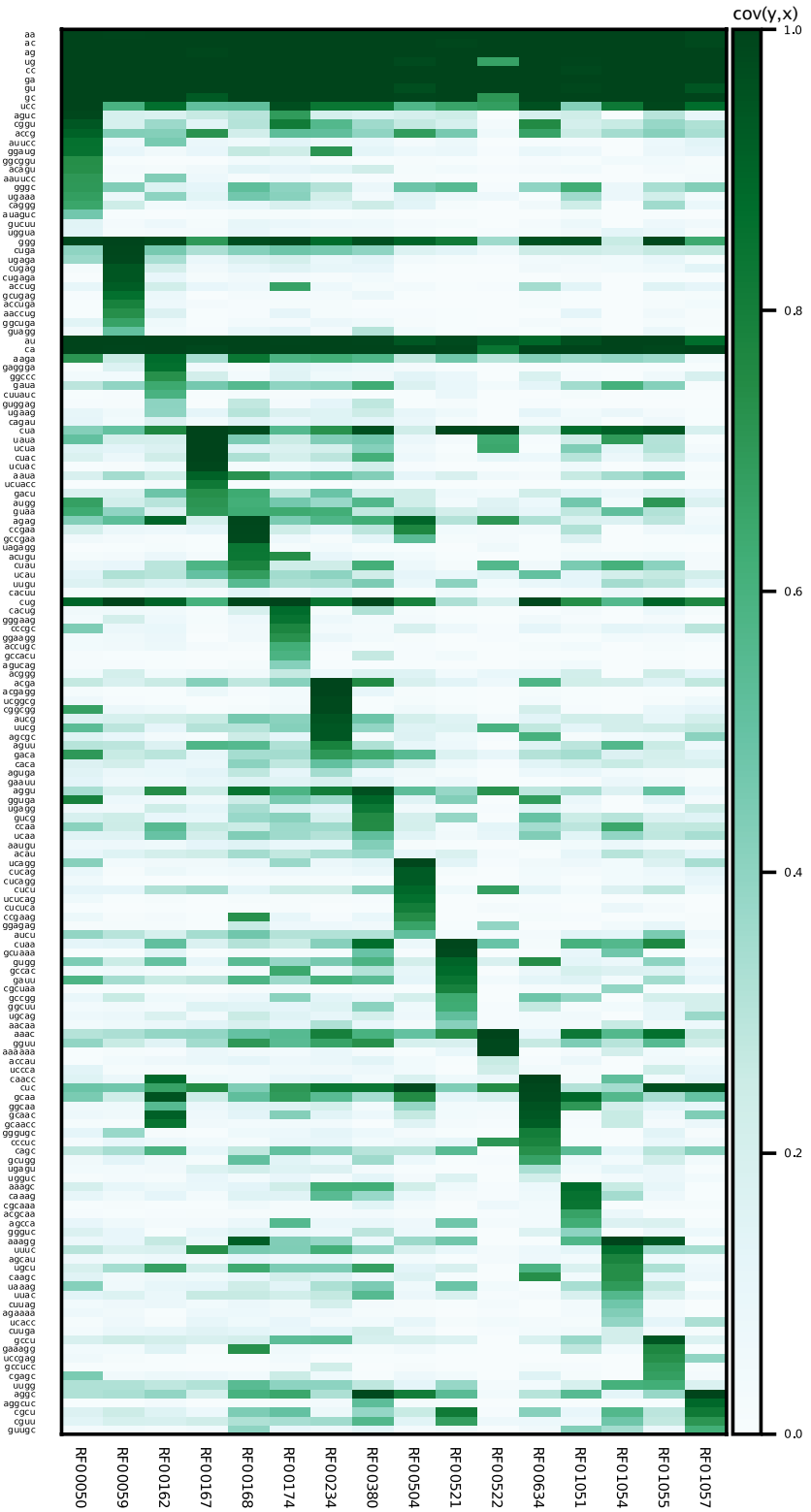

Figura 12: *k*-mer covMap agrupado generado utilizando como parámetros de entrada las 16 familias de riboswitches consideradas en el estudio, los *k*-meros del conjunto HEXCFS y las secuencias de entrenamiento. Las filas (y) corresponden a los *k*-meros seleccionados, las columnas (x) a las familias de riboswitches, y los valores representados en la celdas a $cov(y, x)$.

Observe que, para algunas familias, casi todos los *k*-meros asignados a su correspondiente clúster son discriminantes (RF00059, RF00174 y RF00504). Por otro lado, los *k*-meros pertenecientes al clúster de RF00162 (SAM I) ocurren casi con la misma frecuencia en la familia RF00634 (SAM IV). Esto pudiera ser explicado por el hecho de que SAM-I y SAM-IV tienen en común los sitios de unión (aunque forman andamiajes diferentes) (Weinberg *et al.*, 2008). En cambio, los aptámeros de SAM-II (RF00521) son más pequeños y tienen una estructura secundaria más simple comparados con los aptámeros de SAM-I (Corbino *et al.*, 2005). Como consecuencia, los *k*-meros pertenecientes al clúster de SAM-II son diferentes a los de SAM-I (Fig. 12).

Al considerar los *k*-meros agrupados en los *clusters* de la Fig. 12, una observación interesante es que algunos de estos *k*-meros pueden fusionarse en motivos de mayor longitud. Por ejemplo, considere el caso de la familia RF00050 y los *k*-meros `AUUCC`, `UGAAA`, y `AAUUCC`. Observe que, en efecto, `AUUCC` es una subcadena de `AAUUCC`, y que `UGAAA` y `AAUUCC` pueden concatenarse para formar el motivo `UGAAAUUCC`. Para profundizar en esto, se generaron y resaltaron las estructuras consenso de las familias consideradas. En particular, la Figura 13 muestra el consenso resaltado del riboswitch FMN(RF00050). Se puede verificar que la cadena `UGAAAUUCC` está presente en el alineamiento múltiple (bases 20-28). Asimismo, pueden apreciarse otros 4 motivos compuestos de *k*-meros consecutivos o superpuestos: `CAGGGCAGGGUGAAAUUCC` (10-28), `ACCGGCGGU` (31,39), `UCCGGUGAAAUUCC` (90-103) y `CGGUAUAGUCCGGAUG` (102-127).

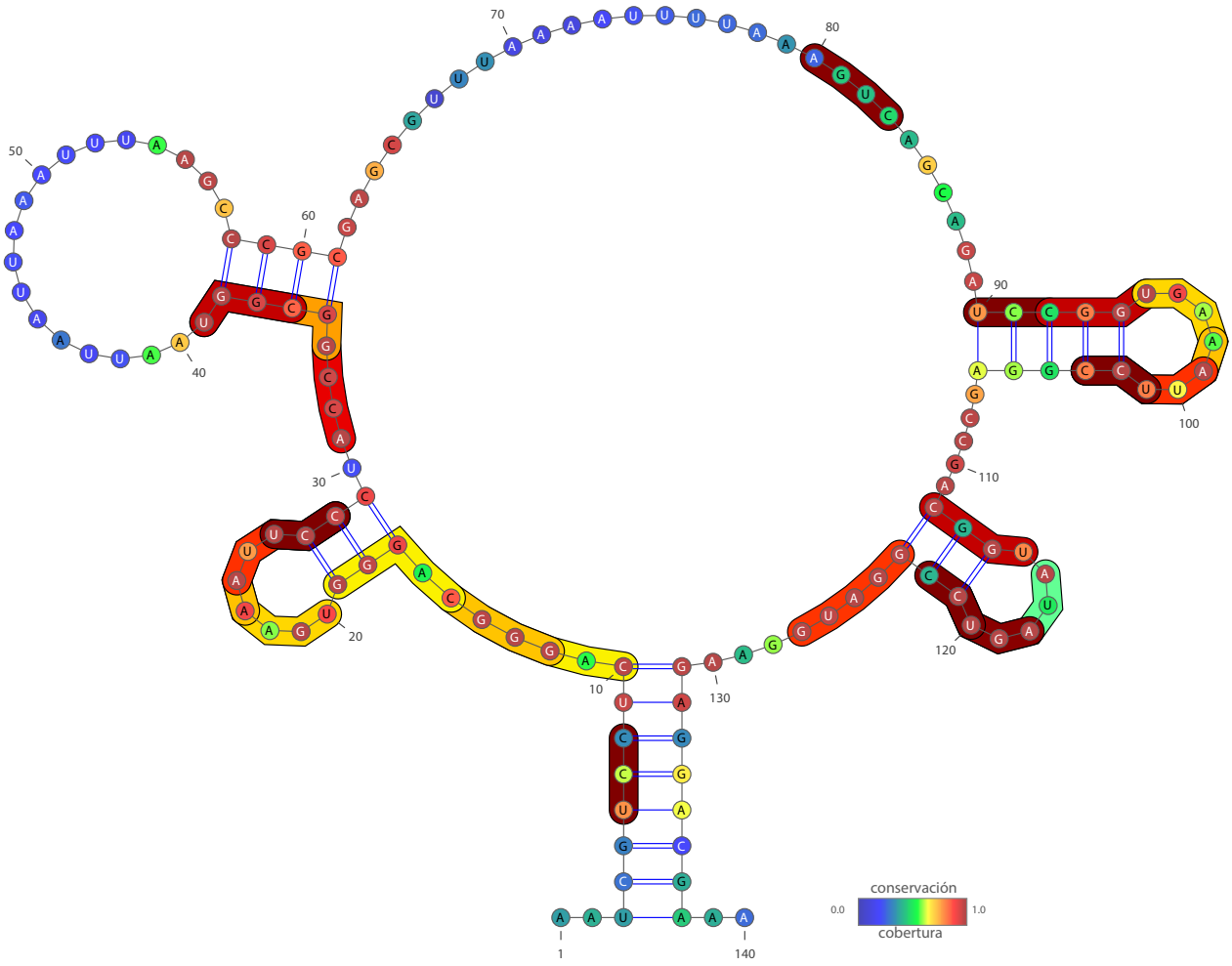

Figura 13: Secuencia y estructura consenso de la familia RF00050. En cada base, el color representa su conservación en el alineamiento (a nivel de secuencia). En cada región resaltada, el color representa la cobertura de la región en la familia.

Motivado por lo anterior, los *k*-meros contenidos en el *cluster* de cada una de las familias que también resultaron estar presentes en los alineamientos de RFAM se buscaron en las estructuras de riboswitches reportadas en la literatura. A su vez, también se consideraron las superposiciones que pudieran formar entre ellos (Tabla 4).

Tabla 4: Ejemplos de patrones formados con *k*-meros presentes en el conjunto HEXCFS que forman parte de motivos reportados en riboswitches. Entre paréntesis se indica la posición del motivo en el alineamiento múltiple de RFAM (Figura 10).

Familia	Motivo	Descripción
RF00059	UGAGA (32-36)	Región de la hélice sensor de pirimidina en el riboswitch Thi-Box (Edwards y Ferré-D'Amaré, 2006).
RF00162	GAGGGA (19-24)	Región adyacente a un motivo *kink-turn* que permite la interacción con un pseudonudo. La formación estable de este dominio durante su interacción con SAM causa que la plataforma de expresión río abajo genere un terminador de la transcripción independiente de rho, lo que detiene la expresión génica (Montange y Batey, 2006).
RF00167	CUAC (54-57) y (80-83)	Ambas locaciones están dentro de la misma unión de 3 *stems*, la cuál es crítica para la unión con el metabolito (Mandal *et al.*, 2003).
RF00168	UAGAGG (9-14)	Parte del elemento Lys (unión de 4 tallos), la cuál se encarga de la regulación de la lisina (Sudarsan *et al.*, 2003).
RF00234	AGCGC (12-16)	Región ubicada en el núcleo de la ribozima. El sitio de corte principal se encuentra entre los nucleótidos A (12) y G (13). (Winkler *et al.*, 2004).
RF00234	CGGCGG (79-84)	Región ubicada en el núcleo de la ribozima (Winkler *et al.*, 2004).
RF00234	ACGAGG (54-59)	Región no pareada del núcleo de la ribozima (Winkler *et al.*, 2004).
RF00380	AGGU (14-17) y UGAGG (17-21)	*k*-meros superpuestos que forman parte de una region bulto-tallo conservada, con enlaces internucleótido en la estructura terciaria (Dann III *et al.*, 2007).
RF00504	CUCAGG (68-73)	Segmento del *stem* de un *internal loop* presente en ambos aptámeros del riboswitch de la glicina (Kwon y Strobel, 2008).
RF00522	AAAAAA (28-33) y AAAC (33-36)	*k*-meros superpuestos que son parte de la cola 3' rica en adenina del dominio del aptámero, la cuál forma un pseudonudo único compacto que encapsula a PreQ1 (Kang *et al.*, 2010).
RF00634	GGCAA (39-43), CAACC (41-45), y CCCUC (44-48)	*k*-meros superpuestos que están en una hélice conservada de SAM-IV (Weinberg *et al.*, 2008).
RF01051	CAAAG (37-41)	Región que forma un enlace entra las hélices P1 y P2, necesarias para la formación del bolsillo de unión de c-di-GMP (Smith *et al.*, 2009).
RF01054	AAAGG (85-89)	Porción de la hélice P3 del riboswitch PreQ1-II. Los últimos tres nucleótidos (AGG) forman parte de una secuencia Shine-Dalgarno conservada (Meyer *et al.*, 2008).
RF01055	GAAAGG (112-117)	Región de una unión *multi-stem* central. Se ha predicho que los últimos tres nucleótidos (AGG) son sitios de unión al ribosoma en el ORF adyacente en algunos representantes de Moco RNA (Regulski *et al.*, 2008).
RF01057	AGGCUC (40-45)	Segmento ubicado en la unión entre los *stems* P1 y P2 del riboswitch SAH. Los últimos dos nucleótidos (UC) forman parte de la hélice P2 (Wang *et al.*, 2008).

Se encontraron algunas instancias de motivos discriminantes para las familias PreQ1-I (RF00522) y PreQ1-II (RF01054). Aunque ambas clases son subcategorías de la superfamilia PreQ1, PreQ1-I se ha relacionado con regulación transcripcional o traduccional, mientras que PreQ1-II solo participa en la regulación traduccional (Eichhorn *et al.*, 2014). Consecuentemente, todas las secuencias de PreQ1-II identificadas a la fecha contienen una secuencia Shine-Dalgarno en el extremo 3' (la oclusión de dicha

secuencia origina la inhibición de la iniciación de la traducción (Rinaldi *et al.*, 2016)), y efectivamente este no es el caso para todas las secuencias de PreQ1-I. Además, PreQ1-I contiene una cola rica en adenina (poli `A`) que no está presente en PreQ1-II. Ambas características (la secuencia Shine-Dalgarno y la cola de poli `A`) se presentaron en los *k*-meros de HEXCFS. Otro ejemplo es la secuencia `GAAAGG` para el motivo Moco (RF01055), la cuál incluye un sitio de pegado de ribosoma (RBS por sus siglas en inglés). Esto es notable debido a que la función de este riboswitch es reprimir la expresión génica. Asimismo, HEXCFS contiene algunos patrones *k*-méricos localizados en los bolsillos de pegado de metabolitos (por ejemplo, `CUAC` en la familia RF00167 y `UAGAGG` en la familia RF00168). Finalmente, el motivo seleccionado para el riboswitch glmS (RF00234) es una subcadena del núcleo catalítico conservado que se une a GlcN6P y utiliza este metabolito como cofactor para promover el auto-corte por ribozima, estando el principal sitio de corte entre los nucleótidos 12 y 13 (McCown *et al.*, 2012).

3.3.3. Validación cruzada

La Tabla 5 muestra los resultados de la validación cruzada de 10 pliegues en los clasificadores inducidos con las características HEXCFS. La Figura 14 muestra las matrices de confusión resultantes de la prueba.

Tabla 5: Resultados de la validación cruzada de 10 pliegues. Los resultados se encuentran ordenados por $F1$ en orden descendente. La celdas en negrita representan el mejor valor por columna.

Clasificador	Sen	Spe	Acc	F1
SMO-HEXCFS	**0.996**	**0.999**	**0.999**	**0.996**
MLP-HEXCFS	0.994	**0.999**	**0.999**	0.994
RF-HEXCFS	0.988	0.997	0.997	0.988
J48-HEXCFS	0.934	0.990	0.984	0.934
NB-HEXCFS	0.914	0.991	0.981	0.916
HP-HEXCFS	0.463	0.878	0.821	0.386

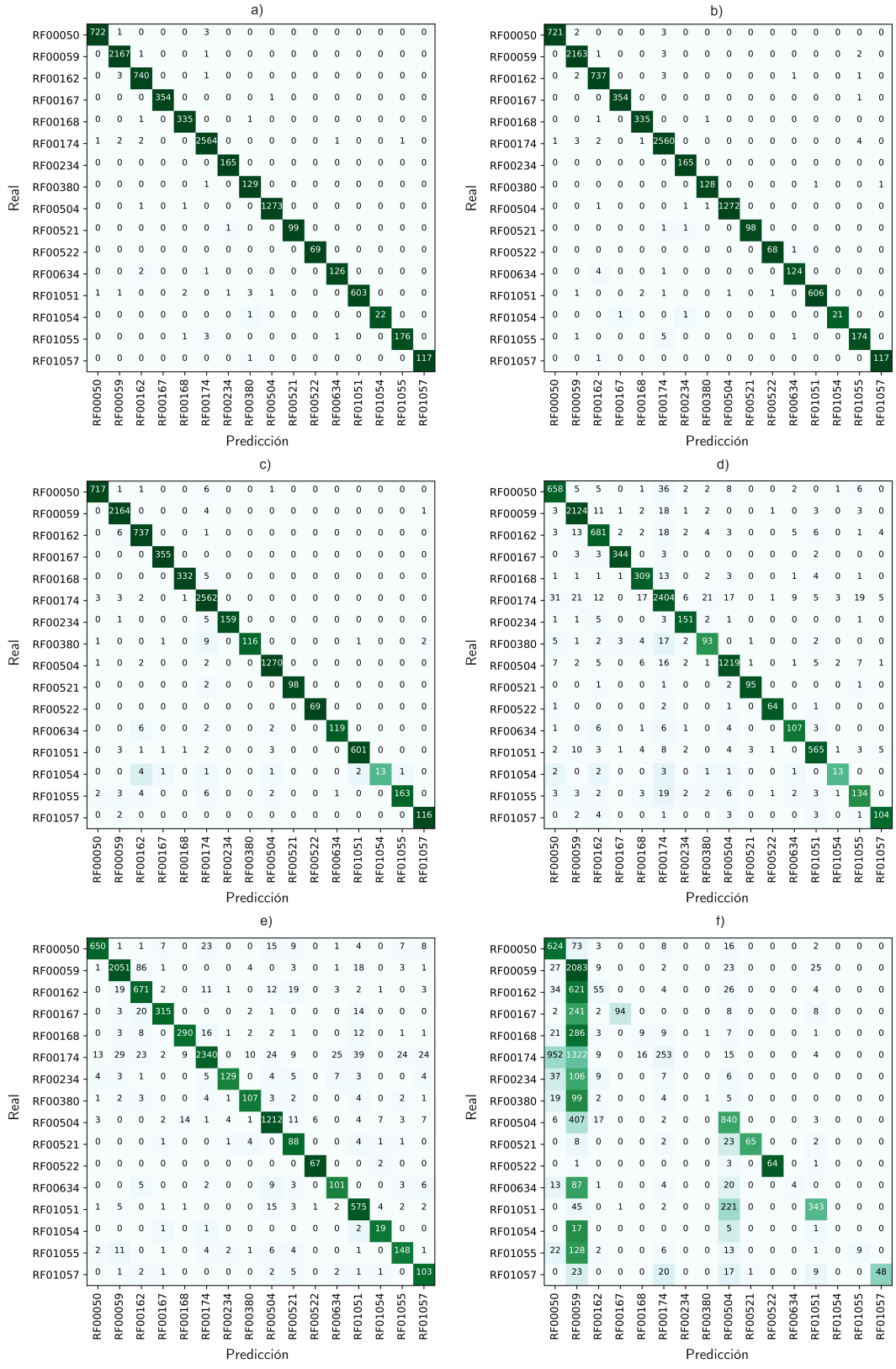

Figura 14: Matrices de confusión para los experimentos de validación cruzada. a) SMO-HEXCFS; b) MLP-HEXCFS; c) RF-HEXCFS; d) J48-HEXCFS; e) NB-HEXCFS; f) HP-HEXCFS. Los clasificadores están ordenados por valor-F.

Es interesante resaltar que, con la excepción de HP, los clasificadores alcanzaron valores arriba de 0.91 para todas las métricas evaluadas. En general, todos los clasificadores obtuvieron valores más altos de especificidad y precisión que de sensibilidad y valor-F. Por ejemplo, la especificidad estuvo en el intervalo de 0.990 a 0.999, la precisión de 0.981 a 0.999, la sensibilidad de 0.914 a 0.996 y el valor-F de 0.916 a 0.996. Particularmente, el mejor clasificador en todas las medidas de desempeño fue SMO-HEXCFS (sensibilidad= 0.996, especificidad= 0.999, precisión= 0.999 y valor-F= 0.996), seguido de cerca por MLP-HEXCFS (sensibilidad= 0.994, especificidad= 0.999, precisión= 0.999 y valor-F= 0.994). Respecto al clasificador HP-HEXCFS, consistentemente obtuvo los peores resultados en todas las métricas (sensibilidad= 0.463, especificidad= 0.878, precisión= 0.821 y valor-F= 0.386).

Desempeño de MLP y SMO

Los mejores clasificadores en esta etapa fueron SMO-HEXCFS y MLP-HEXCFS, ambos presentando un desempeño similar. Sin embargo, considerando el valor-F, SMO-HEXCFS obtuvo valores ligeramente mayores a los de de MLP-HEXCFS en todas las clases con excepción de RF00168, RF00380 y RF01051. Puesto que ambos clasificadores mostraron desempeño similar en todas las métricas excepto sensibilidad y valor-F, se discutirá su conteo de verdaderos positivos (TP). SMO-HEXCFS fue capaz de predecir cada instancia de las familias RF00234 y RF00522, incluso cuando esta última es la segunda clase más pequeña del conjunto (69 instancias de entrenamiento). Además, SMO-HEXCFS alcanzó un TP mayor en 11 familias: RF00050, RF00059, RF00162, RF00174, RF00380, RF00504, RF00521, RF00634, RF01054 y RF01055. Ambos clasificadores obtuvieron el mismo TP en cuatro familias: RF00167, RF00168, RF00234 y RF01057. MLP-HEXCFS fue mejor que SMO-HEXCFS solo en el riboswitch di-GMP-I (RF01051), el cuál fue la familia más difícil de clasificar para SMO-HEXCFS (0 y 9 falsos negativos, respectivamente). Esto es destacable porque RF01051 ni siquiera es una clase minoritaria; sin embargo, SMO-HEXCFS fue capaz de distinguir a las seis familias con menos secuencias, todas de ellas con menos de 130 instancias (RF00380, RF00521, RF00522, RF00634, RF01054 y RF01057).

El efecto del desbalance en las familias en el desempeño de la clasificación

Las clases minoritarias tuvieron un gran impacto en el desempeño de los algoritmos diferentes a SMO y MLP. Por ejemplo, J48-HEXCFS y NaiveBayes-HEXCFS no llegaron a valores de 0.8 en las clases con menos de 200 secuencias de entrenamiento, particularmente para las métricas sensibilidad, F1 y precisión. Al contrario, en clases con más de 1000 secuencias, el intervalo de valores en las métricas fue de 0.912 a 0.991. En cuanto a HP-HEXCFS, este clasificador obtuvo mal desempeño en todas las clases independientemente del número de instancias de entrenamiento.

Tanto RF-HEXCFS, J48-HEXCFS, NB-HEXCFS como HP-HEXCFS no fueron capaces de superar a los dos mejores clasificadores. Las Figuras 14c-f) muestran cómo estos clasificadores presentaron un sesgo hacia la familia mayoritaria (RF00174), con un total de 164, 70, 67 y 45 falsos positivos para J48-HEXCFS, HP-HEXCFS, NB-HEXCFS y RF-HEXCFS, respectivamente. La Figura 14f) muestra que HP-HEXCFS prefirió asignar las etiquetas de las secuencias en alguna de las clases con más instancias (RF00050, RF00059 y RF00504), lo que derivó en un pobre desempeño (valor-F= 0.386). A pesar de esto, alcanzó un valor-F de 0.962 en la familia RF00522, la cuál contó únicamente con 69 instancias para el entrenamiento.

El efecto de la selección de características en el desempeño de la clasificación

Con la finalidad de evaluar el impacto de reducir el conjunto de características usando un algoritmo de selección, comparamos los resultados presentados arriba con la validación cruzada de 10 pliegues utilizando el conjunto completo de características HEX 6.

Tabla 6: Comparación del desempeño en validación cruzada de 10 pliegues para los clasificadores entrenados con los conjuntos HEX y HEXCFS.

	SEN		SPE		ACC		F1	
Clasificador	HEXCFS	HEX	HEXCFS	HEX	HEXCFS	HEX	HEXCFS	HEX
SMO	0.996	0.999	0.999	1.000	0.999	1.000	0.996	0.999
MLP1	0.994	0.278	0.999	0.814	0.999	0.758	0.994	0.207
RF	0.988	0.983	0.997	0.995	0.997	0.995	0.988	0.982
J48	0.934	0.926	0.990	0.988	0.984	0.982	0.934	0.926
NB	0.914	0.867	0.991	0.983	0.981	0.963	0.916	0.874
HP	0.463	0.506	0.878	0.876	0.821	0.830	0.386	0.451

En general, todas las métricas fueron ligeramente mejores para los clasificadores entrenados con HEXCFS respecto a aquellos entrenados con HEX. Por ejemplo, la mejora en F1 fue de 4.81 %, 0.86 % y 0.61 % para NB, J48 y RF, respectivamente. En cuanto a los dos clasificadores SMO, la diferencia en F1 entre SMO-HEX y SMO-HEXCFS fue despreciable (0.003), aunque éste último con un *speedup* de ∼40x.

El efecto de la elección de n en la K-composición

En la sección anterior se mostró que la selección de características es una parte importante del preprocesamiento en la resolución del problema que estamos resolviendo. Respecto a la extracción de características, el valor del parámetro n se estableció en la fase de diseño del estudio. Para validar la elección de $n = 6$, se generó la composición $\{k \mid 1 \leq k \leq n\}$-mérica para $n = [1, .., 8]$, reduciéndose posteriormente los conjuntos resultantes con el algoritmo CfsSubsetEval. Estos conjuntos fueron usados para inducir clasificadores SMO que se evaluaron con validación cruzada de 10-pliegues (Tabla 7).

Tabla 7: Comparación del desempeño en validación cruzada de 10-pliegues para clasificadores SMO entrenados con las composiciones K-méricas reducidas ($n = [1, .., 8]$).

n	#Características	Sen	Spe	Acc	F1
1	4	0.388	0.821	0.790	0.322
2	14	0.648	0.910	0.888	0.621
3	33	0.876	0.973	0.965	0.874
4	74	0.965	0.993	0.990	0.965
5	147	0.993	0.999	0.998	0.993
6	156	0.996	0.999	0.999	0.996
7	185	0.995	0.999	0.999	0.995
8	169	0.995	0.999	0.999	0.995

Puede observarse una mejora en todas las métricas hasta $n = 6$, y después de ese punto, los valores se estabilizan. La especificidad máxima (0.999) se alcanzó en $n = 5$, mientras que los valores para sensibilidad, precisión y F1 (0.996, 0.999 y 0.996, respectivamente) corresponden a HEXCFS ($n = 6$). Estos resultados confirman que el valor seleccionado de n fue el apropiado para este conjunto de datos en particular.

De manera similar al análisis de las características de HEX y HEXCFS (Tabla 3), se calculó la distribución de características seleccionadas por categoría en los conjuntos reducidos (Tabla 8). El porcentaje de características seleccionadas respecto a los conjuntos originales para cada valor de n fueron de 100 %, 70.00 %, 39.28 %, 21.76 %, 10.77 %, 2.85 %, 0.84 % y 0.19 %, respectivamente. La categoría con más elementos de todos los conjuntos fue los 5-meros en $n = 5$ (89 de 147 características seleccionadas en ese valor de n). Además, se puede observar que, para $n \leq 5$, los n-meros fueron las categorías más seleccionadas. Sin embargo, para $n > 5$, los 4-meros y 5-meros continuaron siendo las mayores categorías. Puede notarse que el número de características seleccionadas por categoría se estabilizó cuando $n > k$; en estos casos, el promedio y desviación estándar fue de 0.6 ± 0.5, 10 ± 0.6, 5.4 ± 0.8, 46 ± 2.4, 57.7 ± 2.4, 24.5 ± 2.5 y 19 ± 0 para k =1, 2, 3, 4, 5, 6 y 7, respectivamente. Más aún, para $k \geq 5$, el número de k-meros seleccionados fue mayor que para los $(k+1)$-meros; es decir, cada conjunto reducido contenía menos 6-meros que 5-meros, menos 7-meros que 6-meros y menos 8-meros que 7-meros. Esta tendencia pareciera estar relacionada a la cobertura de los motivos discriminantes discutida en la Sección 3.3.2.1.

Tabla 8: Resultados de la selección de características en las K-composiciones para $1 \leq n \leq 8$.

Categoría	*n* 1	2	3	4	5	6	7	8
1-mero	4	0	0	1	1	0	1	1
2-mero	14	11	10	9	10	10	10	
3-mero	22	4	6	5	6	6		
4-mero	59	42	46	48	48			
5-mero	89	56	61	56				
6-mero	39	27	22					
7-mero	32	19						
8-mero	7							
Total	4	14	33	74	147	156	185	169
%Conjunto completo	100.00	70.00	39.28	21.76	10.77	2.85	0.84	0.19

3.3.4. Desempeño en datos de prueba

Los resultados de la clasificación utilizando los datos de prueba (40 % del conjunto original) se muestran en la Tabla 9. Las matrices de confusión para todos los clasificadores (exceptuando HP) se presentan en la Fig. 15.

Tabla 9: Resultados de la prueba con datos independientes. Los resultados se encuentran ordenados por $F1$ en orden descendente. La celdas en negrita representan el mejor valor por columna.

Clasificador	Sen	Spe	Acc	F1
BLAST (Camacho *et al.*, 2009)	**0.995**	**0.999**	**0.999**	**0.995**
SMO-HEXCFS	0.966	0.993	0.991	0.966
MLP-HEXCFS	0.963	0.993	0.991	0.963
RF-HEXCFS	0.947	0.985	0.984	0.946
J48-HEXCFS	0.881	0.976	0.969	0.879
NB-HEXCFS	0.858	0.987	0.970	0.863
HP-HEXCFS	0.422	0.872	0.811	0.337
MLP-DNC (Singh y Singh, 2017)	0.914	0.261	0.810	0.331

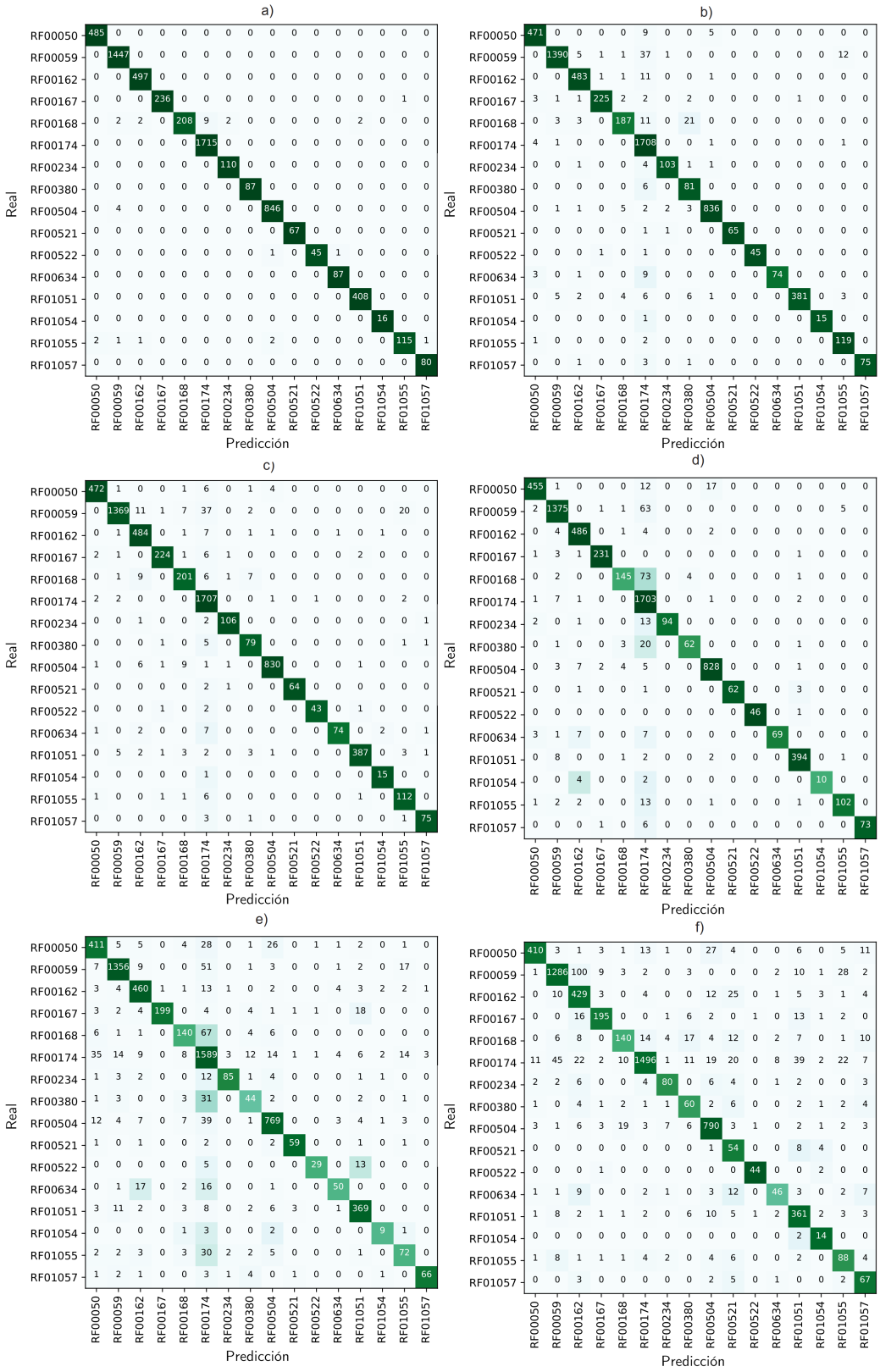

Figura 15: Matrices de confusión para los experimentos con datos independientes. a) BLAST; b) SMO-HEXCFS; c) MLP-HEXCFS; d) RF-HEXCFS; e) J48-HEXCFS; f) NB-HEXCFS. Los clasificadores están ordenados por valor-F.

Resultados al entrenar con HEXCFS

En general, SMO fue el mejor clasificador en todas las métricas evaluadas, alcanzando valores de sensibilidad, especificidad, precisión y valor-F de 0.966, 0.993, 0.991 y 0.966, respectivamente. Sin embargo, MLP obtuvo los mismos valores de especificidad y precisión que SMO, con valores ligeramente menores en sensibilidad y valor-F (0.963 y 0.963, respectivamente). Todos los clasificadores tuvieron mayor especificidad que sensibilidad. Exceptuando HP, la sensibilidad estuvo en el intervalo de 0.858 a 0.966, especificidad de 0.976 a 0.993, precisión de 0.969 a 0.991 y el valor-F de 0.863 a 0.966.

MLP y SMO fueron los mejores clasificadores en esta prueba. Ambos presentaron un desempeño similar en valor-F en cada familia. No obstante, obtuvieron un desempeño distinto para tres familias en particular: ykoK leader (RF00380), PreQ1-II (RF01054) y Moco RNA motif (RF01055). Con respecto a la familia RF00380, SMO y MLP alcanzaron un valor-F de 0.801 y 0.872, respectivamente. El número de falsos positivos (FP) en esa familia fue mayor para SMO (34 vs. 15). De estos 34 positivos, SMO etiquetó erróneamente 21 secuencias como lysine riboswitch (RF00168), mientras que MLP cometió este error en tan solo 7 secuencias de prueba (Figs. 15b) y 15c)).

En cuanto a la familia con menos instancias de entrenamiento (PreQ1-II), SMO y MLP tuvieron el mismo número de verdaderos positivos (15) y falsos negativos (1). Sin embargo, fue la diferencia entre el total de falsos positivos de MLP (3) y SMO (0) lo que afectó el valor-F (0.967 y 0.882, respectivamente). Respecto a la familia RF01055, el valor-F fue de 0.926 para SMO y 0.858 para MLP. En general, etiquetar secuencias de la familia RF00059 como RF01055 fue problemático para ambos clasificadores (SMO y MLP lo hicieron en 12 y 20 instancias, respectivamente).

En las matrices de confusión resultantes (Fig. 15), es posible observar que algunos clasificadores presentan un alto número de falsos positivos para RF00162, RF00174 y RF00504. Estas familias son las que tuvieron más instancias de entrenamiento, lo que pudiera indicar que estos clasificadores tuvieron problemas al discriminar debido al desbalance en dichas clases. Por ejemplo, los clasificadores J48-HEXCFS y NaiveBayes-

HEXCFS (Figs. 15d) y 15f), respectivamente) tendieron a predecir correctamente las secuencias pertenecientes a las clases más grandes, pero tuvieron un bajo desempeño en las clases minoría.

De nuevo, SMO-HEXCFS fue el mejor clasificador sin ser afectado por la diferencia de tamaño de las clases (Tabla 10). El valor mínimo de valor-F en clases individuales fue de 0.802 y 0.880 para RF00380 y RF00168, respectivamente. En el resto de las 14 familias, el valor-F fue mayor o igual a 0.919. Observe que incluso las familias más pequeñas ($\leq$ 129 secuencias de entrenamiento) alcanzaron un valor-F muy alto (RF00521: 0.985, RF00522: 0.978, RF00634: 0.919, RF01054: 0.968 y RF01057: 0.968). En general, este clasificador fue muy específico ($\geq$ 0.978).

Tabla 10: Resultados por clase del clasificador SMO-HEXCFS en los datos de prueba.

Familia	Sen	Spe	Acc	F1
RF00050	0.971	0.998	0.996	0.974
RF00059	0.961	0.998	0.990	0.976
RF00162	0.972	0.997	0.996	0.971
RF00167	0.949	1.000	0.998	0.968
RF00168	0.831	0.998	0.992	0.880
RF00174	0.996	0.978	0.983	0.968
RF00234	0.936	0.999	0.998	0.949
RF00380	0.931	0.995	0.994	0.802
RF00504	0.984	0.998	0.996	0.986
RF00521	0.970	1.000	1.000	0.985
RF00522	0.957	1.000	1.000	0.978
RF00634	0.851	1.000	0.998	0.919
RF01051	0.934	1.000	0.996	0.965
RF01054	0.938	1.000	1.000	0.968
RF01055	0.975	0.997	0.997	0.926
RF01057	0.938	1.000	0.999	0.968

Es notable que, para todos los clasificadores, RF00168 fue la familia más difícil. En esta familia en particular, el mejor clasificador fue MLP-HEXCFS (sensibilidad= 0.893, especificidad= 0.996, valor-F= 0.895 y precisión= 0.891). Al contrario de lo que pudiera esperarse, la familia RF00168 no es de las más pequeñas (337 secuencias de entrenamiento).

Todos estos resultados tienen diferencias marginales respecto a los estimados en

la validación cruzada, también es posible observar el mismo efecto de las clases desbalanceadas.

Errores por secuencia de los clasificadores HEXCFS

Una vez obtenido el desempeño de cada uno de los clasificadores HEXCFS exploramos la posibilidad de si al unirlos en un ensamble se mejora la discriminación. Primero, investigamos si todos los clasificadores cometieron errores en las mismas secuencias. Para lograr esto, se agruparon las predicciones individuales de cada secuencia de prueba según el número de clasificadores que no fueron capaces de distinguirla. Así como muestra la Figura 16, solo 36.68 % de las secuencias fueron clasificadas correctamente por todos los modelos en consenso. Por el otro lado, ningún clasificador fue capaz de distinguir al 0.82 % de las secuencias, lo que pudiera considerarse como el error irreducible del ensamble a producir (Friedman *et al.*, 2001).

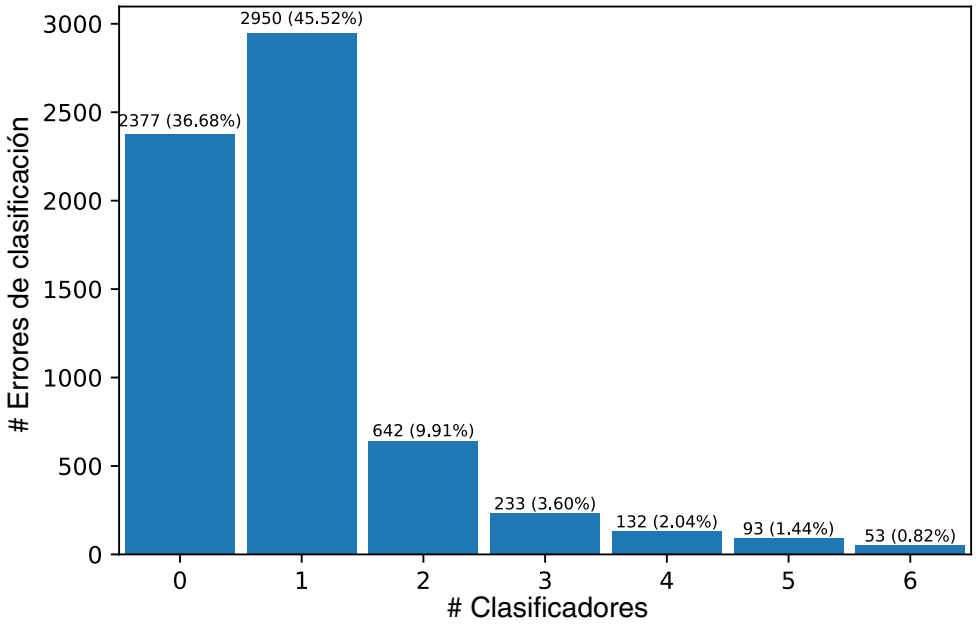

Figura 16: Errores de clasificación agrupados por el número de clasificadores que incurrieron en dicho error.

Segundo, se calculó la matriz de comparación de acuerdo a los errores en común entre pares de clasificadores y se mostró el porcentaje del error respecto al número total de secuencias de prueba (Figura 17). Se puede observar que los pares de clasifi-

cadores con la mayor cantidad de errores en común fueron (SMO, NB) = 2.7 %, (MLP, NB) = 2.7 %, (RF, J48) = 4.2 %, (RF, HP) = 4.2 %, (J48, HP) = 9.0 % y (NB, HP) = 11.0 %. Esto nos lleva a considerar el ensamble por votación como una opción viable para la combinación de las predicciones.

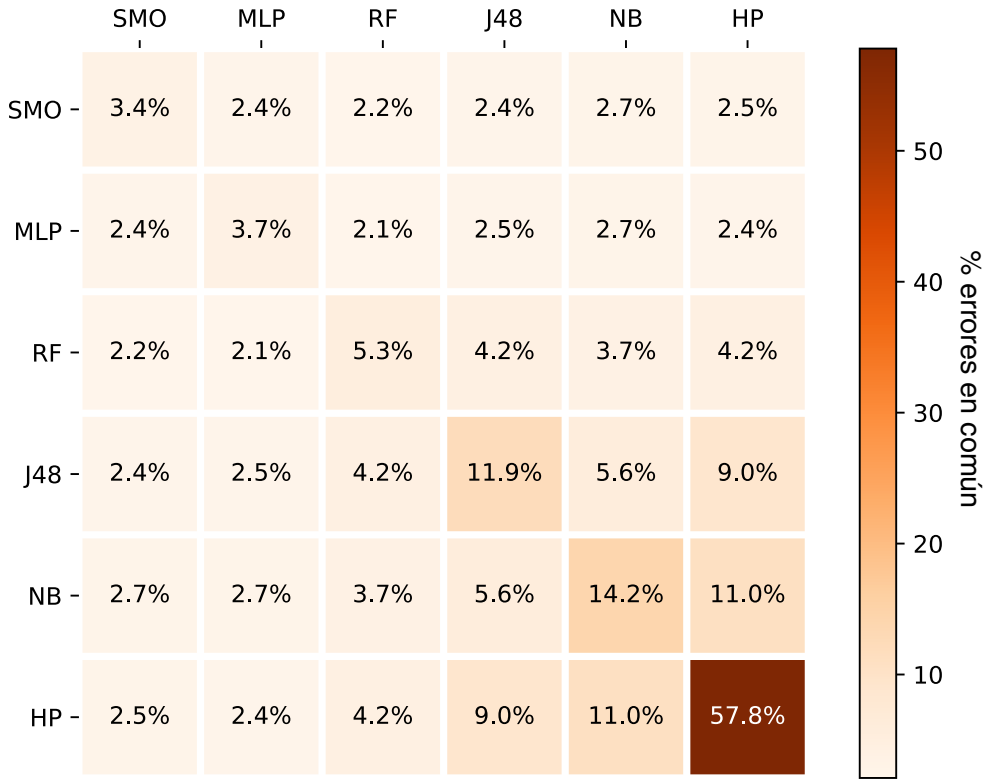

Figura 17: Porcentaje de secuencias mal clasificadas en común por cada par de modelos respecto del conjunto total de secuencias de prueba. La diagonal corresponde al error individual del modelo y también puede calcularse como $(1 - Sen) \times 100$.

A continuación, se construyeron un total de cinco ensambles de votación con las siguientes reglas: MAJ (votación por mayoría), AVG (promedio de probabilidades), PRODUCT (producto de probabilidades), MAX (máxima probabilidad), and MIN (probabilidad mínima). Los resultados se muestran en la Figura 18. A pesar de que el mejor ensamble (MAJ) alcanzó un F1 de 0.960, no es suficiente para superar a SMO-HEXCFS (F1 = 0.966).

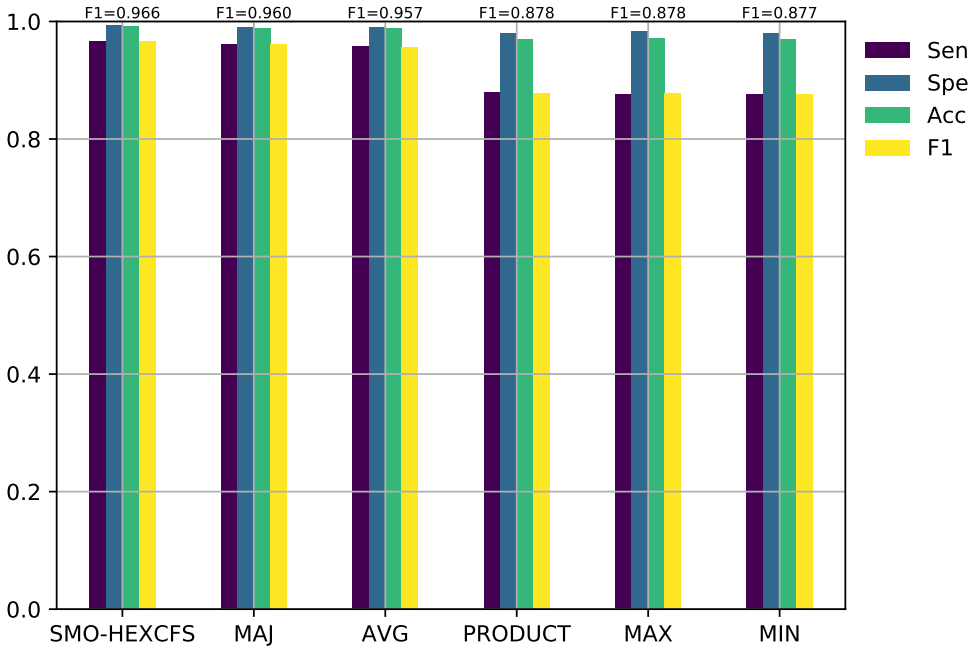

Figura 18: Comparación de SMO-HEXCFS y cinco ensambles de votación formados por los clasificadores HEXCFS.

3.3.5. Comparación con otros enfoques

3.3.5.1. MLP-DNC (Singh y Singh, 2017)

En esta sección se comparan los resultados obtenidos en la sección anterior con los del mejor clasificador reportado por Singh y Singh (2017) (Tabla 9), que consiste en un perceptrón multicapa inducido con frecuencias de monómeros y dímeros (de aquí en adelante referido como MLP-DNC).

En general, todos los clasificadores HEXCFS mejoraron el desempeño de MLP-DNC en las métricas valor-F y especificidad. Particularmente, nuestros tres mejores clasificadores (SMO-HEXCFS, MLP-HEXCFS y RF-HEXCFS) lo superaron en cada métrica evaluada. La Tabla 9 muestra que, en su mejor métrica, MLP-DNC alcanzó una sensibilidad de 0.914, rebasando a J48-HEXCFS, NB-HEXCFS y HP-HEXCFS. Sin embargo, HP-HEXCFS, el cual resultó nuestro peor clasificador, lo superó en valor-F (0.337 vs. 0.331). Por otro lado, al comparar MLP-DNC con nuestro mejor clasificador, SMO-HEXCFS (sensibilidad= 0.966, especificidad= 0.993, precisión= 0.991 y valor-F= 0.966),

podemos afirmar que el primero fue superado en cada métrica evaluada (sensibilidad= 0.914, especificidad= 0.261, precisión= 0.810 y valor-F= 0.331).

Es interesante notar que el clasificador MLP-HEXCFS se construyó usando la misma configuración reportada para MLP-DNC, obteniendo incluso mejores resultados (sensibilidad: 0.963 vs 0.914, especificidad: 0.993 vs. 0.261, precisión: 0.963 vs. 0.810 y valor-F: 0.963 vs. 0.331). Por consiguiente, se podría conjeturar que el conjunto de características HEXCFS induce mejores clasificadores para este conjunto de secuencias que el conjunto DNC.

3.3.5.2. BLAST+ (Camacho *et al.*, 2009)

Considerando que el conjunto de características empleado en este estudio está basado solo en la estructura primaria y no la secundaria, un punto de comparación intuitivo es comparar su desempeño contra un algoritmo de búsqueda de secuencias como BLAST. Con este objetivo, se utilizó BLAST+ (Camacho *et al.*, 2009). Se espera que los hits reportados por BLAST sean secuencias de prueba que tengan suficiente similitud con las de entrenamiento, y aquellas diferentes no reporten un alto nivel de confianza (*E*-value). En esta sección investigamos si nuestros clasificadores son capaces de superar tal limitación.

Con la finalidad de usar BLAST como un clasificador, se construyó una base de datos con las secuencias de entrenamiento y se tomó el mejor hit de BLAST como el vecino más cercano. El valor-F obtenido por BLAST (0.995) lo posicionó como el mejor clasificador en las secuencias de prueba. No obstante, el mejor clasificador HEXCFS resultó ser competitivo en especificidad (0.993 vs. 0.999) y precisión (0.991 vs. 0.999). Al comparar el valor-F por familia, SMO-HEXCFS fue capaz de alcanzar a BLAST solo en la familia RF00522 (0.978). Los valores más bajos que obtuvo BLAST fueron en las familias RF00168 (valor-F = 0.961) y RF01055 (valor-F = 0.966). En particular, la familia RF00168 (lysine riboswitch) fue la familia en la que BLAST arrojó la mayor cantidad de falsos negativos (17 de 225). De estas instancias, BLAST etiquetó erróneamente 9 como RF00174 (cobalamin riboswitch). Este fue un error común en todos los clasificadores evaluados con la excepción de MLP, que cometió el mismo error en solo 6

secuencias (Fig. 15).

3.3.5.3. Secuencias difíciles para BLAST

Puesto que virtualmente no hubo límite para la búsqueda BLAST, el valor mínimo del *E*-value de los errores de BLAST (1.3E-02) se consideró como umbral para recuperar las secuencias de prueba y poder evaluar el desempeño de los demás clasificadores (Tabla 11 y Figura 19).

Tabla 11: Resultados al clasificar los hits con *E*-value ≥ 1.3E-2. Los resultados se encuentran ordenados por *F*1 en orden descendente. La celdas en negrita representan el mejor valor por columna.

Clasificador	Sen	Spe	Acc	F1
MLP-HEXCFS	**0.783**	0.982	**0.914**	**0.829**
SMO-HEXCFS	0.739	**0.998**	0.913	0.815
BLAST	0.730	0.990	0.907	0.784
RF-HEXCFS	0.496	0.978	0.797	0.563
NB-HEXCFS	0.443	0.971	0.805	0.525
J48-HEXCFS	0.296	0.978	0.766	0.381
HP-HEXCFS	0.061	0.984	0.723	0.077

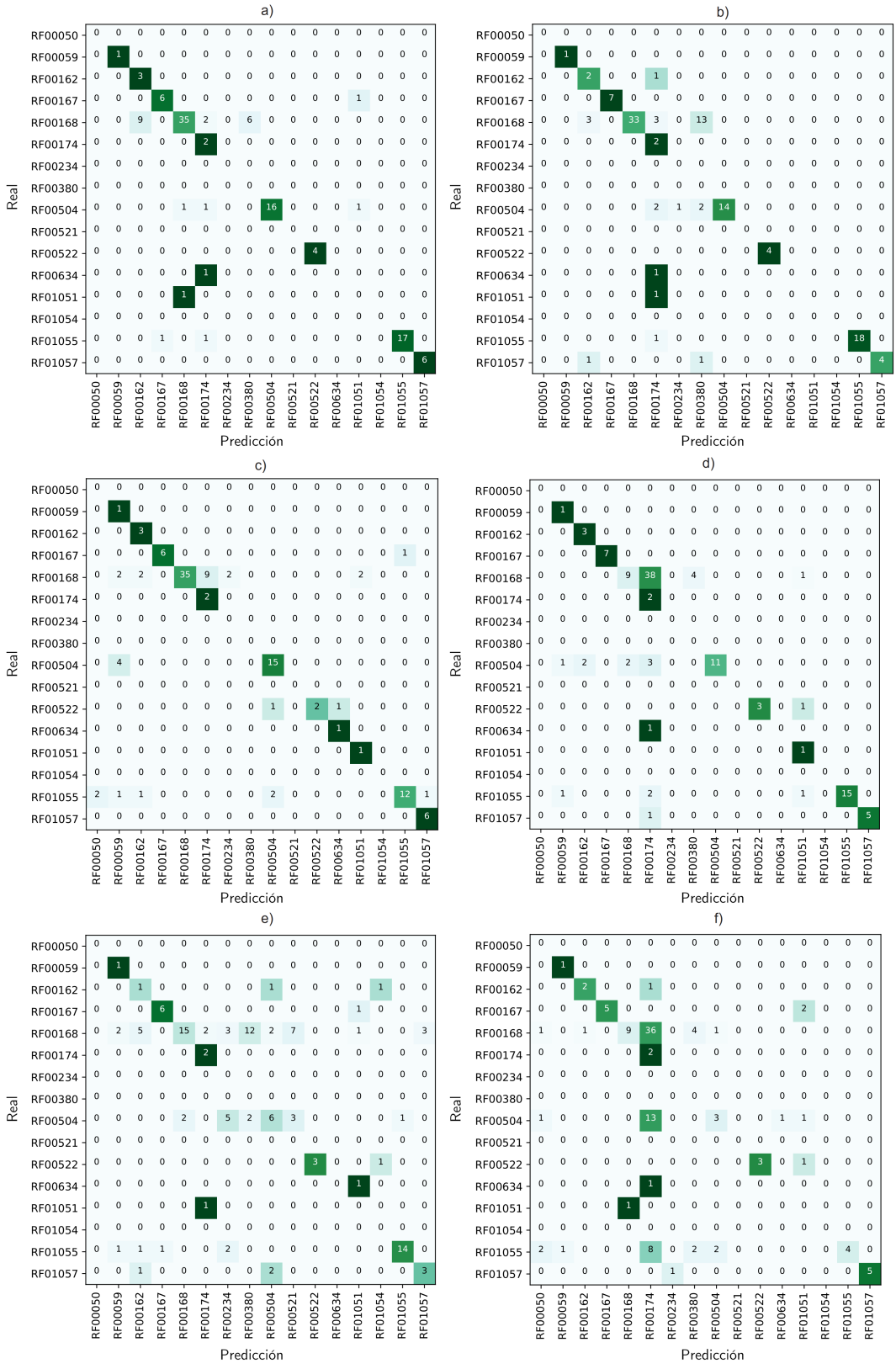

Figura 19: Matrices de confusión para la clasificación de hits con *E*-value ≥ 1.3E-02. a) MLP-HEXCFS; b) SMO-HEXCFS; c) BLAST ; d) RF-HEXCFS; e) NB-HEXCFS; f) J48-HEXCFS. Los clasificadores están ordenados por valor-F.

En total se obtuvieron 115 secuencias (RF00059: 1, RF00162: 3, RF00167: 7, RF00168: 52, RF00174: 2, RF00504: 19, RF00522, 4: RF00634: 1, RF01051: 1, RF01055: 19 y RF01057: 6).

En este conjunto, el mejor clasificador fue MLP-HEXCFS (sensibilidad = 0.783, especificidad = 0.982, precisión = 0.914 y valor-F = 0.829). Curiosamente, en dicho subconjunto MLP-HEXCFS y SMO-HEXCFS superaron a BLAST en sensibilidad (0.783 y 0.739 vs. 0.730), precisión (0.914 y 0.913 vs. 0.907) y valor-F (0.829 y 0.815 vs. 0.784); solo SMO-HEXCFS pudo superar a BLAST en especificidad (0.998 vs. 0.990).

El desempeño por familia para BLAST y MLP-HEXCFS se muestra en la Tabla 12.

Tabla 12: Comparación entre MLP-HEXCFS y BLAST en los hits con *E*-value ≥1.3E-2.

Familia	Soporte	Sen		Spe		Acc		F1	
		BLAST	MLP	BLAST	MLP	BLAST	MLP	BLAST	MLP
RF00059	1	1.000	1.000	0.939	1.000	0.939	1.000	0.222	1.000
RF00162	3	1.000	1.000	0.973	0.920	0.974	0.922	0.667	0.400
RF00167	7	0.857	0.857	1.000	0.991	0.991	0.983	0.923	0.857
RF00168	52	0.673	0.673	1.000	0.968	0.852	0.835	0.805	0.787
RF00174	2	1.000	1.000	0.920	0.956	0.922	0.957	0.308	0.444
RF00504	19	0.789	0.842	0.969	1.000	0.939	0.974	0.811	0.914
RF00522	4	0.500	1.000	1.000	1.000	0.983	1.000	0.667	1.000
RF00634	1	1.000	0.000	0.991	1.000	0.991	0.991	0.667	0.000
RF01051	1	1.000	0.000	0.982	0.982	0.983	0.974	0.500	0.000
RF01055	19	0.632	0.895	0.990	1.000	0.930	0.983	0.750	0.944
RF01057	6	1.000	1.000	0.991	1.000	0.991	1.000	0.923	1.000

Aunque ambos algoritmos tuvieron un desempeño similar, MLP-HEXCFS alcanzó un valor-F de 1.0 en tres familias y superó a BLAST en dicha métrica en 6 familias de 11: RF00059 (1.0 vs. 0.222), RF00174 (0.444 vs. 0.380), RF00504 (0.914 vs. 0.811), RF00522 (1.0 vs. 0.667), RF01055 (0.944 vs. 0.750) y RF01057 (1.0 vs. 0.923). Sin embargo, se puede apreciar en la Fig. 19 que solo BLAST fue capaz de clasificar a las familias RF00634 y RF01051. Respecto a la familia con más secuencias en esta configuración, RF00168, ambos clasificadores obtuvieron resultados similares (0.805 y 0.787 para BLAST y MLP, respectivamente). La sensibilidad fue igual de baja (0.673) porque obtuvieron 17 falsos negativos de 52 instancias (curiosamente, de diferentes familias).

3.4. Conclusiones

Este estudio tuvo la finalidad de mejorar el desempeño de algoritmos de aprendizaje máquina inducidos con características derivadas de la estructura primaria del ARN para la clasificación de secuencias putativas de riboswitches en 16 familias conocidas. Para lograr esto, se propuso un conjunto de 156 características, HEXCFS, el cuál se obtuvo utilizando selección de características en el conjunto de frecuencias *k*-méricas desde monómeros hasta hexámeros.

Se entrenaron seis algoritmos de clasificación con el conjunto HEXCFS: Multilayer perceptron, Naive Bayes, Random Forest, SMO, J48 e HyperPipes. Al evaluar en datos de prueba, todos los clasificadores con excepción de HyperPipes fueron capaces de alcanzar un valor-F mayor o igual a 0.863. Los dos mejores clasificadores, SMO-HEXCFS y MLP-HEXCFS, obtuvieron un valor-F de 0.996 y 0.963, respectivamente. En cuanto a los resultados por familia, ambos clasificadores obtuvieron un valor mayor o igual a 0.802 en esta métrica, sin importar que los datos estuvieran desbalanceados.

Al comparar los resultados obtenidos con otro enfoque basado en características de secuencia (Singh y Singh, 2017), SMO-HEXCFS mejoró dicho trabajo en cada métrica considerada: valor-F (0.966 vs. 0.331), especificidad (0.993 vs. 0.261), precisión (0.991 vs. 0.810) y sensibilidad (0.966 vs. 0.914). El hecho de que todos los clasificadores superaron al trabajo previo en valor-F implica que HEXCFS constituye un mejor conjunto de características para esta tarea en particular.

Adicionalmente, comparamos nuestros clasificadores con una búsqueda BLAST, los cuales fueron capaces de obtener resultados competitivos (valor-F de 0.966 y 0.995 para SMO-HEXCFS y BLAST+, respectivamente). Sin embargo, al evaluar el desempeño de la clasificación de los hits tales que *E*-value $\geq$ 1.3E-2, encontramos que nuestro enfoque de aprendizaje máquina obtuvo mejores resultados (valor-F de 0.829 y 0.784 para MLP-HEXCFS y BLAST+, respectivamente). Reconocemos el hecho de que los clasificadores presentados en este trabajo pudieran no ser aptos todavía para sustituir a BLAST, pero quizás pudieran ser empleados para obtener una segunda opinión en hits con alto *E*-value, o incluso como el componente de un ensamble en un flujo de trabajo

de toma de decisiones.

Con la finalidad de representar gráficamente la cobertura de los *k*-meros en un conjunto de secuencias dado, se describió una herramienta de visualización llamada *k*-mer covMap. El covMap para los *k*-meros presentes en el conjunto HEXCFS nos ayudó a elucidar que algunas de las características seleccionadas pudieran tener la función de motivos discriminantes (o una parte de un motivo), lo que pudiera explicar los altos resultados de clasificación. Esto también abre la posibilidad de explorar los algoritmos de selección de características como métodos libres de alineamiento para obtener motivos discriminantes. Además, lo anterior nos llevó a considerar el hecho de que no es suficiente con seleccionar regiones altamente conservadas como motivos para clasificación, sino que es también necesario asegurar que esas regiones no sean tan frecuentes en otras familias.

La caracterización de elementos funcionales de un genoma en clases funcionales usando motivos es una parte básica del proceso de anotación (Yip *et al.*, 2013). Aunque en este trabajo nos centramos en la discriminación inter-clase de riboswitches putativos, estos hallazgos pudieran ser potencialmente útiles en flujos de trabajo integrales que busquen descubrir este tipo de moléculas en una escala genómica. Como trabajo futuro, este trabajo se extenderá para distinguir riboswitches de otras secuencias genómicas para ir más allá de la discriminación inter-clase, así como considerar símbolos de base degenerados (alfabeto IUPAC) en la construcción de conjuntos de características para aumentar la cobertura de los motivos.

Finalmente, pensamos que esta metodología pudiera emplearse en la clasificación de otras familias de ARN no codificante que tengan conservación parcial a nivel de secuencia.

Capítulo 4. Clasificación de ARN inmunomodulador

4.1. Introducción

El ARN no codificante regulatorio (especialmente de pequeño tamaño como miRNA, snoRNA y siRNA) es de interés debido a su participación en la regulación de genes, como biomarcador y en la genética del desarrollo (Stefani y Slack, 2008). También presenta potencial terapéutico, como por ejemplo su uso como activador eficiente del proceso de interferencia del ARN (RNAi por sus siglas en inglés) (Olejniczak *et al.*, 2010).

Aunque hoy en día es posible encontrar reactivos comerciales diseñados para aplicaciones de RNAi, en escenarios reales dichos reactivos pudieran desencadenar respuestas inmunes no específicas a las secuencias, las cuales son capaces de causar inmunotoxicidad (Olejniczak *et al.*, 2012). Sin embargo, de ser controladas, estas propiedades inmunoestimuladoras pudieran aumentar el efecto de las terapias que involucran el silenciamiento de genes en enfermedades tales como cáncer e infecciones virales. Las secuencias de cadena sencilla de ARN (ssRNA) que tienen estas propiedades son llamadas ribonucleótidos inmunomoduladores (IMORNs por sus siglas en inglés).

Discriminar entre ribonucleótidos que son inmunomoduladores o no es un paso clave en el diseño de adyuvantes inmunológicos basados en ARN con un potencial inmunomodulador deseado, por lo que efectuar esta clasificación de manera efectiva es fundamental en un proceso de cribado virtual que busque este tipo de secuencias (Chaudhary *et al.*, 2016; Nagpal *et al.*, 2017). En un trabajo previo se realiza esta clasificación mediante aprendizaje supervisado en un conjunto de 1122 ssRNAs reportadas en la literatura con longitudes de 17 a 27 nucleótidos. En este capítulo se propone un conjunto diferente de características basadas en secuencia, así como la utilización de un algoritmo de selección de características y distintos métodos de clasificación para mejorar los resultados previos en la discriminación de IMORNs. Demostramos que el uso de características a nivel de secuencia obtiene resultados similares en sensibilidad, mientras que se mejoran las métricas de precisión, especificidad y el coeficiente

de correlación de Matthews (MCC). El resto del capítulo se organiza como sigue: en primer lugar se presenta la metodología, en segundo se muestran y discuten los resultados obtenidos, para finalizar con las conclusiones[1].

4.2. Metodología

4.2.1. Conjunto de datos

En este trabajo se utilizó el conjunto de secuencias propuestas por Chaudhary *et al.* (2016). Dicho conjunto de datos se compone de 1122 ssRNAs con longitudes de 17 a 27 bases. En total se tienen 602 secuencias no redundantes con actividad inmunomoduladora verificada experimentalmente. Estas instancias de la clase positiva se obtuvieron de la base de datos RNAimmuno (Olejniczak *et al.*, 2012) y cuatro patentes no especificadas. Respecto al conjunto negativo, no se encontraron secuencias de ARN que experimentalmente hayan sido demostrado no tener ninguna actividad inmunomoduladora, por lo que los autores del trabajo previo seleccionaron 520 secuencias miRNA no redundantes como instancias negativas provenientes de la base de datos miRandola (Russo *et al.*, 2012). Los miRNAs seleccionados fueron recuperados de fluidos extracelulares de tejidos humanos tanto saludables como cancerosos; por esta razón, los autores hicieron la suposición de que no tenían actividad inmunomoduladora.

4.2.2. Extracción de características

Para los experimentos se diseñaron dos conjuntos de características derivados de la estructura primaria. El primero es el mejor conjunto reportado por el trabajo previo, denominado PNF. PNF contiene las frecuencias de los pentanucleótidos, sumando un total de 1024 características. Como segundo conjunto, se propone en este trabajo el conjunto HEXNF, el cuál contiene las siguientes características:

1. El contenido `G+C` de la secuencia.

[1] Este capítulo es una versión extendida del artículo presentado en la *2017 International Joint Conference on Neural Networks* (Guillen-Ramirez *et al.*, 2017).

2. La longitud de la secuencia.

3. La composición $\{1, 2, 3, 4, 5, 6\}$-mérica, es decir, las frecuencias desde monómeros a hexanucleótidos (la formalización de este conjunto se discute en el Capítulo 3).

En total, el conjunto HEXNF contiene 5462 características, incluyendo completamente a las 1024 características del conjunto PNF.

Si bien no se agregaron al conjunto propuesto, también se consideró en un inicio el uso de características derivadas de la estructura secundaria en la extracción. Al realizar el análisis preliminar, el primer paso consistió en generar las estructuras secundarias de todas las secuencias. Para este fin se utilizó el programa RNAfold (Lorenz *et al.*, 2011). Sin embargo, se observó que puesto que la longitud de las secuencias era tan pequeña, la mayoría no presentaba una estructura secundaria predicha. Por lo tanto, para este conjunto particular de secuencias, la estructura secundaria no provee información relevante para discriminar entre las clases positiva y negativa, por lo que fue descartada para los experimentos subsecuentes.

4.2.3. Diseño experimental

El objetivo principal de este trabajo es analizar si es posible mejorar el desempeño de clasificación en el conjunto de IMORNs propuestos en el trabajo previo. El primer paso consistió en replicar la metodología e intentar mejorar los resultados usando clasificadores de distintas clases a SVM. Por consiguiente, se consideraron dos escenarios para los experimentos:

1. Reproducción de la metodología descrita por Chaudhary *et al.* (2016). En el artículo, el conjunto total de secuencias se divide en entrenamiento (80 %) y prueba (20 %), manteniendo el porcentaje de cada clase en ambos conjuntos (muestreo estratificado). Este muestreo se realiza diez veces para evitar el riesgo potencial de sesgo en el proceso de selección. Posteriormente, se realiza sintonización de parámetros en una SVM por medio del promedio del desempeño en la validación

cruzada de 5-pliegues para cada una de las muestras. Finalmente, se reporta el desempeño promedio de los clasificadores seleccionados con los datos de entrenamiento y prueba.

2. Puesto que el número de instancias totales es relativamente pequeño, se consideró emplear todo el conjunto para entrenamiento y evaluar mediante validación cruzada estratificada de 10 pliegues. Este método ha probado ser el mejor para evaluación de modelos en términos de sesgo y varianza (Kohavi, 1995).

Todos los experimentos se realizaron en Weka 3.6 (Hall *et al.*, 2009a) y LIBSVM (Chang y Lin, 2011).

4.2.4. Métricas de desempeño

El desempeño de clasificación se midió en términos de precisión (ACC), sensibilidad (SEN), especificidad (SPE) y el coeficiente de correlación de Matthews (MCC). Las fórmulas para calcular cada medida son las siguientes:

$$ACC = 100 \times \frac{TP + TN}{TP + FN + TN + FP}, \tag{27}$$

$$SEN = 100 \times \frac{TP}{TP + FN}, \tag{28}$$

$$SPE = 100 \times \frac{TN}{TN + FP}, \tag{29}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TN + FN)(TP + FN)(TN + FP)}}, \tag{30}$$

donde TP, FN, TN y FP corresponden al número de verdaderos positivos, falsos negativos, verdaderos negativos y falsos positivos, respectivamente.

4.2.5. Análisis exploratorio

Como una primera aproximación, se evaluaron los clasificadores disponibles en Weka y sus parámetros por omisión en validación cruzada de 5-pliegues en las secuencias de entrenamiento con ambos conjuntos de características (PNF y HEXNF). Los algoritmos que presentaron mejor desempeño en esta prueba preliminar fueron los siguientes: *Random Forest* (Breiman, 2001) (RF), *Support Vector Machine* (Boser *et al.*, 1992) (SVM), *Stochastic Gradient Descent* (Bottou, 2010) (SGD), *Sequential Minimal Optimization* (Platt, 1998) (SMO), *Stochastic Primal Estimated sub-GrAdient SOlver for Svm* (Shalev-Shwartz *et al.*, 2011) (SPegasos) e *Hyper Pipes* (Kalapanidas *et al.*, 2003a) (HP). Posteriormente, se siguió la metodología de prueba descrita por Chaudhary *et al.* (2016) en estos clasificadores. Los resultados del entrenamiento así como los correspondientes al mejor reportado en el trabajo previo (llamado SVM* de aquí en adelante) se muestran en la Tabla 13. Es posible observar que el clasificador RF con los parámetros por omisión superó el resultado reportado en la literatura en ambos conjuntos de características. Respecto al desempeño por conjunto de características, HEXNF obtuvo mejores resultados que PNF para cada clasificador inducido.

Basando la decisión en los resultados anteriormente descritos, se seleccionaron los clasificadores RF, SMO, SGD y SPegasos para la tercer ronda de experimentos. Puesto que las SVM ya fueron probadas en el enfoque previo y que HP no tiene parámetros por sintonizar que pudieran mejorar su desempeño, se descartaron ambos tipos de clasificadores en los experimentos posteriores.

4.2.6. Sintonización de parámetros

La sintonización de parámetros de los cuatro clasificadores seleccionados se realizó evaluando los resultados de validación cruzada de 10 pliegues. Los detalles de las configuraciones exploradas se muestran a continuación. El valor λ indica la cardinalidad del espacio de configuraciones explorado. Los valores de parámetros omitidos corresponden a los valores por omisión de Weka 3.6.

Tabla 13: Resultados preliminares de entrenamiento utilizando parámetros por defecto en los clasificadores y la metodología presentada por Chaudhary *et al.* (2016). CC denota "conjunto de características". Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
RF	HEXNF	92.86 ± 3.15	98.37 ± 1.50	95.41 ± 1.77	0.91 ± 0.03
RF	PNF	93.63 ± 2.48	94.38 ± 2.58	93.98 ± 1.72	0.88 ± 0.03
SVM*	PNF	93.96 ± 0.67	91.88 ± 1.41	93.00 ± 0.83	0.86 ± 0.02
HP	HEXNF	93.96 ± 2.46	91.68 ± 3.91	92.91 ± 2.22	0.86 ± 0.04
SGD	HEXNF	93.13 ± 2.64	92.50 ± 2.65	92.84 ± 1.85	0.86 ± 0.04
SMO	HEXNF	93.46 ± 2.52	91.95 ± 2.81	92.76 ± 1.94	0.85 ± 0.04
SVM	HEXNF	93.46 ± 2.52	91.95 ± 2.81	92.76 ± 1.94	0.85 ± 0.04
SPegasos	HEXNF	93.20 ± 2.59	92.02 ± 3.03	92.65 ± 1.93	0.85 ± 0.04
HP	PNF	92.12 ± 3.23	89.98 ± 4.18	91.13 ± 2.48	0.82 ± 0.05
SGD	PNF	92.78 ± 2.63	88.78 ± 3.04	90.92 ± 1.89	0.82 ± 0.04
SPegasos	PNF	92.55 ± 2.62	88.13 ± 3.01	90.50 ± 1.96	0.81 ± 0.04
SVM	PNF	92.76 ± 2.45	87.84 ± 3.11	90.48 ± 1.94	0.81 ± 0.04
SMO	PNF	92.76 ± 2.45	87.81 ± 3.12	90.47 ± 1.94	0.81 ± 0.04

- RF: número de árboles $I = 25, 50, 100, 200$; número de características por árbol $K = 0, 25, 50, 75$ ($\lambda = 16$).
- SGD: número de épocas de entrenamiento $E = 100, 250, 500, 1000$; constante de regularización lambda $R = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$ ($\lambda = 16$).
- SMO: tres funciones kernel distintas (PolyKernel, NormalizedPolyKernel y RBFKernel); constante de complejidad $C = 1.0, 2.0, 4.0$ ($\lambda = 9$).
- SPegasos: número de épocas de entrenamiento $E = 100, 250, 500, 1000$; constante de regularización lambda $R = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$ ($\lambda = 16$).

4.3. Resultados y discusión

4.3.1. Clasificación

La Tabla 14 muestra el desempeño de clasificación en el entrenamiento, mientras que la Tabla 15 los resultados en los datos de prueba.

Los resultados en los datos de entrenamiento obtenidos por los clasificadores RF y SMO en ambos conjuntos de características superaron los del SVM* en las métricas

Tabla 14: Mejores resultados en el entrenamiento por clasificador y conjunto de características utilizando la metodología de Chaudhary *et al.* (2016). CC denota "conjunto de características". Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
RF ($I = 200, K = 0$)	HEXNF	92.80 ± 3.17	98.63 ± 1.25	95.50 ± 1.72	0.91 ± 0.03
SMO ($C = 2.0$,NPK)	HEXNF	90.44 ± 3.08	99.42 ± 0.69	94.60 ± 1.73	0.90 ± 0.03
RF ($I = 200, K = 0$)	PNF	93.51 ± 2.60	94.83 ± 2.40	94.12 ± 1.79	0.88 ± 0.04
SMO ($C = 2.0$,NPK)	PNF	89.38 ± 3.39	99.52 ± 0.76	94.08 ± 1.85	0.89 ± 0.03
SVM*	PNF	93.96 ± 0.67	91.88 ± 1.41	93.00 ± 0.83	0.86 ± 0.02
SPegasos ($E = 100, R = 10^{-3}$)	HEXNF	93.46 ± 2.39	92.21 ± 2.92	92.88 ± 1.84	0.86 ± 0.04
SGD ($E = 500, R = 10^{-3}$)	HEXNF	93.17 ± 2.67	92.50 ± 2.62	92.86 ± 1.83	0.86 ± 0.04
SGD ($E = 100, R = 10^{-6}$)	PNF	92.82 ± 2.68	88.80 ± 3.08	90.96 ± 1.93	0.82 ± 0.04
SPegasos ($E = 1000, R = 10^{-5}$)	PNF	92.49 ± 2.78	88.92 ± 3.38	90.84 ± 2.01	0.82 ± 0.04

SEN, SPE y MCC. Nótese que el clasificador SMO casi alcanza una especificidad de 100, mientras que RF tiene casi la misma sensibilidad reportada para SVM* (92.80 vs. 93.96), pero con una clara mejora en especificidad (98.63 vs. 91.88). En cambio, SGD y SPegasos se desempeñaron igual o peor que SVM*.

La Tabla 15 muestra los resultados de la clasificación en los datos de prueba. En general, los clasificadores tuvieron buen rendimiento, obteniendo valores de precisión arriba de 91. De manera similar a los resultados del entrenamiento, los clasificadores RF y SMO superaron a SVM* utilizando cualquiera de los dos conjuntos de características. En términos de especificidad, puede observarse que los primeros cuatro clasificadores listados muestran una mejora considerable respecto al resultado de referencia, aunque la sensibilidad sea ligeramente menor que SVM*. La semejanza entre los resultados de entrenamiento y prueba indican que nuestros clasificadores no están sobreoptimizados (Chaudhary *et al.*, 2016).

La Tabla 16 muestra los resultados de la validación cruzada de 10 pliegues al inducir los clasificadores con el conjunto completo de datos. Puede observarse que los resultados son más altos que los mostrados en la Tabla 15, a excepción de la sensibili-

Tabla 15: Mejores resultados en los datos de prueba por clasificador y conjunto de características utilizando la metodología de Chaudhary *et al.* (2016). CC denota "conjunto de características". Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
RF ($I = 200, K = 0$)	HEXNF	92.00 ± 2.24	98.85 ± 1.20	95.18 ± 1.26	0.91 ± 0.02
SMO ($C = 2.0$,NPK)	HEXNF	90.75 ± 2.19	99.90 ± 0.29	95.00 ± 1.18	0.90 ± 0.02
SMO ($C = 2.0$,NPK)	PNF	90.58 ± 2.21	100.00 ± 0.00	94.96 ± 1.18	0.90 ± 0.02
RF ($I = 200, K = 0$)	PNF	93.50 ± 2.17	96.44 ± 2.47	94.87 ± 1.81	0.90 ± 0.04
SGD ($E = 500, R = 10^{-3}$)	HEXNF	93.42 ± 2.09	94.13 ± 2.45	93.75 ± 1.45	0.88 ± 0.03
SVM*	PNF	95.00 ± 2.39	91.73 ± 2.09	93.48 ± 1.49	0.87 ± 0.03
SPegasos ($E = 100, R = 10^{-3}$)	HEXNF	93.25 ± 1.99	91.15 ± 3.30	92.28 ± 2.07	0.85 ± 0.04
SPegasos ($E = 1000, R = 10^{-5}$)	PNF	92.58 ± 1.84	90.77 ± 2.79	91.74 ± 1.33	0.83 ± 0.03
SGD ($E = 100, R = 10^{-6}$)	PNF	92.75 ± 1.97	90.19 ± 2.57	91.56 ± 1.50	0.83 ± 0.03

dad. De manera congruente con los resultados anteriores, los clasificadores RF y SMO se desempeñaron mejor, y el mejor clasificador en esta prueba, RF-HEXNF, alcanzó valores de precisión, sensibilidad y especificidad de 96 %, 94 % y 98 %, respectivamente. Así como en los datos mostrados anteriormente, algunos clasificadores se acercan al valor de sensibilidad reportada para SVM*, mejorando al mismo tiempo las otras métricas evaluadas. Valores altos de especificidad tienen una consecuencia directa en procesos de cribado virtual, donde es de vital importancia reducir las muestras para validación en laboratorio. En general, los resultados presentados en este trabajo sugieren que el conjunto de características HEXNF puede inducir clasificadores capaces de distinguir qué secuencias no son IMORNs de una mejor manera que los clasificadores entrenados con el conjunto PNF. Con la intención de ver qué características son más importantes en HEXNF, se calculó el valor de ganancia de información (IG por sus siglas en inglés) para este conjunto. Las cinco características mejor puntuadas fueron el contenido `G+C` (IG = 0.64), longitud de la secuencia (IG = 0.59) y las frecuencias de `G`, `U` y `A` (valores IG de 0.56, 0.54 y 0.54, respectivamente). Curiosamente, todas estas características pertenecen a HEXNF pero no a PNF.

Tabla 16: Mejores resultados en validación cruzada de 10 pliegues por clasificador y conjunto de características. CC denota “conjunto de características”. Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
RF ($I = 50, K = 75$)	HEXNF	94.02 ± 4.15	98.46 ± 1.44	96.08 ± 2.33	0.92 ± 0.04
RF ($I = 100, K = 0$)	PNF	94.51 ± 3.50	96.54 ± 2.40	95.45 ± 2.84	0.91 ± 0.06
SMO ($C = 2.0$,NPK)	HEXNF	91.53 ± 3.20	99.62 ± 0.77	95.28 ± 1.83	0.91 ± 0.03
SMO ($C = 2.0$,NPK)	PNF	91.19 ± 2.99	99.62 ± 0.77	95.10 ± 1.56	0.91 ± 0.03
SPegasos ($E = 500, R = 10^{-4}$)	HEXNF	94.35 ± 3.51	93.85 ± 3.92	94.12 ± 2.26	0.88 ± 0.04
SGD ($E = 100, R = 10^{-3}$)	HEXNF	94.02 ± 2.99	94.04 ± 3.69	94.03 ± 1.87	0.88 ± 0.04
SVM*	PNF	95.00 ± 2.39	91.73 ± 2.09	93.48 ± 1.49	0.87 ± 0.03
SPegasos ($E = 500, R = 10^{-5}$)	PNF	93.52 ± 2.94	89.81 ± 3.55	91.80 ± 2.39	0.84 ± 0.05
SGD ($E = 100, R = 10^{-4}$)	PNF	93.85 ± 3.58	89.23 ± 3.12	91.71 ± 2.29	0.83 ± 0.05

Tabla 17: Mejores resultados de los ensambles en validación cruzada de 10 pliegues por clasificador y conjunto de características. CC denota “conjunto de características”. Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
RF (Tabla 16)	HEXNF	94.02 ± 4.15	98.46 ± 1.44	96.08 ± 2.33	0.92 ± 0.04
MAJV	HEXNF	93.36 ± 3.65	98.85 ± 1.28	95.90 ± 1.75	0.92 ± 0.03
MAJV	PNF	93.36 ± 2.98	98.08 ± 1.72	95.54 ± 2.15	0.91 ± 0.04
MAXP	HEXNF	94.69 ± 2.95	93.85 ± 3.92	94.30 ± 2.11	0.89 ± 0.04
SVM*	PNF	95.00 ± 2.39	91.73 ± 2.09	93.48 ± 1.49	0.87 ± 0.03
MAXP	PNF	94.35 ± 2.60	89.62 ± 3.77	92.16 ± 2.35	0.84 ± 0.05

Tabla 18: Mejores resultados en validación cruzada de 10 pliegues para los conjuntos de características reducidos. CC denota “conjunto de características”. Los resultados están ordenados por precisión en orden descendente.

Modelo		**Desempeño promedio**			
Clasificador	CC	SEN	SPE	ACC	MCC
MAJV	HEXNF_FSA	94.02 ± 3.17	99.04 ± 0.96	96.35 ± 1.62	0.93 ± 0.03
RF (Tabla 16)	HEXNF	94.02 ± 4.15	98.46 ± 1.44	96.08 ± 2.33	0.92 ± 0.04
MAJV	PNF_FSA	93.35 ± 3.80	98.08 ± 1.92	95.54 ± 2.62	0.91 ± 0.05
RF	HEXNF_CFS	95.18 ± 2.41	93.08 ± 3.00	94.21 ± 1.45	0.88 ± 0.03
RF	HEXNF_IGA	94.02 ± 2.70	93.08 ± 2.31	93.58 ± 1.43	0.87 ± 0.03
SVM*	PNF	95.00 ± 2.39	91.73 ± 2.09	93.48 ± 1.49	0.87 ± 0.03
MAJV	HEXNF_PCA	86.70 ± 5.07	97.31 ± 1.96	91.62 ± 2.97	0.84 ± 0.05
MAJV	PNF_PCA	82.72 ± 4.91	98.65 ± 1.23	90.11 ± 2.66	0.82 ± 0.05
RF	PNF_CFS	82.06 ± 3.86	97.12 ± 1.97	89.04 ± 2.09	0.79 ± 0.04
RF	PNF_IGA	80.08 ± 5.62	95.38 ± 2.61	87.17 ± 2.43	0.76 ± 0.04

4.3.2. Ensambles de los mejores clasificadores

Motivados por los resultados mostrados en la Tabla 16, se decidió explorar si era posible combinar las ventajas que presenta cada clasificador en un ensamble. Para lograr esto, se entrenaron ensambles compuestos de los tres mejores clasificadores por conjunto de características, es decir, RF, SMO y SPegasos. Los parámetros utilizados en cada uno son los que se reportan en la Tabla 16. El ensamble elegido fue votación, y se consideraron dos posibles reglas para cada ensamble:

- *Mayoría de votos* (MAJV) (Kuncheva, 2004): la clase seleccionada como la predicha es aquella que tenga la mayor cantidad de votos. Dado que el ensamble tiene un número impar de clasificadores, no existen los empates.
- *Probabilidad máxima* (MAXP) (Kittler *et al.*, 1998): este esquema supone que los clasificadores son mútuamente independientes dada una etiqueta de clase (independencia condicional), y adapta la regla de decisión Bayesiana para el cálculo de la probabilidad posterior con el objetivo de maximizarla. La regla de probabilidad máxima entonces aproxima la suma de las probabilidades posteriores de cada clasificador.

Los resultados de los ensambles se muestran en la Tabla 17. Aunque los ensambles propuestos no mejoran al mejor clasificador encontrado en la etapa de experimentos anterior, casi todos ellos mejoran a SVM* en términos de MCC.

4.3.3. Selección de características

En general, los clasificadores entrenados con el conjunto de características HEXNF tuvieron mejor desempeño. Puesto que HEXNF (5462 características) es cinco veces más grande que PNF (1024 características), se exploró la aplicación de selección de características a ambos conjuntos. Para lograr lo anterior se aplicaron los siguientes cuatro algoritmos a los conjuntos de características:

- *Correlation-based Feature Subset Selection* (CfsSubsetEval) (Hall, 1999): evalúa

la importancia de un subconjunto de característica considerando la capacidad predictiva individual de cada característica junto con el grado de redundancia entre ellas. Para lo anterior se utilizó la implementación de Weka, seleccionando la búsqueda voraz del espacio de subconjuntos. El resultado de este algoritmo fueron los conjuntos PNF_CFS (39) y HEXNF_CFS (83).

- *Information Gain Attribute Evaluation* (InfoGainAttributeEval) (Yang y Pedersen, 1997): evalúa la importancia de un atributo por medio de la ganancia de información de dicha característica respecto a la clase. El resultado de este algoritmo fueron los conjuntos PNF_IGA (50) y HEXNF_IGA (50).
- Componentes principales (PrincipalComponents) (Pearson, 1901): realiza el análisis de componentes principales y la transformación de los datos. El resultado de este algoritmo fueron los conjuntos PNF_PCA (365) y HEXNF_PCA (522).
- Selección por algoritmo genético (GeneticAlgorithmSVM) (Beltrán Verdugo, 2014): este algoritmo se compone de dos elementos principales: un algoritmo genético encargado de la generación y búsqueda eficiente de subconjuntos de características, y un SVM que evalúa la calidad del subconjunto seleccionado. El resultado de este algoritmo fueron los conjuntos PNF_FSA (594) y HEXNF_FSA (2939).

Posteriormente, se evaluaron los clasificadores RF, SMO, MAJV y MAXP en validación cruzada de 10 pliegues utilizando todas las secuencias con los conjuntos reducidos (Tabla 18). Los parámetros de cada clasificador son los mismos que están indicados en la Tabla 16 y la Tabla 17.

Obsérvese que el ensamble MAJV entrenado con las características HEXNF_FSA alcanzó el mejor resultado en las métricas de especificidad (99.04), precisión (96.35) y MCC (0.93). Aunque no fueron los mejores clasificadores, tanto RF-HEXNF_CFS como RF-HEXNF_IGA y MAJV-PNF_FSA obtuvieron mejor desempeño que SVM* en términos de MCC. En general podemos afirmar que la selección de características demostró ser de utilidad para mejorar los resultados obtenidos hasta el momento. Además, es importante destacar que, a excepción de HEXNF_FSA, todos los conjuntos contienen menos características que PNF, lo que permite generar clasificadores más rápidos.

4.4. Conclusiones

El objetivo principal de este trabajo fue explorar hasta qué punto era posible mejorar los resultados presentados en el trabajo de Chaudhary *et al.* (2016) para la predicción de ARN inmunomodulador al extraer diferentes características de las secuencias, utilizando diversos clasificadores e incluso selección de características.

Como primer aporte, se propuso el conjunto de características HEXNF. En un análisis preliminar, se seleccionaron seis clasificadores, los cuales fueron inducidos con este nuevo conjunto. Con la finalidad de realizar una comparación justa, se consideró el mejor conjunto de características reportado en el trabajo previo, llamado PNF, y se desarrolló la metodología de clasificación de la misma manera en la que fue reportada. Al comparar los mejores clasificadores con el mejor modelo reportado (SVM*), observamos que los nuestros lo superan en todas las métricas excepto en sensibilidad, donde los resultados eran competitivos. Posteriormente, se hizo una prueba de validación cruzada de 10 pliegues con todas las secuencias. En este escenario, el comportamiento del desempeño de los clasificadores fue similar a los anteriores.

Motivados por los resultados obtenidos hasta ese momento, se consideraron ensambles de clasificadores, así como métodos de selección de características para mejorar la clasificación. Al usar un ensamble de votación con regla de mayoría con los clasificadores RF, SMO y SPegasos inducido con las características reducidas por el algoritmo GeneticAlgorithmSVM, los resultados obtenidos en validación cruzada de 10 pliegues en todas las secuencias fueron 96.35 en precisión, 94.02 en sensibilidad, 99.04 en especificidad y 0.93 en MCC. Por consiguiente, se puede concluir que el enfoque propuesto resulta apropiado para la clasificación de IMORNs.

Capítulo 5. Ingeniería de características basadas en la secuencia del ARN

5.1. Introducción

Un motivo en una secuencia biológica es un patrón sobrerrepresentado con respecto a un modelo de fondo (Song y Gu, 2014). Los motivos por lo general están asociados a la evolución: debido a la función que realizan, pasan de generación en generación. Por ejemplo, existen motivos asociados al inicio de la transcripción, la identificación del punto de auto-corte para ribosimas, entre otros.

Sin embargo, en ocasiones no basta con identificar que una cadena sea un motivo, sino que además existan en un conjunto de secuencias pero no en algún otro (usualmente llamado conjunto negativo). Estos motivos son llamados discriminantes y tienen aplicaciones en el análisis de expresión diferencial y clasificación de secuencias (Song y Gu, 2014). Generalmente, los programas que encuentran motivos discriminantes lo hacen entre el conjunto positivo y negativo (es decir, dos clases). Ejemplos de estos algoritmos son WordSpy (Wang *et al.*, 2005), DEME (Redhead y Bailey, 2007) y MERCI (Vens *et al.*, 2011). Estos métodos son lentos debido a que trabajan con alineamientos múltiples. Además, no todos los problemas de este tipo están limitados a dos clases, lo que aumenta la complejidad de las soluciones.

En el Capítulo 3 se mostró que algunos k-meros seleccionados por el algoritmo CFS pueden considerarse motivos discriminantes. Particularmente, esta observación se derivó del mapa de cobertura k-mérica (k-mer covMap), donde fue posible apreciar que ciertos k-meros son más conservados en una familia de riboswitches que en el resto de ellas. En este capítulo se introduce la función de evaluación q para evaluar de manera rápida la unicidad de un k-mero, así como el conjunto de éstos con la función de cobertura cov (denominado el framework QCOV). A su vez, se muestran tres estrategias para la selección de características basadas en este framework.

5.2. Framework QCOV

En esta sección se presentan las funciones de evaluación que conforman el framework QCOV. A continuación se enuncian las definiciones formales que se utilizarán a lo largo del capítulo.

5.2.1. Definiciones

De aquí en adelante, las cadenas consideradas están sobre el alfabeto de ribonucleótidos $\Sigma_{RNA} = \{\texttt{A,U,C,G}\}$, sin embargo el enfoque puede generalizarse a otros alfabetos como el de nucleótidos $\Sigma_{DNA} = \{\texttt{A,T,C,G}\}$, nucleótidos ambiguos de IUPAC $\Sigma_{DNAext} = \{\texttt{ G, A, T, C, R, Y, W, S, M, K, H, B, V, D, N }\}$ o el de aminoácidos IUPAC $\Sigma_{AA} = \{\texttt{ A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y }\}$.

Un k-mero se define como una cadena de longitud k. Para Σ_{RNA}, existen 4^k k-meros diferentes y $\frac{4}{3}(4^n - 1)$ k-meros de longitudes 1 a k. Sean p un k-mero y s una cadena, ambos sobre el alfabeto Σ_{RNA}. La frecuencia de p en s ($freq(p, s)$) se define como la razón del número de veces que p aparece como una subcadena de s ($count(p, s)$) y el número de k-meros en s, esto es,

$$freq(p, s) = \frac{count(p, s)}{|s| - k + 1}. \tag{31}$$

Sea I una función tal que:

$$I(p, s) = \begin{cases} 1 & \text{si } p \text{ es una subcadena de } s, \\ 0 & \text{otro caso.} \end{cases} \tag{32}$$

Las Ec. 31 y 32 se pueden relacionar de la siguiente manera:

$$I(p,s) = \begin{cases} 1 & \text{si } freq(p,s) > 0, \\ 0 & freq(p,s) = 0. \end{cases} \tag{33}$$

Sea S un conjunto de secuencias sobre el alfabeto y C un conjunto de etiquetas (categorías), de tal manera que $f : S \twoheadrightarrow C$ es una función suprayectiva; es decir, para cada categoría $c \in C$ existe al menos una secuencia $s \in S$ etiquetada como perteneciente a la clase c y cada $s \in S$ se etiqueta con una única categoría. De esta manera, se denota como $S_c = f^{-1}(c)$ a la partición de secuencias de S etiquetadas con la categoría c.

Entonces, la cobertura del k-mero p en la categoría c se define como:

$$cov(p,c) = \frac{\sum_{s \in S_c} I(p,s)}{|S_c|}. \tag{34}$$

En otras palabras, $cov(p,c)$ indica la proporción de secuencias de la clase c donde p aparece como subcadena al menos una vez, por lo que el rango de esta función es $[0,1]$. Si $cov(p,c) = 1$, entonces todas las secuencias que pertenecen a la clase c contienen a p. En caso contrario, cuando $cov(p,c) = 0$, p no está presente en ninguna secuencia de la clase c.

Sin embargo, note que un k-mero con una cobertura de 1 no significa que sea discriminante, por lo que de manera complementaria, se propone la función $q(p,s)$, la cual mide la "calidad" del motivo (qué tan discriminante resulta un k-mero para una familia en particular):

$$q(p,c) = \begin{cases} \frac{\sum_{s \in S_c} I(p,s)}{\sum_{s \in S} I(p,s)} & \text{si } \sum_{s \in S} I(p,s) > 0, \\ 0 & \text{otro caso.} \end{cases} \tag{35}$$

La función se define por partes debido a que es posible que p no aparezca en ninguna $s \in S$ y en tal caso se asigna el valor de 0. Si al menos una secuencia contiene a p, el numerador de la función q es el mismo que el de cov; sin embargo, el denominador representa el número de apariciones de p en las secuencias de todas las

clases, por lo que también el intervalo de esta función es $[0, 1]$. Puede verificarse que $\sum_{c \in C} q(p, c) = 1$ si $\sum_{s \in S} I(p, s) > 0$.

Si existe alguna clase c tal que $q(p, c) = 1$, significa que todas las secuencias que contienen a p están incluidas en c. Note que a diferencia de cov, la función q no está ponderada por el número de secuencias en la clase c. Por ejemplo, suponga un conjunto de secuencias S distribuidas en dos clases c_1 y c_2 tales que $|S_{c_1}| = 1$ y $|S_{c_2}| = 10000$. Suponga además que hay una p tal que $cov(p, c_1) = 1$ y $cov(p, c_2) = 0.001$. Esto significa que p está presente en la única secuencia de c_1 y en 10 secuencias de c_2, por lo que $q(p, c_1) = \frac{1}{1+10} = 0.09$ y $q(p, c_2) = \frac{10}{1+10} = 0.90$. Este ejemplo muestra claramente la necesidad de combinar ambas funciones para seleccionar motivos discriminantes (Tabla 19).

Tabla 19: Interpretación de los valores límites QCOV.

	$cov \to 0$	$cov \to 1$
$q(p, c) \to 0$	El motivo no está presente en c.	El motivo está altamente presente en c, pero se presenta más en otras clases.
$q(p, c) \to 1$	El motivo es específico para la clase c pero aparece en muy pocas secuencias (sobreajuste).	El motivo está altamente presente en c y es específico para la clase c (ideal).

Una primera impresión de la Tabla 19 consiste en que los motivos discriminantes de un conjunto de datos se ubicarían en la esquina inferior derecha de la tabla (valores q y cov cercanos a 1). Sin embargo, encontrar k-meros con estos valores depende enteramente del conjunto de datos. Además, el hecho de que se tenga que elegir entre un valor u otro sugiere que la selección de k-meros basados en q y cov puede plantearse como un problema multi-objetivo, el cuál se enuncia como *encontrar el conjunto de k-meros* $\{p \mid p \in \Sigma^+_{RNA}\}$ *que maximicen las funciones* $cov(p, c)$ *y* $q(p, c)$ *para una clase* $c \in C$. En la siguiente sección se desarrolla cómo ambas funciones pudieran estar en compromiso.

5.2.2. Funciones q y cov como objetivos en compromiso

Para demostrar que cov y q están en compromiso es necesario probar que cuando cov decrece, q se incrementa y viceversa. Sean $c_i \in C$ una etiqueta y $p_j \in \Sigma_{RNA}^j$ un j-mero sobre el alfabeto de ribonucleótidos tal que $\sum_{s \in S_{c_i}} I(p_j, s) > 0$. Observe que lo anterior implica que $\sum_{s \in S} I(p_j, s) > 0$. Suponga también que p_{j+1} es la concatenación de p_j con algún símbolo del alfabeto Σ_{RNA}^j.

En primer lugar, se demostrarán un lema y un corolario para apoyar los argumentos que se van a presentar.

Lema 5.1. $\sum_{s \in S_{c_i}} I(p_j, s) \geq \sum_{s \in S_{c_i}} I(p_{j+1}, s)$.

Demostración. Por contradicción, suponga que existe una p_{j+1} tal que $\sum_{s \in S_{c_i}} I(p_j, s) < \sum_{s \in S_{c_i}} I(p_{j+1}, s)$. Esto significa que existe una secuencia en S_{c_i} que contiene a p_{j+1} como subcadena pero no a p_j, lo cual es una contradicción debido a que p_j es una subcadena de p_{j+1}. □

Corolario 5.1. $\sum_{s \in S_{cm}, m \neq i} I(p_j, s) \geq \sum_{s \in S_{cm}, m \neq i} I(p_{j+1}, s)$.

Demostración. Esto es una consecuencia directa del Lema 5.1. □

Corolario 5.2. $cov(p_j, c_i) \geq cov(p_{j+1}, c_i)$.

Demostración. Utilizando el Lema 5.1 se tiene que:

$$\sum_{s \in S_{c_i}} I(p_j, s) \geq \sum_{s \in S_{c_i}} I(p_{j+1}, s),$$

por lo que

$$\left(\frac{1}{|S_{c_i}|}\right) \sum_{s \in S_{c_i}} I(p_j, s) \geq \left(\frac{1}{|S_{c_i}|}\right) \sum_{s \in S_{c_i}} I(p_{j+1}, s).$$

Sustituyendo por la Ec. 34 se concluye que

$$cov(p_j, c_i) \geq cov(p_{j+1}, c_i).$$

□

El Corolario 5.2 muestra que cov es una función monótona decreciente para un conjunto incremental de k-meros sobre una clase c_i.

Lema 5.2. *La relación de orden entre* $q(p_j, c_i)$ *y* $q(p_{j+1}, c_i)$ *es la misma que existe entre las razones* $\frac{\sum_{s \in S_{cm}, m \neq i} I(p_{j+1}, s)}{\sum_{s \in S_{cm}, m \neq i} I(p_j, s)}$ *y* $\frac{\sum_{s \in S_{c_i}} I(p_{j+1}, s)}{\sum_{s \in S_{c_i}} I(p_j, s)}$.

Demostración. Primeramente, se buscará la relación de orden entre $q(p_j, c_i)$ y $q(p_{j+1}, c_i)$. Trivialmente, si $q(p_{j+1}, c_i) = 0$ significa que p_{j+1} no es subcadena de ninguna secuencia en c_i, por lo que $q(p_j, c_i) \geq q(p_{j+1}, c_i)$. Otro caso trivial es cuando $q(p_j, c_m) = 0$, para todas las clases $c_m \neq c_i$. Si $\exists s_x \mid p_j \in s_x \wedge s_x \in S_c$, entonces $q(p_j, c_i) = 1$ y $q(p_j, c_i) \geq q(p_{j+1}, c_i)$.

Ahora suponga el caso no trivial donde $q(p_{j+1}, c_i) \neq 0$ y $q(p_j, c_m) \neq 0$, $m \neq i$. Esto implica que existen al menos un par de secuencias s_x, s_y tales que $(p_{j+1} \in s_x \wedge s_x \in S_{c_i}) \wedge (p_j \in s_y \wedge s_y \in S_{c_m})$. A continuación, con el objetivo de ver cuándo las funciones q y cov se encuentran en conflicto, suponga que se cumple $q(p_j, c_i) < q(p_{j+1}, c_i)$.

Sustituyendo por la Ec. 35:

$$\frac{\sum_{s \in S_{c_i}} I(p_j, s)}{\sum_{s \in S} I(p_j, s)} < \frac{\sum_{s \in S_{c_i}} I(p_{j+1}, s)}{\sum_{s \in S} I(p_{j+1}, s)}.$$

Separando las sumas de los denominadores,

$$\frac{\sum_{s \in S_{c_i}} I(p_j, s)}{\sum_{s \in S_{c_i}} I(p_j, s) + \sum_{s \in S_{cm}, m \neq i} I(p_j, s)} < \frac{\sum_{s \in S_{c_i}} I(p_{j+1}, s)}{\sum_{s \in S_{c_i}} I(p_{j+1}, s) + \sum_{s \in S_{cm}, m \neq i} I(p_{j+1}, s)},$$

haciendo multiplicación cruzada,

$$\sum_{s \in S_{c_i}} I(p_j, s) \left(\sum_{s \in S_{c_i}} I(p_{j+1}, s) + \sum_{s \in S_{cm}, m \neq i} I(p_{j+1}, s) \right) < \sum_{s \in S_{c_i}} I(p_{j+1}, s) \left(\sum_{s \in S_{c_i}} I(p_j, s) + \sum_{s \in S_{cm}, m \neq i} I(p_j, s) \right).$$

Expandiendo los productos en ambos lados de la desigualdad y eliminando el término común resultante,

$$\left(\sum_{s \in S_{c_i}} I(p_j, s) \right) \left(\sum_{s \in S_{cm}, m \neq i} I(p_{j+1}, s) \right) < \left(\sum_{s \in S_{c_i}} I(p_{j+1}, s) \right) \left(\sum_{s \in S_{cm}, m \neq i} I(p_j, s) \right),$$

Reordenando,

$$\frac{\sum_{s \in S_{c_m}, m \neq i} I(p_{j+1}, s)}{\sum_{s \in S_{c_m}, m \neq i} I(p_j, s)} < \frac{\sum_{s \in S_{c_i}} I(p_{j+1}, s)}{\sum_{s \in S_{c_i}} I(p_j, s)}.$$

La demostración es análoga para las relaciones igual y mayor que. □

El Lema 5.2 muestra que q es una función no monótona para un conjunto incremental de k-meros sobre una clase c_i, y esto se debe a que el valor resultante depende también del término independiente a la clase, $\sum_{s \in S_{c_m}, m \neq i} I(p, s)$, por lo que no es posible afirmar que q y cov estén siempre en conflicto.

5.2.2.1. Revisita a los riboswitches

Para mostrar la relación entre los valores QCOV, el poder discriminante y el conjunto no dominado de k-meros, se calcularon las funciones q y cov para el conjunto de entrenamiento utilizado en el Capítulo 3. En las Figuras 20-22, el eje X representa los valores q y el eje Y los valores cov. Cada k-mero se identifica por un color dependiendo de su pertenencia al conjunto HEXCFS, además de que se marcan con una cruz los k-meros que pertenecieron al conjunto no dominado de la familia en cuestión.

En general, puede observarse que la región delimitada por $cov < 0.2$ y $q < 0.2$ tiene una alta densidad de k-meros. En esta region se concentran los k-meros que son poco frecuentes en cada familia. Existen algunas familias que se expanden por todo el cuadrante (Figs. 20 b) y f), 21 c)). Sin embargo, en algunas otras la gráfica aparece como una columna vertical, denotando una baja calidad y distintos valores de cobertura, como por ejemplo las Figs. 21 d), e) y f) y 22 b), c) y d). Estas familias son las que menos instancias tienen en el conjunto de datos (Tabla 1).

En total, 210 k-meros diferentes pertenecieron al conjunto de Pareto de alguna familia. En las figuras puede observarse cómo algunos de estos k-meros no dominados coinciden con los seleccionados en el conjunto HEXCFS. Para detallar lo anterior, se contó el número de k-meros HEXCFS que fueron no dominados para alguna familia, agrupándolos por el número de familias en que esto ocurrió (Figura 23). Se puede observar que solamente 37.2 % de los k-meros HEXCFS fueron encontrados en algún

conjunto no dominado.

Una posible explicación a esta discrepancia entre el conjunto HEXCFS y los *k*-meros no dominados por familia es la redundancia de las características. Para profundizar en esto, la Fig. 24 muestra los *k*-meros no dominados de longitud mayor a 3 para las familias RF00059 y RF00504. Observe por ejemplo los *k*-meros *cugag* y *cugaga* en la Fig. 24 a). Aunque *cugaga* tiene una mayor calidad, *cugag* es una subcadena de la anterior, lo que significa que ambas representan al mismo subconjunto de secuencias puesto que tienen la misma cobertura, con la diferencia de que *cugaga* discrimina mejor entre las demás familias. Lo mismo puede observarse en la Fig. 24 b) con los *k*-meros *ucag* y *ucagg*, así como también con el par *cucag* y *cucagg*. Por lo tanto, un esquema de selección en este espacio no puede basarse sólo en la no dominancia, sino que debería también considerar la redundancia inherente a los *k*-meros.

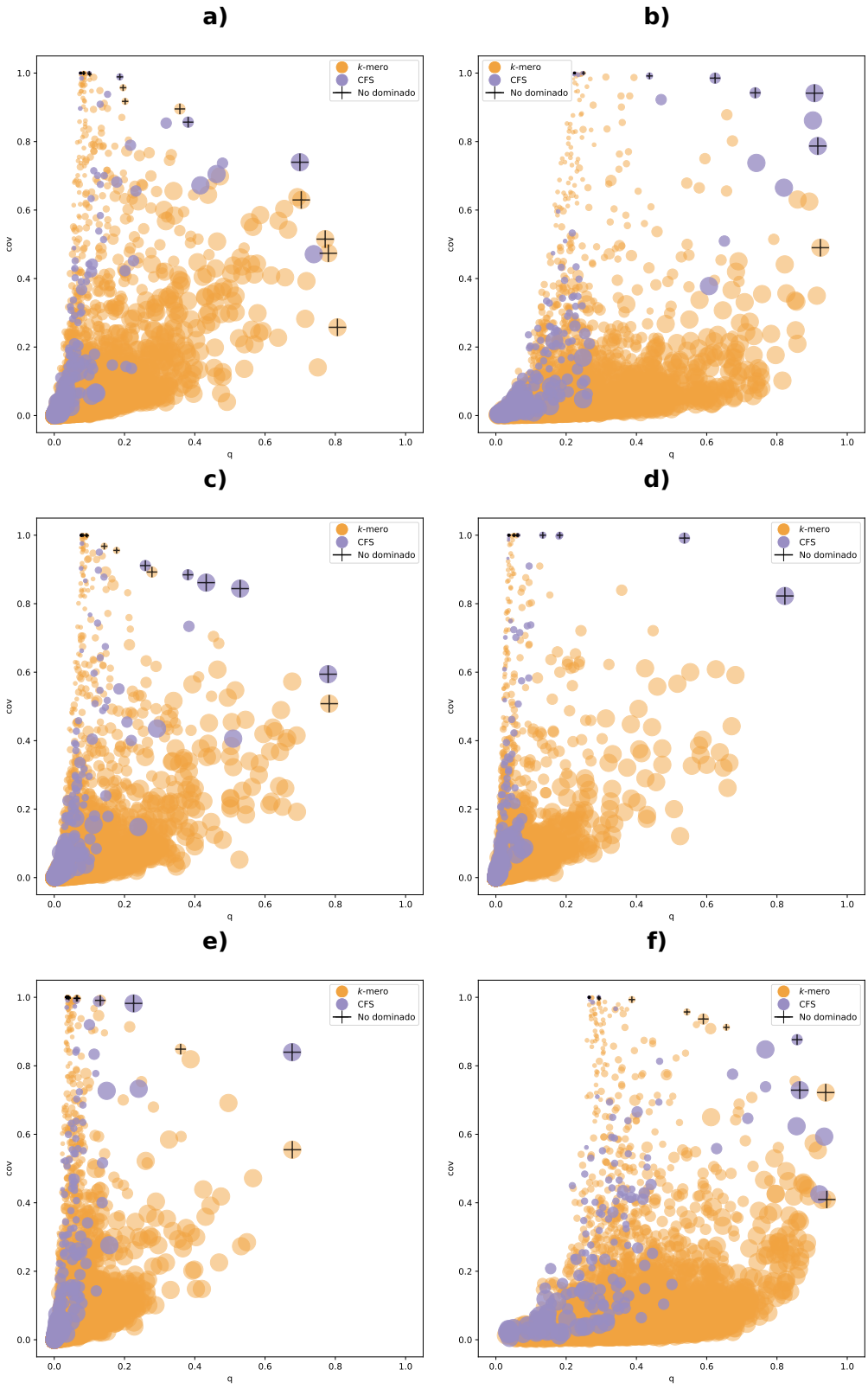

Figura 20: *k*-meros localizados en el espacio QCOV para las familias de riboswitches. a) RF00050; b) RF00059; c) RF00162; d) RF00167; e) RF00168; f) RF00174. El tamaño del punto es proporcional al tamaño del *k*-mero representado. El color violeta indica que el *k*-mero también pertenece al conjunto HEXCFS y la cruz que el *k*-mero pertenece al frente no dominado si se consideran *q* y *cov* como funciones a maximizar.

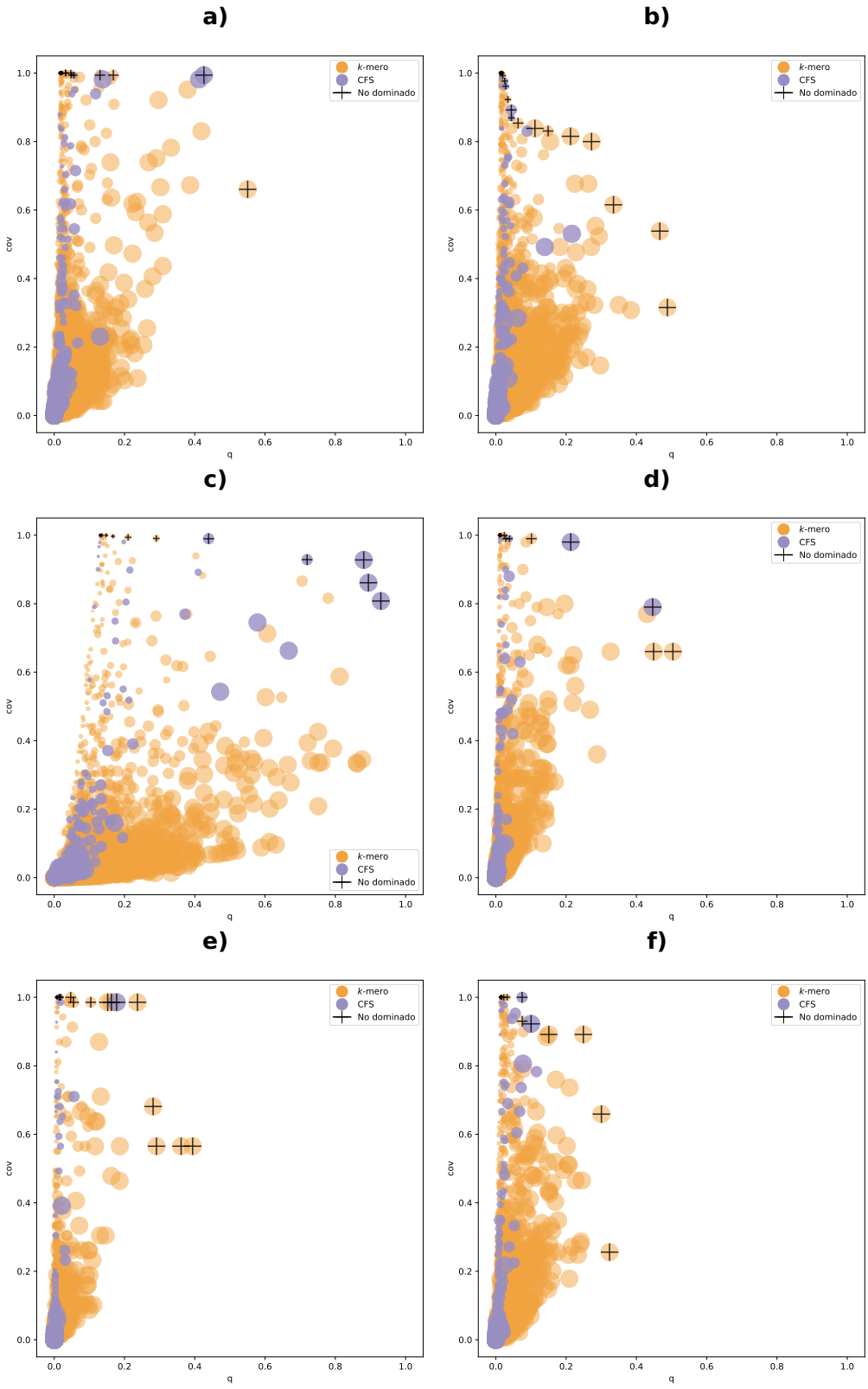

Figura 21: *k*-meros localizados en el espacio QCOV para las familias de riboswitches. a) RF00234; b) RF00380; c) RF00504; d) RF00521; e) RF00522; f) RF00634. El tamaño del punto es proporcional al tamaño del *k*-mero representado. El color violeta indica que el *k*-mero también pertenece al conjunto HEXCFS y la cruz que el *k*-mero pertenece al frente no dominado si se consideran *q* y *cov* como funciones a maximizar.

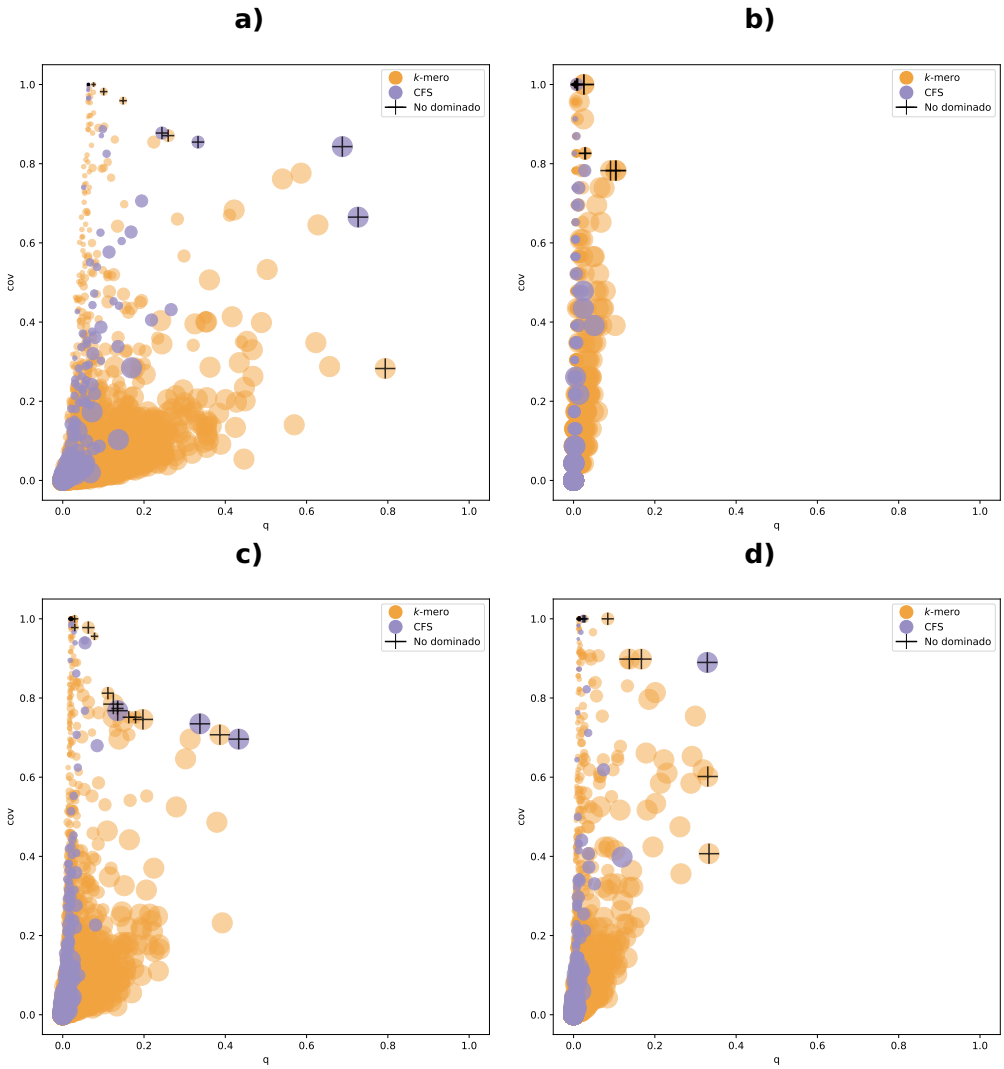

Figura 22: *k*-meros localizados en el espacio QCOV para las familias de riboswitches. a) RF01051; b) RF01054; c) RF01055; d) RF01057. El tamaño del punto es proporcional al tamaño del *k*-mero representado. El color violeta indica que el *k*-mero también pertenece al conjunto HEXCFS y la cruz que el *k*-mero pertenece al frente no dominado si se consideran *q* y *cov* como funciones a maximizar.

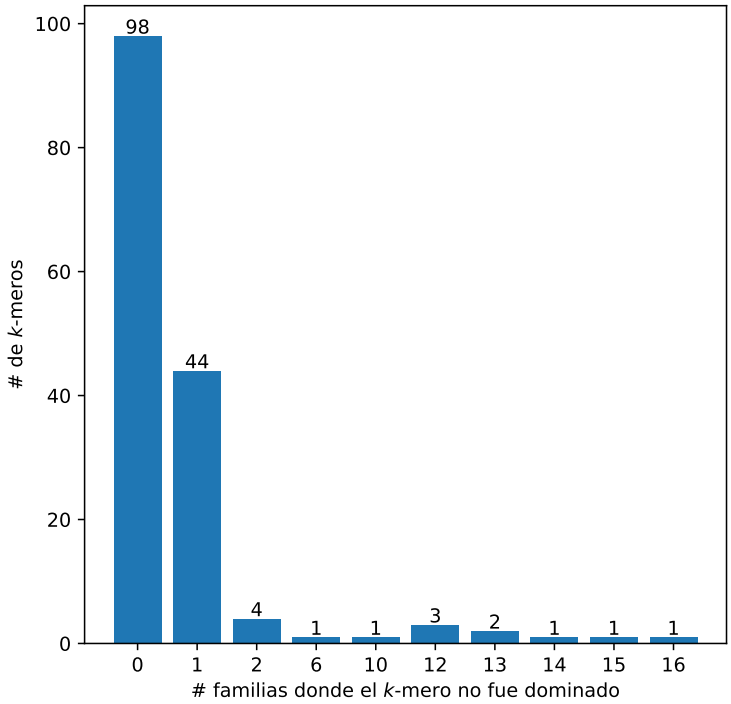

Figura 23: Agrupación de *k*-meros HEXCFS según las veces que pertenecieron a conjuntos no dominados en el espacio QCOV.

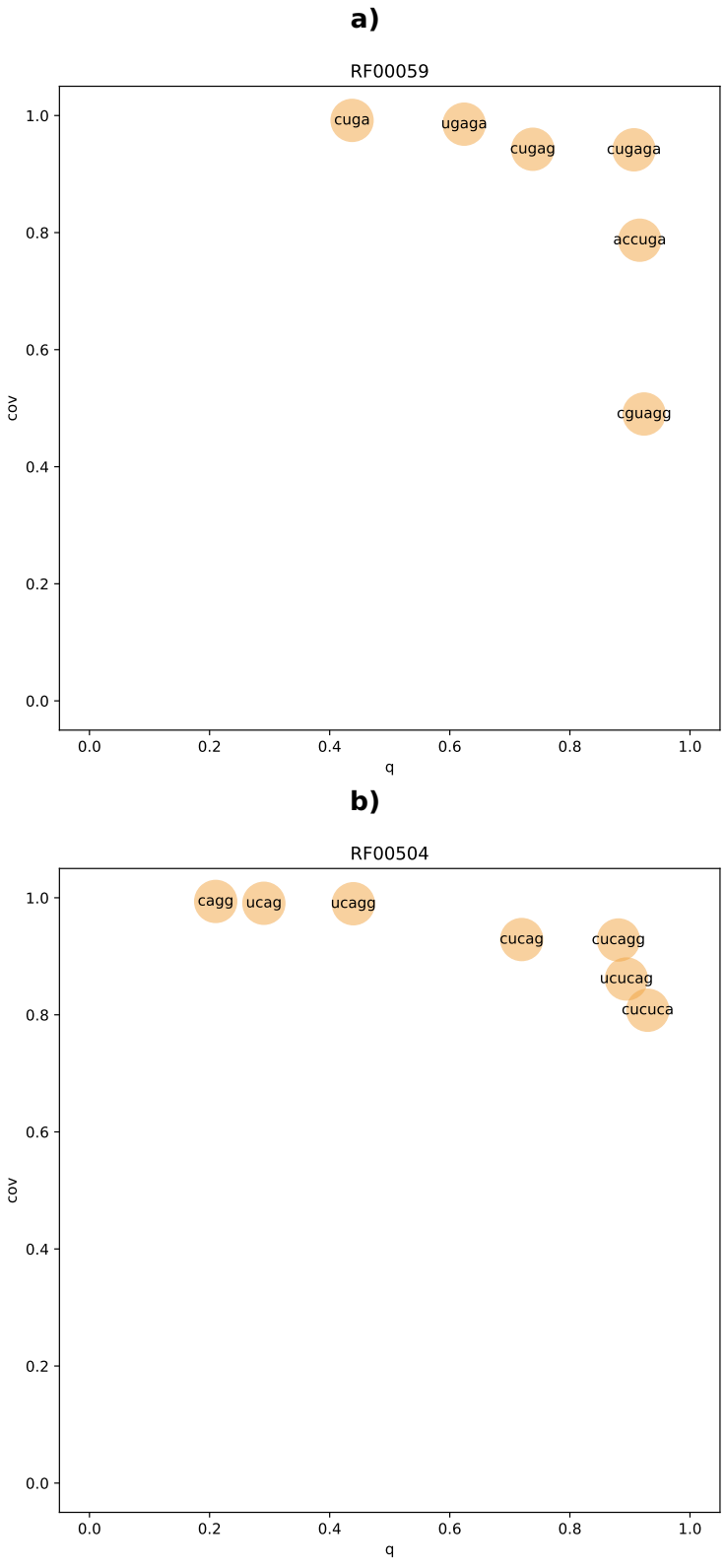

Figura 24: *k*-meros no dominados ($k > 4$) en dos familias de riboswitches. a) RF00059; b) RF00504.

5.3. QCOVGraph

En esta sección se presenta un algoritmo para organizar los k-meros en un bosque de cuatro árboles por cada clase de secuencias llamado `QCOVGraph`. Utilizando esta estructura de datos, se proponen tres métodos de selección de vértices discriminantes, los cuales se prueban como características para clasificar el conjunto de secuencias de riboswitches descrito en la sección anterior.

5.3.1. Construcción

La idea básica del algoritmo es que las bases (1-meros) se convierten en las raíces del bosque correspondiente a la clase c. Después, de manera iterativa, se agregan vértices al grafo de tal manera que el padre del k-mero p es su prefijo de longitud $k-1$, siempre y cuando p esté en al menos una secuencia de S_c y $cov(p, c)$ sea mayor a un umbral t_{cov} determinado por el usuario (Algoritmo 1).

Algoritmo 1: QCOVGRAPH(c, n, t_{cov})

Input : c, una categoría.
n, tamaño máximo de k-mero.
t_{cov}, umbral de tolerancia para el valor de cov.
Output: G_c, un grafo dirigido.

```
1  V ← arreglo vacío
2  E ← arreglo vacío
3  kmers ← arreglo de longitud n
4  kmers[0] ← Σ^1_RNA
5  foreach kmer in kmers[0] do
6  |  V.ADD(CREATEVERTEX(kmer))
7  end
8  for 1 ≤ i < n do
9  |  newkmers ← arreglo vacío
10 |  foreach kmer in kmers[i − 1] do
11 |  |  foreach base in Σ^1_RNA do
12 |  |  |  newkmer ← kmer · base
13 |  |  |  if cov(newkmer, c) > 0  and  cov(newkmer, c) ≥ t_cov then
14 |  |  |  |  v = CREATEVERTEX(newkmer)
15 |  |  |  |  V.ADD(v)
16 |  |  |  |  E.ADD(CREATEEDGE(V.FINDVERTEX(kmer), v)
17 |  |  |  |  newkmers.ADD(newkmer)
18 |  |  |  end
19 |  |  end
20 |  end
21 |  kmers[i] ← newkmers
22 end
23 return G_c = (V, E)
```

Las funciones ADD(*objeto*), CREATEVERTEX(*objeto*), CREATEEDGE($objeto_1, objeto_2$) y FINDVERTEX(*objeto*) se describen a detalle en la Sección 6.3.2. El algoritmo inicia creando una raíz por cada ribonucleótido (líneas 3-7). El arreglo $kmers[i]$ contiene los k-meros de longitud $i+1$ presentes en la clase c. Este arreglo se puebla de manera incremental. Para cada longitud $k \geq 2$, se recuperan los $(k-1)$-meros, es decir, los prefijos (línea 10). Posteriormente, se generan los nuevos k-meros concatenando cada prefijo con cada base (líneas 11-12). Si el nuevo k-mero está presente en al menos una secuencia de S_c ($cov(newkmer, c) > 0$) y si la cobertura es mayor a la especificada por el usuario ($cov(newkmer, c) \geq t_{cov}$), entonces se crea un nodo para representar al k-mero, se crea una arista desde el prefijo hacia el k-mero y se agrega el k-mero a la lista de k-meros de longitud $i+1$ (líneas 13-22). Finalmente, se crea el grafo y se devuelve (línea 23).

El peor caso del Algoritmo 1 sucede cuando las secuencias S_c contienen a todos los k-meros posibles hasta $k = n$, por lo que la complejidad es de $O(4^n)$. En la práctica, se recomiendan valores de n menores a 10.

Observe que en la línea 13 del algoritmo se realiza el filtrado por t_{cov} y $cov > 0$. Sin embargo, aún cuando $t_{cov} = 0$, cov tomará un valor mínimo definido como:

$$cov_{min}(c) = \frac{1}{|S_c|}. \tag{36}$$

En otras palabras, $cov_{min}(c)$ es la cobertura que tendría un motivo específico a una única secuencia de la clase c.

Como se mencionó anteriormente, q es independiente de lo frecuente que es una clase en toda la base de datos. Sin embargo, es posible calcular el valor de q en una clase c que tendría un motivo que está en todas las secuencias:

$$q_{background}(c) = \frac{|S_c|}{|S|}. \tag{37}$$

Los `QCOVGraphs` resultantes para las secuencias de riboswitches se muestran en las Figuras 25-28.

En las figuras la línea punteada horizontal indica el valor de $q_{background}(c)$ y la línea punteada vertical al de $cov_{min}(c)$. Mediante las aristas, es posible observar el compromiso entre la longitud de los k-meros, la cobertura y la calidad. Por ejemplo, en la Figura 25 a), el k-mero `AUUCC` tiene valores de cov y q de 0.87 y 0.38, respectivamente. Este motivo es prefijo de `AUUCCC`. Al incrementar la longitud del motivo, la calidad aumenta hasta 0.77, pero la cobertura decrece a 0.28. En otros casos, puede verse que existen motivos que se hacen más específicos para la familia sin que la cobertura se reduzca, como es el caso de los k-meros `ACGAGG`, `UCGGCG` y `GACGAG` en la Figura 25 c). Observe cómo al relacionarse los k-meros en el grafo, se permite la detección de motivos redundantes. Sin embargo, la simple inspección visual no es suficiente para seleccionar los k-meros más relevantes. Para este fin, se proponen tres esquemas de

selección, los cuales serán descritos en la siguiente sección.

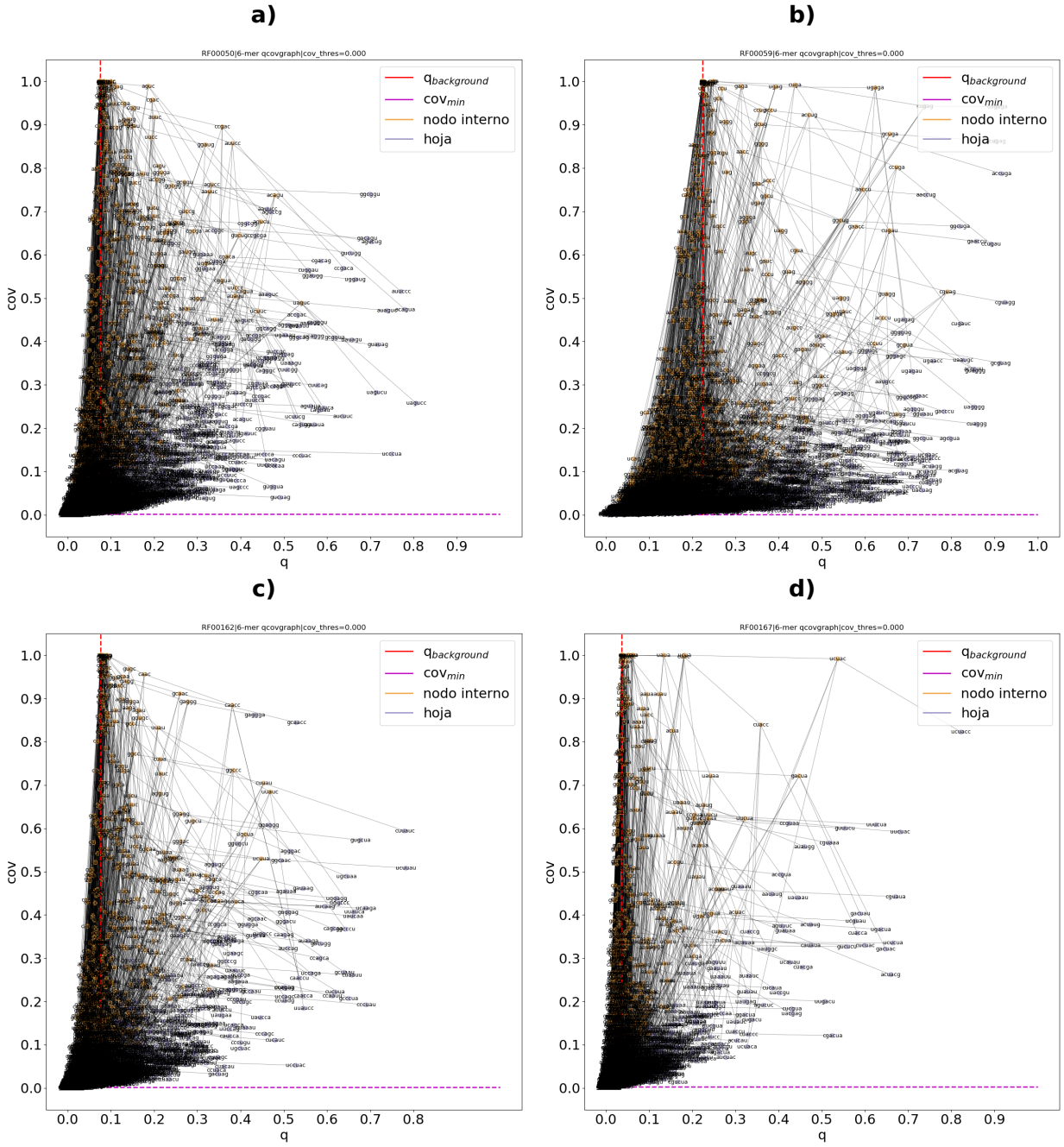

Figura 25: QCOVGraphs obtenidos para distintas familias de riboswitches. a) RF00050; b) RF00059; c) RF00162; d) RF00167.

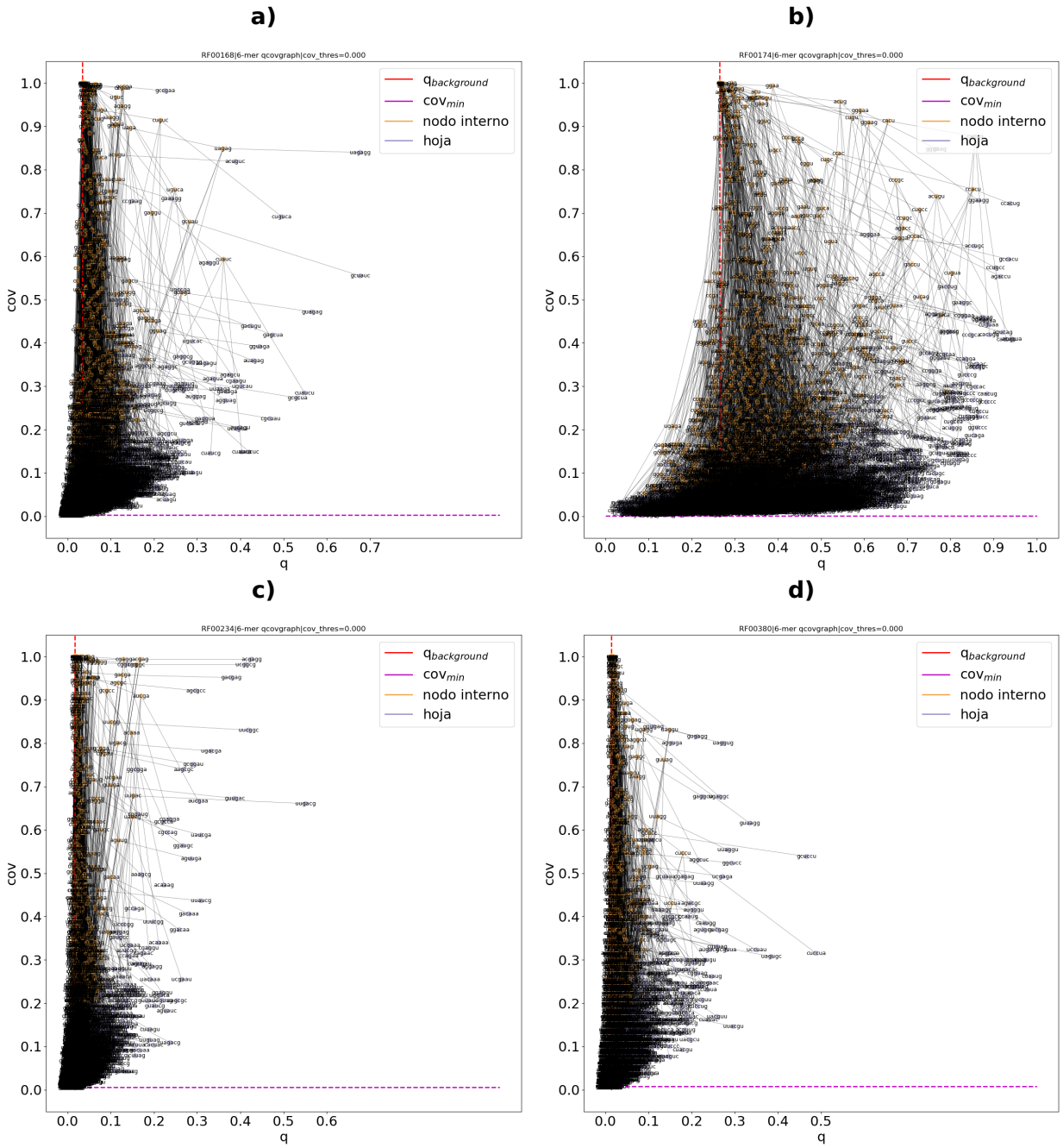

Figura 26: QCOVGraphs obtenidos para distintas familias de riboswitches. a) RF00168; b) RF00174; c) RF00234; d) RF00380.

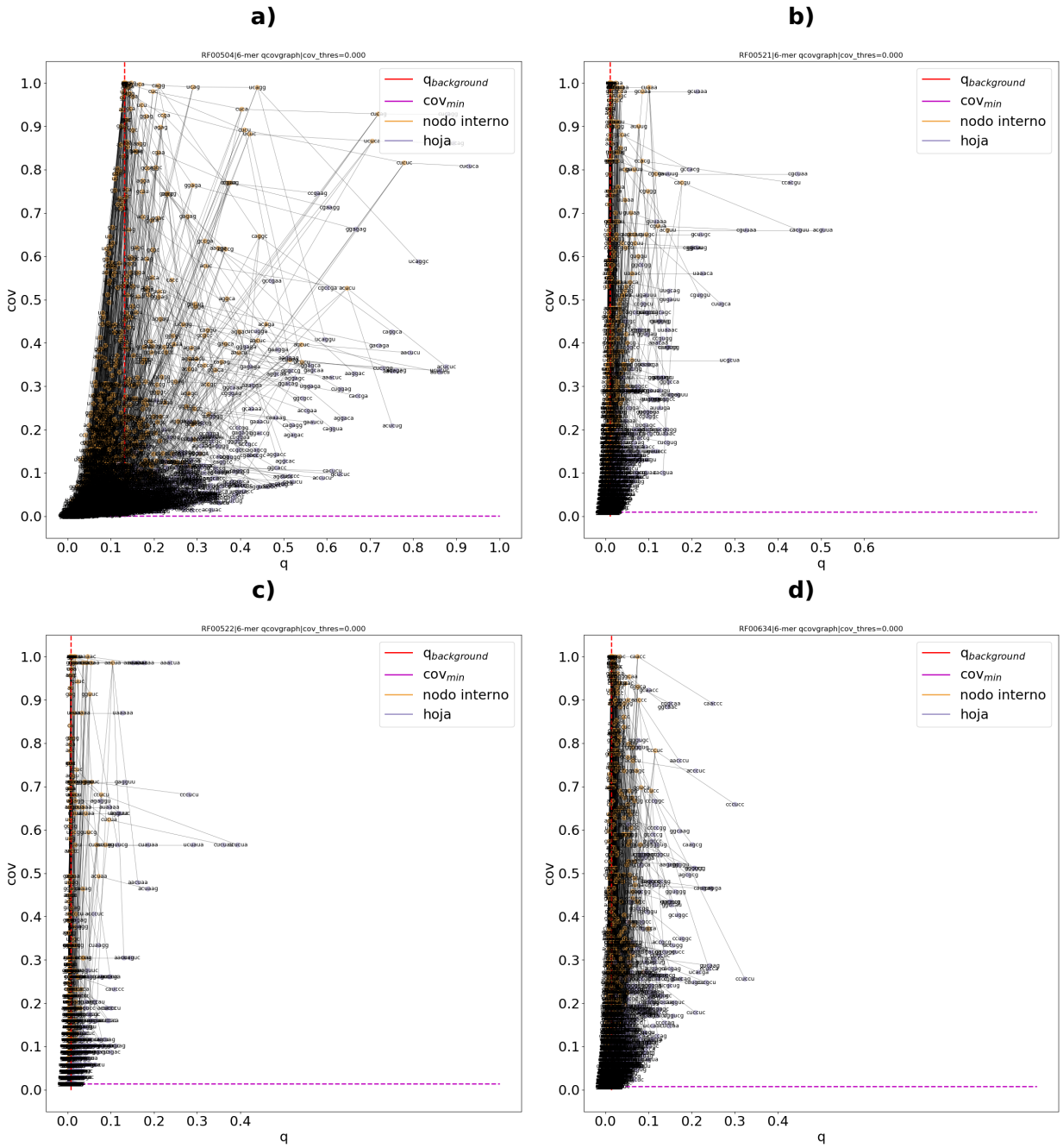

Figura 27: QCOVGraphs obtenidos para distintas familias de riboswitches. a) RF00504; b) RF00521; c) RF00522; d) RF00634.

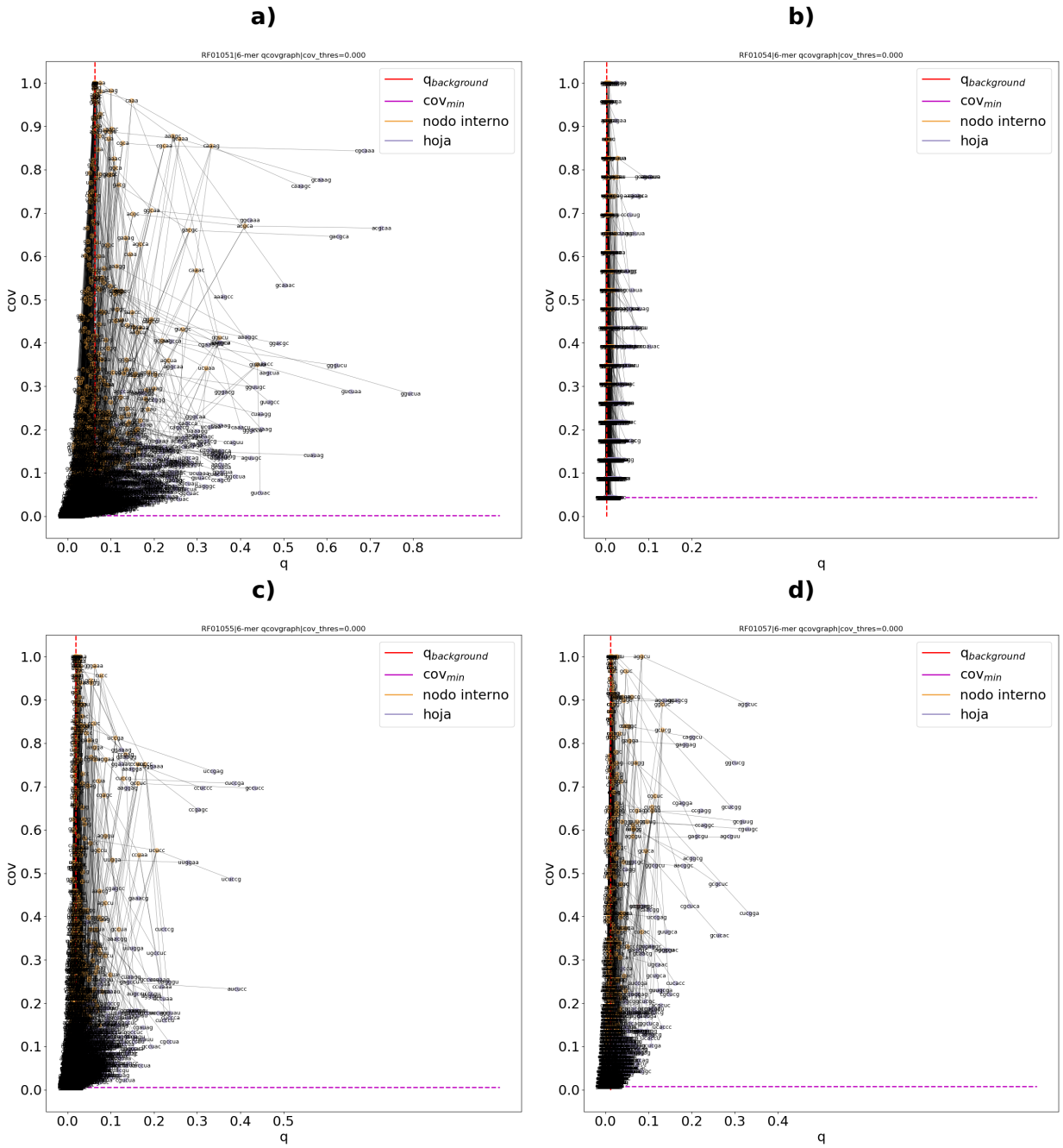

Figura 28: QCOVGraphs obtenidos para distintas familias de riboswitches. a) RF01051; b) RF01054; c) RF01055; d) RF01057.

5.3.2. Selección de motivos

El primer esquema de selección propuesto, M_1, consiste en recuperar todos los k-meros de los QCOVGraphs generados en ciertos valores de n y t_{cov} especificados. La estrategia se muestra en el Algoritmo 2.

Algoritmo 2: SELECCIÓNM1(G)

Input : G, el conjunto de QCOVGraphs de S.
Output: M_1, un conjunto de k-meros.
1 **return** $M_1 = \bigcup_{c \in C} \{v : v \in G_c.V\}$

El algoritmo recorre los QCOVGraphs de todas las clases, recuperando los k-meros distintos de longitud máxima n cuya cobertura supere el umbral t_{cov} (línea 1). Este algoritmo presenta dos desventajas: la primera, es que el número de características es muy grande debido a que no hay otro filtrado además del umbral t_{cov}. La segunda, es que no se considera la redundancia entre los k-meros.

Una manera de seleccionar motivos que no sean redundantes, es considerar solamente las hojas de los QCOVGraphs. Si bien pueden tener prefijos en común, ninguna hoja es un prefijo de alguna otra. La estrategia M_2 implementa esta idea y se muestra en el Algoritmo 3.

Algoritmo 3: SELECCIÓNM2(G)

Input : G, el conjunto de QCOVGraphs de S.
Output: M_2, un conjunto de k-meros.
1 **return** $M_2 = \bigcup_{c \in C} \{v : v \in G_c.V \wedge \text{OUTDEGREE}(v) = 0\}$

El algoritmo recorre los QCOVGraphs de todas las clases, recuperando los k-meros distintos de longitud máxima n cuya cobertura supera el umbral t_{cov} y además sean hojas, es decir, que no tengan ninguna arista de salida (línea 1).

Una de las opciones para filtrar aún más el conjunto M_2 consiste en considerar únicamente las hojas con una calidad mayor a la calidad del fondo ($q(p, c) \geq q_{background}(c)$). La lógica detrás de esta estrategia es que si se supera la calidad del fondo, entonces la representación del motivo es mayor que la representación de la clase en todo el conjunto de secuencias, lo que lo convierte en potencialmente discriminante para la clase. Esta estrategia es llamada M_3 y se presenta en el Algoritmo 4.

Algoritmo 4: SELECCIÓNM3(G)

Input : G, el conjunto de QCOVGraphs de S.
Output: M_3, un conjunto de k-meros.
1 **return** $M_3 \leftarrow \bigcup_{c \in C} \{v : v \in G_c.V \wedge \text{OUTDEGREE}(v) = 0 \wedge q(v,c) \geq q_{background}(c)\}$

El algoritmo recorre los QCOVGraphs de todas las clases, recuperando los k-meros distintos de longitud máxima n cuya cobertura supere el umbral t_{cov}, sean hojas, y su calidad sea mayor a la del fondo (línea 1). La región que contiene a las hojas que cumplen con estas condiciones en el espacio QCOV se muestra en la Figura 29.

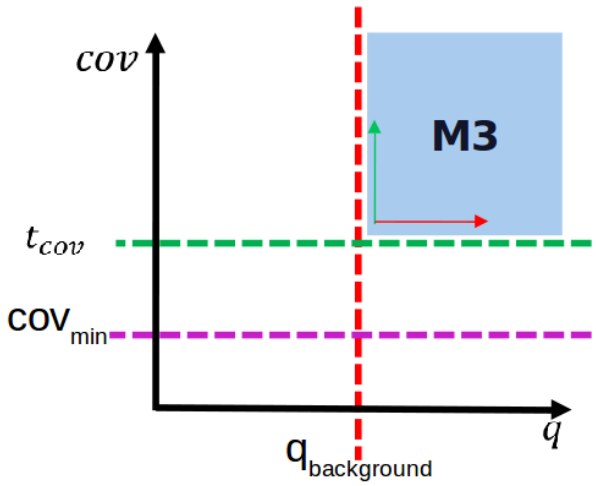

Figura 29: Esquema de la región M_3.

Con la finalidad de mostrar la reducción del número de k-meros para cada valor de t_{cov} en cada una de las estrategias propuestas, se generaron los QCOVGraphs para el conjunto de riboswitches tomando valores de t_{cov} del 0 al 1 espaciados por una distancia de 0.05 (Figura 30). En ambos paneles de la figura se indica con una línea punteada el valor correspondiente para el conjunto de k-meros no dominados. En la Figura 30 a) puede observarse cómo para valores pequeños de t_{cov} la diferencia entre el número de motivos es de más de 1000. Sin embargo, esta diferencia se acorta a medida que se incrementa el umbral, al punto en que es mínima. Por ejemplo, en $t_{cov} = 0.85$ la diferencia de M_1 a M_2 es de 60 motivos, la diferencia de M_1 a M_3 es de 63 y la diferencia de M_2 a M_3 es de 3. En $t_{cov} = 1.00$ el conjunto M_2 es igual al M_3 y ambos difieren de M_1 en 16 motivos.

Para estimar qué tan buenos serían los motivos seleccionados, se calculó el porcentaje de k-meros HEXCFS contenidos en cada conjunto seleccionado (Figura 30 b)). El comportamiento de la curva de M_1 es esperado debido a que el podado de nodos se hace progresivamente sobre el valor de t_{cov}; al inicio, todos los elementos de HEXCFS están contenidos y van decreciendo a medida que aumenta el umbral. No obstante, esto no ocurre con M_2 y M_3 porque ambas estrategias filtran las hojas del árbol podado. Dicho de otra manera, si se elimina una hoja al podar según el umbral t_{cov}, entonces su antecesor que era un nodo interno se transforma en una hoja del nuevo grafo. De esta forma, se descubren nuevos motivos que no se habían considerado en umbrales menores, y se explica el incremento en porcentaje de elementos de HEXCFS contenidos. Este porcentaje es mayor que el conjunto no dominado en $t_{cov} \leq 0.85$ para M_1, $0.05 \leq t_{cov} \leq 0.75$ para M_2 y $0.05 \leq t_{cov} \leq 0.70$ para M_3.

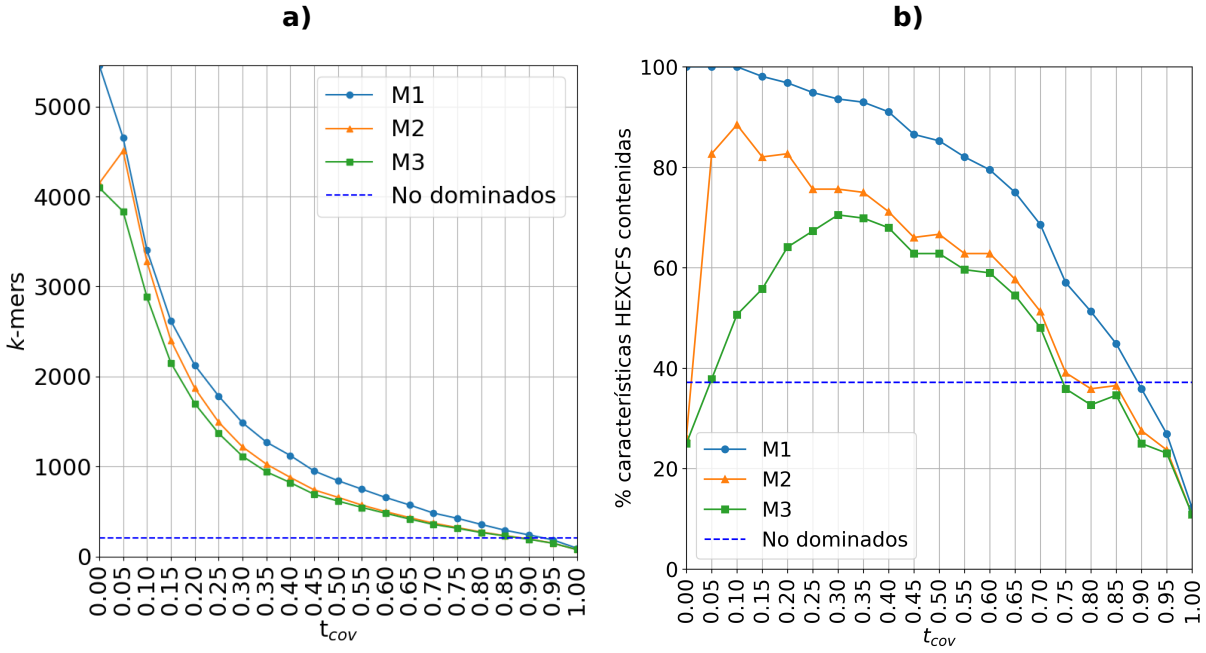

Figura 30: Resultado de las estrategias M_1, M_2 y M_3 en las secuencias de riboswitches. a) Número de k-meros seleccionados para cada valor de t_{cov}; b) Porcentaje del número de características HEXCFS contenidas en cada conjunto M_1, M_2 y M_3 para cada valor de t_{cov}.

5.4. Conclusiones

En este capítulo se propone el framework QCOV, el cual permite distinguir frecuencias k-méricas en un conjunto de clases según su cobertura y "calidad". Estas frecuencias son características que presentan alta redundancia, lo cual está reportado

en modelos basados en *n*-gramas (Brown *et al.*, 1992). A diferencia de un lenguaje como el español o el inglés, el tomar solamente la raíz en común de un grupo de palabras no tiene la misma utilidad cuando éstas son derivadas de secuencias biológicas, puesto que en los motivos las regiones variables no están necesariamente al inicio o al final de las cadenas. Por otro lado, se mostró que un enfoque multi-objetivo no es tan apto para seleccionar *k*-meros bajo estas funciones, por lo que fue necesario recurrir a otras estrategias. Considerando lo anterior, creemos que los métodos de selección basados en el `QCOVGraph` tienen el potencial para encontrar motivos discriminantes en secuencias biológicas, tal como se mostró con el acuerdo que se logró con las características HEXCFS.

Capítulo 6. Abstracciones basadas en grafos para la estructura secundaria del ARN

6.1. Introducción

Una secuencia de ARN conservada es aquella que es altamente similar entre varias especies. Un ejemplo de ARN altamente conservado son los miRNA y los snoRNA (Pang *et al.*, 2006). Sin embargo, la gran mayoría de los genes de ARN no se conservan a nivel primario, sino a nivel secundario y terciario (Menzel *et al.*, 2009). Los predictores *de novo* explotan esta posibilidad al predecir la estructura secundaria a través de un modelo de energía libre, generalmente realizando un escaneo por ventanas en genomas completos (Gorodkin y Hofacker, 2011b). La utilidad de usar solamente la energía libre resultante de la estructura como criterio determinante en la predicción se encuentra todavía en debate. Por un lado, Rivas y Eddy (2000) mostraron que las estructuras predichas en ciertos ncRNA no son significativamente más estables que secuencias aleatorias generadas a partir del trasfondo genómico, mientras que Clote *et al.* (2005) afirmaron que el ARN estructural tiene menor energía en el plegado que secuencias aleatorias con la misma frecuencia de dinucleótidos.

En general, la búsqueda y predicción de estructura secundaria y terciaria tiene un alto costo computacional (Manzourolajdad y Arnold, 2015). Además, los métodos de predicción varían tanto en resultados que ha sido necesario establecer métodos para comparar y evaluar la validez de las predicciones (Allali *et al.*, 2012). Más aún, el comparar las predicciones de estructura contra algún *estándar de oro* resulta problemático debido a que dichas estructuras verificadas pueden no ser representativas de los ncRNA que todavía no han sido descubiertos o caracterizados, por lo que una posible vía de acción consiste en integrar al modelo información de distintas fuentes que puedan generar una estructura secundaria consenso (Gorodkin *et al.*, 2010).

Incluso cuando las secuencias están conservadas a nivel primario, las homologías de nucleótidos no resultan tan fáciles de detectar como las homologías en proteínas, lo cual limita el poder de los enfoques basados en evidencia, haciendo necesario el uso

de modelos de clasificación (Yuan y Sun, 2013). Usualmente, los enfoques de clasificación de moléculas de ncRNA incluyen en su flujo de trabajo la extracción de características basadas en la estructura secundaria (y/o terciaria) con la finalidad de compensar la falta de conservación en la estructura primaria (Karklin *et al.*, 2005; Childs *et al.*, 2009; Gudyś *et al.*, 2013; Panwar *et al.*, 2014). En los Capítulos 3 y 4 se mostró la importancia de extraer características significativas en el conjunto de secuencias a clasificar, por lo que, aunado a lo anterior, podemos afirmar que es crítico elegir una representación de la estructura secundaria que sea adecuada para la extracción de características.

El resto del capítulo se organiza de la manera siguiente: primero, se presenta un panorama general de las tareas del procesamiento de secuencias de ncRNA que involucran representaciones alternativas de la estructura secundaria. Segundo, se presenta el framework NR-PR, una abstracción generalizada que permite modelar otras representaciones basadas en grafos. Finalmente, se abordan las posibles aplicaciones de una de las representaciones propuestas sobre un conjunto de secuencias de riboswitches.

6.2. Procesamiento de secuencias de ncRNA

La necesidad de contar con una representación alternativa de la estructura secundaria del ARN se presenta en diversos problemas relacionados al procesamiento de secuencias de ncRNA, no solo en la clasificación. La Figura 31 muestra un panorama general del flujo de trabajo en estos problemas. A continuación se detalla en qué consiste cada parte principal del diagrama.

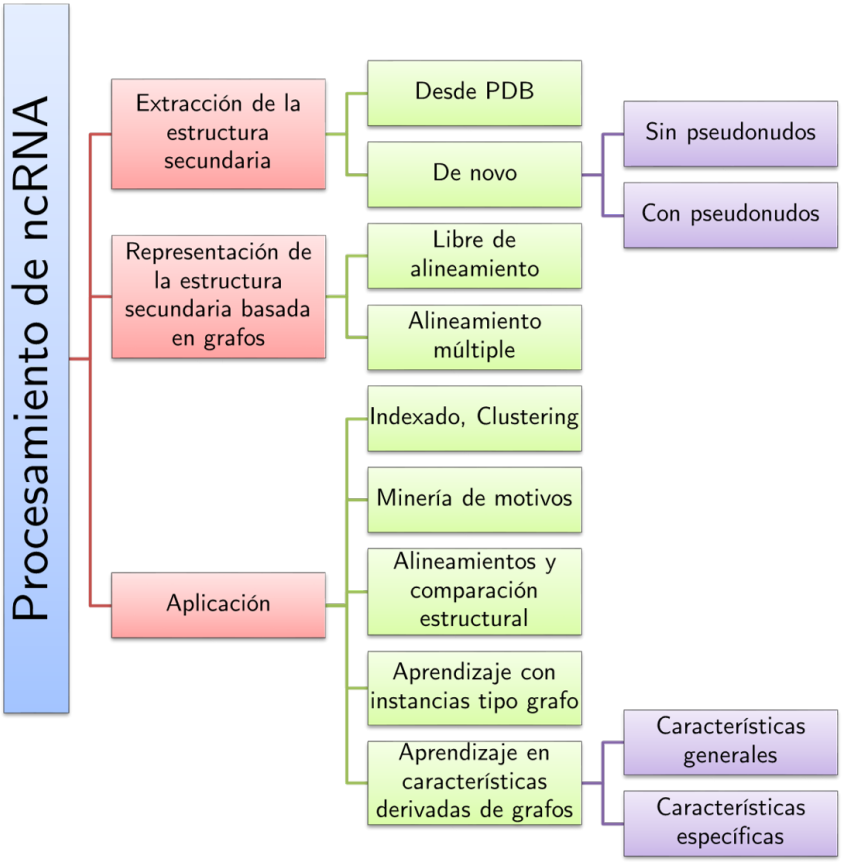

Figura 31: Ejemplos de pasos en el procesamiento de secuencias de ncRNA y la importancia de las representaciones alternativas basadas en grafos para la estructura secundaria del ARN.

6.2.1. Extracción de la estructura secundaria

El paso inicial en el flujo de trabajo que involucra la estructura secundaria del ARN es su extracción de alguna base de datos o la predicción *de novo*.

En caso de contar con la estructura real anotada en PDB (Sussman *et al.*, 1998), RNApdbee (Zok *et al.*, 2018) es una herramienta que permite obtener la estructura secundaria a partir de dicha anotación. Sin embargo, el proceso de predecir estructuras *de novo* no es tan sencillo debido a que es necesario tomar ciertas decisiones. Por ejemplo, si se desea incluir la predicción de pseudonudos o no, si se requiere sólo la mejor estructura o un conjunto de las mejores estructuras o si se desea visualizar el ensamble termodinámico completo.

El paquete Vienna RNA (Lorenz *et al.*, 2011) es un software ampliamente utilizado para el plegado y análisis de estructuras secundarias. Las implementaciones que contiene para la predicción de la estructura son RNAfold, RNAsubopt, RNALfold y RNA2Dfold. RNAfold calcula la energía mínima libre (MFE por sus siglas en inglés) y encuentra la estructura óptima por backtracking. También es capaz de calcular la función de partición, la matriz de probabilidades de apareamiento entre bases y la estructura centroide utilizando el algoritmo de McCaskill (McCaskill, 1990a). RNAsubopt en general produce estructuras subóptimas. Pueden ser estructuras que caen dentro de cierta banda de energía, una muestra aleatoria generada con backtracking estocástico (como sfold (Ding y Lawrence, 2003)) o un conjunto de estructuras tal que por cada par de bases que pudiera formarse contiene la estructura más favorable energéticamente (algoritmo de Zucker (Zuker, 1989), usado también en aplicaciones de mfold (Zuker, 2003)). RNALfold escanea secuencias largas de ARN y calcula subestructuras locales estables mediante ventanas. Por último, la librería RFA2Dfold implementa minimización de la energía, cómputo de la función de partición y *backtracking* estocástico para la proyección en dos dimensiones del espacio de estructuras secundarias definido por las distancias entre pares de bases de dos estructuras de referencia que son dadas como entrada.

RNAPKplex también está contenido en el paquete Vienna y es un programa capaz de predecir estructuras secundarias con pseudonudos. Pknots (Rivas y Eddy, 1999) es el algoritmo de programación dinámica más general para resolver esta tarea, aunque con una complejidad de $\Theta(n^6)$. Otra herramienta popular para predecir estructuras secundarias con pseudonudos es IPknot (Sato *et al.*, 2011), la cual usa programación entera para maximizar el valor esperado de la precisión de la estructura predicha. Knotty (Jabbari *et al.*, 2018) es una herramienta basada en CCJ (Chen *et al.*, 2009) que busca un compromiso entre las clases de pseudonudos detectadas y la complejidad en tiempo y espacio.

6.2.2. Representación de la estructura secundaria basada en grafos

Una vez calculadas las estructuras secundarias, el siguiente paso es decidir si se trabajará una representación para cada secuencia individual (libre de alineamiento), o

si va a estar basada en el consenso de las secuencias o estructuras.

Un ejemplo clásico de representación de grafo basada en alineamiento es la que mostraron Cary y Stormo (1995), donde el alineamiento se transforma a un grafo donde las aristas representan el valor de información mútua a nivel de pares de bases.

En general, los métodos libres de alineamiento para análisis de estructuras de ARN se basan en frecuencias *k*-méricas, modelos de Markov y representaciones alternas (Schwende y Pham, 2013). Por ejemplo, St-Onge *et al.* (2007) proponen modelar motivos estructurales a nivel terciario utilizando gramáticas de grafos (*graph-grammars*). La estructura secundaria se representa como un grafo, pero las aristas están etiquetadas según el tipo de interacción entre los átomos. El enfoque se prueba realizando un alineamiento con 800 ARNs ribosomales 23S bacteriales sobre el sitio SRL (*sarcin-ricin loop*). Por otro lado, GraphClust (Heyne *et al.*, 2012) es un método de comparación y *clustering* a nivel de estructura libre de alineamiento el cual utiliza una representación de grafos basada en formas abstractas (Janssen *et al.*, 2008).

6.2.3. Aplicaciones

Indexado y *clustering*

Una de las primeras representaciones basadas en grafos son los *dual graphs*, propuesta para integrar la base de datos RAG (RNA-As-Graphs database) (Gan *et al.*, 1987), la cual se ha usado en aplicaciones que van desde la construcción de índices para la búsqueda de escructuras (y suponer que si una estructura no está registrada por ellos entonces es un nuevo tipo de molécula (Gan *et al.*, 2003)), hasta su extensión para modelar estructuras tridimensionales (Zahran *et al.*, 2015). RNAshapes (Steffen *et al.*, 2005) utiliza una abstracción llamada *Abstract Shapes*, la cual reduce elementos consecutivos del mismo tipo en cadenas *dot-bracket*. Los autores definieron cinco niveles de abstracción, siendo el quinto el más reducido. Para probar su enfoque, compararon sus representaciones con las estructuras consenso de RFAM, y entre otras cosas, concluyen que es necesario considerar la energía libre para poder diferenciar las estructuras a nivel de representación, además, encontraron que a menudo la es-

tructura consenso de RFAM no representa una forma compartida. Este trabajo derivó en otras aplicaciones como RNAsifter (RNA shape index filter (Janssen *et al.*, 2008)), el cual funciona como un filtro para las búsquedas en RFAM basado en el indexado de las formas abstractas. Las formas abstractas que se comparan para realizar la búsqueda son representantes de todo el espacio de posibles estructuras secundarias de la base de datos. Bakhtin y Heitsch (2009) diseñaron una representación de la estructura secundaria basada en árboles planos llamada *plane tree*, la cual trata de modelar la configuración y preservar información sobre el arreglo básico de *loops* y *stems* convirtiéndolos en vértices y aristas, respectivamente. En su trabajo, aplican el principio de desviaciones grandes (*large deviation principle*) a los grados de vértices de los árboles planos y a estructuras aleatorias. Chiu y Chen (2012) utilizan una abstracción similar a las *Abstract Shapes* para analizar topológicamente la base de datos RNA STRAND (Andronescu *et al.*, 2008).

Existen diversos enfoques para *clustering* basado en representaciones alternas de estructuras secundarias. Saito *et al.* (2011) usan ensambles de alineamientos a nivel de secuencia y aproximaciones al alineamiento estructural mediante una nueva medida de similitud entre las estructuras. Por otro lado, Zhong y Zhang (2012) utilizan *clustering* y la representación de Bon *et al.* (2008) para identificar motivos estructurales *de novo*. Wang *et al.* (2013) clasificaron ncRNA como funcional o no utilizando *k*-medias sobre una representación llamada grafo semántico-estructural. Para esto, la cadena de bases sin alinear incluida en un *loop* se convierte a un vértice, y un *stem* se convierte a una arista. La similitud se basa en la trayectoria común más larga entre un par de grafos y el contenido de la secuencia. BlockClust (Videm *et al.*, 2014) realiza *clustering* estructural con kernels de grafos aplicado directamente a transcritos cortos provenientes de RNA-seq. Para lograrlo, proponen una representación propuesta basada en los perfiles de expresión. GraphClust (Heyne *et al.*, 2012; Miladi *et al.*, 2017) utilizan grafos basados en *Abstract Shapes* para comparar y hacer *clustering* de moléculas de ARN utilizando conjuntamente información de secuencia y estructura. Wiedemann (2014) añade una capa de interpretabilidad a GraphClust para visualizar y analizar los resultados del clustering.

Minería de motivos estructurales

Anteriormente mencionamos cómo los trabajos de Gan *et al.* (1987) y Saito *et al.* (2011) enumeran estructuras con la finalidad de indexar o realizar clustering. En esta sección se discutirán otros cuyo enfoque principal es el enumerar y minar las (sub)estructuras posibles en un conjunto de secuencias.

Hamada *et al.* (2006) presentaron RNAmine, un método que emplea representación de estructura secundaria de ARN mediante grafos para enumerar exhaustivamente los motivos posibles de un conjunto de secuencias utilizando un algoritmo de minería de grafos, con el objetivo de minar patrones en los *stems* sin realizar alineamientos. RSSVM (*RNA Sampler* + *Support Vector Machine*) (Xu *et al.*, 2009) utiliza una SVM para identificar eficientemente motivos funcionales en estructuras secundarias de ARN aleatorias. Gawronski y Turcotte (2014) proponen un grafo dual dirigido extendido para modelar estructuras de ARN y sus interacciones y el uso de la minería de subgrafos frequentes (*Frequent subgraph mining*) para extraer patrones significativamente más frecuentes que en grafos aleatorios. Su hipótesis es que los patrones obtenidos representan mecanismos biológicos.

Algunos trabajos de este tipo derivaron en bases de datos de motivos. Por ejemplo, RmotifDB (Wen y Wang, 2009) es una base de datos de motivos estructurales de ARN que se basa en una representación derivada de un árbol ordenado de etiquetas y además presentan otra representación basada en un modelo de *loops*. Los motivos se obtuvieron de tres fuentes: 1) manualmente de literatura biomédica; 2) enviados por científicos de todo el mundo; 3) descubiertos por una amplia variedad de métodos de minería de motivos. Por otro lado, RMfam (Gardner y Eldai, 2014) es una base de datos basada en RFAM donde los motivos se minan con métodos bioinformáticos.

Alineamientos y comparación estructural

En ocasiones el objetivo principal de presentar una representación alternativa consiste en idear también cómo comparar de manera coherente dos estructuras. Benedetti y Morosetti (1996) utilizan un enfoque basado en topologías de grafos para esta

tarea. Los autores utilizan un grafo donde los *loops* son vértices y los *stems* aristas. Esta representación se basa en la idea básica de que si dos estructuras son similares, entonces es de esperarse que tengan la misma apariencia de ramificación (en otras palabras, tengan la misma relación topológica).

MiGaL (*Multiple Graph Layers*) (Allali y Sagot, 2005) es una estructura de datos compuesta de varios grafos unidos por relaciones de abstracción/refinamiento. Los autores proponen el algoritmo para comparar dos MiGaLs, y lo aplican en estructuras de ARN. Liao *et al.* (2006) presentan una representación gráfica de dos dimensiones (2DGRR) usando un sistema de dos coordenadas cartesianas que han sido derivadas de la notación matemática de la estructura secundaria. Los autores argumentan que su representación resuelve el problema de la degeneración de la estructura y evita la pérdida de información. ESA (*Elastic Shape Analysis*) (Laborde *et al.*, 2011) toma a las estructuras de ARN como curvas 3D y realiza alineamiento flexible entre dos moléculas de ARN por medio de doblar y estirar una de las moléculas para hacer coincidir una con la otra. La cantidad de estiramiento y doblamiento se cuantifica por la medida de distancia geodésica. Basándose en ESA, es posible construir un framework para comparar estructuras de ARN y los autores lo explotan para la predicción de funciones de moléculas de ARN en conjuntos de datos de prueba. Zhong *et al.* (2010) presentan la herramienta RNAMotifScan, la cual implementa la representación geométrica propuesta por Leontis y Westhof (2001) (que considera 12 tipos diferentes de enlaces), así como un método de alineamiento estructural para la identificación de motivos.

Por otro lado, otros autores aprovechan la representación directa de una estructura de árbol y aplican algoritmos de comparación entre estas estructuras de datos a problemas de ARN. Churkin *et al.* (2012) argumentan que las estructuras secundarias de ARN pueden representarse por *coarse grain tree-graphs*, por lo que pueden utilizarse índices topológicos significativos para distinguir entre varias estructuras, aunque RNAs pequeños necesitarían ser representados por grafos completos. Dado que no se ha sugerido ningún índice topológico para lo anterior, proponen uno basado en cortes elementales: el índice Szeged (Khadikar *et al.*, 1995). En la misma dirección, Ouangraoua y Ferraro (2009) presentan un algoritmo de programación dinámica para comparar dos árboles ordenados usando una distancia de edición restringida, mientras que

el presentado por Zhong y Zhang (2013) utiliza programación dinámica *sparse*. Milo *et al.* (2012) presentan un algoritmo que generaliza los algoritmos previos de comparación de ARN utilizando árboles, es decir, sin importar si el alineamiento es local, global, si el problema es asimétrico o si el árbol es enraizado u ordenado. Bon *et al.* (2008) atacan el problema de comparar estructuras con pseudonudos basados en el genus topológico del diagrama circular asociado con la estructura de bases pareadas de ARN. El genus es un entero positivo cuyo valor cuantifica la complejidad topológica de la estructura plegada de ARN.

Aprendizaje con instancias tipo grafo

Algunos autores utilizan los grafos directamente para inducir clasificadores. Por ejemplo, Sato *et al.* (2008) presentan una extensión a kernels de cadenas, llamados *stem kernels*, con la finalidad de medir la similitud entre dos secuencias de ARN desde el punto de vista de las estructuras secundarias. Los kernels se derivan de la construcción de grafos acíclicos dirigidos basados en la matriz de probabilidad de apareamiento de bases. Sin embargo, los autores admiten que aplicar dichos *stem kernels* directamente a conjuntos grandes de ncRNA es impráctico debido a la alta complejidad computacional del método. Mahé y Vert (2009) desarrollaron una familia de kernels definidos para grafos basados en la detección de subárboles comunes. Los autores probaron los kernels en problemas de toxicidad y predicción de actividad anti-cáncer para pequeñas moléculas con SVMs. Por otro lado, Uhl (2014) se enfoca en microRNA para modelar las interacciones entre objetivos de microRNA como grafo, codificando también información de las secuencias en el mismo. En su trabajo modifican kernels para grafos existentes y proponen uno nuevo adaptado a la representación que proponen.

Aprendizaje en características derivadas de grafos

Anteriormente se mencionaron algunos ejemplos de aprendizaje no supervisado utilizando las representaciones alternativas de la estructura secundaria. Respecto al aprendizaje supervisado, Childs *et al.* (2009) presentan GraPPLE, un clasificador de familias de RFAM. Los autores usan la representación *bracketed graph* (las bases son

vértices y todos los enlaces son aristas) y 20 métricas derivadas de los grafos tales como número de puntos de articulación, diámetro, grado, densidad y transitividad entre otros para entrenar una SVM. RNAcon (Panwar *et al.*, 2014) es una SVM que utiliza características estructurales para clasificar el mismo conjunto de secuencias de ncRNA del enfoque anterior. Es una clasificación de dos niveles, donde en el primero discriminan si una molécula es ncRNA o no utilizando frecuencias de trímeros. Retsi *et al.* (2015) presentan un método basado en la combinación de algoritmos genéticos y SVMs para distinguir entre tRNAs, miRNAs, snoRNAs, rRNAs y otros tipos con una precisión de 93 %. Para entrenar el modelo se utilizan características secuenciales, estructurales, termodinámicas y estadísticas.

6.3. El framework NR-PR y representaciones basadas en grafos propuestas

Usualmente, los trabajos previos en representaciones alternativas de la estructura secundaria del ARN abordan el problema de la falta de conservación en los distintos niveles estructurales con la implementación de alguna capa de abstracción. Por ejemplo, se espera que una representación genere objetos similares para un par de secuencias que difieran en un nucleótido perteneciente a una región altamente conservada. En esta sección se propone una generalización llamada framework NR-PR, la cual tiene como finalidad permitir la construcción o especificación de abstracciones donde lo importante es mantener la topología general de la estructura secundaria.

6.3.1. Framework NR-PR

En la estructura secundaria, cada base puede tener dos estados: pareada y no pareada. Por ejemplo, en la notación *dot-bracket* se utiliza el punto para representar a los nucleótidos no pareados, mientras que los paréntesis identifican a las bases pareadas. En caso de contar con pseudonudos, se utilizan distintos tipos de paréntesis con lo que se identifican de manera específica las interacciones entre los nucleótidos.

Partiendo de una estructura sin pseudonudos representada en notación *dot-bracket* y modificando el algoritmo que verifica si los paréntesis de una expresión están balanceados (es decir, que cada paréntesis de apertura tenga uno de cierre y que el

anidamiento de paréntesis sea correcto), es posible obtener qué bases están apareadas entre sí. Considerando esta información, la estructura puede descomponerse en un conjunto de componentes elementales llamados NR y PR. NR denota a una región de bases consecutivas no pareadas (*consecutive u**N**paired **R**egion*). En la nomenclatura de subestructuras del ARN, se pueden categorizar como NR los *bulges*, el *loop* de un *hairpin*, los *loops* de un *internal loop* (simétrico o asimétrico), y las secuencias iniciales o finales sin aparear de la estructura. Por otro lado, PR denota *consecutive **P**aired **R**egion* el cual está dado por los nucleótidos consecutivos pareados (i.e. hélices, componentes de los *stems*). La posición de inicio, final y el tipo de componente forman una 3-tupla que representa a cada uno de los componentes NR y PR. Formalmente, sea s una secuencia de ARN de longitud n y sea ss la estructura secundaria de s. Entonces, cada componente se define por la tupla *component*:

$$
\begin{aligned}
&component = (start, end, type),\\
&\quad \text{donde:}\\
&\quad start \in [1, n],\\
&\quad end \in [start, n],\\
&\quad type \in \{PR, NR\}.
\end{aligned}
\tag{38}
$$

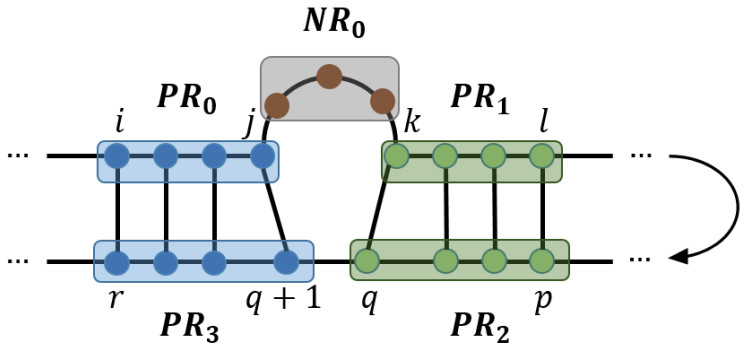

Figura 32: Ejemplo de componentes PR y NR de un fragmento de una estructura secundaria de ARN. La lista de componentes se expresa como $components = [\ldots, (i, j, PR), (j+1, k-1, NR), (k, l, PR), \ldots, (p, q, PR), (q+1, r, PR), \ldots]$.

La Figura 32 muestra un ejemplo de componentes presentes en una estructura secundaria. Puede observarse que no todas las bases pareadas consecutivas corresponden a un único componente PR. Por ejemplo, considere la región $ss_{p..r}$. Utilizando la notación *dot-bracket*, esta subcadena se representa con $p - r + 1$ paréntesis de cierre consecutivos. En este caso, la región $ss_{i..l}$ contiene todas las bases complementarias

a aquellas de $ss_{p..r}$; esto es, $ss_{p..q}$ es complementaria a $ss_{k..l}$ y $ss_{q+1..r}$ es complementaria a $ss_{i..j}$. Sin embargo, $ss_{j+1..k-1}$ es una región no pareada dentro $ss_{i..l}$, lo que significa que $ss_{i..l}$ no es una región pareada consecutiva, por lo que el componente PR correspondiente a la región p a r se divide en dos componentes diferentes ($ss_{p..q}$ y $ss_{q+1..r}$).

6.3.1.1. Derivación de componentes no elementales: *stems*

Con el objetivo de simplificar la notación en el resto de la sección, considere una cadena *dot-bracket ss* de longitud n sin pseudonudos y dos enteros $i, j \in [1, n], i \neq j$. Se define la operación binaria $i \updownarrow j$ como:

$$i \updownarrow j = \begin{cases} Verdadero & \text{si } ss_i \text{ y } ss_j \text{ son bases apareadas en } ss, \\ Falso & \text{en otro caso.} \end{cases} \tag{39}$$

Un *stem* en la lista NR-PR puede identificarse como un par de componentes PR complementarios de la misma longitud. De manera formal, un *stem* S es una tupla de componentes (c_i, c_j) tales que:

$$\begin{gathered} i < j, \\ c_i.type = c_j.type = \text{PR}, \\ c_i.end - c_i.start + 1 = c_j.end - c_j.start + 1, \\ c_i.start \updownarrow c_j.end, \\ c_i.end \updownarrow c_j.start. \end{gathered} \tag{40}$$

6.3.2. Algoritmos propuestos para obtener la lista NR-PR y de *stems*

A continuación se presentan algunas propuestas en pseudocódigo para generar la lista NR-PR y los *stems* de la estructura en tiempo lineal. Estos algoritmos fueron implementados en Python 3 y la librería networkX (Hagberg *et al.*, 2008). En aras de la claridad, se muestra el grafo de dependencias entre las entradas y salidas de los

algoritmos que se describirán (Figura 33).

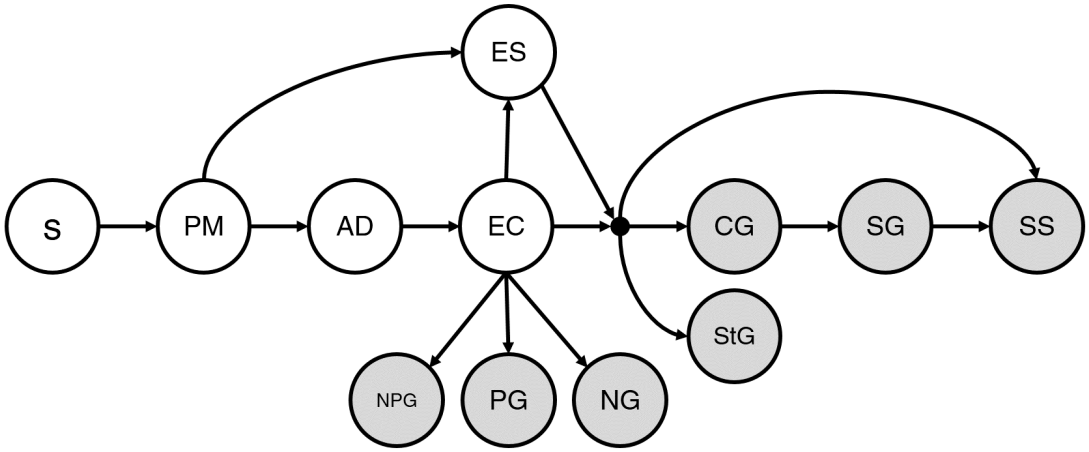

Figura 33: Grafo de dependencias para los algoritmos presentados. **s**: estructura de entrada. **PM**: ParMatching (Algoritmo 5). **AD**: AdjacentDifference (Algoritmo 6). **EC**: ExtractComponents (Algoritmo 7). **ES**: ExtractStems (Algoritmo 8). **NPG**: NPGraph (Algoritmo 9). **PG**: PGraph (Algoritmo 10). **NG**: NGraph (Algoritmo 11). **StG**: StemGraph (Algoritmo 12). **CG**: ComponentGraph (Algoritmo 13). **SG**: StructuralGraph (Algoritmo 14). **SS**: StructuralString (Algoritmo 15). Los nodos sombreados indican las representaciones propuestas en esta sección.

Los algoritmos utilizan algunas funciones y estructuras de datos en común, las cuales se describen a continuación.

- Length(*objeto*): devuelve el número de elementos de una estructura (*objeto*).
- Pila: estructura de datos donde el último elemento en entrar es el primero en salir.
  - Push(*objeto*): almacena un objeto en la pila.
  - Pop(*objeto*): extrae el último objeto almacenado en la pila.
- Arreglo: estructura de datos donde los objetos guardados pueden recuperarse por medio de su posición (índice) en la estructura.
  - *arreglo*[*i*]: devuelve el objeto almacenado en el índice *i*. La notación equivalente, $arreglo_i$, será usada de manera indistinta en los algoritmos.
  - Add(*objeto*): si no se especifica el tamaño del arreglo, esta operación añade el objeto al final del arreglo, es decir, lo asigna a la posición Length(*arreglo*), lo que sería el equivalente a Push(*objeto*).

  - Find(*objeto*): devuelve el índice donde se encuentra almacenado el objeto en el arreglo. Haciendo búsqueda secuencial, esta operación tiene una complejidad de $O(n)$ por cada búsqueda. Sin embargo, si se genera un diccionario que almacena los objetos del arreglo como llave y los índices como valor, el preprocesamiento tiene un costo $O(n)$ pero cada consulta se realiza en $O(1)$.

- Diccionario: estructura que almacena tuplas del tipo $(llave, valor)$, donde *llave* es un objeto de cualquier tipo que funciona como índice para recuperar *valor*. Una implementación común de los diccionarios es una tabla *hash*.

  - *diccionario*[*llave*]: devuelve el *valor* de la tupla $(llave, valor)$ almacenada en el diccionario.
  - AddEntry(*llave* : *valor*): almacena la tupla $(llave, valor)$ en el diccionario.
  - Items(): devuelve la lista de tuplas $(llave, valor)$ almacenadas en el diccionario.

- Grafos y multigrafos: un grafo G es una tupla (V, E) donde V es un conjunto de objetos denominados vértices y E es una conjunto de tuplas (u, v, I) donde $u, v \in V$ e I representa la información de la arista. Usualmente esta información es un valor numérico denominado peso; en nuestro caso, se utilizará para almacenar qué componente `NR-PR`/*stem* representa.

  - CreateVertex(*objeto*): crea una estructura tipo vértice la cual almacena al objeto.
  - *ListaDeVertices*.FindVertex(*objeto*): devuelve la referencia al vértice que contiene al objeto. Esta operación pudiera implementarse con un preprocesamiento en $O(n)$ y consulta en $O(1)$ usando un diccionario donde la llave sea el objeto y la clave la referencia al vértice.
  - CreateEdge($u, v,$ *objeto*): crea una tupla donde los dos primeros elementos son referencias a dos estructuras tipo vértice. La nueva arista almacena a *objeto*.
  - CycleDetection(G): función que regresa todos los ciclos simples del grafo G. Usualmente se implementa en tiempo $O(|E| + |V|)$. En el caso particular de las representaciones propuestas, el grado de los vértices no es mayor a 4 por lo que la cota resulta en $O(|V|)$. Más aún, puesto que los vértices son

componentes NR-PR o una reducción de ellos (*stems*), puede inferirse que nunca habrá más vértices que nucleótidos en la secuencia de entrada.

Paréntesis balanceados

El problema de la coincidencia de los paréntesis o paréntesis balanceados (*parentheses matching*) consiste, en su forma más simple, en verificar que por cada paréntesis de apertura exista uno de cierre y viceversa, y el anidamiento de los paréntesis sea correcto. En este caso, además de verificar que la cadena esté balanceada, es necesario almacenar el índice del paréntesis de apertura/cierre y considerar que pueden existir caracteres punto entre los paréntesis. En otras palabras, si s es una estructura en formato *dot-bracket* de longitud n, el Algoritmo ParMatching debe devolver un arreglo *parmatch* tal que:

$$parmatch_i = \begin{cases} -1 & \text{si } s_i = \text{'.'} \\ j \mid i \updownarrow j & \text{en otro caso.} \end{cases} \tag{41}$$

El Algoritmo 5 (ParMatching) se muestra a continuación.

Algoritmo 5: ParMatching(s)

```
   Input  : s, una cadena en formato dot-bracket.
   Output: parmatch, parenthesis matching de s.
 1 n ←Length(s)
 2 parmatch ← arreglo de longitud n inicializado en −1
 3 stack ← pila vacía
 4 for 0 ≤ i < n do
 5 |   if s[i] == '(' then
 6 |   |   stack.Push(i)
 7 |   else
 8 |   |   if s[i] == ')' then
 9 |   |   |   j = stack.Pop()
10 |   |   |   parmatch[i] ← j
11 |   |   |   parmatch[j] ← i
12 |   |   end
13 |   end
14 end
15 return parmatch
```

Cada vez que el algoritmo encuentra un paréntesis de apertura, el índice donde lo encuentra se almacena en una pila (líneas 5-6). Si el carácter resulta ser un paréntesis de cierre, se recupera el índice del último paréntesis de apertura, se elimina de la pila y se actualiza el *parmatch* (líneas 8-11). Los puntos son ignorados. De aquí podemos observar que si la cadena tiene más paréntesis de cierre o un anidamiento incorrecto, la operación *stack.pop()* arroja error por estar la pila vacía. Por otro lado, si la cadena tiene más paréntesis de apertura que de cierre, el algoritmo no termina con la pila vacía. Por motivos de claridad, estas validaciones no se muestran en el pseudocódigo.

Una de las ventajas de tener el arreglo *parmatch* es que es posible responder ciertas preguntas en tiempo constante. Por ejemplo, para encontrar la base complementaria a s_i basta con obtener el valor $parmatch[i]$ y la respuesta a si dos bases i y j están pareadas en la estructura corresponde al resultado de la comparación $parmatch[i] == j$.

Algoritmo Adjacent difference

La lista de componentes `NR-PR` se compone de bases consecutivas con el mismo estado. Si bien en las bases no pareadas esto no representa problema, en las pareadas

es necesario considerar el *parmatch*. El Algoritmo 6 (AdjacentDifference) muestra cómo identificar las bases pareadas consecutivas en tiempo lineal por medio de la diferencia entre los índices de las bases pareadas consecutivas.

Algoritmo 6: AdjacentDifference(*parmatch*)

Input : *parmatch*, *parenthesis matching* de alguna estructura.
Output: *adj*, la diferencia de adyacencias de *parmatch*.

```
1 n ←Length(parmatch)
2 adj ← arreglo de longitud n inicializado en 0
3 for 0 ≤ i < n do
4 |   if parmatch[i] ≠ −1 then
5 |   |   adj[i] ← parmatch[i] − parmatch[(i + 1) mód n]
6 |   end
7 end
8 return adjacency
```

El algoritmo ignora los puntos de la estructura y les asigna el valor cero en el arreglo *adj* puesto que el valor de inicialización no se modifica (líneas 2-4). Después, en cada índice i que corresponde a un paréntesis, se obtiene la diferencia de $j-k$, donde j es el índice de la base complementaria a i y k es el índice de la base complementaria a $(i+1)$ mód n (línea 5). La operación de módulo se considera en el caso de que el primero y último nucleótidos estén apareados. Si j y k son consecutivas, la diferencia es 1 o un número mayor a 1. Para mostrar un ejemplo del arreglo resultante y visualizar cómo quedan las fronteras de los componentes definidas, considere la siguiente secuencia:

Secuencia de ejemplo

```
>AACY023290437.1/381-448 (SAH riboswitch, RFAM_ID = RF01057)
CGCCCCAAGGAGCGUUGCAACCCGCGCAUUGGGUGAGGCUUGAGGGCGAACGUCGGCGACAGAAUACA
(((((....((((.((...(((((.....)))))))).))))..))))))...(((...))).......
```

El resultado de aplicar los Algoritmos 5 y 6 a la secuencia de ejemplo se muestra en la Figura 34. Puede observarse que los componentes NR son ceros consecutivos, mientras que los PR son secuencias monótonamente crecientes que inician en 1.

S	I	P	A
(	0	47	1
(	1	46	1
(	2	45	1
(	3	44	1
(	4	43	44
.	5	-1	0
.	6	-1	0
.	7	-1	0
.	8	-1	0
(	9	40	1
(	10	39	1
(	11	38	1
(	12	37	38
.	13	-1	0
(	14	35	1
(	15	34	35
.	16	-1	0
.	17	-1	0
.	18	-1	0
(	19	33	1
(	20	32	1
(	21	31	1
(	22	30	1
(	23	29	30
.	24	-1	0
.	25	-1	0
.	26	-1	0
.	27	-1	0
.	28	-1	0
)	29	23	1
)	30	22	1
)	31	21	1
)	32	20	1
)	33	19	4
)	34	15	1
)	35	14	15
.	36	-1	0
)	37	12	1
)	38	11	1
)	39	10	1
)	40	9	10
.	41	-1	0
.	42	-1	0
)	43	4	1
)	44	3	1
)	45	2	1
)	46	1	1
)	47	0	1
.	48	-1	0
.	49	-1	0
.	50	-1	0
(	51	59	1
(	52	58	1
(	53	57	58
.	54	-1	0
.	55	-1	0
.	56	-1	0
)	57	53	1
)	58	52	1
)	59	51	52
.	60	-1	0
.	61	-1	0
.	62	-1	0
.	63	-1	0
.	64	-1	0
.	65	-1	0
.	66	-1	0
.	67	-1	0

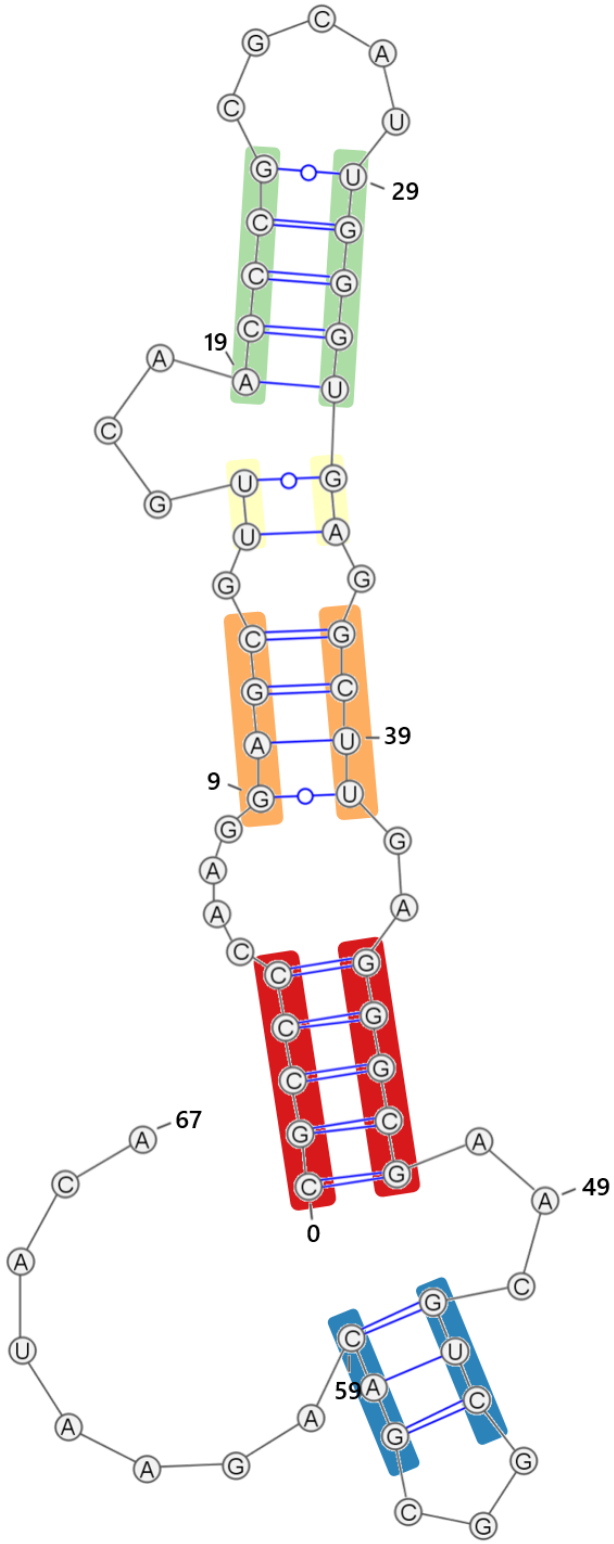

Figura 34: Resultado de aplicar los Algoritmos 5 y 6 a la secuencia de ejemplo. La figura muestra la estructura secundaria con las regiones complementarias consecutivas resaltadas. Las bases están numeradas iniciando en 1. En la tabla, **S** es el caracter de la estructura *dot-bracket* en el índice indicado en **I** (iniciando en 0). **P** es el *parmatching* de la estructura y **A** es el *adj* de la estructura.

Extracción de componentes NR-PR

Considerando las secuencias monótonamente crecientes de *adj*, al analizar el arreglo en pares y almacenar qué tipo de componente se está creando puede completarse la lista de componentes NR-PR (Algoritmo 7).

Algoritmo 7: ExtractComponents(*adj*)

Input : *adj*, la diferencia de adyacencias de una estructura.
Output: *components*, la lista NR-PR de la estructura.

```
n ← Length(adj)
components ← arreglo vacío
for 0 ≤ i < n do
    Aplicar acción de la Tabla 20 según (adj[idx − 1], adj[idx])
end
return components
```

Tabla 20: Configuraciones posibles para cada par de elementos adyacentes ($adj[idx-1], adj[idx]$) en *adj*. Un componente se define por la tripleta ($start, end, type$). La notación X||Y indica que $adj[idx-1]$ es la última base de un componente tipo X y $adj[idx]$ es la primer base de un componente tipo Y, mientras que "inter Z" indica que tanto $adj[idx-1]$ como $adj[idx]$ están contenidos en el mismo componente de tipo Z.

$adj[idx-1]$	$adj[idx]$	Descripción	Acción
Null	{0,1,otro}	Inicio cadena	$start = 0$
0	0	inter NR	Ignorar
1	0	PR||NR	Agregar $(start, idx-1, \text{PR})$ $start = idx$
otro	0	PR||NR	Agregar $(start, idx-1, \text{PR})$ $start = idx$
0	1	NR||PR	Agregar $(start, idx-1, \text{NR})$ $start = idx$
1	1	inter PR	Ignorar
otro	1	PR||PR	Agregar $(start, idx-1, \text{PR})$ $start = idx$
0	otro	NR||PR	Agregar $(start, idx-1, \text{NR})$ $start = idx$
1	otro	inter PR	Ignorar
otro	otro	PR||PR	Agregar $(start, idx-1, \text{PR})$ $start = idx$
0	*Null*	NR||	Agregar $(start, n-1, \text{NR})$
1	*Null*	PR||	Agregar $(start, n-1, \text{PR})$
otro	*Null*	PR||	Agregar $(start, n-1, \text{PR})$

En la Figura 34 pueden apreciarse casi todos los casos considerados en la Tabla 20.

Extracción de *stems*

Retomando la Ecuación 40, un *stem* S es un par de componentes (c_i, c_j), tales que son diferentes y el inicio de uno es complementario al final del otro, en ambos componentes. Indexando los inicios de los PR es posible generar la lista de *stems* en $O(n_c)$, donde n_c es el número de componentes de la estructura s (Algoritmo 8).

Algoritmo 8: ExtractStems(*parmatch*, *components*)

```
   Input  : parmatch, parenthesis matching de alguna estructura.
            components, lista de componentes NR-PR de la estructura.
   Output: stems, lista de stems de la estructura.
 1 PRstarts ← diccionario vacío
 2 foreach c in components do
 3 |   if c.type = PR then
 4 |   |   PRstarts.AddEntry(c.start : c)
 5 |   end
 6 end
 7 complementaryPR ← Diccionario vacío
 8 foreach start : component in PRstarts.Items() do
 9 |   complementaryPR.AddEntry(component :
   |    PRstarts[parmatch[component.end]])
10 end
11 stems ← arreglo vacío
12 foreach c_i : c_j in complementaryPR.Items() do
13 |   if c_i.start < c_j.start then
14 |   |   stems.Add((c_i, c_j))
15 |   end
16 end
17 return stems
```

El algoritmo indexa el inicio de cada componente PR (líneas 2-6). Posteriormente, asigna cada componente a su complementario en un diccionario (líneas 8-10). Finalmente, se recorren todas las entradas del diccionario y se almacenan los *stems* (c_i, c_j) con $i < j$ (líneas 11-16).

6.3.3. Generando representaciones tipo grafo a partir de la lista NR-PR

A continuación se muestran y analizan siete ideas de representaciones a partir de la lista NR-PR. Cabe destacar que, si bien se muestra la construcción de grafos a nivel de estructura secundaria, la secuencia primaria de cada componente es recuperable utilizando las posiciones almacenadas en los atributos *start* y *end*. Para ejemplificar el grafo resultante de cada representación analizada, se emplea una secuencia del riboswitch de purina (RF00167) y del riboswitch SAH (RF01057), las cuales se muestran a continuación:

Secuencia de ejemplo de Purine riboswitch

```
>AGTN01223743.1/435-371 (Purine riboswitch, RFAM_ID = RF00167)
UUGGUAUAUGCCGAAUAAUUUGGAUUUGGCGUUCUACCCGGUUGCCGAAACAACCGGACUACCAA
.(((((.(((((((((.......)))))))))).....(((((((......)))))))..))))).
```

Secuencia de ejemplo de SAH riboswitch

```
>AACY023290437.1/381-448 (SAH riboswitch, RFAM_ID = RF01057)
CGCCCCAAGGAGCGUUGCAACCCGCGCAUUGGGUGAGGCUUGAGGGCGAACGUCGGCGACAGAAUACA
(((((....((((.((...(((((.....)))))))..))))..))))))...(((...))).......
```

Las transformaciones resultantes para las secuencias de los riboswitches de purina y SAH se muestran en las figuras 35 y 36, respectivamente.

La figuras 35a) y 36a) muestran el *squiggle plot* de las estructuras secundarias de las secuencias con los componentes NR y PR anotados. La estructura secundaria se predijo con el programa RNAfold implementado en el ViennaRNA Package 2.0 (Lorenz *et al.*, 2011), mientras que el diagrama se generó con VARNA (Darty *et al.*, 2009).

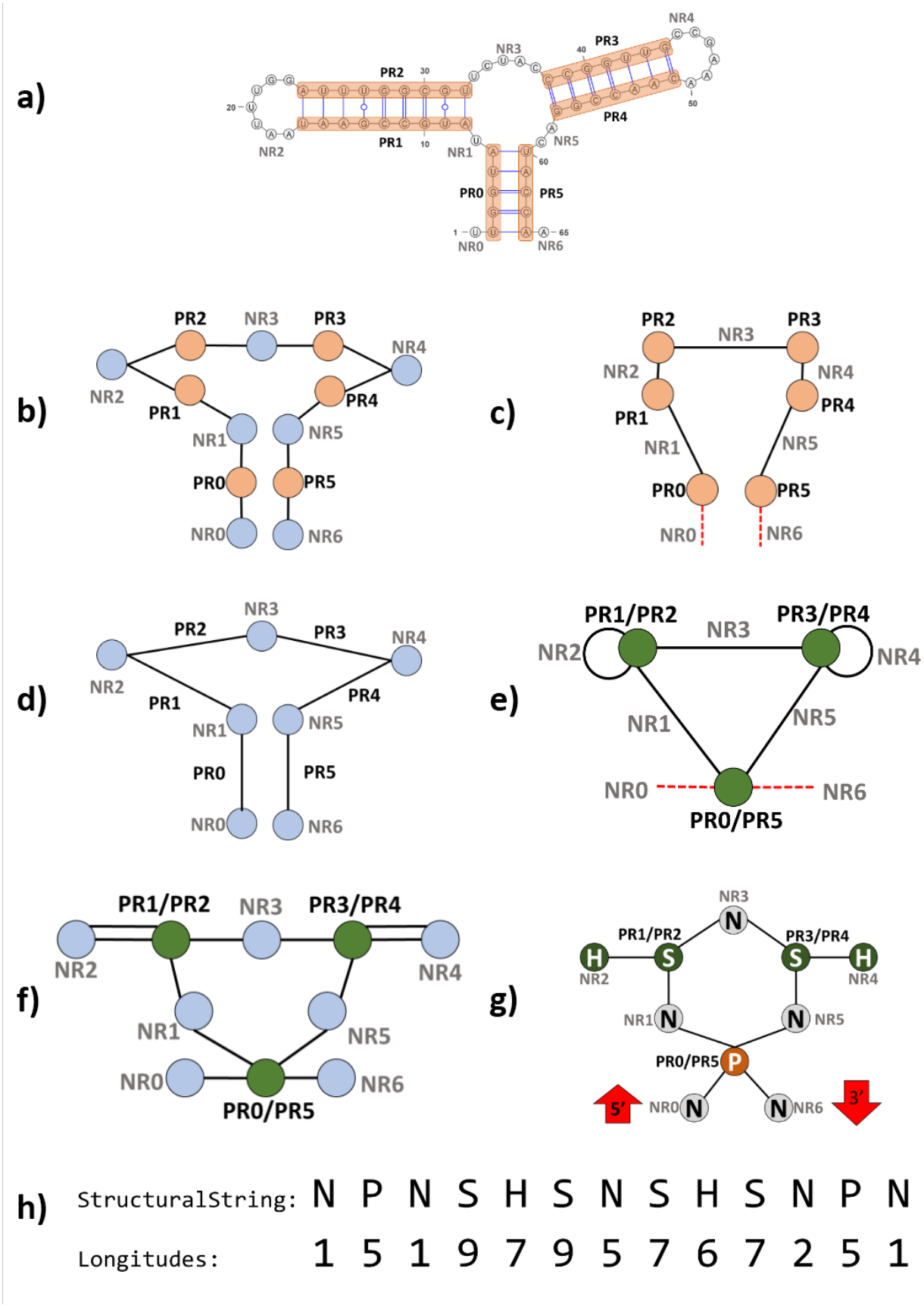

Figura 35: Representaciones de la secuencia del riboswitch de purina. a) Descomposición NR-PR. b) NPGraph. c) PGraph. d) NGraph. e) StemGraph. f) ComponentGraph. g) Structural-Graph. h) StructuralString.

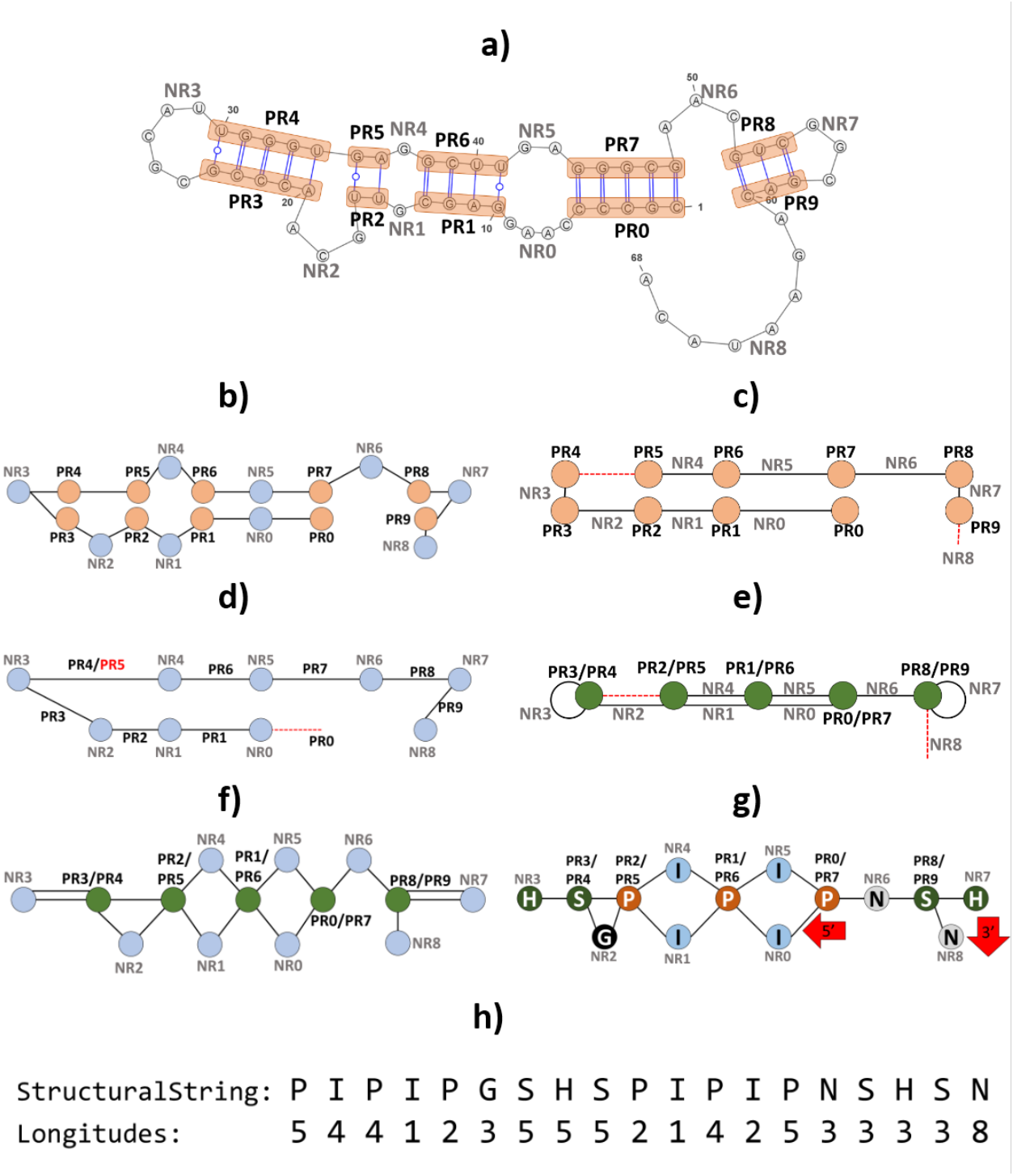

Figura 36: Representaciones de la secuencia del riboswitch de SAH. a) Descomposición NR-PR. b) NPGraph. c) PGraph. d) NGraph. e) StemGraph. f) ComponentGraph. g) StructuralGraph. h) StructuralString.

6.3.3.1. NPGraph

La idea principal de la representación NPGraph es crear un vértice por cada componente en C y una arista por cada par de componentes adyacentes en la lista NR-PR. El Algoritmo 9 muestra cómo generar la representación.

Algoritmo 9: NPGraph(C)

Input : C, la lista de componentes NR-PR.
Output: G, un grafo no dirigido.

```
1  n ←Length(C)
2  V ← arreglo vacío
3  E ← arreglo vacío
4  for 0 ≤ i < n do
5  |  V.Add(CreateVertex(c_i))
6  end
7  for 0 ≤ i < n − 1 do
8  |  E.Add(CreateEdge(V[i], V[i + 1], null))
9  end
10 return G = (V, E)
```

El arreglo itera sobre C y crea el conjunto de vértices (líneas 4-6). Posteriormente, crea las aristas entre cada par de componentes adyacentes (líneas 7-9).

Los grafos de ejemplo se muestran en las figuras 35b) y 36b). Esta representación es la más sencilla de generar, y provee abstracción en los componentes contiguos del mismo tipo (debido a la lista NR-PR), haciéndola similar al *Abstract Shape* de nivel uno (Steffen *et al.*, 2005) si se recorren los nodos en dirección $5' \rightarrow 3'$. Sin embargo, al no considerar los enlaces entre componentes PR pareados (a diferencia del *Abstract Shape*), la información de las bases complementarias se pierde, lo que le resta utilidad práctica.

6.3.3.2. PGraph

En esta representación, se crea un vértice por cada componente PR en la lista C y cada componente NR adyacente a dos componentes PR se convierte en una arista entre ellos. El Algoritmo 10 muestra cómo generar esta representación.

Algoritmo 10: PGraph(C)

Input : C, la lista de componentes NR-PR.
Output: G, un grafo no dirigido.

```
1  n ←Length(C)
2  V ← arreglo vacío
3  E ← arreglo vacío
4  foreach c in C do
5  |  if c.type == PR then
6  |  |  V.Add(CreateVertex(c))
7  |  end
8  end
9  for 1 ≤ i < n − 1 do
10 |  if c_i.type == NR and c_{i−1}.type == PR and c_{i+1}.type == PR then
11 |  |  e ←CreateEdge(V.FindVertex(c_{i−1}), V.FindVertex(c_{i+1}), c_i)
12 |  |  E.Add(e)
13 |  end
14 end
15 return G = (V, E)
```

El algoritmo primero crea los vértices usando los componentes PR (líneas 4-8). Después, itera sobre la lista de componentes. Cuando encuentra un componente NR verifica si sus dos vecinos son PR, en caso afirmativo crea una arista entre ellos (líneas 9-13).

Los grafos de ejemplo se muestran en las figuras 35c) y 36c). Esta representación provee abstracción en los componentes contiguos PR y descarta todos los componentes NR al transformarlos en aristas. Al recorrerse el grafo en dirección $5' \rightarrow 3'$, la cadena generada es similar al *Abstract Shape* de nivel dos. Sin embargo, de la misma forma que la representación NPGraph, al no considerar los enlaces entre componentes PR pareados, la información de las bases complementarias se pierde. Existen dos problemas adicionales con esta representación: el primero es que la información de los componentes NR al inicio o final de la secuencia se pierden ya que no es posible generar la arista al faltar otro vértice incidente (componentes $NR0$ y $NR6$ en la Figura 35c) y el componente *NR8* en la Figura 36c)). El segundo es que dos componentes PR contiguos no tienen una arista entre ellos, por lo que esta representación en realidad genera un conjunto de grafos desconectados.

6.3.4. NGraph

Esta representación consiste en crear un vértice por cada componente NR en la lista C, y cada componente PR adyacente a dos componentes NR se convierte en una arista entre ellos. El Algoritmo 11 muestra cómo generar la representación.

Algoritmo 11: NGraph(C)

```
   Input  : C, la lista de componentes NR-PR.
   Output: G, un grafo no dirigido.
 1 n ←Length(C)
 2 V ← arreglo vacío
 3 E ← arreglo vacío
 4 foreach c in C do
 5 |   if c.type == NR then
 6 |   |   V.Add(CreateVertex(c))
 7 |   end
 8 end
 9 for 1 ≤ i < n do
10 |   if c_i.type == PR and c_{i-1}.type == NR and c_{i+1}.type == NR then
11 |   |   e ←CreateEdge(V.FindVertex(c_{i-1}), V.FindVertex(c_{i+1}), c_i)
12 |   |   E.Add(e)
13 |   end
14 end
15 return G = (V, E)
```

El Algoritmo es bastante similar al Algoritmo 10, con la diferencia de que los componentes PR ahora son aristas y los NR vértices.

Los grafos de ejemplo se muestran en las figuras 35d) y 36d). Esta representación provee abstracción en los componentes contiguos NR y descarta todos los componentes PR al transformarlos en aristas. Al recorrerse el grafo en dirección $5' \rightarrow 3'$, la cadena generada no es similar al *Abstract Shape* de ningún nivel. De la misma forma que las representaciones NPGraph y PGraph, al no considerar los enlaces entre componentes PR pareados, la información de las bases complementarias se pierde. Más aún, de manera similar a la representación PGraph, la información de los componentes PR al inicio o final de la secuencia se pierden ya que no se puede generar la arista porque falta otro vértice incidente (componentes $PR0$ en la Figura 36d)). Además, es nece-

sario considerar el caso de dos componentes PR adyacentes (*PR*4 y *PR*5 en la Figura 36d)).

6.3.5. StemGraph

La representación StemGraph trabaja directamente con los *stems* de la estructura en lugar de los componentes PR. Cada *stem* se convierte en un vértice y cada componente NR entre ellos se considera como una arista. El Algoritmo 12 muestra cómo generar la representación.

Algoritmo 12: StemGraph(C, S)

Input : C, la lista de componentes NR-PR de una estructura.
S, la lista de *stems* de la estructura.
Output: G, un multigrafo no dirigido.

```
1  n ←Length(C)
2  V ← lista vacía
3  E ← lista vacía
4  CmapV ← diccionario vacío
5  foreach s in S do
6  |  v ←CreateVertex(s)
7  |  V.Add(v)
8  |  CmapV.AddEntry(s[0] : v)
9  |  CmapV.AddEntry(s[1] : v)
10 end
11 for 1 ≤ i < n − 1 do
12 |  if C_i.type == NR and C_{i−1}.type == PR and C_{i+1}.type == PR then
13 |  |  e ←CreateEdge(CmapV[C_{i−1}], CmapV[C_{i+1}], C_i)
14 |  |  E.Add(e)
15 |  end
16 end
17 return G = (V, E)
```

En primer lugar, se crean los vértices segun los *stems* de la estructura y a su vez se crea un diccionario donde las llaves son los componentes del *stem* y el valor el vértice recién creado (líneas 5-10). Posteriormente, se itera sobre la lista de componentes. Cada componente NR adyacente a dos *stems* se convierte en una arista entre ellos; para identificar el vértice que corresponde a dicho *stem*, se utiliza el diccionario creado anteriormente (líneas 11-16).

Los grafos de ejemplo se muestran en las figuras 35e) y 36e). A diferencia de las anteriores, esta representación contiene información sobre las bases pareadas al usar los *stems* como vértices. Además, la topología del grafo tiene cierto parecido al *squiggle plot* mostrado, y es posible identificar ciertos componentes a simple vista. Por ejemplo, las autoaristas $NR2$ y $NR4$ en la Figura 35e), y $NR3$ y $NR7$ en la Figura 36e) indican la presencia de *hairpin loops* en la estructura secundaria. Sin embargo, de la misma manera que en el PGraph, la información de los componentes NR al inicio o final de la secuencia se pierden ya que no se puede generar la arista al faltar otro vértice incidente (componentes $NR0$ y $NR6$ en la Figura 35e) y el componente *NR8* en la Figura 36e)).

6.3.6. ComponentGraph

La representación ComponentGraph crea un vértice por cada *stem* y cada componente NR, y dichos vértices se conectan por aristas si son adyacentes en la lista NR-PR. El Algoritmo 13 muestra cómo generar la representación.

Algoritmo 13: ComponentGraph(C, S)

Input : C, la lista de componentes NR-PR de una estructura.
S, la lista de *stems* de la estructura.
Output: G, un grafo dirigido.

```
1  n ←Length(C)
2  V ← lista vacía
3  E ← lista vacía
4  CmapV ← diccionario vacío
5  foreach c in C do
6      if c.type == NR then
7          v ←CreateVertex(c)
8          V.Add(v)
9          CmapV.AddEntry(c : v)
10     end
11 end
12 foreach s in S do
13     v ←CreateVertex(s)
14     V.Add(v)
15     CmapV.AddEntry(s[0] : v)
16     CmapV.AddEntry(s[1] : v)
17 end
18 for 0 ≤ i < n − 1 do
19     e ←CreateEdge(CmapV[C_i], CmapV[C_{i+1}], null)
20     E.Add(e)
21 end
22 return G = (V, E)
```

Este algoritmo es bastante similar al Algoritmo 12, con la diferencia de que ahora los componentes NR también son vértices y por consiguiente también son indexados (líneas 5-10).

Los grafos de ejemplo se muestran en las figuras 35f) y 36f). En esta representación no se pierde la información de ningún componente. Además, la topología del grafo tiene cierto parecido al *squiggle plot* mostrado, y es posible identificar ciertos componentes a simple vista. Por ejemplo, un par de vértices *stem* y NR pertenecientes al mismo ciclo simple corresponden a un *hairpin-stem* y un *hairpin-loop*, respectivamente. Los vértices que mapean componentes NR y tienen grado uno corresponden con regiones no pareadas iniciales o finales de la estructura. Los ciclos simples del grafo de tamaño 3 o 4 pueden corresponder a *bulges* o *internal loops*.

6.3.7. StructuralGraph

Esta representación se basa en el ComponentGraph para modelar la topología de las subestructuras en la estructura secundaria en un grafo. Básicamente se etiquetan los vértices del ComponentGraph según las siguientes condiciones:

- `S`: si es un *stem* perteneciente a un *hairpin*.
- `H`: si es un *loop* perteneciente a un *hairpin*.
- `P`: si es un *stem* que no pertenece a un *hairpin*.
- `I`: si es un *internal loop*.
- `G`: si es un *bulge*.
- `N`: si es un componente *NR* que no fue etiquetado como `H`, `I` o `G`.

Posteriormente, el grafo se transforma a uno no dirigido. El Algoritmo 14 muestra cómo generar la representación.

Algoritmo 14: StructuralGraph(CG)

Input : CG, ComponentGraph de s.
Output: SG, un grafo dirigido.

```
1  ciclos ←CycleDetection(G)
2  foreach ciclo in ciclos do
3  |  Aplicar acción de la Tabla 21 según ciclo
4  end
5  foreach v in CG.V do
6  |  if v.label == null then
7  |  |  if v.type == NR then
8  |  |  |  v.label = N
9  |  |  else
10 |  |  |  v.label = P
11 |  |  end
12 |  end
13 end
14 return SG = (CG.V, supp(CG.E))
```

Tabla 21: Identificación de subestructuras en el ComponentGraph.

Length(*ciclo*)	Composición	Subestructura	Acción
2	2 S	N/A	–
	1 S, 1 NR	*Hairpin*	Etiquetar S como `S` y NR como `H`
	2 NR	N/A	–
3	3 S	*Junction*	–
	2 S, 1 NR	*Bulge*	Etiquetar NR como `G`
	1 S, 2 NR	N/A	–
	3 NR	N/A	–
4	4 S	*Junction*	–
	3 S, 1 NR	*Junction*	–
	2 S, 2 NR	*Internal loop*	Etiquetar ambos NR como `I`
	1 S, 3 NR	N/A	–
	4 NR	N/A	–
>5	Cualquiera	*Junction*	–

En primer lugar, el algoritmo busca todos los ciclos simples del ComponentGraph (línea 1). Posteriormente, dependiendo de la composición de cada ciclo encontrado se etiquetan los componentes (línea 2-4). Finalmente, los componentes que no fueron etiquetados se nombran como `N` o `P` dependiendo de si son un componente NR o un *stem*, respectivamente (líneas 5-13).

La parte crucial del algoritmo reside en la Tabla 21. El argumento que apoya porque ciertos ciclos no son posibles (marcados como N/A) se basa en la construcción de los componentes *NR* y *stems*. Si un par de componentes NR están conectados por una arista entonces son consecutivos, lo que significa que deberían estar fusionados en un solo componente NR y no separados. De manera análoga, un ciclo entre dos *stems* implica que sus componentes PR son adyacentes, por lo que deberían estar fusionados en un solo *stem*. El resto de los casos no posibles implican que dos componentes NR sean adyacentes. Los componentes etiquetados (`S`, `H`, `G` e `I`) se derivan de la representación gráfica de las subestructuras. Los grafos de ejemplo se muestran en las figuras 35g) y 36g). Puede observarse como esta representación mapea directamente

las estructuras de alto nivel presentes en el *squiggle plot* sin perder información.

6.3.8. StructuralString

Finalmente, se presenta una representación de la estructura no en forma de grafo, sino de cadena. El StructuralString se deriva de recorrer los vértices resultantes del StructuralGraph en dirección 5′ → 3′ y devolver la secuencia de las etiquetas de los vértices recorridos.

Algoritmo 15: StructuralString(*SG*)

Input : *C*, la lista de componentes NR-PR de una estructura.
S, la lista de *stems* de la estructura.
SG, el StructuralGraph de la estructura.
Output: *structuralstring*, la representación StructuralString de la estructura.

```
1  structuralstring ← cadena vacía
2  foreach c in C do
3      if c.type == NR then
4          label = CG.V.FindVertex(c).label
5      else
6          s = S.FindStem(c)
7          label = CG.V.FindVertex(s).label
8      end
9      structuralstring = structuralstring · label
10 end
11 return structuralstring
```

En la entrada se considera la lista de componentes NR-PR (*C*) y el StructuralGraph (*SG*). Al iterarse la lista de componentes (líneas 2-9), sólo existen dos opciones para el tipo del componente: NR o PR. En caso de ser un componente NR, se recupera la etiqueta del vértice en el grafo estructural correspondiente a dicho componente (líneas 3-4). En caso contrario, el componente PR ya está considerado como parte de un *stem*, por lo que primero es necesario recuperar el objeto *stem* correspondiente (línea 6) para posteriormente buscarlo en el grafo estructural y recuperar la etiqueta (línea 7). Finalmente, se concatena la etiqueta al *structuralstring* (línea 9).

Las cadenas de ejemplo se muestran en las figuras 35h) y 36h). Los `StructuralStrings` resultantes alineados con sus secuencias primarias y secundarias son los siguientes:

Purine

```
>AGTN01223743.1/435-371 (Purine riboswitch, RFAM_ID = RF00167)
UUGGUAUAUGCCGAAUAAUUUGGAUUUGGCGUUCUACCCGGUUGCCGAAACAACCGGACUACCAA
.(((((.(((((((((.......)))))))))).....(((((((......)))))))..))))).
NP    NS        H      S        N    S      H     S      N P    N
15    19        7      9        5    7      6     7      2 5    1
```

SAH

```
>AACY023290437.1/381-448 (SAH riboswitch, RFAM_ID = RF01057)
CGCCCCAAGGAGCGUUGCAACCCGCGCAUUGGGUGAGGCUUGAGGGCGAACGUCGGCGACAGAAUACA
(((((....((((.((...(((((.....)))))))..))))..)))))...(((...))).......
P    I   P   IP G  S    H    S    P IP   I P    N  S  H  S  N
5    4   4   12 3  5    5    5    2 14   2 5    3  3  3  3  8
```

Esta representación permite distinguir a simple vista los componentes estructurales y su longitud. Al recuperar los *stems* de la lista de componentes, es posible identificar de manera inequívoca los PR complementarios que los forman. Por otro lado, la elección de las letras para etiquetar cada subestructura no fue aleatoria, sino que corresponden con alguna de las utilizadas para representar a los 20 aminoácidos, esto con la finalidad de poder aplicar algún algoritmo de alineamiento o clustering de proteínas en un grupo de StructuralStrings dado.

6.4. Aplicaciones basadas en el StructuralString

En esta sección se presentan algunas pruebas de concepto de aplicaciones potenciales de la representación StructuralString. En particular, se cuantifican las estructuras redundantes en un conjunto de secuencias, se calcula el porcentaje de estructuras compartidas entre diversos tipos de ncRNA y finalmente, se identifican los motivos estructurales. Estas pruebas se realizaron en secuencias de riboswitches reportadas en RFAM versión 12.1.

6.4.1. Metodología

Con la finalidad de obtener el conjunto de datos para las pruebas, se descargó la base de datos RFAM 12.1 en su totalidad. Después, en el archivo de descripciones, se buscaron las familias que contuvieran la palabra “riboswitch”. En total se obtuvieron 28 familias, de las cuales se recuperaron sus secuencias y se eliminaron los duplicados (es decir, filtrado al 100 % de identidad), resultando en un total de 33540. Posteriormente, se predijo la estructura secundaria de cada secuencia con la herramienta RNAfold y se calculó su StructuralString.

La identificación de estructuras únicas se realizó a nivel de StructuralString; es decir, se obtuvo un conjunto a partir de la lista de cadenas estructurales por familia. Segundo, se calculó la matriz de comparación (no simétrica) de acuerdo al número de estructuras en común entre pares de familias como sigue: por cada par de familias (f_i, f_j), se calculó el porcentaje que representan las cadenas en común respecto al total de secuencias contenidas en la familia f_i. Tercero, se utilizaron dos programas de MEME SUITE (Bailey *et al.*, 2009) para las tareas relacionadas con la búsqueda de motivos estructurales: MEME (Bailey y Elkan, 1994) para encontrar motivos en cada familia y MAST (Bailey y Gribskov, 1998) para buscar los motivos encontrados en estructuras consenso reportadas en RFAM 12.1. Para obtener dichos consensos, se descargaron los alineamientos estructurales y se sanitizaron eliminándose los *gaps* estructurales y los pseudonudos. Se calculó el StructuralString de la estructura consenso para cada familia. Por último, estas cadenas estructurales se depositaron en un solo archivo fasta, y por medio de MAST se buscó cada uno de los mejores motivos reportados por familia.

6.4.2. Resultados

6.4.2.1. Conjunto de datos

En total se obtuvieron 33540 secuencias de 28 diferentes familias de riboswitches (Tabla 22). La diferencia en número de familias con respecto al Capítulo 3 (16) reside en el hecho de que la versión 11 fue liberada en 2012, mientras que la versión

12.1 en 2016. Esto es una muestra de la rapidez de los avances que se logran en la identificación de nuevas moléculas de ncRNA. De aquí en adelante, las familias de riboswitches pueden referirse indistintamente por su nombre o por su identificador de RFAM (columna RFAM_ACC).

Puede observarse que el conjunto de riboswitches es desbalanceado: la familia más grande es RF00174 con 6740 secuencias, mientras que la más pequeña es RF01510 con solamente 7. Una vez calculados los `StructuralStrings`, se obtuvo el conjunto de cadenas no redundante, es decir, las cadenas estructurales distintas por familia (columna #SSnr). Esto se logró de manera simple en Python al transformar la lista de cadenas estructurales (tipo `list`) a un conjunto (tipo `set`) y después hacer la transformación del conjunto resultante a una lista. Con la finalidad de apreciar mejor la proporción de cadenas estructurales redundantes en cada familia, se calculó el porcentaje de cadenas reduntantes (%SSr) como:

$$\%\mathrm{SSr} = \frac{\#\mathrm{Snr} - \#\mathrm{SSnr}}{\#\mathrm{Snr}} \times 100. \tag{42}$$

Note que 11 familias presentaron un porcentaje de cadenas estructurales redundantes arriba del 40 % (RF01051, RF00050, RF00504, RF01510, RF01056, RF00522, RF00167, RF00162, RF01826, RF01727 Y RF00059). En particular, las tres familias más redundantes fueron PreQ1-II (85.6 %), SAM_V (77.8 %) y SAM-SAH (75 %). Estas familias usualmente son reportadas como altamente conservadas a nivel estructural (Eichhorn *et al.*, 2014; Poiata *et al.*, 2009; Weinberg *et al.*, 2010). Vale la pena destacar que este resultado no se obtuvo utilizando alineamiento múltiple, sino mediante una transformación de tipo de dato, la cuál tiene una complejidad de $O(n)$, donde n es el número de cadenas en la lista. Respecto a la longitud de las cadenas estructurales resultantes (columna Long(SS)), puede observarse una reducción general del tamaño en relación a la secuencia de nucleótidos, alcanzando reducciones desde 3.3 a 1 (SAM-IV) hasta 6.3 a 1 (PreQ1).

Tabla 22: Secuencias utilizadas para las pruebas de la representación StructuralString. **Nombre:** nombre de la molécula. Entre corchetes se muestra su nombre corto; **RFAM_ACC:** identificador de la familia en RFAM; **#Snr:** número de secuencias primarias no redundantes (filtrado a 100 % de identidad); **#SSnr:** número de `StructuralStrings` no redundantes (filtrado a 100 % de identidad); **%SSr:** porcentaje de `StructuralStrings` redundantes; **Long(S):** promedio y desviación estándar (de las longitudes) de las secuencias primarias; **Long(SS):** promedio y desviación estándar (de las longitudes) de las `StructuralStrings`.

Nombre	RFAM_ACC	#Snr	#SSnr	%SSr	Long(S)	Long(SS)
AdoCbl riboswitch	RF01482	222	165	25.7	140.5±19.8	39.2±6.6
Cobalamin riboswitch	RF00174	6740	6094	9.6	186.7±45.2	48.8±13.2
Cyclic di-GMP-I riboswitch	RF01051	1576	856	45.7	86.3±11.8	23.3±4.8
Cyclic di-GMP-II riboswitch	RF01786	183	166	9.3	85.2±7.0	22.6±5.2
drz-agam-1 riboswitch	RF01787	3793	2583	31.9	110.3±14.5	33.1±6.8
drz-agam-2-2 riboswitch	RF01788	596	495	16.9	180.6±26.0	43.4±8.3
FMN riboswitch (RFN element)	RF00050	1986	1014	48.9	144.8±29.7	40.8±9.8
glmS glucosamine-6-phosphate activated ribozyme	RF00234	528	421	20.3	173.0±37.4	40.1±10.3
Glycine riboswitch	RF00504	2786	1094	60.7	95.7±16.4	25.0±6.1
Lysine riboswitch	RF00168	1281	957	25.3	178.8±16.7	43.7±7.5
M. florum riboswitch [MFR]	RF01510	7	4	42.9	63.3±0.5	13.6±2.8
Magnesium Sensor [Mg_sensor]	RF01056	62	37	40.3	114.2±4.9	22.9±4.1
Moco (molybdenum cofactor) riboswitch [MOCO_RNA_motif]	RF01055	585	438	25.1	135.6±18.6	34.6±7.1
PreQ1 riboswitch	RF00522	174	25	85.6	45.8±3.9	7.2±2.8
PreQ1-II riboswitch	RF01054	99	61	38.4	108.9±14.4	21.6±5.1
Purine riboswitch	RF00167	1001	369	63.1	99.9±4.1	19.3±4.2
S-adenosyl methionine (SAM) riboswitch [SAM-IV]	RF00634	354	293	17.2	115.4±9.6	34.7±5.3
S-adenosyl-L-homocysteine riboswitch [SAH_riboswitch]	RF01057	362	295	18.5	94.3±19.8	25.6±7.6
SAM riboswitch (alpha-proteobacteria) [SAM_alpha]	RF00521	343	261	23.9	77.7±3.4	19.4±4.8
SAM riboswitch (S box leader)	RF00162	2076	944	54.5	111.2±16.8	29.5±5.2
SAM-I/IV variant riboswitch [SAM-I-IV-variant]	RF01725	602	394	34.6	101.9±13.2	27.0±5.3
SAM-V riboswitch [SAM_V]	RF01826	559	124	77.8	65.1±4.0	14.6±3.0
SAM/SAH riboswitch [SAM-SAH]	RF01727	76	19	75.0	48.8±1.3	12.1±2.5
SMK box translational riboswitch	RF01767	70	50	28.6	89.1±10.3	19.5±4.1
THF riboswitch	RF01831	294	201	31.6	104.6±18.5	27.3±6.1
TPP riboswitch (THI element)	RF00059	6383	3576	44.0	105.9±16.1	26.9±6.6
ykkC-yxkD leader	RF00442	352	230	34.7	107.5±12.0	28.6±5.8
yybP-ykoY leader	RF00080	450	338	24.9	154.5±34.4	35.0±11.7

6.4.2.2. Identificación de estructuras comunes entre familias de riboswitches

En la sección anterior se mostró que todas las familias consideradas contienen cadenas estructurales redundantes, lo cuál pudiera explicarse con la conservación estructural esperada en este tipo de moléculas. A pesar de estar catalogadas como familias diferentes, no hay que perder de vista que el conjunto de datos pertenece a

la superfamilia de riboswitches. Debido a que su función está íntimamente ligada a su estructura (Poot *et al.*, 1997; Ilyinskii *et al.*, 2009), resulta interesante el explorar las coincidencias de cadenas estructurales entre las diferentes familias. Con este fin, se generó un mapa de calor de la matriz de comparación de estructuras compartidas entre los posibles pares de familias de riboswitches considerados (Figura 37). En la matriz, la celda $M_{i,j}$ contiene el porcentaje de cadenas estructurales distintas de la familia f_i que se encuentran también en la familia f_j. Formalmente,

$$M_{i,j} = \frac{|SS_i \bigcap SS_j|}{|SS_i|} \times 100, \tag{43}$$

donde SS_k es el conjunto de cadenas estructurales pertenecientes a la familia f_k. Trivialmente $M_{i,i} = 100$, por lo que la diagonal se omitió en la visualización presentada.

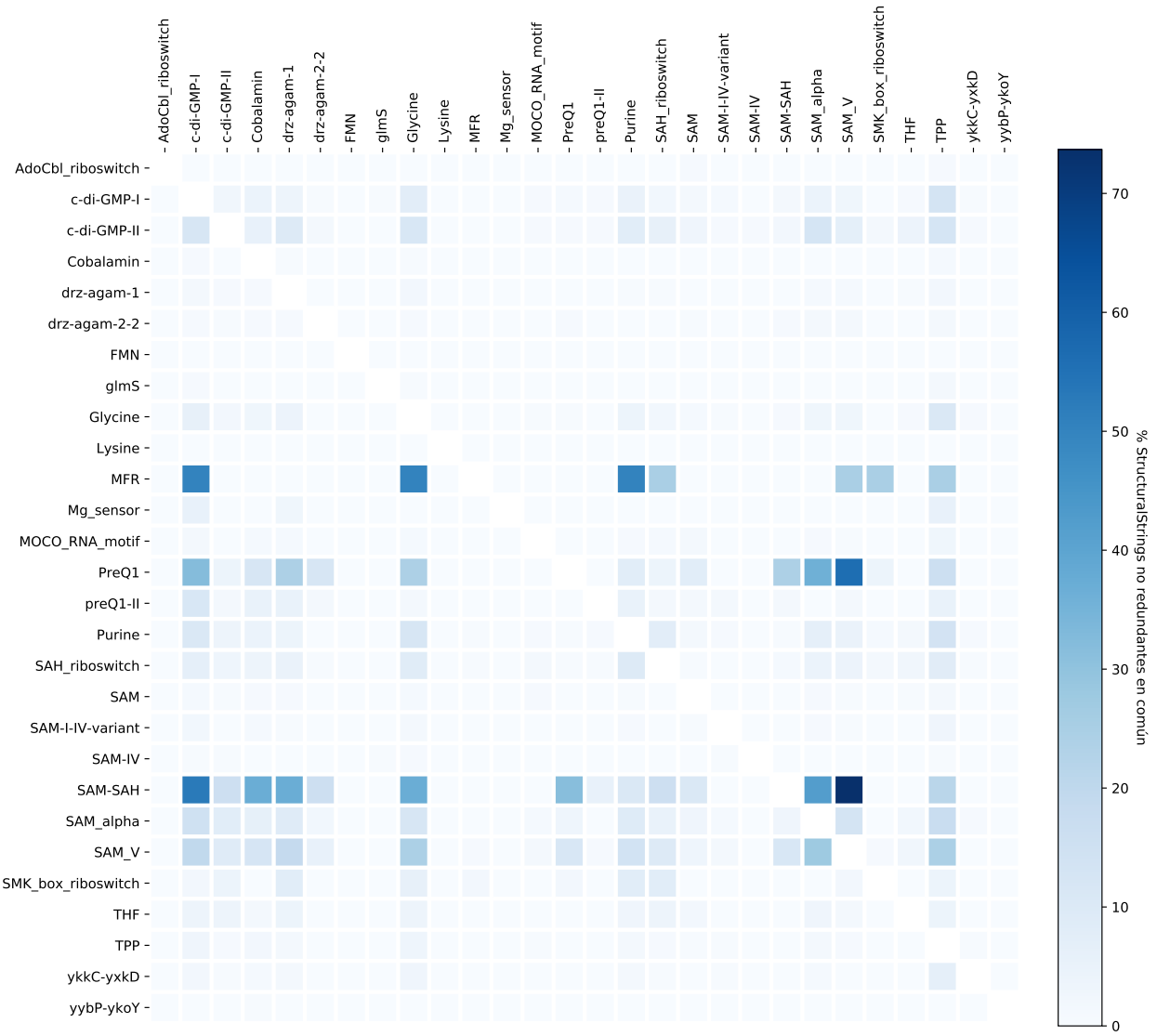

Figura 37: `StructuralStrings` no redundantes en común entre las familias de riboswitches. La diagonal se omitió debido a que tiene un valor de 100 % para cualquier familia.

Figura 38: `StructuralStrings` no redundantes en común entre el subconjunto de familias de riboswitches seleccionado. La diagonal muestra el número de `StructuralStrings` distintas contenidas en la familia.

En la Figura 37 es fácil identificar 10 familias tales que comparten muy pocas cadenas con las otras familias: AdoCbl_riboswitch, FMN, glmS, Lysine, Mg_sensor, MOCO_RNA_motif, SAM-I-IV-variant, SAM-IV, ykkC-yxkD y yybP-ykoY. Por el contrario, también es posible observar celdas que muestran un alto porcentaje de cadenas estructurales compartidas. Para analizar mejor estos casos, seleccionamos 13 de estas familias y generamos el mapa de calor anotado (Figura 38). Para facilitar la comparación, la diagonal muestra el número de estructuras no redundantes de la familia correspondiente.

Observe que la mayor coincidencia de cadenas se encuentra entre SAM-SAH con SAM_V (73.7 %) y PreQ1 con SAM_V (56 %). En general, SAM-SAH comparte muchas de sus cadenas con las otras familias seleccionadas a excepción de MFR, donde esta última tiene la mitad de sus cadenas estructurales no redundantes contenidas en c-di-GMP-I, Glycine y Purine.

Consideramos que el analizar las cadenas estructurales compartidas incluyendo aquellas redundantes, es de vital importancia si se llegasen a emplear solamente características derivadas de esta representación en la tarea de clasificación, puesto que el nivel de redundancia inter-familia pudiera estimar un límite inferior del error irreducible (Friedman *et al.*, 2001).

Con este objetivo, se obtuvo la matriz M de porcentajes de cadenas compartidas (incluyendo las redundantes). La celda $M_{i,j}$ contiene el porcentaje de cadenas de la familia f_i que se encuentran también en la familia f_j. En otras palabras:

$$M_{i,j} = \frac{100}{n_i} \left(\sum_{s \in (SS_i \bigcap SS_j)} m^i(s) \right), \tag{44}$$

donde SS_k es el conjunto de cadenas estructurales (no redundante) pertenecientes a la familia f_k, la función $m^k(s)$ devuelve la multiplicidad de la cadena s en la familia f_k y $n_k = \sum_{s \in SS_k} m^k(s)$ es el número total de cadenas estructurales de la familia f_k (contando duplicados). Trivialmente, $M_{i,i} = 100$, por lo que la diagonal se omite en la visualización presentada. La Figura 39 muestra la matriz de comparación de estructuras compartidas entre los pares posibles de las familias de riboswitches consideradas.

Figura 39: `StructuralStrings` en común entre las familias de riboswitches. La diagonal se omitió debido a que tiene un valor de 100 % para cualquier familia.

Figura 40: `StructuralStrings` en común entre el subconjunto de familias de riboswitches seleccionado. La diagonal muestra el número de secuencias contenidas en la familia.

Esta figura resulta bastante similar a la Figura 37, sin embargo hay diferencias sutiles. Por ejemplo, el nuevo máximo de la escala de color es 90 % y resaltan nuevas celdas como Glycine con Cobalamin y PreQ1 con Purine.

Para fines de comparar las Figuras 37-39, se seleccionaron las mismas 13 familias y se generó el mapa de calor de la submatriz correspondiente (Figura 40). Puede apreciarse que existen 11 celdas con un porcentaje de cadenas en común arriba del 50 %, y la mayoría de estas se concentran en la familia SAM-SAH. Los casos más altos de coincidencia son las cadenas de SAM-SAH contenidas en SAM_V (90.8 %), así

como las cadenas de PreQ1 también contenidas en SAM_V (86.2 %). Aunque pareciera haber una relación entre el número de secuencias no redundantes de una familia y en qué medida están contenidas en otras familias, esto depende del contenido de las secuencias consideradas. Los factores que explicarían esto pudieran remontarse hasta el criterio que tomó el experto que determina que una secuencia pertenecía a una familia y no a otra similar estructuralmente. Esta situación es similar a las observaciones de la demostración del (no) conflicto en las funciones QCOV: aunque haya secuencias redundantes (otra manera de ponerlo es que tienen mucha cobertura en una familia), la calidad (qué tanto identifican a una familia sobre las otras) sigue siendo un valor independiente.

Sin embargo, en este caso en particular, observar este alto nivel de cadenas compartidas entre las familias sugiere que, en caso de realizar tareas de clasificación, es necesario considerar otras características adicionales que permitan discriminar entre las clases superpuestas.

6.4.2.3. Identificación de motivos estructurales en las familias de riboswitches

Como se mencionó en la Sección 6.2.3, muchos esfuerzos se han realizado en la minería de motivos estructurales. Con la finalidad de mostrar cómo pudiese ser aplicada la representación propuesta en esta tarea, se ejecutó MEME con los parámetros por defecto sobre las cadenas estructurales de cada familia. Debido a un requerimiento de MEME, no se consideraron las cadenas con una longitud menor a 8. El mejor motivo encontrado para cada familia se muestra en la Tabla 23.

Tabla 23: Mejor motivo por familia encontrado por MEME en las cadenas estructurales. **Motivo:** expresión regular del mejor motivo encontrado. **#SS:** número de secuencias en donde se realizó la búsqueda. **Cobertura:** porcentaje de secuencias donde el motivo tuvo al menos una coincidencia.

Familia	Motivo	#SS	Cobertura	*E-value*	*p-value*
AdoCbl_riboswitch	`[IH][PS][NI]PISHSIP[NI]`	165	86.06	1.0e-349	5.6e-588
Cobalamin	`P[NGI]PISHSIP[PIN]P`	6093	89.25	1.5e-1660	4.6e-3253
FMN	`NPNPISHSIPN`	1014	86.98	3.6e-2372	3.1e-3879
Glycine	`IPISHSIPIP[NI]`	1094	62.16	2.6e-1727	1.5e-2839
Lysine	`PIPISHSIPIP`	957	84.74	9.6e-1758	1.0e-3193
MFR	`NPISHSIP`	4	50.00	1.1e+002	2.3e-001
MOCO_RNA_motif	`P[NI]P[IGN]PISHSIP`	438	87.44	1.3e-1130	2.3e-1739
Mg_sensor	`[HIS][PS][NIP]SHS[NI]P[NI]P[NI][PS][NI]`	37	86.49	3.4e-053	1.6e-093
PreQ1	`PISHSIP[INP]`	18	100.00	7.0e-012	1.9e-024
Purine	`N[SP][HI][PS][NI]SHS[NI]PI`	368	96.47	1.6e-697	1.7e-1093
SAH_riboswitch	`P[IG]PISHSIP[IPG]P`	295	72.20	7.8e-528	1.9e-870
SAM	`P[NIG]PISHSIP[PIN][PG]`	944	84.22	1.1e-1909	2.8e-3159
SAM-I-IV-variant	`[HS][SH][NS]PGPSHSGPPN[SP]H`	394	55.08	4.7e-932	6.1e-1318
SAM-IV	`[NP]P[IG]PISHSIP[IP]`	293	73.38	7.6e-734	4.2e-1113
SAM-SAH	`PISHSIP[GIN]`	19	100.00	7.0e-035	3.2e-050
SAM_V	`[ING]P[IG]SHS[IP]P`	118	88.98	8.9e-222	2.5e-341
SAM_alpha	`IPISHSIPIP[INP]`	261	79.69	2.8e-491	1.3e-765
SMK_box_riboswitch	`N[PS][NI]P[IN]SHS[IN]P[NP]`	50	98.00	1.1e-059	1.7e-108
THF	`[IP]P[IN]P[ING]P[GI]SHS[PI]`	201	84.58	3.7e-421	7.9e-673
TPP	`[IN]PISHSIP[IN]P[IN]`	3576	68.40	1.3e-1593	4.7e-2759
c-di-GMP-I	`PISHSIP[NIG]`	856	88.55	3.8e-1458	3.0e-2545
c-di-GMP-II	`P[IP]SHS[IG]P[IG]P[IP]`	166	99.40	1.9e-341	1.1e-526
drz-agam-1	`IP[IG]PIPISHSI`	2583	85.48	1.0e-1379	2.6e-2732
drz-agam-2-2	`PIPISHSIPIP`	495	81.21	8.2e-1078	7.4e-1807
glmS	`P[IN]PISHSIP[IN]P`	421	94.54	2.9e-1094	2.0e-1722
preQ1-II	`[PH][INS]P[IN]SHS[IN]`	61	100.00	2.7e-090	1.4e-161
ykkC-yxkD	`[PI][IP]SHS[IG]P[IG]P[IP]P`	230	99.13	5.4e-447	2.4e-743
yybP-ykoY	`[NS][SH][HPS][SN]PGSHSP[NSP]`	338	89.64	1.0e-601	2.5e-1086

MEME evalúa los motivos encontrados según varias funciones objetivo. En este caso particular, se utilizaron el *E-value* y el *p-value*. La base de ambas funciones es la significancia estadística del *log likelihood ratio* (LLR) de las ocurrencias del motivo. El *E-value* del motivo es un estimado del número de motivos (con la misma longitud y número de ocurrencias) que pudieran tener igual o mayor LLR si las secuencias de entrada se hubieran generado aleatoriamente según el modelo de fondo. Se utiliza un algoritmo aproximado para estimar la probabilidad del LLR observado (o incluso mejor). El *p-value* se multiplica por un estimado del número de posibles motivos que generen el mismo *E-value*.

En general, todos los motivos presentan *E-value* menor a $7.0e$-12, *p-value* menor a $1.9e$-24 y cobertura mayor a 55.08 %. La única excepción fue la familia MFR (*E-value* = $1.1e$+2, *p-value* = $2.3e$-1 y cobertura = 50.00 %). Esto se debe a que sólo hay 4 cadenas estructurales distintas para dicha familia.

Los motivos se presentan como expresión regular para mostrar las regiones no tan conservadas (entre corchetes). Usualmente estas regiones corresponden a *internal loops*, *bulges* y raramente a *hairpins* (componentes `S` y `H`). Aún así, los motivos muestran una buena aproximación de las regiones conservadas que se presentarían en la estructura consenso.

Para indagar más en esto, se buscaron los motivos resultantes en todas las cadenas estructurales correspondientes a los alineamientos estructurales de las familias contenidos en RFAM utlizando MAST de manera simultánea. Los resultados se muestran en la Figura 41.

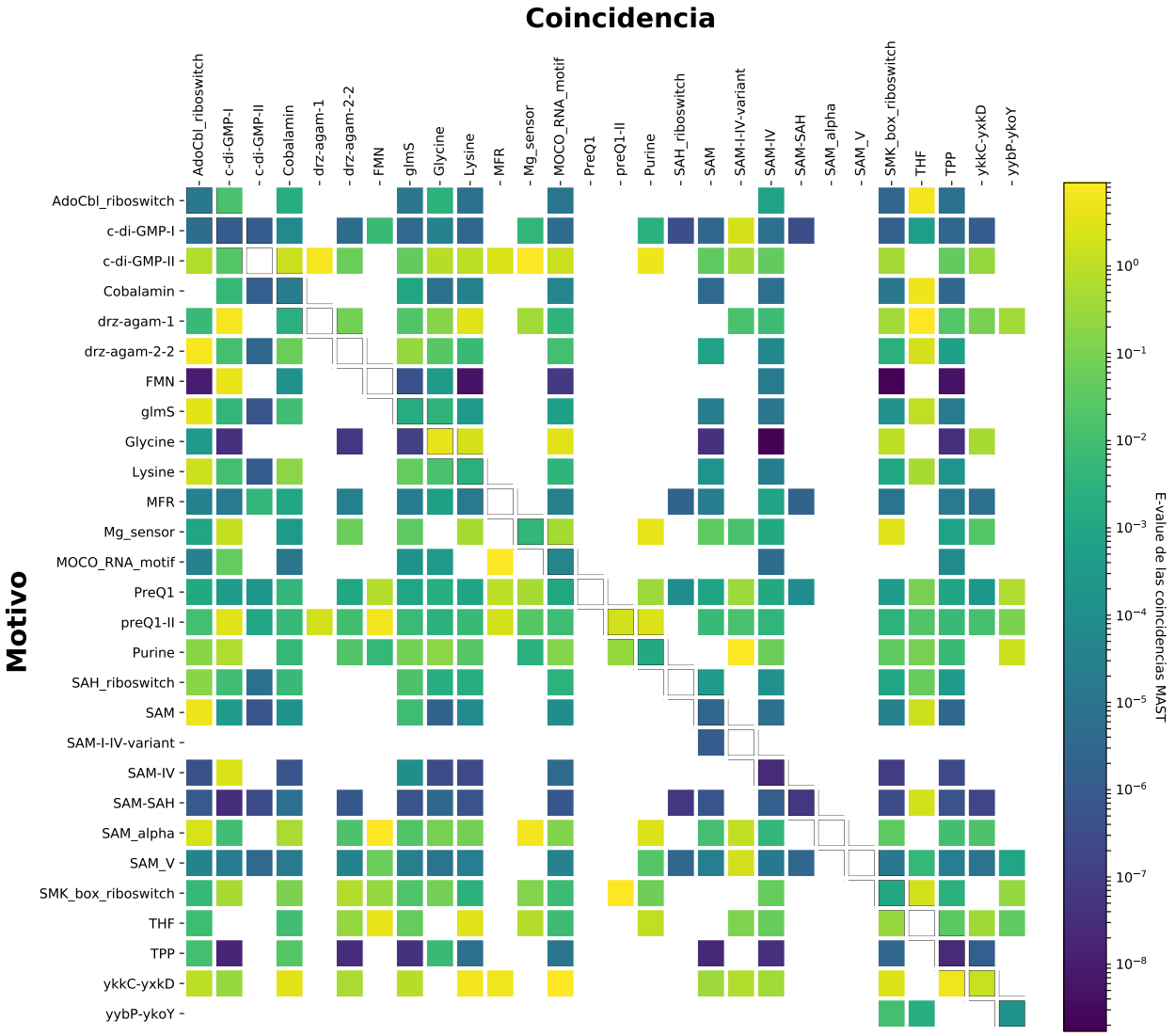

Figura 41: Resultados de la búsqueda de los motivos estructurales representativos de cada familia con MAST (Bailey y Gribskov, 1998) en las estructuras consenso. El color muestra el valor del *E*-value en escala logarítmica. Las celdas en blanco representan que no hubo una coincidencia para esa familia en particular. La diagonal se resalta para mostrar las coincidencias con la estructura consenso.

Puede observarse que muchos motivos fueron encontrados en otras familias diferentes a la objetivo. Esto se debe a que MEME evalúa los motivos con el LLR y no con el poder discriminante de los motivos en otras secuencias. Por otro lado, en 17 de las 28 familias hubo al menos una coincidencia del motivo encontrado con la estructura consenso de RFAM con un *p-value* superior al umbral por omisión de MAST.

A continuación se muestran algunos ejemplos de coincidencias (*matches*) del mo-

tivo en la cadena estructural consenso de la familia donde fue obtenido (figuras 42 y 43).

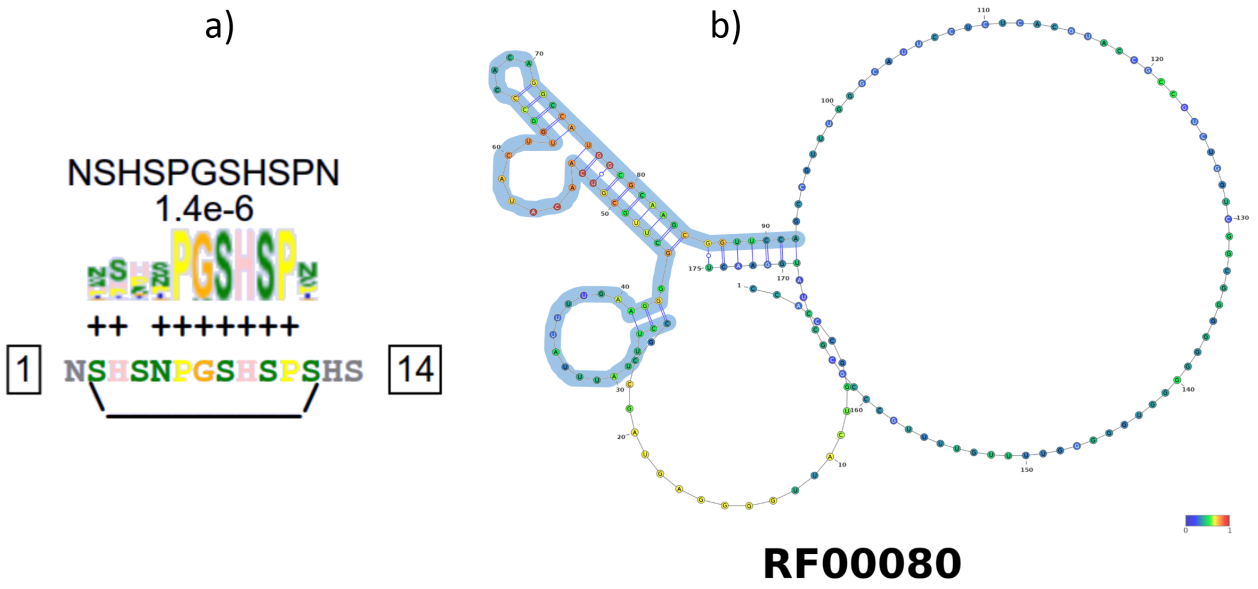

Figura 42: Coincidencia del mejor motivo encontrado por MEME en la estructura consenso predicha por RFAM para la familia yybP-ykoY (RF00080). a) Alineamiento del motivo con la cadena estructural consenso. b) Resaltado del motivo en la estructura consenso.

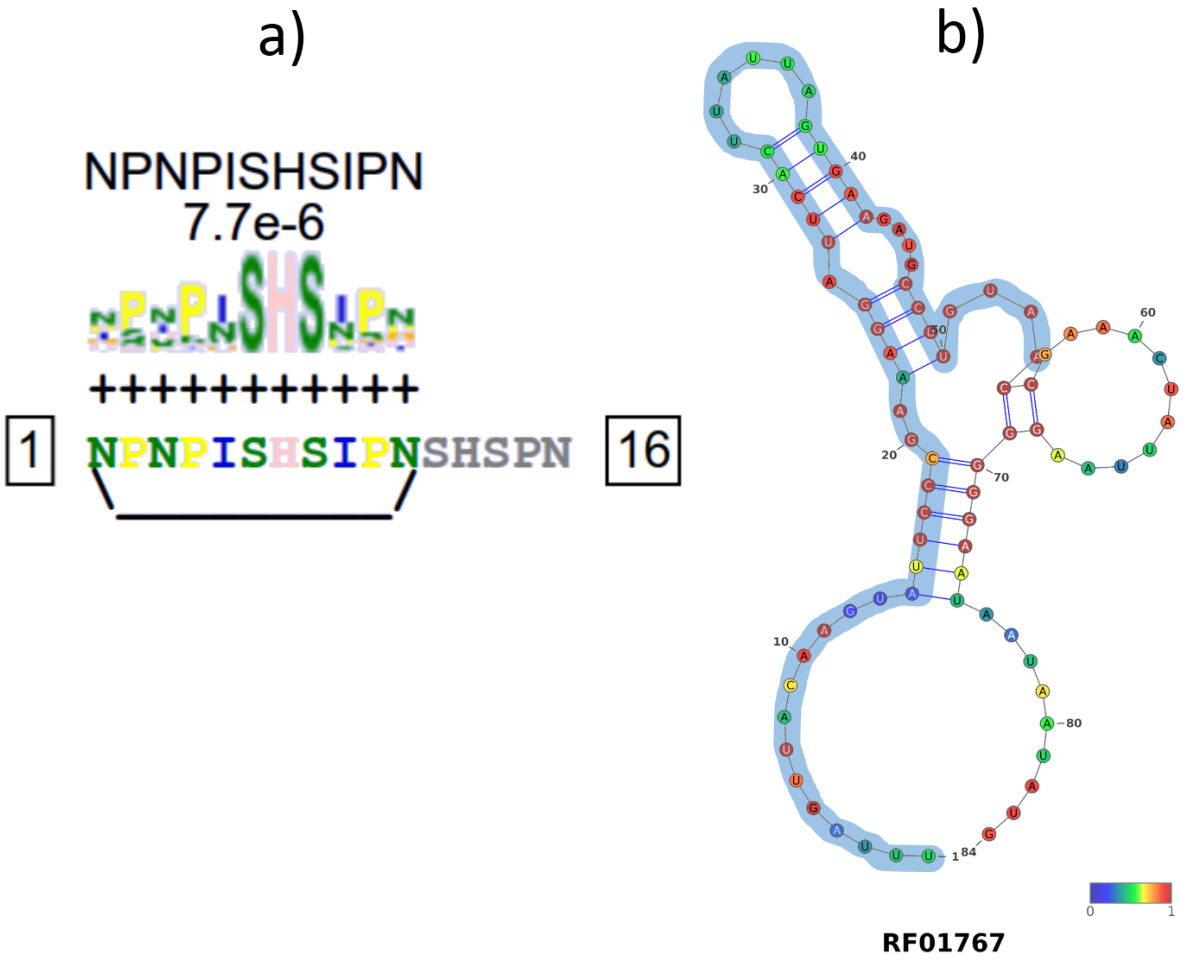

Figura 43: Coincidencia del mejor motivo encontrado por MEME en la estructura consenso predicha por RFAM para la familia SMK_box_riboswitch (RF01767). a) Alineamiento del motivo con la cadena estructural consenso. b) Resaltado del motivo en la estructura consenso.

El motivo encontrado para yybP-ykoY fue el que menor número de coincidencias presentó en las otras familias y se encontró solamente en SMK_box_riboswitch y THF. Puede observarse como el motivo encontrado para la familia SMK_box_riboswitch abarca casi toda la estructura consenso.

6.5. Conclusiones

En este capítulo se presentó un panorama general de las representaciones alternativas de la estructura secundaria del ARN, así como un framework que pudiera servir de base para la generalización de éstas. Se proponen algunas representaciones, en particular el StructuralString, el cual permite utilizar métodos bioinformáticos a nivel

de secuencia para la búsqueda de motivos en estructuras de ARN. En general, la representación permite identificar de manera rápida clases de secuencias que pudieran tener una alta similitud a nivel de estructura secundaria en el conjunto de *riboswitches* considerado. Si bien es a nivel de abstracción, representa una primer etapa de filtrado de resultados para realizar análisis más profundos.

Respecto a la identificación de motivos, las pruebas se realizaron por familia, por lo que no mostraron ninguna capacidad discriminante. Sería interesante utilizar métodos dedicados a la búsqueda de motivos discriminantes y analizar el potencial de los resultados que arrojen como características para problemas de clasificación de ncRNA.

Capítulo 7. Conclusiones

En este trabajo de investigación se aborda el problema de la predicción de ARN no codificante, el cual forma parte del proceso de anotación genómica. En general, dicho proceso es extenso y con varios niveles, dentro de los cuales se encuentra la distinción entre clases similares (clasificación inter-familia). En esta tesis se aborda diversas aristas de este problema de clasificación *fine-grained*. En este capítulo se presenta un resumen con los puntos más relevantes de la tesis. Posteriormente, se enuncian las conclusiones finales del estudio y las propuestas de trabajo a futuro.

7.1. Sumario

Dentro del marco de un flujo de trabajo de clasificación típico, los puntos desarrollados en esta tesis se resumen de la siguiente forma:

- **Ingeniería de características**: en primer lugar, se propusieron conjuntos de características basados en secuencia para la clasificación de *riboswitches* y ARN inmunomodulador. En segundo, en el contexto de utilizar frecuencias *k*-méricas como características, se describen las funciones q y cov así como tres estrategias de selección basadas en estos valores. En tercero, se describe el *framework* NR-PR, el cual permite generar representaciones alternativas de la estructura secundaria del ARN con la finalidad de potencialmente extraer nuevos conjuntos de características.
- **Visualización del conjunto de datos**: se propusieron tanto el covMap como el QCOVGraph, herramientas que ayudan a la identificación de frecuencias *k*-méricas como características discriminantes a simple vista.
- **Clasificación y evaluación**: se clasificaron dos conjuntos de secuencias de ARN no codificante (*riboswitches* e inmunomoduladores) utilizando los conjuntos de características propuestos. Al comparar con los trabajos previos, se obtuvieron resultados de clasificación superiores.

7.2. Conclusiones finales

7.2.1. Elección del nivel estructural de las características

La elección del nivel estructural del que se extraigan las características para la clasificación dependerá de la tarea principal que se esté resolviendo. Por ejemplo, si la clasificación es inter-familia entre un número pequeño de secuencias putativas, el costo computacional añadido por la extracción de características pudiera ser despreciable. Sin embargo, si el clasificador toma secuencias provenientes de la búsqueda por ventanas superpuestas de un genoma completo o todos los *reads* provenientes de un experimento RNA-seq, usar la estructura secundaria conlleva un alto costo computacional que pudiera convertir el enfoque prohibitivo en el tiempo. Por ello, consideramos que es necesario examinar las propiedades de las secuencias a nivel intra e inter-familia. El enfoque considerado en este trabajo fue iniciar la clasificación con frecuencias *k*-méricas, y sólo si no hay un nivel de discriminación inter-familia suficiente, entonces emplear las características computacionalmente más costosas. Por otro lado, si a nivel intra-familia no hay conservación a nivel primario de las secuencias, entonces conviene considerar las características estructurales.

Si el enfoque es totalmente centrado en estructura, el paso lógico sería entonces preparar los casos de prueba a ciertos niveles de porcentaje de identidad a nivel de secuencia, incluso evaluar el conjunto utilizando la búsqueda BLAST como clasificador y poder descartar a las características a nivel de secuencia como el factor determinante en el desempeño de la clasificación y evaluar solamente el poder discriminante de la representación o características extraídas de la estructura. Sin embargo, esta decisión de diseño sólo se encontró en el caso de prueba de Infernal.

7.2.2. Datos de entrenamiento, casos de prueba y reproducibilidad

RFAM es el origen de las secuencias en la mayoría de los trabajos de clasificación de ARN no codificante. No obstante, no todas las secuencias son validadas experimentalmente y muchas de ellas son putativas. Más aún, en cada versión de la base de datos se cambian tanto el número de familias (pueden eliminarse, fusionarse o crear-

se), por lo que datos previamente etiquetados como de una familia pueden ya no ser correctos, o incluso la secuencia en sí misma puede haber cambiado al modificarse la locación de la anotación en el genoma. Esto conlleva un problema de consistencia en los datos de entrenamiento que no ha sido discutido debidamente en la literatura previa. Una posible solución a esta situación sería limitar el entrenamiento a datos validados experimentalmente. Sin embargo, para poner el problema en perspectiva, en una búsqueda en la versión actual del *Protein Data Bank* (PDB) sólo se encontraron 256 estructuras reportadas de *riboswitches*, lo cual representa tan solo el 2.63 % del conjunto de entrenamiento descrito en el Capítulo 3 o el 0.76 % del conjunto utilizado en el Capítulo 6. Específicamente para el Capítulo 3, el conjunto de secuencias fue seleccionado según el estado del arte con fines de comparación justa. Sin embargo, una recomendación derivada del problema mencionado anteriormente consiste en actualizar incrementalmente los modelos de clasificación desarrollados con cada iteración de RFAM.

Lamentablemente, la dificultad de reproducir los resultados de la literatura fue constante durante el periodo del estudio. Este problema se presentó en diversas situaciones:

- Bases de datos. Las bases de datos ya no se encontraban disponibles para descargar en la página web de los autores o el formato cambiaba drásticamente de una versión a otra (esto último ocurrió con NONCODE y la misma RFAM).
- Código y configuraciones. La mayoría del código no se encontraba disponible para ejecutar y verificar resultados. En algunos casos, las configuraciones para ejecutar software existente no se especificaban; por ejemplo, umbrales de selección, tipos y parámetros del kernel de una SVM y semillas aleatorias para métodos que así lo requerían.
- Conjuntos de datos. En general, no hay *benchmarks* aceptados por la comunidad que validen el avance en los algoritmos de predicción/clasificación, por lo que en la mayoría de los casos es el autor el que propone el propio. Si bien en la mayoría de los trabajos analizados se obtienen las secuencias de bases de datos públicas, el flujo de trabajo para obtener el conjunto final no está disponible y mucho

menos las secuencias resultantes. Esto es particularmente importuno cuando se especifican pasos como “X secuencias fueron seleccionadas aleatoriamente para balancear las clases”.

7.3. Trabajo a futuro

Dado lo cambiante que es la base de datos RFAM, convendría actualizar el modelo de clasificación de *riboswitches* con cada nueva versión. Además, resultaría interesante evaluar la clasificación incorporando características a nivel secundario. En este sentido, conviene extraer características generales y específicas a las representaciones alternativas propuestas y comparar con la literatura en clasificación a nivel meramente estructural. A su vez, podría realizarse el aprendizaje con las instancias tipo grafo directamente.

Otro trabajo por realizar es la comparación de la selección de frecuencias k-méricas usando las funciones q y cov con otros selectores de características. Asimismo, este framework se extiende naturalmente a otras secuencias biológicas como proteínas, por lo que es de interés probarlo en ese contexto.

Otra línea de trabajo consiste en enunciar las representaciones basadas en grafos existentes usando el framework `NR-PR` y proponer casos de prueba para evaluar diversas propiedades como la pérdida de información y la precisión al comparar dos o más estructuras utilizando características intrínsecas a la representación.

A largo plazo, la idea es integrar este trabajo en un enfoque más integral que considere el descubrimiento *de novo* en genomas completos.

Literatura citada

Abreu-Goodger, C. y Merino, E. (2005). Ribex: a web server for locating riboswitches and other conserved bacterial regulatory elements. *Nucleic acids research*, **33**(suppl 2): W690–W692.

Aghdam, E. M., Hejazi, M. S., y Barzegar, A. (2016). Riboswitches: From living biosensors to novel targets of antibiotics. *Gene*, **592**(2): 244 – 259.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., y Watson, J. (2014). *Molecular Biology of the Cell*. Garland, 6mo edición.

Alioto, T. (2012). Gene prediction. En: *Evolutionary Genomics*. Springer, pp. 175–201.

Allali, J. y Sagot, M.-F. (2005). A multiple graph layers model with application to RNA secondary structures comparison. En: *String Processing and Information Retrieval*. Springer, pp. 348–359.

Allali, J., Saule, C., Chauve, C., dŔAubenton Carafa, Y., Denise, A., Drevet, C., Ferraro, P., Gautheret, D., Herrbach, C., Leclerc, F., *et al.* (2012). Brasero: A resource for benchmarking RNA secondary structure comparison algorithms. *Advances in bioinformatics*, **2012**.

Andronescu, M., Bereg, V., Hoos, H. H., y Condon, A. (2008). RNA strand: the RNA secondary structure and statistical analysis database. *BMC bioinformatics*, **9**(1): 340.

Arrial, R. T., Togawa, R. C., y Brigido, M. M. (2009). Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus paracoccidioides brasiliensis. *BMC bioinformatics*, **10**(1): 239.

Arruda, W., Ralha, C. G., Raiol, T., Brígido, M. M., Walter, M. E. M., y Stadler, P. F. (2013). ncRNA-Agents: A multiagent system for non-coding RNA annotation. En: *Advances in Bioinformatics and Computational Biology*. Springer, pp. 136–147.

Bailey, T. y Elkan, C. (1994). Fitting a mixture model by expectation maximization to discover motifs in biopolymers. En: *Proceedings. International Conference on Intelligent Systems for Molecular Biology*. Vol. 2, p. 28.

Bailey, T. L. y Gribskov, M. (1998). Combining evidence using p-values: application to sequence homology searches. *Bioinformatics (Oxford, England)*, **14**(1): 48–54.

Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., Ren, J., Li, W. W., y Noble, W. S. (2009). Meme suite: tools for motif discovery and searching. *Nucleic acids research*, **37**(suppl_2): W202–W208.

Bakhtin, Y. y Heitsch, C. E. (2009). Large deviations for random trees and the branching of RNA secondary structures. *Bulletin of mathematical biology*, **71**(1): 84–106.

Bandyopadhyay, S., Maulik, U., y Roy, D. (2008). Gene identification: classical and computational intelligence approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **38**(1): 55–68.

Beltrán Verdugo, J. (2014). *Métodos para la selección de características y clasificación de péptidos antimicrobianos*. Tesis de maestría, Centro de Investigación Científica y de Educación Superior de Ensenada.

Benedetti, G. y Morosetti, S. (1996). A graph-topological approach to recognition of pattern and similarity in RNA secondary structures. *Biophysical chemistry*, **59**(1): 179–184.

Bengert, P. y Dandekar, T. (2004). Riboswitch finder - wa tool for identification of riboswitch RNAs. *Nucleic acids research*, **32**(suppl 2): W154–W159.

Berens, C. y Suess, B. (2015). Riboswitch engineering - w making the all-important second and third steps. *Current Opinion in Biotechnology*, **31**: 10 – 15. Analytical Biotechnology.

Blount, K. F. y Breaker, R. R. (2006). Riboswitches as antibacterial drug targets. *Nature biotechnology*, **24**(12): 1558–1564.

Bon, M., Vernizzi, G., Orland, H., y Zee, A. (2008). Topological classification of RNA structures. *Journal of molecular biology*, **379**(4): 900–911.

Boser, B. E., Guyon, I. M., y Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. En: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, pp. 144–152.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. En: *Proceedings of COMPSTAT'2010*. Springer, pp. 177–186.

Breaker, R. R. (2012). Riboswitches and the RNA world. *Cold Spring Harbor perspectives in biology*, **4**(2): a003566.

Breiman, L. (2001). Random forests. *Machine learning*, **45**(1): 5–32.

Brosius, J. y Gould, S. J. (1992). On "genomenclature": a comprehensive (and respectful) taxonomy for pseudogenes and other "junk DNA". *Proceedings of the National Academy of Sciences*, **89**(22): 10706–10710.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., y Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, **18**(4): 467–479.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., y Madden, T. L. (2009). Blast+: architecture and applications. *BMC bioinformatics*, **10**(1): 421.

Cary, R. B. y Stormo, G. D. (1995). Graph-theoretic approach to RNA modeling using comparative data. En: *ISMB*. Vol. 95, pp. 75–80.

Chang, C.-C. y Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, **2**(3): 27.

Chang, T.-H., Huang, H.-D., Wu, L.-C., Yeh, C.-T., Liu, B.-J., y Horng, J.-T. (2009). Computational identification of riboswitches based on RNA conserved functional sequences and conformations. *Rna*, **15**(7): 1426–1430.

Chang, T.-H., Huang, H.-Y., Hsu, J. B.-K., Weng, S.-L., Horng, J.-T., y Huang, H.-D. (2013). An enhanced computational platform for investigating the roles of regulatory RNA and for identifying functional RNA motifs. *BMC bioinformatics*, **14**(2): 1.

Chaudhary, K., Nagpal, G., Dhanda, S. K., y Raghava, G. P. (2016). Prediction of immunomodulatory potential of an RNA sequence for designing non-toxic siRNAs and RNA-based vaccine adjuvants. *Scientific reports*, **6**: 20678.

Chen, C., Ridzon, D. A., Broomer, A. J., Zhou, Z., Lee, D. H., Nguyen, J. T., Barbisin, M., Xu, N. L., Mahuvakar, V. R., Andersen, M. R., *et al.* (2005). Real-time quantification of microRNAs by stem–loop rt–pcr. *Nucleic acids research*, **33**(20): e179–e179.

Chen, H.-L., Condon, A., y Jabbari, H. (2009). An $O(n^5)$ algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *Journal of Computational Biology*, **16**(6): 803–815.

Childs, L., Nikoloski, Z., May, P., y Walther, D. (2009). Identification and classification of ncRNA molecules using graph properties. *Nucleic acids research*, **37**(9): e66–e66.

Chiu, J. K. H. y Chen, Y.-P. P. (2012). Conformational features of topologically classified RNA secondary structures. *PloS one*, **7**(7): e39907.

Churkin, A., Gabdank, I., y Barash, D. (2012). On topological indices for small RNA graphs. *Computational biology and chemistry*, **41**: 35–40.

Clancy, S. (2008). RNA functions. *Nature Education*, **1**(1): 102.

Clote, P., Ferré, F., Kranakis, E., y Krizanc, D. (2005). Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA*, **11**(5): 578–591.

Corbino, K. A., Barrick, J. E., Lim, J., Welz, R., Tucker, B. J., Puskarz, I., Mandal, M., Rudnick, N. D., y Breaker, R. R. (2005). Evidence for a second class of s-adenosylmethionine riboswitches and other regulatory RNA motifs in alpha-proteobacteria. *Genome biology*, **6**(8): R70.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., y Stein, C. (2009). *Introduction to algorithms*. MIT press.

Danger, R., Segura-Bedmar, I., Martínez, P., y Rosso, P. (2010). A comparison of machine learning techniques for detection of drug target articles. *Journal of biomedical informatics*, **43**(6): 902–913.

Dann III, C. E., Wakeman, C. A., Sieling, C. L., Baker, S. C., Irnov, I., y Winkler, W. C. (2007). Structure and mechanism of a metal-sensing regulatory RNA. *Cell*, **130**(5): 878–892.

Darty, K., Denise, A., y Ponty, Y. (2009). VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, **25**(15): 1974–5.

Ding, Y. y Lawrence, C. E. (2003). A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic acids research*, **31**(24): 7280–7301.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, **55**(10): 78–87.

Eddy, S. R. (2001). Non–coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, **2**(12): 919–929.

Eddy, S. R. (2002). A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC bioinformatics*, **3**(1): 18.

Eddy, S. R. y Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucleic acids research*, **22**(11): 2079–2088.

Edwards, T. E. y Ferré-D'Amaré, A. R. (2006). Crystal structures of the thi-box riboswitch bound to thiamine pyrophosphate analogs reveal adaptive RNA-small molecule recognition. *Structure*, **14**(9): 1459–1468.

Eichhorn, C. D., Kang, M., y Feigon, J. (2014). Structure and function of preq 1 riboswitches. *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms*, **1839**(10): 939–950.

Fernández-Delgado, M., Cernadas, E., Barro, S., y Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, **15**(1): 3133–3181.

Freyhult, E. K., Bollback, J. P., y Gardner, P. P. (2007). Exploring genomic dark matter: a critical assessment of the performance of homology search methods on noncoding RNA. *Genome research*, **17**(1): 117–125.

Friedman, J., Hastie, T., y Tibshirani, R. (2001). *The elements of statistical learning*, Vol. 1. Springer series in statistics New York, NY, USA.

Fürtig, B., Nozinovic, S., Reining, A., y Schwalbe, H. (2015). Multiple conformational states of riboswitches fine-tune gene regulation. *Current Opinion in Structural Biology*, **30**: 112 – 124. Folding and binding/Nucleic acids and their protein complexes.

Gan, H. H., Fera, D., Zorn, J., Shiffeldrim, N., Tang, M., Laserson, U., Kim, N., y Schlick, T. (1987). Rag: RNA-as-graphs database—concepts, analysis, and features. *Nutrition and Health*, **5**(1-2): 1285–1291.

Gan, H. H., Pasquali, S., y Schlick, T. (2003). Exploring the repertoire of RNA secondary motifs using graph theory; implications for RNA design. *Nucleic acids research*, **31**(11): 2926–2943.

Gardner, P. P. y Eldai, H. (2014). Annotating RNA motifs in sequences and alignments. *Nucleic acids research*, p. gku1327.

Gawronski, A. R. y Turcotte, M. (2014). Ribofsm: Frequent subgraph mining for the discovery of RNA structures and interactions. *BMC bioinformatics*, **15**(Suppl 13): S2.

Gelfand, M. S., Mironov, A. A., y Pevzner, P. A. (1996). Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Sciences*, **93**(17): 9061–9066.

Gorodkin, J. y Hofacker, I. L. (2011a). From structure prediction to genomic screens for novel non-coding RNAs. *PLoS computational biology*, **7**(8): e1002100.

Gorodkin, J. y Hofacker, I. L. (2011b). From structure prediction to genomic screens for novel non-coding RNAs. *PLoS computational biology*, **7**(8): e1002100.

Gorodkin, J., Hofacker, I. L., Torarinsson, E., Yao, Z., Havgaard, J. H., y Ruzzo, W. L. (2010). De novo prediction of structured RNAs from genomic sequences. *Trends in biotechnology*, **28**(1): 9–19.

Gorodkin, J., Hofacker, I. L., y Ruzzo, W. L. (2014). Concepts and introduction to RNA bioinformatics. En: *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*. Springer, pp. 1–31.

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., y Eddy, S. R. (2003). Rfam: an RNA family database. *Nucleic acids research*, **31**(1): 439–441.

Griffiths-Jones, S., Grocock, R. J., Van Dongen, S., Bateman, A., y Enright, A. J. (2006). miRBase: microRNA sequences, targets and gene nomenclature. *Nucleic acids research*, **34**(suppl 1): D140–D144.

Gudyś, A., Szcześniak, M. W., Sikora, M., y Makałowska, I. (2013). Huntmi: an efficient and taxon-specific approach in pre-miRNA identification. *BMC bioinformatics*, **14**(1): 1.

Guillen-Ramirez, H. A. y Martinez-Perez, I. M. (2014). Retos en la predicción de ARN no codificante. En: *Memorias del Encuentro Nacional de Ciencias de la Computación, ENC 2014*. SMCC, pp. 1–4.

Guillen-Ramirez, H. A., Colbes, J., Brizuela, C. A., y Martinez-Perez, I. M. (2017). Accurate classification of immunomodulatory RNA sequences. En: *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, pp. 236–241.

Guillén-Ramírez, H. A. y Martínez-Pérez, I. M. (2018). Classification of riboswitch sequences using k-mer frequencies. *Biosystems*, pp. 1–14.

Hagberg, A., Swart, P., y S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Reporte técnico, Los Alamos National Lab.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., y Witten, I. H. (2009a). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, **11**(1): 10–18.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., y Witten, I. H. (2009b). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, **11**(1): 10–18.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. Tesis de doctorado, The University of Waikato.

Hamada, M., Tsuda, K., Kudo, T., Kin, T., y Asai, K. (2006). Mining frequent stem patterns from unaligned RNA sequences. *Bioinformatics*, **22**(20): 2480–2487.

Havill, J. T., Bhatiya, C., Johnson, S. M., Sheets, J. D., y Thompson, J. S. (2014). A new approach for detecting riboswitches in DNA sequences. *Bioinformatics*, **30**(21): 3012–3019.

He, H. y Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, **21**(9): 1263–1284.

Heyne, S., Costa, F., Rose, D., y Backofen, R. (2012). Graphclust: alignment-free structural clustering of local RNA secondary structures. *Bioinformatics*, **28**(12): i224–i232.

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., y Schuster, P. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly*, **125**(2): 167–188.

Hofacker, I. L., Fekete, M., y Stadler, P. F. (2002). Secondary structure prediction for aligned RNA sequences. *Journal of molecular biology*, **319**(5): 1059–1066.

Hofacker, I. L., Bernhart, S. H., y Stadler, P. F. (2004). Alignment of RNA base pairing probability matrices. *Bioinformatics*, **20**(14): 2222–2227.

Holmes, I. y Rubin, G. (2002). Pairwise RNA structure comparison with stochastic context-free grammars. En: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*. p. 163.

Hopcroft, J. E. y Ullman, J. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Howe, J. A., Wang, H., Fischmann, T. O., Balibar, C. J., Xiao, L., Galgoci, A. M., Malinverni, J. C., Mayhood, T., Villafania, A., Nahvi, A., *et al.* (2015). Selective small-molecule inhibition of an RNA structural element. *Nature*.

Hua, J., Xiong, Z., Lowey, J., Suh, E., y Dougherty, E. R. (2004). Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, **21**(8): 1509–1515.

Huang, H.-Y., Chien, C.-H., Jen, K.-H., y Huang, H.-D. (2006). Regrna: an integrated web server for identifying regulatory RNA motifs and elements. *Nucleic acids research*, **34**(suppl 2): W429–W434.

Hung, T. y Chang, H. Y. (2010). Long noncoding RNA in genome regulation: prospects and mechanisms. *RNA biology*, **7**(5): 582–585.

Ilyinskii, P. O., Schmidt, T., Lukashev, D., Meriin, A. B., Thoidis, G., Frishman, D., y Shneider, A. M. (2009). Importance of mRNA secondary structural elements for the expression of influenza virus genes. *OMICS A Journal of Integrative Biology*, **13**(5): 421–430.

Jabbari, H., Wark, I., Montemagno, C., y Will, S. (2018). Knotty: Efficient and accurate prediction of complex RNA pseudoknot structures. *Bioinformatics*, **1**: 8.

Janssen, S., Reeder, J., y Giegerich, R. (2008). Shape based indexing for faster search of RNA family databases. *BMC bioinformatics*, **9**(1): 131.

Japkowicz, N. y Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, **6**(5): 429–449.

John, G. H. y Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. En: *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 338–345.

Kalapanidas, E., Avouris, N., Craciun, M., y Neagu, D. (2003a). Machine learning algorithms: a study on noise sensitivity. En: *Proc. 1st Balcan Conference in Informatics*. pp. 356–365.

Kalapanidas, E., Avouris, N., Craciun, M., y Neagu, D. (2003b). Machine learning algorithms: A study on noise sensitivity. En: *Proc. 1st Balcan Conference in Informatics*. pp. 356–365.

Kang, M., Peterson, R., y Feigon, J. (2010). Structural insights into riboswitch control of the biosynthesis of queuosine, a modified nucleotide found in the anticodon of trna. *Molecular Cell*, **39**(4): 653–655.

Karklin, Y., Meraz, R. F., y Holbrook, S. R. (2005). Classification of non-coding RNA using graph representations of secondary structure. En: *Pacific symposium on biocomputing*. World Scientific, Vol. 10, pp. 4–15.

Khadikar, P. V., Deshpande, N. V., Kale, P. P., Dobrynin, A., Gutman, I., y Domotor, G. (1995). The szeged index and an analogy with the wiener index. *Journal of chemical information and computer sciences*, **35**(3): 547–550.

Kittler, J., Hatef, M., Duin, R. P., y Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, **20**(3): 226–239.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. En: *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc., pp. 1137–1143.

Kolbe, D. L. y Eddy, S. R. (2009). Local RNA structure alignment with incomplete sequence. *Bioinformatics*, **25**(10): 1236–1243.

Kong, L., Zhang, Y., Ye, Z.-Q., Liu, X.-Q., Zhao, S.-Q., Wei, L., y Gao, G. (2007). Cpc: assess the protein-coding potential of transcripts using sequence features and support vector machine. *Nucleic acids research*, **35**(suppl 2): W345–W349.

Kuncheva, L. (2004). *Combining pattern classifiers: methods and algorithms*. Wiley. John & Sons.

Kwon, M. y Strobel, S. A. (2008). Chemical basis of glycine riboswitch cooperativity. *Rna*, **14**(1): 25–34.

Laborde, J., Srivastava, A., y Zhang, J. (2011). Structure-based RNA function prediction using elastic shape analysis. En: *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*. IEEE, pp. 16–21.

Lagesen, K., Hallin, P., Rødland, E. A., Stærfeldt, H.-H., Rognes, T., y Ussery, D. W. (2007). Rnammer: consistent and rapid annotation of ribosomal RNA genes. *Nucleic acids research*, **35**(9): 3100–3108.

Leontis, N. B. y Westhof, E. (2001). Geometric nomenclature and classification of RNA base pairs. *Rna*, **7**(4): 499–512.

Lesnik, E. A., Fogel, G. B., Weekes, D., Henderson, T. J., Levene, H. B., Sampath, R., y Ecker, D. J. (2005). Identification of conserved regulatory RNA structures in prokaryotic metabolic pathway genes. *Biosystems*, **80**(2): 145–154.

Lestrade, L. y Weber, M. J. (2006). snorna-lbme-db, a comprehensive database of human h/aca and c/d box snornas. *Nucleic acids research*, **34**(suppl 1): D158–D162.

Lewis, B. P., Shih, I.-h., Jones-Rhoades, M. W., Bartel, D. P., Burge, C. B., *et al.* (2003). Prediction of mammalian microRNA targets. *Cell*, **115**(7): 787–798.

Liao, B., Luo, J., Li, R., y Zhu, W. (2006). RNA secondary structure 2d graphical representation without degeneracy. *International journal of quantum chemistry*, **106**(8): 1749–1755.

Liu, C., Bai, B., Skogerbø, G., Cai, L., Deng, W., Zhang, Y., Bu, D., Zhao, Y., y Chen, R. (2005). NONCODE: an integrated knowledge database of non-coding RNAs. *Nucleic acids research*, **33**(suppl 1): D112–D115.

Liu, J., Gough, J., y Rost, B. (2006). Distinguishing protein-coding from non-coding RNAs through support vector machines. *PLoS Genet*, **2**(4): e29.

Lorenz, R., Bernhart, S. H., Siederdissen, Z., H., C., Tafer, H., Flamm, C., Stadler, P. F., y Hofacker, I. L. (2011). ViennaRNA package 2.0. *Algorithms for Molecular Biology*, **6**: 26.

Lowe, T. M. y Eddy, S. R. (1997). trnascan-se: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic acids research*, **25**(5): 0955–964.

Mahé, P. y Vert, J.-P. (2009). Graph kernels based on tree patterns for molecules. *Machine learning*, **75**(1): 3–35.

Maji, S. y Garg, D. (2013). Progress in gene prediction: Principles and challenges. *Current Bioinformatics*, **8**(2): 226–243.

Majoros, W. H. (2007). *Methods for computational gene prediction*, Vol. 1. Cambridge University Press Cambridge.

Mandal, M., Boese, B., Barrick, J. E., Winkler, W. C., y Breaker, R. R. (2003). Riboswitches control fundamental biochemical pathways in bacillus subtilis and other bacteria. *Cell*, **113**(5): 577–586.

Manzourolajdad, A. y Arnold, J. (2015). Secondary structural entropy in RNA switch (riboswitch) identification. *BMC bioinformatics*, **16**(1): 1.

Mathé, C., Sagot, M.-F., Schiex, T., y Rouzé, P. (2002). Current methods of gene prediction, their strengths and weaknesses. *Nucleic acids research*, **30**(19): 4103–4117.

McCaskill, J. (1990a). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**(6-7): 1105.

McCaskill, J. S. (1990b). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**(6-7): 1105–1119.

McCown, P. J., Winkler, W. C., y Breaker, R. R. (2012). Mechanism and distribution of glms ribozymes. En: *Ribozymes*. Springer, pp. 113–129.

Menzel, P., Gorodkin, J., y Stadler, P. F. (2009). The tedious task of finding homologous noncoding RNA genes. *RNA*, **15**(12): 2075–2082.

Mercer, T. R., Dinger, M. E., y Mattick, J. S. (2009). Long non-coding RNAs: insights into functions. *Nature Reviews Genetics*, **10**(3): 155–159.

Meyer, M. M., Roth, A., Chervin, S. M., Garcia, G. A., y Breaker, R. R. (2008). Confirmation of a second natural preq1 aptamer class in streptococcaceae bacteria. *Rna*, **14**(4): 685–695.

Miladi, M., Junge, A., Costa, F., Seemann, S. E., Havgaard, J. H., Gorodkin, J., y Backofen, R. (2017). Rnascclust: clustering RNA sequences using structure conservation and graph based motifs. *Bioinformatics*, **33**(14): 2089–2096.

Milo, N., Zakov, S., Katzenelson, E., Bachmat, E., Dinitz, Y., y Ziv-Ukelson, M. (2012). RNA tree comparisons via unrooted unordered alignments. En: *Algorithms in Bioinformatics*. Springer, pp. 135–148.

Mituyama, T., Yamada, K., Hattori, E., Okida, H., Ono, Y., Terai, G., Yoshizawa, A., Komori, T., y Asai, K. (2009). The functional RNA database 3.0: databases to support mining and annotation of functional RNAs. *Nucleic acids research*, **37**(suppl 1): D89–D92.

Montange, R. K. y Batey, R. T. (2006). Structure of the s-adenosylmethionine riboswitch regulatory mRNA element. *Nature*, **441**(7097): 1172.

Mukherjee, S. y Sengupta, S. (2015). Riboswitch scanner: an efficient phmm-based web-server to detect riboswitches in genomic sequences. *Bioinformatics*, p. btv640.

Nagpal, G., Chaudhary, K., Dhanda, S. K., y Raghava, G. P. S. (2017). Computational prediction of the immunomodulatory potential of RNA sequences. En: *RNA Nanostructures*. Springer, pp. 75–90.

Nawrocki, E. P. y Eddy, S. R. (2013). Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, **29**(22): 2933–2935.

Nawrocki, E. P., Kolbe, D. L., y Eddy, S. R. (2009). Infernal 1.0: inference of RNA alignments. *Bioinformatics*, **25**(10): 1335–1337.

Olejniczak, M., Galka-Marciniak, P., y Krzyzosiak, W. (2010). Sequence-non-specific effects of RNA interference triggers and microRNA regulators. *Nucleic acids research*, **38**(1): 1–16.

Olejniczak, M., Galka-Marciniak, P., Polak, K., Fligier, A., y Krzyzosiak, W. (2012). Rnaimmuno: a database of the nonspecific immunological effects of RNA interference and microRNA reagents. *RNA*, **18**: 930–935.

Ouangraoua, A. y Ferraro, P. (2009). A new constrained edit distance between quotiented ordered trees. *Journal of Discrete Algorithms*, **7**(1): 78–89.

Pang, K. C., Frith, M. C., y Mattick, J. S. (2006). Rapid evolution of noncoding RNAs: lack of conservation does not mean lack of function. *Trends in Genetics*, **22**(1): 1–5.

Pang, K. C., Stephen, S., Dinger, M. E., Engström, P. G., Lenhard, B., y Mattick, J. S. (2007). Rnadb 2.0 – an expanded database of mammalian non-coding RNAs. *Nucleic acids research*, **35**(suppl 1): D178–D182.

Panwar, B., Arora, A., y Raghava, G. P. (2014). Prediction and classification of ncRNAs using structural information. *BMC genomics*, **15**(1): 127.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11): 559–572.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pp. 185–208.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pp. 185–208.

Poiata, E., Meyer, M. M., Ames, T. D., y Breaker, R. R. (2009). A variant riboswitch aptamer class for s-adenosylmethionine common in marine bacteria. *Rna*.

Poot, R. A., Tsareva, N. V., Boni, I. V., y Van Duin, J. (1997). RNA folding kinetics regulates translation of phage MS2 maturation gene. *Proceedings of the National Academy of Sciences*, **94**(19): 10110–10115.

QIAGEN (2011). mirneasy mini kit. recuperado de http://www.qiagen.com/products/catalog/sample-technologies/rna-sample-technologies/mirna/mirneasy-mini-kit.

Quinlan, J. R. (1993). C 4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning, San Mateo, CA*, **1**.

Redhead, E. y Bailey, T. L. (2007). Discriminative motif discovery in DNA and protein sequences using the deme algorithm. *BMC bioinformatics*, **8**(1): 385.

Regulski, E. E., Moy, R. H., Weinberg, Z., Barrick, J. E., Yao, Z., Ruzzo, W. L., y Breaker, R. R. (2008). A widespread riboswitch candidate that controls bacterial genes involved in molybdenum cofactor and tungsten cofactor metabolism. *Molecular microbiology*, **68**(4): 918–932.

Retsi, V., Leonti, I. A., Korfiati, A., Theofilatos, K., Likothanassis, S., y Mavroudi, S. (2015). Predicting and classifying short non-coding RNAs using a multiclass evolutionary methodology. En: *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS)*. ACM, p. 20.

Retwitzer, M. D., Kifer, I., Sengupta, S., Yakhini, Z., y Barash, D. (2015). An efficient minimum free energy structure-based search method for riboswitch identification based on inverse RNA folding. *PloS one*, **10**(7): e0134262.

Rinaldi, A. J., Lund, P. E., Blanco, M. R., y Walter, N. G. (2016). The shine-dalgarno sequence of riboswitch-regulated single mrnas shows ligand-dependent accessibility bursts. *Nature communications*, **7**: 8976.

Rivas, E. y Eddy, S. R. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, **285**(5): 2053–2068.

Rivas, E. y Eddy, S. R. (2000). Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, **16**(7): 583–605.

Rosenberg, M. S. (2009). *Sequence alignment: methods, models, concepts, and strategies*. Univ of California Press.

Russo, F., Di Bella, S., Nigita, G., Macca, V., Lagana, A., Giugno, R., Pulvirenti, A., y Ferro, A. (2012). miRandola: extracellular circulating microRNAs database. *PloS one*, **7**(10): e47786.

Saito, Y., Sato, K., y Sakakibara, Y. (2011). Fast and accurate clustering of noncoding RNAs using ensembles of sequence alignments and secondary structures. *BMC bioinformatics*, **12**(Suppl 1): S48.

Sato, K., Mituyama, T., Asai, K., y Sakakibara, Y. (2008). Directed acyclic graph kernels for structural RNA analysis. *BMC bioinformatics*, **9**(1): 1.

Sato, K., Kato, Y., Hamada, M., Akutsu, T., y Asai, K. (2011). Ipknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, **27**(13): i85–i93.

Schattner, P., Brooks, A. N., y Lowe, T. M. (2005). The tRNAscan-SE, snoscan and snoGPS web servers for the detection of tRNAs and snoRNAs. *Nucleic acids research*, **33**(suppl 2): W686–W689.

Schwende, I. y Pham, T. D. (2013). Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Briefings in bioinformatics*, p. bbt070.

Senter, E., Sheikh, S., Dotu, I., Ponty, Y., y Clote, P. (2012). Using the fast fourier transform to accelerate the computational search for RNA conformational switches. *PloS one*, **7**(12): e50506.

Serganov, A. y Nudler, E. (2013). A decade of riboswitches. *Cell*, **152**(1): 17–24.

Shalev-Shwartz, S., Singer, Y., Srebro, N., y Cotter, A. P. (2011). Primal estimated sub-gradient solver for SVM. *Mathematical programming*, **127**(1): 3–30.

Singh, P., Bandyopadhyay, P., Bhattacharya, S., Krishnamachari, A., y Sengupta, S. (2009). Riboswitch detection using profile hidden markov models. *BMC bioinformatics*, **10**(1): 325.

Singh, S. y Singh, R. (2017). Application of supervised machine learning algorithms for the classification of regulatory RNAriboswitches. *Briefings in Functional Genomics*, **16**(2): 99.

Sleator, R. D. (2010). An overview of the current status of eukaryote gene prediction strategies. *Gene*, **461**(1): 1–4.

Smith, K. D., Lipchock, S. V., Ames, T. D., Wang, J., Breaker, R. R., y Strobel, S. A. (2009). Structural basis of ligand binding by a c-di-gmp riboswitch. *Nature structural & molecular biology*, **16**(12): 1218.

Sokolova, M. y Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, **45**(4): 427–437.

Song, T. y Gu, H. (2014). Discriminative motif discovery via simulated evolution and random under-sampling. *PloS one*, **9**(2): e87670.

St-Onge, K., Thibault, P., Hamel, S., y Major, F. (2007). Modeling RNA tertiary structure motifs by graph-grammars. *Nucleic acids research*, **35**(5): 1726–1736.

Stefani, G. y Slack, F. (2008). Small non-coding RNAs in animal development. *Nature reviews Molecular cell biology*, **9**(3): 219–230.

Steffen, P., Voß, B., Rehmsmeier, M., Reeder, J., y Giegerich, R. (2005). Rnashapes: an integrated RNA analysis package based on abstract shapes. *Bioinformatics*, **22**(4): 500–503.

Sudarsan, N., Wickiser, J. K., Nakamura, S., Ebert, M. S., y Breaker, R. R. (2003). An mRNA structure in bacteria that controls gene expression by binding lysine. *Genes & development*, **17**(21): 2688–2697.

Sussman, J. L., Lin, D., Jiang, J., Manning, N. O., Prilusky, J., Ritter, O., y Abola, E. E. (1998). Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica Section D*, **54**(6-1): 1078–1084.

Tam, O. H., Aravin, A. A., Stein, P., Girard, A., Murchison, E. P., Cheloufi, S., Hodges, E., Anger, M., Sachidanandam, R., Schultz, R. M., *et al.* (2008). Pseudogene-derived small interfering RNAs regulate gene expression in mouse oocytes. *Nature*, **453**(7194): 534–538.

Tian, B., Bevilacqua, P. C., Diegelman-Parente, A., y Mathews, M. B. (2004). The double-stranded-rna-binding motif: interference and much more. *Nature reviews Molecular cell biology*, **5**(12): 1013.

Tripathi, R., Patel, S., Kumari, V., Chakraborty, P., y Varadwaj, P. K. (2016). Deeplnc, a long non-coding RNA prediction tool using deep neural network. *Network Modeling Analysis in Health Informatics and Bioinformatics*, **5**(1): 1–14.

Uhl, M. (2014). *Improving microRNA target prediction in humans using a highly descriptive graph-based machine-learning model*. Tesis de doctorado, Albert-Ludwigs-University Freiburg.

van Leeuwen, S. y Mikkers, H. (2010). Long non-coding RNAs: guardians of development. *Differentiation*, **80**(4): 175–183.

Vens, C., Rosso, M.-N., y Danchin, E. G. (2011). Identifying discriminative classification-based motifs in biological sequences. *Bioinformatics*, **27**(9): 1231–1238.

Videm, P., Rose, D., Costa, F., y Backofen, R. (2014). Blockclust: efficient clustering and classification of non-coding RNAs from short read RNA-seq profiles. *Bioinformatics*, **30**(12): i274–i282.

Wang, G., Yu, T., y Zhang, W. (2005). Wordspy: identifying transcription factor binding motifs by building a dictionary and learning a grammar. *Nucleic acids research*, **33**(suppl 2): W412–W416.

Wang, J. X., Lee, E. R., Morales, D. R., Lim, J., y Breaker, R. R. (2008). Riboswitches that sense s-adenosylhomocysteine and activate genes involved in coenzyme recycling. *Molecular cell*, **29**(6): 691–702.

Wang, S., Hou, S., Wu, J., y Wei, J. (2013). Clustering of ncRNA based on structural and semantic similarity. *Journal of Bionanoscience*, **7**(1): 20–25.

Washietl, S., Will, S., Hendrix, D. A., Goff, L. A., Rinn, J. L., Berger, B., y Kellis, M. (2012). Computational analysis of noncoding RNAs. *Wiley Interdisciplinary Reviews: RNA*, **3**(6): 759–778.

Weinberg, Z., Regulski, E. E., Hammond, M. C., Barrick, J. E., Yao, Z., Ruzzo, W. L., y Breaker, R. R. (2008). The aptamer core of SAM-IV riboswitches mimics the ligand-binding site of SAM-I riboswitches. *Rna*, **14**(5): 822–828.

Weinberg, Z., Wang, J. X., Bogue, J., Yang, J., Corbino, K., Moy, R. H., Breaker, R. R., *et al.* (2010). Comparative genomics reveals 104 candidate structured RNAs from bacteria, archaea, and their metagenomes. *Genome Biol*, **11**(3): R31.

Wen, D. y Wang, J. T. (2009). Design of an RNA structural motif database. *International Journal of Computational Intelligence in Bioinformatics and Systems Biology*, **1**(1): 32–41.

Wiedemann, K. (2014). *Detection of secondary structure motifs in long non-coding RNAs*. Tesis de doctorado, Hochschule Mittweida University of applied sciences.

Winkler, W. C., Nahvi, A., Roth, A., Collins, J. A., y Breaker, R. R. (2004). Control of gene expression by a natural metabolite-responsive ribozyme. *Nature*, **428**(6980): 281.

Xu, X., Ji, Y., y Stormo, G. D. (2009). Discovering cis-regulatory RNAs in shewanella genomes by support vector machines. *PLoS computational biology*, **5**(4).

Yandell, M. y Ence, D. (2012). A beginner's guide to eukaryotic genome annotation. *Nature Reviews Genetics*, **13**(5): 329–342.

Yang, Y. y Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. En: *Proc. 14th International Conference on Machine Learning (ICML'97)*. pp. 412–420.

Yip, K. Y., Cheng, C., y Gerstein, M. (2013). Machine learning and genome annotation: a match meant to be? *Genome biology*, **14**(5): 205.

Yuan, C. y Sun, Y. (2013). RNA-CODE: A noncoding RNA classification tool for short reads in NGS data lacking reference genomes. *PloS one*, **8**(10): e77596.

Zahran, M., Sevim Bayrak, C., Elmetwaly, S., y Schlick, T. (2015). RAG-3D: a search tool for RNA 3D substructures. *Nucleic acids research*, **43**(19): 9474–9488.

Zhang, S., Borovok, I., Aharonowitz, Y., Sharan, R., y Bafna, V. (2006). A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements. *Bioinformatics*, **22**(14): e557–e565.

Zhang, Z., Carriero, N., y Gerstein, M. (2004). Comparative analysis of processed pseudogenes in the mouse and human genomes. *Trends in Genetics*, **20**(2): 62–67.

Zhong, C. y Zhang, S. (2012). Clustering RNA structural motifs in ribosomal RNAs using secondary structural alignment. *Nucleic acids research*, **40**(3): 1307–1317.

Zhong, C. y Zhang, S. (2013). Efficient alignment of RNA secondary structures using sparse dynamic programming. *BMC bioinformatics*, **14**(1): 269.

Zhong, C., Tang, H., y Zhang, S. (2010). Rnamotifscan: automatic identification of RNA structural motifs using secondary structural alignment. *Nucleic acids research*, p. gkq672.

Zok, T., Antczak, M., Zurkowski, M., Popenda, M., Blazewicz, J., Adamiak, R. W., y Szachniuk, M. (2018). RNApdbee 2.0: multifunctional tool for RNA structure annotation. *Nucleic acids research*.

Zuker, M. (1989). On finding all suboptimal foldings of an RNA molecule. *Science*, **244**(4900): 48–52.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research*, **31**(13): 3406–3415.

Zwieb, C. (2014). The principles of RNA structure architecture. En: *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*. Springer, pp. 33–43.